DEC-11-HR2B-D

KA11
processor manual

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

1st Printing September 1970
2nd Printing (Rev) March 1971
3rd Printing October 1971
4th Printing February 1972
5th Printing May 1972

The material in this manual is for informational purposes and is subject to change without notice.

CONTENTS

CONTENTS (cont.)

ILLUSTRATIONS

DRAWINGS

TABLES

FOREWORD

INTRODUCTION

This document describes the KA11 Processor, which operates as a device on the Unibus data transmission path. The KA11 Processor is a component of a Programmed Data Processor 11 (PDP-11) and operates on data supplied by other devices (such as core memory) on the Unibus.

SCOPE

The information included in this document pertains only to the KA11 Processor. Other devices attached to the Unibus are covered in similar documents of this series, and the operation of the Unibus is detailed in the *Unibus Interface Manual.* The processor is described in terms of the set of instructions that it executes and the machine states that implement these instructions.

Sections of this document describe:

a. The logical structure of the processor

b. The theory of operation

c. The logical implementation

d. Basic maintenance procedures.

PURPOSE

This document provides the reader with the information necessary to understand the normal operation of the KA11 Processor. The processor is a complex digital device. To fully use its capabilities or to recognize and correct the cause of improper operation, the user must understand the normal operation of the processor. This document provides the information on processor operation. In addition, the information is organized to serve as a reference for specific data on the level of detailed operations and specific circuit implementation.

ORGANIZATION

The discussion of the KA11 Processor is divided into four sections. Four sections describe the processor as follows:

a. The block diagram

b. Instruction set

c. Machine state flow

d. Module-by-module logic description level.

The fifth section presents general maintenance information.

The overall structure of the KA11 Processor and the flow of data are presented in a series of block diagrams, which also introduce the connection of the processor to the Unibus. The interaction of the processor with external devices and with the data from those devices is covered in the discussion of the instruction set. The sequence or flow of machine states that determines what operations will occur is detailed in the description of the flow charts, while the implementation of this control sequence and of the operations performed on the data are shown in the logic descriptions.

CHAPTER 1
BLOCK DIAGRAMS

This section presents three block diagrams of the KA11 Processor to illustrate the overall structure of the processor from different viewpoints. The block diagrams also serve to relate the information contained in other sections of this document to an overview of the processor.

The first block diagram concentrates on the flow of data and machine states within the processor and describes the control signals that effect the data flow required by the individual machine states.

The second block diagram is a *one-bit slice* of the machine; the interconnections of all the data paths in the machine are shown for a single bit of the 16-bit word. All the control signals required to select a route for each data transfer are shown on this diagram.

A third block diagram breaks the processor down by modules, illustrating the general physical implementation of the structure and relating the block diagrams to the module descriptions.

1.1 BASIC BLOCK DIAGRAM

Figure 1-1 is a simplified block diagram that shows the relationship of data flow and machine state flow in the KA11 Processor. In addition, the diagram illustrates the relationship of the processor to the Unibus. All data inputs to the processor (such as instructions and data from core memory) are transmitted on the Unibus; equally, all outputs, whether to core memory or to I/O devices, are on the Unibus.

The block diagram divides the processor into four areas: the Bus Interface; the Data Paths; the Register; and the Control section. As the other block diagrams show, this breakdown is only basically correct and omits several exceptions or qualifications. However, this breakdown is useful to explain the interaction of data and machine states to generate new data and new machine states.

1.1.1 Data Sections

Three of the processor sections are concerned with the flow of data. The Bus Interface controls the gating of information from and to the processor; the Data Paths include all logic which performs modification on the data; and the Register is used for temporary storage within the processor. All data transfers within the processor, that is, from the Bus Interface to the Registers or from the Register to the Bus Interface, are through the Data Paths. As this structure shows, the KA11 Processor is a *general register* machine; an instruction may use any one of eight program-selected registers as data, as an address, or as a pointer to a stack. Stacks can be formed within address modes of any instruction. One of the eight registers is also the Program Counter (PC); thus, the address calculation to fetch the next instruction uses the same Register-to-Data Paths-to-Bus Interface flow that is used to fetch operands. This feature gives a very simple structure a great deal of flexibility and power.

1.1.2 Control Section

The fourth section of the block diagram is labeled Control. This section is concerned with the flow of machine states, which include the timing states (such as the Major States shown in the diagram), the instruction being executed, and various flags and condition code bits. The KA11 Processor is a variable state sequence machine; unlike computers that step through a fixed sequence of states and are active in only some of these states for each instruction, the KA11 enters only the states necessary to execute the current instruction. Therefore, the machine state flow is not a simple loop but includes many points at which not only the next operation on data, but also the next machine state, is selected by the current state and inputs.

The KA11 Processor has five major states: fetch, source, destination, execute, and service. The first four states are used during normal processor operation, while the last state (service) is used during special operations, such as traps and interrupts.

> The fetch (F) major state locates and decodes an instruction. When this major state is completed, the processor enters another major state, depending on the type of instruction decoded. It is possible to go from fetch to any other state, including back to fetch. Every instruction must first enter the fetch major state.
>
> The source (So) major state decodes the source field of a double-operand group instruction and transfers data to appropriate locations. The source major state is entered only if the instruction belongs to the double-operand group.
>
> The destination (D) major state decodes the destination field of the appropriate instruction. Destination fields are present in all double-operand, single-operand, rotate/shift, and subroutine call instructions.
>
> The execute (E) major state uses the information from previous major states to perform the specified operation. During this state arithmetic operations, logic functions, and tests are performed.
>
> The service (Se) major state is used to execute special operations, such as interrupts, traps, etc.

Although the major states follow the sequence of fetch, source, destination, execute, and service, not all major states are required for every instruction. The minimum sequence is from fetch of one instruction directly to fetch of the next instruction. A more normal sequence might be fetch, source, destination, execute and then to fetch of the next instruction. Maximum sequence is fetch, source, destination, execute, service, and back to fetch.

The Control section of the processor selects data operations by transmitting control signals to the other three sections of the processor. These control signals, which are discussed in more detail in the bit-slice block diagram, control the routing of data, which in turn controls the type of modifications performed on the data.

Inputs to the Control section consist of the present instruction, which is transferred from the Bus Interface section, and data from the Data Paths which generates the condition codes to indicate arithmetic and logical values of the data (such as positive or negative and zero or non-zero). These inputs, together with the control signals transmitted by the Control section constitute the only communication between the controlling and data handling sections of the machine.

1.1.3 The Unibus

Communications with external devices (with the exception of the console) is entirely through the Unibus. All external data transfers that the processor executes are under the control of the Bus Interface; the data enters and leaves the processor through the Bus Interface gating. Inputs to the Control section from the Unibus and the console do not transfer data. Their sole function is to determine the next bus master, which may require stopping or modifying the machine state flow in the control section.

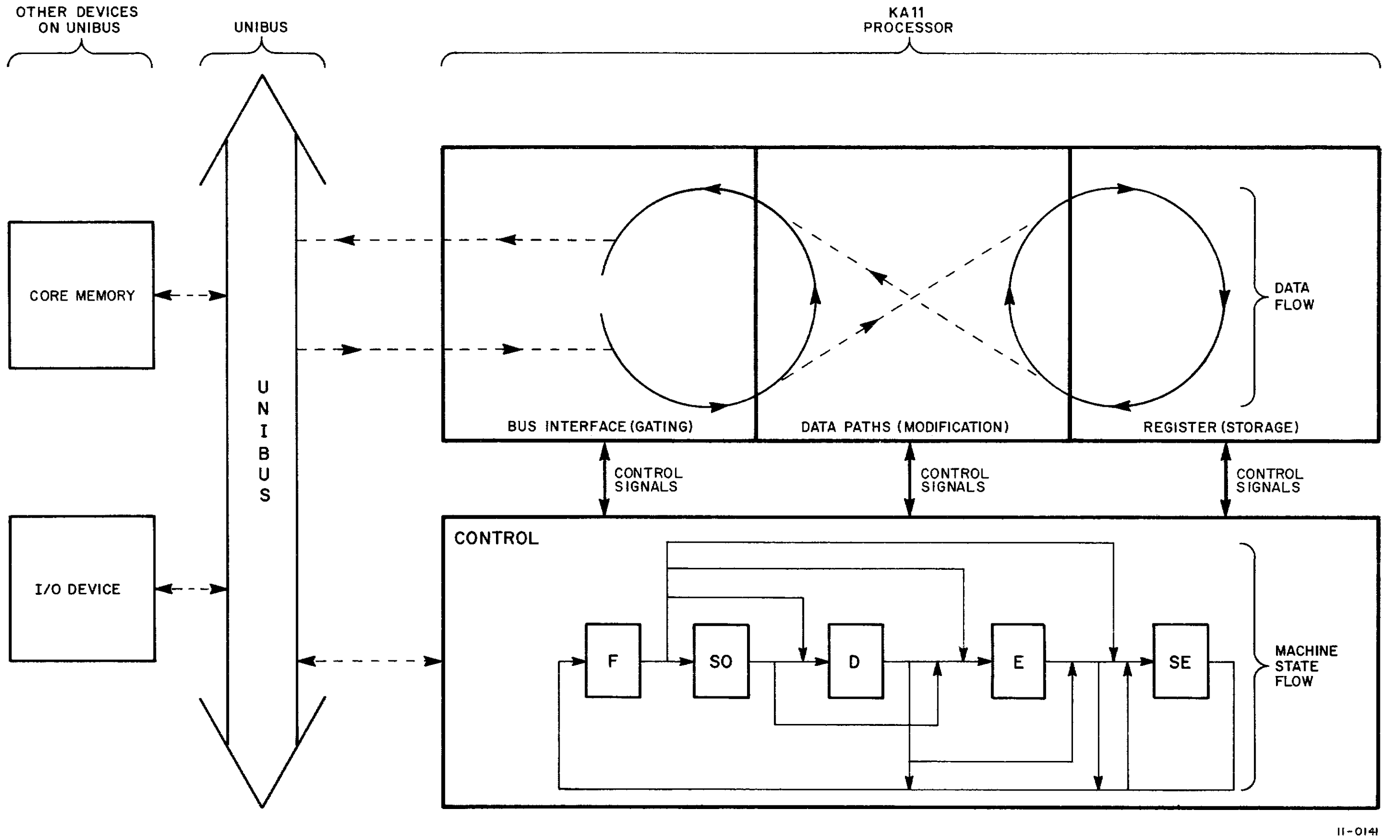


Figure 1-1 KA11 Simplified Block Diagram

1.1.4 Summary Description

Figure 1-1 can relate the structure of the KA11 Processor to the organization of this document. The control signals transmitted by the Control section are detailed in the bit slice block diagram, along with the structure of the data flow. The structure of the Control section is described in the module block diagram. The machine state flow is detailed by the flow charts, which also include sections on the priority transfers which allow the console or devices on the Unibus to *interrupt* or *cycle steal* from the processor. The logical implementation of the structure is presented in the module descriptions.

1.2 BIT-SLICE BLOCK DIAGRAM

Figure 1-2 illustrates the data flow structure of the KA11 Processor. Only one bit of the 16-bit word is shown, because this *bit slice* is representative of any of the 16 similar parallel paths in the processor. The block diagram includes the logic that routes and manipulates the data, the interconnection between these elements, and the control signals that determine the routing.

The logic elements appearing on the block diagram also appear on the module schematics, but the block diagram illustrates the interconnections more clearly. In some cases, the data pass through additional selection logic between the elements shown, as in the inputs to the Processor Status word (STATUS); for clarity, these have been omitted. The paths shown represent all the paths on which data, rather than machine states, can flow in the processor.

1.2.1 Control Signals

The control signals that appear on the block diagram are generated in the Control section of the processor. Because the processor uses a variable state sequence, a signal may be asserted at various times during the execution of an instruction; many of the signals are generated by combinational logic arrays.

The inputs to the combinational logic are time states, instructions, flags, and condition codes that are decoded to select the specific machine states during which each signal is asserted.

The bit-slice block diagram illustrates more clearly several of the distinctions made in the discussion of the basic block diagram: the only communication with external data is via the Bus Interface to the Unibus; all transfers between the Register and the Bus Interface pass through the Data Paths; and the only data inputs to the Control section are to the Instruction Register (IR) and, through the codes data logic, to the Processor Status register (STATUS).

The structure of the Control section as related to the generation of the control signals is discussed in the description of the module block diagram.

1.2.2 Structure of the Data Flow

The heart of the data flow structure is the Data Paths section, where all combination and modification of data are performed. The Data Paths are divided into three parts: the input gating, including the latches; the adder; and the output, or rotate/shift, gating.

The input gating consists of two similar structures, the A input and latch and the B input and latch. Each structure includes three input gates, each controlled by a gating signal, and a latch input gate. The purpose and operation of the latches is discussed first.

1.2.2.1 **Latches** – The latches perform two functions. In Register-to-Register or Bus-to-Bus transfers, data are stored in a latch during the transition from a read operation to a write operation. In addition, combining two variables, by addition or by a logical OR, requires the temporary storage of one variable in the data path while the second variable is being fetched from the Register.

Each input structure (as shown in Figure 1-3) is basically a Set/Reset flip-flop with three gated set inputs and one reset input. This structure is better illustrated in Figure 1-4, which is logically identical to Figure 1-3. Note that the Latch signal acts as a reset input when it is unasserted, not when it is asserted, and also that the output to the adder is asserted when low.

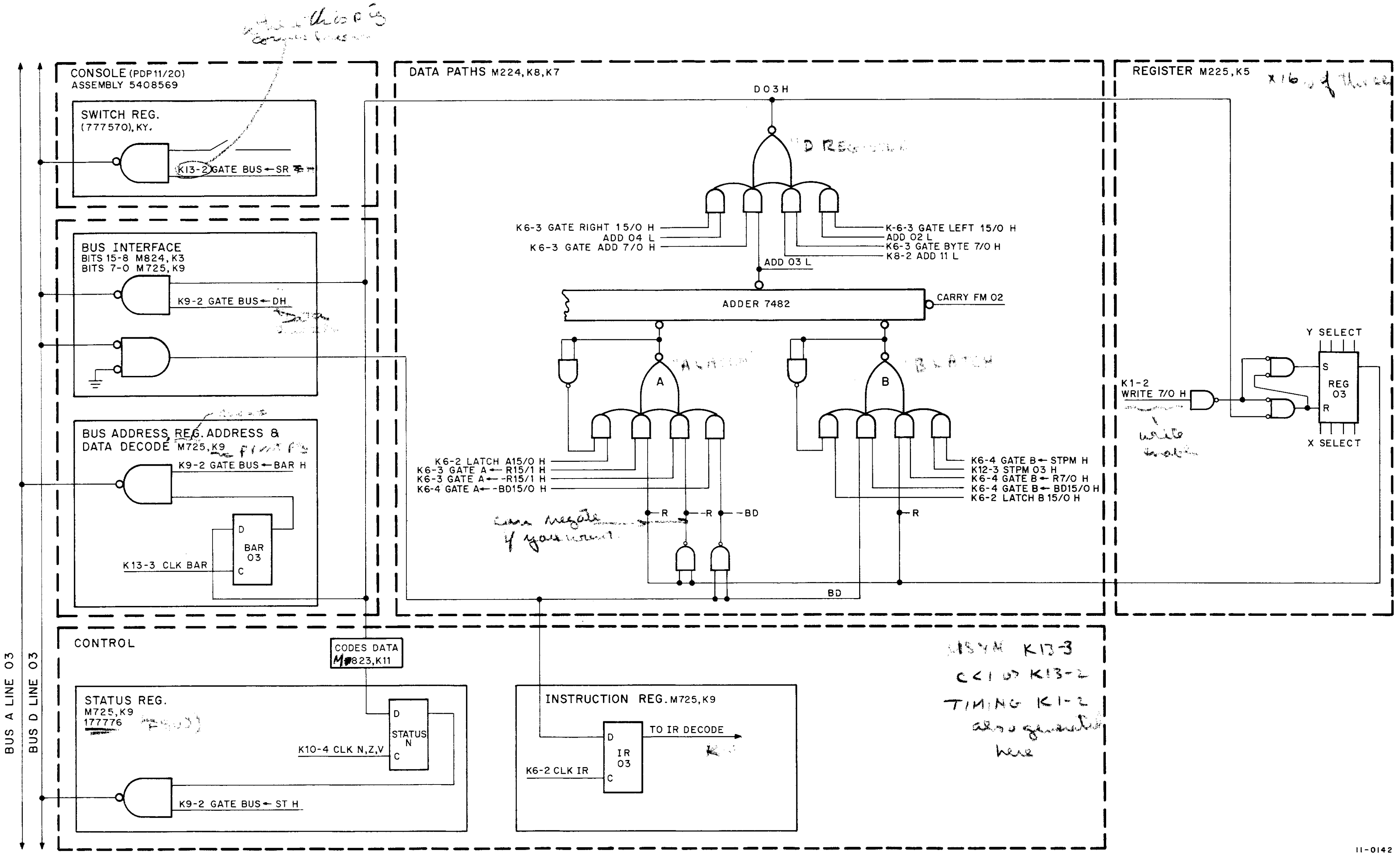


Figure 1-2 Processor Block Diagram Showing One-Bit (03) Data Slice

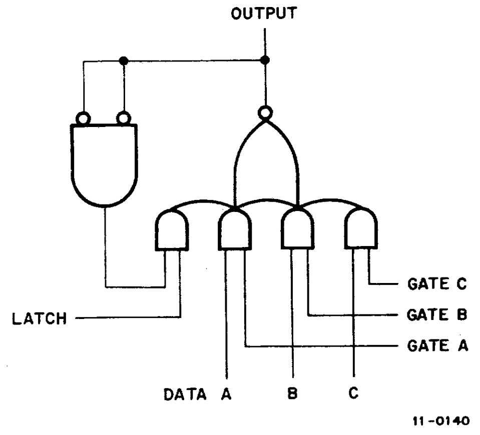


Figure 1-3 Latch Input – Logic Diagram

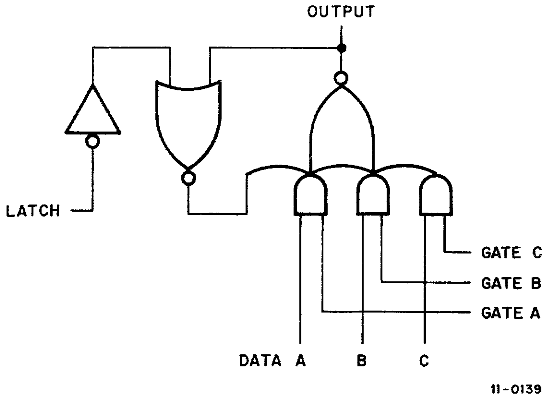


Figure 1-4 Latch Input – Functional Diagram

In use, the Latch input is cleared whenever new data are to be gated in (except when these data are to be ORed with the data already in the latch) and then reasserted before the gated data are removed. Often the latch is cleared (although no input is gated) to ensure that there are no unwanted inputs to the adders. Figure 1-5 illustrates the timing constraints placed on the input gating, latch, and data signals.

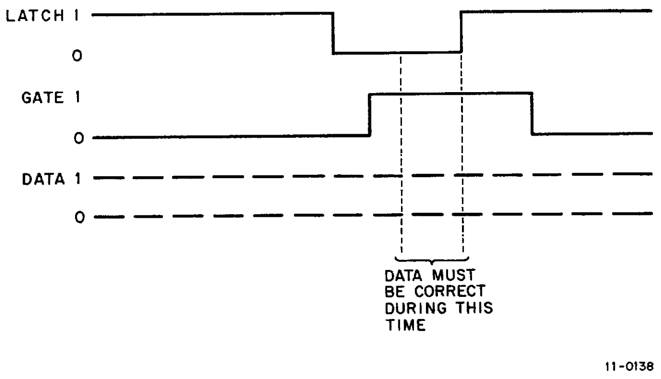


Figure 1-5 Input Timing Constraints

Each latch has three inputs. The A latch receives an input from the registers and complemented inputs from both the registers and the Bus Interface. The B latch receives inputs from both the Registers and the Bus Interface, and a third, special input to generate special trap addresses (this input will not be discussed further here). The gating signals, present at different times, are selected by combinational logic and determine the action to be taken on the input data.

The input gating can also be used to generate constants for use in the adder. For instance, a two's complement -1 can be generated by gating both register data and its complement into Latch A ($X + \overline{X} = 1$). The constants that can be generated include -1, -2, and +1. The constant +2 is generated by a combination of +1 in Latch A and the setting of the carry flip-flop.

1.2.2.2 **Adders** – The KA11 Processor uses a standard binary adder. The A, B, and carry inputs, and the sum and carry outputs are low when asserted. More information on the circuit of the adder is available in Chapter 4. Often only one input to the adder is used; in this case, the adder does not modify the input, but transmits the input unchanged.

1.2.2.3 **Output Gating** – The output, or rotate/shift, gating selects the form in which the information from the adder outputs is to be transmitted. This information is transmitted to both the Bus Interface and the Register, where other control signals determine the routing (destination) of the information. The shift gating can transmit the outputs of the adders unmodified, shifted one bit right or left, or with the 8-bit bytes swapped. The unmodified output is the one most commonly used; most data transfers and the results of most summations are unshifted.

The shift and swap outputs are selected by the Rotate/Shift and SWAB instructions, respectively. However, these output gates are also used by the processor to perform data conversions necessary in the execution of instructions. The offset of a branch instruction is shifted left during the calculation of the relative address; byte data from an odd address are swapped into the even byte (D⟨07:00⟩) on the data lines before calculations are performed and swapped back to the odd byte (D⟨15:08⟩) data lines before being retransmitted. (The data also appear on the even byte data lines for condition code sensing.)

1.2.3 Registers and Bus Interface

Outputs from the Data Paths are connected to the inputs to the Register and to the Bus Interface. There are other destinations, which include the Address decoding logic and the Processor Status word (STATUS), but these destinations are not part of the direct flow of data. The Register is used for the temporary storage of data that the processor can use in later calculations, or as address data, or to calculate address data. Two of the eight program-accessible registers and two that are not accessible to instructions are used by the processor; these include the Program Counter (PC) and Stack Pointer (SP), which are Registers 07 and 06 respectively, and two inaccessible registers labeled Temp and Source, which are registers 10 and 11 respectively.

1.2.3.1 **Registers** – Register selection is a function of logic in the Control section. The register selected is a function of the Instruction Register (IR) or of the logic, depending on the machine state.

The registers are implemented by logic elements that store 16 bits each. These bits are accessible individually, not simultaneously, so each logic element contains one bit and it is the same bit in each register. There are 16 elements, 1 element for each bit of the processor word. The logic element shown on the bit-slice block diagram contains 1 bit (bit 03) of each of the 10 registers used by the processor.

The logic elements of the registers function like Set/Reset flip-flops; thus, gating is provided on the inputs to prevent changes in the contents of the addressed register, while the outputs of the Data Paths are routed elsewhere. Set/Reset flip-flops do not provide usable outputs while new data is being written, and only one register can be addressed at a time, so the latches are used for temporary storage during register-to-register transfers.

1.2.3.2 **Bus Interface** – Data is received from the Unibus and gated onto the Unibus through the Bus Interface. The receivers are ungated, because selection of the input data (and selection of word, low byte, or high byte data) is done by the Data Paths input gating. The output gating is controlled by a signal from the Control section of the processor.

The Bus Interface also includes the Bus Address Register (BAR), which selects the bus address that is to respond to a transfer. The BAR is loaded from the Data Paths outputs and gated to the bus address lines by two signals from the Control section.

1.3 MODULE BLOCK DIAGRAM

The KA11 Processor is constructed on 14 modules. The logic of the processor has been allocated to these modules by function, where possible; this scheme reduces the number of interconnections between modules, provides functional sub-units for testing and modification, and facilitates fault isolation to the module level.

1.3.1 Module Distribution

Figure 1-6 is a block diagram of the processor that segments the processor logic following the module allocation. Several modules that perform functions both of control and of data transmission are divided into more than one block. Only the major paths of data flow are illustrated; this data flow structure expands the structure in the bit-slice block diagram but is not as detailed. The modules are also interconnected by control signals, which are not shown. Two modules that perform control functions for external operations (bus and console data transfers and release of control to the bus or the console) appear in the Bus Interface section. These modules are Priority (K3) and Bus and Console Control (K13).

Figure 1-6 includes the console in the block diagram. The console is not discussed in this document, and the discussion of that part of the block diagram is limited to mentioning the interconnections between the console and the processor.

1.3.2 Processor Structure

The processor can be divided into two major sections: the data handling section and the control section. The data handling section can be further divided into the three previously mentioned sections: the Bus Interface, the Data Paths, and the Register. These sections perform gating, modification, and storage of data respectively. The data handling section is discussed in detail in the bit-slice block diagram description.

1.3.3 Control Section

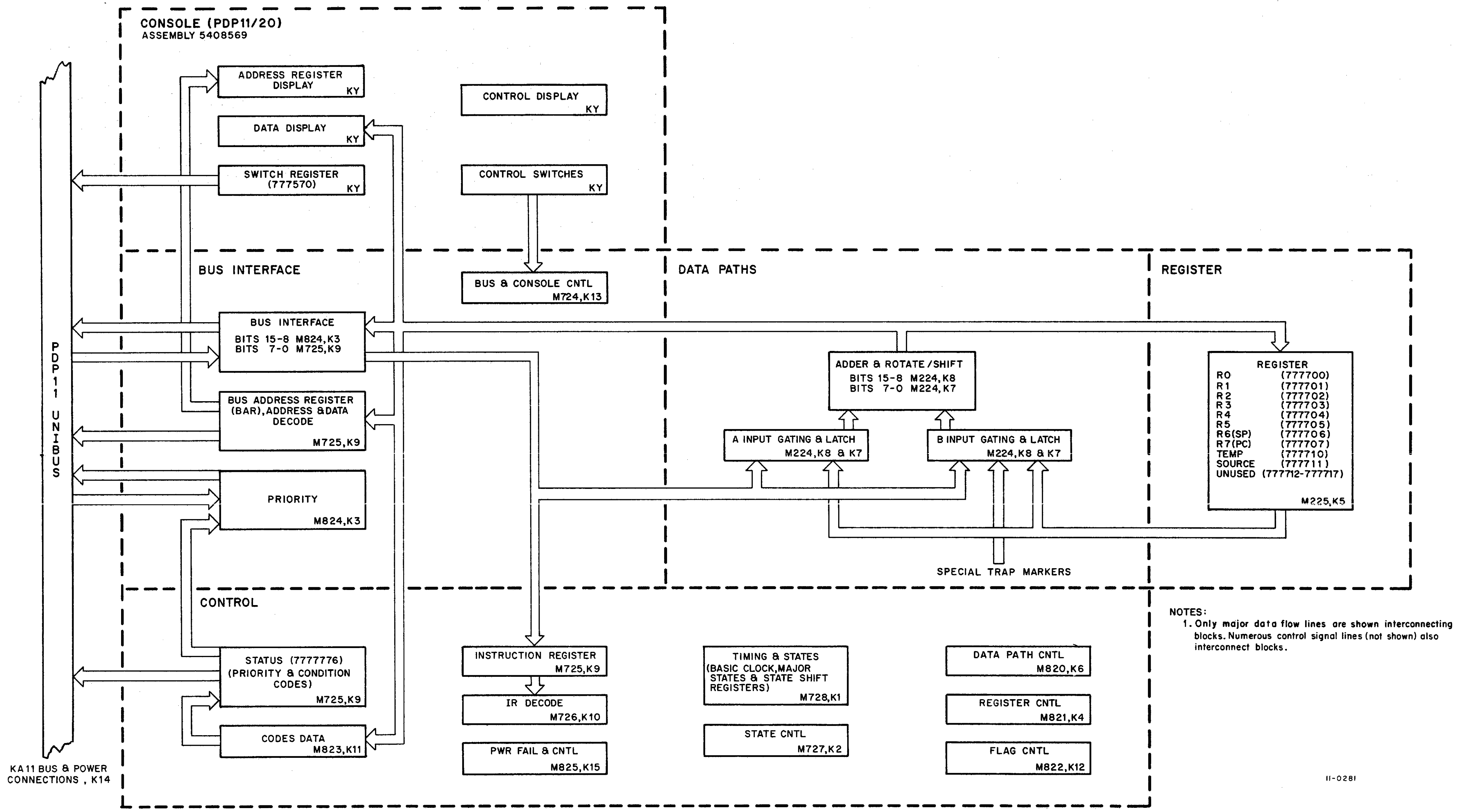
The control section can also be divided in three subsections that perform functions of storing machine states, controlling transformations from one machine state to another, and generating control signals for the current machine state. These three functions can be generalized to the same storage, modification, and gating functions described for the data handling section.

1.3.3.1 **Machine State Storage** – The KA11 Processor performs operations on data; these operations are determined by the *machine state*. This state is defined by the current instruction, the current time state within that instruction, and the values of several types of flags and conditions. Information on these state variables is stored on several modules as follows: the current instruction is held in the Instruction Register (IR) on module K-9; the time states are generated by logic on the Timing and States module, K-1; the condition codes and processor priority are stored in the Processor Status word (STATUS) on module K-9; the current operand address is stored in the Bus Address Register (BAR) on module K-9; and other flags for asynchronous conditions and error conditions are stored on the Flag Control module, K-12. The IR, the BAR, and STATUS are connected to the data flow structure of the processor.

1.3.3.2 **Machine State Selection** – The variable state sequence used by the KA11 Processor requires the processor to specify the next state explicitly. The new machine state is a function of the current machine state and of the data being processed. (If the next state were a function of the current state only, and were not influenced by the data, the processor would not be a stored program computer and would be limited to linear calculations without decision points or branches.)

The logic that selects the next machine state by generating new inputs to the state storage logic is contained on the following modules: condition codes are generated by the Codes Data module, K-11; time states are selected by the State Control module, K-2; flags are selected by the Flag Control module, K-12, and the Power Fail and Control modules, K-15 and K-3. The Instruction Register Decode module, K-10, provides signals that are used as inputs by the State Control, Codes Data, and Timing and States modules, as well as by the logic that provides control signals to the data handling logic.

1.3.3.3 **Control Signal Generation** – Four modules contain the logic that generates control signals to the data handling logic, the console, and the Unibus. These modules are: Register Control, K-4; Data Paths Control, K-6; Bus and Console Control, K-13; and Priority, K-3. The functions of each module are indicated by the name; Priority controls the release of bus control by the processor to requesting devices of various bus priority levels. Combinational logic on the modules selects combinations of inputs from the Instruction Register Decode (IRD) and Timing and States modules; these inputs represent specific machine states. For each selected machine state, the logic determines the control signals to generate.


CONSOLE (PDP11/20)
ASSEMBLY 5408569
ADDRESS REGISTER DISPLAY KY
DATA DISPLAY KY
SWITCH REGISTER (777570) KY
CONTROL DISPLAY KY
CONTROL SWITCHES KY
BUS INTERFACE
BUS & CONSOLE CNTL M724,K13
DATA PATHS
REGISTER
BUS INTERFACE BITS 15-8 M824,K3 BITS 7-0 M725,K9
ADDER & ROTATE/SHIFT BITS 15-8 M224,K8 BITS 7-0 M224,K7
REGISTER
R0 (777700)
R1 (777701)
R2 (777702)
R3 (777703)
R4 (777704)
R5 (777705)
R6(SP) (777706)
R7(PC) (777707)
TEMP (777710)
SOURCE (777711)
UNUSED (777712-777717)
M225,K5
BUS ADDRESS REGISTER (BAR), ADDRESS & DATA DECODE M725,K9
A INPUT GATING & LATCH M224,K8 & K7
B INPUT GATING & LATCH M224,K8 & K7
PDP11 UNIBUS
PRIORITY M824,K3
SPECIAL TRAP MARKERS
CONTROL
STATUS (7777776) (PRIORITY & CONDITION CODES) M725,K9
INSTRUCTION REGISTER M725,K9
TIMING & STATES (BASIC CLOCK, MAJOR STATES & STATE SHIFT REGISTERS) M728,K1
DATA PATH CNTL M820,K6
IR DECODE M726,K10
REGISTER CNTL M821,K4
CODES DATA M823,K11
STATE CNTL M727,K2
PWR FAIL & CNTL M825,K15
FLAG CNTL M822,K12
KA11 BUS & POWER CONNECTIONS, K14
11-0281

NOTES:
1. Only major data flow lines are shown interconnecting blocks. Numerous control signal lines (not shown) also interconnect blocks.

Figure 1-6 KA11 Component Block Diagram

CHAPTER 2
THE INSTRUCTION SET

The KA11 Processor is defined by its instruction set. The sequences of processor operations are selected by decoding instructions. A knowledge of the instruction set is basic to understanding what the processor is attempting to do during the execution of a program. This chapter describes the KA11 instruction set from an internal point of view; for further information on the instructions, see description in the *PDP-11 Handbook*, Appendix B, the *Unibus Interface Manual* (DEC-11-HIAA-D), and the *Paper Tape Software Manual* (DEC-11-GGPA-D).

2.1 PURPOSE

This chapter treats the instruction set as a description of the processor on a level between that of the block diagrams and the more detailed level of the flow charts. For this purpose, each instruction is defined in terms of the data manipulations and transfers which are performed to execute that instruction. These definitions illustrate the use of the processor data logic and the operations that the processor can perform.

2.2 SCOPE

This chapter describes the instruction set in a notation called the Instruction Set Processor (ISP) language. This notation is a concise method of describing the operations of any digital computer on a register or data transfer level. The ISP notation is described briefly, then the processor state information that affects the execution of the instructions is described in ISP notation. The addressing modes used by the instructions to access operands in Unibus locations or in processor general registers are listed; all the instructions are given with the ISP descriptions.

The ISP language description of the instruction set omits the processor response to:

a. Illegal or reserved instructions

b. Reference to nonexistent bus locations

c. Internal traps such as stack overflow or power failure

d. Console operations.

2.3 ISP NOTATION

The Instruction Set Processor (ISP) notation is a more concise and more carefully defined form of the common method of describing instructions. For each instruction, the operations that take place are illustrated by a string of symbols.

The string of symbols is called an instruction expression. Each expression has two parts: the conditions and the action. The expression defines a set of conditions, often a code in the Instruction Register (IR), and the actions that take place if those conditions exist. The conditions are followed by a right arrow, which is followed by the actions.

The action sequence, which is usually interpreted as one or more actions taking place simultaneously, consists of one or more pairs of destination expressions and data expressions. A destination expression specifies the location that will be set equal to the value of the data expression.

A data expression consists of location expressions combined by operators. The operators include all the standard arithmetic and logical operators (+, -, *, /, >, <, ⩾, ⩽, =, ≠, ∧, ∨, and ⊕) and a variety of other operators such as modulo (ignore carry). Concatenation (consider the combined registers as one, with length equal to the sum of the lengths) is indicated by a □. Each location expression defines the address of a register (or Unibus location) the contents of which will be operated on in the manner required by the operators.

The ISP language has several additional features relating to the assignment of names for locations. Each name can have several forms (aliases), such as a full spelling and an abbreviation. These forms are presented in sequence, separated by slashes (/).

A name can be followed by one or more array declarations. Each declaration is a pair of numbers, separated by a colon, within square brackets, and represents a sequence of numbers from the first to the last, inclusive, to be used as addresses with the name. Multiple array declarations signify a multi-dimensioned array.

Following the array declarations, if any, may be a length declaration. This has the same form as an array declaration with angle brackets substituted for the square brackets. The length declaration signifies that the name represents bits or characters numbered from the first to the last number, inclusive. The examples clarify the use of array and length declarations.

Examples

A register declaration defines a name and the data format that it represents by giving the array declarations and length declarations with the name.

The declaration:

R/Registers [0:7] ⟨15:0⟩

defines a data structure called *Registers* (abbreviated R) which consists of 8 words numbered from 0 to 7, where each word includes 16 bits numbered from 15 to 0, going from most significant (leftmost) bit to least significant (rightmost).

Similarly, an instruction declaration of the form:

ADD(:=bop+0010)→(CC,D←D+S)

defines an ADD instruction, where the sum (+) of the Source (S) and Destination (D) operands replaces the contents of the Destination operand, and sets the condition codes (CC). The comma signifies that the leftmost portion of the memory expression (CC,D) is treated in a manner defined elsewhere. The equivalence symbol (:=) signifies that the information following it defines the name (ADD) preceding it. The parenthetical statements are added for clarity.

Table 2-1 summarizes the information presented in the preceding paragraphs.

Table 2-1
ISP Summary

a/b	alias	– the names may be used interchangeably; *a* means the same as *b*.
a b	conditional	– the action sequence *b* is performed if the condition expression *a* is true.
a:=b	equivalence	– wherever *a* appears, the value of *b* may be substituted for it.
a b	replacement	– the value of the data expression *b* replaces the contents of the memory expression *a*.
a:b	length declaration	– the expression describes a series of bits or characters, numbered from *a* to *b* inclusive. *a* is the most significant, *b* the least significant.
a:b	array declaration	– the expression describes a series of words, each of a length determined by a following length declaration, numbered from *a* to *b* inclusive.
"next"	sequential execution	– the action sequence following a "next" is performed after, not simultaneous with, the sequence preceding the "next".
+ /add operators		used in data expressions to define operations performed on the data which are the contents of the locations represented by the name symbols.
- /subtract		
* /multiply		
/ /divide		
∧ /and		
∨ /or		
⊕ /exclusive or		
⌐ /not		
= /equal		
≠ /not equal		
> /greater than		
< /less than		
⩾ /greater than or equal		
⩽ /less than or equal		
□ /concatenation		

2.4 PDP-11 DATA STRUCTURE

The data structure used by the processor is defined as the structure of the processor state (those storage locations within the processor that define the current state) and the current instruction, and the structure of the Unibus locations used as processor memory. The structure of the processor state and of the memory is described by the ISP notation in Table 2-2. Table 2-3 illustrates the instruction format.

Table 2-2
Memory and Processor State Structure

Primary Memory State	
M/Mb/Memory[0:2^{16}–1]⟨7:0⟩	Byte memory
Mw[0:2^{15}–1]⟨15:0⟩: = M[0:2^{16}–1]⟨7:0⟩	Word memory mapping
Processor State (9 words)	
R/Registers[0:7]⟨15:0⟩	Word general registers
SP⟨15:0⟩: = R[6]⟨15:0⟩	Stack pointer
PC⟨15:0⟩: = R[7]⟨15:0⟩	Program counter
PS⟨15:0⟩	Processor state register
Priority/P⟨2:0⟩: = PS⟨7.5⟩	Under program control; priority level of the process currently being interpreted a higher level process may interrupt or trap this process

Table 2-2 (Cont)
Memory and Processor State Structure

CC/Condition⌴Codes⟨3:0⟩: = PS⟨3:0⟩	Under program control; when set, each instruction executed will trap; used for interpretive and breakpoint debugging
Carry/C: = CC⟨0⟩	A result condition code indicating an arithmetic carry from bit 15 of the last operation
Negative/N: = CC⟨3⟩	A result condition code indicating last result was negative
Zero/Z: = CC⟨2⟩	A result condition code indicating last result was zero
Overflow/V: = CC⟨1⟩	A result condition code indicating an arithmetic overflow of the last operation
Trace/T: = ST⟨4⟩	Denotes whether instruction trace trap is to occur after each instruction is executed
Undefined⟨7:0⟩: = PS⟨15:8⟩	Unused
Run	Denotes normal execution
Wait	Denotes waiting for an interrupt

Table 2-3
Instruction Format

Instruction Format (Bit assignments used in the various instruction formats)	
i/instruction⟨15:0⟩	
bop⟨3:0⟩: = i⟨15:12⟩	Binary operation code
uop⟨15:6⟩: = i⟨15:6⟩	Unary operation code
brop⟨15:8⟩: = i⟨15:8⟩	Branch operation code
sop⟨15:6⟩: = i⟨15:6⟩	Shift operation code
s/source⟨5:0⟩: = i⟨11:6⟩	Source control byte
sm⟨0:1⟩ : = s⟨5:4⟩	Source mode control
sd : = s⟨3⟩	Source defer bit
sr : = s⟨2:0⟩	Source register
d/destination⟨5:0⟩: = i⟨5:0⟩	Destination control byte
dm⟨0:1⟩: = d⟨5:4⟩	
dd : = d⟨3⟩	
dr⟨2:0⟩ : = d⟨2:0⟩	
offset⟨7:0⟩: = i⟨7:0⟩	Signed 7 bit integer
address⌴increment/ai	Implicit bit derived from i to denote byte or word length operations

2.5 ADDRESS MODES

The data used by an instruction is often determined by a complex address calculation, depending on the address modes and registers used in the instruction format. The address modes and the resulting operations are described in ISP notation in Table 2-4.

Table 2-4
Address Modes

Source/S and Destination/D Calculation	
S/Source⟨15:0⟩: = (¬ sd → (	Direct access
(sm = 00) → R[sr];	Register
(sm = 01) ∧ (sr ≠ 7) → (M[R[sr]]; next R[sr] ← R[sr] + ai);	Auto increment
(sm = 01) ∧ (sr = 7) → (M[PC]; PC ← PC +2);	Immediate
(sm = 10) → (R[sr] ← R[sr] – ai; next M[R[sr]]);	Auto decrement
(sm = 11) ∧ (sr ≠ 7) → (M[M[PC] + R[sr]]; PC ← PC + 2);	Indexed
(sm = 11) ∧ (sr = 7) → (M[M[PC] + PC]; PC ← PC + 2));	Relative
sd → (	Indirect access
(sm = 00) → M[R[sr]];	Indirect via register
(sm = 01) ∧ (sr ≠ 7) → (M[M[R[sr]]]; next R[sr] ← R[sr] + ai);	Indirect via stack, auto decrement
(sm = 01) ∧ (sr = 7) → (M[M[PC]]; PC ← PC +2;	Direct absolute
(sm = 10) → (R[sr] ← R[sr] – ai; next M[R[sr]]);	Indirect via stack, auto increment
(sm = 11) ∧ (sr ≠ 7) → (M[M[PC] + R[sr]]; PC ← PC + 2);	Indirect, indexed
(sm = 11) ∧ (sr = 7) → (M[M[M[PC] + PC]]; PC ← PC +2))	Indirect relative

The operations done for the same address modes in destination operands are identical, except for the JMP and JSR instructions, in which the final data is replaced by the calculated address.

2.6 INSTRUCTION EXECUTION PROCESS

Instructions are fetched from memory by the process:

(i← M[PC]; PC ← PC + 2

The instruction is decoded and the operations that are done are illustrated in Table 2-5.

Table 2-5
Instruction Execution

Instruction␣execution: = (	
MOV(: = bop = 0001) → (CC,D ← Sb);	Move word
MOVB(: = bop = 1001) → (CC,Db ← Sb);	Move byte
Binary Arithmetic: D ← DbS;	
ADD(: = bop = 0110) → CC,D ← D + s);	Add
SUB(: = bop = 1110) → (CC,D ← D – S);	Subtract
CMP(: = bop = 0010) → (CC ← D – S);	Word compare
CMPB(: = bop = 1010) → (CC ← Db – Sb);	Byte compare
Unary Arithmetic D ← uS;	
CLR(: = uop = 050_8) → (CC,D ← 0);	Clear word
CLRB(: = uop = 1050_8) → (CC,Db ← 0);	Clear byte
COM(: = uop = 051_8) → (CC,D ← ¬ D);	Complement word
COMB(: = uop = 1051_8) → (CC,Db ← ¬ Db);	Complement byte
INC(: = uop = 052_8) → (CC,D ← D + 1);	Increment word
INCB(: = uop = 1052_8) → (CC,Db ← Db + 1);	Increment byte
DEC(: = uop = 053_8) → (CC,D ← D – 1);	Decrement word
DECB(: = uop = 1053_8) → (CC,Db ← Db – 1);	Decrement byte
NEG(: = uop = 054_8) → (CC,D ← – D);	Negate
NEGB(: = uop = 1054_8) → (CC,Db ← – Db);	Negate byte
ADC(: = uop = 055_8) → (CC,D ← D + C);	Add the carry
ADCB(: = uop = 1055_8) → (CC,Db ← Db + C),	Add to byte the carry
SBC(: = uop = 056_8) → (CC,D ← D – C);	Subtract the carry
SBCB(: = uop = 1056_8) → (CC,Db ← Db – C);	Subtract from byte the carry

Table 2-5 (Cont)
Instruction Execution

TST(: = uop = 057_8) → (CC ← D);	Test
TSTB(: = uop = 1057_8) → (CC ← Db);	Test byte .
Shift operations: D ← D x 2^n;	
ROR(: = sop = 060_8) → (C□L ← C□D/2[rotate]);	Rotate right
RORB(: = sop = 1060_8) → (C□Db ← C□Db/2[rotate]);	Byte rotate right
ROL(: = sop = 061_8) → (C□D ← C□D x 2[rotate]);	Byte rotate left
ROLB(: = sop = 1061_8) → (C□Db ← C□Db x 2[rotate];	Byte rotate left
ASR(: = sop = 062_8) → (CC,D ← D x 2);	Arithmetic shift right
ASRB(: = sop = 1062_8) → (CC,Db ← Db/2);	Byte arithmetic shift right
ASL(: = sop = 063_8) → (CC,D ← D x 2);	Arithmetic shift left
ASLB(: = sop = 1063_8) → (CC,Db ← Db x 2);	Byte arithmetic shift left
ROT(: = sop = 064_8) → (C□D ← D x 2^s);	Rotate
ROTB(: = sop = 1064_8) → (C□Db ← D x 2^s);	Byte rotate
SWAB(: = sop = 3) → (CC,D ← D⟨7:0, 15:8⟩)	Swap bytes
Logical Operations	
BIC(: = bop = 0100) → (CC,D ← D ← D ∧ ¬ S);	Bit clear
BICB(: = bop = 1100) → (CC,Db ← Db ∧ ¬ Sb);	Byte bit clear
BIS(: = bop = 0101) → (CC,D ← D ∨ S);	Bit set
BISB(: = bop = 1101) → (CC,Db ← Db ∨ Sb);	Byte bit set
BIT(: = bop= 0011) → (CC ← D ∧ S);	Bit test under mask
BITB(: = bop = 1011) → (CC ← Db ∧ Sb);	Byte bit test under mask
Branches and Subroutine Calling: PC ← f;	
JMP(: = sop = 0001_8) → (PC ← D′);	Jump unconditional
BR(: = brop = 01_{16}) → (PC ← PC + offset);	Branch unconditional
BEQ(: = brop = 03_{16}) → (Z → (PC ← PC + offset));	Equal to zero
BNE(: = brop = 02_{16}) → (¬ Z → (PC ← PC + offset));	Not equal to zero
BLT(: = brop = 05_{16}) → (N ⊕ V → (PC ← PC + offset));	Less than (zero)
BGE(: = brop = or_{16}) → (N ≡ V → (PC ← PC + offset));	Greater than or equal (zero)
BLE(: = brop = 07_{16}) → (Z ∨ (N ⊕ V) → (PC ← PC + offset));	Less than or equal (zero)
BGT(: = brop = 06_{16}) → (¬ (Z ∨ (N ⊕ V)) → (PC ← PC + offset));	Less greater than (zero)
BCS/BHIS(: = brop = 87_{16}) → (C → (PC ← PC + offset));	Carry set; higher or same (unsigned)
BCC/BLO(: = brop = 86_{16}) → (C ¬ C → (PC ← PC + offset));	Carry clear; lower (unsigned)
BLOS(: = brop = 83_{16}) → (C ∧ Z → (PC ← PC + offset));	Lower or same (unsigned)
BHI(: = brop = 82_{16}) → ((¬ C ∨ Z) → (PC ← PC + offset));	Higher than (unsigned)
BVS(: = brop = 85_{16}) → (V → (PC ← PC + offset));	Overflow
BVC(: = brop = 84_{16}) → (¬ V → (PC ← PC + offset));	No overflow
BMT(: = brop = 81_{16}) → (N → (PC ← PC + offset));	Minus
BPL(: = brop = 80_{16}) → (¬ N → (PC ← PC + offset));	Plus

JSR(: = sop = 0040_8) → (
SP ← SP – 2; next
M[SP] ← R[sr];
R[sr] ← PC;
PC ← D);
	Jump to subroutine by putting R[sr], PC on stack and loading R[sr] with PC, and going to subroutine at D

RTS(: = i = 00200_8) → (
PC ← R[dr];
R[dr] ← M[SP];
SP ← SP + 2);
	Return from subroutine

Miscellaneous processor state modification:	

RTI(: = i = 2_8) → (PC ← M[SP];
SP ← SP + 2; next
PS ← M[SP];
SP ← SP + 2);
	Return from interrupt

HALT(: = i = 0) → (Run ← 0);	
WAIT(: = i = 1) → (Wait ← 1);	
TRAP(: = i = 3) → (SP ← SP + 2; next M[SP] ← PS;	Trap to M [34_8] store status and PC

Table 2-5 (Cont)
Instruction Execution

```
                    SP ← SP + 2; next
                    M[SP] ← PC;
                    PC ← M[$34_8$];                                    Enter new process
                    PS ← M[12]);
      EMT(: = brop − $82_{16}$) → (                                    Emulator trap
                          SP ← SP + 2; next
                          M[SP] ← PS;
                          SP ← SP + 2; next
                          M[SP] ← PC;
                          PC ← M[$30_8$];
                          PS ← M[$32_8$]);
IOT(: = i = 4) → (see TRAP)                                            I/O trap to M[$20_8$]
      RESET(: = i = 5) → (not described)                               Reset to external devices
      OPERATE(: = i⟨5:15⟩ = 5) → (                                     Condition code operate
         i⟨4⟩ → (CC ← CC ∨ i⟨3:0⟩);                                     Set codes
         ¬ i⟨4⟩ → (CC ← CC ∧ ¬ i⟨3:0⟩));                                Clear codes
                                        end Instruction␣execution
```

CHAPTER 3
MACHINE STATE FLOW CHARTS

The instruction set of the KA11 Processor is implemented by control logic that produces a variable sequence of states, depending on the instruction and on the previous state of the processor. The structure of the control logic is described in the block diagram discussions and detailed in the module descriptions. This chapter describes the machine states in more detail and illustrates the sequences of states that the processor follows to execute various instructions. The flow charts presented in this chapter list the machine states, the control signals generated in each state, and the transfers from one machine state to the next.

The flow of data through the processor is implicit in the operation of the control section, because the data flow is controlled by the signals generated in each machine state. The flow charts do not explicitly describe the data flow, although most data transfers are described in the comments accompanying the line of flow. To follow the flow of data through the processor, study the Bit-Slice Block Diagram (see Figure 1-2), and determine the effects of the control signals generated during each machine state.

3.1 PURPOSE OF THE FLOW CHARTS

The flow charts provide information for two purposes. First, the reader who is learning the KA11 Processor for the first time can follow the sequence of control statements and indirectly follow the flow of data to gain an understanding of the instruction set and its implementation and to learn more about the interaction of the processor with other devices on the Unibus.

Second, the reader who is familiar with the normal operation of the processor can use the flow charts as a guide for troubleshooting when working with a processor. Often a good way to determine the cause of improper processor operation is to check the flow charts to determine the signals that should be present during the affected portion of the processor sequence. Then, check the particular signals to determine which ones are not present. The conditions listed on the flow chart for the generation of the signal will appear on the print set as inputs to the combinational logic that generates the signal. These inputs can be tested and the logic traced to determine where the failure occurred.

3.2 KA11 FLOW CHART SYSTEM

The flow chart system used for the KA11 Processor provides a unique departure from standard flow charts. Information given on the flow chart enables the user to look at a circuit schematic and identify a specific gate, or gates, which normally causes generation of a specified logic signal during a certain machine timing state. The system used to provide this information is discussed in subsequent paragraphs.

3.2.1 Signal Names

Signals, when generated, are shown in the following manner on the flow chart:

K13-4 CONSF ◄—— 1 ——— $\boxed{\text{HALT F (1)}}$

The alphanumeric symbol preceding the signal name is the page number of the circuit schematic where the signal is generated (in the above case, schematic K13, sheet 4). The Boolean statement within the box to the right defines the gate, or gates, of the combinational logic that are active for generation of the signal. A signal name with no box to the right indicates unconditional generation of the signal at that time.

3.2.2 Symbols

There are 14 special symbols used on the flow charts. These symbols are shown in Table 3-1, which also includes the name of each symbol, its general function, and the number of the paragraph containing a detailed explanation of the symbol.

Table 3-1
Flow Chart Symbols

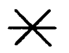
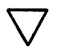
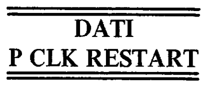
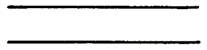
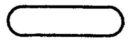
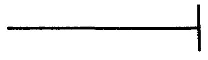
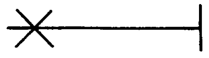
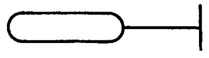
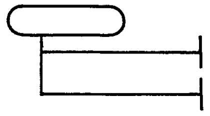
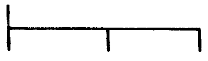
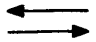
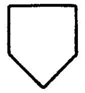
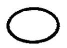
Symbol	Name	Function	Paragraph Reference
	System clock (SCLK)	Indicates that a system clock cycle begins at this point.	3.2.2.1
	Major state entry and exit	Denotes an entry into, or exit from, some processor major state.	3.2.2.2
 DATI P CLK RESTART	Major operation	Indicates that some special sequence is to be performed before continuing in the flow.	3.2.2.3
	Delays	Indicate that some additional time is required before proceeding in the flow.	3.2.2.4
	Time states	Indicate that the time state within the symbol has been entered and is controlling the operations.	3.2.2.5
	Signal block – general	A gathering point for various signals that occur at, or during, the point encountered.	3.2.2.6
	Signal block – used with system clock	Indicates those signals that are generated as a direct result of SCLK.	3.2.2.6
	Signal block – used with time state	Indicates the signals that generate the next possible time state.	3.2.2.6
	Signal block – used below time state	Indicates the signal(s) that may be generated in the time state for control of operations.	3.2.2.6
	Parallel operation connector	Indicates that more than one path can occur and that, if an operation reference statement is encountered down the leg, then that condition must be met before proceeding.	3.2.2.7

Table 3-1 (Cont)
Flow Chart Symbols

Symbol	Name	Function	Paragraph Reference
	Arrows	Indicate direction of flow; used only for clarity on flow chart.	3.2.2.8
	Page interconnection	This symbol contains a letter and identifies the exit point of one page and the entry point on the new page.	3.2.2.9
	Pulse	Designates that pulse signal which causes the next sequential operation that is pertinent to machine flow.	3.2.2.10

3.2.2.1 **System Clock (SCLK)** – The system clock (SCLK) symbol (see Table 3-1) indicates that a SCLK cycle begins at this point. This symbol implies that once the clock cycle is started, it completes the cycle and that all of its associated clocking levels (R/WO, R/W1, R/W3, R/W2) are generated in their proper sequence before the next SCLK symbol is encountered. This conforms to the KC11 clock circuit, which, once a cycle is started, cannot be stopped until R/W2. The R/W states are implied in the signal conditions, when used, except at those points on the flow chart where they are shown for clarity.

3.2.2.2 **Major State Entry and Exit** – The major state entry and exit symbol (see Table 3-1) denotes entry into, or exit from, some major state. The states that may be encountered are:

Mnemonic	Name
F	Fetch
S	Source
D	Destination
E	Execute
SVC	Service
C	Console
DATI	Data in
DATIP	Data in, pause
DATO	Data out
DATO#	Modified data out
EXIT	Exit

The appropriate mnemonic from the above list is located within the symbol. When used for entry, the symbol indicates that this is the major state being entered and the state is retained until another symbol, used as an exit, is encountered.

The DATI, DATIP, DATO, and DATO# states are entry points into the bus operation sequence and are a result of a major operation symbol encountered in the instruction flow charts. (Refer to Paragraph 3.2.2.3 for a description of major operations.)

The EXIT state is the normal termination point for a bus operation. This symbol indicates that it is necessary to return to the instruction flow signal point that initiated the bus operation (refer to Paragraph 3.2.2.3).

3.2.2.3 **Major Operation** – The major operation symbol (see Table 3-1) indicates that some special sequence must be performed before continuing in the flow chart. It may, or may not, require time delays and SCLK timing to perform the required operation. The initial statement within the symbol (DATI in the example shown in Table 3-1) indicates the operation to be performed. The final statement (P CLK RESTART in the example) is the signal that terminates the operation when asserted. In the example given in the table, the major operation symbol indicates that a DATI bus operation is to be performed and that flow on the chart continues when P CLK RESTART occurs. It also implies that the DATI operation can be seen by following the bus flow chart and that the EXIT symbol on the bus flow chart returns the user to the main flow where the symbol was encountered. In some instances, the termination statement is an OR function, and the output then encounters a parallel operation connector symbol (Paragraph 3.2.2.7) in order to proceed.

3.2.2.4 **Delays** – The delay symbol (see Table 3-1) indicates that additional time is required before proceeding in the flow. In most cases, this time element is defined and an indication given as to the amount of time involved. If it is a control operation, the name of the operation is indicated, followed by the time quantity, or just the time quantity is shown. If a delay is used in order to wait for a specific signal, then the time quantity is listed followed by the signal required in order to proceed.

3.2.2.5 **Time States** – The time state symbol (see Table 3-1) indicates that the time state indicated with the symbol has been entered and is controlling the operations being performed. The state indicated within the symbol is either ISR or BSR with its associated numerical value.

3.2.2.6 **Signal Block** – The signal block symbol (see Table 3-1) is a gathering point for the various signals that occur at, or during, the point where encountered.

If the signal block symbol is used in conjunction with the SCLK symbol (see example in Table 3-1), it indicates those signals that are generated as a direct result of the system clock.

If the symbol is used in conjunction with the time state symbol (see example in Table 3-1), it indicates the signals that generate the next possible time state. At this point, the instruction flow charts also include the set-up signals for bus flow timing changes, when applicable.

If the signal block symbol is used below the time state symbol (see example in Table 3-1), it indicates the signal, or signals, that may be generated in the time state for control of operations. The signal block closest to the time state symbol encompasses the operation control signals. The lower signal block, when used, contains major state change signals only. This latter block is found just prior to the exit symbols.

Whenever the signal block symbol is encountered in a flow path and is independent of a time state symbol line, it indicates that the signal(s) occurs as controlled by the conditions associated with that point on the diagram.

3.2.2.7 **Parallel Operation Connector** – The parallel operation connector (see Table 3-1) indicates that more than one path can occur and that, if an operation reference statement is encountered down the leg, then that condition must be met before proceeding.

There are cases where operations are performed at the same time and, in this instance, no operation reference statement is given. In those cases where operations are performed in parallel but only one path is the main path for continuous flow, the operations are indicated in the manner shown in Figure 3-1.

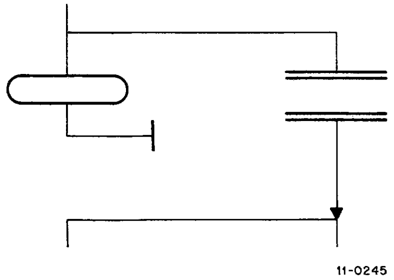


Figure 3-1 Parallel Operation – Main Path

3.2.2.8 **Arrows** – The arrow symbol (see Table 3-1) is shown at those junction points where it is not obvious as to what is expected, or at those points where more than one signal meet, or at those points where clarity is necessary to properly define the flow.

3.2.2.9 **Page Interconnection** – The page interconnection symbol (see Table 3-1) contains a letter designation within the symbol to correlate an exit point on one page with the associated entry point on the new page. A comment accompanies the symbol to indicate which new page the flow is continued on.

3.2.2.10 **Pulse** – The pulse symbol (see Table 3-1) is inserted into a flow line at a point where the pulse would occur in time. The symbol designates that pulse signal which causes the next sequential operation pertinent to machine flow. A signal block may be appended to this symbol.

3.2.3 Signal Statements

Signal statements are those signal names that do not include circuit schematic references and that define the signal requirements in order to proceed. A typical statement of this type is BUS SSYN, which states that if, or when, this signal occurs, the user proceeds down the flow path concerned. If -(BUS SSYN) is the statement, then it indicates to proceed when the true signal is not present. The previous example is an operation reference statement. A time state reference statement is: BSR ← 15 or ISR ← 0. An exit reference statement is: FETCH ← 1.

3.2.4 Comments

Comments that are required to clarify an operation are shown to the left of the time state symbol. These comments present an overview of what can be expected to occur during that time state. The comment may be supplemented with a symbol such as Ⓡ or Ⓦ. These symbols indicate a read from, or a write into, one of the KA11 general registers.

3.2.5 Time State Sequence

Time state flow sequences may exist in one of two major configurations depending on whether or not a bus operation is involved. An example of each of these configurations is presented in the following paragraphs.

3.2.5.1 **Time State Sequence – Without Bus Operation** – Figure 3-2 shows part of the EXECUTE * JSR flow chart, at the point where the ISR1 time state is entered. The comments indicate that register 7 (PC) is read into Latch B. The time state symbol signal block shows that the ISR shifts to its next state since this is not a unary (single-operand), binary (double-operand), or rotate/shift operation. During the time state, the PC is gated from the register output into the latch by gating each individual byte. The operation signals occur as a function of EXEC * JSR * ISR.

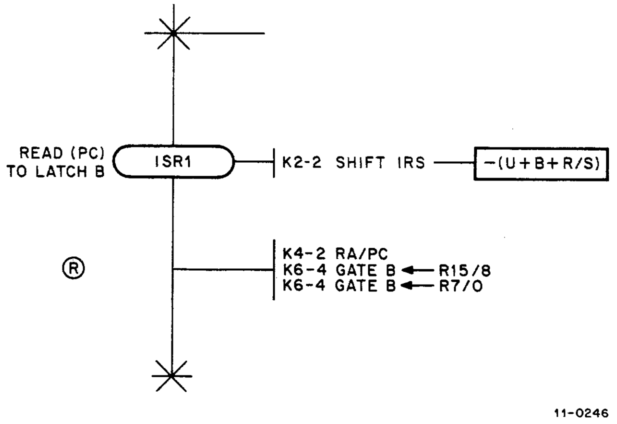


Figure 3-2 Portion of EXECUTE * JSR Flow Chart

3.2.5.2 **Time State Sequence – With Bus Operation** – Figure 3-3 shows a portion of the EXECUTE * RTI flow chart with EXEC * RTI at the entry point. The system clock (SCLK) clocks EXECUTE to a 1 and BSR to a 1. Simultaneously, the ISR is clocked to 15 and the DATI bus operation starts. The comments indicate that the stack pointer (SP), which is general register 6, is to be used for the bus address. This address is incremented after it is used and then reloaded into register 6. The data contained in the address location specified by register 6 is returned and stored temporarily in latch B. When the clock restarts, via P CLK RESTART, the ISR goes to 14. The DATI bus operation used two additional SCLK cycles (BSR3 and BSR7) to perform all of its operations while the ISR remained at 15.

There is no connection below the ISR15 time state symbol; thus, the flow chart indicates that primary flow proceeds through the major operation path.

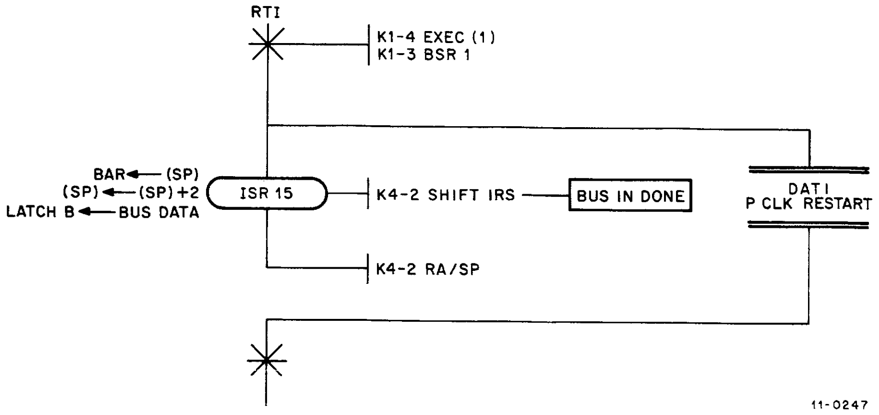


Figure 3-3 Portion of EXECUTE * RTI Flow Diagram

3.2.6 Composite Layout

Figure 3-4 illustrates all of the flow chart designations and symbols in a manner in which they may normally be encountered on the flow chart. This figure provides the user with one illustration of most of the flow chart system components discussed in previous paragraphs.

3.3 INFORMATION IN THE FLOW CHARTS

The flow charts contain four major classes of information about the processor. The present machine state is indicated by the position on the flows; the signals that are generated in each machine state are named; the inputs that determine the signals to be generated are shown as conditions; and the selection of the next machine state as a function of the inputs (conditions) is shown.

3.3.1 Current Machine State

The machine state represented by any point on the flow charts is determined by the time states entered by the line of flow in reaching that point and by the instruction being executed. For brevity and clarity, only a few of the components of a machine state are shown at any point on the flow; the reader must keep track of the time states and conditions that the flow has entered, as well as the instruction and major state represented by the flow.

3.3.2 Input Information

At various points in the flow where signals are generated, the flow charts show the conditions required to generate each signal. These conditions are composed of groups of input signals. The input signals include both parts of the machine state and input information from the data handling section of the processor. The name assigned to a condition on the flow chart represents some specific combination of signals that can often be found as physical inputs to the combinational logic that generates the control signal or data handling control signal shown.

3.3.3 Next Machine State

The control signals generated in a machine state can be divided into two major groups: the signals that generate the next machine state and the signals that control the data handling. Each machine state is followed by one or more machine states, which are selected by the control signals generated in the machine state. These control signals are shown, along with the conditions (inputs) that generate each signal.

3.3.4 Output Signals

The remainder of the control signals are outputs to the data handling section; these signals control what operations are performed on the data and are shown with the inputs that select the activity to occur.

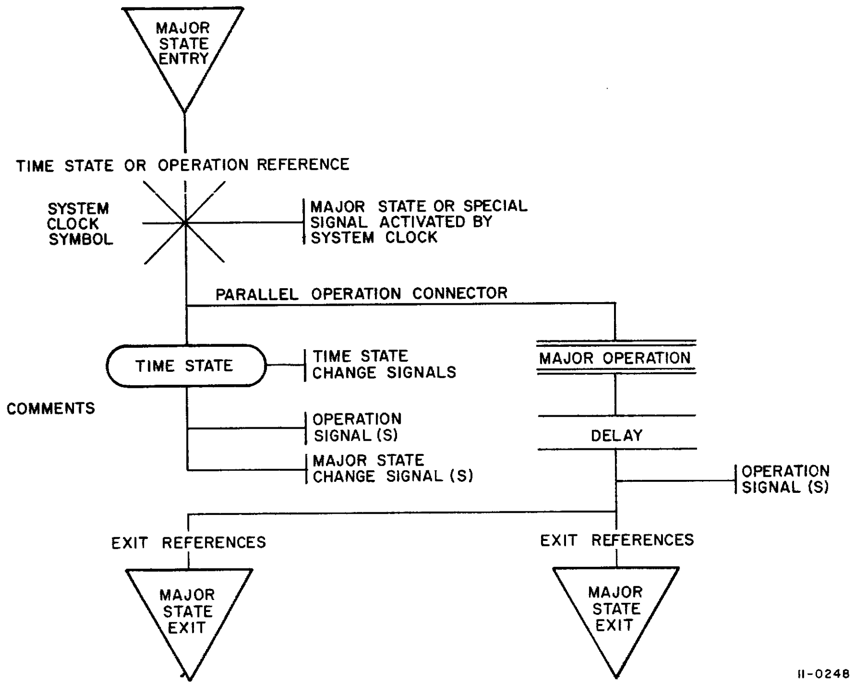


Figure 3-4 PDP-11 Processor Flow Chart Composite Layout and Symbol System

3.4 A NOTE ON PROCESSOR DESCRIPTION

This manual presents the information needed to acquire a practical understanding of the KA11 Processor operation. However, it is interesting to consider the processor in a more general theoretical manner to enable comparisons between the KA11 Processor and other machines. This section is presented for information only and is not necessary to the discussion of the KA11 Processor flow of operation.

The most general description of an information processing machine considers the machine in terms of an object that can be in any one of a finite (for digital devices) number of states. In each state, the machine is subject to certain inputs that determine both the next state the machine will be in and the outputs from the machine. One way of representing the structure and operation of such a machine is to list each possible combination of machine state and inputs, with the corresponding next machine state and outputs. This listing takes the form of a series of quadruplets (CS,I,NS,O), where CS is the current machine state, I is the input, NS is the next machine state, and O is the output.

For a digital computer, the number of quadruplets that can be listed is much too large for such a listing to be practical. Therefore, the listing is replaced by a shorthand description (such as the flow charts presented here) from which the reader can assemble the information for each quadruplet when he needs it. Each point in the flow chart represents a machine state, which is specified by the information about time states, instructions, and processor conditions which lead to the selected point. The possible inputs to this machine state are represented by the conditions listed at that point; some, or all, of the possible inputs may be present.

Each combination of a machine state and possible inputs leads to a next machine state. The possible next states for a given state are represented by the branching flow leading from the point on the flow chart that represents the present state; often the conditions that select a particular next state are given at the branch point.

The outputs possible in a given machine state are also a function of the inputs; thus, various outputs are presented at the appropriate point on the flow chart with the input conditions that generate them.

To compile the information corresponding to a given quadruplet, the reader must examine the flow charts to determine all the state information that specifies a location on the flow and examine all the inputs or conditions at that location to determine the ones that are involved in the specific operation of interest. The signals shown with the active inputs are the corresponding outputs; these outputs determine both the next machine state and the actual outputs from the control section.

Note that this discussion is concerned only with the control section of the KA11 Processor. Some outputs are control signals to the data handling section of the processor. When considering the processor as a whole, the inputs and outputs are the data on which the processor operates. However, because the processor operation is controlled by the data that pass through the processor only to a very limited extent, it is helpful to consider the data handling section as a separate device under the control of the processor. The operations that can be performed on the data have been described in the discussion of the instruction set and in the discussion of the data flow structure.

3.5 PROCESSOR TIMING

The processor timing consists of five major processor states (or cycles) during which specific processor operations are carried out (such as fetching an instruction from memory). In addition, each major state is divided into a number of ISR states during which individual steps of the operation are performed. The ISR states may also consist of a number of BSR states which are entered when the processor needs to obtain information from the bus.

The major states are so called because each major state controls the performance of a series of functions that result in one overall action; for instance, the Fetch major state controls the transfer of an instruction word from memory to the Instruction Register and the decoding of that word to determine the next activity. This action requires operations in two ISR states, one of which is divided into several BSR states.

The major states are shown in Figure 1-1 and their functions are listed in Table 3-2.

Table 3-2
KA11 Major States

State	Abbreviation	Function
Fetch	F	The instruction word is transferred from memory to the Instruction Register and decoded to determine the next state to enter.
Source	So	This state is used by two address instructions only to perform the same function for a first operand that Destination performs for a second.
Destination	D	The address of an operand is calculated and the operand is transferred from a storage location to the processor.
Execute	E	Manipulates the operands and transfers the results to the storage location the address of which was calculated in Destination.
Service	Se	All trap and interrupt service routines, as well as certain processor housekeeping functions, are performed in this state.

Each major state is divided into several time states by the operations of the Instruction Shift Register. Typically, each ISR state results in the execution of a data transfer; the individual steps in the transfer are controlled by the detailed timing of the processor. In external transfers over the Unibus, the ISR state is divided into several states of the Bus Shift Register (BSR). The BSR is similar in implementation to the ISR and controls the execution of detailed address calculations and Unibus control during a data transfer. The detailed timing within BSR states is controlled by the Read/Write Shift Register (R/W), in a similar manner as the detailed timing within ISR states.

The following sections discuss the prints, which contain the flow charts for the various major states of the KA11. Each section covers one major state, and refers to the prints that illustrate that state.

First, a basic flow chart of the flow of control among the major states is discussed; then, the execution of the external data transfers is discussed by reference to the Bus Operation Flow Charts. The major states are described in the order given in Table 3-2.

3.6 MAJOR STATE FLOW

The major state flow is shown in Figure 3-5, which is an enlarged version of the major state flow portion of Figure 1-1.

3.6.1 Fetch

At the beginning of the processing of each instruction, the processor enters the Fetch major state, performs a DATI bus operation to transfer the new instruction from memory to the Instruction Register (IR), and then decodes the contents of the Instruction Register. Depending on the operations required by the current instruction, the processor may perform modifications on the processor state or processor control flip-flops during the current major state, or the processor may transfer to any other major state (including entering Fetch again, to get the next instruction).

For the purposes of this discussion, the instructions can be divided into two categories: data manipulation instructions, which can access information stored outside the processor, and processor control instructions, which generally work on the processor state and information stored in the processor. There is a good deal of overlap between the two categories (such as jump and branch instructions). For data manipulation instructions, the processor usually enters Source or Destination major state from Fetch, then enters Execute and finally enters either Service or Fetch (for the next instruction). Many processor control instructions are completed in Fetch, while others are finished in Execute and do not require entry into Source or Destination. In general, instructions are completed before the processor enters the Service major state, which is primarily involved in trap and interrupt service.

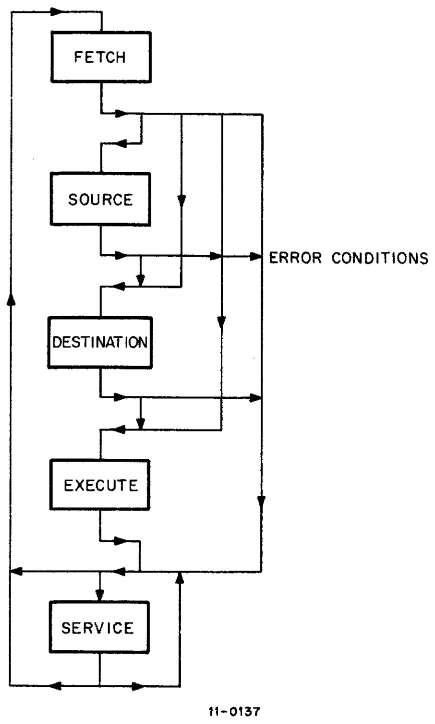


Figure 3-5 KA11 Major State Flow

3.6.2 Source and Destination

Source and Destination are the operand acquisition major states; any necessary address calculation is done in these states. Because the KA11 Processor uses a variable state sequence that enters only the machine states necessary to process the current instruction, the processor skips an operand acquisition state if the corresponding address mode is zero (i.e., if the operand is in a processor register and no Unibus transfers are needed). Two-address instructions can therefore use any sequence of operand acquisition major states; both Source and Destination, Source only, Destination only, or neither state. The last sequence occurs when both operands are in processor registers that can be accessed directly in the Execute major state.

Certain processor control instructions use the Destination major state to calculate addresses and, thereby, replace the contents of the Program Counter. These instructions include the JMP and JSR instructions. The Destination major state is aborted before the contents of the specified address are transferred. The address contains the next instruction, which will be transferred in the next Fetch major state.

3.6.3 Execute

The Execute major state controls all manipulation done on data (with the exception of address calculations) and controls all transfers of data from the processor to Unibus locations (except during trap and interrupt service). Because of the varying sequences of major states which can be used to access operands, Execute can be entered from Fetch, Source, or Destination.

3.6.4 Service

The final major state of the KA11 Processor is called Service. Only a few instructions (e.g., the trap instructions, wait, and halt and reset) require entry into the Service state. For all other instructions, Service may be entered if bus requests, interrupts, or console switch operations, are present. Some demands are internal to the processor; these include traps for illegal or reserved instructions and for conditions such as low power supply voltage.

3.7 UNIBUS OPERATIONS

The Unibus executes data transfers in four types of bus operations. These operations are conducted by a Master device that controls transfers with a Slave device. The processor is never a Slave device in data transfers; it can act as a Slave device only during Interrupt bus operations. The direction of a bus data transfer is described in terms of the Master device; a data in (DATI) or data in, pause (DATIP) transfers data from the Slave into the Master, while a data out (DATO) or data out, byte (DATOB) transfers data out of the Master to the Slave. A more detailed discussion of the theory of operation of the Unibus is in the *Unibus Interface Manual*, DEC-11-HIAB-D.

3.7.1 Processor Control of Data Transfers

Whenever the processor executes a Unibus data transfer, the processor enters a sequence of machine states in which the Bus Shift Register (BSR) controls a series of operations. At all other times, the BSR is in a state decoded as BSR state 0. The specific sequence of BSR states depends on which of the four types of transfers is being done; normally the BSR state 1 is the initial non-zero state, but the modified DATO cycle, which is used following a DATIP transfer, begins in BSR state 12.

The following discussion refers to the flow diagrams of the KA11 Bus Flow (drawing FD-KA11-0-KBF, 2 sheets). The flow chart illustrates the timing and all the significant signals involved in Unibus transfers and processor control; therefore, only a general discussion of the purpose of each operation is given.

Each data transfer can be divided into three or four major parts. First, the address with which the transfer is to be conducted is calculated and stored in the Bus Address Register (BAR). Second, the data are fetched into the data paths. Optionally, some manipulation of the data may occur at this time. Finally, the data is transferred from the processor data paths to the destination of the transfer. For DATI and DATIP transfers, the destination of the transfer is one of the latches on the data paths; thus, no further operations are required. For DATO and DATOB transfers, the data is transferred into the data paths from the latches, and the destination is the bus address specified by the calculated address.

Address calculation is done in BSR states 1, 3, and 7. Modified DATO and DATOB transfers (DATO#) bypass these states, because the address used in the preceding transfer (a DATIP) is still in the BAR and is reused. This address calculation should not be confused with the total calculation of operand addresses, which requires the use of the Source and Destination major states and sequences of this simpler operation.

Non-processor requests (NPRs) can occur at the end of any bus cycle (except DATIP). During each bus cycle, the arbitration logic first checks for an NPR request (since these requests always take precedence over processor use of the bus). If an NPR is present, the logic issues an NPG signal and receives a selection acknowledge (SACK) signal in return from the requesting device. This procedure occurs simultaneously with the current data transfer.

3.7.2 BSR1

During BSR1, the instruction, major state, and ISR state are monitored by the Register Control module to select a processor register that contains the address base of the Unibus location with which the transfer is being done. In some cases, the contents of the register are decremented before the BAR is loaded. Decrementing is done by loading a constant (-2 for word transfers and for all references to the SP or PC; -1 for byte transfers) into Latch A at the same time that the contents of the register are loaded into Latch B.

The contents of the selected register are gated into Latch B except during a Deposit console function or during certain operations of the operand access major states. For the last transfer to access an indirectly addressed operand, the address is already in Latch B from the previous transfer, and no register is gated. This occurs in ISR3 for address modes 3 and 5, and in ISR7 for address mode 7. In ISR3 with address modes 6 and 7, the contents of the selected register are added to the contents of Latch B, which contains an index or offset word accessed by the previous transfer. The contents of the register are therefore gated into Latch A.

3.7.3. BSR3

At the beginning of BSR3, the output of the adder paths is clocked into the BAR. The processor register addressed in BSR1 continues to be accessed, and positive constants (1 for a byte address or 2 for a word address) can be added by forcing Latch A and Carry 00 to the adder to the appropriate value. The sum will appear at the outputs of the data paths and may be written back into the addressed register during BSR7. If no constant is added, the sum is simply the address clocked into the BAR; this may include a negative constant. The only time that a value other than +2, +1, 0, -1, or -2 can be added to the contents of a register is during ISR3 of Source or Destination with address mode 6 or 7; the contents of the register remain unchanged during such an indexing operation, as no write is done during BSR7.

Also during BSR3, the processor senses the state of the Bus SSYN signal. If SSYN is asserted, the slave device from the previous bus transfer is still active, and the bus is not available. Therefore, the processor stops until SSYN is cleared and the processor can enter BSR7. If the bus operation that the processor is doing is a DATI or DATIP, MSYN is asserted at the beginning of BSR7; the BAR is always gated to the bus A lines provided the processor or console has bus control.

From BSR3, the processor shifts to BSR7, unless an odd address error has been detected. An odd address for anything other than a byte operand causes the processor to enter BSR0 instead of BSR7 and simultaneously enter Service major state to execute a trap service sequence.

3.7.4 BSR7

The address calculation and setting of the BAR is the same for all types of bus data transfers and is completed in BSR 1 and 3. In BSR7, the operations performed depend on the type of transfer. For an input transfer (a DATI or a DATIP), MSYN is asserted; the register contents are written into the register; and the processor stops to await either a SSYN signal or a timeout restart. For an output transfer (a DATO), the register contents are written and the processor enters BSR0. The actual transmission of data on the bus is delayed to allow the processor to get the data from a processor register and put it in the data paths.

After the processor has transmitted an address and MSYN for an input transfer, the processor stops until it receives an SSYN signal. If no signal is received for 25 μs, the processor restarts, enters Service major state, and does a trap service sequence for a Timeout Error. Otherwise, the processor receives SSYN, waits 100 ns for the data lines to settle, and gates the data in Latch B (except for operands in certain instructions that complement the data; these are received in Latch A).

The processor release function, which allows other devices access to the Unibus, occurs at the end of the bus transfer in an input operation as well and, in addition, can occur after the byte swapping operation which follows the input of an odd byte of data.

3.7.5 BSR15 and 14

The processor determines whether to use the entire word of data or only one byte; if only the low byte is required, the data is handled as a complete word, and the processor must keep track of signs, overflows, and byte sums until the data is transferred out of the processor. A byte with an odd address is referred to as a high byte and requires special handling to permit the proper operation of the data paths. The byte must be shifted to the even, or low, byte half of the data paths for proper operation. This shift is done by passing the data from the high byte through the byte swapping gates on the outputs of the data paths, writing the modified word into the Temp register, and then reading the word back into the appropriate latch.

NOTE
The following discussion refers only to DATI and DATIP operation since output data is not handled under the bus cycle.

The swapping and writing are done in BSR15, the reloading of the latch in BSR14. Following these operations the processor enters BSR0, simultaneously shifting to the next ISR state as required by the present machine state. The processor release function, which allows other devices access to the Unibus, occurs at the end of the bus transfer in an input operation as well and, in addition, can occur after the byte swapping operation that follows the input of an odd byte of data. When no byte swapping is done, the processor enters BSR0 directly from BSR7 for input transfers.

3.7.6 BSR0 for Output Transfers

For output transfers, the processor enters BSR0 from BSR7. Note that the only time that the processor does byte output transfers is for byte operands. Because the DATOB done during execute follows a DATIP, the DATOB is modified and the BSR enters BSR12 directly without entering BSR7. Any justification of odd-byte data is done by the rotate/shift gating data paths. In BSR0 the contents of the appropriate processor register are loaded into Latch B, and Latch A is cleared. The processor then enters BSR12. This use of BSR0 for an output transfer does not compromise the use of BSR0 as a rest state in other situations.

3.7.7 BSR12

When an output transfer follows a DATIP bus operation, the address with which the transfer is done remains the same (in the BAR), and the data are supplied directly from the data paths, rather than from a register. Therefore, a modified form of the DATO and DATOB operations is possible (this modified form is called DATO#), and the variable state sequence used by the processor accomplishes this DATO# output transfer by entering BSR12 directly. This occurs only in Execute major state, following the DATIP that terminates the Destination major state. The value placed in the BAR during the DATIP is used again, and the data transmitted are the outputs of the data paths, which are the result of whatever manipulations of data have occurred. Justification of output odd-byte data occurs in the rotate/shift gating of the data paths in the Execute major state. Unlike input justification it is not part of the bus cycle.

For either type of output transfer, the address is in the BAR, and the data are in the data paths at the beginning of BSR12. The processor first determines the state of the SSYN bus line, delaying any operations if SSYN is asserted, to allow previous bus operations to be completed. Then the processor gates address, data, and control lines to the bus and shifts to BSR8.

3.7.8 BSR8

As the processor enters BSR8, MSYN is asserted and the processor stops. The processor clock is restarted by either a SSYN signal or, after 10 μs, a Timeout. If the clock is restarted by a SSYN signal, the processor clears the bus lines and, if no other device has control of the bus, proceeds with the next machine state. If the restart was caused by a timeout trap, the processor enters Service major state to execute the trap service sequence. The processor release function, which allows other devices access to the Unibus, occurs at the end of the bus transfer in an input operation as well, and in addition can occur after the byte swapping operation which follows the input of an odd byte of data.

3.8 FETCH MAJOR STATE

Drawing D-FD-KA11-0-KIF, Sheet 1 illustrates the Fetch major state. When the processor is first turned on, or receives a reset instruction, it is in SERVICE*ISR0. When the start switch is pressed, Fetch and ISR0 (also BSR1) states are entered. In Fetch and ISR0, the processor does a DATI bus transfer, using the contents of the Program Counter (PC) register for the address. The address is incremented by two before being re-written into the PC, and the data received from the Unibus are clocked into the Instruction Register (IR) as well as into Latch B.

The processor then shifts to ISR1 to decode the contents of the IR. The specific instruction decoded determines the next major state and the next ISR state and, in addition, may cause the setting of certain flags and condition code bits. The latter are set if the instruction that is decoded is a condition codes operate instruction. Halt and reset instructions set the halt flag, while trap instructions set a trap flag. If the trace bit in the processor status word (PS) is set, the trace flag is set.

Branch instructions which have conditions unmet do not enter other states besides Fetch, nor do condition code operate instructions. The condition code operate instruction changes the condition code bits in the STATUS word. This change in the status word does not require entry into Service and ISR 2. All other instructions enter at least one major state besides Fetch. For HALT, WAIT, RESET, EMT and TRAP instructions, the only other state entered is Service; the rest of the legal instructions require operations upon data from the Unibus or from the IR, and enter either Execute or an operand access major state.

Note that TRAP instructions include all illegal instructions and reserved instruction codes; the trap vectors for these classes of codes are different from the trap vectors for the trap type instructions, but the basic trap service sequence in Service major state is the same.

3.9 OPERAND ACCESS MAJOR STATES

The operations that are done in the Source and Destination major states are very similar. Therefore, the two major states are illustrated by a common flow chart (Drawing D-FD-KA11-0-KIF, Sheet 1) and are discussed together.

The operations that transfer an operand from a Unibus location to the central processor are entirely dependent on the address mode by which the operand is selected. These major states are the clearest example of the variable state sequence by which the processor performs only the operations required and avoids unproductive machine states.

3.9.1. Address Modes

For an instruction that can access data from the Unibus, an operand access major state is entered only if the address mode is not zero (data in a general register). As discussed in Paragraph 3.6.2, the processor skips the major state if the address mode is zero and gets the operand from the processor register during the Execute major state. Address mode zero is illegal for the JMP and JSR instructions.

If an operand access major state is entered, the processor first calculates the address of the operand and then transfers the operand to the processor. In Source major state the processor stores the operand in the Source register until the beginning of the Execute major state, when it is transferred to a latch; this frees the data paths during the Destination major state. The address calculation that precedes the operand transfer may be as simple as loading the BAR from a processor register, or as complex as performing two Unibus transfers, with additional address calculation to load the BAR for each one. The following paragraphs discuss the specific operations that perform the address calculation for each non-zero address mode; in every case the address calculation is followed by the remaining steps of a DATI (or DATIP) bus operation, as described in Paragraph 3.7.

3.9.1.1 **Address Mode 1** – For address mode 1, a single bus operation is done. The address calculation is performed by the bus operation in fetching the address from the specified processor register; the data are obtained and stored.

3.9.1.2 **Address Mode 2** – Address mode 2 is similar to address mode 1, with the distinction that the bus operation increments the contents of the register. The register is incremented by one for a byte operand or by two for a word operand; if the register is the PC or the Stack Pointer (SP), the increment is always by two. The increment is performed *after* the register contents are used as an address.

3.9.1.3 **Address Mode 3** – Address mode 3 performs a bus operation identical to that performed for address mode 2 but does not store the transferred data. Instead, the data are loaded into the BAR, and a second bus operation is performed to fetch the data. The appropriate register is incremented *after* use.

3.9.1.4 **Address Mode 4** – Address mode 4 is similar to address mode 2, except that the change in the contents of the accessed register is a decrement rather than an increment. The decrement is performed *before* the contents of the register is used as an address.

3.9.1.5 **Address Mode 5** – Address mode 5 is the same as address mode 3, except that a decrement is performed instead of an increment.

3.9.1.6 **Address Mode 6** – Address mode 6 requires two bus operations. The first is a DATI, which uses the same address calculation as the DATI in Fetch major state; the PC is loaded into the BAR, incremented by two, and written back into the PC. The second transfer uses a unique address calculation; the contents of the specified register is added to the contents of the first word that was transferred, and the sum is loaded into the BAR. The word transferred by this DATI is the data.

3.9.1.7 **Address Mode 7** – Address mode 7 uses three bus operations. The first two are the same as the two in address mode 6, but the result is not stored as data; instead it is loaded into the BAR, and a third bus operation occurs, which transfers the desired operand to the processor.

3.9.2 Disposition of the Operand

The major difference between the two operand access major states is the disposition of the data. Where the Source major state temporarily stores the operand in a processor register, the Destination major state forces the last bus transfer to be a DATIP and leaves the result in a latch (unless the instruction is a JMP or JSR, for which a DATI is aborted and the address is stored in the PC or Temp register, respectively).

3.9.3 Entries to and Exits from Source and Destination

Both Source and Destination are entered in ISR1. The last ISR state in Fetch is also ISR1; therefore, if the processor enters an operand access major state from Fetch the Instruction Shift Register is not changed, and two ISR1 states are performed in succession. Because both operand access states end in ISR0, the ISR must be shifted to a 1 state if the processor enters Destination from Source. The ISR state entered when entering Execute major state depends on the instruction to be executed; the exits from the Fetch, Source, and Destination major states select the appropriate ISR state (and BSR state) as shown on the flow charts.

3.10 EXECUTE MAJOR STATE

The operations that take place in the Fetch major state are generally the same for all instructions; the operations done in the operand access major states are controlled by the address mode, not the specific instruction. In the Execute major state, the sequence of events that takes place varies for almost every instruction. (See Drawing D-FD-KA11-0-KIF, Sheets 2 through 4).

The flow charts for Execute are divided into five major sections. The first section (Sheet 2) illustrates the operations done for all data manipulation instructions. The remaining sections illustrate the sequences of operations for four types of processor control instructions; Branch, RTI, and RTS instructions, which do not require operands transferred from Unibus locations during an operand access state, and the JSR instruction, which requires an address only.

3.10.1 Data Manipulation Instructions

Data manipulation instructions include three groups of instructions and one unique instruction. The three groups are: two-operand instructions; one-operand instructions; and rotate/shift instructions. In addition, the SWAB instruction is performed by a similar sequence of operations.

3.10.1.1 **Entries into Execute** – There are two entries into Execute for data manipulation instructions. If the destination operand is specified with an address mode of zero, the entry is in ISR0 to perform a transfer from the selected register to a latch. The particular latch that the data is gated into is a function of the specific instruction, as shown on the flow charts. If the address mode of the Destination operand is not zero, the entry into Execute is an entry into ISR1; the destination operand must be in the appropriate Latch as a result of the operations conducted in the Destination major state.

3.10.1.2 **The Extra ISR States** – For most data manipulation instructions, the data modifications are complete when the data have been loaded into the latches through the selected gates; for example, a COM instruction loads the operand into Latch A through a complementing set of gates, while the BIS instruction loads both operands into Latch B, holding the latch set for both operations, to produce the inclusive OR of the operands. The only remaining operation is to store the data in the destination address.

However, two double-operand instructions (BIT and BIC) and Rotate/Shift Byte instructions operating on an odd byte require extra machine states to complete the data manipulations. The BIT and BIC instructions perform variations of a logical AND on the operands by complementing the OR of the complements (using de Morgan's principal); this procedure requires a second pass through the data paths to perform the final complement. Therefore, the partial result must be stored in the Temp register in ISR3 and then gated into Latch A in ISR7.

Normally, the Rotate/Shift instructions use the shift gating on the outputs of the data paths in ISR15, when the data from the Latches is transmitted through the shift gating to the bus gating or Register input. However, when a byte operand with an odd address is manipulated, the swap byte gating must be asserted in ISR15; thus, the shift gating operation is performed in the two extra states. This situation requires the same operations of writing the operand into a register and then reading it back into Latch A that is required for the BIT and BIC instructions, but the data manipulation is performed during the write port'on, rather than during the read portion.

3.10.1.3 **Disposition of the Result** – The final ISR state entered for the Execute major state of data manipulation instructions is ISR15. In ISR15, the result of the data manipulation is transferred to the destination operand address. If this address is external, the processor does a modified DATO or DATOB operation. This DATO# operation follows the last transfer in the Destination major state, which was a DATIP transfer; therefore, the same address is left in the BAR, and the processor enters BSR12 at the same time it enters ISR15.

If the destination operand is internal, the processor writes the data into the selected register. For Test instructions, which include BIT, CMP, and TST, the only operation done is the loading of the status register from the selected source as shown in the flow charts. A DATIP was not the last bus operation and no DATO or DATOB is necessary; the Destination data is unchanged.

The Processor Status word (STATUS) includes the condition codes, which are normally clocked after the outputs of the data paths are settled. If the destination operand address is the address of STATUS, the processor must enter an additional ISR state in SERVICE to allow the new STATUS to be compared against any previous commitment to servicing a BUS REQUEST. This time state is provided by entering ISR2 of the Service major state.

3.10.2 JSR Instructions

When the processor enters Execute for the JSR instruciton, the address of the destination operand is in the Temp register, and the processor is in ISR0 and BSR1. A bus operation is done, which is a DATO using the decremented Stack Pointer (SP) register for the address, to store the contents of a register (which is selected by bits 8 through 6 of the IR) in a bus location that acts as a location on the push down stack. If the transfer is successful and the stack pointer does not decrement to a value of less than 400 (the lowest 400 locations of bus addresses are reserved for interrupt and trap vectors and are protected from the hardware stack by an overflow trap), the processor enters ISR1 to load the contents of the PC into Latch B.

The processor then sequences into ISR3 to write the contents of the data paths into the register, the previous contents of which were transferred to the stack. This register is selected by the same bits of the IR that select the register used to address source operands.

The next two ISR states repeat the process of reading a register into Latch B and then writing the contents into a register, the previous contents of which have been saved. The contents of the Temp register, which represent the address calculated in an aborted Destination major state, are transferred to the PC. This step completes the processing for a JSR instruction, and the processor exits from the Execute major state into either Fetch or Service; the state entered depends on external requirements for control of the Unibus and on the state of the flags in the processor.

3.10.3 Branch Instructions

When a branch instruction is unconditional or has conditions that are met by the present state of the processor, the sequence of operations transfers directly from Fetch to Execute. The processor reenters ISR1, with the instruction word in Latch B. The offset portion of this word is written into the Temp register with one modification; it is shifted one bit position left, which is equivalent to multiplying the offset value by two.

The ISR then shifts to ISR3 state, and the contents of the Temp register are read into Latch B. The offset is a two's complement number, but it is only eight bits long, so the sign bit must be extended to the rest of the high byte by the sign extension logic.

A second read operation occurs in ISR7 to transfer the contents of the PC to Latch A. The sum of the numbers in the two latches is written back into the PC in ISR15. This sum is the offset address at which the next instruction to be executed is stored.

The exit from Execute for a branch instruction is dependent only on the state of the processor flags and on requests for control of the Unibus.

3.10.4 RTS Instruction

The Return from Subroutine (RTS) instruction is the reverse of the JSR instruction. The order and direction of the transfers of data are reversed. The processor enters Execute and ISR15 directly from the Fetch major state and transfers the contents of a register specified by bits 2 through 0 of the IR (the same bits that specify the selected register for a Destination operand) to Latch B.

The processor then shifts to ISR14 and writes the output of the data paths into the PC, which reverses the storing of the PC during the execution of the JSR. The processor continues by entering ISR12.

In ISR12, the processor does a DATI bus operation, using the contents of the SP register for an address. The SP is incremented by 2. When the outputs of the data paths are written into the selected register, this reverses the stacking of the contents of the selected register in the JSR instruction.

3.10.5 RTI Instruction

The Return from Interrupt (RTI) instruction has the same relationship to the Interrupt and Trap service sequence (occurring in the Service major state) that the RTS instruction has to the JSR instruction. The transfers of data are made in the opposite order and in the opposite direction from those made in Service.

The RTI instruction, like the RTS instruction, sequence directly from the Fetch major state to Execute and ISR15. In ISR15, a DATI bus operation is done, similar to that in ISR12 for the RTS instruction, to transfer a word from the hardware stack to the processor. In ISR14, this word is written into the PC, reversing the stacking of the PC that occurs in ISR7 of Service.

The processor continues into ISR12, where a second unstacking bus transfer occurs. The data are written into the processor Status word (STATUS) in ISR8 and in ISR2 of Service, reversing the stacking operation that occurs in ISR3 of Service. ISR2 of Service is also entered.

3.11 SERVICE MAJOR STATE

Service major state is not a part of most instructions; with the exception of trap, halt, wait, and reset instructions, Service is entered only if external BUS REQUESTS for Unibus control or internal processor flags require the asynchronous processes that take place in Service.

The Service major state consists of two parts: the processes that occur in ISR2 and ISR0 and the trap and interrupt service sequence performed in seven ISR states beginning with ISR1.

3.11.1 Condition Code Clocking in ISR2

When the Service major state is entered from Execute with data to be loaded into the Processor Status word (STATUS) (This occurs after the execution of an RTI instruction or after any data manipulation instruction that addresses STATUS as a destination operand.), the processor enters ISR2 to reclock the priority determination circuits for BUS REQUESTS. This is necessary because the processor priority has been altered.

ISR2 is separate from the rest of the Service major state, because the processor can proceed either to Service and ISR0 or to Fetch and ISR0. The major state entered is determined by the presence or absence of requirements for the granting of BUS requests.

3.11.2 Priorities for Service in ISR0

When ISR0 is entered, the processor determines what requirements for bus mastership are present and performs the operations necessary for the highest priority requirement. The priority is determined by the order in which the need for processor action is determined: traps (such as trap instruction, illegal instructions, power fail traps, or the halt flag) are serviced first; if there are no traps to be serviced, requests for Unibus control are serviced; if there are neither traps nor requests, the wait state can be entered.

Because the trap service sequence enters other ISR states and the request and wait service sequences normally do not, the sequences are discussed in the reverse order from their priority.

3.11.3 Wait Service

When a WAIT instruction is decoded in the Fetch major state, the processor enters the Service major state. The processor continues to recycle through Service and ISR0 until a higher priority requirement for service takes precedence. Typically, a program will execute the WAIT instruction after setting up conditions where a device will cause a BUS REQUEST; when the BUS REQUEST from the device is received, the processor will release control of the bus because the request service sequence has higher priority than the wait service sequence. If the BUS REQUEST results in an INTR sequence, the processor leaves the wait loop.

3.11.4 Request Service

At the end of every processor-controlled bus transfer and several times during each major state, the processor clocks requests for bus mastership into a set of flip-flops (one for each bus request level). The priority arbitration logic in the processor determines whether any request has a higher priority than the processor; if such a request exists, an NPR is immediately granted or at the end of an instruction the processor enters the Service major state and responds to a BUS REQUEST.

When the processor services a request, the processor clock stops, the processor clears the BBSY bus line, and the highest priority requesting device is granted control of the Unibus. The console has a higher priority than any other device. When the SACK and BBSY bus lines are both cleared, the processor clock is restarted.

3.11.5 Interrupt Recognition

If the device which gained control of the Unibus under a BUS REQUEST performs an INT (interrupt) bus operation, the processor responds as a Slave device by clocking the contents of the bus D lines into Latch B and setting the Interrupt flag (INTRF). When the processor restarts after the peripheral releases control, the processor enters the trap and interrupt service sequence. If the device that had control of the bus did not do an interrupt operation and the last instruction was a WAIT instruction, the processor continues to cycle through Service in a wait service sequence. If neither a wait service sequence nor an interrupt service sequence is required, the processor enters the Fetch major state.

3.11.6 Trap and Interrupt Service

Either an internal trap (such as a trap instruction or a stack overflow) or an interrupt from a bus device can cause the processor to enter the trap and interrupt service sequence. For an interrupt, the vector address is loaded into Latch B by the INTR bus operation; for a trap, the vector address is loaded into Latch B during ISR0 from the Special Trap Markers (STPM) input. The vector address is used to locate two words, which are loaded into processor registers to begin a program for servicing the cause of the service requirement. Refer to logic descriptions and prints K3-2 (Priority) and K15-2 (Power Fail) for a discussion of power fail servicing.

3.11.6.1 **ISR1** – The processor shifts to ISR1, and the contents of the data paths are written into the Temp register to be stored until ISR15. The processor then shifts to ISR3 unless the power up flag (PUPF) is set; for a power-up trap, the contents of the registers (particularly the stack pointer) are unknown, so the stacking operations must be avoided by transferring directly to ISR15.

3.11.6.2 **ISR3** – In ISR3 the processor does the first of two stacking operations. The contents of the Processor Status word (STATUS) are transferred to a Unibus location, the address of which is the contents of the Stack Pointer (SP) decremented by two. This transfer is done by a DATO bus transfer. The contents of the SP are loaded into Latch B, and Latch A is loaded with a -2 during BSR1; the Bus Address Register (BAR) is loaded in BSR3; in BSR7 the decremented contents of the SP are written back into that register; in BSR0 the contents of STATUS are loaded into Latch B; in BSR12 the processor transmits the data and address and stops the processor clock; and when the clock restarts, the processor enters BSR8 and completes the transfer.

3.11.6.3 **ISR7** – The processor shifts to ISR7 and repeats the stacking operation with the contents of the PC as data. The data are transferred to a word the address of which is two less than the address of the word storing the contents of the PS.

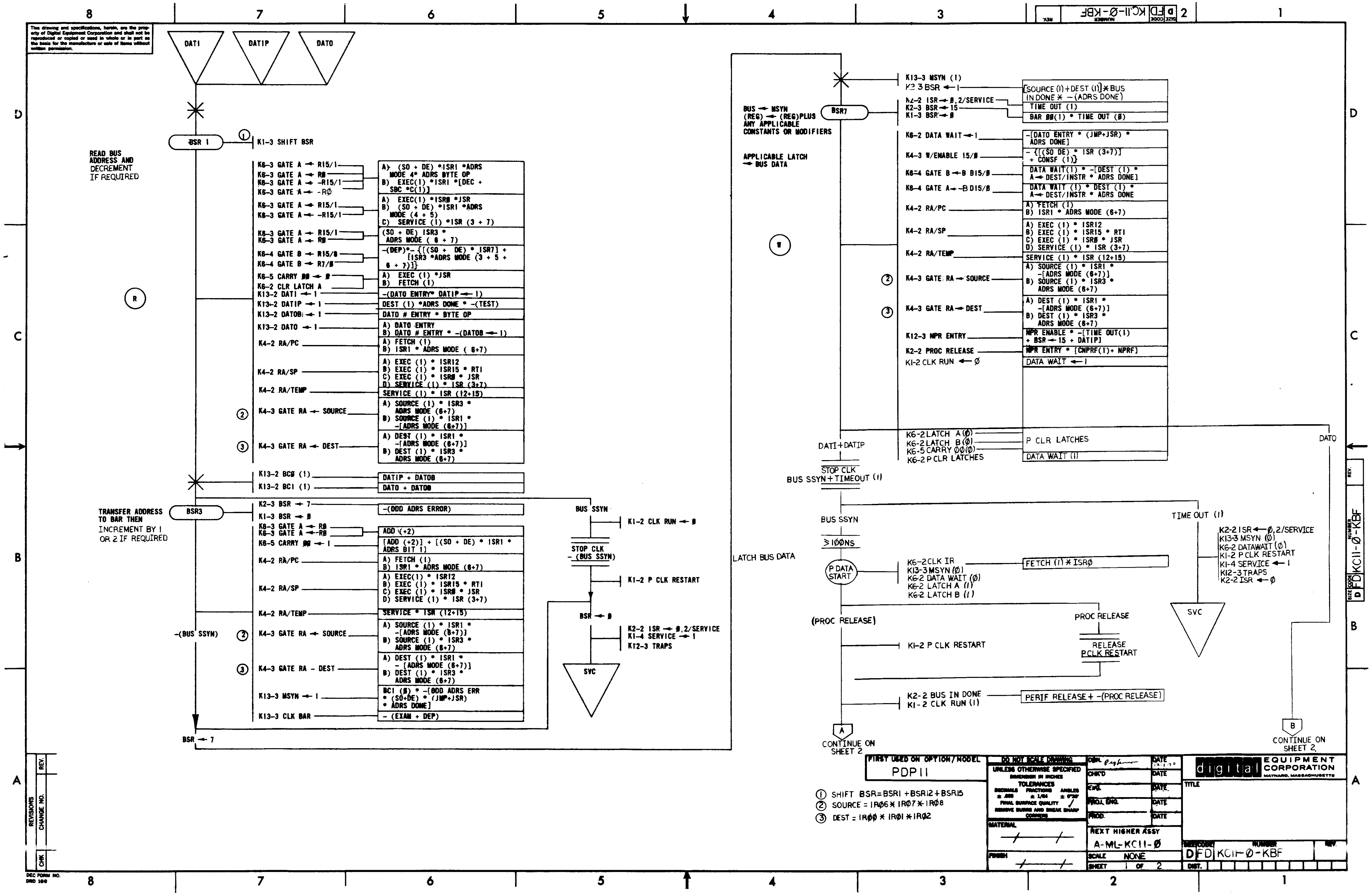
3.11.6.4 **ISR15 and ISR14** – When the processor enters ISR15, the contents of the Temp register are loaded into the BAR, and the processor does a DATI bus operation to load the contents of the word addressed by the BAR into Latch B. The contents of the Temp register are incremented by two after the BAR is loaded. The processor shifts to ISR14, and the contents of the data paths are written into the PC, replacing the previous contents with the address of the first instruction of the service routine.

3.11.6.5 **ISR12 and ISR8** – The processor now does a second DATI operation in ISR12, again using the contents of the Temp register for an address and reloading Temp with the result of incrementing that address. The second word is loaded into the processor status register during ISR8, replacing the old STATUS word which includes the processor priority, the contents of the condition codes, and the trace bit. The contents are replaced with new values appropriate to the service routine.

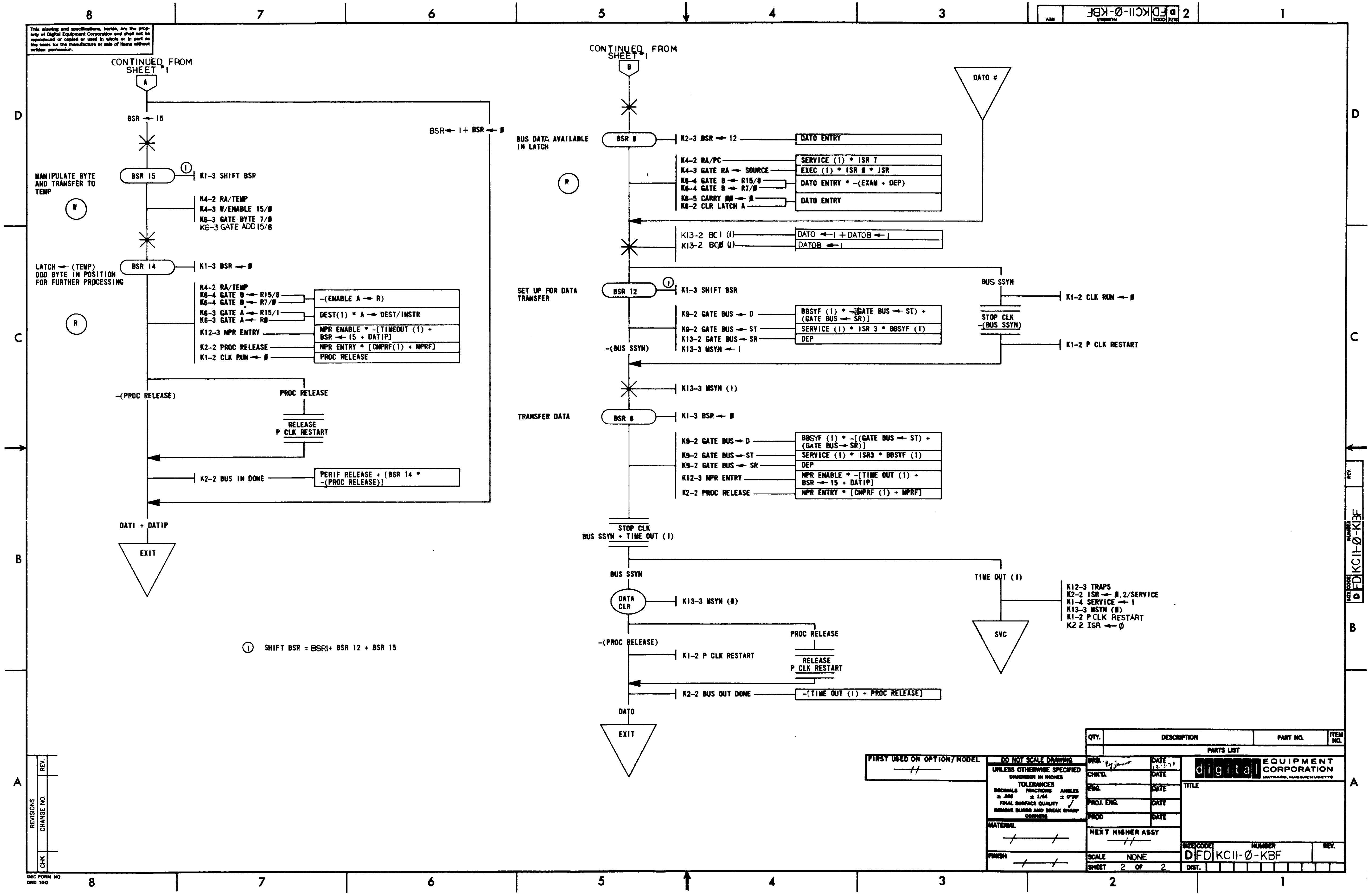
3.11.6.6 **Exits from the Service Sequence** – When the service sequence is done, the processor checks that no internal flags have been set during the performance of the sequence. The overflow flag may be set if the trap and interrupt service sequence has caused the stack to store words at locations with address below 400_8. If the overflow trap or any other flag is present, the processor reenters Service to handle the flag; otherwise, the processor enters Fetch to get the first instruction of the program that services the original interrupt or trap.

3.12 OPERATIONS DONE WHEN THE HALT FLAG IS SET

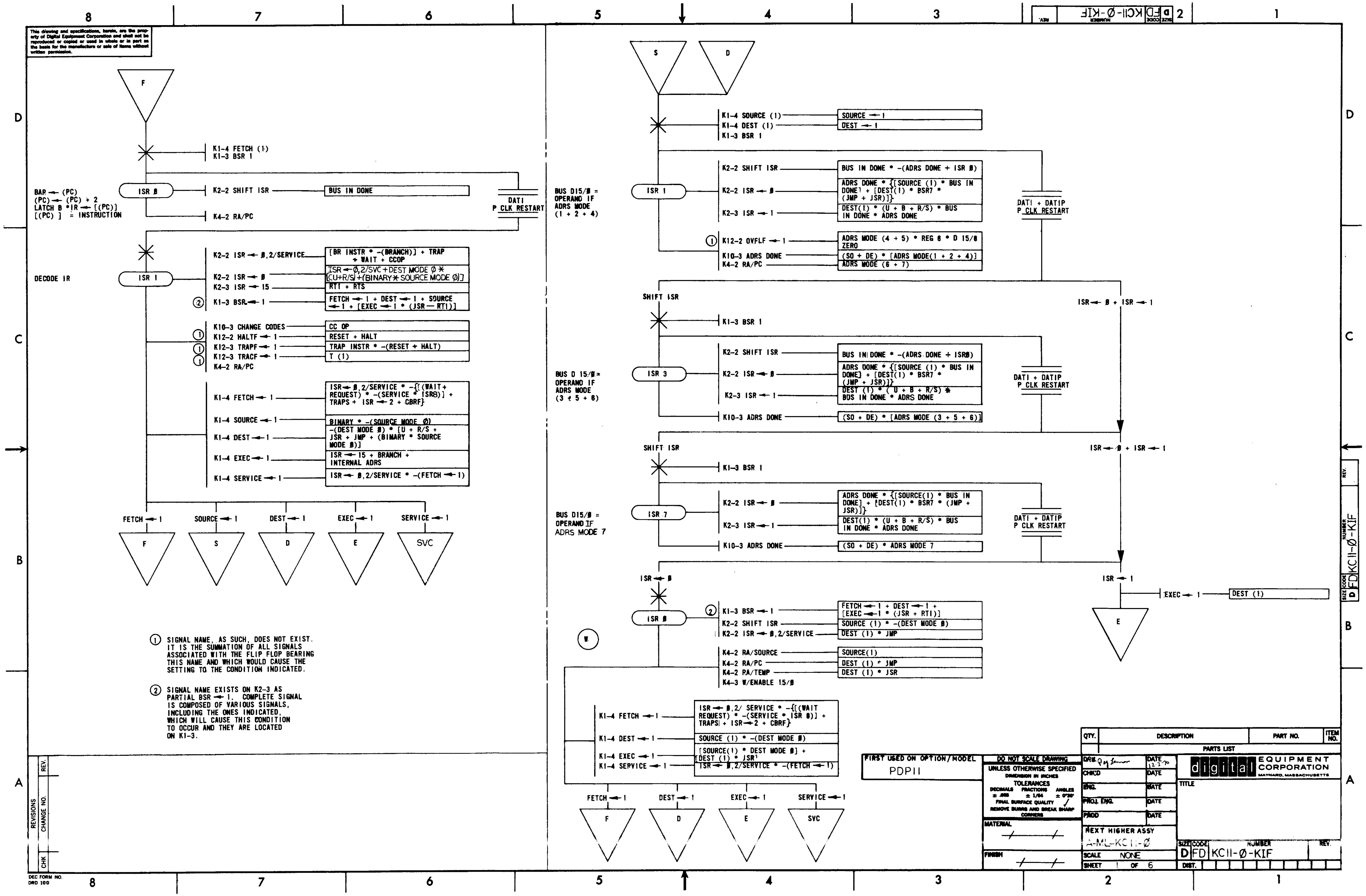
The Halt Flag (HALTF) is set by one of three occurrences: a halt instruction, a reset instruction, or a bus error (such as an odd address error) occurring before a previous bus error is serviced. If the HALTF is set, the processor releases control of the Unibus to the console, thus effectively stopping until operator intervention can be supplied. For a Reset instruction, the halt is accompanied by a 20 ms initialization level, which resets the processor and all devices on the Unibus to a starting condition; a restart pulse occurs afterward. The processor can also be halted with control transferred to the console by the assertion of a Console Bus Request (CBR) or a Console Non Processor Request (CNPR), which occurs when the Halt/Enable switch on the console is in the Halt position and the Single Instruction/Single Cycle switch is in the Single Instruction or Single Cycle position, respectively; however, these transfers of control occur only if no higher priority (internal) requirements for service occur.



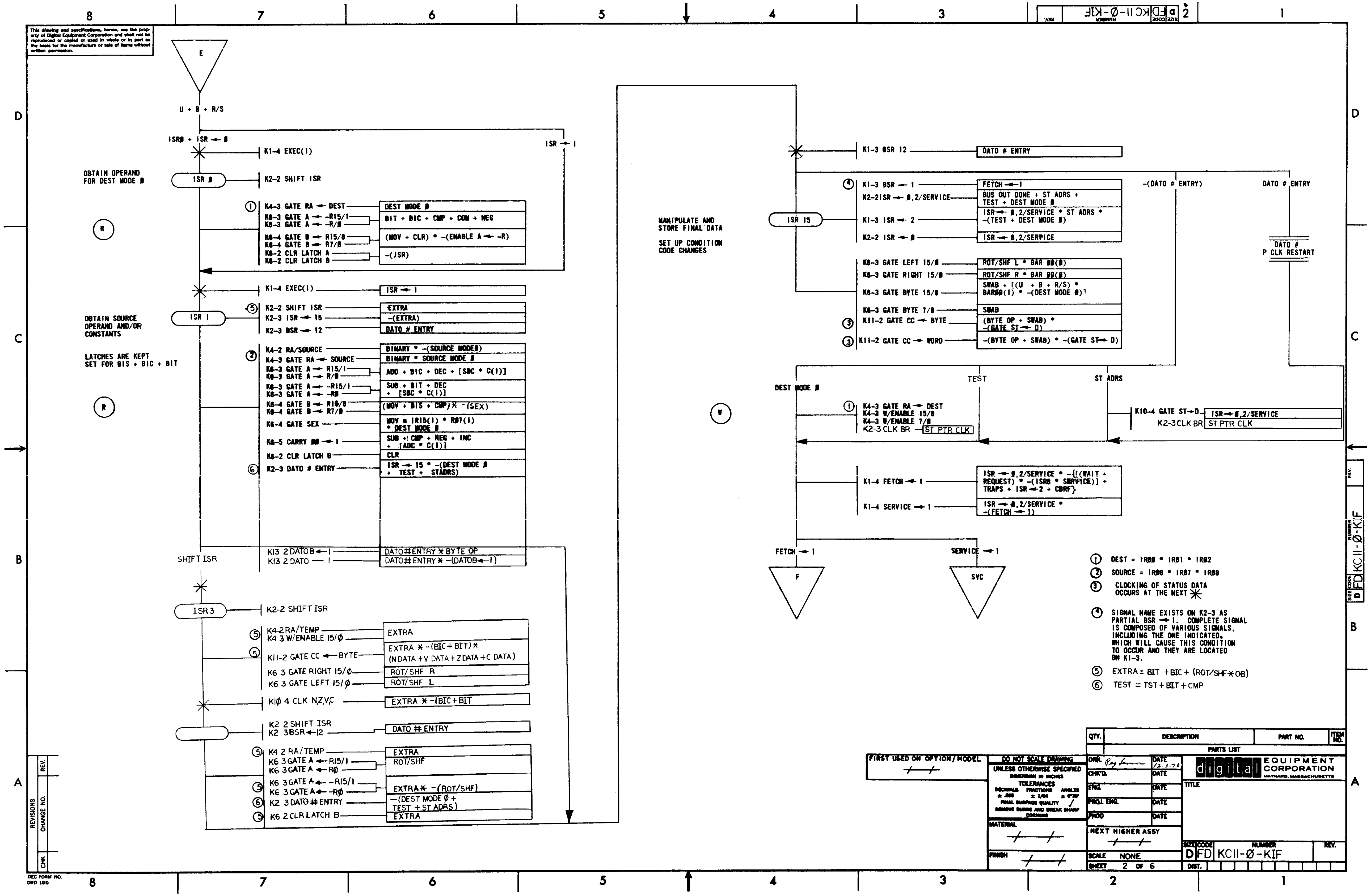
KA11 Bus Flow



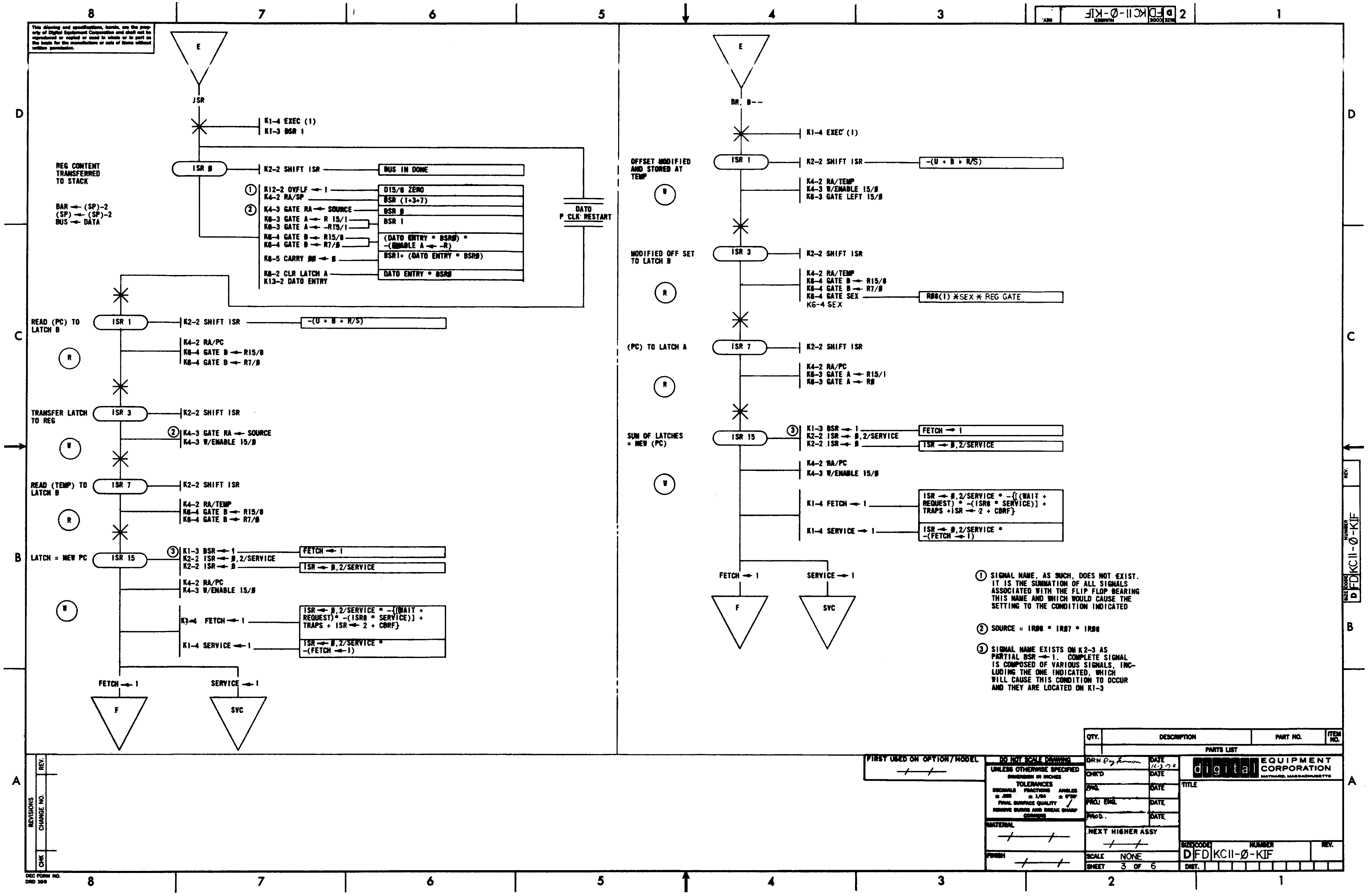
KA11 Bus Flow


This drawing and specifications, herein, are the property of Digital Equipment Corporation and shall not be reproduced or copied or used in whole or in part as the basis for the manufacture or sale of items without written permission.
F
K1-4 FETCH (1)
K1-3 BSR 1
BAR ← (PC)
(PC) ← (PC) + 2
LATCH B *IR ← [(PC)]
[(PC)] = INSTRUCTION
ISR ∅
K2-2 SHIFT ISR
BUS IN DONE
K4-2 RA/PC
DATI
P CLK RESTART
DECODE IR
ISR 1
K2-2 ISR ← ∅,2/SERVICE
[BR INSTR * -(BRANCH)] + TRAP + WAIT + CCOP
K2-2 ISR ← ∅
ISR ← ∅,2/SVC + DEST MODE ∅ * [CU+R/S]+(BINARY * SOURCE MODE ∅)]
K2-3 ISR ← 15
RTI + RTS
2
K1-3 BSR ← 1
FETCH ← 1 + DEST ← 1 + SOURCE ← 1 + [EXEC ← 1 * (JSR — RTI)]
K10-3 CHANGE CODES
CC OP
1
K12-2 HALTF ← 1
RESET + HALT
1
K12-3 TRAPF ← 1
TRAP INSTR * -(RESET + HALT)
1
K12-3 TRACF ← 1
T (1)
K4-2 RA/PC
K1-4 FETCH ← 1
ISR ← ∅,2/SERVICE * -{[(WAIT + REQUEST) * -(SERVICE * ISR8)] + TRAPS + ISR ← 2 + CBRF}
K1-4 SOURCE ← 1
BINARY * -(SOURCE MODE ∅)
K1-4 DEST ← 1
-(DEST MODE ∅) * [U + R/S + JSR + JMP + (BINARY * SOURCE MODE ∅)]
K1-4 EXEC ← 1
ISR ← 15 + BRANCH + INTERNAL ADRS
K1-4 SERVICE ← 1
ISR ← ∅,2/SERVICE * -(FETCH ← 1)
FETCH ← 1
SOURCE ← 1
DEST ← 1
EXEC ← 1
SERVICE ← 1
F
S
D
E
SVC
1 SIGNAL NAME, AS SUCH, DOES NOT EXIST. IT IS THE SUMMATION OF ALL SIGNALS ASSOCIATED WITH THE FLIP FLOP BEARING THIS NAME AND WHICH WOULD CAUSE THE SETTING TO THE CONDITION INDICATED.
2 SIGNAL NAME EXISTS ON K2-3 AS PARTIAL BSR ← 1. COMPLETE SIGNAL IS COMPOSED OF VARIOUS SIGNALS, INCLUDING THE ONES INDICATED, WHICH WILL CAUSE THIS CONDITION TO OCCUR AND THEY ARE LOCATED ON K1-3.
S
D
K1-4 SOURCE (1)
SOURCE ← 1
K1-4 DEST (1)
DEST ← 1
K1-3 BSR 1
BUS D15/∅ = OPERAND IF ADRS MODE (1 + 2 + 4)
ISR 1
K2-2 SHIFT ISR
BUS IN DONE * -(ADRS DONE + ISR ∅)
K2-2 ISR ← ∅
ADRS DONE * {[SOURCE (1) * BUS IN DONE] + [DEST(1) * BSR7 * (JMP + JSR)]}
K2-3 ISR ← 1
DEST(1) * (U + B + R/S) * BUS IN DONE * ADRS DONE
DATI + DATIP
P CLK RESTART
1
K12-2 OVFLF ← 1
ADRS MODE (4 + 5) * REG 6 * D 15/8 ZERO
K10-3 ADRS DONE
(SO + DE) * [ADRS MODE(1 + 2 + 4)]
K4-2 RA/PC
ADRS MODE (6 + 7)
SHIFT ISR
ISR ← ∅ + ISR ← 1
K1-3 BSR 1
BUS D 15/∅ = OPERAND IF ADRS MODE (3 + 5 + 6)
ISR 3
K2-2 SHIFT ISR
BUS IN DONE * -(ADRS DONE + ISR∅)
K2-2 ISR ← ∅
ADRS DONE * {[SOURCE (1) * BUS IN DONE] + [DEST(1) * BSR7 * (JMP + JSR)]}
K2-3 ISR ← 1
DEST (1) * (U + B + R/S) * BUS IN DONE * ADRS DONE
DATI + DATIP
P CLK RESTART
K10-3 ADRS DONE
(SO + DE) * [ADRS MODE (3 + 5 + 6)]
SHIFT ISR
ISR ← ∅ + ISR ← 1
K1-3 BSR 1
BUS D15/∅ = OPERAND IF ADRS MODE 7
ISR 7
K2-2 ISR ← ∅
ADRS DONE * {[SOURCE(1) * BUS IN DONE] + [DEST(1) * BSR7 * (JMP + JSR)]}
K2-3 ISR ← 1
DEST(1) * (U + B + R/S) * BUS IN DONE * ADRS DONE
DATI + DATIP
P CLK RESTART
K10-3 ADRS DONE
(SO + DE) * ADRS MODE 7
ISR ← ∅
ISR ← 1
EXEC ← 1
DEST (1)
E
ISR ∅
2
K1-3 BSR ← 1
FETCH ← 1 + DEST ← 1 + [EXEC ← 1 * (JSR + RTI)]
K2-2 SHIFT ISR
SOURCE (1) * -(DEST MODE ∅)
K2-2 ISR ← ∅,2/SERVICE
DEST (1) * JMP
W
K4-2 RA/SOURCE
SOURCE(1)
K4-2 RA/PC
DEST (1) * JMP
K4-2 RA/TEMP
DEST (1) * JSR
K4-3 W/ENABLE 15/∅
K1-4 FETCH ← 1
ISR ← ∅,2/ SERVICE * -{[(WAIT REQUEST) * -(SERVICE * ISR ∅)] + TRAPS] + ISR ← 2 + CBRF}
K1-4 DEST ← 1
SOURCE (1) * -(DEST MODE ∅)
K1-4 EXEC ← 1
[SOURCE(1) * DEST MODE ∅] + [DEST (1) * JSR]
K1-4 SERVICE ← 1
ISR ← ∅,2/SERVICE * -(FETCH ← 1)
FETCH ← 1
DEST ← 1
EXEC ← 1
SERVICE ← 1
F
D
E
SVC
FIRST USED ON OPTION/MODEL
PDP11
DO NOT SCALE DRAWING
UNLESS OTHERWISE SPECIFIED DIMENSION IN INCHES
TOLERANCES
DECIMALS ± .005 FRACTIONS ± 1/64 ANGLES ± 0°30'
FINAL SURFACE QUALITY
REMOVE BURRS AND BREAK SHARP CORNERS
MATERIAL
FINISH
QTY.
DESCRIPTION
PART NO.
ITEM NO.
PARTS LIST
DR'N
DATE 12-1-70
CHK'D
DATE
ENG.
DATE
PROJ. ENG.
DATE
PROD
DATE
NEXT HIGHER ASSY
A-ML-KC11-∅
SCALE NONE
SHEET 1 OF 6
digital EQUIPMENT CORPORATION MAYNARD, MASSACHUSETTS
TITLE
SIZE D
CODE FD
NUMBER KC11-∅-KIF
REV.
DIST.
REVISIONS
REV.
CHANGE NO.
CHK
DEC FORM NO. DRD 100
8
7
6
5
4
3
2
1
D
C
B
A

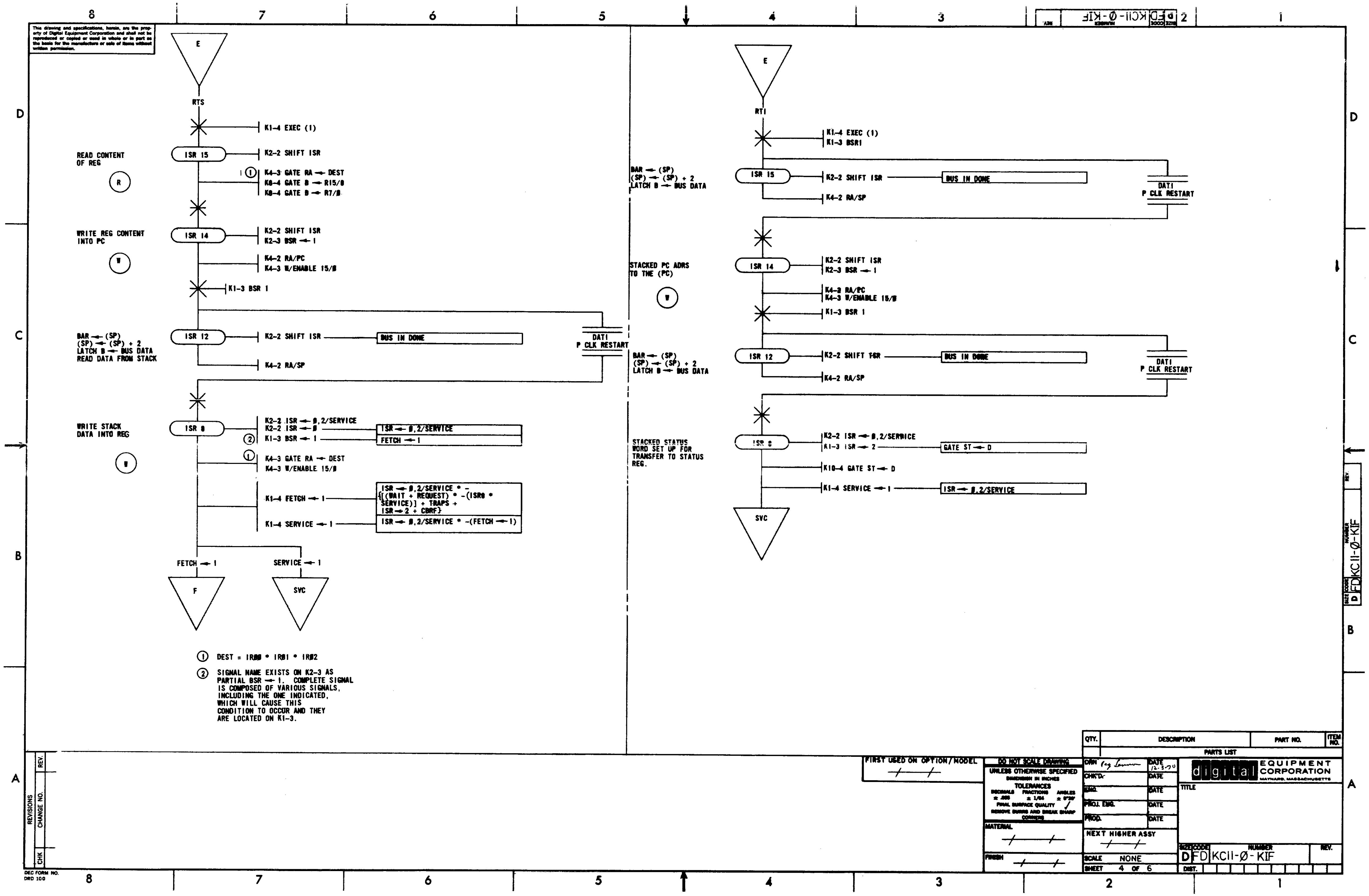
KA11 Instruction Flow


This drawing and specifications, herein, are the property of Digital Equipment Corporation and shall not be reproduced or copied or used in whole or in part as the basis for the manufacture or sale of items without written permission.
E
U + B + R/S
ISRØ + ISR ← Ø
ISR ← 1
K1-4 EXEC(1)
OBTAIN OPERAND FOR DEST MODE Ø
ISR Ø
K2-2 SHIFT ISR
K4-3 GATE RA ← DEST
DEST MODE Ø
K6-3 GATE A ← -R15/1
K6-3 GATE A ← -R/Ø
BIT + BIC + CMP + COM + NEG
K6-4 GATE B ← R15/Ø
K6-4 GATE B ← R7/Ø
(MOV + CLR) * -(ENABLE A ← -R)
K6-2 CLR LATCH A
K6-2 CLR LATCH B
-(JSR)
R
K1-4 EXEC(1)
ISR ← 1
OBTAIN SOURCE OPERAND AND/OR CONSTANTS
LATCHES ARE KEPT SET FOR BIS + BIC + BIT
ISR 1
K2-2 SHIFT ISR
EXTRA
K2-3 ISR ← 15
-(EXTRA)
K2-3 BSR ← 12
DATO # ENTRY
K4-2 RA/SOURCE
BINARY * -(SOURCE MODEØ)
K4-3 GATE RA ← SOURCE
BINARY * SOURCE MODE Ø
K6-3 GATE A ← R15/1
K6-3 GATE A ← R/Ø
ADD + BIC + DEC + [SBC * C(1)]
K6-3 GATE A ← -R15/1
K6-3 GATE A ← -RØ
SUB + BIT + DEC + [SBC * C(1)]
K6-4 GATE B ← R16/Ø
K6-4 GATE B ← R7/Ø
(MOV + BIS + CMP)* -(SEX)
K6-4 GATE SEX
MOV * IR15(1) * RØ7(1) * DEST MODE Ø
K6-5 CARRY ØØ ← 1
SUB + CMP + NEG + INC + [ADC * C(1)]
K6-2 CLR LATCH B
CLR
K2-3 DATO # ENTRY
ISR ← 15 * -(DEST MODE Ø + TEST + STADRS)
K13 2 DATOB ← 1
DATO# ENTRY * BYTE OP
K13 2 DATO — 1
DATO# ENTRY * -(DATOB ← 1)
SHIFT ISR
ISR3
K2-2 SHIFT ISR
K4-2 RA/TEMP
K4 3 W/ENABLE 15/Ø
EXTRA
K11-2 GATE CC ← BYTE
EXTRA * -(BIC + BIT)* (N DATA + V DATA + Z DATA + C DATA)
K6 3 GATE RIGHT 15/Ø
ROT/SHF R
K6 3 GATE LEFT 15/Ø
ROT/SHF L
K1Ø 4 CLK N,Z,V,C
EXTRA * -(BIC + BIT
K2 2 SHIFT ISR
K2 3BSR ← 12
DATO # ENTRY
K4 2 RA/TEMP
EXTRA
K6 3 GATE A ← R15/1
K6 3 GATE A ← RØ
ROT/SHF
K6 3 GATE A ← -R15/1
K6 3 GATE A ← -RØ
EXTRA * -(ROT/SHF)
K2 3 DATO # ENTRY
-(DEST MODE Ø + TEST + ST ADRS)
K6 2 CLR LATCH B
EXTRA
MANIPULATE AND STORE FINAL DATA
SET UP CONDITION CODE CHANGES
W
K1-3 BSR 12
DATO # ENTRY
ISR 15
K1-3 BSR ← 1
FETCH ← 1
K2-2ISR ← Ø,2/SERVICE
BUS OUT DONE + ST ADRS + TEST + DEST MODE Ø
K1-3 ISR ← 2
ISR ← Ø,2/SERVICE * ST ADRS * -(TEST + DEST MODE Ø)
K2-2 ISR ← Ø
ISR ← Ø,2/SERVICE
-(DATO # ENTRY)
DATO # ENTRY
DATO # P CLK RESTART
K6-3 GATE LEFT 15/Ø
ROT/SHF L * BAR ØØ(Ø)
K6-3 GATE RIGHT 15/Ø
ROT/SHF R * BAR ØØ(Ø)
K6-3 GATE BYTE 15/8
SWAB + [(U + B + R/S) * BARØØ(1) * -(DEST MODE Ø)]
K6-3 GATE BYTE 7/Ø
SWAB
K11-2 GATE CC ← BYTE
(BYTE OP + SWAB) * -(GATE ST ← D)
K11-2 GATE CC ← WORD
-(BYTE OP + SWAB) * -(GATE ST ← D)
DEST MODE Ø
TEST
ST ADRS
K4-3 GATE RA ← DEST
K4-3 W/ENABLE 15/8
K4-3 W/ENABLE 7/Ø
K2-3 CLK BR
ST PTR CLK
K10-4 GATE ST ← D
ISR ← Ø,2/SERVICE
K2-3CLK BR
ST PTR CLK
K1-4 FETCH ← 1
ISR ← Ø,2/SERVICE * -{[(WAIT + REQUEST) * -(ISRØ * SERVICE)] + TRAPS + ISR ← 2 + CBRF}
K1-4 SERVICE ← 1
ISR ← Ø,2/SERVICE * -(FETCH ← 1)
FETCH ← 1
F
SERVICE ← 1
SVC
① DEST = IRØØ * IRØ1 * IRØ2
② SOURCE = IRØ6 * IRØ7 * IRØ8
③ CLOCKING OF STATUS DATA OCCURS AT THE NEXT ✳
④ SIGNAL NAME EXISTS ON K2-3 AS PARTIAL BSR ← 1. COMPLETE SIGNAL IS COMPOSED OF VARIOUS SIGNALS, INCLUDING THE ONE INDICATED, WHICH WILL CAUSE THIS CONDITION TO OCCUR AND THEY ARE LOCATED ON K1-3.
⑤ EXTRA = BIT + BIC + (ROT/SHF * OB)
⑥ TEST = TST + BIT + CMP
QTY. DESCRIPTION PART NO. ITEM NO.
PARTS LIST
FIRST USED ON OPTION/MODEL
DO NOT SCALE DRAWING
UNLESS OTHERWISE SPECIFIED DIMENSION IN INCHES TOLERANCES DECIMALS ± .005 FRACTIONS ± 1/64 ANGLES ± 0°30' FINAL SURFACE QUALITY REMOVE BURRS AND BREAK SHARP CORNERS
MATERIAL
FINISH
DRN. DATE 12-3-70
CHK'D. DATE
ENG. DATE
PROJ. ENG. DATE
PROD DATE
NEXT HIGHER ASSY
SCALE NONE
SHEET 2 OF 6
digital EQUIPMENT CORPORATION MAYNARD, MASSACHUSETTS
TITLE
SIZE D CODE FD NUMBER KC11-Ø-KIF REV.
DIST.
REVISIONS REV. CHANGE NO. CHK
DEC FORM NO. DRD 100
8 7 6 5 4 3 2 1
D C B A

KA11 Instruction Flow



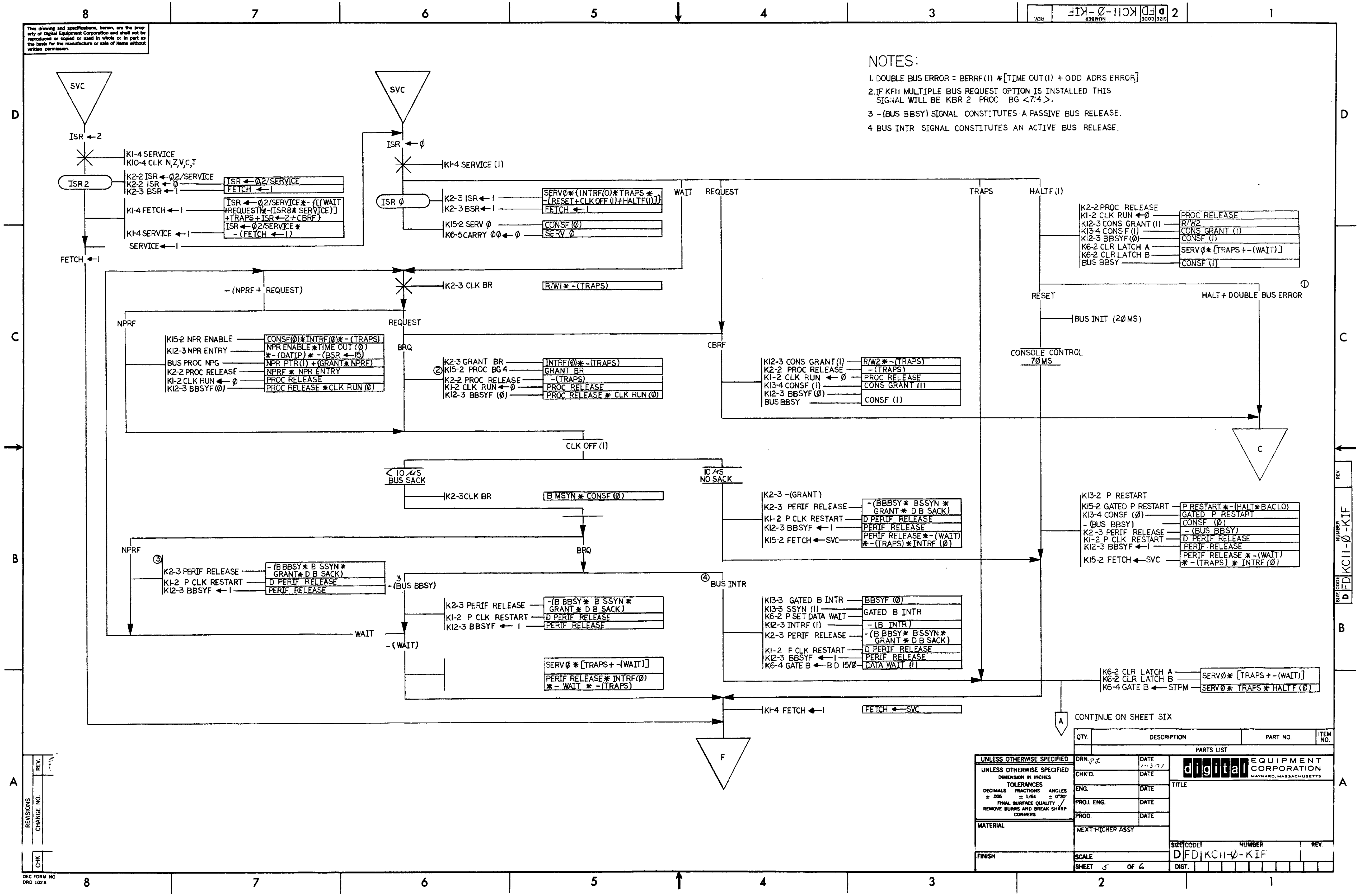
KA11 Instruction Flow



① DEST = IRØØ * IRØ1 * IRØ2

② SIGNAL NAME EXISTS ON K2-3 AS PARTIAL BSR ← 1. COMPLETE SIGNAL IS COMPOSED OF VARIOUS SIGNALS, INCLUDING THE ONE INDICATED, WHICH WILL CAUSE THIS CONDITION TO OCCUR AND THEY ARE LOCATED ON K1-3.

KA11 Instruction Flow

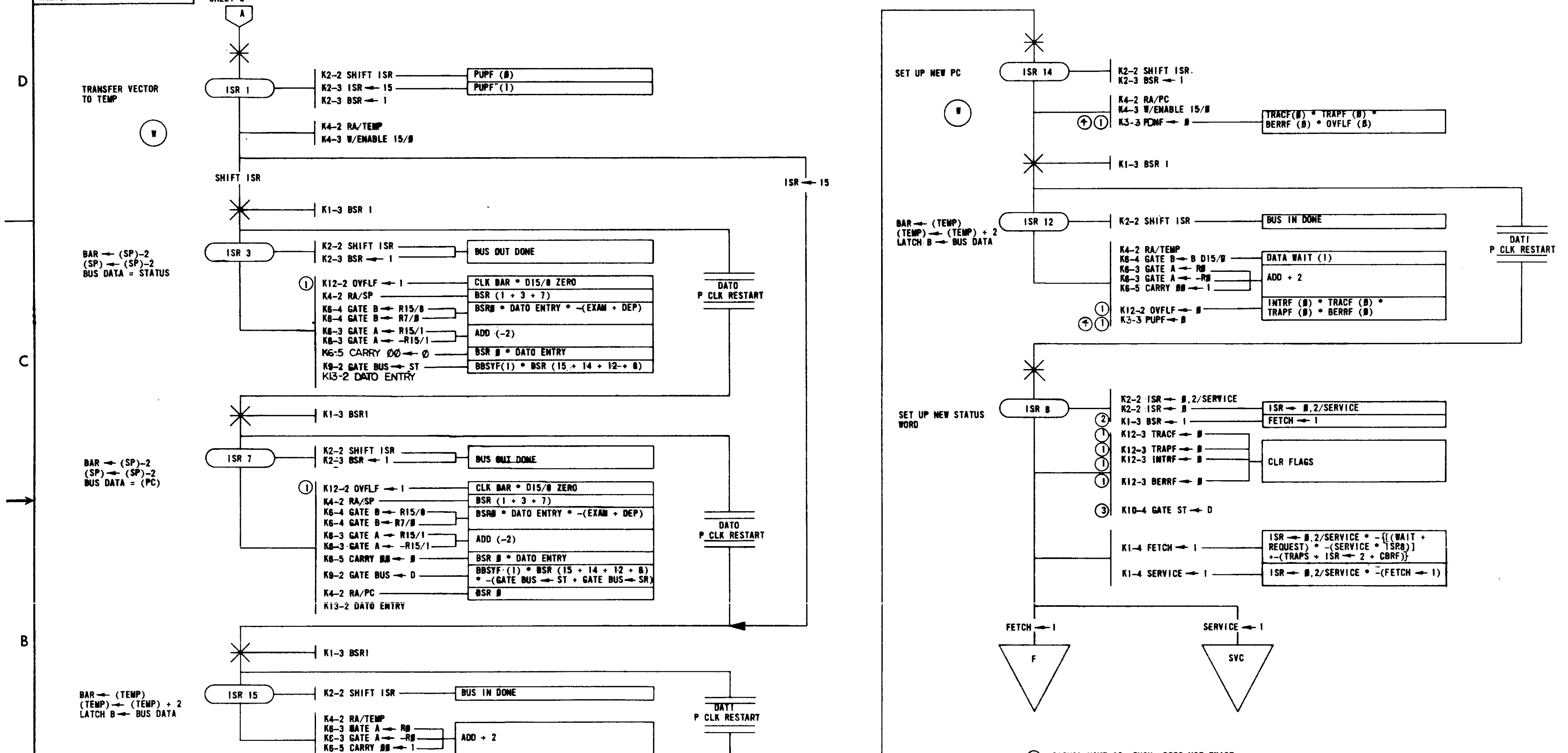


NOTES:

1. DOUBLE BUS ERROR = BERRF(1) * [TIME OUT(1) + ODD ADRS ERROR]
2. IF KF11 MULTIPLE BUS REQUEST OPTION IS INSTALLED THIS SIGNAL WILL BE KBR 2 PROC BG <7:4>.
3 - (BUS BBSY) SIGNAL CONSTITUTES A PASSIVE BUS RELEASE.
4 BUS INTR SIGNAL CONSTITUTES AN ACTIVE BUS RELEASE.

KA11 Instruction Flow

CONTINUED FROM SHEET 5



1. SIGNAL NAME AS SUCH DOES NOT EXIST. IT IS THE SUMMATION OF ALL SIGNALS ASSOCIATED WITH THE FLIP FLOP BEARING THIS NAME AND WHICH WOULD CAUSE THE SETTING TO THE CONDITION INDICATED.

2. SIGNAL NAME EXISTS ON K2-3 AS PARTIAL BSR ← 1. COMPLETE SIGNAL IS COMPOSED OF VARIOUE SIGNALS, INCLUDING THE ONE INDICATED, WHICH WILL CAUSE THIS CONDITION TO OCCUR AND THEYARE LOCATED ON K1-3.

3. CLOCKING OF STATUS DATA OCCURS AT NEXT ✳.

4. PRINT PREFIX BECOMES KP-2 IN THE KCII

QTY.	DESCRIPTION	PART NO.	ITEM NO.
	PARTS LIST		

FIRST USED ON OPTION/MODEL
—/—/—

DO NOT SCALE DRAWING
UNLESS OTHERWISE SPECIFIED
DIMENSION IN INCHES
TOLERANCES
DECIMALS ± .005 FRACTIONS ± 1/64 ANGLES ± 0°30'
FINAL SURFACE QUALITY ✓
REMOVE BURRS AND BREAK SHARP CORNERS

MATERIAL —/—/—

FINISH —/—/—

DRN Pazhenn DATE 12-3-70
CHK'D. DATE
ENG. DATE
PROJ ENG DATE
PROD DATE

NEXT HIGHER ASSY —/—/—

SCALE NONE
SHEET 6 OF 6

digital EQUIPMENT CORPORATION
MAYNARD, MASSACHUSETTS

TITLE

SIZE D CODE FD NUMBER KCII-Ø-KIF REV.

DIST.

DEC FORM NO. DRD 100

KA11 Instruction Flow

CHAPTER 4
KA11 PROCESSOR LOGIC DESCRIPTION

4.1 INTRODUCTION

Beyond the theory of machine operation is the actual implementation and associated logic description. The logic description consists of those drawings and discussions that relate directly to implementation and hardware. In the KA11, these discussions and drawings are physically and conceptually integral. The location of text adjacent to the logic drawing eases cross-reference, while the previous presentation of operational theory allows the discussion to concentrate on implementation.

A measure of information exists in the mechanics of presentation. Certain conventions in the logic drawings, logic usage, and discussions convey information; these are noted below.

4.2 PRINT ORGANIZATION

The KA11 prints and wire list correlate all signal names and allow the forward or reverse tracing of signals. The prints conform, in general, to DEC STD 056, "Distinctive Shape Logic Symbology." The following characteristics are important:

a. The *logic drawings* comprise individual print sets ordered toward the individual modules. The TIMING & STATES print set, for instance, contains four sheets that document the TIMING & STATES module, M728. A cover sheet (K1-1) provides: component reference and placement; supply voltage filter capacitors; and notes upon signal and circuit conventions. The remaining sheets (K1-2, K1-3, K1-4) provide the logic drawings of the module. Signal names within the logic relate this logic to the rest of the processor. It is this interrelationship between the several modules that allows separate print sets to adequately document the processor.

b. *Signal names* contain a print prefix (K1-2, for instance) and a polarity suffix (H or L).

The print prefix identifies the logic print from which the signal originated. In the KA11, there are fifteen such multiple-page print sets with the print prefix located in each title block. In the print prefix, the number immediately after the K identifies the print set, while the next number identifies the page within the set. The print prefixes KY and KM refer to the KY11-A Console and KM11-A Maintenance Panel, respectively. Signal names beginning with "BUS" are an exception; they represent a "wired-or" situation with multiple sources.

The polarity suffix identifies the logic level at which the named condition is true. Thus for the signal K1-2 DATA CLR H, DATA CLR is true when the signal level is high. Logic gates are enabled by the named signal condition when the input signal's polarity suffix coincides with the input state indicator. The gate is disabled by the named condition if a conflict occurs. For example, the logic gate below is enabled by the named conditions A, B and C, and disabled by the named condition D.

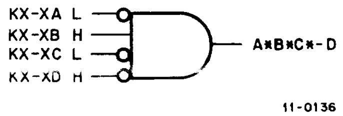


Figure 4-1 Typical Gate Showing State Indicators

c. *Signal flow*, as indicated by gate orientation, is from left to right or from bottom to top. The majority of prints flow left to right with all module output signals brought to the extreme right. This technique eases the search for a source signal referenced from another module set. For example: on the K6-2 print (DATA PATH CNTL) at drawing reference 4C, the signal K1-2 REG LATCH H is used; the source of this signal is easily found on the K1-2 print (TIMING & STATES) on the extreme right at drawing reference 1C. If the source signal is within the same print set, it is on the same module and may not have a module pin. If no module pin exists, the signal source would be within the drawing and not at the extreme right.

The DATA PATHS prints (K7 and K8) have signal flow from the bottom to the top. Module output signals end in vertical lines; input control signals have horizontal lines; input data signals begin in vertical lines.

d. The *wire list* supplements the logic drawings and discussions. It lists those module pins under common signal name that are wired together, and allows a signal to be traced from its source to all inputs. It is also possible to trace from inputs to source, but this is more easily provided for in the print prefix of the signal name.

Each signal name entry in the wire list notes: the signal name (RUN NAME and A/P); the module pin for this entry (PIN NAME); the order in which the pin is wire wrapped (BAY ORDER); the level at which the wrap is made (Z); and the drawing(s) upon which the module pin appears (DRAW). Since multiple prints exist for a given module, a single module pin might appear on several prints; such situations are noted by entries under DRAW with comma's separating the sheet numbers (K1-2,3,4, for example). The manufacture process ensures that specific module pins are interconnected; the order or level of interconnection is not tested or guaranteed.

Some differences in nomenclature exist between the prints and the wire list. Most notable are:

1. The use of leading 0s in numerical fields to order signals. The print signal K1-2 S CLK H becomes K01-2 S CLK H in the wire list.
2. The wire list substitutes the letters FM for a left arrow. The print signal K2-2 ISR ← 0 L becomes K02 ISR FM 00 L in the wire list.
3. Some signal symbols have been changed. For example, the print signal K2-3 DATO# ENTRY H becomes K2-3 DATO = ENTRY H in the wire list.

4.3 LOGIC USAGE

The logic descriptions assume knowledge of logic conventions (MIL STD 806B, for instance) and usage. The majority of logic is combinational with simple NAND and NOR gates; sequential logic, in general, utilizes either a simple D-edge flip-flop or a shift register with shift or direct load capabilities. Certain medium scale integration (MSI) circuits are used throughout the processor and are noted in the *PDP-11 Conventions Manual.*

4.4 LOGIC DISCUSSIONS

The logic discussions are directed at the module output signals that interconnect the separate modules. A flexible format is utilized to present maximum usable information directly adjacent to the prints. The discussion consists of: a general module description; specific module signal descriptions; and equations of combinational logic signals.

The general module description provides introductory information and relates the module to processor operation. The specific module signal descriptions are concerned with the effect or use of the signal; this section is expanded for a sequential logic signal to show derivation. The logic equations provide an economical presentation of signal content or what activates the output. The equations' location next to the logic prints is most necessary in maintenance situations.

In presenting the discussions on the logic prints, certain assumptions have been made:

a. A knowledge of logic circuits and operation is basic.

b. The use of machine state sequences (as noted in the Flow Diagrams) to enable data and sequence alteration is known. Specifically, within the discussions, the uses of the various machine states on the TIMING & STATES module are not detailed.

c. Certain machine procedures are known. Specifically, the clearing of machine states by the INIT signal is not noted in each discussion.

d. The theory of processor operation has been gained from previous sections; these discussions are concerned with the implementation.

NOTE

The remaining paragraphs in this section refer to specific engineering drawings which are contained in the second volume entitled, *KA11 Processor, Engineering Drawings.*

The information on each of the following pages relates to only one specific print. The print number is in the upper corner of the page. For example, if a page has K2-3 in the upper corner, then all material on that page refers to print K2-3.

4.5 TIMING & STATES

This module provides: the basic processor clock and its control logic (RW0, RW1, CLK OFF and CLK RUN flip-flops); machine states for instruction and bus cycles (Instruction Shift Register and Bus Shift Register); and major machine states (STATE SR). Many of the outputs have driving or inverting logic.

Internal processor timing is synchronous with the major processor clock being the System CLocK (K1-2 S CLK H, K1-2 S CLK l); all machine states are based upon this timing interval. This clock is derived from the Read/Write shift register (E3) and the basic oscillator (E12 and associated discrete components). Both phases of the basic oscillator (K1-2 CLK H and K1-2 CLK L) are used to clock the Read/Write flip-flops (R/W0, R/W1) through their transient-free cycle (00, 01, 11, 10) with one bit change per clock. The outputs provide the noted S CLK signals, as well as other clocking signals. A timing diagram in the logic discussion for Print K1-2 shows the relationship of these various signals.

Asynchronous processor operation is required for the transfer of data and bus control. This is provided for by the CLK RUN and CLK OFF flip-flops (E9) which control the basic oscillator and the gating of basic clocking signals, respectively. The basic oscillator and clocking is halted during the R/W2 state for asynchronous transfers. Since this state is the fourth quarter of a S CLK period, extended machine states occur upon such transfers. Overall machine operation appears as segments of synchronous operation with asynchronous pauses.

The Instruction Shift Register (ISR), Bus Shift Register (BSR), and State Shift Register (FETCH, SOURCE, DEST, EXECUTE and SERVICE) provide the time state signals used throughout the machine to direct and alter machine flow. The shift and load input terminals of these shift registers have combinations of time states, address mode, and instructions as their input signals; and are clocked at S CLK intervals. The shift register can be shifted or loaded according to the enabled input control; the clock is disabled if neither the shift or load inputs are enabled. This latter feature allows simplification of the state control logic: an enabling signal is needed only when a change in machine state is necessary. Instead of defining the input control signal for all time only the boundary (of the state change) need be defined. The ISR utilizes this and remains constant while the BSR cycles through data transfers; major states are held in the STATE SR unless a change is needed. The load signals (K1-4 FETCH ← 1 L, for example) provide the data input to the shift register as well as enable the load control input on these boundaries.

The use of shift registers and the proper selection of consecutive machine states can provide transient free transitions between machine states. Within the KA11, this technique is utilized only for bus gating signals derived from K1-3 BSR (15 + 14 + 12 + 8) H. The other machine states outputs cannot be considered transient free near the machine state boundaries effected by the clocking of the S CLK signal. (The use of decoded shift register outputs and the usual depth and complexity of the combinational logic contribute to these transients.) Various clocking signals against these machine states and signals derived from them eliminate the effects of these transients.

The basic purpose of the processor timing and machine states is to provide a series of discrete machine states that control machine operation. To this end, the registers are loaded or shifted to various machine states; no particular importance should be attached to the name assigned to an ISR or BSR state (ISR 1 or 3 or 7, for instance) or the manner of entry (load or shift). Since the machine states and the clocking signals are used throughout the machine, driver gates and inverters are provided.

K1-2 DATA CLR H direct clears the MSYN flip-flop (K13-3 MSYN (1)H) and enables the direct clear of the processor's BBSYF (K12-3 BBSYF (1)H) for transfer of bus mastership to a requesting peripheral.

K1-2 P CLK RESTART L provides a test point for the 100 nanosecond pulse that direct sets the CLK RUN flip-flop (E9) to restart the processor's basic oscillator.

K1-2 CLK RUN (0)H enables the direct clear of the processor's BBSYF flip-flop (K12-3 BBSYF (1)H) for transfer of bus mastership. Primarily the CLK RUN flip-flop controls the processor's basic oscillator (E12): enabling the oscillator when set; and disabling the oscillator when loaded to zero.

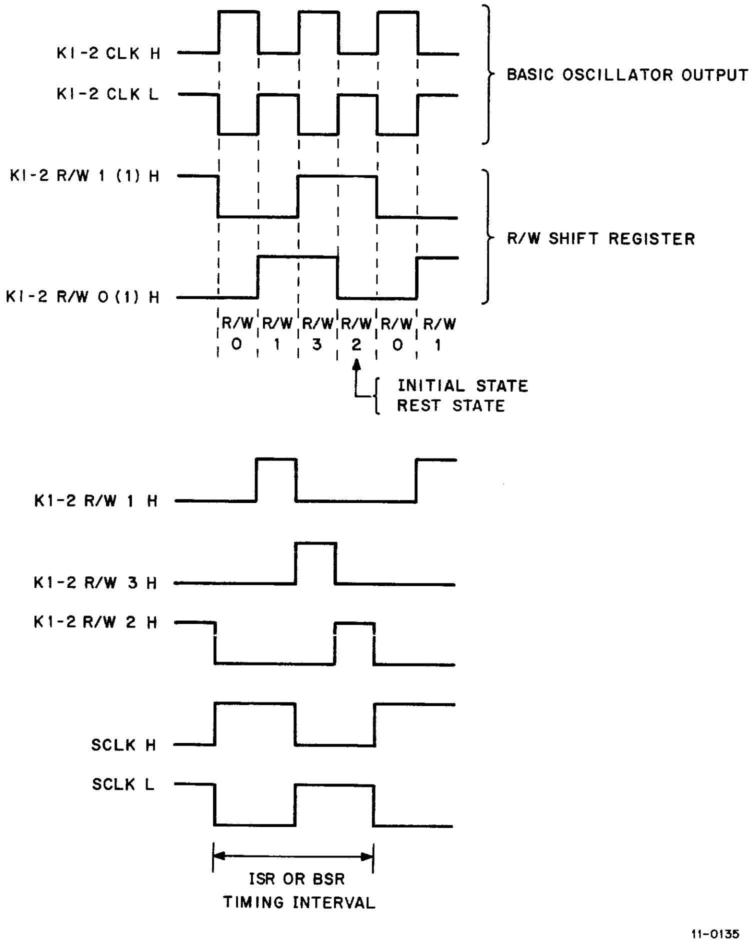


Figure 4-2 Basic Processor Clock Timing Diagram

K1-2 S CLK H provides the basic processor clock. It is derived from the R/W shift register and utilized throughout the processor. Its period of 280 nanoseconds is the period of single, synchronous, ISR or BSR machine states. See K1-2 Timing Chart.

K1-2 S CLK L provides the inverse phase of K1-2 S CLK H and is utilized throughout the processor. See K1-2 Timing Chart.

K1-2 R/W1 H
K1-2 R/W2 H
K1-2 R/W3 L
K1-2 R/W3 H
} provide specific timing signals within each S CLK period. These signals are used throughout the processor and are derived, transient free, from the R/W shift register (E3). See K1-2 Timing Chart.

K1-2 CLK OFF (1)H is used to enable machine state changes for console functions (K2-3 PARTIAL BSR ← 1 L).

The CLK OFF flip-flop (E9) is used to enable console timing when the processor's basic oscillator is disabled; it also provides a disable during the first cycle after restart. In both cases, a delay between CLK RUN alteration and CLK OFF alteration is desired. When CLK RUN is clocked to zero, CLK OFF is clocked to one only after the rest state R/W2 has existed for 50 nanoseconds – this allows completion of processor timing. When CLK RUN is direct set and processor clocking begins, the CLK OFF is not direct cleared until R/W1 time state – this allows a recycle in SERVICE * ISR0 to obtain trap vectors.

K1-2 CLK OFF (0) H provides the entry into a DATO bus cycle for the console function, DEP (K13-2 DATO ENTRY H); the disabling of processor timing inputs to K1-2 REG LATCH H and K1-2 REG GATE H for console operation; and a restart recycle in SERVICE * ISR0 to obtain trap vectors by inhibiting K2-3 ISR ← 1 L. Details of CLK OFF flip-flop (E9) operation are noted in signal discussion K1-2 CLK OFF (1)H.

K1-2 REG LATCH provides the LATCH flip-flops of DATA PATH CNTL with a direct set signal for non-Unibus data transfers. Both processor basic timing (R/W2 state) and console timing (P3 CSR state) contribute. Details of LATCH control are noted in the DATA PATH CNTL discussion.

K1-2 REG GATE provides the LATCH flip-flops of DATA PATH CNTL with a clocking signal for a conditional load to zero; the signal is also used to enable this module's control signals to the DATA PATHS inputs. Both processor basic timing (R/W1 (1) state) and console timing (P2 CSR state) contribute. Details of LATCH control are noted in the DATA PATH CNTL discussion.

K1-2 WRITE 15/8 H provides the write pulse for bits ⟨15:8⟩ of the processor's internal memory, REGISTER. Both processor basic timing (R/W3 state) and console timing (P1 CSR state) contribute.

K1-2 WRITE 7/0 H provides the write pulse for bits ⟨07:00⟩ of the processor's internal memory, REGISTER. Both processor basic timing (R/W3 state) and console timing (P1 CSR state) contribute.

K1-2 CARRY 00 L provides the timing signal to the carry flip-flop (K6-5 CARRY 00 (DL)), which is set or cleared according to enabling inputs. Both processor basic timing (R/W3 state) and console timing (P1 CSR state) contribute.

K1-2 CLK L
K1-2 CLK H
} are test points for each phase of the basic oscillator output.

K1-2 DATA CLR H = CLK OFF (1) * PROC CNTL * B SSYN * BSR8 + P DATA START

K1-2 P CLK RESTART L = -(REG ADRS) * P1 CSR0 * (EXAM + DEP) + D PERIF RELEASE + P TIME OUT + -(B SSYN) * CLK OFF (1) * BSR (3+12) + PROC RELEASE { CLK OFF (1) * PROC CNTL * B SSYN * BSR8 + P DATA START } + P CLK RESTART

K1-2 CLK RUN ← 0 L = (DATA WAIT ← 1) + PROC RELEASE + BSR8 + B SSYN (BSR3 + BSR12)

K1-2 WRITE 15/8 H = W/ENABLE 15/8 * (P1 CSR + R/W3)

K1-2 WRITE 7/0 H = W/ENABLE 7/0 * (P1 CSR + R/W3)

K1-2 CARRY 00 L = P1 CSR + R/W3

Both the Bus Shift Register (BSR) and the Instruction Shift Register (ISR) provide numerous machine states that are used throughout the processor. Both polarities are usually provided. Since these signals are used as conditions in most of the combination logic, no details of usage are presented here; this information is available in the flow diagrams.

K1-3 ST CLK PTR L provides an additional clocking signal (K2-3 CLK BR L) to the PRIORITY bus request buffer and associated PTR flag flip-flops. This clocking occurs prior to machine entry in SERVICE * ISR0 and accommodates changes in the processor's STATUS word that would affect priority determination.

K1-3 ST CLK PTR L = (GATE ST ← D) * – [SERVICE (ISR0 + ISR8)] * R/W3

K1-4

The State Shift Register provides major state information throughout the machine. Since these register states exist singularly, they are represented as flip-flop outputs at their respective buffered drive outputs. No details of usage are presented here; this information is available in the flow diagrams.

K1-4 EXEC ← 1 H provides the entry signal (load control and data to the State Shift Register) to EXECUTE major state. It is also utilized as an enabling condition for entry into BSR 1 state (K2-3 PARTIAL BSR ← 1 L).

K1-4 EXEC ← 1 H = B FETCH (1) * ISR1 * {(ISR ← 15) + BRANCH + INTERNAL ADRS} + DEST (1) {(ISR ← 1) + (JSR * ISR0)} + {DEST MODE 0 * SOURCE (1) * ISR0}

K1-4 SOURCE ← 1 L = B FETCH (1) * ISR1 * ~SOURCE MODE 0 * BINARY

K1-4 PART 2 DEST ← 1 L = B SOURCE (1) * ~(DEST MODE 0) * ISR0

K1-4 PART 1 DEST ← 1 L = B FETCH * ~(DEST MODE 0) * ISR1 * {(U + R/S) + JSR + JMP + (SOURCE MODE 0 * BINARY)}

K1-4 FETCH ← 1 H = (FETCH ← SVC) * SERVICE + [(ISR ← 0,2/SERVICE) * { ~(ISR ← 2) * SERVICE * ISR8 * (WAIT + REQUEST) * -TRAPS * -CBRF }]

K1-4 DEST ← 1 H = (PART 1 DEST ← 1) + (PART 2 DEST ← 1)

K1-4 LOAD STATE H = (SOURCE ← 1) + (PART 2 DEST ← 1) + (PART 1 DEST ← 1) + (EXEC ← 1) + (SERVICE ← 1) + (FETCH ← 1)

4.6 STATE CNTL

The module provides signals to alter ISR and BSR time states during instruction and priority transfer operations. Depending on machine condition, time state changes occur by loading or shifting to the appropriate ISR and BSR time state.

Combinational circuitry detects completion of instructions and bus data transfers to enable non-processor grants (K2-3 BUS PROC NPGH) and to gate processor bus grants (K2-2 GRANT BRH). Clocking for bus request and non-processor request flip-flops (K2-3 CLK BRL) is provided to gate new requests. The clocking is derived from the processor and bus data transfers. The clocking is inhibited if bus grants are in process or if BUS SACK is asserted. Priority transfer flip-flops (BR PTR, NPR PTR) are clocked 220 ns after K2-3 CLK BRL if no bus interrupt is present, to gate grants to the Unibus. Direct clearing of BR PTR and NPR TRP occurs 100 ns after BUS SACK is asserted and drops the enabled bus grants. Inhibit (K2-2 PROC RELEASE L) and restart (K2-3 D PERIF RELEASE L) signals are provided for control of the processor's clock during priority transfers. Further discussion on priority transfers can be found in the *PDP-11 Unibus Interface Manual.*

K2-2 SHIFT ISR H enables the shift input of the Instruction Shift Register (ISR).

K2-2 BUS IN DONE H detects the completion of a bus data in transfer and is used to enable K2-2 SHIFT ISR H in ISR0 of FETCH or, if additional bus transfers are needed, to complete the address calculation in SOURCE or DEST; and to enable K2-2 ISR ← 0 L during the last bus data transfer in SOURCE address calculation.

K2-2 BUS OUT DONE H detects completion of a bus data transfer out and is used: to enable K2-2 SHIFT ISR H in ISR0 of EXECUTE; to enable K2-2 ISR ← 0,2/SERVICE L in ISR15 of executing a Unary, Binary, or Rotate/Shift instructions.

K2-2 PROC RELEASE L detects a NPR request to be honored at completion of a bus cycle, or a HALTF or bus request (BR ⟨7:4⟩) to be honored in ISR0 of SERVICE. The signal is used: to direct clear the BBSYF and to enable K1-2 CLK RUN ← 0 L to stop the processor's clock.

K2-2 ISR ← 0,2/SERVICE H provides for the entry into ISR0 or ISR2 and SERVICE which occurs at the end of EXECUTE, K13-2 TIME OUT (1) or K10-3 ODD ADRS ERR L during a processor bus data transfer. It is used: to enable K1-4 SERVICE ← 1L if traps or bus requests are to be serviced; and to enable K1-4 FETCH ← 1L allowing the processor to skip the SERVICE major state if no traps or bus requests exist.

K2-2 ISR ← 0L enables the ISR load input to ISR0 for a machine state alteration.

K2-2 INTERNAL ADRS enables K1-4 EXEC ← 1H during ISR1 of FETCH if a double operand (Binary) instruction has SOURCE MODE 0 and DEST MODE 0 or a single operand (Unary) or Rotate/Shift instruction has DEST MODE 0.

K2-2 WAITING L = (SERVICE * ISR0) * WAIT * - CONSF (1)

K2-2 PROC RELEASE L = SERVICE * ISR0 * (HALTF (1) + (REQUEST * - TRAPS) + NPR ENTRY * (CNPRF (1) + NPRF)

K2-2 BUS IN DONE L =PERIF RELEASE+ - PROC RELEASE * { BSR7 * - BAR00 * - TIME OUT (1) + BSR14 }

K2-2 BUS OUT DONE H = BSR8 * - TIME OUT (1) * - PROC RELEASE

K2-2 SHIFT ISR H = EXEC[ISR1 * EXTRA * - (U + B + R/S) + ISR0 {BUS OUT DONE + (U + B + R/S)}] + B EXEC (1) { ISR (3+7) + RTS + ISR15 } + BUS IN DONE { B FETCH (1) * ISR0 + - B EXEC (1) * ISR (12+15) * (U + B + R/S) } + ISR14 + SERVICE {ISR1 * - PUPF (1) + ISR (3+7) * BUS OUT DONE } + BUS IN DONE * - ISR0 * (SO + DE) * - (ADRS DONE) + - DEST MODE 0 * ISR0 * SOURCE (1)

K2-2 ISR ← 0,2/SERVICE = TIME OUT (1) * BSR (7+8) + BSR3 * ODD ADRS ERR + ISR8 + ISR2 + B DEST (1) * JMP * ISR0 + B FETCH (1) * F INSTR * ISR1 + B EXEC (1) * ISR15 * { (U + R + R/S) * (BUS OUT DONE + ST ADRS + TEST + DEST MODE 0) + BRANCH + JSR }

K2-2 ISR ← 0 L = (ISR ← 0,2/SERVICE) + ADRS DONE { B SOURCE (1) * BUS IN DONE + B DEST (1) * BSR7 (JMP + JSR) } + INTERNAL ADRS

K2-2 INTERNAL ADRS L = + B FETCH (1) * ISR1 * DEST MODE 0 {BINARY * SOURCE MODE 0 + (U + R/S) }

K2-3 BSR ← 12 L provides for the entry into BSR12 machine state during bus data out transfers.

K2-3 PARTIAL BSR ← 1 L provides some of the entry signals into BSR1 machine state for the beginning of a bus data transfer.

K2-3 DATO # ENTRY H provides for the data out transfer to return an operand that the instruction has modified. This DATO# transfer utilizes only the latter portion of the processor data out cycle; the address of the data is already determined and in the Bus Address Register (BAR).

K2-3 ISR ← 15 L enables the ISR Load input to ISR15 and is used to enable K1-4 EXEC ← 1 H in ISR1 of FETCH for a ReTurn from Interrupt or ReTurn from Subroutine instruction.

K2-3 ISR ← 1 L enables the ISR load input to ISR1 when destination address calculation is complete for a Unary, Binary, or Rotate/Shift instruction or when servicing an INTR F or a processor trap.

K2-3 PERIF RELEASE L detects that the Unibus is clear of BBSY, SSYN, GRANT, and SACK signals. The signal is used to provide data to set the processor's BBSYF when the processor clock restarts; and to enable K2-3 D PERIF RELEASE L to restart the clock. This signal is the means by which bus control is transferred back to the processor.

K2-3 D PERIF RELEASE L is enabled 600 ns after K2-3 PERIF RELEASE is asserted. The signal direct sets the CLK RUN flip-flop to restart the processor's clock.

K~~2-3~~REQUEST H detects a console bus request (K3-2 CBRF(1)L) or a qualified bus request (K3-2 BRQ L) and enables K2-2 PROC RELEASE L in ISR0 of SERVICE if no traps are present.

K2-3 BSR ← 15 enables the BSR load input to BSR15 in BSR7 if the BAR contains an odd address. The processor bus cycle is extended to right justify odd byte bus data.

K2-3 CLK BR L provides the clock signal to the Bus Request flip-flops BR ⟨7:4⟩ NPRF, CBRF, and CNPRF. If no peripheral is waiting for bus mastership and no grant is being issued, the signal is enabled by: K1-2 R/W 1 H during a WAIT instruction; assertion of K13-3 BMSYN H without console control; or K1-3 ST PTR CLK L. The clock also activates the 220 nanosecond one-shot (E43), which clocks the PTR flip-flops after time out. This delayed clock is inhibited during the INTR cycle.

The delay in clocking the PTR flip-flops is necessary to allow the settling of the PRIORITY comparison logic. If an NPR or BR is to be serviced, the respective PTR flip-flop is activated; these flip-flops (NPR PTR or BR PTR) provide the flag signals for altering the processor operation.

K2-3 GRANT BR H enables the processor bus grants (K3-2 PROC BG ⟨7:4⟩) to the Unibus in ISR0 of SERVICE if no traps, interrupts or NPR's exist.

K2-3 GRANT L inhibits K2-3 CLK BR L when bus grants are gated to the Unibus to prevent a grant priority level change during grant.

K2-3 GRANT H clocks the TIME SACK flip-flop to a (1) state to begin a 10 microsecond time out for peripheral Selection ACKnowledgement (SACK). If no SACK occurs, the signal K13-2 NO SACK (1) L activates the 100 nanosecond pulser, (E44, pin 11) to clear the PTR flags; processor operation continues.

K2-3 BSR ← 7 L if no ODD ADRS ERR is detected during BSR3; the signal enables the BSR Load input and provides input data to change the BSR time state to BSR7.

K2-3 BUS PROC NPG H is the non-processor grant issued to the Unibus.

K2-3 DATO# ENTRY H = B EXEC (1) { - DEST MODE 0 * - TEST * - ST ADRS } * (U + B + R/S) * { ISR7 + ISR ← 15 }

K2-3 BSR ← 12 L = BSR0 { DATO ENTRY + DATO# ENTRY }

K2-3 ISR ← 15 L = ISR1 [B EXEC (1) * - EXTRA * (U + B + R/S) + B FETCH (1) * { RTI + RTS } + PUPF]

K2-3 PARTIAL BSR ← 1 L = SERVICE { ISR1 + BUS OUT DONE * ISR (3+7) } + (EXEC ← 1) * (JSR + RTI) + CLK OFF (1) * - (REG ADRS) * (EXAM + DEP) + BUS IN DONE * - (ADRS DONE) * - ISR0 * (SO + DE)

K2-3 ISR ← 1 L = B DEST (1) * BUS IN DONE * ADRS DONE * (U + B + R/S) + SERV0 [INTR F (1) + TRAPS * - { RESET + CLK OFF (1) + HALT F (1) }]

K2-3 PERIF RELEASE H = - BUS BUSY * - B SSYN * - GRANT * - BUS SACK

K2-3 CLK BR L = { ST CLK + CONSF (0) * B MSYN + WAITING * - TRAPS * R/W1 } * - BUS SACK * - GRANT

4.7 **PRIORITY,** M824

The PRIORITY module provides: priority buffering, comparison and selection; trap flags for power fail operation; and the bus receivers and drivers for upper byte data. (Refer also to KY11-A Console flow diagrams in the *KY11-A Programmer's Console Manual, DEC-11-HR6A-D.)*

4.7.1 **Priority**

The priority portion of the module consists of a register buffer, comparison logic, and bus grant gating. The incoming requests for bus control are strobed (K2-3 CLK BR L) into the buffer and then compared to the processor status. If the comparison indicates that a requesting device has higher priority, appropriate signals alter machine flow and grant to the specific request. The manner of this response is dependent upon the request type.

Console requests occur in the HALT mode and result in either a Console Non-Processor Request Flag (CNPRF flip-flop) for a S-CYCLE mode or a Console Bus Request Flag for a S-INST mode. Since console operations have highest priority comparison is inhibited and bus control is transferred directly from the processor to the console by setting the CONS GRANT flip-flop (K1-3 CONS GRANT (1) H) and CONSF flip-flop (K13-4 CONSF (1) H). After the transfer of control, the K13-4 CONSF (1) H clears all the request buffers except CNPRF, which it sets.

The release of bus control by the console results in K13-4 CONSF (0) L clearing even CNPRF.

Requests from the bus result in a Non-Processor Request Flag (NPRF flip-flop) and Bus Requests (BR ⟨7:4⟩ flip-flops). The comparison and subsequent gating of these requests are made according to the following:

a. console requests have priority over processor status and requests from the bus.

b. Non-Processor Requests (NPRs) have priority over Bus Requests (BRs).

c. With no NPRs, the highest existing BR is granted if processor status (K9-4 ST ⟨07:05⟩ (1) H) is lower.

If the requesting bus device has higher priority, appropriate signals (K3-2 NPRF H or K3-2 BRQ L) load the PTR flags on STATE CNTL and machine flow is altered (K2-2 PROC RELEASE H). A specific grant is enabled according to the request level (BUS PROC NPG H or K3-2 PROC BG ⟨7:4⟩ H) and gated by K2-3 GRANT BR H. The term PROC appears in the above grant signals because the signals pass through the two general peripheral slots (CDEF 13 and 14) before physically becoming a Unibus signal.

Note that the control logic for the Priority register and combinational logic is located on STATE CNTL.

4.7.2 Power Fail

The power fail portion of PRIORITY consists of a power down flag (PDNF flip-flop) and a power up flag (PUPF flip-flop), each flagging a trap routine to bus address 24.

On power down, the AC LO asynchronous transition is synchronously clocked to PNDF by K15-2 CLK PDNF H. A load to 1 is effected unless KY-3 HALT L is enabled; if HALT is enabled the power down trap does not occur and bus control is transferred to the console. If PDNF is set both K3-3 PWRF H and K3-3 PWRF L are active and the power fail trap will occur in sequence after all internal processor traps and before any requests from the bus. This sequence is apparent in the generation of the Special Trap Markers (STPM) on FLAG CNTL and in the clearing of the flags. The PNDF flag is cleared only after other internal flags are cleared (K12-2 SVC CLR OVFLF and K12-2 OVFLF (0) L).

On power up, the initializing signal (K13-2 PWR UP H) sets the PUPF flip-flop unless KY-3 HALT L is active. If HALT is enabled then K13-2 INIT H clears the PUPF as it does most other flags.

In this mode CONSF (K13-4 CONSF (1) H which is set by K13-2 INIT H) is not cleared by K15-2 GATED P RESTART; the machine does not restart and is halted under console control. No power up trap is sequenced on START or CONT as the PUPF is cleared.

Without KY-3 HALT L, power up results in the PUPF flag set. Since other flags are cleared when K15-2 GATED P RESTART restarts the machine, a power up trap to bus location 24 is immediately sequenced. (Since the location and sequence are similar to power down, a combination flag K3-3 PWRF H or L is used for PUPF and PDNF). The K3-3 PUPF (0) H does, however, alter the usual trap sequence to avoid the DATO operations; the address base for these is a solid state register and unknown on power up. The PUPF is cleared after the trap sequence.

4.7.3 Bus Receivers and Drivers

High impedance bus receivers effect the Unibus input interface for upper byte data from BUS D ⟨15:08⟩ L to provide buffered inputs, K3-3 B D ⟨15:08⟩ H. The Unibus driver gates enable upper byte data from the DATA PATHS outputs K8-2 D ⟨15:08⟩ H. Note that BUS D ⟨15:08⟩ L signals are "wire or'ed" and receive outputs from the other sources both within the processor (console SR) and without (general peripheral slots and the bus).

```
K3-2 CBRF H = CBRF (1) * - CNPRF (1)

K3-2 NPRF H = NPRF (1) * - [CNPRF (1) + CBRF (1)]

K3-2 PROC BG7 H = (GRANT BR) * BR7 (1) * - [CNPRF (1) + CBRF (1) + NPRF (1) + ST07 (1) * ST06 (1) * ST05 (1)]
                  PROC STATUS 7

K3-2 PROC BG6 H = (GRANT BR) * BR6 (1) * - [CNPRF (1) + CBRF (1) + NPRF (1) + BR7 (1) + ST07 (1) * ST06 (1)]
                  PROC STATUS (7+6)

K3-2 PROC BG5 H = (GRANT BR) * BR5 (1) * - { CNPRF (1) + CBRF (1) + NPRF (1) + BR7 (1) + BR6 (1) + ST07 (1) * [ST06 (1) + ST05 (1)]}
                  PROC STATUS (7+6+5)

K3-2 PROC BG4 H = (GRANT BR) * BR4 (1) * -{CNPRF (1) + CBRF (1) + NPRF (1) + BR7 (1) + BR6 (1) + BR5 (1) + ST07 (1)}
                  PROC STATUS (7+6+5+4)

K3-2 BRQ L = PROC BG7 + PROC BG6 + PROC BG5 + PROC BG4
```

4.8 REGISTER CNTL, M821

The REGISTER CNTL module consists of the combination logic necessary to the control of REGISTER. Machine state, flag and instruction information is combined to select the address source for the REGISTER. Selection of a specific source, and providing the source for Specific ADdress (SAD), is a "read"; the "write" also requires write signals from TIMING & STATES. Some other logic signals, unrelated to memory, are formed on REGISTER CNTL.

Signal discussions and equations of the output signals follow.

K4-2 JMP H provides the logic inversion of signal K10-3 JMP L.

K4-2 GATE RA ← SAD H enables the Specific ADdress bits to the Register Address. It is active when specific processor registers are called in the machine flow. This address selection differs from the gating of a register's unknown bit combination to obtain one of a group of registers. A specific register is selected: Program Counter (PC); Stack Pointer (SP); TEMP register; and SOURCE register. Which of the registers is selected depends upon the respective internal signals: K4-2 RA/PC L; K4-2 RA/SP L; K4-2 RA/TEMP L; and K4-2 RA/SOURCE.

K4-2 SAD ⟨03:00⟩ H provides the individual bit inputs to the REGISTER for selection of a Specific ADdress. Certain bit patterns are enabled by the internal signals: K4-2 RA/PC L; K4-2 RA/SP L; K4-2 RA/TEMP L, and K4-2 RA/SOURCE L. The SAD bit patterns and selected registers are, respectively: $(0111)_2$ or R7 for PC; $(0110)_2$ or R6 for SP; $(1000)_2$ or R8 for TEMP; and $(1001)_2$ or R9 for SOURCE.

K4-2 RA/PC L = B FETCH (1) + B EXEC (1) [BRANCH {ISR7 + ISR15} + JSR {ISR15 + ISR1}] + ISR14 + (SO + DE) {JMP * ISR0 + ADRS MODE (6+7) * ISR1} + START * CSR3 + BSR0 * ISR7 * SERVICE

K4-2 RA/TEMP L = B EXEC (1) { BRANCH * ISR (1+3) + EXTRA * ISR (3+7) + JSR * ISR7} + SERVICE { ISR1 + ISR (12+15) } + BSR14 + BSR15 + CSR3 { LOAD ADRS + (EXAM + DEP) } + CSR1 (ST + EX + DEP) + B DEST (1) * ISR0 * JSR

K4-2 RA/SOURCE L = B SOURCE (1) * ISR0 + B EXEC (1) * ISR1 * BINARY * ~SOURCE MODE 0

K4-2 RA/SP L = B EXEC (1) [ISR12 + ISR15 * RTI + ISR0 * JSR * BSR (1+3+7)] + SERVICE * BSR (1+3+7) * ISR (3+7)

K4-2 GATE RA ← SAD H = RA/PC + RA/TEMP + RA/SOURCE + RA/SP

K4-2 SAD03 H = RA/TEMP + RA/SOURCE

K4-2 SAD00 H = RA/SOURCE + RA/PC

K4-2 SAD02 H = RA/PC + RA/SP

K4-2 SAD01 H = RA/PC + RA/SP

K4-3 (EXEC * JSR) H provides a logical signal of major state and instruction from a convenient internal point.

K4-3 GATE RA ← SOURCE H enables the bits of the Instruction Register which specify a SOURCE operand. The selected register is unknown and a function of the specific bits, but gating occurs when SOURCE operands are necessary in machine flow.

K4-3 ADRS MODE (6+7) L provides a logical signal for indexed or indexed deferred SOURCE or DEST address calculations.

K4-3 GATE RA ← DEST H enables the bits of the Instruction Register which specify a DEST operand. The selected register is unknown and a function of the specific bits, but gating occurs when DEST operands are necessary in machine flow.

K4-3 W/ENABLE 7/0 L enables the write signal to REGISTER for lower byte. The enable is not direct but is coupled with console and processor timing signals on TIMING & STATES.

K4-3 W/ENABLE 15/8 L enables the write signal to REGISTER for upper byte. The enable is not direct, but is coupled with console and processor timing signals on TIMING & STATES.

K4-3 TEST L provides the logical inversion of signal K10-4 TEST H.

K4-3 GATE RA ← BAR H enables the bits of the Bus Address Register to the Register Address during console operation. All sixteen words of the REGISTER may be addressed as a function of the appropriate BAR bits.

K4-3 REG ADRS L provides the logical inversion of signal K9-2 REG ADRS H.

K4-3 GATE RA ← SOURCE H = B EXEC (1) [JSR { BSR0 * ISR0 + ISR3} + SOURCE MODE 0 * BINARY * ISR1] + B SOURCE (1) * BSR (1+3+7)
[ISR3 * ADRS MODE (6+7) + ISR1 * - ADRS MODE (6+7)]

K4-3 GATE RA ← DEST H = B EXEC (1) [DEST MODE 0 (U + B + R/S) { ISR0 + ISR15 } + RTS { ISR8 + ISR15 } + B DEST (1) * BSR (1+3+7) *
[ISR3 * ADRS MODE (6+7) + ISR1 * - ADRS MODE (6+7)]

K4-3 W/ENABLE 7/0 L = B EXEC (1) [ISR3 (EXTRA + JSR) + ISR15 (JSR + BRANCH) + ISR1 * BRANCH + RTS * ISR8 + ISR15 (U + B + R/S) *
DEST MODE 0 (- ST ADRS + - TEST) + BSR15 + ISR14 + CSR3 + ISR1 * SERVICE + ISR0 * (SO + DE) + BSR7 { ISR (3+7) * -
(SO + DE) + - CONSF } + CSR0 * REG ADRS * DEP

K4-3 W/ENABLE 15/8 L = - W ENABLE 7/0 + [B EXEC (1) * (U + B + R/S) * DEST MODE 0 * ISR15 * (- ST ADRS + - TEST) * (WORD + MOV)]

K4-3 GATE RA ← BAR H = CSR0 * REG ADRS (EXAM + DEP)

4.9 REGISTER, M225

The REGISTER module provides data storage essential to the instruction set and processor data flow. Sixteen words, each of 16 bits, are available; the KA11 utilizes eight registers for data, address, program counter and stack pointer; and two registers for temporary storage. The REGISTER module consists of storage elements, address selection drive, and selection of address inputs.

Register storage is effected by sixteen Medium Scale Integration circuits, each 1 bit x 16 words. The parallel output of the sixteen circuits provides a memory of 16 bits x 16 words. Selection of a specific word requires the activation of one of four X address lines and one of four Y address lines in each circuit (see Figure 4-3). A "Read" is accomplished by this address selection; a "Write" is accomplished by address selection and the enabling of set and reset inputs by a write signal and appropriate data.

The address selection consists of a 4 x 4 matrix of address lines. The address selected is at the intersection of the specific X line activated and the specific Y line activated.

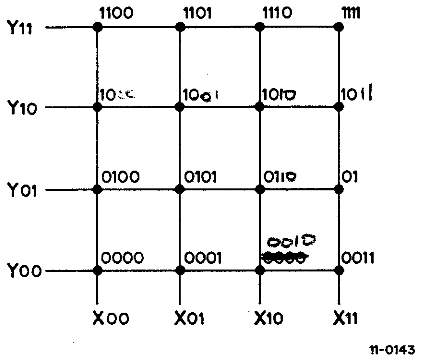


Figure 4-3 4-by-4 Address Matrix

Prior to these drive circuits, decoding and selection of address sources is made. Decoding relates the X and Y address lines to the bits of the selected address sources. Four possible sources exist: three of these sources are direct gating of portions of registers (IR and BAR) and represent instruction or console selection; one source represents specific address (PC, SP, TEMP) necessary for internal processor operation.

Since the REGISTER is important to the processor's data flow and the storage of operand data and address information, both the inputs and outputs (K5-2 prefixes) are discussed.

K5-2R ⟨15:00⟩ (1) H provides the data output of the selected word. This output can be either data or address information and is used within or passed through the DATA PATHS modules. The circuit output is open collector with resistor pull-up and clamp provided to limit signal amplitude.

K5-2 Y 11, 10, 01, 00 H provides test points for the Y drive lines of the address selection matrix. Each of these drive lines goes to the sixteen memory circuits. A discrete transistor switch provides high current drive from an open collector circuit with resistor pull-up and clamp. No requirement is made on the drive lines that the same absolute address be selected in each memory circuit; it is necessary that a unique bit is selected for each address.

The activation of a specific Y drive line (Y11, Y10, Y01, Y00) results from the decoding of the two low order bits of the selected address source. The decoding network selects Y00 when it is inactive.

K5-2 X 11, 10, 01, 00 H provides test points for the X drive lines of the address selection matrix. These lines are similar to the Y lines noted above except that the upper two bits of the selected address source are decoded to activate the specific X drive line (X11, X10, X01, X00).

K8-2 D ⟨15:07⟩ H provides the upper byte data input to the REGISTER from the output of DATA PATHS 2.

K7-2 D ⟨08:00⟩ H provides the lower byte data input to the REGISTER from the output of DATA PATH 1.

K1-2 WRITE 15/8 H enables a write into the upper byte of the address selected word. The actual write is accomplished by address selection and the combination of the write signal with the data input to set or reset the appropriate memory bits. For example: if K8-2 D 15 H was true, the application of K1-2 WRITE 15/8 H would activate the set (S) input (pin 13) of R15 (E31); if the data input had been untrue, the reset (R) input (pin 9) of R15 (E31) would have been activated.

K1-2 WRITE 7/0 H enables a write into the lower byte of the address selected word. The write operation is similar to that noted for the upper byte.

K1-2 GATE RA ← SOURCE H enables specific bits of the Instruction Register (SOURCE) for use in the Register Address (RA) selection. This input would be activated during the major states SOURCE or EXECUTE.

K9-4 IR ⟨08:06⟩ H provides the binary address bits for the selection of one of eight possible SOURCE registers. These Instruction Register (IR) bits are noted in instruction discussion with the selected addresses being the registers R0 through R8.

GND12 provides a low signal input for the higher order bit on SOURCE and DEST selection. This disabling input limits the register address to the lower eight registers, R0 through R8.

K4-3 GATE RA ← DEST H enables specific bits of the Instruction Register (DEST) for use in the Register Address (RA) selection. This input would be activated during the major states DEST or EXECUTE.

K9-5 IR ⟨02:00⟩ H provide the binary address bits for the selection of one of eight possible DEST registers. These Instruction Register (IR) bits are noted in instruction discussion with the selected addresses being the registers R0 through R8. See the signal GND12 for additional information.

K4-3 GATE RA ← BAR H enables specific bits of the Bus Address Register (BAR) for use in the Register Address (RA) selection. This input would be activated during Console operations; the processor registers respond to explicit bus addresses only during these operations.

K9-5 BAR ⟨03:00⟩ (1) H provides the binary address bits for selection of one of 16 possible registers. Four address bits are provided and the full sixteen words of the REGISTER may be addressed.

K4-2 GATE RA ← SAD H enables bits of a Specific ADdress code for use in the Register Address (RA) selection. Input activation occurs throughout machine operation whenever certain registers (PC, SP, TEMP, SOURCE) are needed for specific operation. The registers, PC and SP, may also be called as general registers (R7 and R6) in instructions or by the Console. Register address selection, in this case, occurs by selecting DEST, SOURCE or BAR inputs.

K4-2 SAD ⟨03:00⟩ H provides the binary address bits for selection of one of four specific addresses: PC, SP, SOURCE and TEMP. These registers are selected under SAD in the following instances: PC is specifically selected during Fetch, in jump or branch instructions, and as a Stack operand; SP provides the address of the Stack; SOURCE register temporarily stores the SOURCE operand in two operand instructions; and the TEMP has general use as temporary storage, especially in the movements of operands from the output to the input of the data paths.

4.10 DATA PATH CNTL

This module provides: all gating signals (control) for both DATA PATHS modules; clocking signals related to bus data transfers; and carry input data.

The logic requirements upon the signals for the DATA PATHS are related to its three segments: the rotate/shift gating; the adder; and the A and B input gating. The control to the rotate/shift gating requires only combinational logic. The carry input data to the adder must exist across machine state boundaries and requires latch storage. The control for the A and B input gating requires both storage and transient-free gating. Storage is necessary on latch signals to hold data across state boundaries; transient-free gating is needed on the other inputs to protect the data.

The DATA PATH CNTL module must effect processor data transfers to the DATA PATHS from both the Unibus and the REGISTER. The REGISTER transfers occur with normal processor timing and require no special logic. Bus transfers, however, are asynchronous and bus signals control. Logic to translate the bus signals to processor signals is needed, with these signals being used for data control, bus response and basic timing.

K6-2 P DATA START H provides a pulse signal for the restarting of processor operation after a data in or data out transfer. Bus control signals for data transfers (K13-3 B SSYN H) and for interrupt vector transfers (K13-3 B INTR H) are used directly after deskewing (δ = 100NS) to produce the signal. (The pulse nature of the output results from bus constraints on the output signals.) If no response is made by the addressed peripheral after a suitable time, a restart signal occurs (K13-2 P TIME OUT L). The K6-2 P DATA RESTART H signal is used: to restart the processor clock if no NPR is to be serviced (K1-2 P CLK RESTART L); and to generate an intermediate clearing signal (K1-2 DATA CLR H) for MSYN (K13-3 MSYN (1) H) and for BBSYF (K12-3 BBSY F (1) H) if an NPR is to be serviced.

K6-2 CLK IR H provides the clock signal to the Instruction Regsiter (IR) during FETCH major state. The signal is derived from the usual latching signal to DATA PATHS. Bus data is loaded to the IR and the DATA PATHS; in the latter it is utilized in offset instructions.

K6-2 LATCH A15/0 H provides the storage signal for all A inputs of the DATA PATHS; this signal holds the data in the A inputs. The signal is usually active, except for a clear before load. A flip-flop (LATCH A) is used to insure transient-free storage across machine state boundaries.

Latch operation is required in two types of data transfers to the DATA PATHS: one transfer is from REGISTER and the other is from the Unibus. Different timing signals are involved and are noted below with appropriate timing diagrams.

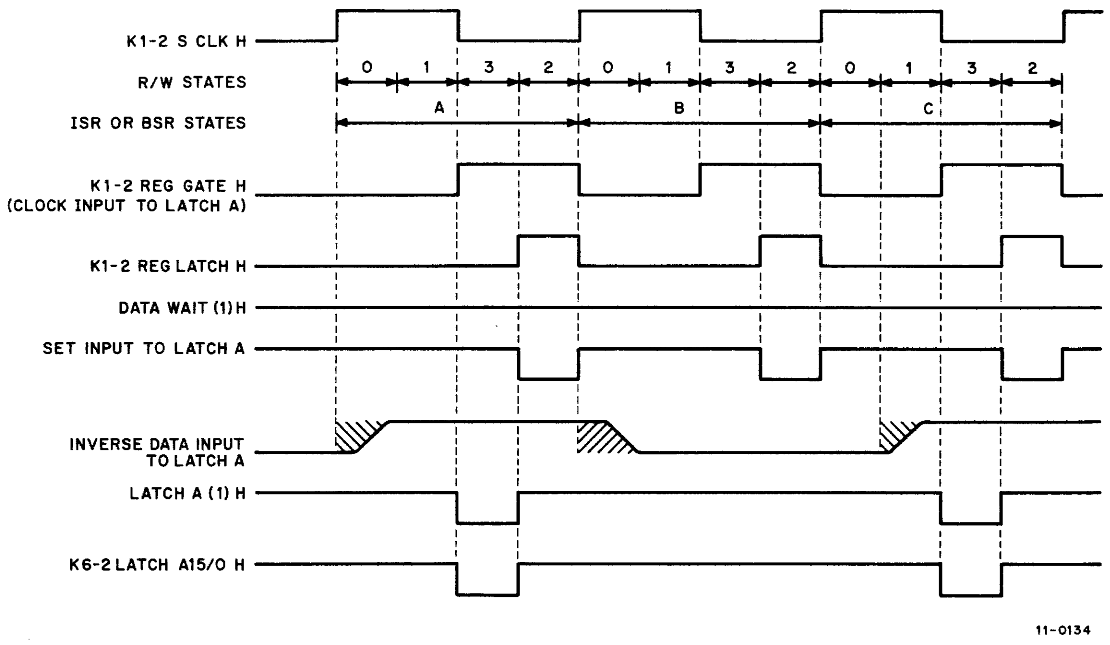


Figure 4-4 Register Data Transfer Timing Diagram

K6-2 P DATA START H = DATA WAIT (1) { B SSYN * PROC CNTL * B INTR } + P TIME OUT

K6-2 CLK IR H = B FETCH (1) [{ DEP * CSR2 + LOAD ADRS * CSR1 } * P2 CSR + P TIME OUT + DATA WAIT { B SSYN * PROC CNTL + B INTR }]

K6-2 CLR LATCH A L = ISR0 { SERV 0 * - (WAIT * - TRAPS) + JSR * B EXEC (1) + CSR2} + CLR ON BSR

K6-2 CLR LATCH B L = ISR0 {SERV0 * - (WAIT * - TRAPS) + JSR * B EXEC (1) + CSR2 } + B EXEC (1) { ISR7 * EXTRA + CLR * ISR1}

K6-2 DATA WAIT ← 1 L = BSR7 * - DATO ENTRY * - (ADRS DONE) * - (JMP + JSR)

4.10.1 Register Data Transfer

The REGISTER data transfer (Figure 4-4) utilizes normal processor timing; the control signals to the LATCH A flip-flop are derived from the basic clock (K1-2 S CLK H) and are directly related to machine time states (ISR or BSR). During operation the LATCH A flip-flop is clocked to zero (K1-2 REG GATE H) if the inverse data input indicates a data change (load) to the DATA PATHS during this ISR or BSR state. A non-active or non-asserted K6-2 LATCH A 15/0 H allows the DATA PATHS latches to be cleared. Independent of whether the LATCH A flip-flop is cleared or not, the flip-flop is set by the inverse of K1-2 REG LATCH H and the absence of DATA WAIT (1) H. This assertion or continuation of assertion of K6-2 LATCH A15/0 H holds data in the DATA PATHS latches regardless of machine state or transients at machine state change (noted by shaded areas on "Inverse Data Input to LATCH A").

Either the LATCH A or LATCH B flip-flop may be independently "loaded to zero" and set in this way. The enabling signals for the "load to zero" on the LATCH flip-flops are also the load enable signals to the DATA PATHS inputs.

4.10.2 Unibus Data Transfer

The Unibus data transfer (Figure 4-5) utilizes some processor timing, the asynchronous bus signals, and an additional machine state called DATA WAIT. During the BSR state in which the processor is to halt for an asynchronous data input, the DATA WAIT flip-flop is "loaded to one" by the inverse data input (K6-2 DATA WAIT 1 L) and clock (K1-2 R/W2 H). Note that this occurs on the entry to R/W2, the clock phase in which the machine halts. The clocking of DATA WAIT to one, produces a pulse (K6-2 P CLR LATCHES L) that clears both LATCH flip-flops (and both sets of input latches on DATA PATHS). The machine remains in this state (clock halted) until a bus control signal (here typified by K6-2 P DATA START H and enabled by K6-2 DATA WAIT (1) H) sets the LATCH flip-flops. This stores the bus data in the appropriately gated latches on DATA PATHS and feeds around through K6-2 LATCH B (1) L and K6-2 P CLR DATA WAIT L to clear DATA WAIT.

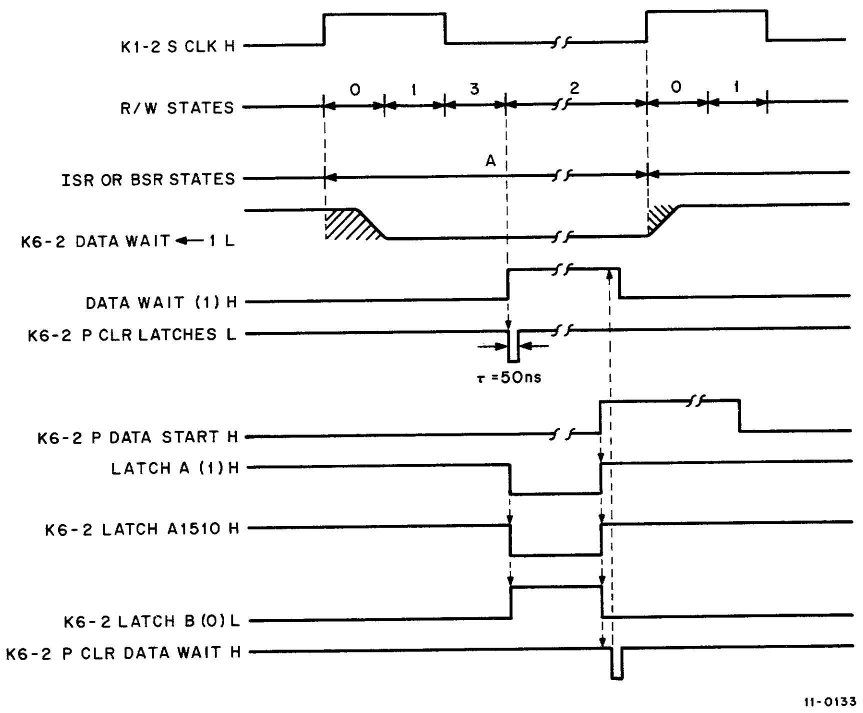


Figure 4-5 Unibus Data Transfer Timing Diagram

Both LATCH A and LATCH B are simultaneously loaded to one and set. No data is held in the DATA PATHS latches during a bus data in transfer; the data transfer, however, can be to either the A Input or B Input.

The above sequences are also initiated for certain console operations. Console Shift Register (CSR) timing is used instead of the basic processor clock (K1-2 S CLK L). The DATA WAIT flip-flop, in this instance, is set (K6-2 P SET DATA WAIT L) instead of "loaded to one". The asynchronous bus control signal is present on Unibus transfers; on internal transfers (REGISTER) the signal K13-4 P2 CSR H into pin 5 of E36 provides the set signal to the LATCH flip-flops. Note that this signal does not become part of the processor clock restart signal (K6-2 P DATA START H); the processor clock is not used for console timing. Details of console operation are noted in the KY11-A manual and the BUS & CONS CNTL discussion of the KA11 manual.

K6-2 LATCH B (0) H provides a test point for the LATCH B flip-flop.

K6-2 LATCH B15/0 H provides the storage signal for all B Inputs of the DATA PATHS; this signal holds the data in the B inputs. This signal is usually active, except for a clear before load. A flip-flop (LATCH B) is used to insure transient-free storage across machine state boundaries.

Details of latch operation are noted under the signal K6-2 LATCH A15/0 H. The B LATCH is activated at different times during internal data transfers; this difference is apparent from the inverse data inputs to LATCH A and LATCH B flip-flops.

K6-2 P CLR DATA WAIT H provides a test point for the clear signal to the DATA WAIT flip-flop. Its pulse activation from the resetting of LATCH B flip-flop has been noted under the signal K6-2 LATCH A 15/0 H. It is incidentally activated by K15-2 WAIT CLR H whose main purpose is to clear the latches on DATA PATHS (K6-2 P CLR LATCHES L) for a possible INTR cycle; the DATA WAIT flip-flop is already cleared at that time.

K6-2 DATA WAIT ← 1 L provides an inverted data input to "load to one" the DATA WAIT flip-flop. Its use has been noted under the signal K6-2 LATCH A 15/0 H; it is also used to turn the processor clock off (K1-2 CLK RUN ← 0 L). The DATA WAIT flip-flop is active only on a data in transfer from the Unibus. On a data out transfer, the latches of DATA PATHS are not altered and the restart signal (K6-2 P DATA START H) is inhibited from the LATCH flip-flops.

K6-3 GATE A ← R15/1 H enables REGISTER data for bits ⟨15:1⟩ to the A Input of the DATA PATHS. Transients from machine state changes, which might destroy data, are eliminated by the gating timing signal, K1-2 REG GATE H. This signal is not active until two R/W states after machine state changes (K1-2 S CLK H).

K6-3 GATE A ← -R15/1 H enables the inverse of REGISTER DATA FOR BITS ⟨15:1⟩ to the A Input of the DATA PATHS. Transient-free activation is as noted in signal K6-3 GATE A ← R15/1 H.

K6-3 GATE A ← R0 H enables the REGISTER data for bit 00 to the A Input of the DATA PATHS. Segmentation exists on the A Input bits to allow the generation of constants. Transient-free activation is as noted in signal K6-3 GATE A ← R15/1 H.

K6-3 GATE A ← -R0 H enables the inverse of REGISTER data for bit 00 to the A Input of the DATA PATHS. Segmentation exists on the A Input bits to allow the generation of constants. Transient free activation is as noted in signal K6-3 GATE A ← R15/1 H.

K6-3 GATE LEFT 15/0 H enables the rotate/shift gating of the DATA PATHS; bit n of the adder output is enabled to bit (n + 1) of the DATA PATHS output; specific data is supplied bit 00 (K10-4 LEFT DATA 00 L).

K6-3 GATE RIGHT 15/0 H enables the rotate/shift gating of the DATA PATHS; bits (n + 1) of the adder output is enabled to bit n of the DATA PATHS output; specific data is supplied bit 15 (K10-4 RIGHT DATA 15 L).

K6-3 GATE ADD 15/8 H enables the adder output, bits ⟨15:08⟩ directly to the DATA PATHS outputs whenever no other function of the rotate/shift gating is enabled.

K6-3 GATE ADD 7/0 H enables the adder outputs, bits ⟨07:00⟩, directly to the DATA PATHS outputs whenever no other function of the rotate/shift gating is enabled.

K6-3 EXTRA H provides for extra machine cycles necessary on certain instructions (BIC, BIT and ROT/SHF to an odd byte). The signal is used: to alter machine flow (K2-2 SHIFT ISR H and K2-2 ISR ← 0 L); to alter the Condition Codes clocking on ROT/SHF to an odd byte (K10-4 CLK N, Z, V, H and K10-4 CLK C H); to recycle data through the TEMP location of REGISTER (K4-3 W/ENABLE 15/8 L, K4-3 W/ENABLE 7/0 L and K4-2 RA/TEMP L) to load the DATA PATHS (K6-2 CLR LATCH B L).

K6-3 GATE BYTE 7/0 H enables the upper byte data from the adder bits ⟨15:08⟩ to the DATA PATHS outputs bits ⟨07:00⟩.

K6-3 GATE BYTE 15/8 H enables the lower byte data from the adder bits ⟨07:00⟩ to the DATA PATHS outputs bits ⟨15:08⟩.

```
K6-3 GATE A ← R 15/1 H = REG GATE {(ENABLE A ← R) + ADD (-1) + ADD (-2)}
K6-3 GATE A ← R15/1 H = REG GATE { (ENABLE A ← -R) + ADD (-2) + ADD (-1)}
K6-3 GATE A ← R0 H = REG GATE {(ENABLE A ← R) + ADD (-1) + ADD (+2) }
K6-3 GATE A ← -R0 H = REG GATE { (ENABLE A ← -R) + ADD (-1) + ADD (+2)}
K6-3 GATE LEFT 15/0 H = B EXEC (1) [ { ISR3 + ISR15 * BAR00 (0)} ROT/SHF L + ISR1 * BRANCH]
K6-3 GATE RIGHT 15/0 H = B EXEC (1) [ { ISR3 + ISR15 * BAR00 (0) } ROT /SHF R]
K6-3 GATE ADD 15/8 H = - [GATE LEFT 15/0 + GATE RIGHT 15/0 + GATE BYTE 15/8]
K6-3 GATE ADD 7/0 H = - [GATE LEFT 15/0 + GATE RIGHT 15/0 + GATE BYTE 7/0]
K6-3 EXTRA H = BIT + BIC + ROT/SHF * BAR00 (1)
K6-3 GATE BYTE 7/0 H = B EXEC (1) * ISR15 * SWAB + BSR15
K6-3 GATE BYTE 15/8 H = B EXEC (1) [ISR15 { SWAB + (U + B + R/S) * BAR00 (1) * - (DEST MODE 0) } ]
K6-3 ADD (-1) L = B EXEC (1) * ISR1 * { SBC + DEC + C (1) } + (SO + DE) * ISR1 * BSR1 * ADRS BYTE OP * ADRS MODE 4
K6-3 ENABLE A ← R L = B DEST (1) * BSR14 * (A ← DEST/INSTR) + B EXEC (1) [ISR7 * (BRANCH + ROT/SHF) + ISR1 * (ADD + BIC)] + (SO +
                      DE) * BSR1 * ISR3 * ADR MODE (6+7)
K6-3 ADD (-2) L = B EXEC (1) * ISR0 * BSR1 * JSR + (SO + DE) * ISR1 * BSR1 * ADRS MODE (4+5) + SERVICE * ISR (3+7) * BSR1
K6-3 ENABLE A ← -R L = B EXEC (1) [ISR0 * (A ← DEST/INSTR) + ISR1 * (SUB + BIT) + ISR7 * (- ROT/SHF * EXTRA)]
```

K6-4 GATE B ← B D15/0 H enables buffered bus data for bits ⟨15:00⟩ to the B Input of the DATA PATHS. Transient-free activation is effected by the gating of K6-2 DATA WAIT (1) H.

K6-4 GATE A ← -B D15/0 H enables the inverse of buffered bus data for bits ⟨15:00⟩ to the A Input of the DATA PATHS. Transient-free activation is effected by the gating of K6-2 DATA WAIT (1) H.

K6-4 GATE B ← R7/0 H enables the REGISTER data for bits ⟨07:00⟩ to the B Input of the DATA PATHS. Transients from machine state changes, which might destroy data, are eliminated by the gating timing signal, K1-2 REG GATE H.

K6-4 GATE B ← STPM H enables the Special TraP Markers (STPM) to the B Input of the DATA PATHS.

K6-4 GATE B ← R15/8 H enables the REGISTER data for bits ⟨15:08⟩ to the B Input of the DATA PATHS. Transient-free activation is as noted in K6-4 GATE B ← R7/0 H.

K6-4 GATE SEX H enables Sign EXtension (SEX) into the upper byte B Inputs of DATA PATHS. Transient-free activation is as noted in K6-4 GATE B ← R7/0 H.

```
K6-4 GATE B ← B D15/0 H = { - B DEST (1) + - ADRS DONE } (A ← DEST/INSTR) * DATA WAIT (1)

K6-4 GATE A ← - B D 15/0 H = B DEST (1) * DATA WAIT (1) * (A ← DEST/INSTR) (ADRS DONE)

K6-4 ENABLE B ← R L = B EXEC (1) * [ISR1 * GATE B/ISR1 + ISR15 * RTS + ISR7 * JSR + ISR3 * BRANCH] + SERVICE * ISR0 * HALT F (1) +
                    BSR 14 (- ENABLE A ← R) + BSR1 * (- DEP) * - [ISR7 * (SO + DE) + ADRS MODE (3+5+6+7)]

K6-4 GATE B ← R7/0 H = REG GATE * (ENABLE B ← R)

K6-4 GATE B ← STPM H = SERVICE * TRAPS * SERV 0 * - (HALT F (1))

K6-4 GATE B ← R15/8 H = REG GATE * (ENABLE B ← R) * - SEX

K6-4 SEX H = B EXEC (1) * [ISR3 * BRANCH + ISR1 * IR15 * MOV * DEST MODE 0]

K6-4 GATE SEX H = REG GATE * - SEX * { MOV * R07 (1) + BRANCH * R08 (1) }
```

K6-5

K6-5 CARRY 00 (1) L provides the carry input data to bit 00 of the DATA PATHS adder. Storage of this input information across machine state boundaries is effected by the discrete gate Set-Reset flip-flop. This flip-flop is set by the signal K6-5 CARRY 00 ← 1 L and cleared by the signal K6-2 CARRY 00 ← 0 L, K6-2 P CLR LATCHES L and K6-2 CSR 2 L.

K6-5 ADD (+2) L = BSR3 [B FETCH (1) * ISR0 + SERVICE * ISR (12+15) + B EXEC (1) * { ISR (12+15) * RT1 + ISR12 * RTS } + (DO + DE) * ISR1 * ADRS BIT 1 * (- ADRS BYTE OP + - ADRS MODE 2)] + CSR1 (DEP + EXAM) * INCF * - (REF ADRS)

K6-5 CARRY 00 ← 0 L = B EXEC (1) * BSR1 * JSR + SERV0 + FETCH (1) * BSR1 + BSR0 * DATO ENTRY

K6-5 CARRY 00 ← 1 L = B EXEC (1) * ISR1 * CARRY INSTR + (SO + DE) * BSR3 * ADRS BIT 1 + ADD (+2) + CSR1 * INCF * (DEP + EXAM)

K6-5 CARRY 00 (1) L = (CARRY 00 ← 1) * CARRY 00 + CARRY 00 (1) * - (P CLR LATCHES) * - (CSR2) * [- CARRY 00 + - CARRY 00 ← 0]

4.11 DATA PATHS, M224

4.11.1 General

The DATA PATHS are central to the flow of data through the KA11 Processor. Its input receives all data from the bus; its output provides both address and data information to the bus; its inputs and outputs to the REGISTER complete an internal data flow loop. All transfer and alteration of data within the processor is done through the DATA PATHS. Two DATA PATHS modules are used within the KA11 to provide the 16 parallel DATA PATHS. The following discussion relates to both M224 modules, DATA PATHS 1 and DATA PATHS 2.

The DATA PATHS consist of three main segments: the input gating and latches; the adder; and the output rotate/shift gating.

4.11.2 Input Gating and Latches

The input gating provides each of the two adder inputs (A input and B input) with input selection, necessary *input* inversion, and storage. A latch storage technique, integral to the selection gate, provides rapid data transmission to the adder with subsequent data retention. The A input gating selects: the complement of bus data; register data; and the complement of register data. The B input gating selects: bus data; register data; and special trap markers or sign extend.

The A input gating is further used to generate constants within the DATA PATHS. The simultaneous enabling of data and its complement into an oring gate (the 74H53 gate) produces a logical "1" input. This logic certainly ($X + \overline{X} = 1$) is used within the A inputs on register data, with the A inputs bit segmented to provide different constants. This segmentation is used in conjunction with non-gating (no inputs enabled) which produces a logical "0" input. For example, the constant -2, $(177776)_8$, is produced by enabling data and its complement on bit 15 through 01, and non-gating bit 00. The constant -1, $(177777)_8$, is produced by enabling data and its complement on all 16 bits. The constant +1, $(000001)_8$, and +2 $(000002)_8$ use the carry input to the adder to provide a constant +1. For the constant +2, the A input gating is also used to produce an additional +1 with non-gating on bits 15 through bit 01, and data and its complement on bit 00.

The latches provide data storage for the A and B inputs to the adder. The 74H53 gate with its multiple inputs and the feedaround 74H00 gate provides a Set-Reset type of flip-flop. A simplified redrawing (Figure 4-6) of the input gating emphasizes this:

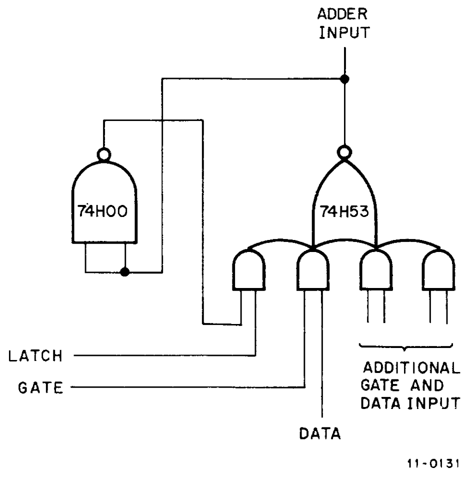


Figure 4-6 Latch Input Gating

Disabling the LATCH input with the GATE (GATE here representing any of the gating inputs) signal low results in the flip-flop being reset or cleared. The "0" side of the flip-flop, the ADDER INPUT, is high. Disabling the LATCH input with the GATE signal high loads the flip-flop with the DATA signal. If the LATCH signal is enabled before the GATE signal is removed, the DATA signal is stored within the latch.

Storage of information within the DATA PATHS provides information across machine state boundaries. The first of two operands from REGISTER is stored while the second is being obtained. A write back into REGISTER with the data, a function of REGISTER output, is possible. Bus data is latched into the DATA PATHS, allowing the bus or peripheral read/write memory to finish their cycles.

4.11.3 Adder

A conventional MSI adder is used, two bits within each package. Each bit has a summation output (Σ), an A and a B input. Every two bits have a carry-out and a carry-in signal. All signals into and out of the adder are asserted at the low level. Note that if only one operand (A or B) exists in the adder (the other input zero) the adder is merely a through gate.

4.11.4 Rotate/Shift Gating

The rotate/shift gating on the adder output allows selection of certain data as the Data Path output. Selections include: the direct output of the adders; the adder output shifted (or rotated) one bit position right or left; and lower or upper segments (bytes) of the complete (16 bits) adder. The direct output is used in most instances (the adder is altering data or merely transmitting it). The shifted (or rotated) output is utilized with ROT/SHF instructions and when a signal bit left justification is required of an offset. The gating of segments (bytes) is used for: right justifying upper byte data for instruction manipulation (all byte data is right justified within the processor); the gating of output byte data to the upper bus data lines (they also appear on the lower data path outputs for Conditions Codes sensing); and for swapping upper and lower byte segments in the SWAB instruction.

The output signal of the data path provides asserted data levels high. If the direct output is to be used, the output of the adder provides asserted data at the low level.

4.12 DATA PATHS 1

Since the Data Paths are redundant and heavily concerned with selection, both input and output (K7-2 prefixes) signals are discussed. This module provides the data paths for bits ⟨07:00⟩, most having common operation. The discussion will be ordered toward this lower portion of word data, with the exceptions for byte operation and bit 00 noted. Some of the gating signals reference wider bit segments than DATA PATHS 1 encompasses, extract appropriate bit segments for DATA PATHS 1. Multiple inputs of the same signal are occasionally used as the M224 module accommodates different bit functions on DATA PATHS 1 and DATA PATHS 2.

K7-2 D ⟨07:00⟩ H provides main data path output for bits 07 through 00 which is used for bus data, bus address (BAR input) STATUS data, or processor operands (REGISTER input). This output represents the complete interaction of present and past Data Paths gating and input data.

K7-2 ADD ⟨07:00⟩ L provides the direct adder output of the data paths prior to the rotate/shift gating. For single shifts (or rotates) most of these outputs are used internal to the module, only the end bits are utilized elsewhere. For byte manipulation, the adder outputs are the input to the DATA PATHS 1 rotate/shift gating. For situations where the adder output is directly gated, it is the complement data of the main data path output.

K6-3 GATE RIGHT 15/0 H enables the rotate/shift gating to transfer the (nth + 1) bit adder output to the nth bit data path output.

K6-3 GATE ADD 7/0 H enables the rotate/shift gating to transfer directly the nth bit data path output.

K6-3 GATE BYTE 7/0 H enables the rotate/shift gating to transfer the upper byte data (adder output bits 15 to 08) to the lower data paths outputs, bits 07 to 00 respectively.

K6-3 GATE LEFT 15/0 H enables the rotate/shift gating to transfer the nth-1 bit adder output to the nth bit data path output.

K10-4 RIGHT DATA 07 L provides proper rotate or shift data to the bit 07 data path output upon a right rotate or a right shift. See discussion on K10-4 outputs.

K7-2 ADD ⟨15:08⟩ L provides the inputs to the rotate/shift gates for the transfer of upper byte information to the lower byte by K6-3 GATE BYTE 7/0 H.

K10-4 LEFT DATA 00 L provides the (nth -1) bit data for bit 00 for shift left or rotate left; gated by K6-3 GATE LEFT 15/0 H.

K7-2 CARRY 07 L is the output carry from bit 07 of the adder. This is input data for adder bit 08 and the C bit of Condition Codes for byte operation.

K6-5 CARRY 00 (1) L is the input carry to bit 00 of the adder, providing for the incrementation of data for instruction execution (INC, ADC) and constant generation (+1, +2).

K7-2 DATA 07 L provides the A input to bit 07 of the adder, after input selection. Bit 07 is the sign bit of an arithemetic byte; this output is used in calculating overflow for the V bit of the Condition Codes.

K6-2 LATCH A 15/0 H enables the A input latch to store the selected A input data. The load operation requires that this signal be disabled during the loading (to clear the latch) and enabled prior to data removal. Data remains stored until the disabling of the LATCH A 15/0 signal. If a subsequent load sequence omits the disabling of the signal (clearing) an "oring" of latch content and input data occurs. Simultaneously enabling of more than one input also effects this (the use of X and X to produce constants).

K6-3 GATE A ← R 15/1 H enables the REGISTER output to the A input of the adder for bits 07 to 01.

K6-6 A ← R 15/1 H enables the complement of the REGISTER output to the A input of the adder for bits 07 to 01. The complement is provided by NAND gate inversion upon the modules. The simultaneous enabling of the true and complement of the REGISTER into the same adder input bit allows the generation of a "1" in that bit. The restriction of the gating signal to bits ⟨15:1⟩ allows flexibility in generating the constant in the low order bit.

K6-3 GATE A ← -B D15/0 H enables the complement of buffered bus data to the A input of the adder. The complement is provided by NAND gate inversion upon the module.

K6-3 GATE A ← R0 H enables the REGISTER output to the A input of the adder for bit 00. This input is segmented from the R input gate to allow the generating of constants (+R, -1 and -2). This signal is enabled with K6-3 GATE A ← R15/1 for the loading of REGISTER data.

K6-3 GATE A ← -R0 H enables the complement of the REGISTER output to the A input of the adder for bit 00. The complement is provided by NAND gate inversion on the module. This input is segmented from the other -R inputs to allow the generation of constants (+2, -1, and -2). This signal is enabled with K6-3 GATE A ← R15/1 for the loading of REGISTER data.

K7-2 B DATA 07 L provides the B input to bit 07 of the adder after input selection. Bit 07 is the sign bit of an arithmetic byte; this output of the adder input is used in calculating overflow for the V bit of Condition Codes.

K6-4 GATE B ← D15/0 H enables the buffered bus data to the B input of the adder.

K6-2 LATCH B 15/0 H enables the A Input Latch to store the selected B input data. See discussion under K6-2 LATCH A 15/0 H for loading and storing. No "oring" takes place within the B Latch.

GND 09 disables some unnecessary A inputs.

K6-4 GATE B ← R7/0 H enables the REGISTER outpput to the B input of the adder.

K6-4 GATE B ← STPM H enables the address data of the Special TraP Markers used during trap sequences. Only bits 05, 04, 03, and 02 are enabled with the data of bit 05 disabled by the GND 09 signal.

K3-3 B D ⟨15:08⟩ provides the buffered bus data for the A and B adder input selection gates. Inversion of this data to produce the complement occurs on the module.

K5-2 R ⟨15:08⟩ (1) H provides the REGISTER data for the A and B adder input selection gates. NAND gate inversion of this data to produce the complement occurs on the module.

K12-3 STPM ⟨04:02⟩ provides the address data for the Special TraP Markers. This address reflects the specific trap being sequenced by the processor and is enabled to the data path for temporary storage in REGISTER.

4.13 DATA PATHS, M224

4.13.1 General

The DATA PATHS are central to the flow of data through the KA11 Processor. Its input receives all data from the bus; its output provides both address and data information to the bus; its inputs and outputs to the REGISTER complete an internal data flow loop. All transfer and alteration of data within the processor is done through the DATA PATHS. Two DATA PATHS modules are used within the KA11 to provide the 16 parallel data paths. The following discussion relates to both M224 modules, DATA PATHS 1 and DATA PATHS 2.

The DATA PATHS consist of three main segments: the input gating and latches; the adder; and the output rotate/shift gating.

4.13.2 Input Gating and Latches

The input gating provides each of the two adder inputs (A input and B input) with input selection, necessary *input* inversion, and storage. A latch storage technique, integral to the selection gate, provides rapid data transmission to the adder with subsequent data retention. The A input gating selects: the complement of bus data; register data; and the complement of register data. The B input gating selects: bus data; register data; and special trap markers or sign extend.

The A input gating is further used to generate constants within the DATA PATHS. The simultaneous enabling of data and its complement into an oring gate (the 74H53 gate) produces a logical "1" input. This logic certainly ($X + \overline{X} = 1$) is used within the A inputs on register data, with the A inputs bit segmented to provide different constants. This segmentation is used in conjunction with non-gating (no inputs enabled) which produces a logical "0" input. For example, the constant -2, $(177776)_8$, is produced by enabling data and its complement on bit 15 through 01, and non-gating bit 00. The constant -1, $(177777)_8$, is produced by enabling data and its complement on all 16 bits. The constant +1, $(000001)_8$, and +2 $(000002)_8$, use the carry input to the adder to provide a constant +1. For the constant +2, the A input gating is also used to produce an additional +1 with non-gating on bits 15 through bit 01, and data and its complement on bit 00.

The latches provide data storage for the A and B inputs to the adder. The 74H53 gate with its multiple inputs and the feedaround 74H00 gate provides a Set-Reset type of flip-flop. A simplified redrawing of the input gating emphasizes this:

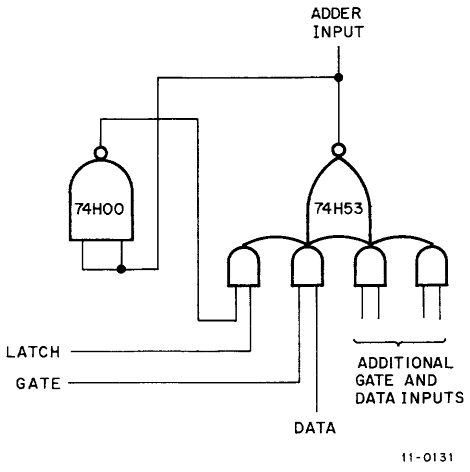


Figure 4-7 Latch Input Gating

Disabling the LATCH input with the GATE (GATE here representing any of the gating inputs) signal low results in the flip-flop being reset or cleared. The "0" side of the flip-flop, the ADDER INPUT, is high. Disabling the LATCH input with the GATE signal high loads the flip-flop with the DATA signal. If the LATCH signal is enabled before the GATE signal is removed, the DATA signal is stored within the latch.

Storage of information within the DATA PATHS provides information across machine state boundaries. The first of two operands from REGISTER is stored while the second is being obtained. A write back into REGISTER with the data, a function of REGISTER output, is possible. Bus data is latched into the DATA PATHS, allowing the bus or peripheral read/write memory to finish their cycles.

4.13.3 Adder

A conventional MSI adder is used, two bits within each package. Each bit has a summation output(Σ), an A and a B input. Every two bits have a carry-out and a carry-in signal. All signals into and out of the adder are asserted at the low level. Note that if only one operand (A or B) exists in the adder (the other input zero) the adder is merely a through gate.

4.13.4 Rotate/Shift Gating

The rotate/shift gating on the adder output allows selection of certain data as the Data Path output. Selections include: the direct output of the adders; the adder output shifted (or rotated) one bit position right or left; and lower or upper segments (bytes) of the complete (16 bits) adder. The direct output is used in most instances (the adder is altering data or merely transmitting it). The shifted (or rotated) output is utilized with ROT/SHF instructions and when a signal bit left justification is required of an offset. The gating of segments (bytes) is used for: right justifying upper byte data for instruction manipulation (all byte data is right justified within the processor); the gating of output byte data to the upper bus data lines (they also appear on the lower data path outputs for Conditions Codes sensing); and for swapping upper and lower byte segments in the SWAB instruction.

The output signal of the data path provides asserted data levels high. If the direct output is to be used, the output of the adder provides asserted data at the low level.

4.14 DATA PATHS 2

Since the Data Paths are redundant and heavily concerned with selection, both input and output (K8-2 prefixes) signals are discussed. This module provides the data paths for bits ⟨15:08⟩, most having common operation. The discussion will be ordered toward this upper portion of word data, with the exception for byte operation noted. Some of the gating signals reference wider bit segments than DATA PATHS 2 encompasses; extract appropriate bit segments for DATA PATHS 2. Multiple inputs of the same signal are occasionally used, as the M224 module accommodates different bit functions on DATA PATHS 1 and DATA PATHS 2.

K8-2 D ⟨15:08⟩ H – provides the main data path output for bit 15 through bit 08 which is used for bus data, bus address (BAR input) or processor operands (REGISTER input). This output represents the complete interaction of present and past data paths gating and input data.

K8-2 ADD ⟨15:08⟩ L provides the direct adder output of the data paths prior to the rotate/shift gating. For single shifts (or rotates) most of these outputs are used internal to the module, only the end bits are utilized elsewhere. For byte manipulation, the adder outputs are the input to the DATA PATHS 1 rotate/shift gating. In situations where the adder output is directly gated; it is the complement data of the main data path output.

K6-3 GATE RIGHT 15/0 H enables the rotate/shift gating to transfer the (nth + 1) bit adder output to the nth bit data path output.

K6-3 GATE ADD 15/8 H enables the rotate/shift gating to transfer directly the nth bit adder output to the nth bit data path output.

K6-3 GATE BYTE 15/8 H enables the rotate/shift gating to transfer the lower byte data (adder output bits 07 to 00) to the upper data paths outputs, bits 15 to 08, respectively.

K6-3 GATE LEFT 15/0 H enables the rotate/shift gating to transfer the nth -1 bit adder output to the nth bit data path output.

K10-4 RIGHT DATA 15 L provides proper rotate or shift data to the bit 15 data path output upon a right rotate or a right shift. See discussion on K10-4 outputs.

K7-2 ADD ⟨07:00⟩ L provides the inputs to the rotate/shift gates for the transfer of lower byte information to the upper byte by K6-3 GATE 15/8 H.

K7-2 ADD 07 L provides the (nth -1) bit data for bit 08 on shift left or rotate left; gated by K6-3 GATE LEFT 15/0 H.

K8-2 CARRY 15 L is the output carry from bit 15 of the adder. This is input data for the C bit of the STATUS word.

K7-2 CARRY 07 L is the input carry to bit 08 of the adder, during addition in the data paths.

K8-2 A DATA 15 L provides the A input to bit 15 of the adder after input selection. Bit 15 is the sign bit of an arithmetic word and this output is used in calculating overflow for the V bit of Condition Codes.

K6-2 LATCH A 15/0 H enables the A input latch to store the selected A input data for the load operation requires that this signal be disabled during the loading (to clear the latch) and enabled prior to data removal. Data remains stored until the disabling of the LATCH A 15/0 signal. If a subsequent load sequence omits the disabling of the signal (clearing) an "oring" of latch content and input data occurs. Simultaneously enabling of more than one input also effects this (the use of X and X to produce constants).

K6-3 GATE A ← R15/1 H enables the REGISTER output to the A input of the adder for bits 15 to 08. The signal inputs twice to all for the gating of the lower bit required of the M224 in DATA PATH 1 use.

K6-3 A ← R15/1 H enables the complement of the REGISTER output to the A input of the adder for bits 15 to 08. The complement is provided by NAND gate inversion upon the module. The signal inputs twice to allow for the gating of the lower bit required of the M224 in DATA PATH 1 use. The simultaneous enabling of the true and complement of the REGISTER into the same adder input bit allows the generation of a "1" in that bit. The restriction of the gating signal to bits ⟨15:1⟩ allows flexibility in generating the constant in the low order bit (see DATA PATHS 1).

K6-3 GATE A ← -B D15/0 H enables the complement of buffered bus data to the A input of the adder. The complement is provided by NAND gate inversion upon the module.

K8-2 B DATA 15 L provides the B input to bit 15 of the adder after input selection. Bit 15 is the sign bit for an arithmetic word and this output of the adder input is used in calculating overflow for the V bit of Condition Codes.

K6-4 GATE B ← B D15/0 H enables the buffered bus data to the B input of the adder.

K6-2 LATCH B 15/0 H enables the A Input Latch to store the selected B input data. See discussion under K6-2 LATCH A 15/0 H for loading and storing. No "oring" takes place within the B latch.

K6-4 GATE SEX H enables all 1s into the B input latch bits ⟨15:8⟩ and effects the Sign EXtension of negative byte data (sign bit is 1) through all bits of the upper byte when desired. The extension of a positive byte sign bit (sign bit is 0) utilizes the cleared state of the latches and requires no active signal. Multiple inputs both as gate and data signals accommodate diverse function of the M224 modules in DATA PATHS 1 and DATA PATHS 2.

K6-4 GATE B ← R15/8 H enables the REGISTER output to the B input of the adder.

K3-3 B D ⟨15:08⟩ provides the buffered bus data for the A and B adder input selection gates. Inversion of this data to produce the complement occurs on the modules.

K5-2 R ⟨15:08⟩ (1) H provides the REGISTER data for the A and B adder input selection gates. NAND gate inversion of this data to produce the complement occurs on the module.

4.15 BUS INTERFACE & IR

Besides the Instruction Register, the module contains: combination logic for output gating, address sensing, and data detection; the Bus Address Register (BAR); the processor Status word (STATUS); and the bus interface for BAR, STATUS and lower byte data (D⟨07:00⟩).

The instruction Register (IR ⟨15:00⟩) accepts instruction data directly from the bus after input buffering; its outputs and decoded outputs are used throughout the processor for machine flow decisions and register address data. The Bus Address Register (BAR) is used by the processor and console for all bus addresses. Data input is directly from the DATA PATH outputs with the Unibus gated from the register, the register contents appear in the consoles ADDRESS REGISTER display.

Processor STATUS word occupies the lower byte of bus address $(777776)_8$ and consists of three bits for Priority (ST⟨07:05⟩) and five bits for Condition Codes (T, N, Z, V, C). The upper portion is directly loaded from the DATA PATHS while the Condition Codes require the additional gating of CODES DATA and specific gating dependent on the last instruction. Also associated with the N, Z, V, and C bits are set and clear gating to implement the microprogrammed instruction to set or clear specific Condition Codes. The direct STATUS register outputs are used within the processor on PRIORITY, CODES DATA, IR DECODE and DATA PATH CNTL for data and for decision.

The output of STATUS to the Unibus data lines is "wired or'ed" within the module to the Unibus output of lower byte data from the DATA PATHS. A minimum of module pins are expended in this arrangement.

K9-2

The module is essentially a register module (IR, BAR, STATUS) with its combinational logic directly related to the input, output or output (bus) control signals. Two signals (K9-2 GATE BUS ← D H and K9-2 GATE BUS ← BAR H) are derived to gate data from the module to the Unibus. The remaining signals conserve back panel wiring by directly decoding the register output or the data inputs on the module.

A discussion of these signals follows:

K9-2 GATE BUS ← D H enables the DATA PATH output to the Unibus. The data bus interface from DATA PATHS uses this for lower byte data on this module (prints K9-4 and K9-5) and on PRIORITY for upper byte data. The signal is usually enabled during the latter portion of DATO or DATOB processor operations with specific inhibit by the signals that enable the STATUS word or SR (console) to the bus.

K9-2 GATE BUS ← ST H enables the STATUS word to the Unibus. The signal is used only within this module (prints K9-4 and K9-5); and is active when STATUS is explicitly (instruction or console address) or implicitly (DATO in a trap sequence) addressed.

K9-2 GATE BUS ← BAR H enables the BAR output to the address lines of the Unibus. This signal is used only within the module and is active whenever the console or the processor is in control of the bus (K13-4 PROC CNTL H).

K9-2 REG ADRS H detects in the BAR the bus addresses assigned to the processor registers, $(777700)_8$ to $(777717)_8$. These addresses are not detected in the processor and are used only by the console. During console operation the signals are used: to inhibit the start of the processor clock for EXAM and DEP (K1-3 P CLK RESTART) to select the BAR address input to REGISTER (K4-3 GATE RA ← BAR) and to write if a DEP (K4-3 W/ENABLE 15/8 L and K4-3 W/ENABLE 7/0 L); to gate the REGISTER output to the B input (K6-4 ENABLE B ← RL); and to limit the console incrementing (for EXAM and DEP) to one for REGISTER addresses (K6-5 ADD (+2) L. After inversion (K4-3 REG ADRS L) the signal inhibits an entry into BSR 1 for EXAM or DEP.

K9-2 ST ADRS H detects in the BAR the bus address assigned to the processor's STATUS word, $(777776)_8$. This address is used only when the STATUS word is directly addressed by an instruction or the console. On a read from STATUS, the word is gated onto the bus and the processor supplies the SSYN signal in response to its own MSYN signal. In this bus operation, the ST ADRS signal is used: to enable the STATUS word to the bus (K9-2 GATE BUS ← ST H) and to load the processor's SSYN flip-flop (K13-3 SSYN (1) H). The load to STATUS is done off the DATA PATHS outputs, through CODES DATA, without a bus cycle. The ST ADRS signal is used here: to inhibit the usual bus cycle entry (K2-3 DATO ENTRY H); to inhibit the write to REGISTER (K4-3 W/ENABLE 7/0 L and K4-3 W/ENABLE 15/8 L); and to enable the DATA PATHS to STATUS (K10-4 GATE ST ← D H) with appropriate clocking (K10-4 CLK N, Z, V, H, K10-4 CLK C H and K10-4 CLK T H).

K9-2 SR ADRS H detects in the BAR the bus address assigned to the console's SWITCH REGISTER, $(777570)_8$. This address is used only when the SWITCH REGISTER (SR) is directly addressed by an instruction. A pseudo bus operation, similar to the reading of STATUS, is done. The SR is gated to the bus (K13-2 GATE BUS ← SR H) and SSYN (K13-2 SSYN (1) H) is used.

K9-2 D 15/8 ZERO L detects all zeros on the upper byte data from the DATA PATHS. The signal is used to detect a stack address less than $(400)_8$ and set the stack OVer FLow Flag (K12-2 OVFLF (1) H).

K9-2 D 15/0 ZERO L detects all zeros on the complete word from DATA PATHS. The signal is gated within CODES DATA as the Z bit word data (K11-2 Z DATA H).

K9-2 D 7/0 ZERO L detects all zeros on the lower byte data from the DATA PATHS. The signal is gated within CODES DATA as the Z bit byte data (K11-2 Z DATA H).

K9-2 BAR ⟨17:16⟩ (1) H provides the upper two bits of the Bus Address Register and are used in the console's ADDRESS REGISTER display. These bits are in excess of the 16 bit data and address paths of the KA11 and accommodate future expansion. The bits are set if an address has D bits ⟨15:13⟩ set; they are cleared otherwise. These outputs are:

BUS A ⟨17:16⟩ L provides the proper Unibus "wire or'ed" signal for bits ⟨17:16⟩ of the Bus Address Register.

(continued on page 4-58)

K9-2 GATE BUS ← ST H = SERVICE * ISR3 * BSR (15+14+12+8) * BBSYF (1) + ST ADRS * PROC CNTL * SSYN (1)

K9-2 GATE BUS ← D H = BSR (15+14+12+8) * BBSYF (1) * [- GATE BUS ← SR + - (GATE BUS ← ST)]

K9-2 GATE BUS ← BAR H = PROC CNTL

K9-2 REG ADRS H = BAR17 (1) * BAR16 (1) * BAR15 (1) * BAR14 (1) * BAR13 (1) * BAR12 (1) * BAR11 (1) * BAR10 (1) * BAR09 (1) * BAR08 (1) * BAR07 (1) * BAR06 (1) * BAR05 (0) * BAR04 (0)

K9-2 D 15/0 ZERO L = - D15 * - D14 * - D13 * - D12 * - D11 * - D10 * - D09 * - D08 * - D07 * - D06 * - D05 * - D04 * - D03 * - D02 * - D01 * - D00

K9-2 D 7/0 ZERO L = - D07 * - D06 * - D05 * - D04 * - D03 * - D02 * - D01 * - D00

K9-2 ST ADRS H = BAR17 (1) * BAR16 (1) * BAR15 (1) * BAR14 (1) * BAR13 (1) * BAR12 (1) * BAR11 (1) * BAR10 (1) * BAR09 (1) * BAR08 (1) * BAR07 (1) * BAR06 (1) * BAR05 (1) * BAR04 (1) * BAR03 (1) * BAR02 (1) * BAR01 (1) * BAR00 (1)

K9-2 SR ADRS H = BAR17 (1) * BAR16 (1) * BAR15 (1) * BAR14 (1) * BAR13 (1) * BAR12 (1) * BAR11 (1) * BAR10 (1) * BAR09 (1) * BAR08 (1) * BAR07 (1) * BAR06 (1) * BAR05 (1) * BAR04 (1) * BAR03 (1) * BAR02 (0) * BAR01 (0) * PROC CNTL

K9-2 D 15/8 ZERO L = - D15 * - D14 * - D13 * - D12 * - D11 * - D10 * - D09 * - D08

K9-3 IR ⟨15:12⟩ (1) H provides the Instruction Register output for bits ⟨15:12⟩ that are used for instruction decoding on IR DECODE. Bit 15 is also directly used to generate a byte sign extension (K6-4 SEX H and K6-4 GATE SEX H) on a MOV to REGISTER.

K9-3 IR ⟨11:09⟩ (0) H provides the Instruction Register output for bits ⟨11:09⟩ that are used for instruction decoding on IR DECODE.

K9-3 IR 09 (1) H provides the Instruction Register output for bit 08 that is used for instruction decoding on IR DECODE. Bit 08 is also used as address data to the REGISTER for the instruction SOURCE base register.

K9-3 BAR ⟨15:08⟩ (1) H provides the Bus Address Register outputs for bits ⟨15:08⟩ and are used in the console's ADDRESS REGISTER display.

BUS A ⟨15:08⟩ H provides the proper Unibus signals for bits ⟨15:08⟩ of the Bus Address Register.

K9-4 BAR ⟨07:04⟩ (1) H provides the Bus Address Register output for bits ⟨07:04⟩ and are used in the console's ADDRESS REGISTER display.

BUS A ⟨07:04⟩ L provides the proper Unibus signals from bits ⟨07:04⟩ of the Bus Address Register.

K9-4 B D ⟨07:04⟩ H is the Unibus buffered inputs for bits ⟨07:04⟩ of lower byte data. The signal is used directly on the module by the Instruction Register (IR ⟨07:04⟩) and as inputs to DATA PATHS.

K9-4 IR ⟨07:06⟩ (1) H provides the Instruction Register output for bits ⟨07:06⟩ that is used for instruction decoding on IR DECODE. These bits are also used as address data to the REGISTER for the instruction SOURCE base register.

K9-4 IR ⟨05:04⟩ (0) H provides the Instruction Register output for bits ⟨05:04⟩ that are used for instruction decoding on IR DECODE. The output of IR04 is also used on BUS INTERFACE & IR to set or clear the Condition Codes N, Z, V, or C in the OPR instruction.

BUS D ⟨07:04⟩ L provides the Unibus interface for lower byte data from the STATUS word and DATA PATHS for bits ⟨07:04⟩. The gated outputs of each of these data sources are "wire or'ed" within the module. The enabling gates are activated separately (K9-2 GATE BUS ← D H or K9-2 GATE BUS ← ST H).

K9-4 ST ⟨07:05⟩ (1) H is the processor priority of the STATUS word and is used on PRIORITY in deciding bus control. This section of status is loaded directly from the DATA PATHS outputs by a loading signal, K10-4 CLK T H.

K9-4 T (1) H is the Trace (T) bit of the Condition Codes and is used to set the TRACF flip-flop on FLAG CNTL. The T bit is bit 04 of the STATUS word and is loaded (K7-2 D04 H) and gated to the bus (BUS D04 L) accordingly.

K9-5 BAR ⟨03:00⟩ (1) H provides the Bus Address Register output for bits ⟨03:00⟩ and is used in the console's ADDRESS REGISTER display. The bits ⟨03:00⟩ are also used as address data to the REGISTER for EXAM and DEP console operations. Bit 00 indicates whether a processor bus address is odd or even and is used: to initiate an odd address error (K10-3 ODD ADRS ERR L); and to provide for extra machine cycles if an upper byte operand must use the rotate/shift gating first for instruction operation (K6-3 EXTRA H and K2-2 SHIFT ISR H) and then for shifting the data (K6-3 GATE RIGHT 15/8 H).

K9-5 BAR 00 (0) H provides an addition output for bit 00 of the Bus Address Register; this output is used to allow the normal shift or rotate operation (K6-3 GATE LEFT 15/0 H or K6-3 GATE RIGHT 15/0 H) during ISR15 or to inhibit it because it has previously occurred.

BUS A ⟨03:00⟩ L provides the proper Unibus signals from bits ⟨03:00⟩ of the Bus Address Register.

K9-5 BD ⟨03:00⟩ L provides the proper Unibus signals from bits ⟨03:00⟩ of lower byte data. The signal is used directly on the module by the Instruction Register (IR ⟨03:00⟩ and as inputs to DATA PATHS.

K9-5 IR 03 (0) H provides the Instruction Register output for bit 03 that is used for instruction decoding on IR DECODE. The complementary output of IR03 is also used to set or clear the N bit of Condition Codes in the OPR instruction.

K9-5 IR ⟨02:00⟩ (1) H provides the Instruction Register for bits ⟨02:00⟩ that is used for instruction decoding on IR DECODE. These bits are also used as address data to the REGISTER for the DEST base register and to set or clear bits Z, V or C of the Condition Codes in an OPR instruction.

BUS D ⟨03:00⟩ L provides the Unibus interface for lower byte data from the STATUS word and DATA PATHS for bits ⟨03:00⟩. The gated output of each of these data sources are "wire or'ed within the module. The enabling gates are activated separately (K9-2 GATE BUS ← D H or K9-2 GATE BUS ← ST H).

K9-5 N (1) H is the Negative (N) bit of the Condition Codes and is used on IR DECODE as a BRANCH instruction enabling condition. The N bit is bit 03 of the STATUS word, reflecting last operands data, and is loaded (K11-2 N DATA L) and gated to the bus (BUS D03 L) accordingly.

K9-5 Z (1) H is the Zero (Z) bit of the Condition Codes and is used on IR DECODE as a BRANCH instruction enabling condition. The Z bit is bit 01 of the STATUS word, reflecting last operand's data, and is loaded (K11-2 V DATA L) and gated to the bus (BUS D01 L) accordingly.

K9-5 C (1) H is the Carry (C) bit of the Condition Codes and is used: on IR DECODE as a BRANCH instruction enabling condition; for data in ROTATE instructions (K10-4 I RIGHT DATA 07 L); for enabling an increment if carry (K10-4 CARRY INSTR H); and for enabling a decrement if carry (K6-3 ADD (-1) L). The C bit is bit 00 of the Condition Codes, reflecting last operands data, and is loaded (K10-4 C DATA H) and gated to the bus (BUS D00 L) accordingly.

K10-1

4.16 IR DECODE

The module decodes instructions and address modes. Decoders use the instruction register (IR) contents as input data; decode the instructions; and provide appropriate instruction signals, used throughout the machine for bus flow and time state alterations. IR 14/12 DCDR (E1) decodes bits 14, 13, and 12 of the instruction word to detect the double operand (Binary) instructions. IR 11/9 DCDR (E7) decodes the Branch instructions. IR 8/6 DCDR (E27) decodes the single operand (Unary) instructions excluding the Rotate/Shift instructions. IR 7/5 DCDR (E45) decodes the SWAB, Condition Codes OPR and JMP instructions. IR 2/0 DCDR (E54) decodes the OPERATE GROUP of instructions (HALT, WAIT, RTI, TRT, IOT, RESET). ADRS MODE DCDR (E/8) inputs are switched between SOURCE and DEST major states. In SOURCE, IR bits 03, 04, 05, the address mode field, are gated by K1-4 B SOURCE (1) H. In DEST, IR bits 09, 10, 11, the address mode field, are gated by K1-4 B DEST (1) H. Decoded address modes are gated with ISR time states to detect completion of address calculation (K10-3 ADRS DONE L).

Decoded Branch instructions are gated with STATUS word Condition Codes to test that conditions for the branch are met. Verification of conditions provides K10-2 BRANCH H to alter machine time states.

Special TraP Marker vector addresses (K12-3 STPM ⟨02:01⟩) are formed to service the OPERATE GROUP trap instructions. K10-4 INSTR STPM ⟨02:04⟩ reflect the specific instruction and gate with K12-3 TRAPF (1) H to provide the appropriate address.

Combinational logic provides the input gating and clocking for the STATUS word. Clock signals are separated (CLK N, Z, V; CLK C; CLK T) to alter portions of the STATUS word at times dependent upon decoded instructions. Input data for the C bit of STATUS is provided (K10-4 C DATA H); appropriate data is gated by the specific instruction.

For Rotate/Shift instructions, the input data for DATA PATH bits D14, D07, D00 is provided by K10-4 RIGHT DATA 07 L, K10-4 RIGHT DATA 15 L, K10-4 LEFT DATA 00 L.

K10-2 RSVD INSTR L detects an instruction code other than those specified in the KA11 repertoire.

K10-2 ADD L detects an ADD instruction.

K10-2 SUB L detects a SUBtract instruction.

K10-2 BINARY L detects a double operand (Binary) instruction.

K10-2 BINARY BYTE OP L detects a double operand (Binary) byte instruction.

K10-2 JSR L
K10-2 BIS L
K10-2 BIC L
K10-2 BIT L
K10-2 CMP L
K10-2 MOV L
K10-2 TRAP L
detect their respective instructions

K10-2 ROT L detects any Rotate instruction.

K10-2 ROT RIGHT H
K10-2 ROT LEFT H
detect specific Rotate instructions

K10-2 ROT/SHF L detects any Rotate/Shift instruction.

K10-2 Rot/SHF R H
K10-2 ROT/SHF L H
detect specific Rotate/Shift instructions.

K10-2 BRANCH H detects that a Branch instruction was verified for execution.

K10-2 TRUE BR L detects a verified Branch instruction that is dependent on appropriate Condition Codes being set.

K10-2 FALSE BR L detects a verified Branch instruction that is dependent on the appropriate Condition Codes being clear.

K10-2 BR L detects an unconditional branch.

K10-2 BR INSTR H detects any Branch instruction.

K10-2 TST L
K10-2 SBC L
K10-2 ADC L
K10-2 NEG L
K10-2 DEC L
K10-2 INC L
K10-2 COM L
K10-2 CLR L
detect their respective instructions.

K10-2 UNARY L detects a single operand (Unary) instruction.

K10-3

K10-3 SWAB L
K10-3 CC OP H
K10-3 JMP L
K10-3 RTS H
K10-3 RESET L
K10-3 IOT L
K10-3 RTI L
K10-3 WAIT L
K10-3 HALT L
K10-3 TRT L
} detect their respective instructions

K10-3 CHANGE CODES L gates direct set and direct clear signals to the Condition Code flip-flop as directed by a CC OP. Change occurs in ISR1 of FETCH major state.

K10-3 ILL INSTR L detects an ILLegal instruction (JMP or JSR with register mode destinations).

K10-3 RSVD OPS H detects instruction codes ReSerVeD for future instructions and is used as a component of K10-2 RSVD INST L.

K10-3 (U+R/S) H detects a Unary or Rotate/Shift instruction.

K10-3 BYTE OP H detects a byte instruction from IR bit 15 in the (1) state.

K10-3 ODD ADRS ERR L detects a reference to a word at an odd address.

K10-3 (U+B+R/S) L detects a Unary or Binary, or Rotate/Shift instruction.

K10-3 SOURCE MODE 0 L
K10-3 DEST MODE 0 L
K10-3 REG 6 L
K10-3 ADRS MODE (4+5) L
K10-3 ADRS MODE (6+7) H
K10-3 ADRS MODE (3+5+6+7) H
K10-3 ADRS MODE 5 L
K10-3 ADRS MODE 4 L
K10-3 ADRS MODE 2 L
} detects these respective addressing modes

K10-3 ADRS DONE L detects the last bus cycle for address calculation in SOURCE and DEST major states. Address modes 1, 2, 4 requiring one bus cycle, complete in ISR1. Modes 3, 5, 6 requiring two bus cycles, complete in ISR3. Address mode 7 requiring three bus cycles, completes in ISR7. The signal is used to alter machine flow (K2-2 ISR0 L) and suppress the data acquisition cycle for JMP or JSR instructions (K6-2 DATA WAIT1 L and K13-3 MSYN1 H).

K10-3 ADRS BIT 1 H detects IR bit 10 in SOURCE major state and IR bit 04 in DEST major state.

K10-4 RTI H detects a return from interrupt instruction.

K10-4 CLK N, Z, V H provides the clock signal for the N, Z, V bits of the STATUS word. Generated by specific instruction and time state combinations gated with K1-2 S CLK L.

K10-4 CLK C H provides the clock signal for the C bit of the STATUS word.

K10-4 GATE ST ← D H provides enabling signal to gate data from DATA PATHS 1 to the processor STATUS word when the STATUS word is the destination.

K10-4 CLK T H provides the clock signal to the T bit and priority bits of the STATUS word.

K10-4 GATE C H enables the carry data from the adder in the DATA PATHS to the data input of the C bit of the STATUS word.

K10-4 GATE V H enables data from the data paths to the data input of the V bit of the STATUS word for arithmetic overflow indication.

K10-4 GATE ROT/SHF H enables the exclusive oring of the N and C bits on Rotate operations and enables data to the C bit for Rotate/Shift instructions.

K10-4 CARRY INSTR H enables setting of the carry flip-flop for instructions requiring a +1 constant in the EXECUTE major state.

K10-4 A ← DEST/INSTR H used to enable K6-4 GATE A ← -B D15/0 H for instructions that require complemented bus data to be put into the A latch.

K10-4 GATE B/ISR1 H enables K6-4 ENABLE B ← RH in ISR1 of EXECUTE for BIS, JSR, CMP, and MOV instructions.

K10-4 GATE B/ISR0 H disables K6-4 ENABLE B ← RH in ISR0 of EXECUTE for MOV, JSR, and CLR instructions.

K10-4 TRAP INSTR H detects a Trap instruction.

K10-4 SVC ← F INSTR H enables ISR ← 0, 2/SERVICE for Trap instructions to take the processor from ISR1 of FETCH to SERVICE and ISR0.

K10-4 INSTR STPM ⟨04:02⟩ H provide the gating signals for appropriate vector addresses for servicing trap instructions (K12-3 STPM 04, STPM 03, STPM 02).

K10-4 ROT/SHF C DATA H provides the input data to the STATUS word C bit for Rotate/Shift instructions.

K10-4 C DATA H provides the data input to the STATUS word C bit.

K10-4 RIGHT DATA 15 L provides the input data to bit 15 of the DATA PATH for Rotate/Shift Right Instructions. For Rotate Right it is the C bit of STATUS; for Arithmetic Shift Right it is bit 15 of the adders.

K10-4 LEFT DATA 00 L provides input data to bit 00 of the DATA PATH for Rotate/Shift Left Instructions. For Rotate Left it is the C bit; for Arithmetic Shift Left it is a zero.

K10-4 RIGHT DATA 07 L provides the input data for bit 07 of the DATA PATH for Rotate/Shift instructions. For Rotate Right and Arithmetic Shift Right it is DATA PATH bit 08; for Rotate Right Byte it is the C bit and for Airthmetic Shift Right Byte it is DATA PATH bit 07.

K10-4 GATE ST ← D H = ISR8 * (SERVICE - RTI) + ST ADRS [ISR 15 * (U + B + R/S) * (ISR ← 0, 2/SERVICE) * - TEST * (- DEST MODE 0) + CSR0 * DEP]

K10-4 CLK N, Z, V H = SCLK * [- B EXEC (1) * - EXTRA * - ISR 3 * BIC * BIT + - (GATE ST ← D) + - (ISR ← 0, 2/SERVICE) * - (U + B + R/S) * - ISR15 * - BIC * - BIT * EXTRA]

K10-4 CLK CH = [CLK N, Z, V] · [BIS + BIT + BIC + MOV + DEC + INC]

K10-4 CLK T H = (GATE ST ← D) * S CLK

K10-4 GATE - C H = - (GATE ST ← D) * - ROT/SHF {COM + NEG + SUB + CMP + SBC * C (1)}

K10-4 GATE C H = - GATE - C * - GATE ROT/SHF

K10-4 GATE V H = - (GATE ST ← D) * - GATE ROT/SHF

K10-4 GATE ROT/SHF H = - (GATE ST ← D) * ROT/SHF

K10-4 CARRY INSTR H = B C (1) * ADC + SUB + NEG + INC + CMP

K10-4 A ← DEST/INSTR H = BIT + BIC + CMP + COM + NEG

K10-4 TEST H = BIT + TST + CMP

K10-4 GATE B/ISR1 H = BIS + JSR + CMP + MOV

K10-4 GATE B/ISR0 H = MOV + JSR + CLR

K10-4 TRAP INSTR H = HALT + RSVD INSTR + ILL INSTR + TRT + IOT + RESET + TRAP + EMT

K10-4 SVC ← F INSTR H = TRAP INSTR + WAIT + CC OP + BR INSTR + - BRANCH

K10-4 INSTR STPM 04 H = IOT + TRAP + EMT

K10-4 INSTR STPM 03 H = EMT + TRAP + TRT + RSVD INSTR

K10-4 INSTR STPM 02 H = TRAP + TRT + ILL INSTR

K10-4 ROT/SHF C DATA H = HIGH C DATA * ROT/SHF L + (- ADD 00) * ROT/SHF R

K10-4 C DATA H = (ADD 00 + - GATE ST ← D) * {CARRY DATA + - (GATE - C)} * { - (CARRY DATA) + - (GATE - C)} *{ - (GATE ROT/SHF) + (ROT/SHF C DATA)}

K10-4 RIGHT DATA 15 L = - HIGH C DATA * SHF RIGHT + C (1) * ROT RIGHT

K10-4 LEFT DATA 00 L = C (1) * ROT LEFT

K10-4 RIGHT DATA 07 L = - ACROSS DATA * (ROT WD + SHF) + C (1) * ROT BYTE

4.17 CODES DATA, M823

4.17.1 General

The CODES DATA modules provide data inputs to the condition codes portion of STATUS and certain data for the DATA PATHS during Rotate/Shift operations. Since this data reflects the results of a number of arithmetic or logic operations (word or byte), diverse input data is required. CODES DATA provides this by the selection of various output data.

Selection is effected by common enabling inputs on the various AND/NOR gates (GATE CC WORD, GATE CC BYTE, GATE V, GATE ROT/SHF, etc). This selection may exist for a complete instruction, with the output reflecting the various data passing through the DATA PATHS. It is necessary for the data to be proper only when it is used. For the Condition Codes, it is when they are clocked; for the DATA PATHS outputs, it is when they are stored. Loading of STATUS (addressed directly by bus address or indirectly within machine flow) is a special situation, in which the action of the Condition Codes portion of STATUS is suspended and the data directly loaded without change.

4.17.2 Output Signals

K11-2 GATE CC ← WORD H provides the gating signal for word operation within and external to CODES DATA. The signal is mutually exclusive to loading STATUS and byte operations.

K11-2 GATE CC ← BYTE H provides the gating signal for byte operation within and external to CODES DATA. The signal is mutually exclusive to loading STATUS and word operations.

K11-2 N DATA L provides information upon the sign bit of last operand to the data input of the N bit of the Condition Codes. A word operation gates the data line K8-2 K15 H to N; a byte operation gates the data line K7-2 D07 H to N; the loading of STATUS gates the data line K7-2 D03 H to N. If a byte or word sign bit is negative (set), then the N data input is asserted. The loading of STATUS requires the use of the 03 data line as this correlates to the bit position of N within the STATUS word.

K11-2 Z DATA H provides information upon a zero byte or word in the last operation, to the data input of the Z bit of the Condition Codes. A word operation gates K9-2 D15/0 ZERO L to Z; the loading of STATUS gates the data line K7-2 ADD02 L to Z. If a word or byte had all zero bits in its resultant data, then the Z data input would be asserted. The signal K7-2 ADD 02 L is used in the load situation to provide the proper data polarity.

K11-2 DATA L provides overflow information to the V bit of the Condition Codes. Both the calculation of the overflow situation and further gating as a function of instruction requires discussion.

Overflow occurs in an arithmetic operation when the signs of the input data are similar and the sign of the result data is dissimilar (i.e., if the A and B input signs were positive, and the result sign is negative, overflow has occurred). This comparison [(A DATA) * (B DATA) * -(N DATA) + -(A DATA) * -(B DATA) * (N DATA)] is made in gates E4 and E9, respectively. A word operation gates K8-2 A DATA 15 L and K8-2 B DATA 15 L, respectively (gates E8 and E3); a byte operation gates K7-2 A DATA 07 L and K7-2 B DATA 07 L, respectively (gates E8 and E3). Note that the comparison is further gated (in E5, E3, and E4) by the enabling signal K10-4 GATE V H. (The K10-4 GATE V H signal is active except when K10-4 GATE ST ← D H or K10-4 GATE ROT/SHF H are active). When K10-4 GATE ROT/SHF is active the data to the V bit of the Condition Codes is the "exclusive-or" of the sign and the carry of the result. This "exclusive-or" (C DATA * -N DATA + ROT/SHF C DATA * N DATA) is generated in gate E5 and enabled by K10-4 GATE ROT/SHF H.

The loading of STATUS gates the data line K7-2 D01 H with K10-4 GATE ST ← D H through E3.

K11-2 CARRY DATA L provides the carry data from K8-2 CARRY 15 L if a word operation occurred and from K7-2 CARRY 07 L if a byte operation occurred.

K11-2 HIGH C DATA L provides a portion of the data to the C bit of the Condition Codes. The selection of this output for input to the C bit is made on IR DECODE for a ROT/SHF L situation. Only the selection of byte (K7-2 ADD07 L) or word (K8-2 ADD15 L) information is done on the CODES DATA module. These signals are purposely selected from the DATA PATHS prior to the rotate/shift gating. This upper bit of data becomes the carry data on a rotate or shift to the left.

K11-2 ACROSS DATA L provides the data for bit 07 of the result during a rotate or shift operation to the right. A word operation gates K8-2 ADD08 L; a byte operation gates K7-2 ADD07 L. For the word the next highest bit is moved right; for the byte the information in its highest bit is regenerated; the data source for both is purposely selected prior to the rotate/shift gating in the DATA PATHS.

K11-2 GATE CC ← WORD H = -(GATE ST ← D) -(BYTE OP + SWAB)

K11-2 GATE CC ← BYTE H = -(GATE ST ← D) (BYTE OP + SWAB)

K11-2 N DATA L = (GATE CC ← WORD) D15 + (GATE CC ← BYTE) D07 + (GATE ST ← D) D03

K11-2 Z DATA H = (GATE CC ← WORD) (D15/0 ZERO) + (GATE CC ← BYTE) (D7/0 ZERO) + (GATE ST ← D) ADD02

K11-2 V DATA L = (GATE V) [(GATE CC ← WORD){(A DATA 15) (B DATA 15) -(N DATA) + -(A DATA 15) -(B DATA 15) (N DATA)}
+ (GATE CC ← BYTE){(A DATA 07) (B DATA 07) -(N DATA) + -(A DATA 07) -(B DATA 07) (N DATA)}]
+ (GATE ROT/SHF) [(C DATA -(N DATA) + -(ROT/SHF C DATA) (N DATA)] + (GATE ST ← D) D01

K11-2 CARRY DATA L = (GATE CC ← WORD) -(CARRY 15) + (GATE CC ← BYTE) -(CARRY 07)

K11-2 HIGH C DATA L = (GATE CC ← WORD) -ADD15 + (GATE CC ← BYTE) -ADD07

K11-2 ACROSS DATA = (GATE CC ← WORD) -ADD08 + (GATE CC ← BYTE) -ADD07

4.18 FLAG CNTL, M822

The FLAG CNTL module contains seven flag flip-flops that alter processor flow for: stack overflow (OVFLF); bus error (BERRF); halting (HALTF); trace operations (TRACF); trap instructions (TRAPF); and the bus INTR cycle (INTRF). Two additional flip-flops are concerned with transferring bus control to the console (CONS GRANT) and supplying the processor's BUS BBSY signal for bus mastership (BBSYF). Combinational logic is used in conjunction with the flip-flops to load, set, and clear; it is also used for the Special Trap Marker signals providing the trap vectors associated with certain flags.

The function of the flag flip-flops is to alter processor operation; they are set at specific times (either loaded or set) and cleared after use. Five of the flags (OVFLF, BERRF, TRACF, TRAPF AND INTRF) alter machine flow to process a trap sequence; one flag (HALTF) transfers bus control to the console. In the KA11, multiple flags may be set, and these are serviced in the descending order noted below:

Flag	STPM	Use
HALTF	Display R0	Halts the machine on HALT and RESET instructions and on double bus errors.
BERRF*	$(000004)_8$	Data time out (no response) and odd bus address error.
TRAPF*		Various instruction traps
	$(000004)_8$	Illegal instruction
	$(000010)_8$	Reserved instruction
	$(000014)_8$	TRT instruction
	$(000020)_8$	IOT instruction
	$(000030)_8$	EMT instruction
	$(000034)_8$	TRAP instruction
TRACF*	$(000014)_8$	T bit of STATUS set for trace operation.
OVFLF	$(000004)_8$	Stack overflow
PWRF	$(000024)_8$	Power-fail.

*A common clearing signal after service precludes multiple trap sequences.

Other information in the table relates the trap vector (STPM) and its use to the flag flip-flop. The power fail trap is noted because its trap vector is generated and gated on FLAG CNTL; the power up (PUPF) and power down (PDNF) flip-flops, which combine for the single PWRF flag, are located on PRIORITY.

The bus interrupt flag (INTRF) is not noted in the order of service; processor operation makes this unnecessary. Flags are serviced in the processor's major state SERVICE. In SERVICE, the flow diverges into two major paths: one path returns to FETCH unless a Bus Request (BR) or Non-Processor Request (NPR) is outstanding, no trap flags must be raised; the other path services trap flags in order and returns to FETCH, no Bus Requests (BR's) are honored (NPR's are honored at the end of each bus cycle within the trap sequence). Since INTRF is set as the result of a Bus Request and subsequent bus INTR cycle, INTRF is not set while traps are serviced. It is possible for a bus error to set BERRF during the INTR trap sequence; this error trap is serviced immediately. Another bus error would enable HALTF, which would halt the machine. The halting on double bus error is also true for regular trap servicing.

K12-2 OVFLF (1) H provides the flag output for stack overflow. It is set when the address contents of the processor stack (R6) is below $(000400)_8$ and information is "pushed" onto the stack. The processor stack (R6) can be implicitly addressed in machine flow (JSR or trap service) or explicitly addressed in an auto decrement mode. The OVFLF flag is of low priority and is serviced after the completion of the instruction and after other traps (TRACF, TRAPF and BERRF). The OVFLF flip-flop is cleared after service (K12-2 SVC CLR OVFLF H).

The OVFLF signal is used on PRIORITY to inhibit the reset of the PDNF flip-flop; the signal is used on FLAG CNTL to gate the appropriate STPM address $(000004)_8$ and alter machine flow through K12-3 TRAPS H.

K12-2 SVC CLR OVFLF H provides an inhibit input to the clear signal for the PDNF flip-flop on PRIORITY. This is used with the OVFLF signal to insure the order of service of flags; a flag will not be cleared while higher order flags are being serviced. The flag is cleared when it is the highest order flag.

K12-2 BERRF (0) H provides the flag output for two bus errors: the use of an illegal odd bus address (ODD ADRS ERR); and the non-response of a bus address (TIME OUT). All odd bus addresses (bit 00 of BAR is set) are illegal unless upper byte data is transferred as an instruction operand. Non-response of a peripheral SSYN signal to a MSYN signal indicates a non-existent or defective peripheral. Both of the above errors result in a trap sequence to location $(000004)_8$, unless the bus cycle occurred during EXAM or DEP. The trap sequence is second in order of service, with a bus error during the trap sequence resulting in a machine halt (HALTF).

The input signal to BERRF initially alters the processor flow directly to SERVICE. After the trap (K12-3 TRAPS H) sequence the BERRF is of high priority and the entry to SERVICE is immediate. The BERRF signal, itself, is used: to inhibit the clearing of lower order flags (K12-2 SVC CLR OVFLF H); to provide the proper STPM address; and to control machine flow (K12-3 TRAPS H).

K12-2 HALTF (1) H provides the flag output for halting the KA11 processor upon: a HALT or RESET instruction; or a double bus error. The HALTF flip-flop is set during FETCH if either a HALT or RESET instruction is decoded. A new bus error during the trap service for a bus error (BERRF) results in a double bus error and the HALTF flip-flop is loaded to "one". The HALTF signal is used: to transfer bus control to the console (K12-3 CONS GRANT (1) H and K2-2 PROC RELEASE H); to inhibit the continuation of the usual trap sequence (K2-3 ISR1 L); to inhibit the gating of any STPM address (K6-4 GATE B←STPM H); and to enable the display of R0 on the console (K6-4 ENABLE B ← R L and the address selection of R0 by asserting no address).

DIRECT SET ON OVFLF = B EXEC (1) * ISR0 * JSR + (SO ± DE) * ISR1 * AM (4+5) * REG 6 + SERVICE * ISR (3+7) * D 15/8 ZERO * CLK BAR

DIRECT CLR ON OVFLF = SERVICE * ISR12 * - INTRF * SVC CLR OVLF + INIT

K12-2 **SVC CLR OVLF H** = R/W3 * TRACF (0) * TRAPF (0) * BERRF (0)

K12-3 STPM ⟨04:02⟩ H provide the three active bits of address for Special TraP Markers used in processor initiated traps. The address is dependent upon the highest order trap flag – this flag enables its address and disables the others. Since the highest order trap flag is cleared after service and the next highest flag is then service, the addresses are properly sequenced. The STPM signals are enabled as data inputs to DATA PATHS 1 during SERVICE * ISR0. The INTR bus cycle effects the same input for address vectors that will be utilized in an INTRF trap sequence.

K12-3 TRACF (1) H provides the flag output for a trace operation. It is set during FETCH if the T bit of Condition Codes (STATUS) is set. Since the T bit can be altered in any trap sequence (STATUS is changed), the TRACF is serviced only if it is the first (or highest order) trap being serviced. A common clearing signal (K12-3 CLR FLAGS L) for TRACF, TRAPF and BERRF flip-flops clears all if any one is serviced. (The service of HALTF does not require a trap sequence, merely the transfer of control to the console; if console control is relinquished by CONT, the outstanding trap flags will be sequenced.)

The TRACF signal is used: to control machine flow (K12-3 TRAPS H); and to provide the proper STPM address.

K12-3 TRAPS L provides a single signal indicating any set trap flag; it is used to alter processor flow. This signal, K12-3 TRAPS L, is used: to gate the trap address to DATA PATHS 1 (K6-4 GATE B ← STPM H); and to inhibit the release of bus control (K2-2 PROC RELEASE H) during SERVICE * ISR0.

K12-3 TRAPS H provides a single signal indicating any set trap flag; it is used to alter processor flow. This signal, K1-23 TRAPS H, is used: to inhibit the entry into FETCH state from SERVICE * ISR0 (K15-2 FETCH ← SVC H); and to prohibit the falling through SERVICE on the entry signal, ISR ← 0, 2/SERVICE H (K1-4 FETCH ← 1 H).

K12-3 SERV 0 L provides a signal to identify that portion of SERVICE * ISR0 in which the processor is in control. The signal is used to enable the gating of STPM addresses K6-4 GATE B ← STPM H).

K12-3 NPR ENTRY H enables the service of Non–Processor Requests (NPR's); such service occurs at the end of bus cycles (DATIP excluded) or during SERVICE * ISR0 (K2-2 PROC RELEASE H).

K12-3 INTRF (1) H provides a flag output for an interrupt operation. Bus control is acquired by a peripheral using a Bus Request (BR); after control the peripheral can return control and a trap address by a bus INTR cycle. The INTRF flip-flop is set by this sequence and cleared after trap service. The INTRF signals are used: to inhibit the return to FETCH (K15-2 FETCH ← SVC H); to alter machine flow (K2-3 ISR ← 1 L); and to inhibit the granting of bus control (K2-3 GRANT L and the inhibit of PTR clocking). The INTRF signal also ends the cycling of the WAIT instruction.

K12-3 BBSYF (1) H provides for processor bus control; when this signal is asserted the processor has bus mastership. The flip-flop is loaded to one upon peripheral bus release and then maintains itself until cleared. Direct clear comes with release of processor control to either the bus or the console after a completed bus cycle. The BBSYF are used: to assert processor bus mastership on the Unibus (BUS BBSY L); to identify control within the processor (K13-4 PROC CNTL H); to enable bus gating (K9-2 GATE BUS ← D H and K9-2 GATE BUS ← ST H); and to disable processor response to BUS INTR L when BBSYF is asserted (print K13-3).

K12-3 CONS GRANT (1) H is utilized as a pulser in transferring bus control from the processor to the console. The flip-flop is loaded to one, sets the CONSF flip-flop (K13-4 CONSF (0) H) which then clears CONS GRANT. The CONS GRANT signal is also used to clear the HALTF which may have initiated the transfer; it also initializes the EXAMF and DEPF of BUS & CONSOLE CNTL.

K12-3 STPM 04 H = TRAPF (1) * - OVFLF (1) * TRACF (1) * INSTR STPM 04 + BERRF (0) PWRF * - TRAPF (1) * - TRACF (1) + INSTR STPM 04 * TRAPF (1)

K12-3 STPM 03 H = TRAPF (1) * TRACF (1) * - OVFLF (1) * INSTR STPM 03 + BERRF (0) TRAPF * INSTR STPM 03 + TRACF (1) * - TRAPF (1)

K12-3 STPM 02 H = TRAPF (1) * INSTR STPM 02 + TRACF (1) + (OVFLF (1) + PWRF) * - TRACF (1) * - TRAPF (1) + - BERRF (0)

K12-3 TRAPS H = TIME OUT (1) + ODD ADRS ERR + PWRF + TRAPF (1) + TRACF (1) + - OVFLF (0) + - HALTF (0) + - BERRF (0)

K12-3 SERV 0 L = SERVICE * ISR0 * - CONSF (1)

K12-3 NPR ENTRY H = NPR ENABLE * - (BSR15) * - DATIP * - (TIME OUT)

DIRECT CLEAR on BBSYF = (PROC RELEASE) * (CLK RUN (0)) (BSR14 + SVC * ISR0) + (DATA CLR) + CONSF (1) + INIT

DATA INPUT on CONS GRANT = (SERVICE * ISR0) * HALTF (1) + CBRF * - TRAPS + (NPR ENTRY) * CNPRF (1)

4.19 BUS & CONSOLE CNTL

The module provides the logic control necessary to bus and console operation. Bus control involves: initializing timing for power up and power down situations; time-out for data and bus control transfers; and processor bus control (BUS C ⟨1:0⟩) and bus timing (BUS MSYN and BUS SSYN). Console control involves: the enabling of console switches; separate console timing and console flag flip-flops for bus control and automatic incrementation on EXAM and DEP. The console operations are further described within the KY11-A manual; both flow charts and waveforms are presented and are directly related to the KA11 and the BUS & CONSOLE CNTL module.

K13-2 INIT L initializes flip-flops (singularly or in registers) throughout the processor. The signal is active in power-up and power-down situations upon a BUS DC LO L positive or negative transition. (BUS DC LO L is a signal activated by the power supply when d.c. voltages are low). Edge sensitivity to the signal is effected by the PWR DOWN and PWR UP one-shots (E35 and E34), each producing a 5 millisecond output signal for K13-2 INIT L. The console START switch produces this signal as long as it is depressed. Both signal sources utilize the discrete transistor switch of this module to obtain sufficient drive.

K13-2 INIT H provides the inverted signal for K13-2 INIT H with standard load capabilities.

K13-2 PWR UP H is activated only on the power-up transition of BUS DC LO L. The PWR UP one-shot (E34) produces a 5 millisecond pulse (also used in K13-2 INIT L) to set the PUPF flip-flop (K3-3 PUPF (0) H) for the Restart portion of Power/Fail Restart.

BUS INIT L provides an initializing signal from the processor to the Unibus. It combines the sources noted in K13-2 INIT L and the output of the RESET one-shot (E47). The RESET instruction activates the RESET one-shot (E47) which provides a 20 millisecond bus initializing signal. The RESET output also activates the RESTART one-shot (E43).

K13-2 P RESTART L provides a restart pulse upon power-up or RESET instruction. The 100 nanosecond pulse occurs 70 milliseconds after the last activating input to the RESET one-shot, (E43). The power-up activation occurs through the 100 nanoseconds pulser (E38, E52) on the positive transition of BUS AC LO L. This is gated by the console HALT mode (KY-3 HALT L) or a power-down situation within the last 5 milliseconds (PWR DOWN one-shot). Multiple power-up activations within 70 milliseconds delays the K13-2 P RESTART L until 70 milliseconds after the last one. The RESET one-shot (E47) upon activation, activates RESTART to restart the processor after 70 milliseconds. This restart is necessary as the RESET instruction sets the HALTF (K12-2 HALTF (1) H) which halts the processor and transfers control to the console; restart occurs through the signal K15-2 GATED P RESTART L.

K13-2 TPI 1 provides Test Point for the RESTART one-shot (E43).

K13-2 NO SACK (0) H provides a time-out flag for the non-acknowledgement of a bus control grant. An input flip-flop (TIME SACK, E48) is clocked to one by the K2-3 GRANT H signal; this flip-flop initiates a 10 microsecond timing one-shot (E51) and provides a delayed (δ = 150 nanoseconds) data input to the flag flip-flop (NO SACK). If a Select ACKnowledge occurs, the K2-3 SACK H signal clears the input flip-flop and removes the data input to the flag before clocking the one-shot; the flag flip-flop is also held off by the clearing signal. If no K2-3 B SACK H signal occurs, the flag flip-flop is clocked to one. The signal, K13-2 NO SACK (0) H is used for continuation of processor operation without trapping, by clearing the PTR flags (K2-3).

K13-2 TIME OUT (1) H provides a time-out flag for the non-acknowledgement of a bus data transfer. Operation is similar to the time out logic of signal K13-2 NO SACK (0) H: The input flip-flop (TIME DATA, E48) is clocked by K13-3 MSYN CLK H; the flag flip-flop (TIME OUT, E46) is clocked after a selection of delays; and the clearing signal is a composite of bus (K13-3B SSYN H), console and machine functions. The selection of delays for time-out is effected by removing one or two of the capacitors C11, C67 and C68 giving a range of 25 microseconds to 5 microseconds increments. The signal is used: to inhibit machine state alteration (K2-2 SHIFT ISR H); and to provide for the trap (K12-2 BERRF (0) H and ISR 0,2/SERVICE H). The complement flag flip-flop output is also used.

K13-2 TIME OUT (0) H provides the complement output for the signal K13-2 TIME OUT (1) H and is used: to provide for the trap sequence (K12-3 TRAPS H) and to alter machine flow (K2-2 BUS OUT DONE H and K2-3 BSR 15 H).

K13-2 P TIME OUT L provides a 100 nanosecond pulse upon the setting of the data transfer time-out flag (TIME OUT). The signal is used: to restart the processor clock (K1-2 CLK P RESTART L); and to imitate a data completion (K6-2 P DATA RESTART L).

K13-2 GATE BUS ← SR H enables the Switch Register to the Unibus on the console and inhibits the gating of Data Paths Outputs to the Bus (K9-2 GATE BUS ← D H).

K13-2 BC ⟨1:0⟩ (1) H provides test point for the bus control flip-flops that determine processor bus cycles. These flip-flops are clocked to the appropriate bus control states by the data inputs: K13-2 DATO ← 1 L; K13-2 DATO B ← 1 L; K13-2 DATIP ← 1 L; and K13-2 DATI ← 1 L. The flip-flops are cleared by K13-2 BUS INDICATOR which is asserted on peripheral control, other than console.

BUS C ⟨1:0⟩ L provides the Unibus signals reflecting the processor's bus control flip-flops (BC ⟨1:0⟩).

K13-2 DATIP L provides a signal indicating a DATa In Pause bus cycle. This bus cycle renders a peripheral liable to data damage; Non-Processor Request (NPR's) are not allowed after this cycle (K12-3 NPR ENTRY H).

K13-2 DATO ENTRY H provides the entry signal for the bus data transfer of DATO. The signal is used: to alter machine flow (K2-3 BSR ← 12 L); and to effect console operation (K6-5 CLR ON BSR H and K6-4 ENABLE B ← R L).

K13-2 DATO ENTRY L provides the complement output of signal K13-2 DATO ENTRY H. It is used: to disable a load to DATA WAIT on BSR7; and to disable a K15-2 NPR ENABLE L during BSR(7+14).

GATE BUS ← SR H
K13-2 DATO ← 1 L
K13-2 DATOB ← 1 L
K13-2 DATIP ← 1 L
K13-2 DATI ← 1 L
K13-2 DATO ENTRY L

K13-2 GATE BUS ← SR H = SR ADRS * PROC CNTL * SSYN (1) H + LOAD ADRS + DEP * [CSR2 + BSR (15+14+12+8)]

K13-2 DATO ← 1 L = DATO# ENTRY * - (DATOB ← 1) + DATO ENTRY * BSR1

K13-2 DATOB ← 1 L = BYTE OP * DATO # ENTRY

K13-2 DATIP ← 1 L = B DEST (1) * BSR1 * ADRS DONE * - TEST

K13-2 DATI ← 1 L = BSR1 * - (DATIP ← 1) * - DATO ENTRY

K13-2 DATO ENTRY L = BSR1 * CLK OFF (0) + EXEC * ISR0 * JSR + SERVICE * (ISR 3+7)

BUS MSYN L provides the Unibus signal from the processor MSYN flip-flop (K13-3 MSYN (1) H).

K13-2 MSYN (1) H provides a test point for the processor MSYN flip-flop; the signal is also used to drive the bus interface gate to produce BUS SSYN L. Activation of this flip-flop initiates a processor data transfer on the Unibus. It is enabled to one by the gated clock (K1-2 S CLK L) during the appropriate Bus Shift Register (BSR) machine state. For a data out transfer, this is BSP 12; for a data in transfer, this is BSR3. Data in transfers are aborted for last JMP or JSR address operation (K13-3 MSYN ← 1 H) with the MSYN flip-flop not being set. The signal K6-2 DATA CLR H clears the flip-flop when set.

K13-3 CLK BAR H provides the clocking signal to the Bus Address Register (BAR) for processor bus cycles and console operation. It is also used as the timing portion of the OVFLF set signal (K12-2 OVFLF (1) H).

BUS SSYN L provides the Unibus signal from the processor SSYN flip-flop (K13-3 SSYN (1) H).

K13-3 SSYN (1) H provides the processor bus response for an INTR cycle or processor data transfer from Switch Register (SR) or STATUS (ST). These situations enable the data input to the SSYN flip-flop. The 100 nanosecond pulser on the clock input provides deskewing for the bus timing (MSYN or INTR) by clocking with the lagging pulse edge (K13-3 P (MSYN ↓ + INTR ↓) L with the arrow indicating the actual bus transition for assertion). The SSYN flip-flop is cleared after bus master's removal of timing signals (K13-3P (MSYN ↑ + INTR ↑) L. The signal K13-3 SSYN (1) H is used to: drive the bus interface gate (BUS SSYN L) and enable processor operands to the bus (K13-2 GATE BUS ← SR H and K9-2 GATE BUS ← ST H).

K13-3 MSYN ← 1 H = BSR12 + BSR3 * BC 1(0) * -{ODD ADRS ERR + [(SO+DE) * (JMP+JSR) * ADRS DONE]}

K13-3 CLK BAR H = CSR3 + BSR3 * R/W 1 * - (EXAM + DEP)

K13-3 GATED B INTR L = SERVICE * ISR0 * BUS INTR * BBSYF (0)

K13-3 B INTR H = BUS INTR * BBSYR (0)

K13-4 STARTF (1) H provides a test point for the START F flip-flop. The START sequence is a unique console function; it must perform a console sequence and begin processor program operation. The STARTF flip-flop provides a temporary flag enabling the console sequence (CSR timing); it is immediately reset upon completion to allow normal processor clocking (SCLK Timing).

K13-4 CONS BR H provides a CONSole Bus Request to PRIORITY when the appropriate switches enable the HALT and S/INSTR modes.

K13-4 CONS NPR H provides a CONSole Non-Processor Request to PRIORITY when the appropriate switches enable the HALT and S/CYCLE modes.

K13-4 CSR ⟨3:0⟩ H provides Console Shift Register (CSR) timing states throughout the processor. A fixed sequence of timing states are provided by decoding the shift register CSR ⟨1:0⟩; these states and additional console timing are shown in Figure 4-8.

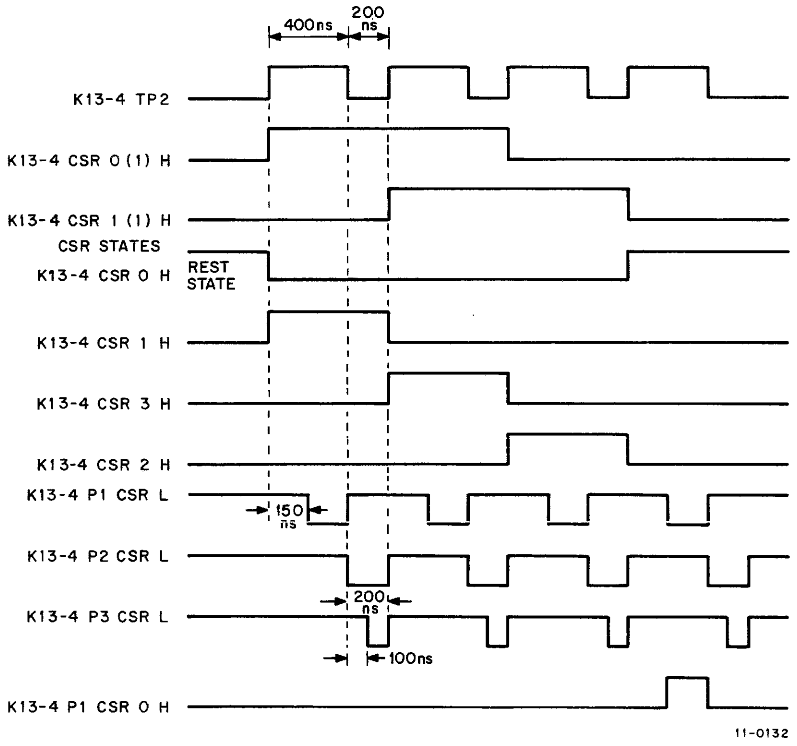


Figure 4-8 Console CSR Timing Diagram

The basic timing loop consists of the 400 nanosecond one-shot (E25) with output K13-4 TP2 and 200 nanosecond pulser (E20, pin 3) with output K13-4 P2 CSR L. The one-shot is activated by a gated console switch and clocks the CSR from its rest states of 00 to 01; after time-out (400 nanoseconds) the pulser is activated with its lagging edge (200 nanoseconds) refiring the one-shot. This recycle continues through the CSR flip-flop states of 11 and 10; until the pulser (E20, pin 2) is inhibited by the CSR rest state (K13-4 CSR 0L).

The CSR flip-flops are clocked by the one-shot from state 00 through 01, 11, 10, to 00. The non-transient decoding of these states results in K13-4 (CSR ⟨3:0⟩ H. In addition to the CSR states and associated with each are the pulses K13-4 P ⟨1:3⟩ CSR L. The pulse K13-4 P1 CSR L is delayed 150 nanoseconds (E20, pin 8) from one-shot activation and terminates with its time-out; a 250 nanosecond pulse is produced. Note that the three pulses occur within the first three CSR states (CSR1, CSR3, CSR2) with only the K13-4 P1 CSR L occurring in the end rest state (CSR0).

The CSR states and pulses are used within the processor to effect console operation. The details of this gating are noted in the flow diagrams of the KA11.

K13-4 CSR0 (1) H provides the console enabling equivalent to the CSR3 or CSR2 states, (K-4-3 GATE RA ← BAR H).

K13-4 P1 CSR0 H provides the single pulse (P1) in the end state (CSR0) of the console sequence.

K13-4 CSR ⟨0,2:3⟩ L provides the inverse signal of those noted under K13-4 CSR ⟨3:0⟩ H.

K13-4 P ⟨3:1⟩ CSR H provides the pulse outputs noted under K13-4 CSR ⟨3:0⟩ H.

K13-4 P ⟨3:1⟩ CSR L provides the inverse pulse signal of those noted under K13-4 P ⟨3:1⟩ CSR H.

K13-4 TP2 provides a test point for the one-shot (E25) basic to console timing and noted under K13-4 CSR ⟨3:0⟩ H.

K13-4 EXAM H
K13-4 DEP H
K13-4 LOAD ADRS H
K13-4 START H
K13-4 LOAD ADRS L

provides a gated output signal for the appropriate switch activated. Gating is provided by (SERVICE * ISR0) and CONSF (1) to insure activation only when console functions are allowed.

K13-4 INCF L provides a flag signal for a second EXAM or DEP which requires word incrementation of the bus address before use. The EXAMF and DEPF flip-flops are loaded according to the last console operation and cleared if the operation is contrary. Coincidence between the last operation and the present activate the signal.

K13-4 P CONSF (0) L provides a 100 nanosecond pulser (E23, pin 3) output when the CONSF is cleared. The signal is used to clear the CNPRF flip-flop on PRIORITY.

BUS BBSY L provides the console assertion of Unibus mastership from CONSF flip-flop.

K13-4 CONSF (1) H provides the major console flag that indicates console control. Control is passed to the console by the set input involving K12-3 CONS GRANT (1) H. Release of bus control is effected on the switch functions of START or CONT to the clock input of K15-2 GATED P RESTART to the clear input on power-up or RESET instruction.

K13-4 BUS INDICATOR H provides a peripheral bus mastership signal that requires the assertion of bus mastership (K2-3 B BBSY H) but not by the console (K13-4 CONSF (0) H) or processor (K12-3 B BSYF (0) L). The signal is used for the BUS console indicator.

K13-4 ADDRESS 0 H provides the console display signal for bit 0 of ADDRESS in major machine states of SOURCE or DEST.

K13-4 ADDRESS 1 H provides the console display signal for bit 1 of ADDRESS cycle in major machine state of SOURCE or DEST.

K13-4 ADDRESS 0 H = (SO+DE) * (ISR1+ISR7)

K13-4 ADDRESS 1 H = (SO+DE) * (ISR7+ISR3)

4.20 PWR FAIL & CNTL, M825

Timing logic for the power fail sequence and some general control gating are located on the M825 module. (Refer also to console flow diagrams in the *KY11-A Programmer's Console Manual,* DEC-11-HR6B-D.)

4.20.1 Power Fail

Logic synchronizes the AC LO request for power fail and manipulates both the AC LO and DC LO signals during the power fail sequence. The output signals K15-2 CLK PDN F H, BUS AC LO L, and BUS DC LO L and associated logic are discussed relative to this sequence.

In a normal power failure, the power supply activates the AC LO and DC LO signals (active low) as shown in Figure 4-9.

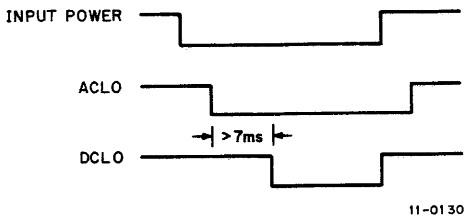


Figure 4-9 Power Fail Timing Diagram

The power down requirements for the power supply are: the AC LO signal must occur 7 milliseconds before the DC LO signal; a DC LO signal cannot occur without a preceding AC LO. The power-up requirements are merely that the DC LO signal ceases before the AC LO signal. A momentary power interruption can cause the power supply to temporarily activate the AC LO signal without activating the DC LO signal; processor logic corrects this and sequences both AC LO and DC LO. Once a power fail sequence is begun, it is completed.

The loss of input power activates the input signal K13-2 BAC LO L which fires the DELAY DC LO and AC LO one-shots (E4 and E3) and attempts to set the ACLOF flip-flop (E13). The AC LO one-shot pulls the BUS AC LO L signal low for 15 milliseconds, effectively extending a short AC LO signal to a workable duration.

The DELAY DC LO one-shot after its 7 milliseconds delay provides DC LO signal through the 1 microsecond pulser circuit. A BUS DC LO L signal, therefore, is generated even if the power supply does not provide one. (BUS INIT L is activated on the BUS & CONSOLE CNTL module on each edge of the DC LO signal.)

The AC LO Flag (ACLOF flip-flop) is set by the original AC LO transition if the DELAY PWR DOWN one-shot (E5) is inactive; it is inactive if the machine has not restarted (K15-2 GATED P RESTART L) within the last 3 milliseconds. This delay provides a fixed operating time after a power up sequence. If a AC LO occurs during it, the end of the delay sets the AC LO flip-flop. The pulser circuit responds only to signal transitions with the set pulse width (τ = 300 nanoseconds) sufficient to override any clock signal loading the flag to 0. (The resistors R10 and R14 provide a clamp load for the open collector drive of E1.)

The function of the ACLOF flip-flop is to asynchronously accept the AC LO transition and provide for a synchronous transfer to the Power DowN Flag (PDNF flip-flop) on PRIORITY. This latter transfer requires K1-2 S CLKL for synchronizing and results in the signal K15-2 CLK PDNF H; the transfer also clears (loads to 0) the ACLOF flip-flop through a discrete circuit inverter. The ACLOF flip-flop is cleared on power supply K13-2 INIT L and on restart by the DELAY PWR DOWN one-shot. In addition to clearing the ACLOF flip-flop, DELAY PWR DOWN gates its output.

4.20.2 General Control Gating

The combinational logic on the PWR FAIL & CNTL module provides some general control signals used throughout the processor. They are discussed below:

K15-2 WAIT CLR H clears the DATA WAIT and LATCH flip-flops on DATA PATHS CNTL upon power up and entry into a WAIT loop. The latches must be cleared during the WAIT loop for possible interrupt vectors.

K15-2 FETCH ← SVC H provides state control with an entry signal to the FETCH major state (K1-4 FETCH ← 1 H) from SERVICE after servicing a Bus Request (without an INTR) or a Non-Processor Request. Other signals are also used to enter FETCH.

K15-2 NPR ENABLE L enables the release of processor control (K2-2 PROC RELEASE H) for a Non-Processor Request at specific places in machine flow. This enable signal is further qualified to produce K12-3 NPR ENTY H, but essentially enables processor release after a bus operation and during the WAIT loop.

K15-2 SERV 0 H provides for a determination of machine state within SERVICE * ISR0. If K15-2 SERV 0 H is asserted, the processor and not the console is in control.

K15-2 GATED P RESTART L clears the CONSole Flag (K13-4 CONSF (1) H and BBSY L) which returns bus control to the processor (K2-3 D PERIF RELEASE L) and restarts the processor clock (K1-2 P CLK RESTART L). This 100 ns pulse occurs 70 milliseconds after the RESTART instruction or power up (positive transition) of BUS AC LO L). The delay, an integrating one-shot, is on BUS & CONSOLE CNTL and is reflected in the signal K13-2 P RESTART L which is further gated by KY-3 HALT H and K13-2 B AC LO L. KY-3 HALT H prohibits restart if the ENABLE/HALT switch is in the HALT mode, control remains with the Console. K13-2 B AC LO L prohibits restart, as does the integrating one-shot, until the input power has stabilized for more than 70 milliseconds.

K15-2 WAIT CLR H = FETCH * ISR1 * WAIT * R/W2 + INIT

K15-2 FETCH ← SVC H = ISR0 * PERIF RELEASE * INTR F (0) * - WAIT * - TRAPS

K15-2 NPR ENABLE L = SERVICE * ISR0 * - TRAPS * INTR F (0) * - CONSF (1) + BSR8 + BSR(7+14) * - (DATO ENTRY)

K15-2 SERV 0 H = SERVICE * ISR0 * - CONSF (1)

K15-2 GATED P RESTART L = PRESTART * - (BACLO) * - HALT

CHAPTER 5
MAINTENANCE

5.1 SCOPE

The maintenance philosophy of the KA11 processor consists of presenting information that enables the user to understand how the system should function during normal operation. The user can then use this information when analyzing trouble symptoms to determine necessary corrective action. It is beyond the scope of this manual to provide detailed trouble-shooting information. However, this section does provide general maintenance information, such as required equipment, module layout, interconnection for multiple box systems, and power control. In addition, this section conatins a procedure for adjusting the processor clock, special installation and removal instructions, and maintenance tips. The appropriate engineering drawings are listed in Table 5-1.

Table 5-1
Engineering Drawings

Drawing No.	Title	Sheets
A-ML-KA11-0	Central Processor master list	2
C-DI-KA11-0-1	Drawing index	1
D-AD-7006537-00	Wired assembly	2
D-BD-KA11-0-BD	KA11 block diagram	1
D-TD-KA11-0-WF	Waveforms	1
D-MU-KA11-0-MU	Module Utilization	1
D-WL-KA11-0-WL	Wire list	1

5.2 TEST EQUIPMENT AND TOOLS

Table 5-2 lists various equipment, devices, and tools which may be required to perform tests and maintenance on the KA11 Processor.

5.3 INSTALLATION OF ECOs

The procedures for installing engineering change orders (ECOs) on modules used in the PDP-11 system are contained in the Module Rework Standard, DEC SP 7605845-0-0, dated 7 August, 1970.

5.4 MODULE IDENTIFICATION AND LAYOUT

The modules associated with the PDP-11 system unit are designated by an alpha-numeric scheme as indicated on the various schematic prints. This scheme consists of a 4-character designation but, at times, may consist of only a 3-character designation. The 3-character scheme provides the same information as the 4-character method by inferring the value of the fourth character. This latter situation occurs on a system unit that can be housed at any system unit position within the overall PDP-11 system.

Table 5-2
Test Equipment and Tools

	Item	Type
Test Equipment	Oscilloscope	Tektronix Model 453 (or equivalent)
	Volt-Ohmmeter	Triplett Model 630 (or equivalent)
Devices	Extender Board	One W984A double extender board
		Two single extender boards (a W984A board cut in half)
	Maintenance Module Set	One W-130 One W-131
	IC Test Clip	
	Pin probe tip	Tektronix #30
	Pointed tip solder iron	Rated at less than 40 watts
	Package of Solderwick	This is used for removal of solder on printed circuit boards
Tools	Screw drivers Allen wrench set Needle nose pliers Wire strippers	

A typical system layout of a PDP-11/20 with six system units installed is shown in Figure 5-1. This figure shows the units as viewed from the back panel pin side. The module and pin identification is as follows:

a. Assume a designation of F4A1. The first character (F) designates column F (the lower column running from front to back).

b. The second character (4) designates row 4. (Sometimes the character is written as 04.) The rows run from side to side. Row 4 is the fourth from the front. Therefore, F4 designates that group of pins associated with column F and row 4 (this group is identified as I in the figure). Row numbers 5, 10, 15, 20, and 25 are taken into account for the numbering sequence but do not contain modules or pins. They are the spacing between blocks.

c. The final two characters identify a specific pin within the F4 block. This identification method conforms to the standard DEC pin identification method used for double-sided modules. This method is shown on Figure 5-1.

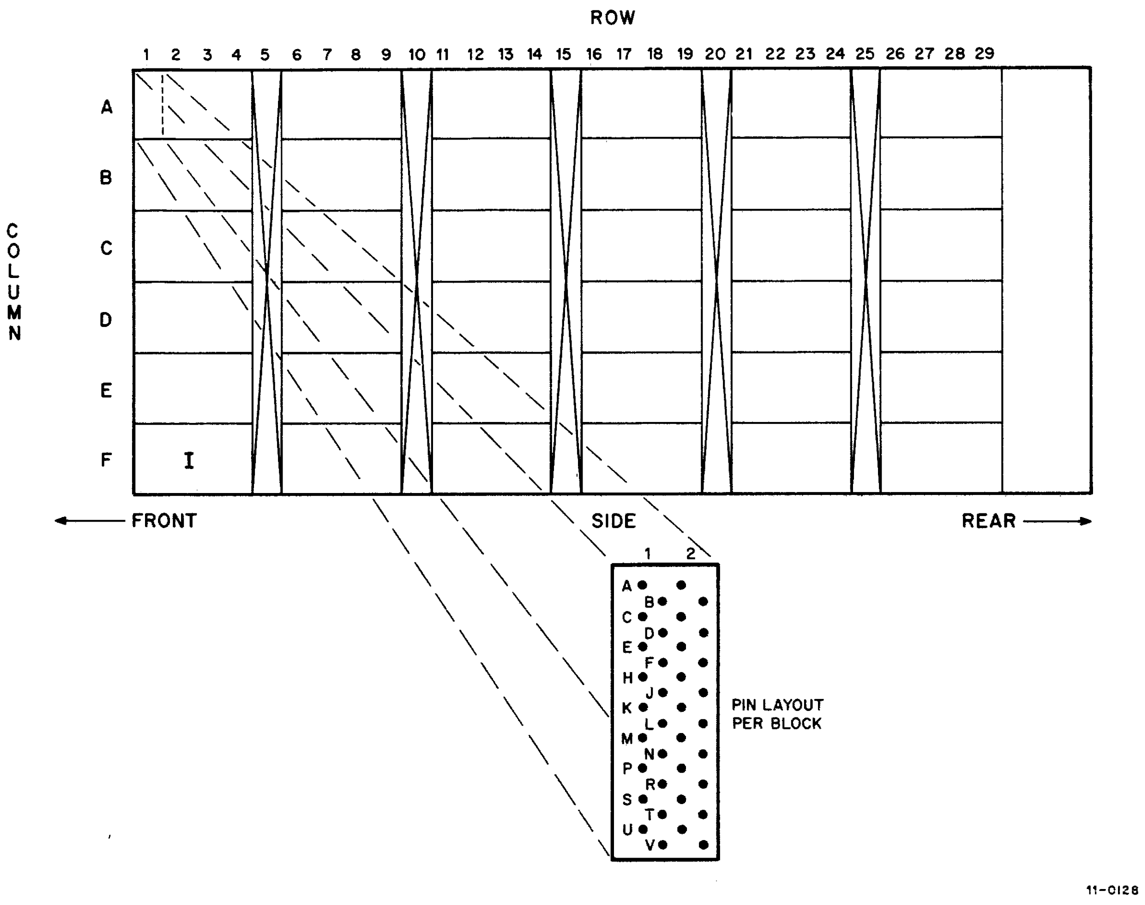


Figure 5-1 Typical System Layout

All KA11 Processor prints use this 4-character identification method. The MM11-E core memory, the HSR/HSP PC-11 reader, and the KL11 Teletype control all use the 3-character method. The prime reason for doing this is to make all prints equally applicable regardless of which slot is used for system unit installation. In this method, the column is given the pin identification. No matter which row the modules are inserted into (provided the wiring is for that device), the modules have the same column and pin identification.

5.5 MODULE COMPONENT IDENTIFICATION

The individual components located on any module of the PDP-11 system are identified, along with physical location, on the first sheet of the specific module print set.

5.6 UNIBUS CONNECTIONS

Instructions for connecting to the Unibus or adding additional devices to an installation are covered in the *Unibus Interface Manual, DEC-11-HIAA-D.*

5.7 MULTIPLE BOX SYSTEMS

Whenever BA11 extension mounting boxes are added to an existing basic mounting box configuration, it is necessary to interconnect (by means of the Unibus) the AC LO and DC LO functions of each additional power supply. This is required to ensure that power failure in any box causes proper processor response. This requirement is also necessary for any user manufactured and/or interfaced device which needs processor response to power failure and does not receive its power from a BA11 box.

5.8 POWER CONTROL

Primary power for a basic PDP-11/20 System is 120 VAC, 50/60 Hz (H720-A power supply). Variations in these values are possible by adhering to the wiring requirements shown on the H720 power supply schematics and assembly drawings which are part of the system print set.

The power receptacle at the rear of the H720 power supply always furnishes the same power as that supplied on the input line. The power at this receptacle is directly controlled by the POWER/OFF/PANEL LOCK switch on the programmer's console. The internal fans for individual BA11 mounting boxes are always across a 120 VAC source as long as wiring requirements for the H720 power supply are followed.

Those systems using a free-standing base cabinet (H960) have additional fans mounted in them. The number of fans and their power requirements are dependent on the system procurement specifications. In 120 VAC systems, cabinet fans are plugged into the H720 receptacle. Some 240 VAC systems may have two 120 VAC fans wired in series. Another possible 240 VAC configuration may use an H722 step-down transformer. In this case, the fan(s) are wired to the 120 VAC side of the transformer along with other 120 VAC options.

NOTE

If any change in input line power from its original configuration is anticipated, the procurement specifications must be considered.

The control of the entire system (including options) can be attained by using the POWER/OFF/PANEL LOCK switch on the programmer's console. This is accomplished by parallel connecting all additional mounting boxes, options, and peripherals from the receptacle to the main mounting box. Most DEC manufactured equipment has receptacles for interconnecting other units in this fashion.

5.9 PROCESSOR CLOCK ADJUSTMENT

This adjustment establishes the basic operating frequency of the processor.

a. Load address 0.

b. Deposit WAIT instruction (000001) into location 0.

c. Load address 0 again.

d. Place ENABLE/HALT switch in ENABLE position.

e. Depress START. At this point, the processor executes the WAIT instruction continuously, causing the processor clock to run synchronously.

f. Apply an oscilloscope probe to pin E03F1. This is the SCLK L output from the M728 Timing and States module. A complete cycle output should be 280 nanoseconds. The waveform should resemble that described on the K1-2 Timing and States print.

g. If necessary, adjust trimpot R8 on the M728 module. The trimpot is at location CDEF03.

h. At this point, the KA11 system unit can be removed by releasing the six screws mounted on the outside of the pins along the side of the system unit.

Installation of the KA11 system unit is the reverse of the above procedure.

5.10 REMOVAL/INSTALLATION

The following procedure is to be used whenever removing a KA11 system unit from the mounting box. Refer to Figure 5-2 for location of items mentioned in this procedure.

a. Make certain that all power is turned off.

b. Remove the top and bottom covers of the mounting box.

c. Release the front bezel by removing the Phillip's head screw at each of the four corners.

d. Remove the bezel.

CAUTION

Handle the bezel with care as this unit can be easily broken.

e. Remove the two screws that secure the KY11-A programmer's console. These screws are located approximately two inches from the bottom on both side panels.

f. Open the fan door by releasing the two fasteners which are mounted on the door.

g. Once the fan door is open, the following units are to be removed:

1. Power bus (3 modules)
2. M780 Teletype Control interconnecting cables
3. M781 HSR/HSP Control interconnecting cables
4. M920 Unibus connector
5. KY11-A Console panel

CAUTION

Due to the physical size of the KY11-A console and the extremely tight fit, extreme care must be used when removing this component to prevent damage.

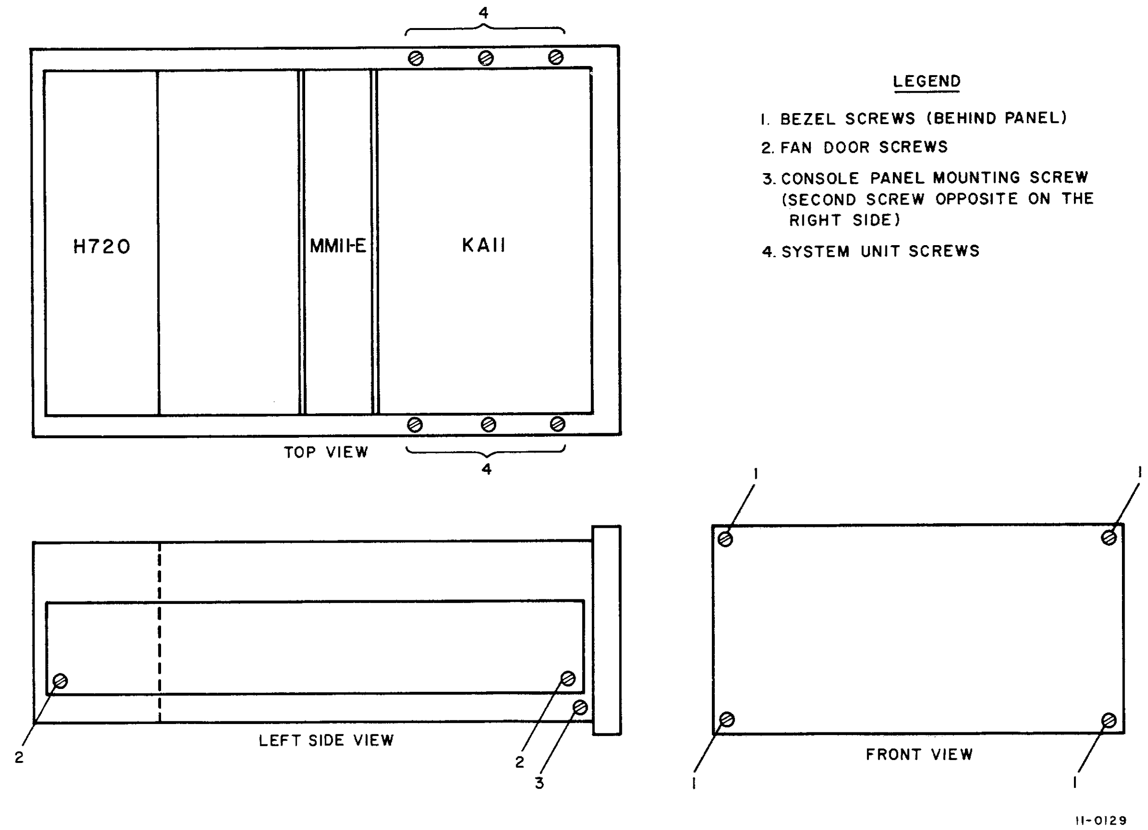


Figure 5-2 Mounting Hardware

5.11 MAINTENANCE TIPS

5.11.1 Diagnostic Programs

The MainDEC-11 diagnostic programs are designed to test particular devices, operations, or functions. Their purpose and operating instructions are included in the documentation supplied with each test tape. Processor Test 17 is an overall system exerciser and, as such, is a prime vehicle for isolating malfunctions to a device. Once a fault has been isolated to a specific device by Test 17, then the special tests for the device, operation, or function can be used to further determine the cause of the malfunction. This method, in most cases, determines the hardware problem areas. However, a problem may exist, all diagnostic programs perform satisfactorily, yet user equipment and/or programs fail. In this instance, a more likely place to look for the cause of the problem is the user program and/or equipment.

5.11.2 KM11 Maintenance Set

The KM11 Maintenance Set consists of the W130 and W131 modules and is one of the most valuable tools that can be used to troubleshoot the KA11 Processor. The Maintenance Set provides a capability for single clock stepping through programs, disabling time out, and providing BUS SSYN, all under operator control. It also furnishes indicators of ISR and BSR time states, MSYN, BSYN, Traps, R/W2, and condition codes. Three test indicators are available for connection of signals which may be desired in the course of troubleshooting. The connections can be made to the appropriate pins on the back panel wiring. Complete instructions for using the Maintenance Set are given in the *KM11 Maintenance Set manual.*

5.11.3 Observation of Service Major State Operation

The ability to observe operation through the various ISR states requires that the machine be single clocked by the KM11 Maintenance Set with ENABLE/HALT switch in the ENABLE position. If this switch is set to HALT, the console always gains control in SERVICE * ISR0 and the processor never proceeds through the rest of the major states.

APPENDIX A
PROCESSOR SIGNALS

Signal Name	Polarity	Drawing
A		
A DATA 07	L	K07-2
A DATA 15	L	K08-2
A FM DEST/INSTR	H	K10-4
ACROSS DATA	L	K11-2
ADD	L	K10-2
ADD00	L	K07-2
ADD01	L	K07-2
ADD02	L	K07-2
ADD03	L	K07-2
ADD04	L	K07-2
ADD05	L	K07-2
ADD06	L	K07-2
ADD07	L	K07-2
ADD08	L	K08-2
ADD09	L	K08-2
ADD10	L	K08-2
ADD11	L	K08-2
ADD12	L	K08-2
ADD13	L	K08-2
ADD14	L	K08-2
ADD15	L	K08-2
ADDRESS 0	H	K13-4
ADDRESS 1	H	K13-4
ADRS BIT 1	H	K10-3
ADRS BYTE OP	H	K10-3
ADRS DONE	L	K10-3
ADRS DONE	H	K13-3
ADRS MODE 2	L	K10-3
ADRS MODE 2	H	K13-3

Signal Name	Polarity	Drawing
ADRS MODE 4	L	K10-3
ADRS MODE 4	H	K13-3
ADRS MODE 5	L	K10-3
ADRS MODE (3+5+6+7)	H	K10-3
ADRS MODE (4+5)	H	K10-3
ADRS MODE (4+5)*REG6	H	K10-3
ADRS MODE (6+7)	H	K10-3
ADRS MODE (6+7)	L	K04-3
B		
B ACLO	L	K13-2
B ACLO	H	K13-2
B BBSY	H	K02-3
B DATA 07	L	K07-2
B DATA 15	L	K08-2
B D00	H	K09-5
B D01	H	K09-5
B D02	H	K09-5
B D03	H	K09-5
B D04	H	K09-4
B D05	H	K09-4
B D06	H	K09-4
B D07	H	K09-4
B D08	H	K03-3
B D09	H	K03-3
B D10	H	K03-3
B D11	H	K03-3
B D12	H	K03-3
B D13	H	K03-3
B D14	H	K03-3
B D15	H	K03-3

Signal Name	Polarity	Drawing
B DEST (0)	H	K01-4
B DEST (1)	H	K01-4
B EXEC (0)	H	K01-4
B EXEC (1)	H	K01-4
B FETCH (0)	H	K01-4
B FETCH (1)	H	K01-4
B INTR	H	K13-3
B MSYN	H	K13-3
B SACK	H	K02-3
B SERVICE	H	K01-4
B SOURCE (1)	H	K01-4
B SSYN	L	K13-3
B SSYN	H	K13-3
BAR00 (1)	H	K09-5
BAR01 (1)	H	K09-5
BAR02 (1)	H	K09-5
BAR03 (1)	H	K09-5
BAR04 (1)	H	K09-4
BAR05 (1)	H	K09-4
BAR06 (1)	H	K09-4
BAR07 (1)	H	K09-4
BAR08 (1)	H	K09-3
BAR09 (1)	H	K09-3
BAR10 (1)	H	K09-3
BAR11 (1)	H	K09-3
BAR12 (1)	H	K09-3
BAR13 (1)	H	K09-3
BAR14	H	K09-3
BAR14 (1)	H	K09-3
BAR15 (1)	H	K09-3
BAR16 (1)	H	K09-2
BAR17 (1)	H	K09-2
BBSYF (0)	H	K12-3
BBSYF (1)	H	K12-3
BC0 (1)	H	K13-2
BC1 (1)	H	K13-2
BERRF (0)	H	K12-2
BIC	L	K10-2
BINARY	L	K10-2
BINARY	H	K13-3

Signal Name	Polarity	Drawing
BIS	L	K10-2
BIT	L	K10-2
BR	L	K10-2
BRANCH	L	K06-3
BRANCH	H	K10-2
BRQ	L	K03-2
BSR FM 12	L	K02-3
BSR FM 15	L	K02-3
BSR FM 17	L	K02-3
BSR 00	H	K01-3
BSR 01	L	K01-3
BSR 01	H	K01-3
BSR 03	H	K01-3
BSR 07	H	K01-3
BSR 08	L	K01-3
BSR 08	H	K01-3
BSR 12	L	K01-3
BSR 12	H	K01-3
BSR 14	L	K01-3
BSR 14	H	K01-3
BSR 15	L	K01-3
BSR 15	H	K01-3
BSR (1+3+7)	H	K01-3
BSR (3+7)	H	K01-3
BSR (7+8)	H	K01-3
BSR (7+14)	H	K01-3
BSR (15+14+12+8)	H	K01-3
BSR14 + SVC * ISR0	L	K15-2
BUS IN DONE	L	K02-2
BUS IN DONE	H	K02-2
BUS INDICATOR	H	K13-4
BYTE OP	H	K10-3
C		
C (1)	H	K09-5
C DATA	H	K10-4
CARRY 00	L	K01-2
CARRY 00 (0)	H	K06-5
CARRY 07	L	K07-2
CARRY 15	L	K08-2
CARRY DATA	L	K11-2

Signal Name	Polarity	Drawing
CARRY INSTR	H	K10-4
CBRF	H	K03-2
CBRF (0)	H	K03-2
CC OP	H	K10-3
CHANGE CODES	L	K10-3
CLK	L	K01-2
CLK	H	K01-2
CLK BR	L	K02-3
CLK BAR	H	K13-3
CLK C	H	K10-4
CLK IR	H	K06-2
CLK N'Z'V	H	K10-4
CLK OFF (0)	H	K01-2
CLK OFF (1)	H	K01-2
CLK PDNF	H	K15-2
CLK RESTART	L	K01-2
CLK RUN (1)	H	K01-2
CLK T	H	K10-4
CLR	L	K10-2
CMP	L	K10-2
CNPRF (1)	H	K03-2
CONS BR	H	K13-4
CONS GRANT (0)	H	K12-3
CONS GRANT (1)	H	K12-3
CONS NPR	H	K13-4
CONSF (0)	H	K13-4
CONSF (1)	H	K13-4
CONT	L	KY-3
CONT	H	KY-3
CSR 00	H	K13-4
CSR 00 (1)	H	K13-4
CSR 01	H	K13-4
CSR 02	L	K13-4
CSR 02	H	K13-4
CSR 03	H	K13-4
D		
D PERIF RELEASE	L	K02-3
DATA CLR	L	K01-2
DATA CLR	H	K01-2
DATA WAIT FM 1	L	K06-2

Signal Name	Polarity	Drawing
DATIP	L	K13-2
DATO ENTRY	L	K13-2
DATO ENTRY	H	K13-2
DATO = ENTRY	H	K02-3
DEC	L	K10-2
DEP	L	KY-3
DEP	H	K13-4
DEST MODE 0	L	K10-3
DEST MODE 0	H	K13-3
D00	H	K07-2
D01	H	K07-2
D02	H	K07-2
D03	H	K07-2
D04	H	K07-2
D05	H	K07-2
D06	H	K07-2
D07	H	K07-2
D08	H	K08-2
D09	H	K08-2
D10	H	K08-2
D11	H	K08-2
D12	H	K08-2
D13	H	K08-2
D14	H	K08-2
D15	H	K08-2
D07/0 ZERO	L	K09-2
D15/0 ZERO	L	K09-2
D15/8 ZERO	L	K09-2
E		
EXAM	L	KY-3
EXAM	H	KY-3
EXAM	H	K13-4
(EXAM+DEP)	L	K06-5
(EXAM*DEP)	H	K02-3
EXEC FM 1	H	K01-4
(EXEC*ISR0)	H	K06-3
(EXEC*ISR1)	H	K02-2
(EXEC*JSR)	H	K04-3
EXTRA	H	K06-3

Signal Name	Polarity	Drawing
F		
FETCH FM SVC	H	K15-2
(FETCH*ISR0)	H	K02-2
(FETCH*ISR1)	H	K01-4
G		
GATE C	H	K10-4
GATE CC FM BYTE	H	K11-2
GATE CC FM WORD	H	K11-2
GATE LEFT 15/0	H	K06-3
GATE RAFM DEST	H	K04-3
GATE RAFM SOURCE	H	K04-3
GATE RAFMBAR	H	K04-3
GATE FARMSAD	H	K04-2
GATE RIGHT 15/0	H	K06-3
GATE ROT/SHF	H	K10-4
GATE SEX	H	K06-4
GATE STFMD	H	K10-4
GATED B INTR	L	K13-3
GATED P RESTART	L	K15-2
GRANT BR	H	K02-3
GRANT	H	K02-3
GATE A FM-BD15/0	H	K06-4
GATE A FM R0	H	K06-3
GATE A FM-R0	H	K06-3
GATE A FM R15/1	H	K06-3
GATE A FM-R15/1	H	K06-3
GATE ADD 7/0	H	K06-3
GATE ADD 15/8	H	K06-3
GATE B FM BD15/0	H	K06-4
GATE B FM R15/8	H	K06-4
GATE B FM STPM	H	K06-4
GATE B/ISR0	H	K10-4
GATE B/ISR1	H	K10-4
GATE BUS FM D	H	K09-2
GATE BUS FM SR	H	K13-2
GATE BYTE 7/0	H	K06-3
GATE BYTE 15/8	H	K06-3
H		
HALT	L	KY-3
HALT	H	KY-3
HALT	H	K10-3
HALT F (1)	L	K11-2
HIGH C DATA	L	K11-2
I		
INCF	L	K13-4
INIT	L	KY-3
INIT	L	K13-2
INIT	H	K13-2
INSTR STPM 02	H	K10-4
INSTR STPM 03	H	K10-4
INSTR STPM 04	H	K10-4
INTERNAL ADRS	H	K02-2
INTERNAL ADRS	L	K02-2
INTRF (0)	H	K12-3
INTRF (1)	H	K12-3
IR00 (1)	H	K09-5
IR01 (1)	H	K09-5
IR02 (1)	H	K09-5
IR03 (0)	H	K09-5
IR04 (0)	H	K09-4
IR05 (0)	H	K09-4
IR06 (1)	H	K09-4
IR07 (1)	H	K09-4
IR08 (1)	H	K09-3
IR09 (0)	H	K09-3
IR10 (0)	H	K09-3
IR11 (0)	H	K09-3
IR12 (1)	H	K09-3
IR13 (1)	H	K09-3
IR14 (1)	H	K09-3
IR15 (1)	H	K09-3
ISR F M02/SERVICE	L	K02-2
ISR F M02/SERVICE	H	K02-2
ISR FM 00	L	K02-2
ISR FM 1	L	K02-3
ISR FM 15	L	K02-3
ISR 00	H	K01-3
ISR 00	L	K01-3
ISR 01	H	K01-3
ISR 01	L	K01-3

Signal Name	Polarity	Drawing
ISR 02	H	K01-3
ISR 03	H	K01-3
ISR 03	L	K01-3
ISR 07	H	K01-3
ISR 07	L	K01-3
ISR 08	H	K01-3
ISR 08	L	K01-3
ISR 12	H	K01-3
ISR 21	L	K01-3
ISR 14	H	K01-3
ISR 15	H	K01-3
ISR (1+3)	H	K01-3
ISR (3+7)	H	K01-3
ISR (12+15)	H	K01-3
(ISR12* - INTRF)	L	K13-2
J		
JMP	L	K10-3
JMP	H	K04-2
(JMP*JSR)	L	K10-3
JSR	L	K10-2
JSR	H	K10-2
L		
LATCH A 15/0	H	K06-2
LATCH B (0)	H	K06-2
LATCH B 15/0	H	K06-2
LEFT DATA 00	L	K10-4
LOAD ADRS	L	K13-4
LOAD ADRS	L	KY-3
LOAD ADRS	H	K13-4
LOAD ADRS	H	KY-3
LTC	L	K14-2
M		
M CLK	L	KM-2
M CLK ENABLE	L	KM-2
MOV	L	K10-2
MSYN (1)	H	K13-3
N		
N (1)	H	K09-5
N DATA	L	K11-2

Signal Name	Polarity	Drawing
NO SACK (0)	H	K13-2
NPR ENABLE	L	K15-2
NPR ENTRY	H	K12-3
NPRF	H	K03-2
O		
ODD ADRS ERR	L	K10-3
ODD ADRS ERR	H	K13-2
OVFLF (1)	H	K12-2
P		
P CLR DATA WAIT	H	K06-2
P CONSF (1)	H	K13-4
P DATA START	H	K06-2
P RESTART	L	K13-2
P TIME OUT	L	K13-2
PARTIAL BST FM 1	L	K02-3
PERIF RELEASE	L	K02-3
PERIF RELEASE	H	K02-3
PROC BG 04	H	K03-2
PROC BG 05	H	K03-2
PROC BG 06	H	K03-2
PROC BG 07	H	K03-2
PROC CNTL	L	K15-2
PROC CNTL	H	K13-4
PROC RELEASE	L	K02-2
PROC RELEASE	H	K02-2
PUPF (0)	H	K03-3
PWRF	L	K03-3
PWRF	H	K03-3
PWR UP	H	K13-2
P1 CSR 00	H	K13-4
P1 CSR	H	K13-4
P1 CSR	L	K13-4
P2 CSR	H	K13-4
P2 CSR	L	K13-4
P3 CSR	H	K13-4
P3 CSR	L	K13-4
R		
REG ADRS	L	K04-3
REG ADRS	H	K09-2

Signal Name	Polarity	Drawing
REG GATE	H	K01-2
REG LATCH	H	K01-2
REG 6	L	K10-3
RESET	L	K10-3
(RESET+ HALT)	H	K10-3
REQUEST	H	K02-3
RIGHT DATA 07	L	K10-4
RIGHT DATA 15	L	K10-4
ROT/SHF	L	K10-3
ROT/SHF C DATA	L	K10-4
ROT/SHF L	H	K10-3
ROT/SHF R	H	K10-3
RTI	L	K10-3
RTI	H	K10-4
RTS	H	K10-3
R/W1	H	K01-2
R/W2	H	K01-2
R/W3	L	K01-2
R/W3	H	K01-2
R00 (1)	H	K05-2
R01 (1)	H	K05-2
R02 (1)	H	K05-2
R03 (1)	H	K05-2
R04 (1)	H	K05-2
R05 (1)	H	K05-2
R06 (1)	H	K05-2
R07 (1)	H	K05-2
R08 (1)	H	K05-2
R09 (1)	H	K05-2
R10 (1)	H	K05-2
R11 (1)	H	K05-2
R12 (1)	H	K05-2
R13 (1)	H	K05-2
R14 (1)	H	K05-2
R15 (1)	H	K05-2
S		
S CLK	L	K01-2
S CLK	H	K01-2
SAD 00	H	K04-2
SAD 01	H	K04-2

Signal Name	Polarity	Drawing
SAD 02	H	K04-2
SAD 03	H	K04-2
SBC	L	K10-2
S/CYCLE	L	KY-3
SERV 0	L	K15-2
SERV 0	H	K15-2
SERVICE	L	K01-4
SERVICE	H	K01-4
(SERVICE*ISR0)	L	K12-3
(SERVICE*ISR0)	H	K12-3
(SERVICE*ISR0)	L	K01-3
(SERVICE*ISR8)	L	K01-3
SHIFT 1 SR	H	K02-2
S/INST	L	KY-3
(SO+DE)	H	K01-4
(SO+DE)	L	K01-4
SOURCE MODE 0	L	K13-3
SOURCE MODE 0	H	K10-3
SR ADRS	H	K09-2
SR16 (switch reg.)	H	KY-3
SR16 (switch reg.)	L	KY-3
SR17	H	KY-3
SR17	L	KY-3
SSYN (1)	H	K13-3
ST ADRS	H	K09-2
(ST+EX+DEP)	H	K06-4
ST PTR CLK	L	K01-3
STADRS	L	K02-3
START	L	KY-3
START	H	KY-3
START	H	K13-4
START F (1)	H	K13-4
STPM 02	H	K12-3
STPM 03	H	K12-3
STPM 04	H	K12-3
SUB	L	K10-2
SVC CLR OVFLF	H	K12-2
SVC FM INSTR	H	K10-4
SWAB	H	K10-3
ST05 (1)	H	K09-4

Signal Name	Polarity	Drawing
ST06 (1)	H	K09-4
ST07 (1)	H	K09-4
T		
T (1)	H	K09-4
TEST	L	K04-3
TEST	H	K10-4
TIME OUT (0)	H	K13-2
TIME OUT (1)	L	K13-2
TP1	L	K13-2
TP2	H	K13-4
TP2	H	K02-3
TRAPS	L	K12-3
TRAPS	H	K12-3
TRACF (1)	H	K12-3
TST 01	H	KM-2
TST 02	H	KM-2
U		
(U+B+R/S)	L	K10-3
(U * B * R/S)	H	K13-3
(U * R/S)	L	K01-4
(U+R/S)	H	K10-3
V		
V (1)	H	K09-5
V DATA	L	K11-2
W		
WAIT	L	K10-3
WAIT	H	K02-3
WAIT CLR	H	K15-2
(WAIT * -TRAPS)	L	K15-2
W/ENABLE 7/0	L	K04-3
W/ENABLE 15/8	L	K04-3
(WORD+MOVE)	L	K06-4
WRITE 7/0	H	K01-2
WRITE 15/8	H	K01-2
X		
X00	H	K05-2
X01	H	K05-2
X10	H	K05-2
X11	H	K05-2

Signal Name	Polarity	Drawing
Y		
Y00	H	K05-2
Y01	H	K05-2
Y10	H	K05-2
Y11	H	K05-2
Z		
Z (1)	H	K09-5
Z DATA	H	K11-2

READER'S COMMENTS

PDP-11 KA11 PROCESSOR
DEC-11-HR2B-D

Your comments and suggestions will help us in our continuous effort to improve the quality and usefulness of our publications.

What is your general reaction to this manual? In your judgment is it complete, accurate, well organized, well written, etc.? Is it easy to use?

What features are most useful?

What faults do you find with the manual?

Does this manual satisfy the need you think it was intended to satisfy?

Does it satisfy *your* needs? Why?

Would you please indicate any factual errors you have found.

Please describe your position.

Name Organization

Street Department

City State Zip or Country

Fold Here

Do Not Tear - Fold Here and Staple