Publication Number B29-381

OS/32-ST
PROGRAM LOGIC MANUAL
VOLUME 1

The information contained in this manual is subject to design change and product improvement.

THIS MANUAL CONTAINS PROPRIETARY INFORMATION AND IS SUPPLIED BY INTERDATA FOR THE SOLE PURPOSE OF USING AND MAINTAINING INTERDATA SUPPLIED EQUIPMENT AND SHALL NOT BE USED FOR ANY OTHER PURPOSE UNLESS SPECIFICALLY AUTHORIZED IN WRITING.

INTERDATA®

Subsidiary of PERKIN-ELMER
Oceanport, New Jersey 07757, U.S.A.

© INTERDATA INC., 1974
All Rights Reserved
Printed In U.S.A.
October 1974

OS/32 ST
PROGRAM LOGIC MANUAL

VOLUME I

TABLE OF CONTENTS

CHAPTER 1		INTRODUCTION	1-1
CHAPTER 2		SYSTEM STRUCTURE	2-1
	2.1	INTRODUCTION	2-1
	2.2	EXECUTIVE	2-1
	2.2.1	Task Management	2-1
	2.2.2	Executive Services	2-1
	2.2.3	Internal Interrupts	2-5
	2.2.4	Event Service Handler	2-5
	2.2.5	Crash Handler	2-7
	2.2.6	System Journal	2-7
	2.3	I/O SYSTEM	2-7
	2.3.1	Device/Volume Mnemonic Table (DMT/VMT)	2-7
	2.3.2	Logical Unit Table (LTAB)	2-9
	2.3.3	Device Control Block (DCB)	2-9
	2.3.4	Channel Control Block (CCB) and Interrupt Service Pointer Table (ISPTAB)	2-9
	2.3.5	SVC 1 Processor	2-9
	2.3.6	Drivers	2-9
	2.3.7	Termination Event Coordination Table (EVT)	2-10
	2.4	COMMAND PROCESSOR	2-10
	2.4.1	Command Processing	2-10
	2.4.2	Command Substitution System (CSS)	2-10
	2.4.3	Loader	2-10
	2.4.4	Direct Access Support	2-11
	2.4.5	Console Support	2-11
	2.5	FILE MANAGEMENT	2-11
	2.5.1	SVC 7 Processor	2-11
	2.5.2	Bit Map and Directory Handler	2-12
	2.5.3	Contiguous File Access Method	2-12
	2.5.4	Chain File Access Method	2-12
	2.5.5	File/Volume Utility Routines	2-12
	2.6	FLOATING POINT TRAPS	2-12
CHAPTER 3		SYSTEM CONVENTIONS	3-1
	3.1	SYSTEM STATES	3-1
	3.1.1	User Task State (UT)	3-1
	3.1.2	Executive Task State (ET)	3-1
	3.1.3	Reentrant System State (RS)	3-1
	3.1.4	Reentrant System State, Alternate Save Area (RSA)	3-2
	3.1.5	Event Service State (ES)	3-2

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

3.1.6 Non-reentrant System State (NS) 3-2
3.1.7 Non-reentrant System State, User Register Set (NSU) 3-3
3.1.8 Interrupt Service State (IS) 3-3
3.2 SVC CONVENTIONS 3-4
3.3 INTERNAL INTERRUPT CONVENTIONS 3-5
3.4 SUBROUTINE CONVENTIONS 3-5
3.4.1 RS, RSA and NSU Subroutines 3-5
3.4.2 NS Subroutines 3-5
3.4.3 Calling Sequences 3-5
3.4.4 Exits 3-5
3.5 GENERAL NAMING CONVENTIONS 3-6
3.5.1 Data Structures 3-6
3.5.2 Bits 3-6

CHAPTER 4 EXECUTIVE DESCRIPTION 4-1
4.1 TASK MANAGEMENT 4-1
4.1.1 Task Control 4-1
4.1.2 Task Management Facilities 4-2
4.1.2.1 Dispatch Current Task (TMDISP, TMRDISP) . . 4-2
4.1.2.2 Suspend the Current Task (TMSTOP) 4-2
4.1.2.3 Chain (TMCHN) 4-2
4.1.2.4 Unchain (TMUCHN) 4-3
4.1.2.5 Enter System State (TMRSIN, TMRSNIN, TMRSAIN) 4-3
4.1.2.6 Exit From System State (TMRSOUT, TMRSNOUT, TMRSAOUT) 4-3
4.1.2.7 Dispatch From Top of EVT Queue (EVDISP) . . 4-3
4.1.2.8 Remove Wait (TMREMW) 4-3
4.1.2.9 Start User Task (TMSTART) 4-4
4.1.3 Task States 4-4
4.2 EVENT SERVICE HANDLER 4-5
4.2.1 Event Coordination Table 4-5
4.2.2 System Queue 4-5
4.2.3 Coordination 4-5
4.2.3.1 Connection 4-5
4.2.3.2 Queuing 4-7
4.2.3.3 Assertion 4-7
4.2.4 Event Service Facilities 4-7
4.2.4.1 Connect (EVCON, EVQCON) 4-8
4.2.4.2 Disconnect (EVDIS) 4-9
4.2.4.3 Release (EVREL) 4-9
4.2.4.4 System Queue Service (SQS) 4-9
4.2.4.5 Dispatch From EVT (EVTDISP) 4-9
4.2.4.6 Return From Event (EVRTE) 4-10
4.2.4.7 Propagation (EVPROP) 4-11
4.2.5 Dispatch Priority 4-11
4.3 SVC HANDLER 4-12
4.3.1 First Level Interrupt Handler (FLIH) . . . 4-12
4.3.2 SVC 1 Executor (SVC1) 4-12
4.3.3 SVC 1 Termination (IODONE) 4-13
4.3.4 SVC 2 Executors (SVC2 and SVC2.xx) 4-14
4.3.5 SVC 3 Executor (SVC3) 4-14

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

	4.3.6	SVC 5 Executor (SVC5)	4-15
	4.3.7	ADCHK	4-15
	4.4	SYSTEM JOURNAL	4-16
	4.5	EXECUTIVE MESSAGES	4-16
	4.6	CRASH HANDLER	4-17
	4.7	INTERNAL INTERRUPT HANDLERS	4-17
	4.7.1	Machine Malfunction Handler (MMH)	4-17
	4.7.2	Illegal Instruction Handler (IIH)	4-18
	4.7.3	Memory Fault Handler (MFH)	4-19
	4.7.4	Arithmetic Fault Handler (AFH)	4-19
	4.8	SYSTEM INITIALIZATION	4-19
CHAPTER 5		THE COMMAND PROCESSOR	5-1
	5.1	INTRODUCTION	5-1
	5.2	COMMAND PROCESSOR INITIALIZATION (COMMAND)	5-1
	5.3	COMMAND INPUT/PARSING (COMMANDR)	5-1
	5.3.1	Command Prompts	5-1
	5.3.2	Command Parsing	5-2
	5.4	COMMAND ERROR HANDLING (CMDERROR)	5-2
	5.5	COMMANDS	5-3
	5.5.1	Task Related Commands	5-3
	5.5.2	Device/File Commands	5-3
	5.5.3	General Commands	5-5
	5.6	COMMAND SUBSTITUTION SYSTEM (CSS)	5-6
	5.6.1	Calling CSS (CSSTEST)	5-6
	5.6.2	Preprocessor/Expansion (PREPRO)	5-6
	5.6.3	Additional Commands	5-7
	5.6.4	Building CSS Files (BUILD, $BUILD)	5-8
	5.7	LOADER	5-8
	5.7.1	Common Loader (LOAD)	5-9
	5.7.2	Fullword Loader (LOADFULL)	5-9
	5.7.3	Halfword Loader (LOADHALF)	5-9
	5.7.4	Overlay Loads (LOADOVLY)	5-9
	5.7.5	Load Errors (LOADFAIL)	5-9
	5.8	CONSOLE HANDLING	5-9
	5.9	THE BREAK KEY	5-10
CHAPTER 6		FILE MANAGEMENT SYSTEM	6-1
	6.1	FILE HANDLER	6-1
	6.2	VOLUME ORGANIZATION AND INITIALIZATION	6-1
	6.3	DIRECTORY MANAGEMENT	6-2
	6.3.1	Directory Entry Creation And Deletion (ALLOD, RELED)	6-4
	6.3.2	Directory Access (DIRLOOK, GETD, PUTD)	6-4
	6.4	BIT MAP MANAGEMENT	6-4
	6.4.1	File Allocation and Deletion (GETSECTR, RELEB, GETB, PUTB)	6-4
	6.5	SVC 7 SECOND LEVEL INTERRUPT HANDLER (SVC7)	6-5
	6.6	SVC 7 FUNCTION EXECUTORS	6-5
	6.6.1	Allocate (ALLO)	6-5
	6.6.2	Assign (OPEN, OPEN.DEV, OPEN.CO, OPEN.CH)	6-5
	6.6.3	Change Access Privileges (CAP)	6-6

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

6.6.4	Rename (RENAME)	6-7
6.6.5	Reprotect (REPRO)	6-7
6.6.6	Close (CLOSE)	6-7
6.6.7	Delete (DELETE)	6-8
6.6.8	Checkpoint (CHECKPT)	6-8
6.6.9	Fetch Attributes (FETCH)	6-8
6.6.10	SVC 7 Integrity Checking Subroutines	6-9
6.7	SVC 1 INTERCEPT ROUTINES	6-9
6.7.1	Contiguous File Handler	6-9
6.7.1.1	Data Transfer Requests For Contiguous Files (CONTIG)	6-9
6.7.1.2	Command Requests For Contiguous Files (CMD.CO)	6-10
6.7.2	Chain File Handler	6-11
6.7.2.1	Chain File Handler Subroutines	6-11
6.7.2.2	Data Transfer Requests for Chain Files (CHAIN)	6-12
6.7.2.3	Command Requests for Chain Files (CMD.CH)	6-12
6.7.2.4	Error Recovery For Chain Files	6-13
CHAPTER 7	DRIVER DESCRIPTION	7-1
7.1	DRIVERS	7-1
7.2	DRIVER CONTROL BLOCKS	7-2
7.2.1	Device Control Block (DCB)	7-2
7.2.2	Channel Control Block (CCB)	7-6
7.3	DRIVER INITIALIZATION ROUTINE (DIR)	7-8
7.4	INTERRUPT SERVICE ROUTINES (ISR)	7-9
7.5	EVENT SERVICE ROUTINES (ESR)	7-10
7.6	HALT I/O ROUTINE (TIMEOUT)	7-11
CHAPTER 8	SYSTEM FLOW EXAMPLES	8-1
8.1	SYSTEM STARTUP	8-1
8.2	I/O REQUEST	8-1
8.3	LOG MESSAGE REQUEST	8-4
8.4	FETCH OVERLAY REQUEST	8-6
8.5	CHAIN FILE ACCESS	8-7
CHAPTER 9	EXECUTIVE TASKS AND SYSTEM EXTENSIONS	9-1
9.1	INTRODUCTION	9-1
9.2	EXECUTIVE TASKS	9-1
9.3	SYSTEM EXTENSIONS	9-1
9.4	PATCHING	9-2
CHAPTER 10	JOURNAL AND CRASH CODES	10-1
10.1	CRASH CODES	10-1
10.2	JOURNAL CODES	10-2
CHAPTER 11	DATA STRUCTURES	11-1
11.1	INTRODUCTION	11-1
11.2	CHANNEL CONTROL BLOCK (CCB)	11-1
11.3	DEVICE CONTROL BLOCK (DCB)	11-3
11.4	DIRECTORY ENTRY (DIR)	11-5

	11.5	DEVICE MNEMONIC TABLE (DMT)	11-6
	11.6	EVT LEAF (EVL)	11-6
	11.7	EVT NODE (EVN)	11-7
	11.8	FILE CONTROL BLOCK (FCB)	11-8
	11.9	INITIAL VALUE TABLE (IVT)	11-12
	11.10	SYSTEM POINTER TABLE (SPT)	11-13
	11.11	TASK CONTROL BLOCK (TCB)	11-15
	11.12	VOLUME MNEMONIC TABLE (VMT)	11-17
	11.13	VOLUME DESCRIPTOR (VD)	11-17
	11.14	SYSTEM DATA STRUCTURE RELATIONSHIPS . . .	11-18
CHAPTER 12		MODULE DEFINITIONS	12-1
	12.1	INTRODUCTION	12-1
	12.2	EXECUTIVE MODULES	12-2
	12.3	COMMAND PROCESSOR MODULES	12-4
	12.4	FILE MANAGER MODULES	12-6
	12.5	FLOATING POINT TRAPS	12-8

ILLUSTRATIONS

FIGURE	2-1	OS/32 FUNCTIONAL BLOCK DIAGRAM	2-2
	2-2	OS/32 ST MEMORY MAP	2-4
	2-3	EXAMPLE OF EVT STRUCTURE	2-6
	2-4	ELEMENTS OF I/O SYSTEM	2-8
FIGURE	3-1	SYSTEM STATES	3-3
FIGURE	4-1	TASK CONTROL	4-1
	4-2	PORTION OF EVT	4-6
	4-3	CONNECTION AND CONNECTION WAIT	4-8
FIGURE	6-1	VOLUME DESCRIPTOR	6-2
	6-2	DIRECTORY EXAMPLE	6-3
FIGURE	7-1	DCB ATTRIBUTE BIT DEFINITION	7-3
	7-2	DCB FLAG DEFINITION	7-5
	7-3	CCW BIT DEFINITIONS	7-7
FIGURE	8-1	SYSTEM START UP	8-2
	8-2	SVC 1 (I/O AND WAIT)	8-3
	8-3	LOG MESSAGE REQUEST	8-5
	8-4	FETCH OVERLAY REQUEST	8-7
	8-5	READ REQUEST TO CHAIN FILE	8-9
FIGURE	11-1	CHANNEL CONTROL BLOCK (CCB)	11-1
	11-2	DEVICE CONTROL BLOCK	11-3
	11-3	DIRECTORY ENTRY (DIR)	11-5
	11-4	DEVICE MNEMONIC TABLE (DMT)	11-6
	11-5	EVT LEAF (EVL)	11-6
	11-6	EVT NODE (EVN	11-7
	11-7	FILE CONTROL BLOCK (FCB)	11-8
	11-8	INITIAL VALUE TABLE (IVT)	11-12
	11-9	SYSTEM POINTER TABLE (SPT)	11-13
	11-10	TASK CONTROL BLOCK (TCB)	11-15
	11-11	VOLUME MNEMONIC TABLE (VMT)	11-17
	11-12	VOLUME DESCRIPTOR (VD)	11-17
	11-13	SYSTEM DATA STRUCTURE RELATIONSHIPS	11-18

CHAPTER 1

INTRODUCTION

This Program Logic Manual (PLM) is a guide to the internal structure of the operating system OS/32 ST. It is intended for use by people involved in maintaining and modifying the system. It normally should be used with program listings.

Use of this manual requires the reader to be knowledgeable of the features, functions and conventions of OS/32 ST from the user's point of view as documented in Program Reference Manual, Publication Number 29-380, and Program Configuration Manual, Publication Number 29-379. Documentation for I/O drivers is in OS/32 Series General Purpose Driver Manual, Publication Number 29-384.

OS/32 ST is an operating system that provides program management for single task programs. System control via console operator, interrupt handling and I/O servicing are built-in functions of OS/32 ST. Disc file management features are also provided when the system is equipped with a disc, and as such, OS/32 ST is oriented towards a disc operating system environment. A file directory and allocation bit map are maintained on each disc volume to allow for disc portability.

OS/32 ST is compatible on the program level with the real-time multi-task operating system OS/32 MT in most respects not related to multi-programming, and can serve as a development tool and "test bed" for many OS/32 MT-oriented programs. Moreover, many system routines, I/O drivers in particular, are identical in both systems; many other routines are quite similar in structure.

Chapter 2 of this manual describes the general structure of OS/32 ST. Chapter 3 discusses the conventions followed by the system in terms of interfacing between modules, naming of fields and flags and structure of modules. Chapters 4, 5 and 6 contain a detailed technical description of the major module groupings in OS/32 ST. These chapters are designed to provide a technical overview to the detailed module descriptions in Chapter 12. Chapter 7 contains the Driver Description. Chapter 8 contains examples of system flow of control. Chapter 9 contains an explanation of Executive tasks and discusses user added extensions to OS/32 ST. Chapter 10 contains a list of CRASH and JOURNAL Codes and their meanings. Chapter 11 contains the format of system control blocks. Chapter 12 contains detailed module descriptions for each module in OS/32 ST. These module descriptions are intended to be used together with the corresponding flow charts in Volume 2 of the OS/32 ST Program Logic Manual, Publication Number 29-382.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

CHAPTER 2

SYSTEM STRUCTURE

2.1 INTRODUCTION

This chapter describes, in a broad fashion, the general structure of OS/32 ST from a technical viewpoint. As illustrated in Figure 2-1, OS/32 ST is composed of four major module groupings. These are Executive, Command Processor, File Manager and the OS/32 Series General Purpose drivers. This chapter discusses each of these module groupings and how they interact. I/O support is provided by the OS/32 Series General Purpose drivers together with major portions of the Executive, so the drivers are discussed in the context of this I/O subsystem.

2.2 EXECUTIVE

The executive contains logic for processing Supervisor Calls (SVCs) and other internal interrupts, a memory manager, task manager, Event Service handler, a crash handler, general utility function routines and a system journal handler. Portions of the Event Service handler and SVC processor support the I/O subsystem and are discussed in section 2.2.

2.2.1 Task Management

All functions in OS/32 ST are performed on behalf of a task. A task is controlled by OS/32 ST through a Task Control Block (TCB). In OS/32 ST there are two TCB's, one for the system task (Command Processor), and one for the user task. A task may be in one of the following states: current, ready or wait. A task is in the Wait state when some external event must take place before the task can proceed. A task is in the ready state when all external events have occurred that are necessary to let the task proceed. A task is in the current state when it is the highest priority ready task. In OS/32 ST, the system task has higher priority than the user task.

2.2.2 Executive Services

SVC handlers

All SVC interrupts cause entry to the SVC First Level Interrupt Handler. This module performs common preprocessing for SVC 1,2,3, 5, and 7, such as making a Journal entry for the particular SVC, checking the parameter block address for validity, saving the

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

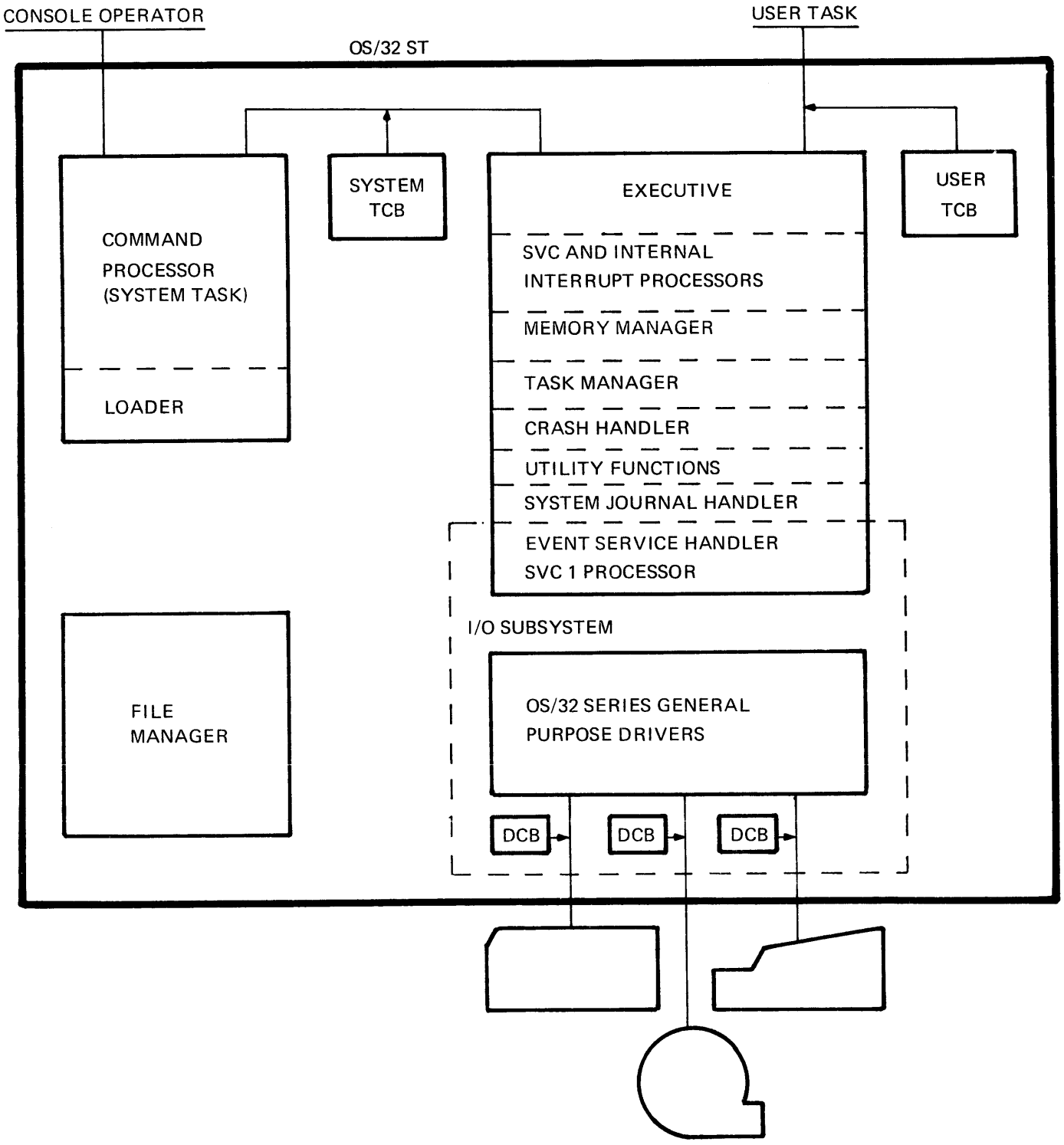


Figure 2-1. OS/32 ST FUNCTIONAL BLOCK DIAGRAM

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

user registers, if necessary, and branching to the executor for the particular SVC. SVC 2 Second Level Interrupt Handler performs similar common pre-processing for the different codes of SVC 2.

Memory Manager

This routine keeps track of UTOP, CTOP, UBOT, MTOP and FBOT in the System Pointer Table (see Figure 2-2). It processes GET/RELEASE storage and EXPAND/CONTRACT Allocation SVCs. It is called by the file manager to obtain space for File Control Blocks (FCB). It also contains a routine to check the validity of addresses passed by the user task to the system via SVCs. (Any such address must be between UBOT and CTOP+2). The way in which memory is allocated in OS/32 ST is as follows (refer to Figure 2-2):

When a task is loaded, space is allocated upwards from the value of UBOT. The new program top address is stored at UTOP. CTOP is then set to UTOP + a SYSGENd number of 256 byte blocks -2. (CTOP is the last halfword of memory in the program's allocation).

Space for new FCB's is allocated from MTOP down. (MTOP is the last physical byte address in memory). FBOT points to the first byte in the FCB area. If allocation of a new FCB would cause FBOT to be less than CTOP + 2, the allocation request is rejected, causing the file manager to return a Buffer Error status.

EXPAND/CONTRACT Allocation calls cause CTOP to be incremented/ decremented by the specified number of 256 byte blocks. If the call would cause CTOP + 2 to overlap UTOP or FBOT, the call is rejected.

GET/RELEASE Storage calls cause UTOP to be incremented/ decremented by the specified number of bytes. If the call would cause UTOP to overlap UBOT or CTOP + 2, the call is rejected.

General Utility Functions

This package is primarily used for processing the miscellaneous SVC 2 routines, but it is also entered directly by the other system elements from time to time. Some of the features

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

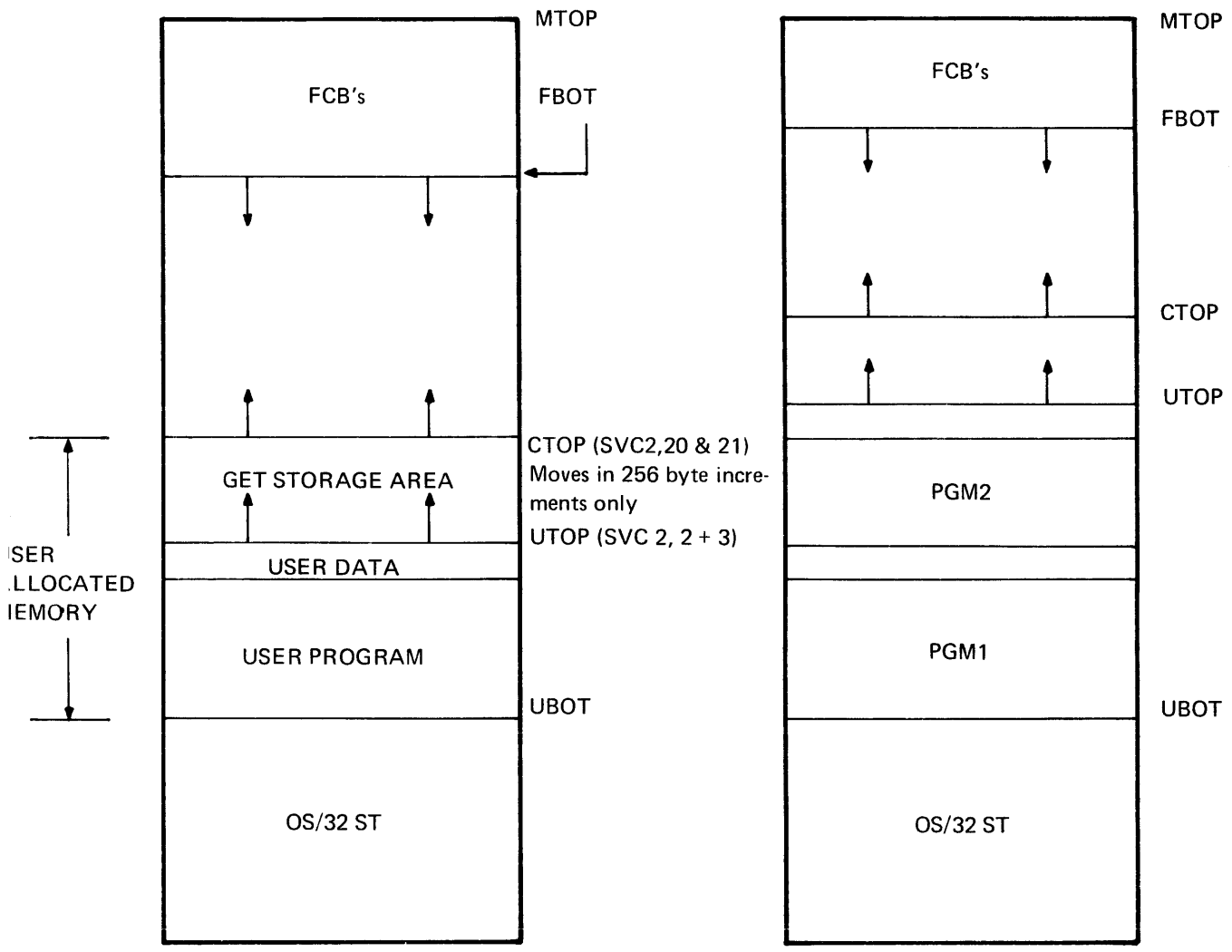


Figure 2-2. OS/32 ST Memory Map

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

provided are:

UNPACK routine
EXECUTIVE MESSAGE routine

Any subroutine called by more than one system module is normally placed in this package.

2.2.3 Internal Interrupt Handlers

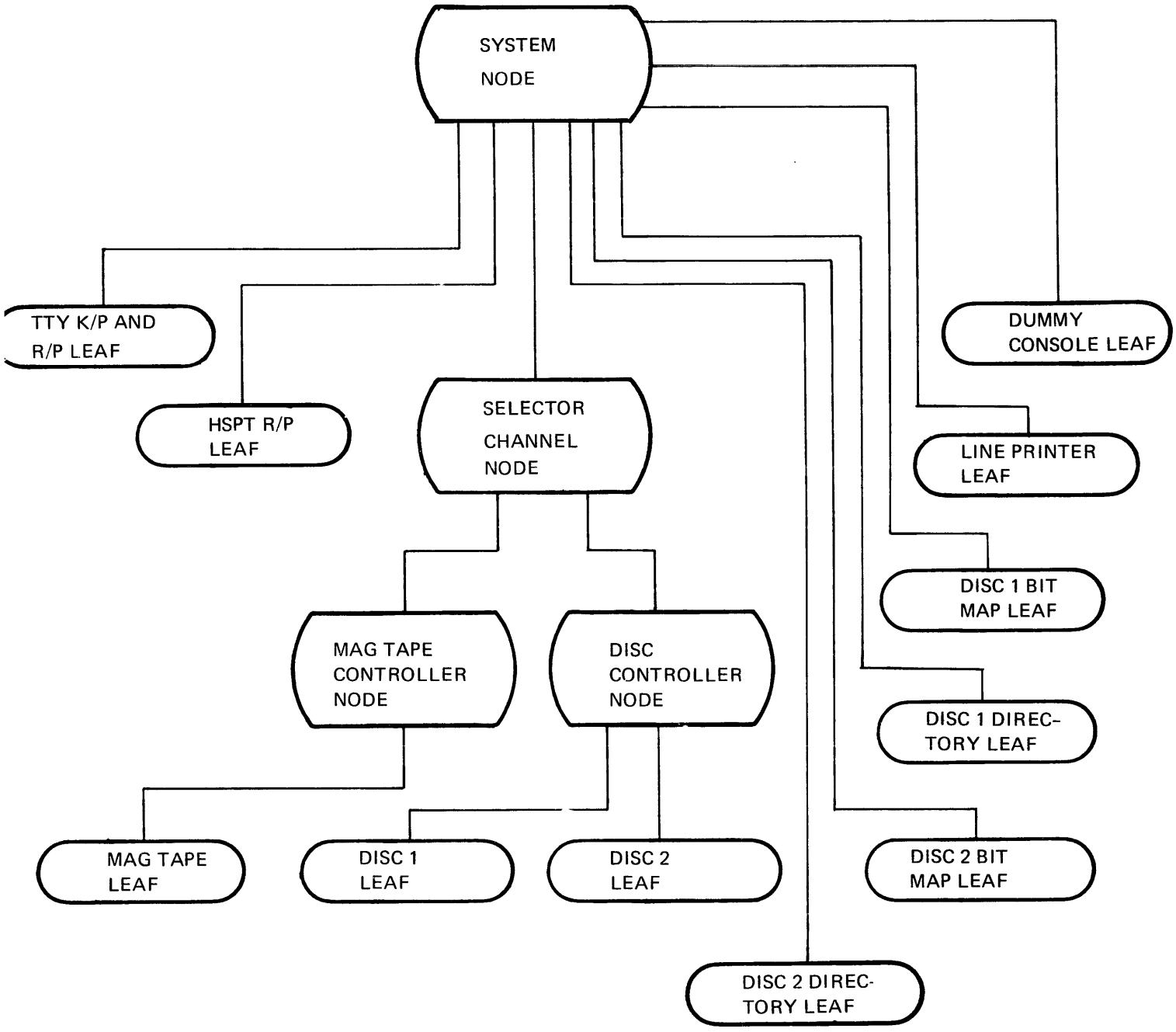
This package handles interrupts due to machine malfunction, illegal instruction, and arithmetic fault. Illegal instructions, arithmetic faults or parity errors encountered within the executive cause an immediate entry to the Crash handler. Arithmetic faults encountered while a task is running cause a message to be logged to the system console and the task to be PAUSEd or continued as specified by the user. Illegal instructions encountered while the task is running may enter the SYSGENable floating-point trap package to be executed. Otherwise, the task is PAUSEd via an entry into the task manager, after logging a message to the system console. Memory parity machine malfunctions within a task cause the task to be aborted after logging a message to the system console. Power failure causes all registers to be saved, whereupon the system waits for power restoration. When power is restored, the following actions take place:

All direct-access data transfers are retried.
All other I/O operations are terminated.
A message is logged to the system console, requesting the operator to reset peripherals if necessary. The system waits for the operator to enter 'GO'.
The active task (if there was one) is PAUSEd.

2.2.4 Event Service Handler

Coordination of system resources (mainly I/O devices) is controlled through the Event Coordination Table (EVT). The Event Service Handler contains routines to manage the EVT. The EVT is a tree structure consisting of nodes (entries with descendents) and leaves (entries without descendents). (Figure 2-3 illustrates an example of an EVT structure). Each path in the tree corresponds to a group of system resources that must be coordinated as one resource. For example, the system node, selector channel node and mag tape leaf path corresponds to all the resources that must be coordinated to control access to the magnetic tape. Coordination is implemented by providing routines to connect to, queue to, disconnect from, and release entries in the EVT. Only one task may be connected to an EVT entry at a time. The EVT is generated at SYSGEN time by the OS/32 ST Configuration Utility Program, 03-076. A task is not connected to any required entry until it can be connected to all required entries, thus preventing deadlock conditions.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.



Peripherals: ASR TTY
High Speed Paper Tape Reader/Punch
Line Printer
Selector Channel
Mag Tape
Disc Controller
Disc 1
Disc 2

Figure 2-3. Example of Evt Structure

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

2.2.5 Crash Handler

This routine is entered when the system cannot continue without the risk of destroying system or user information. A CRASH CODE is displayed on the Display Panel and is also stored in the SYSTEM POINTER TABLE (SPT) at SPT.CRSH. (See Chapter 10 for crash codes and meanings). System Initialization does not reset SPT.CRSH. Some of the conditions which cause the Crash handler to be entered are:

Illegal Instruction within the system.
Invalid Item on the System queue.
Invalid TCB ID passed to Task Management.
Interrupt from undefined device.

2.2.6 System Journal

The system journal is a circular list of historical data maintained by the system. Each entry on the journal consists of five fullwords of information: the task id of the task which was active at the time of the entry, the reason for making the entry (Journal Code) and information pertinent to that call. The system journal is established at SYSGEN time by OS/32 ST Configuration Utility Program 03-076. System Journal processing may be eliminated at SYSGEN time. See Chapter 10 for a list of the Journal Codes and their meanings.

2.3 I/O SYSTEM

The I/O system consists of system routines and control blocks necessary to provide a device independent facility for performing I/O requests. It is composed of the SVC 1 executor, IODONE, device drivers, Device/Volume Mnemonic Tables (DMT/VMT), Device Control Blocks (DCB), Channel Control Blocks (CCB), Interrupt Service Point Table (ISPTAB), Logical Unit Table (LTAB), the Event Coordination Table (EVT), and the Event Service Handler (see Figure 2-4).

2.3.1 Device/Volume Mnemonic Tables

All devices and direct-access volumes are referred to throughout the system either by logical unit or by an ASCII identifier. These tables, DMT and VMT, bind these ASCII identifiers to the devices' DCB's.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

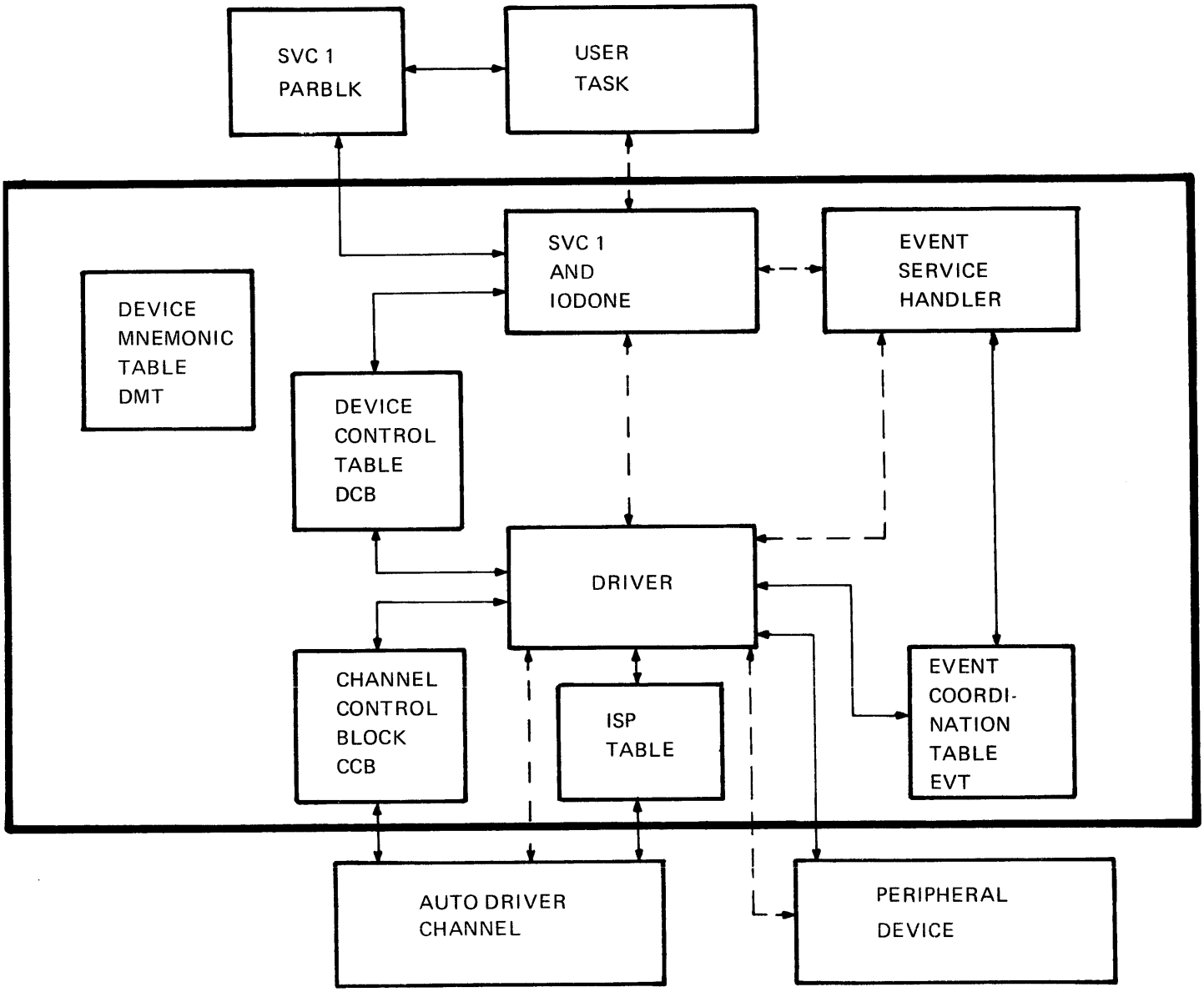


Figure 2-4. Elements of I/O System

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

2.3.2 Logical Unit Table

This table is physically present in the TCB for both the system and user tasks; however, it is of interest to the I/O subsystem and the file manager. It consists of a table of DCB or FCB addresses, one for each logical unit. If the logical unit is not assigned to any device, the entry is set to zero. The size of the user Logical Unit table is fixed at SYSGEN time. Access privileges are placed in a Logical Unit entry at ASSIGN time.

2.3.3 Device Control Block (DCB)

A DCB is provided for each device in the system. This control block contains device-dependent information such as the attributes of the device, flags and register save areas if needed. Pointers are provided to the driver initialization, interrupt service, and termination phases, as well as the event service leaf which coordinates access to this device (see below).

2.3.4 Channel Control Block (CCB) and Interrupt Service Pointer Table (ISPTAB)

CCB's and the ISP table are used to control I/O requests through the Auto Driver Channel capability of the 32-bit series processor.

2.3.5 SVC 1 Processor

The SVC 1 Processor saves the user's registers, picks up the user's parameter block for the driver, and then makes several error checks. These are done primarily through the mechanism of checking the attributes bytes in the device control block against the function code specified in the call. If the call is in order, the system enters a reentrant state, places the data from the parameter block into the DCB and vectors to the appropriate driver.

2.3.6 Drivers

These are the same as the OS/32 MT drivers and are consequently fully reentrant, with the exception of the interrupt-handling phase. The initiation phase of an OS/32 driver runs as a reentrant subroutine of the task, i.e., using the user registers and with queue service enabled. The interrupt-handling phase runs with all interrupts inhibited, except for illegal instruction and machine malfunction. The termination phase of the driver runs in a reentrant state but as though it were an interrupt-handling routine of the task.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

2.3.7 Termination Event Coordination Table

The OS/32 Termination Event Coordination Table (EVT) is used to coordinate access to all devices, controllers, selector channels and bus switches in the system as well as other system resources that must have controlled access, such as bulk storage directories and allocation bit-maps. This table contains busy flags for all devices and pointers for each device that requires coordination, to the controller, channels and bus switches with which that device must be coordinated. See Section 2.1.4.

2.4 COMMAND PROCESSOR

The Command Processor provides the operator interface to OS/32 ST. It executes as one of the two tasks in OS/32 ST and is designed so that many functions are performed through Supervisor Calls. The Command Processor contains routines to support the Command Substitution System (CSS), the resident loader and routines to support Direct Access devices. The Command Processor controls all I/O requests to the Console and Log devices.

2.4.1 Command Processing

The Command Processor accepts commands from the system console device, decodes them and calls the appropriate executor. Some commands are executed via Supervisor calls (e.g., EXPAND, ASSIGN) while others are executed by the Command Processor routines (e.g., MARK, DISPLAY). The Command Processor contains logic to provide the console operator with informative messages in case of error.

2.4.2 Command Substitution System (CSS)

The Command Substitution System routines provide the ability to build, execute and control files of OS/32 ST operator commands. CSS consists of routines to execute CSS operator commands, to manage the CSS buffers and to provide the command parameter substitution facility. The CSS buffers are established at SYSGEN time by OS/32 ST Configuration Utility Program, 03-076.

2.4.3 Loader

The OS/32 ST resident loader loads tasks and overlays. The input medium must be in one of two loader formats: CAL 32-bit object output, for fullword mode tasks, and CAL 16-bit or OS Assembler or Fortran IV object format (M16/M17) for halfword

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

mode tasks. Since the two modes have disparate formats, in essence two loaders are provided in a full OS/32 ST system; the halfword mode loader and the fullword mode loader. The halfword mode loader may be SYSGENed out. The resident loader supports the OS/32 ST SVC 5 capability for loading overlays.

2.4.4 Direct Access Support

The Command Processor provides the operator with the command functions necessary to initialize a disc pack and name it, allocate and delete files, display files, save an OS image suitable for boot loading, perform functions such as Rewind, backspace record to a file assigned to the user task and for mounting and dismounting direct access volumes. Most of these functions are executed via SVC 1 and SVC 7 calls.

2.4.5 Console Support

The Command Processor provides the user task access to the keyboard/printer device used as the system console. This is accomplished via a dummy driver which intercepts all log messages and SVC 1 requests to the console device and executes them for the user task. Because of this feature and the structure of Task Management, most commands can be entered and executed while a user task is active, even if the task has assigned the console device.

2.5 FILE MANAGEMENT

The file management routines handle all access to bulk storage files, either by the user task or by the system. There are five basic modules in this package: the directory and bit-map handler, the Contiguous file access method, the Chained file access method, the SVC 7 processor, and the file/volume utility module.

2.5.1 SVC 7 Processor

This package processes all SVC 7 calls. It calls on the directory and bit-map handler when a file is assigned, allocated, deleted, or check-pointed. When a file is closed, it calls the disc driver as required to make sure all valid data is written on the disc. Protection keys are checked by this module, which also performs all assignment of devices to logical units.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

2.5.2 Directory and Bit-Map Handler

This package handles all access to and modifications of the directory and bit-map for each bulk storage device. Entries are provided to look up a file in the directory, to enter a new file name in the directory, to modify or delete a directory entry, to allocate one or more contiguous sectors of storage, or to release allocated bulk storage.

2.5.3 Contiguous File Access Method

This package is entered on an SVC 1 to a Contiguous file. It performs sector address computations and enters the disc driver.

2.5.4 Chained File Access Method

This package is entered on an SVC 1 to a Chained file. It handles all buffering and unbuffering, calls the disc driver for read or write whenever a buffer is filled, and allocates new space on bulk storage as required for file expansion.

2.5.5 File/Volume Utility

This package performs certain miscellaneous and non-SVC 7 functions associated with the maintenance of bulk storage volumes. It contains routines to list a directory, to mount or dismount a volume, and to initialize a new volume, writing on it a clean directory and bit-map. It is capable of marking defective sectors as permanently allocated (i.e., inaccessible) at volume initialization time. This module is called by the Command Processor to perform these functions.

2.6 FLOATING POINT TRAPS

OS/32 ST provides a SYSGEN option to include software support for the floating point instructions of the 32-bit series processors. This support consists of a trap routine which is passed control on every Illegal instruction interrupt. If the illegal instruction is a floating point instruction, it is then executed by the routine; if it is not a floating point instruction, control is passed to the Illegal instruction handler.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

CHAPTER 3

SYSTEM CONVENTIONS

3.1 MACHINE STATES

OS/32 programs, tasks and routines run in one of eight well-defined states. These states are differentiated by a combination of PSW bits and status bits of the active task, if a task is active. Any state not defined below is not permissible. At any given instant in time, the Processor is executing code in one of these states. They are, in increasing order of priority and privilege:

1) User Task (UT)
2) Executive Task (ET)
3) Reentrant System (RS)
4) Reentrant System, Alternate Save Area (RSA)
5) Event Service (ES)
6) Nonreentrant System (NS)
7) Nonreentrant System, User Registers (NSU)
8) Interrupt Service (IS)

The definition of these states in terms of PSW and TCB status bits is shown in Figure 3-1.

3.1.1 User Task State - UT

The UT state is the state in which all user tasks run. The PSW Protect bit is set. Internal and external interrupts are enabled, with the possible exception of the Arithmetic Fault interrupt bit, which is the only interrupt bit that is under user control. This state may only be exited via interrupt or execution of SVC. The User register set is active.

3.1.2 Executive Task State - ET

The ET state is the state in which all executive tasks (E-tasks) run (see Chapter 9). Protect mode is disabled. All interrupts other than Arithmetic Fault are enabled. Arithmetic Fault is under control of the E-task. All SVC's are permitted. The User register set is active. This state should only be exited via interrupt or execution of SVC.

3.1.3 Reentrant System State - RS

The RS state is the state in which reentrant system code is executed on behalf of a task. Machine constraints are the same as for the ET state; however, software constraints are

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

more stringent: No code may be executed which would cause the RS state to be entered (e.g., a TYPE II SVC - see Section 3.2). Note that RS, RSA and ES code is executed on behalf of a task and is scheduled and dispatched as though it were a privileged routine of the task. For this reason, Queue Service interrupts are enabled. The User register set is active; the previous contents of the User registers is assumed to have been stored by the task manager in the RS Save area of the calling task's TCB. This state may be exited in several ways:

Branch to task manager to return to UT/ET state
LPSW or EPSR that enters NS or NSU state
I/O or internal interrupt

3.1.4 Reentrant System State, Alternate Save Area - RSA

This state is identical to RS state except that the previous contents of the user register set have been stored in an alternate save area (other than in the TCB); a pointer to this save area is in TCB.ASV. This allows routines executing in RSA state to issue SVC's. This state may be exited in several ways:

LPSW or EPSR that enters NS or NSU state
Branch to task manager to return to UT or ET state
I/O or internal interrupt
SVC

3.1.5 Event Service State - ES

This state is identical to RS state in all respects except that the ES (Event Service) status bit in the task's TCB is set, disabling the dispatching of an event for that task (see Section 4.2). It is used only in drivers, principally in driver termination code. This state may be exited only via a call to the Return from Event routine in the event handler package.

3.1.6 Nonreentrant System State - NS

The NS state is the state in which the system executes system code which changes critical system information such as EVT,TCB. This code is nonreentrant and Queue Service interrupts are disabled. The Executive register set is active, of which NS code may use registers 8-F. As no new task may be dispatched while the system is in this state, routines that run in this state must necessarily be short and quick to execute. No SVC's may be executed. This state is exited via LPSW, EPSR, or external interrupt.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

3.1.7 Nonreentrant System State User Register Set - NSU

This state is identical to the NS state (see Section 3.1.6) except that the User register set is enabled. It is used when the user registers are stored in a TCB or alternate save area.

3.1.8 Interrupt Service State - IS

The IS state is used only for interrupt service routines within drivers and in the machine malfunction handler. All interrupts are disabled except machine malfunction. The Executive register set is active, of which IS code may use registers 2-7. This state is only exited via LPSWR on register 0 and 1.

	PSW Status Bits						TCB Status		TCB Option
	I 17	MM 18	AF 19	QS 22	P 23	R 24:27	ES 0	RS 1	ET 0
UT	1	1	d	1	1	F	0	0	0
ET	1	1	d	1	0	F	0	0	1
RS/RSA	1	1	1	1	0	F	0	1	d
ES	1	1	1	1	0	F	1	d	d
NSU	1	1	1	0	0	F	d	d	d
NS	1	1	1	0	0	0	d	d	d
IS	0	1	1	0	0	0	d	d	d

0 means bit must be ZERO
1 means bit must be ONE
d means bit may be ZERO or ONE
F means all 4 bits are ONE
I - Immediate Interrupt
MM - Machine Malfunction
AF - Arithmetic Fault
QS - System Queue Service
P - Protect
R - Register Select
ES - Event Service State
RS - Reentrant System State
ET - Executive Task

FIGURE 3-1 System States

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

3.2 SVC DEFINITIONS AND CONVENTIONS

SVC		Function	Type
1		I/O	II
2	code 1	Pause	II
	2	Get Storage	II
	3	Release Storage	I
	4	Set Status	I
	5	Fetch Pointer	II
	6	Unpack	II
	7	Log Message	II
	15	Pack	II
	16	Pack File Descriptor	II
	17	Mnemonic Scan	II
	18	Move Characters	II
	19	Peek	I
	20	Expand Allocation	I
	21	Contract Allocation	I
3		End of Job	II
5		Fetch Overlay	II
7		File Management	II

All SVC interrupts cause the system to enter NS state. Each SVC enters a separate entry point in the First Level Interrupt Handler (FLIH). FLIH decodes the SVC number and passes control to the appropriate executor. There are two types of SVC executors: those that are short and do not require access to the user register set (Type I) and those that are lengthy or require access to the user register set (Type II). Type I SVC's execute in NS state, thus eliminating the overhead of saving the user registers. Type II SVC's execute for some portion in RS or RSA state.

FLIH passes control to an SVC executor with:

1) address of the Task Control Block of the invoking task in register 9.
2) address of the SVC parameter block in register 13.
3) resume PSW in registers 14 and 15.

Entry is in the state (NS or RS) indicated in a table contained in FLIH. On entry to the executor, the parameter block has been checked to insure it is on a fullword boundary and the address is between UBOT and CTOP+2. It is the responsibility of the executor to perform validity checking of any addresses passed in the parameter block.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

3.3 INTERNAL INTERRUPT CONVENTIONS

Internal interrupts cause control to be passed to the individual interrupt handler in NS state. In all cases, a message is output to the system console indicating the nature of the interrupt and the address at which it occurred. In addition to the interrupts generated by the 32-bit architecture, illegal SVC calls and invalid addresses passed in SVC calls are handled by the internal interrupt handler as for illegal instruction.

3.4 SUBROUTINE CONVENTIONS

Two levels of subroutine linkage are defined for the reentrant system states, RS and RSA, and for non-reentrant state, user register set, NSU. One level of subroutine linkage is defined for non-reentrant system state, NS.

3.4.1 RS, RSA and NSU Subroutines

The mainline level is allowed to use the full register set U0-UF. First level subroutines are linked through register 8 and may use Registers U8-UF without save/restore. Second level subroutines are linked through Register 12 and may use Registers UC-UF without save/restore. This is a general definition used as a guideline; individual modules that violate this definition are described in Chapter 12.

3.4.2 NS Subroutines

The mainline is allowed to use Registers E8-EF of the executive register set. The first level subroutine is linked through Register 8 and may use Register E8-EB without save/restore. Subroutines that may be called from either NSU or NS must be written as an NS subroutine.

3.4.3 Calling Sequences

Parameters are passed in registers or in memory. Parameters may be passed in memory immediately following the BAL instruction only if the parameters require halfword alignment. Parameters may be passed in system tables such as SPT, TCB, etc.

3.4.4 Exits

The normal exit from a subroutine should be to the address contained in the link register, or to a specified number of halfwords past the address contained in the link register. Alternate exits must be to locations passed as parameters. Exits to unlabeled addresses are not permitted.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

3.5 GENERAL NAMING CONVENTIONS

3.5.1 Data Structures

All data structures (defined by CAL STRUC statements) in OS/32 are named with three character symbolic names, e.g., TCB, SPT, DMT. All fields within these structures are defined by a name of the form SSS.FFF, where SSS is the structure name, and FFF is the field ID. (See Chapter 11 for structure definitions).

3.5.2 Bits

Certain fields in a data structure contain flag bits to denote information. These flag bits are manipulated with either bit instructions (e.g., TBT, SBT, RBT) or logical immediate instructions (e.g., THI, OHI, NHI). For each flag bit there are two definitions - one for the bit number and one for the mask. These definitions are of the form SFFF.XXB and SFFF.XXM where S is a character which refers to the structure ID, FFF are three characters which refer to the field, XX identifies the function of the flag bit, B denotes a bit number, and M denotes a bit mask. For example, in the TCB there is field TCB.OPT which contains the option bits; Bit 0 = 1 means the task is an EXEC TASK (E-TASK), Bit 0 = 0 means that the task is a USER TASK (U-TASK). The bit number and bit mask definitions of this flag are:

```
TOPT.ETB    EQU    0
TOPT.ETM    EQU    X'8000'
```

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

CHAPTER 4

EXECUTIVE DESCRIPTION

4.1 TASK MANAGEMENT

4.1.1 Task Control

In OS/32 ST there are two tasks: the system task and the user task. A task may be in Wait state or in Ready state. Wait state indicates that some external event must take place before the task may proceed. Ready state indicates that all such necessary events have taken place. The task manager controls tasks through the use of several control blocks (see Figure 4-1).

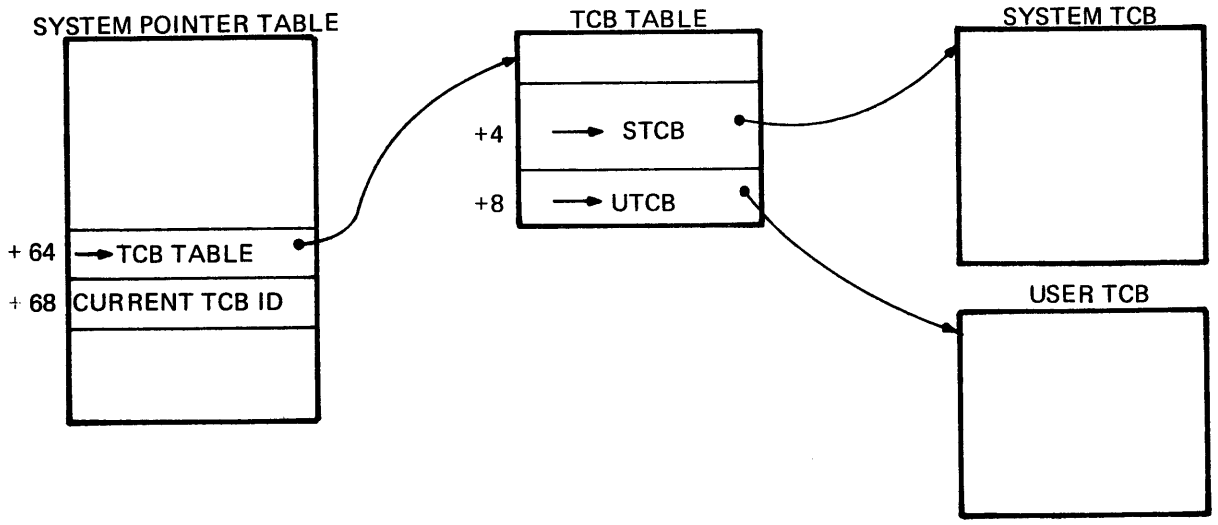


FIGURE 4-1 TASK CONTROL

Each task is described by a Task Control Block (TCB). The addresses of the TCB's are maintained in the TCB table which is pointed to by the System Pointer Table (SPT). A chain, of one or two entries, is maintained of the tasks that are in ready state. This task ready chain is maintained in priority order. In OS/32 ST the system task has priority 1 (highest) and the user task has priority 2. TCBs are referenced by their address or by their id. TCB id is the index, starting at 1, of the TCB in the TCB table.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

4.1.2 Task Management Facilities

The task manager provides subroutines to:

Dispatch the current task
Suspend the current task
Put a task on the ready chain
Remove a task from the ready chain
Cause a task to enter RS, RSA or ES state
Dispatch from top of EVT
Remove a wait condition from a task
Start the user task

In order to maintain the control of a task as it is in various states, the task manager uses three PSW and register save areas in the TCB, the dispatch save area, the RS save area and the ES save area. The state of these save areas is indicated by the ready chain, RS and ES bits in the status field of the TCB.

4.1.2.1 Dispatch Current Task (TMDISP, TMRDISP)

The task manager dispatches the current task by deciding which is higher priority, the task at the top of the ready chain or the task at the top of the EVT queue (see Section 4.2). The top of the ready chain is maintained in the SPT. If this TCB ID is zero, there is no task ready and the task manager places the system in an enabled wait state. If the TCB ID is non-zero, the task manager loads the user register set from the TCB dispatch save area and then passes control to the task by loading the resume PSW in the dispatch save area.

4.1.2.2 Suspend the Current Task (TMSTOP)

This task manager facility is called to prepare the current task for removal as current task. The user register set is saved in the dispatch save area of the TCB and the task's resume PSW is saved in the dispatch PSW save area of the TCB. This facility is used before placing the task in a wait state or when an event has occurred which has made a higher priority task ready.

4.1.2.3 Chain (TMCHN)

When a task has become ready it is placed on the ready chain in priority order. In OS/32 ST the ready chain always takes one of the following forms: empty, system task only, user task only, system task then user task.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

4.1.2.4 Unchain (TMUCHN)

When a condition exists that prevents a task from proceeding until an external event occurs, the task manager removes the TCB from the ready chain. The task may or may not be the current task.

4.1.2.5 Enter System State (TMRSIN, TMRSNIN, TMRSAIN)

All reentrant system routines execute as privileged subroutines of the invoking task. On entry to one of these routines the task manager saves the current PSW and user register values in one of three places: the RS save area in the TCB if the task is entering RS state, the ES save area of the TCB if the task is entering ES state, and the specified alternate save area if the task is entering RSA state. The condition of these various save areas is indicated by the state of the corresponding bits in the TCB status field; the bit on means that the save area contains valid data. These routines must be entered from NS state.

4.1.2.6 Exit From System State (TMRSOUT, TMRSNOUT, TMRSAOUT)

On exit from one of the reentrant system states, RS, RSA or ES, the task manager restores the state of the task to the environment saved in the appropriate save area. The corresponding bit in the status field of the TCB is reset to indicate no valid data in that save area. These routines also check for I/O wait pending or pause pending in the status field of the TCB, and if set, put the task into the corresponding wait state by moving the PSW and registers from the specified save area to the TCB dispatch save area, removing the TCB from the ready chain, and branching to TMDISP.

4.1.2.7 Dispatch from Top of EVT Queue (EVTDISP)

This function of the task manager is discussed more fully in Section 4.2.4.5. In brief, if the task at the top of the EVT queue is higher priority than the task at the top of the ready chain, it is chained and then dispatched as the current task.

4.1.2.8 Remove Wait (TMREMW)

This facility of the task manager removes the specified wait conditions from the specified task by resetting the corresponding bits in the wait field of the TCB. If no wait conditions remain the task is placed on the ready chain.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

4.1.2.9 Start User Task (TMSTART)

The task manager starts the user task by constructing the start PSW from the options field in the TCB and the specified location. This PSW is placed in the dispatch save area of the TCB and the TCB is chained on the ready chain. When the system task enters the Wait state, the user task becomes top of ready chain and is dispatched at the saved PSW with all the user registers set to zero, if the task had been just loaded.

4.1.3 Task States

The state of a task is defined by the settings of the bits in status and wait fields of the TCB and the value of the current TCB field of the SPT. The following is a list of detailed task states and their meanings:

State	Indication	Meaning
Dormant	Dormant bit TCB wait field	No user task has been loaded. (user task only)
Loader Wait	Load wait bit TCB wait field	Loader executing on behalf of task. (user task only)
Ready	Ready chain bit TCB status field	Task on ready chain
Current	TCB ID in SPT current TCB field	Task is the executing task.
RS	RS bit in TCB status field	TCB RS save area contains valid data. Task may be Ready or in wait state.
RSA	AS bit in TCB status field	Save area pointed to by TCB contains valid data. Task may be ready or in wait. RS bit must also be set in TCB status field.
ES	ES bit in TCB status field	TCB ES save area contains valid data. Task must be Ready. I/O wait bit may also be set.
Wait	Ready chain bit reset TCB status field	Task needs external event before it may proceed.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

4.2 EVENT SERVICE HANDLER

4.2.1 Event Coordination Table - EVT

Coordination of system resources is controlled through the Event Coordination Table. The EVT is a tree structure consisting of nodes (entries with descendants) and leaves (entries without descendants). Each path in the tree corresponds to a group of system resources that must be coordinated as one resource. Figure 4-2 illustrates a simple portion of an EVT. All paths in the tree are descendants of the system node.

4.2.2 System Queue

The 32 bit architecture provides for the facility of a system queue. This queue is maintained in the standard list format, and is pointed to by a fixed location (X'80') in low memory. Whenever the status portion of the PSW is updated with the Queue Service bit set, an internal interrupt is generated if there is an entry on this queue. OS/32 ST uses the system queue to schedule events coordinated by the EVT. The entries made to the system queue are always in the form of an address of a leaf in the EVT. When a system queue service interrupt occurs, the leaf is said to have evented.

4.2.3 Coordination

In order to explain the Event Service Handler, the following terms must be defined:

4.2.3.1 Connection

In order to assume control of a system resource reflected in the EVT a task must be connected to the resource. A task is connected to a leaf and ancestor nodes, up to system node, by placing the task ID and priority in the leaf and the task ID, the task priority and the connected leaf address in the upper nodes. An unconnected leaf has a TCB ID of X'00' and a priority of X'FF'; unconnected nodes have, in addition, a connected leaf address of 0. Only one task may be connected to a leaf or node at a time. A task must be connected to all entries between a connected leaf and the highest connected node in the path. Referring to Figure 4-2, a task could be

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

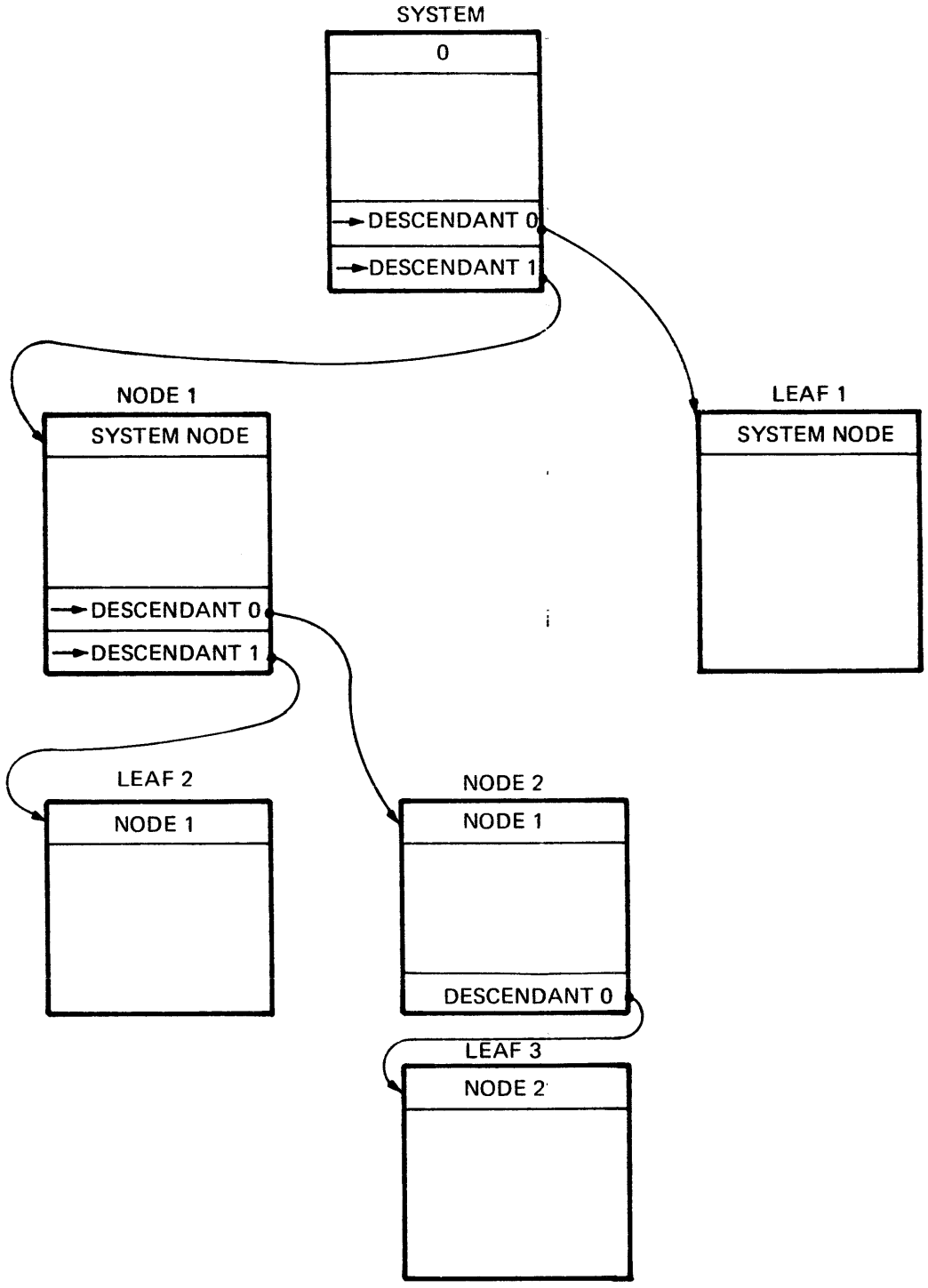


Figure 4-2. Portion of EVT

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

connected to the following EVT entries:

Leaf 1
Node 1, Leaf 2
Node 1, Node 2, Leaf 3
Node 2, Leaf 3
Leaf 2
Leaf 3

A task could not be connected to just Node 1. Tasks never connect to the system node.

4.2.3.2 Queueing

Before connection is made to an EVT entry, all upper nodes must be unblocked (not connected). If an upper node or the leaf being connected to is blocked, the Event Service Handler, upon request, will queue the task for the leaf. This leaf queue is maintained in priority order. While a task is on the leaf's queue, it is placed in a connection wait state. Each unblocked node maintains a pointer to the descendant subtree of the highest priority. Thus the highest priority task on a leaf queue is queued to the highest unblocked upper node.

4.2.3.3 Assertion

Although a task must be initially connected to an entire path in the EVT, it can release upper nodes if they are not necessary for some portions of an operation. When the task again requires these upper nodes, it is said to be asserting reconnection and an assert flag is set in the highest connected entry of this path, a pending flag is set in the leaf and the priority of the highest connected entry is propagated. When the asserting task becomes top of EVT, the dispatcher reconnects required upper nodes before dispatching the Event Service Routine (see Section 4.2.4.5).

4.2.4 Event Service Facilities

The Event Service Handler provides subroutines to:

Connect a task to a path in the EVT
Disconnect a task from EVT entries
Release upper nodes
Service System Queue Service interrupts
Dispatch from top of EVT
Return from event
Propagate a priority up the EVT

All Event Service Handler routines execute in NS state except for return from event.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

4.2.4.1 Connect (EVCON, EVQCON)

Requests for connection always specify a leaf address to be connected to. The connection routines check all upper nodes up to system node. If any upper nodes are connected in some other path, a condition code of X'F' is returned (EVCON) or the task is placed on the leaf's queue and into connection wait (EVQCON). If the path is unblocked, the task is connected to the leaf and the leaf is added to the task's connected leaf chain and a condition code of X'0' is returned.

The connected leaf chain is maintained as a bi-directional list of leafs pointed to by the TCB. The connection wait chain is maintained as a bi-directional list of TCBs pointed to by the leaf. A task can be connected to a leaf and in connection wait for it at the same time. Refer to Figure 4-3 for an example of the user task connected to two leaves and both the system task and the user task in connection wait for the first leaf. This would occur if the user task and the system task issued I/O requests to a device prior to the completion of a previous I/O and proceed request by the user task to the same device.

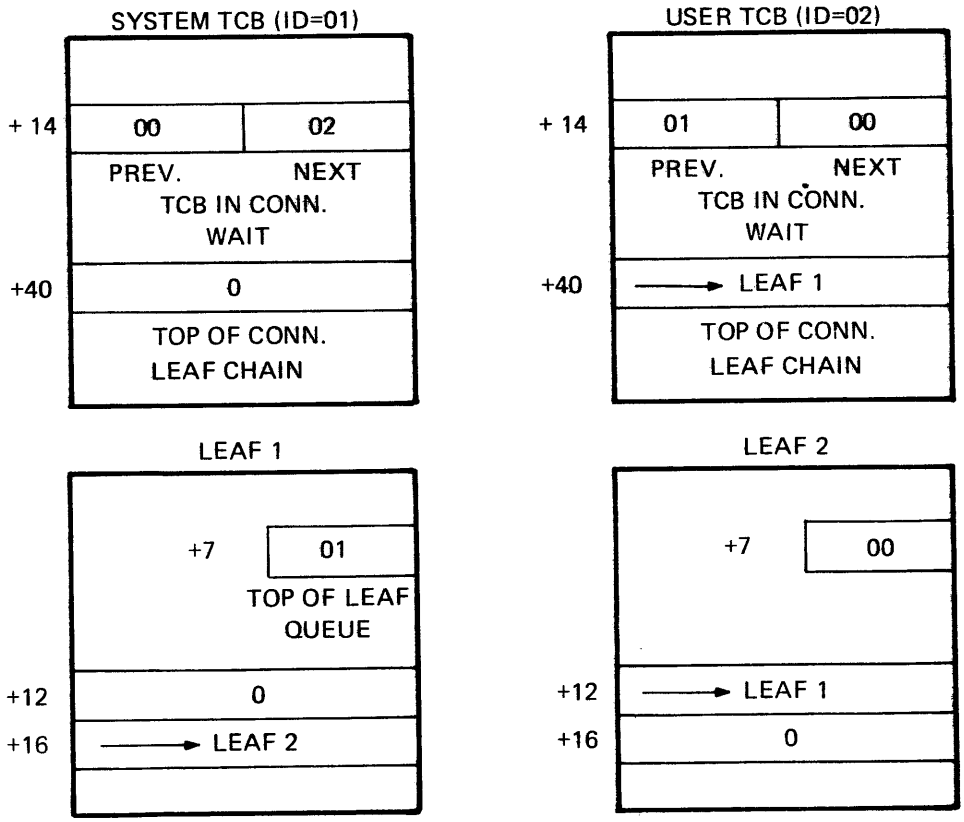


FIGURE 4-3 CONNECTION AND CONNECTION WAIT

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

4.2.4.2 Disconnect (EVDIS)

Requests for disconnection always specify a leaf address. A task is disconnected from the specified leaf and all upper nodes. After each node is unblocked, the priority of the task on top of the queue for that node, if any, is propagated up to the node.

4.2.4.3 Release (EVREL)

Requests for release specify a leaf address and the level above the leaf that is to be released. All nodes from the specified level up are unblocked as in disconnection. For example, the disc driver requires only the disc for a seek operation, so the nodes from level 2 up can be released (the disc controller node and the selector channel node).

4.2.4.4 System Queue Service (SQS)

SQS is entered on a queue service interrupt from the microcode. The leaf address on the bottom of the system queue is removed. SQS obtains the address of the connected TCB from the TCB ID stored in the leaf. The event count in the leaf and in the connected TCB are incremented by one. SQS checks the status flag to see if the task is eventable. If the task is already in ES state it is non-eventable and SQS simply loads the PSW at the time of the interrupt. The non-zero occurrence counts in the leaf and the TCB queue the event to the task. If the task is eventable, SQS sets a pending flag in the leaf and an assertion flag in the highest connected node. The priority of the task is then propagated up the tree from the highest connected node (see Section 4.2.4.7). If the priority was able to propagate up to the top of the EVT (system node), then the current task, if any, is suspended by saving the current PSW and user registers in the dispatch save area of its TCB and the event service routine for the connected task (which may be the current task) is scheduled by EVTDISP (Section 4.2.4.5), unless the current task is higher priority. If the current task is higher priority, it is redispatched and the connected task is dispatched into the event service routine when it becomes top of ready chain.

4.2.4.5 Dispatch From EVT (EVTDISP)

Every routine that causes the current task to be changed branches to the task manager routine TMDISP to determine the next task to be dispatched. TMDISP determines the next current task by comparing the priority of the task at the

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

top of the ready chain, if any, with the priority of the task queued to the system node, if any. If the top of EVT priority is equal to or higher than the priority of the task at the top of ready chain, EVTDISP is entered to dispatch the task queued to the EVT. The task queued to the top of the EVT is queued for one of two reasons: it is in connection wait or it is queued for the dispatching of an event service routine (asserting reconnection).

EVTDISP walks down the EVT by loading the pointer of the highest queued descendant at each level until it reaches a leaf or entry with the assert flag set. If it reaches a leaf, the task at the top of the queue for that leaf is in connection wait. EVTDISP then removes the TCB from the queue, connects the task to all upper nodes, removes the task from connection wait (thus putting it on the ready chain) and branches to the task manager to dispatch the task.

If EVTDISP reached an entry with the assert flag set, it resets the assert flag, resets the pending flag in the connected leaf, connects the task to all upper nodes and branches to the task manager routine TMRSNIN which puts the task in the ready chain, if necessary, saves the current state of the task in the ES save area of the TCB, decrements the event count in the leaf and the TCB by one, and dispatches the task at the event service routine pointed to by the leaf.

4.2.4.6 Return From Event (EVRTE)

All event service routines terminate through EVRTE. EVRTE checks the event count in the TCB in case an event has occurred for the task during the event service routine. If no events are queued, EVRTE simply pas es control to the task manager routine TMRSNOUT to exit from ES state. If the TCB event count is non-zero, EVRTE searches the connected leaf chain for the first leaf with a non-zero event count. If the task is connected to all upper nodes, the task is redispatched in ES state at the event service routine pointed to by the leaf. If it is not connected to all upper nodes, the pending flag is checked. If the leaf's pending flag is set, then the task has been propagated for this leaf and the routine continues to search the connected leaf chain of the task for a leaf with a non-zero event count.

If a leaf with a non-zero event count is found without the pending flag set, pending is set and EVRTE walks up the EVT to the highest connected node, sets the assert flag in that node and propagates the priority up the EVT.

When all leaves in the task's connected leaf chain have been processed, EVRTE passes control to TMRSNOUT to exit from ES state.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

4.2.4.7 Propagation (EVPROP)

Whenever the priority that is queued to an entry in the EVT is changed, that priority is propagated up the EVT to insure that the highest priority task is always queued to the top of the EVT. Since a task is never connected to an entry in the EVT until it is able to be connected to all the entries in the path required, propagation insures that the highest priority task will be connected first at the time the path is free. Requests for propagation specify the address of the starting EVT entry and the priority to be propagated up from that entry.

The priority is propagated by stepping up the EVT one level at a time and comparing the propagating priority to the highest queued priority of that node. If the propagating priority is lower than the priority of the top of the node's queue, EVPROP returns normally. If the propagating priority is higher, EVPROP replaces the node's highest queued priority with the propagating priority and stores the descendant number of the entry just stepped up from in the node's highest queued descendant pointer. This continues until the priority has been propagated up to the system node or a blocked node is encountered. If the priority is propagated up to the system node, EVPROP returns with a top-of-tree indication.

If a blocked node is encountered, EVPROP stops queuing descendants but if the propagating priority is greater than the priority of the task connected to the node, it replaces the connection priority and the propagation process continues.

4.2.5 Dispatch Priority

In OS/32 ST each task has two priorities associated with it: the task priority and the dispatch priority. The ready chain is maintained in dispatch priority order. The system task priority is 01 and the user task priority is 02. In most cases the dispatch priority of the task is equal to the task priority. However, if the user task is connected to an EVT entry that is blocking a path requested by the system task, the user task will have its dispatch priority raised to 01 for the time it is connected to that entry. Although this has no effect on the order of execution in OS/32 ST, it is necessary since the Event Service Handler routines are used in OS/32 MT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

4.3 SVC HANDLER

4.3.1 First Level Interrupt Handler (FLIH)

Each SVC interrupt vectors to a separate entry point in the First Level Interrupt Handler, in NS state. FLIH stores the SVC number in a save area and branches to a common SVC preprocessing routine. FLIH maintains a table which controls the preprocessing for each SVC. All SVCs except for SVC 3, require a parameter block. The address passed in register 13 is checked to insure that it is between UBOT and CTOP+2 and is on a fullword boundary. The end of the parameter block is checked except for SVC 2 which may have different length parameter blocks. FLIH then makes entry in the system journal and checks the table for the entry state required by the requested SVC executor. If the executor requires NS entry, FLIH branches to executor; if the executor requires RS entry, FLIH calls the task management routine TMRSIN to pass control in RS state. On entry to an executor, register 9 contains the TCB address of the invoking task and registers 14 and 15 contain the resume PSW.

4.3.2 SVC 1 Executor (SVC1)

Entry to the SVC 1 executor is in NS state. The executor checks the Logical Unit (LU) specified and if it is valid it loads the address of the DCB (for a device) or FCB (for a file) from the TCB of the invoking task. Since the fields of the DCB or FCB used by SVC 1 are identical, processing is done independent of the device or file being referenced. SVC 1 checks the validity of the request for the device or file by comparing the request against the attributes in the TCB LU table (for data transfer requests) or the attributes field of the DCB/FCB (for command function requests). Requests to the null device are returned with normal status immediately. If any errors are detected in the request, the appropriate status is placed in the parameter block and control is returned to the invoking task.

If the request is for wait only or test I/O complete, SVC 1 processes the request in NS state by loading the address of the leaf in the EVT associated with the device from the DCB. If the connected TCB ID of the leaf is different from the TCB ID of the invoking task, normal status is returned immediately to the task. If the TCB ID is the same, then the task has I/O proceeding on the specified device. If it is a test I/O complete request, SVC 1 sets the condition code to X'F' and returns; if it is a wait only call, SVC 1 checks to see if the incomplete I/O is to the same LU as specified and, if not, returns normal status to the task. If the I/O

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

is to the same LU then SVC 1 places the task into I/O wait for the I/O request by setting the I/O wait bit in the TCB wait field, removing the TCB from the ready chain, storing a function code of X'08' (wait only) in the function code field of the DCB and exiting to the task manager routine TMDISP.

SVC 1 then enters either RS state (requests for device or unbuffered file) or RSA state (request to a buffered file) depending on the state of the buffered access method flag in the flags field of the DCB/FCB. It then processes the information in the parameter block.

For data transfer requests, SVC 1 checks the validity of the start and end addresses and if the request is not for a buffered file, it calls the event service handler routine EVQCON to connect the task to the EVT entries pointed to by the leaf address in the DCB for requests with the unconditional proceed bit reset, or it calls the routine EVCON for requests with unconditional proceed set. If EVCON cannot connect the task to the requested path in the EVT it returns a non-zero condition code to SVC 1 which sets the condition code in the resume PSW in the RS save area of the TCB to X'F' and branches to the task manager routine TMRSOUT to return control to the invoking task. If EVQCON cannot connect the task to the requested path, it puts the task in connection wait. When the task is connected, control returns to the instruction following the EVQCON call in SVC 1. If the leaf address in the DCB is zero, no connection is performed. After processing the required connections, SVC 1 stores the start, end and random address from the parameter block in the DCB and branches to the driver or file manager entry point specified in the DCB/FCB. If the request is for I/O and wait, before branching to the driver, SVC 1 sets the I/O wait pending flag in the status field of the TCB. Upon exit from the driver or file manager initialization routine, the task manager puts the task into I/O wait. For command function requests, SVC 1 performs a call to EVQCON for all device requests. For all requests, SVC 1 then passes control to the command function entry point specified in the DCB/FCB.

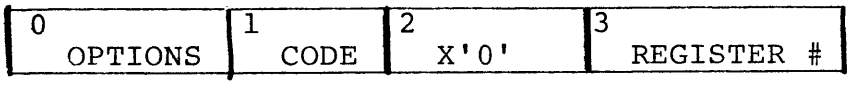
4.3.3 SVC 1 Termination (IODONE)

All drivers and file manager access routines exit to IODONE to complete I/O requests. IODONE is entered in ES state (IODONE) or RS state (IODONE2) with a DCB address and a leaf address. IODONE places the status returned in the DCB into the parameter block if there is one, calls EVDIS to disconnect the task from the specified leaf and its upper nodes, removes the I/O wait condition if this call is an I/O and wait call, and branches to EVRTE if called from termination routine of a driver, or to TMRSOUT if called from an initialization routine at entry IODONE2.

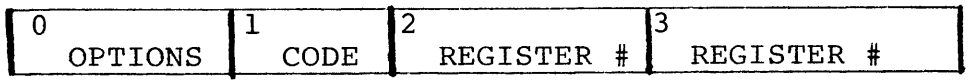
This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

4.3.4 SVC 2 Executors (SVC2 and SVC2.xx)

SVC 2 requests are all vectored to the SVC 2 second level interrupt handler SVC2 for common preprocessing. SVC2 maintains a table of valid SVC 2 codes indicating the type of preprocessing required and the entry state required by the individual executors. SVC2 checks the validity of the code and then performs validity checking on the register specifications passed in the parameter block if necessary. For SVC 2 codes that require parameters to be passed in registers, SVC2 assumes 2 formats of the parameter block:



for parameter blocks requiring one register specification, and:



for parameter blocks requiring two register specifications.

SVC2 then branches to the SVC 2 executor for the specified code for NS entry executors, or branches to the task manager routine TMRSIN to enter RS entry executors.

4.3.5 SVC 3 Executor (SVC3)

SVC3 is entered in NS state from the first level interrupt handler. In NS state, SVC3 stores the specified return code in the System Pointer Table and goes down the connected leaf chain pointed to by the TCB and halts any read requests by calling the routine TIMEOUT (see Section 7.6). It then enters RS state for the rest of processing. SVC3 issues an SVC 7 checkpoint call for each LU in the TCB. This insures that all writes have been normally completed, that the timeout of all reads is also complete and that all files are in a safe condition. It then puts the return code in the end of task message and sets up the TCB and the message function code so that the task is put into I/O wait while the message is printed on the system console and that control is returned to the SVC 3 processor in NSU state on completion of the message. After issuing the end of task message, SVC3 removes the TCB from the ready chain, sets the dormant bit in the TCB wait field and zeroes out the TCB except for the ID, priority, number of LUs and options fields. SVC3 then exits to the task manager to wait for the system task to become ready.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

4.3.6 SVC 5 Executor (SVC5)

Entry to fetch overlay request is in NS state. SVC5 prepares the system task for the overlay request by storing the option specified in the command processor loader request word, SVC5 then picks up the entry in the task's LU table specified by the LU in the SVC 5 parameter block and stores the entry in the LU 1 slot of the system task. This effectively assigns the overlay device or file to the system task temporarily. It then puts the address of the TCB dispatch save area slot for register 12 into the field in the command processor which points to the word that is filled with the status of the load. SVC5 sets the loader wait bit in the wait field of the invoking task's TCB and removes the task from the ready chain.

The task manager routine TMRSIN is called to save the resume PSW and user registers in the RS save area of the TCB and to return control to SVC5 still in NS state. SVC5 stores the address of the parameter block into the register 13 slot of the TCB dispatch save area and stores a PSW with RS status and a location counter of entry RSVC 5 in the dispatch PSW save area. It then branches to a secondary entry point in the dummy driver (see Section 5.7) in IS state to schedule the system task. The system task performs the load of the overlay and upon completion stores the status of the load in the user task register 12 slot in the TCB dispatch save area, removes the loader wait condition from the user task and puts it back on the ready chain.

When the user task becomes top of ready chain, the task manager dispatches it by loading the user register set from the TCB dispatch save area and loading the resume PSW in the dispatch PSW save area. This, in effect, resumes the task at entry point RSVC 5 in SVC5 with the status of the load in register 12. SVC5 stores the status into the parameter block and returns to the user task via TMRSOUT.

SVC 7 is discussed in Chapter 6.

4.3.7 ADCHK

All SVC executors check any addresses passed to insure that the address is between UBOT and CTOP+2. This prevents the user task from overwriting the system routines or file control blocks through the use of an SVC. It is also necessary for OS/32 MT compatibility. This checking is performed by the executive routine ADCHK. Entry to ADCHK is in NS or RS state. If the address checked is not valid, the SVC execution routine branches to MEMFAULT (from NS) or MEMFLTRS (from RS) to process the error (see Section 4.7). In OS/32 MT the address is checked to insure it is within the task program space and also for writability; in OS/32 ST all addresses are writeable.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

4.4 SYSTEM JOURNAL

OS/32 ST provides a facility for recording significant events in the system in a system journal. The journal is a standard circular list with a length specified at Configuration Utility Program time. The address of the journal is kept in the System Pointer Table. Entries to the journal are made from system routines by executing a BAL instruction to the journal routine followed by a halfword journal code. Each entry in the journal consists of five fullwords of information in the following format:

WORD 1	TCB ID X'0' JOURNAL CODE
WORD 2	CONTENTS OF REGISTER 12
WORD 3	CONTENTS OF REGISTER 13
WORD 4	CONTENTS OF REGISTER 14
WORD 5	CONTENTS OF REGISTER 15

where TCB ID is the ID of the current task at the time of the journal call, and the last four words are the contents of registers 12-15 at the time of the journal call. Entry to the journal routine must be in NS state. When the journal list is full, the journal routine resets the slots-used field and reuses the list, thus maintaining the most recent entries. For a complete list of journal codes made by the system, see Chapter 10.

In order to allow the system task and other Executive tasks (see Chapter 9) to make journal entries from other than NS state, an SVC 2 code 0 call is provided. The parameter block for SVC 2 code 0 is:

```
+0     OPT   X'0'     JOURNAL CODE
+4       VALUE 1
+8       VALUE 2
+12      VALUE 3
+16      VALUE 4
```

where values 1-4 are stored in the second through fifth word of the journal entry. The journal code is OR'd with X'8000' before being stored in the journal to identify it as a user code. This SVC is only valid from a task executing in privileged mode (bit 23 of the PSW status reset).

4.5 EXECUTIVE MESSAGES

Since the executive routines cannot issue SVC calls, all messages output by the executive are processed by the system task (command processor). This is accomplished by connecting to the dummy leaf (see Section 5.7), storing the start and end address of the message in the dummy DCB and branching to the dummy driver. All messages are processed in this way by branching to the executive message subroutine EXECMSG.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

4.6 CRASH HANDLER

Throughout OS/32 ST are checks for normally impossible states of the system, such as invalid leaf address on the system queue or illegal instruction interrupt in system code. When such a condition is found the system brings itself to a halt before further destroying the conditions that led up to the impossible situation. This is done by entering the Crash handler.

The Crash handler is entered by issuing a SINT instruction to device number 0 followed by a halfword crash code. The first entry in the ISP table is set by SYSINIT (see Section 4.8) to branch to the Crash handler, CRSEP. CRSEP on entry loads the address of the last entry made to the system journal into register 6 of the executive register set, displays the crash code on the display panel and loads a PSW with only the wait bit and the machine malfunction enable bit set, thus stopping the system in an uninterruptable state. See Chapter 10 for a complete list of crash codes and their meanings.

4.7 INTERNAL INTERRUPT HANDLERS

The internal interrupt handlers process the interrupts generated by the microcode for illegal instruction, arithmetic fault, memory parity error, power fail and power restore. In addition, this package processes illegal SVC calls and invalid addresses passed in SVC calls.

4.7.1 Machine Malfunction Handler (MMH)

On detection of memory parity error, power fail or power restore, the Machine Malfunction Handler (MMH) is entered. Entry is in a state with all interrupts masked off. The condition code is used to determine the type of interrupt and the appropriate routine is entered.

On parity error in the system, the Crash handler is entered. If the parity error is detected while the user task is executing, MMH loads a pointer to the memory parity error message and enters the illegal instruction handler for common interrupt processing.

On power fail detect, MMH tests an internal flag to see if a power restore sequence was in execution at the time of the power fail. If this is so, MMH simply loads an enabled wait PSW to wait for the power restore interrupt. If a power restore sequence was not in execution, the executive and user register sets are saved in an internal save area, the machine malfunction old PSW is loaded from reserved memory and stored in an internal save area. The OS does not use

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

the Power Restore Auto/Restart save area since multiple power fails would destroy the original state of the system. MMH then sets an internal flag to indicate a power restore sequence is in effect and loads an enabled wait PSW to wait for the power restore interrupt.

On power restore detect, the Machine Malfunction Handler enters IS state, clears the Display Panel and loads the address of the TCB table. For each TCB, all active I/O is halted by passing the address of all leaves on the connected leaf chain pointed to by the TCB to the time-out routine (see Section 7.6). The pause pending bit is set in the user TCB if it is not dormant. MMH then issues a message to the system console stating that a power restore has occurred and that all peripherals must be reset. This is necessary primarily due to the fact that on a true power fail/restore sequence the 2.5 and 10 Mbyte disc systems come up in a write protected state, making it impossible to retry any active disc I/O. When the operator has reset all necessary peripherals and typed GO, MMH reloads the register sets from the internal save area, resets the power restore sequence in effect flag and reloads the machine malfunction old PSW from the internal save area.

MMH issues the power restore message from IS state to insure that no interrupts from the timed out I/O can occur before the peripherals have been reset. In order to achieve this, MMH sets up a special CCB, initializes the status field of the DCB to -1, enters the Teletype Keyboard/Printer driver first ISR and sits in a loop testing the status field. When the cleanup ISR is finished, it sets the status field of the DCB to zero. On completion of the I/O, MMH removes from the system queue the address of the console leaf which was put there by the cleanup ISR, in order to prevent the scheduling of a termination routine on exit from the machine malfunction handler. The operator reply is read from the system console in the same manner.

4.7.2 Illegal Instruction Handler (IIH)

Entry to IIH is in NS state. IIH contains common processing for illegal instruction, illegal SVC call, invalid address passed in an SVC call and memory parity error. Each error causes control to be passed to a separate entry point which loads a pointer to the appropriate message and branches to the common interrupt processing.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

Common interrupt processing passes control to the crash handler if the error is detected in system code. If the error occurred during user task execution, the pause pending bit is set in the user TCB and the message is scheduled by branching to the executive message routine. On exit from the executive message routine, the task manager routine TMRSOUT detects the pause pending bit and puts the task into console wait after issuing the task paused message.

An illegal SVC call enters the internal interrupt handler at entry ISH, if the error is detected in NS state, or at entry ISHRS, if the error is detected in RS state. Invalid address passed in an SVC call enters the internal interrupt handler at entry point MEMFAULT, if the error is detected in NS state, or at entry MEMFLTRS, if the error is detected in RS state.

4.7.3 Memory Fault Handler (MFH)

Memory fault interrupts should not occur in OS/32 ST since the Memory Access Controller (MAC) is never enabled. A memory fault indicates a hardware failure so OS/32 ST enters the Crash handler on this interrupt.

4.7.4 Arithmetic Fault Handler (AFH)

Entry to the Arithmetic Fault Handler is in NS state. If the error occurred in the system, the Crash handler is entered. If the error occurred during user task execution, the pause pending bit is set in the status field of the user TCB unless the Arithmetic Fault Continue bit is set in the options field of the TCB. In either case, the interrupt is logged by loading a pointer to the arithmetic fault message and branching to the common interrupt processing in IIH.

4.8 SYSTEM INITIALIZATION

System initialization is performed by the routine SYSINIT. It is entered whenever the system is started at location X'60'. On entry, the status of the PSW is unknown, so the first operation performed by SYSINIT is to put the Processor into an uninterruptable, privileged state. The Display Panel is cleared, the ISP table is set to ignore all interrupts, FBOT is reset to MTOP, all DCBs are reset to initial values, the EVT is refreshed, the TCBs are reset, and SYSINIT puts the system TCB on the top of the ready chain and branches to the Command Processor initialization routine in the ET state.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

Warmstart capability is supported by SYSINIT and the command processor initialization routine. If location X'64' contains a non-zero value, SYSINIT puts the processor into an uninterruptable, privileged state, clears the Display Panel, times out the read outstanding to the console device, if any, puts the system TCB at the top of the ready chain, sets pause pending in the user task if it is active and branches to the Command Processor initialization routine in ET state. This has the effect of restarting the Command Processor as on system initialization but the user task is in a state as if a PAUSE command had been issued. A warmstart should not be attempted if the system crashes in processing a command.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

CHAPTER 5

THE COMMAND PROCESSOR

5.1 INTRODUCTION

The Command Processor is the highest priority task in OS/32 ST. It is an Executive Task (see Chapter 9), and it is the medium through which the operator communicates with the Operating System, and controls the system environment. Whenever possible, the Command Processor tries to perform functions by executing Supervisor Calls. The Command Processor is also responsible for control of the system console, the resident loader, and the Command Substitution System (CSS).

5.2 COMMAND PROCESSOR INITIALIZATION (COMMAND)

After System Initialization has been performed, the Command Processor is entered. The mnemonic of the console device is obtained from the Initial Value Table (IVT) and the Device Mnemonic Table is searched for a matching mnemonic. When found, the device's read and write counts are forced to -1 and it's keys to X'FFFF', thereby making the device unavailable to any other task in the system (see Section 5.7, System Console Device). The OS identifier is now printed. If a "warm start" is in progress, the Command Processor merely prompts the user, however if a "cold start" is taking place, the RESET command is executed.

5.3 COMMAND INPUT/PARSING (COMMANDR)

The Command Processor reads commands from the system console or from the device/file indicated by the current CSS level (see Section 5.5, CSS). A single command, or multiple commands may appear on a given input line. A new command line is not input/requested until all commands on the current line have been executed. The Command Processor executes all commands on a line until it hits an invalid command, and all commands on a line after an invalid command are ignored.

5.3.1 Command Prompts

If input to the Command Processor is from the system console device, and no task is active, an "*" is output to indicate to the operator that the Command Processor is ready to receive another input line. If a task is active, the Command Processor does not prompt with the "*", but still accepts input from the console. While a task is active, if previous command input was from a CSS file, the processing

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

of that file is suspended until the task goes to End-of-Job. During the interval that the task is active, the console responds to command input.

5.3.2 Command Parsing

After the line is read, it is expanded (see Section 5.5, CSS), and logged (see Section 5.4.3, Set Log). This is true unless one of the BUILD commands is in effect, (see Section 5.5.4, BUILD). After the logging, a scan is made of the command line for the first non-blank, non-terminator. Note that the Command Processor uses Register 1 as the pointer to the current character being processed, and that this register is never used for anything else. When the first non-blank, non-terminator is found, it is compared against the Command Mnemonic Table (COMANTAB). If it is not found and there is CSS present in the system, the command is assumed to be a CSS call (see Section 5.5, CSS), or else a MNEMonic error has occurred (see 5.3, ERROR HANDLING).

A check is made to see if any IF statements have set the "skip" flag (see Section 5.5.3, IFs). If so, and if this statement is an IF, then the IF count is incremented, and a new command is searched for. If the statement is an $ENDC the IF count is decremented. If it is a $TERMJOB, it is also executed.

The normal path makes a "user journal entry" (X'8001'), and exits to the executor.

5.4 COMMAND ERROR HANDLING (CMDERROR)

When an error in syntax, or an invalid parameter, or a number of other parsing errors occur, the Command Processor enters the error handler. All entries are made via a BAL on UC, the next 4 bytes after the BAL contain the error mnemonic. An error message is constructed of the form:

```
XXXX-ERR   POS = XX.....
```

The position field attempts to display the last parameter parsed. It may not always be meaningful. The return code is set to 255. If the JOB flag indicates a $JOB is in effect, the JOB "skip" flag is set to indicate all statements until a $TERMJOB are to be skipped. All CSS levels are closed down to the level of the $JOB.

If an I/O error or SVC7 error is encountered while processing a command, additional information on the error is supplied. In the case of an I/O error, or SVC7 error, the error message

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

is:

```
XXXX-ERR   TYPE=XXXX   POS=XX.....
```

where TYPE indicates the error type (DU, NAME, BUFF, PRTY, etc.). If an SVC7 error is an I/O error, then

```
XXXX-ERR   TYPE=IO   TYPE=XXXX   POS=XX.....
```

is displayed, where the second type field indicates the I/O error type.

5.5 COMMANDS

5.5.1 Task Related Commands

The task related commands in OS/32 ST are: START, CANCEL, PAUSE, CONTINUE, OPTIONS, EXPAND, ASSIGN, and CLOSE.

START - obtains the start location, moves in the starting 'options' above UTOP and calls TMSTART.

CANCEL - Closes all CSS levels and calls CANEOJ.

PAUSE - Calls S21PAUSE to put task in Console wait.

CONTINUE - calls TMREMW to remove the console wait and put user task back on ready chain.

OPTIONS - used to set/reset the options bits in the user TCB options field (TCB.OPT).

EXPAND - used to increase the value of CTOP. It merely executes an SVC 2,20.

ASSIGN - used to assign a file/device to a user logical unit. It defaults access privileges to SRW. Initially, it assigns the device/file to Command Processor LU3. If the task is in user state (UT) then the command processor temporarily puts itself in UT state. After a successful assign the command processor LU3 is copied to the appropriate user LU and the Command Processor LU3 is zeroed out.

CLOSE - Copies the user LU to Command Processor LU3, and puts zero in the user's LU entry, then executes an SVC 7 close.

5.5.2 Device/File Commands

The device/file related commands are: ALLOCATE, DELETE, INITIALIZE, MARK, RENAME, and REPROTECT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

ALLOCATE - Builds an SVC7 parameter block from data as defined in the input line. If type is chained, it defaults LRECL to 126 and BKSZ to 1. It then executes an SVC7.

DELETE - Executes an SVC7 with the File Descriptor specified.

MARK - This will mark a device on or off line. If a non-bulk device, the on-line bit is merely set/reset. If the device specified is a bulk device (disc) then:

> MARK ON: Reads the volume descriptor and moves the directory pointer to the DCB, and the volume ID to the VMT.
>
> MARK OFF: Flushes the bit map buffer and directory buffer, and resets the presence bits. Clears the VMT entry name portion.

INITIALIZE - may (1) change the name of a disc pack, (2) save an OS image on a disc pack, (3) check a disc for bad sectors, build a clean descriptor and bit map.
(1) In the case of just changing the pack name, the VD is read, the new name stored, and the VD rewritten.
(2) To save an OS image, if an OS image is currently present, these sectors are released. The size of the OS is computed, and that number of sectors are allocated. The OS is written out, and the address of the image is put into the VD.
(3) To clear a disc pack several operations are performed. First, sector 0 is checked to see if it is good. If not, the pack may not be used. Then an attempt is made to find enough good contiguous sectors in which to put a bit map. These sectors are then cleared to zero, marked as allocated in the appropriate place, and written out. The information as to the location of the bit map is now put in the VD, along with the pack name, and the VD is written out.

The entire pack is now "read-checked". All sectors found to be bad are now marked as allocated. If a bit map sector is found to be bad (remember, bit map sectors have previously been checked), the disc is considered bad.

NOTE

> If both CLEAR and SAVE are present in the command, CLEAR must precede SAVE.

RENAME - Executes an SVC 7 to assign the specified file/device and an SVC 7 to rename it.

REPROTECT - Executes an SVC7 to assign the specified file/device and an SVC7 to reprotect it.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

5.5.3 General Commands

The other commands are: BIAS, EXAMINE, MODIFY, RESET, SET, DISPLAY, and VOLUME.

BIAS - read the BIAS value to be used by the EXAMINE and MODIFY commands and saves it.

EXAMINE - gets the starting location and adds the bias. It then checks for a "/" or ",". If a "/" is found, it gets the ending address and adds the bias. If a "," is found, it gets the number of halfwords to be displayed, and computes an ending address from it. The contents of memory from the starting location through the ending location inclusive is displayed, 8 halfwords per line, to the console/log device.

MODIFY - This command obtains the starting address and adds the bias. Data is obtained from the command line as halfwords, and is stored in successive memory locations.

RESET - Closes all user logical units. If the CLOSE option is not specified, the default assignments are made. CTOP, UTOP are set to initial values.

VOLUME - Sets up SPT.VOL for the default volume name.

SET - (A) SET CODE - put value in SPT.RC
(B) SET LOG - gets the file/device and assigns it to LU2.

If no file/device is specified, LU2 (current log) is closed.

If the COPY option is specified, then the copy flag is set. This command causes all input lines to be logged to the device specified. If the COPY option is specified, all input/ messages will also appear on the system console. If COPY is not specified, system messages will not be logged to the console.

DISPLAY: There are 3 DISPLAYs:

1. DISPLAY LU - This displays a list of all user LUs that are currently assigned, and to what device/file they are assigned.

2. DISPLAY PARAMETERS - This displays various system parameters (number of logical units, task options, CTOP, UTOP, etc.).

3. DISPLAY FILES - This displays the files on the specified volume. It gets the pack ID, finds which drive it is on and assigns the drive SRO. The extent of display is then checked for. In the syntax "-" means all, and hence ABC.- means any files with name ABC and all extensions, or -.- means any file name and any extension.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

The directory is searched for a match and when found the information for the file is displayed.

5.6 COMMAND SUBSTITUTION SYSTEM (CSS)

The Command Substitution System (CSS) is a means for the user to create catalogued but dynamically variable command input streams to perform a predetermined job. CSS consists of the preprocessor and CSS commands.

5.6.1 Calling CSS (CSSTEST)

Whenever a command is parsed, and it is determined that the command is not in the table of standard mnemonics, then it is assumed that a CSS call is being made (true only if CSS is in the system, as is this entire discussion). The mnemonic is treated as a file descriptor, and an attempt to assign the file/device is made. If the file/device does not exist, and the file descriptor did not have an extension, the extension of '.CSS' is appended, and an attempt is again made to assign it. If it does not exist, a MNEMonic error has occurred;

EXAMPLE: ABC is the command, an attempt is made to assign ABC (default system volume) and if it does not exist, ABC.CSS is assigned.

Since the Command Processor uses LUs 0-4 for its executors, CSS files are assigned starting at LU5 (level 1 = LU5, level 2 = LU6, etc.). The pointer to the current buffer is saved (in PTRSTACK) to be used by the preprocessor for parameter substitution. The address of a new buffer (for expansion) is also calculated.

5.6.2 Preprocessor/Expansion (PREPRO)

After each command line is read, it is sent to the preprocessor to be expanded. The preprocessor moves characters from the input buffer to the appropriate expansion buffer. When an "@" is encountered, CSS is alerted that parameter substitution is needed. The number of @'s are counted to determine how many levels back to go, and the parameter number is obtained. The address of the appropriate CSS call is obtained from PTRSTACK, and that call is scanned for the appropriate parameter. The parameter is moved into the expansion buffer, and then the moving of characters from the input buffer resumes.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

EXAMPLE:

```
CSS call - ABC   PAR1,PAR2
FILE ABC - read -              *INPUT LINE @1
         - expands to -        *INPUT LINE PAR1

         - read -              *@1@2
         - expands to -        *PAR1PAR2

         - read -              *XX@1YY@2
         - expands to -        *XXPAR1YYPAR2
```

Whenever a request for substitution is made, and the parameter does not exist, a null is substituted.

```
EXAMPLE:  read           *@@1ABC
          expands to     *ABC
```

@0 is a special parameter. A call for substitution of @0 (or @@0, etc), will cause the file descriptor that called the appropriate CSS level to be substituted. Substitution is made of the file descriptor exactly as it appears in the call (without default system volume, or .CSS extension).

5.6.3 Additional Commands

Several additional commands are supplied to allow the user great flexibility in building CSS files and testing conditions. They are $COPY, $NOCOPY, $CLEAR, $EXIT, $JOB, $TERMJOB, $SKIP, $IFE, $IFNE, $IFG, $IFNG, $IFL, $IFNL, $IFX, $IFNX, $IFNULL, $IFNNULL, $FNDC.

$COPY and $NOCOPY - These commands turn on ($COPY) or off ($NOCOPY) the display of CSS command lines read from a CSS file. They will or will not be listed depending on whether $COPY or $NOCOPY is in effect. These 2 executors merely set/reset a flag (CSSLIST) used by MSGLOG to determine whether to print the line.

$CLEAR - This command terminates all CSS processing, closes all CSS LUs, and returns the input function to the console.

$JOB; $TERMJOB - These are used to delimit a given sequence of input as a unit. If a $JOB is in effect and any command error is detected, then all commands read are skipped until a $TERMJOB is read. $JOB merely saves the level number that the $JOB appears on. $TERMJOB resets the $JOB saved, resets any IFSKIP that is set (see below).

$SKIP - If $JOB is in effect, causes all commands to be skipped until a $TERMJOB. It closes all CSS levels down to the level the $JOB was on.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

$EXIT - This indicates that input from the current CSS level is done and that the current CSS LU should be closed. Input then begins from the next higher CSS level, or the console if there are no higher levels.

$IFE $IFNE, $IFG, $IFNG, $IFL, $IFNL - These commands pick up the value specified in the operand field of the command, and compare the current value of the return code to the value on the line. If the compare satisfies the condition specified, then the next statement is merely read. If the compare does not meet the condition then IFSKIP is set, and no statements are processed until a corresponding $ENDC is found. If IFSKIP is set, reading another $IF increments IFSKIP, each $ENDC read decrements it. When IFSKIP is 0, skipping is done.

$ENDC - Terminator of a $IF (as described above). Parsing one causes IFSKIP to be decremented.

$IFX, $IFNX - These two check to see if the file/device specified by the operand exists. If the condition specified is not met, IFSKIP is incremented. The fd is obtained, and an attempt is made to assign it. The result returned by SVC7 determines whether it exists. Success indicates it exists, but certain errors also indicate the file exists.

$IFNULL, $IFNNULL - These check to see if the parameter specified is null or not null. Since substitution has been performed, the line is scanned for the next non-blank; if it is a terminator, then the parameter was null.

5.6.4 Building CSS Files (BUILD, $BUILD)

BUILD and $BUILD are used to create a CSS file. BUILD copies input lines to the file/device specified, $BUILD performs substitution first. When a BUILD or $BUILD is encountered, an attempt is made to ALLOCATE and ASSIGN the file specified in the operand field. If it exists, it is assigned. The BUILD flag (BUILDFLG) is then set positive, for a BUILD, and negative for a $BUILD. Each time a line is read, if BUILDFLG is set, BUILDDSP is entered. If $BUILD is in effect, CSS expansion is done; if BUILD, no expansion is done. The line is now checked for the special terminator either $ENDB or ENDB, and if it is found, then the BUILDFLG is reset. If not, the line is then copied to the file/device and another line is read.

5.7 LOADER

The resident loader is part of the Command Processor in OS/32 ST. It performs loads as directed by console command and also performs overlay loading functions.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

5.7.1 Common Loader (LOAD)

This command gets the file/device specified and assigns it to LU1. It then obtains the impure and pure biases. If none are present, it substitutes the value of UBOT for the impure bias, and calculates the pure bias from the size of the impure segment. The fullword loader or halfword loader is then entered.

5.7.2 Fullword Loader (LOADFULL)

The loader is entered if the user task options specify fullword mode at the time of the load. It reads records, does sequence checks, checksum, etc. It loads only object programs. When it is done, it sets up UTOP and CTOP.

5.7.3 Halfword Loader (LOADHALF)

The halfword loader loads 16 bit object programs. The halfword loader does not run in halfword mode, it is part of the Command Processor, and runs in fullword mode. It is entered if the user task options specify halfword mode at the time of the load.

5.7.4 Overlay Loads (LOADOVLY)

Before reading a new command line the Command Processor checks a flag, LOADSTAT, to see if an overlay load is being requested. If so, the LOADOVLY routine is entered. It calls the appropriate loader (LOADFULL, LOADHALF) to load the overlay.

5.7.5 Load Errors (LOADFAIL)

If an error is encountered while loading, a check is made to see if an overlay was being loaded. If so, LOADSTAT contains the location at which to store the error code. If not, a message is built LDRX-ERR, and CMDERROR is called. If an overlay is being loaded, TMREMW is called to remove the loader wait condition.

5.8 CONSOLE HANDLING

The system console device is handled by a special interrupt routine in OS/32 ST. The Command Processor always has the real device assigned to LU0. Tasks that try to assign a device that has the console bit set, are instead assigned to the dummy device. When an I/O request is issued to this dummy device, the dummy driver is entered.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

The dummy driver sets a flag to indicate to the Command Processor that I/O is being requested by a task to the console. The dummy driver also times out any read to the console being performed by the Command Processor, if no characters have been read.

When the Command Processor sees that I/O is being requested by a task (CMDPEND is non-zero), it performs the I/O for the task. It picks up the starting address, ending address, and function code from the dummy DCB (DCBCMD). If a read is requested, a ">" prompt is printed to indicate to the user that a task is requesting data. The read is then executed. If a write is requested it is performed. Any write is governed by the current value of logging. Hence, if log is set with no copy, the task's write goes only to the log device.

When the I/O has been performed the Command Processor adds an item to the System Queue for the dummy leaf (DMLV) in order to schedule the termination phase of the dummy driver. The termination phase merely calls IODONE.

5.9 THE BREAK KEY

The BREAK Key on the console has special meaning to OS/32 ST. It will cause any Command Processor initiated output (a display, EXAMINE, etc.) to be terminated. If a task write is in progress, it will be stopped, a "*" prompt printed and a command line will be read. After the command line has been read, the task write is retried. It may be interrupted as often as desired and keeps retrying until successful or until the task is cancelled. If the task is cancelled, the write will still be printed, but any queued writes are lost.

If a task is in read mode (": " has been printed) and Break is depressed, a "*" is printed and a command line is read. As above, when the command is executed, the read is retried, until completion. If a task is cancelled while a read is outstanding (a prompt has been interrupted by BREAK), the read must be completed. Simply hit carriage return.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

CHAPTER 6

FILE MANAGEMENT SYSTEM

6.1 FILE HANDLER

The routines in this package include all the logic needed to support the OS/32 ST file management system. The File handler is invoked by the SVC First Level Interrupt Handler (FLIH) any time a task issues an SVC 7 supervisor call. When an SVC 7 call is intercepted by FLIH, control is passed to the SVC 7 Second Level Interrupt Handler SVC7. This routine then decodes each function specified by the SVC 7 parameter block and invokes the necessary executors. The SVC 7 executors contain routines to:

Allocate a new file
Assign a file or device to a logical unit
Change the access privileges of a file or device
Rename a file or device
Reprotect (change the protect keys of) a file or device
Close the assignment between a logical unit and a file or device
Delete a file
Checkpoint a file or device
Fetch the attributes associated with a file descriptor

More than one function can be performed by a single SVC 7 request. Each executor that completes successfully returns to SVC7 to determine if any other requests are still outstanding. When all functions have been processed, control is returned to the calling task via TMRSOUT. If any of the SVC 7 executors encounter an error, the appropriate error status is returned in the calling task's parameter block and control returns directly to the task via TMRSOUT. These executors make use of the following routines contained within the File handler:

A directory management package for maintaining information on all currently allocated files.

A bit map management package which provides a method for allocating and deleting files on direct-access volumes.

The file manager also contains SVC 1 intercept routines which intercept all I/O calls to a file.

6.2 VOLUME ORGANIZATION AND INITIALIZATION

Any direct-access volume to be used within an OS/32-ST environment must be formatted by the STANDALONE DISC TEST and FORMAT PROGRAM (06-122), or equivalent.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

Since OS/32-ST handles file allocations in multiples of one sector, the arguments to this program must specify a DEFSEC of 1. Once a volume has been formatted using this procedure, it should not have to be formatted again unless a hardware failure occurs on the volume. After a disc is formatted, it must be INITIALIZEd, using the INITIALIZE function provided as an operator command. The INITIALIZE command will read-check each sector on the volume; any sector found to be defective will be marked as permanently allocated. INITIALIZE then creates a bit map and volume descriptor for this volume.

A Volume Descriptor is shown in Figure 6-1. The Volume Descriptor (VD) contains the volume name, a pointer to the bit map and first directory block, and a pointer and the size of an OS boot loadable image, if one exists.

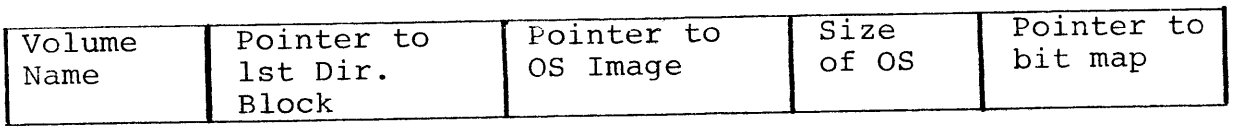


Figure 6-1. Volume Descriptor

The size of the bit map is determined by the size of the volume; each complete bit map sector represents 2048 allocatable sectors on the volume. The final sector within the bit map represents between 1 - 2048 sectors. A sector is marked as allocated when the bit representing it is set; free when the bit is reset.

The Volume Descriptor is placed on CYLINDER 0, SECTOR 0; the bit map may be located anywhere on the volume, since it is pointed to by the VD.

6.3 DIRECTORY MANAGEMENT

A file directory is maintained as a chain of directory blocks, where each directory block contains the following fields (see Figure 6-2):

A chain field containing either a zero (indicating it is the last block in the chain) or the logical block address (sector) of the next block in the chain.

A file that has just been INITIALIZEd contains no directory. The INITIALIZE logic sets the VD directory pointer (VD.FDP) to zero.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

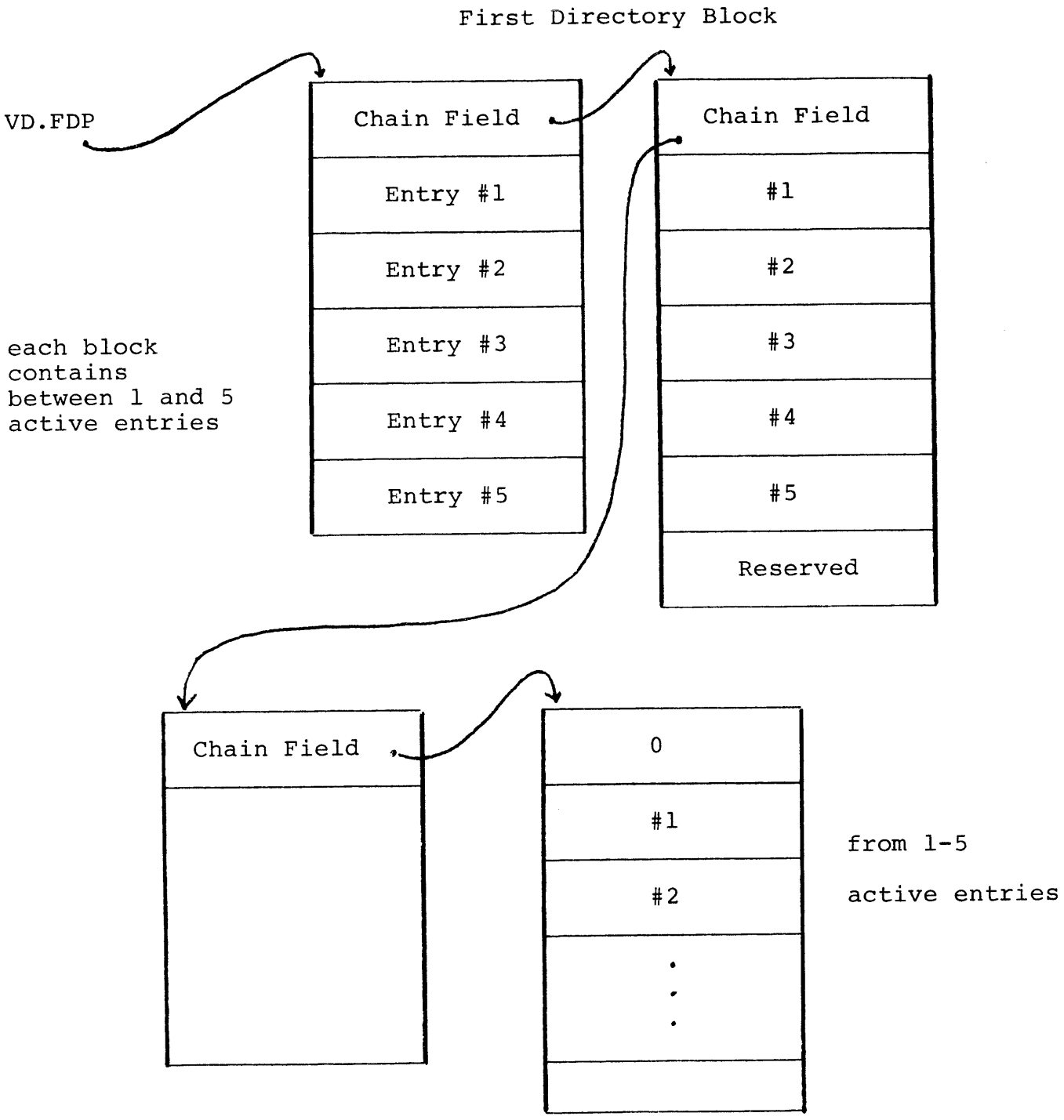


Figure 6-2. Directory Example

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

6.3.1 Directory Entry Creation and Deletion (ALLOD, RELED)

When the first file is allocated on a disc volume, a directory block is allocated. The first entry represents the new file and the remaining 4 entries are marked inactive and therefore available for additional new files. Subsequent allocations search the chain for the first unused directory entry and if none is found, a new directory block is allocated. The first directory block is always pointed to by the VD. If a file is deleted, its entry is marked inactive. If all entries in a directory block are marked inactive, the directory block is released and the chain relinked. If all the directory blocks are thus released, the VD.FDP field is again set to zero.

6.3.2 Directory Access (DIRLOOK, GETD, PUTD)

When a function is requested on a currently existing file, the directory block containing the directory entry (DIR) for the file must first be found via a call to DIRLOOK. The I/O routines used to read directory blocks into memory or to write out modified blocks are GETD and PUTD.

When a new file is allocated, and one or more directory blocks currently exist, the routine DIRLOOK searches each block until an inactive entry is found. If all entries are marked active, a new directory block is allocated as described above.

6.4 BIT MAP MANAGEMENT

Allocation of OS/32 direct access files is in multiples of one sector; the status (free or allocated) of each sector on the volume is maintained in the volume's bit map. When an INITIALIZE command is entered, all non-defective sectors within the volume are marked as free by resetting the corresponding bit in the bit map. Then the VD and bit map are created; the sectors they occupy are marked as allocated by setting the appropriate bits. The INITIALIZE logic also provides a pointer from the VD to the bit map (VD.MAP).

6.4.1 File Allocation and Deletion (GETSECTR, RELEB, GETB, PUTB)

When a request is received by the bit map management routines to allocate a string of contiguous sectors, GETSECTR searches the bit map for a corresponding number of bits that are reset, thus indicating available sectors. Since allocations may span bit map sector boundaries, one or more calls to GETB may be required to read bit map sectors into memory. When enough available sectors have been found in this manner, GETSECTR then sets each bit in the bit map within this allocation. As bit map sectors are modified, they are written back to disc via PUTB.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

When a file is deleted, the procedure is reversed by RELEB. Each bit representing the allocation is reset, indicating the sector is again available. GETB and PUTB may again be invoked, to read and write the bit map sectors.

6.5 SVC 7 SECOND LEVEL INTERRUPT HANDLER (SVC7)

The FLIH transfers control to the SVC 7 driver routine, SVC7, with two arguments, the address of the current TCB and the address of the calling task's SVC 7 parameter block.

SVC7 processes the function code specified by the parameter block from left to right. If the function code is initially zero, the call is a FETCH Attributes. Otherwise, the function code is saved in TCB.SYS, and each SVC 7 function specified within it is performed by branching to the appropriate executor. Each executor that completes successfully returns control to SVC7. As each function is performed, the bit representing it in TCB.SYS is reset, until each bit of TCB.SYS has been reset. Control then returns to the calling task via a branch to the Task Management routine TMRSOUT.

6.6 SVC 7 FUNCTION EXECUTORS

6.6.1 Allocate (ALLO)

The SVC 7 executor ALLO is called directly from SVC7 when the function code in the parameter block specifies an allocate operation.

The logic in ALLO proceeds as follows: The directory management routines are called to insure that the specified file descriptor is unique to that file, and establishes a directory entry for the file being allocated. For a contiguous type file, the complete file allocation size is established at allocation time; this is performed by the bit map management routines. Since a chain file is open-ended and has no predefined size, no allocations are performed on behalf of a chain file at allocation time. The necessary initial information is established in the directory for both file types. Control returns directly to SVC7 upon the successful completion of ALLO.

6.6.2 Assign (OPEN, OPEN.DEV, OPEN.CO, OPEN.CH)

The SVC 7 executor OPEN performs all common assign processing for direct and non-direct access devices. OPEN establishes the validity of the logical unit being assigned. If the OPEN function is being performed upon a non-direct access device,

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

OPEN transfers control to OPEN.DEV which completes any necessary validity checks, using the subroutines described in Section 6.6.10, and sets up the entry in the LU table to contain the DCB address and device attributes. If the device being opened is a direct access device, OPEN completes the assignment process itself and places the DCB address and device attributes in the LU table. Otherwise, control is transferred to OPEN.CO, if the file being assigned is a contiguous file, or OPEN.CH, if it is a chain file. In either case, the first event to occur is a call to the memory management routine, GETFCB, to allocate a File Control Block (FCB) within dynamic system space.

OPEN.CO and OPEN.CH also obtain current control information about the file from its directory entry and move this to the FCB.

OPEN.CH positions the file to the requested data block and allocates a new block for data via the bit map management routines, if the file is being opened with write privileges.

Upon successful completion, both OPEN.CO and OPEN.CH set up the entry in the LU table to contain the FCB address and a file attribute byte, which indicates the allowable data transfers to the file.

All OPEN processing successfully terminates by returning directly to SVC7.

6.6.3 Change Access Privileges (CAP)

The SVC 7 executor, CAP, performs the function of changing the access privileges associated with a given logical unit. The logical unit can be assigned to a file or device. CAP is a two pass operation.

On pass one, the routine insures that the new access privileges are legal, but makes no modifications to any control blocks. When pass one has completed successfully, the routine proceeds to pass two, this time making updates to all required control blocks to reflect the new access privileges. This requires modifying the write and read count fields in the DCB or FCB (DCB.WCNT, DCB.RCNT, FCB.WCNT, FCB.RCNT) to reflect the current access privileges. The current access privileges associated with a file are reflected in the WCNT and RCNT fields of the control block in the following manner:

```
WCNT/RCNT = 0  implies no task having write/read privileges
WCNT/RCNT = -1 implies one task having exclusive write/read
               privileges
WCNT/RCNT = +n implies n tasks sharing write/read privileges
```

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

6.6.4 Rename (RENAME)

RENAME is the SVC 7 executor that changes the name of a file or device. If the rename function is directed at a device, RENAME insures that the new name does not currently exist in the Device Mnemonic Table (DMT) and then replaces the device's previous name in the DMT with its new name. To RENAME a file, the procedure is similar except it is the directory that is checked for a duplicate name. The directory management routines are used to read the directory, search for a name match, and rewrite it with the new file name. RENAME returns to SVC7 upon successful completion.

6.6.5 Reprotect (REPRO)

The SVC 7 executor, REPRO, contains the logic to modify the protect keys associated with a given LU. The LU can be assigned to a file or device. The protect keys associated with a device are kept in its DCB (DCB.WKEY, DCB.RKEY); the protect keys associated with a file are kept in its directory entry (DIR.WKEY, DIR.RKEY). A file or device may be unconditionally protected (Keys = X'FF'), unconditionally unprotected (Keys = X'00') or conditionally protected with Write, Read Keys between X'01' and X'FE'. The logic in REPRO insures that the new protect keys are not in violation of the former protect keys, and updates the control block (DCB or Directory) with the new protect keys. Control returns to SVC7 for further SVC 7 processing.

6.6.6 Close (CLOSE)

The purpose of the CLOSE executor is to disconnect an open logical unit from a file or device. The logic of CLOSE insures that the given LU is currently assigned.

If the LU was assigned to a device, the read and write count fields in the DCB are modified as follows:

1. if old WCNT,RCNT = -1, new WCNT,RCNT = 0
   (previously one exclusive user)

2. if old WCNT,RCNT = 0, new WCNT,RCNT = 0
   (implies there were no users of this privilege)

3. if old WCNT,RCNT = n, n > 0, new WCNT,RCNT = n-1
   (previously n shared users)

If the LU is assigned to a file the WCNT,RCNT fields in the directory and FCB are updated as specified above. A test is then made to determine if the FCB should be released or if

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

it is being shared. (Current file implementations preclude the possibility of sharing FCB's. The logic is included in CLOSE for future use). If the FCB is not being shared, its memory allocation is returned to system space by a call to the memory management routine, RELEFCB. The directory management routines are then used to update the directory with the information about the file which was in the FCB. CLOSE finishes by setting this LU's entry in the LU table to zero and exiting to SVC7.

6.6.7 Delete (DELETE)

The DELETE executor is used to delete contiguous and chain files; it has no meaning with regard to devices and will generate an error if an attempt is made to delete a device. The logic in delete requires dynamic system space; this is obtained by the memory management routine GETFCB. A file is deleted by releasing its allocated storage on the volume containing it via the bit map management routines and by relinquishing its directory entry via the directory management routines. Finally, the system space is returned via a call to the memory management routine RELEFCB. DELETE returns control to SVC7 on completion.

6.6.8 Checkpoint (CHECKPT)

The SVC 7 executor CHECKPT contains the logic to checkpoint to an LU; the LU may be assigned to a file or a device. If the checkpoint function is directed to a device, a subroutine is invoked to perform an SVC 1 I/O wait only operation on the device. To checkpoint a file, all current information about the file is moved from its FCB to its directory entry. The bit map and directory management routines are used to insure that the bit map and directory on the volume reflect the current file allocations. A chain file is also positioned to read random mode using the chain file reset routine, RESET.CH, described in 6.7.2. CHECKPT exits to SVC7.

6.6.9 Fetch Attributes (FETCH)

The purpose of the FETCH executor is to obtain the attributes associated with the file or device assigned to a given LU. The device/file attributes, device code, and name are moved from the DCB/FCB to the task's SVC 7 parameter block. FETCH returns directly to the calling task via the task management routine, TMRSOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

6.6.10 SVC 7 Integrity Checking Subroutines

This section briefly describes the integrity checking subroutines used by the SVC 7 executors:

1. APCHECK - Verifies the legality of the requested access privileges; converts the requested access privilege to a numeric quantity, to be saved in the WCNT and RCNT field of the control block.

2. LUCHECK - determines if a given LU is assigned, and picks up its LU entry from the LU table.

3. DMTLOOK/VMTLOOK - searches the DMT/VMT for a given device/volume.

4. FDCHECK - checks the syntax of a given file or volume name.

6.7 SVC 1 INTERCEPT ROUTINES

When the SVC 1 processor (SVC1) determines that an SVC 1 call is directed to a file, the file management SVC 1 intercept routines are entered to process the request.

6.7.1 Contiguous File Handler

The contiguous file handler package consists of the following two routines:

CONTIG - processes data transfer requests to a contiguous file
CMD.CO - processes command function requests to a contiguous file

6.7.1.1 Data Transfer for Contiguous Files (CONTIG)

The routine CONTIG is entered directly from SVC1; the length of the data transfer request is computed and the random address is obtained either from the FCB random address (for a random I/O request) or from the FCB current sector pointer (for a sequential I/O request). CONTIG copies the FCB information into the DCB and if the I/O request is a read or write, CONTIG exits by transferring control directly to the disc driver. If the I/O request is a test and set (both read and write bits set in the SVC 1 function code), CONTIG enters RSA state, moving the RS save area to a save area in the FCB via TMRSRSA, and then modifies the following fields in the TCB:

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

TCB.RPSW - the location field of the resume PSW is set up to contain a secondary entry point within CONTIG

TCB.RGPR - the general purpose register save area is set up to contain the current values of user register set.

The routine then transfers control to the disc driver for the read portion of the test and set operation. By modifying the TCB.RPSW and TCB.RGPR fields as specified above, upon termination of the read the disc driver returns to CONTIG at its secondary entry point. CONTIG then processes the remainder of the test and set operation itself. If a write is to be performed (the first halfword of the buffer read contained a X'0000'), CONTIG does the write by issuing an SVC 1 WRITE, WAIT call. Control is returned to the calling task upon successful completion of CONTIG via the task management routine, TMRSAOUT. If CONTIG receives an EOM status following an I/O operation, EOM status is saved in the FCB and control is returned to the task by branching to IODONE2 to complete the request.

6.7.1.2 Command Requests to Contiguous Files (CMD.CO)

The command function intercept routine for contiguous files, CMD.CO, contains six command executors. Each executor and its function is briefly described below:

Rewind (CMD.REW) - Set current sector (FCB.CSEC) in the FCB to 0 and return via a branch to IODONE2.

Backspace Record (CMD.BSR) - Decrement FCB.CSEC by 1, enter RSA state and issue an SVC 1 read of new current sector to check for any I/O problems. Exit to TMRSAOUT.

Forward Space Record (CMD.FSR) - Increment FCB.CSEC by 1 and proceed as in CMD.BSR.

Write End of File (CMD.WEOF) - Increment FCB.CSEC by 1, enter RSA state and write a pseudo-file mark (X'1313') at that random address via an SVC 1 WRITE,WAIT call. Exit via TMRSAOUT.

Forward Space File (CMD.FSF) - Enter RSA state and issue SVC 1 read commands starting at FCB.CSEC, until a pseudo-file mark, X'1313' is found. Exit to TMRSAOUT.

Backward Space File (CMD.BSF) - Same as Forward Space File, except the X'1313' is searched for starting at FCB.CSEC and backing up one sector at a time. Exit to TMRSAOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

6.7.2 Chain File Handler (CHAIN, CMD.CH)

The Chain File Handler consists of the following two routines, CHAIN and CMD.CH, and various subroutines, described in 6.6.2.1. The purpose of each is:

CHAIN - process data transfer requests to a chain file

CMD.CH - process commands to a chain file

6.7.2.1 Chain File Handler Subroutines

The Chain File Handler requires various subroutines in order to process chain files. Each is briefly described below:

POSITN - position the chain file to a specific block and record beginning within that block. The current position of a chain file is indicated by the value of the FCB.CBLK.

GETCHL - move logical record from a system buffer to the task's buffer. If the logical record spans more than 1 physical block, a call is made to GETCHPR, to read the next block into a system buffer.

PUTCHL - move logical record from task's buffer to system buffer. If a logical record spans physical blocks, the block is written via a call to PUTCHP.

GETCHPR, GETCHPL - perform physical reads to a chain file to the right (entry point GETCHPR) or to the left (entry point GETCHPL). If the file is currently in sequential mode, double buffering is used; in random mode, single buffering is used.

PUTCHP - perform physical writes to a chain file. If the file is in sequential mode, the write logic uses double buffering; in random mode, single buffering is used. If the file is in write sequential mode, a new block of sectors is preallocated at this time, via a call to the bit management routine, GETSECTR. If an EOM status on the disc is returned from GETSECTR during PUTCHP processing, the file is returned to a known state by writing out the current buffer and backing up until the last logical record within the file ends in the current block.

CHDIR - establish the direction in which a chain file is to be processed (right or left).

RESET.CH - change the current state of a chain file. At any point in time, the contents of the FCB.FLGS field indicate the state of the chain file, where a state is defined as being one

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

of the following:

Operation	Processing Mode
Read	Sequential
Read	Random
Write	Sequential
Write	Random

Therefore, there are sixteen possible state changes a file may undergo, where four of these are no-ops. RESET.CH performs whatever functions are required to change a file from one state to another.

6.7.2.2 Data Transfer for Chain Files (CHAIN)

The routine CHAIN is entered directly from SVC1 in RSA state. The routine determines the state the file should be processed in based upon the function code within the FCB. EOM status is generated if a Read at the end of the file or a Write beyond the end of the file is attempted. The current state of the file is established via a call to RESET.CH. Control is transferred directly to either GETCHL or PUTCHL, to perform the logical I/O operation.

6.7.2.3 Command Requests For Files (CMD.CH)

The command function intercept routine for chain files contains 5 executors to perform the 5 allowable commands to a chain file. These executors all make use of the chain file subroutines described in Section 6.7.2.1. The following is a brief description of each executor:

Rewind (CCH.REW) - The first block in the file (block 0) is positioned to by a call to POSITN, and the current logical record field in the FCB (FCB.CLRL) is set to zero.

Backspace Record (CCH.BSR) - The previous record in the file is positioned to by decrementing the FCB.CLRL field by 1 and then positioning to the block containing this record via a call to POSITN.

Forwardspace Record (CCH.FSR) - The logic of CCH.FSR is to increment the FCB.CLRL by 1 and proceed as CCH.BSR.

Forwardspace File (CCH.FSF) - The last block in the file is positioned to (FCB.NBLK -1) via POSITN.

Backward Space File (CCH.BSF) - identical to CCH.REW.

All the executors return to the calling task via TMRSAOUT since entry to CMD.CHN is in RSA state.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

6.7.2.4 Error Recovery For Chain Files

I/O errors may occur during the processing of an SVC 1 Data Transfer or command request to a chain file. An End-of-Media status (X'90') is a software generated status that means an attempt to write to a chain file could not be satisfied because no more allocatable space exists on the direct-access volume. The file is then 'closed' in the sense that the last block in the file is written with a proper link field. The user can then continue to process the file in any way (i.e., Close, Delete, etc.) or, after making more direct-access space available on the volume, continue writing to the file.

CAUTION

Any other type of I/O error is caused by some hardware problem and may require user intervention to correct. If the user was updating an existing logical record within a file that has been closed or checkpointed, and an I/O error occurs, the file may be closed, the error corrected, and processing of that file may resume. If however the file was being processed in any manner and had not been previously closed or checkpointed, some link fields may not be set properly which causes the file to be unusable. The action taken by the user in this case should be to execute the Disc Integrity Checker Utility Program (program # 03-080) before continuing to process the file. Failure to do so may result in other files being inadvertently destroyed if the user attempts to process this file.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

CHAPTER 7

DRIVER DESCRIPTION

7.1 DRIVERS

Each driver consists of three phases: Initialization, Interrupt Service and Termination (or Event Service). Initiation phase runs as a reentrant subroutine (interrupts are enabled) of the task issuing the I/O request. In general, the initiation phase uses the information stored in the DCB by the SVC 1 executor to prepare the device dependent information required to execute the required function. After all processing has been done, the Initiation phase starts the physical I/O process by causing an interrupt on the device requested. The Initiation phase then enters the task manager which returns control to the calling task on an I/O and proceed call, or puts the calling task into I/O wait on an I/O and wait call.

When an interrupt is detected from the device, the microcode causes control to pass to the Auto Driver Channel or to the Interrupt Service Phase. If the Auto Driver Channel is employed, end of buffer or error conditions cause control to be passed to the Interrupt Service phase of the driver specified. The Interrupt service phase of the drivers execute with all interrupts disabled(except for Machine Malfunction). This phase controls the actual I/O to the device, either in conjunction with the Auto Driver Channel or by I/O instructions. Error conditions cause status to be set in the DCB. On completion of the I/O, the Interrupt Service phase disables interrupts from the device and adds the address of the device's leaf (EVT entry) to the system queue.

When a PSW is loaded that has queue service interrupts enabled, the System Queue Service routine (SQS) is entered by the microcode. SQS removes the address of the device's leaf from the system queue and schedules the termination phase of the driver specified in the leaf. This scheduling of the termination phase is called an event. The Termination phase (or event service routine - ESR) of the driver executes as an asynchronous, reentrant, non-eventable subroutine of the task which requested the I/O. The termination phase is asynchronous because it is scheduled as the result of a queue service interrupt. If the calling task is executing (or about to execute) at the time the Termination phase of the driver is scheduled, the state of the task is saved in the TCB until the ESR is complete. The ESR executes with all interrupts enabled, so it is reentrant. Non-eventable means that if another queue service interrupt occurs for the calling task while the ESR is executing, the second ESR will be queued by the System Queue Service handler for scheduling when the first ESR completes.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

The ESR performs post-processing on the I/O performed and either schedules another ISR and enters the Event Service Handler which passes control back to the calling task or schedules a queued ESR, or it enters IODONE to complete the I/O request.

The executive routine, IODONE, performs common post-processing for all drivers. It passes status and length of transfer from the DCB to the SVC 1 parameter block, calls Event Service Routines to disconnect the task from all EVT entries that were necessary to coordinate the I/O request, resets the ISP table entry for the device so that subsequent interrupts will not cause entry to the driver, removes the I/O wait condition from the task, if necessary, and enters the Event Service Handler to return control to the task or to schedule a queued ESR.

It is the responsibility of the executive to schedule driver routines in the proper state - Initiation Routines in RS, Interrupt Service Routines in IS, and Event Service Routines in ES State. It is the responsibility of the Driver Initialization and Event Service Routines to enter and exit from NSU state via LPSW, LPSWR or EPSR instructions if necessary.

7.2 DRIVER CONTROL BLOCKS

7.2.1 Device Control Block (DCB)

All standard drivers make use of the device independent portion of the DCB (see Figure 11-2). The DCB is used to pass information between the executive and the drivers; it is also used by the drivers, SVC 1 and SVC 7 to control I/O requests. The use of the Event Service Handler allows a driver to assume exclusive access to a DCB for the duration of an I/O request to the device associated with that DCB. The following section describes each field in the DCB and its usage:

DCB.DMT - Address of Device Mnemonic Table entry for this DCB. Established by the Configuration Utility Program. Used by File Manager at assign time.

DCB.LEAF - Address of Event Coordination Table entry for device described by the DCB. Established by the Configuration Utility Program. Used by the SVC 1 executor to establish task connection to the required EVT entries before passing control the the driver.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

DCB.INIT - Driver entry point for data transfer requests. Established at DCB assembly time by referencing the data transfer entry in the driver initialization routine. This entry point must have a name of the form INITxxxx where xxxx designates the driver. This address is used by the SVC 1 executor to enter the driver for data transfer requests.

DCB.FUNC - Driver entry point for command function requests. Established at DCB assembly time by referencing the command function entry in the driver initialization routine. This entry point must have a name of the form CMDxxxx where xxxx designates the driver. This address is used by the SVC 1 executor to enter the driver for command function requests.

DCB.TERM - Driver entry point for first Event Service Routine to be scheduled. Established at DCB assembly time by referencing the entry address of the desired Event Service Routine. This address is placed in the device EVT entry (leaf) at connection time (SVC 1 executor).

DCB.WCNT; DCB.RCNT - Read and Write count fields used by the File Manager to control access at assign time.

DCB.ATRB - Attributes of device. Used by File Manager to determine the attributes to associate with the device at assign time. The File Manager copies the attributes to the Logical Unit being assigned, possibly resetting the read or write bit. The Logical Unit is a field in the Task Control Block. The SVC 1 executor uses this copy of the attributes to determine the validity of an I/O request. The attribute bits are defined in Figure 7-1.

BIT	ATTRIBUTE
0	RESERVED
1	SUPPORTS READ
2	SUPPORTS WRITE
3	SUPPORTS BINARY
4	SUPPORTS WAIT I/O
5	SUPPORTS RANDOM
6	SUPPORTS UNCONDITIONAL PROCEED
7	SUPPORTS IMAGE
8	RESERVED
9	SUPPORTS REWIND
10	SUPPORTS BACKSPACE RECORD
11	SUPPORTS FORWARD SPACE RECORD
12	SUPPORTS WRITE FILEMARK
13	SUPPORTS FORWARD SPACE FILEMARK
14	SUPPORTS BACKSPACE FILEMARK
15	RESERVED

FIGURE 7-1. DCB Attribute Bit Definitions

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

DCB.RECL - This field defines the maximum length of a record for the device. Established at DCB assembly time. Used by the driver to truncate requests larger than maximum.

DCB.TOUT - Time-out constant. Established by Interrupt Service Routines to indicate desired treatment by the executive or the Timer Routines. The value of the timeout constant is defined as follows:

> -1 (X'FFFF') means the I/O request is in the process of normal termination by the driver. An ESR has been scheduled for the I/O request.
>
> 0 (X'0000') means the driver should abnormally terminate the I/O request. An ESR has been scheduled for the I/O request.
>
> $2^{15}-1$ (X'7FFF') means the I/O request is not to be timed out by a timer interrupt.
>
> 1 through X'7FFE' means the timeout constant is to be decremented by 1 every second by the system clock if there is one, until value is ZERO.

An ISR normally sets the timeout constant to the appropriate value for the device and request. After the timeout constant has been set to a positive value, all subsequent ISRs and ESRs check for timeout constant = 0. If the request has been timed out, the appropriate status is placed in the DCB, an ESR is scheduled if necessary, and the timeout constant is set to -1.

DCB.RTRY - Retry count. Established by driver if necessary. Used by standard drivers to control number of error retries.

DCB.FLGS - Flag bytes used to describe various characteristics of the device to the File Manager, the Executive and to drivers. The flag bits are defined in Figure 7-2.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

Bit	Name	Meaning and Usage
0	Bulk device flag	Set at DCB assembly time, used by File Manager.
1	On-line flag	Set at DCB assembly time, Modified by Command Processor. Used by File Manager at assign time.
2	Directory Presence	Set by system initialization routine. Used by File Manager for directory processing on device. Bulk Device Flag must also be set.
3	Bit map Presence bit	Similar to Directory presence bit.
4	Check Pseudo File Mark flag	Set by the File Manager. designates to the disc driver to check for pseudo file mark.
5	Bit Map Modify flag	Set and modified by File Manager. Indicates Bit Map must be updated on device. Bulk device flag must also be set.
6	Console flag	Device is the console device. Set by System Initialization Routine.
7	Uncancellable flag	Set at DCB assembly time. Designates to Executive not to timeout I/O to this device on End of Task.
8-15		Reserved - must be zero.

FIGURE 7-2 DCB Flag Definitions

DCB.STAT - Status field. Set by driver to indicate status of I/O request. SVC 1 executor sets value to zero before passing control to driver. Executive routine, IODONE, copies this field to status field in SVC 1 parameter block on completion of I/O request.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

DCB.DCOD - Device Code. Established at DCB assembly time. Must correspond to nnn in name of DCB module (DCBnnn).

DCB.WKEY; DCB.RKEY - File protect keys. Established by File Manager Reprotect function. Used by file manager to control access at assign time.

DCB.PBLK - Parameter Block Address. Established by SVC 1 executor. Contains the relocated physical address of the SVC 1 parameter block for current I/O request. Drivers generally do not use this address.

DCB.FC - Function Code. Established by SVC 1 executor. Used by driver initialization routine to determine nature of I/O request. Subsequent Wait only request may modify this field, therefore, ISRs and ESRs should not depend on contents.

DCB.LU - Logical Unit of current I/O request. Established by SVC 1 executor. Used by File Manager and SVC 1 executor. Must not be modified by driver.

DCB.DN - Device number. Established by Configuration Utility Program. Used by drivers to determine physical device to perform I/O request.

DCB.SADR; DCB.EADR - Data transfer start and end addresses. Established by SVC 1 executor for data transfer requests. Contains the relocated physical addresses. Used by drivers to define buffer for request.

DCB.RAND - Data transfer random address. Established by SVC 1 executor.

DCB.LLXF - Length of last transfer. Established by driver. This value is copied to length of last transfer field of SVC 1 parameter block by executive routine IODONE for data transfer requests.

7.2.2 Channel Control Block (CCB)

The Channel Control Block is used to control Auto Driver Channel operations. The address of the CCB+1 (to make it odd) is placed in the ISP table entry for a device before any servicable interrupts are generated. The CCB must reside in the first 64K of memory since the ISP table entry must contain a halfword entry. The following section describes each field in the CCB. Refer to Figure 11-1.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

CCB.CCW - Channel Command Word. Established and modified by the driver before enabling interrupts on the device. Used by Auto Driver Channel to control I/O request. The CCW bits are defined in Figure 7-3.

BITS	MEANING	
0-7	Status mask	
8	Execute Bit	
12	Buffer bit.	Zero value selects buffer 0, One value selects buffer 1 if Fast bit reset.
13	Write bit	When reset, indicates read operation.
14	Translate bit	Specifies translation if Fast bit reset.
15	Fast bit	Specifies no translate, no buffer switch, no redundance check.

FIGURE 7-3 CCW Bit Definitions

CCB.LB0 - Length of buffer 0. Used to specify length of data pointed to by buffer 0. Length is expressed as a negative number whose value is equal to start address minus end address. Thus, at any time, the length added to the ending address gives the next character to be processed.

CCB.EB0 - End Address of Buffer 0. Last character to be processed by Auto Driver Channel. Established by driver.

CCB.CW - Check Word. Used by Auto Driver Channel to accumulate redundancy check. Not used by standard drivers.

CCB.LB1 - Length of Buffer 1 (See CCB.LB0). Established by driver.

CCB.LB1 - End Address of buffer 1. Established by driver.

CCB.XLT - Translation table address. Specifies the translation table to be used by Auto Driver Channel when CCB.CCW flag bit 14 is set. Established at CCB assembly time by referencing the translation table address in the driver module.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

CCB.SUBA - Subroutine address. Specifies an ISR entry point which is branched to by the Auto Driver Channel in the following cases:

Execute bit (CCB.CCW bit 8) is reset
End of Buffer Condition
Error condition detected

Since this is a halfword field, the ISR entry point must exist in the first 64K of memory. Established by the driver.

CCB.MISC - Miscellaneous field. Established and used by drivers to pass information between Initiation, Interrupt Service and Termination phases.

CCB.FLGS - Established and used by drivers to pass information between Initiation, Interrupt Service and Termination phases.

CCB.DCB - Address of DCB for device being controlled by CCB. Established at CCB assembly time by referencing the DCB name.

7.3 DRIVER INITIALIZATION ROUTINES

Each Driver Initiation Phase has two entry points: data transfer request (INIT) entry and command request (FUNC) entry. These entry points are named INITxxxx and CMDxxxx, where xxxx designates the driver. At driver assembly time, these entry point addresses are coded in each DCB the driver controls.

Driver Initialization Routines (DIR) execute in Reentrant System (RS) state, thus executing as reentrant sub-routines of the calling task. The user task's registers and resume PSW are stored in the TCB RS save area by the SVC 1 executor. The user task is connected to the Event Coordination Table entries corresponding to the peripherals required for the I/O request. This insures that no other I/O requests can be initiated to the device until the driver requests the task's disconnection. Register 13 of the user register set contains the address of the SVC 1 parameter block, function code, logical unit, physical start and end addresses and the random address as required by the function code in the DCB.

The DIR performs the preprocessing necessary to translate the device independent SVC 1 parameter block quantities into the device dependent information to be used by the ISR and ESR portions of the Driver or by the Auto Driver Channel (see 32 Bit Series Reference Manual, Publication Number

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

29-365). After preprocessing, the DIR modifies the Interrupt Service Pointer Table entries for the devices required to point to the proper CCB or ISR and then issues a Simulate Interrupt instruction on the device address. The Driver Initialization Routine then exits to the Task Management routine TMRSOUT. This routine returns control to the calling task following the SVC 1 if the call is for I/O and proceed or it places the calling task into I/O wait state if the call is for I/O and wait.

The DIR may determine that I/O to the device is not necessary due to an error condition or because of the nature of the request. In this case, no ISR will execute. In order to terminate the I/O request, the driver does one of two things:

1. Exits to executive routine IODONE at the alternate entry IODONE2.

2. Schedules an ESR by adding the address of the leaf contained in the DCB (DCB.LEAF) to the top of the system queue.

7.4 INTERRUPT SERVICE ROUTINES

Interrupt Service Routines (ISR) execute in the Interrupt Service (IS) state. They are entered as the result of an interrupt on a device involved in the I/O request. On entry, registers 0 and 1 of the executive register set contain the resume PSW for the program that was executing at the time the interrupt was serviced. Register 2 contains the device number of the interrupting device. In the case of drivers which employ the Auto Driver Channel, Register 4 contains address of the CCB which is controlling the Auto Driver Channel. ISRs may use Registers 2 through 7 of the executive register set.

In general, all I/O instructions (e.g., SS, RD, WB) are issued from ISRs. The Auto Driver Channel is used both to perform I/O requests through the appropriate Channel Command Word and to simply transfer control to an ISR, as would Interrupt Driven I/O but with the addition of the CCB pointer in register 4. An ISR always exits by loading the PSW in Registers 0 and 1. An ISR may place another ISR entry in the Interrupt Service Pointer Table or CCB to process the next interrupt. If the ISR detects that the I/O request is complete or that some portion of the I/O request is complete (e.g., SEEK complete on a disc I/O request), it disarms the device to prevent further interrupts. The ISR then schedules the Event Service routine pointed to by the leaf (EVT entry) for the device by placing the address of the leaf on the top of

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

the system queue with an ATL instruction. It is the responsibility of the driver to insure that the ESR address contained in the leaf is the proper address before adding the leaf address to the system queue. The address in the leaf is initially set by the SVC 1 executor to the value contained in the DCB (DCB.TERM). If the driver determines that some other ESR should be scheduled, it modifies the ESR address contained in the leaf by calling the Event Service Handler routine EVMOD with the address of the new ESR in Register 14 and the leaf address in Register 15.

If an ISR detects an error condition it sets the Status field of the DCB to the appropriate value (see respective driver program descriptions). If Auto Driver Channel translation is employed, the translation subroutines are ISRs.

7.5 EVENT SERVICE ROUTINES

Event Service Routines are scheduled in the Event Service (ES) state by the Task Manager, as a result of a System Queue Service interrupt. All interrupts are enabled and the user register set is used. On entry, the registers and PSW of the task which initiated the I/O request are saved in the Task Control Block, register 13 contains the address of the DCB and register 15 contains the address of the leaf corresponding to the device. ESRs can use Registers 0 through 15 of the user register set.

ESRs perform post-processing on the I/O request being terminated, such as calculating length of last transfer, or they process intermediate I/O events in the case where the request requires more than one I/O sequence to complete (e.g., a seek and then a data transfer is required to complete a DISC read). If additional ISRs are required, the ESR may modify the CCB to schedule a different ISR, or change the address in the leaf to schedule a different ESR on completion of the ISR. If further I/O must be initiated, the ESR causes an interrupt on the device and exits by branching to the Event Service Handler routine EVRTE (return from Event). If I/O request is complete, the ESR exits to the executive routine IODONE with DCB address in register 13 and leaf address in register 15.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

7.6 HALT I/O ROUTINE (TIMEOUT)

At certain times it is necessary to cancel I/O requests that have already been started, such as in cancel processing. This is accomplished by a pseudo timeout facility in OS/32 ST. In order to halt I/O that is in progress, TIMEOUT is called from IS state with the address of the leaf corresponding to the device. TIMEOUT loads the DCB pointer from the leaf and returns if the pointer is zero (as in the case of the dummy leaf - see Section 5.7). If the DCB pointer is non-zero, TIMEOUT checks the timeout constant in the DCB. If it is zero or negative, an event service routine has already been scheduled for this request and the routine returns to the caller. If the timeout constant in the DCB is positive, TIMEOUT checks the value of the last entry in the system queue, since a power fail/restore sequence may have interrupted a driver ISR in between adding the leaf address to the system queue and setting the DCB timeout constant to -1. If the address of the leaf is not the last entry in the system queue, TIMEOUT adds it to the top of the queue, thus scheduling a termination routine for that request.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

CHAPTER 8

SYSTEM FLOW EXAMPLES

8.1 System Start Up

Figure 8-1 illustrates system flow during initialization of the system, loading and starting a task. At location X'60' is a branch to the first location of the SPT which contains a branch to SYSINIT. SYSINIT initializes the ISP Table, the EVT, DCBs and TCBs. The system TCB is placed on the ready chain. All processing is performed with all interrupts disabled. The Command Processor is branched to in ET state. The Command Processor initializes all its internal flags and buffers and uses SVC 1 to display the OS ID on the system console. It then issues a write image SVC 1 to prompt with an * and issues an SVC 1 read and wait to the system console. The system task enters I/O wait state, placing the Processor in a System Wait state. When the load command is entered, the Teletype driver ESR is scheduled by the task manager, the I/O is completed and the Command Processor resumes processing after the SVC 1. The Command Processor decodes the command and branches to the resident loader which loads the task. SVC 1 is used to write the prompt, followed by SVC 1 read and wait to the system console. The system task enters I/O wait state. The START command causes the task manager to schedule the Teletype ESR, the I/O is completed and the Command Processor resumes execution following the SVC 1. The command is decoded and the Command Processor branches to the executive routine TMSTART with the start address. TMSTART constructs a start PSW in the dispatch PSW save area of the user TCB, calls TMCHN to put the user TCB on the ready chain behind the system task and returns to the Command Processor. The Command Processor tests the user TCB to see if it is dormant and since it is not, no prompt is written to the console; an SVC 1 read and wait is issued thus leaving the user task at top of ready chain. The task manager dispatches the user task by loading the user register set from the dispatch register save area in the user TCB and loading PSW from the user TCB dispatch PSW save area.

8.2 I/O Request

Figure 8-2 illustrates system flow during an SVC 1 Write Image and wait request to the line printer. The task issues an SVC 1 write, image and wait. The First Level Interrupt Handler decodes the SVC and passes control to the SVC 1 Processor. SVC1 checks the validity of the data in the parameter block and enters RS state, saving the user registers and resume PSW in the TCB. EVQCON is called to connect to the line printer leaf. On return the information in the parameter block is stored in the line printer DCB and SVC1 branches to the DIR.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

Current Task	Executive	External Event
None	SYSINIT - Init system tables - Put system TCB on ready chain	Start processor at X'60'
Command Processor - init Command Processor data structures - SVC 1 write OS32ST - SVC 1 write * - SVC 1 read to console	Task Manager - Unchain system TCB - Enter wait state I/O	
None	IODONE - Task Manager - Complete I/O - remove I/O wait - Chain system TCB	LOAD Command
Command Processor - Decode cmd - load task - SVC 1 write * - SVC 1 read to console	Task Manager - Unchain system TCB - Enter I/O wait state	
None	IODONE - Task Manager - Complete I/O - remove I/O wait - Chain system TCB	START command
Command Processor - Decode command - Check validity of start - Branch to TMSTART - - - -	TMSTART - Set up user TCB dispatch PSW from OPTIONS and start address - Chain user TCB	
- SVC 1 read to console	Task Manager - Unchain system TCB - Set I/O wait in system TCB	
User Task - - - -		

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

Figure 8-1. System Start Up

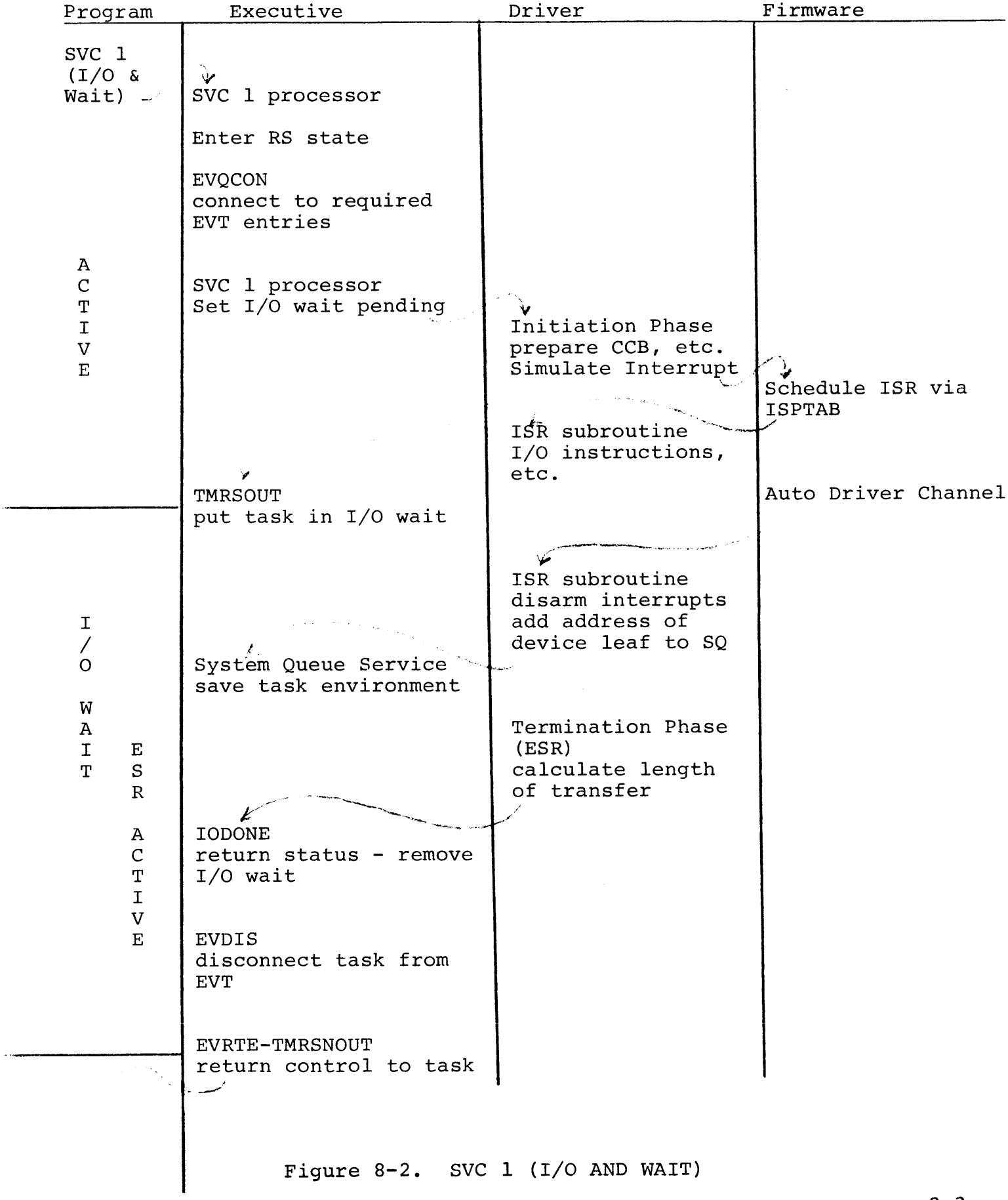


Figure 8-2. SVC 1 (I/O AND WAIT)

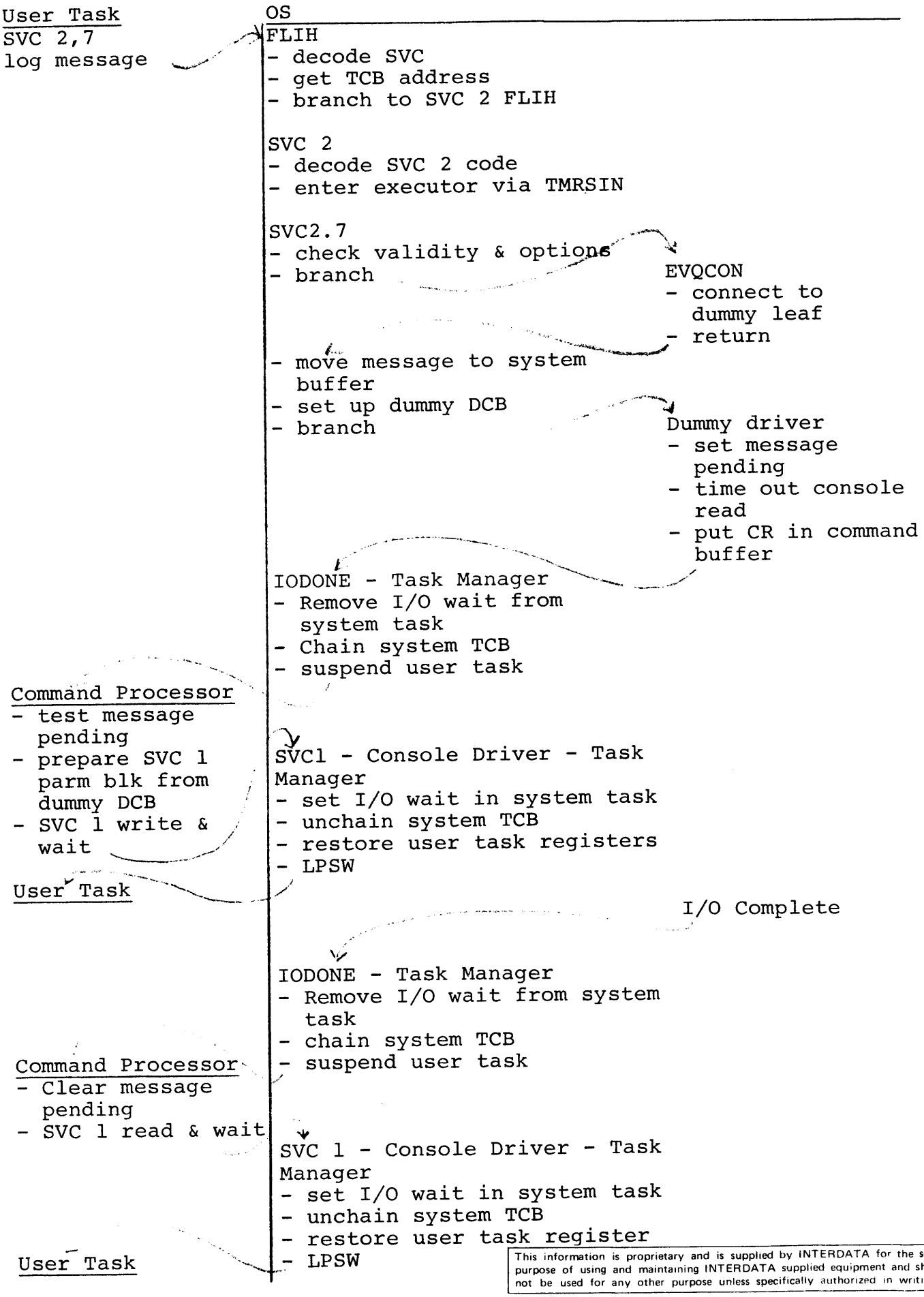
This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

The Line Printer DIR sets up the CCB with the proper CCW, subroutine address and buffer information and then SINTs the device. The microcode transfers control in IS state to the ISR routine pointed to by the CCB. The ISR sets the timeout constant, checks the device status and enables interrupts on the device. It picks up the first character in the buffer and writes it to the device and returns control to the DIR following the SINT. The DIR exits to the task manager which puts the task into I/O wait by setting the I/O wait bit, unchaining the TCB and moving the user registers and resume PSW from the RS save area to the dispatch save area of the user TCB. The Auto Driver Channel completes the transfer and passes control to the ISR on buffer empty. The ISR disarms interrupts on the printer and adds the address of the line printer leaf to the system queue. The ISR exits by loading the PSW in Registers 0 and 1 of Register set 0. This PSW is the system wait PSW since no task was active. The Queue Service enable bit is set, so the microcode causes a queue service interrupt, passing control to SQS. SQS removes the address of the printer leaf from the system queue, queues the user task to the top of the EVT by calling EVPROP and branches to EVTDISP. EVTDISP saves the user task environment by moving the dispatch save area to the ES save area in the user TCB, chains the user TCB and schedules the ESR pointed to by the leaf. The ESR calculates length of transfer, stores it in the DCB and exits to IODONE. IODONE moves the status and length of transfer to the SVC 1 parameter block, disconnects from the leaf by calling EVDIS, resets the I/O wait bit in the TCB and exits to EVRTE to return from the event. EVRTE finds no queued events and exits to TMRSNOUT which returns control to the user task by loading the user registers and resume PSW from the TCB ES save area.

8.3 Log Message

Figure 8-3 illustrates system flow during a log message request. The user task issues an SVC 2 code 7. The SVC First Level Interrupt Handler decodes the SVC and passes control to the SVC 2 Second Level Interrupt Handler. SVC 2 SLIH enters the SVC 2 code 7 executor (SVC2.7) via TMRSIN. SVC2.7 checks the options and calls EVQCON to connect to the EVT leaf for the dummy device. On return SVC2.7 moves the text to an internal buffer, sets up the DCB for the dummy device to point to the internal buffer and branches to the dummy driver. The dummy driver sets the message pending flag in the Command Processor, times out the read outstanding to the system console by storing a carriage return in the command buffer and scheduling the Teletype ESR. This causes the system task to be scheduled, suspending the user task inside the dummy driver. The Command Processor finds the command buffer empty and message pending set. The data in the DCB for the dummy device

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.



This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

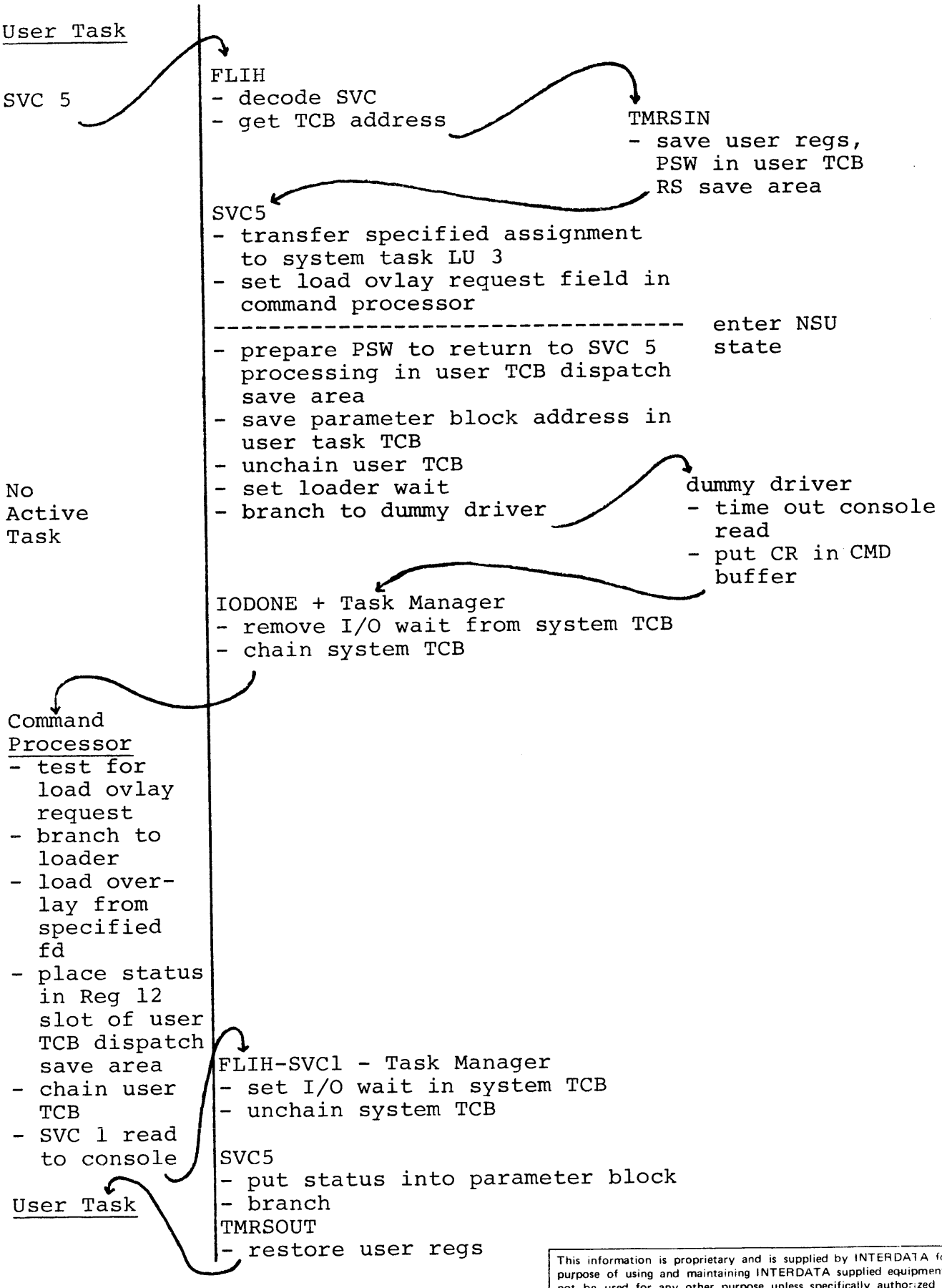
Figure 8-3. Log Message Request

is used to prepare a parameter block for the SVC 1 write and wait to the system log. The task manager puts the system task into I/O wait thus making the user task top of ready chain. The user task is dispatched at the exit from the dummy driver. The dummy driver exits to TMRSOUT which returns control to the user task following the SVC 2 code 7. When the message completes, the Teletype ESR is scheduled for the system task, suspending the user task. The Command Processor clears message pending flag and issues an SVC 1 read and wait to the system console. The task manager puts the system task into I/O wait and returns control to the user task.

8.4 Fetch Overlay Request

Figure 8-4 illustrates system flow during an SVC 5 Fetch Overlay request. The user task executes an SVC 5. The First Level Interrupt Handler decodes the SVC and enters the SVC 5 executor via TMRSIN which puts the user task into RS state. SVC5 temporarily assigns the specified device/file to the system task logical unit 3 by moving the LUTAB entry from the specified user task LU. The option field of the parameter block is stored in the Command Processor's load overlay request field. SVC 5 then enters NSU state to prevent the Command Processor from becoming active. The dispatch PSW save area of the user TCB is prepared with a PSW with RS status and a location counter of RSVC5 which is the return address in SVC 5 executor. The parameter block address is stored in the register 13 slot of the dispatch save area. SVC 5 then stores the address of the Register 12 slot of the dispatch save area into the Command Processor LOADSTAT field, sets loader wait in the user TCB, unchains the user TCB and branches to the dummy driver secondary entry point in IS state. The dummy driver times out the console read. This terminates the I/O wait condition and schedules the system task. The Command Processor detects the overlay request and branches to the loader which loads the overlay from the device/file assigned to LU3 of the system task. On completion, the status is placed in the word pointed to by the address in LOADSTAT and TMREMW is called to remove the load wait condition from the user task and to chain the user TCB. The Command Processor issues an SVC 1 read and wait which places the system task in I/O wait, thus making the user task top of ready chain. The user task is dispatched inside SVC5 at RSVC5 by loading the user register set and PSW from the user TCB dispatch save area which puts the load status in Register 12 and parameter block address in Register 13. SVC5 completes the request by storing the status into the parameter block and branching to TMRSOUT which returns control to the user task following the SVC 5.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.



This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

Figure 8-4. Fetch Overlay Request

8.5 READ REQUEST TO CHAIN FILE

Figure 8-5 illustrates system flow during an SVC 1 read request to a chain file. On execution of the SVC 1, the First Level Interrupt Handler passes control to SVC1. SVC1 checks the validity of the request and since the LU entry points to a chain file FCB, SVC1 does not attempt to perform a connection since the leaf field in the FCB contains zeroes. SVC1 enters RSA state since the request is to an FCB with the buffered access method flag set. Entry to the file manager is made at CHAIN. CHAIN resets I/O wait pending in the task, determines that it is a read request and calls GETCHL. GETCHL moves the data from the current FCB buffer to the user task buffer. When the data in the current buffer is exhausted, GETCHL calls GETCHPR to refill the buffer. GETCHPR issues an SVC 1 read and proceed call for the next sector in the file to be read into the just exhausted buffer. This causes the SVC 1 Processor to enter RS state, connect to the disc leaf and branch to the disc driver. The disc driver initiates the read and exits via TMRSOUT. Since CHAIN reset I/O wait pending, TMRSOUT returns control to the file manager in RSA state following the SVC 1 request. GETCHPR returns to GETCHL which completes the data move from the other buffer (now current). On completion, the file manager exits via TMRSAOUT which returns control to the user task following the SVC 1 read and wait.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

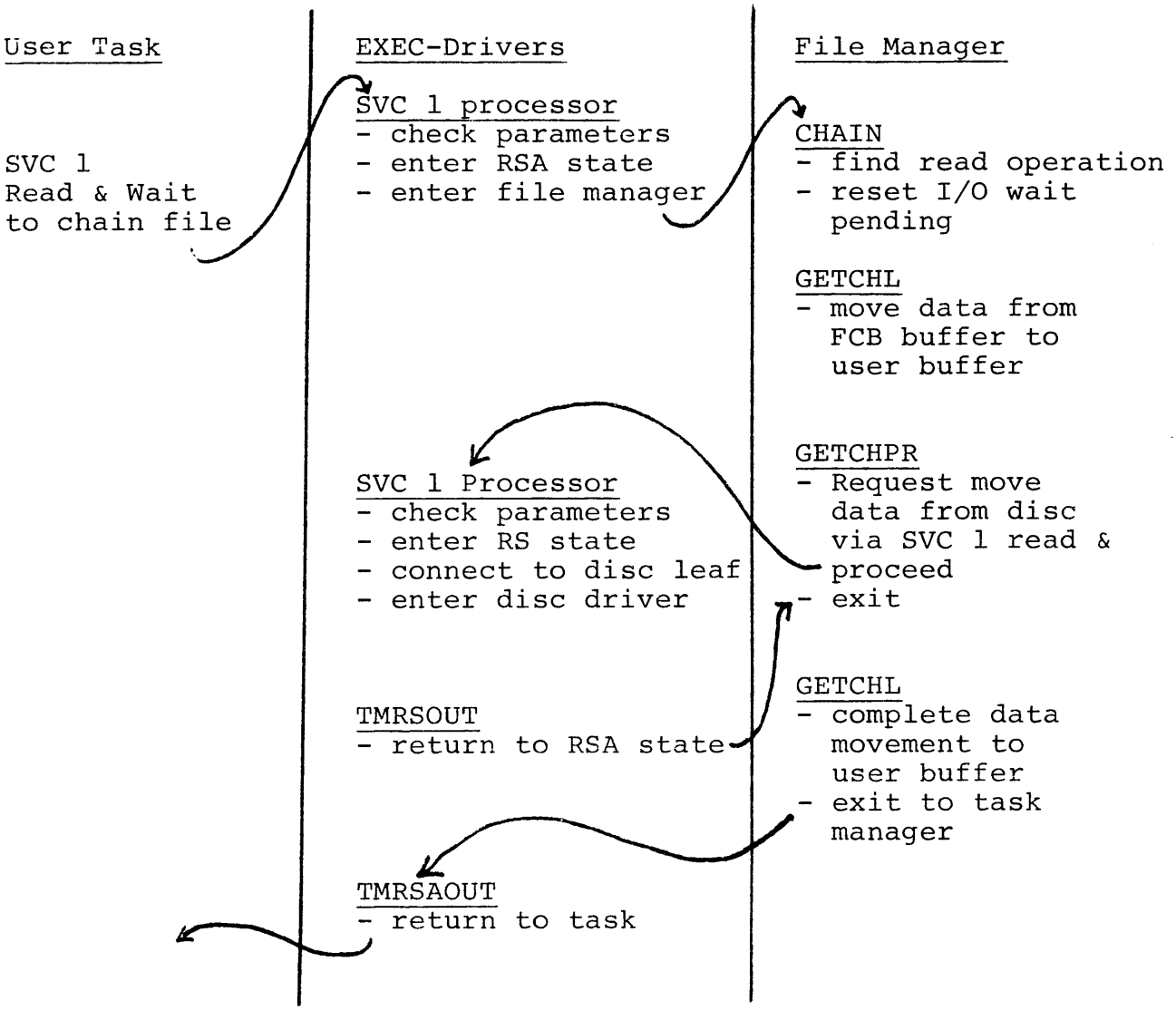


Figure 8-5. Read Request to Chain File

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

CHAPTER 9

EXECUTIVE TASKS AND SYSTEM EXTENSIONS

9.1 INTRODUCTION

There are several ways of extending or modifying the capabilities of OS/32 ST. This chapter discusses the features designed into OS/32 ST to facilitate such extensions. The user may wish to incorporate the modification directly into the system by modifying one or more system modules or by adding a system module. For example, the user may support a non-standard peripheral device by writing a driver. On the other hand, the user may wish to support infrequently used extensions to the system by writing an executive task (E-task) which may be loaded and executed on demand.

9.2 EXECUTIVE TASKS

An Executive Task (E-Task) is written as a user task and executed in ET state by specifying OPTIONS ET before starting the task. E-Tasks execute in a hardware and software privileged mode. All machine instructions are allowed and these additional capabilities are provided:

- All addresses are valid in SVC calls
- A disc device (rather than a file on the disc) may be assigned to the E-Task
- SVC 2 code 0 (Journal Entry) is valid
- REPROTECT (SVC 7) for a key of X'FF' and to non-bulk device is valid
- RENAME (SVC 7) for a key of X'FF' and to non-bulk device is valid

As a direct result of these added capabilities, E-Tasks must be designed and coded with extreme caution to prevent crashing the system. E-Tasks may not execute in halfword mode.

Access to system tables and control information is provided through the System Pointer Table (SPT). The address of the SPT is contained in the halfword at location X'62' in low memory. E-Tasks may use all SVCs. An example of a function which might require an E-Task is a disc utility program. The OS/32-ST Command Processor executes as an E-Task.

9.3 SYSTEM EXTENSIONS

OS/32 ST may be extended or modified by incorporating changes into the source of one or more system modules or adding a system module and using the Configuration Utility Program (CUP) MODULE statement to include the modified or new module in the

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

system (refer to the OS/32 ST Program Configuration Manual 29-379). All system data structures should be referenced by copying the STRUC defining the data structure from the Parameters and Control Block Module (07-063) at assembly time and using these field definitions in all instructions referencing the structure.

9.4 PATCHING

In making modifications to OS/32 ST, debugging usually entails making patches to new or existing code to avoid reassembling every time a bug is found. In order to insert a patch in OS/32 ST:

1) locate the address of SPT.UBOT in the map of the system produced by the OS/32 Library Loader (03-065).

2) Use the MODIFY command to increase the value of UBOT by an amount sufficient to contain the patch.

3) Use the MODIFY command to insert the patch starting at the old value of UBOT.

4) Use the MODIFY command or the console panel to insert a branch to the patch area.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

CHAPTER 10

JOURNAL AND CRASH CODES

10.1 CRASH CODES

After a system crash, Register 5 of Register set 0 contains a pointer to the system journal and Register 6 contains a pointer to the most recent entry to the journal. The following is a list of crash codes, their meanings and in some cases additional information concerning the cause of the crash. (Ex denotes register X of the Executive Register set (set 0); Ux denotes register X of user register set, Rx denotes register X of register set in use at time of crash).

CRASH CODE (HEX)	DESCRIPTION
1	Console device mnemonic not found in DMT
2	Unrecoverable error on system console
7	Invalid VMT during MARK processing
10	Invalid file descriptor during MARK processing
100	Arithmetic fault not in UT/ET state E9 contains address of current TCB. EE-EF contain PSW at time of fault.
101	Arithmetic fault not in user task. E9 contains current TCB ID, EE-EF contain PSW at time of fault.
102	Illegal instruction, illegal SVC or illegal address passed in SVC not in user task. E9 contains current TCB ID, ED contains pointer to 4 bytes before pointer to message, EE-EF contain PSW at time of fault.
103	Illegal instruction, illegal SVC or illegal address passed in SVC-user task not in UT/ET state. E9 contains address of user TCB, ED-EF same as for 102.
104	Memory parity error during Auto Driver Channel operation.
105*	Attempt to pause system task.
106	Illegal SVC or illegal address passed in SVC with PSW not pointing after an SVC 1 instruction. EE-EF contain PSW at time of interrupt.
107*	Attempt to remove illegal TCB from ready chain. R9-TCB ID, R8-return address.
108*	Attempt to remove a wait condition from or chain an illegal TCB ID. R8-return address, R9-TCB ID.
109*	Attempt to dispatch illegal TCB ID from top of ready chain. E9-TCB ID.
10A*	Attempt to dispatch ESR for illegal TCB ID. E9-TCB ID; EA-ES priority, EF-leaf address.
110*	Attempt to start illegal TCB ID. U9-TCB ID, UF-start location.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

CRASH CODE	DESCRIPTION
111*	Attempt to remove illegal wait bits from TCB. R8-return address, R9-TCB address, RD-wait bits.
112*	Attempt to put illegal TCB into RS state. E9-TCB ID; EA-EB-return PSW.
113*	Attempt to take illegal TCB out of RS state. U9-TCB ID.
115*	Attempt to suspend illegal TCB. E8-return address; E9-TCB ID.
118	TCB has ready chain bit set but is not on ready chain. E8-return address, E9-TCB ID.
119	Memory fault interrupt-hardware error. EE-EF-PSW at time of fault.
200*	System Queue Service interrupt-hardware error. EE-EF-PSW at time of fault.
201*	Invalid leaf address on system queue. ED-leaf address.
202*	Event for unconnected leaf. ED-leaf address.
205*	Attempt to disconnect or release leaf not connected to current task. U8-return address, U9-connected TCB ID; UF-leaf address.
206*	Release level ≤ 2 or greater than connection level for leaf. Same as for 205 with UE-release level.
207*	Attempt to connect to invalid leaf address. U8-return address; UD-DCB/FCB pointer, UF-leaf address.
208*	Attempt to modify a leaf not connected to current task. U8-return address, UE-new ESR address; UF-leaf address.
20A*	Leaf queued to system node with no task queued to leaf. EB-leaf address.
210	Entry to EVRTE not in ES state. U9-TCB ID; E0-E1-PSW at entry to EVRTE.
211	Task event count non-zero but all leaf occurrence counts zero. U9-TCB address.
212*	Leaf being disconnected has occurrence count greater than TCB event count. U8-return address; U9-TCB address; UF-leaf address.
300	I/O error reading bit map. (Disc may be write protected).
301	I/O error writing bit map.
302	Attempt to read non-existent directory block.
303	I/O error reading directory block.
304	Attempt to release non-existent directory block.
305	I/O error writing volume descriptor. (Disc may be write protected).
306	I/O error reading volume descriptor.
307	Request for FCB of invalid size.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

CRASH CODE	DESCRIPTION
308	Attempt to release FCB with FBOT=MTOP
30A	FCB not found during release attempt
30B	Invalid DCB link field during release FCB
30C	Invalid FCB chain found during release attempt
30D	Bit map or directory leaf added to system queue
30E	Invalid save attributes
30F	Attempt to close invalid file type

* denotes crash check present only if SGN.SAFE = 1

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

CODE	DESCRIPTION AND REGISTER CONTENTS
6x	Execution of SVC x. (FLIH) 12 - First word of SVC parameter block (if any) 13 - Address of parameter block 14 - SVC old PSW status 15 - SVC old PSW location (updated)
71	Task dispatched from suspended state or from NS state (TMRDISP) 12 - n.i. 13 - n.i. 14 - Status portion of PSW to be loaded 15 - Location counter of PSW to be loaded
72	Task exit from RS state. (TMRSOUT) 12 - address of TCB RS save area 13 - n.i. 14 - n.i. (if 15=0); status portion of exit PSW (15≠0) 15 - 0 means load PSW in TCB.RPSW; location of exit PSW if non-zero
73	Task entered ES state (TMRSNIN) 12 - n.i. 13 - DCB address 14 - n.i. 15 - leaf address
74	Task exit from ES state (TMRSNOUT) 12 - address of TCB ES save area 13 - n.i. 14 - n.i. 15 - 0
75	Task exit from RSA state (TMRSAOUT) 12 - address of alternate save area 13 - n.i. 14 - n.i. (15=0); status portion of exit PSW (15≠0) 15 - 0 means load PSW from save area; location of exit PSW if non-zero

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

CODE	DESCRIPTION AND REGISTER CONTENTS
76	Task entered RS state (TMRSIN) 12 - address of RS save area 13 - n.i. 14 - status portion of resume PSW 15 - location counter of resume PSW
77	Task entered RSA state (TMRSAIN) 12 - address of alternate save area 13 - n.i. 14 - status portion of resume PSW 15 - location counter of resume PSW
80*	Remove wait or Chain call (TMREMW) 12 - n.i. 13 - wait bits if Remove wait call; n.i. if chain call 14 - n.i. 15 - n.i.
91*	Illegal Instruction Interrupt (IIH) 12 - n.i. 13 - n.i. 14 - status portion of PSW at time of interrupt 15 - location counter of PSW at time of interrupt
92*	Arithmetic Fault Interrupt (AFH) 12 - n.i. 13 - n.i. 14 - status portion of PSW at time of interrupt 15 - location counter of PSW at time of interrupt
94	System Queue Service Interrupt (SQS) 12 - n.i. 13 - leaf address 14 - status portion of old PSW 15 - location counter of old PSW
95*	Connect to Leaf (EVQCON) 12 - 0 means QCON call; -1 means CON call 13 - DCB address 14 - ESR address 15 - leaf address

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

CODE	DESCRIPTION AND REGISTER CONTENTS
96	Disconnect from Leaf (EVDIS) 12 - n.i. 13 - DCB address 14 - n.i. 15 - leaf address
8001	Command Processor Command decoded (COMMANDR) 12 13 14 - Index of Command in command table 15
8002	Command Processor Dummy Driver Call (COMMANDR) 12 - n.i. 13 - n.i. 14 - n.i. 15 - n.i.
8xxx	User Journal Code (SVC 2 code 0)

* denotes journal code present only if SGN.SAFE = 1

n.i. means register contains no information or information meaningful only in context.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

CHAPTER 11

DATA STRUCTURES

11.1 INTRODUCTION

This chapter presents the formats of system control blocks and table entry. Each field is identified by its name and a descriptive title. All control blocks and table entries are referenced in OS/32 ST by copying the CAL STRUC of the same name from the OS/32 ST Parameters and Control Block module (07-063). The full field identifier is of the form:

BBB.FFFF

where BBB is the control block name and FFFF is the field name. Most fields are self explanatory; those which are not are explained following the figure for that control block. Offsets are given in the form:

DD(HH)

where DD is the offset in decimal and HH is the offset in hexadecimal.

11.2 CHANNEL CONTROL BLOCK (CCB)

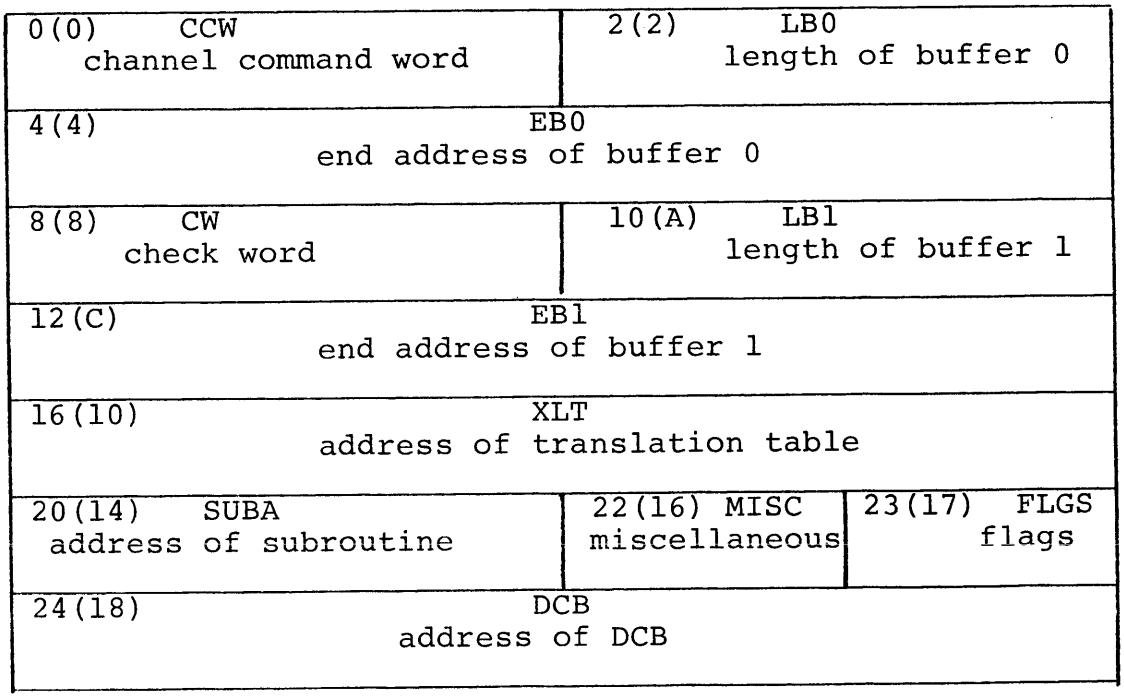


Figure 11-1. Channel Control Block (CCB)

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

- Channel Command Word (CCW)

Bit	Flag Name	Meaning
0-7	CCWSTAT	This byte is ANDed with device status; if result is non-zero, control is passed to CCB subroutine.
8	CCWEX	Execute bit. If set Auto Driver performs operation specified by CCW; if reset, control is passed to CCB subroutine.
9-11		Reserved.
12	CCBBl	Buffer bit. If reset buffer 0 in use; if set buffer 1 in use (unless bit 15 also set).
13	CCBWR	Read/Write bit. Reset means read; set means write.
14	CCBTL	Translate bit. If set translation is performed using translation table pointed to by CCB.XLT.
15	CCWFST	Fast Bit. Set indicates fast mode - no translation, buffer 0, no buffer switch, no redundancy checking.

- Miscellaneous and Flags (MISC and FLGS)

These fields are used by drivers to pass and maintain information controlling the request from DIR to ISRs and ESRs. Sometimes referenced as a halfword field, sometimes as two byte fields.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

11.2 DEVICE CONTROL BLOCK (DCB)

0(0)	DMT address of DMT entry		
4(4)	LEAF address of leaf		
8(8)	INIT address of driver data xfer entry		
12(C)	FUNC address of driver cmd function entry		
16(10)	TERM address of driver termination routine		
20(14)	WCNT write count	22(16)	RCNT read count
24(18)	ATRB attributes of device	26(1A)	RECL record length
28(1C)	TOUT time out constant	30(1E)	RTRY retry count
32(20)	FLGS flags halfword	34(22)	 reserved
36(24) STAT I/O status	37(25) DCOD device code	38(26) WKEY write key	39(27) RKEY read key
40(28)	PBLK relocated SVC 1 parameter block address		
44(2C) FC function code	45(20) LU logical unit	46(2E) DN device number	
48(30)	SADR relocated SVC 1 start address		
52(34)	EADR relocated SVC 1 end address		
56(38)	RAND SVC 1 random address		
60(3C)	LLXF length of last transfer		
64(40)	 reserved		
68(44)	 reserved		

Figure 11-2. Device Control Block (DCB)

The DCB is used by the I/O subsystem to identify characteristics of each device configured in the system and to serve as a work space for drivers during an I/O request. DCBs are pointed to by the Device Menmonic Table (DMT) entry for the device represented. DCBs are included in the system by the Configuration Utility Program (CUP) at object SYSGEN time.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

- Attributes (ATRB)

This field is used at assign and SVC 1 time to check the validity of the request.

Bit	Meaning
0	reserved
1	supports read
2	supports write
3	supports binary formatted records
4	supports wait I/O
5	supports random requests
6	supports unconditional proceed
7	supports image
8	reserved
9	supports rewind
10	supports backspace record
11	supports forward space record
12	supports write file mark
13	supports forward space file mark
14	supports backspace file mark
15	supports device dependent command

- Time out constant (TOUT)

This field is used to control device time out and halt I/O functions.

Value	Meaning
X'001'-X'7FFF'	Device active for request
X'0000'	Device has been timed out (I/O halted)
X'FFFF'	ESR has been scheduled for this request

- Flags (FLGS)

Bit	Flag Name	Meaning
0	DFLG.BLM/B	Bulk device flags
1	DFLG.LNM/B	On-line flag. Set indicates device online.
2	DFLG.DRM/B	Directory presence flag. Set indicates valid directory record in memory for this device. Bulk device flag must also be set.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

Bit	Flag Name	Meaning
3	DFLG.MPM/B	Bit Map presence bit. Set indicates valid bit map record in memory for this device. Bulk device flag must also be set.
4	DFLG.PFM/B	Set indicates moving head disc driver should check record for pseudo file mark.
5	DFLG.BMM/B	Bit map modify bit. Set indicates bit map record in memory has been modified and must be rewritten to disc. Bulk device flag must also be set.
6	DFLG.CNM/B	Console bit. Device is a console device.
7	DFLG.UCM/B	Uncancellable flag. Device not to be halted on cancel.

- Device Code (DCB)

This field is used to identify the particular device. Value must be greater than X'0F'. A value of X'FF' indicates the null device.

11.4 DIRECTORY ENTRY (DIR)

0(0)	FNM File Name		
8(8)	EXT Extension		11(B) VERS Version
12(C)	FLBA First Logical Block Address		
16(10)	LLBA Last Logical Block Address		
20(14) WKEY Write Key	21(15) RKEY Read Key	22(16) LRCL Logical Record Length	
24(18)	DATE Creation Date/Time		
28(1C)	LUSE Last Used Date/Time		
32(20) WCNT Write Count		32(22) RCNT Read Count	
36(24) ATRB Attributes	32(25) BKSZ Blocksize	38(26) FLRO First Logical Record Offset	
40(28)	CSEC Current Sector/# of Logical Records		

Figure 11-3. Directory Entry (DIR)

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

Each directory record contains up to five directory entries. Version (VERS), Creation date (DATE), date last used (LUSE), are unused in OS/32 ST.

- Current Sector/# of Logical Records (CSEC)

This field contains a pointer to the current relative sector for a contiguous file; number of logical records in a chain file.

11.5 DEVICE MNEMONIC TABLE (DMT)

0(0)	DM Device Mnemonic
4(4)	DCB Address of DCB

Figure 11-4. Device Mnemonic Table (DMT)

The DMT consists of 1 entry for each device configured in the system. The table is terminated by a doubleword of zeroes. The DMT is pointed to by the SPT. There is no structure for the DMT.

11.6 EVT LEAF (EVL)

0(0)	CORD coordination-address of parent		
4(4) CPRI connection priority	5(5) FLGS flag byte	6(6) QPRI highest queued priority	7(7) QTCB 1st TCB ID in queue
8(8) DSCN descendent number		10(A) OCNT occurrence count	
12(C)	PREV previous leaf in connected chain		
16(10)	NEXT next leaf in connected chain		
20(14)	DCB connected DCB		
24(18)	ESR entry point of ESR		
28(1C) CLEV connection level	29(1D) TSIZ tree size	reserved	31(1E) CTCB connected TCB

Figure 11-5. EVT Leaf (EVL)

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

- Flags (FLG)

Bit Offsets are from the halfword boundary - EVL.CPRI.

Bit	Flag Name	Meaning
8	EVF.LEFM/B	Leaf bit. EVT entry is a leaf.
9	EVF.ASSM/B	Assert bit. Task is asserting reconnection to upper nodes of leaf.
10	EVF.PENM/B	Pending flag. Leaf has evented.
11	EVF.DUMM/B	Non-eventing flag. This leaf doesn't represent a physical device, thus it should never appear on the system queue.

- Tree Size (TSIZ)

This is the number of entries in the path up to the system node including the leaf. For example, TSIZ = 1 for a TTY leaf; TSIZ = 3 for a disc leaf (disc leaf, disc controller node, selch node). When connection level (CLEV) = tree size (TSIZ) the task is connected to all required entries for this path. EVL is contained in the STRUC named EVT.

11.7 EVT NODE (EVN)

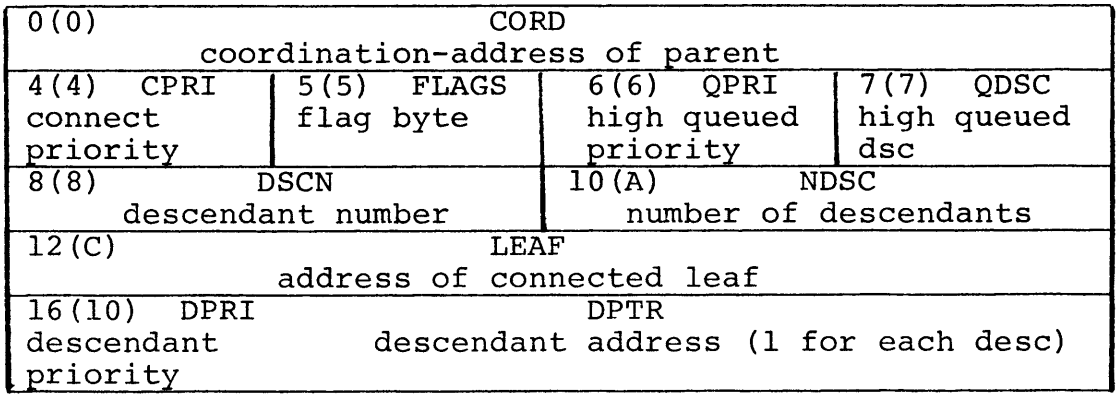


Figure 11-6. EVT Node (EVN)

The bit definitions of the flags field (EVN.FLGS) is identical to those for the EVT leaf (EVL). The leaf bit (bit 8) or the pending bit (bit 10) should never be set in a node. The length of a node is variable since the descendant pointer list occupies 4 bytes for each direct descendant.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

11.8 FILE CONTROL BLOCK (FCB)

0(0)	VMT address of VMT entry		
4(4)	LEAF address of leaf		
8(8)	INIT address of driver data xfer entry		
12(C)	FUNC address of driver cmd-function entry		
16(10)	TERM address of driver term entry		
20(14) WCNT write count		RCNT read count	
24(18) ATRB attributes of file		RECL record length	
28(1C) OFF directory offset	29(1D) BKSZ file block size	30(1E) reserved	
32(20) FLGS flag halfword		34(22) reserved	
36(24) STAT DCB I/O Status	37(25) DCOD device code	38(26) WKEY write key	39(27) RKEY read key
40(28) PBLK relocated SVC 1 parameter block address			
44(2C) FC function code	45(2D) LU logical unit	46(2E) PA physical address	
48(30) SADR relocated SVC 1 start address			
52(34) EADR relocated SVC 1 end address			
56(38) RAND SVC 1 random address			
60(3C) LLXF length of last transfer			
64(40) NAME file name			
72(48) EXT file extension			75(48) VERS reserved
76(4C) DIR address of directory block			
80(50) DCB address of DCB			

Figure 11-7. File Control Block (FCB)

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

84(54)	FLBA first logical block address
88(58)	LLBA last logical block address
92(5C)	CSEC current sector logical block address
96(50)	RPSW PSW save area
104(68)	RGPR general register save area

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

168(A8)	BAPB parameter block of buffer A		
188(BC)	FCB FCB Linkage Field		
192(CO)	SLU saved LU entry		
196(C4)	BAPT buffer for contiguous files/buffer A parm block ptr. for chained files		
200(C8)	BBPT address of buffer B parm block		
204(CC)	PBUF previous buffer		
208(D0)	NBLK number blocks in file		
212(D4)	CBLK current block number		
216(D8)	NLR number logical records in file		
220(DC)	CLRL current logical record number		
224(E0)	COFF offset of current block	218(DA) reserved	219(DB) CBUF current buffer
228(E4)	BBPB parameter block of buffer B		

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

248(F8) BUFA

BUFB

Chained Files

Buffers

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

The first portion of the FCB is defined as the DCB with the exception of a VMT pointer instead of a DMT pointer and different flag definitions. The value of the device code identifies this control block as an FCB rather than a DCB.

- Flags (FLGS)

Bit	Flag Name	Meaning
0	FFLG.BAM/B	Buffered access method flag. Set indicates a buffered access file.
1	FFLG.USM/B	Set implies FCB in use.
2	FFLG.OPM/B	Operation flag. Set implies write.
3	FFLG.ABM/B	Active buffer bit. Set implies buffer active (chain files only)
4	FFLG.BMM/B	Current buffer flag (chain files only)
5	FFLG.MOM/B	Mode flag. Set indicates random (chain files only)
6	FFLG.DIM/B	Direction flag. Set implies left (chain files only)

- Device Code (DCOD)

Device code must be X'00' (contiguous file) or X'01' (chain file).

11.9 INITIAL VALUE TABLE (IVT)

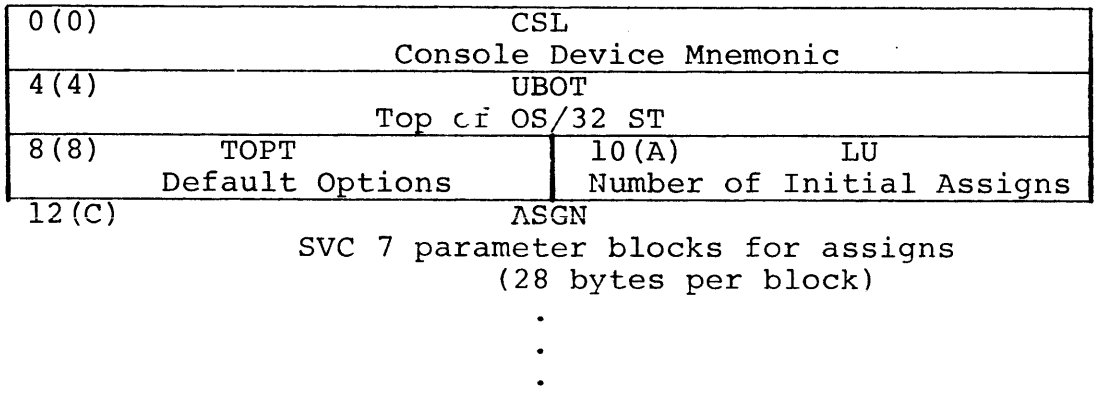


Figure 11-8. Initial Value Table (IVT)

IVT is pointed to by the SPT.

- Default Options

The default options are defined as the TCB option field.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

11.10 SYSTEM POINTER TABLE (SPT)

0(0)	INIT branch to SYSINIT		
		6(6)	CRSH system crash code
8(8)	FLV address of first leaf		
12(C)	LLV address of last leaf		
16(10)	MLBL message log buffer length	18(12)	CTSP ctop expand quantity
20(14)	CSLV number of CSS levels		
24(18)	CSBF size of CSS buffer +2		
28(1C)	CHBK maximum chain file block size	30(1E)	ISPT Top of ISP + MAC
32(20)	CTOP last halfword in allocated memory		
36(24)	UTOP lst byte above user program space		
40(28)	UBOT lst byte in user program space		
44(2C)	FBOT lst byte in system storage area		
48(30)	MTOP lst byte above configured memory		
52(34)	OSID system ID = OS32STRR RR = release level		
60(3C)	IVT address of initial value table		
64(40)	TTAB address of TCB table		
68(44)	CTCB current TCB ID	69(45) NTCB max TCB ID +1	70(46) reserved
72(48)	DMT address of DMT		
76(4C)	VMT address of VMT		

Figure 11-9. System Pointer Table (SPT)

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

80(50)	SVOL name of default volume	
84(54)	SNOD address of system node	
88(58)	JRNL address of system journal	
92(5C)	RC return code	90(5A) reserved
96(60)	PSV save area used by Task Manager	
104(68)	RSV save area used by Task Manager	
108(6C)	TSV save area used by Task Manager	
112(70)	AFSV save area used by Task Manager	

The SPT is built by CUP at object SYSTEM TIME. Location X'62' contains a halfword pointer to the first byte of the SPT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

11.11 TASK CONTROL BLOCK (TCB)

0(0) ID tcb ID #	1(1) PRI tcb priority	2(2) DPRI dispatch prio.	3(3) NLU # logical unit
4(4) OPT options halfword		6(6) STAT status halfword	
8(8) WAIT wait condition halfword		10(A) EVC event occurrence count	
12(C) PTCB previous tcb on ready	13(D) NTCB next tcb on ready	14(E) PCWT previous tcb in conn wt.	15(F) NCWT next tcb in conn wt.
16(10) SLOC default starting address			
20(14) SYS system tcb word			
24(18) USER reserved for user			
32(20) ASV alternate save area pointer			
36(24) LEAF leaf pointer during connection wait			
40(28) CLC connected leaf chain			
44(2C) DPSW dispatcher save psw			
52(34) DGPR dispatcher save registers			
116(74) RPSW rs-state save psw			
124(7C) RGPR rs-state save registers			
188(8C) EPSW es-save psw			
196(C4) EGPR es-save psw			
260(104) FMLU file manager LU			
264(108) LTAB logical unit table		(4 bytes per logical unit)	
.			
.			
.			

Figure 11-10. Task Control Block (TCB)

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

- Options (OPT)

Bit	Flag Name	Meaning
0	TOPT.ETM/B	Set means E-task, reset means user task.
1	TOPT.ACM/B	Set means continue on arithmetic fault; reset means pause.
2-14		Reserved
15	TOPT.HWM/B	Set means halfword mode task; reset means fullword task.

- Status (STAT)

Bit	Flag Name	Meaning
0	TSTT.ESM/B	Set implies valid data in TCB ES-save area. Task is non-eventable. Task in ES state.
1	TSTT.RSM/B	Set implies valid data in TCB RS-save area or alternate save area. Task may be in RS, RSA or ES state depending on other status bits.
2	TSTT.PPM/B	Pause pending. Set means task to be put into console wait on dispatch into UT/ET state.
3	TSTT.RCM/B	Set implies task on ready chain.
4	TSTT.ASM/B	Set means valid data in save area pointed to by TCB.ASV. TSTT.RSM/B must also be set.
5	TSTT.IPM/B	I/O wait pending. Task to be put into I/O wait on exit from RS or RSA state.
6	TSTT.SYM/B	Set means TCB is the system task.
7	TSTT.CPM/B	Cancel pending. Task is in SVC 3 processing.

- Wait (WAIT)

Bit	Flag Name	Meaning
0	TWT.IOM/B	I/O wait
1	TWT.CWM/B	Connection wait. Task on an EVT queue.
2	TWT.CNM/B	Console wait. Task paused.
3	TWT.LWM/B	Load wait. No task has been loaded or task in SVC 5 processing.
4	TWT.DMM/B	Dormant. Task loaded but not started or task has gone to EOT.
5-15		Reserved.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

In general, if any wait bits are set, the task is not on the ready chain. The exception to this is a task may be on ready chain in ES state (TSTT.RCM/B and TSTT.ESM/B both set) while I/O wait is set (TWT.IOM/B).

- User Field (TCB.USER)

8 bytes of the TCB are reserved for use by E-tasks for task dependent work area.

11.12 VOLUME MNEMONIC TABLE (VMT)

0(0)	VM Volume Mnemonic
4(4)	DMT address of corresponding DMT entry

Figure 11-11. Volume Mnemonic Table (VMT)

There is one entry in the VMT for each disc device configured in the system. When the disc is marked online the volume name is read from the volume directory and placed in the VMT entry. The VMT is terminated with a doubleword of zeroes. There is no structure for the VMT.

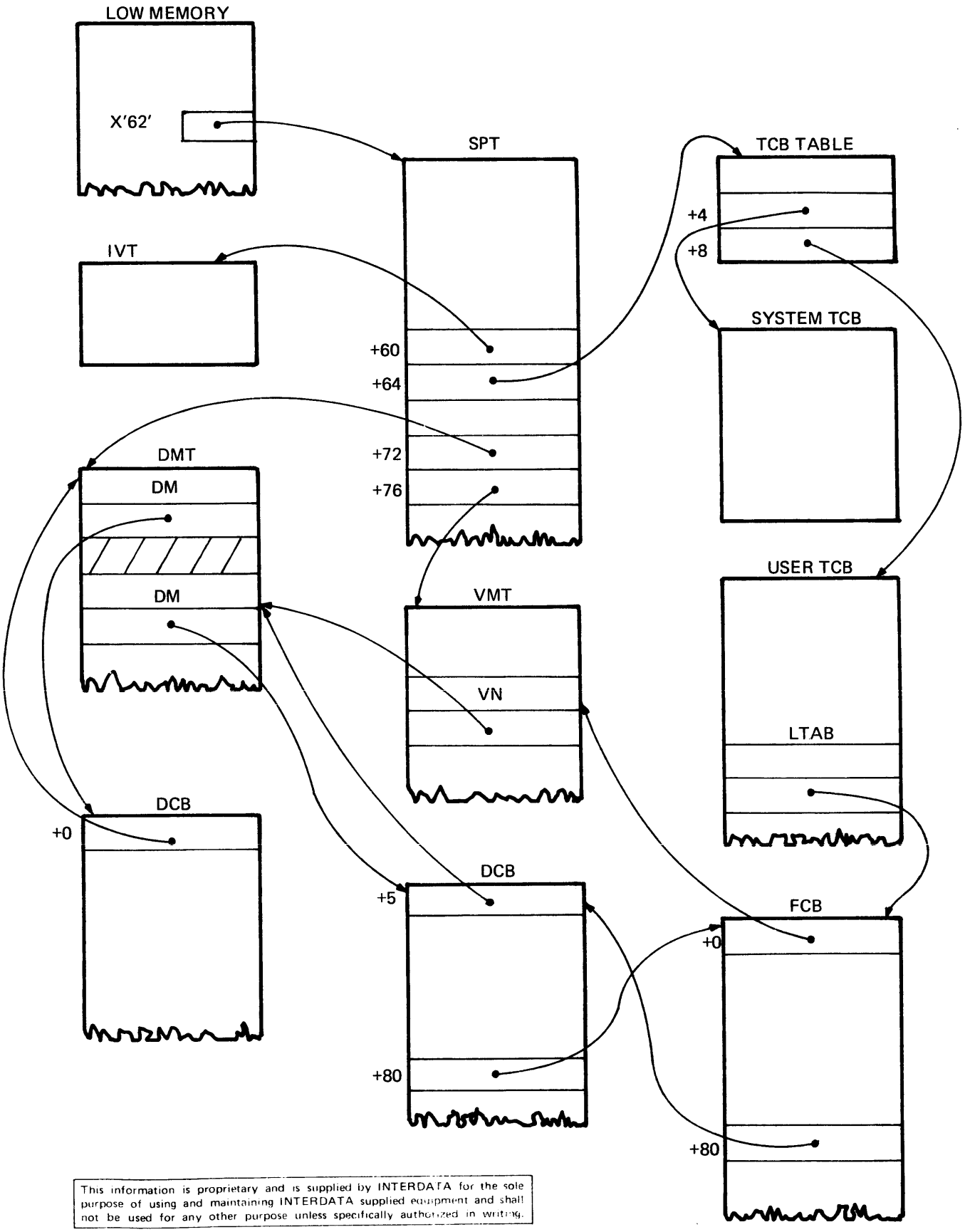
11.13 VOLUME DESCRIPTOR (VD)

0(0)	VOL Volume Name
4(4)	ATRB Volume Attributes
8(8)	FDP First Directory Block Pointer
12(C)	OSP Pointer to OS Image
16(10)	OSS Size of OS Image
20(14)	MAP Pointer to Bit Map

Figure 11-12. Volume Descriptor (VD)

The volume descriptor is written onto sector 0 of a disc pack by the INITIALIZE command. Volume attributes field is not used in OS/32 ST.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.



This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

Figure 11-13. System Data Structure Relationships

CHAPTER 12

MODULE DEFINITIONS

12.1 INTRODUCTION

The following sections include module definitions for each module in the three major portions of OS/32 ST: Executive, Command Processor and File Manager, as well as the Floating Point Trap routine. The module definitions are meant to be used in conjunction with the corresponding flowchart in OS/32 ST Program Logic Manual, Publication Number Volume II 29-382 and the appropriate object listing. The ENTRY and EXTRN names are names which are referenced within the major module groupings as well as between the major module groupings. At the start of each section is listed the ENTRIES and EXTRNS which are needed to bind the major module groupings together. For module descriptions of the OS/32 General Purpose Driver, see the OS/32 General Purpose Driver Manual, Publication Number 29-384.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

12.2 EXECUTIVE MODULES

ENTRIES:	ADCHK	CANEOJ	EVCON	EVDIS
	EVMOD	EVQCON	EVREL	EVRTE
	EXEC.XXX	FPHWLBIA	IIH	III
	IODONE	IODONE2	ISPTAB	MEMFAULT
	MEMFLTRS	NSEVREL	S21PAUSE	SPT.AFSV
	SPT.CHBK	SPT.CRSH	SPT.CSBF	SPT.CSLV
	SPT.CTCB	SPT.CTOP	SPT.CTSP	SPT.DMT
	SPT.FBOT	SPT.FLV	SPT.INIT	SPT.ISPT
	SPT.IVT	SPT.JRNL	SPT.LLV	SPT.MLBL
	SPT.MTOP	SPT.NTCB	SPT.OSID	SPT.PSV
	SPT.RC	SPT.RSV	SPT.SNOD	SPT.SVOL
	SPT.TSV	SPT.TTAB	SPT.UBOT	SPT.UTOP
	SPT.VMT	TMREMW	TMRSAIN	TMRSAOUT
	TMRSOUT	TMSTART		

EXTRNS:	CMDBUFFS	CMDLR	COMMAND	DCBCMD
	DMT	INITCMD$	IVTBL	JRNLBKS
	LGMBUFF	LOADSTAT	SNOD	SPT.STCB
	SPT.UTCB	SQ	SVC7	TCBTAB
	UBOT	VMT	XLTTYKP	

SYSTEM PARAMETERS:
SGN.JRNL
SGN.HWRD
SGN.SAFE

LIBRARY ROUTINES:
CCB
DCB
EVT
FCB
REGS
SGN
SVC1.
TCB

MODULE	PAGE
FLIH	12-9
IIH	12-10
AFH	12-11
MMH	12-12
SQS	12-13
SVC1	12-14
SVC2	12-15
SVC2.0	12-16
SVC2.1	12-17
SVC2.2	12-18
SVC2.3	12-19

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

MODULE	PAGE
SVC2.4	12-20
SVC2.5	12-21
SVC2.6	12-22
SVC2.7	12-23
SVC2.15	12-24
SVC2.16	12-25
SV2ADCHK	12-26
SVC2.17	12-27
SVC2.18	12-28
SVC2.19	12-29
SVC2.20	12-30
SVC3	12-31
SVC5	12-32
TMDISP	12-33
TMRDISP	12-34
EVTDISP	12-35
TMSTART	12-36
TMSTOP	12-37
TMRSIN	12-38
TMRSOUT	12-39
TMRSNIN	12-40
TMREMW	12-41
TMUCHN	12-42
EVCON	12-43
EVHOOK	12-44
EVMOD	12-45
IODONE	12-46
EVDIS	12-47
EVRTE	12-48
EVPROP	12-49
CANEOJ	12-50
EXECMSG	12-51
EXECRSCC	12-52
UNPACK	12-53
JOURNAL	12-54
TIMEOUT	12-55
ADCHK	12-56
SYSINIT	12-57
CRSH	12-58

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

12.3 COMMAND PROCESSOR MODULES

ENTRIES:	CMDLR	CMDP.XXX	COMMAND	DCBCMD
	INITCMD	INITCMD$	LOADSTAT	TERMCMD
EXTRNS:	CANEOJ	CMDBUFFS	DMLV	DMTLOOK
	GETSECTR	IODONE	IVTBL	PTRSTACK
	RELEB	S21PAUSE	SPT.CHBK	SPT.CSBF
	SPT.CSLV	SPT.CTOP	SPT.CTSP	SPT.DMT
	SPT.FBOT	SPT.IVT	SPT.MTOP	SPT.OSID
	SPT.RC	SPT.STCB	SPT.SVOL	SPT.UBOT
	SPT.UTCB	SPT.UTOP	SPT.VMT	SQ
	TMREMW	TMRSAIN	TMRSAOUT	TMRSOUT
	TMSTART	VMTLOOK		

SYSGEN PARAMETERS:
- SGN.BCMD
- SGN.CSS
- SGN.DA
- SGN.HWRD
- SGN.JRNL

LIBRARY ROUTINES:
- DCB
- DIR
- FCB
- IVT
- REGS
- SGN
- SVC1.
- SVC7.
- TCB
- VD

MODULES	PAGE
DUMMY DRV	12-59
COMMAND	12-60
COMMANDR	12-61
CMDERROR	12-62
CONTINUE	12-63
START	12-64
CANCEL	12-65
BIAS	12-66
WRITE	12-67
READ	12-68
JOB	12-69
OPTIONS	12-70
SET	12-71
SETLOG	12-72
PAUSE	12-73
CMCLOSE	12-74

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

MODULE	PAGE
CLOSSUB	12-75
EXPAND	12-76
CMRENAME	12-77
REPROTEC	12-78
ASSIGN	12-79
MARK	12-80
MARKSUB	12-81
MARKOFF	12-82
MARKSUB2	12-83
DISPLAY	12-84
DEVICES	12-85
LU	12-86
DSPARMS	12-87
$CLEAR	12-88
$COPY, $NOCOPY	12-89
$EXIT	12-90
SETCND	12-91
CSSIFS	12-92
BUILDS	12-93
$JOB	12-94
$TERMJOB	12-95
$SKIP	12-96
SETLU	12-97
CHECKCSL	12-98
COMMACK	12-99
SCANNER, TERMCHK	12-100
MNMFIND	12-101
BUFFINIT	12-102
CSSBUF	12-103
CSSCLOSE	12-104
MSGLOG	12-105
PREPRO	12-106
BLANKBUF	12-107
DISPFD	12-108
CMDCLOSE	12-109
CMDASGN	12-110
CMDWRITE	12-111
MAG TAPE CMDS	12-112
MOFFBLK	12-113
MONBLK	12-114
VOLUME	12-115
CMDELETE	12-116
ALLOCATE	12-117
DISPFILE	12-118
INITIAL	12-119
LOADER	12-120
LOADOVLY	12-121
LOADFULL	12-122
LOADHALF	12-123
CHEWING	12-124
CHECKER	12-125/12-126

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

12.4 FILE MANAGER MODULES

ENTRIES:	DCB25500	DMTLOOK	FMGR.XXX	GETSECTR
	RELEB	SVC7	VMTLOOK	
EXTRNS:	DCBCMD	DMT	EVDIS	EVQCON
	IODONE2	SPT.CHBK	SPT.CTOP	SPT.DMT
	SPT.FBOT	SPT.IVT	SPT.MTOP	SPT.VMT
	TMRSAOUT	TMRSOUT		

SYSGEN PARAMETERS:
SGN.CH
SGN.CO
SGN.DA

LIBRARY ROUTINES:
DCB
DIR
EVT
FCB
IVT
REGS
SGN
SVC1.
SVC7.
TCB
VD

MODULES	PAGE
SVC7	12-127
OPEN	12-128
OPEN.DEV	12-129
OPEN.CO	12-130
OPEN.CH	12-131
APCHECK	12-132
GETSECTR	12-133
GETB	12-134
RELEB	12-135
FDCHECK	12-136
LUCHECK	12-137
DIRLOOK	12-138
GETD	12-139
RELED	12-140
DMTLOOK	12-141
ALLOD	12-142
GETFCB	12-143
RELEFCB	12-144
ALLO	12-145
DELETE	12-146

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

MODULE	PAGE
RENAME	12-147
REPRO	12-148
CLOSE	12-149
CAP	12-150
CHECKPT	12-151
FETCH	12-152
CONTIG	12-153
CMD.CO	12-154
CHAIN	12-155
CMD.CH	12-156
GETCHL	12-157
PUTCHL	12-158
SET.LRCL	12-159
CHN.WAIT	12-160
GETCHRR	12-161
PUTCHP	12-162
POSITN	12-163
CHDIR	12-164
RESET.CH	12-165/12-166
PUTD	12-167
PUTB	12-168
TMRSRSA (TMRSARS)	12-169/12-170

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

12.5 FLOATING POINT TRAPS

ENTRY: FLTP.S00

EXTRNS: IIH

MODULES	PAGE
TRAPS	12-171/12-174

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: First Level Interrupt Handler (FLIH)

ABSTRACT: All SVC requests are vectored by the microcode to FLIH which performs validity checking on the parameter block address, calculates the TCB address, makes a Journal entry and enters the SVC executor in the state indicated in the FLIH SVC table.

ENTRYS: FLIH1, FLIH2, FLIH3, FLIH5, FLIH7

SOURCE LIBRARY ROUTINES: SGN, TCB, SPT

EXTRN: JOURNAL, ADCHK, MEMFAULT, TMRSIN, ISH

REGISTERS USED: E8-EF

ON ENTRY:
EE-EF contain PSW at time of SVC interrupt
ED - contains parameter block address
NS state

ON EXIT:
R9 - TCB address
RD - parameter block address
RE, RF - PSW at time of SVC interrupt
NS or RS state depending on SVC table

PRINCIPLES OF OPERATION:

Each SVC enters FLIH in NS state at FLIHx, where x is the SVC number, via the SVC new location table in low memory. FLIH also contains the low memory location data ORGed at absolute X'0'. The appropriate journal code X'6x' is stored in FLIHJC and common processing is entered. FLIH calculates the TCB address and checks the system state for UT/ET or RSA state. If state is not one of these, the crash handler is entered. Otherwise, the journal code is picked up and the SVC number extracted and used to pick up an attribute word describing the SVC in the FLIH SVC table (FLIHTAB). If a parameter block is required, the address in register 13 is checked for validity and alignment by a call to ADCHK. The end address in the parameter block is checked and the first work of the parameter block is loaded into register 12 for the Journal call. FLIH then enters the specific executor by branching to the address in FLIHTAB if NS entry is required or by calling TMRSIN to enter the executor in RS state. The format of a FLIHTAB entry is:

Bit	Meaning
0	Reset means NS entry; Set means RS entry
1	Set means parameter block must be writeable
2	Set means perform parameter block checking
3-14	reserved
15	set means SVC allowed in halfword mode
16-23	Fullword parameter block length
24-31	Halfword parameter block length
32-63	Executor address

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: Illegal Interrupt Handler (IIH)

ABSTRACT: IIH performs error processing for illegal instruction, illegal SVC and illegal address in SVC faults as well as common processing for memory parity error. MAC Faults and Ignore Immediate Interrupt routines are also contained in IIH.

ENTRYS: IIH, ISH, ISHRS, MEMFAULT, MEMFLTRS, MFH, III, IIHCOM

SOURCE LIBRARY ROUTINES: SGN, TCB, SPT

EXTRN: JOURNAL, TMRSIN1, EXECMSG

REGISTERS USED: E8-EF, U0-UF

ON ENTRY: EE, EF - PSW at time of interrupt
NS state (IIH, ISH, MEMFAULT)
RS state (ISHRS, MEMFLTRS)

ON EXIT: Pause Pending bit set in TCB
Exit via EXECMSG in RS state

PRINCIPLES OF OPERATION:

On entry, the fault handler load Register 13 with a pointer to a parameter block describing the appropriate error. Register 12 is loaded with the address at the time of the fault. IIHCOM is then entered. IIHCOM checks the TCB ID and task status and crashes if error occurred during execution of system code. If error occurred in user task, RS state is entered (unless entered at ISHRS, MEMFLTRS), the pause pending bit is set in the user TCB, and the message start, end and address fields and the address of the fault are loaded into registers and exit is made to EXECMSG.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: Arithmetic Fault Handler (AFH)

ABSTRACT: AFH processes arithmetic fault interrupts by placing task in pause state (unless Arithmetic Fault Continue is set) and issuing a message describing the error.

ENTRYS: AFH

SOURCE LIBRARY ROUTINES: SGN, TCB, SPT

EXTRN: JOURNAL, TMRSIN1, EXECMSG

REGISTERS USED: E8-EF, U0-UF

ON ENTRY: EE, EF - PSW at time of interrupt
NS state

ON EXIT: Pause Pending set (unless AFCONT set)
Exit via EXECMSG in RS state

PRINCIPLES OF OPERATION:

AFH checks the TCB ID and system state for validity and enters the crash handler if the fault occurred in system code. If the fault occurred in user task code, the TCB option field is checked and if AF continue is reset, the pause pending bit is set in the status field. AFH then enters RS state via TMRSIN1 entry point in TMRSIN and prepares registers with the message start, end and address pointers and the address at the time of the fault (instruction following the instruction causing the fault) and exits to EXECMSG.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

NAME: Machine Malfunction Handler (MMH)

ABSTRACT: MMH processes memory parity, power fail and power restore interrupts. Parity errors are processed by branching to common interrupt processing. Power fail/restore sequence is processed by saving the state of the system on power fail and on power restore, pausing the active task, if there is one, and halting all I/O except disc transfers which are retried.

ENTRYS: MMH, MMWPSW, CRSPSW

SOURCE LIBRARY ROUTINES: SPT, TCB, DCB, CCB

EXTRN: TIMEOUT, IIHCOM, TCBTAB, SQ

REGISTERS USED: E0-EF

ON ENTRY: EE, EF - PSW at time of interrupt
PSW condition code - C bit means Auto Driver Channels active
L bit means Power Fail interrupt
V, G bits mean parity error
if no bits set it is a power restore interrupt

ON EXIT:
parity error - IIHCOM
power fail - load a wait PSW
power restore - return to state of system at time of power fail

PRINCIPLES OF OPERATION:

Entry to MMH is an interruptable state; the condition code is used to branch to the appropriate routine. For parity errors, MMH enters NS state, loads register 13 with a pointer to the parameter block describing the memory parity error, loads register 12 with the location of the interrupt and branches to IIHCOM for common interrupt processing.

On power fail, MMH checks for power restore incomplete and if found loads an enabled wait PSW. If not found, MMH saves the Machine Malfunction old PSW and both register sets in an internal save area, sets power restore incomplete and loads an enabled wait PSW.

On power restore, MMH enters IS state, goes down the connected leaf chain for each TCB and halts I/O on all connected devices and sets pause pending in the user TCB. MMH then issues the power restore message by obtaining the console device DCB from LU0 of the system TCB, setting up a CCB to control the write, setting the DCB status to -1 and SINTing the device. The message is output by the TTY driver ISRs. On completion of the message, the final TTY ISR sets DCB status to zero and adds the TTY leaf to SQ. MMH loops, testing the DCB status for zero in NS state. When DCB status goes to ZERO, MMH resets the status to -1, removes the leaf address from SQ to prevent a spurious SQS interrupt and reads the GO command by the same method. If anything other than GO was entered, the messa : process is repeated. After GO is entered, MMH reloads both register sets, resets the power restore incomplete flag and loads the PSW at time of power fail.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: System Queue Service (SQS)

ABSTRACT: SQS handles system queue service interrupts by scheduling ESRs for the eventing task or queuing the event to the task if it is non-eventable.

ENTRYS: SQS

SOURCE LIBRARY ROUTINES: SGN, SPT, TCB, EVT

EXTRN: JOURNAL, TCBTAB, TMSTOP, TMRDISP, EVTDISP, EVPROP

REGISTERS USED: E9-EF

ON ENTRY: EE,EF - PSW at time of SQS interrupt
ED - address of system queue
NS state

ON EXIT: NS state to TMRDISP or EVTDISP

PRINCIPLES OF OPERATION:

SQS removes the entry from the bottom of SQ and checks for a valid leaf address. After making a journal entry, SQS derives the address of the TCB of the connected task. The event count is incremented in the leaf and the TCB. If the task is non-eventable (already in ES state) or the pending flag is set in the leaf, SQS returns to the interrupted task (or system Wait state) by loading the PSW in EE-EF. Otherwise SQS sets pending in the eventing leaf, walks up the EVT to the highest connected ancestor and sets the assert flag. SQS then calls EVPROP to propagate the connection priority of the highest connected entry. On return, if the priority propagated up to the system node and if the priority of the interrupted task (current task) is lower than the eventing task's priority, the current task is suspended by a call to TMSTOP and SQS exits to EVTDISP to dispatch the eventing task's ESR. Otherwise, SQS returns control to the interrupted task by loading the PSW in register EE-EF.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: SVC 1 Handler (SVC1)

ABSTRACT: SVC 1 performs validity checking on data passed in the SVC 1 parameter block, connects to necessary EVT entries, prepares the DCB and enters the appropriate driver or file manager routine for SVC 1 I/O requests.

ENTRYS: SVC1

SOURCE LIBRARY ROUTINES: SGN, SVC1., SPT, TCB, DCB, FCB, EVT

EXTRN: TMNSOUT, TMSTOP, TMUCHN, TMDISP, TMRSIN1, TMRSAIN, EVCON, EVQCON, ADCHK, MEMFLTRS, TMRSOUT

REGISTERS USED:
E9-EF, U0-UF

ON ENTRY: EE,EF - PSW at time of SVC 1 interrupt
ED - parameter block address
E9 - TCB address
NS state

ON EXIT: To Driver/File Manager: UD - parameter block address
RS or RSA state

PRINCIPLES OF OPERATION:

SVC1 picks up the DCB/FCB address from the specified LTAB entry. If the null device is assigned, SVC1 returns with valid status. If the LU is not assigned, SVC1 returns a status of X'81'. If the request is for a command function, SVC1 compares the specified function against the second byte of the attribute field of the DCB. If any bit is set in the function code that is not set in the DCB, an error status of X'C0' is returned (illegal function). If the request is for data transfer, SVC1 makes a similar check against the attributes field of the LTAB entry.

If the request is for wait only or test I/O complete, SVC1 loads the address of the leaf from the DCB, checks the state of the previous I/O request and returns the appropriate condition code for test I/O complete or a completed wait only call. For a wait only call referencing an incomplete I/O and proceed, SVC1 stores a wait only function code in the DCB function code field, sets the I/O wait bit, removes the TCB from the ready chain and calls TMSTOP to save the user registers and resume PSW, thus turning the I/O and proceed request into an I/O and wait. SVC1 then exits to TMDISP to dispatch the new current task.

If the request is not wait only/test I/O complete, SVC1 enters RS state (if not a buffered access method request) or RSA state. The start and end addresses are checked for data transfers. SVC1 then calls EVQCON or EVCON (unconditional proceed) to connect the required EVT entries unless the leaf address is zero in the DCB or the request is for a command function to a file (FCB). SVC1 then prepares the DCB or FCB with the function code, logical unit, parameter block address, and for data transfers, start, end and random addresses. SVC1 then exits to the address in INIT DCB/FCB.INIT (data transfers) or DCB/FCB.FUNC (command functions).

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: SVC 2 Second Level Interrupt Handler (SVC2)

ABSTRACT: SVC2 performs common preprocessing and decoding for the various SVC 2 codes. SVC2 then passes control to the appropriate executor.

ENTRYS: SVC2

SOURCE LIBRARY ROUTINES: SGN, SPT, TCB

EXTRN: TMRSIN, ISH

REGISTERS USED: E8-EB, ED

ON ENTRY:
EE, EF - PSW at time of interrupt
ED - parameter block address
E9 - TCB address
NS state

ON EXIT:
RE, RF - PSW at time of interrupt
RD - parameter block address
R9 - TCB address
NS or RS state depending on SVC 2 code

PRINCIPLES OF OPERATION:

SVC2 checks the validity of the code and checks register specifications for SVC 2 codes that require them. SVC2 then picks up the address of the executor from an internal table (SVC2TAB) and branches to the routine (NS entry executors) or calls TMRSIN (RS entry executors).

SVC 2 codes 8 and 9 cause SVC2 to return control to the task via TMNSOUT, thus ignoring these calls.

SVC2TAB consists of 4 bytes for each code.

Bit	Meaning
0	set means RS entry; reset means NS entry
1	set means not valid in halfword mode
5-6	00 means no register check performed 10 means check halfword register number 01 means check two byte register numbers
8-31	executor entry

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: Journal Entry Call (SVC2.0)

ABSTRACT: SVC2.0 makes a user journal entry for privileged tasks

ENTRYS: SVC2.0

SOURCE LIBRARY ROUTINES: SGN

EXTRN: JOURNAL, TMNSOUT, ISH

REGISTERS USED: E8-EF

ON ENTRY: EE, EF - PSW at time of interrupt
ED - parameter block address
NS state

ON EXIT: NS state

PRINCIPLES OF OPERATION:

SVC2.0 checks to insure task is privileged (Protect mode bit reset in PSW), loads the journal code from the parameter block, sets the high order bit, stores the code in the halfword following the call to JOURNAL, loads EC-EF from the last four words of the parameter, BALs to JOURNAL and exits via TMNSOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: SVC 2 CODE 1 PAUSE (SVC2.1)

ABSTRACT: SVC2.1 executes pause requests via SVC 2 code 1 or PAUSE command.

ENTRYS: SVC2.1, S21PAUSE

SOURCE LIBRARY ROUTINES: SGN, TCB

EXTRN: TMRSOUT, TCBTAB

REGISTERS USED: U8-UA, UF

ON ENTRY:
U9 - TCB address (SVC2.1); TCB ID (S21PAUSE)
U8 - return address (S21PAUSE)
RS or ET state

ON EXIT: RS or ET state

PRINCIPLES OF OPERATION:

SVC2.1 loads U8 with address of TMRSOUT for exit and U9 with TCB ID. S21PAUSE checks the TCB to insure validity, sets the pause pending bit in TCB status field and exits via register 8.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: SVC 2 CODE 2 GET STORAGE (SVC2.2)

ABSTRACT: SVC2.2 processes GET STORAGE requests by adjusting UTOP appropriately and returning the old value of UTOP in user register zero. UTOP is maintained on a fullword boundary.

ENTRYS: SVC2.2

SOURCE LIBRARY ROUTINES: SGN, SPT, TCB

EXTRN: ADCHK, MEMFLTRS, EXECRSCC

REGISTERS USED: U8-UF

ON ENTRY: U9 - TCB address
UD - parameter block address
RS state

ON EXIT: RS state

PRINCIPLES OF OPERATION:

SVC2.2 obtains the value of CTOP+2 and then checks the options specified. For get all requests, SVC2.2 checks the end of the parameter block, subtracts UTOP from CTOP+2 and stores the result in the SIZE field of the parameter block. The old value of UTOP is stored in the register zero slot of the TCB RS save area, the new UTOP is stored in the SPT and SVC2.2 exits to EXECRSCC to set the condition code and exit to the task via TMRSOUT.

If request is for a specific number of bytes, the SIZE field is added to the value of UTOP (and checked), the result is rounded up to a fullword boundary, the old value of UTOP is stored in the register zero slot of the TCB RS save area, the new value of UTOP is stored in the SPT and SVC2.2 exits to EXECRSCC to set the condition code and exit to the task via TMRSOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: SVC 2 CODE 3 RELEASE STORAGE (SVC2.3)

ABSTRACT: SVC2.3 processes release storage calls by modifying UTOP by the amount requested

ENTRYS: SVC2.3

SOURCE LIBRARY ROUTINES: SGN, SPT, TCB

EXTRN: TMNSOUT

REGISTERS USED: E9-EB, ED-EF

ON ENTRY:
EE, EF - PSW at time of SVC
ED - parameter block address
E9 - TCB address
NS state

ON EXIT:
NS state

PRINCIPLES OF OPERATION:

SVC2.3 calculates the new UTOP by subtracting the specified size from UTOP. If the new value of UTOP is greater than UBOT, SVC2.3 rounds it up to a fullword boundary, stores the value in the SPT, sets the condition code in register 14 and exits to the task via TMNSOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: SVC 2 CODE 4 SET STATUS (SVC2.4)

ABSTRACT: SVC2.4 provides the user task with the capability to modify the condition code and arithmetic fault enable bits in the PSW.

ENTRYS: SVC2.4

SOURCE LIBRARY ROUTINES: SGN, TCB

EXTRN: TMNSOUT

REGISTERS USED: E9-EB, ED-EF

ON ENTRY:
EE,EF - PSW at time of SVC
ED - parameter block address
E9 - TCB address
NS state

ON EXIT:
NS state

PRINCIPLES OF OPERATION:

SVC2,4 tests the options and if AF modify bit is reset, it skips AF bit processing. Otherwise, the AF bit in register 14 is set or reset to the value specified. SVC2.4 then stores the specified condition code into register 14 and exits via TMNSOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: SVC 2 CODE 5 FETCH POINTER (SVC2.5)

ABSTRACT: SVC2.5 returns the address of the memory management fields in the SPT to the user.

ENTRYS: SVC2.5

SOURCE LIBRARY ROUTINES: SGN, SPT, TCB

EXTRN: TMRSOUT

REGISTERS USED: U9-UB, UD-UF

ON ENTRY:
UD - parameter block address
U9 - TCB address
RS state

ON EXIT: RS state

PRINCIPLES OF OPERATION:

If calling task is in halfword mode, SVC2.5 copies the memory management values from the SPT into an internal table which is DOS compatible and returns a pointer to this table. Otherwise SVC2.5 stores the address of SPT.CTOP into the designated register slot in the RS save area of the TCB and returns via TMRSOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: SVC 2 CODE 6 UNPACK (SVC2.6)

ABSTRACT: SVC2.6 uses the values passed in the parameter block to call UNPACK to convert a binary number to decimal or hexadecimal ASCII.

ENTRYS: SVC2.6

SOURCE LIBRARY ROUTINES: SGN, TCB

EXTRN: ADCHK, MEMFLTRS, TMRSOUT, UNPACK

REGISTERS USED: U6, U8-UF

ON ENTRY: UD - parameter block address
U9 - TCB address
RS state

ON EXIT: RS state

PRINCIPLES OF OPERATION:

SVC2.6 checks the destination start and end addresses for validity and loads the options, length, start of destination into registers from the parameter block and the binary number from the Register 0 slot in the RS save area of the TCB. After calling UNPACK to do the work, SVC2.6 exits via TMRSOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: SVC 2 CODE 7 LOG MESSAGE (SVC2.7)

ABSTRACT: SVC2.7 calls EXECMSG to schedule a log message for the calling task.

ENTRYS: SVC2.7

SOURCE LIBRARY ROUTINES: SGN, SPT, TCB

EXTRN: TMRSOUT, EXECMSG, EXECMSG1, LGMBUFF, ISHRS, ADCHK, MEMFLTRS

REGISTERS USED: U1-UA, UD

ON ENTRY: UD - parameter block address
U9 - TCB address
RS state

ON EXIT: RS state

PRINCIPLES OF OPERATION:

SVC2.7 picks up the address of the text and calls EXECMSG to connect to the dummy leaf. On return, the message text is moved to the system log message buffer. SVC2.7 then sets up the function code with image or format as specified and exits to EXECMSG1 which calls the dummy driver which exits to the task via TMRSOUT. Subsequent calls to SVC2.7 before the message is completed by the Command Processor cause the task to enter connection wait on the call to EXECMSG.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: SVC 2 CODE 15 PACK (SVC2.15)

ABSTRACT: SVC2.15 packs a string of decimal or hexadecimal ASCII characters into a binary number.

ENTRYS: SVC2.15

SOURCE LIBRARY ROUTINES: TCB

EXTRN: SV2ADCHK, EXECRSCC

REGISTERS USED: U0-U3, U5-U6, U8-UA, UD, UF

ON ENTRY:
UD - parameter block address
U9 - TCB address
RS state

ON EXIT:
UF - condition code
U9 - TCB address
RS state

PRINCIPLES OF OPERATION:

SVC2.15 calls SV2ADCHK to check the validity of the input string, initializes the work registers and skips leading blanks in the input string, if specified. The number is then processed from left to right by multiplying the accumulator by the base (10 or 16) and adding the next byte after stripping the zone (and converting if hexadecimal A-F). If the value exceeds 8 digits (hex) or $2^{31}-1$ (dec) the 'V' bit is set in the condition code. If no characters were converted, the 'L' bit is set. On the first byte that is not 0-9 (dec) or 0-9,A-F (hex), SVC2.15 updates the input string pointer by adding the index to the specified register slot in the RS save area of the TCB and exits to EXECRSCC to set the condition code and return to the task via TMRSOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: SVC 2 CODE 16 PACK FILE DESCRIPTOR (SVC2.16)

ABSTRACT: SVC2.16 converts an ASCII string containing a valid file descriptor into the form necessary in an SVC 7 parameter block.

ENTRYS: SVC2.16

SOURCE LIBRARY ROUTINES: SPT, TCB

EXTRN: SV2ADCHK, SV2ADC1, MEMFLTRS, EXECRSCC

REGISTERS USED: U0-UF

ON ENTRY: UD - parameter block address
U9 - TCB address
RS state

ON EXIT: RS state

PRINCIPLES OF OPERATION:

SVC2.16 calls SV2ADCHK to check the validity of the target start and end and the input start. SVC2.16 then uses an internal subroutine PFDGET to extract the fields of the file descriptor from the input string.

PFDGET4 returns up to 4 alphanumeric characters, starting with an alphabetic, left justified and right filled with blanks, in register 11. The input pointer is updated to point past the last character loaded into 11. If less than 4 characters were processed the condition code is non-zero.

PFDGET4A returns 1-4 alphanumeric characters as specified by register 12.

PFDGET4E decrements register 12 on entry and so returns 1-3 alphanumeric characters in register 11.

After processing the volume, name and extension fields, SVC2.16 checks for syntax errors, updates the input and destination pointers in the specified register slots in the RS save area of the TCB and exits to EXECRSCC to set the condition code and return to the task via TMRSOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: SV2ADCHK

ABSTRACT: SV2ADCHK performs common address checking for SVC 2 executors

ENTRYS: SV2ADCHK, SV2ADC1

SOURCE LIBRARY ROUTINES: TCB

EXTRN: ADCHK, MEMFLTRS

REGISTERS USED: U0, U1, U8, U9, UA

ON ENTRY:
U9 - TCB address
U1 - register number of register containing address
U0 - return
RS state

ON EXIT:
UA - checked address
U1 - offset of register in RS save area

PRINCIPLES OF OPERATION:

SV2ADCHK loads the address to be checked from the specified slot in the RS save area of the TCB, calls ADCHK to perform the validity checking and returns.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: SVC 2 CODE 17 Mnemonic Table Scan (SVC2.17)

ABSTRACT: SVC2.17 scans the input string for a match against the specified mnemonic table and returns the index of the mnemonic which matches.

ENTRYS: SVC2.17

SOURCE LIBRARY ROUTINES: TCB

EXTRN: SV2ADCHK, ADCHK, MEMFLTRS, EXECRSCC

REGISTERS USED: U0-U1, U5-UF

ON ENTRY:
UD - parameter block address
U9 - TCB address
RS state

ON EXIT: RS state

PRINCIPLES OF OPERATION:

SVC2.17 loads the input address from the specified register in the RS save area, loads the mnemonic table address and scans the input string for the first non-alphabetic following the first character. This yields the length of the input mnemonic. SVC2.17 then compares the input mnemonic with the first mnemonic in the table character by character, stripping the high order bit of each table byte. If an unequal compare results before the byte of X'00' in the table, SVC2.17 steps to the next mnemonic in the table and starts the compare again. If all the characters of the input mnemonic match a table entry, SVC2.17 checks the high order bit of the next table byte. If set, no match is present and SVC2.17 skips to the next table entry; if reset, SVC2.17 updates the input pointer in the specified register slot in the RS save area and exits to EXECRSCC to set the condition code and return control to the task via TMRSOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: SVC 2 CODE 18 MOVE ASCII CHARACTERS (SVC2.18)

ABSTRACT: SVC2.18 moves a string of bytes, ending on a specified set of characters or when a specified number of bytes have been moved.

ENTRYS: SVC2.18

SOURCE LIBRARY ROUTINES: TCB

EXTRN: SV2ADCHK, ADCHK, MEMFLTRS, EXERSCC

REGISTERS USED: U0-U3, U5-UF

ON ENTRY: U0 - parameter block address
U9 - TCB address
RS state

ON EXIT: RS state

PRINCIPLES OF OPERATION:

SVC2.18 loads the start of the input, destination and ending character string if present.

If ending character processing is specified U3 is loaded with the length of the ending character string. Each byte of the input string is then compared with each ending character. If there is no match or no ending character string (U3=0) the byte is moved to the next byte in the destination string. This process is continued until an input character matches an ending character or the input counter goes to ZERO. SV2.18 then updates the input and destination pointers in the specified register slots in the RS save area and exits to EXECRSCC to set the condition code and return control to the task via TMRSOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

NAME: SVC 2 CODE 19 PEEK (SVC2.19)

ABSTRACT: SVC2.19 returns the OS ID, number of LUs, task options and wait status in the SVC 2 code 19 parameter block.

ENTRYS: SVC 2

SOURCE LIBRARY ROUTINES: SPT, TCB

EXTRN: ADCHK, MEMFAULT, TMNSOUT

REGISTERS USED: E8-EF

ON ENTRY: ED - parameter block address
E9 - TCB address
NS state

ON EXIT: NS state

PRINCIPLES OF OPERATION:

SVC2.19 checks the validity of the end of the parameter block and sets up the parameter block from the SPT and TCB and exits via TMNSOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: SVC 2 CODE 20/21, EXPAND/CONTRACT ALLOCATION (SVC2.20/SVC2.21)

ABSTRACT: SVC 2 code 20/21 processes memory allocation expand/contract calls by modifying CTOP as requested.

ENTRYS: SVC2.20, SVC2.21

SOURCE LIBRARY ROUTINES: SET

EXTRN: TMNSOUT, ADCHK, MEMFAULT

REGISTERS USED: E8-EF

ON ENTRY: ED - parameter block address
E9 - TCB address
NS state

ON EXIT: NS state

PRINCIPLES OF OPERATION:

SVC2.20/21 loads the parameter block into register 10 and converts the number of blocks specified into number of bytes. Expand-all requests are processed by subtracting CTOP from FBOT, rounding the result down to the nearest 256 byte boundary and storing the number blocks in the parameter block. Other expand requests and contract requests are processed by adjusting CTOP by the specified amount. CTOP is updated in the SPT, the condition code is set by modifying register 14 and control is returned to the task via TMNSOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: SVC 3 END OF TASK,(SVC3)

ABSTRACT: SVC 3 performs end of task processing by halting all outstanding reads, check-pointing all LUs, issuing the End of Task message and resetting the TCB.

ENTRYS: SVC3

SOURCE LIBRARY ROUTINES: SPT, TCB, EVT, IVT, DCB, FCB

EXTRN: TIMEOUT, TMRSIN1, EXECMSG, EXECMSG2, TMDISP

REGISTERS USED: E8-EF, U0-UA, UD-UF

ON ENTRY: ED - return code
E9 - TCB address
NS state

ON EXIT: NSU state

PRINCIPLES OF OPERATION:

SVC3 goes down the connected leaf chain and calls TIMEOUT to halt all reads outstanding. RSA state is entered by calling TMRSIN and then setting the alternate save area bit. This is to allow SVC 7 calls from SVC3. SVC3 then issues an SVC 7 checkpoint for each LU, using the area TCB.SYS for a temporary parameter block. On return from SVC 7, the alternate save area status bit is reset. SVC3 then prepares the status field with RS, I/O wait pending and ready chain bits set and enters NSU state. SVC3 prepares the EOT message pointers and function code. EXECMSG connects to the dummy leaf and returns. SVC3 sets cancel pending bit to allow the command processor to issue the correct command prompt. SVC3 then issues the EOT message as a subroutine of the ending task via EXECMSG2. Since I/O wait pending is set and the RS PSW save area has been set up with an NSU PSW pointing to EOJME, on completion of the message, SVC3 regains control, unchains the TCB, sets the dormant wait bit, zeroes the TCB except for ID, NLU, PRI, DPRI, and LTAB and exits to TMDISP.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: SVC 5 FETCH OVERLAY (SVC5)

ABSTRACT: SVC5 processes load overlay requests by passing the load information to the command processor, waiting for completion and passing status back to the calling task.

ENTRYS: SVC5

SOURCE LIBRARY ROUTINES: SPT, TCB

EXTRN: CMDLR, TCBTAB, LOADSTAT, TMRSIN1, TMDISP, TMNSOUT, TMRSOUT, INITCMD$

REGISTERS USED: EA-EF, UC-UD

ON ENTRY: ED - parameter block address
E9 - TCB address
NS state

ON EXIT: RS state

PRINCIPLES OF OPERATION:

On entry SVC5 checks the validity of the options and LU specified. SVC5 stores the option byte into CMDLR in the command processor and copies the attributes and DCB/FCB address from the specified LU of the user TCB into LU1 of the system TCB. It then prepares to wait for completion of the load by:

1. Storing the address of the Register 12 slot of the user TCB dispatch save area in LOADSTAT in the command processor.
2. Setting load wait in user TCB.
3. Unchaining the user TCB.
4. Calling TMRSIN to save the user registers and resume PSW in the user TCB RS save area.
5. Saving the parameter block address in the Register 13 slot of the user TCB dispatch save area.
6. Storing a PSW with RS status and location of RSVC5 into the PSW save slot of the user TCB dispatch save area.

SVC5 then enters the dummy driver at the secondary entry point INITCMD$ to complete the SVC1 I/O & wait to the console and schedule the system task. The command processor detects t e load overlay request in CMDLR, performs the load, stores the status in the register 12 slot of the user TCB dispatch save area (pointed to by LOADSTAT) and calls TMREMW to remove the load wait and chain the user TCB. When the user task becomes top of ready chain, TMRDISP dispatches the user task by loading the PSW and registers from the dispatch save area thus passing control to SVC5 at RSVC5. The status is converted in Register 12, stored in the parameter block and SVC5 exits to the task via TMRSOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: TMDISP

ABSTRACT: TMDISP compares the priority of the current task with top of EVT and enters TMRDISP or EVTDISP accordingly.

ENTRYS: TMDISP

SOURCE LIBRARY ROUTINES: SPT, TCB, EVT

EXTRN: TMRDISP, EVTDISP

REGISTERS USED: E9-EB

ON ENTRY: NS state

ON EXIT: NS state

PRINCIPLES OF OPERATION:

TMDISP compares the priority of the task at top of ready chain (SPT.CTCB) with the priority queued to the system node (EVN.QPRI). If CTCB = 0, (no ready task) the ready chain priority is set to X'FE'. TMDISP enters TMRDISP or EVTDISP as follows:

TMRDISP:
1. EVN.QPRI of system node = X'FF'
2. priority of CTCB greater than EVN.QPRI of system node

EVTDISP:
1. EVN.QPRI greater than or equal to priority of CTCB
2. EVN.QPRI not X'FF' and CTCB = 0.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: TMRDISP (TMNSOUT)

ABSTRACT: TMRDISP dispatches from SPT.CTCB (top of ready chain) or puts system into wait state. TMRDISP processes pause pending and issues the task paused message if set.

ENTRYS: TMRDISP, TMNSOUT

SOURCE LIBRARY ROUTINES: SPT, TCB, SGN

EXTRN: TMRSIN1, EXECMSG, EXECMSG1, MMWPSW, JOURNAL

REGISTERS USED: E8-EF, U0-U4, U7-U9

ON ENTRY: TMNSOUT requires EE, EF to have the resume PSW
U0-UF to have valid user register values
NS state

ON EXIT: state defined in EE-EF or dispatch save area

PRINCIPLES OF OPERATION:

TMRDISP enters NS state via an EPSR and loads SPT.CTCB. If ZERO an enabled wait PSW is loaded to put the system into a wait state. If non-zero, TMRDISP loads the user register set from the dispatch save area in NSU state, reenters NS state and loads the dispatch save area PSW into EE-EF. This state is equivalent to exit from a type I SVC (TMNSOUT).

TMRDISP then checks for pause pending. If pause pending is not set or the task is being dispatched into RS, RSA or ES state, a journal entry is made and the task dispatched by doing an LPSWR on EE, EF. If pause pending is set and the task is being dispatched into UT/ET state, RS state is entered via TMRSIN1, and EXECMSG is called to schedule the task paused message. On exit from the dummy driver, TMRSOUT places the task into console wait.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

NAME: EVTDISP

ABSTRACT: EVTDISP connects the task queued to the top of the EVT and places it on the ready chain

ENTRYS: EVTDISP

SOURCE LIBRARY ROUTINES: SPT, EVT, TCB

EXTRN: TCBTAB, EVHOOK2, TMREMW, TMDISP, TMRDISP, TMRSNIN

REGISTERS USED: E8-EF

ON ENTRY: NS state

ON EXIT: NS state

PRINCIPLES OF OPERATION:

EVTDISP processes tasks queued for one of two reasons: connection wait and assertion. If the highest queue task is in connection wait, EVTDISP walks down the EVT to the highest queued leaf and removes the task at the top of the connection queue for that leaf. The task is connected to the leaf by placing the DCB and ESR address from the register 13 and register 14 slots of the task's TCB dispatch save area into the leaf along with the task's ID and dispatch priority. The leaf is added to the task connected leaf chain. EVTDISP calls EVHOOK to connect all ancestor nodes and then calls TMREMW to remove the connection wait and chain the TCB. On return, if CTCB was not changed TMDISP is entered to see if EVHOOK propagated a new task to the top of the EVT. If the current task was changed the new task is dispatched via TMRDISP.

If the highest queued task is asserting reconnection, EVTDISP encounters an EVT entry with the assert flag set while walking down to the highest queued leaf. In this case, EVTDISP resets the assert flag, loads the connected leaf pointer, if the leaf is not the asserting entry, resets the pending flag in the connected leaf, calls EVHOOK to connect from the asserting entry up and branches to TMRSNIN to schedule the ESR.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: TMSTART

ABSTRACT: TMSTART prepares the user TCB for dispatching as a result of a START command.

ENTRYS: TMSTART

SOURCE LIBRARY ROUTINES: SGN, SPT, TCB

EXTRN: TMREMW, TCBTAB

REGISTERS USED: U9-UF

ON ENTRY UF - starting location
U9 - TCB ID
ET state

ON EXIT ET state

PRINCIPLES OF OPERATION:

TMSTART derives the TCB address; establishes the starting PSW status from the TCB options in register 14 and stores registers 14, 15 in the PSW of the TCB dispatch save area. TMSTART then calls TMREMW to remove all wait bits and to chain the user TCB. TMREMW returns to the command processor via register 8.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

NAME: TMSTOP

ABSTRACT: TMSTOP saves the state of the current task in the TCB dispatch save area.

ENTRYS: TMSTOP

SOURCE LIBRARY ROUTINES: SGN, SPT, TCB

EXTRN: TCBTAB

REGISTERS USED. E8-EF, U0-UF

ON ENTRY. EE,EF - PSW to be saved in dispatch save area
E9 - TCB ID
E8 - return address
NS state

ON EXIT: NS state
E9 - TCB address

PRINCIPLES OF OPERATION:

TMSTOP stores the resume PSW in the TCB.DPSW, enters NSU state, saves U0-UF in TCB.DGPR, reenters NS state and exits via Register 8.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: TMRSIN (TMRSAIN)

ABSTRACT: TMRSIN saves the user register set and resume PSW in the specified save area and exits via the PSW in EA-EB.

ENTRYS: TMRSIN, TMRSIN1, TMRSAIN

SOURCE LIBRARY ROUTINES: SGN, SPT, TCB

EXTRN: TCBTAB, JOURNAL

REGISTERS USED: E8-EF, U0-UF

ON ENTRY: EE,EF - PSW to be saved in RS save area
EA-EB - return PSW
E9 - TCB address (TMRSIN1, TMRSAIN)
NS state

ON EXIT: U9 - TCB address
UA-UF - values passed in EA-EF
state specified by EA-EB

PRINCIPLES OF OPERATION:

TMRSIN, TMRSAIN make separate journal entries, save E9-EF in the SPT Task Manager save areas and load a pointer to TCB.RGPR or the alternate save area pointed to by TCB.ASV, respectively. TMRSIN calculates the address of the current TCB; TMRSIN1 requires the TCB address in E9. Common processing enters NSU state, saves U0-UF in the specified save area, loads the saved values of EA-EF into UA-UF and exits via the PSW in UA, UB.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: TMRSOUT (TMRSAOUT, TMRSNOUT)

ABSTRACT: TMRSOUT loads the task environment from the specified save area. Pause pending, I/O wait pending and wait bits are processed.

ENTRYS: TMRSOUT, TMRSOUT1, TMRSAOUT, TMRSNOUT, TMRSAOU1

SOURCE LIBRARY ROUTINES: SGN, TCB, SPT

EXTRN: JOURNAL, TMRDISP, TMUCHN, TMDISP, TCBTAB

REGISTERS USED: U0-UF

ON ENTRY: SPT.CTCB - TCB ID (TMRSOUT)
U9 - TCB ID (TMRSOUT1, TMRSAOUT, TMRSAOU1, TMRSNOUT)
RS, RSA, ES state (TMRSOUT/TMRSOUT1, TMRSAOUT/TMRSAOU1, TMRSNOUT)
UE - UF - resume PSW (TMRSOUT1, TMRSAOU1)

ON EXIT: state in specified save area

PRINCIPLES OF OPERATION:

This module can be regarded as several functional submodules:

RSOUT: RSOUT sets dispatch flag (UF=0); RSOUT1 enters with dispatch flag reset (UF≠0). The TCB RS status bit is reset unless the alternate save area bit is set, EC is loaded with a pointer to the TCB RS save area, and a journal entry is made. TESTF is entered.

RSAOUT: RSAOUT sets the dispatch flag (UF=0); RSAOU1 enters with dispatch flag reset. The TCB RS and alternate save area bits are reset, EC is loaded with a pointer to the alternate save area (TCB.ASV), a journal entry is made and TESTF is entered.

RSNOUT: RSNOUT sets the dispatch flag, resets the ES status bit, makes a journal entry. If any wait bits are set WAIT is entered. If not, the task is rechained (no-op in ST), if necessary, by switching PRI and DPRI, calling TMCHN and setting PRI to DPRI on return. If task is still current (always in ST) TESTRE is entered, if not TDISP is entered to dispatch the new current task.

TESTF: TESTF tests the dispatch flag and if set (UF=0) enters WAIT if any wait bits are set or TESTRE if no waits are set; if reset (UF≠0) TESTF saves the PSW in UE and UF in SPT.PSV and enters MOVE.

TESTRE: TESTRE checks if task is about to enter UT/ET level. If not, DISP is entered to dispatch from the save area. If so the following checks are made:
1) Except for exit to RSA state (AS bit still set in status) pause pending status causes a PSW with NS states and a location of TMRDISP to be saved in SPT.PSV and MOVE is entered.
2) Except for exit from ES state (EC contains address TCB.EPSW) I/O wait pending causes the pending bit to be reset, I/O wait to be set, and WAIT to be entered.

WAIT: WAIT unchains the TCB and enters TDISP.

MOVE: MOVE moves the PSW and registers from the specified save area (EC) to the TCB dispatch save area and exits via the PSW in SPT.PSV.

DISP: DISP saves the PSW from the specified save area (EC) in SPT.PSV, loads the user registers from the save area and exits via the PSW in SPT.PSV.

TDISP: Saves a PSW with NS status and a location of TMDISP in SPT.PSV and enters MOVE.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

NAME: TMRSNIN (ESRGO)

ABSTRACT: TMRSNIN schedules an ESR in ES state for the current task. The interrupted environment of the task is saved in the TCB ES save area.

ENTRYS: TMRSNIN, TMRSNIN1, ESRGO

SOURCE LIBRARY ROUTINES: SGN, SPT, TCB, EVT

EXTRN: TCBTAB, JOURNAL, TMCHN

REGISTERS USED: E8-EF, U0-UF

ON ENTRY:

TMRSNIN:
- EF - leaf address
- EA - ES priority
- E9 - TCB address (TMRSNIN1)
- NS state

ESRGO:
- UF - leaf address
- UD - DCB address
- U7 - TCB address
- NSU state
- ES bit set in TCB status

ON EXIT:
- ES state
- UF - leaf address
- UD - DCB address

PRINCIPLES OF OPERATION:

TMRSNIN derives the TCB address, sets the task DPRI to the specified ES priority and calls TMCHN to put the task in the proper place in the ready chain (no-op in ST). It then enters NSU state, moves the dispatch save area contents to the ES save area, sets the ES status bit, loads UD with the DCB address from the leaf and enters ESRGO.

ESRGO resets the pending flag in the leaf, walks up the tree to the highest connected ancestor (CLEV), resets assert in the highest connected ancestor, makes a journal entry, decrements the event count in the leaf and TCB and exits to the ESR routine pointed to by the leaf in ES state.

B126

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: TMREMW (TMCHN)

ABSTRACT: TMREMW resets the specified wait bits. If no wait bits remain set, TMCHN puts the task on the ready chain before the first task with lower priority.

ENTRYS: TMREMW, TMCHN

SOURCE LIBRARY ROUTINES: SGN, SPT, TCB

EXTRN: TMUCHNS, TCBTAB

REGISTERS USED: R8-RB

ON ENTRY: RD - bits 16-31 contain the wait bits to be reset (TMREMW)
R9 - TCB ID
R8 - return address
any state except IS

ON EXIT: entry state

PRINCIPLES OF OPERATION:

On entry, TMREMW and TMCHN set a flag to indicate which entry, save the entry state, and enter NS state (if state on entry is NS) or NSU (if entry state was NSU, RS, ET, RSA). TMREMW calls cause the specified bits to be reset in the TCB wait field (TCB.WAIT). If any bits remain set, TMREMW returns via the saved state and Register 8.

TMCHN saves the value of SPT.CTCB and checks the ready chain bit; if set and DPRI=PRI, TMCHN returns via the saved state and R8. If the ready chain bit is set and DPRI ≠ PRI, TMUCHNS is called to unchain the TCB from its present position and on return, the TCB is chained before the first task on the chain whose DPRI is less than the DPRI of the task being chained. The value of SPT.CTCB is compared with the value on entry. If the same, the condition code in the saved entry status is set to 0; otherwise, it is set to X'F' and TMCHN exits via the saved state and R8.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: TMUCHN

ABSTRACT: TMUCHN removes a TCB from the ready chain.

ENTRYS: TMUCHN, TMUCHNS

SOURCE LIBRARY ROUTINES: SPT, TCB, SGN

EXTRN: TCBTAB

REGISTERS USED: R8-RB

ON ENTRY: R9 - TCB ID (TMUCHN) TCB address (TMUCHNS)
R8 - return address
any state except IS (TMUCHN) NS/NSU (TMUCHNS)

ON EXIT: entry state

PRINCIPLES OF OPERATION:

TMUCH saves the entry state and enters NS (if NS entry) or NSU (all other entries). TMUCHN then searches the ready chain for the specified TCB and removes it. The value of CTCB is compared with the CTCB value on entry. If equal, the saved status is modified to have a condition code of zero; if unequal, it is modified to have a condition code of X'F'. Exit is via the saved status and R8.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

NAME: EVCON (EVQCON)

ABSTRACT: EVCON connects a task to a leaf and its ancestor nodes in the EVT. If connectien cannot be made to all entries in the specified path, a condition code of X'F' is returned (EVCON) or the task is placed in connection wait.

ENTRYS: EVCON, EVQCON

SOURCE LIBRARY ROUTINES: SGN, EVT, SPT, TCB

EXTRN: TCBTAB, EVHOOK2,TMSTOP, TMUCHN, TMDISP, EVPROP1

REGISTERS USED: U8-UF, E8-EA, EE-EF

ON ENTRY:
UF - Leaf address
UE - event service routine pointer
UD - DCB address
SPT.CTCB - TCB ID
RS STATE

ON EXIT:
UF - leaf address
UE - ESR pointer
UD - DCB address
RS STATE

PRINCIPLES OF OPERATION:

On entry, EVCON enters NSU state. EVCON walks up the EVT from the leaf and checks for a blocked node (CPRI ≠ FF). If none are found, the task is connected to the leaf by placing the TCB ID, DCB address, ESR address and task priority into the leaf. The leaf is added to the task connected leaf chain, EVHOOK is called to connect any upper nodes and EVCON exits by loading a PSW with RS status and a location of the return address.

If any node in the path is blocked, EVCON entries cause return to the task in RS state with the 'G' bit set in the condition code. If entry is to EVQCON, the task is placed in connection wait by calling TMSTOP to save the values of register set X'F' in the TCB dispatch save area, saving a resume PSW to return to caller in RS state in the dispatch save area, calling TMUCHN to take the TCB off the ready chain. The task is placed on the leafs connection queue in priority order. If the task is not at the head of the queue EVCON exits to TMDISP to schedule the next task. If the task is the new head of the queue, EVQCON calls EVPROP to propagate the new queued priority up the EVT. EVPROP exits to TMDISP.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: EVHOOK

ABSTRACT: EVHOOK completes connection to a path in the EVT from the specified level up to system node.

ENTRYS: EVHOOK, EVHOOK2, EVHOOK1

SOURCE LIBRARY ROUTINES: SPT, EVT

EXTRN: None

REGISTERS USED: R8-RB (EVHOOK2), R8-RE (EVHOOK1)

ON ENTRY: R8 - starting node address
RA - leaf address
R9 - connection priority
NS/NSU state

ON EXIT: entry state

PRINCIPLES OF OPERATION:

EVHOOK saves RC-RF in the SPT and connects the ancestor of the specified node by stepping up one level and storing the specified leaf address and connection priority in EVN.LEAF and EVN.CPRI. The descendant pointer of the entry just stepped up from is given a priority of X'FF' to prevent the descendent from being queued (since it is connected). EVHOOK then scans the descendent list and places the highest descendent number and its priority in the node's queue. This process is continued until system node is reached. The leaf is not connected to system node.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: EVMOD

ABSTRACT: EVMOD changes the ESR address in the specified connected leaf.

ENTRYS: EVMOD

SOURCE LIBRARY ROUTINES: SGN, EVT, SPT

EXTRN: None

REGISTERS USED: R8, R9, RE, RF

ON ENTRY:
RF - leaf address
RE - ESR address
R8 - return
any state

ON EXIT:
RF - leaf address
RE - ESR address
entry state

PRINCIPLES OF OPERATION:

EVMOD stores the ESR address into the specified leaf.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: IODONE

ABSTRACT: IODONE performs common post processing of SVC 1 requests.

ENTRYS: IODONE, IODONE2

SOURCE LIBRARY ROUTINES: SPT, DCB, SVC1., TCB, SGN

EXTRN: EVRTE, TMRSOUT, EVDIS, TCBTAB, TMREMW

REGISTERS USED: U0, U1, U2, U7-UA, UD-UF

ON ENTRY: UF - leaf address
UD - DCB address or FCB address
ES (IODONE) or RS (IODONE2) state

ON EXIT: UF - leaf address
entry state

PRINCIPLES OF OPERATION:

IODONE sets the exit routine to EVRTE; IODONE2 sets it to TMRSOUT. The parameter block address is loaded from the DCB and the device number is used to reset the ISP table entry for the device to III. The status is moved from the DCB into the parameter block. EVDIS is called to disconnect from the leaf. The function code and length of last transfer fields are loaded from the DCB before disconnection since they may not be valid on return. If the function is a command, IODONE exits. If it is a data transfer, the length of last transfer is stored in the parameter block and if it is an I/O and proceed call IODONE exits. For an I/O and wait, the I/O wait pending bit is reset. If it had been set, IODONE exits since this means the I/O completed before the task entered I/O wait state. If it had been reset, IODONE calls TMREMW to remove the I/O wait condition and to exit to EVRTE or TMRSOUT.

IODONE skips parameter block processing if the parameter block address is zero in the DCB/FCB.

IODONE skips the EVDIS call if the leaf address is zero in the DCB/FCB.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: EVDIS (EVREL)

ABSTRACT: EVDIS disconnects a task from a path in the EVT. EVREL disconnects a task from the upper portion of a path.

ENTRYS: EVDIS, NSEVDIS, EVREL, NSEVREL

SOURCE LIBRARY ROUTINES: SGN, SPT, TCB, EVT

EXTRN: JOURNAL, EVPROP, TMRDISP, TMSTOP, EVTDISP, TCBTAB

REGISTERS USED: R8-RB

ON ENTRY:
RF - leaf address
RE - release level (EVREL, NSEVREL)
R8 - return
any state except IS

ON EXIT:
RF - leaf address
entry state

PRINCIPLES OF OPERATION:

This module consists of three separate sub-modules DIS, REL and DLOOP.

DIS: DIS unconnects the task from the leaf by zeroing out the CTCB and connection level and setting the connection priority to X'FF'. The leaf is removed from the task connected leaf chain. The priority of the task at the top of the leaf's connection queue is stored in EVDPSV and DLOOP is entered.

REL: REL sets the new connection level in the connected leaf and walks up the EVT to the release level, stores a priority of X'FF' in EVDPSV and enters DLOOP.

DLOOP: DLOOP is entered with R8 = starting level number, RA = starting node address, EVDCSV = level to stop at, EVDPSV = highest queued priority of subtree, EVDDS = highest queued descendant #. DLOOP unblocks the node at its current level and insures that the highest priority subtree (descendant) is queued to the node. When the stop level is reached, EVPROP is called to propagate the queued priority at the stop level to the system node. If the priority is new top of EVT, DLOOP checks the priority of the current task. If the current task is higher priority EVDIS returns to caller; if not, or if EVPROP produced a new top of ready chain, the current task is suspended via TMSTOP and EVTDISP is entered to dispatch new top of tree.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: EVRTE

ABSTRACT: EVRTE checks for queued events and schedules one if possible. If no events are queued to the task TMRSNOUT is entered to exit from ES state.

ENTRYS: EVRTE

SOURCE LIBRARY ROUTINES: SPT, TCB, EVT

EXTRN: TCBTAB, TMRSNOUT, ESRGO, EVPROP

REGISTERS USED: U4-UB, UD, UF

ON ENTRY: ES state
SPT.CTCB - TCB ID

ON EXIT: ES state

PRINCIPLES OF OPERATION:

EVRTE checks the TCB event count and if zero, exits to TMRSNOUT. If non-zero, the connected leaf chain is searched for a leaf with a non-zero event count. If a leaf is found with non-zero event count and connection level = tree size, EVRTE exits to ESRGO to schedule the ESR pointed by the leaf.

If CLEV ≠ TSIZE and the pending flag is set then this leaf has already been processed by EVPROP so EVRTE goes to the next leaf. If pending is not set, EVRTE sets it, sets assert in the highest connected ancestor and calls EVPROP to propagate the priority up from connection level before going on to the next leaf.

After all leaves have been processed, if no queued ESR can be scheduled, TMRSNOUT is entered to exit from ES state.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

NAME: EVPROP

ABSTRACT: EVPROP is called to propagate a priority up the EVT and to queue descendant subtrees to entries in the EVT.

ENTRYS: EVPROP, EVPROP1

SOURCE LIBRARY ROUTINES: EVT, TCB

EXTRN: TCBTAB, TMCHN

REGISTERS USED: R8-RB

ON ENTRY:
RB - propagation priority
RA - starting node
NS/NSU state

ON EXIT:
RB - propagation priority
CC = 0 normal exit
CC = 1 top of ready chain changed
CC = 8 priority propagated to top of EVT
NS/NSU state

PRINCIPLES OF OPERATION:

EVPROP performs two basic functions: propagating a priority up a path in the EVT until a node with a higher connected priority is encountered and queuing a task to each higher entry in the EVT until a blocked node is encountered or a node with a higher queued priority is enccuntered.

EVPROP checks the propagating priority against the connected priority (CPRI) at each level. If the CPRI is not X'FF' then the node is blocked and if a higher priority than the propagating priority EVPROP returns with CC=0 or 1. If CPRI is lower, EVPROP replaces it with the propagation priority and if the task that is connected to that node is executing an ESR, EVPROP rechains the task at the new priority (no-op in ST). If the new priority places the task at the top of ready chain EVPROP returns with CC=1. If not, EVPROP attempts to continue propagating until a higher priority is encountered.

If the node is unblocked, EVPROP sets the current descendant (the one just stepped up from) as the node's highest queued descendant unless the propagating priority is lower than the already queued priority. In the latter case, this higher queued priority replaces the propagating priority and EVPROP attempts to queue and propagate the highest queued descendant of the node.

If at any unblocked node, the propagating priority is lower than the queued priority but the descendant numbers are the same, EVPROP forces the lower priority. This allows a task to be unqueued from the EVT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: CANEOJ

ABSTRACT: CANEOJ executes as an ET subroutine of the Command Processor and is the CANCEL command executor.

ENTRYS: CANEOJ

SOURCE LIBRARY ROUTINES: TCB, EVT, DCB

EXTRN: TCBTAB, TMRSOUT, TMREMW, TIMEOUT

REGISTERS USED: U8-UF

ON ENTRY: U9 - TCB ID
ET state

ON EXIT: ET state

PRINCIPLES OF OPERATION:

CANEOJ calculates the TCB address and enters NSU state. Each leaf in the connected leaf chain is passed to TIMEOUT to halt all ongoing I/O (except for leaves corresponding to devices which have the uncancellable bit set in the DCB).

TIMEOUT is entered in IS state via an LPSW.

The task is then removed from connection wait, if necessary, by moving the task from the leaf queue and if top of queue, calling EVPROP with a propagating priority of X'FF'. The resume PSW is set to go to TMRSOUT instead of the instruction following the EVQCON call.

The UT/ET level PSW is then found and modified to execute an SVC 3,255 on dispatching into UT/ET level. Pause pending is reset and TMREMW is called to remove connection and console waits if necessary. CANEOJ returns to the Command Processor.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: EXECMSG

ABSTRACT: EXECMSG is called to schedule a message for the executive on behalf of the current task. It also allows an address to be unpacked into the message.

ENTRYS: EXECMSG, EXECMSG1, EXECMSG2

SOURCE LIBRARY ROUTINES: DCB.

EXTRN: DCBCMD, EVQCON, UNPACK

REGISTERS USED: U0-U4, U8-UF, U8

ON ENTRY:
U8 - exit address if U1=0
U4 - message start
U3 - message end
U2 - message function code and unpack options
U1 - unpack dest or 0

ON EXIT:
U0 - value to unpack
RS State

Exit to Dummy Driver

PRINCIPLES OF OPERATION:

EXECMSG connects to the dummy leaf. If U1=0, EVQCON returns to caller to allow processing. If U1≠0, UNPACK is called to unpack the value in U0 into the message. EXECMSG1 entry causes the dummy DCB to be set up pointing to the message and the dummy driver is entered. The dummy driver exits via TMRSOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: EXECRSCC

ABSTRACT: EXECRSCC sets a condition code in the RS resume PSW and exits to TMRSOUT.

ENTRYS: EXECRSCC

SOURCE LIBRARY ROUTINES: TCB

EXTRN: TMRSOUT

REGISTERS USED: U2, UE, UF

ON ENTRY: UF - desired condition code
U9 - TCB address
RS state

ON EXIT: RS state

PRINCIPLES OF OPERATION:

EXECRSCC loads the status portion of the RS resume PSW, zeroes the condition code, ORs in the condition code passed in UF, stores the updated status back in the RS PSW and exits.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: UNPACK

ABSTRACT: UNPACK is called to unpack a binary number to decimal or hexadecimal ASCII characters.

ENTRYS: UNPACK

SOURCE LIBRARY ROUTINES: None

EXTRN: None

REGISTERS USED: U8-UD

ON ENTRY:
UD - options and length (same as SVC2.6)
UC - start of destination
UA - binary number
U8 - return address (bit 0 must be 0)
RS State

ON EXIT:
RS State
UD - options
UC - start of destination

PRINCIPLES OF OPERATION:

UNPACK processes the number from least significant digit to most significant digit by repeatedly dividing the argument by 10 or 16 until the length specified has been unpacked.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: JOURNAL

ABSTRACT: JOURNAL makes an entry in the system journal

ENTRYS: JOURNAL

SOURCE LIBRARY ROUTINES: SPT, SGN

EXTRN: None

REGISTERS USED: E8 - EF

ON ENTRY: EC-EF values to be placed in journal
NS/NSU state

ON EXIT: NS/NSU state
EC-EF - same as entry
exit to 2+ contents of register 8.

PRINCIPLES OF OPERATION:

JOURNAL picks up the journal code, ORs in SPT.CTCB, loads the journal address and executes an ABL instruction. If list overflow results, the list is reset to 0 slots used and the ABL is reexecuted. EC-EF are ABL'd to the list and JOURNAL exits.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: TIMEOUT

ABSTRACT: TIMEOUT halts the I/O outstanding to the specified leaf by setting the DCB timeout constant to zero and scheduling the ESR.

ENTRYS: TIMEOUT

SOURCE LIBRARY ROUTINES: EVT, DCB

EXTRN: None

REGISTERS USED: E8-EB, EF

ON ENTRY: EF - leaf address
IS State

ON EXIT: IS State

PRINCIPLES OF OPERATION:

TIMEOUT picks up the DCB address from the leaf and if zero, returns. If non-zero, TIMEOUT loads the DCB timeout constant. If the timeout constant is non-positive then an ESR has already been scheduled and exit is made via E8. If timeout constant is positive, TIMEOUT checks the system queue to see if this leaf is on the top of queue. If so, TIMEOUT replaces it. If no, TIMEOUT replaces the top and ATLs the leaf. The timeout constant is set to zero and TIMEOUT exits via E8.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA equipment and it may not be used for any other purpose without written permission.

NAME: ADCHK

ABSTRACT: ADCHK checks an address to insure it is within UBOT and CTOP+2.

ENTRYS: ADCHK

SOURCE LIBRARY ROUTINES: SPT, TCB

EXTRN: None

REGISTERS USED: R8-RB

ON ENTRY:
RA - address to be checked
R9 - TCB address
R8 - return
any state

ON EXIT:
RB - logical segment # (always 0)
RA - address
R9 - TCB address
CC = 0 means valid; CC = 8 means invalid
entry state

PRINCIPLES OF OPERATION:

ADCHK checks the options and status in the TCB and if the task is an E-task or in RSA state, exits via register 8 with zero condition code and Register 11. If a user task, the address is compared with CTOP+2 and UBOT. If valid, register 11 is set to zero, the condition code is set to zero and ADCHK exits. If the address is outside CTOP+2 and UBOT, register 11 is set to -1, the 'G' and 'L' bits are set in the condition code and ADCHK exits.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: SYSINIT

ABSTRACT: SYSINIT initializes the TCbs, DCBs, EVT and ISPTAB and starts the Command Processor.

ENTRYS: SYSINIT, SYSINITE

SOURCE LIBRARY ROUTINES: SPT, DCB, EVT, TCB

EXTRN: COMMAND, TCBTAB, TIMEOUT, TMCHN, CMDBUFFS

REGISTERS USED: E0-E3, E7-EF

ON ENTRY: Not applicable

ON EXIT: Et State

PRINCIPLES OF OPERATION:

SYSINIT first palces the Processor in a known state with all interrupts masked off. The ISPTAB is then set to contain all entries of III except the first entry which points to the Crash handler. The Display Panel is cleared. FBOT is set to MTOP. The DMT is used to address all DCB's. Each DCB has RCNT, WCNT, RTRY set to zero and TOUT to -1. The EVT is cleared next by unblocking all nodes and leaves and restoring all descendant pointers. The TCBs are then zeroed out except for ID, PRI and NLU. DPRI is set to PRI. The system TCB is chained by storing X'01' in the SPT.CTCB and setting the ready chain bit in TCB status. The system queue is cleared of any entries and the Command Processor is entered in ET state at entry COMMAND.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

NAME: CRSH

ABSTRACT: CRSH handles system crashes by displaying the crash code on the Display Panel and loading a disabled wait PSW.

ENTRYS: CRSEP

SOURCE LIBRARY ROUTINES: SPT

EXTRN: CRSPSW

REGISTERS USED: E0-E6

ON ENTRY: E1 - crash code address
IS State

ON EXIT: wait state
E5 - address of system journal
E6 - last valid journal entry

PRINCIPLES OF OPERATION:

CRSH loads the journal address, calculates the last valid entry, picks up the crash code, stores it in SPT.CRSH, displays the crash code on the Display Panel and loads a PSW with machine malfunction enable and wait status bits set.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: DUMMY DRV.

ABSTRACT: Console Intercept Handler

ENTRYS: TERMCMD, INITCMD, INITCMD$

SOURCE LIBRARY ROUTINES: TCB, DCB

EXTRN: IODONE, CMDPEND, CMDBUFFS, SQ, TMRSOUT

REGISTERS USED: E0-E7

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

INITCMD - Enters IS state. Sets CMDPEND to indicate there is I/O to do. Checks the command read buffer (CMDBUFFS). If empty, just exits. Else it stores a C/R in the buffer, issues a Disable (X'C4') command, and schedules the ESR for the real console.

TERMCMD - resets CMDPEND and exits to IODONE to finish the IO.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: COMMAND

ABSTRACT: This module is entered by SYSINIT; it inits the Command Processor, and exits to JOB to reset the user task.

ENTRYS: COMMAND

SOURCE LIBRARY ROUTINES: IVT, DCB, SVC1

EXTRN: JOB

REGISTERS USED: U1,2,3,5,6,7,C

ON ENTRY:

ON EXIT: Register U1 points to Level Ø buffer

PRINCIPLES OF OPERATION:

COMMAND is entered by SYSINIT. It first sets all Command Processor flags to Ø's. It sets up the SVC1 parameter block for command ready to point to the read buffer. The mnemonic of the console device, (as specified in the IVT) is obtained, and a check is made to see if it is in the CMT. If it is not, SYSTEM CRASH is entered with code Ø. The Command Processor now sets up the DCB to look as if it is assigned ERW, and sets up the LUØ entry in the LU tab to the specified DCB. COMMAND exits to JOB, to initialize the user TCB.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: COMMANDR

ABSTRACT: This module does command line scanning, parsing, and coding. The interaction with the $JOB, $IF, $BUILD is controlled here.

ENTRYS: NOPARM, CMDNEXT, $ENDC, CSSTST, CKDORM, CKDORM3, OUTSTAR, CKPASS

SOURCE LIBRARY ROUTINES: TCB.SPT, SVC1, SVC7

EXTRN:

REGISTERS USED: UØ - UF

ON ENTRY: U1 points to next character in the command line to be examined.

ON EXIT: U1 points to first character to be processed by a command.

PRINCIPLES OF OPERATION:

Scanning for a command is done at CMDNEXT. First blanks are skipped. If the first non-blank is a terminator (carriage return), a new line is read. If it is a semi-colon, then the next non-blank is found. If this is a semi colon, an error condition has been found, and we exit to CMDERROR (code FORM). If the first character of the command (first non-blank, non-terminator) is a "*", then the rest of the line is a comment, and we go to read a new line.

An SVC2,17 is executed to see if the command is in the command table. A journal entry is made indicating this result. Next, the JOBSKIP is tested. If it is set, the current command is tested to see if it was a TERMJOB. If it was a TERMJOB, then it is executed, else it is skipped.

If JOBSKIP is not set, IFSKIP is checked. If it is not set we exit to the executor. If it is set, we check the current command to see if it is an "IF". If so, the IFLEVEL is incremented, the command is skipped, and we go to get another command. If it is not an "IF", we check for an ENDC, and if it is an ENDC, the IFLEVEL is decremented. Otherwise, the command is skipped.

When all commands on a line have been processed, we go to read another line. If the task is dormant, the LU to read is determined. If the LU is LUØ (system console) then an "*" is output and then a line is read. If the task is paused or running the Read is done to LUØ, and if the task is paused, the "*" is printed first. After the line is read, the I/O is checked to see if there was an error. If no error, then we go to do the preprocessing, etc.

If I/O error is detected, we see if the line was being read from the console. If not, then the LU is closed, all CSS files open are closed, we exit to CMDERROR to output an IO error message, and then return to read another line. If the I/O was from the console, we check to see if its a recoverable error. If not, we go to CRASH with code = 2.

If recoverable, we repeat the process by outputting an "*" and forcing a command line to be read.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

NAME: CMDERROR

ABSTRACT: This routine builds and displays error messages

ENTRYS: CMDERROR, ERR.ASGN, ERR.IO

SOURCE LIBRARY ROUTINES: SPTE

EXTRN: SPT.RC, CMDWRITE, CMDERMV, CMDPOSV, CSSCLOSE

REGISTERS USED: UC, U2, U6, U7, U8, U9

ON ENTRY: UC - points to error code call is:
U1 - points to last character processed
U7 - error type if SVC 1 or SVC 7 error

```
BAL UC,CMDERROR
DC  C'XXXX'
```

ON EXIT: Exits to read new command line, registers not meaningful.

PRINCIPLES OF OPERATION:

This sets the task error code to 255, moves in the XXXX-ERR by calling CMDERMV, and the POS = XXXX.... by calling CMDPOSV. The message is displayed to LU2 (log device) by calling CMDWRITE. All CSS levels down to the JOB level are closed, and a new command line is read.

If entry is made to ERR.10 or ERR.ASGN, REG 7 contains error code (in hex). The mnemonic is obtained from a table, and a call to CMDTYMV is made to build the TYPE = XXXX message. If the ASGN error is an IO error two TYPE = XXXX fields are built.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: CONTINUE

ABSTRACT: Resume PAUSEd Task

ENTRYS: CONTINUE

SOURCE LIBRARY ROUTINES: TCB

EXTRN: TMREMW

REGISTERS USED: U5, U1, U8, U9, UD

ON ENTRY:

ON EXIT:
U8 - A(NOPARM)
U9 - 2
UD - TWT.CWM

PRINCIPLES OF OPERATION:

Makes sure the task is paused, and calls TMREMW to put it back on the ready chain.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: START

ABSTRACT: Start a task

ENTRYS: START

SOURCE LIBRARY ROUTINES: SPTE, TCB

EXTRN: SPT.UTCB, CMDERROR, SCANNER, TMSTART

REGISTERS USED.

ON ENTRY.

ON EXIT: U8 = A(CMDEMPTY)
U9 = 2
UF = task start address

PRINCIPLES OF OPERATION:

Checks state of task to make sure it is dormant or paused. Moves the starting options (or a C/R) above UTOP. Gets the starting location specified (or default if none specified) and goes to TMSTART to start the task.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: CANCEL

ABSTRACT: Cancel a task

ENTRYS: CANCEL

SOURCE LIBRARY ROUTINES: TCB

EXTRN: SPT.UTCB, CMDEMPTY, CSSCLOSE

REGISTERS USED:

ON ENTRY:

ON EXIT:
U9 = 2
U8 = A(CMDEMPTY)

PRINCIPLES OF OPERATION:

Close CSS LU's down to the $JOB level, and calls CANEOJ to cancel the task.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: BIAS

ABSTRACT: Sets up the bias for EXAMINE and MODIFY.

ENTRYS: BIAS

SOURCE LIBRARY ROUTINES: SPTE

EXTRN: SPT.MTOP

REGISTERS USED: U1, U0

ON ENTRY: U1 points to current position in command line

ON EXIT: U1 pointing past paramters

PRINCIPLES OF OPERATION:

Does a SVC 2.15 to get the bias value, and stores it at CMDBIAS for later use. If the value is invalid (bad characters, or greater than MTOP), PARMERR is taken as the exit.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: WRITE

ABSTRACT: This does the MODIFY command. It modifies the contents of memory.

ENTRYS: WRITE

SOURCE LIBRARY ROUTINES: SPTE

EXTRN: SPT.MTOP, CMDNEXT

REGISTERS USED: U1, U0, UF, UC

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

It obtains the relative starting location (if invalid exit to PARMERR, if not present, NOPR error). The bias is added to obtain the physical start. Successive halfwords are obtained and stored. If no more data is to be had, exit is made to CMDNEXT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: READ

ABSTRACT: This command displays the contents of memory to the log device/console

ENTRYS: READ

SOURCE LIBRARY ROUTINES: SPT

EXTRN: SPT.MTOP, BLANKBUFF, CMDWRITE, COMMACK

REGISTERS USED: U0, U1, U2, U5, U6, U7, U8, UA, UB, UC, UD, UF

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

The relative starting location is obtained, and relocated to a physical address. A check to see if next parameter is a "," or a "/". If a comma is found then "n" is obtained, multiplied by 2 and added to the starting location to obtain the ending location. If a "/", the relative ending location is obtained, and converted to a physical ending location.

Display lines are built by putting current address (physical), and then 8 halfwords of data/per line. The last line contains only as many halfwords as are necessary to complete the display. Before each line is built BLANKBUF is called to clear the buffer. CMDWRITE is called to write the line to LU2.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: JOB

ABSTRACT: Puts the system into a known state

ENTRYS: RESET

SOURCE LIBRARY ROUTINES: SPTE, IVT, TCB, SVC 7

EXTRN: CLOSSUB, SPT.STCB, SPT.UTCB, SPT.UBOT, SPT.UTOP, SPT.CTOP

REGISTERS USED: U0, U7, U9, UA, UE

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

The CLOSE option is checked for and R9 is used as a flag to indicate its presence. The user task is located, and all his logical units are closed, by calling CLOSSUB. If "CLOSE" was not specified, the IVT is scanned to find if any default assigns are to be done. If so, they are made for the user. Next memory pointers (CTOP, UBOT, etc.) are set back to their initial values.

NOPARM is exited to, to perform the next command.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: OPTIONS

ABSTRACT: Sets/Resets task options

ENTRYS: OPTIONS

SOURCE LIBRARY ROUTINES: TCB

EXTRN: SPT.UTCB

REGISTERS USED: U5, UB, UE, UC

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

The state of the task is checked. It must be Dormant or Paused to change its options. If so, then the line is scanned for each option specified, the appropriate bit is set/reset in the task's options halfword (TCB.OPT).

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: SET

ABSTRACT: Finds appropriate executor, and exits to it

ENTRYS: SET

SOURCE LIBRARY ROUTINES:

EXTRN: MNMFIND

REGISTERS USED: UB, UF

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

Calls MNMFIND to see if modifier is valid, and if so, exits to the appropriate executor.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: SETLOG

ABSTRACT: Establishes/Terminates Command Logging

ENTRYS: SETLOG

SOURCE LIBRARY ROUTINES: SVC7

EXTRN: SCANNER, COMMACK1,NOPARM, CHECKCSL

REGISTERS USED: UC, UA, UE, U2, U8

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

If no operands are specified, LU2 is closed and LOGFLAG is set to 0, to indicate no logging.

If an operand exists, an attempt is made to assign it to LU2. Next the LU is checked to see if the user has tried to assign the log to the console, and if so, an error is indicated. If not, the COPY option is checked for. If specified, LOGFLAG is set to -1, if not, to +1.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: PAUSE

ABSTRACT: Pauses a task

ENTRYS: PAUSE

SOURCE LIBRARY ROUTINES: TCB

EXTRN: S21PAUSE

REGISTERS USED: U5, U8, U9

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

Sets up a call to, and return register for, a call to S21PAUSE, who pauses the task.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

NAME: CMCLOSE

ABSTRACT: This module closes user logical units.

ENTRYS: CMCLOSE

SOURCE LIBRARY ROUTINES:

EXTRN: CLOSSUB, ERR.ASGN, CMDNEXT, FORMERR

REGISTERS USED: U1,U7

ON ENTRY: U1 - pointer to current position on input line

ON EXIT: U1 - bumped past last characters on input line processed
U7 - status of close

PRINCIPLES OF OPERATION:

CMCLOSE calls CLOSSUB to scan the input line and perform the actual close. It is returned the status in U7. If U7 ≠ 9 (LU not opened) or Ø, then exit is taken to ERR.ASGN.

Else, the next delimiter is checked. If it is a comma, the pointer is bumped past it, and the above sequence is repeated. If a terminator is found, we return to CMDNEXT. If any other character is found, we go to FORMERR.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: CLOSSUB

ABSTRACT: This routine does the actual closing of a given user lu.

ENTRYS: CLOSSUB, CLOSSUB2

SOURCE LIBRARY ROUTINES: TCB, SVC7

EXTRN: PARMERR, NOPRERR

REGISTERS USED: U1, U2, U4, U5, U6, U7, UØ

ON ENTRY: U1 = input string pointer
UC = return pointer
if entering
at CKISSYB2 UØ = LU #

ON EXIT: U7 = status
U4 = user tcb address
U5 = system tcb address

PRINCIPLES OF OPERATION:

The LU number is obtained from the input line; if it is not present we exit to NOPRERR, if the syntax is bad we exit to PARMERR. The LU number is compared against the task's maximum number of LU's. If it is not valid, we load U7 with 2 and return.

The user's LU table entry is moved to the Command Processor's LU3 slot, and the user's slot is zeroed out. An SVC 7 to close the Command Processor LU3 is executed, the status is loaded into Register 7, and we return.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: EXPAND

ABSTRACT: Increases a task's memory allocation

ENTRYS: EXPAND

SOURCE LIBRARY ROUTINES:

EXTRN:

REGISTERS USED: U0

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

This routine obtains the expand value from the command line, and puts it in an SVC 2,20 parameter block. An SVC 2,20 is executed, and the results, if error, are reported.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: CMRENAME

ABSTRACT: This module renames a file/device

ENTRYS: CMRENAME, CMREN2

SOURCE LIBRARY ROUTINES: SVC7

EXTRN: ASGNERR, CMDERROR, CMDNEXT, ERR.ASGN, CMDASGN

REGISTERS USED: U3, U2, UC, UB, U7

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

CMDASGN is called to assign the specified FD to LU3 for ERW. The mnemonic is checked to make sure it is not 'NULL'. The new fd is put into the SVC7 parameter block. Register 7 is set up to indicate "CLOSE, RENAME". Register C points to 'RENM'. CMREN2 stores register 7 into the function code of the SVC7 parameter block and executes the SVC7. If error occurs, it is reported, else exit is made to CMDNEXT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: REPROTEC

ABSTRACT: Reprotect a file/device

ENTRYS: REPROTEC

SOURCE LIBRARY ROUTINES:

EXTRN: CMREN2, CMDASGN

REGISTERS USED: U3, U2, U7, U0, UB

ON ENTRY:

ON EXIT: U7 = function code to indicate CLOSE, REPROTECT
UC = address of the string C'REPR'

PRINCIPLES OF OPERATION:

CMDASGN is called to assign the specified FD to LU3 for ERW. The new keys are obtained and stored in SVC7PBLK. Register 7 is set up for "CLOSE, REPROTECT" (X'C000') and Register C points to C'REPR'. CMREN2 is entered to do the Close and Reprotect.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: ASSIGN

ABSTRACT: Assigns a file/device to a user logical unit

ENTRYS: ASSIGN

SOURCE LIBRARY ROUTINES: SVC7, TCB

EXTRN: SPT.UTCB, SPT.STCB, CMDASSGN1, SCANNER, CLOSSUB2

REGISTERS USED: U0 - U7 U9, UB, UC

ON ENTRY:

ON EXIT: If error, R7 = SVC7 status

PRINCIPLES OF OPERATION:

The LU number is obtained and checked for validity. If the LU is assigned, the task is checked for DORMANT. If not dormant, then an error is indicated. If the task is dormant, the LU is closed by calling CLOSSUB, the FD is obtained and put into SVC7PBLK. If no more parameters are specified, keys of 0, and AP of SRW are put into the parameter block. If AP and/or keys are specified, they are put into the parameter block.

If the user task is in UT state, then the Command Processor resets its ET status in TCB.OPT. CMDASGN is called to assign the file/device to LU3. The Command Processor now goes back to ET state. If successful, the LU3 entry is copied to the appropriate user LUTAB entry, and system LU3 is zeroed.

If failure is detected, then ERR.ASGN is entered, with code in Reg 7.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: MARK

ABSTRACT: Calls routines to do the MARK

ENTRYS: MARK

SOURCE LIBRARY ROUTINES:

EXTRN: MARKSUB

REGISTERS USED: UB

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

Calls MARKSUB to do work. If condition code returned is not zero, then CMDERROR is entered.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: MARKSUB

ABSTRACT: Decodes the command line for mark

ENTRYS: MARKSUB

SOURCE LIBRARY ROUTINES: TCB

EXTRN: CMDASGN, SPT.STCB, COMMACK, MNMFIND

REGISTERS USED: U2, U3, U6, U7, UB, UC, UE

ON ENTRY: U7 = return to MARK

ON EXIT: U6 = address of DCB to mark off/on line
U7 = return to MARK

PRINCIPLES OF OPERATION:

This routine calls CMDASGN to assign the specified fd to LU3 for ERW. If there is an error, STATERR is exited to. A call to MNMFIND to detect ON/OFF option, and call the appropriate executor is made.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: MARKOFF

ABSTRACT: Mark a device offline

ENTRYS: MARKOFF

SOURCE LIBRARY ROUTINES: DCB

EXTRN: MOFFBLK

REGISTERS USED: U8, U6, U7

ON ENTRY: R7 = return to MARK
R6 = DCB address

ON EXIT: R7 = return to MARK
R6 = DCB address

PRINCIPLES OF OPERATION:

Resets the devices on-line bit (DFLG.LNB). If the device is a bulk device, MOFFBLK is called.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: MARKSUB2

ABSTRACT: Mark device on-line

ENTRYS: MARKSUB2, MARKS2

SOURCE LIBRARY ROUTINES: DCB

EXTRN: MONBLK

REGISTERS USED: U8, U0, U7, U6

ON ENTRY: U6 = DCB address
U7 = return to MARK

ON EXIT: Same as above

PRINCIPLES OF OPERATION:

Tests to see if device is a bulk device. If so, MONBLK is called. Else, the device's on-line bit is set (DFLG.LNB). Exit by BR U7.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: DISPLAY

ABSTRACT: Selects appropriate executor

ENTRYS: DISPLAY

SOURCE LIBRARY ROUTINES:

EXTRN: SCANNER, MNMFIND

REGISTERS USED: UA, UB, UC, UE

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

Calls MNMFIND to determine if modifier is appropriate, and exit to the appropriate executor.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: DEVICES

ABSTRACT: Display List of devices/PA/Keys/State

ENTRYS: DEVICES

SOURCE LIBRARY ROUTINES: SPTE, DCB

EXTRN: SPT.DMT, SPT.VMT, CMDNEXT, CMDWGB, DUPFD, GENBUFF, BLANKBUF

REGISTERS USED: U0, U2, U4, U5, U6, U7, U8, UE, UF

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

Calls DISPFD to get/assign any necessary display device. The GENBUF is cleared, and the header is written out. The DMT is located for each DMT entry, the mnemonic, physical address and keys are put in the line. If the device is off-line, "OFF" is put after the entry. If the device is a bulk device, and is on-line, then the VMT entry for the device is obtained, and placed in the line.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: LU

ABSTRACT: Display a task's logical units

ENTRYS: LU

SOURCE LIBRARY ROUTINES: TCB, DCB, FCB

EXTRN: SPT.UTCB, DISPFD, CMDWGB

REGISTERS USED: U0, U2, U4, U5, U6, U8, U9, UD

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

DISPFD is called to locate/assign any display device. The buffer is cleared, and the header is displayed. Each task LU entry is obtained. If it is not assigned nothing is displayed. If it is assigned, then the LU number and device mnemonic are displayed. If the LU is assigned to a file, then the volume name, file name and extention are obtained and displayed.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: DSPARMS

ABSTRACT: Displays parameters

ENTRYS: DSPARMS

SOURCE LIBRARY ROUTINES: SPTE, TCB

EXTRN: SPT.UTCB, SPT.SVOL, TCB, SPT.CTOP, SPT.UBOT, SPT.UTOP, SPT.MTOP, CMDWGB

REGISTERS USED: U0, U8, U9, UC, U6, U7

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

DISPFD is called to locate/assign any display device. The buffer is cleared. The name of each successive parameter is obtained from DPMTBL, each parameter is obtained, converted if necessary, and displayed.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: $CLEAR

ABSTRACT: Resets CSS to Level 0

ENTRYS: $CLEAR

SOURCE LIBRARY ROUTINES:

EXTRN: CSSCLOSE

REGISTERS USED: U9, U8

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

Sets register 9 to 0 (to indicate level to close to) and calls CSSCLOSE.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: $COPY, $NOCOPY

ABSTRACT: Sets CSS copy flag

ENTRYS: $COPY, $NOCOPY

SOURCE LIBRARY ROUTINES:

EXTRN:

REGISTERS USED:

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

$COPY sets CSSLIST to 1, $NOCOPY sets CSSLIST to 0.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: $EXIT

ABSTRACT: leave current CSS level

ENTRYS: $EXIT

SOURCE LIBRARY ROUTINES:

EXTRN: CSSCLOSE

REGISTERS USED: U9, U8

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

Gets current CSS level, and subtracts 1. Uses that value to call CSSCLOSE to close levels down to current -1.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

NAME: SETCND

ABSTRACT: Sets value of return code

ENTRYS: SETCND

SOURCE LIBRARY ROUTINES: SPTE

EXTRN: SPT.RC

REGISTERS USED: U0

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

Gets the value specified and stores at SPT.RC.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: CSSIFS

ABSTRACT: These commands are the CSS $IFn commands. They allow conditional execution of CSS streams.

ENTRYS: $IFG, $IFNG, $IFE, $IFNE, $IFL, $IFNL, $IFNULL, $IFNNULL, $IFX, $IFNX, GETCC, UPSKP

SOURCE LIBRARY ROUTINES: SPTE

EXTRN: CMDASGN, SPT.RC

REGISTERS USED: U0, U2, U3, U7, UC

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

$IFy y = E, NE, G, NG, L, NL

Calls get CC to get value specified, and compare it to SPT.RC. If the result of the compare meets the condition specified, then the next command is executed. If not, CSSSKIP is incremented.

$IFX, $IFNX

Call CMDASGN to assign the fd specified. If $IFX is specified and the assign is successful, or an error code indicating the file exists is returned, the next command is executed, else CSSSKIP is incremented. $IFNX checks for non-existence.

$IFNULL, $IFNNULL

Calls scanner to find next non-blank. If it is a terminator, then the parameter specified was null, else it was not. CSSSKIP is set if the condition specified is not set.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: BUILDS

ABSTRACT: Module BUILDS CSS files

ENTRYS: BUILD, $BUILD, BUILDDSP

SOURCE LIBRARY ROUTINES: SVC7, SPTE

EXTRN. SPT.CSBF, CMDWRITE, MSGLOG, PREPRO

REGISTERS USED: U0, U2, U3, U4, U6, U7, U8, UE

ON ENTRY: To BUILDDSP CC = + if BUILDFLG is +
CC = - if BUILDFLG is -

ON EXIT:

PRINCIPLES OF OPERATION:

BUILD and $BUILD call BUILDUP to do the necessary assignments of the file. BUILD sets BUILDFLG to -1, $BUILD to 1.

BUILDUP allocates and assigns (or just assigns, if the file/device exists) a file.

BUILDDSP is entered with CC + or -. It exits to BUILD1 if -(BUILD in effect).

BUILD1 logs the line just read, compares it to an ENDB. If not, it writes it out, and gets a new line. If ENDB it resets BUILDFLG to 0.

If CC is +, the $BUILD is in effect. It calls PREPRO to expand the line, MSGLOG to log it. $ENDB is checked for. If not, then the line is written out. If so, the BUILDFLG is set to 0.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: $JOB

ABSTRACT: Delimits a run-unit

ENTRYS: $JOB

SOURCE LIBRARY ROUTINES. SPTE

EXTRN: SPT.RL, CSSCLOSE

REGISTERS USED: U2/U9, U8, UC

If $JOB is in effect (JOBFLAG ≠ 0), then an error occurs. All CSS LEVELS are closed by a call to CSSCLOSE, and JOBFLAG is set to 0.

If $JOB is not in effect, the current CSSLEVEL is stored at JOBFLG, and SPT.RC is set to 0.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment. It shall not be used for any other purposes nor disclosed to unauthorized persons.

NAME: $TERMJOB

ABSTRACT: Terminates a $JOB delimited run-unit

ENTRYS: $TERMJOB

SOURCE LIBRARY ROUTINES:

EXTRN:

REGISTERS USED: U2, U3

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

Reset all JOB associated flags to 0.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: $SKIP

ABSTRACT: Sets flag to indicate that a $JOB run-unit should not execute any more commands

ENTRYS: $SKIP

SOURCE LIBRARY ROUTINES: SPTE

EXTRN: SPT.RC

REGISTERS USED: U9

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

This sets JOBSKIP to indicate to COMMANDR that all statements until a $TERMJOB should be skipped. It also sets SPT.RC to 255.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: SETLU

ABSTRACT: Used to determine which LU command read should come from

ENTRYS: SETLU

SOURCE LIBRARY ROUTINES: TCB

EXTRN: SPT.UTCB

REGISTERS USED U4, U2

ON ENTRY: UC = return address

ON EXIT: U2 = LU to read from

PRINCIPLES OF OPERATION:

If the task is dormant or paused, returns 0. Otherwise, it gets CSSLEVEL. If CSSLEVEL = 0 (no CSS files open), it returns 0. Else LU = CSSLEVEL+4.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: CHECKCSL

ABSTRACT: Used to see if an LU is assigned to the DCBCMD

ENTRYS: CHECKCSL

SOURCE LIBRARY ROUTINES: SVC7,TCB

EXTRN: SPT.STCB

REGISTERS USED: U4, U2

ON ENTRY: U2 = LU #

ON EXIT: CC = 0 if LU assigned to DCBCMD

PRINCIPLES OF OPERATION:

Check the LU table entry against DCBCMD, if not equal returns. If equal, it closes the LU, and exits with CC = 0.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

NAME: COMMACK

ABSTRACT: Checks to see if next non-blank is a comma

ENTRYS: COMMACK, COMMACK1

SOURCE LIBRARY ROUTINES:

EXTRN: SCANNER

REGISTERS USED: U1, UA, UC

ON ENTRY:
UB - return
U1 - to current position in command line if to COMMACK1
UA - character to be checked

ON EXIT:
UA - current character
U1 - pointing to that character

PRINCIPLES OF OPERATION:

Calls SCANNER to get next non-blank into UA. Compares it against a comma. If it is it returns, otherwise exits to FORMERR.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: SCANNER, TERMCHK

ABSTRACT: Finds next non-blank character in command line, and compares it against a ";" or carriage return.

ENTRYS: SCANNER

SOURCE LIBRARY ROUTINES:

EXTRN: TERMCHK

REGISTERS USED: U1, UA

ON ENTRY: UC - return, if call to TERMCHK UA contains the character

ON EXIT:
UA - next non-blank character
CC - 0 if terminator
U1 - updated to point to non-blank

PRINCIPLES OF OPERATION:

Finds a non-blank, put it in UA, goes to TERMCHK to check it against C/R or ";".

TERMCHK compares it to a C/R or ";" and returns CC = 0 if it is one.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: MNMFIND

ABSTRACT: Find a mnemonic in a mnemonic table, exit to a routine

ENTRYS: MNMFIND

SOURCE LIBRARY ROUTINES:

EXTRN: SCANNER, PARMERR

REGISTERS USED: UC, UB, UE

ON ENTRY: UE = address of mnemonic table
UB = address of branch table

ON EXIT:

PRINCIPLES OF OPERATION:

Does an SVC 2,17 against the table pointed to by UE. If found, it branches to the routine specified by the index into the table and the table pointed to by UB.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: BUFFINIT

ABSTRACT: Set up a CSS buffer

ENTRYS: BUFFINIT

SOURCE LIBRARY ROUTINES: SPTE

EXTRN: SPT.CSBF

REGISTERS USED: U2, U3

ON ENTRY: UC = return
U1 = address of buffer

ON EXIT:

PRINCIPLES OF OPERATION:

Put a semicolon in first position of buffers, and a C/R at last.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: CSSBUFF

ABSTRACT: Return address of a buffer, appropriate for this CSSLEVEL

ENTRYS: CSSBUF

SOURCE LIBRARY ROUTINES: SPTE

EXTRN: SPT.CSBF

REGISTERS USED: U2, U3, U1

ON ENTRY: U2 = return

ON EXIT: U1 = address of buffer

PRINCIPLES OF OPERATION:

Buffer address = CMDBUFFS + (CSSLEVEL*SPT.CSBK)

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

NAME: CSSCLOSE

ABSTRACT: CLOSE CSSLEVELS down to a specified level

ENTRYS: CSSCLOSE

SOURCE LIBRARY ROUTINES:

EXTRN: CMDCLOSE

REGISTERS USED: U2, UA, U9

ON ENTRY: U9 = CSSLEVEL to close to
U8 = return

ON EXIT: CSSLEVEL update to reflect level closed to

PRINCIPLES OF OPERATION:

Gets CSSLEVEL, compares against limit, if equal exits. If not, it calls CMDCLOSE to close this CSSLEVEL's LU, decrements CSSLEVEL and stores, loops to do next level.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: MSGLOG

ABSTRACT: Used to log a line just read

ENTRYS: MSGLOG

SOURCE LIBRARY ROUTINES: SPTE

EXTRN: SPT.CSBF, CMDWRITE, SETLU

REGISTERS USED: U2, U5, U6, U7, UB, UC, U8

ON ENTRY: U1 = address of line just read
UB = return

ON EXIT:

PRINCIPLES OF OPERATION:

Gets the value of the current LU. If it is 0, then it is not a CSS input, and need not be logged to console (if LU = 0 then it was read from console). If LU2 is not assigned then the exit is made. If LU2 is assigned, then LOGFLAG is saved and temporarily set to 0. The line read is written to LU2 and the value of LOGFLAG is restored.

If LU≠0 then CSSLIST is checked. If CSSLIST=0 ($NOCOPY in effect) then nothing is done. Else the line just read is written to LU2.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

NAME: PREPRO

ABSTRACT: Does CSS expansion

ENTRYS: PREPRO

SOURCE LIBRARY ROUTINES:

EXTRN: TERMCHK

REGISTERS USED: U1, U2, U3, U5, U6, U8, U9, UC, UB

ON ENTRY: U1 = address of buffer read

ON EXIT: U1 = address of buffer to process

PRINCIPLES OF OPERATION:

This calls CSSBUFF to get address of buffer to move characters to. Characters are moved until a C/R is encountered. When a C/R is encountered, it is moved, and PREPRO is exited.

If an @ is encountered, successive @'s are counted. This is subtracted from CSSLEVEL to get address of buffer (found in PTRSTACK) to scan for parameters. If total is negative then a null should be substituted. The parameter number is obtained. If not zero, then the appropriate level buffer is scanned for that parameter, and when found, it is moved to the expansion buffer. If not found, then a null is substituted. If parameter 0 is specified, then the line is scanned backward for a terminator. When found, characters are moved from the characters after the terminator until the next blank.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: BLANKBUF

ABSTRACT: Clears GENBUFF to blanks

ENTRYS: BLANKBUF

SOURCE LIBRARY ROUTINES:

EXTRN:

REGISTERS USED: U5, U6

ON ENTRY: UC = return

ON EXIT:

PRINCIPLES OF OPERATION:

Puts blanks in GENBUFF.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: DISPFD

ABSTRACT: Routine to find/assign a file/device for a DISPLAY

ENTRYS: DISPFD

SOURCE LIBRARY ROUTINES:

EXTRN: SCANNER, CMDASGN, COMMACK1, CHECKCSL

REGISTERS USED: U2, U3, U8, UB, UC

ON ENTRY: U8 = return address

ON EXIT:

PRINCIPLES OF OPERATION:

Finds next non-blank, if it is a terminator it just returns. COMMACK1 is called to make sure it is a comma. CMDASGN is then called to assign the FD to LU3 for SWO. If an error, the FDERROR is exited to. Else, the LU is checked for DCBCMD by CHECKCSL. Return is made to FDERROR if it is to DCBCMD, else return is made on UC.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: CMDCLOSE

ABSTRACT: To close a Command Processor LU

ENTRYS: CMDCLOSE

SOURCE LIBRARY ROUTINES: SVC7

EXTRN:

REGISTERS USED: U2

ON ENTRY: U3 = LU to CLOSE
UC = return

ON EXIT:

PRINCIPLES OF OPERATION:

Store U2 into an SVC7 parameter block, and executes an SVC7, then returns without checking status.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: CMDASGN

ABSTRACT: To assign a file/device to a Command Processor LU

ENTRYS: CMDASGN, CMDASGN1

SOURCE LIBRARY ROUTINES: SVC 7

EXTRN:

REGISTERS USED:

ON ENTRY: To CMDASGN or CMDASGN1
U2 = LU #
U3 = access privilege (3 bits)
UC = return

ON EXIT: U7 = status of assign
CC = 0 if successful

PRINCIPLES OF OPERATION:

At CMDASGN a pack-file descriptor to put the file into an SVC7 parameter block is performed.

At CMDASGN1 the access privilege is shifted left to its appropriate place in a halfword, and store it in the SVC7PBLK. Register 2 will be stored as the LU in the SVC7PBLK.

The SVC7 is executed, the status loaded into U7, and return is made to the caller.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

NAME: CMDWRITE

ABSTRACT: Used to write a line to any Command Processor LU

ENTRYS: CMDWRITE, CMDWRIMG, CMDWGB, CMDWGDX

SOURCE LIBRARY ROUTINES: SVC1, TCB

EXTRN: SPT.STCB, OUTSTAR

REGISTERS USED:

ON ENTRY:
```
To: CMDWRITE - U2 = LU   U6 = Buffer Start   U7 = Buffer End
    CMDWIMG  - U2 = LU   U6 = Buffer Start   U7 = Buffer End
                         U5 = Function Code
    CMDWGBX  - U2 = LU #
```

ON EXIT: U7 = status

PRINCIPLES OF OPERATION:

```
Entry to CMDWGB - loads U2  with a 3
                        U6 = GENBUFF
                        U7 = GENBUFFE

          CMDWGBX- loads U6 = GENBUFF
                        U7 = GENBUFFE

          CMDWRITE-loads U5 = X'2800'
```

Gets the LU and sees if assigned. If not, sees if LU = 2, if not sets LU = 2, goes to see if LU2 assigned. If LU = 2, sets LU = 0.

When assigned LU is found, LU is stored in CMDSVC1 as LU, U5 is stored as function code, and U6 and U7 are stored as starting and ending address.

The SVC1 is executed. If no error then the LU is checked to see if it was LU2. If so and if LOGFLAG = -1 (Log with COPY) then the write is repeated to LU0.

If error, and the I/O was not to the console, the LU is closed, and exit to IOERR. If the I/O was to the console, and the error was recoverable, exit is made to OUTSTAR to force a command read. If a I/O error other than recoverable error is found on the console, the system is crashed with code = 2.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: Magnetic Tape Commands

ABSTRACT: Executes a magnetic tape command to a device/file

ENTRYS: WFILE, FFILE, FORREC, BFILE, BACREC, REWIND

SOURCE LIBRARY ROUTINES: SVC1, SVC7, TCB, SPT.UTCB

EXTRN: CMDCLOSE, COMMACK, SCANNER

REGISTERS USED: U5, UC, U3, U2, U4, U0, U9

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

At each entry point the appropriate function code for each function is loaded. Next, CMDASGN is called to assign the file/device to system LU4 for SRW. If the fd specified a device, the command is executed.

If the fd specified a file, then LU number to which this file is assigned is obtained. The user LU is copied to a Command Processor LU and the command is executed.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

NAME: MOFFBLK

ABSTRACT: Marks a disc off-line

ENTRYS: MOFFBLK

SOURCE LIBRARY ROUTINES: SPTC, DCB

EXTRN: SPT.VMT

REGISTERS USED: U2, U3, U7, U8, U9

ON ENTRY: U6 = points to DCB
U7 = return

ON EXIT:

PRINCIPLES OF OPERATION:

The VMT entry for the disc is found and zeroed. If the bit-map modify flag is set, then the bit map is written out, and the flag is reset. If an error occurs writing out the bit map, BPAC-ERR is displayed.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

B/26

NAME: MONBLK

ABSTRACT: Used to mark a disc on-line

ENTRYS: MONBLK

SOURCE LIBRARY ROUTINES: DCB, SPTE

EXTRN: SPT.VMT

REGISTERS USED: U3, U2, U4

ON ENTRY: U6 = DCB address
U7 = return to mark

ON EXIT: U6 - DCB address
U7 - return to mark

PRINCIPLES OF OPERATION:

This routine sets up the bit-map and direc ory-read parameter blocks in the DCB. It reads the VD. The directory pointer an bit map pointer are stored in the DCB. The VMT entry is found, and the name inserted. It exists to MARKS2 to set the on-line bit.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: VOLUME

ABSTRACT: Sets up default system volume

ENTRYS: VOLUME

SOURCE LIBRARY ROUTINES: SPTE

EXTRN: SPT.SVOL

REGISTERS USED: UC, U2, U5

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

Get a 4 character name and puts it in SPT.SVOL.

This information is proprietary and [illegible] by TERADATA [illegible] purpose of using and maintaining TERADATA [illegible] not be used for any other purposes [illegible]

NAME: CMDELETE

ABSTRACT: Deletes a file

ENTRYS: CMDELETE

SOURCE LIBRARY ROUTINES: SVC 7

EXTRN: ERR.ASGN

REGISTERS USED:

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

Gets the fd into a SVC7 parameter block, and executes an SVC 7 delete. If unsuccessful, it displays a message.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: ALLOCATE

ABSTRACT: Allocates a file

ENTRYS: ALLOCATE

SOURCE LIBRARY ROUTINES: SVC 7, SPT

EXTRN: SPT.CHBK, SCANNER, COMMACK1

REGISTERS USED: U5, U6, U7, U0, U8, UC, UA, UB

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

The fd is obtained and put in the SVC7 parameter block. Next the file type option is obtained, and the appropriate sub-executor is entered. If contiguous, the size must be specified. It is obtained and stored in the SVC7PBLK. Keys default to 00. Keys are scanned for if they exist they are stored, else they are set to the default.

If chained, default blocksize = 1, lrecl = 126, Keys = 00. If any of these parameters are specified they are stored in the SVC7PBLK.

The SVC7 is executed, and if error status is displayed.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: DISPFILE

ABSTRACT: Display files on a disc

ENTRYS: DISPFILE

SOURCE LIBRARY ROUTINES: DMT, DCB, SVC1, VD, DIR

EXTRN: BLANKBUFF, SCANNER, COMMACK

REGISTERS USED: U0-UF

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

All indicators are reset. The command is now scanned. The volid is obtained and checked. It's associated device is then found and assigned SRO to LU4. The rest of the specified display is obtained. If "-" is specified in the filename field DISPIND1 is set to a positive number. If "-" is specified in the extention, then DISPIND2 is set to a positive number. If "-" is not specified, then the name is saved at DIRFN and DIREXT respectively.

DISPFD is called to find/assign the display device. The VD is read to obtain the directory pointer. Each directory sector is checked and only active entries are processed. For an active entry if DISPIND1 is not set, then the filename is compared against the specified filename. If no match, then the next entry is processed. If DISPIND1 is set, no filename compare is made.

If DISPIND2 is not set then an extention compare is made. If a match is not found then the next entry is processed. If DISPIND2 is set no extention match is made.

If all match, or if a combination of DISPIND1, DISPIND2 and match selects a directory entry, then it is displayed.

If neither DISPIND1 or DISPIND2 is set the search stops after a match on both filename and extention is found. Otherwise, the entire directory is searched.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

B/26

NAME: INITIAL (NOTE: User should be aware of OS/32 File Structure/Allocation before reading)

ABSTRACT:

This routine initializes a disc. It may give it a name, save an OS image, and/or clear it to unused state.

ENTRYS: INITIAL

SOURCE LIBRARY ROUTINES: SPTE, TCB, DCB, VD, SVC 1

EXTRN: CMDASGN, SCANNER, COMMACK

REGISTERS USED: All

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

CMDASGN is called to find/assign the specified device to LU 3 for ERW. It then verifies that the device is a bulk device. The volid is obtained and saved.

If no options are specified - the volume descriptor is read, the new name inserted, and the volume descriptor is rewritten.

Options are scanned against a table for CLEAR or SAVE. If CLEAR is specified, sector 0 of the disc is read-checked to verify it is good. If not, the pack cannot be used and a message to that effect is logged. The DCB is examined to determine the size of the disc. From the size of the disc the size of the bit map is calculated (bit map size = $\frac{\text{disc size}}{2048}$ rounded up to the nearest integer). Next the disc is searched to find enough good, contiguous sectors in which to put the bit map. If not enough exist anywhere on the pack, it cannot be used, and a message is logged to that effect.

When the sectors have been found, they are cleared to 0, and written out. The sectors used for the bit map are marked as being used in the appropriate place in the bit map. A volume descriptor is built with pack-id, and bit map pointer written in, and with the rest of the sector zeroed. This is now written out to sector 0. The entire disc is now read-checked. If a sector is found to be bad it is marked as allocated in the appropriate place in the bit map. If a sector found to be bad is a bit map sector, the disc cannot be used, and a message is logged to say so.

If CLEAR is specified, SAVE is checked for. CLEAR and SAVE can both be given, but CLEAR must precede SAVE. If SAVE is the first option encountered, it may not be followed by CLEAR.

If SAVE is specified - the Command Processor puts itself in what appears to be RSA state. It does this so that it may call SVC 7 subroutines. First the VD is read. If an OS image is present on the disc, the RELEB is called to release the sectors it occupies. Next the save of the OS is computed (UBOT/256 rounded up) and GETSECTR is called to obtain the necessary space. The OS image is written out to that area of the disc. RSA state is left, and the address of the OS image is stored in the VD, along with the volume-id, and is rewritten.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

NAME: LOADER

ABSTRACT: Parses the load command, sets biases, and goes to appropriate loader.

ENTRYS: LOAD/LOADFAIL

SOURCE LIBRARY ROUTINES: TCB

EXTRN: LOADHALF, LOADFULL

REGISTERS USED:

ON ENTRY: U1 points to command line

ON EXIT: U1 points after last parameter parsed

PRINCIPLES OF OPERATION:

On entry to LOADER:

The UTCB is examined to see that the RS and ES bits in status are not set, and that the LW bit in "waits" is 0, and the dormant bit is 1. If these conditions are not met, a branch to SEQERR is taken. An SVC 2,16 to get the FD to load from is now performed, putting the fd into an SVC7 parm block for later use. If SVC 2,16 detects syntax error, a branch to PARMERR is taken. A zero is placed at LOADSTAT.

The fd just obtained is assigned ERO to SYSLU1. If the assign returns an error register D is loaded with a 1, branch to LOADFAIL. If assign is IO, see if any biases are specified. If not, put UBOT to IMPBASE. If so, set imp bias, round up and store at imp base, set pure bias, if non, store 0 at pure base, else, store value. Then go to appropriate loader.

LOADFAIL - If LOADSTAT = 0, build a LDRn where n is value in UD. BAL to load register C with address of nat msg. Branch to CMDERROR.
If LOADSTAT ≠ 0, store value in UD at location contained in LOADSTAT, load U9 with 2, load U8 with A(CMDEMPTY), load UD with TWT.LWM, clear CMDLR to 0, branch to TMREMW.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

NAME: LOADOVLY

ABSTRACT. Position for overlay

ENTRYS: LOADOVLY

SOURCE LIBRARY ROUTINES.

EXTRN: CMDLR, LOAD5

REGISTERS USED:

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

Sees if CMDLR specifies a REWIND is to be done. If so, it does it, if not it exits to do the load (LOAD5).

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: LOADFULL

ABSTRACT: Reads and processes 32-bit object tapes

ENTRYS: LOADFULL

SOURCE LIBRARY ROUTINES:

EXTRN:

REGISTERS USED: ALL

ON ENTRY:

Undefined

ON EXIT:

UD = error code (0 if successful)

PRINCIPLES OF OPERATION:

This module reads 126 byte object records. It calls CHECKER to do sequence check and checksumming. Loader items are obtained as bytes and used as an index to branch to the appropriate processing routine. Processing routines return to LDFUL3 to get the next loader item. At end it sets up CTOP, UTOP, etc.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: LOADHALF

ABSTRACT: Reads and processes 16-bit object tapes

ENTRYS: LOADHALF

SOURCE LIBRARY ROUTINES:

EXTRN:

REGISTERS USED: ALL

ON ENTRY: Undefined

ON EXIT: If error during the load, UD = error number

PRINCIPLES OF OPERATION:

This module reads a 108 byte record, and calls CHECKER to do sequence check and checksumming. Loader items are obtained as nibbles (4 bits). The nibble is used as an index to branch to the appropriate processing subroutine. After each processing routine has performed its function it returns to the main dispatcher to get the next nibble. At end it exits to LDFUL8 to set up system pointers (UTOP, CTOP, etc.).

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: CHEWING

ABSTRACT: This routine gets bytes or nibbles from SECTORBF starting from relative nibble pointer as contained in CURRNIBL.

ENTRYS: BYTER, NIBBLER

SOURCE LIBRARY ROUTINES:

EXTRN:

REGISTERS USED: U2, U3, U4, U5, U7

ON ENTRY. U2 contains count of bytes or nibbles requested

ON EXIT: U5 contains the bytes or nibbles requested

PRINCIPLES OF OPERATION:

A count of nibbles requested is given directly if entered at Nibbler or calculated by doubling count given to Byter. A byte is fetched from loader record. If the current nibble count was even the left nibble of this byte is used else the right nibble is used. This nibble is logically ORed into the returning register after the returning register is shifted left 4 bits. The nibble pointer is incremented, the count is decremented, if non-zero loop to fetch another nibble.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

NAME: CHECKER

ABSTRACT: Performs sequence checking and checksum verification on loader records

ENTRYS: CHECKER

SOURCE LIBRARY ROUTINES:

EXTRN:

REGISTERS USED: U8, U9, UA, UB, UD

ON ENTRY: Loader record in SECTORBF, address of UTCB in U4

ON EXIT: On error exit to LOADFAIL, UD contains error code

PRINCIPLES OF OPERATION:

LASTSEQ is decremented and compared with sequence number just read. If they match, replace LASTSEQ with this new number. If no match occurs, error exit to LOADFAIL.

Determine if in halfword or fullword mode, generate checksum for corresponding length record, and compare to checksum contained in record. If match, return to caller. If no match, error exit to LOADFAIL.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: SVC7

ABSTRACT: Decode the SVC7 command byte and invoke the proper executor.

ENTRYS: SVC7, SV7.CMD

SOURCE LIBRARY ROUTINES: TCB

EXTRN: FETCH, ALLO, OPEN, CAP, RENAME, REPRO, CLOSE, DELETE, CHECKPT, TMRSOUT

REGISTERS USED: UØ-UF without SAVE/RESTORE

ON ENTRY:
RS STATE
U9 - Address of TCB
UD - SVC7 parameter block

ON EXIT:
RS STATE
U2 - Address of TCB
U5 - SVC7 parameter block

PRINCIPLES OF OPERATION:

SVC7 is entered from the First Level Interrupt Handler (FLIH). The routine obtains the command byte from the user's parameter block and decodes the command; multiple commands are processed left to right. The routine branches to the appropriate executor; upon completion of all commands, exit is via a branch to TMRSOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: OPEN

ABSTRACT: Perform common open processing for all devices and files and branch to the appropriate OPEN logic routine.

ENTRYS: OPEN, OPN.BDSK

SOURCE LIBRARY ROUTINES: DCB, DIR, TCB

EXTRN: LUCHECK, DMTLOOK, VMTLOOK, FDCHECK, APCHECK, EVQCON, TMRSRSA, DIRLOOK, OPEN.CO, OPEN.CH, OPEN.DEV, SV7.CMD, EVDIS, TMRSARS, SVC7.ERR

REGISTERS USED: U0-UF without SAVE/RESTORE

ON ENTRY:
RS STATE
U2 - address of the TCB
U5 - SVC 7 parameter block address

ON EXIT:
RS STATE (on exit to OPEN.DEV or SV7.CMD); RSA state otherwise
U2 - address of the TCB
U5 - SVC7 parameter block address
U7 - address of the DCB
U1 - address of the LU table entry
UB - DCB attribute byte (on exit to OPEN.CH and OPEN.CO)

PRINCIPLES OF OPERATION:

This routine is invoked via a branch from SVC7 and performs all common open processing. The routine begins by performing SVC7 parameter block validation; any errors cause a branch to SVC7.ERR. OPEN then calls DMTLOOK to search for a matching device; if a match is found and the device is not a direct-access device, the routine exits via a branch to OPEN.DEV, which completes the open for a non-direct access device. If the device is direct-access, it may only be opened by an E-task; attempting to open a direct-access device by a U-TASK causes an exit via a branch to SVC7.ERR. The routine enters NSU state to complete the open of the direct access device, returns to RS State and returns to SV7.CMD.

If no match was found in the DMT, the VMT is searched for a match; no match in the VMT causes an error return via SVC7.ERR. If a match is found in the VMT, OPEN obtains control of the volume directory via EVQCON and enters RSA state to allow SVC1 I/O calls. Further error checks are performed; then the routine ends by branching to OPEN.CO (to complete the open process for contiguous files) or OPEN.CH (to complete the process for chained files).

OPEN contains the subroutine OPN.BDSK, which assigns the bare disc for requested privileges prior to allocating or opening files. It is called directly from common OPEN processing prior to branching to OPEN.CO or OPEN.CH, and also called from ALLO.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: OPEN.DEV

ABSTRACT: Complete the Logical Unit Assignment to a non-direct access device.

ENTRYS: OPEN.DEV

SOURCE LIBRARY ROUTINES: DCB

EXTRN: APCHECK, SV7.CMD

REGISTERS USED: U0-UF

ON ENTRY:
U2 - TCB Address
U5 - SVC 7 Parameter Block Address
U7 - DCB address

ON EXIT:
U2 - TCB Address
U5 - SVC 7 Parameter Block Address

PRINCIPLES OF OPERATION:

OPEN.DEV receives control directly from OPEN to complete the assignment of a non-direct access device.

If the assignment is directed towards the console device, the address of the dummy DCB (DCBCMD) and its attributes (DCB.ATRB) are moved to the LU table entry for the LU being assigned. Return is to SV7.CMD.

If the device is not the console device, the device must be on-line or the calling task must be an E-task. Then NSU state is entered and the access privileges are converted to numeric quantities via APCHECK. If the device is the null device, no further tests are necessary. Otherwise, the DCB Read and Write keys are checked with the specified keys in the SVC 7 parameter block.

Finally, the new read and write counts are saved in the DCB. The LU table entry is set up to point to the device's DCB. The DCB attribute byte is also moved to the LU Table.

The routine returns to SV7.CMD.

If any errors are encountered, the appropriate error status is saved in the task's SVC 7 Parameter Block, and OPEN.DEV exits directly back to the calling task via TMRSOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: OPEN.CO

ABSTRACT: Complete the open of Contiguous Files. The routine is entered directly from OPEN, and upon successful completion returns to SV7.CMD to continue SVC 7 processing. The routine is entered in RSA state, having control of the Directory Leaf. OPEN.CO establishes the initial values in the FCB required to process a Contiguous file.

ENTRYS: OPEN.CO

SOURCE LIBRARY ROUTINES: FCB, DIR, DCB

EXTRN: LUCHECK, PUTD, APCHECK, DIR.SCAN, TMRSARS, EVDIS, SV7.CMD, TMRSOUT, CLO.BLK

REGISTERS USED: U0-UF, without save/restore

ON ENTRY:
RSA STATE
UC, UF - New Write, Read Counts
UB - Attribute byte
U7 - DCB pointer
U2 - TCB pointer
U5 - User parameter block pointer

ON EXIT:
RS STATE
U5 - User parameter block pointer
U2 - TCB pointer

PRINCIPLES OF OPERATION:

OPEN.CO begins by calling GETFCB, to allocate a contiguous FCB. Upon successful return from GETFCB, OPEN.CO establishes the starting and ending addresses of the Contiguous file buffer in the FCB. It then calls LUCHECK to obtain the pointer to the LU table entry for this file, and stores the FCB pointer and file attribute byte into the LUTABLE. Next the first logical block address, last logical block address, and current sector number are obtained and saved in the FCB. The initialization address for data transfers and the command entry point for the disc driver are saved into the FCB. The directory is updated via a call to PUTD. The read and write counts are obtained by a call to APCHECK, and saved into the FCB. A pointer to this FCB's Directory block, and its offset within that block are established by a call to DIR.SCAN.

The routine then returns to RS state by calling TMRSARS, releases control of the Directory by calling EVDIS, and returns to SV7.CMD.

If the call to GETFCB is unsuccessful, a buffer error (X'08') is saved into the user's parameter block. The Bare disc is closed via CLO.BLK, RS state is entered, the directory is released, and the task returns directly by branching to TMRSOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: OPEN.CH

ABSTRACT: Complete the open of Chain Files. The routine is entered directly from common OPEN processing, and returns directly to SV7.CMD upon successful opening of a Chain File. The routine is entered in RSA state, having control of the Directory. OPEN.CH establishes the initial values in the FCB required to process Chain Files.

ENTRYS: OPEN.CH

SOURCE LIBRARY ROUTINES: DIR, FCB, DCB

EXTRN: GETFCB, EVQCON, GETSECTR, EVDIS, TMRSOUT, RELEFCB, PUTD, LUCHECK, APCHECK, TMRSARS, CLO.BLK, SV7.CMD

REGISTERS USED: U0-UF, without save/restore

ON ENTRY:

RSA STATE

U2	TCB pointer	UB	Attribute byte
U5	User parameter block pointer	UC, UF	New Write, Read Counts
U7	DCB pointer		

ON EXIT:

RS STATE

U2	TCB pointer
U5	User parameter block pointer

PRINCIPLES OF OPERATION:

OPEN.CH begins by computing the size of the Chain File FCB and calling GETFCB, to allocate the FCB. Upon successful return from GETFCB, the Directory pointer in memory is saved into the DCB and the FCB, and the parameter block starting and ending addresses for the two buffers are computed and saved into the FCB. The initialization entry point for data transfers and the command entry point for the disc driver are moved to the FCB. The first logical block address, last logical block address, number of logical records, blocksize, and logical record length are moved from the directory to the FCB. The number of blocks currently in the file is computed and saved in the FCB.

The access privileges are checked to insure that the file can be assigned the desired access privileges. If the file is open for write only, the I/O flags are set to indicate a write sequential operation. For a new file, one block is established on the disc via a call to GETSECTR. The sector pointer returned from GETSECTR is saved in both the first logical block address and the last logical block address fields in the FCB and the routine continues by updating the directory.

If the File already exists, the current logical block is read into memory, via an SVC 1 Random Wait call. For both new and existing files, the directory is updated by a call to PUTD. LUCHECK is called to obtain this file's slot in the LUTABLE, and the FCB pointer and attribute byte are stored into the LUTABLE. The read and write counts are converted by APCHECK, and saved into the FCB along with the attribute byte, indicating allowable data transfers for this file.

If the file is being open for read and write privileges, and it is a new file, the logic proceeds exactly as if it is being open for write privileges only. For an existing file being opened, the current sector is read into memory via an SVC1 Random Wait call. If there exists more than one block in the file already, the next block is read into memory (using the alternate buffer in the FCB) via an SVC 1 Read, Random, and Proceed call. The rest of the logic is identical to that specified above, starting with the update of the directory.

The routine returns to RSA state via a call to TMRSARS, releases control of the Directory by a call to EVDIS, and closes the bare disc by calling CLO.BLK. The routine returns directly to SV7.CMD, if no errors were encountered following any of the subroutine calls. If any error conditions were returned, the routine sets the appropriate error status in the user's parameter block, releases the FCB via RELEFCB, and returns by a branch to TMRSOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: APCHECK

ABSTRACT: Compare the Access Privileges specified in the user SVC 7 parameter block against the access privileges in the system control block (DCB, FCB, or Directory). If the access privileges are valid, APCHECK converts the desired access privilege to a numeric value as follows: 0 if the access is not requested, -1 if the access is requested exclusively, and +1 if the access is requested for shared privileges. The returned access privileges are then added to read and write count fields in the control block.

ENTRYS: APCHECK

SOURCE LIBRARY ROUTINES: None

EXTRN: None

REGISTERS USED: U8-UF, U4, U5; U3 cannot be used

ON ENTRY:
U4 - Write, Read count from DCB, FCB, or Directory
U7 - DCB pointer
U5 - SVC 7 parameter block pointer
UB - attribute byte

ON EXIT:
UB - attribute byte
UC - Read Count
EE - Write Count

PRINCIPLES OF OPERATION:

The logic of APCHECK proceeds as follows: An internal flag is maintained, to indicate whether read or write is desired, and whether the access is for exclusive control, or shared usage. Then the attribute byte is checked to determine if the required access privilege is supported for the device; the request is ignored if it cannot be supported. The read and write counts are next set as indicated above. Finally the routine checks to make sure that at least read or write was granted. If any of these tests fail, the routine returns a condition code of 0. Otherwise, the read and write counts are returned, and the condition code is set positively.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: GETSECTR

ABSTRACT: Allocate a requested number of sectors on a directory device. Each allocatable sector is represented by a bit in an allocation (bit) map, maintained by the operating system. Each newly allocated sector is indicated by setting a corresponding bit in the bit map.

ENTRYS: GETSECTR

SOURCE LIBRARY ROUTINES: DCB

EXTRN: None

REGISTERS USED: U0, U1, U8-UF

ON ENTRY:
U3 - number of contiguous sectors desired
U7 - DCB pointer

ON EXIT:
UC - Starting Logical Block in Allocation
UE - Ending Logical Block in Allocation
CC = 0 - not enough free sectors
CC ≠ 0 - allocation completed

PRINCIPLES OF OPERATION:

GETSECTR begins by checking if there currently exists a bit map sector in memory. If so and if the allocation desired is one sector, then the searching begins with the sector in memory. If the allocation desired is more than one contiguous sector, the first bit map sector is read into memory via a call to GETB. GETSECTR then searches for 'n' contiguous sectors. If the allocation spans more than one bit map sector, internal flags are maintained, indicating the bit map sector the allocation begins in and another bit map is read into memory using GETB. Once the proper number of sectors are found, the routine returns to the start of the allocation to mark each sector as allocated. Then the first logical address and last logical address of the allocation are computed and returned to the calling routine. If there are not enough free sectors within the bit map to satisfy the allocation request, a completion code of 0 is returned.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: GETB

ABSTRACT: Read a desired bit map sector into memory

ENTRYS: GETB, GETBA

SOURCE LIBRARY ROUTINES: DCB

EXTRN: None

REGISTERS USED: U9, UC-UE: cannot use UA, UB, UF, U0-U8

ON ENTRY:
U9 - current relative block address
U7 - DCB address

ON EXIT:
U9 - last sector allocatable this bit map sector
UD - previous current relative block address

PRINCIPLES OF OPERATION:

If there currently exists a bit map sector in memory, its relative block address is compared against the desired block address. If they are the same, no I/O is performed. If no sector exists in memory, the desired one is read in. If one is currently in memory, and if it has been modified, it is written out before a new sector is read in. If it has not been modified, the write is not performed, and the desired sector is read in. If the write encounters any I/O errors, the system crashes X'301'. If the read encounters any I/O errors, the system crashes X'300'.

The DCB.SIZE field is used to compute the last allocatable sector on the last sector in the bit map. (The first n-1 sectors of the bit map represent 2048 allocatable sectors; sector n represents the remainder of the allocatable sectors).

If the bit map sector read in is the last sector in the bit map, the ending sector index is returned in a register. Otherwise, 2047 is returned, indicating a complete sector.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

NAME: RELEB

ABSTRACT: The purpose of RELEB is to release the allocation of a given number of sectors from a disc.

ENTRYS: RELEB

SOURCE LIBRARY ROUTINES: DCB

EXTRN: GETB

REGISTERS USED: U3, U4, U8-UF

ON ENTRY: U3 - Logical Block Address of sectors to be released
U4 - number of sectors to be released
U7 - DCB address

ON EXIT: U7 - DCB address

PRINCIPLES OF OPERATION:

RELEB is the inverse of the subroutine GETSECTR. The arguments to RELEB are two integer numbers, the LBA of the sectors to be released and the number of sectors to release. The LBA is converted to an ordered pair (S,O) (sector, offset) corresponding to the bits in the bit map representing this LBA. The ending bit in the bit map is converted to another ordered pair (S1,O1). GETB is then called to reset all the bits in the bit map between these two points.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

NAME: FDCHECK

ABSTRACT: The purpose of FDCHECK is to check the syntax of a file descriptor.

ENTRYS: FDCHECK, FDCHECK1

SOURCE LIBRARY ROUTINES:

EXTRN:

REGISTERS USED: UØ, U8-UF

ON ENTRY: U5-SVC7 Parm Block Pointer

ON EXIT: U5-SVC7 Parm Block Pointer

PRINCIPLES OF OPERATION:

FDCHECK checks the FD fields in the user's parameter block for syntactic errors. Entry at FDCHECK checks the file name and extension fields; entry at FDCHECK1 checks the volume field.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment; and shall not be used for any other purpose unless specifically authorized in writing

NAME: LUCHECK

ABSTRACT: The purpose of LUCHECK is to validate the logical unit specified in the user's parameter block.

ENTRYS: LUCHECK

SOURCE LIBRARY ROUTINES: TCB

EXTRN: TMRSOUT

REGISTERS USED: U1, U2, U5, U7, UF

ON ENTRY:
U2 - TCB pointer
U5 - SVC7 parameter block pointer

ON EXIT:
U1 - LUTAB entry address
U2 - TCB pointer
U5 -.SVC7 parameter block pointer
U7 - DCB/FCB address

PRINCIPLES OF OPERATION:

This routine checks the LU in the user's parameter block to make sure it is less than the maximum LU number. It sets the condition code to 0 if the LU is not currently assigned and otherwise if the LU is currently assigned.

It returns both a pointer to the matching entry in the LUTABLE and the address obtained from this entry. This address is either of an FCB (for a contiguous or chain file) or a DCB. The File Manager LU in the TCB is set up for all files.

If the LU was invalid, return is to TMRSOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: DIRLOOK

ABSTRACT: Search directory for file descriptor

ENTRYS: DIRLOOK, DIR.SCAN

SOURCE LIBRARY ROUTINES: DIR, DCB

EXTRN: GETD

REGISTERS USED: U8-UF

ON ENTRY:
U2 - TCB address
U5 - SVC 7 parameter block pointer
U7 - DCB address

ON EXIT:
CC = 2 if requested FD found
CC = 1 if directory empty
CC = 0 - available entry pointer returned

PRINCIPLES OF OPERATION:

Search volume directory searching for a requested FD. When an FD match is found, the routine returns. The required directory block is in memory. The block and offset of the first empty entry within the directory are also returned.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: GETD

ABSTRACT: GETD reads a requested directory block into memory.

ENTRYS: GETD

SOURCE LIBRARY ROUTINES: DCB

EXTRN: NONE

REGISTERS USED: UC, UE, UF, U7

ON ENTRY:
U7 - DCB pointer
UE - -1 if the next directory block is desired
0 if the first directory block is desired
n if the nth directory block is desired

ON EXIT:
U7 - DCB pointer
CC - 1 if directory block read in successfully
0 if the directory is empty

PRINCIPLES OF OPERATION:

According to the argument in UE, GETD will read the first, next, or nth directory block into memory. If the next block is requested, and no block is currently in memory, the routine crashes X'302'. If the read returns an I/O error, the routine crashes X'303'.

Once the required directory block is read into memory, the directory presence bit is set to insure that the directory is updated prior to another GETD.

The condition code is set as indicated above and the routine returns to the caller.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: RELED

ABSTRACT: Release a directory block when all its directory entries have become inactive.

ENTRYS: RELED

SOURCE LIBRARY ROUTINES: DCB, VD

EXTRN: RELEB, GETD, PUTD

REGISTERS USED:

ON ENTRY: U6 - LBA of the directory block to be freed
U7 - DCB pointer

ON EXIT: U7 - DCB pointer

PRINCIPLES OF OPERATION:

The routine begins by reading via GETD, successive directory blocks, searching for the desired directory block. If the directory was empty on the release attempt, the routine crashes X'304'. If the first directory block is the one to be released, the VD must be read in, to be updated to point to the next directory block as the start of the directory chain. If an I/O error occurs during the read or write of the VD, the system crashes X'305'.

Once the correct directory block is found, the directory chain is relinked, excluding the deleted block. If any I/O errors occur during this operation, the system also crashes X'305'. The allocation for this block is released by a call to RELEB, and the routine returns to the caller.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: DMTLOOK/VMTLOOK

ABSTRACT: Search DMT/VMT for match on device/volume name; return associated DCB pointer.

ENTRYS: DMTLOOK, VMTLOOK

SOURCE LIBRARY ROUTINES: SPT

EXTRN: None

REGISTERS USED: U8-UF

ON ENTRY: U5 - address of SVC 7 parameter block

ON EXIT: U5 - as above
U7 - pointer to DCB

PRINCIPLES OF OPERATION:

The routine searches either the DMT (enters at DMTLOOK) or the VMT (enters at VMTLOOK) for a match between the name field of the DMT/VMT and the name contained in the task's SVC 7 parameter block. Upon finding a match, the associated DCB address is returned and a positive condition code is returned.

If no match is found, a zero condition code is returned.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: ALLOD

ABSTRACT: Allocate 1 sector for use by Directory.

ENTRYS: ALLOD

SOURCE LIBRARY ROUTINES: DCB, DIR

EXTRN: GETSECTR, GETD, PUTD

REGISTERS USED: U3, U4, U8-UF

ON ENTRY: U7 - DCB address

ON EXIT: U7 - DCB address

PRINCIPLES OF OPERATION:

ALLOD allocates 1 sector for use by the Directory in the following manner: GETSECTR is called to allocate 1 sector on the volume. If the directory did not previously exist (DCB.DIRP=0) then the VD is read into memory and the VD.FDP pointer is set to contain the address of the sector obtained from GETSECTR. The VD is rewritten at this time. Then the DCB.DIRP field is set to contain this same address.

If a directory block already exists, the directory is read via calls to GETD, until the last existing directory block is found. Once found, the link field of the last directory block is set to point to the new sector and this block is rewritten via PUTD.

In either case, the link field of the new block is zeroed and all its entries marked inactive.

The directory is updated via call to PUTD; the routine then returns with a positive condition code.

If there exists no free sectors on the volume, ALLOD returns with a condition code of 0.

If an I/O error occurs during the read or write of the VD, the system crashes X'306'.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: GETFCB

ABSTRACT: The purpose of this routine is to allocate new FCBs, starting at the top of memory. GETFCB allocates both chain and contiguous types of FCBs.

ENTRYS: GETFCB

SOURCE LIBRARY ROUTINES: FCB, DCB, SPT

EXTRN:

REGISTERS USED: U3, U8-UF without SAVE/RESTORE

ON ENTRY:
RS STATE
U3 - size of FCB to be allocated in bytes

ON EXIT:
RS STATE
UA - pointer to FCB
CC = 0 if FCB successfully allocated
CC < 0 if allocation not possible

PRINCIPLES OF OPERATION:

This routine manages the chain of FCBs. The chain of FCBs are built starting at SPT.MTOP, down in memory to SPT.FBOT. The first two words of each FCB contain a pointer to the next FCB and the size of the current FCB. The bottom of the FCB chain (SPT.FBOT) must be greater than SPT.CTOP+2. If not, there is not enough memory available for the allocation, so GETFCB returns with a Condition Code < 0.

Each FCB associated with a DCB is chained from the DCB, starting at DCB.FCB.

GETFCB will Crash with a system crash code of X'307' if the FCB to be allocated is less than the size of a contiguous FCB.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: RELEFCB

ABSTRACT: The purpose of RELEFCB is to release a specified FCB; if the FCB was on the bottom of the FCB chain, the memory is freed up by adjusting SPT.FBOT.

ENTRYS: RELEFCB

SOURCE LIBRARY ROUTINES: DCB, FCB, SPT

EXTRN:

REGISTERS USED: U3, U8-UF

ON ENTRY: RS STATE
U3 - pointer to FCB to be released

ON EXIT: RS STATE

PRINCIPLES OF OPERATION:

RELEFCB will release an FCB in one of two ways; if the FCB is the last FCB in the chain of FCBs in the system, it is removed from the chain and the bottom of the FCB (SPT.FBOT) is adjusted to reflect the freed FCB. If the FCB is anywhere else in the chain, the FCB active bit is reset, indicating that the area for this FCB is available for a subsequent call from GETFCB.

RELEFCB will CRASH for each of the following reasons:

X'308' - attempt to release a non-existent FCB

X'30A' - many FCBs in chain and addresses do not match

X'30B' - FCB chain incorrect in DCB

X'30C' - FCB chain in DCB contains no matching address

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: ALLO

ABSTRACT: The purpose of this module is to allocate new files. For contiguous files, both a directory entry and a disc allocation are obtained. For chain files, only the directory entry is obtained.

ENTRYS: ALLO

SOURCE LIBRARY ROUTINES: TCB, DCB, FCB, DIR

EXTRN: FDCHECK, DMTLOOK, OPN.BDSK, EVQCON, TMRSRSA, DIRLOOK, ALLOD, GETD, GETSECTR, PUTB, PUTD, SV7.CMD, CLO.BLK, TMRSARS, EVDIS

REGISTERS USED: U0-UF

ON ENTRY: U2 - TCB address
U5 - SVC 7 parameter block address

ON EXIT: U2 - TCB address
U5 - SVC 7 parameter block address

PRINCIPLES OF OPERATION:

ALLO is the SVC 7 executor entered when an allocate function is indicated in the task's SVC 7 parameter block. If the system is SYSGENed to delete direct access support, ALLO returns directly to the task with a type error.

The first syntax for the file descriptor is verified by calling FDCHECK. Then the DMT and VMT are searched to make sure that the allocate is directed to a volume known to the system. To allow I/O operations to the disc on behalf of the file being created, the disc is assigned to the file manager LU via OPN.BDSK.

The bit map and directory leaves are connected to via calls to EVQCON and the routine enters RSA state (TMRSRSA) to allow the file manager to perform I/O. The directory management routines are used to search the directory to insure that the new file name is unique to that volume (DIRLOOK) and to find an empty entry for the new file. If the directory is currently empty, a new directory block is allocated by ALLOD. GETD is called to insure that the required directory block is currently in memory, and the directory entry for the file is initialized to contain the new file name.

If the file being created is contiguous, the complete file allocation is obtained via a call to GETSECTR. The directory pointers DIR.FLBA, DIR.LLBA are set to contain the addresses returned from GETSECTR as the first and last logical block (sector) addresses. The bit map is then updated by a call to PUTB. For a chain file, no allocation is performed. Rather, the block size and logical record length are moved from the parameter block to the directory entry.

For both file types, the Read and Write keys are moved to the Directory. Then PUTD is called to update the Directory. The assignment to the File Manager LU is closed by CLO.BLK, the routine returns to RS state (TMRSARS) and the directory and bit map leaves are disconnected via EVDIS. Return is to SV7.CMD, to continue scanning the function code.

If any errors are encountered during ALLO processing, the appropriate error status is saved in the task's SVC 7 parameter block and the task is returned to directly via TMRSOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: DELETE

ABSTRACT: For systems with Direct Access support, the DELETE function is included to delete Contiguous and Chain Files. The delete is accomplished by freeing the allocation of the file within the bit map, and marking as inactive the directory entry pertaining to the file.

ENTRYS: DELETE

SOURCE LIBRARY ROUTINES: DCB, TCB, FCB, DIR

EXTRN: DMTLOOK, VMTLOOK, EVQCON, TMRSRSA, TMRSARS. OPN.BDSK, DIRLOOK, RELED, PUTD, RELEB, EVDIS, RELEFCB, CLO.BLK, TMRSOUT, SV7.CMD

REGISTERS USED: U0-UF, without save/restore

ON ENTRY: U2 - TCB address
U5 - SVC 7 parameter block address of user

ON EXIT: U2 - TCB address
U5 - SVC 7 parameter block address

PRINCIPLES OF OPERATION:

DELETE is called directly from the SVC 7 driver routine, SVC7. If no direct access support exists in the system, the routine returns immediately via TMRSOUT, indicating a type error (X'0A')in the SVC 7 parameter block.

DELETE searches the DMT and the VMT via calls to DMTLOOK and VMTLOOK. If a match is found in the DMT, a type error is returned. If a match is not found in the VMT, a volume error is returned. A contiguous FCB is obtained, for use by DELETE itself, via a call to GETFCB. The routine then obtains control of the Directory by connecting to the Directory leaf via EVQCON and enters RSA state via TMRSRSA.

The bare disc is assigned by a call to OPN.BDSK, and the directory is searched via DIRLOOK, looking for a match between the File Descriptor specified within the user's parameter block and the directory entries. Once the proper directory entry is obtained, the write and read count fields are checked. Both fields must equal zero to allow the delete to take place. If the calling task is a U-task, the Read and Write keys must also be compared against those specified by the user in the parameter block.

To release the file, the directory entry is marked inactive. If all of the entries within this directory block are now inactive, the directory block is released by a call to RELED. Otherwise, the directory block is just updated by PUTD.

For a contiguous file, the size of the allocation is computed, and the allocation is released by RELEB. The directory leaf is released by EVDIS and RS state is returned to by TMRSARS. The FCB allocated at the start of the DELETE function is released by RELEFCB, the bare disc is unassigned by CLO.BLK, and the routine returns to SV7.CMD.

For a chain file, the directory leaf is released right away, and each link in the chain is read into memory in order to compute the logical block address of the next block. As each block is read into memory, the link field is saved and then the block is freed via RELEB. The routine then returns to RS state and terminates as above.

If any errors are encountered during the DELETE processing, the appropriate error condition is set into the user's SVC 7 parameter block, RS state is returned to, the directory leaf is released and the routine returns to the user via TMRSOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: RENAME

ABSTRACT: The function of RENAME is to change the name of devices and direct-access files.

ENTRYS: RENAME

SOURCE LIBRARY ROUTINES: DCB, FCB, DIR

EXTRN: SV7.CMD, SVC7.ERR, TMRSOUT, TMRSRSA, TMRSARS, EVDIS, EVQCON, FDCHECK, FDCHECK1, DMTLOOK, DIRLOOK, GETD, PUTD

REGISTERS USED: U0 - UF

ON ENTRY:
RS STATE
U2 TCB ADDRESS
U5 SVC 7 PARAMETER BLOCK

ON EXIT:
RS STATE
U2 TCB ADDRESS
U5 SVC 7 PARAMETER BLOCK

PRINCIPLES OF OPERATION:

RENAME is an SVC 7 function, called directly from the SVC 7 driver routine, SV7. RENAME proceeds along two separate paths, depending upon whether the RENAME is being performed on a device or a direct-access device.

In both cases, the LU must be valid and currently assigned. To rename devices, the caller must be an E-TASK, the new device mnemonic must be syntactically correct, and the new name must not currently exist in the Device Mnemonic Table (DMT). If all these tests are completed successfully, the access privilege is checked to insure that the device is open for ERW. If so, it has exclusive access. The new device name is moved to the DMT and the routine returns to SV7.CMD.

If the RENAME function is being performed on a file, the File Descriptor is checked for syntax. The access privileges are checked to insure that the file is open for ERW. If so, it has exclusive access. The routine obtains control of the directory via an EVQCON and searches the directory for the new name. If the new name does not currently exist in the directory (i.e., is a valid new name), the directory block pertaining to this file is brought into memory and the new FD is moved to the directory entry. The FCB is also updated to contain the new file name. The directory is then rewritten, and control of it is released via an EVDIS. Return is directly to SV7.CMD.

If any of the above checks fail, the appropriate status is set in the user's parameter block and SVC 7 processing is terminated by returning directly to TMRSOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: REPRO

ABSTRACT: Change the protect keys associated with an LU.

ENTRYS: REPRO

SOURCE LIBRARY ROUTINES: TCB, FCB, DCB, DIR

EXTRN: LUCHECK, REN.EPCK, SV7.CMD, TMRSRSA, EVQCON, GETD, PUTD, TMRSARS, EVDIS

REGISTERS USED: U0-UF

ON ENTRY:
U2 - TCB address
U5 - SVC 7 parameter block pointer

ON EXIT:
U2 - TCB address
U5 - SVC 7 parameter block pointer

PRINCIPLES OF OPERATION:

The FCB/DCB address associated with the LU is obtained from the LU table via LUCHECK.

A reprotect may be directed to a device by an E-task only. The read and write keys are moved from the parameter block to the DCB; return is to SV7.CMD.

To reprotect a file, RSA state is entered (TMRSRSA), the directory leaf is connected to (EVQCON) and the required directory entry is obtained (GETD).

The new read and write keys are checked for validity; if the current keys are X'FF' and the calling task is not an E-task, the reprotect is not allowed.

Normal return is to SV7.CMD in RS state. Any errors encountered during REPRO are saved into the SVC 7 parameter block and the task is returned to via TMRSOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: CLOSE

ABSTRACT: Logically disconnect the assignment between a task and a file or device.

ENTRYS: CLOSE, CLO.BLK

SOURCE LIBRARY ROUTINES: DCB, FCB, DIR

EXTRN: EVQCON, EVDIS, TMRSRSA, TMRSARS, LUCHECK, RESET, GETD, RESET.CH, RELEFCB, PUTD

REGISTERS USED: U0 - UF

ON ENTRY: RS STATE
U2 - TCB Address
U5 - SVC 7 Parameter Block Address

ON EXIT: RS STATE
U2 - TCB Address
U5 - SVC 7 Parameter Block Address

PRINCIPLES OF OPERATION:

CLOSE is the SVC 7 function which removes the logical connection between a task and a file or device which was established at OPEN time. CLOSE is called directly from the SVC 7 driver routine, SVC7. The logical unit entry in the LU table is obtained via a call to LUCHECK; if it was not previously assigned, unassigned status is placed in the user's parameter block, and the routine returns via TMRSOUT. This entry is set to zero and the processing continues in one of two paths, depending upon whether the CLOSE is directed to a device or a file.

For the null and console device, the routine terminates immediately, exiting back to SV7.CMD. Otherwise, the read and write counts in the DCB are reset to reflect the disconnection and the return is taken to SV7.CMD.

For files, an EVQCON is used to obtain control of the directory and the required directory block is read into memory via a call to GETD.

For contiguous files, the current sector pointer is moved from the FCB to the directory. For chain files open for Write or Read/Write, the file is check-pointed via the routine RESET.CH and the first logical block address, last logical block address, current sector number and number of logical records is moved from the FCB to the directory. For all file types, the read and write counts in the directory and the FCB are reset. If the FCB read and write counts now equal zero, the FCB is released via a call to RELEFCB. Whether or not the FCB is released, the DCB read and write counts are updated and the directory is rewritten, via a call to PUTD. Control of the directory is released via EVDIS and return is directly to SV7.CMD.

CLO.BLK is an additional entry point to CLOSE to unassign the bare disc; it is also called directly from ALLO and DELETE, prior to their exiting to SV7.CMD.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: CAP

ABSTRACT: Change access privileges associated with a given LU.

ENTRYS: CAP

SOURCE LIBRARY ROUTINES: TCB, FCB, DCB, DIR

EXTRN: LUCHECK, EVQCON, EVDIS, TMRSARS, TMRSRSA, APCHECK , PUTD, GETD

REGISTERS USED: U0-UF

ON ENTRY: U2 - TCB address
U5 - SVC 7 parameter block address

ON EXIT: U2 - TCB address
U5 - SVC 7 parameter block address

PRINCIPLES OF OPERATION:

CAP is the SVC 7 executor invoked by SVC7 to change the access privileges associated with a given LU. The LU may be assigned to a device or a file.

CAP is a two pass operation. On pass one, all required error checking is performed. If the new access privilege is not allowed, a privilege error is saved into the task's SVC 7 parameter block, and the task is returned to via TMRSOUT.

If the requested access privilege change is valid, CAP proceeds to pass two. During pass two, all changes in the DCB (for a device) or the FCB and Directory (for files) are actually made. The following are the subroutines used internally within CAP, and a brief description of each:

(1) CHNORW - remove an access currently granted to the LU; U6 acts as a flag indicating which access to remove, U6 = 0 implies remove read; U6 = 4 implies remove write. U0 and U1 contain current Write and Read counts.

(2) CKRWCT - determines if new access is in violation of former access; U6 indicates desired new access; U0, U1 contain current read and write counts associated with the DCB/FCB. If exclusive access is requested of a privilege (either read or write) the routine insures that no more than one shared user is currently granted that privilege.

Once a new access privilege is valid, it is indicated in the appropriate control block via the read write count fields (DCB.WCNT, DCB.RCNT) for a device, (FCB.WCNT, FCB.RCNT, DIR.WCNT, DIR.RCNT) for a file.

CAP uses the directory management routines GETD and PUTD to read and write the required directory block. It uses the Task Management routines TMRSRSA and TMRSARS to switch between RS and RSA states, to allow SVC 1 calls from the file manager and it uses the Event Coordination Routines, EVQCON and EVDIS to connect to and release control of, the directory leaf.

CAP returns directly to SV7.CMD upon successful completion.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: CHECKPT

ABSTRACT: Checkpoint a file or device

ENTRYS: CHECKPT

SOURCE LIBRARY ROUTINES: FCB, DCB, DIR

EXTRN: LUCHECK, TMRSRSA, RESET.CH, EVQCON, PUTB, EVDIS, PUTD, TMRSARS, SV7.CMD

REGISTERS USED: U0-UF

ON ENTRY:
U2 - TCB address
U5 - SVC 7 parameter block address

ON EXIT:
U2 - TCB address
U5 - SVC 7 parameter block address

PRINCIPLES OF OPERATION:

CHECKPT is the SVC 7 executor used to checkpoint an LU. To checkpoint a device, the routine simply validates its LU via LUCHECK and issues an SVC 1 WAIT call to the device.

To checkpoint a file, the FCB associated with it is obtained from the LU table via LUCHECK. Then RSA state is entered via TMRSRSA. The state of a chain file is switched to READ, RANDOM via RESET.CH to flush any outstanding buffers. Then the bit map is connected to via EVQCON; the bit map is updated via PUTB and the bit map leaf is released via EVDIS.

For both Contiguous and Chain files, the directory leaf is connected to, and the directory block is read into memory via GETD. Current file information is then moved from the FCB to the directory. The directory is updated by PUTD, and the directory leaf is released. RS state is returned to via TMRSARS and the routine exits to SV7.CMD.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: FETCH

ABSTRACT: FETCH file/device attributes of a given FD.

ENTRYS: FETCH

SOURCE LIBRARY ROUTINES: FCB, DCB

EXTRN: LUCHECK, TMRSOUT

REGISTERS USED: U0-UF without Save/Restore

ON ENTRY: U2 - TCB address
U5 - SVC 7 parameter block address

ON EXIT: U2 - TCB address
U5 - SVC 7 parameter block address

PRINCIPLES OF OPERATION:

FETCH is called directly from the SVC 7 driver routine, SVC 7. The attributes, logical record length and device code field of the DCB/FCB are moved to the user's parameter block. For files, the file name and size are obtained from the FCB and moved to the user's parameter block. For devices the device name is obtained from the DMT and moved to the user's parameter block. Exit from FETCH is to TMRSOUT.

If the Logical Unit the FETCH is being performed upon is unassigned, FETCH will exit to the SVC 7 error routine to issue a X'09' error code and exit immediately to TMRSOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

NAME: CONTIG

ABSTRACT: Intercept all data transfers to a contiguous file.

ENTRYS: CONTIG, CTG.EP

SOURCE LIBRARY ROUTINES: FCB, DCB, TCB

EXTRN: TMRSRSA, TMRSAOUT, IODONE2

REGISTERS USED: U0-UF

ON ENTRY:
U2 - TCB address
U3 - FCB address
U4 - Task SVC 1 parameter block address
U5 - data start address
U6 - data end address

ON EXIT:
U7 - random address

On Exit: None

PRINCIPLES OF OPERATION:

CONTIG receives control from SVC1 (IOSET) when it intercepts an SVC1 data transfer call to a contiguous file.

The length of the data transfer request is computed and the random address is obtained either from the FCB random address (for a random I/O request) or from the FCB current sector pointer (for a sequential I/O request). CONTIG copies the FCB information into the DCB and if the I/O request is a read or write, CONTIG exits by transferring control directly to the disc driver. If the I/O request is a test and set (both read and write bits set in the SVC 1 function code), CONTIG enters RSA state, moving the RS save area to a save area in the FCB via TMRSRSA, and then modifies the following fields in the TCB:

TCB.RPSW - the location field of the resume PSW is set up to contain a secondary entry point within CONTIG

TCB.RGPR - the general purpose register save area is set up to contain the current values of user register set.

The routine then transfers control to the disc driver for the read portion of the test and set operation. By modifying the TCB.RPSW and TCB.RGPR fields as specified above, upon termination of the read the disc driver returns to CONTIG at its secondary entry point. CONTIG then processes the remainder of the test and set operation itself. If a write is to be performed (the first halfword of the buffer read contained a X'0000'), CONTIG does the write by issuing an SVC 1 WRITE, WAIT call. Control is returned to the calling task upon successful completion of CONTIG via the task management routine, TMRSAOUT. If CONTIG receives an EOM status following an I/O operation, EOM status is saved in the FCB and control is returned to the task by branching to IODONE2 to complete the request.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: CMD.CO

ABSTRACT: Intercept and Execute all commands to a contiguous file

ENTRYS: CMD.CO

SOURCE LIBRARY ROUTINES: FCB, DCB

EXTRN: TMRSRSA, TMRSAOUT, IODONE2

REGISTERS USED: U0 - UF

ON ENTRY:
RS STATE
U2 - TCB Address
UD - FCB Address

ON EXIT:
RS/RSA STATE
U2 - TCB Address
UD - FCB Address

PRINCIPLES OF OPERATION:

The following are the executors contained in CMD.CO:

Rewind (CMD.REW) - Set current sector (FCB.CSEC) in the FCB to 0 and return via a branch to IODONE2.

Backspace Record (CMD.BSR) - Decrement FCB.CSEC by 1, enter RSA state and issue an SVC 1 read of new current sector to check for any I/O problems. Exit to TMRSAOUT.

Forward Space Record (CMD.FSR) - Increment FCB.CSEC by 1 and proceed as in CMD.BSR.

Write End of File (CMD.WEOF) - Increment FCB.CSEC by 1, enter RSA state and write a pseudo-file mark (X'1313') at that random address via an SVC 1 WRITE, WAIT call. Exit via TMRSAOUT.

Forward Space File (CMD.FSF) - Enter RSA state and issue SVC 1 read commands starting at FCB.CSEC, until a pseudo-file mark, X'1313' is found. Exit to TMRSAOUT.

Backward Space File (CMD.BSF) - ame as Forward Space File, except the X'1313' is searched for starting at FCB.CSEC and backing up one sector at a time. Exit to TMRSAOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: CHAIN

ABSTRACT: Intercept SVC 1 data transfer requests from the SVC 1 First Level Interrupt Handler (FLIH) which are directed to a chain file.

ENTRYS: CHAIN

SOURCE LIBRARY ROUTINES: TCB, FCB, DCB

EXTRN: RESET.CH, POSITN, GETCHL, PUTCHL, TMRSAOUT

REGISTERS USED: U0-UF

ON ENTRY:
U2 - TCB address
U3 - FCB address
U4 - SVC 7 parameter block address
UD - DCB address

ON EXIT:
U2 - TCB address
U3 - FCB address
U4 - SVC 7 parameter block address
U7 - DCB address

PRINCIPLES OF OPERATION:

CHAIN recieves control from the FLIH when an SVC 1 call is directed to a chain file. The function code is obtained from the FCB and the I/O operation desired is decoded (Read or Write, Random or Sequential access).

If the call is random, the current logical record is updated with the random address. The block and offset within that block of the logical record being processed are computed by the subroutine, CCH.BKOF, and the FCB is updated to contain the new current block and offset. If this requires repositioning of the file, RESET.CH is called first to flush any outstanding buffers and to switch the file to a known state (READ, RANDOM). Then RESET.CH is called again to switch to the mode requested.

If the I/O to be performed is a read, CHAIN transfers control directly to GETCHL, to perform the next logical record read. Otherwise, control is transferred to PUTCHL, to perform the next logical record write.

If a read is requested of FCB.NLR+1 (a logical record beyond the end of the file), or a write of FCB.NLR+2 (a logical record more than one record beyond the end of the file) an EOM status is generated and saved in the task's SVC 7 parameter block. Exit is directly back to the task via TMRSAOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

NAME: CMD.CH

ABSTRACT: Intercept SVC 1 command requests from the SVC1 First Level Interrupt Handler (FLIH) which are directed to a chain file.

ENTRYS: CMD.CH

SOURCE LIBRARY ROUTINES: FCB

EXTRN: POSITN, TMRSAOUT

REGISTERS USED: U0-UF

ON ENTRY:
U2 - TCB address
U3 - FCB address
U4 - SVC 7 parameter block address
UD - DCB address

ON EXIT: None

PRINCIPLES OF OPERATION:

CMD.CH is entered from the FLIH when a SVC 1 command is directed to a chain file. CMD.CH contains 5 executors, containing the logic necessary to perform the following functions: Rewind, Backspace Record, Forward Space Record, Backspace Filemark, and Forward space Filemark. If any other commands are attempted, the routine ignores them by returning directly to the task via TMRSAOUT.

Before decoding the command, CMD.CH sets up the file manager LU in the TCB (TCB.FMLU) and resets the file to Read Random Mode via RESET.CH. The following executors perform the

1. CMD.REW - The file is rewound by positioning the file to block 0 by calling POSITN with a zero argument. The current logical record is set to zero (FCB.CLRL) and the offset within that block is set to 4 (FCB.COFF).

2. CMD.BSR - If the file is currently positioned at the start, an EOM status is placed in the SVC 7 parameter block. Otherwise, the block and offset of the previous logical record (FCB.CLRL-1) is computed and the file is positioned t the appropriate block. Then the current logical record (FCB.CLRL), a d current offset (FCB.COFF) are updated.

3. CMD.FSR - To forward space one record, a test is made to see if the file is currently positioned at the end. If it is, an EOM status is returned as above. Otherwise, the block and offset of the next logical record are computed (FCB.CLRL+1) and the routine proceeds as above.

4. CMD.FSF - To position the file at the end, POSITN is called with an argument of FCB.NBLK-1. Then the number of logical records is set as the current logical record (FCB.NLR → FCB.CLRL) and the offset is computed and saved.

5. CMD.BSF - Identical to CMD.REW.

All executors return to the calling task via TMRSAOUT.

The subroutine CCH.BKOF is used within many of the chain file handling routines to compute the block number a given logical record begins in and its offset within that block.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

NAME: GETCHL

ABSTRACT: The purpose of GETCHL is to move a logical record from a system buffer to the user's buffer. This routine performs no physical I/O and is only used for buffered chain files.

ENTRYS: GETCHL

SOURCE LIBRARY ROUTINES: FCB

EXTRN: TMRSAOUT, SET.LRCL, GETCHPR

REGISTERS USED: U1, U2, U3, U4, U7, U8, UC

ON ENTRY:
RSA STATE
U2 TCB pointer
U3 FCB pointer
U4 User parm block pointer
U7, UD DCB pointer

ON EXIT:
RSA STATE
U2 TCB pointer
U3 FCB pointer
U4 User parm block pointer
U7 DCB pointer

PRINCIPLES OF OPERATION:

When a task requests a read to a chain file, logical records must be moved from a system buffer to the user's buffer. GETCHL performs this movement of data.

The data request transfer length is established via a call to SET.LRCL. Data is then moved, one byte at a time, from the system to the user buffer. If a system buffer is exhausted before the data transfer is complete, a new buffer must be obtained via a call to GETCHPR.

The current logical record number in the FCB is incremented by 1 after the data transfer is completed; exit is to TMRSAOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: PUTCHL

ABSTRACT: Move the current logical record from the system buffer to its preallocated disc location.

ENTRYS: PUTCHL

SOURCE LIBRARY ROUTINES: FCB

EXTRN: SET.LRCL, PUTCHP, TMRSAOUT

REGISTERS USED: U0 - UF

ON ENTRY: U2 - TCB address
U3 - FCB address
U4 - SVC 1 parameter block address
U7 - DCB address

ON EXIT: None

PRINCIPLES OF OPERATION:

PUTCHL receives control directly from CHAIN, to complete the SVC 1 operation to a chain file. No physical I/O is performed by PUTCHL. The record is moved, byte by byte, from the system buffer to the disc file until a complete record is moved. The length of the move operation is the smaller of the size of the logical record, or the length requested in the parameter block.

If a block is filled before the whole record has been moved the block is physically written to the disc by the routine PUTCHP. When the data transfer is complete, the FCB is updated by incrementing the FCB current logical record number.

If the file is in sequential mode, the current logical record number is compared with the number logical records (FCB.CLRL: FCB.NLR).

If the current is greater than or equal to the number of logical records, the number of logical records is updated with the value of the current logical record (FCB.CLRL → FCB.NLR).

The routine returns control to the user via TMRSAOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: SET.LRCL

ABSTRACT: Set up various parameters to be used by GETCHL and PUTCHL to move data between system and task buffers.

ENTRYS: SET.LRCL

SOURCE LIBRARY ROUTINES: FCB

EXTRN: None

REGISTERS USED: U3, U4, U6, U9, UE, UF

ON ENTRY:
U3 - DCB address
U4 - SVC 1 parameter block address

ON EXIT:
U3 - DCB address
U4 - SVC 1 parameter block address
U6 - logical record length
UE - system buffer address
UF - system parameter block address

PRINCIPLES OF OPERATION:

SET.LRCL performs the following functions:

Computes the length of the data transfer by comparing the length requested with the logical record length; the smaller of the two is returned in U6.

The current logical record number, parameter block pointer and buffer pointer are picked up from the FCB and returned in registers.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: CHN.WAIT

ABSTRACT: Perform an SVC 1 Wait call on the File Manager LU

ENTRYS: CHN.WAIT

SOURCE LIBRARY ROUTINES: None

EXTRN: None

REGISTERS USED: UC

ON ENTRY: None

ON EXIT: None

PRINCIPLES OF OPERATION:

CHN.WAIT is used by the file management routines to perform a WAIT call on the file manager LU.

This information is proprietary and is supplied by INTERDATA for the purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: GETCHPR

ABSTRACT: Perform physical reads to a chain file. The length of the I/O is in n multiples of 1 or more sectors, where n is the blocksize of the file being processed.

ENTRYS: GETCHPR, GETCHPL

SOURCE LIBRARY ROUTINES: FCB

EXTRN: TMRSAOUT

REGISTERS USED: U0 - UF

ON ENTRY:
U3 - FCB address
U4 - SVC 1 parameter block address
U5 - cannot be used; contains block, offset information
U7 - DCB address
UE - System buffer address
UF - System parameter block address

ON EXIT:
U3 - FCB address
U4 - SVC 1 parameter block address
U5 - cannot be used; contains block, offset information
U7 - DCB address
UE - System buffer address
UF - System parameter block address

PRINCIPLES OF OPERATION:

This routine is called whenever a new physical block is to be read into memory. It has two entry points: the routine is entered at GETCHPR if the file is to be read to the right (FCB.NBLK+n); it is entered at GETCHPL if the file is to be read to the left (FCB.NBLK-n). If the file is in sequential mode, one block is read. If the file is in random mode, the Read process is continued until the required block is read in.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: PUTCHP

ABSTRACT: Perform physical writes to a chain file. The length of the data transfer is in n multiples of 1 sector, where n is the blocksize of the file.

ENTRYS: PUTCHP

SOURCE LIBRARY ROUTINES: FCB

EXTRN: EVQCON, EVDIS, GETSECTR, TMRSAOUT, RELEB, CHDIR

REGISTERS USED: U0 - UF

ON ENTRY: U3 - FCB address
U4 - User parameter block address
U7 - DCB address

ON EXIT: None

PRINCIPLES OF OPERATION:

PUTCHP is invoked whenever a physical block is to be written to disc. In random mode the block is written via an SVC 1 WRITE, RANDOM, WAIT call. Upon successful completion of the I/O, the routine returns to the task via TMRSAOUT.

In sequential mode, a new block of sectors is obtained on the disc via GETSECTR prior to issuing the SVC1. The SVC1 WRITE, RANDOM, PROCEED is issued on the current buffer; then the current and alternate buffers are switched. The FCB.NBLK and FCB.CBLK fields are updated to include the new block allocation.

If an I/O error occurs, the I/O status is set into the task's SVC 1 parameter block and the task returned to.

If an EOM status on the disc is returned from GETSECTR, the routine must set the file into a known state; i.e., a state that will allow the file to be closed, or deleted. Also, if space on the disc is subsequently available, the file should be in a state that the write sequential operation could continue. This is accomplished by computing which block the last sector in the file ends in. Once it is determined that FCB.NLR+1 starts in the current block, this has been accomplished. Any sectors deleted from the file are released via RELEB.

The backing up procedure is accomplished by using the routine CHDIR to process the file to the left. Once the file has backed up far enough, CHDIR is called again to change the processing of the file back to the right.

The routine sets an EOM status in the SVC1 parameter block and returns directly to the task via TMRSAOUT.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: POSITN

ABSTRACT: Perform physical positioning of a chain file

ENTRYS: POSITN

SOURCE LIBRARY ROUTINES: FCB

EXTRN: TMRSAOUT, CHDIR, GETCHPL, GETCHPR

REGISTERS USED: U0 - UF

ON ENTRY:
U3 - FCB address
U4 - SVC1 parameter block address
U7 - DCB address
U5 - Required Block

ON EXIT: U3-U7 as above

PRINCIPLES OF OPERATION:

The purpose of POSITN is to set up the FCB to point to a specific block in the file. The routine is entered with a block number in U5. POSITN moves the file either to the right or left, until the current block (FCB.CBLK) equals this required block.

To move the FCB.CBLK pointer, the routine performs SVC1 Read WAIT calls on the file. If the file is to be positioned at its start (0 →FCB.CBLK) or its end (FCB.NBLK-1→ FCB.CBLK), only one Read is necessary. Otherwise, the routine determines which direction from the current block the required block is. Then CHDIR is called, if needed, to set the processing mode into the proper direction. Subsequent blocks are Read, until the required and current blocks are equal. The routine always changes the direction back to the right, if necessary, before returning.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing

NAME: CHDIR

ABSTRACT: The purpose of CHDIR is to change the direction by which a given chain file is processed.

ENTRYS: CHDIR

SOURCE LIBRARY ROUTINES: FCB, DCB

EXTRN:

REGISTERS USED: U1, U3, U8, U9, UA, UE, UF

ON ENTRY:
U1 - linkage
U2 - return linkage for higher level subroutines
U4 - user parm block pointer
U5 - argument for higher level subroutines
UA - argument to CHDIR

ON EXIT:
U3 - FCB address

U2, U3, U4, U5 as above

PRINCIPLES OF OPERATION:

A chain file can be processed in either direction; i.e., from any position in the file, processing may continue either to the right or to the left. This routine is called whenever the processing of a file is to be changed. Changing direction of the processing of a file is allowed only when the file is in the read random mode.

When the direction is changed, the previous and current block pointers in the FCB are switched to indicate the new direction of the file. Also the new direction is indicated by setting (for left) or resetting (for right) the appropriate bit in the FCB.FLGS field.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: RESET.CH

ABSTRACT: Change the current state of a chain file.

ENTRYS: RESET.CH

SOURCE LIBRARY ROUTINES: FCB

EXTRN: TMRSAOUT

REGISTERS USED: U1, U3, U4, U7-UF

ON ENTRY:
U0 - Reserved; cannot be used
U1 - New state of file
U3 - FCB address
U4 - SVC 7 parameter block pointer
U7 - DCB address

ON EXIT:
U3, U4, U7 as above

PRINCIPLES OF OPERATION:

The purpose of RESET.CH is to switch state. The state of a chain file is defined to be both its operation (READ or WRITE) and its access method (RANDOM or SEQUENTIAL). Each mode change is discussed separately as follows: (The state of a file is indicated by the FCB.FLGS field).

From Read Random To:

(1) Read Random - Return
(2) Read Sequential -
  (a) If at last block, set mode to sequential and return.
  (b) Otherwise issue Read, Random, Proceed to alternate buffer, reading in link of current block. Then set mode to sequential and return.
(3) Write Random - Set operation to Write and return.
(4) Write Sequential - Set state to write sequential and return.

From Read Sequential To:

(1) Read Random - Issue SVC 1 WAIT call on alternate buffer. If I/O completes successfully, the mode is set to Random and the routine returns. If an I/O status is returned, this status is stored in the task's parameter block and the calling task is returned to directly via TMRSAOUT.
(2) Read Sequential - Return
(3) Write Random - Set operation to write and issue SVC 1 WAIT on alternate buffer. Proceed as Read Random above.
(4) Write Sequential - Set operation to write and issue SVC 1 WAIT on alternate buffer. Proceed as above if an I/O error is encountered. Otherwise, set mode to sequential; switch current buffer to alternate and set the last logical block of the file as the current (FCB.LLBA → FCB.CSEC) and total number of logical records to current (FCB.NLR → FCB.CLRL), thus positioning the file at the end.

An SVC 1 READ, RANDOM, WAIT call is issued with a random address of the last logical block in the file. If the status returned is good, the routine returns. Otherwise, the I/O status is saved and exit is via TMRSAOUT.

From Write Random To:

(1) Read Random - If the current buffer modify bit is set (indicating a buffer needs flushing) the current buffer is written via an SVC1 WRITE, WAIT, RANDOM call. If an I/O error occurs, the I/O error status is saved and the routine exits via TMRSAOUT. Whether or not the above I/O is necessary, the operation is set to Read and the routine returns to the caller.
(2) Read Sequential - Perform all the above functions as in Write Random to Read Random switch. Then, if the file is currently positioned at the end (FCB.CSEC = FCB.LLBA) the mode is set to sequential and the routine returns.

  If the file is not currently positioned at the end, the next block in the file is read into the alternate buffer via an SVC 1 READ, RANDOM, PROCEED call. Then the mode is set to sequential and RESET.CH returns.
(3) Write Random - Return
(4) Write Sequential - Again, the current buffer is flushed, if necessary. Then the file is switched to sequential mode and the routine positions the file at the end by setting the last block in the file as the current block (FCB.LLBA FCB.CSEC) and the last logical record as the current (FCB.NLR FCB.CLRL). The last block is read into a system buffer and, upon successful completion of the I/O, the routine returns. An I/O error causes a return to the task via TMRSAOUT, as described above.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: RESET.CH (continued)

ABSTRACT:

ENTRYS:

SOURCE LIBRARY ROUTINES:

EXTRN:

REGISTERS USED:

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

From Write Sequential to all states: (except Write Sequential):

The current buffer is written via SVC1 WRITE, RANDOM, WAIT call, and the alternate buffer is written via SVC1 WRITE, RANDOM, PROCEED call.

Then to:

1. Read Random - Set Read Random and return
2. Read Sequential - Set Read and return
3. Write Random - Set Random Mode and Return
4. Write Sequential - return

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: PUTD

ABSTRACT: Write Directory Block

ENTRYS: PUTD

SOURCE LIBRARY ROUTINES: DCB

EXTRN: PUTD

REGISTERS USED: UC, UE, UF

ON ENTRY: U7 - DCB address

ON EXIT: U7 - DCB address

PRINCIPLES OF OPERATION:

PUTD issues an SVC1 WRITE, RANDOM and WAIT call to the current directory block. If an I/O error is returned from this SVC1 call, the system crashes X'303'.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: PUTB

ABSTRACT: Write a modified Bit Map Sector

ENTRYS: PUTB

SOURCE LIBRARY ROUTINES: DCB

EXTRN: PUTB

REGISTERS USED: UC, UE

ON ENTRY: U7 - DCB address

ON EXIT: U7 - DCB address

PRINCIPLES OF OPERATION:

PUTB tests the bit map modify flag to see if the bit map has been modified. If not, the bit map has not been modified since the last I/O operation to the bit map, and PUTB returns directly to the caller. If the bit map has been modified, an SVC1 WRITE, RANDOM and WAIT call is issued. If an I/O error status is returned from the SVC1, GETB crashes the system with a X'301' crash code.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: TMRSRSA, TMRSARS

ABSTRACT: Switch between RS and RSA state to allow I/O via SVC1 calls from the file manager.

ENTRYS: TMRSRSA; TMRSARS

SOURCE LIBRARY ROUTINES: TCB

EXTRN: None

REGISTERS USED: U6-UF, U2

ON ENTRY: U6 - points to alternate save area in DCB or FCB (entry to TMRSRSA)
U2 - TCB address

ON EXIT: None

PRINCIPLES OF OPERATION:

TMRSRSA is called by any file management routine which is about to issue an SVC 1 I/O call. TMRSRSA moves the 16 general purpose registers saved in TCB.RGPR to an alternate save area in the DCB or FCB, thus allowing the TCB save area to be reused by the SVC 1 processor. It sets the RS state bit in the TCB status field to indicate that this has occurred. TMRSARS is called when the file management routine has completed the I/O and wants to return to RS state. The registers are moved from the alternate save area to the TCB and the alternate save area bit in the TCB status field is reset.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: OS/32 ST Floating Point Trap

ABSTRACT: This routine simulates all floating point instructions.

ENTRYS: FLTP.S00

SOURCE LIBRARY ROUTINES: None

EXTRN: IIH

REGISTERS USED: E8, E9, EA, EB, EC, ED, EE, EF. All floating point registers.
Index registers.

ON ENTRY: EE-EF = PSW in user program
All floating point registers: input values
Index register: index value to compute effective address

ON EXIT: All floating point registers = output values
EE-EF = PSW of next instruction in user program

NOTE: All references to condition code in this MDF refer to the condition code in user program after executing the input floating point instruction.

PRINCIPLES OF OPERATION:

This routine is entered via an illegal instruction trap and in NS state. When entered, the location counter of user PSW is picked up. If the user program is in halfword mode, the upper halfword of the location counter is masked off. The opcode and R1, R2 fields are then picked up. The opcode is then used to index into VECTAB1.

- If it is not a floating point instruction but a real illegal one, control exits to IIH.
- If the user program is in halfword mode and the opcode shows a floating point instruction not supported in HW mode, control exits to IIH, illegal instruction handler.
- If it is a FLR instruction, control goes to FLR.1 routine.
- If it is a FXR instruction, control goes to FXR routine.
- If it is a RR floating point instruction other than FLR and FXR, control goes to first level handler XER routine.
- If it is a RX1, RX2 or RX3 floating point instruction, control goes to first level handler XE routine.

In XER, R1 and R2 fields are converted into the real address of floating point registers. The proper second level interrupt handler is entered. In return, condition code is checked to see if overflow occurred. If overflow did occur and arithmetic fault interrupt is enabled, control exits to arithmetic fault interrupt handler. Otherwise, control returns to the user program.

In XE, the displacement is picked up. Checks are made to see the instruction format. Note: only RX1 format is supported in halfword mode.

- If the instruction is in RX1, the displacement will be added to the index value, if any. Control goes to second interrupt handler.
- If it is in RX2, the displacement is added to the location counter. Indexing is then checked and effective address is computed. Second interrupt handler is then entered.
- If it is in RX3, the displacement is added to the first index value and then the second index value. Control then goes to second interrupt handler.

NSU state is entered when indexing in order to obtain the value in index registers.

In FXR, the floating point R2 is converted to its real address, the floating number is then picked up. Checks are made to see if the number is too big or too small to be an integer.

- If the float number has no integer part, a zero value is returned to the user.
- If the float number is too large to be expressed as an integer, X'7FFFFFFF' is returned for positive number. X'10000001', the 2's complement of X'7FFFFFFF' is returned for negative number.
- If the float number is neither too large or too small, the magnitude is truncated if necessary.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: OS/32 ST Floating Point Trap (continued)

ABSTRACT:

ENTRYS:

SOURCE LIBRARY ROUTINES:

EXTRN:

REGISTERS USED:

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

In ME, the condition code is cleared first. If either operand is zero on entrance, ME branches immediately to STA, with final result equal to zero. Otherwise, the exponents and signs are added. The program then tests for carry out of the exponent field. This information may be determined from the status of the condition code C and V bits as follows:

Let C_S = carry from the sign field (bit 0)

C_E = carry from the exponent field (bit 1) into sign field.

By definition, $V = C_S \oplus C_E$ ($\oplus$ exclusive or)

and $C = C_S$

Therefore, $C_E = C_S \oplus C_E \oplus C_S = V \oplus C$

If there was a carry from the exponent field, the routine goes to ME1.

The original exponents were in excess 64 form, so 64 must be subtracted from the new exponent for correction. However, if the most significant bit of the exponent is equal to 1, the subtraction will not be sufficient to compensate for the carry. Therefore, exponent overflow has taken place. If there was no carry from the exponent field, the reverse is true. Unless the most significant bit of the exponent is equal to 1, exponent underflow will take place when the excess 64 correction takes place. This condition is tested at ME2. The excess 64 correction is performed at ME3. The magnitudes are left-shifted by 7 bits for best precision. Before rounding, it is necessary to determine whether the result must be adjusted by 2 or 6 bits. This determination is made by testing the most significant 4 bits of the final product. The result is then rounded in the appropriate place (the bit that will be least significant after adjusting). If the result required adjustment by 2 bits only, rounding might have caused a carry. This condition is tested. If it is true, 6 bits will be shifted, otherwise, 2 bits will be shifted. One will be subtracted from the final exponent for correction.

In DE, the user condition code is set to zero. The second operand is tested for zero. If it is zero, DIV.BY.0 sets the condition code to 12. If the first operand is zero, a zero result is returned. The exponent and sign of the second operand are then subtracted from the exponent and sign of the first operand. Carry out of the exponent field is tested as described above. In this case, 64 must be added to the result exponent in order to restore excess 64 notation. Therefore, if carry from the exponent has taken place, the result exponent is tested at DE.1 for an underflow condition. If this is true, UNDERFLO is entered. If there was no carry, the result exponent is tested at DE2 for overflow condition. If this is true, OVERFLO2 is entered. OVERFLO2 is entered instead of OVERFLO because the sign bit is already good at this point. If there was no exponent overflow or underflow, DE.3 is entered where excess 64 correction takes place. The divisor is shifted left 6-bits for best precision. Before rounding, the result is tested to determine whether it must be adjusted by 2 or 6 bits. This determination is made by testing the most significant 4 bits of the final product. The result is then rounded in the appropriate place (the bit that will be least significant after adjusting). If the result required adjustment of 2 bits only, rounding might have caused a carry. This condition is tested. If it is true, 6 bits will be shifted. Otherwise, 2 bits will be shifted. One will be added to the final exponent for correction.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

NAME: OS/32 ST Floating Point Trap (continued)

ABSTRACT:

ENTRYS:

SOURCE LIBRARY ROUTINES:

EXTRN:

REGISTERS USED:

ON ENTRY:

ON EXIT:

PRINCIPLES OF OPERATION:

The second interrupt handlers comprises FLR2, LE, AE, SE, ME, DE, CE, STME, LME, STE, which are described by their labels.

In FLR.2, the user CC is first set to zero. Checks are made to the integer number in the general register.

- If the integer is zero, program goes to store which is in LE handler.
- If the integer is positive or negative, the initial exponent is set to X'46' or X'C6' correspondingly. The integer value is then normalized to 6 digits. Program then goes to store routine which is in LE handler.

In LE, the user CC is first set to zero, the datum to be loaded is checked. If it is normalized already, further checks are put to the datum and the resulting condition code is set. If it has not been normalized, magnitude is checked for zero. If it is, the result is set to zero. If it is not, the float number is then in normalization process. If underflow is produced in the process, the V flag is set. Otherwise, the condition code is set to L, G, or not set.

In STE, the number is picked up from the register and stored in memory. The condition code from the old PSW is used without change.

In CE, the user CC is cleared, the source number A and destination number B are picked up. They are compared by subtraction. C and L bits are set if A < B. Otherwise, if B > A, G bit is set.

The add and subtract routines are the same with the exception that the floating-point subtract reverses the sign of the second operand upon entry. At AESE, the common process of AE and SE, the exponents of the two operands are picked up and tested for equality. If the two exponents are not equal, the smaller operand must be shifted to the right by 4 bits times the exponent difference. However, if the exponent difference is 6 or greater, no addition need take place and the larger operand is returned. If the exponents were equal, the magnitudes are compared, and a branch is taken to AGB (A greater than B) or BGA (B greater than A) depending on the result. These points are also entered following operand adjustment in case of exponent difference. At AGB, the signs of the operands are checked to see whether an add or subtract is required. If a subtract is required, the 'B' magnitude is subtracted from the A magnitude. A zero result returns zero to the result. A non-zero result goes to LE.4A to be normalized. The same thing occurs at BGA, except that the A magnitude is subtracted from the B magnitude. If an effective add is required, both BGA and AGB transfer control to AESE.2, the larger exponent will be used as the initial value of the final exponent. The magnitude of B is added to that of A, and the result is checked to see if a post-add adjustment is required. If not, a return is made to LE.5. If a carry occurred, the sum is shifted right 4 bits and at AESE.2B, the exponent is incremented to compensate for the shift. If the exponent overflowed, the sign bit is reversed, thus compensating for overflow into the sign, and then the result is set to X'7FFFFFFF' or its two's complement form depending on the sign of the result, and exits.

This information is proprietary and is supplied by INTERDATA for the sole purpose of using and maintaining INTERDATA supplied equipment and shall not be used for any other purpose unless specifically authorized in writing.

PUBLICATION COMMENT FORM

Please use this postage-paid form to make any comments, suggestions, criticisms, etc. concerning this publication.

From _____ Date _____

Title _____ Publication Title _____

Company _____ Publication Number _____

Address _____

_____

_____

FOLD FOLD

Check the appropriate item.

☐ Error Page No. _____ Drawing No. _____

☐ Addition Page No. _____ Drawing No. _____

☐ Other Page No. _____ Drawing No. _____

Explanation:

FOLD FOLD

CUT ALONG LINE

Fold and Staple

No postage necessary if mailed in U.S.A.

STAPLE

STAPLE

FOLD

FOLD

FIRST CLASS
PERMIT No. 22
OCEANPORT, N. J.

BUSINESS REPLY MAIL

NO POSTAGE NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY:

INTERDATA®

Subsidiary of PERKIN-ELMER
Oceanport, New Jersey 07757, U.S.A.

TECH PUBLICATIONS DEPT. MS 229

FOLD

FOLD

STAPLE