

**SECURE COMMUNICATIONS
PROCESSOR SPECIFICATION**

Honeywell Information Systems, Inc.
Federal Systems Division
7900 Westpark Drive
McLean, VA 22101

June 1976

Approved for Public Release;
Distribution Unlimited.

Prepared for

DEPUTY FOR COMMAND AND MANAGEMENT SYSTEMS
ELECTRONIC SYSTEMS DIVISION
HANSCOM AIR FORCE BASE, MA 01731

20100827220



LEGAL NOTICE

When U.S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

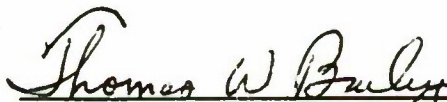
OTHER NOTICES

Do not return this copy. Retain or destroy.

This technical report has been reviewed and is approved for publication.



WILLIAM R. PRICE, Captain, USAF
Techniques Engineering Division



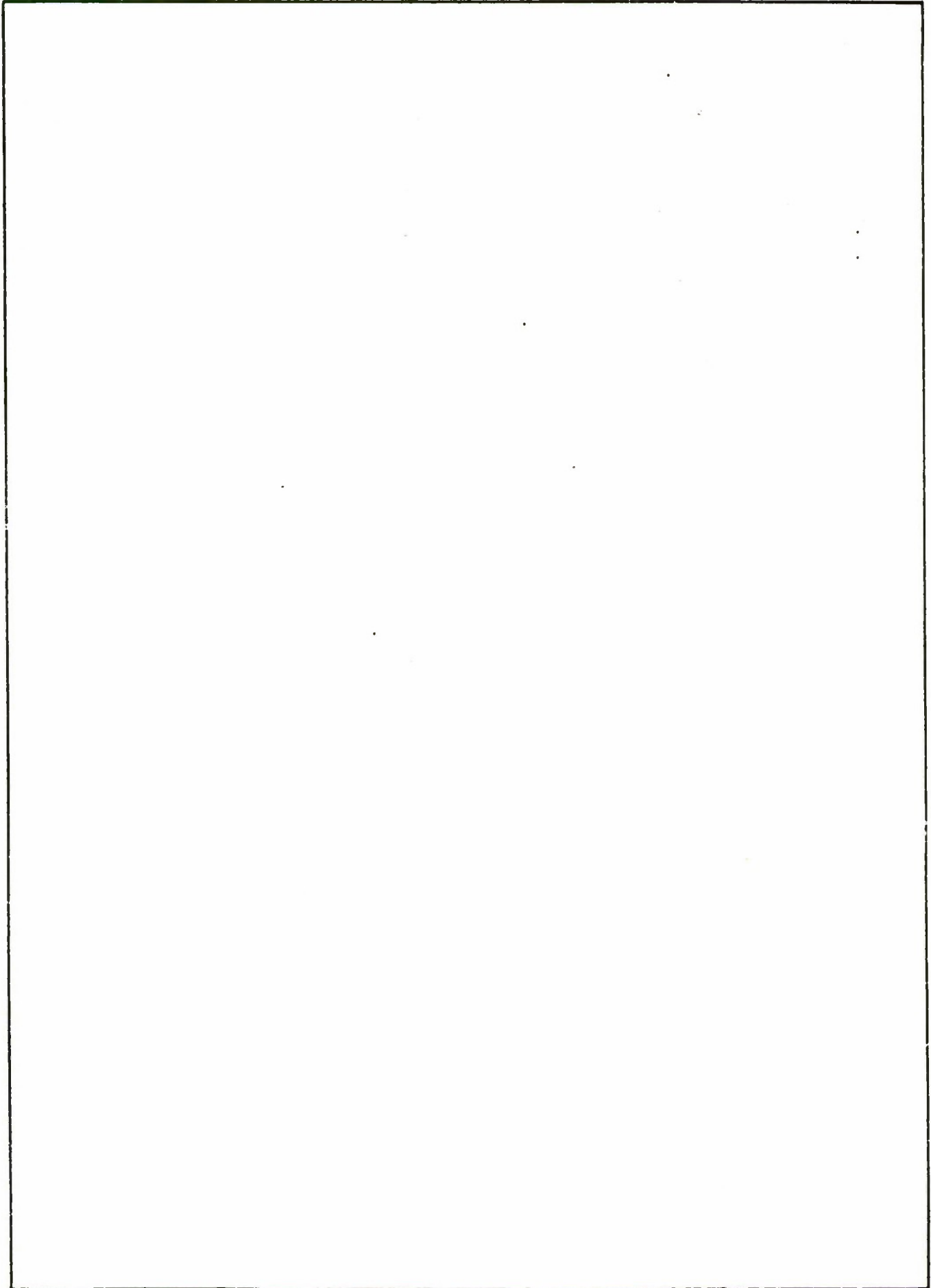
THOMAS W. BAILEY, Lt Col, USAF
Chief, Computer Security Group

FOR THE COMMANDER



STANLEY P. DERESKA, Colonel, USAF
Deputy Director, Computer
Systems Engineering
Deputy for Command & Management Systems

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ESD-TR-76-351, Vol. II	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SECURE COMMUNICATIONS PROCESSOR SPECIFICATION	5. TYPE OF REPORT & PERIOD COVERED	
	6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s) R. Broadbridge J. Mekota	8. CONTRACT OR GRANT NUMBER(s) FI9628-74-C-0205	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Honeywell Information Systems, Inc. Federal Systems Division 7900 Westpark Drive, McLean, VA 22101	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS CORL Item A004	
11. CONTROLLING OFFICE NAME AND ADDRESS Deputy for Command and Management Systems Electronic Systems Division Hanscom AFB, MA 01731	12. REPORT DATE June 1976	
	13. NUMBER OF PAGES 86 pages	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report) UNCLASSIFIED	
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
16. DISTRIBUTION STATEMENT (of this Report) Approved for Public Release; Distribution Unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Secure Communications Processor, Security Protection Module, Descriptor Structure, Ring Structure, Memory Management, I/O Mapping		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) A specification which establishes the performance, design, de- velopment, and test requirements for a Secure Communications Processor. A generic specification to be used is establishing lower level functional specifications. Requirements for hard- ware implementation of the functionality needed in an ADP system to operate in a multilevel secure environment.		



PREFACE

This document presents the specification of the performance, design, development, and test requirements for the Secure Communications Processor. This document was prepared by Mr. Robert Broadbridge of Honeywell Informations System, Inc.

Federal Systems Operations
7900 Westpark Drive
McLean, Virginia 22180

TABLE OF CONTENTS

1.0	Scope	4
1.1	Item Established	4
1.2	Approach	4
1.3	Conceptual Environment	4
2.0	Applicable Documents	7
3.0	Requirements	11
3.1	SPM Definition	11
3.1.1	SPM Functional Overview	11
3.1.2	SPM Interfaces	15
3.1.2.1	Processor to Memory Interface	15
3.1.2.1.1	Address Translation	15
3.1.2.1.2	Access Control	19
3.1.2.1.2.1	Effective Ring (Reff)	21
3.1.2.1.2.2	Argument Validation	22
3.1.2.1.3	Descriptors	23
3.1.2.1.3.1	Memory Descriptor	24
3.1.2.1.3.2	Descriptor Base Root	28
3.1.2.1.4	Descriptor Structure Dynamics	30
3.1.2.1.5	Fast Access Descriptor Store (FADS)	31
3.1.2.1.6	Cross Ring Movements	31
3.1.2.1.6.1	Call and Return	32
3.1.2.1.6.2	Trap and Trap Return	34
3.1.2.2	Device to Memory Interface	36
3.1.2.2.1	I/O Flow	36
3.1.2.2.1.1	Premapped I/O Flow	37
3.1.2.2.1.2	Mapped I/O Flow	37
3.1.2.3	Processor to Device Interface	40
3.1.2.3.1	I/O Naming Structure	40
3.1.2.3.1.1	Explicit Names	43
3.1.2.3.1.2	Devices in Memory	43
3.1.2.3.2	I/O Descriptor	43
3.1.2.3.2.1	Explicit Names Descriptor	44
3.1.2.3.2.2	Devices as Memory Descriptor	47
3.1.2.3.3	I/O Initiation	48
3.1.2.3.3.1	Premapped Initiation	48
3.1.2.3.3.2	Mapped Initiation	49
3.1.2.4	Device to Processor Interface	50
3.1.2.4.1	Interrupt Structure	51
3.1.2.4.2	Interrupt Storage and Entry	51
3.1.2.5	Processor to Processor Interface	51
3.1.2.5.1	Maintenance of Descriptor Structure	51
3.1.2.5.2	General Interprocessor Signalling	52

3.1.2.6	Operator to Processor Interface	52
3.1.2.6.1	Standalone Bootstrap	52
3.1.2.6.2	Front End Bootstrap	54
3.1.3	Major Component List	54
3.1.4	Government Furnished Property List	55
3.1.5	Government Loaned Property List	55
3.2	Characteristics	55
3.2.1	Performance	55
3.2.1.1	Relative Performance	55
3.2.1.2	Data Cache	55
3.2.2	Physical Characteristics	56
3.2.3	Reliability	57
3.2.4	Maintainability	58
3.2.5	Environmental Conditions	58
3.3	Design and Construction	59
3.3.1	Materials Processes and Parts	59
3.3.2	Compromising Emanations Control (TEMPEST) and Electromagnetic Compatibility	59
3.3.3	Nameplates and Product Marking	60
3.3.4	Workmanship	60
3.3.5	Interchangeability	60
3.3.6	System Safety	60
3.3.7	Human Engineering	61
3.4	Documentation	61
3.5	Logistics	61
3.6	Personnel and Training	61
3.6.1	Training	61
3.7	Major Component Characteristics	61
3.7.1	SPM	62
3.7.2	Processor	62
3.7.3	Memory	63
3.7.4	Controllers	63
3.8	Precedence	64
4.0	Quality Assurance Provisions	65
4.1	General	65
4.1.1	Responsibility for Tests	65
4.1.2	Special Tests	65
4.2	Quality Conformance	65
4.2.1	Interface Definitions	66
4.2.2	Throughput	66
4.2.3	Physical Characteristics	66
4.2.4	Environmental Conditions	66
5.0	Preparation for Delivery	67

6.0	Notes	68
6.1	Descriptor Format and Field Encoding	68
6.2	Virtual Device Addresses	68
6.3	Virtual Address Pointers	69
6.4	Process/Event Reporting	71
6.5	Simple Indirect Descriptor	71
6.6	Page Descriptors	72
6.7	Physical Address Size and Precision	74
6.8	SPM Algorithmic Description	75
10.0	Appendix	86
10.1	Interconnection Unit	86
10.1.1	Intersystem Geometry	86
10.1.2	Internal Geometry	86
10.1.3	Interprocessor Communication	86

1.0 Scope

1.1 Item Established

This specification establishes the performance, design, development and test requirement for the secure data communications system prime item. This is a generic specification for use in establishing requirements for specific functional specifications for individual applications.

1.2 Approach

A secure communications system is achieved through the interconnection of the elements of a commercial minicomputer system by a Security Protection Module (SPM). The central position of the SPM is shown in Figure 1. This document specifies the required properties of both the SPM and the commercial units. The secure communications system shall meet the hardware requirements stated in DoD 5200.28-M.

1.3 Conceptual Environment

While the Software concepts underlying the design of a security kernel are beyond the scope of this document, a brief summary of some central notions will facilitate the understanding of this document (ref. section 2.0, Schiller).

A process is defined to consist of a collection of resources and a state. The state of a process includes a single execution point. A resource is available to a process only through a descriptor.

The resources of a process are subdivided into partitions. Each partition defines a domain. The single execution point of the process may be in any one domain. The domains of a process are well-ordered based on access privilege. This domain organization is termed a ring structure (ref. section 2.0, Schroeder and Saltzer). When operating within a ring, the process has access to the resources of that ring and of all rings of less privilege. The most privileged ring (ring zero, or innermost) will normally contain the security kernel, although the kernel may be constructed to occupy more than one ring (e.g., zero and one). Each process operating on a system will contain the security kernel (distributed security kernel), so that a switch to a security kernel function will occur within a process.

1A-48, 147

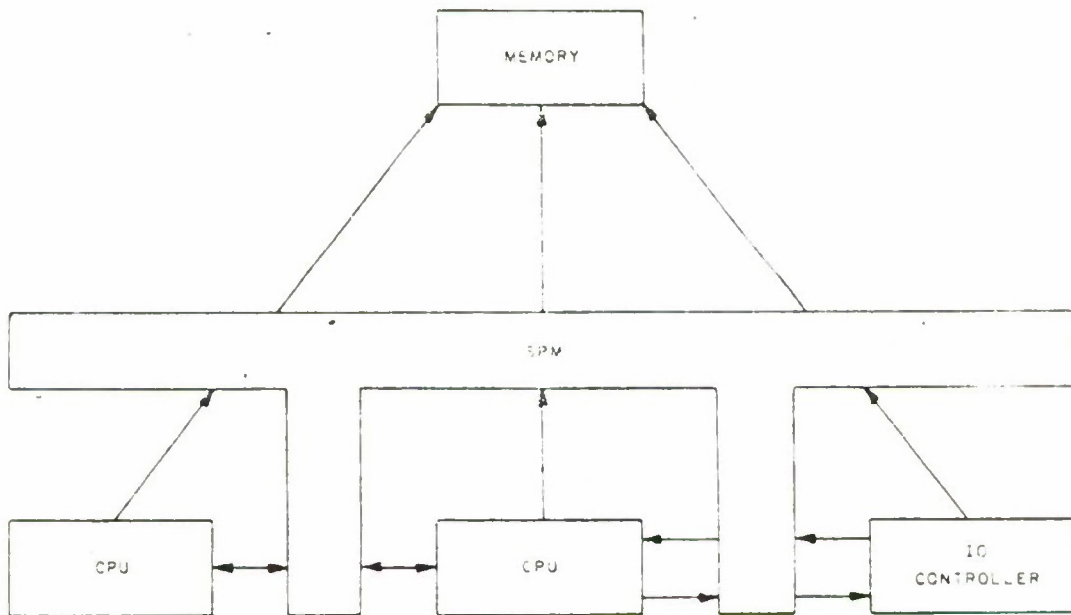


Figure 1. SPM Central Position

The access privilege of a process is controlled through a comparison within the SPM of an effective ring number generated by the SPM during address formation and pertinent access control fields contained in the descriptor controlling access to the resource. A definition of the access control rules is contained in sections 3.1.2.1.3.1, Memory Descriptor, and 3.1.2.3.2, I/O Descriptor. A specification of how the descriptor controlling access is found is contained in sections 3.1.2.1.1, Address Translation, and 3.1.2.3.1, I/O Naming Structure.

2.0 Applicable Documents

It is considered that the Secure Communications Processor will have DOD tri-services applicability, and thus the prime governing specification for communications with its respective latest revision and class will be one of the following:

MIL-E-4158E	Class 1,2 & 3	General Requirements for Ground Electronic Equipment
MIL-E-16400	Class 3 & 4	General Specification for Electronic Equipment, Ship and Shore
MIL-E-21200	Class 2 & 3	General Specification for Control Systems Electronic Test Equipment
MIL-E-5400	Class 1	General Specification for Aircraft Electronic Equipment

In addition, the following related disciplinary standards and specifications are representative of minimal requirements:

Military Specifications

MIL-P-9042G	6 Jun 72	Packaging, Handling, and Transportability in system/equipment acquisition.
MIL-M-7793D	31 Dec 69	Meter, Time Totalizing
MIL-E-5087B	15 Oct 64	Bonding, Electrical and Lightning Protection for Aerospace Systems
MIL-C-38999C	31 Jan 72	Connector, Electrical, Circular, Miniature; High density quick disconnect
MIL-C-8872313	21 Dec 73	Connector, Electrical Circular, Environment resisting
MIL-E-8189G	2 Jul 73	Electronic Equipment, Missiles, Boosters, Allied Vehicles
MIL-H-46855A	2 May 72	Human Engineering Requirements for Military Systems, Equipment and Facilities

MIL-C-1472E 31 Dec 74

Human Engineering Design Criteria for
Military Systems, Equipment and
Facilities

MIL-Standards

MIL-STD-188-100 15 Nov 72

Common Long Haul and Tactical
Comm. Sys. Tech. Std.

MIL-STD-280A 7 Jul 69

Definitions of Item Levels, Item
Exchangeability, Models and Related
Terms

MIL-STD-454D 31 Aug 73

Standard, General Requirements for
Electronic Equipment

MIL-STD-461A 1 Aug 68

Electromagnetic Interference
Characteristics, Requirements for
Equipment

& Chg. Notice 3 1 May 70

MIL-STD-462 31 Jul 67

Electromagnetic Interference
Characteristics, Measurement of

& Chg. Notice 1 1 May 70

MIL-STD-883 1 May 68

Test Methods and Procedures for
Micro-Electronics

MIL-STD-470 21 Mar 66

Maintainability Program Requirements
for Sys & Equipments

MIL-STD-471A 27 Mar 73

Maintainability/Verification
Demonstration/Evaluation

MIL-STD-483(USAF) 31 Dec 70

Configuration management practices
for systems, equipment, munitions,
and computer programming.

MIL-STD-490 1 Feb 69

Military standard specification
practices.

MIL-STD-756A 15 May 63

Reliability Prediction

MIL-STD-781B 15 Nov 67

Reliability Tests, Exponential
Distribution

MIL-STD-810B	15 Jun 67	Environmental Test Methods
& Chg Notice 1	20 Oct 69	
MIL-STD-1130	12 Nov 68	Connections, Electrical Solderless Wrapped
MIL-STD-1472A	15 May 70	Human Eng. Design Criteria for Military
MIL-STD-1521	1 Sep 72	Technical Reviews and Audits for Systems, Equipment and Computer Programs Systems, Equipment and Facilities
MIL-STD-3100.35G	15 Oct 68	Integrated Logistics Requirements Planning Guide for DoD Systems and Equipment
MIL-Q-9858A	16 Dec 63	Quality Program Requirements
<u>Manuals</u>		
DoD 5200.28 - M	2 Jan 73	Techniques and procedures for implementation, deactivation, testing, and evaluation of secure resource sharing ADP systems.
<u>Handbooks</u>		
MIL-HDBK-217B	20 Sep 74	Reliability Stress & Failure Rate Data for Electronic Equipment
MIL-HDBK-232(C)	14 Nov 72	DCS Red/Black Engineering and Installation Guidelines (U)
AFSC Design Handbook 1-3	1 Jan 72	Personnel Subsystems
<u>NSA Documents</u>		
NACSEM 5100(C)	Oct 70	Compromising Emanations Laboratory Test Standard Electromagnetic (U)
NACSEM 5200(S)		Compromising Emanations Design Handbook (U)

AF Commsec Publications

AFNAG-5B (C)	Mar 74	Red and Black Engineering and Installation Criteria
AFNAG-9A (C)	14 Apr 72	Using NACSEM Documents and TEMPEST Emanation Limits (U)

Army

DDC AD619666	1 Aug 64	Interference Reduction Guide for Design Eng Vol. 1
DDC AD619667	1 Aug 64	Interference Reduction Guide for Design Eng Vol 2

DASA

DASA 2028		Tree Preferred Procedures
-----------	--	---------------------------

OTHER

AFR 300-8	3 Jan 74	Security Requirements for Automatic Data Processing Systems (ADPS)
DIA-M-50-4A(C)	14 Jan 75	Security of Compartmental Computer Operations
Schiller, W. L.	Mar 75	The Design and Specification of a Security Kernel for the PDP-11/45, MTR-2934, MITRE Corporation, Bedford, MA.
Schroeder, M. D. & Saltzer, J. H.	Mar 72	"A Hardware Architecture for Implementing Protection Rings", <u>Communications of the ACM</u> 15(3), pp. 157-170.
Honeywell, AG95	Jun 72	<u>The Multics Virtual Memory</u>

3.0 Requirements

3.1 SPM Definition

3.1.1 SPM Functional Overview

The function of an SPM is to mediate, through a descriptor structure, all interactions between elements of a protected minicomputer. The logical structure that the introduction of an SPM imposes on the protected minicomputer is diagrammed in Figure 2. An SPM is intimately associated (for purposes of SPM control) with each processor of the system. Through its SPM, each processor may communicate with the other processors, I/O devices, and memory. An I/O device communicates to memory through an SPM. Thus, each SPM may be thought of as an address translation resource for a number of requesters, the requesters being the attached processors and I/O devices. The address translation operation is the conversion of virtual addresses presented by the requesters, via the descriptor structure, to absolute resource addresses (using information contained in the descriptors).

Each SPM logically contains the mechanism diagrammed in Figure 3. It contains the following functions:

- 1) the current protection state (current and effective ring) of each requester it services;
- 2) a pointer (Descriptor Base Root) to the set of descriptors which describe the accessible resources for each requester;
- 3) a mechanism by which the protection state and set of resource descriptors may be initialized for each requester: this mechanism is generally under the control of the associated processor;
- 4) a mechanism by which the SPM may search through the descriptor structure to locate the proper descriptor applying to a requested resource;
- 5) a mechanism by which the SPM may evaluate the propriety of a requested access based on the following information: the identity of the requester, the access mode of the request, the resource requested, the current protection state of the SPM for the requester, and the requester's descriptor for the resource;

IA-48,149

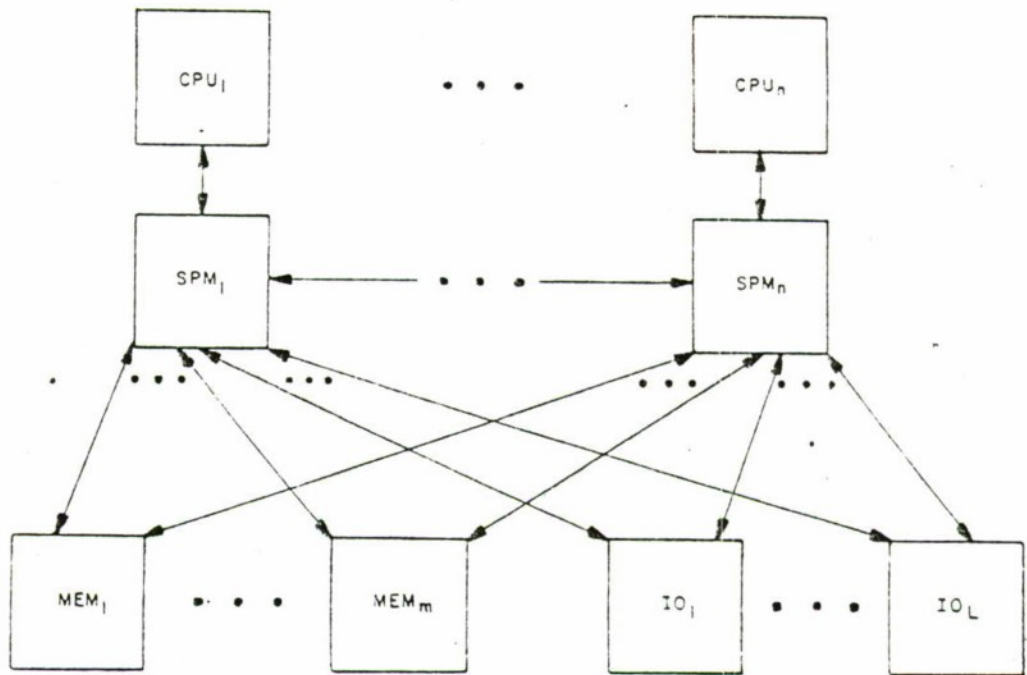


Figure 2. General System Structure

IA-48,164

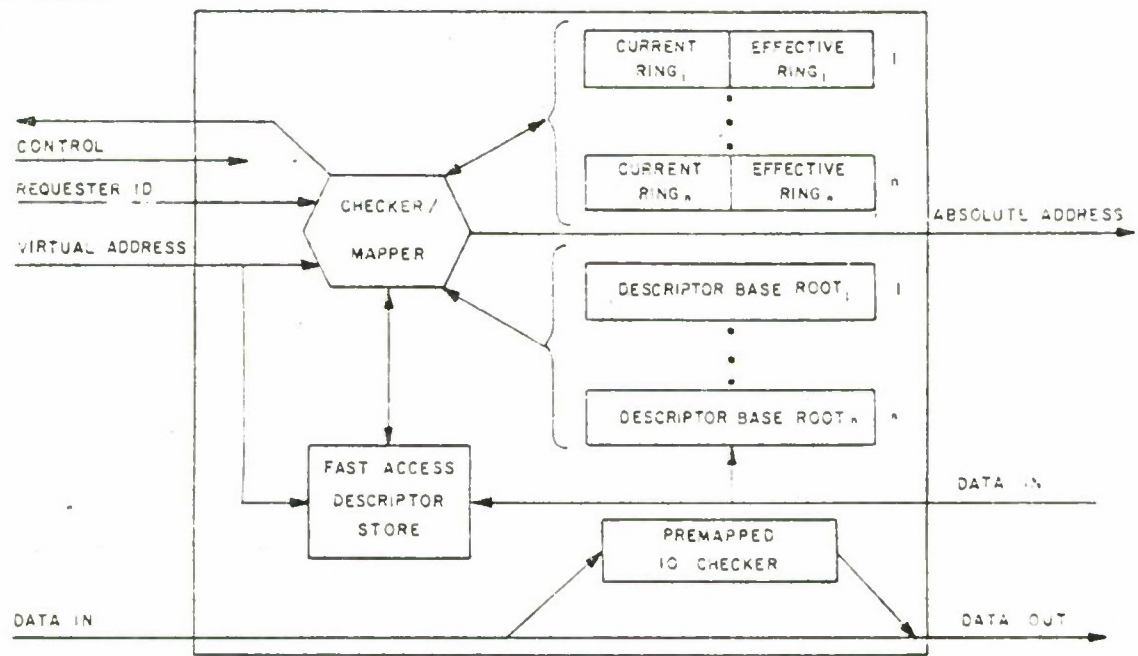


Figure 3. SPM Function

- 6) a mechanism by which the protection state of a requester may be changed, in a well-defined manner; and
- 7) a Fast Access Descriptor Store (FADS) in which the SPM may place fast access copies of recently referenced descriptors.

The SPM shall have sufficient protected storage to remember the protection state and descriptor set of each of the requesters (processors and I/O devices) that it services. In Figure 3 the number of requesters for a specific SPM is designated by n.

The SPM shall mediate each request by a processor to:

- 1) reference memory; and
- 2) initiate an I/O operation.

The SPM shall mediate each request by an I/O device to reference memory.

Most minicomputer systems utilize a bus architecture to transmit information between elements of the system. Thus, for an SPM implementation for such systems, the information flows of Figure 2 will occur on one or more busses.

The above discussion assumes that each memory request, by an I/O device, is actively mediated by the SPM. Due to performance difficulties imposed by the bus architecture of most minicomputers, the SPM's active mediation may cause an unacceptable performance loss (particularly in high I/O bandwidth applications). Thus an alternative form of I/O mediation is specified in this document. This architecture, termed premapped I/O, imposes substantially more responsibility and complexity on the controlling system software. Thus its use in secure systems must be carefully considered. Premapped I/O mediation imposes a "one-time" check of the propriety of an I/O device's memory requests. This checking, equivalent to the dynamic checking discussed above, is performed at I/O initiation time. The virtual memory address and extent to or from which the I/O device is to transfer data, is transmitted to the SPM which interprets the addresses in the descriptor structure of the requesting process. These addresses, if valid, are then translated into absolute addresses and transmitted to the device. The device must be guaranteed not to modify the addresses passed to it. The SPM must guarantee that the processor does not modify the set of descriptors used in the translation until the completion of the I/O operation. If an SPM supports premapped I/O it shall contain a premapped I/O checker that shall validate and translate addresses passed to devices with respect to the descriptors of the process initiating the I/O operation.

3.1.2 SPM Interfaces

The SPM enforces security through mediation of all communication between the non-secure hardware components. The interfaces are:

- processor to memory
- device to memory
- processor to device
- device to processor
- processor to processor

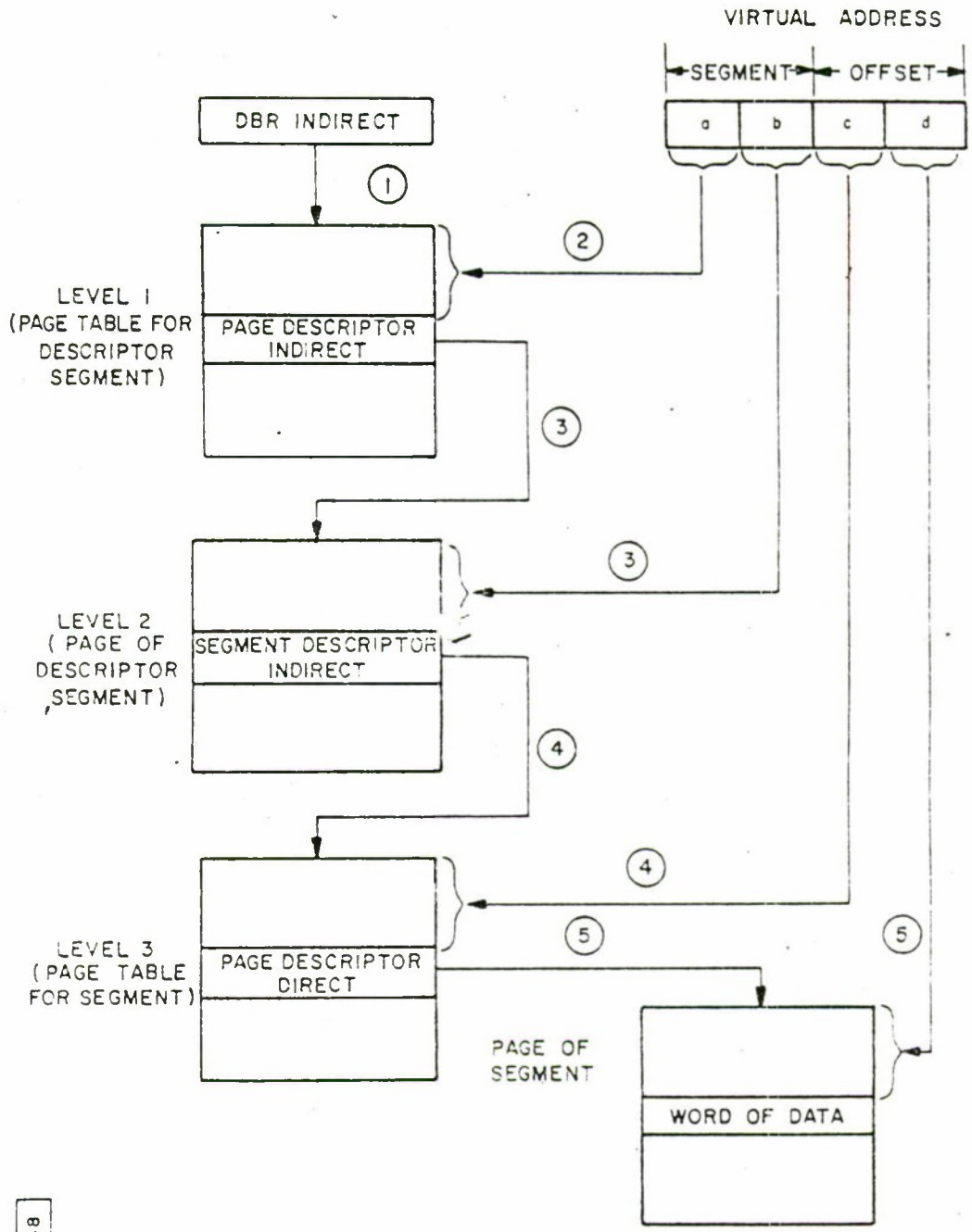
3.1.2.1 Processor to Memory Interface

3.1.2.1.1 Address Translation

The SPM shall mediate all processor to memory references. When the processor makes a memory reference, the memory address presented on the bus is intercepted by the SPM and is treated as a virtual address. The SPM translates this virtual address into a physical memory address through a series of lookups in descriptor tables resident in memory. The physical address is then presented to memory, and the appropriate read or write access is made. The data going to or from memory is not examined by the SPM.

Each memory descriptor in the descriptor tables contains, among various control fields (see section 3.1.2.1.3.1), a pointer to an absolute memory location (i.e., a physical memory address). There are several types of descriptors, as designated by particular encodings in the descriptor control fields. If the descriptor is indirect, the descriptor's pointer is the address of another descriptor table. If the descriptor is direct, the object described is either an area of memory or an I/O device. If an area of memory, the descriptor's pointer is the address of a area (page or segment) of data to be referenced. This section will discuss in detail indirect descriptors and direct memory descriptors. See 3.1.2.3.2 for a discussion of I/O descriptors.

The virtual address presented by the processor can, in the general case, be considered to consist of four fields, designated a,b,c,d, as shown at the top of Figure 4. There are several ways in which a virtual address can be translated into a physical address, depending on the types of descriptors encountered. The steps below, as illustrated in the figure, describe the procedure by which a virtual address shall be translated for the specific case of two levels of indirect descriptors and a third level of direct descriptor.



IA - 48, 158

Figure 4. Address Translation

- (1) The SPM, given a virtual address, makes its first reference to the first level descriptor table pointed to by the descriptor base root (DBR) known to the SPM (see 3.1.2.1.3.2 for a discussion of the DBR). It is assumed that the T field of the DBR specifies an indirect DBR.
- (2) The offset into this descriptor table is the first field of the virtual address (a), and the descriptor at that location is referenced.
- (3) If the descriptor is an indirect descriptor, the pointer in that descriptor is used to access a second descriptor table, and the second part of the virtual address (b) is used as an offset into this second table.
- (4) If the second level descriptor is indirect, it similarly is used to access a third descriptor table and the third part of the virtual address (c) is used to get the third level descriptor.
- (5) The third level descriptor must be a direct descriptor. Its pointer is used to find the page of data, and the last part of the virtual address (d) is an offset into the page to obtain the actual word being referenced.

The above steps describe the address translation in the case where the DBR is indirect and the descriptors in the first two levels are indirect. If either the first or second level descriptors are direct, their respective pointers shall be used to directly access the defined area of data, and the unused parts of the virtual address (either fields b,c,d or c,d) shall be combined to form the offset into the defined area.

The three-level descriptor system is the most general in that it allows for the implementation of segments, pages, and paged descriptor segments. The first descriptor table can be considered to be the page table of the descriptor segment, the second table is a page of the descriptor segment, and the third table is the page table for the segment. The indirect descriptors in the descriptor segment are called segment descriptors and the direct descriptors in the page tables are called page descriptors. The terms segment, page, and descriptor segment will be used in this specification to refer to the appropriate level descriptors and pages. It is not the intention, however, to preclude some other use or structuring of descriptors. The only requirement is that the address translation mechanism perform as specified.

A process's view of memory is that of a series of segments, each identified by a segment number (composed of fields a,b combined).

Within each segment there is a word offset (composed of fields c,d). Since each segment may not be the maximum size, there will be "holes" in the virtual address space for high values of the word offset (c,d) for some segments. Within a segment, however, all values of the word offset from 0 to the current size of the segment are usually defined. (The "end" of a segment is the offset corresponding to the last word of the last page of the segment as defined by the page table size contained in the segment descriptor. An "undefined" offset in a segment that occurs before the end of the segment is one that corresponds to a location in that segment for which there is no page descriptor in the page table. A "missing" page descriptor in a page table must be marked by software using the directed trap (DT) field in the descriptor, so that the SPM will not attempt to continue address translation using the contents of that descriptor. Though there may be little real use for having undefined offsets before the end of a segment, this specification does not require all page descriptors in a segment to point to valid memory addresses as long as the DT field is appropriately marked.)

Depending on the system software structure, the user can in general ignore the fact that segments and descriptor segments are paged. Thus, the breakdown of segment number and word offset into separate fields is of no concern. Only the lowest level software, i.e. the kernel, need be aware of paging. It is of course required that the kernel be in complete control of all descriptors.

In a multiprocessing environment, each process usually has its own virtual address space as characterized by its descriptor segment. In such an environment, and particularly in a communications processor where many processes have common data bases, it is essential that sharing of segments between processes be supported. When a segment is shared between processes, there is no inherent requirement that the segment have the same segment number for each active process that currently has access to the segment. Thus, the segment number portion of a given virtual address only has meaning from within the process using the descriptor segment to which the virtual address applies. However, the word offset portion of the address of a particular word in a segment is the same for every process that accesses the same word in the segment. This means a segment's page table can be shared among all processes currently accessing the segment. In fact, memory management in a demand paging environment is greatly simplified if the page descriptors (i.e., those that describe the actual physical resource) are shared (ref. section 2.0, The Multics Virtual Memory). The requirement for sharing of direct descriptors in different virtual address spaces is important because it dictates the manner in which access control is interpreted (see 3.1.2.1.2).

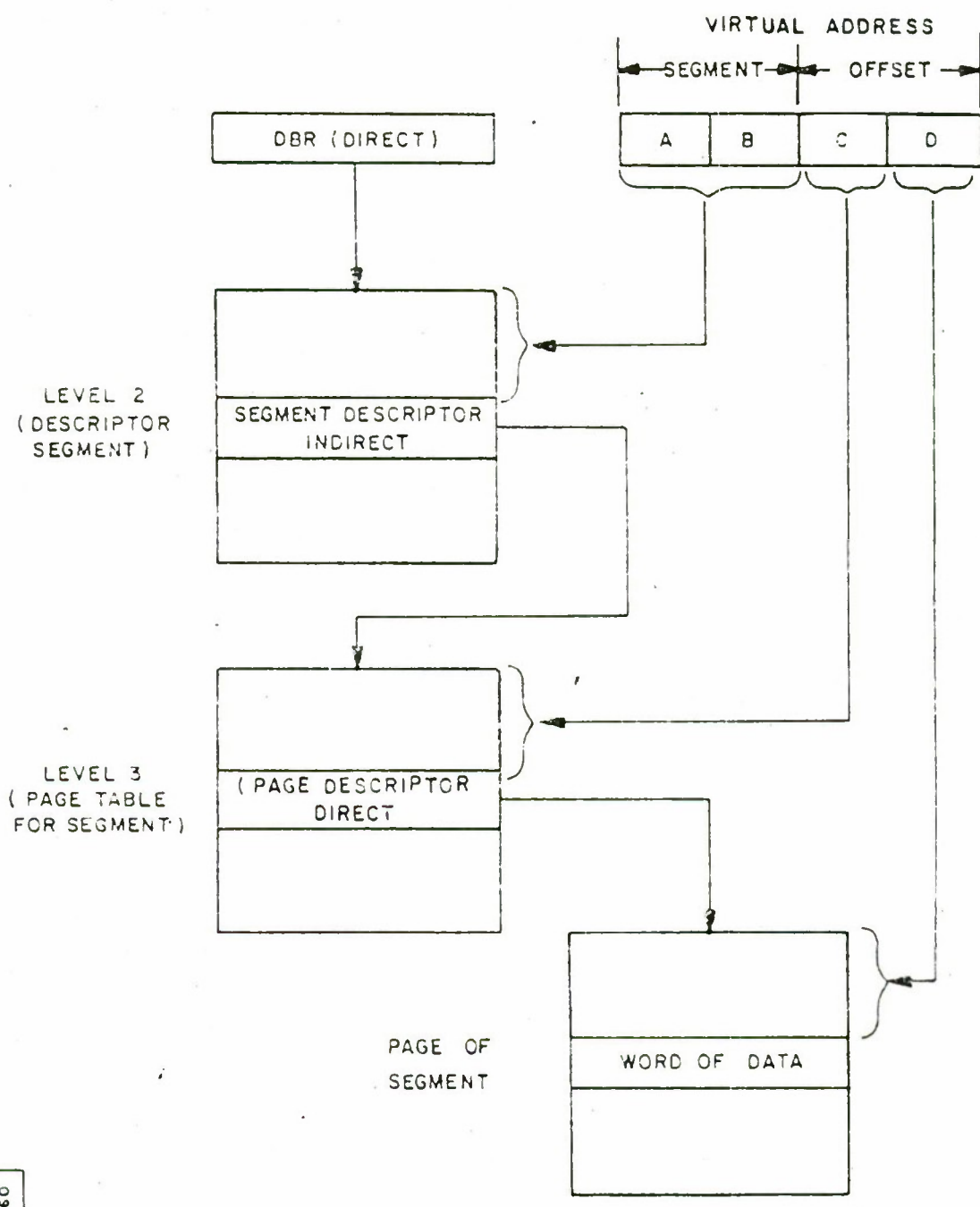
One final variation on the address interpretation shall be implemented to allow unpagged descriptor segments. If, in a given application, it is determined that a process's descriptor segment will be no greater than one page, or that it is not necessary to page descriptor segments, it is useful to specify that the descriptor segment is not pagged. This means that the segment descriptor must be directly accessed by the offset specified in the combined a,b field (see Figure 5). In this case, the DBR shall point directly to the second level descriptor table, and the combined a,b field shall be used to index into this table. The T field in the DBR shall specify this form of DBR interpretation (see 3.1.2.1.3.2).

The number of bits in each of the four fields of the virtual address depends on the particular application, since page size, segment size, and usable virtual address space are affected. For example, if the size of the combined a,b field is small, and c,d is large, a process can only access a small number of large segments. For some applications, this may leave a large amount of unusable space at the end of each segment, resulting in insufficient usable virtual address space. The use of a large number of small segments, on the other hand, may degrade system performance due to the large amount of memory required for segment descriptors that can't be shared. Generally, the relative sizes of the c and d fields affect paging overhead and memory management, and the segment size affects the granularity of access control (see next section) in terms of the size of the atomic unit of protection.

Though the field sizes cannot be exactly specified, there are certain restrictions in the relative field sizes that are required to simplify interpretation. For example, the size of the a,b field should be no larger than c,d -- otherwise it would be very difficult to access the descriptor segment itself as a segment. The decision as to how to divide the fields shall be justified with respect to the particular application.

3.1.2.1.2 Access Control

In addition to performing the function of address translation, the SPM shall verify that the process has the required access to the memory location referenced. Access to a memory location is defined to be in one of the three modes: read (R), write (W), or execute (E). Read refers to a data or address constant fetch from memory, write is a store into memory, and execute is an instruction fetch from memory. There is a set of three ring brackets (R1, R2, R3) that are also used to determine the type of access allowed. The ring brackets restrict the process to certain types of access when executing in a given domain, or ring. Each memory descriptor shall be capable of



IA - 48,160

Figure 5. Unpagged Descriptor Segment

containing the access permission and ring bracket information that is to apply to the location referenced. During the address translation phase, the access control information in the appropriate descriptor is used to calculate the final effective access mode to the location in memory. The effective mode is compared to the desired access mode, and an access violation trap shall be signalled by the SPM if the required access is not allowed by the effective mode.

Since, during any memory reference, there are up to three descriptors that are accessed, a decision has to be made as to where to put the access control information. The general three level descriptor structure shown in Figure 4 requires that the access control information be placed in the second level descriptor, i.e., the segment descriptor. This is because access to a segment may not be the same for every process currently using the segment. If the access control information were in the direct page descriptor, all processes using the page descriptor (which is shared) are forced to have the same access. If the access control information were placed in a page descriptor for the descriptor segment, the granularity of access control would be on the order of many segments.

There are specific cases, however, such as the use of unpagged or unshared segments, in which it is convenient to place the access control information in direct descriptors. Thus, in order to support full generality, the SPM shall be prepared to accept access control information from any descriptor encountered during address translation. The A field in the descriptor shall specify that the access control information it contains is to be applied to the memory reference. It is the responsibility of the security kernel to properly set the access control bits in each descriptor.

3.1.2.1.2.1 Effective Ring (Reff)

The actual effective access to a location in memory shall be determined by comparing a calculated effective ring number, Reff, to the three ring brackets associated with a descriptor for that memory reference, and then factoring in the three access permission bits. See 3.1.2.1.3.1 for a description of the exact algorithm used to calculate the effective mode.

For the simple memory reference, the value of Reff used in this determination is the current ring number (Rcur) maintained by the SPM. More discussion on the use of Rcur can be found in 3.1.2.1.6. In the general case, however, as part of the address preparation cycle, the processor may make one or more memory references to fetch indirect addresses (address constants) before operand fetch. (The fetch of an address constant from memory is subject to the same access control and

address translation as a simple read access to data.) If an address constant is contained in a segment that can be written from a higher ring than Rcur, as is the case when an inner ring procedure is referencing arguments through an indirect address passed to it from an outer ring, the ultimate location referenced by the address constant must be subject to access control defined by the ring of the segment in which the address constant resides, rather than Rcur. If the address constant were only subject to Rcur restrictions, the inner ring procedure would, in software, have to verify that the Baddress constant pointed to a segment to which the outer ring had access. In order to eliminate the need for software validation of arguments, the SPM shall validate indirect references with respect to the ring of the segment in which the address constant resides.

The SPM shall accomplish the automatic address validation by keeping track, in terms of Reff, the maximum value of the ring number Rl in all descriptors encountered during address preparation. The value of Reff shall be initialized to Rcur at the beginning of each instruction cycle and shall apply to the instruction fetch and all references until the next instruction fetch. For each descriptor encountered between instruction fetch and operand fetch, a new value of Reff shall be computed as the maximum of the current Reff and Rl in the descriptor and this new Reff shall apply to the fetch of subsequent indirect addresses or data. It can be seen from this scheme that Reff can only increase from its initial value of Rcur. In addition, if the processor allows more than one level of indirection, subsequent indirect address fetches are subject to the constantly increasing value of Reff.

3.1.2.1.2.2 Argument Validation

One specific problem involving argument validation is solved by the use of Reff as discussed in the previous section. On a system with multiple levels of address indirection, the Reff mechanism properly checks argument validity as long as the addresses in the indirection chain are left in memory segments where they originated. On processors that achieve the addressing effects of multiple indirection through the software loading of successive links of an indirect chain into internal registers, the loading of each link shall generate an indication of the results of the Reff check applied to the entire remaining chain.

A more general problem is that, if the argument pointer is copied from an outer ring to the current inner ring, the automatic Reff type of validation performed by the SPM has no effect, since now the address constant resides in the current ring. Inner ring procedures must be able to copy argument pointers to their own ring, and they must be

able to indirect through these pointers without having to check, in software, that these pointers will do no harm. Therefore the SPM shall provide a mechanism whereby software running in a given ring can force the validation of a reference to an arbitrary virtual address with respect to any higher ring number. Two methods for accomplishing this validation are proposed below, though this specification does not preclude other implementations that satisfy the argument validation requirement.

The most convenient solution to this problem from the software point of view is to store a ring number along with the pointer at the time it is copied, which would be the value of Reff computed as if the pointer were referenced directly. On subsequent indirect references through this copied pointer, the SPM would use the ring number in the pointer as another factor in computing Reff. In order to implement this scheme, there must be sufficient bits in the format of the indirect word(s) unused by the processor. Assuming software appropriately sets these unused bits to some ring number, the SPM could examine the bits during the indirect address fetch. Note that addresses presented on the bus by the processor need not contain ring numbers in order for this scheme to work. Only indirect addresses as they are obtained from memory need be examined. A special processor order recognized by the SPM could be used to facilitate insertion of the correct ring number into an address constant as it is copied.

If there are insufficient unused bits in the indirect address to store a ring number, the SPM shall provide, as a minimum, a method whereby a given indirect address fetch be subject to an initial software specified value of Reff. For example, software could load the initial value of Reff into some register, and two special orders (a load and a store) could be recognized by the SPM to perform the normal load and store function, but with an initial value of Reff taken from this register rather than from Rcur. This latter approach limits programming generality because it requires the subroutine using an address to know when an address could possibly have originated in an outer ring. However, any approach allowing software to specify the initial Reff for indirect addresses is acceptable.

3.1.2.1.3 Descriptors

Every resource that is allocated to a process shall be represented by descriptors. Descriptors are constructed by the security kernel and are structured in memory for use by the SPM. The descriptor structure is the prime data base for the state of allocation of the system resources. Copies of descriptors in use in the SPM are only valid if they reflect the memory originals. This section will specify the format and semantics of a memory descriptor and a Descriptor Base

Root. I/O descriptors are specified in section 3.1.2.3.2, I/O Descriptors.

3.1.2.1.3.1 Memory Descriptor

The logical format of the memory descriptor recognized by the SPM is diagrammed in Figure 6. This section specifies the information required to be contained in a descriptor. The intent of this section is not to precisely specify a particular implementation. However, every implementation shall provide, in some manner, the required descriptor information. Each piece of required information is identified and its purpose and semantics specified.

Descriptor type: The T field identifies the type of the descriptor. It shall have sufficient size to identify each type of descriptor supported by a particular SPM implementation. As a minimum it shall identify three types of descriptor: an indirect descriptor that describes an array of descriptors; a direct memory descriptor which directly describes an array of memory locations; and a direct I/O device descriptor which directly describes an I/O device.

Directed traps: The DT field of the descriptor provides for software directed hardware traps. At least four values of this field must be provided. One value describes a normal descriptor that does not cause a trap. The three other values cause traps. It shall be possible for the system to distinguish between page not in memory (page fault), missing segment or segment not in memory (segment fault), and descriptor segment page fault.

Access control: Three pieces of information are defined: the A field, the Ring Brackets, and Permissions. The A (Access) field determines whether the access control fields of the descriptor are to be used to control access to all resources described by the descriptor (regardless of the number of subsequent levels of address translation). Two values must be provided: if the A field is ON then this descriptor's access control fields apply, if OFF, either an inferior or superior descriptor must provide the necessary access control. If more than one descriptor is encountered, during address translation, with the A field ON, the first descriptor (defining the largest resource) with the A field ON defines the appropriate access control. Of course, at least one descriptor with the A field on must be found.

The R1, R2, and R3 fields define the privilege rings. Each field shall contain at least three values (Integers: 0,1,2) so that the system supports at least three rings of access privilege. The

1A - 88,156

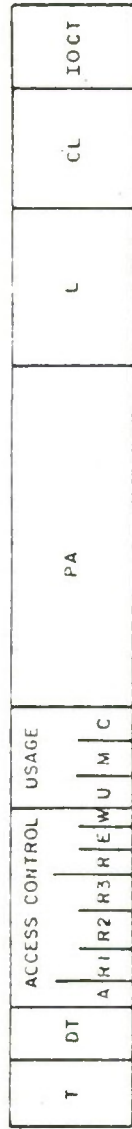


Figure 6. Memory Descriptor Format

interpretation of these fields is described below. It is the responsibility of the software to ensure that $R1 \leq R2 \leq R3$.

The Read, Write, and Execute (R, E, and W) fields define allowed modes of access to the described resource. Each field must have two values (ON and OFF): if ON the respective mode of access is allowed, if OFF the respective mode of access shall be denied.

The following rules specify the required interpretation of the above access control information. The item Reff is the effective ring number computed by the SPM during effective address formation (ref. section 3.1.2.1.2.1, Effective Ring).

- 1) Write permission if and only if ($W = ON$) and ($Reff \leq R1$);
- 2) For any resource other than the resource that is the source of the request, read permission if and only if ($R = ON$) and ($Reff \leq R2$) (The source of the request is defined as the resource to which the program counter points and from whose descriptor the access control information, for the current access, was obtained, i.e., the descriptor whose A field was on.);
- 3) For the resource that is the source of the request (e.g., address and data constants embedded in procedures): Read permission if and only if ($R = ON$ or $E = ON$) and ($Reff \leq R2$). This rule is a desirable extension of rule 2. In an implementation where software considerations do not allow the Read (R) bit to be OFF when the Execute bit (E) is ON, this extension may be omitted;
- 4) Execute permission if and only if ($E = ON$) and ($R1 \leq Reff \leq R2$); and
- 5) The use of $R3$ and the precise rules for entry/return to/from a procedure resource are specified in section 3.1.2.1.6, Cross Ring Movement. In general, Call permission if and only if ($E = ON$) and ($R1 \leq Reff \leq R3$).

Certain sequences of ring numbers are termed brackets to denote a range of allowed rings in which certain modes of access are possible. The term write bracket shall apply to rings 0 to $R1$ inclusive. The term read bracket shall apply to rings 0 to $R2$ inclusive. The term execute bracket shall apply to rings $R1$ to $R2$ inclusive. The term call bracket shall apply to rings $R1$ to $R3$ inclusive.

Usage: The U, M, and C fields record and limit the usage of the described resource. The U field has two values (ON and OFF): if OFF and the resource is successfully accessed (in any mode: read, write, or execute) the SPM shall update the value to ON. The M field has two values (ON and OFF): if OFF and the resource is successfully accessed in write mode the SPM shall update the value to ON. The C field controls the entry of elements of the described resource into a data cache (if such a cache is supported by the system). It has two values (ON and OFF): if ON the described resource may enter the cache, if OFF the resource shall not be placed in cache storage. This field is specified to ensure consistent copies of shared read/write resources in multiprocessor systems (for example, resources composed of descriptors). This field need not be included if the cache technology used in a particular system ensures the consistency of the resource with the cache copy in a multiprocessor environment.

Location: The PA field supplies the physical address of the base (in memory) of the resource described. The PA field shall have sufficient size to address all physical memory that the system may support. The PA field shall have sufficient precision to address memory resources (particularly small arrays of descriptors) without waste of physical address space. The choice of the size of the PA field shall be justified in each implementation.

Limit: The L field defines the size of the defined resource. An access request having an offset greater than the value of the L field of any descriptor encountered during address formation shall cause the SPM to generate a trap. The L field shall have sufficient size to specify the maximum size of a resource as determined by its offset (ref. section 3.1.2.1.1, Address Translation). The L field shall have sufficient precision to specify resource sizes that do not waste physical memory. The choice of the size of the L field shall be justified in each implementation.

Call limiter: The CL field defines the number of entry points within the described resource for use by the call order (ref. section 3.1.2.1.6, Cross Ring Movements). The CL field defines the maximum offset, within the resource, that may be addressed using a call order. A call order addressing an element within the resource with an offset greater than the CL field shall cause the SPM to generate a trap. The size of the CL field shall be large enough to define sufficient entry points to minimize software interpretation of cross ring transfers (call orders). It is suggested that the CL field allow for at least 64 entry points.

In some implementations software considerations or hardware properties may make it desirable to include some other form of the call limiting function. The call limiting function shall always be implemented, and

any departure from the inclusion of a call delimiter field in a descriptor must be justified.

Concurrent access: The IOCT field of direct memory descriptors shall be incremented by the SPM at each initiation of an I/O operation in/out of the described resource. This field is intended to be used by system software to determine the existence of I/O operations in progress within a resource. This information shall then be used, by system software, to keep the resource in memory until all outstanding I/O has completed.

If the requested mode of access, for a resource, is not permitted by the access control information, the SPM shall generate a trap.

The interpretation of descriptor fields is dependent on the descriptor level (ref. section 3.1.2.1.1, Address Translation). The T, DT, C, PA, and L fields are applicable for each level of descriptor. The IOCT, U, and M fields are referenced and updated only for direct descriptors. The access control fields R1, R2, R3, R, E, and W are only applicable for a descriptor which has the A field ON.

3.1.2.1.3.2 Descriptor Base Root

A special form of descriptor is recognized by the SPM. This descriptor is called the Descriptor Base Root, (DBR) and is diagrammed in Figure 7. It is used by the SPM to establish the set of descriptors for a process. The DBR is a construct similar in format to a memory descriptor. It describes where (in physical memory) and how to find the descriptors defining the resources accessible to the currently executing process. The DBR shall describe both the set of I/O device descriptors and memory descriptors accessible to a process. There are two alternative methods by which the I/O and memory descriptors may be structured. The first structure defines distinct name spaces for I/O devices and memory both rooted in a process's DBR. A name designated as an I/O device name will be interpreted in the I/O device descriptor name space; a name designated as a memory address will be interpreted in the memory descriptor name space. This structure requires the DBR to have two components: the first describes the set of memory descriptors, the second describes the set of I/O device descriptors. The second structure interprets both I/O device names and memory addresses in the same name space of descriptors. The DBR for this alternative has only one component describing all descriptors for both I/O devices and memory. If the sizes of the virtual memory space and the virtual device space (generated by the processor) are different, the SPM shall manipulate the smaller virtual address so that both memory and device direct descriptors may be conveniently accessed. The interpretation of each component of the

IA - 48. 153



Figure 7. DBK Format

DBR by the SPM shall be:

Location: The PA field provides the physical memory address of the tree of descriptors describing the resources of a process. It shall have sufficient size to address all of physical memory so that descriptors may be placed anywhere in physical memory. It shall have sufficient precision to address the smallest resource (a single descriptor) without waste of physical memory. The choice of the size of the PA field shall be justified in each implementation.

Limit: The L field defines the size of the described resource. It shall have sufficient size to accommodate the maximum number of descriptors, determined by the chosen size and interpretation of the virtual address (ref. section 3.1.2.1.1, Address Translation). It shall have sufficient precision to accommodate the minimum number of descriptors (one) without waste of physical memory space. The choice of the size of the L field shall be justified in each implementation.

Type: The T field defines the type of the DBR. It has at least two values: direct and indirect. A direct DBR type indicates that the DBR describes an array of segment descriptors (ref. 3.1.2.1.1, Address Translation). Note that either segment or page descriptors may directly describe a resource. The resource described may either be memory (via a direct descriptor) or an I/O device (via a device descriptor; ref. 3.1.2.3.2, I/O Descriptors).

If the distinct I/O device and memory descriptor structure is implemented, the DBR shall contain two components, each with the above format: one component describing the set of I/O device descriptors, the other describing the set of memory descriptors. An indirect DBR type indicates that the DBR describes an array of descriptors, each of which describes an array of segment descriptors.

3.1.2.1.4 Descriptor Structure Dynamics

The processor shall include a dispatch function. The dispatch function shall cause a DBR, for the process being dispatched, to be loaded into the SPM. The SPM shall use the information in the loaded DBR to locate descriptors for resources accessed by the process (ref. section 3.1.2.1.1, Address Translation, and section 3.1.2.3.1, I/O Naming Structure). As descriptors are referenced by the SPM, they shall be retained in a fast access descriptor store for possible reuse in subsequent accesses, by the process, to the resource. It shall be a software responsibility to ensure that changes made to the

descriptor structure in memory are reflected in the fast access copies. The SPM shall provide a function to the processor by which outdated copies of descriptors may be cleared from the fast access store under kernel software control. It shall be a hardware responsibility to ensure that any changes made, by the SPM, to fast access copies of descriptors are promptly and unambiguously reflected into the memory originals.

3.1.2.1.5 Fast Access Descriptor Store (FADS)

The SPM shall have a fast access store to retain recently used descriptors in order to meet and exceed the performance constraints of section 3.2.1.1, Relative Performance. This specification does not constrain the specific organization and technology used in an implementation of a fast access descriptor store. However, every implementation shall have the following properties:

- 1) descriptors in the FADS shall have copies of the access control information contained in the descriptor that controls access to the resource described by the FADS retained descriptor (e.g., a page descriptor in the FADS must contain a copy of the access control information obtained from its immediately superior segment descriptor)
- 2) a Clear Fast Access Store (CFAS) function shall be provided which will clear the FADS of descriptors; and
- 3) changes made to the descriptors in the FADS by the SPM (particularly to the U, M, and IOCT fields) shall be promptly and unambiguously reflected into the memory originals.

3.1.2.1.6 Cross Ring Movements

The SPM shall maintain a current ring number (Rcur) at which the processor is running. This ring number is used in the calculation of the effective ring number (Reff) associated with a particular reference to memory that is compared to the ring brackets (R1, R2, and R3) of the referenced segment. Ring changes are initiated at the request of a process using the call and return instructions, or automatically by a trap or interrupt. This section discusses the call, return, and trap requirements. Interrupts are discussed in 3.1.2.4.1.

3.1.2.1.6.1 Call and Return

Two processor orders that shall be recognized by the SPM are the call and return orders. The call order is very similar to a transfer, except that the the SPM can change the current ring number to a lower value. The return is also a transfer with a possible increase in the current ring number. Calls are normally used to transfer to inner ring procedures to accomplish more privileged operations than those allowed at the current ring, and returns are used to return from an inner ring procedure back to the outer ring from which the call originated.

Access checking on the operand of the call instruction is somewhat different from that of other instructions. The operand of a normal transfer instruction need not be accessed until the next instruction fetch cycle, and thus access to the operand may not be required or checked until the program counter is loaded with the new virtual address generated by the transfer instruction. Since the call instruction can change Rcur to a lower number and thus put the processor in a more privileged state, the SPM must guarantee that entry into the inner ring is tightly and completely controlled by that inner ring. This means that the SPM must check that calls can only be made to specific locations within specific procedures belonging to the inner ring.

The mechanism that accomplishes this control shall perform as follows. An inner ring procedure that is callable from an outer ring is defined as a "gate" by specifying in the ring brackets of the descriptor for the procedure segment a value of R3 that is different from R2. Normally, transfers to a segment cannot be made from rings above R2. However, a call instruction is allowed to a procedure if the call is made from a ring less than or equal to R3. If such a call is made, the new value of Rcur becomes R2, and execution continues. The value of Reff after address preparation for the call instruction is used in the comparison with R2 and R3. The tests made in the call are as follows:

Reff > R3 entry denied, trap (outside call
bracket)

$R2 < \text{Reff} \leq R3$ entry allowed, R2 becomes Rcur

$R1 \leq \text{Reff} \leq R2$ entry allowed, Rcur unchanged

Reff < R1 entry denied, trap (outside call
bracket)

The checks on call shall not preclude using the call instruction to transfer to a procedure from within its execute bracket. Nor shall it be required that a segment be a gate (i.e., $R2 < R3$) in order to be called from within its execute bracket. Thus, the call bracket is defined as $R1$ to $R3$, with $R2$ being the new ring of execution if the segment is a gate and the call is from outside $R2$.

It is not sufficient to simply specify which segments are gates. There must also be a mechanism for specifying which locations in the gate segment are valid entry points. As noted in the description of the call limiter field (3.1.2.1.3.1), this function can be achieved through the use of the call limiter field or through equivalent functionality justified for a particular implementation. If the call limiter field is used, the call limiter specifies the maximum offset within the segment to which a call can transfer. Thus, if a segment is a gate, a certain number of locations at the beginning of the segment must be reserved for entry points, and the called procedure must be prepared to accept entry at any of these locations. The SPM, during the access check discussed above, shall verify that the offset of the virtual address is no greater than the call limiter field before changing the ring of execution.

In the general case the list of entry points in a gate segment would consist of a transfer vector -- a series of transfer instructions that transfer to various locations in the segment beyond the gate boundary or within other segments in the new ring that are not gates. Since transfer to other segments requires a sequence of several instructions to form a virtual address containing another segment's segment number, it is more useful, though not necessary, to specify that the transfer instruction at the gate is a transfer to a relative location within the same segment as the gate, but beyond the call limiter. Note if such transfer instructions occupy more than one word of memory, such as even-odd pairs, the SPM must only permit calls to transfer to even-numbered words. The SPM must know the number of words occupied by transfer instruction in a gate. This number can be a system convention that is fixed in hardware.

In addition to the gate and access checks, the SPM shall make the caller's program counter and ring number (R_{cur}) available to the called procedure. The called procedure must know the caller's ring number so that access to arguments not already controlled by the argument validation mechanism can be verified by software. The SPM shall also make available the current ring number so that the called procedure knows what its own access rights are. This is important because a procedure may have been entered from a ring within the execute bracket, and therefore is not necessarily executing at $R2$. Such multi-ring procedures that perform software argument validation must be able to compare the current ring with the ring of the caller.

In conjunction with being able to obtain the current ring number, the SPM shall provide a mechanism whereby a multi-ring gate procedure (defined as $R1 < R2 < R3$) can obtain, upon entry via a call from an outer ring, the address of the stack segment for the current ring without having to first write into memory. This is a requirement because a multi-ring process must have a separate stack segment for each ring, and a gate procedure, upon entry, cannot rely on any information as to the location of the stack segment that may have been passed to it by the caller. In the simplest case the stack segment number is keyed by convention to the current ring number, thereby simply requiring that the ring number be used to index into a table of preset stack pointers. Note that this requirement only applies to multi-ring gates when called from an outer ring. Procedures that are not gates, or gates entered from within the execute bracket, always execute in the ring of the caller and can thus believe the value of the stack address stored in some register by the caller. Only when the caller is from an outer ring must the stack address be independently calculated by the called procedure.

Another transfer instruction that shall be recognized by the SPM is the return instruction. The only requirements for return are that the returning procedure be able to specify the ring to which to return and that returns to inner rings be prohibited. Otherwise the return operates just like the transfer. Assume that R_{to} is the ring to which the procedure desires to return:

$Reff < R_{to}$ R_{to} becomes R_{cur}

$Reff > R_{to}$ return denied, trap (inward return)

3.1.2.1.6.2 Trap and Trap Return

Traps are software initiated events (either intentional or unintentional) to which the processor responds by saving the current state of the processor in such a way that it can later be restored, and transferring control to a specified memory location. In a secure system, many traps occurring in a given ring are best handled by software executing in that ring. Some traps, however, such as some of those generated by the SPM, are best handled by inner ring software. Page faults, for example, must be handled by the kernel since the user must not be able to determine whether paging is taking place. In addition, if the data stored during a trap is security sensitive in that the user should not be allowed to restore the processor with such data upon trap return, the trap (and probably therefore all traps) must be handled by the kernel. An example of the latter can be found in the case where a processor might generate a trap in the middle of instruction execution during the formation of an indirect address. If

the trap return sequence supported by the processor allows restarting in the middle of the instruction, then part of the information saved and restored by software must be the value of R_{eff}. In such a case there must be a mechanism whereby software cannot specify a restored value of R_{eff} lower than R_{cur} of the trap handling procedure. If the SPM can make this check, then the trap may be handled by any ring. If the SPM cannot verify this, then all traps must be handled by the kernel and there need be no facility for handling traps in an arbitrary ring. The requirements below assume that the hardware does not restore security-sensitive information during a trap return. If the hardware requires that security-sensitive information be restored by software, and the hardware cannot verify this information, then there is no requirement that traps to an arbitrary ring be allowed (i.e., all traps shall be to the security kernel).

Typically in a processor there is a trap vector that consists of a list of entry points for each class of traps. There may also be a parallel list of storage areas in which the processor is to store the state information for each trap. This parallel list may degenerate into a single address of a storage area for all traps. Since at least some traps must be handled by the kernel, the kernel must be in control of the trap vectors and the storage area pointers. If the processor provides for software to specify the location of the trap vector, this operation must be privileged so that it can be restricted to the kernel (see 3.7.2). The only responsibility of the SPM is to translate the virtual address of the trap vector, the entry point and the storage area, and to calculate a new ring of execution in the case where a trap occurring in one ring must be handled in an inner ring.

A sample scenario of a trap sequence is as follows. Assume a trap of type N has occurred. The processor indexes into the trap vector and loads the program counter with the entry point specified for trap N. It also stores state information in the proper storage area. Other than address translation, the SPM need perform no verification on these addresses since they were set up by the kernel. However, the SPM must set a new current ring of execution. In addition, sufficient information must be made available to the trap handling procedure so that the exact cause of the trap can be determined and so that the trap return can be used to restore the processor to a state existing just before the trap occurred. As a minimum, such information shall include the value of the program counter, the ring number, and the name or type of the trap.

A trap is very much like a call, except that the trap handler should not be directly callable by the user. Thus, on a trap, the SPM shall set R_{cur} equal to R₂ of the trap handling procedure if the trap occurred while in a ring greater than R₂. If the trap occurred from a ring less than or equal to R₂ the value of R_{cur} should not change.

This allows a given trap handler to work in several rings. Since the trap handler should not be callable directly from an outer ring, no check for a valid gate procedure ($R2 < R3$) shall be made. Also, the call limiter shall not be checked on entry into the trap procedure.

As stated previously, if the processor requires security-sensitive information to be restored upon trap return, then it is sufficient for the SPM to force the new value of Rcur to zero, rather than to the software specified value R2.

The kernel can provide a mechanism whereby the user can specify a procedure to handle a certain trap, and it is the responsibility of the kernel to verify that the specified procedure is properly accessible to the user before storing the entry point in the trap vector. If the pointer for storage of the state information is associated with the trap, the user should also be able to set that pointer, subject to kernel validation. If there is only one storage area for all traps, the kernel should provide a facility whereby the user's trap handling procedure can examine the state information for the trap that occurred.

The trap return instruction is identical to the return instruction except that new state information must be loaded into the processor and SPM so that the process can resume operation from the point of the trap. In particular, for SPM traps generated due to access violations or the directed trap field (DT) set in a descriptor (e.g., page faults) it should be possible for the kernel trap handling procedure to set the proper access or change the descriptor causing the trap so that execution in the faulting procedure can be resumed as if the descriptor were correct in the first place. Since a trap return may possibly occur from a ring outside the kernel, the SPM shall check that the new value of Rcur is no less than the current value of Rcur.

3.1.2.2 Device to Memory Interface

3.1.2.2.1 I/O Flow

There are two alternative data paths from device to memory specified. Each device attached to a secure data communications processor shall use at least one. The basic difference between the alternatives is defined by the nature of the information resident in a DMA device, where a Direct Memory Access (DMA) device, once initiated, will control a series of data transfers to (from) memory.

The first type of device to memory mediation, premapped I/O, interprets and translates memory addresses at I/O initiation and the device subsequently uses absolute addresses. The alternative, mapped

I/O, requires SPM mediation of each memory request by the device. An implementation of the SPM to handle both types of flow is desirable, although nothing herein shall preclude an implementation of the SPM that would handle only one type. If both types are handled, at I/O initiation the SPM shall use information within the I/O device describing mechanism (ref. section 3.1.2.3.2, I/O Descriptors) to determine which flow is applicable.

3.1.2.2.1.1 Premapped I/O Flow

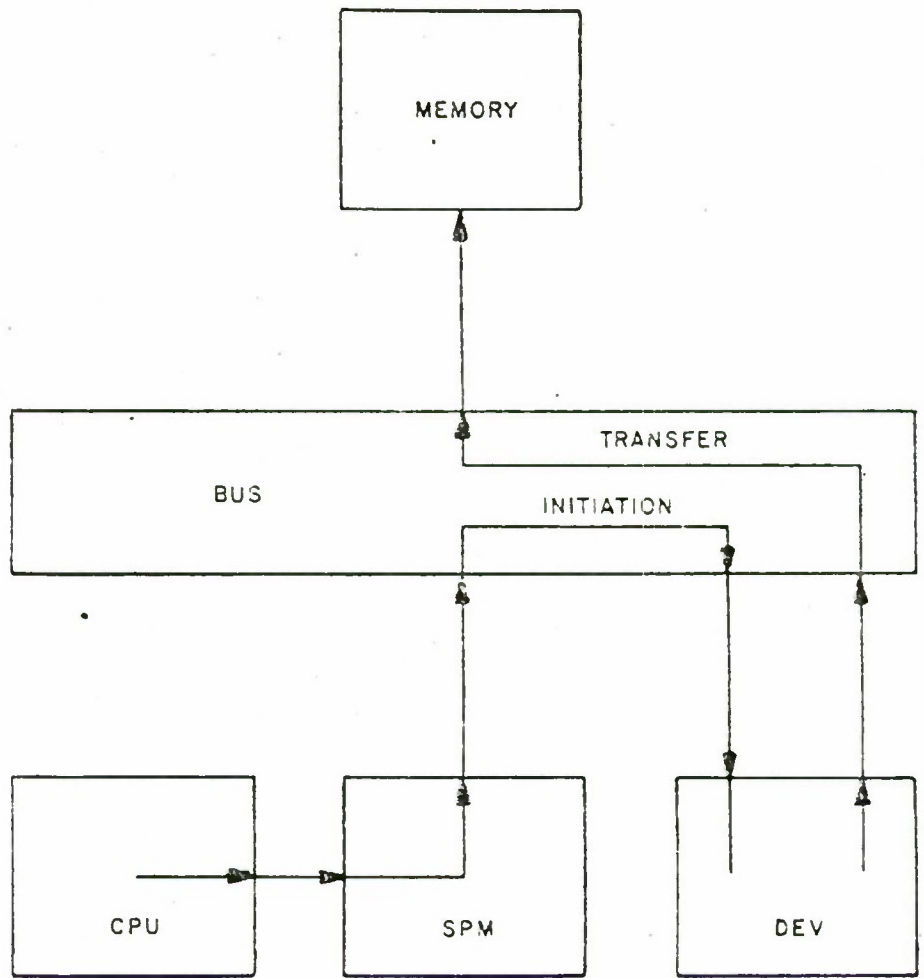
The premapped I/O flow is shown in Figure 8. This figure is meant to illustrate the flow of the addresses associated with an I/O transfer, and is not meant to constrain system physical geometry. At premapped I/O initiation the virtual address associated with the transfer is delivered to the SPM. After suitable checking (3.1.2.3.3) by the SPM's premapped I/O checker the address is mapped by the SPM to an absolute memory address and loaded into the device. Transfer of data will occur directly between the device and memory using absolute addresses. The SPM shall mark memory descriptors, for all memory locations to be referenced by the device, at I/O initiation time so that the kernel will know not to invalidate these descriptors during the I/O operation.

3.1.2.2.1.2 Mapped I/O Flow

The address flow for the mapped I/O flow is illustrated in Figure 9. At mapped I/O initiation the virtual address associated with the transfer is delivered to the SPM, and then is loaded into the device as a virtual address. The address of each item of data transferred shall be delivered to the SPM for mapping and checking. Each address delivered to the SPM shall be accompanied by the identification of the transferring device so that the correct memory descriptor may be obtained by the SPM. The SPM shall retain, for each active I/O device, the following information. (An active I/O device is one for which an initiated I/O operation has not yet terminated.)

- 1) the effective ring number the device is to operate at; and
- 2) some method by which the SPM may access the memory descriptors of the process that initiated the I/O operation. For example, the SPM may remember the DBR contents at the time the I/O operation was initiated.

Each access by the device, to memory, is to be mediated by the SPM. The access checking performed by the SPM is equivalent to the checking performed for memory accesses by a processor. Each access is



IA - 48, 151

Figure 8. Premapped I/O Flow

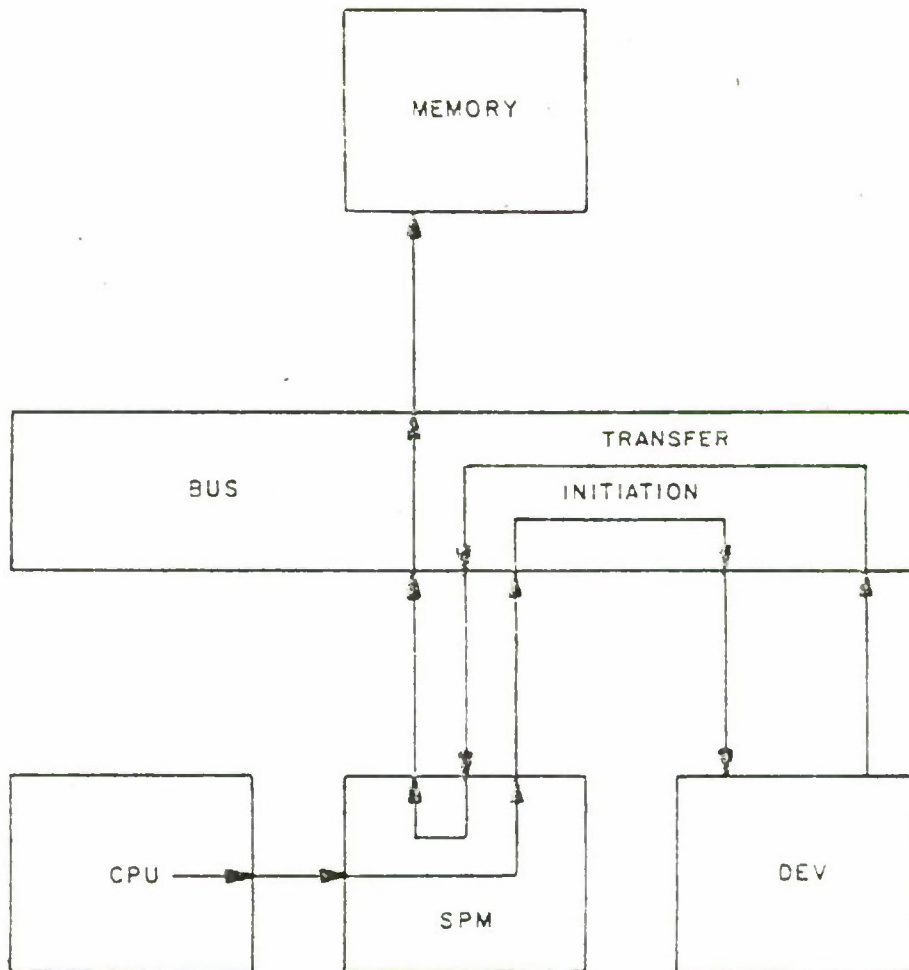


Figure 9. Mapped I/O Flow

evaluated at the effective ring number of the device, in the mode of the device (read or write), using the descriptors contained in the address space of the process that initiated the I/O operation.

If the SPM retains descriptors recently used by I/O devices in its Fast Access Descriptor Store, the SPM shall provide the capability to clear the FADS selectively, by device.

3.1.2.3 Processor to Device Interface

3.1.2.3.1 I/O Naming Structure

The software structure by which a processor names an I/O device will vary with the particular processor chosen as the base for a secure data communications processor. The naming structure of a secure data communications processor shall have certain properties. It shall allow for a virtual to absolute mapping of the device name on a per process basis. It shall allow for the unique assignment of a device to a process. It shall allow for the unique assignment (and naming) of each physical device attached to a multidevice controller.

Two classes of I/O devices may be supported by a secure data communications processor. The first class are Direct Memory Access (DMA) devices. This class, once initiated by a processor order, independently references memory to perform the required data transfer. This class of device is the primary object of discussion in the following sections. The second class of device are Programmed I/O (PI/O). These devices do not independently reference memory and return information from the device directly to the processor. Included in this class are status requests to DMA devices whose information may also be routed directly to the requesting processor. For this class of device, the SPM need not support a data path from device to memory but shall ensure that the requested information is returned from the named device to the processor that made the request.

For both classes of device, the SPM shall require that accesses made by processors to devices be mediated through I/O descriptors. Section 3.1.2.1.3.2 discussed two alternative means by which I/O descriptors may be addressed within the set of descriptors rooted in the DBR. One alternative places I/O descriptors in their own descriptor tree, distinct from memory descriptors, rooted in the DBR. For the other alternative, I/O descriptors are embedded in the same descriptor tree as memory descriptors, and, in general, may be placed anywhere in the descriptor tree that a direct memory descriptor may be placed. Figure 10 illustrates the placement of an I/O descriptor at the memory segment descriptor level: the device assumes the process local name of an entire segment of virtual memory. Figure 11 illustrates the

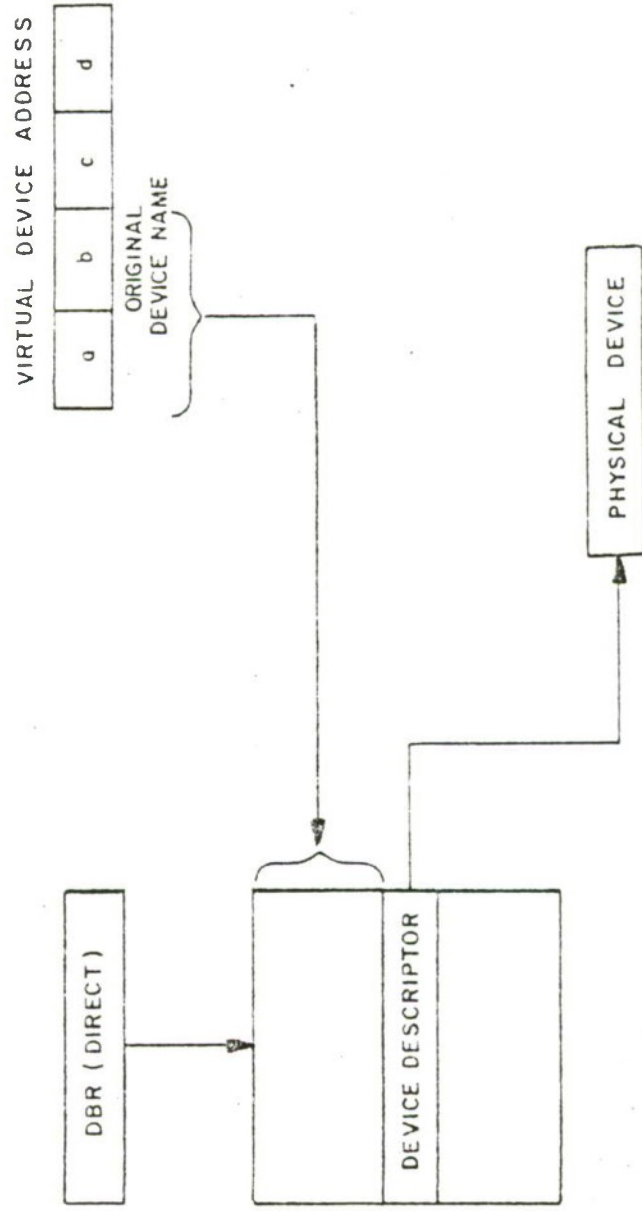


Figure 10. I/O Descriptor Interpretation

IA - 40, 157

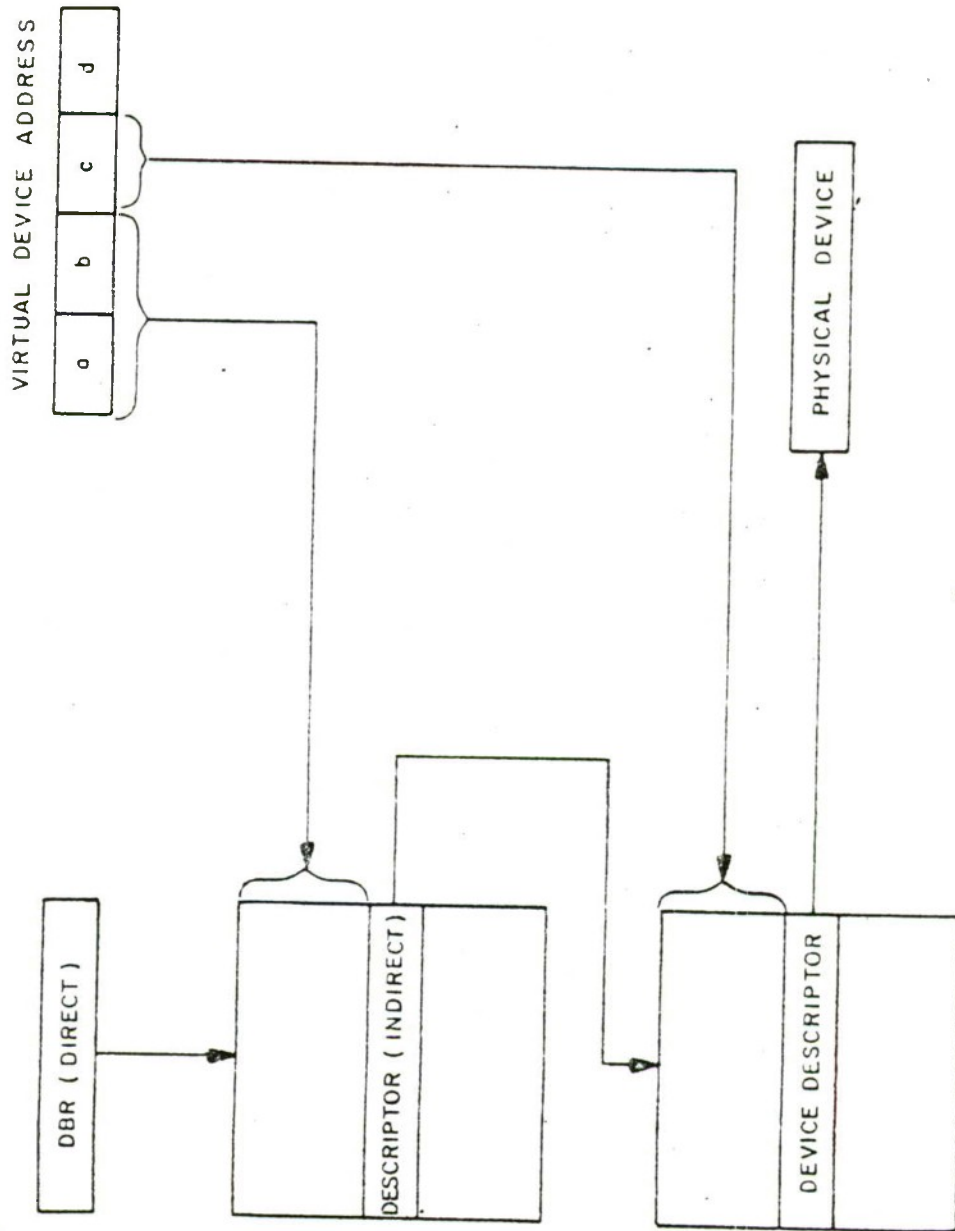


Figure 11. I/O Description Interpretation

placement of an I/O descriptor at the memory page descriptor level: the device assumes the process local name of the page of a segment of virtual memory. For each of the above examples the diagrams assumes a direct DBR. Of course, every SPM implementation shall also support indirect DBRs (section 3.1.2.1.1 and 3.1.2.1.3.2).

3.1.2.3.1.1 Explicit Names

In this structure the processor is aware of explicit I/O names, and issues them identified as I/O names. The SPM uses the descriptor tree rooted in the DBR to find an I/O descriptor. As discussed in sections 3.1.2.1.3.2 and 3.1.2.3.1, there are two alternative means by which explicit names I/O descriptors may be found in the set of descriptors rooted in the DBR. The first requires a distinct tree of I/O descriptors rooted in the DBR. The SPM, when presented with an address identified as an I/O device name shall search for the I/O descriptor within the I/O descriptor tree. The second structure embeds explicit names I/O device descriptors within the memory descriptor tree. For this alternative, the SPM, when presented with an address identified as an I/O device name, shall transform the presented virtual device name into a virtual memory address and search the memory descriptor tree for an explicit name I/O descriptor. The access control information applicable to the I/O descriptor is used to check access rights.

3.1.2.3.1.2 Devices in Memory

In this structure a process has no explicit names for I/O devices, and there may not be specific I/O instructions. The set of registers that control a device is represented by a range of absolute memory addresses. The SPM will find the direct descriptor defining these memory addresses in the memory descriptor tree rooted in the DBR. The access control information applicable to this memory descriptor is used to control access rights.

3.1.2.3.2 I/O Descriptor

The SPM mediates the processor to device interface by treating all references by a process to devices as virtual references, and mapping them through an I/O describing mechanism. Two types of I/O descriptors are specified: the type supported by a specific implementation of this specification will be determined by the form of I/O instruction supported by the processor. This specification assumes that any one minicomputer system will support one or the other type of I/O instruction: devices in memory or explicit names. If a

system supports both kinds, the descriptors for each type of device shall be distinguished by distinct descriptor type fields (T field). Both types of I/O descriptors are direct descriptors: the I/O device is directly described. Indirect descriptors describing an array of direct I/O descriptors have precisely the same format as the memory descriptors specified in section 3.1.2.1.3.1, Memory Descriptor.

3.1.2.3.2.1 Explicit Names Descriptor

In the explicit names structure, I/O descriptors are direct descriptors contained in the tree of descriptors rooted in (located by) the DBR (Figure 13). The SPM shall obtain the appropriate descriptor when presented with a virtual device name by the process. The logical format of the explicit names I/O descriptor is diagrammed in Figure 12. The descriptor fields are interpreted by the SPM as:

Directed traps: The DT field of the descriptor provides for software directed traps on access. At least two values of this field must be provided, one of which does not cause a trap. All other values of the field cause an SPM generated trap.

Access control: Three pieces of information are defined: the A field, the Ring Brackets, and Permissions. The A field determines whether access control fields of the descriptor are to be used to evaluate the propriety of the access. Its interpretation is identical to that of the corresponding field in the memory descriptor (ref. 3.1.2.1.3.1, Memory Descriptor). The R1, R2, and R3 fields define rings. Their definition is identical to that of memory descriptors. Their interpretation differs, only for the R3 field, as specified below. The R, W, and E fields define allowed modes of access. Their definition is similar to that of memory descriptors. The only difference is the interpretation of the E field. This field, for I/O descriptors, shall be undefined except for pertinent control operations, e.g. diagnostic operations, microcode loading/modification. The system specific operations shall be allowed if and only if (E = ON) and (Reff < R3). Note that it is not required that $R3 \geq R2$. In developing an I/O virtual device address, an effective ring number (Reff) is developed by the SPM in an identical manner to a memory access. The following rules specify the allowed modes of access to an I/O device:

- 1) Read permission (initiate a read from the device) if and only if (R = ON) and (Reff \leq R2); and
- 2) Write permission (initiate a write to the device) if and only if (W = ON) and (Reff \leq R1).

IA - 43, 154



Figure 12. "Explicit Names" IO Descriptor Format

IA - 46,155

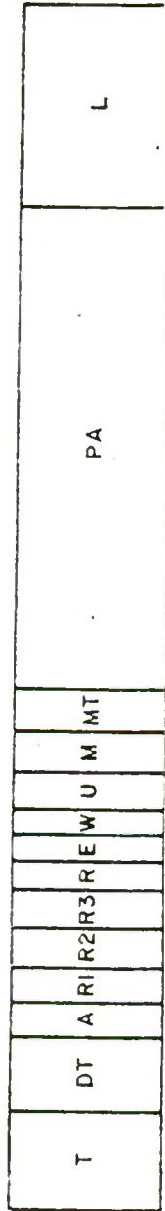


Figure 13. "Devices in Memory" IO Descriptor Format

Type: The T field identifies the type of the descriptor. I/O descriptors shall be identified by at least one distinct value of this field, which identifies a direct device descriptor.

Usage: The U and M fields shall have an interpretation similar to memory descriptors. For each initiation of an input/output operation from/to a device, if the device descriptor's U field is OFF, the SPM shall update it to ON. These fields shall be updated only if there was no access violation. For each initiation of an output operation to a device, if the device descriptor's M field is OFF, the SPM shall update it to ON.

Mapping type: The MT field has two values: ON and OFF. If MT=ON then the described device is a premapped I/O device, if MT=OFF then the described device is a mapped I/O device.

Actual device: The PD identifies the physical device name to be used upon access to this descriptor. This field must have sufficient size and precision to accommodate all device names addressable by the processor.

3.1.2.3.2.2 Devices as Memory Descriptor

In the devices in memory structure the I/O description mechanism shall be a descriptor contained in the tree of memory descriptors rooted in the DBR (ref. Figure 13). The interpretation of the fields of the descriptor will be the same as the interpretation of the explicit names I/O descriptor with the following exceptions:

Location: The PA field provides the physical address of the base of the set of contiguous memory locations that constitute the control registers for the device. It shall have sufficient size and precision to address all device addresses supported by the minicomputer system. This field replaces, for the devices in memory type of device, the function of the PD field in the explicit names I/O descriptor.

Limit: The L field limits the number of memory locations which constitute the device's control registers. It shall have sufficient size to accommodate the largest set of device control registers. It shall have sufficient precision to accommodate the smallest set of device control registers without waste of physical address space.

3.1.2.3.3 I/O Initiation

In either I/O naming structure the SPM must be aware that an I/O operation is being requested. In the explicit names structure the SPM shall be informed by processor signal. In the devices in memory naming structure the SPM shall conclude that an I/O operation is requested by the type of the direct descriptor accessed (the T field shall indicate an I/O descriptor). The SPM shall determine whether the device is to be treated as a premapped device or a mapped device from an examination of the MT field in the I/O descriptor.

In either I/O mapping structure, the SPM shall validate and translate each access to memory by a device with respect to the descriptors of the initiating process. An I/O operation is defined as a request for a transfer to/from memory in which the processor specifies a device, the direction (mode: read/write) of the transfer, and a single starting address and extent. Some devices, particularly fast mass storage (disks), support chaining of I/O operations. Under this structure, this device reads a list of I/O requests from memory and executes them sequentially. For chained I/O, the addresses used by the device to locate I/O requests in memory shall also be validated and mapped by the SPM. These devices are perhaps most conveniently supported as mapped I/O devices.

In multiprocessor systems, each processor shall be capable of initiating I/O operations to any configured I/O device.

3.1.2.3.3.1 Premapped Initiation

When a device is to be treated as a premapped device, the SPM shall ensure the following conditions before permitting the initiation of each I/O operation:

- 1) that the device has been assigned to the process requesting transfer; the assignment is indicated by the presence of a descriptor;
- 2) that all memory addresses affected by the transfer (starting address through starting address plus extent) have the proper access permission for the effective ring number and access mode of the process requesting the transfer;
- 3) that the range of affected memory addresses falls within the range of memory described by one direct memory descriptor -- segment or page; and

- 4) that the descriptor defining the I/O device allows access in the requested mode at the effective ring number of the process requesting the transfer.

If any of the above checks fail, the SPM shall initiate a trap to the requesting processor. If all of the SPM checks are successfully passed, the SPM shall cause the following actions:

- 1) the descriptor for the affected memory addresses is marked (I/O count field incremented) to provide notice to the security kernel software that an I/O operation using this memory descriptor has been initiated;
- 2) the descriptor for the I/O device is marked (U and/or M fields) to indicate that an I/O operation has been initiated;
- 3) the SPM translates the requested virtual memory address and extent to a physical address and extent and causes this information to be loaded into the device; and
- 4) the SPM allows the requested I/O operation to be initiated.

Transfer of data will occur directly between the device and memory.

3.1.2.3.3.2 Mapped Initiation

When a device is to be treated as a mapped device, the SPM shall ensure:

- 1) that the device has been assigned to the process requesting the transfer -- determined by the presence of an appropriate I/O descriptor; and
- 2) that the descriptor defining the device permits access in the requested mode at the effective ring number of the process requesting the transfer.

If either of these checks fail, the SPM shall initiate a trap to the requesting processor. If all of the checks are successfully passed, the SPM shall cause the virtual address to be loaded into the device and the requested operation initiated. The SPM shall remember, for each active device, the following information, stored at the time that an I/O operation is initiated:

- 1) the effective ring number of the process initiating the I/O operation; and

- 2) the DBR or the set of memory descriptors, of the requesting process, that will be referenced by the device during the I/O operation, so that the SPM shall consistently interpret device memory requests with respect to the descriptors of the initiating process.

When the virtual address associated with each request by the device for data transfer and the device identifier arrive at the SPM, the SPM shall be able to retrieve the effective ring number previously stored and the memory descriptor required by the virtual address. The memory descriptor shall be one of the memory descriptors defined by the security kernel for the process requesting the I/O transfer. The checking undertaken by the SPM, for a device memory request, during a mapped transfer shall be identical to the checking of a memory access by a process running on a processor.

The SPM (or SPMs in a multiprocessor configuration) shall ensure the following conditions on mapped I/O operations.

- 1) each device knows which SPM to use to make requests for memory access;
- 2) the SPMs shall ensure that there exists only one outstanding I/O operation per device -- this condition is imposed to guarantee that, for any I/O operation, the virtual addresses presented by the device are consistently interpreted with respect to the descriptor structure of the process that initiated the I/O request; and
- 3) the SPMs shall ensure that fast access copies of descriptors retained for use by active mapped devices accurately reflect the memory originals of the process requesting the I/O operation in progress on the device. The SPMs shall provide sufficient mechanism to invalidate copies of descriptors retained for the use of devices if the original process descriptors are altered by the kernel.

3.1.2.4 Device to Processor Interface

The only device to processor interface is the signalling of interrupts by a device. In the discussion below, the term "device" will be used to refer to not only I/O devices, but devices such as timers that can interrupt a processor.

3.1.2.4.1 Interrupt Structure

When an I/O device signals an interrupt, that signal shall be directed to the processor with which the device is currently associated. This association may be determined at the time of initiation of the I/O order, or fixed in hardware. For processor-local devices, such as interval timers, this association may be fixed in hardware.

The processor shall have the ability to allow the interrupt handler to determine the cause of the interrupt, and all information stored by or in the SPM at initiation of the last I/O order on that device shall be available to software at interrupt time. In addition, the state of the processor at interrupt shall be made available so that it can later be restored on return from interrupt.

Since, on return from interrupt, it is required that the interrupted procedure continue to execute as if no interrupt had occurred, the specific points in the instruction cycle and descriptor fetch during which interrupt can occur must be well defined with respect to the SPM. Another processor requirement is that the security kernel, at its discretion, have the capability of delaying any interrupts for certain periods of time. The delaying of interrupts must be a privileged operation as defined in section 3.7.2.

3.1.2.4.2 Interrupt Storage and Entry

The sequence of events occurring at interrupt time are exactly the same as for traps, though the amount and type of information saved at interrupt might be different depending on the requirements of the processor. Upon recognition of an interrupt, the processor makes references to an interrupt vector of entry points and possible storage areas and the SPM shall set a new value of Rcur depending on the ring brackets of the entry location. It is the responsibility of the security kernel to set the interrupt vectors properly. As for traps, specification of the interrupt vector location must be a privileged operation if allowed by the processor.

3.1.2.5 Processor to Processor Interface

3.1.2.5.1 Maintenance of Descriptor Structure

In a system configured with multiple processors each processor will work with its own SPM. Changes to the descriptor structure will be made by the security kernel software, and in certain limited cases by SPMS.

The processors shall use a system wide semaphore to coordinate operations on descriptors. The processors shall have a Clear Fast Access Store (CFAS) function. The CFAS function shall be capable of directing any configured SPM (initiated from any processor) to clear all or some of its fast access descriptor store. The CFAS function shall incorporate a response mechanism to inform the processor of the completion of the order by an SPM. The CFAS function shall incorporate the ability to clear by descriptor type (the T field). If mapped I/O is implemented the CFAS function shall have the ability to clear an SPM by specific device name.

SPMs will modify the U, M, and IOCT fields of memory descriptors. Logically, an SPM must seize and release the system wide "changing descriptor" semaphore, with a trap generated on failure to seize. An indivisible memory Read-Alter-Rewrite function for setting U or M or incrementing IOCT is highly desirable.

The processors shall provide indivisible memory test-and-set instructions to coordinate processor access to shared memory.

3.1.2.5.2 General Interprocessor Signalling

Each processor shall have the capability to signal any processor in the system. The signal function shall invoke the system interrupt structure, causing the receiving processor to execute the software defined interrupt routine.

3.1.2.6 Operator to Processor Interface

The requirements for the interaction of an operator with a secure data communications processor are beyond the scope of this document. It is anticipated that this interaction will be defined and controlled by the security kernel. The secure data communications processor shall have the ability to allow some portion of the security kernel to be bootstrapped.

3.1.2.6.1 Standalone Bootstrap

When the secure data communications processor is to be operated in a standalone environment, some I/O device shall be controlled by the system operator to effect an initial memory load. In Figure 14 is shown the contents of memory following the initial memory load. This figure is meant to be illustrative, and is not intended to preclude other designs of the bootstrap mechanism. In this example, the SPM is addressed as an I/O device using I/O descriptors. In Figure 14 a DBR,

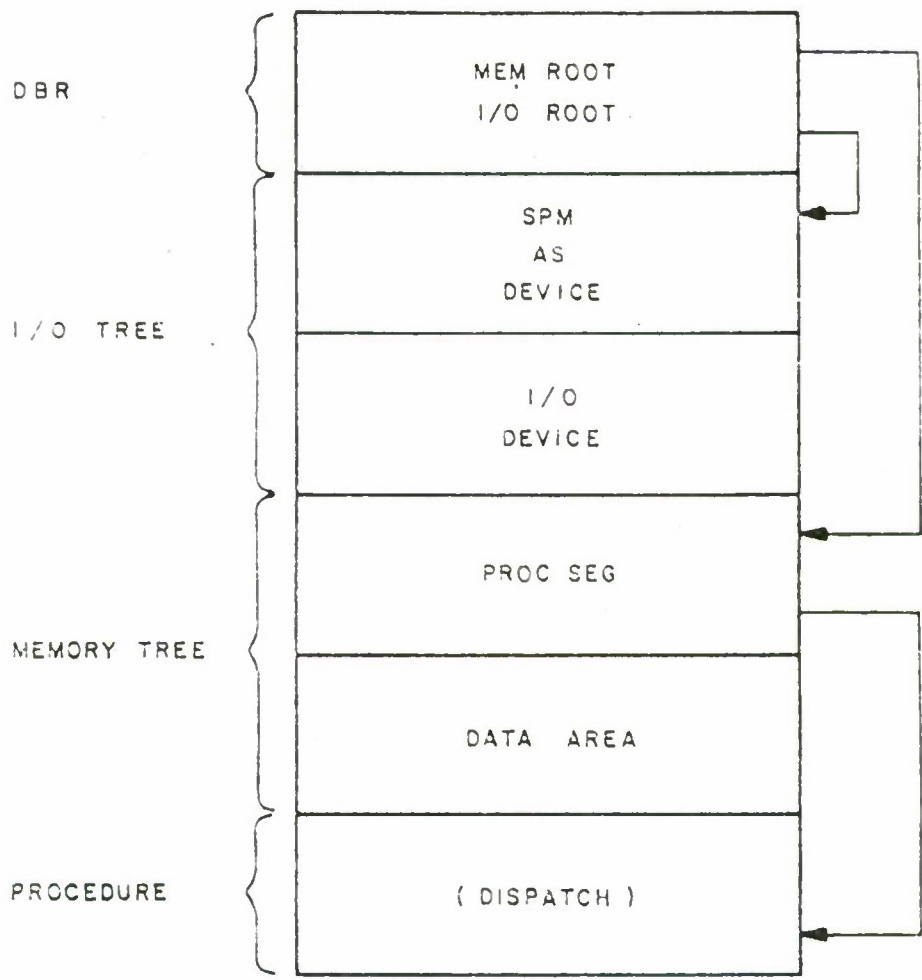


Figure 14. Bootstrap

1A - 48, 150

two I/O descriptors, two memory descriptors and a procedure segment have been loaded. The DBR establishes the trees of I/O (2) descriptors and memory (2) descriptors. The first I/O descriptor establishes the SPM as an I/O device, the second establishes the device for further memory loading. The first memory descriptor establishes the loaded procedure, the second establishes a memory area for further I/O input. It is assumed that the processor Program Counter can be set to extract the first order of the procedure segment. The DBR is initialized either externally or by convention, by the bootload function, to a predefined value. The current ring is initialized to be zero. The contents of the Program Counter is assumed to be a virtual address and the corresponding instruction is fetched from memory using the initial DBR and memory descriptors. Processing continues in ring zero (until explicitly changed by software) with all addresses interpreted as virtual addresses.

3.1.2.6.2 Front End Bootstrap

When the secure data communications processor is used as a front end for some host processor, it shall have the ability to be bootstrapped from the host processor. Within the illustrative protocol of Figure 14, the initial memory load would be performed by the host processor through an interconnecting unit.

3.1.3 Major Component List

The secure data communications system is composed of the following components:

1. SPM
2. Processor
3. Memory Units
4. Controllers

In applications where the secure data communications system is acting as a front end for a host processor, an Interconnecting Unit will be required (ref. section 10.1).

3.1.4 Government Furnished Property List

As specified by procuring agency

3.1.5 Government Loaned Property List

As specified by procuring agency

3.2 Characteristics

As secure data communications processors are intended for use over a variety of applications and in a variety of environments, there is no one set of characteristics that will be universally desirable. It is anticipated that each procurement for a secure data communications processor will refer to the architectural requirements specified herein and to a specification of the precise characteristics required for the intended application. Reference to a particular specification established for a particular application is denoted herein by the phrase "per individual specification".

3.2.1 Performance

3.2.1.1 Relative Performance

The performance of a secure system incorporating the SPM is dependent on the implementation of the SPM and its method of connection to the nonsecure components. Performance for a secure system can be roughly estimated relative to a similar but nonsecured system. For a secure minicomputer system, conforming to this specification, the performance degradation introduced by the SPM, relative to an equivalent unsecured system, shall not exceed 25%.

3.2.1.2 Data Cache

A data cache is one method (among many) to increase the processor performance of a system. It is not the intent of this document to require the incorporation of a data cache in the SPM. In the event that a system implementation includes a data cache in the SPM, certain characteristics are required. The C field is in the memory descriptors to allow the software to instruct the SPM to keep data out of the data cache. This facility must be provided in order to ensure the consistency of the descriptor structure in the fast access descriptor store, data cache, and memory when descriptors are manipulated (by security kernel software) as data. This is most

easily ensured by the restriction that the areas of memory containing descriptors not be placed in a data cache. Data cache must be responsive to the CFAS function in clearing any data or physical address information associated with a cleared descriptor.

The above requirement does not preclude cache implementations in which the consistency of shared read/write data areas (particularly if composed of descriptors, and most emphatically in multiprocessor systems) is ensured by mechanisms other than not placing descriptors in the data cache.

3.2.2 Physical Characteristics

To reduce cost, "off-the-shelf" commercial grade equipment is utilized to a maximum extent. However, due to the criticality of communications systems in many DoD strategic and tactical applications, coupled with the need for sparing, repairing and replacing attritioned equipment, additional requirements will be imposed which require modifications to commercial equipment or in some cases new designs. More stringent requirements are necessary in the following areas:

- A. Availability - Redundant systems (without single-point failures) with bit correction and low mean times to repair (MTTRs) and roll-back and recovery capability may be required to meet on-line availability requirements.
- B. Reliability - More stringent parts selection, control, screening and system/subsystem burn-in may be required.
- C. RFI Compatibility - More stringent requirements may be imposed to mitigate jamming and upset.
- D. Tempest - Control of compromising emanations is required.
- E. Radiation Survivability and Vulnerability - For certain strategic applications, EMP and nuclear emissions survivability is required.
- F. Configuration Control - Suitable identification control and accounting procedures will be required for critical-item provisioning.

In general, the quality level of equipment suitable for digital communications applications fall into three broad categories:

- A. Ruggedized commercial equipment (MIL-E-4158(-1))
- B. MIL spec commercial compatible equipment (MIL-E-4158(-1), MIL-E-16400(4), MIL-E-5400, curve II and III)
- C. Sophisticated ad-hoc designs (MIL-E-5400, curve IV, MIL-E-8189)

Category A equipments are suitable for most ground fixed-site applications, Category B equipments are required for most mobile applications and Category C equipments are required for space and/or "high-rel" applications.

Packaging - Table top, self contained and/or rack mountable (19") configurations.

Air Cooling - ambient air (coolers may be utilized subject to noise constraints)

Size - per individual specification

Weight - per individual specification

Power - per individual specification

Mounting and Access - per individual specification for ease of maintenance and repair.

3.2.3 Reliability

The probability of failure, per hour of operation, shall be less than 0.00005/hr for the Security Protection Module alone, and less than the following figures for units of the system which it is to protect, except as may be relaxed for particular installations.

Processor: <0.00005/hr

Memory Unit
(128K bytes): <0.00004/hr

Controllers: < .00004/hr

The SPM design shall be such that the probability that a hardware component failure may cause a security breach may be reliably considered to be less than 0.000001/hour of operation.

Reliability Plan - per individual specification

Quality Control - per individual specification

Service Life - per individual specification

Specified MTBF - per individual specification

3.2.4 Maintainability

The following maintainability features are required:

Self-Test

A manually activated self-test feature shall be provided, to be used as a means of detecting traps, exercising the equipment during troubleshooting, verifying proper performance, and performing self-test. Status-indicating capabilities shall be included. This feature shall be self-contained.

Mean-Time to Repair - per individual specification

Organizational Level Maintenance - per individual specification

Intermediate Level Maintenance - per individual specification

3.2.5 Environmental Conditions

Vibration

Operating: per individual specification

Non-Operating: (Requirements are for transportation, ergo transportation cases and shock mounts may be utilized).

Land Carriers:	1.5 g's	5-500 Hz
Air Carriers:	1.5 g's	17-28 Hz
	3.5 g's	44-500 Hz

Shock

Operating: per individual specification

Non-Operating:

- (a) Bench drop tests
- (b) 20 g's peak, 18 milliseconds

Temperature

Operating: Ambient air from 50 DT to 100 DT (10 C - 38 C)

Altitude/Pressure

Non-Operating: Sea level to 35,000 feet

Operating: per individual specification

Humidity

Operating: 5% to 95%

Transportability

The system hardware shall be designed and constructed to withstand military and commercial transport via air, rail and truck to installation sites without degradation of the equipment performance.

3.3 Design and Construction

The equipment shall be designed and constructed to meet the general design requirements of the prime governing specifications and the general requirements for electronic equipment as specified in MIL-STD-454D.

3.3.1 Materials Processes and Parts

Materials, processes and parts for use in the manufacture of the equipment shall be in accordance with the prime governing specification and MIL-STD-454D, Requirement 4. Parts shall be in accordance with MIL-STD-454, Requirement 64.

3.3.2 Compromising Emanations Control (TEMPEST) and Electromagnetic Compatibility

Because system equipment will be connected to communications lines and controllers of various security levels and need-to-know categories, all system equipment shall be designed to reduce compromising emanations below the applicable radiation and conduction limits of NACSEM 5100 as modified by AFNAG-9A unless otherwise specified. All TEMPEST design shall follow the principles outlined in NACSEM 5200. Installation of system equipment shall conform to the applicable

requirements of AFNAG-5B. Red/black separation requirements are described in MIL-HDBK-232.

Electromagnetic interference criteria shall be in accordance with the emission and susceptibility requirements established by MIL-STD-461A(Notice3). The interference reduction guide for Design Engineers, DDC AD619666 and AD619667 shall be used to establish bonding and shielding criteria and to optimize interference suppression circuits.

3.3.3 Nameplates and Product Marking

The identification nomenclature marking and labeling for the several elements of the system shall be provided in accordance with MIL-E-4158E., paragraph 3.6.

3.3.4 Workmanship

All units, component parts, and accessories supplied shall be in accordance with MIL-STD-454D, Requirement 9.

3.3.5 Interchangeability

All units shall be designed and constructed in accordance with the provisions of MIL-STD-454D, Requirement 7, concerning the use and selection of interchangeability items, and with the definitions contained in MIL-STD-280A. Like units, assemblies, subassemblies, and replaceable parts shall be physically and functionally interchangeable, without modifications of such items or the equipment. Module replacement shall be possible without removal of adjacent modules. Software modules shall be interchangeable among like processors.

3.3.6 System Safety

System safety engineering principles shall be applied throughout the design, development manufacture, test, checkout, operation and maintenance of all systems/equipment in accordance with MIL-STD-454D, Requirement 1 and 8 (class 1 equipment).

3.3.7 Human Engineering

Human engineering design criteria and principles shall be applied in accordance with MIL-H-46855A, MIL-STD-1472, and AFSC Design Handbook 1-3 in the design of the equipment, computer programs, and facilities so to achieve safe, reliable, and effective performance by operator, maintenance and control personnel, and to optimize personnel skill requirements and training time.

3.4 Documentation

Documentation required for reviews and audits shall be provided in accordance with MIL-STD-1521 (USAF).

3.5 Logistics

The Logistics Disciplines of DoD Directive 3100.35G shall be integrated into the design and engineering constraints. Such integration shall include provisions for logistics support for the system's life cycle.

Maintenance, Supply Support, Support Equipment and Facilities are normally per the individual Equipment Specifications.

3.6 Personnel and Training

The communications processor system shall be designed to be maintainable by Air Force skill level 5 personnel after completion of formal training course (s).

3.6.1 Training

Formal training requirements will include training equipment requirements. This formal training includes theory and hands-on equipment training.

3.7 Major Component Characteristics

The secure communications system is composed of a SPM and commercially available components (3.1.3). The properties of the SPM are delineated in 3.1 and 3.2. The commercially available components will have features normally considered necessary in the area of application. This paragraph lists the properties of the commercial

units required by their interconnection with the SPM in a secure system.

3.7.1 SPM

Properties as specified in this document.

3.7.2 Processor

The processor when used in conjunction with the SPM shall implement the following functions:

- 1) Call/Return;
- 2) Clear Fast Access Store;
- 3) Interprocessor Signal;
- 4) Dispatch; and
- 5) Argument validation.

The processor and SPM shall provide sufficient mechanism so that the following instructions may be restricted to the most privileged domain of execution (ring zero):

- 1) Clear Fast Access Store;
- 2) Interprocessor Signal;
- 3) Dispatch;
- 4) instructions which control the setting of trap and interrupt vectors as well as the recognition and control of traps and interrupts;
- 5) instructions which control the state of the SPM (except for Call/Return);
- 6) instructions which may change microcode either within a processor or device; and
- 7) any function which influence the SPM's interpretation of the descriptor structure defined by the security kernel.

The processor shall deliver all memory addresses identified as addresses to the SPM. When addresses are generated by the hardware in response to special conditions (traps and interrupts), the special nature of the addresses shall be signalled to the SPM. If the explicit-names structure is implemented (3.1.2.3.1.1), the I/O name shall be delivered to the SPM identified as an I/O name.

The processor shall deliver all information concerning its internal state to the SPM necessary to implement the functions of this specification. It shall at least signal:

- 1) Order initiation;
- 2) Occurrence of trap or interrupt condition;
- 3) I/O initiation for explicit names;
- 4) Mode of access (read, write, execute, call, return); and
- 5) SPM control instructions.

The processor shall be capable of being controlled from the SPM. The SPM shall be able to generate processor traps.

The processor shall be capable of being restarted after a trap (ref. section 3.1.2.1.6.2).

The processor shall provide an indivisible Test-Read-Alter-Rewrite memory instruction so that software may cooperate harmoniously in multiprocessor systems.

3.7.3 Memory

Memory subsystems shall have the capability to implement indivisible Read-Alter-Rewrite functions that can be used by the software as a nonambiguous semaphore mechanism.

3.7.4 Controllers

If the mapped I/O is implemented (3.1.2.3.1), a controller shall present to the SPM the identification of a device requesting transfer. A controller shall provide storage for each of the devices it controls so that any interdevice (and hence potentially interprocess)

interference is discernable only in the time domain.

3.8 Precedence

This document takes precedence over all documents in section 2.0 when a specific conflict in specification arises.

4.0 Quality Assurance Provisions

4.1 General

Requirements for formal tests/verifications of the secure communications processor performance, design characteristics and operability shall be per the individual specification. Tests/verifications will include design evaluation and operational capability verification.

4.1.1 Responsibility for Tests

Unless otherwise specified, the contractor is responsible for verifying that all specifications requirements have been satisfied. The contractor may utilize his own or other facilities acceptable to the government. The procuring agency will monitor the contractor's effort and reserves the right to perform any of the verifications set forth in the specification. The Government also reserves the right to designate special existing Government facilities for the performance of specialized tests such as TEMPEST and/or nuclear environmental testing.

4.1.2 Special Tests

- A. Reliability Demonstration. Reliability demonstration shall be per the individual specification.
- B. Probabilistic Measure of Security Compromise. Collection, recording and analysis of all failure data during testing shall be performed. Tests or analyses should establish to a high degree of statistical confidence that the specified probabilistic measure of security compromise is satisfied.
- C. Security Features. The contractor shall rigorously demonstrate that all security features of each secure data communications processor are functionally correct.
- D. Qualification Tests. Per individual specification.

4.2 Quality Conformance

The contractor shall perform the following inspections, analyses and tests to verify the requirements of Section 3 of this specification.

4.2.1 Interface Definitions

The contractor shall test and demonstrate the performance of the dedicated communication lines, electrical interfaces and logical protocols. The contractor shall also demonstrate the standard peripheral interfaces.

4.2.2 Throughput

The contractor shall demonstrate that the security controls do not degrade throughput by greater than 25%.

4.2.3 Physical Characteristics

The contractor shall verify by analysis or tests the following physical characteristics.

- A. Weight limitations
- B. Cooling requirements
- C. Maintainability requirements

4.2.4 Environmental Conditions

Each production unit shall be subjected to the following environmental acceptance test per the individual specification.

- A. Temperature
- B. Humidity
- C. Vibration
- D. Shock
- E. Nuclear environment
- F. Electromagnetic compatibility
- G. TEMPEST

5.0 Preparation for Delivery

The secure communications processor shall be delivered as an integrated unit after appropriate testing in accordance with 4.0. When used as the front end to a host processor, it will be delivered in an integrated system with the host processor.

6.0 Notes

6.1 Descriptor Format and Field Encoding

The formats given for descriptors, and other constructs, are logical ones, intended to identify required information. A specific implementation of this specification may rearrange and reorganize these formats as long as pertinent information is preserved. For instance, the A field of a descriptor may be encoded into marginally useful encodings of the permissions field. One such permission is (R = OFF) and (E = ON) and (W = ON). A second instance of an alternative encoding is the MT field of an I/O descriptor. This field may be encoded into the T field yielding distinct descriptor types for premapped and mapped I/O devices.

6.2 Virtual Device Addresses

In section 3.1.2.3.2 an option was discussed that uses one descriptor tree for both I/O and memory descriptors. If such an option is implemented, there must be a well defined mapping between virtual I/O device names and generalized virtual addresses so that the descriptor for a specific I/O device can be located in the descriptor tree. Since the size of virtual device addresses (for the explicit I/O naming structure) is often much smaller than the size of virtual memory addresses, the secure data communications processor (particularly the SPM) shall perform appropriate transformations of the virtual device address so that the device address may be interpreted in the same name space as virtual memory addresses. For example, assume virtual memory addresses are 18 bits wide, structured as 2^{*6} segments of 2^{*6} pages of 2^{*6} words, and that the virtual device address is 6 bits wide. One transformation (ref. Figure 10) assumes one segment per device and forms a new virtual address by concatenating 12 low-order zeroes to the original device name. A second method (ref. Figure 11), more conservative of virtual address space, assumes the 64th segment (e.g., segment number 63) is composed of "pages", each page descriptor being a direct device descriptor. For this case the new virtual device address may be formed by concatenating 6 high-order ones and 6 low-order zeroes to the original device name.

The device name transformation algorithm for a specific implementation is determined by the following factors:

- 1) the size of the virtual memory address;
- 2) the size of the virtual device address;

- 3) the intended organization of the descriptor tree (interpretation of virtual addresses); and
- 4) the intended placement of devices in the virtual address space. For example, devices may be placed either as segments or pages (see above discussion).

6.3 Virtual Address Pointers

It is highly desirable that the secure data communications processor support a general form of virtual address pointer. A logical format is diagrammed in Figure 15. It is intended that this pointer replace, in function, indirect address constructs in memory (and in processor address registers, if possible). The interpretation of the fields shall be:

Virtual address: The VA field provides a virtual indirect address of whatever size and precision supported by the processor. The specific use of this field in effective address formation is, of course, processor specific.

Validation ring: The VR field provides a software defined ring number at which the address provided in the VA field should be evaluated. The VR field should be of sufficient size and precision to accommodate any ring number defined by the system. Before the VA field is used as an address by the processor, the SPM's Reff must be updated to $\max(\text{Reff}, \text{VR}, \text{Rl})$ where Rl is taken from the descriptor controlling access to the resource in which the pointer is contained.

Directed trap: The DT field provides a software defined mechanism for causing hardware traps on access (as a pointer) to a virtual address pointer. The DT field shall have at least two values (ON and OFF): if ON a processor trap is generated before the VA is used as an address, if OFF the indirect operations proceeds as specified above.

In conjunction containing providing virtual address pointers containing ring numbers, it would be useful if hardware supported a special set of pointer copying instructions (or pointer load instructions). A pointer copy instruction would copy the pointer, replacing the VR field of the pointer with the maximum of Reff and the previous value of the VR field in the pointer. Thus, whenever an inner ring procedure copies a pointer from a segment in an outer ring to the inner ring, the effective ring of all references made through that pointer will never be less than the ring in which the pointer was originally located. Without a pointer copying instruction, software



Figure 15. Virtual Address Pointer

in the inner ring must check the validity (ring number) of the VR field of any copied pointer to make sure that the value of VR is never less than the ring from which the pointer was copied.

6.4 Process/Event Reporting

Operation complete events should be directed to the process initiating the operation. Security kernels require sophisticated process structures. Embedding the mechanism for reporting external events in a well-defined manner directly to the process requires the following capabilities of both the processor and SPM:

- 1) both processor and SPM must understand the concept of process and all attributes associated with a process: the processor (with SPM) must be capable of performing a sophisticated dispatch algorithm;
- 2) the hardware must provide support for arbitrary scheduling algorithms; and
- 3) the hardware must provide sophisticated primitives for process cooperation (PV primitives, monitors, or their equivalent).

Conceptually the SPM could be designed to include this functionality. Such functionality would be impacted in a major way by software design considerations. Insufficient understanding of these implications precludes the requirement of this functionality in the SPM for a secure system.

Within the constraints of a particular application or software environment, the implementation of a direct event/process reporting mechanism in the SPM would be highly desirable.

6.5 Simple Indirect Descriptor

The semantics of the indirect descriptor (ref. section 3.1.2.1.1, Address Translation) specifies that these descriptors point to an array of lower level descriptors. An important special case is the simple indirect descriptor: a descriptor that points to a single lower level descriptor. The location field of this descriptor type gives the physical memory address of the lower level descriptor and the limit field may be ignored. No information from the fields in the virtual address are needed to locate the next level descriptor once the simple indirect descriptor is found.

This construct is very useful in to support sharing the construction of segmented, unpagged virtual memory systems. Such a memory structure is diagrammed in Figure 16.

A secure data communications processor, satisfying this specification, need not implement a simple indirect descriptor construct. However, its presence increases the range of application of a particular SPM implementation.

6.6 Page Descriptors

In the full three-level descriptor structure illustrated in Figure 4, the access control information is almost always placed in the segment descriptors. This access control information includes the ring brackets, mode fields, call limiter, and limit field. (The limit field is included in this sense because the segment descriptor can specify, to the resolution of one page, the size of the segment. The simplest memory management scheme would not attempt to further subdivide memory.) The only remaining information in a page descriptor that actually needs to be there and cannot be inferred by the SPM is the physical address and directed trap (DT) field. If, in a given implementation, it is known that a structure such as that in Figure 4 will be the only structure used, the actual size of the page descriptor can be much smaller than that of the segment descriptor. Since there must be at least one page descriptor for every page of physical memory (and most likely one page descriptor with DT field set for every page of every active segment not in physical memory) the page descriptors usually use up a great deal of physical memory. A significant improvement in performance may be realized if the size of page descriptors can be reduced, for example, from 4 to 2 words of real memory. There may also be an impact on performance due to fewer accesses to memory for page descriptors.

Another method for reducing the size of page descriptors, that would work in a more general structure, would be to include a field in the first word of the descriptor that specifies whether that descriptor is "short" or "long". The simplest hardware implementation of this scheme might be to define a descriptor structure such that the physical address, directed trap, and short/long indicator were packed in the first one or two words, with the additional information in subsequent words. By examining the first word fetched, the SPM would know the descriptor format and either inhibit or simply ignore subsequent word fetches.

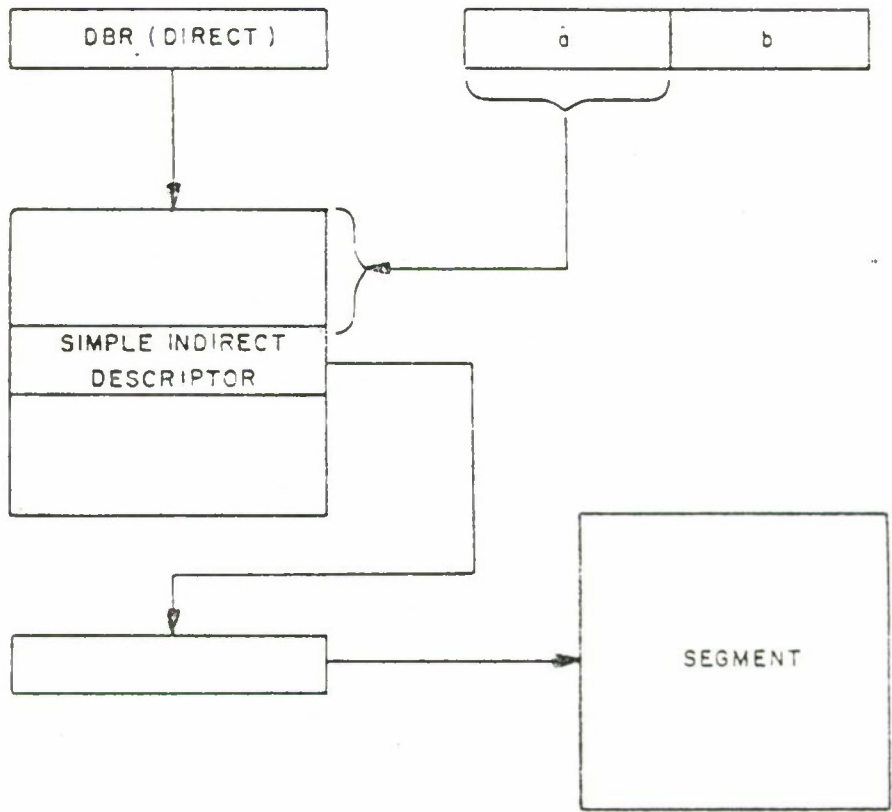


Figure 16. Simple Indirect Descriptor

6.7 Physical Address Size and Precision

Each descriptor that describes a construct located in memory shall have sufficient size and precision in its location and limit fields so that the construct may be addressed anywhere in physical memory without waste of memory space. While memory space waste is the responsibility of hardware, the hardware structure shall not preclude efficient space management by software. A specific implementation must trade off software flexibility against SPM cost. In this case, SPM cost may be measured with respect to descriptor size. The most flexible SPM implementation provides location and limit resolution to the word or byte.

While specific requirements are implementation specific, bounds can be placed on location size and precision requirements.

Precision: A descriptor need not address memory with greater precision than the size of the smallest allocatable element: in most implementations this element will be a descriptor. It shall at least have a precision sufficient to precisely address and limit the minimum of: the smallest array of descriptors, the smallest area of allocatable memory, or the smallest set of device registers for the devices in memory I/O naming structure (if applicable). The precision of a descriptor may vary with its level of interpretation. For example, direct memory descriptors at the segment and page levels of interpretation may have differing precision. This may be accomplished, using a fixed descriptor format, by allowing the significance of the location and limit fields to be a function of the level of interpretation or additional encodings of the type field.

Size: Each location field shall be capable of addressing all of physical memory at the precision of the descriptor. For example, if the precision of the descriptor is the size of a descriptor (several bytes) then the location field shall have sufficient size to address all of memory in increments of its precision: the size of a descriptor. Each limit field shall have sufficient size to delimit the largest addressable construct at the precision of the descriptor. Using the above example, if the precision of the descriptor is the size of a descriptor, then the limit field shall have sufficient size to delimit the largest construct addressable from the descriptor (for instance, a page) modulo the size of a descriptor.

6.8 SPM Algorithmic Description

A sample flowchart detailing the address translation and access control functions of the SPM is shown in Figure 17. This flowchart is not intended to define the implementation, but rather to present an algorithm that approximates the functionality of the SPM. The flowchart is constructed as a procedure named REF that is called by the CPU or I/O device on each memory reference. The arguments to REF, appearing at the top of the flowchart, have the following meaning:

VADDR - the virtual address of the memory reference.

MODE - one of the five values: Read, Execute, Write, Call, Return. The MODE argument specifies the type of permission required. If TYPE (see below) is I/O, the MODE is either Read or Write.

DATA - either an input argument, in which case this is data to be written into memory at the virtual address VADDR, or an output argument, in which case the contents of the memory location read is returned here.

RESET - can be 0 or have the value RESET or IND. If RESET, the SPM is instructed to reset R_{eff} to the value of R_{cur} before performing any access check or address translation. If 0, R_{eff} is left unchanged from the previous memory reference, and may possibly be further incremented in subsequent descriptor fetches. If IND, the reference is a fetch of an indirect address (virtual address pointer) by the CPU. This value instructs the SPM to examine the DT and VR fields of the virtual address pointer in determining whether a trap should be generated or whether to update R_{eff}.

DEVICE - identifies the active module requesting the reference. It is either the value CPU, indicating the processor is requesting the reference, or an I/O device number.

TYPE - is either I/O, indicating that the reference is to an I/O device and the instruction being executed is an I/O instruction, or MEMORY, indicating that the reference is to memory.

In order to implement the algorithm in the flowchart, several internal registers for storage of permanent and temporary values have been created. These are illustrated in Figure 18 and are defined as follows:

CALLING SEQUENCE: REF (VADDR, MODE, DATA, RESET, DEVICE, TYPE)

VADDR = VIRTUAL ADDRESS TO REFERENCE

MODE = READ, EXECUTE, WRITE, CALL, OR RETURN

DATA = DATA TO BE READ OR WRITTEN

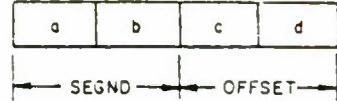
RESET = IF "RESET", RESET Ref. IF "IND",

THIS IS INDIRECT FETCH.

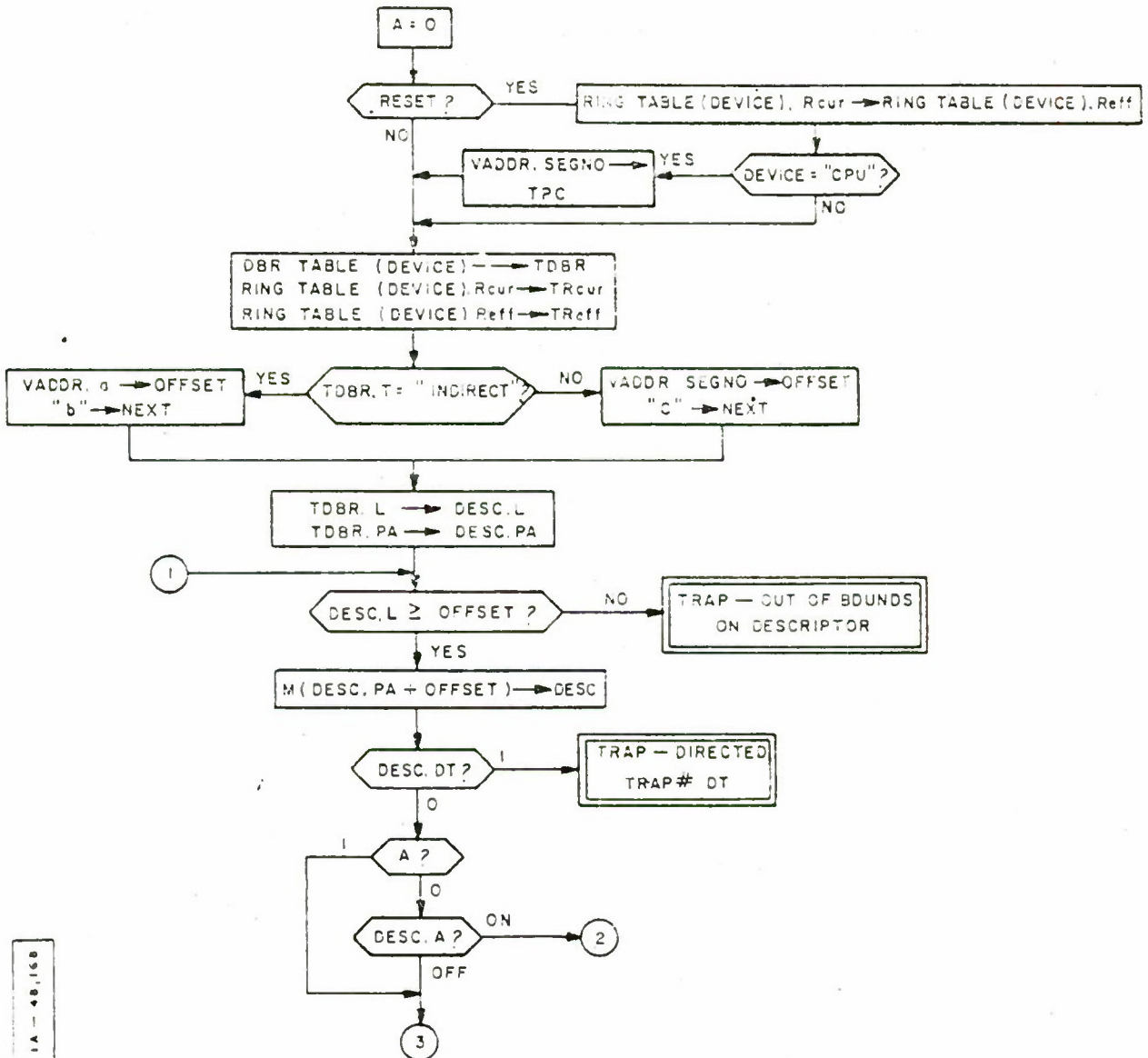
DEVICE = "CPU" OR I/O DEVICE NUMBER (REQUESTER)

TYPE = "I/O" OR "MEMORY" (OBJECT)

VADDR =

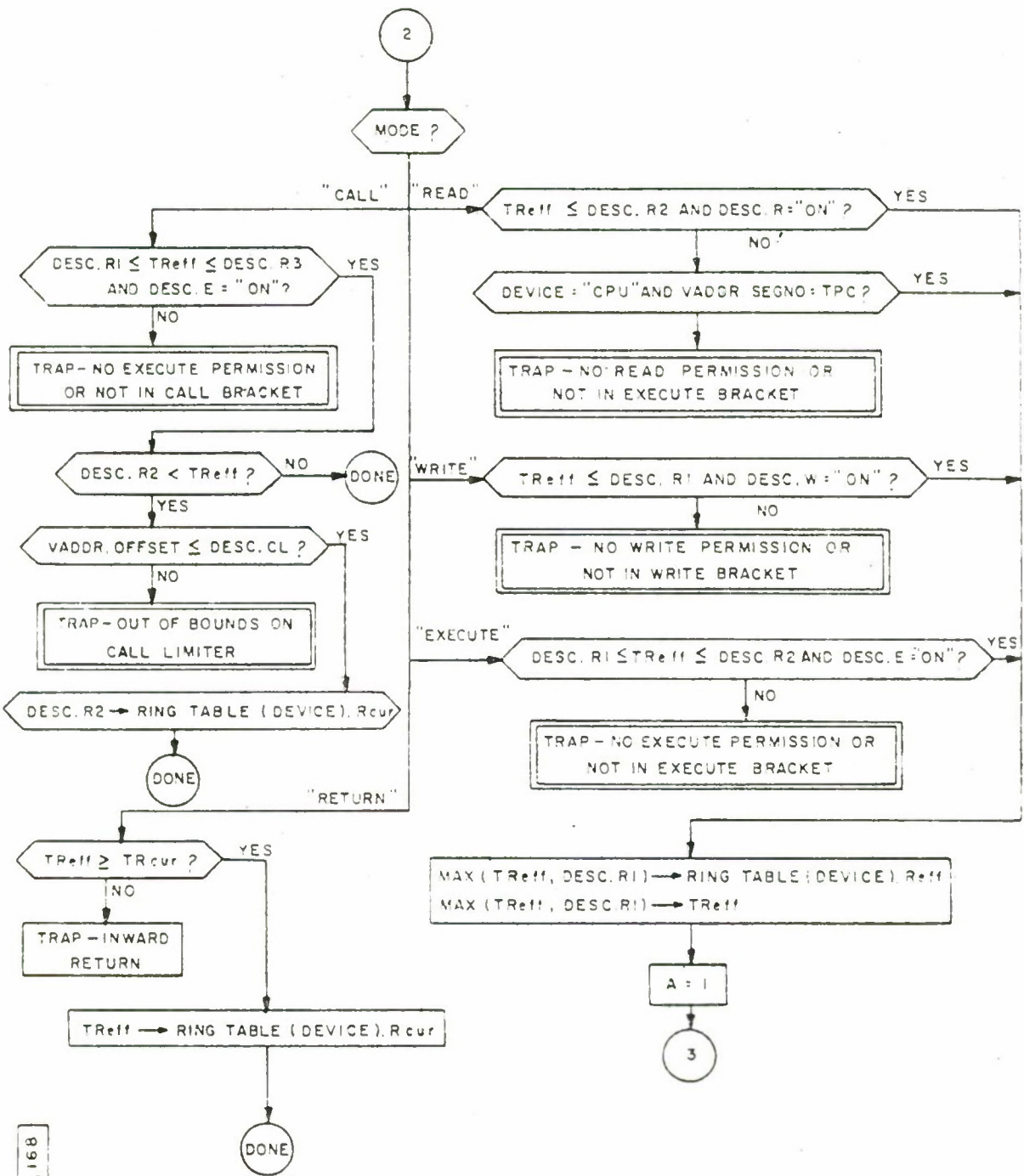


DATA =



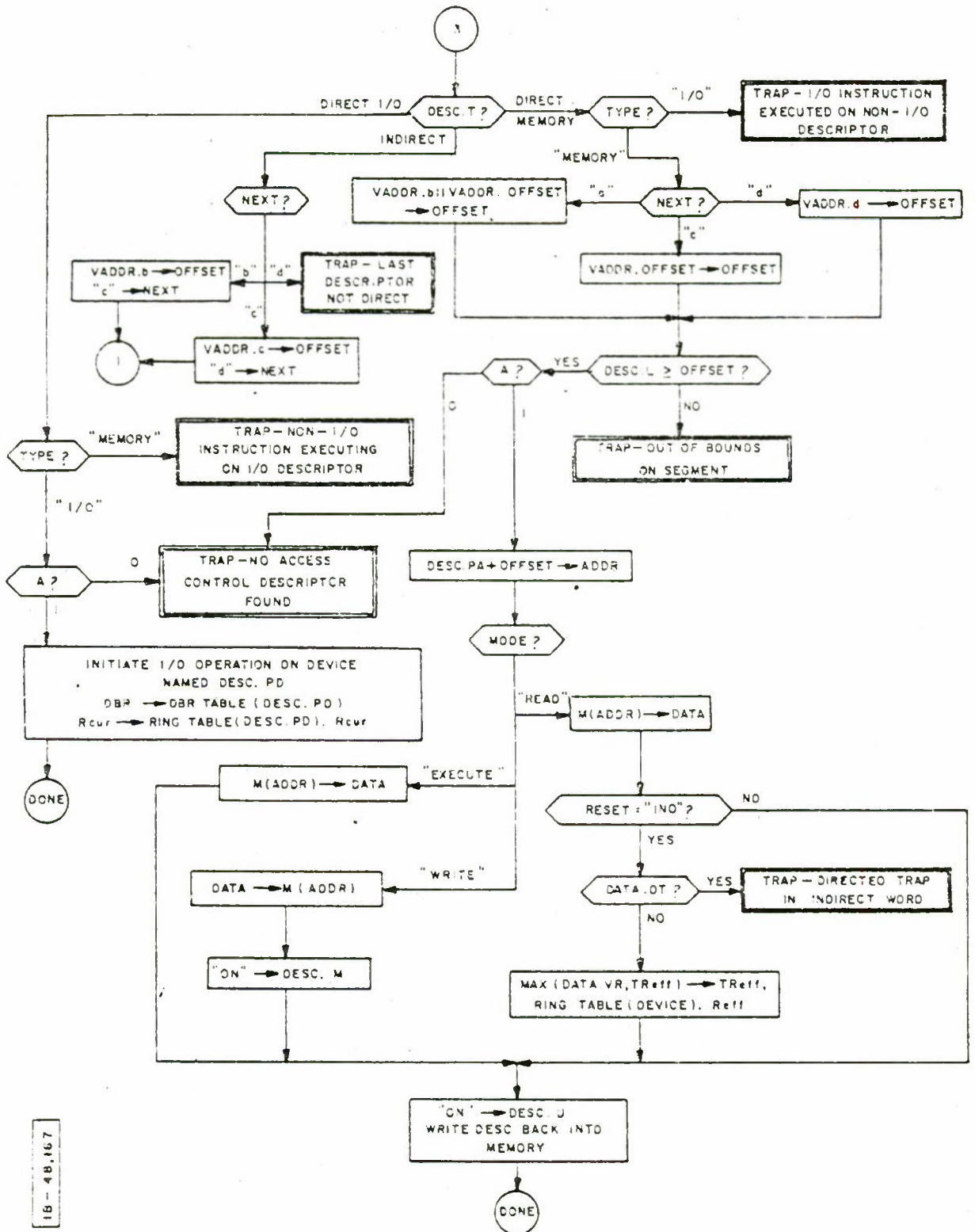
IA - 48,168

Figure 17. SPM Flowchart



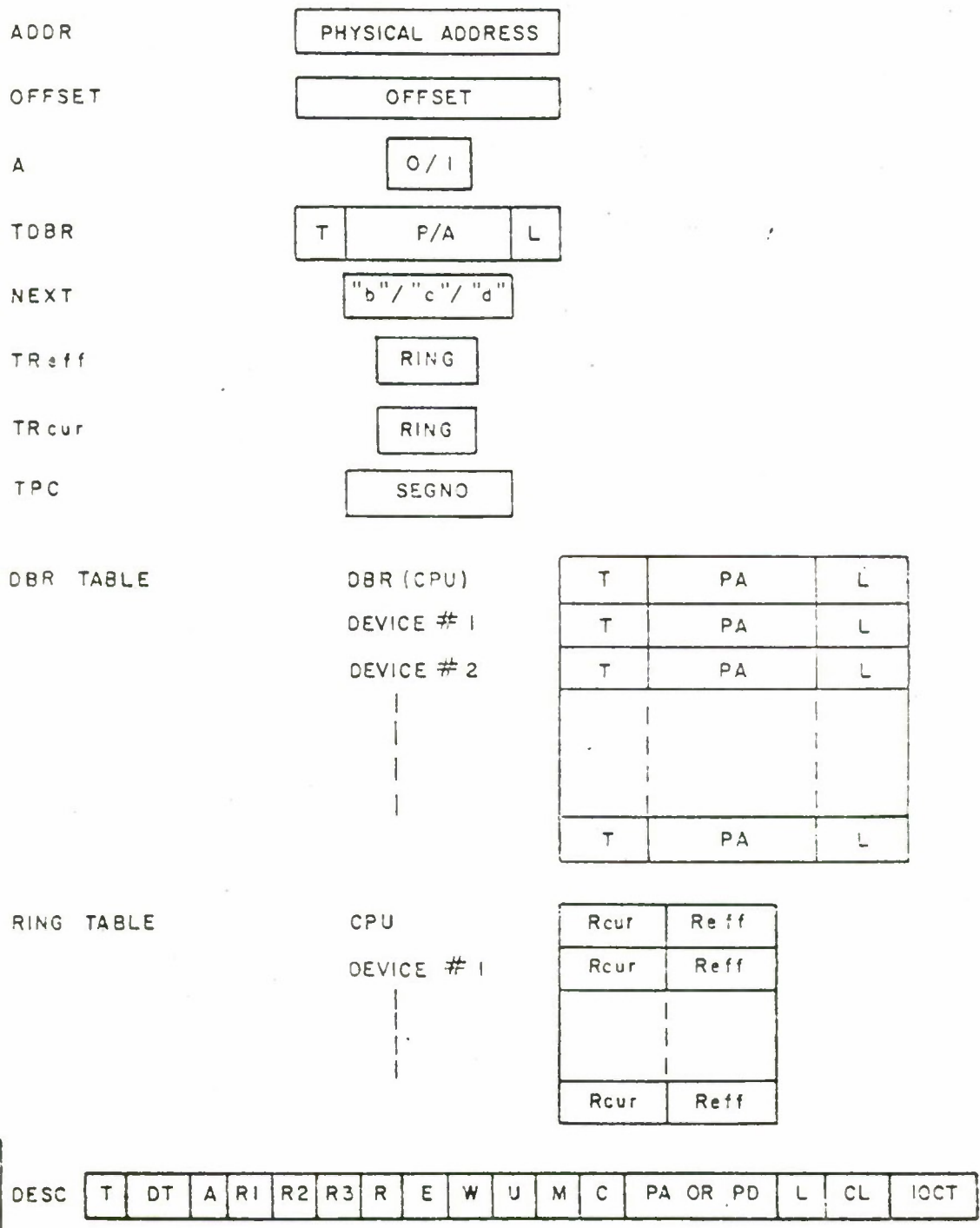
1A-48,16B

Figure 17. SPM Flowchart (Continued)



IB-48,167

Figure 17. SPM Flowchart (Concluded)



1A - 48, 165

Figure 18. Internal SPM Storage

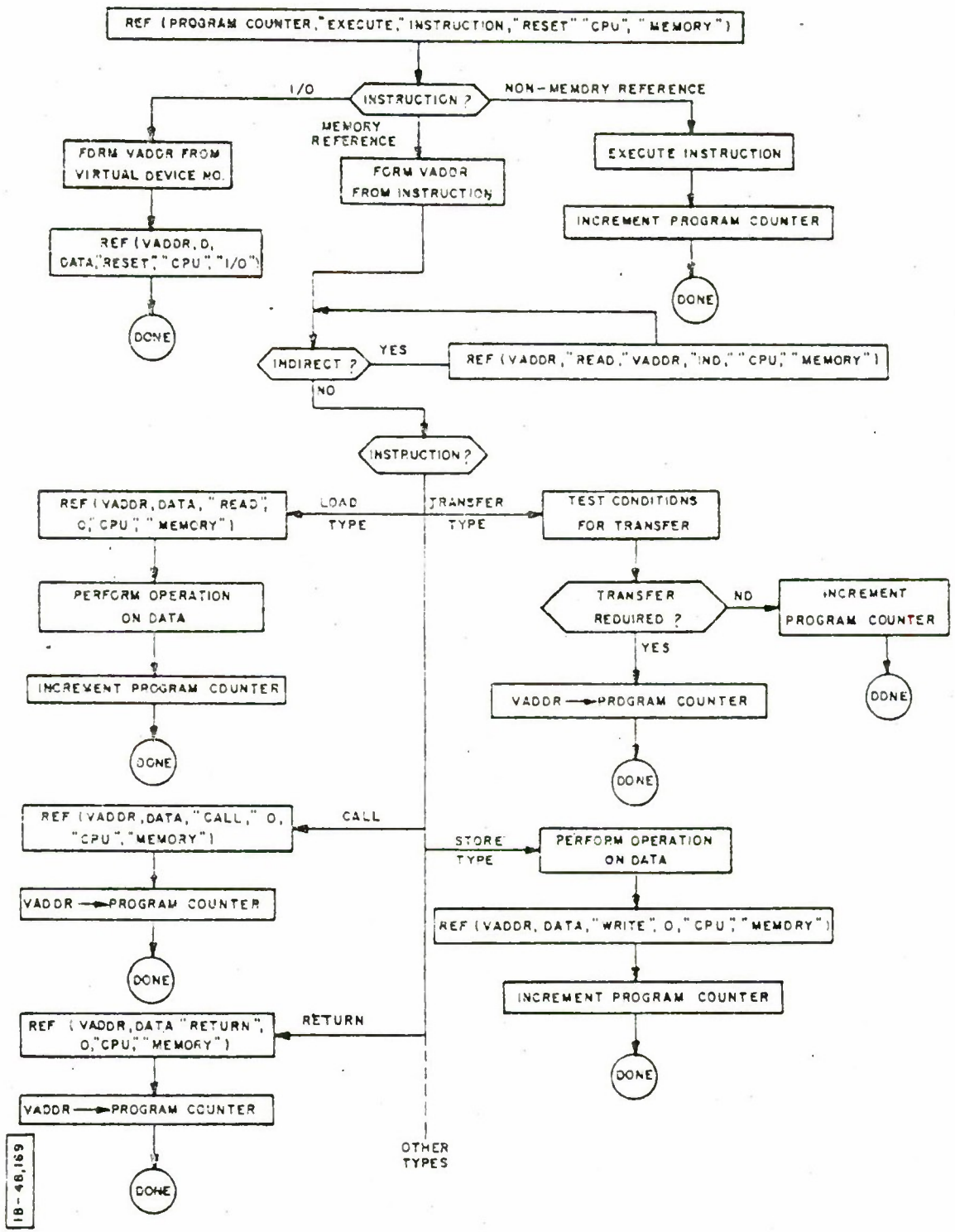


Figure 19. Typical Instruction Loop

- ADDR - temporary register in which a physical memory address is saved.
- OFFSET - temporary register in which an offset into a page or descriptor table is saved.
- A - a flag that is set to TRUE when a descriptor has been encountered with its A field set. If, after the final descriptor fetch, this flag is still FALSE, a trap will be generated.
- TDBR - a register, structured like the DBR, in which to temporarily store the current DBR that applies to the reference. See Figure 7 for a description of the internal fields.
- TReff, TRcur - temporary holding registers for the value of Rcur and Reff used in this reference.
- NEXT - has the value "b", "c", or "d", indicating which portion of VADDR is to be used as the next offset into a descriptor table or page.
- TPC - temporary storage for the segment number in which the current instruction resides. This value is loaded with VADDR.SEGNO at each RESET. Any subsequent read from memory (until the next RESET) is allowed from this segment even though there may be no explicit read permission.
- DBR TABLE - is a table containing one DBR value for the CPU and each device currently active. This table may or may not have a physical realization as memory internal to the SPM. A new DBR is loaded into this table when an I/O operation is initiated on a device.
- RING TABLE - is a table corresponding to the DBR table that contains the values of Reff and Rcur for the CPU and for each active device. The values of Reff must be saved, in addition to Rcur, because different devices may access memory between requests to reset Reff for any one device.
- DESC - is internal storage for a descriptor that has been fetched from memory. The internal fields are described in Figure 10 and Figure 12.

The algorithm represented by the flowchart makes the following assumptions:

I/O is mapped (see 3.1.2.2.1.2).

The same descriptor tree is used for both I/O descriptors and memory descriptors (see 3.1.2.3.2).

The Explicit names structure is used for I/O (see 3.1.2.3.1.1)

Indirect addresses (virtual address pointers) contain ring number and directed trap fields (see 3.1.2.1.2.2 and 6.3).

The algorithm is not reentrant. This means that the SPM is not interrupted by a request for another memory reference until DONE has been reached.

The intended use of REF by the processor in a typical instruction cycle is illustrated in Figure 19. This figure is only meant to be illustrative and does not restrict the implementation. An actual processor instruction cycle may be somewhat different in the placement of various tests and computations. Each call to REF in this figure represents either a memory reference or an I/O device reference. Note that the algorithm defining REF keeps track of Reff for each active device. This allows an I/O device access to memory at any time between successive calls to REF in Figure 19.

6.9 Switchable Field Widths

In 3.1.2.1.1 on address translation the requirement was stated that the breakdown of the virtual address into four fields a,b,c,d be carefully chosen to be optimal for the intended application. Since it may be undesirable to construct hardware best suited for only one application, an optional implementation can allow for more than one specific breakdown of the virtual address into four fields. For example, the optimal segment size for one application may be 2048 words, whereas the most useful segment size for another application may be 128 words. The change from one interpretation of virtual address to another can be implemented through the use of switches or jumpers, or more dynamically in software through the setting of certain modes in the SPM or use of special fields in the descriptors or DBR. In general, only a small number (two or three) choices of virtual address breakdown need be available to handle all foreseeable applications.

6.10 Stack Base Register

In 3.1.2.1.6.1 the call order was discussed, with some references to its use. Normally, non-gate procedures with an execute bracket of more than one ring (i.e., $R1 \neq R2$ and $R2 = R3$) must be structured so that they will execute properly in any of the rings from which they

are entered. Since, in a multi-ring environment there must be a separate stack segment for each ring, there is a requirement that such procedures must obtain the address of the stack segment before beginning execution. This requirement is easily satisfied in software by defining a standardized calling sequence that provides the stack segment number in some register. The called procedure can safely believe that this register truly points to the proper stack segment because the caller, who provided the value, is running in the same ring as the called procedure.

If the procedure is a gate, and it is called from above R2, then the value of such a stack segment number cannot be believed because the caller was running in a higher ring and might try to spoof the gate procedure into using the wrong stack. However, the stack number is not really required in this case because the called procedure can assume that it is always executing at R2 when entered at its gate entry. (There is an assumption that R1 = R2 or that, by convention, software does not enter a gate procedure unless the current ring is R2 or greater.) Generally, when the current ring is known, the stack segment number can easily be determined by convention (e.g., the stack segment number for ring N is N).

A problem arises, though, when programming generality dictates that a gate procedure be programmed independent of the ring in which it is to be run. Another requirement for generality may be that a gate procedure be entered at the same location, via the call order, from any ring between R1 and R3. In order for such a procedure to find its stack, it must first determine from which ring it was called. If it was called from within its execute bracket, a register can be believed. If it was called from above R2, the stack is that for R2. The difficulty here is that computation of the stack address must take place on every call, and must not require use of temporary storage (i.e., the stack). Also, the fact that the value of the stack address for R2 must be programmed into the procedure makes the procedure dependent on the virtual address of a given segment. Moreover, it is difficult, when writing in a higher level language, to require that the intended ring of execution be specified to the compiler.

The problem is easily solved through the use of a simple SPM feature. Upon request, or perhaps automatically at every call, the SPM would create a virtual address pointer to the stack by using the current ring number as the segment number (or by some similar simple transformation), and load that pointer into a register through which software can indirect. In such a structure, all procedures could assume, for example, that a given register always points to the base of the stack for the current ring, regardless of which ring they were entered from.

A more general variant of this feature, required for multiple execution points within a process, is an SPM-maintained stack base register. Instead of simply using the current ring number to find the stack, the stack base register provides the base address of a set of stack segments. The current ring number is added to (or concatenated with) the stack base register to find the right stack segment. The stack base register itself is set by the kernel and is a function of which of several execution points for the current process is being invoked.

6.11 Ring Alarm Register

In a multi-ring environment each ring has a set of asynchronous interrupts that it has set up, such as timers or external devices. The interrupt vector, and the interrupts themselves, are all handled by the kernel. However, if the interrupt is one that is of interest to the outer ring, the kernel must invoke the appropriate outer ring interrupt handler. This is no problem if the interrupt occurs while the process is executing in that outer ring, since in that case the kernel can simply invoke the proper procedure and save the state information so that the outer ring will appear to have been asynchronously interrupted.

If the interrupt occurs while the process is executing in the kernel ring, particularly while in some kernel procedure that may not be interrupted arbitrarily, the actual interrupt must be deferred until the kernel is finished and ready to return to the outer ring. An unsatisfactory solution would be to defer interrupts while in the kernel using the hardware interrupt inhibit feature. (Hardware interrupt inhibit should only be used in parts of kernel interrupt handlers and other critical kernel code.) The software solution is to remember that the interrupt occurred, and then, just before returning to the outer ring, check if any interrupt handlers should be invoked. This procedure makes kernel calls indivisible from the outer ring's point of view.

The software solution has problems due to extra overhead at each return, and timing problems. The problem is more difficult if the interrupt is not for the ring to which the kernel is returning, but for an outer ring (e.g., an interrupt to be handled by ring 3 that occurs during ring 0). A very simple hardware feature solves the problem. A ring alarm register, maintained by the SPM, is loaded by the software interrupt handler when an interrupt occurs to which an outer ring is to respond. The number that is loaded is the ring that is to handle the interrupt. The SPM simply compares this ring number to the ring being returned to on all return orders and generates a ring alarm trap whenever the ring being returned to is greater than or equal to the ring alarm register. The ring alarm trap handler can then easily invoke the proper interrupt handler in the outer ring.

10.0 Appendix

10.1 Interconnection Unit

10.1.1 Intersystem Geometry

The connection of the secure communications processor to a host processor through an Interconnection Unit (IU) is shown in Figure 20. On the communications processor side the IU is addressable as absolute memory and as a I/O device. When addressed as absolute memory it may be the source or sink for data flow from an I/O device. When addressed as an I/O device, it may be set to various operating modes by the secure communications processor. If it is deemed necessary in a particular application, the range of absolute addresses assigned to the IU may be subdivided, and a connection path to the host processor implemented for each subdivision. Each such subdivision would be required to respond to its own I/O name.

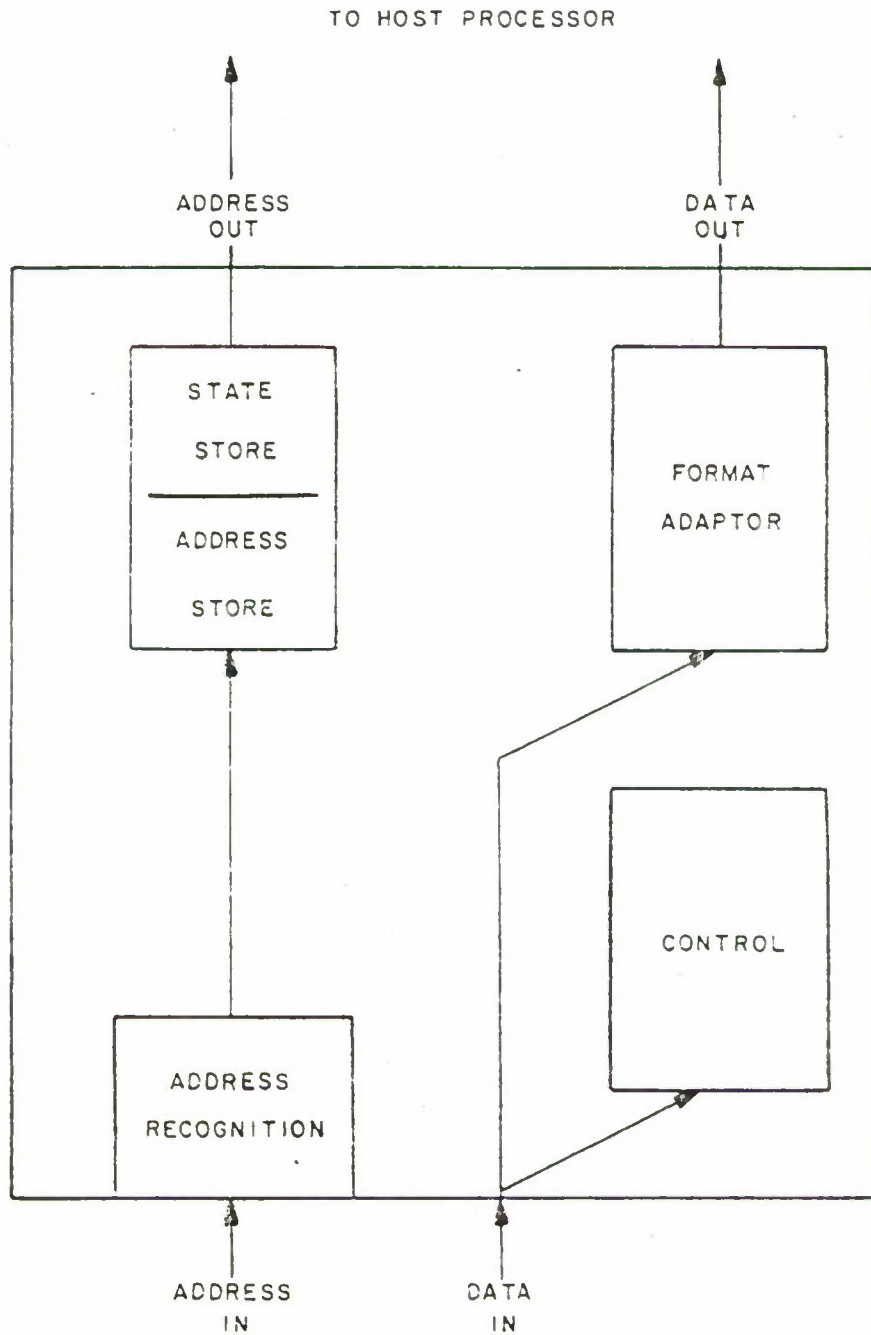
10.1.2 Internal Geometry

A functional diagram of a connection path in the IU is shown in Figure 21. This diagram is not intended to constrain the implementation of an IU. The format adapter is intended to match the formats of the secure communications system and the host system. The storage element would contain state or mode information provided by the secure communication processor, and any addresses required for interaction with the host processor. The address response logic must respond to the assigned range of absolute addresses and the appropriate I/O name.

10.1.3 Interprocessor Communication

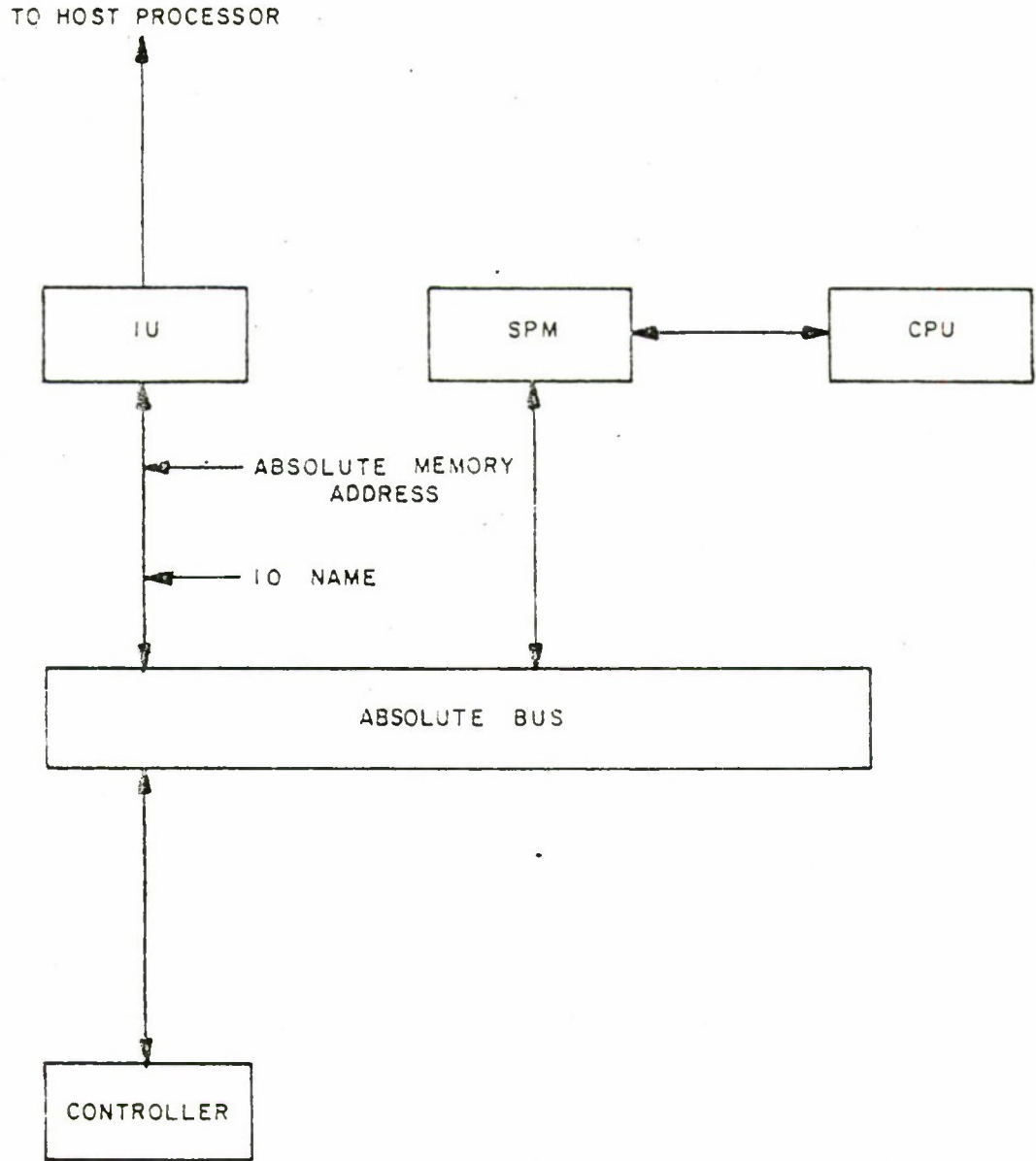
The IU shall extend the interprocessor signal function of the secure data communications processor to include the capability for interprocessor interrupts between the host processor and a secure data communications processor.

To facilitate the transference of information between the secure communications processor and the host processor, the IU will be able to be set so that all of host processor memory addresses can be accessed by the secure communications processor. In addition, all information bits in a host processor memory unit shall be accessible from the secure data communications processor.



IA - 48,163

Figure 20. IU Function



IA - 48, 162

Figure 21. IU Intersystem Geometry