pdp11

KT11-D
memory management option
user's manual

digital

EK-KT11D-OP-001

KT11-D
memory management option
user's manual

digital equipment corporation • maynard, massachusetts

1st Edition, September 1976

Printed in U.S.A.

CONTENTS

ILLUSTRATIONS

TABLES

INTRODUCTION

This manual describes the KT11-D Memory Management Unit, which is a hardware option available for use with the PDP-11 Programmed Data Processor. The purpose of this manual is to:

1. Provide an overall understanding of how the KT11-D functions in a PDP-11 system.
2. Explain how the KT11-D hardware can be used in the development of the memory management module of a software operating system.
3. Describe the KT11-D logic in sufficient detail to enable maintenance personnel to perform on-site troubleshooting and repair.

The KT11-D interacts with the KD11-A Central Processor Unit and operating system software to achieve PDP-11 system management objectives. The use of the KJ11 Stack Limit option is also utilized for expanded kernel stacking flexibility. For this reason, a background description of memory management system objectives and programming information is included in this manual.

Chapter 1 introduces the purpose and use of the memory management unit.

Chapter 2 references the installation procedures provided in the *PDP-11/40, PDP-11/35 System Manual (21 Inch Chassis).* There are KT11-D wiring change procedures provided in this manual.

Chapter 3 contains operation and programming reference information. It describes the internal registers and their application from a software viewpoint. Programming details, hints, and exceptions of interest to programmers are included.

Detailed descriptions of the processor, console, Unibus, and memory logic that interface with the memory management unit are provided in the following related documents.

PDP-11/40, PDP-11/35 System Manual (21 Inch Chassis)	EK-11040-TM-002
KD11-A Processor Maintenance Manual	EK-KD11A-MM-001

CHAPTER 1
GENERAL DESCRIPTION

1.1 PURPOSE AND USE OF KT11-D OPTION

The KT11-D Memory Management Unit is a PDP-11 hardware option that:

a. Expands the basic 32K-word address capability of the KD11-A to 128K words.

b. Provides a "virtual" address space with memory relocation and protection for multi-user timesharing systems.

c. Implements the separate address spaces for the PDP-11 Kernel, and User modes of operation.

d. Provides memory management information for use of memory in multi-user, multi-program systems.

These features are briefly reviewed in the following paragraphs. The essential details required to thoroughly understand the memory management option from the PDP-11 system programmer's viewpoint are presented in Chapter 3, Operation and Programming.

1.1.1 Memory Expansion

The KT11-D option extends the basic PDP-11 physical address capability to 128K words. The 16-bit word length of the KD11 limits the memory address capability of the basic PDP-11 to 32K words. (The least significant address bit is used for byte addressing.) The upper 4K of the address space is always reserved for internal register and external device addresses. Therefore, the total memory address capability is extended from 28K to 124K words.

This is accomplished by converting the 16-bit virtual address generated by the processor to an 18-bit physical address. Several sets of relocation registers are the key to this feature. A complete description of how the active page address registers (PAR) are used to construct the physical address is provided in Paragraph 3.2.2.

1.1.2 Virtual Address Space

Because the KT11-D relocates, if enabled, all addresses automatically, the KD11-A may be considered to be operating in a virtual address space. This means that no matter where a program is loaded into physical memory, it will not have to be "re-linked"; it always appears to be at the same virtual location in memory.

1.1.3 Minimal Memory Fragmentation

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

In addition, the KT11-D provides a means of allocating a page as small as 32 words, so that short procedures or data areas need occupy only as much memory as required. This is a useful feature in real-time control systems that contain many separate small tasks. It is also a useful feature for stack and buffer control.

1.1.4 Memory Protection

Each virtual page has a separate protection key associated with it. There are three possible basic protection levels. These are listed, in order of increasing protection, as follows:

1. All read or write accesses allowed.
2. Only read accesses allowed.
3. No access allowed.

Any attempt to violate any of these forms of protection is prevented by the KT11-D hardware. For example, an illegal "read" attempt (attempting to read from a page that is protection keyed for no access) does not result in obtaining the contents of the location. An illegal "write" attempt does not result in the modification of the contents of the location. All such illegal access attempts cause an immediate trap (called an "abort") to the Kernel space.

The KT11-D hardware records and preserves abort status information so that the offending user can be notified of the violation. It is not generally possible to recover from these aborts.

1.1.5 Operating Mode Control

In a multi-programmed, timeshared system, user programs must be prevented from modifying or destroying the operating system and each other. The KT11-D implements the Kernel/User modes of PDP-11 operation upon which the timeshared system is based. A page address register/page descriptor register (PAR/PDR) set is provided for each mode of operation. The KT11-D analyzes every memory reference, based on the processor status word, to enable the correct PAR/PDR set. Thus, a User mode program, for example, is prevented from operating in space assigned to Kernel programs.

Compilers, utility programs, and other shared source programs, might be assigned to User mode address space, with access codes keyed to permit read-only access.

1.1.6 Memory Management

In a multi-program, multi-user environment, memory space must be used in the most efficient way, to accommodate as many users as possible with minimum delay. The KT11-D logic maintains a bit which indicates whether the associated page has ever been written into. The software memory management system can interrogate each PDR to determine whether or not that page has been used. If a current active page has been written into, the memory management operating system needs to be informed, so that the modified program can be rewritten into secondary storage before that page is overlayed.

1.2 KT11-D MEMORY MANAGEMENT UNIT SPECIFICATIONS

A summary of specifications and technical characteristics for the KT11-D Memory Management Unit option is listed in Table 1-1.

Table 1-1
Abridged Specifications Summary

Characteristic	Specification or Description
Memory Expansion	Expands PDP-11 memory address capability up to 124K words.
Interface	Address line outputs compatible with PDP-11 Unibus.
Delay	Adds 150 ns to every memory reference.

Table 1-1 (Cont)
Abridged Specifications Summary

Characteristic	Specification or Description
Modes of Operation	Implements the KD11 Central Processor Kernel, and User modes.
Available Pages	Provides eight pages for each mode.
Page Length	A page can vary in length from one 32-word block up to 128 32-word blocks, in 32-word increments. Maximum page length is therefore 4096 words.
Program capacity	Eight 4096-word pages will accommodate a 32K-word program.
Physical description	Option consists of one standard hex module (15 x 8.5 in.) that mounts in PDP-11/40 CPU backplane assembly.
Module M7236	Located in slot 8 rows A through F.
Environmental	Refer to overall PDP-11/40 specifications listed in system manual, EK-11040-TM-002

1.3 REFERENCE LITERATURE

The following list of references covers some of the more general aspects of "memory management" tasks of interest to systems programmers. It is part of the recommended bibliography of the National Academy of Engineering for its course outline on "Operating System Principles".

The following abbreviations are used in the bibliography:

ACM	Association for Computing Machinery
IEEE	Institute for Electrical and Electronics Engineers
IEEETC	IEEE Transactions on Computers
CACM	Communications of the ACM
JACM	Journal of the ACM
CS	Computing Surveys (ACM)
FJCC	Fall Joint Computer Conference
SJCC	Spring Joint Computer Conference
2SOSP	Second Symposium on Operating Systems Principles (proceedings available from ACM, 1133 Avenue of the Americas, New York, N.Y. 10036.

Abate, J., and Dubner, H. *Optimizing the Performance of a Drum-Like Storage.* IEEE Trans. C-18, 11 (Nov. 1969), 992-997.

Belady, L.A. *A Study of Replacement Algorithms for Virtual Storage Computers.* IBM Sys. J. 5, 2 (1966), 78-101.

Bensoussan, A., Clingen, C.T., and Daley, R.C. *The Multics Virtual Memory.* Proc. 2SOSP (Oct. 1969).

Denning, P.J. *The Working Set Model for Program Behavior.* Comm. ACM 11, 5 (May 1968), 323-333.

Denning, P.J. *Thrashing: Its Causes and Prevention.* AFIPS Conf. Proc. 33 (1968 FJCC), 915-922.

Dennis, J.B. *Segmentation and the Design of Multiprogrammed Computer Systems.* JACM 12, 4 (Oct. 1965), 589-602.

Kilburn, T. et al. *One-Level Storage System.* IRE Trans. EC-11, 2 (Apr. 1962), 223-235.

Knuth, D.E. *The Art of Computer Programming (Vol. 1).* Addison-Wesley (1969), Ch.2.

Mattson, R.L., Gecsei, J., Slutz, D.R., and Traiger, I.L. *Evaluation Techniques for Storage Hierarchies.* IBM Sys. J. 9, 2 (1970), 78-117.

Randell, B., and Keuhner, C.J. *Dynamic Storage Allocation Systems.* Comm. ACM11, 5 (May 1968), 297-305.

Sayre, D. *Is Automatic Folding of Programs Efficient Enough to Displace Manual?* CACM 12, 12 (Dec. 1969), 656-660.

Wilkes, M.V. *Slave Memories and Dynamic Storage Allocation.* IEEE Trans. EC-14 (Apr. 1965), 270-271.

Wilkes, M.V. *Time-Sharing Computer Systems.* Am. Elsevier (1969).

CHAPTER 2
INSTALLATION INFORMATION

The installation procedure for the KT11-D Memory Management Unit option is included as part of the complete PDP-11 system installation procedure described in Chapter 2 of the *PDP-11/40, PDP-11/35 System Manual (21 Inch Chassis).* Specific procedures for wiring changes to the Processor are given below with descriptions of hook operations that require no wiring changes. When the KT11-D is included as part of the initial PDP-11 system, the M7236 module is installed prior to shipment. If the KT11-D option is being added to an existing PDP-11 system, the installation procedure is straightforward. The M7236 module is installed in slot 8, rows A-F of the CPU backplane assembly.

The wiring modifications to the KD11-A Processor that are necessary are as follows:

a. Refer to Processor Block Schematic K1-6, location A-6, and move jumper W10 to the ground inputs of the 8881 gates. This change disables the KD11-A from giving an address to the Unibus (Figure 2-1).

b. Refer to Processor Block Schematic K1-8, location C/D-8, and remove jumpers W5, W6, and W9. This change enables the correct selection of User and Kernel stacks in either explicit or implicit operations (Figure 2-2).

c. Refer to Processor Block Schematic K1-7, location C/D-3, and remove jumpers W1, W2, W3, and W4. This change enables the KT11-D to use an 18-bit virtual address to decode all internal register addressing (Figure 2-3).

d. Refer to Processor Block Schematic K4-4, location C-5. Remove jumper W2 and connect jumper W2A. This change connects pins 10, 11, and 12 of 74H55 module at location E-6 to A07H2 (KT-3 FAULT H). This change enables the KT11-D to start a trap sequence for a KT Abort condition (Figure 2-4).

e. Refer to Processor Block Schematic K1-9, location D-5. Remove jumpers W7 and W8. This change disables automatic operation of console address lights BA (17:16) on assertion of K1-6 BA (17:16) from the Processor (Figure 2-5).

f. Refer to Processor Block Schematic K4-4, location C-4. Add capacitor C113 (680 mmf, 100 WVDC) DEC Part Number 10-00026. This change extends the CLK MSYN H delay from 150 ns to 300 ns to enable memory management logic to propagate a new address to the Unibus while retaining bus specifications (Figure 2-6).

g. Refer to Processor Block Schematic K4-4, location D-4. Add capacitor C114 (560 mmf, 100 WVDC) DEC Part Number 10-00025. This change increases the delay of fast MSYN in the Processor from 75 ns to 225 ns (Figure 2-6).

When the KT11-D Memory Management Option is added to an existing PDP-11 system, the KJ11-A Stack Limit Register Option must also be added.

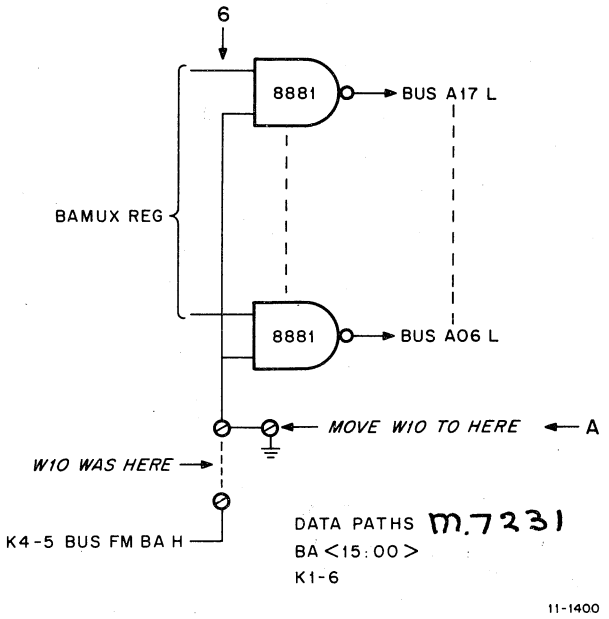


Figure 2-1 KD11-A Wiring Change No. 1

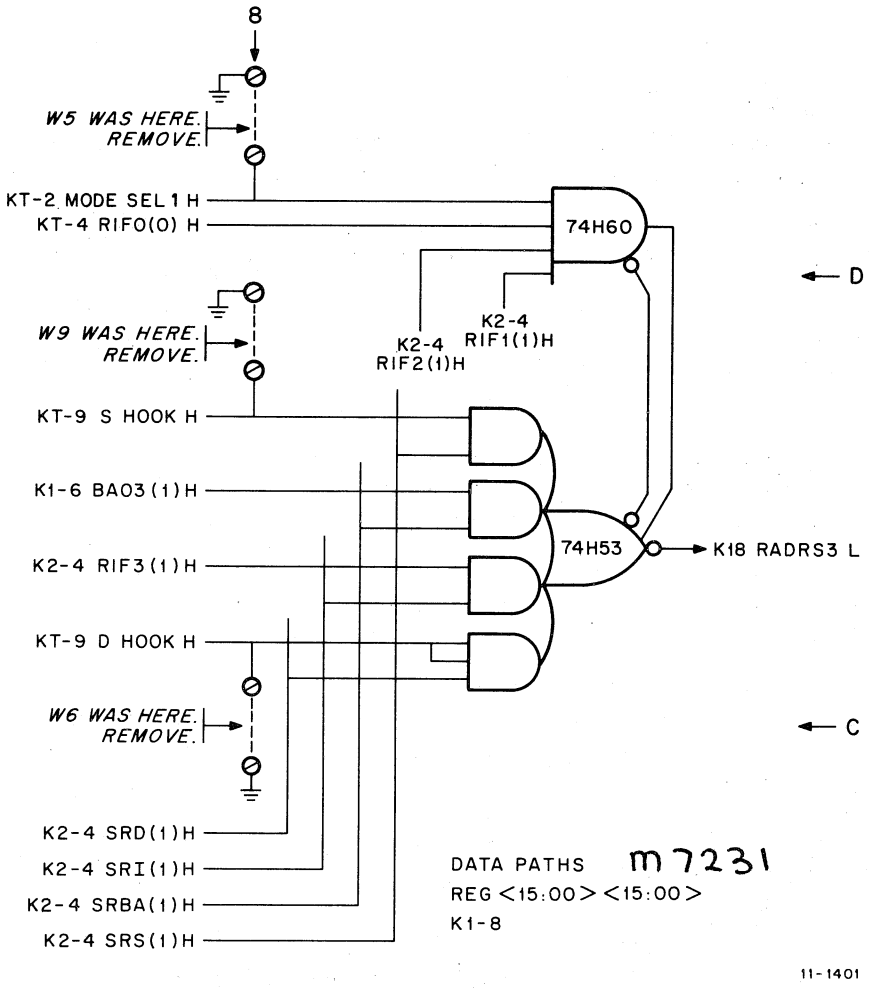


Figure 2-2 KD11-A Wiring Change No. 2

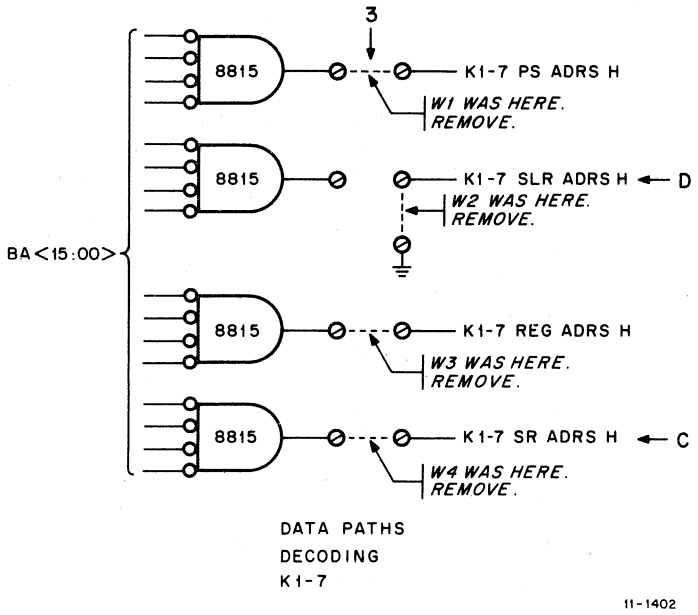


Figure 2-3 KD11-A Wiring Change No. 3 M7231

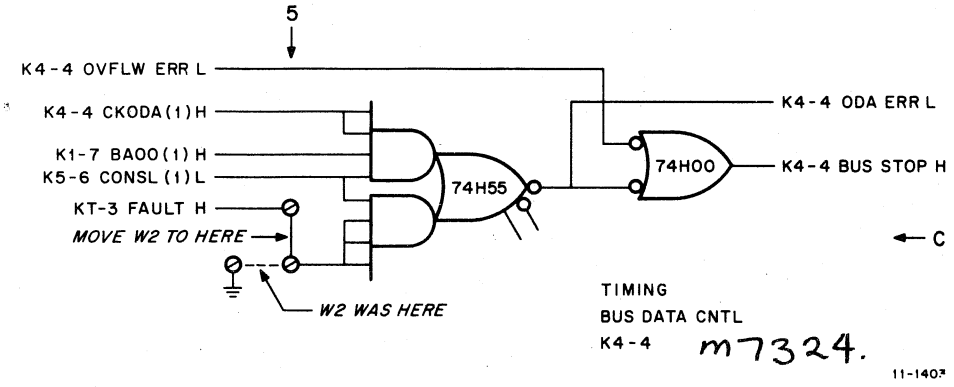


Figure 2-4 KD11-A Wiring Change No. 4

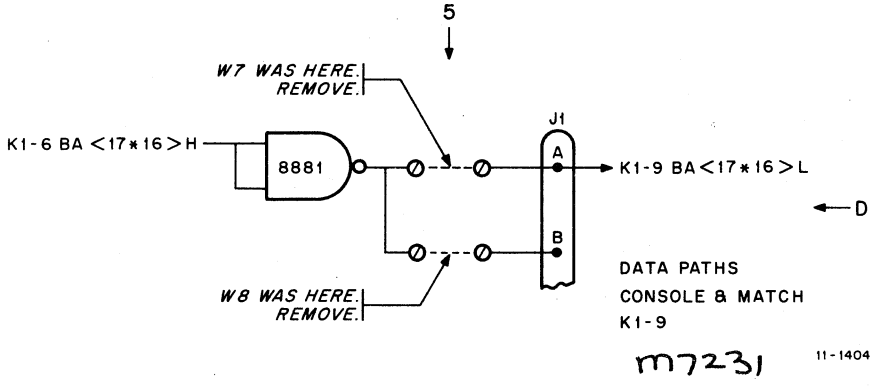


Figure 2-5 KD11-A Wiring Change No. 5

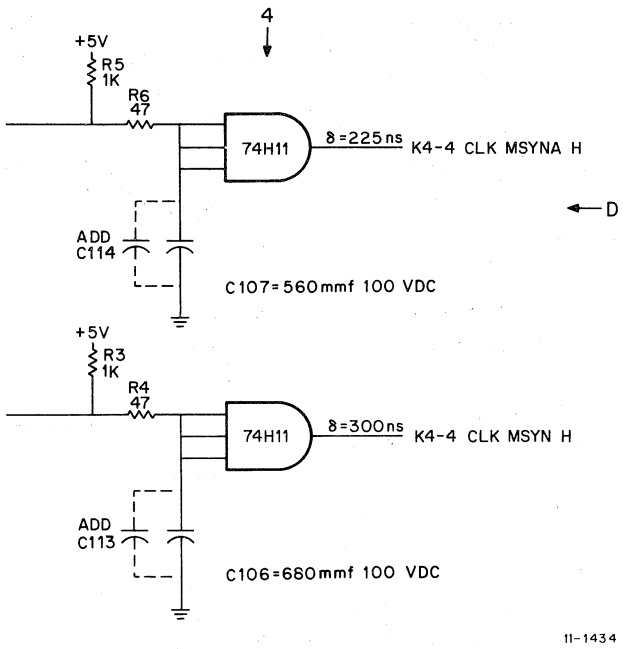


Figure 2-6 KD11-A Wiring Changes No. 6 and No. 7

The wiring modifications to the KD11-A Processor that are necessary for the KJ11-A are as follows:

a. Refer to Processor Block Schematic K4-4. Move jumper W1 on the M7234 in accordance with the notes on the drawing. This change permits the KJ11-A to create a stack overflow.

b. Refer to Processor Block Schematic K5-4. Move jumper W1 on the M7235 in accordance with the notes on the drawing. This change allows the stack overflow flip-flop to be set.

Once installed, the KT11-D option is ready to be checked out, using the diagnostic programs supplied with the option.

Other hooks, that do not include wiring changes but do represent signals sent from the KT11-D to the KD11-A to modify its operation with the Memory Management Option, include the following functions;

a. As shown on Processor Drawing K3-2, location B-3, and in Figure 2-7, as a result of KT-3 FAULT H starting a Trap Sequence, KT-3 FAULT L going to F05M2 forces a jam to the correct flow path in the Trap Sequence on a UBF 10.

b. As shown on Processor Drawing K3-5, location C-4, and in Figure 2-8, on a detection of an instruction unique to memory management, the KT11-D signals the KD11-A to recycle in the fetch flow to decode the new instruction.

c. As shown on Processor Drawing K3-6, location C-3 and in Figure 2-9, KT-2 PS15(0)H indicates User mode. This signal controls the restricted use of a reset in User mode and the KT11-D creates the KT INSTR L signal to recycle the Reset Instruction as a NOP Instruction.

d. As shown on Processor Drawing K4-4, location D-6 and in Figure 2-10, KT-2 PS15(0)H (when LOW) is used to disable stack overflow detection in User mode.

e. As shown on Processor Drawing K4-4, location D-3 and in Figure 2-11, KT-6 NO MSYN L is created by the KT11-D on an internal access and will disable BUS MSYN to the Bus.

f. As shown on Processor Drawing K5-2, location C-6 and in Figure 2-12, KT-2 INH PS CLK L disables the clocking of CLK PS (07:T) in User mode.

g. As shown on Processor Drawing K5-5, location B-2 and in Figure 2-13, the signal KT-9 MFP SMO L is a signal that chooses the correct destination stack address on a MFP R6 instruction.

h. As shown on Processor Drawing K5-7, location C-3 and in Figure 2-14, the signal KT-9 RELOCATE ENB L enables the VIRTUAL lamp on the Console. KT-9 PS15 L is used to enable the USER lamp on the Console. Signals KT-9 SR16 L and KT-9 SR17 L are used to enable the BA 16 and BA 17 lamps on the Console.

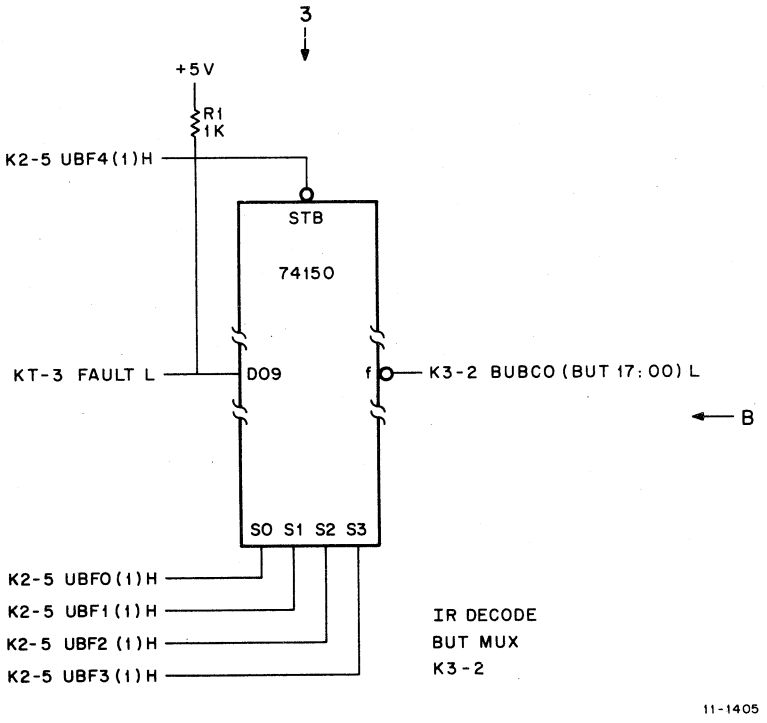


Figure 2-7 KT-3 Fault H Hook In KD11-A

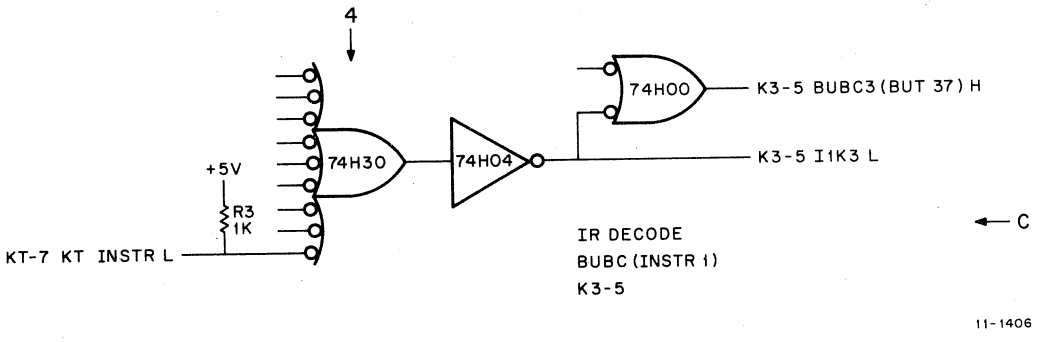


Figure 2-8 KT-7 KT INSTR L Hook In KD11-A

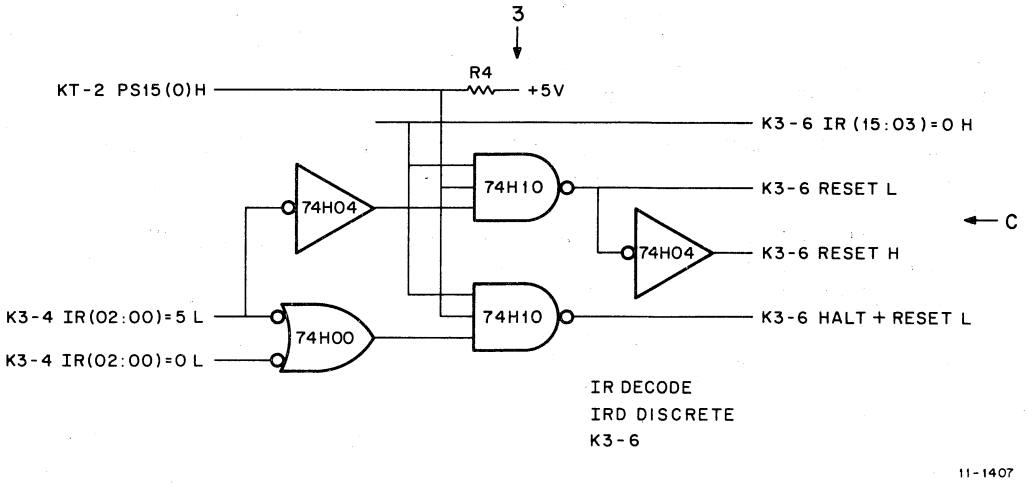


Figure 2-9 KT-2 PS15(0)H Hook In KD11-A

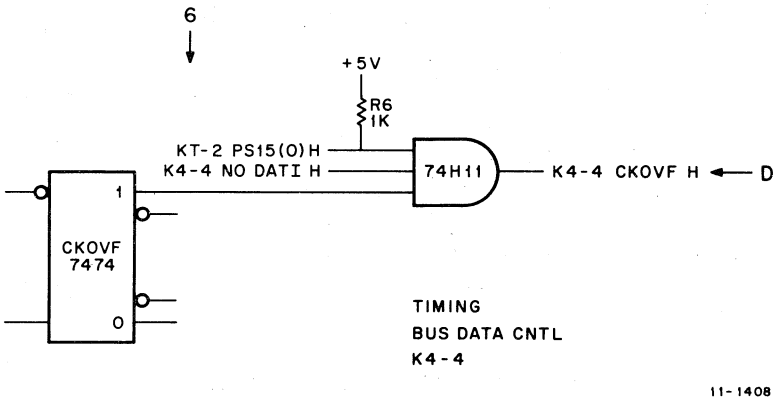


Figure 2-10 KT-2 PS15(0)L Hook In KD11-A

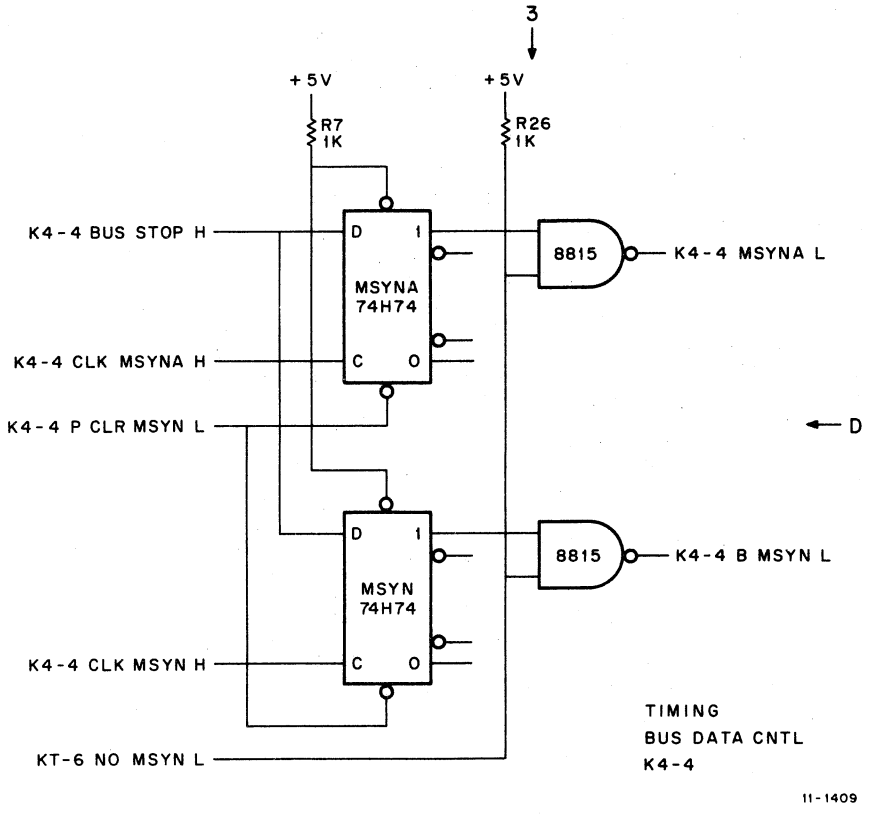


Figure 2-11 KT-6 No MSYN L Hook in KD11-A

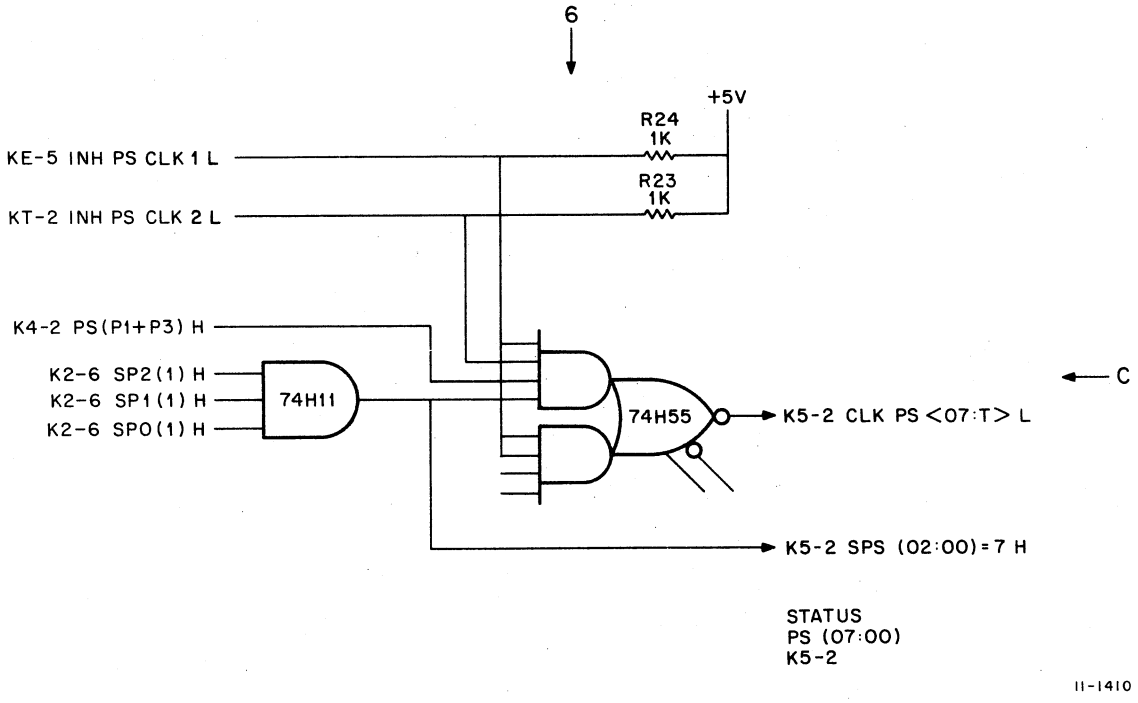


Figure 2-12 KT-2 INH PS CLK 2 L Hook in KD11-A

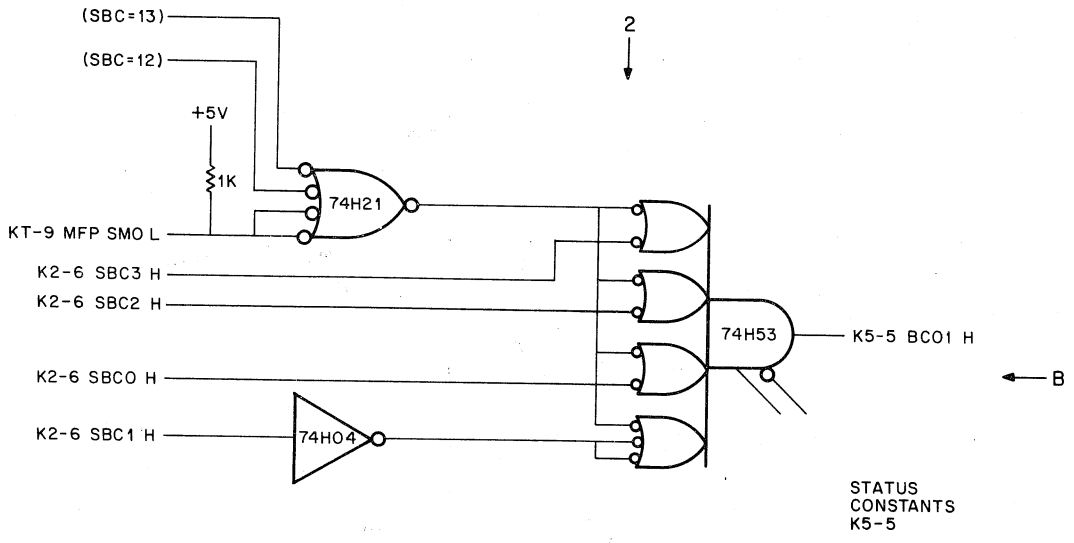


Figure 2-13 KT-9 MFP SM0L Hook In KD11-A

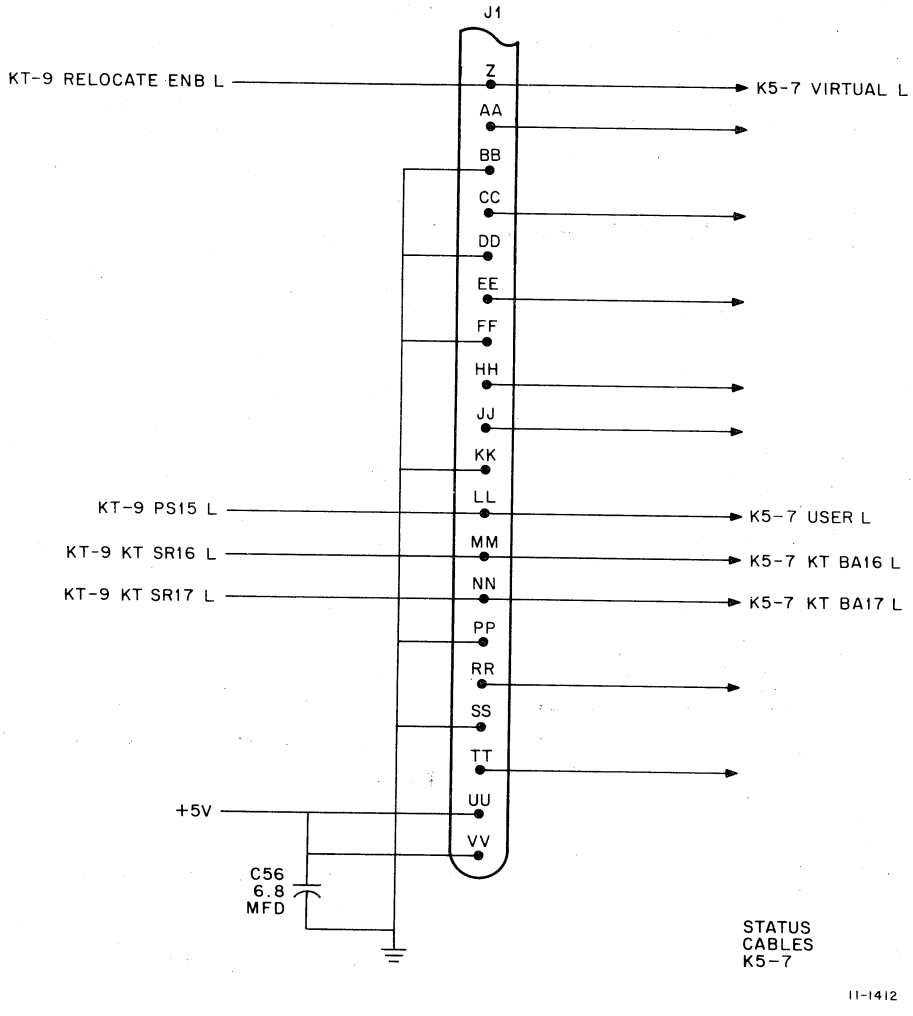


Figure 2-14 Console MM Enable Hooks In KD11-A

CHAPTER 3
OPERATION AND PROGRAMMING

3.1 MEMORY MANAGEMENT SYSTEM INTRODUCTION

The purpose of this chapter is to describe the capabilities and objectives of the PDP-11 memory management system. The operating characteristics of the KT11-D Memory Management Unit, which performs the hardware functions in the paging system, are described from the system programmer's viewpoint.

Suggested techniques that are included in this chapter are presented only for the purpose of illustrating hardware operating characteristics and as examples of how system programmers can use the KT11-D in developing an operating system. This information is also presented to provide a better understanding of the hardware for maintenance purposes.

NOTE
The information in this chapter does not describe the DEC Operating System for the PDP-11.

In the following paragraphs, the general features of the KT11-D Memory Management Unit are introduced, beginning with the basic memory relocation and extended memory addressing capability. Next, some of the general requirements of a memory management system are described, along with illustrations of how the KT11-D hardware can be used to implement such a system. Following that, the overall memory protection requirements and corresponding facilities provided by the KT11-D are described.

The system programmer has the option of using any or all of the KT11-D memory management capabilities, depending upon whether simple relocation into extended memory is required or complex dynamic memory allocation systems are required.

3.2 MEMORY RELOCATION

A basic KT11-D function is to perform memory relocation and provide extended memory addressing capability for systems with more than 28K of physical memory. The KT11-D uses two sets of page address registers to relocate virtual addresses to physical addresses in memory. These sets are used as hardware relocation registers that permit several user's programs, each starting at virtual address 0, to simultaneously reside in physical memory.

3.2.1 Program Relocation

The page address registers are used to determine the starting address of each relocated program in physical memory. Figure 3-1 shows a simplified example of the relocation concept. A more detailed example of how memory relocation works is shown in Figure 3-3.

In Figure 3-1, Program A starting address 0 is relocated by a constant to provide physical address 6400_8.

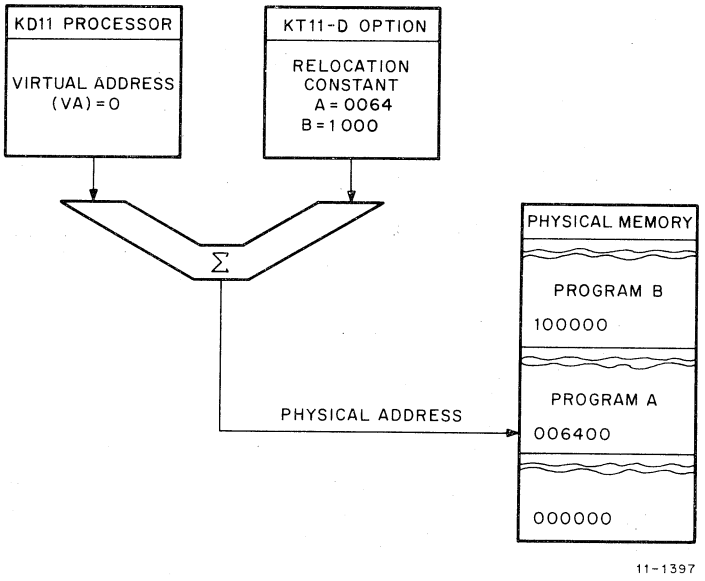


Figure 3-1 Simplified Memory Relocation Example

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

In the PDP-11 systems, a program is relocated in pages. A page can consist of from 1 to 128 blocks. Each block is 32 words in length. Thus, the maximum length of a page is 4096 (128 x 32) words. Using all of the eight available active page registers in a set, a maximum program length of 32,768 words can be accommodated. Each of the eight pages can be relocated anywhere in the physical memory, as long as each relocated page begins on a boundary that is a multiple of 32 words. However, for pages that are smaller than 4K words, only the memory actually allocated to the page may be accessed.

The relocation example shown in Figure 3-2 illustrates several points about memory relocation. These points are:

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

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

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

4. Each page is relocated independently. There is no reason why two or more pages could not be relocated to the same physical memory space. Using more than one page address register in the set to access the same space would be one way of providing different memory access rights to the same data, depending upon which part of a program was referencing that data. Further information about memory protection is provided in Paragraph 3.4. In the example shown in Figure 3-2, note the relocation constant assigned to pages 4 and 6. As a result, virtual addresses within both address ranges access the same physical addresses in memory, using separate page address registers.

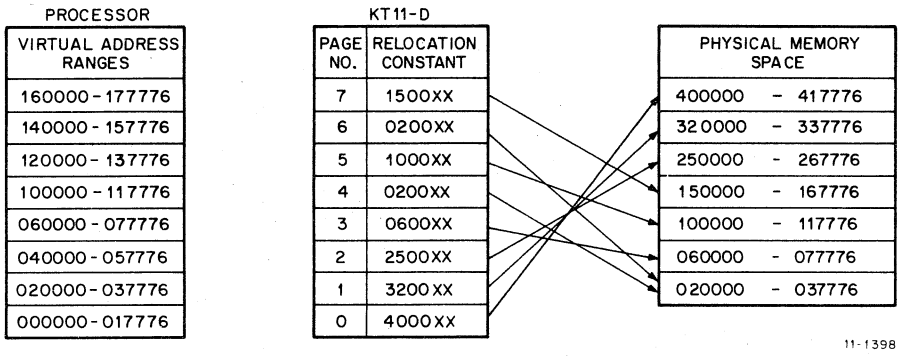


Figure 3-2 Relocation of a 32K Word Program
Into 124K Word Physical Memory

NOTE
Where xx is the address within the block number given by the PAR

3.2.2 Extended Memory Addressing

When the KT11-D Memory Management Unit option is added to the PDP-11 system, the 16-bit KD11 address output is no longer interpreted as the direct physical address of a device or a memory location. Instead, it is considered as a 16-bit virtual address that contains information to be used by the KT11-D to construct an 18-bit physical address.

Refer to Figure 3-3 which shows how the 18-bit physical address is constructured. Virtual address bits VA (15:13) are interpreted as an active page field (APF) to select one of eight active page registers in a set. Virtual address bits (12:06) provide the block number (0 to 127_{10}) within the page. VA (05:00) indicate the displacement within each 32-word block.

The PAR contains a page address field (PAF) that is written into the PAR under program control at the time the complete PAR/PDR set is defined for a program page. Consider the PAF as the base address of the page. The block number, VA (12:06) is added to the base address PAF (11:00) to provide the 12 most significant bits of the physical address. This, plus virtual address bits VA (05:00) (unchanged by relocation), forms the 18-bit physical address.

3.3 MEMORY MANAGEMENT

The following paragraphs describe some of the memory management tasks that might be required of an operating system. The KT11-D hardware features aid the system software in the performance of these tasks.

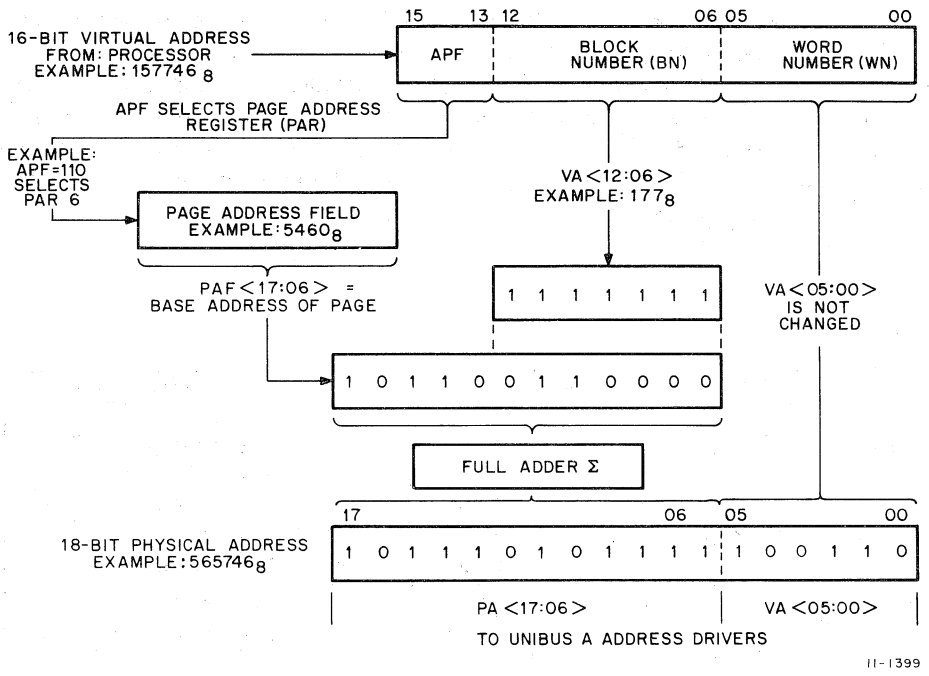


Figure 3-3 Construction of an 18-Bit Physical Address

3.3.1 Program Relocation

A timesharing system which swaps programs between physical memory and some backing store (such as an RF11 disk) can be efficiently implemented, because the programs need not be restricted to run in any particular memory locations.

When it is time to swap a program in, all that is required is that there be sufficient memory available in which to read the program and then set up the KT11-D to "relocate" the program so that it thinks it has been loaded at location 0.

3.3.2 Dynamic Memory Allocations

The KT11-D provides hardware-implemented features that enable the operating system to dynamically allocate memory upon demand, while a program is being run. These features are particularly useful when running higher-level language programs, such as ALGOL and PL/1, where, for example, arrays are constructed at execution time. No fixed space is reserved for them by the compiler. Lacking the dynamic memory allocation capability, the programmer must calculate and allow sufficient memory space to accommodate the worst case. This is time consuming and in many cases memory space is wasted.

Monitor primitives to allocate (de-allocate) memory, either by adding (deleting) a page or by increasing (decreasing) the size of an existing page, may be implemented. A running program can request, and subsequently return, memory used for a temporary buffer and the like, thus efficiently using the physical store and removing the worst case memory size restrictions.

To illustrate how dynamic memory allocation might be used, suppose an assembler initially requests enough memory for 64 symbols, say 256 words. If this symbol table overflows, then the assembler requests an additional 256 words, and so on, until a request is denied, at which time (and only at which time) an actual symbol table overflow has

occurred. This algorithm is clearly more efficient with respect to memory utilization than one which initially grabs a large chunk of memory for symbols.

NOTE

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

3.3.3 Memory Management Statistics

In a multiprogramming timeshared system, programs tend to be swapped between the main memory and some backing store, such as a disk. Needless to say, the performance of such a system is certainly dependent on the efficiency of the swapping algorithm.

Those portions of memory which have not been modified, (i.e., written into) since the last time they were swapped in, need not be swapped out when the space they occupy is required for something else. Instead, the memory can be used as is, because the copy of it on the backing store is still current. Thus, "half" the swapping time can be saved for those unmodified portions of main memory.

The KT11-D logic provides two mechanisms to help implement efficient swapping:

1. Pages may be designated read only. Such pages cannot be modified.
2. A flag, the "W-bit", is automatically set by hardware whenever a potentially writable page is actually written into.

3.3.4 Memory Management Instructions

Memory Management provides the ability to communicate between two spaces, as determined by the Extended Processor Status Word, PS(15–12). This capability is implemented by the addition of two unique instructions to the KD11-A Instruction repertoire:

MTPI – Move To Previous Space (0066 DD).

MFPI – Move from Previous Space (0065 SS).

These instructions are operational in a KD11-A system, otherwise an Illegal Instruction Trap will result on an attempted execution.

Memory Management does not have to be enabled (SRO bit 0 set) for interstack communications although relocation and protection will be disabled. If these two instructions are examined from a programmer's point of view, they appear somewhat complex. However, from a hardware viewpoint, it is evident that they are modified MOV instructions.

In investigating the memory management instructions, the following facts must be kept in mind:

1. There are two possible modes of operation:
   a. Kernel (Monitor)
   b. User.

2. The selection of mode is made by expanding and utilizing the Processor Status Word (PSW). The possible machine states specified by PS(15:12) are as follows:

PS(15:12)	Current Mode (CM)	Previous Mode (PM)
00 00	Kernel	Kernel
11 11	User	User
11 00	User	Kernel
00 11	Kernel	User

3. The MFPI and MTPI instructions are most likely to be used in current mode Kernel, previous mode User.

4. The current mode specifies the Page Relocation and Descriptor Register set used to convert the KD11-A virtual address to a KT11-D physical address.

Examining the MFPI instruction, the general instruction format is:

MFPI (0065 SS) i.e., OP CODE and a Source address field.

When MFPI SS is fetched, the KD11-A will transform and encode it into the following:

MFPI SS → MOV SS, – (SP)

It will send it back to the processor over the RD Bus, reclock it into the IR Register, and execute it in conjunction with Memory Management space selection logic.

The calculation of the Source address is done in current space. That is, any index word or indirect addresses used in the address calculation are fetched using the KT11-D Page Address Registers selected by the current mode bits of the PSW. The final fetch of the Source operand in which data is to be moved *from* is made in previous space, i.e., using the KT11-D Page Address Registers selected by the previous mode bits in the PSW. Note that if the Source field is mode 0 Register 6, the SP selected is made by the previous mode bits of the PSW. But with any other mode and R6, the SP selected is by the current mode bits of PSW; since in these cases, the register is part of the address calculation and is not the final operand. The Source operand is then pushed on the current mode stack.

Examining the MTPI DD instruction, it has the following general format:

MTPI DD (0066 DD) i.e., op code and a Destination address field.

A similar transformation to a MOV instruction is done to MTPI DD.

MTPI DD → MOV (SP) +, DD

The calculation of the Destination address is done in current space. That is, any index or indirect addresses used in the address calculation are fetched using the KT11-D Page Address Registers selected by the current mode bits of the PSW. The final fetch of the Destination operand in which data is to be stored, is made in previous space. Note that if the Destination field is mode 0 Register 6, the SP selected is made by the previous mode bits of the PSW. But with any other Destination mode and R6, the SP selected is made by the current mode bits of the PSW. Since in these cases, the register is part of the address calculation and is not the final operand, this instruction pops a word off the current stack determined by PS(15:14) and stores that word into a Destination address in previous space determined by PS(13:12).

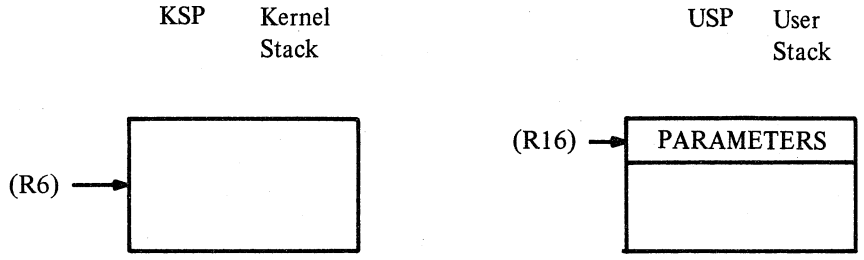
Thus, these instructions are used in memory managed systems to allow the exchange of data between the monitor (Kernel) and a user.

The following is an example of how Memory Management instructions can be used in an operating system. In advanced software systems, a user can not be allowed to handle his own I/O directly. The I/O address space is not available to a user (this is controlled by the contents of the UPAR's, which are set up by the monitor). Users initiate I/O requests to the monitor by means of a trap such as EMT.

Prior to the trap, the user pushes on to his stack (R16) certain parameters such as command, word count, and buffer address. The trap sequence sets up the PS(15:12) such that the current mode is Kernel (monitor) and the previous mode was User.

The Kernel must now retrieve the I/O parameters from the user core space, using the MFPI instruction.

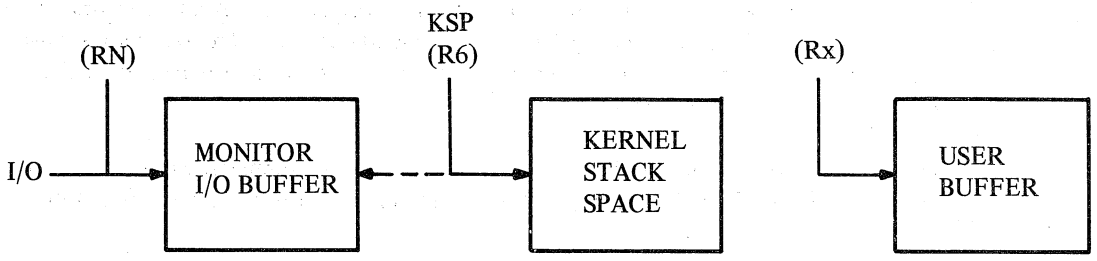
Example 1 – Retrieve parameter from User stack.




```
XX      /MOV #UPCNT, RA          ;STORE NUMBER OF USER
                                 ;PARAMETERS IN A REGISTER.

        /MFPI R6                 ;GET USER STACK POINTER (USP)
        CONVERTS TO MOV R16, - (R6)  ;ONTO THE KERNEL STACK.

        /MOV (R6) +, RN          ;PUT USP IN A REGISTER

LOOP    /MFPI (RN) +             ;PUT USER PARAMETERS
        CONVERTS TO MOV (RN) +, - (R6)  ;ONTO KERNEL STACK.

        /CMP RN, RA + RB         ;HAVE ALL PARAMETERS BEEN
                                 ;TRANSFERRED TO KERNEL.

        /BLT LOOP                ;IF NOT GET ANOTHER
                                 ;PARAMETER.
```

The monitor can now initiate the required command to the I/O. Although all devices have the capability of accessing all of physical core (meaning the ability to access user core), this practice usually is not preferred because the user is subject to being "swapped out" at any time. Therefore, data is normally placed into a monitor (Kernel) buffer. The monitor must then transfer that data to user core by means of the MTPI instruction.

Example 2 – Transferring data read from a device to user core.




```
XX      /MOV I/O BUFADR, RN              ;POINT RN TO I/O BUFADR.

        /MOV UBUFADR, RX                 ;POINT RX TO USER
                                         ;BUFFER.

LOOP    /MOV (RN) +, - (R6)              ;PUSH I/O DATA ONTO THE
                                         ;KERNEL STACK.

        /MTPI (RX) +                     ;MOV I/O DATA TO USER
        CONVERTS TO MOV (R6) +, (RX) +   ;BUFFER CORE.

        /CMP RN, I/O BUFADR + WORD CNT   ;HAS ALL I/O DATA
                                         ;BEEN TRANSFERRED
                                         ;TO USER CORE

        /BLT LOOP                        ;IF NOT GET ANOTHER
                                         ;DATA WORD.
```

Examples 1 and 2 do not reflect any user software system but are meant mearly as examples of the operation and uses of MFPI and MTPI.

3.4 MEMORY PROTECTION

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

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

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

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

The KT11-D option provides the hardware facilities to implement all of the above types of memory protection. The following paragraphs describe the memory protection features afforded by the KT11-D.

3.4.1 Inaccessible Memory

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

3.4.2 Read-Only Memory

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

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

3.4.3 Multiple Address Space

There are two completely separate PAR/PDR sets provided by the KT11-D: one set for Kernel mode and one set for User mode. This affords the timesharing system with another type of memory protection capability. The mode of operation is specified by the Processor Status Word current mode field, or previous mode field, as determined by the current instruction. (MTPI and MFPI are the two instructions that use previous mode.)

Assuming the current mode PSW bits are valid, the active page register sets are enabled as follows:

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

3.4.4 Mode Description

With memory management the modes of operation provide the following flexibility and restrictions:

In Kernel Mode, the operating program has unrestricted use of the machine except for the added time to a bus cycle created by the KT11-D Logic of a 150ns. The User also sees this delay plus the operating restrictions listed below:

1. Attempted execution of the instruction HALT traps as a Reserved Instruction via location 10;
2. Execution of a RESET instruction results in a no-operation execution of a NOP instruction (1.5μsec).

3. User Processor Status restrictions are as follows:

	USER RTI, RTT	USER TRAPS, INTERRUPTS	EXPLICIT PSW ACCESS
CC (3:0)	Loaded from Stack	Loaded from Vector	*
T (4)	Loaded from Stack	Loaded from Vector	Cannot be changed
PRIORITY (7:5)	Cannot be changed	Loaded from Vector	*
PREVIOUS (13:12)	Cannot be changed	Copied from PS (15:14)	*
Current (15:14)	Cannot be changed	Loaded from Vector	*

* = Explicit operations can be made if the Processor Status is mapped in User space.

4. Stack Limit Violations are disabled in User. Stack protection provided by memory protect features. Another difference between the two modes is the use of separate stack pointer registers:

   Kernel – KD11 Register 6 (R6)

   User – KD11 Register 16 (R16)

   On a trap or an interrupt, the vector is always referenced via Kernel space and the old PS and PC are pushed onto the stack determined by the new current mode of the PSW from the vector.

Thus, a User mode program is relocated by its own PAR/PDR set, as are Kernel programs. This makes it impossible for a program running in one mode to accidentally reference space allocated to another mode when the active page registers are set correctly. For example, a user cannot transfer to Kernel space. The Kernel mode address space may be reserved for resident system monitor functions, such as the basic Input/Output Control (IOC) routines, memory management trap handlers, and timesharing scheduling modules. By dividing the types of timesharing system programs functionally between the Kernel and User modes, a minimum amount of space control housekeeping is required as the timeshared operating system sequences from one user program to the next. For example, only the User PAR/PDR set needs to be updated as each new user program is serviced. The two PAR/PDR sets implemented in the KT11-D Memory Management Unit option are shown in Figure 3-4.

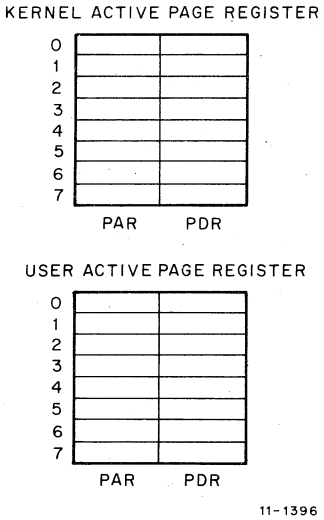


Figure 3-4 KT11-D Memory Management Unit Active Page Registers

3.5 PAR/PDR REGISTERS

The KT11-D Memory Management Unit provides two sets of eight PAR/PDR pairs. Figure 3-4 shows how the two sets are organized. Each pair consists of a Page Address Register (PAR) and a Page Descriptor Register (PDR). These registers are always used as a pair and contain all the information required to locate and describe the current active pages for each mode of operation. As indicated in Figure 3-4, one PAR/PDR set is used in Kernel mode and the other is used in User mode. The current mode bits (or in some cases, the previous mode bits) of the Processor Status Word determine which set will be referenced for each memory access. A program operating in one mode cannot use the PAR/PDR sets of the other mode to access memory. Thus, the two sets are a key feature in providing a full-protected environment for a timeshared multi-programming system.

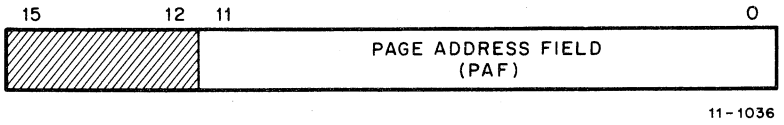


Figure 3-5 Page Address Register (PAR) Format

A specific processor I/O address is assigned to each PAR and PDR of each set. Table 3-1 is a complete list of address assignments.

NOTE
Unibus devices cannot access PARs or PDRs.

In a fully-protected multi-programming environment, the implication is that only a program operating in the Kernel mode would be allowed to write into the PAR and PDR locations for the purpose of mapping user's programs. However, there are no restraints imposed by the KT11-D logic that will prevent User mode programs from writing into these registers. The option of implementing such a feature in the operating system, and thus explicitly protecting these locations from user's programs, is available to the system software designer.

Table 3-1
PAR/PDR Address Assignments

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

3.5.1 Page Address Registers (PAR)

The Page Address Register (PAR), shown in Figure 3-5, contains the 12-bit Page Address Field (PAF) that specifies the base address of the page.

Bits (15:12) of the PAR are not implemented in the hardware.

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

3.5.2 Page Descriptor Registers (PDR)

The Page Descriptor Register (PDR), shown in Figure 3-6, contains information relative to page expansion, page length, and access control.

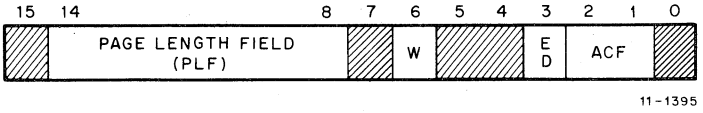


Figure 3-6 Page Descriptor Register (PDR) Format

3.5.2.1 Access Control Field (ACF) – This 2-bit field, ACF (02:01) of the PDR describes the access rights to this particular page. The access codes or "keys" specify the manner in which a page may be accessed and whether or not a given access should result in an abort of the current operation. A memory reference that causes an abort is not completed and is terminated immediately.

Aborts are caused by attempts to access non-resident pages, page length errors, or access violations, such as attempting to write into a read-only page. Traps are used as an aid in gathering memory management information.

In the context of access control, the term "write" is used to indicate the action of any instruction which modifies the contents of any addressable word. A "write" is synonymous with what is usually called a "store" or "modify" in many computer systems. Table 3-2 lists the ACF keys and their functions. The ACF is written into the PDR under program control.

Table 3-2
Access Control Field Keys

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

3.5.2.2 Expansion Direction (ED) – The ED bit located in PDR bit position 03 indicates the authorized direction in which the page can expand. A logic 0 in this bit (ED=0) indicates the page can expand upward from relative zero. A logic 1 in this bit (ED=1) indicates the page can expand downward toward relative zero. The ED bit is written into the PDR under program control. When the expansion direction is upward (ED=0), the page length is increased by adding blocks with higher relative addresses. Upward expansion is usually specified for program or data pages to add more program or table space. An example of page expansion upward is shown in Figure 3-7.

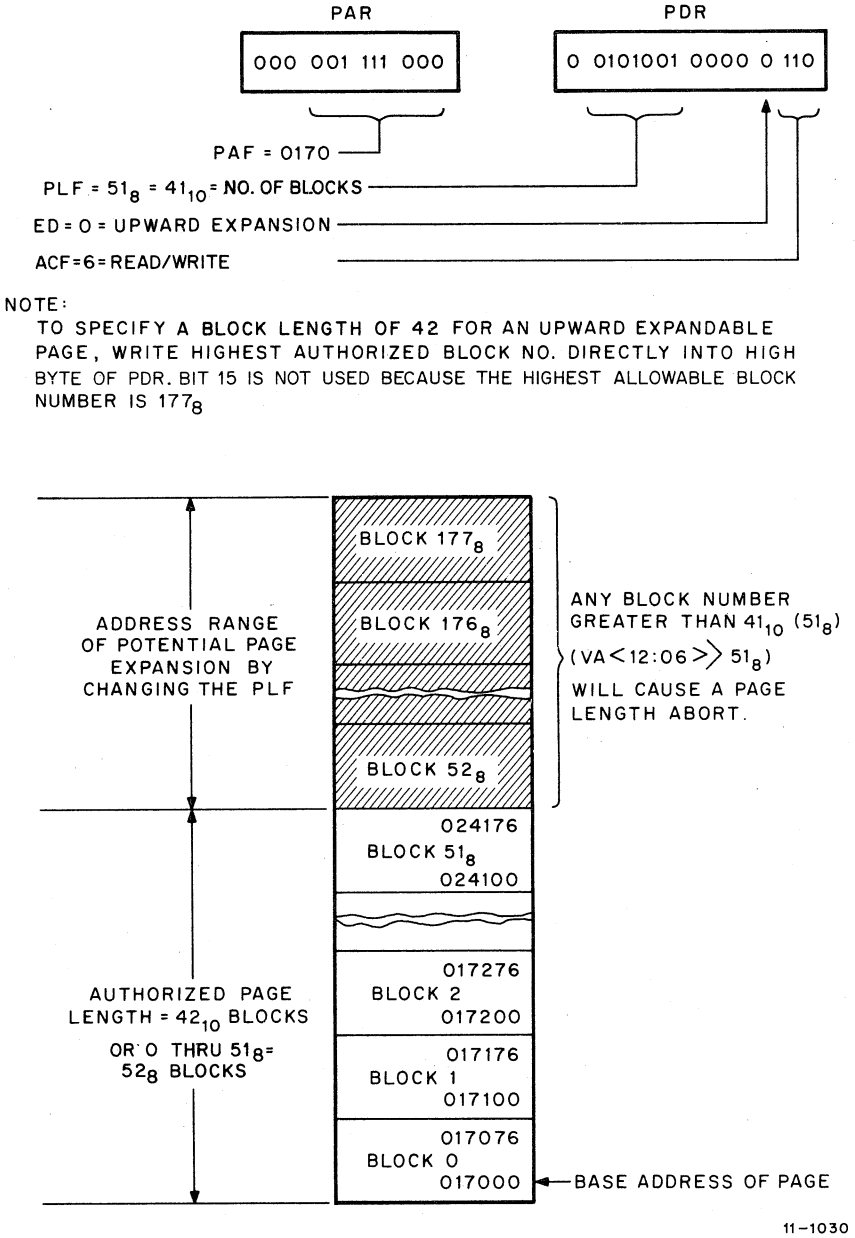


Figure 3-7 Example of an Upward Expandable Page

When the expansion direction is downward (ED=1), the page length is increased by adding blocks with lower relative addresses. Downward expansion is specified for stack pages so that more stack space can be added. An example of page expansion downward is shown in Figure 3-8.

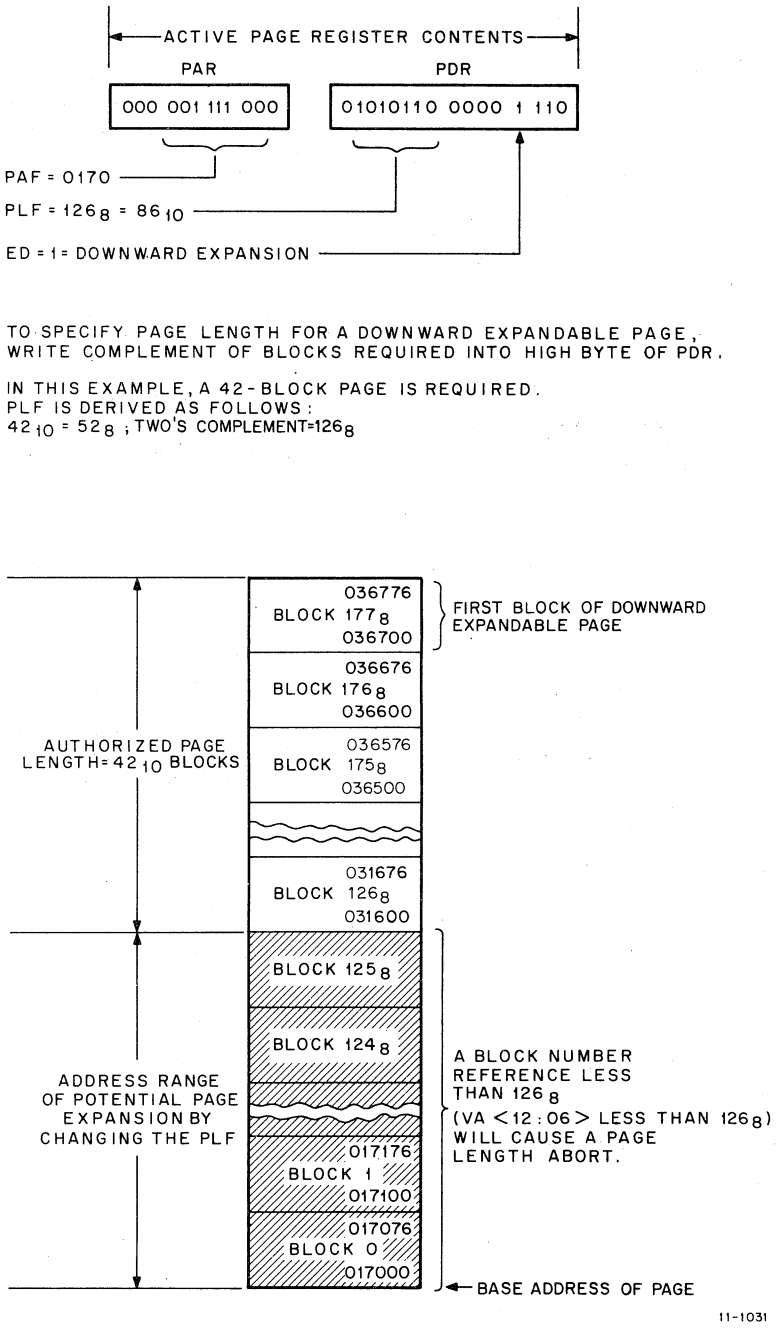


Figure 3-8 Example of a Downward Expandable Page

3.5.2.3 **Written Into (W)** – The W bit located in PDR bit position 06 indicates *whether the page has been written* into since it was loaded into memory. W=1 is affirmative. The W bit is automatically cleared when the PAR or PDR of that page is written into. It can only be set by KT11-D control logic.

In disk swapping and memory overlay applications, the W bit (bit 6) can be used to determine which pages in memory have been modified by a user. Those that have been written into must be saved in their current form. Those that have not been written into (W=0), need not be saved and can be overlayed with new pages, if necessary.

NOTE

The W bit cannot be set by a memory access of a KT11-D internal register (SR0) or a memory access that causes an abort.

3.5.2.4 **Page Length Field (PLF)** – The 7-bit PLF located in PDR bits (14:08) specifies the authorized length of the page, in 32-word blocks. The PLF holds block numbers from 0 to 177_8, thus allowing any page length from 1 to 128_{10} blocks. The PLF is written in the PDR under program control.

3.5.3 PLF for an Upward Expandable Page

When the page expands upward, the PLF must be set to one less than the intended number of blocks authorized for that page. For example, if 52_8 (42_{10}) blocks are authorized, the PLF is set to 51_8 (41_{10}) (Figure 3-7) block 0 being the page boundary and the first block of that page. The KT11-D hardware compares the virtual address block number, VA (12:06) with the PLF to determine if the virtual address is within the authorized page length.

When the virtual address block number is less than or equal to the PLF, the virtual address is within the authorized page length. If the virtual address is greater than the PLF, a page length fault (address too high) is detected by the hardware and an abort occurs. In this case, the virtual address space legal to the program is non-contiguous because the three most significant bits of the virtual address are used to select the PAR/PDR set.

3.5.4 PLF for a Downward Expandable Page

The capability of providing downward expansion for a page is intended specifically for those pages that are to be used as stacks. In the PDP-11, a stack starts at the highest location reserved for it and expands downward toward the lowest address as items are added to the stack. The first block of the downward expandable page being block 177_8.

When the page is to be downward expandable, the PLF must be set to authorize a page length, in blocks, that starts at the highest address of the page. That is always Block 177_8. Refer to Figure 3-8, which shows an example of a downward expandable page. A page length of 42_{10} blocks is arbitrarily chosen so that the example can be compared with the upward expandable example shown in Figure 3-7.

NOTE

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

The rationale for complementing the number of blocks required to obtain the PLF is as follows:

MAXIMUM BLOCK NO.	MINUS	REQUIRED LENGTH	EQUALS	PLF
177_8	–	52_8	=	125_8

3.6 MEMORY MANAGEMENT STATUS REGISTERS

Aborts generated by the KT11-D logic are vectored through Kernel space address location 250.

The KT11-D has three status registers of which two are functional; SR0, SR2 and SR1 responding with all zeros.

Status Register SR0 and SR2 can be referenced by fault recovery routines to determine why the abort occurred. The following paragraphs describe the formats of both status registers.

3.6.1 Status Register 0 (SR0)

SR0 contains abort error flags, memory management enable, plus other essential information required by an operating system to recover from an abort or service a memory management trap. The SR0 format is shown in Figure 3-9.

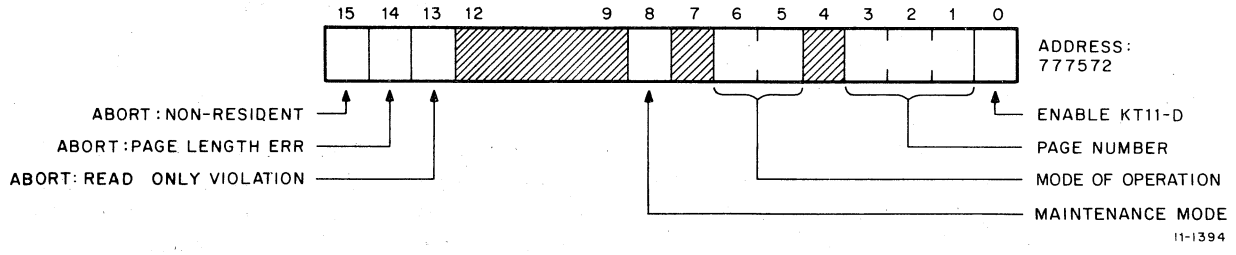


Figure 3-9 Format of Status Register 0 (SR0)

Bits (15:13) are the abort flags and are enabled when an address is being relocated by the KT11-D. This implies that either SR0, bit 0 is equal to 1 (KT11-D operating) or that SR0, bit 8, is equal to 1 and the memory reference is the final one of a destintion calculation (maintenance/destination mode).

NOTE

Bit 15, 14, or 13, when set (abort conditions) cause KT11-D logic to freeze the contents of SR0 bits 1-6 and status register SR2. This is done to determine the cause of the abort.

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

3.6.1.1 Abort-Nonresident – Bit 15 is the "Abort-Nonresident" bit. It is set by attempting to access a page with an access control field (ACF) key equal to 0 or 4 and setting PS (15:14) to an illegal mode.

3.6.1.2 Abort – Page Length – Bit 14 is the "Abort-Page Length" bit. It is set by attempting to access a location in a page with a block number (virtual address bits 12:06) that is outside the area authorized by the Page Length Field (PLF) of the PDR for that page.

3.6.1.3 Abort-Read Only – Bit 13 is the "Abort-Read Only" bit. It is set by attempting to write in a "Read-Only" page having an access key of 2.

NOTE

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

3.6.1.4 Maintenance/Destination Mode – Bit 8 specifies maintenance use of the memory management unit. It is used for KT11-D diagnostic purposes. For the instructions used in the initial diagnostic program, bit 8 is set so that only the final destination reference is relocated. It is useful to prove that the KT11-D is capable of relocating addresses.

3.6.1.5 Mode of Operation – Bits 5 and 6 indicate the CPU mode (User or Kernel) associated with the page causing the abort. (Kernel=00, User=11). These bits are controlled by the KT11-D logic that decodes current previous mode bits of the PSW.

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

3.6.1.7 Enable KT11-D – Bit 0 is the "Enable KT11-D" bit. When it is set to 1, all addresses are relocated and protected by the memory management unit. When bit 0 is set to 0, the memory management unit is disabled and addresses are neither relocated nor protected.

3.6.2 Status Register 2 (SR2)

SR2 is loaded with the 16-bit Virtual Address (VA) at the beginning of each instruction fetch but is not updated if the instruction fetch fails. SR2 is read only; a write attempt will not modify its contents. SR2 is the Virtual Address Program Counter (Figure 3-10). Upon an abort, the results of SR0 bits 15, 14, or 13 being set, SR2 will freeze until the SR0 abort flags are cleared.

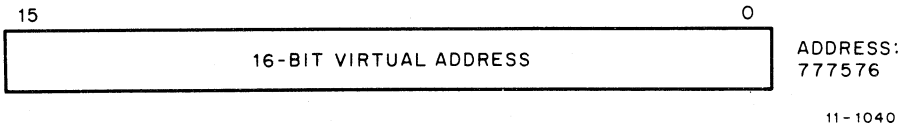


Figure 3-10 Format of Status Register 2 (SR2)

3.7 DETERMINING THE PROGRAM PHYSICAL ADDRESS

A 16-bit virtual address can specify up to 32K words, in the range from 0 to 177776_8 (word boundaries are even octal numbers). The three most significant virtual address bits designate the PAR/PDR set to be referenced during page address relocation. Table 3-3 lists the virtual address ranges that specify each of the PAR/PDR sets.

To calculate the physical address, disregard the three most significant VA bits and add the remainder to the PAR contents, right-shifted six places. Example:

```
VA = 167456 =      xxx0 111   100 101 110
+(PAR) = 3456 =    011  100 101 110
 PA = 355256 =     011  101 101  010 101 110
```

Where x indicates these bits are not used in the calculation.

Table 3-3
Relating Virtual Address to PAR/PDR Set

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

Reader's Comments

KT11-D MEMORY MANAGEMENT
OPTION USER'S MANUAL
EK-KT11D-OP-001

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 _____

CUT OUT ON DOTTED LINE

Fold Here

Do Not Tear - Fold Here and Staple

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

Digital Equipment Corporation
Technical Documentation Department
146 Main Street
Maynard, Massachusetts 01754

digital equipment corporation

Printed in U.S.A.