PHILIPS

PTS 6800 TERMINAL SYSTEM

User Library

PTS 6800 ASSEMBLER PROGRAMMER'S REFERENCE MANUAL

Part 2

Module M06

Data
Systems

Date : December 1977

Code : 5122 993 42031

MANUAL STATUS SURVEY

Module M06, Part 2 "PTS 6800 ASSEMBLER PROGRAMMER'S REFERENCE MANUAL"

This issue comprises following updates :

- U1.42031.0578 (May 1978; complete revision for Release 8.1)
- U2.42031.0778 (July 1978)
- U3.42031.0878 (August 1978)

PREFACE

The Assembler Programmer's Reference Manual provides the information required to write, process and test Assembler application programs for the PTS 6800 computer used in the Philips PTS 6000 Terminal System.

Information is divided into three parts as follows:

Part 1 : Assembler Language
Additional functions
Recommended techniques

Part 2 : Monitor requests
I/O drivers
TOSS utilities

Part 3 : Assembler processor
TOSS system start
Assembler debugging program

Parts 1 and 2 contain the information needed to write an Assembler program. Part 3 contains the information needed to process and test an Assembler program.

Processing of Assembler programs (updating, assembling, etc.) is done under 6800 System Software. The use of those parts of DOS 6800 System Software designed specifically for Assembler programs is described in Part 3 of this Manual. Information concerning the use of the general purpose components of DOS 6800 System Software is contained in the DOS 6800 System Software PRM (M11). Readers of the present Manual are expected to be familiar with the contents of the DOS 6800 System Software PRM.

The testing and production running of Assembler application programs is done under TOSS System Software. Information concerning TOSS System Software which is relevant to the writing, testing and running of Assembler application programs is included in Part 2 of the present Manual.

CONTENTS

	Date	Page
PREFACE	May 1978	0.0.0
1. TOSS MONITOR		
1.1. Introduction	Dec. 1977	1.1.1
1.2. General Description	Dec. 1977	1.2.1
	May 1978	1.2.2
1.3. Interrupt Handlers	Dec. 1977	1.3.1
1.4. LKM Processors	Dec. 1977	1.4.1
1.5. I/O drivers	Dec. 1977	1.5.1
	Dec. 1977	1.5.2
1.6. The Dispatcher	Dec. 1977	1.6.1
	Dec. 1977	1.6.2
	Dec. 1977	1.6.3
1.7. Tables	Dec. 1977	1.7.1
1.8. Other Monitor Programs	Dec. 1977	1.8.1
1.9. TOSS System Software	Dec. 1977	1.9.1
2. LKM REQUESTS		
2.1. Introduction	May 1978	2.1.1
	Dec. 1977	2.1.2
2.2. LKM Request Reference	Dec. 1977	2.2.1
Switch Tasks	Dec. 1977	2.2.2
Normal I/O	May 1978	2.2.3
I/O and Activate	Dec. 1977	2.2.4
Wait	Dec. 1977	2.2.5
Exit	Dec. 1977	2.2.6
Restart	Dec. 1977	2.2.7
Activate	Dec. 1977	2.2.8
Pause	Dec. 1977	2.2.9
Delay	Dec. 1977	2.2.10
Delay and Activate	Dec. 1977	2.2.11
Get Buffer	Dec. 1977	2.2.12
	Dec. 1977	2.2.13
Release Buffer	Dec. 1977	2.2.14
Load Segment	Dec. 1977	2.2.15
Abort	Dec. 1977	2.2.16
Get Time	Dec. 1977	2.2.17
Set Time	Dec. 1977	2.2.18
Assign File Code	May 1978	2.2.19
	May 1978	2.2.20
Assign Index	May 1978	2.2.21
	May 1978	2.2.22
3. I/O DRIVERS		
3.1. Introduction	Dec. 1977	3.1.1
3.2. Drivers for Teller Terminal Printer	Dec. 1977	3.2.1
3.3. Drivers for Data Communication	Dec. 1977	3.3.1
3.4. General Interface Rules	Dec. 1977	3.4.1
	Dec. 1977	3.4.2
	May 1978	3.4.3

	Date	Page
3.5. Driver Reference	May 1978	3.5.1
DRCR01	May 1978	3.5.2
	May 1978	3.5.3
DRDC15	May 1978	3.5.4
	May 1978	3.5.5
	May 1978	3.5.6
	May 1978	3.5.7
	May 1978	3.5.8
	May 1978	3.5.9
DRDC17	May 1978	3.5.10
	May 1978	3.5.11
	May 1978	3.5.12
	May 1978	3.5.13
	May 1978	3.5.14
DRDI01	May 1978	3.5.15
	May 1978	3.5.16
	May 1978	3.5.17
DRDN	May 1978	3.5.18
	May 1978	3.5.19
DRDU01	May 1978	3.5.20
	May 1978	3.5.21
	May 1978	3.5.22
DRDY01	May 1978	3.5.23
	May 1978	3.5.24
	May 1978	3.5.25
	May 1978	3.5.26
	May 1978	3.5.27
	May 1978	3.5.28
DRFD01	May 1978	3.5.29
	May 1978	3.5.30
	May 1978	3.5.31
	May 1978	3.5.32
	May 1978	3.5.33
	May 1978	3.5.33A
	May 1978	3.5.34
	May 1978	3.5.35
	May 1978	3.5.36
	May 1978	3.5.37
DRGP01	May 1978	3.5.38
	May 1978	3.5.39
	May 1978	3.5.40
DRKB01	May 1978	3.5.41
	May 1978	3.5.42
	May 1978	3.5.43
	May 1978	3.5.44
	May 1978	3.5.45
	May 1978	3.5.46
DRKB03	May 1978	3.5.47
	May 1978	3.5.48
	May 1978	3.5.49
	May 1978	3.5.50
	May 1978	3.5.51
	May 1978	3.5.52
	May 1978	3.5.53
DRLP01	May 1978	3.5.54
	May 1978	3.5.55
DRMT01	May 1978	3.5.56
	May 1978	3.5.57
	May 1978	3.5.58
	May 1978	3.5.59

	Date	Page
DRRT01	May 1978	3.5.60
	May 1978	3.5.61
DRSOP1	May 1978	3.5.62
	May 1978	3.5.63
DRTC01	May 1978	3.5.64
	May 1978	3.5.65
	May 1978	3.5.66
	May 1978	3.5.67
	May 1978	3.5.68
DRTP01	May 1978	3.5.69
	May 1978	3.5.70
	May 1978	3.5.71
	May 1978	3.5.72
	May 1978	3.5.73
DRTP02	May 1978	3.5.74
	May 1978	3.5.75
	May 1978	3.5.76
	May 1978	3.5.77
	May 1978	3.5.78
DRTW01	May 1978	3.5.79
	May 1978	3.5.80
	May 1978	3.5.81
	May 1978	3.5.82
TIODM	May 1978	3.5.83
	May 1978	3.5.84
	May 1978	3.5.85
	May 1978	3.5.86
	May 1978	3.5.87
	May 1978	3.5.88
	May 1978	3.5.89
3.6. Special calls	May 1978	3.6.1
Attach/Detach	May 1978	3.6.2
	May 1978	3.6.3
Intertask Communication	May 1978	3.6.4
	May 1978	3.6.5
	May 1978	3.6.6

4. TOSS UTILITIES

4.1. Introduction	Dec. 1977	4.1.1
4.2. Copy Tape to File and Copy File to Tape	Dec. 1977	4.2.1
4.3. Print File and Scan Tape	Dec. 1977	4.3.1
4.4. General Interface Rules	Dec. 1977	4.4.1
4.5. Utility Reference	Aug. 1978	4.5.1
BIX : Build Index File	Aug. 1978	4.5.2
	Aug. 1978	4.5.3
CCF : Copy Cards to Disk File	Aug. 1978	4.5.4
CDD : Copy Disk to Disk	Aug. 1978	4.5.5
	Aug. 1978	4.5.6
CFF : Copy File to File	Aug. 1978	4.5.7
CFT : Copy File to Tape	Aug. 1978	4.5.8
	Aug. 1978	4.5.9
CIT : Copy IBM File to TOSS	Aug. 1978	4.5.10
	Aug. 1978	4.5.11

		Date	Page
CPP	: Copy Program	[illegible]	4.5.12
		[illegible]	4.5.13
		[illegible]	4.5.14
CRF	: Create File	[illegible]	4.5.15
		[illegible]	4.5.16
		[illegible]	4.5.17
			[illegible]
CRV	: Create Volume		[illegible]
			[illegible]
CTF	: Copy Tape to File	[illegible]	[illegible]
			[illegible]
		[illegible]	[illegible]
CTI	: Copy TOSS File to IBM Data Set	[illegible]	[illegible]
		[illegible]	4.5.25
DLF	: Delete File	[illegible]	[illegible]
		[illegible]	[illegible]
PDS	: Print Disk Sector	[illegible]	[illegible]
PIT	: Print Index Track	[illegible]	[illegible]
PRF	: Print File	[illegible]	[illegible]
PVC	: Print Volume Table of Contents	[illegible]	4.5.31
RIX	: Reorganize Index File	Aug. 1978	4.5.32
		Aug. 1978	4.5.33
SCT	: Scan Tape	Aug. 1978	4.5.34
		Aug. 1978	4.5.35
		Aug. 1978	4.5.36
SRT	: Sort File	Aug. 1978	4.5.37
		Aug. 1978	4.5.38
		Aug. 1978	4.5.39
		Aug. 1978	4.5.40
		Aug. 1978	4.5.41
UDS	: Update Disk Sector	Aug. 1978	4.5.42
WIL	: Write IBM Labels	Aug. 1978	4.5.43
		Aug. 1978	4.5.44
		Aug. 1978	4.5.45
APPENDIX A : I/O AND ACTIVATION		Dec. 1977	A.0.1
		Dec. 1977	A.0.2
		Dec. 1977	A.0.3
APPENDIX B : CODES GENERATED FOR KEYBOARD PTS6234		Dec. 1977	B.0.1
APPENDIX C : CODES GENERATED FOR KEYBOARD PTS6231		Dec. 1977	C.0.1
APPENDIX D : ISO CODE CHARACTER SET		Dec. 1977	D.0.1
APPENDIX E : EBCDIC CHARACTER SET		Dec. 1977	E.0.1
APPENDIX F : VOLUME LABEL AND VTOC FORMATS			
	F.1. Volume label	May 1978	F.0.1
	F.2. Volume Table of Contents	May 1978	F.0.2
		May 1978	F.0.3
		May 1978	F.0.4
		May 1978	F.0.5
APPENDIX G : CODES GENERATED FOR KEYBOARD PTS6236		May 1978	G.0.1
		May 1978	G.0.2
		May 1978	G.0.3

	Date		Page
APPENDIX H : MASTER DRIVER DATA COMMUNICATION	July	1978	H.0.1
	July	1978	H.0.2
	July	1978	H.0.3
	July	1978	H.0.4
	July	1978	H.0.5
	July	1978	H.0.6
	July	1978	H.0.7

1. TOSS MONITOR

1.1. Introduction

This chapter describes briefly the structure of the TOSS monitor and the relationships between the Monitor components.
The reasons for including such a description in this Manual are:

- To enable the application programmer to understand the effects of Monitor requests.
- To illustrate the relationships between tasks, Monitor and hardware. This information is especially useful to programmers who wish to generate a TOSS Monitor.
- To introduce TOSS specialist programmers to the general concepts of the TOSS Monitor.

1.2. General Description

The TOSS Monitor is divided into the following logical functions:

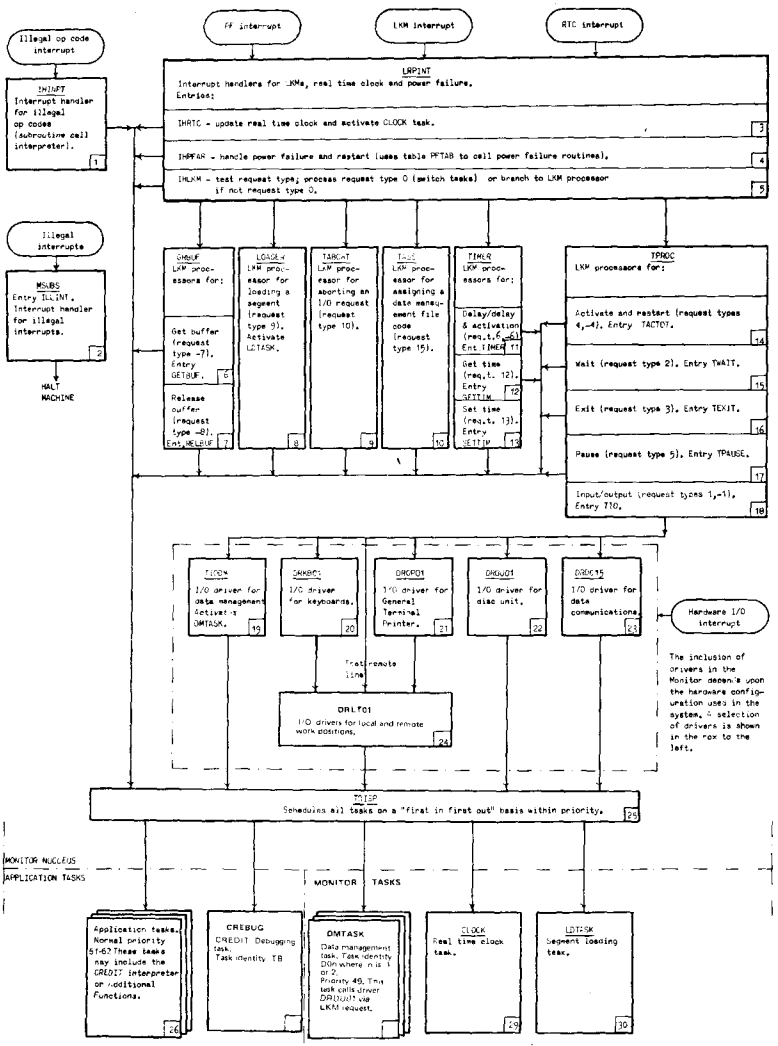
- Interrupt handlers
- LKM processors
- I/O drivers
- Dispatcher
- Tables
- Other Monitor programs.

Reference will be made to the diagram on the next page which illustrates the relationship between the various components of the Monitor. Note that this diagram does not show every module in the Monitor. The diagram has been simplified so that only the major modules and entry points are shown.

The following components are not considered:

- Monitor Initialisation Program
- Monitor Configuration Program
- Assembler Debugging Program

Reference numbers appear in the corner of each box in the diagram. These numbers are referred to in the following sections.
These boxes are grouped into modules. Each box represents an entry point in the containing module. The module name is shown in the top box of the module.


FP interrupt
LKM interrupt
RTC interrupt
Illegal op code interrupt
INTNPT
Interrupt handler for illegal op codes (subroutine call interpreter).
1
LKPINT
Interrupt handlers for LKMs, real time clock and power failure.
Entries:
INRTC - update real time clock and activate CLOCK task.
3
INPFAR - handle power failure and restart (uses table PFTAB to call power failure routines).
4
INLKM - test request type; process request type 0 (switch tasks) or branch to LKM processor if not request type 0.
5
Illegal interrupts
MSGBS
Entry ILLINT. Interrupt handler for illegal interrupts.
2
HALT MACHINE
GMBUF
LKM processors for:
Get buffer (request type -7). Entry GETBUF.
6
Release buffer (request type -8). Ent RELBUF
7
LOADER
LKM processor for loading a segment (request type 9). Activate LDTASK.
8
TARCNT
LKM processor for aborting an I/O request (request type 10).
9
TASS
LKM processor for assigning a data management file code (request type 15).
10
TIMER
LKM processors for:
Delay/delay & activation (req.t.6,-6). Ent TIMER
11
Get time (req.t. 12). Entry GETTIM
12
Set time (req.t. 13). Entry SETTIM
13
TPROC
LKM processors for:
Activate and restart (request types 4,-4). Entry TACTOT.
14
Wait (request type 2). Entry TWAIT.
15
Exit (request type 3). Entry TEXIT.
16
Pause (request type 5). Entry TPAUSE.
17
Input/output (request types 1,-1). Entry TIO.
18
TIOPK
I/O driver for data management Activates DMTASK.
19
DRKBD1
I/O driver for keyboards.
20
DRGP01
I/O driver for General Terminal Printer.
21
DRDU01
I/O driver for disc unit.
22
DRDC15
I/O driver for data communications.
23
Hardware I/O interrupt
Fast remote line
DRLF01
I/O drivers for local and remote work positions.
24
The inclusion of drivers in the Monitor depends upon the hardware configuration used in the system. A selection of drivers is shown in the box to the left.
TOTSP
Schedules all tasks on a "first in first out" basis within priority.
25
MONITOR NUCLEUS
APPLICATION TASKS
MONITOR TASKS
Application tasks. Normal priority 51-62 These tasks may include the CREDIT interpreter or additional Functions.
26
CREBUG
CREDIT Debugging task. Task identity TB
DMTASK
Data management task. Task identity DMn where n is 1 or 2. Priority 49. This task calls driver DRIVUT via LKM request.
CLOCK
Real time clock task.
29
LDTSK
Segment loading task.
30

1.3. Interrupt Handlers

An "interrupt" is an event which causes control to be passed,at the completion of the current instruction, to one of the "interrupt vectors" held in memory words 0 to 63. An interrupt is automatically generated in response to one of the following events:

- Power failure
- LKM request
- Real time clock update
- Execution of an illegal instruction
- Completion of I/O action

The particular vector entry used depends upon the type of interrupt.
Each vector entry contains a pointer to an associated "interrupt handler".

For example memory word 0 contains the interrupt vector for power failure. When power failure occurs the sequence of instructions currently being executed is interrupted, and control is handed to the instruction pointed to by memory word 0 (in the power failure interrupt handler).

As can be seen from the above diagram the interrupt handlers in boxes 1, 2 and 4 are self contained : they do not call any subsidiary modules. The handler in box 2 terminates by halting the machine. The handlers in boxes 1 and 4 terminate by branching to the Dispatcher (box 25).

The interrupt handler for the real time clock (box 3) activates a special clock task (box 29) every 100 milliseconds.
The handler terminates by branching to the Dispatcher (box 25). The interrupts for completion of I/O action are serviced by the appropriate device driver (box 22 for example).

The interrupt handler for LKM requests processes only request type 0. The interrupt handler saves registers A1 to A14 and if necessary branches to one of several "LKM processors" to process the remaining types of LKM requests. These processors are described in the following section.

1.4 LKM Processors

The LKM processors perform the following functions:

- Task scheduling (boxes 11, 14, 16 and 17)
- Input/output (boxes 9, 10, 15 and 18)
- Buffer control (boxes 6 and 7)
- Memory management (box 8)
- Monitor clock control (boxes 12 and 13)

A particular processor is invoked as a result of a LKM instruction executed in a task. As described in section 1.3, this instruction causes an interrupt. The DATA directive following the LKM instruction contains a numeric request type.
This is used by the interrupt handler (box 5) to select the required LKM processor (boxes 6 to 18).

The LKM processors in boxes 6, 7, 9, 11, 12 and 13 (i.e. modules GRBUF, TABORT and TIMER) are only included in the Monitor if they are specifically requested during system generation. The LKM processor in box 8 (i.e. module LOADER) is only included if memory management is requested during system generation.
The LKM processor in box 10 (i.e. module TASS) is only included if data management is requested during system generation. The LKM processors in boxes 6, 7 and 9 to 17 are self contained. They do not call any subsidiary modules. These processors terminate by branching to the Dispatcher (box 25).

The LKM processor for segment loading (box 8) activates a special loading task (box 30), via a normal LKM request, to carry out the loading process. This processor terminates by branching to the Dispatcher (box 25).

The LKM processor for I/O control (box 18) branches to the appropriate I/O driver (boxes 19 to 24 for example) to execute the I/O request. I/O drivers are discussed in the next section.

1.5 I/O drivers

I/O drivers perform device control functions and (optionally) process data communications and data management I/O requests. The type of function to be performed is specified in register A7 by the requesting task. Examples are as follows:

/00 Test Status
/01 Basic Read
/02 Standard Read
/03 Numeric Read
/05 Basic Write
/06 Standard Write
/0A Random Read
/0B Random Write

Further parameters are specified in an "event control block".
The address of the event control block is placed in register A8 by the current task.

A separate driver is available for each type of device (for example boxes 20 to 22 in the diagram) and some devices may have more than one driver (for example keyboard). The required drivers must be built into the Monitor during system generation. Separate drivers are also available for performing device control (e.g. box 24) at either local or local and remote work positions comprising one or more of the following devices:

- Keyboard
- Teller Terminal Printer
- General Printer
- Numeric Display
- Indicator Display/Keyboard Lamps
- Video/Plasma Display

The devices must be connected to the Terminal Computer via a Channel Unit for Local Terminals (CHLT) and/or a Channel Unit for Remote Terminals (CHRT). Devices which are used locally (i.e. not via modems) must be connected to the CHLT. Devices which are used remotely (i.e. via modems) must be connected to the CHRT.

One of two drivers may be used to control devices attached to the CHLT and CHRT. Driver DRLT01 is used to control devices attached to the CHLT. Driver DRRT01 is used to control devices attached either to the CHLT or CHRT. That is driver DRLT01 controls locally connected devices only, and driver DRRT01 controls both locally and remotely connected devices. Only one of these drivers are included in the Monitor during system generation.

Driver DRLT01 or DRRT01 is normally entered only from the individual device drivers for the above devices (for example boxes 20 and 21). The only exception to this rule is the "test remote line" function. In this case the driver DRRT01 is entered directly from the LKM processor (box 18).

Drivers DRLT01 and DRRT01 terminate by branching to the Dispatcher (box 25).

The drivers for the remaining devices not listed above (for example box 27) are self contained. That is they do not enter any additional driver. These drivers all terminate by branching to the Dispatcher.

A separate driver is also available for each type of communication line discipline. Only one type of communication driver can be included in the Monitor at one time (for example box 23). These drivers terminate by branching to the Dispatcher (box 25).

I/O drivers for devices and data communication are only included in the Monitor if they are specifically requested during system generation.

A separate driver TIODM is available for data management (box 19). This driver activates a special data management task (box 27 – one task per disk drive) to process data management requests. This task issues a disk I/O LKM request to perform the actual I/O via the disk driver (box 22). Driver TIODM terminates by branching to the Dispatcher (box 25). The data management driver is only included in the Monitor if data management is requested during system generation.

1.6 The Dispatcher

Previous sections have described the way in which interrupts (LKM request, completion of I/O action etc.) are processed by the Monitor. When the processing of an interrupt is complete, control is handed to the Dispatcher.

The Dispatcher then determines which application or Monitor tasks are able to proceed. If several tasks are able to proceed a task is chosen on a "first in first out" basis within priority level. Registers A1 to A14 are restored for this task and the task is entered via a RTN instruction.

The RTN instruction automatically enables interrupts to occur. Any interrupt of an equal or lower priority which occurred during the processing of the last interrupt would have been queued. On the RTN instruction being executed this interrupt will take effect immediately and control will again be passed to an interrupt handler.

However, if no interrupt has been queued the task will begin execution. Execution of the task will continue until another interrupt occurs (which may of course be a LKM instruction executed by the task). When processing of this interrupt has been completed the task will again become a candidate for scheduling. And so on.

A task may exit by issuing a LKM request type 3. The Dispatcher will then delete all record of the task and the task will cease to exist.

Another task may be activated by the running task issuing LKM request type – 4.

A task may pause itself by issuing a LKM request type 5.

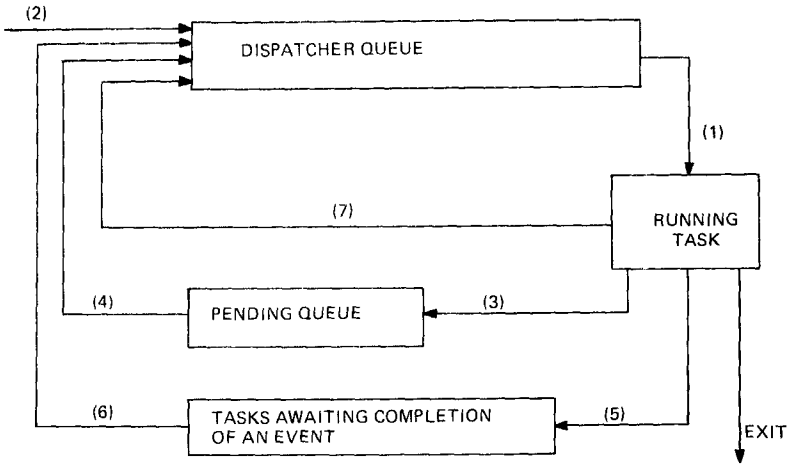
A paused task may then be restarted by another task issuing a LKM request type 4.

A task may be placed at the back of the dispatcher queue and cause the task at the head of the queue to be called into execution by issuing a LKM request type 0 (switch tasks).

Exit, activation, pause, restart and switch are all controlled by the LKM processors and Dispatcher.

In order to identify and schedule tasks, each task must have a task identifier and priority level. Application tasks must be assigned a task identifier and priority level during system generation. Monitor tasks (e.g. data management task, segment loading task) have a predefined task identifier and priority level and need not be specified during system generation.

The scheduling of tasks by the LKM processors and Dispatcher is illustrated in the following diagram.



The following notes refer to the numbers in the diagram:

1. Tasks are dispatched from the dispatcher queue "first in first out" within priority. That is, each task as it becomes available for dispatching is placed at the back of the dispatcher queue. When selecting a task for dispatching, the Dispatcher considers only the tasks with the highest priority level. The task at the front of the queue in this priority level is chosen for dispatching.
2. A running task may activate or restart another task. The new task is placed at the back of the dispatcher queue and the running task continues.
3. A running task may activate itself (normally at a different start point from the original one). However, the Dispatcher cannot schedule two tasks with the same task identity at the same time. For this reason the newly activated task is placed in a pending queue and the running task continues.
4. When the running task performs an exit LKM request the pending task with the same task identity is placed at the back of the dispatcher queue.
5. When the running task issues certain LKM requests (e.g. I/O) it must wait until the requested event is complete. During this time the Dispatcher will select another task from the dispatcher queue and the original task will not be considered for dispatching.

6. When the event is complete the waiting task is placed at the back of the dispatcher queue.
7. A running task may request that it be placed at the back of the dispatcher queue and that the task at the head of the queue be called into execution.

For an example of scheduling using the LKM request I/O and activation see appendix A.

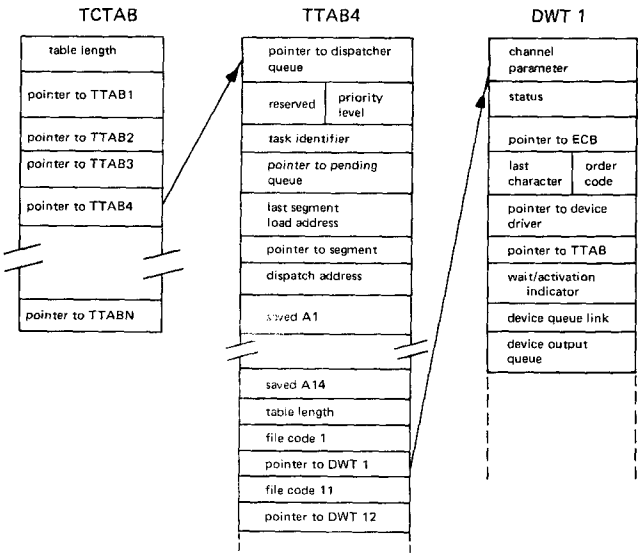
When the dispatcher queue is empty the Dispatcher simply performs an "idle loop" (priority level 63) waiting for a task to be queued. This situation will arise when, for example, all tasks are waiting for I/O to be completed. As soon as an I/O is completed a hardware interrupt will pass control to the appropriate device driver. The driver will then call a routine to queue the waiting task in the dispatcher queue and hand control to the Dispatcher.

1.7 Tables

The Monitor uses a large number of tables to record the status of the various components of the system.
Three of the more important tables are:

- Task control table (TCTAB)
- Task table (TTAB)
- Device work table (DWT)

The following diagram indicates briefly the contents of these tables and the relationships between them:



There is one TCTAB in the system. It points to each TTAB in the system. There is one TTAB per task. TTAB points to each DWT used by the task. There is one DWT for each device in the system, though the CREDIT debugger requires a second DWT for the SOP switches.

TCTAB, TAB's and DWT's are built up from the task definition information supplied to SYSGEN or MONCON. The pointers in TCTAB occur in the order in which tasks were defined during task definition.
The first task in TCTAB is the task activated by the Monitor when the application is first loaded into memory.

1.8. Other Monitor Programs

1.8.1 *Introduction*

The following additional programs (not shown on the above diagram) may be included in the TOSS Monitor if they are requested during system generation:

- Monitor Configuration Program
- Monitor Initialisation Program

These programs are discussed in the following sections.

1.8.2 *Monitor Configuration Program*

The Monitor Configuration Program (MONCON) is executed by the TOSS Monitor during the system start process. MONCON will read certain parameters from a Monitor configuration file supplied by the user. These parameters are used by MONCON to generate certain Monitor tables required for the application tasks.
MONCON is then overwritten in memory.

MONCON is only executed if it has been requested during system generation.
If MONCON is not requested the parameters must be keyed-in to SYSGEN during "task definition" and "common device definition".

The objective of MONCON is to provide the user with a quick and simple method of supplying certain parameters to the Monitor which are likely to change relatively frequently. These parameters may thus be altered without having to re-generate a TOSS Monitor.

1.8.3 *Monitor Initialisation Program*

This program enables the user to restart the TOSS Monitor at address /92 without having to reload the Monitor.

This facility is normally only used for program testing via the Assembler Debugging Program. It is only included in the Monitor if requested during system generation.

1.9. TOSS System software

In addition to the TOSS Monitor, other software components are available which run under the control of the Monitor. They are:

- CREDIT Configurator
- CREDIT Debugging Program
- CREDIT Interpreter
- TOSS Utility Programs
- Additional functions

These components, together with the TOSS Monitor, are known as TOSS System Software.

The CREDIT Configurator, Debugging Program and Interpreter are described in the CREDIT PRM (M04).

The TOSS utility programs are described in the TOSS User Specification (M17) and in chapter 4 of the present Manual.

The additional functions are described in the Assembler PRM (M06 Part 1).

2. LKM REQUESTS

2.1. Introduction

As mentioned in section 1.4. the TOSS Monitor provides application tasks with the following services:

- Task scheduling
- Input / Output
- Buffer control
- Memory management
- Monitor clock control

These services are requested by executing a LKM instruction. The DATA directive following the LKM instruction must contain a numeric request type specifying the type of service required. Certain LKM requests require further information to be loaded into register(s) and possibly further DATA directive(s), depending upon the type of request.

Following a LKM request, registers A1 to A14 are saved by the Monitor. They are restored immediately prior to handing control back to the application task.

In the following lists LKM requests are described briefly under the generic headings mentioned above.
Detailed descriptions of each request type follow in section 2.2.

Task scheduling

Request type	Description
0	Switch the running task for the next available task in the dispatcher queue
3	Exit task
4	Restart a paused task
−4	Activate another task
5	Pause the running task
6	Delay the running task
−6	Activate another task after a delay

Note: if memory management is being used LKM request with activation must specify a start point in the same segment as the LKM request.

Input/output

Request type	Description
1	Perform normal input or output
−1	Perform input or output followed by task activation
2	Wait for completion of input or output
10	Abort a previous input or output request
15	Assign a file code to a file.
16	Assign an index to a data file.

Note: The LKM DATA (−)1 knows two special calls, the ATTACH/DETACH and INTERTASK COMMUNICATION; see I/O drivers, section 3.6.

If memory management is being used the LKM request with activation must specify a start point in the same segment as the LKM request.

Buffer Control

Request type	Description
–7	Allocate buffers to the task
–8	Release buffers from the task.

Memory Management:

Request type	Description
9	Load program segment into memory

Clock Control:

Request type	Description
12	Get time from monitor clock
13	Set time in monitor clock

2.2. LRM Request Reference

This section contains a detailed description of the calling sequence and use of each LRM request.
The descriptions are in request type order.

0	SWITCH TASKS	0

Calling sequence : LKM
DATA 0

Type : Task scheduling

Description : Execution of the requesting task is suspended and control is handed to the next available task in the dispatcher queue (this task will have the same priority as the requesting task). The requesting task is placed at the back of the dispatcher queue.

1 NORMAL I/O 1

Calling sequence : LDK A7, code
LDKL A8, ecb-address
LKM
DATA 1

Type : Input/Output

Description : The I/O operation specified in register A7 will be performed. The device, memory buffer etc. to be used are specified in an event control block whose start address must be held in register A8.

If the wait bit in register A7 is set to 0, the task continues executing instructions (starting at the instruction following the I/O request) while the I/O operation is in progress. If the wait bit is set to 1 execution will be suspended until the I/O operation is complete. Execution will recommence at the instruction following the I/O request.

If the no-wait option is used a wait (request type 2) LKM request must later be issued to synchronise with the I/O completion.

The formats of register A7 and of the event control block are described in chapter 3 (I/O Drivers).

The calls ATTACH/DETACH and INTERTASK COMMUNICATION are described in section 3.6.

−1	I/O AND ACTIVATE	−1

Calling sequence : LDK A1, parameter
LDK A7, code
LDKL A8, ecb-address
LKM
DATA −1
DATA start address

Type : Input/Output

Description : An I/O operation is performed on the specified device. When this I/O operation is completed the current task is activated.
If the task is still active at this time, the activation request is placed in the pending queue, see Appendix A.

The task will be activated at "start-address" with "parameter" in register A1. "Parameter" is a means of passing information to the task being activated and its use depends upon the requirements of the application program.

The I/O operation to be performed is specified in register A7. The device, memory buffer etc. to be used are specified in an event control block whose address must be held in register A8.

The wait bit in register A7 must be zero, and the task continues executing instructions (starting at the instruction following the I/O request) while the I/O operation is in progress. An exit LKM request must be issued subsequent to an I/O and activation request so that the task can be re-activated on completion of the requested I/O.

The calls ATTACH/DETACH and INTERTASK COMMUNICATION are described in section 3.6.

2	WAIT	2

Calling sequence : LDKL A8, ecb-address
LKM
DATA 2

Type : Input/Output

Description : Execution of the current task is suspended until the I/O operation associated with the event control block pointed to by register A8 is complete. Execution of this task will be resumed at the instruction following the wait request.

This request is used after an I/O LKM request with no-wait. The wait LKM request synchronises the execution of the task with the completion of the I/O.

The format of the event control block is described in chapter 3 (I/O Drivers).

3	EXIT	3

Calling sequence : LKM
DATA 3

Type : Task Scheduling

Description : Execution of the current task is ended.
The task can be re-activated subsequently by another task via a LKM request type –4 or –6. However, the previous contents of registers etc. will be lost.

4 RESTART 4

Calling sequence : LDK A7, tid
LKM
DATA 4

Type : Task scheduling

Description : The paused task identified by "tid" is restarted (i.e. put into the dispatcher queue).

The restarted task must be in a paused state as a result of issuing a LKM request type 5. It will begin execution at instruction following the pause request.

The requesting task (i.e. the one issuing the restart request) continues executing at the instruction following the restart request as soon as the restarted task has been placed in the dispatcher queue.

–4	ACTIVATE	–4

Calling sequence : LDK A1, parameter
LDK A7, tid
LKM
DATA –4
DATA start-address

Type : Task scheduling

Description : The task with identifier "tid" is placed in the dispatcher queue. This task will be activated at the instruction indicated by "start-address" with "parameter" in register A1 and "tid" in register A2. "Parameter" is a means of passing information to the task being activated and its use depends upon the requirement of the application program. The contents of other registers is not relevant.

If the task to be activated is already active the request becomes pending.

The current task resumes execution at the instruction following the activate request. Execution of the requesting task recommences as soon as the activated task is placed in the dispatcher queue or becomes pending.

5	PAUSE	5

Calling sequence	:	LKM DATA 5
Type	:	Task scheduling
Description	:	Execution of the current task is suspended and the task is no longer considered for dispatching. The task can be restarted only by another task issuing a LKM request type 4.

6 DELAY 6

Calling sequence : LDKL A8, time
LKM
DATA 6

Type : Task scheduling

Description : Execution of the current task is suspended for the specified period of time. During this period the task will not be considered for dispatching. At the end of this period the task will again be put into the dispatcher queue.

"Time" specifies the delay in multiples of 100 ms.

–6 DELAY AND ACTIVATE –6

Calling sequence : LDK A1, parameter
LDK A7, tid
LDKL A8, time
LKM
DATA –6
DATA start-address

Type : Task scheduling

Description : When the period of time in register A8 has expired the task with identifier "tid" is placed in the dispatcher queue. The new task will be activated at the instruction indicated by "start-address". The task is activated with "parameter" in register A1 and "tid" in register A2. "Parameter is a means of passing information to the task being activated and its use depends upon the requirements of the application program. The contents of the other registers is not relevant.

If the task to be activated is already active the request becomes pending.

"Time" specifies the delay in multiples of 100ms.

Execution of the requesting task is resumed at the instruction following the delay and activate request. Execution of the reques[illegible]ask re-commences as soon as the delay period starts.

–7 GET BUFFER –7

Calling sequence : LDK A7, number-of-buffers
LKM
DATA –7
DATA bcb-address

Type : Buffer control

Description : If the required number of contiguous buffers are free and there are no other requests queued for the buffer pool the specified number of buffers will be allocated to the requesting task. The start point of the first buffer allocated is returned to the task in register A8.

"bcb-address" refers to a two word buffer-control block containing the buffer pool start address and the buffer size in bytes.

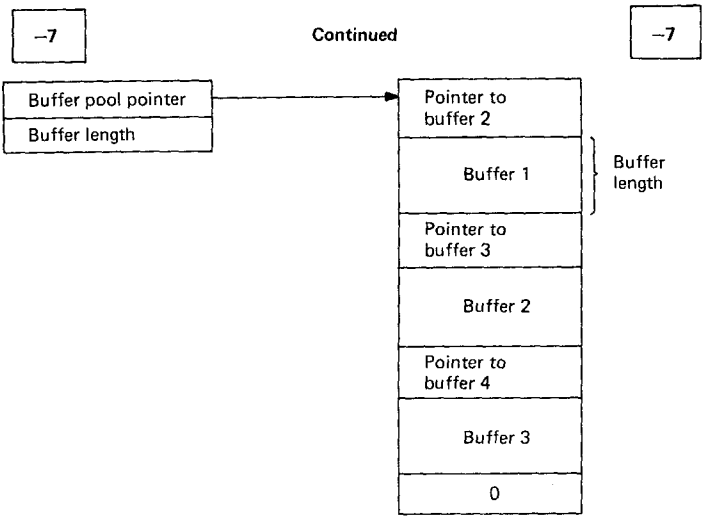
If the required number of buffers are not available the task is placed in a "get buffer request" queue until the buffers are available. This queue is organised "first-in first-out".

In order to use the get buffer and release buffer LKM requests (types–7 and –8) the application program must contain a correctly structured buffer pool. There must be one buffer pool per program. It may be set up using DATA and/or RES directives.

The buffer pool must be divided into fixed length buffers. Each buffer must be prefixed by a pointer containing the address of the next buffer in the pool. A zero pointer indicates the end of the buffer pool.

The get and release buffer requests both refer to a two word block containing the buffer pool start address and the buffer size in bytes. The buffer size excludes the pointer word at the start of the buffer.

The following diagram summarises the structure of a buffer pool.



The pointers at the start of each buffer must be set up by the application program before the first GET BUFFER request. However, they need not be **maintained** by application tasks. That is, when a GET BUFFER request is made an area of memory, including the pointer, is allocated to the task. The task may overwrite the buffer pointer in addition to the associated buffer. When a release buffer request is made the pointers are automatically regenerated by the Monitor.

Note: if memory management is being used, all GET BUFFER requests must be in the root segment of the program.

−8	RELEASE BUFFER	−8

Calling sequence : LDKL A8, buffer-address
LKM
DATA −8
DATA bcb-address

Type : Buffer control

Description : The buffer pointed to by "buffer address" is returned to the buffer pool. The requesting task may not refer to this buffer again.

"bcb-address" refers to a two word buffer control block containing the buffer pool start address and buffer size in bytes. For details on the use of a buffer pool see the GET BUFFER LKM request (type-7).

Note: if memory management is being used, all RELEASE BUFFER requests must be in the root segment of the program.

9 LOAD SEGMENT 9

Calling sequence : LDK A7, segment-no
LDKL A8, segment-index
LKM
DATA 9

Type : Memory management

Description : The program segment identified by "segment-no" is loaded into memory from disk. Segments are numbered sequentially from 0 (root) when they are copied into disk by the TOSS utility CPP.

Register A8 must contain an index (starting at 1) to the segment entry point table SEGENT. This table must contain a list of one word addresses comprising segment entry points relative to the start of the segment.

If the specified segment does not exist, no segment will be loaded. Control will be handed to the instruction following the load segment request.

If the specified segment cannot be loaded due to a hardware error, control is handed to the address at the top of the sub-routine call interpreter table (SUBTAB).

10

ABORT

10

Calling sequence : LDKL A8, ecb-address
LKM
DATA 10

Type : Input/Output

Description : The I/O operation associated with the event control block pointed to by register A8 is aborted. The I/O operation must have been requested by the running task (therefore I/O with wait cannot be aborted).

Return code /C000 is set in the event control block if the abortion was successful. If the abort request is not accepted (e.g. because the I/O is completed) register A7 is set to −1.

An abort request can only be issued for I/O operations controlled by the following drivers.

- DRKB01 − General Keyboard
- DRTW01 − Typewriter
- DRTP01 } Teller Terminal Printer
- DRTP02 }
- DRSOP1 − System Operators Panel

The format of the event-control-block is described in chapter 3 (I/O Drivers).

12 GET TIME 12

Calling sequence : LDKL A8, tcb-address
LKM
DATA 12

Type : Monitor clock control

Description : The time of day is copied from the monitor clock into the timer control block (tcb) pointed to by register A8.

The time is stored in the tcb in ISO–7 characters. The tcb must be set up in the application program and must have the following format:

byte		
0	hour	hour
2	minute	minute
4	second	second

13 SET TIME 13

Calling sequence : LDKL A8, tcb-address
LKM
DATA 13

Type : Monitor clock control

Description : The time of day is copied from the timer control block (TCB) (set up by the application program and pointed to by register A8) into the monitor clock. The monitor colock is periodically updated by the real time clock in order to maintain the correct time of day.

The time must be held in the TCB in ISO–7 characters. The TCB must be set up in the application program and must have the following format :

0	hour	hour
2	minute	minute
4	second	second

15	ASSIGN FILE CODE	15

Calling sequence : LDK A7, task-control
LDKL A8, asblk-address
LKM
DATA 15

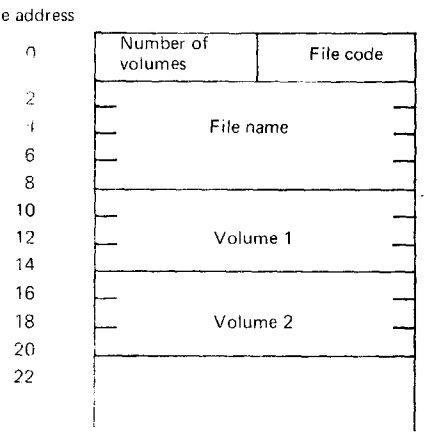
Type : Input/Output

Description : A file code is assigned to a data management file. The register A7 specifies whether the file is under control of one or more tasks. The file code and file name are specified in an assign-block pointed to by register A8.

The file must have been created previously by the data management utility CRF with the file name specified above, see section 4.5.

If task-control is 1 the file will be accessible only by the task which issued this 'assign file code' request.
If task-control is 0 this file is accessible by other tasks too.

The format of the assign block is as follows:



"Number of volumes" must not exceed 4.

"File code" is the file code to be assigned.

"File name" is a string of 8 characters, left-justified and padded with spaces, defining the file to be assigned.

"Volume" is a string of 6 characters, left-justified and padded with spaces, defining the volume where the file resides. A maximum of four volume names may be given.

15 **Continued** 15

A search is made in the volume table of contents (VTOC) for all volumes defined in the assign-block for the requested file name. In memory a file descriptor block (FDB) is built containing all relevant data for the file. The task table or common device table is updated with the FDB-address and the assigned file code. More than one file code may be assigned to the same data file.

Upon completion of the request, the system responds as follows:

A7	= 0	assignment performed
	≠ 0	assignment refused, for one of the following reasons
	= -1	Request error
	= 1	Disk I/O error
	= 2	no free entry in the common device table
	= 3	no file descriptor block available
	= 4	one or more volumes unknown
	= 5	file code already used
	= 6	filename unknown
	= 7	file section missing
	= 8	faulty disk format
	= 9	more than 4 extents exist

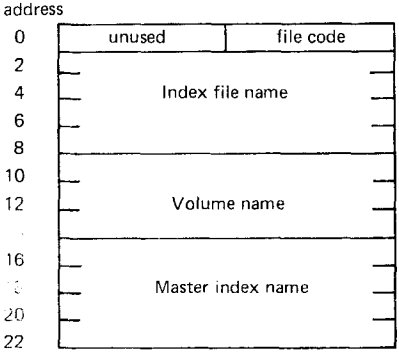
16 ASSIGN INDEX 16

Calling sequence : LDKL A8, asblk-address
LKM
DATA 16

Type : Input/Output

Description : An index file is assigned to a previously assigned data file. The index file is assigned to the file code specified in the assign block in which the index file name, volume name and master index name are specified.

The format of the assign block is as follows:



File code is the data file to which the index is assigned.

Index file name is a string of 8 characters, left-justified and padded with spaces, defining the index file name.

Volume name is a string of 6 characters, left-justified and padded with spaces, defining the volume on which the index file resides. An index file has access on only one volume.

Master index name is a string of 8 characters left-justified and padded with spaces, defining the name of the master index file to be used as index to the index file.

A maximum of 4 index files may be assigned to the same data file. The data file should have then different file codes, one for each index file.

All index-files in a file structure must be assigned within the same CDTAB/TTAB.

16 Continued 16

Upon completion of the request the system responds as follows:

A7	= 0	Assignment performed
	≠ 0	Assignment refused, for one of the following reasons:
	= −1	Request error
	= 1	Disk I/O error
	= 2	No free entry in CDTAB
	= 3	Not sufficient core memory available for Master Index or File descr. blocks
	= 4	Volume name unknown
	= 5	File already assigned from this task
	= 6	File name unknown. Incorrect file format.
	= 7	File section missing or found twice
	= 8	Faulty disk format
	= 9	More than 4 extents exist
	= 10	No data file assigned
	= 11	4 index files already assigned
	= 12	Size of disk buffers not sufficient
	= 13	Request busy. Repeat assign request

3. I/O DRIVERS

3.1 Introduction

The various types of driver available with the TOSS Monitor are described briefly in Section 1.5. The present chapter describes in detail the function of the drivers and the interface rules which must be complied with.

Sections 3.2 to 3.4 describe general aspects of certain related groups of drivers. Section 3.5 describes the general interface rules for all drivers. And section 3.6 contains detailed reference information for each driver. The reference information is given in alphabetical driver name order (e.g. DRCG01, DRCD01).

3.2. Drivers for Teller Terminal Printer

The drivers used to control a Teller Terminal Printer are DRTP01 and DRTP02. Only one of these drivers can be incorporated into the TOSS Monitor. DRTP02 is a version of the driver which can accept CREDIT order codes for the Teller Terminal Printer.

3.3 Drivers for Data Communication

Data communication (DC) drivers control the reception and transmission of data between the Terminal Computer and the remote computers. There is one DC driver for each type of line procedure. However, only driver DRDC15 is described in this manual. More DC drivers will be included in future Manual updates. A single DC driver may only be connected to one remote computer.

The unit of information passed between a DC driver and an application task at a DC I/O request is called a message. This message may have any format, and need not contain any special DC information such as destination address, message length etc. The DC driver generates or discards such information automatically. It also performs blocking and de-blocking when necessary.

The main orders handled by DC drivers are write (order /06), read (order /02) and exchange (order /08). The write order is used to transmit a message from a task. The read order is used to obtain a message which has been sent to the task. The exchange order has the effect of a write followed immediately by a read.

DC drivers comprise the following functions:

- activation
- line procedure
- device simulation

The activation function handles the request and checks the validity of the parameters. The work areas are prepared with respect to queues and timers for the requests.

The line procedure function generates and checks the message framing characters such as STX, ETX, BCC etc. It also decodes the line procedure control characters such as EOT, ACK, polling, selection etc. This function also performs timeout supervision for the procedure and takes actions at timeout.

The device simulation function carries out actions that are related to the simulated terminal type of the remote computer e.g. status signalling.

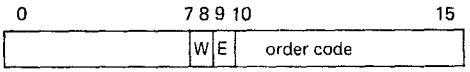
In each application program performing DC there should be a special task dedicated to the reception of random messages from the remote computer. This task is known as the "DC task". It is discussed in the TOSS User Specification (M17).

DC control characters must not appear in the body of a message. If such a character is detected during a write request (or the write part of an exchange request) it will be ignored and bit 13 "code check error" will be set in the return code.

3.4. General Interface Rules

Registers A7 and A8 are used for communication between drivers and application tasks (see LKM request types 1 and −1).

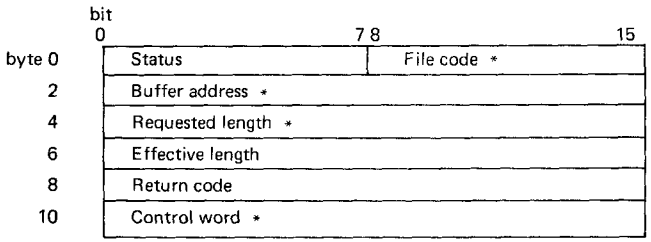
Register A7 has the following format:



where:

bits	0–7	are not significant.
bit	8	is the wait bit. Execution of the task will recommence at the start of the requested I/O operation if the wait bit is set to 0. If the wait bit is set to 1, execution will be suspended until the I/O operation is completed.
bit	9	is the echo bit. This bit may be set to 1 only for keyboard input operations. When set to 1, the driver automatically generates an echo of each input character on the display or printer device. The display device must have been declared as an echo device during system generation.
bits	10–15	is the order code specifying the I/O operation to be performed by the driver. Order codes have a different meaning for each driver. They are discussed in the Driver Reference section (3.6).

Register A8 points to the start of an "event control block" (ECB). The ECB has the following format:



The ECB must be reserved in the application program via the DATA and/or RES directives. The fields marked with an asterisk in the above diagram must be set by the user although the control word can be set by the driver on return e.g. order code /00, Test Status returns the current status of the device to the control word. The remaining fields are set by the drivers.

The significance of each field in the ECB is as follows:

Status — This field is used exclusively by the Monitor. The only bit of possible interest to the application programmer is the most significant (bit 0). This bit is set to 1 on completion of the I/O operation.

File code — The file code specifies the device to be used in the I/O operation and must always be present in the ECB. File codes are associated with devices during system generation. (See the list of recommended file codes below).

Buffer address — This word contains the address of the start of the buffer to be used for the I/O operation. Data is written to this memory area on input. Data is written from this memory area on output. For certain drivers and orders the buffer address is not significant and can take any value (see section 3.6).

Requested length — For an output operation, this word contains a binary number indicating the length of the output record in bytes. For an input operation this word contains a binary number indicating the **maximum** length of the input record in bytes. For certain drivers and orders the requested length is not significant and can take any value (see section 3.6).

Effective length — This word contains a binary number indicating the actual number of bytes, input or output.

Return code — This word contains a bit pattern indicating any exceptional conditions encountered during the I/O operation. The significance of some of these bits is device dependent (see section 3.6). The significance of the remaining bits, when set to 1, is as follows:

Bit	Meaning
0	Illegal request
1–2	Not used/device dependent
3	End of file
4–8	Not used/device dependent
9	Time out/hardware error
10	End of device
11	Illegal order sequence
12	Incorrect length
13	Data fault (parity/CRC/LRC/code)
14	Throughput error
15	Not operable

Control word — This word contains device dependent control information. For certain driver and orders, the control word is not significant and can take any value (see section 3.6).

Note: The ECB is used by the Monitor during I/O. The contents of the ECB should not be changed until any associated I/O operation is completed.

The file codes which are recommended are listed below:

Recommended file code (hexadecimal)	Device
/10	System operator panel-in
/11	System operator panel-out
/12	Cassette recorder no. 1
/13	Cassette recorder no. 2
/15	Remote line test
/20	Keyboard
/25	Reserved for future use
/30 – /30,/31,/32	General printer – terminal printer TJ, TV, TR
/40	Signal display
/41	Numeric display
/50	Character display
/60	Data communication
/70	Magnetic tape
/80	Line printer
/90	Reserved for future use
/A0	Reserved for future use
/B0 – /B3	Reserved for future use
/B6	Reserved for future use
/C0 – /CF	Data management disk files
/D0	Intertask communication – input
/D1	Intertask communication – output
/F0	Fixed disk, drive 1
/F1	Cartridge disk, drive 1
/F2	Fixed disk, drive 2
/F3	Cartridge disk, drive 2
/F4 – /F7	Reserved for future use
/F8	Flexible disk, drive 0
/F9	Flexible disk, drive 1
/FA	Flexible disk, drive 2
/FB	Flexible disk, drive 3
/FC – FF	Reserved for future use

Note: File codes /F0–/FB are automatically assigned at system generation if data management is included.

3.5 Driver reference

This section contains detailed reference information for each driver. The information for each driver is given in alphabetical driver name order.

During system generation, certain parameters concerning the handling of I/O devices can be specified. These parameters are described for each driver in this section. The words "system generation" are printed in bold type throughout this section in order to highlight these parameters. Most parameters are specified during the "conditional assembly" specification for each driver. Where this is not the case the relevant part of system generation appears in brackets.

The table below shows which driver belongs to which peripheral device.

Peripheral device	Drivers
Card reader	DRCR01
Cassette recorder	DRTC01
Console typewriter	DRTW01
Data Communication – BSC multipoint	DRDC15
Data Communication – BSC point-to-point	DRDC17
Disk	DRDU01
Flexible disk	DRFD01
General terminal printer	DRGP01
Keyboards general: only 6236:	DRKB01 DRKB03
Line Printer	DRLP01
Magnetic tape recorder	DRMT01
Numeric display	DRDN01
Remote terminal	DRRT01
Signal displays, lamps on keyboards	DRDT01
SOP Panel	DRSOP1
Teller terminal printer	DRTP01, DRTP02
Video and plasma display	DRDY01

DRCR01 CARD READER **DRCR01**

General information : This driver handles one card reader PTS6885 connected to CPU via CHCD on a programmed channel only.

Calling sequence : Normal I/O:

```
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  1
```

I/O and Activate:

```
LDK   A1, parameter
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  -1
DATA  start address
```

Order code : /02 – standard read

Buffer address : Significant for this order

Requested length : The value must be within 0–80.

Effective length : The number of characters read up to the first space character. If there is a space in column 23, then the effective length will be 22.

Return code : The following bits may be set by this driver:

Bit	Meaning
0	Request error
3	EOF detected
10	Input hopper empty or output stacker full
12	Incorrect length – Requested length is too long (>80). Requested length is too short (there is more information on the card than has been transferred to user buffer).
13	Data fault – Received character cannot possibly be converted to ASCII. The erroneous character is replaced by a "?" (/3F).
14	Throughput error – The card reader offers a new character before the previous one has been handled by the driver.
15	Not operable – Requires an operator intervention.

DRCR01 **Continued** **DRCR01**

Control word	:	Not significant.
Order	:	/02 – standard read. The LKM request must be issued for each card to be read. The cards are read in Hollerith code on 12 bits, converted into ASCII code on 8 bits, and stored until requested length is reached.
Power failure	:	If there is a read request running when power failure occurs, the request is completed with bit 14 Throughput error set in the return code of ECB. To be sure that no information from the actual card is lost, the card has to be read again (for instance, moved from output stacker to input hopper).

DRDC15

BSC MULTIPOINT DATA COMMUNICATIONS

DRDC15

General information : This description covers the standard Binary Synchronous Communication (BSC) Multipoint driver for PTS 6810. The driver is designed to fulfil different BSC requirements, mainly for IBM 3270 simulation.

The transmission speed should be 4800 bps or less. The driver can be used with dialled up (switched) or leased lines, two or four wire.

Calling sequence : Normal I/O:

```
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA 1
```

I/O and Activate:

```
LDK   A1, parameter
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA-1
DATA  start address
```

Order codes : The following orders may be used:

/02 – read
/06 – write
/08 – exchange
/22 – release read buffer
/31 – get write buffer
/37 – transfer parameters

Buffer address : During **system generation** the transmit buffer length and receive buffer length must be specified.

If the length of the transmit buffer is specified as zero then the necessary buffer(s) must be reserved in the application program. In this case the application task must set the value of buffer address in the ECB before requesting order /06 (write) or /08 (exchange). These are the only orders for which the application task must set the buffer address.

Order /31 (get transmit buffer) cannot be requested if the transmit buffer length is specified as zero during **system generation**.

If the length of the transmit buffer is specified as **non** zero, two transmit buffers will be reserved within the driver. They will be allocated dynamically by the driver and need not be reserved in the application program. In this case the transmit buffer address is determined by the driver when order /31

DRDC15 — Continued — DRDC15

(get transmit buffer) is requested. A subsequent order /06 or /08 will use the buffer address set in the ECB by the get transmit buffer request. This buffer address must not be changed during the intervening period.

The first word of the transmit buffer must be reserved for use by the driver.

The length of the receive buffer must be specified as non zero during **system generation**. Two receive buffers will always be reserved within the driver. The buffer address in the ECB will be set by the driver when order /02 (read) or /08 (exchange) is requested. After an order /02 or /08 the receive buffer must be released as soon as possible by requesting an order /22 (release receive buffer).

The length of the receive buffer and transmit buffer specified during **system generation** must be equal to the length of the longest message in words.

Requested length : Only significant for orders /06 and /08. The requested length should be equal to the length of the message to be sent including the reserved word at the start of the buffer.

Effective length : Only significant for orders /02, /06, /08 and /31. For orders /02 and /08 the length will be the number of bytes of application data received. For order /06 the length will be the number of bytes of application data transmitted (including the first word). For order /31 the length will be the length of the buffer allocated.

Return code : The following bits may be set by this driver:

Bit	Meaning
0	Illegal request
2	Status change
5	Calling indicator
9	Time-out (poll time-out for DC task)
10	Carrier off
13	Code check error
14	Throughput error. Set at unsuccessful transmission
15	Not operable. Modem not ready

DRDC15 Continued DRDC15

Control word : Only significant for orders /02, /06, /08, /31 and /37 (i.e. not order /22).

For orders /02, /06, /08 and /31 the control word contains the timeout value in multiples of 100 ms. A zero control word implies no time out function. After a request with timeout the control word will contain the timeout period remaining. After a read request performed by the "DC task" the control word contains the task address (see order /37).

For order /37 the control word holds the line address of the Terminal Computer and the logical device.

Order : /02 – read:

This order has a special function when used in a "DC task". For other tasks this order is used to receive a message addressed to a task. The control word holds the timeout value in multiples of 100 ms. Control word equal to zero means no timeout supervision. The request is completed when a message is correctly received, or because of timeout. The return code is zero if there is a message before timeout, otherwise bit 9 "timeout" will be set.

A message that is correctly received when there is no order /02 or /08 request outstanding for the addressed task is handed over to the "DC task". The DC driver will queue messages and DC status changes when there is no order /02 request outstanding from the DC task.

There is no timeout supervision on an order/02 request issued by the DC task. When a message is received the return code is zero and the task address (see order /37) is returned in the control word. On a status change bit 2 "status change" is set. At a status change one of the following bits is set in the return code:

Bit	Meaning
2	Status change
5	Calling indicator
9	Poll time out
10	Carrier off
15	Not operable. Modem not ready.

All status changes will complete the request for the DC task, e.g. "carrier off" will give return code /2020 and "carrier on" after that /2000.

When a message has been received, either by the DC task or any other task, a receive buffer in the driver has been allocated. This buffer has to be released as soon as possible using order /22 (release read buffer) in order to avoid busy situations due to lack of buffers.

Order : /06 – write

This order will queue a message for transmission on a poll. The control word holds the timeout value in multiples of 100 ms. Control word equal to zero means no timeout supervision.

All message framing characters such as STX, ETX, ... are added by the driver. The first word of the data buffer is reserved for control information. This word must be included in requested length.

The request is completed when the message has been transmitted and an acknowledgement has been received, or a time-out has occurred. If the message is successfully transmitted the return code is zero. If the transmission is unsuccessful due to procedure responses, bit 14 "throughput error" is set. At timeout bit 9 "timeout" will be set.

If the DC driver contains write buffers, they are released on completion of the request.

This request may not be used by the DC task.

Order : /08 – exchange:

This order is a combination of orders /02 and /06. It gives the same return codes as orders /02 and /06 above. Besides, when there is a timeout before the write part has been completed bit 9 "timeout" and bit 14 "throughput error" will be set.

Order : /22 – release read buffer:

This order is used to release a receive buffer to the DC driver. The buffer address is given in the ECB. Return code is always zero.

Order : /31 – get write buffer:

This order is only included in the driver if the transmit buffer is given a non zero length during system generation.

It is used to get a write buffer from the DC driver. The request is completed when a buffer is allocated, or a time-out occurs. When a buffer is allocated the address is given in ECB and the return code is zero. Buffer length is given in ECB field effective length.

DRDC15 Continued DRDC15

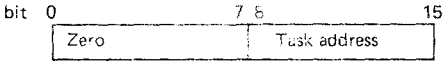
The control word holds the timeout value in multiples of 100 ms. Control word equal to zero means no timeout supervision.

At timeout bit 9 "timeout" will be set.

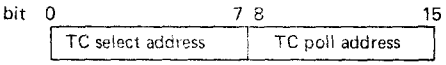
This request may not be used by the DC task.

Order : /37 – transfer parameters

For non-DC tasks this order is used to transfer the "task address" from the control word to the DC driver. The task address is an 8 bit code which uniquely identifies the task to the DC driver. It can have any binary value. The control word must have the following format:



For the DC task this order is used to transfer the terminal computer address from the control word to the DC driver. The control word must have the following format:



The request is completed immediately with return code zero.

When EBCDIC line code is used, the addresses will be transferred in EBCDIC code to the driver.

DC task file code : The file code of the DC device for the DC task is normally /60, but it may be given a different value during **system generation**.

Interrupt logging : Optionally an interrupt logger may be included in the driver. If so, all output, input and status interrupts will be logged in a circular buffer which is 200 words long and which it is possible to access with the debugging console. This facility must be requested during **system generation.**

Poll timeout value : The driver supports a poll timer which is started/restarted every time a poll sequence is received with recognizable address information. If the next poll sequence is not detected within a specified period of time, a poll timeout will occur and this event will be reported to the DC task by means of an order /02 or /08 request completion.

The timeout value in this case must be defined during **system generation** and should be given in multiples of 100 ms.

DRDC15 **Continued** DRDC15

IBM 3270 simulation : If IBM 3270 simulation is required the facilities "status and RVI handling" and possibly "read command handling" must be requested during **system generation.** These facilities are discussed in detail in the TOSS User Specification (M 17).

Transmit block length : The information exchange between the application program and the driver is always carried out on message level. For output messages the driver performs blocking if necessary. This is necessary if the message length is greater than the "transmit block length" specified during **system generation.**
The length does not include control characters such as STX, ETB or ETX.
Text blocking does not divide up the three-byte SBA order sequence, therefore the last characters of a block ending with a SBA sequence would be:
SBA, Address, Address, ETB (or ETX)

Code set : During **system generation** it is necessary to choose either an EBCDIC or ISO–7 DC code set.

This option enables the driver to communicate with ISO–7 or EBCDIC code. If ECBDIC code is chosen all incoming and outgoing characters are translated. The application program does not have to translate any information. The only thing that must be noted is that TC- and DV-addresses must be presented in EBCDIC-code by performing a "transfer parameters" request if EBCDIC code is used.

See appendix D and E for the ISO–7 and EBCDIC codes.

Recovery at power on : At power on the status of the line procedure is reset to neutral and no indication is given to the user program. No I/O requests to the DC driver are completed for this reason.

DRDC17	BSC POINT-TO-POINT DATA COMMUNICATION	DRDC17

General information : This driver enables point-to-point connection over 2 or 4-wire lines between two PTS 6810 or between PTS 6810 and any other equipment using BSC Contention or optionally Siemens MSV2, protocols. The protocols required must be specified at **system generation** time.

The driver handles traffic in both directions as controlled by the application program. Several lines can operate independently of each other in one PTS 6810. The number of lines must be specified at **system generation** time.

The transmission code can be ASCII as well as EBCDIC. The code conversion is performed by the driver. The code used must be specified at **system generation** time. EBCDIC can only be used with BSC protocols.

Optionally the driver can be adapted for EBCDIC Transparency at **system generation** time. When using this feature non-EBCDIC coding can be transmitted as transparent code. This can only be used with BSC protocols.

The driver will take care of all line control characters such as STX, ETX etc.

Blocking must be done by the application program.

Calling sequence : Normal I/O:

```
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  1
```

Order codes :
/02 – read
/06 – write
/22 – send RVI (reverse interrupt)
/31 – accept call
/37 – connect modem to line
/38 – disconnect line

Buffer address :
Requested length :
Effective length :
} Only significant for orders /02 and /06. For order /06 the first word of the buffer should contain control characters and these should be included in the requested length.

DRDC17 **Continued** DRDC17

Return code : The following bits may be set by this driver:

		Order in which bit is set					
Bit	Meaning	/02	/06	/22	/31	/37	/38
0	Always set for order /38						X
1							
2							
3							
4	WACK count out		X				
5							
6							
7	ETB received	X					
8	End of Transmission (EOT)	X	X	X			
9	Time out*	X	X	X	X		
10	RVI received		X				
11	*		X	X	X		
12	Incorrect length	X		X			
13							
14	Throughput error		X				
15	Modem not operable	X	X	X		X	

* The reason for setting these bits is different for each order, see below.

Bit 0 This bit is always set after order /38.
Bit 1 Not used.
Bit 2 Not used.
Bit 3 Not used.
Bit 4 WACK count out
Set after a reception of several WACKs on a message block. The number of WACKs is specified at **system generation** time.
Bit 5 Not used.
Bit 6 Not used.
Bit 7 ETB received
A message block terminated with ETB has been correctly received.
Bit 8 EOT
The EOT control character has been received.

Bit 9 Order /02 Time-out
No message block has been received within the specified time.
Order /06 ENQ time-out
This bit is set at unsuccessful BID or when there has been no acknowledgement on a message block after several ENQs. The number of ENQs is specified at system generation time.
Order /22 Time out
No response to WACK within the specified time. Driver has transmitted WACK due to missing data request.
Order /31 Request time out

Bit 10 RVI received
RVI has been received as response to the message block. Another block ended by ETX should be sent to allow the other party to transmit.

Bit 11 Order /06 ENQ received
This bit is set at BID collision or when the driver is in receive mode.
Order /22 Sequence error
This bit is set when the driver is in write mode or block including ETX already received.
Order /31 Modem already connected.

Bit 12 Incorrect length
A message block has been received that is longer than the buffer.

Bit 13 Not used.
Bit 14 Throughput error
Bit 15 Modem not operable
Set when the modem has not been connected or is switched off.

Control word : Significant for orders /02, /22 and /31. The control word defines the request time-out value in multiples of 100ms. Control word equal to zero means no request time-out supervision.

Order : /02 – read block
This request enables the driver to receive one block into the user buffer.

DRDC17 **Continued** DRDC17

Order : /06 – write block
The first word of the ECB-buffer contains a right-adjusted control character.

If the control character is **not NUL** the request will cause transmission of one message block terminated by ETB.

If the control character is **equal to NUL (/00)** the request will cause transmission of one message block terminated by ETX. After reception of acknowledgement the driver sends EOT and switches to control state.

If the driver is in control mode, that is neither receive nor transmit mode, a BID sequence is sent to set the receiver in receive mode.

Order : /22 – send RVI
This request is used when the reception of message blocks has to be interrupted. Send RVI request is executed instead of the normal Read request. This means that RVI is sent as a response instead of the normal ACK.

Another block has to be received normally terminated by ETX before the flow of messages is interrupted.

Order : /31 – accept call
This order will test the telephone line for a call signal. If there is a call present it will send a ringing signal back to the caller. It is recommended that this instruction is repeated a number of times to give the caller a definite acknowledgement. If a call signal is not present the order will time-out and bit 9 of the return code will be set. When a call is accepted, the modem should be connected to the line by the connect modem order (/37).

Order : /37 – connect modem to line
The modem is connected to the line, even if already connected.

Order : /38 – disconnect modem from line
The modem is disconnected and another call can be accepted. The return code is always zero.

System generation parameters : In addition to the parameters mentioned in General Information there are several other options that must be specified.

- Interrupt logging. A log routine and one circular buffer for each line are included in the driver. If included the parameter is equal to each buffer size.

DRDC17 Continued DRDC17

- BID time-out. A station sends ENQ as a bid for the line. If two stations are connected, the time-out value should be different in case both stations bid for the line at the same time. This is known as 'line contention'.

  The standard is
  – 1 second for the primary station (Master)
  – 3 seconds for the secondary station (Slave).
- Device addresses. The device addresses are alterable as some addresses may already be defined in the system.

The driver is added to the monitor by including:

- driver
- all interrupt entries in the interrupt vector
- one File Code and one Device Work Table in each Task Table
- Power on entry in the PFTAB.

Recovery at power on : At power on any request is compited with return code bit 15, modem not operable, set. The modems are disconnected.

DRDI01 **SIGNAL DISPLAYS, AND LAMPS ON KEYBOARDS** **DRDI01**

General information : This driver handles output to Signal Displays PTS 6241 and 6242, and lamps on Keyboards PTS 6232, 6233, 6234, 6236 and 6331. All devices must be connected to the CPU via a CHLT or CHRT.

Calling sequence : Normal I/O:

```
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  1
```

I/O and Activate:

```
LDK   A1, parameter
LDK   A7, code
LDKL  A8, ebc-address
LKM
DATA  – 1
DATA  start address
```

Order codes : The following orders may be used:

/05 – write lights on
/06 – write lights off
/06 – write program display–PTS 6241 only (CREDIT)
/07 – write program display–PTS 6241 only
/37 – write lights on (CREDIT)
/38 – write lights off (CREDIT)
/39 – write flashing lights.

Buffer address :
Requested length :
Effective length :
} Only significant for orders /07 and /06 when used with a program display on PTS 6241.

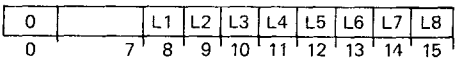
Return code : The following bits may be set by this driver:

Bit	Meaning
0	Illegal request
13	Code check error (only order/07)
15	Not operable. Power off

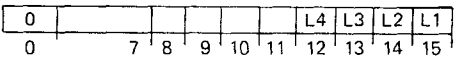
Control word : The control word specifies the lamps that will be affected at orders /05, /37, /06, /38 and /39. The bit pattern has a different meaning for different devices, and is specified below. Lamp L1 is the leftmost lamp on each device.

DRDI01 **Continued** **DRDI01**

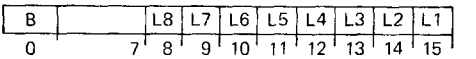
Control word for PTS 6241 and 6242.



Control word for PTS 6232 and 6234

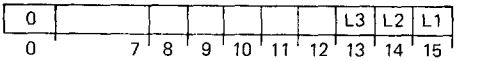


Control word for PTS 6233

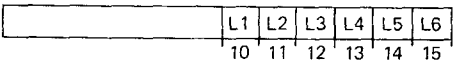


If B = 1, a buzzer is sounded at the keyboard.
This will be done for orders /05, /37, /06 and /38.

Control word for PTS 6331:



Control word for PTS 6236:



Order : /05 or /37 – write lights on:

Order /37 must be used if a CREDIT application is indicated during **system generation.**

Lights corresponding to "1" bits in the control word are turned on. Other lights are not altered.

Order /06 or /38 – write lights off:

Order /38 must be used if a CREDIT application is indicated during **system generation.**

Lights corresponding to "1" bits in the control word are turned off. Other lights are not altered.

DRDI01 Continued DRDI01

Order /07 or /06 – write program display-PTS 6241 only:

/06 must be used if a CREDIT application is indicated during **system generation.**

With these orders the program display on PTS 6241 can be controlled. 1 to 4 characters are sent to the display from the user buffer. Character codes must be in the range /30 to /6F, where:

/30–/3F are sent to the first position
/40–/4F are sent to the second position
/50–/5F are sent to the third position
/60–/6F are sent to the fourth position

The first position corresponds to the leftmost display tube on the indicator unit. Illegal character codes are ignored and bit 13 is set in the return code.

If order /07 or /06 for PTS 6241 is required it must be requested during **system generation.**

Order : /39 – write flashing lights:

Lights corresponding to "1" bits in the control word are lit up once a second. Other lights are not altered. If order /39 is required it must be requested during **system generation.**

Recovery at power on : At power on, all lights are fed with the value existent at power failure time. The information on the program display is also restored.

Note: If an attempt is made to send characters to a device that is not active (power off) the request is immediately completed with bit 15 set in the return code. Thus, to test a terminal line, order /05, /37 /06 or /38 with control word set to zero, can be used.

DRDN01 NUMERIC DISPLAY **DRDN01**

General information : This driver handles the numeric display on indicator unit PTS 6241 connected to the CPU via CHLT or CHRT.

The display may be used as an ordinary output device where numeric information is displayed from the user program. It may also be used as an echo-device, to any keyboard that works under the keyboard driver DRKB01.

Calling sequence : Normal I/O:

```
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  1
```

I/O and Activate:

```
LDK   A1, parameter
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  – 1
DATA  start address
```

Order codes : The following orders may be used:

/05 – erase numeric display
/06 – write numeric display

Buffer address : }
Requested length : } Only significant for order /06.
Effective length : }

Return code : The following bits may be set by this driver:

Bit	Meaning
0	Illegal request
13	Code check error (only order /06)
15	Not operable

Control word : Not significant

Order : /05 – erase numeric display:

The entire display is erased.

Order : /06 – write numeric display:

Characters in the user buffer are sent to the display. Only digits (/30 – /39) and blank (/3F) are accepted. All other codes are ignored and bit 13 is set in return code. Order /06 can be excluded during **system generation** if the display is only to be used as an echo device.

DRDN01	Continued	DRDN01

Echo function : The numeric display may be attached to a keyboard as an echo-device. Only numeric read should be used when echo is wanted at the display.

All received digits are echoed. Clear key (code /18 from the keyboard driver) erases the display. End of record key is echoed if it is a digit. However, the end of the record key will not erase the display. This should be done from user program with order /05.

Recovery at power on : At power on the display is erased. If the old information is to be restored, it has to be done from the application program.

Note: If an attempt is made to send information to a display that is not active (power off) the request is immediately completed with bit 15 set in the return code.

DRDU01 **DISK** **DRDU01**

General information : This driver handles up to two disk drives PTS 6875/PTS 6876 connected to the CPU via a MUX and CHDU on a multiplex channel. The number of disk drives should be specified during **system generation.**
Logically the cartridge disk and the fixed disk on one drive are independent of each other. However, only one disk can be operated at a time.

Each disk has its own file code.

Data is stored per sector with 205 words in each sector. The first word is reserved for a cylinder identifier, thus leaving 204 words on each sector for the user.

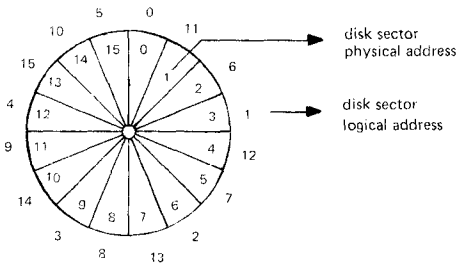
Each disk for PTS 6875 has 32 sectors per cylinder, and a total number of 204 cylinders. Thus, the total number of sectors on one disk is 6528.

Each disk for PTS 6876 has 32 sectors per cylinder and a total number of 408 cylinders. Thus, the total number of sectors on one disk is 13056.

When reading/writing a sector each sector is given a logical sector number from 0 – 6527/13055.

The physical and logical numbering systems are described briefly below and a full description is given in the manual DOS 6810 System Software M11.

There are two sector addressing systems used by the Monitor - physical addressing and logical addressing. Using physical addresses, sectors are numbered according to their physical sequence on the on the disk. Using logical addresses, sector numbers are interlaced on a "factor minus three" basis. This is done to give programs enough time to process the current sector before reading or writing the next sector. The following diagram illustrates this point:



16 sectors

DRDU01 **Continued** **DRDU01**

Calling sequence : Normal I/O:

```
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  1
```

I/O and Activate:

```
LDK   A1, parameter
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  – 1
DATA  start address
```

Order codes : The following orders may be used:

/00 – test status
/01 – basic read
/05 – basic write
/11 – physical read
/15 – physical write

Buffer address : Only significant for orders /01, /05, /11 and /15. The first word in the buffer must be reserved for a cylinder identifier.

Requested length : Only significant for orders /01, /05, /11 and /15. The value
Effective length : of the length should always be 410.

Return code : The following bits may be set by this driver:

Bit	Meaning
0	Illegal request (e.g. illegal order, requested length or sector number)
6	Seek error. The seek operation is not correctly executed.
8	New volume loaded, request aborted. This bit will be set, at first request after the disk drive has become operable and the driver detects that a new volume has been loaded.
12	Incorrect length. End of sector is found before end of exchange.
13	Data fault. CRC error is detected.
14	Throughput error.
15	Not operable.

DRDU01 Continued **DRDU01**

Control word : Only significant for orders /01, /05, /11 and /15. It contains the logical sector number (0–6527) of the sector to be read/written.

Order : /00 – test status:

The disk drive status is checked and bit 15 in the return code is set if it is not operable.

At the completion of this request the control word holds an address pointing to a 6 character area in the Monitor where the volume name of the last loaded volume is stored.

Order : /01 – basic read:

One sector, defined in the control word is read into the user buffer. No recovery is carried out by the driver.

Order : /05 – basic write:

In the first word of the user buffer the driver will store the cylinder identifier before the sector is written. No recovery is carried out by the driver.

Order : /11 – physical read:

One sector, defined in the control word is read to the user buffer. The first word in buffer will contain a cylinder identifier, and is checked by the driver.

If a seek error, incorrect length, data fault or throughput error is detected a new read is carried out. If an error still remains after four attempts, the request is completed and the return code is set to the appropriate value.

Order : /15 – physical write:

Before the requested sector N is written the logical sector N–1 is read to check the cylinder identifier. In the first word of the user buffer the driver will store the cylinder identifier before the sector is written.

After the sector is written it will be read again to check the CRC character.

Read after write, if required, must be requested during **system generation.**

If a seek error, incorrect length, data fault or throughput error is detected the read-write-read sequence is repeated. If after four attempts an error still remains, the request is completed, and the return code is set to the appropriate value.

Recovery at power on : If there is a running request when power failure occurs, the request is immediately completed and bit 14 "throughput error" is set in the return code.
No recovery is carried out by the driver.

After power failure the disk drive will remain not operable until it is started manually.

DRDY01 **VIDEO AND PLASMA DISPLAYS** DRDY01

General information : This driver handles output to the Video Display PTS 6344 and to the Plasma Display PTS 6351. The displays to be handled must be specified during **system generation.** The display must be connected to the CPU via CHLT or CHRT. If a remote connection is made output is possible in block transmission. However, if a TFU is installed on the line, block transmission is not possible.

The driver includes device dependent echo-functions which makes it possible to use the display as an echo-device to any keyboard, that runs under the keyboard driver DRKB01.

Calling sequence : Normal I/O:

```
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  1
```

I/O Activate:

```
LDK   A1, parameter
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  – 1
DATA  start address
```

Order codes : The following orders may be uses:

/00 – test status
/05 – basic write
/06 – standard write
/07 – graphic write
/0B – set cursor and write
/31 – erase

Buffer address : Only significant for orders /05, /06, /07 and /0B. For orders /06
Requested length : and /0B the first word in the buffer is used for a control code.
Effective length : This word is included in the length. For orders /05 and /07 the first word in the buffer is used for normal output data.

Return code : The following bits may be set by this driver:

		Orders in which bits are set					
Bit	Meaning	/00	/05	/06	/07	/0B	/31
0	Illegal request	X	X	X	X	X	X
13	Code check error	X		X			
14	Throughput error		X				
15	Not operable	X	X	X	X	X	X

Control word : Only significant for orders /0B and /31. It contains cursor position for order /0B or number of characters to erase for order /31.

DRDY01 Continued DRDY01

Order : /00 – test status:

A dummy character is sent to the display and bit 15 of the return code is set if the display is not operable (power off).

The actual cursor position is returned in the control word with the line number in the left byte and the column in the right byte.

Order : /05 – basic write:

The requested number of characters are sent to the display without any check. Trailing spaces are suppressed unless they are requested during **system generation.**

If LRC-error occurs for remotely connected displays the request is completed and bit 14 "throughput error" is set in the return code.

Note: the internal cursor position counter of the driver is not updated to correspond to the actual cursor position. Instead it is set to the home position (/0101) after each basic write request, and the value /0101 is returned in the control word when the request is completed.

This implies for display PTS 6351, that the cursor must be sent to the home position using order /0B, before orders /06 standard write or order /0B set cursor and write is used again after a basic write.

For display 6344 order /0B may be used directly since this display uses absolute cursor addressing.

Order : /06 – standard write:

First word in the user buffer is reserved for control information. It must contain one of the following codes in the right hand byte:

/2B: Cursor is not moved before the text is displayed.

/30: Cursor is sent to the leftmost position and advanced two lines before the text is displayed.

/31: The display is erased and the cursor is sent to its home position before the text is displayed.

All other codes will cause the cursor to be sent to the leftmost position and advanced one line before the text is displayed (CR/LF). Trailing spaces are suppressed unless they have been requested during **system generation.**

All alphanumeric characters within /20–/5F, in the user buffer, are accepted and sent to the display. Codes /60–/7F are reduced by /20 giving /40–/5F, if lower case is not requested at **system generation.**

DRDY01 **Continued** DRDY01

Certain special characters may appear in the buffer to control the output operation. A list of these characters is given at the end of this section. These characters, if used, should be counted in the requested length.

Illegal character codes in the user buffer are ignored and bit 13 is set in the return code.

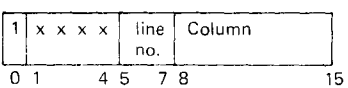
After the write request the actual cursor position is returned in the control word, with line number in left byte and column in right byte.

If LRC-error occurs for remotely connected terminals the message is retransmitted by the driver. No indication is given in the return code.

Order : /07 – graphic write:

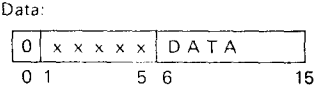
To control the graphic part of the plasma display PTS 6351 order /07 is used.
The display is switched to graphic mode and information in the user buffer is transmitted.
Two types of information may be present in the user buffer: "address" and "data".

- Address:



bit 0 = 1 : Indicating address
bit 1 – 4 : Not significant
bit 5– 7 : Line number (0 – 7) where next data word will be displayed
bit 8–15 : Column (0–255) where next data word will be displayed.

- Data:



bit 0 = 0 : Indicating data
bit 1– 5 : Not significant
bit 6–15 : 10 bits of information, where each bit set will light up a dot in the addressed line and column. Bit 15 corresponds to the lowest dot within a line.

Each address or data word will result in output of two characters to the display.

It is not possible to rub out information from the screen, when the display is in graphic mode.

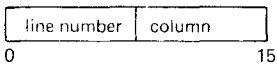
The display will be switched to alphanumeric mode if any of the orders /00, /05, /06 and 0B is given. The cursor will appear in home position when switching from graphic to alphanumeric mode.

After power fail the display will be in alphanumeric mode.

If graphic mode is required (order /07) it must be requested during **system generation.**

Order : /0B – set cursor and write:

By means of this order the cursor may be sent to any position on the screen before the text is displayed. No information on the display is erased. The cursor position is given in the control word as two binary values. The left byte contains a line number and the right byte contains the position within a li /0101 is the cursor home position. (PTS 6344 20 lines, 64 char/line · option 24 lines, 80 char/line . PTS 6351, 8 lines 36 char/line)
Control word:



After the cursor is positioned, the text in user buffer is displayed according to the same rules as for order /06 standard write. The first word in the buffer is not significant but should be included in the requested length. Requested length set to zero or two will cause cursor positioning only.

After the request the new cursor position is returned in the control word.

As for order /06 LRC-errors are handled by the driver.

Order : /31 – erase:

By means of this order a number of characters can be erased on one line. For PTS 6344 this is done in fast output mode. The order is executed on the line at which the cursor is positioned and erasing starts at the cursor position. The cursor remains in that position. The number of characters to be erased is given in binary form in the control word (1–64/80 characters for PTS 6344, 1–36 characters for PTS 6351).

DRDY01 **Continued** DRDY01

Echo function : The display may be attached to a keyboard as an echo-device. All alphanumeric characters within the range /20–5F are echoed. Echo of end of record key must be requested during **system generation** if required.

Backspace key (code /08 from the keyboard driver) will move the cursor one step to the left. The cursor cannot be moved further to the left than the position it had before the read with echo-request. Cursor movement is destructive.

Clear key (code /18 from the keyboard driver) will erase the information that has been echoed for the running read request and the cursor is sent to the position it had before the read-with-echo request.

Recovery at power on : If there is a running request when power fail occurs the request is immediately completed and bit 15 "not operable" is set in the return code.

At power on the display is erased and the cursor is sent to its home position. Power on is also signalled to the application via the keyboard driver DRKB01.

If a write request is done to a display which has power off on the selector unit and/or the display itself, this request is immediately completed with bit 15 "not operable" set in the return code.

Control characters for Video and Plasma Displays : **General**

The following special characters may appear in the buffer to control the output operation.

Characters Valid for All Displays

/AE: Displayed as point (/E2)
/11: Tabulation character. This character should be followed by two ISO–7 digits giving the tabulation position.
/07: Bell is sent to the display.

Characters Valid for PTS 6344 and 6351 only

/0A: Cursor down (line feed).
/08: Cursor left, non-destructive (backspace)
/10: Cursor right, non-destructive.
/20: Cursor right, destructive.
/0B: Cursor home.
/0C: Clear and cursor home.
/0D: Carriage return.

Characters Valid for PTS 6344 only

/12: Underline start. Output of characters which follow this character are provided with underline.

DRDY01

Continued

DRDY01

/13: Underline stop. Output of characters which follow after this character are *not* provided with underline. Underline stop mode will also appear at request end.

/14: Fast output. First character following /14 will be transmitted in fast output mode up to requested length. Note that cursor will remain unchanged.

/1C: Data to keyboard.

/1D: Master clear to keyboard.

/1E: Low intensity start. Output of characters which follow after this character, are displayed at low intensity.

/1F: Low intensity stop. Output of characters which follow after this character are displayed at normal intensity. Normal intensity mode will also appear at request end.

DRFD01 **FLEXIBLE DISK** **DRFD01**

General information : This driver handles up to four chained flexible disk units connected to CPU via channel unit CHFD on multiplex or programmed channel, according to the option specified during system generation. For PTS 6805 only one or two flexible disk drives are included in the configuration.

Logically the disk drives are independent of each other. However, physically only one disk drive can be operated at a time.

Each disk drive has its own filecode.

On the physical level the Flexible Disk should be pre-formatted compatible to IBM 3740. Data is stored per sector with 128 bytes in each sector. Each disk has 26 sectors per track and a total number of 77 tracks. Thus, the total number of sectors in one disk is 2002.

In the text three different abbreviations are frequently used:

Standard: refers to TOSS-labelled disks when the standard type of driver is used.

DM: refers to TOSS-labelled disks when the Data Management type of driver is used.

IBM: refers to IBM-labelled disks.

Types of disk : Two different types of disks are handled by the driver, TOSS- and IBM-labelled disks.

TOSS-labelled disks
When physical reading/writing a sector, each sector is given a **standard** logical sector number from 0-2001. One to four sectors may be read/written at the same request.
Note: Different logical sector numbering may be used, according to the type of driver used, see below, "Types of Driver."

IBM-labelled disks
The driver provides means for sequential access to IBM-labelled disks. **One** Data Set per drive may be sequentially read from or written onto disk. However, it is the user's responsibility to ensure that track 00 contains a correct label before using the IBM-labelled disk. It is possible for the user to create a Data Set and write all necessary labels onto track 00 by means of TOSS utility program WRITE IBM LABELS.

The data fields of the IBM-labelled disk affecting the driver are:

- Volume ID-field
- Beginning of Extent (BOE) of specified Data Set Label
- End of Data (EOD) of Specified Data Set Label
- End of Extent (EOE) of specified Data Set Label

The driver can only affect the EOD-field of the disk.

DRFD01 | Continued | DRFD01

Random access to an IBM-labelled disk is also supported by the driver. Up to four sectors at a time can be addressed and a **standard** logical number from 0—1923 should be given.

All data on IBM-labelled disks should be EBCDIC-coded and the conversion EBCDIC<—>ASCII is performed by the driver.

The driver keeps track of which type of disk is put into each drive by checking the volume label when load request is performed. Thus, any mixed combination of IBM/TOSS-labelled disks is permitted (if IBM-labelled disk handling is specified during **system generation**).

Types of driver : At system generation time it is possible to specify which of two types of driver is required, "Standard" or "Data Management" type. Each type uses the flexible disk in a different manner.

The Standard driver uses standard logical sector numbers from 0—2001. One standard logical sector corresponds to one physical sector of 128 bytes. IBM-labelled disk handling is not included. TOSS-labelled disks should be used, but any different labelled disk may be used, provided the IBM-labelled disk handling is not included. However, the driver will then always assume that the Volume Name is stored at the beginning of sector 0.

The Data Management driver uses DM logical sector numbers from 0—499. Four consecutive physical sectors are combined into one logical sector of 512 characters. However, only 410 of these are used. Thus, to convert a DM logical sector to a standard logical sector the former should be multiplied by four.

Only TOSS-labelled disks should be used.

During **system generation** IBM-labelled disk handling may be included in the Standard or DM-driver. If IBM-labelled disk handling is included, it will not affect the access to TOSS-labelled disks. Thus, for TOSS-labelled disks the logical sector numbering stated above is valid although Standard logical sectors from 0—1923 are used for IBM-labelled disks.

Calling sequence : Normal I/O only

```
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  1
```

DRFD01 **Continued** **DRFD01**

Order codes : The following orders may be used:

/00	test status
/01	basic read
/02	sequential read
/05	basic write
/06	sequential write
~~/0C~~	write deleted data
/11	physical read
/15	physical write
/1A	search key
/26	lock
/31	rewind
/37	load
/38	unload

Buffer address : Only significant for orders /01, /02, /05, /06, /0C, /11 and /15.

Requested length : When writing onto disk, requested length should be:

Effective length :
Standard: $\leqslant$[(N+1) x 128] where N = 0–3
DM: $\leqslant$410
IBM: $\leqslant$[(N+1) x 128] where N = 0–3

However, requested length at sequential read/write should always be 128.

Control word contains the logical sector number to be read/written.

Return code : The following bits may be set by this driver:

		Orders in which bit is set											
Bit	Meaning	/00	/01	/02	/05	/06	/0C	/15	/1A	/26	/31	/37	/38
0	Request error	X	X	Y	X	Y	X	X	X	X	Y	X	X
1	Key not found								X			Y	
2	Zero												
3	End of data			Y									
4	No data		X	Y					X				
5	End of extent					Y							
6	Write protected	X			X	Y	X	X					Y
7	Retries performed		X	Y	X	Y	X	X	X	X		X	Y
8	Zero												
9	Zero												
10	IBM label	Y								Y		Y	
11	Zero												
12	Incorrect length		X	Y	X	Y	X	X	X				
13	CRC-error		X	Y		Y	X	X	X	X		X	Y
14	Seek error		X	Y	X	Y	X	X	X	X		X	Y
15	Not operable	X	X	Y	X	Y	X	X	X	X	Y	X	X

X Standard/DM/IBM
Y IBM only

DRFD01

Continued

DRFD01

Bit	Meaning
0	*Request error* (e.g. illegal order, order not accepted, requested length or sector no.).
1	*Key not found* Search key request: This bit will be set if no record is found within the given limits or if the key has not been found and records could not possibly be read correctly within the limits of the search. Load request: This bit will be set if the contents of BOE/EOD/EOE-fields were impossible to transform to approved logical sector numbers.
3	*End of data* Sequential read request: This bit will be set if a record with a number greater/equal than EOD (end of data) is addressed. Request is aborted.
4	*No data* If any of the read sectors has a deleted data address mark this bit will be set.
5	*End of extent* Sequential write request: Set if an attempt is made to write outside physical space reserved for the data set at creation time.
6	*Write protection* This bit set indicates write protected flexible disk (hole in the envelope).
7	*Retries performed* Retries have been made by hardware due to CRC or seek errors.
10	The disk has an IBM label.
12	Incorrect requested length.
13	*CRC error (data fault)* This bit is set if CRC error still remains after recovery performed by hardware.
14	*Seek error* This bit is set if the requested track is not found after recovery performed.
15	*Not operable* If bits 0 and 15 are set, the request has been aborted due to unlocked drive.

DRFD01 **Continued** **DRFD01**

Control word : The control word requirement can differ in each order depending upon which kind of disk is being handled. Refer to the notes of each order.

Order : /00 – Test status

The addressed disk drive is selected and its status is checked.

At return, control word holds an address pointing to a 6 character area in monitor where the volume name is stored. However, if at return bit 0 or 15 of return code is set, the volume name might not be significant.

Actual Return Code in ECB may be:

- Bit 0 Request error
- Bit 6 Write protected
- Bit 10 IBM-label
- Bit 15 Not operable

The read/write head is not moved by this order.

Order : /01 – Basic read

STD/IBM – One to four sectors are read into the user buffer. The requested length should be ⩽ [(N+ 1) x 128] where N = 0–3, the numbers of sectors.

DM – One DM-sector or part of one DM sector is read into the user buffer. The requested length should be ⩽ 410.

The control word should contain the number of the first sector to be read. All recovery is performed by hardware.

Order : /02 – Sequential read

STD/DM – Not used.

IBM: The CRN is incremented by one and the record pointed to by CRN is read to the user buffer.

Requested length should be 128.

Control word will at return contain the standard logical sector number of the addressed sector.

If a record with a number greater than or equal to EOD is addressed, the request is aborted and bit 3 of return code is set.

This request is only accepted if the corresponding data set has been opened by a LOAD request.

DRFD01 **Continued** **DRFD01**

Order : /05 — Basic write

STD/IBM: One to four sectors are written from the user buffer. The number of sectors to be written is given in the requested length as $\leq [(N+1) \times 128]$ where N = 0–3, the number of sectors.

DM: One DM sector is written from the user buffer. The requested length should be 410.

The control word should contain the logical sector number of the first sector to be written. All recovery is performed by the hardware.

DRFD01 **Continued** **DRFD01**

Order : /06 – Sequential write

STD/DM: Not used

IBM: One sector is written from the user buffer to the sector pointed to by the corresponding EOD number in the driver. The EOD number is incremented by one.

Requested length should be 128.

Control word will at return contain the standard logical sector number of the addressed sector.

If an attempt is made to address a sector behind the EOE, the request is aborted with bit 5 of return code set.

Order : /0C – Write deleted data

STD/IBM: One to four sectors are written from the user buffer. The requested length should be $\leqslant$[(N+1) x 128] where N = 0–3, the number of sectors.

DM: One DM sector is written from the user buffer. The requested length should be $\leqslant$410.

The addressed sectors are preceded by a 'write deleted data address mark'. The control word should contain the logical sector number of the first sector to be written. Read after write is performed by hardware as is all recovery after write/seek errors.

Order : /11 – Physical read

STD/IBM: One to four sectors are read to the user buffer. The number of sectors to be written is given in the requested length as $\leqslant$[(N+1) x 128] where N = 0–3, the number of sectors.

DM: One DM sector is read to the user buffer. The requested length should be $\leqslant$410.

The control word should contain the logical sector number of the first sector to be read. All recovery from read/seek errors is performed by hardware.

Order : /15 – Physical write

STD/IBM: One to four sectors are written from the user buffer. Number of sectors to be written is given in requested length of ECB. [(N+1) x 128], N = 0–3.

DM: One DM-sector is written from the user buffer. The requested length should be 410.

The control word should contain the logical sector number of the first sector to be written.

Read after write is performed by hardware and also all recovery from write/seek-errors.

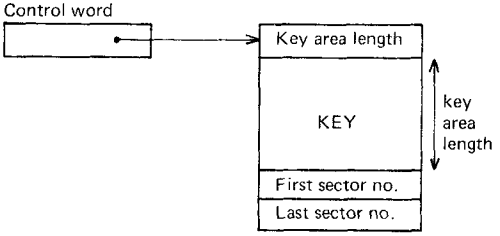
Order : /1A - Search key

DM/IBM: Not used.

STD: This order is used to find and read one particular record, of which the first data are identical to the key.

The key must be defined in an area pointed to by the control word. The first word of this area should contain the key area length.

The key must end with two words defining the first and the last record between which the search is to take place. The control word points to the key buffer area.



Requested length should be 130.

Note: If the key itself contains an odd number of characters, the last character is rejected as not significant.

The data buffer area pointed to by the ECB will contain the requested record preceded by a word containing the record number. Thus, the buffer length used should be 130 characters.

Recovery from read/seek errors is performed by hardware.

Order : /26 – Lock

The driver locks the door of the selected drive. If IBM-labelled disk handling is included in the driver, standard logical sector 0 is read to check whether this disk is TOSS-labelled. If it is not, the driver assumes the disk is IBM-labelled. The volume name is read by the driver and saved.

If an error occurs and the driver is unable to read the volume name, the disk drive will be unlocked at return.
This order must be successfully executed before any other order is accepted for this disk drive.

STD/DM: The control word is not significant.

IBM: The control word should contain the standard logical sector number of the data set label (between 8 and 25 inclusive). The data set is not opened.

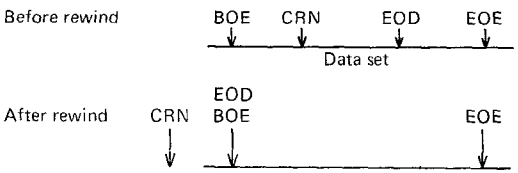
Order : /31 – Rewind

STD/DM: Not used.

IBM: The EOD-number of selected drive is set equal to the BOE number. The current record number is set equal to the BOE –1.

This order is only accepted if the corresponding data set has been opened by a Load request.

The read/write head is not moved by this request.



Order : /37 – Load

The driver locks the door of the selected drive. If IBM-labelled disk handling is included in the driver, standard logical sector 0 is read to check whether this disk is TOSS-labelled. If it is not, the driver assumes the disk is IBM-labelled. The volume name is read by the driver and saved.

If an error occurs and the driver is unable to read the volume name, the disk drive will be unlocked at return.

This order must be successfully executed before any other order is accepted for this disk drive.

STD/DM: The control word is not significant.

IBM: The control word should contain the standard logical sector number of the data set label (between 8 and 25 inclusive).

After the driver has taken the actions described above, the BOE, EOD and EOE fields of the specified data set label are read from the disk. The contents of these fields are transformed to standard logical sector numbers and saved in the driver. In the text these numbers saved in core are called BOE-no, EOD-no, and EOE-no respectively. If any of these numbers is not approved by the driver bit 1 of the return code is set and the drive is unlocked.

The current record no. (CRN) is set equal to the BOE −1.

Note: For a Sequential Read/Write request no data is affected of the data set label on disk. However, the CRN or the EOD-no in the driver is updated. Random access does not affect the data set handling.

Order : /38 – Unload

Door of selected drive is unlocked. Requested length, buffer address and control word are not significant.

IBM: Before unlocking, the driver checks if any data set was opened for this drive. If so the EOD-field of the data set label is updated to the last sector number addressed by a sequential write + 1.

Recovery at power on : All doors locked at the time of power failure will be locked during power off and after power on.

If there was a running request at the time of power failure, the driver will repeat this request. If not successful bit 13 of return code will be set.

DRGP01 GENERAL TERMINAL PRINTER DRGP01

General information : This driver handles General Terminal Printer PTS 6321 connected to the CPU via CHLT or CHRT.

The driver also includes a device dependent echo-function which makes it possible to use the general printer as an echo-device to any keyboard that runs under the general keyboard driver DRKB01.

Calling sequence : Normal I/O:

```
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  1
```

I/O and Activate:

```
LDK   A1, parameter
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  – 1
DATA  start address
```

Order codes : The following orders may be used:

/00 – test status
/05 – basic write
/06 – standard write

Buffer address :
Requested length : Only significant for orders /05 and /06. For order /06 the first word in the buffer should be reserved for a control code.
Effective length : This word is included in the length. For order /05 the first word in the buffer is used for normal output data.

Return code : The following bits may be set by this driver:

		Orders in which bits set		
Bit	Meaning	/00	/05	/06
0	Illegal request	X	X	X
13	Code check error			X
14	Throughput error		X	
15	Not operable, power off	X	X	X

Control word : Not significant.

Order : /00 – test status:

A dummy character is sent to the printer. If time-out is signalled by the channel unit bit 15 of the return code is set.

Common functions for orders /05 and /06:

Continuation of request when the selector unit or printer is inactive may be requested during system generation (bit 15 set in return code).

Order : /05 – basic write:

The requested number of characters are sent to the printer without any check. Trailing spaces are suppressed unless they are requested during **system generation.**

Order : /06 – standard write:

First word in the user buffer is reserved for control information. It can contain one of the following codes in the right hand byte:

/2B: print the line without advancing the paper. The print-head is not moved before printing the text.
/30: advance the paper two lines before print-out and make carriage return.

All other codes will cause carriage return and line feed before print-out.

All alphanumeric characters within the range /20–/5F, in the buffer, are accepted and sent to the printer. Codes /60–/7F are reduced by /20 giving /40–/5F. Furthermore, the following special characters may appear in the buffer to control the output operation:

/AE: Point is printed as roomless. That is, the digit after /AE is code converted and printed as a roomless point digit (point is placed to the left of the digit).

If roomless point is excluded from the driver /AE is printed as an ordinary point.

/13: This code is sent directly to the printer. By hardware, this will cause a special symbol to be generated.
/14: Same as for code /13.
/11 Tabulation character. This character should be followed by two ISO–7 digits giving the tabulation position.
/09: Hardware tabulation. Note: hardware tabulation will reset the head position counter included in the driver. This may cause text to be overwritten in recovery situations.

The special characters and the tabulation position should be included in the requested length.

Illegal character codes in the user buffer are ignored, and bit 13 is set in the return code. A dot is printed in place of the illegal code and printing continues.

Trailing spaces are suppressed at print-out unless they are requested during **system generation.**

DRGP01 Continued DRGP01

If special characters /13 and /14 are to be used they should be requested during **system generation.** Non standard character codes for roomless point digits can also be specified during **system generation.**Alternately the handling of roomless point codes can be suppressed.

Echo function : The General Printer may be attached to a keyboard as an echo-device. All alphanumeric characters within the range /20—/5F are echoed. Each character is echoed together with a space to get visibility of the last printed character. Space must be requested during **system generation** if it is required.

End of record character is echoed if it has a code within the range /20—/5F for standard read or /30—/39 for numeric read. This must be requested during **system generation** if it is required. Backspace key (code /08 from the keyboard driver) is echoed and will be represented by an underline character. After the echoed backspace, the print head advances to the next character position. Cancel key (code /18 from the keyboard driver) is echoed. The paper is then advanced one line and the print head is sent to the position it had before the read with echo-request. Head positioning is carried out with backspace.

The codes to be used for backspace key, cancel key, end of record key and multiple zero key must be specified for the associated keyboard during **system generation** (terminal device class/echo device class).

Recovery at power on : At power up the following actions are taken:

- If the order is /06, the print head is sent to the position it had before the write-request and the line is printed once more. No indication is given in the return code.
- If the order is /05, bit 14 is set in the return code and the request is completed.
- If the printer is in echo-mode no recovery is carried out.

DRKB01 KEYBOARD **DRKB01**

General information : This driver handles input from numeric and alphanumeric keyboards PTS 6231, 6232, 6233, 6234, 6331 and 6342 connected to CPU via CHLT or CHRT.

Only input data from depressed keys are handled by this driver. Output data to signal indicators are processed by a special Signal Display Driver (DRDI01).

For every keyboard in the system there is a circular input buffer, where depressed keys are stored if no read request is running. When a read request is set up the information in this buffer is transferred to user buffer.

An echo-device can be attached to every keyboard.

Calling sequence

Normal I/O:

```
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  1
```

I/O and Activate:

```
LDK   A1, parameter
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  – 1
DATA  start address
```

Order codes : The following orders may be used:

/01 – basic read
/02 – standard read
/03 – numeric read
/04 – SKip circular input buffer
/31 – SKip circular input buffer (CREDIT).

Buffer address :
Requested length : } Significant for all order codes. The end of record key is included in the length.
Effective length :

Return code : The following bits may be set by this driver:

		Orders in which bits set				
Bit	Meaning	/01	/02	/03	/04	/31
0	Illegal request	X	X	X	X	X
9	Time-out	X	X	X		
12	Incorrect length		X	X		
13	Undefined key		X	X		
14	Throughput error	X	X	X		

Control word : Significant for orders /02 and /03 only. For these orders the control word may contain a keytable address. The key table contains a list of end of record keys. If the keytable address is zero then no keytable will be used. The format of the key-table is as follows:

byte		
0	No. of EOR keys	key 1
2	key 2	key 3
4	key 4	key 5

etc.

Order : /01 – Basic read:

The requested number of characters are read and stored in the user buffer without any check. If overflow had occurred in the circular input buffer, at time of request the read request is completed with bit 14 set in the return code.

Common functions for orders /02 and /03 : The driver checks every received character in the following sequence:

- If overflow had occurred in the circular input buffer at the time of request, the request is completed with bit 14 set in the return code.
- Received characters are code converted, if this facility is included, before any further handling of the characters.
- If received character is found in keytable the key is stored in the user buffer and the request is immediately completed. The end of record key is also code converted and stored in the control word so that KEY 1 gives 0, KEY 2 gives 2, KEY 3 gives 4 and so on to enable indexing.

  If a CREDIT application is indicated during **system generation** KEY 1 will give 1, KEY 2 will give 2, KEY 3 will give 3 etc.

  If the keytable address is zero in the control word a standard key, KBEOR, is treated as end of record key.
- Special characters are checked (e.g. multiple zero, clear key etc.) and corresponding functions are carried out.
- Alphanumeric/Numeric characters are stored in the user buffer. If overflow occurs in the user buffer the request is completed with bit 12 set in the return code.

- If received character cannot be identified in the tests above, it is treated as undefined and the request is completed with bit 13 set in the return code. The undefined key is stored in the user buffer and the control word remains unchanged.

Order : /02 – Standard read:

Alphanumeric characters within the range /20–/5F or /20–/7F, depending upon the type of keyboard specified during **system generation** (device class definition) are accepted and stored in the user buffer. If standard read is required it must be requested during **system generation.**

Order : /03 – Numeric read:

Only digits within the range /30–/39 and /70–79 are accepted and stored in the buffer (/70–79 are converted to /30–/39 before storing in the user buffer). Characters within the range /70–/79 may be excluded depending upon the type of keyboard specified during **system generation** (device class definition).

Order : /04 or /31 – skip circular input buffer:

Code /31 must be used if a CREDIT application is indicated during **system generation.**

The information in the circular input buffer is deleted, and the request is completed.

Special characters : Some keys have a special meaning to the driver. The codes for these keys can be user defined. For keyboards 6231, 6233 and the numeric part of 6234 the special keys are independent of key switch position. Codes for key switch in position 1 should be used.
The special keys are:

KBEOR: Standard end of record key. Used by the driver when the keytable address is zero.

KBCLR: Cancel key. The user buffer is cleared. This does not cause completion of the request.

KBBSP: Backspace key. The last reported character is cleared in the user buffer. This does not cause completion of the request.

KBMZ: Multiple zero. Two or three zeros are stored in the user buffer, when this key is received. The number of zeros to be stored must be indicated during **system generation.**

DRKB01 Continued DRKB01

The codes to be used for backspace key, cancel key, end of record key and multiple zero key must be specified for the associated keyboard during **system generation**.

Echo function : Input characters are echoed if the E-bit is set in register A7 during the LKM-request. The read request is accepted only when the echo-device is free. Else the request is queued. The device-dependent echo-function is included in the driver for the output device, and is not described here.

If echo is required it must be requested during **system generation** and an echo device must be associated with the keyboard.

The following information is transferred to the echo-device from the keyboard driver:

- Standard read
  Characters within the range /20–/5F or /20–/7F depending upon the type of keyboard specified during **system generation** (device class definition).
- Numeric read
  Characters within the range /30–/39.
- Basic read
  Echo not allowed, for code check reasons.
- Special Characters
  (both standard and numeric read)
  /08 Backspace key
  /18 Cancel key

Time out : For each keyboard in a system it is possible to include a time-out function. That is, if a key has not been pressed within a certain time, the request will be ended with bit 9 set in the return code. The timer is restarted for each depressed key. The time before time-out is the same for all keyboards. If time-out is required it must be requested during **system generation.**

Double keyboard handling : If two keyboards, one alphanumeric and one numeric are used at the same work position or if keyboard PTS 6234 is used, these keyboards can be treated as one device, despite the fact that two device addresses are used. To avoid code interference the driver can be adapted to convert all function keys of the numeric part by setting the "8–bit" in the received code. Digits will give codes /30–/39 independent of keyswitch position.

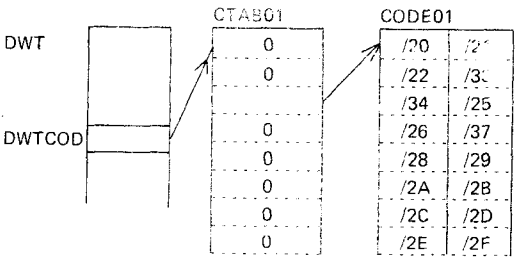
DRKB01 Continued DRKB01

Appendix B shows all codes generated as seen from the application program for keyboard PTS 6234.
Appendix C gives the same information for keyboard PTS 6231.

Note: If keyboard PTS 6231 will work within the same system in conjunction with PTS 6234, input from the 6231 can be code converted by "8–bit" setting to get full code compatibility with PTS 6234.

Code conversion : The driver can, optionally, code convert input characters. That is, the user can at **system generation** time, (device class definition) define one or more columns in the ISO–7 table to be converted for a specific keyboard in the system.

The conversion is achieved with the following table structure.



DWTCOD of the device work table holds an address of an 8–word table CTAB01, indicating the columns to be converted. (DWTCOD) = 0 means no conversion for this keyboard.

Each non-zero entry in CTAB01 gives the address of the code conversion table for the specific column. In the example above codes /23, /24 and /27 are converted to /33, /34 and /37 respectively. The conversion tables can be common for several keyboards.

Only codes between /00–/7F can be converted. So if "8–bit" setting is used (double keyboard handling) these codes cannot be code converted via conversion tables.

Code /FF is used by the driver and must not be used as a converted code.

DRKB01 Continued DRKB01

Completion of read request at power on : If there is a read request this is completed with -2 set in the control word. If not, a power up flag is set causing the first read request after power on to be completed with control word set to -2.
If a CREDIT application is specified during **system generation** power fail indication will be zero instead of -2.

If completion of read request at power on is required it must be requested during **system generation**.

DRKB03 **KEYBOARD** **DRKB03**

General information : This driver handles input from alphanumeric keyboard PTS 6236 connected to CPU via SUM and CHLT/CHRT. Input from keyboards 6231, 6232, 6233, 6234, 6331 and 6342 should be handled by driver DRKB01.
Only input from depressed keys is handled by this driver. Output data to signal indicators are processed by a special Signal Display Driver (DRDI01).
For every keyboard in the system there is a circular input buffer, where depressed keys are stored if no read request is running. When a read request is set up the information in this buffer is transferred to the user buffer.
An echo-device can be attached to every keyboard. The echo function must be defined at system generation time.

Calling sequence : Normal I/O:

```
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  1
```

I/O and Activate:

```
LDK   A1, parameter
LDK   A7, code
LKM
DATA  -1
DATA  start address
```

Order codes : The following orders may be used:

/01 – basic read
/02 – standard read
/03 – numeric read
/31 – skip circular input buffer (CREDIT)

Buffer address :
Requested length :
Effective length :
} Significant for order codes /01, /02 and /03. The end of record key is included in the length.

Return code : The following bits may be set by this driver:

		Orders in which bits set			
Bit	Meaning	/01	/02	/03	/31
0	Illegal request	X	X	X	X
9	Time-out	X	X	X	
12	Incorrect length		X	X	
13	Undefined key		X	X	
14	Throughput error	X	X	X	

DRKB03 **Continued** DRKB03

Control word : Significant for orders /02 and /03 only. For these orders the control word may contain a keytable address. The keytable contains a list of end-of-record keys. If the keytable contains zero at the completion of the request, a power failure has occurred. If it contains a negative value, a key-lock code has been received. A key-lock code cannot be specified as an end-of-record key. The format of the key table is as follows:

byte		
0	No. of EOR keys	key 1
2	key 2	key 3
4	key 4	key 5
	etc.	

Order : /01 – Basic read

The requested number of characters are read and stored in the user buffer without any check. Code conversion is performed according to appropriate conversion table. If overflow has occurred in the circular input buffer at time of the request, the read request is completed with bit 14 "Throughput Error" set in the return code.

Received characters from the key-switch locks (codes /70 – /77) are stored in the user buffer. The internal status indicator of the key-switch position is also updated.

If a power failure occurs during the request no action is taken. (No completion of the request.)

Common functions for orders /02 and /03 : The driver checks every received character in the following sequence:

- If overflow has occurred in the circular input buffer, the request is completed with bit 14, "Throughput Error", set in the return code.
- If the received character derives from a key-lock (codes /70 – /77) the request is completed with a negative value set in the control word. Key-lock code is also stored in the user buffer.
- SHIFT and CTRL keys will only set internal status and are never transferred to the user buffer.
- Received characters are code converted according to the appropriate conversion table before any further handling of the characters.

- If a converted character is found in the keytable that indicates an end-of-record key, the character is stored in the user buffer and the request is immediately completed.
- The EOR-key is also converted and stored in the control word of the ECB so that KEY 1 gives 1, KEY 2 gives 2, KEY 3 gives 3 and so on, to enable indexing.
- Special characters are checked (e.g. multiple zero, clear, etc.) and corresponding functions are carried out.
- If the converted character cannot be identified in the tests above, it is treated as undefined and the request is completed with bit 13 "Undefined Key" set in the return code. The undefined key is stored in the user buffer and the control word remains unchanged.

Order : /02 – Standard read

- Alphanumeric characters within the range of /20–/7F after conversion are accepted and stored in the user buffer. If overflow occurs in the user buffer the request is completed with bit 12, "Incorrect Length" set in the return code.

Order : /03 – Numeric read

The numeric read has the same functions as standard read except that only digits within the range of /30 – /39 after conversion are accepted and stored in the user buffer.

Order : /31 – Skip circular input buffer

The information in the circular input buffer is deleted, and the request is completed.

Special characters : Some keys have a special meaning to the driver. The codes for these keys are user-defined and the converted codes should always be used. The special keys are:

KBCLR:	Clear key. The user buffer is cleared. This gives no completion of the request.
KBBSP:	Backspace key. The last received character is cleared in the user buffer. This gives no completion of the request.
KBMZ2:	Double zero key. Two zeroes are stored in the user buffer.
KBMZ3:	Triple zero key. Three zeroes are stored in the user buffer.

DRKB03 **Continued** DRKB03

Echo function : Input characters are echoed if the E-bit is set in register A7 during the LKM-request and if an echo-device is attached to the keyboard at system generation time. Note that converted characters are echoed. The read request is accepted only when the echo-device is free, else the request is queued. The device-dependent echo function is included in the driver for the output device, and is not described here.

The following information is transferred to the echo-device driver from the keyboard driver:

- Basic read
  All characters but key-lock, SHIFT and CTRL information.
- Standard read
  Characters within the range /20 – /7F.
- Numeric read
  Characters within the range /30 – /39.
- Special characters
  (both standard and numeric read)

/18	Clear key
/08	Backspace key
/30, /30	Double zero key
/30, /30, /30	Triple zero key

Time-out : Each keyboard in a system includes a time-out function. That is, if a key has not been depressed within a certain time, the request will be ended with bit 9 "Time-out" set in the return code. The timer is restarted for each depressed key. The time before time-out is the same for all keyboards. If this facility is required it should be specified at **system generation** time.

Key-lock : Changed key-lock position will be received as input characters. This will always cause a completion of a running Standard or Numeric Read request with a negative value in the control word.

If such a request is not running the completion will be done on subsequent request. If more than one key-lock has been changed only one key-lock change will be reported at a time.

The negative value in control word has the following meaning:

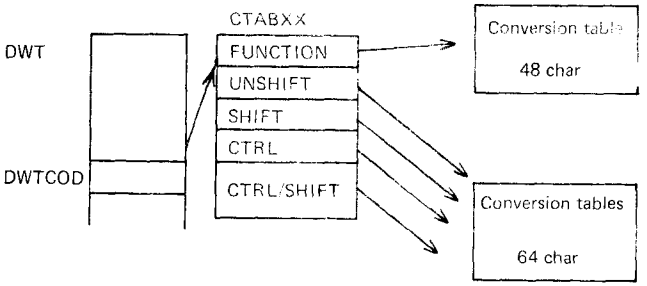
–1	Key-lock no. 4 turned OFF
–2	Key-lock no. 3 turned OFF
–3	Key-lock no. 2 turned OFF
–4	Key-lock no. 1 turned OFF
–5	Key-lock no. 4 turned ON
–6	Key-lock no. 3 turned ON
–7	Key-lock no. 2 turned ON
–8	Key-lock no. 1 turned ON

For Basic Read only the received code is stored in the user buffer and the request is not completed. If all keys are in OFF, the keyboard is regarded as inactive.

Code conversion : To adapt the keyboards to different national keyboard-layouts the driver includes code conversion facilities. The conversion consists of five modes:

- Function/Numeric
- Unshift
- Shift
- CTRL
- CTRL/Shift

The conversion is achieved by means of the following table structure:



Each entry not zero in CTABXX gives the address to the code conversion table. The conversion tables may be used common for several keyboards. One or more conversion tables may be excluded by setting zero or UNSHIFT in corresponding entry in table CTABXX. Zero means no conversion and UNSHIFT conversion according to the UNSHIFT-table.

The code conversion is performed according to the following rules:

Received characters between /00 – /1F and /60 – /6F are converted according to the Function/Numeric conversion table.

DRKB03 Continued DRKB03

Received characters between /20 – /5F are converted according to:

- Unshift table if neither SHIFT nor CTRL-key has been depressed
- Shift table if SHIFT-key has been depressed
- CTRL table if CTRL-key has been depressed
- CTRL/Shift table if both shift and CTRL-key have been depressed.

The conversion tables are set up at **system generation** time.

For the conversion the following recommendations are given:

Function/Numeric – table
The numeric cluster is converted to /30 – /39. All other keys are converted to codes in the range of /80 – /AF.

Unshift – table
If necessary this table is used to adapt the keyboard to different national keyboard layouts. Converted codes should still be in the range of /20 – /5F.

Shift – table
The shift function is realized with this table. Converted codes should be within /20 – /7F or within /20 – /5F if only upper case is used for alpha-keys.

CTRL – table
The CTRL-functions are realized with this table. Converted codes should be within /00 – /1F. The CTRL-function may also be used for special purposes when the alpha-keys are also used as function keys. The converted code shall then be outside the range of /20 – /7F.

CTRL/Shift – table
Same indications as for the CTRL–table. (See above).

Conversion tables
A depressed key on keyboard PTS 6236 generates an internal code. This code is given from the key pad layout see appendix G, for example: Key in position D03 has the code /45 (not converted).
If a conversion table is defined, an index is calculated for every key, see appendix G, for example: Key in position D03 has the index 38 in the UNSHIFT, SHIFT, CTRL and SHIFT/CTRL cluster. If the conversion table in position 38 contains /88, the key in position D03 will give the code /45 without conversion and will give code /88 with conversion.

Index 1 is the first position in the conversion table.

System generation parameters : The following parameters should be defined for the device work table (DWT):

DWTKEY: Codes for special keys KBCLR, KBBSP, KBMZ2, KBMZ3 Note that converted code should be used.

DWTECH: DWT-address of echo-device. Set to zero means no echo device.

DWTTP: Timer-indicator. Set to zero if no timing wanted on this keyboard.

DWTCOD: Address to conversion address table CTABXX

The circular input buffer is also placed in DWT. Its length should be the same in all DWTs, but may be changed between systems.

DRLP01 **LINE PRINTER** **DRLP01**

General information : This driver handles one line printer PTS 6881 on a multiplex or programmed channel. The type of channel should be specified during **system generation.**

Calling sequence : Normal I/O:

```
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  1
```

I/O and Activate:

```
LDK   A1, parameter
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  – 1
DATA  start address
```

Order codes : The following orders may be used:

/00 – test status
/05 – basic write
/06 – standard write

Buffer address
Requested length
Effective length
: Only significant for orders /05 and /06. For order /06 the first word and the last character in the buffer are reserved for control information. For order /05 these parts of the buffer are occupied by normal data.

Return code : The following bits may be set by this driver:

Bit	Meaning
0	Request error
15	Not operable

Control word : Not significant

Order : /00 – test status:
The printer status is tested and bit 15 in the return code is set if not operable.

Order : /05 – basic write:
The requested number of characters are sent to the line printer without any check.
If the buffer of the line printer is full (132 char.), or if a format control character is received, the buffer is printed.
The following format control characters are available:

/0A: advances the paper one line and sets the device at the left-most print position (CR/LF)

/0C: advances the paper to top of form and sets the device at the left-most print position (FF/CR)

/0D: sets the device at the left-most print position (CR)

Order : /06/ – standard write:

The first word in the user buffer is reserved for control information. It can contain one of the following codes in the right hand bytes:

/2B: print the line without advancing the paper (superposition).

/30: advance two lines before printing.

/31: skip to top of form before printing.

All other control codes will advance the paper one line before printing.

At the end of the user buffer one character must be reserved for the system, in which a print code is stored by the driver. This character should **not** be included in the requested length. All other characters in the user buffer should be within /20–/5F but this is not checked by the driver.

Recovery at power on : No recovery is carried out by the driver. If power failure occurs when there is a running print request, this request is completed with bit 15 "not operable" set in the return code.

DRMT01 **MAGNETIC TAPE** **DRMT01**

General information : This driver handles up to eight 1/2 inch magnetic tape recorders, PTS 6872 or 6164 connected to CPU on multiplex channel.

The recorders are operated independently of each other, each having its own file code. However, only one can be working at a time, except at unload.

Data is recorded in blocks with length from 2 to 4095 characters.

Calling sequence : Normal I/O:

```
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  1
```

I/O and Activate:

```
LDK   A1, parameter
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  – 1
DATA  start address
```

Order codes

The following orders may be used:

/00 – test status
/02 – read
/05 – write
/06 – write
/22 – write tape mark
/31 – rewind
/33 – step reverse
/34 – step forward
/37 – load

Buffer address :
Requested length :
Effective length :

Only significant for orders /02, /05 and /06. The last word in each buffer may be used as a block sequence counter. The length must be from 2 to 4095 bytes and must exclude the block sequence counter if this is used.

Return code : The following bits may be set by this driver, see the table on the next page.

DRMT01 **Continued** DRMT01

		Order in which bit is set									
Bit	Meaning	/00	/02	/05	/22	/31	/33	/34	/37	/38	/3F
0	Illegal request	X	X	X	X	X	X	X	X	X	X
1	Zero										
2	Rewinding	X	X	X	X	X	X	X	X		
3	Tape mark		X		X		X	X			
4	No data		X	X	X		X	X			
5	BOT	X				X	X		X		X
6	Write protected	X	X	X	X	X	X	X	X		X
7	Zero										
8	Zero										
9	Hardware error		X	X	X	X	X	X	X	X	X
10	EOT	X	X	X	X		X	X			X
11	Sequence error		X								
12	Incorrect length		X								
13	Data error		X	X			X	X			X
14	Throughput error		X	X							
15	Not operable	X	X	X	X	X	X	X	X	X	X

Control word : Only significant for orders /37, /02, /05, /06 and /3F.

The value of the least significant bit (rightmost) when requesting on order /37 determines whether a block sequence counter will be used:

0 – sequence counter is required

1 – no sequence counter required

The setting of the control word for order /37 will affect the operation of later orders /02, /05, /06 and/or /3F and the recovery procedures at power on.

After a read or write request the control word contains the number of read or write retries performed (orders /02, /05 or /06)

Order : /00 – test status:

The status of the selected recorder is indicated in the return code.

Order : /02 – read:

One block is read from the tape and stored in the buffer. If there is a "data error" or "throughput error" a retry is made. At most three retries are performed.

If used, the block sequence counter is checked and if not correct "sequence error" is set in the return code. Two characters must be reserved at the end of the block for this counter.

If the requested length was less than the actual length "incorrect length" is set in the return code.

When data is not found within two seconds "no data" is set in the return code.

Order : /05 and /06 – write:

One block from the user buffer is written on the tape. If there is a "data error" or "throughput error" a retry is made, after erasing the tape 10 cm from the beginning of the block just written. At most three retries are performed.

When a block sequence counter is used then, before writing the two characters at the end of the buffer are replaced by the counter. These characters are not included in the requested length.

If the tape is write protected the request terminates immediately with "write protected" set. The tape is not moved.

Order : /22 – write tape mark:

One tape mark is written on the tape. Recovery is carried out as in orders /05 and /06 (write).

If the order is successful "tape mark" is set in the return code.

Order : /31 – rewind:

The tape is rewound to beginning of tape (BOT). If BOT is not reached within 3 minutes "rewinding" is set in the return code.

Order : /33 – step reverse:

The tape is reversed one block

Order : /34 – step forward:

The tape is moved one block forward:

It is recommended that this order code is used (or step reverse) when searching for a tape mark since the transport does not delay the CPU.

Order : /37 – load:

The recorder is set on-line and the tape is rewound to BOT. The control word determines if a sequence counter will be used for subsequent orders.

DRMT01 — Continued — DRMT01

If BOT is not reached within 3 minutes "rewinding" is set in the return code.

Note: the PTS 6164 recorder cannot be set on-line from the program, so this must be done by the operator.

Order : /38 – unload:

The tape is rewound and the recorder is switched off-line.

Order : /3F – recover:

The recorder is set on-line and the tape is positioned before the block indicated by the block sequence counter.

If unsuccessful i.e. incorrect block sequence counters on the tape, the recorder is put off-line.

If sequence counters are not used no action is taken.

Recovery at power on : After power fail in the computer a recovery is performed for each recorder which was on-line when the trouble started.

The procedure is the same as in the order "recover".

If the recovery is not successful, due to incorrect block sequence counters the recorder is put off-line.

If sequence counters are not used no action is taken.

When a power fail occurs only in a formatter or recorder this is indicated by the status "not operable" in the return code. It is then possible to put it on-line and recover by using the order "recover".

DRRT01 REMOTE TERMINAL **DRRT01**

General information : The remote terminal driver controls data transfer from the device driver onto the channel units for local and remote terminals.
One special function is available in the remote terminal driver which function can be called by the application program. The test remote line function is valid only for remote connected terminals. The line which connects the channel unit with the selector unit can be tested.

Calling sequence : (LDKL A1, parameter)
LDK A7, code
LDKL A8, ecb-address
LKM
DATA (–)1
(DATA start-address)

Order code : The following order can be used:
/00 -- test remote line

File code : Recommended is file code /15.

Buffer address :
Requested length :
Effective length :
} These parameters are not used.

Return code : The following bits may be set by this driver in the return code word:

Bit	Meaning
0	Request error
9	Channel unit missing or erroneous
14	ACK missing
15	SYNC missing

Control word : The control word contains for this driver identification of the line to be tested.
1 = line of first channel on CHRT1
2 = line of second channel on CHRT1
3 = line of first channel on CHRT2
4 = line of second channel on CHRT2
5 = line of first channel on CHRT3
6 = line of second channel on CHRT3
7 = line of first channel on CHRT4
8 = line of second channel on CHRT4
The control word has to be filled by the application program.

DRRT01 **Continued** **DRRT01**

Order : /00 – test remote line:

This order will test the remote line, if loop connected, by sending a SYNC character each 500 msec. to device address = 7. On return of SYNC from the CHRT an ACK-character is sent and also returned by the loop-connection. A test on receiving ACK is also done.

Information about the state of the line, up to the loop-connection, is specified for this test in the return code word.

The return codes for this order differ as follows:

bit 14: bad line
bit 15: If the line is not loop connected, then the selector unit is inactive. If the line is loop connected, the line is probably broken.

If both bits 14 and 15 are set the line is probably broken.

Miscellaneous : A remote line can be loop-connected via a switch on a used TFU (transfer unit) and sometimes also on the modem. The remote line is tested in this way: the line from CHRT onto the loop-connection and the return from the loop-connection onto the CHRT is used to send a SYNC-character (155) over it and a check is done on receiving the SYNC. After the SYNC detection an ACK-character is sent over the same line and also the return of the ACK-character is checked.

A looped line is out of order for any workstation transaction connected to this line.

A test-remote line should be issued from a local-workstation, a separate task.

DRSOP1	SYSTEM OPERATORS PANEL	DRSOP1

General information : The System Operator's Panel (SOP) is connected to the CPU through the channel unit for cassette recorder CHCR. The panel facilities include 10 switches and 11 lights. The switches may be read and the lamps written.

To facilitate simultaneous operations on the lights and switches, they are treated as independent devices, and are thus assigned different file codes.
Moreover, it is possible to have two independent read requests each with its own file code.

Calling sequence : Normal I/O:

```
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  1
```

I/O and Activate:

```
LDK   A1, parameter
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  – 1
DATA  start address
```

Order codes : The following orders may be used:

/02 – read switches
/05 – write lights on
/06 – write lights off
/37 – write lights on (CREDIT)
/38 – write lights off (CREDIT)
/39 – write flashing lights.

Buffer address :
Requested length : } Not significant
Effective length :

Return code : Only bit zero of the return code is used. This is set if any error is detected.

Control word : The control word contains a SOP switch number after input or a bit pattern before output (lamps corresponding to the bits are illuminated).

Order : /02 – read switches:
When a switch is depressed the switch number is stored in the control word so that SWITCH 1 gives 0, SWITCH 2 gives 2, SWITCH 3 gives 4 and so on to enable indexing. If power failure occurs the read request is completed with the control word set to -2.

DRSOP1 Continued DRSOP1

If a CREDIT application is indicated during **system generation** SWITCH 1 gives 1, SWITCH 2 gives 2, SWITCH 3 gives 3 and so on. Power failure will in this case give zero in the control word.

The rightmost switch corresponds to SWITCH 1.

Order : /05 or /37 – write lights on:
Code /05 must only be used only in an Assembler application.
Code /37 must be used only in a CREDIT application.
The control word bit pattern is transferred to the panel lights. The rightmost light corresponds to bit 15. Lights corresponding to "1" bits are turned on. Other lights are not altered.

Order : /06 or /38 – write lights off:

Code /38 must be used if a CREDIT application is indicated during **system generation**.

The control word bit pattern is transferred to the panel lights. The rightmost light corresponds to bit 15. Lights corresponding to "1" bits are turned off. Other lights are not altered.

Order : /39 – write flashing lights:

The control word bit pattern is transferred to the panel lights. The rightmost light corresponds to bit 15. Lights corresponding to "1" bits are flashed. Other lights are not altered. If order /39 is to be used it should be requested during **system generation.**

Recovery at power on : At power on the following actions are taken:

- Switches are activated
- Lights are fed with the value existent at power failure time
- If there is a read request, this is completed with -2 set in the control word. If not, a power up flag is set, causing the first read request after power on to be completed with the control word set to -2 (if adapted for CREDIT, power fail indication will be zero instead of -2).
  No indication is given in the return code.

  Note: recovery is always carried out after program loading.

DRTC01	CASSETTE	DRTC01

General information : This driver handles one or two recorders connected to the CPU on a programmed channel. The number of recorders must be specified during **system generation.**

Logically the cassette recorders are independent of each other. However, only one can be operated at a time, except at rewind and unload.

Each cassette has its own file code.

Calling sequence : Normal I/O:

```
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  1
```

I/O and Activate:

```
LDK   A1, parameter
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  -1
DATA  start address
```

Order code : The following orders may be used:

/00 – test status
/02 – read
/05 – basic write
/06 – standard write
/22 – write tape mark
/24 – erase
/26 – lock
/31 – rewind
/33 – reverse
/37 – load
/38 – unload

Buffer address :
Requested length :
Effective length :

Only significant for orders /02, /05 and /06. The last byte in each block may be used as a block sequence counter. The length must be from 2 to 256 bytes and must exclude the block sequence counter if this is used.

Return code : The following bits may be set by this driver:

DRTC01 Continued **DRTC01**

		Order in which bit is set										
Bit	Meaning	/00	/02	/05	/06	/22	/24	/31	/33	/37	/38	/26
0	Illegal request	X	X	X	X	X	X	X	X	X	X	X
1	Leader	X	X	X	X	X	X		X		(X)	X
2	BOT missing							X		X		
3	Tape mark detected		X			X			X			
4	No data/erased		X	X	X	X	X		X			
5	BOT/EOT hole		X	X	X	X	X		X		(X)	(X)
6	Write protected	X	X	X	X	X	X	X	X	X	X	X
7	B-side	X	X	X	X	X	X	X	X	X	X	X
8	Zero											
9	Rewind time-out, 55 seconds							X		X		
10	Zero											
11	Sequence error	X	X	X	X	X	X	X	X	X	(X)	X
12	Incorrect length		X									
13	CRC error		X	X	X	X			(X)			
14	Throughput error		X	X	X	X						
15	Not operable	X	X	X	X	X	X	X	X	X	X	X

(X) Not relevant.

Leader - The tape is positioned at the transparent leader.
Throughput error - System overload and should never be set
Not operable - The cassette is not locked, there is no cassette, or it executes an unload command.

Control word : Only significant for orders /37, /02, /05 and /06.

The value of the least significant bit (rightmost) during order /37 determines whether a block sequence counter will be used:

0 - sequence counter is required
1 - no sequence counter required.

The setting of the control word during order /37 will affect the operation of later orders /02, /05 and/or /06 and the recovery procedure at power on.

During a write request (orders /05 and /06) the number of rewrite attempts is returned in the control word.

Order : /00 -- test status:

The cassette is selected and the status is indicated in the return code.

Bit 15 is set "not operable" indicates that the cassette is not locked or it is rewinding.

If bit 15 is set the other bits are not significant else the following are significant:

Bit 1 "leader"
Bit 6 "write protected"
Bit 7 "B-side"

Order : /02 – read:

One block is read from tape and stored in the buffer. If there is an "incorrect length", "CRC" or "throughput error", read recovery is carried out. At read recovery the tape is backspaced one block and the block is read again. If reading is still not successful there is another backspace and read.

If it is used the block sequence counter is checked and if the block is not in sequence bit 11 "sequence error" is set in the return code. The sequence counter is not included in the effective length, but is stored in the read buffer.

Bit 3 "tape mark" is set if a tape mark was read.

Bit 4 "no data" is set if no block is found within 400 mm.

Bit 11 "sequence error" is set if the block is not in sequence when a sequence counter is used or after power failure when a sequence counter is not used.

Bit 12 "incorrect length" is set if the block was longer than the requested length. In this case read recovery is always carried out.

Bit 13 "CRC error" is set if CRC-error remains after read recovery.

For normal applications the last byte is a block sequence character.

Order : /05 – basic write:

This order has the same function as order /06 standard write.

Order : /06 – standard write:

One block is written on tape. If the status after write is "incorrect length", "CRC" or "throughput error", write recovery is carried out.

At write recovery the tape is repositioned after the last correctly written block (or backspaced once if a sequence counter is not being used). The tape is erased, depending on the number of retries, and the block is written again. If it is still not successful the tape is positioned and erased.

Before writing the first block after BOT one block is erased to be compatible with ECMA 34 standard.

If a sequence counter is used this should not be included in the requested length, but one byte must be reserved in the end of the write buffer, where the sequence counter is stored by the driver.

In the return code the following bits are significant for data:

Bit 4 "erased" should be set at successful write recovery.

Bit 5 "BOT/EOT" is set when end of tape hole is found.

Bit 11 "sequence error" is set if positioning of the tape is unsuccessful at write recovery or after power failure when a sequence counter is not used.

Bit 12 "incorrect length" only appears together with bit 0 "request error" indicating that the requested length is less than 2 or greater than 256.

Bit 13 "CRC error" is set after unsuccessful write recovery.

Order : /22 – write tape mark:

A tape mark is written on tape. Write recovery is carried out as for order /06. The bits in the return code are the same as for write. In addition bit 3 "tape mark" should be set.

Order : /24 – erase:

The tape is erased about 570 mm. Erase should be executed after last block written on tape.

Bit 4 "erased" should be set when successful.

Bit 5 "BOT/EOT hole" is set when end of tape hole is found.

If erase is required it must be requested during **system generation.**

Order : /26 – lock:

The cassette drive is locked only. The *tape is not moved.* If lock is required it must be requested during **system generation.**

Order : /31 – rewind:

The tape is rewound to BOT. The block sequence counter is set to zero.

Order : /33 – reverse:

The tape is reversed one block. The block sequence counter is decreased by one.

Bit 3 "tape mark" is set if the reversed block was a tape mark.

Bit 4 "no data/erased" is set if no block is found within 400 mm.

DRTC01 Continued DRTC01

If reverse is required it must be requested during **system generation.**

Order : /37 – load:

The tape is locked and rewound to BOT. The block sequence counter is set to zero.

Order : /38 – unload:

The tape is positioned on the leader and unlocked. This operation is carried out by hardware so the other cassette recorder may be operated simultaneously.

Recovery at power on : After power fail on the CPU, recovery is executed for each recorder.

If the cassette is locked at power fail, it will remain locked. If no blocks have been written or read BOT is searched for, else the tape is positioned. Four blocks are backspaced. If no data is found, BOT is searched for. A block is read, and if it is a tape mark another block is backspaced. The number of blocks to go forward is calculated with help of the sequence counter. The tape is read forward the calculated number of blocks. The sequence number of the read block is checked. Then the running request, if any, is repeated. No information about power fail is given in the return code.

When recovery is unsuccessful, bit 11 "sequence error" in the return code is set for the running request else at a subsequent request.

If there is no block sequence counter the cassette is locked and bit 11 "sequence error" in the return code is set for the running request else at a subsequent request.

DRTP01 — TELLER TERMINAL PRINTER — DRTP01

General information : This driver handles Teller Terminal Printer PTS 6221, 6222 or 6223 connected to the CPU via a CHLT or CHRT, and only for an Assembler application.

Alphanumeric characters are printed according to the user print buffer. Only one line is printed at each request, to get full control at recovery situations.

The user has to specify the movement of the print head and elevator in the control word. That is, no control characters are placed in print buffer.

Calling sequence : Normal I/O

```
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  1
```

I/O and Activate:

```
LDK   A1, parameter
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  – 1
DATA  start address
```

Order codes : The following orders may be used:

/00 – test status
/06 – write journal
/07 – write tally roll
/08 – write voucher/passbook

Buffer address :
Requested length : Only significant for orders /06, /07 and /08. The first word in the buffer must be reserved for control information.
Effective length : This word is only used for order /08. It must be reserved for all orders though the contents are not significant for orders /06 and /07. This word is included in the length.

Return code : The following bits may be set by this driver:

Bit	Meaning
0	Illegal request
6	Voucher in (only order /00)
10	End of journal tape
13	Code check error
14	Throughput error (special feature see order /08)
15	Not operable, power off

DRTP01 Continued DRTP01

Control word : The control word specifies the elevator and printhead movement and has the following format:

H	P	R	D	U		C	P	TAB DECADE	TAB UNIT
0							78		15

where:

HP (bit 0–1) : Head positioning after printing
HP = 0 Standard (current station)
HP = 1 Inverted (other station)
HP = 2 No action

R (bit 2) : Voucher/passbook is released after printing.

D (bit 3) : If the D-bit is set the line number given in the first word of the buffer (see buffer address) is treated as a displacement from the last position. That is, the elevator is sent up/down (depending on U) the requested number of line steps before printing.
If the D-bit is zero the elevator is sent to the top position and waits for grasp. Then the elevator is sent down the requested number of line steps, counted from elevator top position.

U (bit 4) : Elevator is sent up (U = 1) /down (U = 0) the requested number of line steps (Only significant if D = 1).

(bit 5) : Not used.

C (bit 6) : Journal copy is cut off after printing (only significant for PTS 6223).

P (bit 7) : Journal copy is perforated after printing (only significant for PTS 6223).

TAB
(bit 8–15) : A tabulation position can be set as two BCD-digits. Bits 8 – 11 contain tens and bits 12 – 15 contain units.

HP (bit 0–1) is significant for all orders /06 – /08.
R-, D-, U-bit and TAB are only significant for order /08, and C- and P-bit only for order /06.

Order : /00 – test status:

A dummy character is sent to the printer and the return code is set to the appropriate value. (bit 6, 10 and/or 15).

Common functions for orders /06, /07 and /08 : Alphanumeric characters within the range /20 – /5F, in the user buffer, are accepted and sent to the printer. Furthermore, the following special characters are processed:

/AE: Point is printed as roomless. That is, the digit prior to /AE is code converted and printed as roomless point digit (point is placed to the right of the digit).

/13: This code is sent directly to the printer. By hardware, this will cause a special symbol to be generated by the selector unit.

/14: Same function as for code /13.

/11: Tabulation character (only significant for order /08).

The special characters should be included in the requested length.

Leading spaces (/20) in the user buffer are ignored.

Illegal character codes in the user buffer are ignored and bit 13 is set in the return code.

If special characters /13 and /14 are to be used, they should be requested during **system generation.** Non standard character codes for roomless point digits can also be specified during **system generation.** Continuation of write request when the selector unit or printing is inactive may also be requested during **system generation.** (bit 15 set in return code).

Order : /06 – write journal:

Alphanumeric characters in the user buffer are printed on the journal tape. The following sequence is carried out:

- A dummy character is sent to initiate output.
- Carriage return CR1 is sent and the print head is attached.
- Characters according to the user buffer are sent.
- HP of the control word is checked.
  If HP = 0, carriage return is sent to the current station.
  If HP = 1 carriage return is sent to the other station.
  If HP = 2 the print head is not moved after print out.
- Line feed is sent to the journal tape.
- If C-bit is set the journal copy is cut off.
- If P-bit is set the journal copy is perforated.
- A dummy character is sent to end output.

If cut/perforate journal tape is to be used it must be requested during **system generation.**

DRTP01 **Continued** DRTP01

Order : /07 – write tally roll:

Alphanumeric characters in the user buffer are printed on the tally roll.
The following sequence is carried out:

- A dummy character is sent to initiate output.
- The voucher/passbook status is checked. If the voucher/passbook is in, a release voucher/passbook command is sent to the printer and the driver waits until it is removed. If the voucher/passbook is out, carriage return CR2 is sent and the print head is attached.
- Characters according to the user buffer are sent.
- HP of control word is checked.
  If HP = 0, carriage return is sent to the current station
  If HP = 1, carriage return is sent to the other station.
  If HP = 2, the print head is not moved after print out.
- Line feed is sent to the tally roll,
- A dummy character is sent to end output. Order /07 can be excluded during **system generation** if it is not required.

Order : /08 – write voucher/passbook:

The elevator is positioned at the requested line and alphanumeric characters in the user buffer are printed on the voucher/passbook. The line feed should be given in the first word of the user buffer as two ISO–7 digits the number of line feed steps, that the elevator should be moved).
The following sequence is carried out:

- A dummy character is sent to initiate output.
- Carriage return is sent to the current station.
- If D = 0 in the control word, or if the voucher/passbook is not in, the elevator is sent to the top position and waits for grasp.
- Elevator is sent to the requested line.
  That is:
  If D = 0 the line feed is treated as an absolute number and is counted from the elevator top position.
  If D = 1 the line feed is treated as a relative number (up or down depending on U) and is counted from the last elevator position.
- Print head is attached.
- If PTS 6223 a space is sent.
- Characters according to user buffer are sent.
  If a tabulation character (/11) is found the print head is returned. Spaces are sent until the tabulation position is reached, and the print head is attached again.

DRTP01 Continued **DRTP01**

During printing, the voucher/passbook status is checked. If the voucher/passbook is not in, the elevator is sent to the top position and waits for grasp. The request is then repeated. No indication is given in the return code. As an alternative chosen at **system generation** the write request is completed at voucher/passbook out and bit 14 "throughput error" is set in the return code.

- R-bit in the control word is checked. If R = 1, the elevator is sent to the top position and waits until the voucher/passbook is removed. If R = 0, no elevator movement is carried out.
- HP of control word is checked.
  If HP = 0, carriage return CR2 is sent (standard).
  If HP = 1, carriage return CR1 is sent (inverse).
  If HP = 2, the print head is not moved.
- A dummy character is sent to end output.

Recovery at power on : After power failure on the CPU or printer the elevator is sent to the top position and any running request is repeated and the requested line is printed once more. No indication is given in the return code unless requested during **system generation**.

DRTP02 **TELLER TERMINAL PRINTER** **DRTP02**

General information : This driver handles Teller Terminal Printer PTS 6221, 6222 or 6223 connected to the CPU via a CHLT or CHRT, and can be used for both Assembler and CREDIT applications.

Alphanumeric characters are printed according to the user print buffer. Only one line is printed at each request, to get full control at recovery situations.

Journal, tally roll and voucher/passbook parts of the printer are regarded as three independent logical devices. Three different file codes are used: one for each type of printer.

This driver can be used instead of driver DRTP01 in every system, provided the different user interfaces are considered. However, this driver *must* be used in systems where the interpreter language CREDIT is used.

Calling sequence : Normal I/O:

```
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  1
```

I/O and Activate:

```
LDK   A1, parameter
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  -1
DATA  start address
```

Order codes : The following order codes may be used:

- /00 – test status
- /06 – write
- /0B – position lift and write
- /22 – cut journal
- /26 – perforate journal
- /37 – grasp
- /38 – release voucher/passbook

Buffer address
Requested length
Effective length : Only significant for orders /06 and /0B. The first word of the buffer must be reserved for a control code and is included in the length.

Return code : The following bits may be set by this driver:

DRTP02 Continued DRTP02

Bit	Meaning
0	Illegal request
8	Recovery executed on request
10	End of journal tape/voucher out, depending on device
13	Code check error
15	Not operable, power off

Control word : Only significant for order /0B. The control word specifies the lift position. The lift position is given as a binary value in the right byte. Only absolute positioning is used, i.e. the control word gives the number of lift steps from home (lift in top position).

Order : /00 – test status:

A dummy character is sent to the printer and the return code is set to the appropriate value. Note that Bit 10 in the return code means "voucher out" or "end of journal paper" depending on the device used.

Common functions for orders /06, /0B, /22, /26. /37 and /38 : Continuation of request when the selector unit or printer is inactive may be requested during **system generation** (bit 15 set in return code).

Order : /06 – write:

Alphanumeric characters within the range /20–/5F, in the user buffer, are accepted and sent to the printer. Furthermore, the following special characters are processed:

/AE: Point is printed as roomless i.e. the digit prior to /AE is code converted and printed as a roomless point digit.

/13: This code is sent directly to the printer. By hardware, this will cause a special symbol to be generated by the selector unit.

/14: Same function as for code /13.

/09: The print head is moved to the rightmost print position of the voucher print station. This character should be present in the last buffer position.

/0D: The print head is moved to the rightmost journal print position. This character should be present in the last buffer position.

Leading spaces (/20) in the user buffer are ignored.

DRTP02 Continued DRTP02

Illegal character codes in the user buffer are ignored and bit 13 is set in the return code.
If special characters /13 and /14 are to be used, they should be requested during **system generation.** Non standard character codes for roomless point digits can also be specified during **system generation.**

A short summary of the sequence followed when a write request is executed is given below:

- A dummy character is sent to obtain the status of the printer.
  If voucher/passbook printing is being performed, the voucher/passbook must be in.
  If tally roll printing is being performed, the voucher/passbook must be out.
  If these conditions are not fulfilled, grasp is carried out in the first case and release in last case.
- Carriage return is sent and the print head is attached. (if req. length = 0 – 2 no attach).
- If PTS 6223 is being used and voucher/passbook printing is being carried out a leading space is sent.
- Characters according to the user buffer are sent.
- Voucher/passbook printing: If a tabulation character (/11) is found, the print head is returned. Spaces are sent until the tabulation position is reached and the print head is attached again. During printing, the voucher/passbook status is checked. If the voucher/passbook is not in, the elevator is sent to the top position and waits for grasp. Then the request is repeated. As an alternative, chosen at **system generation,** the write request is completed at voucher/passbook out and bit 10 is set in the return code.

  The control code in the first word of the buffer specifies the paper feed. This code has the following significance:

  /28: print with no advance.
  /30: advance two lift steps before printing.
  /31–/39: advance one to nine lift steps before printing.
  Other codes: advance one lift step before printing.
- Tally roll printing: During printing the voucher/passbook status is checked. If the voucher/passbook is in, a release command is sent to the printer and the request is repeated.
- After printing the print head is detached and carriage return is executed if one of the two characters /09 or /0D is present in the buffer.
- Line feed is executed (not voucher/passbook printing) and a dummy is sent to end output.

DRTP02 — Continued — DRTP02

Order : /0B – position lift and write:

Elevator position should be given in the right byte of the control word as a binary value, and indicates the number of line feed steps that the elevator should be moved from home position.

Before positioning a check is made that the voucher/passbook is in. If not the command grasp is initially sent to printer. During positioning the voucher/passbook status is checked and, if the voucher/passbook is out, grasp is executed and the request is repeated. As an alternative, chosen at **system generation** the request is completed at voucher/passbook out and bit 10 in the return code is set.

After lift positioning write voucher/passbook is carried out (see order /06 write)

Order : /22 – cut journal:

Journal copy is cut off (PTS 6223) and one line feed is made. If the cut journal facilitiy is required it should be requested during **system generation.**

Order : /26 – perforated journal:

Journal copy is perforated (PTS 6223) and one line feed is made. If the perforate journal facility is required it should be requested during **system generation.**

Order : /37 – grasp:

The following sequence is carried out:

- a dummy character is sent to initiate output.
- the elevator is sent to the top position.
- a grasp command is sent to the printer.
- a dummy character is sent to the printer. When a data request is returned from the printer the grasp has been performed, i.e. the voucher/passbook is in and the next character can be sent to printer.
- lift to top position is executed.
- the request is completed.

Order : /38 – release voucher/passbook:

The lift is moved to the top position and a release command is executed. After that a dummy character is sent. When a data request is returned from the printer, the command is complete i.e. the voucher/passbook is out and the request is completed.

DRTP02 **Continued** **DRTP02**

Recovery at power on : After power failure on the CPU or printer, the elevator is sent to the top position and any running request is repeated. Normally no indication is given in the return code. There is a possibility, chosen at **system generation**, to get bit 8 set in the return code if recovery has been carried out on a write request (orders /06 or /0B).

DRTW01 **CONSOLE TYPEWRITER** **DRTW01**

General information : This driver handles input from and output to typewriter PTS 6862, connected to the CPU via the teletype-interface or V24 interface.

Input and output can not take place at the same time, since the connection only allows half duplex transmission.

Calling sequence : Normal I/O:

```
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  1
```

I/O and Activate:

```
LDK   A1, parameter
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  – 1
DATA  start address
```

Order codes : The following orders may be used:

/01 – basic read
/02 – standard read
/03 – numeric read
/05 – basic write
/06 – standard write

Buffer address : Significant for all orders: For order /06 the first word in the buffer should be reserved for control information. This word is also included in the length. For orders /02 and /03 the length includes the end of record key.

Return code : The following bits may be set by this driver:

		Order in which bit is set				
Bit	Meaning	/01	/02	/03	/05	/06
0	Illegal request	X	X	X	X	X
9	Time out	X	X	X		
12	Incorrect length		X	X		
13	Code check error		X	X		X
14	Throughput error				X	

Control word : Only significant for orders /02 and /03. For these orders it may contain a keytable address. The keytable contains a list of end of record keys. If the keytable address is zero then no keytable will be used. The format of the keytable is as follows:

DRTW01 **Continued** DRTW01

No. of EOR keys	key 1
key 2	key 3
key 4	key 5

etc

Order : /01 – basic read:

The requested number of characters are read and stored in the user buffer without any check.

Common functions for orders /02 and /03 : The driver checks every received character in the following sequence:

- If the received character is found in the key table, the key is stored in the user buffer and the request is immediately completed. The end of record key is also code converted and stored in the control word so that for Assembler applications KEY 1 gives 0, KEY 2 gives 2, KEY 3 gives 4 and so on to enable indexing. For CREDIT applications KEY 1 gives 1, KEY 2 gives 2, KEY 3 gives 3 and so on. If the keytable address is zero in the control word a standard key TWEOR is used as end of record key.
- Special characters, clear and backspace, are checked and the corresponding functions are carried out.
- Alphanumeric/numeric characters are stored in the user buffer. If overflow occurs in the buffer the request is completed with bit 12 set in the return code. The requested length should include the end of record key.
- If the received character cannot be identified in the tests above, it is treated as undefined and the request is completed with bit 13 set in the return code.
  The undefined key is stored in the user buffer and the control word remains unchanged.

Order : /02 – standard read:

Alphanumeric characters within the range /20–/5F are accepted and stored in the user buffer.

Order : /03 – numeric read:

Only digits within the range /30–/39 are accepted and stored in the user buffer.

Order : /05 – basic write:

The requested number of characters are sent to the printer without any check.

DRTW01 Continued DRTW01

Order : /06 – standard write:

The first word in the user buffer is reserved for control information. It must contain one of the following codes in the right hand byte:

/2B : print the line without advancing the paper. The print head is not moved before printing the text.

/30 : advance the paper two lines before printing and perform carriage return.

/31 : skip to top of the page (form feed) and perform carriage return before printing.

All other codes cause carriage return and line feed before printing.

All alphanumeric characters within the range /20–/5F in the user buffer are accepted and sent to the printer.

Codes /60–/7F are reduced by /20 giving /40–/5F. Furthermore, the following special characters are processed:

/AE : Printed as point (/2E)

/11 : Tabulation character. This character should be followed by two ISO–7 digits giving the tabulation position. Tabulation character and position should be included in the requested length.

/07 : Bell is sent to the printer.

Time out function : For input it is possible to include a time-out function. If a key has not been pressed within a certain time, the request will be ended with bit 9 set in the return code. The timer is restarted for each depressed key.
If time out is required it must be requested during **system generation.**

Special characters : Some keys have a special meaning to the driver. These keys are:

TWEOR : Standard end of record key.
This is used by the system when keytable address is zero.

TWCLR : Clear key. The user buffer is cleared. This does not cause completion of the request.

TWBSP : Backspace key. The last reported character is cleared in the user buffer. This does not cause completion of the request.

DRTW01 **Continued** DRTW01

Recovery at power on : At power up the following actions are taken:

- If the order was /01, /02 or /03 the read request is completed with -2 set in the control word.
  If no read request is running a power up flag is set, causing the first read request after power on to be completed with the control word set to -2.
  Completion of read request, if required, must be requested during **system generation.**
- If the order was /05, bit 14 is set in the return code and the request is completed.
- If the order was /06, the print head is sent to the position it had before the write request and the line is printed once more. No indication is given in the return code.

TIODM	DATA MANAGEMENT	TIODM

General information : This driver provides data management facilities for the files held on upto two PTS 6875 disk drives. The disk drives are controlled via the disk driver DRDU01 (which is called by the data management task). The general information in the driver description for DRDU01 also applies to the data management driver.
For a full explanation of the facilities offered by the data management routines see the Data Management Manual (M07).

Calling sequence : Normal I/O:

```
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  1
```

I/O and Activate:

```
LDK   A1, parameter
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  -1
DATA  start address
```

Order codes : The following orders may be used:

/02 – sequential read
/03 – read VTOC record
/06 – sequential write
/09 – release exclusive access
/0A – random read
/0B – random write
/0C – random delete
/1A – indexed random read
/1B – indexed rewrite
/1C – indexed delete
/1D – indexed insert
/1E – indexed read next
/22 – close
/23 – get current data record number
/24 – get current index record number

Buffer address : Only significant for orders /02, /03, /06, /0A, /0B, /1A, /1B, /1D, /1E.

Requested length
Effective length : Only significant for orders /02, /03, /0A, /1A, /1D, /1E. If the requested length is smaller than the record length only the most significant part of the record will be moved to the user buffer.

Return code : The following bits may be set by this driver:

TIODM

Continued

TIODM

		Order code in which bit is set													
Bit	Meaning	/02	/03	/06	/09	/0A	/0B	/0C	/1A	/1B	/1C	/1D	/1E	/22	/23	/24
0	Request error	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
1	Key not found								X		X					
2	Record protected	X		X		X	X	X	X	X	X	X	X			
3	End of file	X				X	X	X	X		X	X	X			
4	No data	X				X			X		X		X			
5																
6	Next key same								X		X	X	X			
7	Retries performed	X	X	X		X	X	X	X	X	X	X	X			
8	New volume loaded	X	X	X		X	X	X	X	X	X	X	X			
9																
10	End of medium			X		X	X	X	X	X	X	X	X			
11																
12	Incorrect length	X	X			X			X				X			
13																
14	Disk I/O error	X	X	X		X	X	X	X	X	X	X	X	X		
15	Disk not operable	X	X	X		X	X	X	X	X	X	X	X	X		

Bit 0 Request error
Set for request errors such as illegal order, unknown file code etc.

Bit 1 Key not found
Set if the symbolic key required for indexed random instructions was not found in the index file.

Bit 2 Record protected
Set if the accessed record is under "exclusive access" at the time of the read request, or the record is not under "exclusive access" and the record status indicates "USED" at the time of a write request.

Bit 3 End of file
Set if the accessed record has a logical record number greater than the "last record No." updated by sequential write.

Bit 4 No data
Set if the record status character indicates "free" at a read-request.

Bit 5 Not used.

Bit 6 Next key same
Set if the symbolic key in the next used index record is the same as in the current index record.

Bit 7 Retries performed
The driver has retried an I/O action that was in error.

Bit 8 New volume loaded
Set at the first request after a new volume has been loaded.

Bit 9 Not used.

Bit 10 End of medium
Set if the requested record is outside the physical space reserved for the file at creation time.

Bit 11 Not used.

Bit 12 Incorrect length
Set if the requested length is less than the record length at read request.

Bit 13 Not used.

Bit 14 Disk I/O error
Set for hardware errors e.g. seek error, CRC-error, throughput error.

Bit 15 Disk not operable.

Control word : Significant for orders /02, /03, /06, /0A, /0B, /1A, /1B, /1D, /1E. At the time the request is made the control word should contain a logical record number in CW1 and CW2, or the address of a symbolic key in CW1 and key length in CW2. The logical record number starts with 1 and is a 23 bit (right justified) integer. The first characters should contain hexadecimal zero and the sign bit of the third character is not used.

Order : /02 – sequential read:

A record will be read into the user buffer according to the rules for sequential read, described in the Data Management Manual (M07).

The requested length must be filled in by the user and will define the length of the user buffer.

The effective length will, at return, contain the record length (the part of the block moved to the user record area). If the requested length is smaller than the record length defined at file creation time, only the most significant part of the record will be moved.

The control word will be set by data management to the current logical record number.
If a correct return is made the record read is set to exclusive access for the calling task.

Order : /03 – read VTOC:

This order is used to read the volume table of contents (VTOC) for the assigned file into the user buffer. If the requested length is less than the length of the VTOC, only the most significant part of the VTOC is moved to the user buffer.

Order : /06 – sequential write:

A record will be written from the user buffer according to the rules for sequential write, described in the Data Management Manual (M07).

The control word will be set by data management to the logical record number written by this instruction. To check the record status the record space is read before it may be overwritten. If the record is "free" it will be changed to "used" and the record will be put into the sector buffer and the record is immediately written on disk. If the status is "used" the sequential write is not allowed.

Order : /09 – release exclusive access:

The current record in the file specified by the file code in the ECB will no longer be under "exclusive access". This order does not result in any physical I/O.

Order : /0A – random read:

A record will be read into the user buffer according to the rules for random read described in the Data Management Manual (M07).

The requested length and effective length are used in the same way as for order /02 (sequential read).

The control word must be set by the user to the logical record number of the record to be read.

Order : /0B – random write:

A record will be written from the user buffer according to the rules for random write described in the Data Management Manual (M07).

The control word must be set by the user to the logical record number of the record to be written.

To check the record status, the record space is read before it may be overwritten. If the record is "free" it will be changed to "used" and the record is written. The record is written if the record is "used" and under exclusive access. If a "used" record is not under exclusive access the return condition "record protected" is set.

After this order the record written will no longer be under exclusive access.

If the exclusive access check is required it must be requested during **system generation.**

Order : /0C – random delete:

The status of the specified record will be set to "free". This order is only allowed for a record which is under exclusive access. After the order the free record space will no longer be under exclusive access. There is no check that the record was already "free".

If the exclusive access check is required it must be requested during **system generation.**

The control word must be set by the user to the logical record number of the record to be deleted.

Order : /1A – indexed random read

A record will be read into the buffer according to the rules for indexed random read described in the Data Management Manual (M07).

The requested length and effective length are used in the same way as for order /02 (sequential read), that is, they apply to the data record. The control word must contain the symbolic key and key length of the data record to be read.

The current record number of both the index file and the data file will be updated to the index record and data record accessed by this order.

Order : /1B – indexed rewrite

A record will be written from the user buffer and replace the record on the disk that has just been read according to the rules for indexed rewrite described in the Data Management Manual (M07).

Before the record is written, the record that it is replacing must have been read with exclusive access set. Exclusive access is released after this order.

The current record number will be set to the record just rewritten.

If the exclusive access check is required, it must be requested at **system generation** time.

Order : /1C – indexed delete

The status of the referenced data record is set to "free". When this has been done successfully, the index record that refers to the deleted data record is also set to "free".

This order is only allowed for a record that is under exclusive access. There is no check that the record was already "free".

If the exclusive access check is required, it must be requested during **system generation.**

The control word must contain the symbolic key and key length of the data record to be deleted.

The current record number is unaffected by this order.

Order : /1D – indexed insert

A record will be written from the user buffer to the data file after the last record in the file. The last record number will be updated. A new index record, with the logical record number of the new data record, is placed in the correct position in the index file. If an index record exists with the same symbolic key, the new index record will be placed in front of the old one.

The current record numbers for both the data file and index file are set to the new records.

Exclusive access is not set with this order.

Order : /1E – indexed read next

A record will be read into the user buffer from the data file as indicated by the next index record after the current index record number.

The current record numbers in both the data file and the index file will be updated.

The control word is empty at the time the request is made. Exclusive access can be set for this record in the same way as for indexed random read.

Order : /22 – close file:

This order indicates to data management that the file specified by the file code on the ECB is no longer required by any task. It may not be used again by any task until a file code is re-assigned to the file.

Order : /23 – get current record number of the data file:

The current record number of the data file is put into the control word. No I/O operation occurs and only the file code is required as a parameter.

Order : /24 – get current record number of the index file:

The current record number of the index file is put into the control word. No I/O operation occurs and only the file code is required as a parameter.

System generation parameters : In addition to the parameter mentioned above (exclusive access handling) the following parameters must be supplied during **system generation**.

- Number of disk files. This is the maximum number of data management files which can be assigned at the same time.
- Number of file codes. This is the maximum number of file codes which can be used for data management files.
- Number of buffers. This is the number of sector buffers to be used by data management.

Recovery at power on : If there is a running disk I/O-request when power fail occurs, this request is immediately completed at power up and bit 14 "disk I/O error" is set in the return code.

After power fail the disk drive will remain not operable until it is started manually. When the disk drive becomes ready again the volume label is read by the disk driver. If a new volume name is entered the next data management request will be aborted and bit 8 "new volume loaded" is set in the return code.

3.6 Special calls

This section contains detailed reference information about two functions which are called by the LKM, DATA 1, the ATTACH/DETACH function and the INTERTASK COMMUNICATION function.

During system generation, certain parameters concerning the handling of the functions can be specified. The parameters are described for each function in this section. The words "system generation" are printed in bold type throughout this section in order to highlight these parameters. Most parameters are specified during the "conditional assembly" specification. Where this is not the case the relevant part of system generation appears in brackets.

ATTACH/DETACH

ATTACH/DETACH

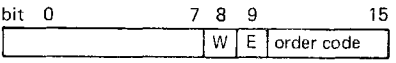
General information : The ATTACH request is used when a task wants to have exclusive access to a device. The other tasks are locked out from I/O on this device.

When a device has been attached to the task all I/O requests from other tasks are put into the device queue and will not be executed until the device is detached.

The DETACH request is used for releasing the attached device. The detach request must be performed by the task that made the attach request.

Calling sequence : LDK A7, code
LDKL A8, ecb-address
LKM
DATA 1

"Code" has the following layout:



bits 0–7 – are not significant.
bit 8 – wait bit, must be set (1).
bit 9 – echo bit, must be reset (0).
bits 10–15 – are the order bits specifying the function.

Order codes : The following order codes are available:

/3B : attach
/3C : detach

ecb-address : The ecb-address is the pointer to the event control block which has the following layout:

byte address

0	status	file code*
2	not used	
4	not used	
6	not used	
8	return code	
10	control word*	

"status" – This field is used exclusively by the monitor. Bit 0 (most significant) is set to 1 on completion of the I/O operation.

"file code" – specifies the device to be attached to or detached from the requesting task. File codes are associated with devices during **system generation** (see list in section 3.4 of recommended file codes).
File code has to be specified by the user.

"return code" – In the word return code two bits indicate how the event is completed.

bit 0 – request error
bit 9 – device not available (only order /3B).

"control word"– The control word is only significant for order /3B. The control word contains the specified time-out value in multiples of 100 ms. That specified time the attach-request stays on and after that time the request is aborted and that is indicated in the return code (bit 9).

A time-out setting of zero implies a test on device for availability. The control is immediately given back to the task which issued the request, with in the return code the information whether the device is attached or not.

Control word has to be specified by the user with order /3B.

Order : /3B – attach

The specified device is attached to the requesting task and stays under control of that task until a detach request is issued by the controlling task. If the device is not available after some time (time-out specified in control word) the request is aborted with bit 9 set in the return code.
I/O requests for the device from other tasks are queued in the device queue. The requests are executed after the device is detached. The attach and I/O requests are queued in the device queue which is organized first in first out within priority.

Order : /3C – detach

The specified device is detached from the task issuing this detach request.
After the detach the device can be used by any other task as specified by the device queue or program.

INTERTASK COMMUNICATION

INTERTASK COMMUNICATION

General information : This intertask communication processor which handles the communication between tasks can be included at **system generation** time. With the intertask communication processor it is possible to transfer data from one task to another task. However, the sending task has to issue a write request and the receiving task has to issue a read request.

A write request is always addressed to a specific task in contrast to a read request which is used for receiving data from any task.

It is strongly recommended to assign the same file codes for intertask communication to all tasks (/D0 – input, /D1 – output) at **system generation**.

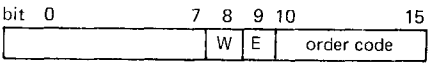
Calling sequence : Intertask communication

```
LDK   A7, code
LDKL  A9, ecb-address
LKM
DATA  1
```

Intertask communication with activation

```
LDK   A1, parameter
LDK   A7, code
LDKL  A8, ecb-address
LKM
DATA  –1
DATA  start-address
```

"Code" has the following layout:



bits 0–7 – not significant
bit 8 – wait bit
bit 9 – echo bit must be reset (0)
Bits 10–15 – are the order bits specifying the function.

Order codes : The following order codes are available:

/02 : read
/06 : write
/39 : set time-out

ecb-address : The ecb-address is the pointer to the event control block which has the following layout:

INTERTASK COMMUNICATION **Continued** **INTERTASK COMMUNICATION**

byte address

0	status	file code*
2	buffer address*	
4	requested length*	
6	effective length	
8	return code	
10	control word*	

"status" – This field is used exclusively by the monitor. Bit 0 (most significant) is set to 1 on completion of the I/O operation.

"file code" – Specifies the file code for input or output in accordance with the order code (input-read, output-write).

"buffer address" – Points to the buffer address used with the intertask communication. (Used with orders /02 and /03.)

"requested length" – Length of the data to be transferred inclusive of the control word in buffer. (Only used with orders /02 and /03.)

"effective length" – In this word the number of transferred characters are administrated. (Not significant with order /39.)

"return code" – The return code gives information about the status of the I/O request after completion.

bit 0 – request error
bit 9 – time-out
bit 12 – Incorrect length (Requested length of read request less than requested length of attached write request.)

"control word" – Only used with order /39 and will contain a time-out value (marked in multiples of 100 msec) or if no time-out is wanted, the value should be –1.

* has to be supplied by the user.

INTERTASK COMMUNICATION **Continued** **INTERTASK COMMUNICATION**

Order : /02 – read

With the order read the task is prepared to receive data from another task, which has issued a write for this task.

The first word of the buffer is reserved and should be included in the requested length. After completion of the read the first word in the buffer contains the task identification (TID) of the task from which the data were transferred.

The words buffer address, requested length, effective length and the return code have the normal meaning.

Order : /03 – write

The write order prepares a data transfer from this task to another task which is addressed by the first word in the buffer.

The write is accepted by the addressed task if that task has issued a read. If no read is issued, the write is put into an input queue (first in–first out) attached to the addressed task.

The words buffer address, requested length, effective length and the return code have the normal meaning.

Order : /39 – set time-out

The set time-out request is issued by the monitor to supervise the read and write request within the intertask communication. If the issued read or write request is not completed within the specified time, bit 9 of the return code word will be set. For read and write different time out values can be set. The time-out value is specified in multiples of 100 msec. in the control word of the ECB. A time-out request for read uses the input file code (recommended /D0).
A time-out request for write uses the output file code (recommended /D1).

A time-out value is valid until changed by a new set-time-out request. If no time supervision is wanted (for a non-expired time-out value) the control word of the set time request should be set to –1.

A set-time-out request should be issued without wait on event before the read or write request is issued which needs the time supervision.

Buffer address, requested length and effective length are not significant.

4. TOSS UTILITIES

4.1 Introduction

The TOSS utilities are a set of routines which carry out various housekeeping functions on data files and program files. The routines are as follows:

Name	*Function*
CDD	Copy disk to disk
CFF	Copy file to file
CFT	Copy file to tape
CPP	Copy program
CRF	Create file
CRV	Create volume
CTF	Copy tape to file
DLF	Delete file
LFD	Load flexible disk
PDS	Print disk sector
PRF	Print file
PVC	Print volume table of contents
SCT	Scan tape
UDS	Update disk sector
UFD	Unload flexible disk.

All of the above routines except CPP operate on data files. CPP operates on disk resident program files.

TOSS utilities can be executed as independent programs or they can be link edited into a normal application program. The function of the present chapter is to describe the application program interface to TOSS utilities. The use of TOSS utilities as independent programs will not be described.

4.2 Copy Tape to File and Copy File to Tape

The utilities CTF and CFT are designed to handle ½" magnetic tape and cassettes labelled according to the ECMA–41 basic or compact labelling system. Multi-volume files are supported, and the different tracks of a cassette are regarded as different volumes.

4.3. Print File and Scan Tape

The following output formats are available with these utilities (PRF and SCT):

- Compact format

The record is printed with no conversion except for unprintable characters, which are replaced with '.'. If unprintable characters have occurred '%%%' is output immediately after the contents of the record.

- Spaced format

Data is printed with spaces separating the characters. Unprintable characters are output as '# followed by two hexadecimal digits representing the internal value. Space is printed as underscore, and underscore as $UN.

- Hexadecimal format

Data is printed in hexadecimal form with 16 bytes/line. The character equivalent is output in the right margin, with periods representing unprintable characters.

4.4. General Interface Rules

The calling sequence for all utilities is as follows:

```
LDKL    A12, pb-address
CF      A14, name
```

Where "pb-address" points to a parameter block which contains all the information needed by the system to run the utility, and "name" is the name of the utility (e.g. CDD, CFF). No validity check is performed by the utilities on the contents of the parameter block.

The utilities write a return code into register A1 before returning to the application program. If A1 is zero then no error has occurred. The significance of non-zero return codes is described in the utility reference section (4.5).

For some utilities register A2 contains an additional return code concerning I/O errors. This is the return code generated by the relevant I/O driver. The significance of these return codes is discussed in the driver reference section (3.4).

4.5 Utility Reference

This section contains detailed reference information for each TOSS utility. The information for each utility is given in alphabetical utility name order. That is, first utility BIX, then CCF, and so on.

BIX BUILD INDEX FILE **BIX**

Description : This utility is used to create a sequential file that contains the index records for a given data file, by reading the data file in conjunction with parameters supplied by the programmer.

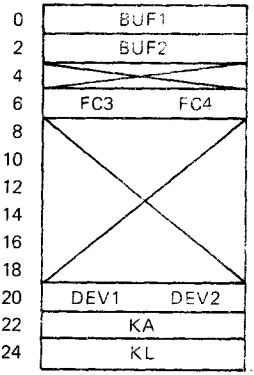
The input file is read until End-of-Medium is reached, and records containing no data are skipped. Both files must reside on disk, and the file to be created by the utility must have been previously defined with the CRF utility.

The record length of the output file must be set to the length of the key field + 6 characters.

Note: the output file from BIX is not ready to be used as an index file, but must first be sorted using the SRT utility, and then be input to the RIX (Reorganize Index) utility, in order to create the actual index files.

Calling Sequence : LDKL A12,pb-address
CF A14,BINDEX

Parameter block : word



BUF1 : Data file sector buffer address
BUF2 : Index (output) file sector buffer address
FC3 : File code of input file
FC4 : File code of output file
DEV1 : Input device indicator; must be set to D
DEV2 : Output device indicator; must be set to D
KA : Relative character address of key in data record; position 0 indicates 1st byte in record
KL : Key length in characters (1–255)

BIX **Continued** **BIX**

Return Code : A1 = 0 No error
A1 = 1 Input I/O error (A2 = return code)
A1 = 2 Output I/O error (A2 = return code)
A1 = 3 Blocksize error
A1 = 4 Not used
A1 = 5 Key definition is outside record boundaries
A1 = 6 File organization is not S
A1 = 7 Record length error
A1 = 8 Output file is not large enough
A1 = 9 Output file is not empty

CCF | **COPY CARDS TO DISK FILE** | **CCF**

Description : This utility is used to copy cards to a disk file previously defined using the CRF utility. The output file must be empty, otherwise an error message is output.

Reading and writing will be carried out until a card is read containing :EOF in columns 1–4. A message is then output to the operator's console stating how many cards have been copied.

If the output record length is greater than 80 bytes, the remainder of the records will be padded with blanks. If the record length of the output file is less than 80 bytes, the cards must contain blanks in columns having a number higher than the output record length, otherwise copying is terminated and an error message displayed on the operator's console.

Calling sequence :
```
LDKL  A12,pb-address
CF    A14,COPYCF
```

Parameter block : word

0	BUF1	
2	BUF2	
4		FC2
6	FC3	FC4

BUF1 : Address of record buffer 1
BUF2 : Address of record buffer 2
FC2 : File code of the operator's output device
FC3 : File code of the card reader
FC4 : File code of the disk output

Return code :
A1 = 0 No error
A1 = 1 Input I/O error (A2 = return code)
A1 = 2 Output I/O error (A2 = return code)
A1 = 3 Card reader not operable
A1 = 4 Output file not large enough
A1 = 5 Output file not empty
A1 = 6 Not used
A1 = 7 Faulty record length
A1 = 8 Disk not operable

CDD

COPY DISK TO DISK

CDD

Description : This utility is used to copy a whole disk to an empty disk already formatted.

The program makes a check that the output disk is empty. A check is also made that the VTOC of the output disk is large enough and that the free space on the output disk is sufficient.

The volume name is not copied to the output disk.

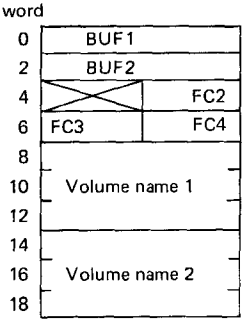
If a file on the input disk is divided into more than one extent, it will, if possible, be put together in one extent on the output disk.

Empty areas and badspots are not copied.

If some sectors cannot be copied due to a disk error the destroyed record numbers will be communicated to the operator.

Calling sequence :
```
LDKL  A12, pb-address
CF    A14,COPVOL
```

Parameter block :



BUF1 : The address of sector buffer 1
BUF2 : The address of sector buffer 2

FC2 : File code of the print device
FC3 : File code of the input disk
FC4 : File code of the output disk

Volume name 1 : Volume name of the input disk
Volume name 2 : Volume name of the output disk

CDD **Continued** CDD

Return code : A1 = 0 No error
A1 = 1 Input I/O error (A2 = driver return code)
A1 = 2 Output I/O error (A2 = driver return code)
A1 = 3 Volume name and file code do not correspond
A1 = 4 Output disk not empty
A1 = 5 VTOC overflow
A1 = 6 Disk overflow
A1 = 7 Faulty disk format
A1 = 8 Flexible disk write protected

CFF	COPY FILE TO FILE	CFF

Description : This utility is used to copy a disk file to an empty file area previously reserved using the CRF utility.

The program can be run with or without packing. If packing is used, records containing no data on the input file will not be copied. This means that randomly written files must be copied without packing.

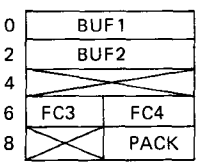
Records placed after the last record number on the input file will also be copied.

The program checks that the last record number is zero on the output file and that the record lengths correspond on the input and output files.

The program also checks that the input and output file organizations are the same.

Calling sequence :
```
LDKL  A12, pb-address
CF    A14,COPFIL
```

Parameter block : word



BUF1 : The address of user buffer 1
BUF2 : The address of user buffer 2
FC3 : File code of the input file
FC4 : File code of the output file
PACK : Pack indicator. Zero means no pack, non-zero means with packing.

Return code :
A1 = 0 No error
A1 = 1 Disk not operable
A1 = 2 Disk I/O error (A2 = driver return code)
A1 = 3 Faulty record length
A1 = 4 Output file not large enough
A1 = 5 Output file not empty
A1 = 6 File organization error.

CFT

COPY FILE TO TAPE

CFT

Description : This utility copies a disk file onto tape cassette or ½" magnetic tape.
All records up to end of file are copied, except for deleted (no data) records. Cassettes and magnetic tapes are written with or without labels. If labels are written, the following fields in the label record are copied from the VTOC:

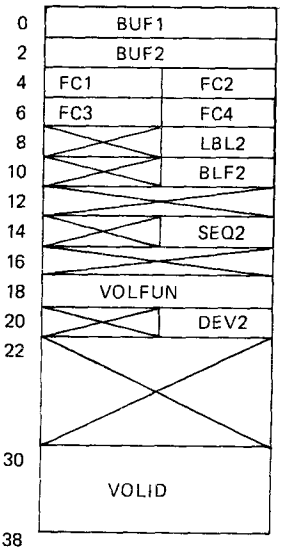
- file name
- creation date
- retention period

If desired, several records can be blocked to form one block on tape.

The cassette or magnetic tape is not rewound at exit: it is therefore possible to copy several files from disk to the same cassette or tape. To rewind the cassette or tape the utility ULD (Unload Device) must be used.

Calling sequence : LDKL A12, pb-address
CF A14,COPYFT

Parameter block : word



CFT **Continued** CFT

BUF1 : The address of a buffer large enough to contain one tape block.
BUF2 : The address of a buffer at least 128 words long.
FC1 : Operators input device.
FC2 : Operators output device.
FC3 : File code of the input file.
FC4 : File code of the output file
LBL2 : Labelled tape flag. This must be 1 if the tape is labelled otherwise 0.
BLF2 : Block factor on output tape (binary value). This must be 1 if no blocking is required.
SEQ2 : Sequence number flag. If = 1, block sequence numbers will be written on to the output tape.
VOLFUN : Subroutine entry address.
When EOT on the output tape is detected, this routine is called (with a CFI A14, 18, A12 instruction), to request mounting of a new volume. Upon return FC4, and eventually VOLID, are assumed to be filled in with appropriate values.
DEV2 : Output device type - one ISO–7 character: 'T' if tape cassette, 'M' if ½" magnetic tape.
VOLID : The new tape volume identifier. Only significant if LBL2 = 1.

Return code : A1 = 0 No error
A1 = 1 Output I/O error (A2 = driver return code)
A1 = 2 Input I/O error (A2 = driver return code)
A1 = 3 Erroneous blocksize - the block factor specified is too big. The blocksize becomes larger than allowed for the output device.

CIT | **COPY IBM FILE TO TOSS** | **CIT**

Description : This utility is used to copy a file from an IBM-labelled flexible disk to an empty file on a TOSS-labelled disk, previously created with the CRF utility.

The program can be run with or without packing. If packing is used, records indicating deleted data and records placed after the end of data on the input file will not be copied.

Data on the IBM-labelled disk must be EBCDIC-coded, and will be converted to ASCII code by the program.

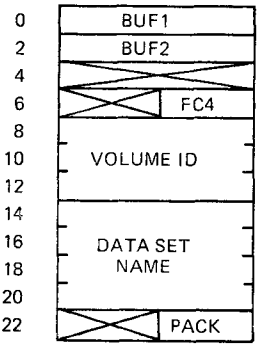
The program checks that the LRN on the output file is zero, and that the input and output files have the same record length.

File codes /F8–/FB are used when searching for the volume identity. Sectors 08–26 are used when searching for the data set name.

Multivolume data sets are not allowed.

Calling sequence : LDKL A12,pb-address
CF A14,COPYIT

Parameter block : word



BUF1 : Address of record buffer 1.
BUF2 : Address of record buffer 2.
FC4 : File code of output file.
VOLUME ID : Volume name of input disk.
DATA SET NAME : Name of input data set.
PACK : Pack indicator. Zero indicates no packing.

CIT **Continued** **CIT**

Return code : A1 = 0 No error
A1 = 1 Disk not operable
A1 = 2 Disk I/O error (A2 = return code)
A1 = 3 Faulty record length
A1 = 4 Output file not large enough
A1 = 5 Output file not empty
A1 = 6 Data set name not found
A1 = 7 Input volume name unknown

CPP	COPY PROGRAM	CPP

Description : This utility is used to move a program from a DOS6810 formatted volume and place it on a TOSS formatted volume.

A special UF-file on the DOS volume must contain the names and userids of the load modules in order to build up a library file. The sequence in which the module names have been generated in this file determines the segment numbers to be used when running the application. The "root" is segment 0 and the other ones are numbered from 1.

While creating the library file the load segment block table (LSBT) is built up in memory and, after the last segment of the file is written on disk, this table is stored in the first sector of the file.

As a result of the CPP utility there is a print out, on the operators typewriter, of all modules and the corresponding segment numbers, with segment lengths in bytes (hexadecimal).

Calling sequence :

```
LDKL   A12,pb-address
CF     A14,COPPRO
```

CPP **Continued** CPP

Parameter block : word

0	BUF1	
2	BUF2	
4	FC1	FC2
6	FC3	FC4
8		
10	INPUT	
12	VOLUMENAME 1	
14		
16		
18	OUTPUT	
20	VOLUMENAME 2	
22		
24	INPUT	
26	FILENAME 1	
28		
30	INPUT	
32	USERID	
34		
36		
38	OUTPUT	
40	FILENAME 2	
42		

BUF1 : The address of the big sector buffer - 1640 words.
BUF2 : The address of the small sector buffer - 205 words.
FC1 : Filecode of the operators keyboard - /20.
FC2 : Filecode of the operators printer - /30.
FC3 : Filecode of the input disk (DOS).
FC3 : Filecode of the output disk (TOSS).
VOLUME NAME 1 : Volume name left adjusted and padded with spaces, written in the volume label of the input disk (DOS). No spaces are allowed in the volume name.

VOLUME NAME 2 : Volume name left adjusted and padded with spaces, written in the volume label of the output disk. (TOSS). No spaces are allowed in the volume name.

FILENAME 1 : File name left adjusted and padded with spaces. This is the name of the UF-file containing the module names and userids of modules to be copied into the file.

USERID : Userid left adjusted and padded with spaces. This is the name of the user where the UF-file resides (FILENAME 1). No spaces are allowed in the userid.

FILE NAME 2 : File name left adjusted and padded with spaces. This is the file where the library file is stored. No spaces are allowed in the file name.

Return code :

A1 = 0	No error
A1 = 1	Disk not operable
A1 = 2	Disk I/O error
A1 = 3	Input filename unknown
A1 = 4	Userid unknown
A1 = 5	Input volume name unknown
A1 = 6	Target area of output volume name too small
A1 = 7	Output volume name unknown
A1 = 8	Not used
A1 = 9	Output filename unknown
A1 = 10	Flexible disk write protected

CRF	CREATE FILE	CRF

Description : This utility is used to create a file on volumes already formatted.

The program searches for an empty space large enough to hold the file. If such an area is not available and a standard file is being created, it searches for a maximum number of 4 areas together being large enough to contain the file. The order in which the maximum of 4 volume names are given determines the order in which the volumes are searched. In the case of a library file only one file extent is allowed.

The volumes on which the file has to be created must be on-line together. File codes /F0–/FB are used when searching for the volume names. If free space has been found, the VTOC s of all volumes are updated. The file organisation in the VTOC is set to 'S' in the case of a standard file and to 'L' in the case of a library file. The created file will be filled with "FREE" records. These records will contain spaces.

Non-standard files will at this moment be treated in the same way as standard files.

On the operator's printer there is also a listing giving the volume name, file extent number, extent base and extent length for all file extents of the created file.

Calling sequence :

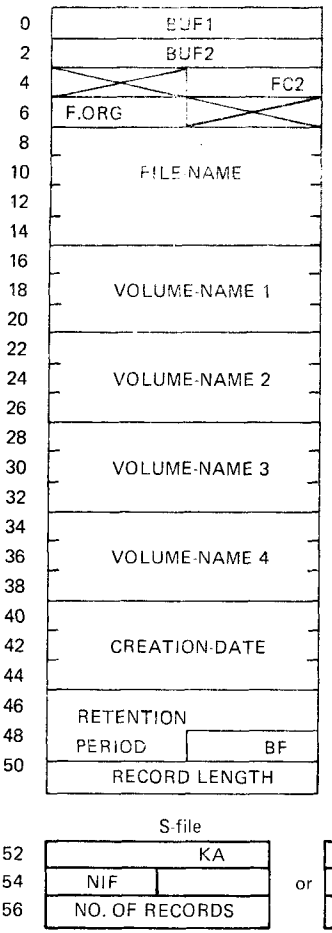
```
LDKL  A12, pb-address
CF    A14,CRFILE
```

CRF

Continued

CRF

Parameter block : word



BUF1 : Address of the sector buffer.
BUF2 : Address of the print buffer.

FC2 ; File code of the operator's print device.
F.ORG : File organisation copied to the VTOC for the created file.
FILENAME : File name left adjusted and padded with spaces, written in the VTOC for the created file. No spaces are allowed in the file name.

VOLUME NAME 1–4 : Names of volumes where a search for free space areas. Unused entries should contain spaces.
CREATION DATE : Six characters copied to the VTOC for the created file.
RETENTION PERIOD : Three characters copied to the VTOC for the created file.

BF : Blocking factor. That is the number of records within one block (one byte binary value). The BF is set to 1 if the file is a library file.

RECORD LENGTH : One word binary value. Note that the status character maintained on disk for each record is not included in the record length. The record length is set to 400 if this is a library file.

For S-files

NIF : One byte binary value, being the number of index files related to this data file.

KA : Binary value, indicating the relative character address of the symbolic key within the data record (0–n). This is only significant for index files and must otherwise be zero.

NO. OF RECORDS : 1 word binary value. Not relevant for L-files. Max. value = 8388607.

For L-files

MONITOR NO. : One byte binary value indicating which Monitor has to be used together with the application.

SOP SWITCH : One byte binary value indicating which switch on SOP-panel to be used at application loading time.

PROGRAM LENGTH : Program length in k bytes.

CRF **Continued** **CRF**

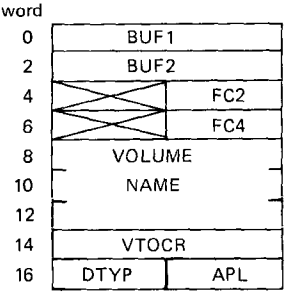
Return code : A1 = 0 No error
A1 = 1 Disk I/O error
A1 = 2 One or more volumes unknown
A1 = 3 Disk overflow
A1 = 4 File could not be allocated within 4 extents.
A1 = 5 Blocklength greater than 400 characters.
A1 = 6 VTOC overflow
A1 = 7 Filename already used
A1 = 8 Flexible disk write protected.

CRV	CREATE VOLUME	CRV

Description : This utility is used to format a disk pack before it will be used. It writes a volume label (VL) and an empty volume table of contents (VTOC) on disk. The program also writes cylinder identifiers in all sectors on the disk and tests the quality of each sector. If an unusable sector is found, this sector is withdrawn from the user available area and a dummy file BADSPOT is created which occupies the sectors not to be used. File organization is set to "B". The free space administration table in the VTOC is also updated for the "badspot" area. If the disk is PTS6875 or PTS6876, the Initial Program Loader (IPL) is stored in Sector 1, Cylinder 0, Track 0; if flexible disk it is stored in physical sectors 5–8, Track 00.
It is possible to choose if the IPL shall read the application directly and start the system (only one application on disk) or first wait for SOP input to define the application that shall be read (more than one application on disk).

Calling sequence :
```
LDKL   A12,pb-address
CF     A14,CRVOL
```

Parameter block : word



BUF1 : Address of sector buffer 1.
BUF2 : Address of sector buffer 2.
FC2 : File code of the operator's print device.
FC4 : File code of the disk to be formatted.
VOLUME NAME : Volume name left adjusted and padded with spaces written in the volume label of the disk to be formatted. No spaces are allowed in the volume name.
VTOCR : Number of empty records reserved in the volume table of contents (VTOC) (1 word binary value).

CRV **Continued** **CRV**

DTYP : Disk unit type indicator. Zero means PTS6875 or Flexible Disk, non-zero means PTS6876.

APL : Indicator for the number of applications to be stored on the disk. Zero means one application. Not zero means more than one application.

Return code : A1 = 0 No error
A1 = 1 Disk spot operable
A1 = 2 Bad spot on track zero. Disk not usable
A1 = 3 More than 5 bad spots. Disk not usable
A1 = 4 Disk I/O error (A2 = driver return code)
A1 = 5 Flexible disk write protected.

CTF — COPY TAPE TO FILE — CTF

Description : This utility copies from a cassette tape or ½" magnetic tape file to a disk file. The disk file is written with sequential write, so that if the output file is not empty, the new data will be appended. Cassettes can be read with or without labels, magnetic tapes only without labels.

The following fields in the label records will be checked:

- label identifier
- file section number
- volume identifier (only if specified)

Blocked records on input are accepted, and deblocking will be performed.

If input records are blocked, the blocksize must be a multiple of the record length for the output file, otherwise records with a different size will be truncated or expanded with blanks, to fit into the output file.

In the case of a labelled input tape, the check of the header label might fail for one of the following three reasons:

- erroneous label identifier: this is not a header label record.
- erroneous volume identifier: the volume identifier specified in the parameter block differs from the corresponding field in the header label record.
- file section sequence error.

In these cases a message and the label record will be displayed on the operators console.

The operator will then be requested to enter one of the following three commands:

GO	- ignore the error
REMOUNT	- another volume should be mounted
ABORT	- terminate the copying

Calling sequence :

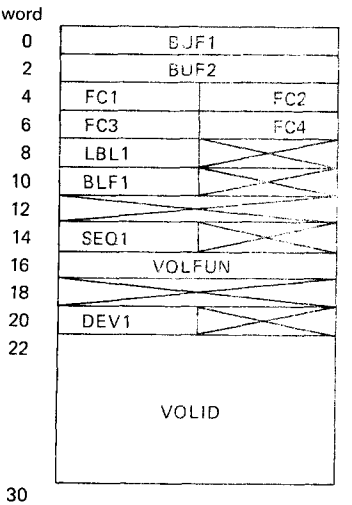
```
LDKL  A12,pb-address
CF    A14,COPYTF
```

CTF

Continued

CTF

Parameter block :



BUF1 : The address of a buffer, large enough to contain one tape block.
BUF2 : The address of a buffer, at least 128 words.
FC1 : Operator's input device.
FC2 : Operator's output device.
FC3 : File code for the input device.
FC4 : File code for the output device.
LBL1 : Labelled tape flag. This must be 1 if the input tape is labelled: 0 otherwise.
BLF1 : Block factor on input tape (binary value). This must be 1 if no blocking is used.
SEQ1 : Sequence number flag (binary value). If = 1, block sequence numbers are expected on the input tape.
VOLFUN : Subroutine address. When the end of volume label is read on tape, this routine is called (with a CF A14, ... instruction), to request mounting of the next volume. Upon return FC4 and - eventually - VOLID are assumed to be filled in with appropriate values.

CTF **Continued** CTF

DEV1 : Input device type - one ISO–7 character:
'T' if tape cassette, 'M' if ½" magnetic tape.

VOLID : Volume identifier of the input tape.
If it is not all spaces, this field is compared with the corresponding field in HDR-label.

Return code :

A1 = 0	No error
A1 = 1	Output I/O error (A2 = driver return code)
A1 = 2	Input I/O error (A2 = driver return code)
A1 = 4	Tape mark missing
A1 = 5	Tape mark after label missing
A1 = 6	Label record missing
A1 = 7	Unexpected tape mark
A1 = 8	Label record error
A1 = 9	Input length error - only when blocked input. This means that the blocksize is not a multiple of the record length.
A1 = 10	End of File detected inside a split record.
A1 = 11	HDR label check, A2 = 1 : Label identifier error – not HDR A2 = 2 : Volume identifier check A2 = 3 : File section sequence error.
A1 = 12	Output file not large enough.

CTI — **COPY TOSS FILE TO IBM DATA SET** — **CTI**

Description : This utility is used to convert a file from a TOSS-labelled disk to a data set on an IBM-labelled disk. It cannot be used to copy L-type files.

The program can copy with or without packing. If packing is used, records indicated as deleted in the input file will not be copied, and the output file will be written sequentially. This means that files which have been updated by random read must be copied without packing.

Records placed after the EOF in the input file will be copied.

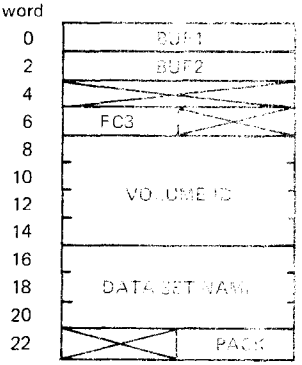
The input records must be in ASCII format and will be converted to EBCDIC by the program.

Multivolume data sets are not allowed.

File codes 128–255 are used when searching for the output volume identity. Sectors 0–3 on Track 00 are used when searching for the data set name.

Calling sequence : LDKL A12,pb address
CF A14,COPYT

Parameter block : word



BUF1 : Address of record buffer 1
BUF2 : Address of record buffer 2
FC3 : File code of input file
VOLUME ID : Volume name of output disk
DATA SET NAME : Name of output data set
PACK : Pack indicator. Zero indicates no packing

CTI **Continued** **CTI**

Return code : A1 = 0 No error
A1 = 1 Disk not operable
A1 = 2 Disk I/O error
A1 = 3 Faulty record length
A1 = 4 Output file not large enough
A1 = 5 Output file not empty
A1 = 6 Data set name not found
A1 = 7 Output volume name unknown
A1 = 8 Flexible disk write protected

DLF — DELETE FILE — **DLF**

Description : This utility deletes a file containing a loadable program or data file. The file to be deleted resides on one or more volumes. Up to eight volume names are entered.

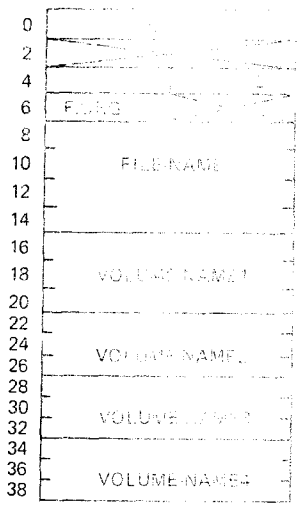
The free space bitmap contained in VTOC is updated for the released space. The file descriptor record in VTOC is set to zero and the first character of the file name.

File codes /F0—/F9 are used for the volume names.

Calling sequence : LDKL A12,pb
CF A14,DLF

Parameter block :

word



BUF1 — Address of the sector buffer
F.ORG — File organisation.
FC2 — File code of the operator's print device.
FILENAME : File name of the file to be deleted.
VOLUME NAME : Name of volumes where a search is made for the file to be deleted.
Unused entries should contain spaces.

DLF **Continued** DLF

Return code	:	A1 = 0	No error
		A1 = 1	Disk I/O error
		A1 = 2	Volume name unknown
		A1 = 3	File name unknown
		A1 = 4	No entry available in free space table of VTOC
		A1 = 5	Flexible disk write protected.

PDS **PRINT DISK SECTOR** **PDS**

Description : This utility will give a listing of the contents of one or more sectors on a specified disk. Output format is hexadecimal together with ISO–7 representation. 410 bytes per sector are printed, except in the case of an IBM-formatted flexible disk, in which case 128 bytes per sector are printed.

Calling sequence :
```
LDKL  A12,pb-address
CF    A14,PRDISC
```

Parameter block :

word		
0	BUF1	
2	BUF2	
4		
6	FC3	FC4
8	SECTOR NUMBER 1	
10	SECTOR NUMBER 2	

BUF1 : Address of the disk sector buffer
BUF2 : Address of the print buffer
FC3 : File code of the print device
FC4 : File code of the disk to be printed
SECTOR NUMBER 1: Sector number of the first sector to be printed
SECTOR NUMBER 2: Sector number of the last sector to be printed.

Return code :
A1 = 0 No error
A1 = 1 Disk not operable
A1 = 2 Disk I/O error
A1 = 3 Printer not operable
A1 = 4 Illegal sector number

PIT

PRINT INDEX TRACK

PIT

Description : This utility is used to print a listing of all the data relevant to the index track of an IBM-formatted flexible disk, i.e. Track 00, Sectors 1–26.

For each data set defined on the volume, the following information is printed –

- Data Set Label Sector
- Data Set Name
- Record Length
- Beginning of Extent Track and Sector Number
- End of Extent Track and Sector Number
- Creation Date
- Expiry Date
- End of Data Track and Sector Number

Calling sequence : LDKL A12,pb-address
CF A14,PRINDX

Parameter block : word

0	BUF1	
2	BUF2	
4		FC2
6	FC3	FC4

BUF1 : Address of disk sector buffer
BUF2 : Address of print buffer
FC2 : File code of operator's print device
FC3 : File code of print device
FC4 : File code of disk containing Index Track to be printed

Return code : A1 = 0 No error
A1 = 1 Disk not operable
A1 = 2 Disk I/O error (A2 = Return Code)
A1 = 3 Printer not operable
A1 = 4 Disk is not IBM format

PRF **PRINT FILE** **PRF**

Description : This utility lists the records in a file on the line printer or operator's console.

Calling sequence : LDKL A12,pb-address
CF A14, PRFILE

Parameter block :

word

0	BUF1	
2		
4		
6	FC3	FC4
8	CNV	PRNR
10	LWB	
12	UPB	

BUF1 : Buffer address.
FC3 : File code for the input file.
FC4 : File code for the print device.
CNV : Output conversion:
= 0 – compact format
= 2 – spaced format
= 4 – hexadecimal format.
PRNR : Record number print flag
≠ 0 – the record number will be printed before data.
LWB : Number of the first record to be printed. } Bit 0 must be zero
UPB : Number of the last record to be printed. }

Return codes : A1 = 0 No error
A1 = 1 Input error (A2 = driver return code)
A1 = 2 Disk I/O error (A2 = return code)
A1 = 3 Printer not operable
A1 = 4 End of medium (A2 = last record number in file)

PVC	PRINT VOLUME TABLE OF CONTENTS	PVC

Description : This utility will give a listing of all relevant data in the VTOC. For all datasets defined in the VTOC the following items will be printed.

- file name
- file extent base
- file extent length
- file organisation
- record length
- blocking factor
- last record number
- file section number
- file extent number
- creation date
- retention period
- number of index files
- symbolic key address in data record
- monitor number
- switch number on SOP

Number of free records in the VTOC and free space administration table are also printed.

Calling sequence :
```
LDKL  A12,pb-address
CF    A14,PRVTOC
```

Parameter block :

word		
0	BUF1	
2	BUF2	
4		
6	FC3	FC4

BUF1 : Address of the disk sector buffer
BUF2 : Address of the print buffer
FC3 : File code of the print device
FC4 : File code of the disk containing the VTOC to be printed.

Return code :
A1 = 0 No error
A1 = 1 Disk not operable
A1 = 2 Disk I/O error (A2 = driver return code)
A1 = 3 Printer not operable
A1 = 4 Not TOSS disk format.

RIX

REORGANIZE INDEX FILE

RIX

Description : This utility is used to create an index file and a master index file from a sorted file previously created by the BIX utility. This file is read until End-of-medium.

All files used must reside on disk, and the output files must have been predefined with the CRF utility. The index file may not span more than one volume. The master index file must reside on the **same** volume as the index file, and must be contained within one extent. It must have a record length equal to RL–3, where RL is the record length of the index file.

The utility moves index records sequentially from the input file to the output file. Free records are inserted at the end of each sector according to the Load Factor parameter supplied by the program. The master index is also written sequentially.

A check is performed on the input record sequence, and the processing terminated if an error is detected.

Each record in the master index file corresponds to one master index entry. The maximum number of index records per master index entry is calculated according to the following formula:

$$NIM = \uparrow \frac{NORI}{FELM \times BFM}$$

where $\uparrow$ = Next higher multiple of Index file blocking factor
NORI = Number of records in Index file
FELM = Number of blocks in Master Index file
BFM = Blocking factor of Master Index file

The number of used index records per sector of the output index file is calculated from the Load Factor parameter, according to the following formula:

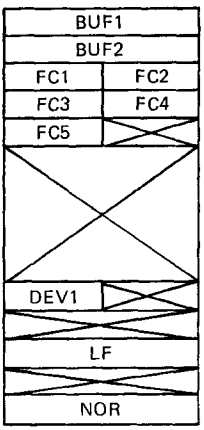
$$NUI = \downarrow BF \times \frac{LF}{100}$$

where $\downarrow$ = Next lower integer value
BF = Blocking factor of Index file
LF = Load factor

Calling sequence :
```
LDKL  A12,pb-address
CF    A14,RINDEX
```

RIX **Continued** RIX

Parameter block : word



BUF1 : Input file sector buffer address
BUF2 : Output index file sector buffer address
FC1 : File code of operator's input device; set to zero if no operator-controlled abort is required.
FC2 : File code of operator's print device; set to zero if no print required
FC3 : File code of input file
FC4 : File code of output Index file
FC5 : File code of output Master Index file
DEV1 : Device indicator; must be set to D
LF : Load Factor for output index file, i.e. percentage of the sector filled with used records.
NOR : Number of reserved records in the output Index file. A binary value in three characters with bit 0 in the second character set to zero.

Return code : A1 = 0 No error
A1 = 1 Input I/O error (A2 = return code)
A1 = 2 Output I/O error (A2 = return code)
A1 = 3 Blocksize error
A1 = 4 Not used
A1 = 5 Key definition error
A1 = 6 Load factor too small
A1 = 7 Record length error
A1 = 8 Output file not large enough
A1 = 9 Output file not empty
A1 = 10 Not used
A1 = 11 Key sequence error
A1 = 12 File organization error

SCT	SCAN TAPE	SCT

Description : This utility is a comprehensive tool for examining the contents of a tape cassette or ½" magnetic tape. The program interacts with the operator by means of a small set of commands for tape positioning and printout control.

On entry, the utility executes a load request for the input device. Then commands are requested, interpreted and executed, until an exit command is encountered, when an unload request is executed on the input device and control is returned to the calling program.

The following control commands are available:

R - rewind tape
F - forward positioning
B - backward positioning
L - list blocks on operator's console
P - list blocks on print device
O - change output format
X - return to calling program

The command syntax is defined as follows:

One command ::= com// eoc | com // parlist // eoc

com ::= one character command-mnemonic
eoc ::= end-of-command character
parlist ::= par | npar // aparlist
| npar // sep // parlist
| apar // parlist
| sep // parlist
par ::= apar | npar
apar ::= one-alphabetical-character
npar ::= unsigned-decimal-number
aparlist ::= parlist -starting-with-an- apar
sep ::= comma | space

The "command list" feature allows the operator to write several commands on the same line, and then have them executed in the order they were entered. A command list is a list of commands, separated by periods and followed by an end-of-record key. A command can be cancelled by pressing (CR) before the end-of-command character * is entered.

The commands are described in detail in the following paragraphs:

R - rewind tape
syntax: R // eoc
function: the tape is rewound to BOT
F - forward positioning
Syntax: F [F] [n1 [,n2]] eoc

SCT **Continued** SCT

F: If this parameter is given the positioning is performed on file (i.e. tape mark) level, otherwise on block level.

n_1: numeric parameter-number of blocks or, if the parameter F is present the number of files-to skip.

n_2: numeric - number of tapemarks after which the operation stops. Only significant if the parameter F is absent.

B - backward positioning

Syntax: B [F] [n_1 [, n_2]] eoc

parameters as for the command F.

L - list blocks on operator's console

Syntax: L: [n_1 [, n_2]]

n_1: number of blocks to list.

n_2: file-count - the operation will stop when n_2 tape marks have been encountered.

P - list blocks on print device

Syntax: P [n_1 [, n_2]]

parameters as for the command L.

O - change output format

Syntax: O [fmt] [n_1 [, n_2]]

fmt: one or more output format mnemonic characters, valid characters are:

A: ISO–7 code
E: EBCDIC code
C: compact format
S: spaced format
H: hexadecimal format

any other character will be ignored.

n_1: first position in block to output.

n_2: last position in block to output.

default for n_1 and n_2 : the contents of the whole block is printed. Character position count starts with 0.

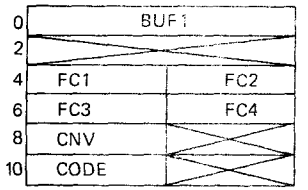
X - exit from the utility

Syntax: X

Calling sequence : LDKL A12, pb-address
CF A14,SCANT

SCT Continued **SCT**

Parameter block :



BUF1 : Address of the buffer
FC1 : File code for the operator's input device.
FC2 : File code for the operator's output device.
FC3 : File code for the input tape.
FC4 : File code for the print device.
CNV : Output conversion format code
0 = compact format
2 = spaced format
4 = hexadecimal format
CODE : Character code
0 = ISO–7
2 = EBCDIC

Return code : A1 = always zero.

SRT

SORT FILE

SRT

Description : This utility is used to sort records in a file into a sequence determined by key fields contained in the records. The key may consist of up to 15 subkeys, each subkey consisting of a number of eight-bit characters.

The utility requires two or three disk files. If only two files are used, the input file is destroyed, so it is important to keep a backup copy of the file until the sort has been successfully processed. If three files are used, the input file is unaffected.

The input and output files may have different record lengths, in which case records are truncated and/or padded with blanks, as appropriate. If the files have different record lengths, the sort keys must be contained within the shortest record length.

The work files and the output file must have been predefined with the CRF utility. All files must reside on disk. Files are handled with Random Access, and the LRN of the output file is not significant.

The utility also requires a work area in memory, which should be defined as large as possible to decrease the length of time required for sorting. The maximum size is 64 Kbyte. The minimum core size can be calculated by the following formula:

$$4 \times RL \times BFC$$

where RL = Effective record length (see parameter block layout)
BFC = Lowest common blocking factor of input, output and work files

A stable sort algorithm is used, which means that the initial relative order of equal keys in the input file is kept in the output file.

The sort executes in the following four phases:

Precalculation: Secondary sort parameters (number of passes, etc) are calculated and the results output to the operator's print device. The operator may abort the sort at this stage.

Presort: Used records are read in sequence from the input file to the memory area. Records in memory are sorted and written in sequence to a work file, forming a 'file division'. This phase is repeated until the entire input file is moved to the other file.

SRT — Continued — SRT

Merge: Records from separate file divisions on one file are merged and output to another file forming larger file divisions. Statistics are printed on the operator's print device. This phase is repeated until the output file contains only one file division.

Check: The output file is read in sequence and fifteen checks are carried out on it. Statistics are gathered and output on the operator's print device.

At the end of the sort processing, the TNR field in the parameter block (see below) is updated with the number of records read from the output file in the Check Phase, excluding free records.

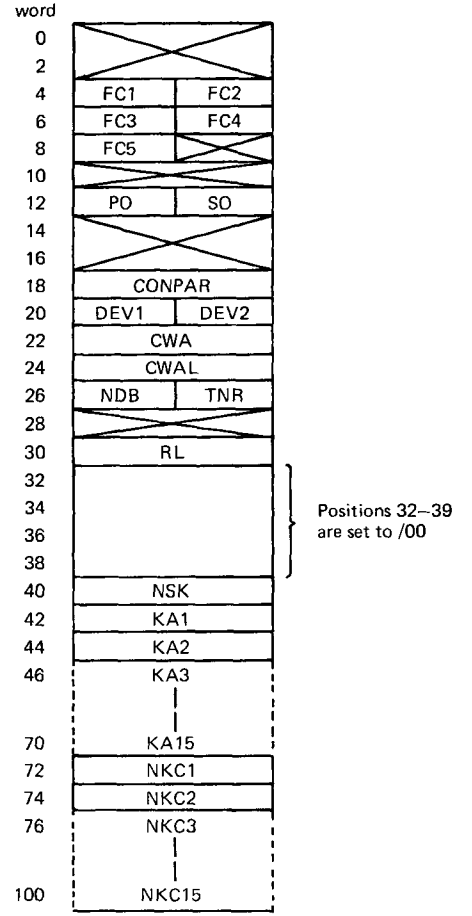
Calling sequence : LDKL A12,pb-address
CF A14,SORTF

SRT

Continued

SRT

Parameter block : word



FC1 : File code of operator's input device; set to zero if operator abort function is not required
FC2 : File code of operator's print device; set to zero if not print required
FC3 : File code of input file
FC4 : File code of output file
FC5 : File code of work file

Note: When FC3 and FC4 are the same, the output file will overwrite the input file.
When FC3 and FC5 are the same, the input file is used as a work file.

PO : Processing order; 0 = single task processing
1 = multitask processing
SO : Sorting order; 0 = descending sequence
1 = ascending sequence
CONPAR : Address of the linked-in object module parameter block used for this sort
DEV1 : Input device indicator; set to D
DEV2 : Output device indicator; set to D
CWA : Core work area address
CWAL : Size of the work area in memory in number of bytes; binary value <65536
NDB : Number of disk buffers available in the MONITOR. A binary value in the range 1–5. The higher the number, the shorter the sorting time.
TNR : Number of records that participate in the sort; both free and used records in the input file are read up to this number of records. If set to zero, the entire input file is used. Binary value in three characters; bit 0 in the second character must be set to zero
RL : Binary field indicating the requested effective record length. This is the number of leading characters in the input record that participate in the sort, and the sort key must be contained within this length. If set to zero the effective record length is the record length of the input file
NSK : Binary field indicating the number of subkeys, 1–15
KA1–15 : Binary fields indicating the relative character positions of the subkeys in the input records. KA1 refers to the most significant subkey. The first position in the record is assumed to be 0 for this field, and the maximum value is 255
NKC1–15 : Binary fields indicating the number of characters in each of subkeys KA1–15 respectively to a maximum value of 255

SRT **Continued** **SRT**

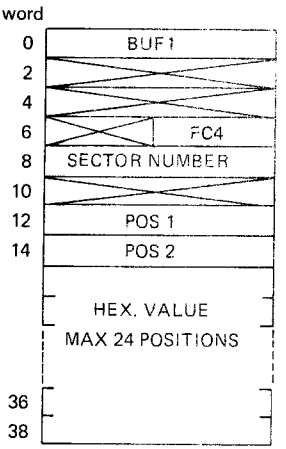
Return code : A1 = 0 No error
A1 = 1 Input I/O error (A2 = return code)
A1 = 2 Output I/O error (A2 = return code)
A1 = 3 Not used
A1 = 4 Not used
A1 = 5 Key definition error
A1 = 6 Work area in memory too small
A1 = 7 Record length error
A1 = 8 Output file not large enough
A1 = 9 File organization not S
A1 = 10 Not used
A1 = 11 Incorrect TNR (see parameter block)

UDS UPDATE DISK SECTOR **UDS**

Description : This utility is used to change one or more positions in a specified sector on disk. The utility may be used on IBM-formatted flexible disks.

Calling sequence : LDKL A12,pb-address
CF A14,UPDISC

Parameter block :



BUF1 : Address of the disk sector buffer.
FC4 : File code of the disk to be updated.
SECTOR NUMBER : Sector number of the sector to be updated.
POS1 : Character position of the first character to be updated.
POS2 : Character position of the last character to be updated.
First position in the disk sector is POS = 0.
Note that position 0 and 1 cannot be changed as they are reserved for the System.
HEX VALUE : A string of up to 24 characters giving the hexadecimal representation of the positions to be changed.

Return code :
A1 = 0 No error
A1 = 1 Disk not operable
A1 = 2 Disk I/O error
A1 = 3 Flexible disk write protected
A1 = 4 Illegal sector number
A1 = 5 Illegal position number (only for IBM disk format).

WIL	WRITE IBM LABELS	WIL

Description : This utility is used for one of three purposes: to write labels on a flexible disk in order to prepare it for further use and create a data set at the same time, to subsequently create further data sets on the disk, or to delete data sets from the disk.

The program initialises track 00 on the disk by writing the volume identity into positions 5—10 of sector 07, and subsequently uses sectors 08—26 for each data set label. It is recommended that sector 08 is used when creating the first data set, and then 09,10, etc. in sequence. When creating further data sets, the program checks that the volume identity is correct and that the data set name does not already exist on the disk. The positions in each sector used to record data set label information are as follows:

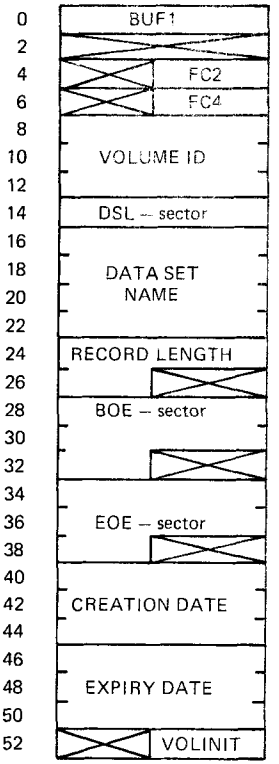
Position	6—13	User name for data set
	23—27	Record length
	29—33	Beginning of extent
	25—29	End of extent
	48—53	Creation date
	67—72	Expiry date
	75—79	End of data

Calling sequence :

```
LDKL  A12,pb-address
CF    A14,IBMLBL
```

WIL Continued WIL

Parameter block : word



BUF1 : Address of disk sector buffer
FC2 : File code of operator's print device
FC4 : File code of disk (/F8–/FB)
VOLUME ID : Volume identity – six alphanumeric characters, left-justified, and with trailing blanks if necessary.
DSL – sector : Binary field, containing the physical sector number of the data set label.

WIL **Continued** WIL

DATA SET NAME : User data set name (max 8 characters)
RECORD LENGTH : 3 digits specifying how much of each 128 position sector contains actual data.
BOE – sector : Physical sector of beginning of extent, in the format TT0SS, where
TT = track number
0 = zero
SS = sector number
EOE – sector : Physical sector of end of extent, in the same format as beginning of extent.
Note: To delete a data set from the disk, the BOE and EOE fields in the block must be set to 74001 and 73026.
CREATION DATE : Six digits in the format YYMMDD or blanks
EXPIRY DATE : Six digits in the format YYMMDD or blanks
VOLINIT : Volume initialisation indicator. Zero means initialisation is not required.

Return code : A1 = 0 No error
A1 = 1 Disk not operable
A1 = 2 Disk I/O error (A2 = return code)
A1 = 3 Volume identity unknown
A1 = 4 Data set name already used
A1 = 5 Flexible disk write protected
A1 = 6 Not used
A1 = 7 Data set definition error

APPENDIX A I/O AND ACTIVATION

This appendix provides an example of the way in which the I/O and activation LKM request can be used. This example is intended to illustrate the circumstances under which the I/O and activation request may be of use.

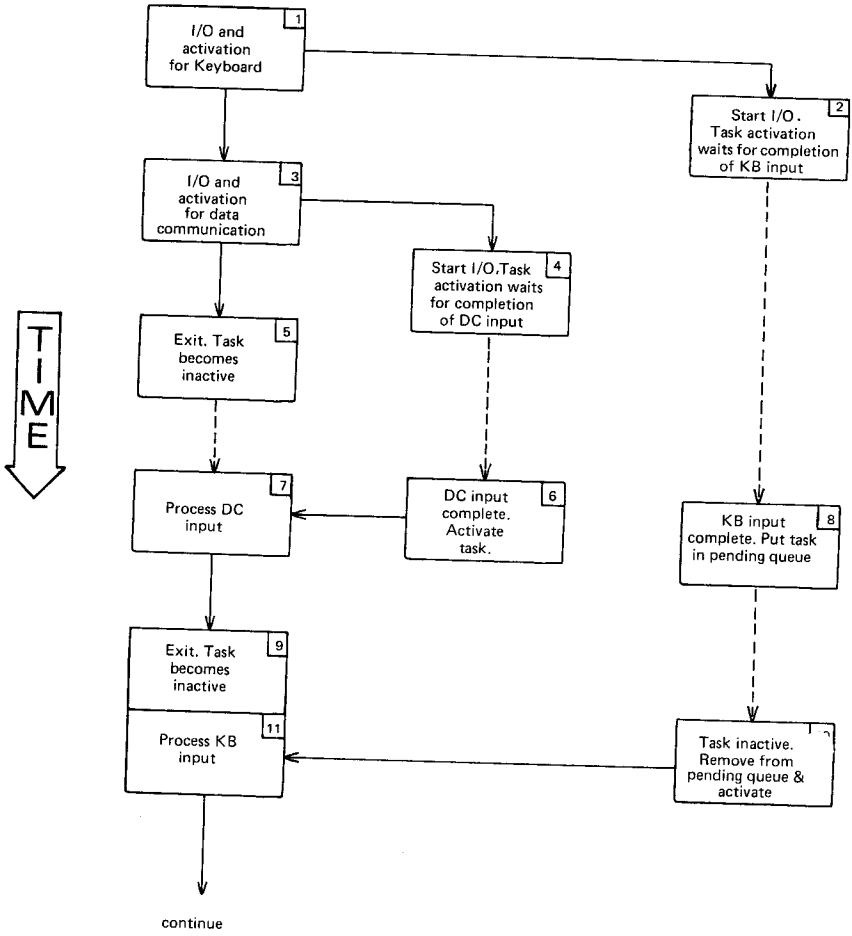
The processing carried out in the example could of course be organised differently. The reasons for using an I/O and activate request and the way in which the associated processing is organised is, of course, a function of application objectives and programming style.

Consider an application task in which the following situations exist:

- Input is needed from the keyboard.
- Input is expected from the data communication line.

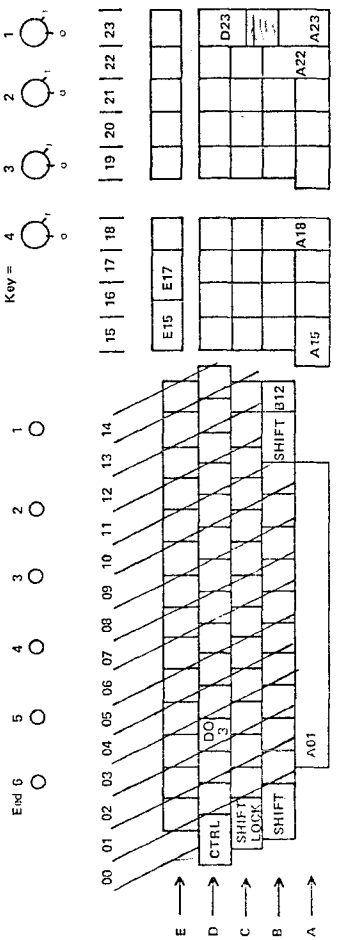
In both these cases the time required to complete an input is indeterminate: it may be milliseconds, seconds, minutes or longer. Also, the input must be processed as soon as it is complete. A solution is to use I/O and activation to handle both input operations and then to perform an exit LKM request. The task will thus be inactive until one of the inputs is complete. The task will be reactivated at a specified start point at the completion of each input.

The following diagram and explanation amplify these points.


1
I/O and activation for Keyboard
2
Start I/O. Task activation waits for completion of KB input
3
I/O and activation for data communication
4
Start I/O.Task activation waits for completion of DC input
5
Exit. Task becomes inactive
TIME
7
Process DC input
6
DC input complete. Activate task.
8
KB input complete. Put task in pending queue
9
Exit. Task becomes inactive
11
Process KB input
Task inactive. Remove from pending queue & activate
continue

1. An I/O and activation request is issued for the keyboard.
2. The Monitor starts the I/O operation but does not yet try to activate the task.
3. An I/O and activation request is issued for the D.C. line.
4. The Monitor starts the I/O operation but does not yet try to activate the task.
5. An exit request is issued. The task becomes inactive.
   Other active tasks may now be dispatched.
6. D.C. input is complete and the task is placed in the dispatcher queue. It is activated at the start point specified in 3. (Activation is immediate in the diagram, but in reality the task would probably have to queue).
7. The D.C. input is processed.
8. While D.C. input is being processed the KB input completes. The Monitor cannot activate the task because it is already active, so the task goes into the pending queue.
9. An exit request is issued. The task again becomes inactive.
10. The Monitor immediately removes the second activation from the pending queue and places the task in the dispatcher queue. It is activated at the start point specified in 1. (Activation is immediate in the diagram, but in reality the task would probably have to queue).
11. The KB input is processed.

APPENDIX A: CODES GENERATED FOR KEYBOARD PTS 6236



Code list key pad layout.

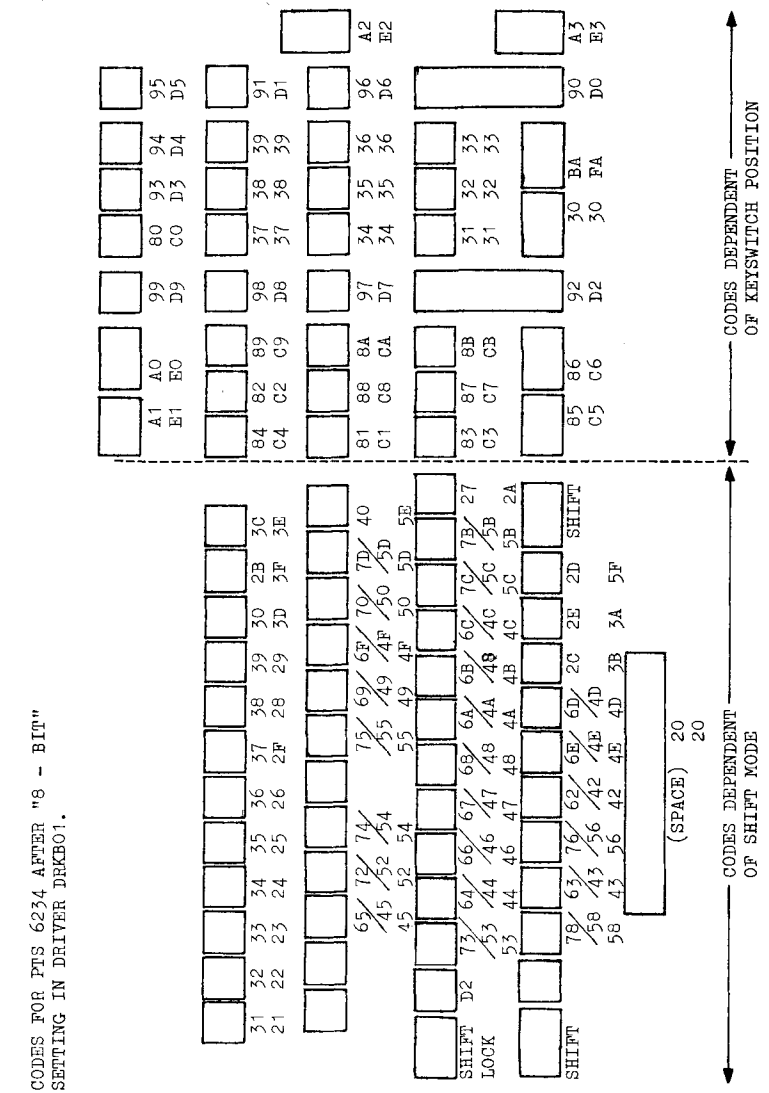
Code List

Row \ Column	0	1	2	3	4	5	6	7
0	A15	A19	A01	E10	E12	D10	E15	key 1 →1
1	B15	B19		E01	C01	D01	E17	key 1 →0
2	B16	B20		E02	B05	D04	E18	key 2 →1
3	B17	B21		E03	B03	C02	E19	key 2 →0
4	C15	C19		E04	C03	D05	E20	key 3 →1
5	C16	C20		E05	D03	D07	E21	key 3 →0
6	C17	C21		E06	C04	B04	E22	key 4 →1
7	D15	D19	C12	E07	C05	D02	E23	key 4 →0
8	D16	D20		E08	C06	B02		SHIFT →1
9	D17	D21		E09	D08	D06		SHIFT →0
A	A16	A20		B12	C07	B01		CTRL →1
B	A17	A21	E11	C13	C08	C11		CTRL →0
C	A18	A22	B08	E13	C09	C10		
D	C18	C22	B10	D13	B07	D11		
E	D18	D22	B09	D14	B06	D12		
F	A23	D23		E14	D09			

Index

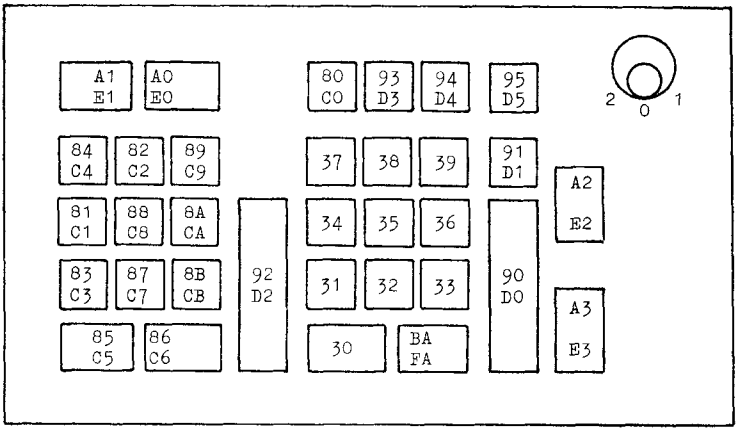
Key	Index in Function Cluster	Key	Index in Function Cluster	Key	Index in Shift Unshift CTRL and SHIFT/CTRL Cluster	Key	Index in Shift Unshift CTRL and SHIFT/CTRL Cluster
A15	1	C22	30	A01	1	C05	40
B15	2	D22	31	C12	8	C06	41
B16	3	D23	32	E11	12	D08	42
B17	4	E15	33	B08	13	C07	43
C15	5	E17	34	B10	14	C08	44
C16	6	E18	35	B09	15	C09	45
C17	7	E19	36	E10	17	B07	46
D15	8	E20	37	E01	18	B06	47
D16	9	E21	38	E02	19	D09	48
D17	10	E22	39	E03	20	D10	49
A16	11	E23	40	E04	21	D01	50
A17	12			E05	22	D04	51
A18	13			E06	23	C02	52
C18	14			E07	24	D05	53
D18	15			E08	25	D07	54
A23	16			E09	26	B04	55
A19	17			B12	27	D02	56
B19	18			C13	28	B02	57
B20	19			E13	29	D06	58
B21	20			D13	30	B01	59
C19	21			D14	31	C11	60
C20	22			E14	32	C10	61
C21	23			E12	33	D11	62
D19	24			C01	34	D12	63
D20	25			B05	35		
D21	26			B03	36		
A20	27			C03	37		
A21	28			D03	38		
A22	29			C04	39		

APPENDIX B CODES GENERATED FOR KEYBOARD PTS 6234



APPENDIX C CODES GENERATED FOR KEYBOARD PTS 6231

Codes for PTS 6231 after "8-bit" setting in driver DRKBO1. Upper figures are valid for key-switch in position 1 and lower figures for key-switch in position 2.



APPENDIX D ISO CODE CHARACTER SET

Character/function declaration

Hex.	ISO Mnemonic	Declaration
0.0	NUL	Null
0.1	SOH	Start of Heading
0.2	STX	Start of Text
0.3	ETX	End of Text
0.4	EOT	End of Transmission
0.5	ENQ	ENQUVIRY
0.6	ACK	Acknowledge
0.7	BEL	Bell
0.8	BS	Backspace
0.9	HT	Horizontal Tab
0.A	LF	Line Feed
0.B	VT	Vertical Tab
0.C	FF	From Feed
0.D	CR	Carrige Return
0.E	SO	Shift Out
0.F	SI	Shift In
1.0	DLE	Data Link Escape
1.1	DC1	X-On Reader
1.2	DC2	X-On Puncher
1.3	DC3	X-Off Reader
1.4	DC4	X-Off Punch
1.5	NAK	Negative Acknowledge
1.6	SYN	Synchronous Idle
1.7	ETB	End of Transmission Block
1.8	CAN	Cancel
1.9	EM	End of Medium
1.A	SUB	Substitute
1.B	ESC	Escape
1.C	FS	File Separator
1.D	GS	Group Separator
1.E	RS	Record Separator
1.F	US	Unit Separator
2.0	SP	Space
7.F	DEL	Delete/Rub Out

There are 12 positions variable for national usage marked with

Per I/O device, the character set may vary.

$b_7b_6b_5$ → / $b_4b_3b_2b_1$ ↓ col / row	000	001	010	011	100	101	110	111
	0	1	2	3	4	5	6	7
0000 0	NUL	DLE	SP	0	@	P		p
0001 1	SOH	DC1	!	1	A	Q	a	q
0010 2	STX	DC2	"	2	B	R	b	r
0011 3	ETX	DC3	#	3	C	S	c	s
0100 4	EOT	DC4	$	4	D	T	d	t
0101 5	ENQ	NAK	%	5	E	U	e	u
0110 6	ACK	SYN	&	6	F	V	f	v
0111 7	BEL	ETB	'	7	G	W	g	w
1000 8	BS	CAN	(	8	H	X	h	x
1001 9	HT	EM	)	9	I	Y	i	y
1010 A	LF	SUB	*	:	J	Z	j	z
1011 B	VT	ESC	+	;	K	[	k	{
1100 C	FF	FS	,	<	L	\	l	|
1101 D	CR	GS	-	=	M	]	m	}
1110 E	SO	RS	.	>	N	^	n	
1111 F	SI	US	/	?	O	-	o	DEL

APPENDIX E EBCDIC CHARACTER SET

Bits 4567 ▼	Hex 1 ▼	00				01				10				11				Bits 0.1
		00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11	◄2.3
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	◄Hex C
0000	0	NUL	DLE			SP	&	-	0	SP	&	-	0	SP	&	-	0	
0001	1	SOH	SBA			A	J	/	1	a	j	/	1	A	J	/	1	
0010	2	STX	EUA		SYN	B	K	S	2	b	k	s	2	B	K	S	2	
0011	3	ETX	IC			C	L	T	3	c	l	t	3	C	L	T	3	
0100	4					D	M	U	4	d	m	u	4	D	M	U	4	
0101	5	PT	NL			E	N	V	5	e	n	v	5	E	N	V	5	
0110	6			ETB		F	O	W	6	f	o	w	6	F	O	W	6	
0111	7			ESC	EOT	G	P	X	7	g	p	x	7	G	P	X	7	
1000	8					H	Q	Y	8	h	q	y	8	H	Q	Y	8	
1001	9		EM			I	R	Z	9	i	r	z	9	I	R	Z	9	
1010	A					¢	!	¦	:	¢	!	¦	:	¢	!	¦	:	
1011	B					.	$	,	#	.	$	,	#	.	$	,	#	
1100	C	FF	DUP		RA	<	*	%	@	<	*	%	@	<	*	%	@	
1101	D		SF	ENQ	NAK	(	)	_	'	(	)	_	'	(	)	_	'	
1110	E		FM			+	;	>	=	+	;	>	=	+	;	>	=	
1111	F		ITB		SUB	|	¬	?	"	|	¬	?	"	|	¬	?	"	

: To be used for national characters

Characters within the white positions are translated to /00.

: Characters used for IBM 3270 Test Request messages.

Normally they are not used.

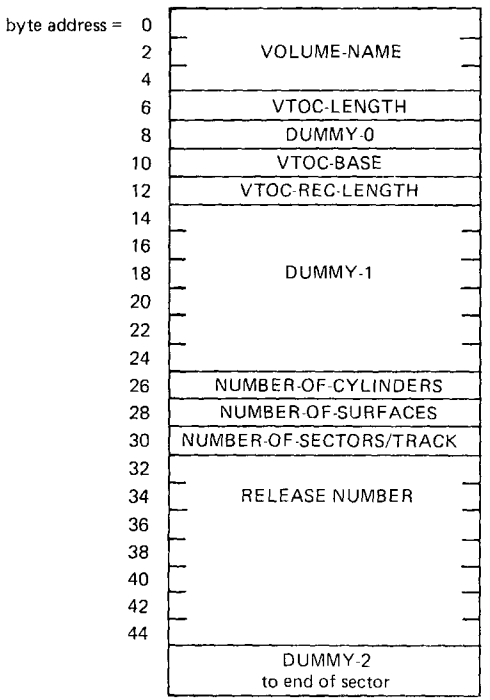
APPENDIX F: VOLUME LABEL AND VTOC FORMATS

F.1 Volume label

The volume label is located at the beginning of all disks and flexible disks.

Disk model	
PTS 6875 PTS 6876	Cylinder 0, track 0, sector 0
Flexible disk	Track 0, sectors 1–4 inclusive

The format of the volume label is shown below:



The fields have the following meaning:

VOLUME NAME
This is a string of 6 characters, left adjusted and padded with spaces. No spaces are allowed within the volume name.

VTOC-LENGTH
Number of sectors occupied by the VTOC.

DUMMY-0
Reserved for future use.

VTOC-BASE
The address of the sector on which the VTOC starts.

VTOC-REC-LENGTH
The size of one VTOC record, in bytes. The status character is not included.

DUMMY-1
Reserved for system use.

NUMBER-OF-CYLINDERS
The number of cylinders available to the user.

NUMBER-OF-SURFACES
For each surface of the disk there is only one track per cylinder. This field therefore shows the number of tracks per cylinder.

NUMBER-OF-SECTORS/TRACK
As stated.

RELEASE NUMBER
This field contains the text 'TOSS REL x.y.' where x is the release number and y is the level.

DUMMY-2
Filler to the end of the sector.

F.2 Volume Table of Contents (VTOC)

The VTOC contains two kinds of information about the volume. The first sector of the VTOC contains a table that describes all the free areas (extents) of the volume. The rest of the VTOC contains records that describe the used areas of the volume. The first is called the 'Free Space Administration Table', the second is called 'VTOC records'.

F.2.1 *Free space administration table*

This table has 50 entries that describe the start address length of all free extents on the volume. The format of each entry is shown below. Unused entries contain all zeros (X '00').

byte address =		
	0	DUMMY-0
	2	EXTENT LENGTH
	4	DUMMY-1
	6	EXTENT-BASE

The fields have the following meaning:

DUMMY-0
Reserved for future extentions.

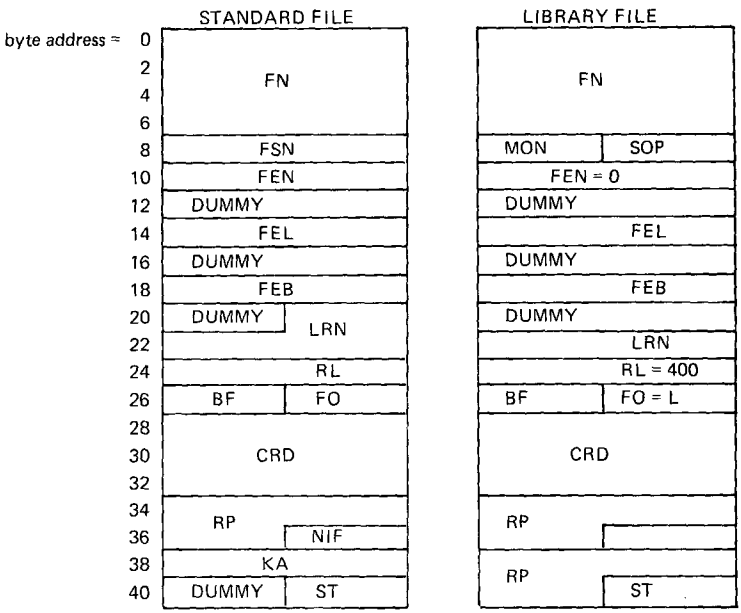
EXTENT LENGTH
The size of the free extent, expressed in number of sectors.

DUMMY-1
Reserved for future extensions.

EXTENT BASE
Logical record number of the first sector in this free extent.

F.2.2 *VTOC records*

Each record is 41 bytes long (40 bytes data + 1 byte status) and they are blocked 9 per sector. One record exists for every used extent in each file. There are two formats for VTOC records, one for standard files, one for library files, as shown below.



The fields have the following meaning:

FN = File Name

A string of 8 characters, left-adjusted and padded with spaces. This field must be set to spaces (X '20') for unused entries. No spaces are allowed within the File Name.

FSN = File Section Number

Two bytes binary value numbering the file sections. A file section is that part of a file which resides on one volume. File-section-nr starts from zero. The field contains other information for library file (see below).

FEN = File Extent Number

Two bytes binary value numbering the extents within each file-section. File-extent-nr starts from zero.

Dummy

Reserved for future extensions. All dummies will contain null characters, X '00'.

FEL = File Extent Length

A binary value that represents the number of sectors in the file extent.

FEB = File Extent Base

A binary value representing the logical sector number of the first sector in the extent.

LRN = Last Record Number

A binary value representing the logical record number of the last 'used' record in the file. There could be 'free' records in the file between the LRN and end-of-file.

RL = Record Length

A binary value representing the number of bytes occupied by the record. This is fixed for all records in this file and does not include the status byte.

BF = Blocking Factor

A binary value representing the number of records per block (i.e. per sector).

FO = File Organization

One character representing the file type. It can be:

S for standard files
L for library files
B for 'Bad spot' files
X for non-standard files

NIF = Number of Index Files

This is a binary value representing the number of index files belonging to this file.

KA = Key Address

This is a binary value representing the position of the first character of the symbolic key in the data file record. This field is only used for index files and is set to zero for other types of file.

ST = Status

This is a single character that indicates whether this VTOC record is used (X 'FF') or free (X '00'). This character is not included in the VTOC-REC-LENGTH defined in the volume label.

CRD = Creation Date

A string of 6 characters representing the date of creation of the file. The format can be either YYMMDD or YYDDD, left adjusted, where

YY = last two digits of the year
MM = month in the year
DD = day in the month
DDD = day in the year.

RP = Retention period

This is a string of 3 characters representing the number of days that this file is to be retained.

MON = Monitor number

This is a binary value representing the Monitor that is to be used together with the application library file. If the file is a monitor library file the monitor number should also be set to the appropriate value. The range of values is 1 to 99 inclusive.

SOP = SOP switch number

This is a binary value representing the SOP switch to be used for loading the application library file. This should be zero if the file is a monitor library file. The range of values is 0 to 10 inclusive.

PHILIPS

PTS 6800 TERMINAL SYSTEM

User Library

PTS 6800 ASSEMBLER PROGRAMMER'S REFERENCE MANUAL

Part 3

Module M06

Data
Systems

Date : January 1978

Code : 5122 993 42131

PREFACE

The Assembler Programmer's Reference Manual provides the information required to write, process and test Assembler application programs for the PTS 6800 computer used in the Philips PTS 6000 Terminal System.

Information is divided into three parts as follows :

Part 1 : Assembler Language
Additional functions
Recommended techniques

Part 2 : Monitor requests
I/O drivers
TOSS utilities

Part 3 : Assembler processor
TOSS system start
Assembler debugging program

Parts 1 and 2 contain the information needed to write an Assembler program. Part 3 contains the information needed to process and test an Assembler program.

Processing of Assembler programs (updating, assembling, etc.) is done under DOS 6800 System Software. The use of those parts of DOS 6800 System Software designed specifically for Assembler programs is described in Part 3 of this Manual. Information concerning the use of the general purpose components of DOS 6800 System Software is contained in the DOS 6800 System Software PRM (M11). Readers of the present Manual are expected to be familiar with the contents of the DOS 6800 System Software PRM.

The testing and production running of Assembler application programs is done under TOSS System Software. Information concerning TOSS System Software which is relevant to the writing, testing and running of Assembler application programs is included in Part 2 of the present Manual.

CONTENTS

	Date	Page
PREFACE	Jan. 1978	0.0.0
1. INTRODUCTION	Jan. 1978	1.0.1
	Jan. 1978	1.0.2
2. ASSEMBLER PROCESSOR		
2.1. General	Jan. 1978	2.1.1
	Jan. 1978	2.1.2
2.2. Input	Jan. 1978	2.2.1
	Jan. 1978	2.2.2
2.3. Output	Jan. 1978	2.3.1
	Jan. 1978	2.3.2
	Jan. 1978	2.3.3
	Jan. 1978	2.3.4
	Jan. 1978	2.3.5
3. TOSS SYSTEM START		
3.1. General	Jan. 1978	3.1.1
3.2. System Start Procedure	Jan. 1978	3.2.1
	Jan. 1978	3.2.2
	Jan. 1978	3.2.3
3.3. Deferred binding of Monitor Configuration Data	Jan. 1978	3.3.1
	Jan. 1978	3.3.2
	Jan. 1978	3.3.3
	Jan. 1978	3.3.4
4. ASSEMBLER DEBUGGING PROGRAM		
4.1. Introduction	Jan. 1978	4.1.1
4.2. Using DEBUG	Jan. 1978	4.2.1
4.3. DEBUG Input	Jan. 1978	4.3.1
	Jan. 1978	4.3.2
	Jan. 1978	4.3.3
	Jan. 1978	4.3.4
	Jan. 1978	4.3.5
	Jan. 1978	4.3.6
	Jan. 1978	4.3.7
	Jan. 1978	4.3.8
	Jan. 1978	4.3.9
4.4. Running DEBUG	Jan. 1978	4.4.1
	Jan. 1978	4.4.2
APPENDIX A : MEMORY ORGANIZATION		
A.1. Memory Layout	Jan. 1978	A.1.1
	Jan. 1978	A.1.2
A.2. Interrupt System	Jan. 1978	A.2.1

1. INTRODUCTION

An Assembler program may be input to the PTS 6000 system via any input device. After input, the source module is held on disk.
All processors and utilities mentioned in the flow-diagram read input from disk and write output to disk.

The following diagram illustrates the sequence of processes needed to develop and run an executable program from Assembler source modules.

Each source module is processed separately by the Assembler.
The Assembler produces object code modules. Each module may contain references to :

– Labels in the same module
– Labels in other Assembler modules
– Assembler System routines

These references are satisfied by the Linkage Editor. This processor builds an application load module from the following object modules :

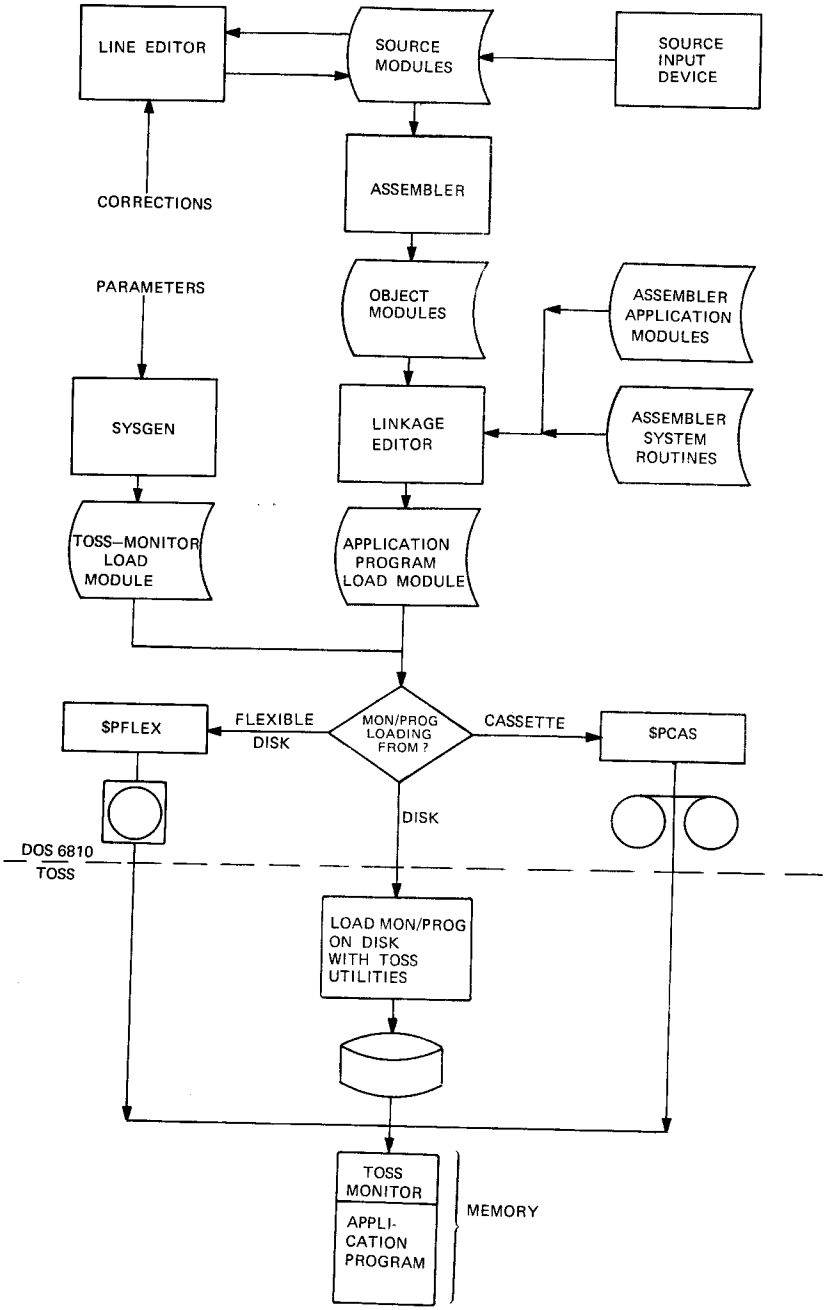
– Assembler application modules
– Assembler system routines (if referenced)

The output result is an application program load module. This module may be stored on magnetic tape cassette, TOSS formatted disk, or flexible disk.
The Assembler and Linkage Editor are run under the DOS 6810 Monitor.
The Load module produced by the Linkage Editor, however, must be run under the TOSS Monitor.

If any application program errors are detected during testing, one or more source modules will have to be corrected. This may be done via the Line Editor – an interactive text editor.
Each corrected source module must be reprocessed by the Assembler, then the whole program must be processed by the Linkage Editor.
The Assembler Debugging Program, if required must be linked to the TOSS Monitor during system generation. The Debugging program is an interactive diagnostic routine which enables the programmer to monitor and control the execution of his program. Control is handed to the Debugging program immediately after the TOSS Monitor and application program are loaded into memory (system start).

Information concerning TOSS System Software which is needed by Assembler programmers is contained in this Manual.

The Assembler processor, though part of DOS 6810 System Software, is discussed in this Manual because it is used by Assembler programmers only.
The remaining DOS 6810 System Software components used by Assembler programmers, notably the Linkage Editor, are described in the DOS 6810 System Software PRM (M11).


LINE EDITOR
SOURCE MODULES
SOURCE INPUT DEVICE
CORRECTIONS
ASSEMBLER
PARAMETERS
OBJECT MODULES
ASSEMBLER APPLICATION MODULES
SYSGEN
LINKAGE EDITOR
ASSEMBLER SYSTEM ROUTINES
TOSS–MONITOR LOAD MODULE
APPLICATION PROGRAM LOAD MODULE
$PFLEX
FLEXIBLE DISK
MON/PROG LOADING FROM ?
CASSETTE
$PCAS
DISK
DOS 6810
TOSS
LOAD MON/PROG ON DISK WITH TOSS UTILITIES
TOSS MONITOR
APPLI-CATION PROGRAM
MEMORY

2. ASSEMBLER PROCESSOR

2.1. General

An Assembly code program may be input to the PTS 6000 System via an input device. After input the source module is held on disk, either in a library or a temporary source file (/S), according to the programmer's requirements. The Assembler processor is held in the system library. It operates under control of the DOS 6810 Operating System and translates source modules from disk and outputs an object code module to disk. The object module can then be linked with other object modules using the Linkage Editor (see PTS 6000 System Software PRM, M11).
Figure 2.1 shows the sequence of events needed to develop and run an executable program from Assembly source modules.

Each source module is processed separately by the Assembler which produces object code modules. The instructions in these modules use a byte-oriented addressing system. Each module may contain references to :

- Labels in the same module
- Labels in other Assembly modules
- Application modules
- System routines

The Assembler processor optionally gives a listing of the program being processed. This listing provides the programmer with all the information he requires for a full record of the program :

- A line count in decimal
- Location counter in hexadecimal
- Hexadecimal representation of the instruction
- Type of address reference
- Source code statement
- Programmer's comments
- An error code at the place an error occurs
- Table of external labels
- Symbol table
- Total error count

The Assembler DEBUG program can be specified at system generation time if required. DEBUG is an interactive diagostic task which is executed in parallel with the Assembly program being tested. With DEBUG the programmer can monitor and control the execution of his program.

When errors have been discovered, the programmer can use the Line Editor to update the module with the correct statement (see DOS 6810 System Software PRM, M11).

The Assembler provides a facility for the programmer to insert directives into his program. These directives, although they are not program instructions, allow the programmer to guide the assembly process.
These directives can be inserted in the source program the first time it is assembled or input via the Line Editor. The directives are fully defined in volume 1 of this manual and a list is shown below.

Directive	Meaning
DATA	Data generation
EJECT	Continue listing on new page
END	End of Assembly
ENTRY	Define entry point name
EQU	Equate symbol to value or another symbol
EXTRN	Define external reference
FORM	Format definition
GEN	Generation directive
IDENT	Program identification
IFF	If false
IFT	If true
LIST	Resume listing output
NLIST	Suspend listing output
RES	Reserve memory area
XFORM	Extension of FORM directive
XIF	End of condition

2.2. Input

2.2.1. Preparation of the Source Module

Source modules can be input via any input device.
The statements can be prepared on these media in either of two formats :

1. They can be prepared in the same format as the program coding sheet with tabulation points set at the 1st, 10th, 19th and 41st column.
2. They can be prepared with a space or backslash (\) between each field.

Whatever media the module is being prepared on, the relevant labelling and structure standards apply (see DOS 6810 System Software PRM, M11). The Assembler always lists instructions in coding sheet format.

2.2.2. Source Input

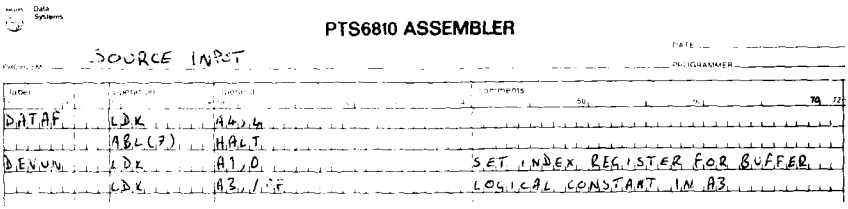
The source module must be input to the system by using the command :

RDS␣[/file-code]

Where /file-code is the device from which the source module is to be read.
The last statement input must be :EOF, even if the input is from the console typewriter. The source module is placed in a temporary file (/S) on disk.

2.2.3. Example Input

The following coding could be input via any input device :



This code could also be input as follows :

```
DATAF\LDX\A4,4
\ABL(7)\HALT
DEVUN\LDX\A1,0\SET␣INDEX-REGISTER␣FOR␣BUFFER
\LDX\A3,/FF\LOGICAL␣CONSTANT␣IN␣A3
```

The listing device gives the listing in the same layout as the coding sheet. An example output listing is shown in paragraph 2.3.1.

Note : The IDENT statement cannot be input using the backslash between fields. It must be input with the correct spacing.

2.2.4. Assembly

The Assembler must be called by using the command :

ASM␣ { /S / name } [,NL]

where :

/S indicates that the source program must be read from the /S file

name indicates the name of a library source module or program to be assembled.

NL if specified, informs the Assembler that no listing is required of the assembled program. If NL is omitted a listing is produced on the printer. Error messages, however, will always be listed and the lines on which the errors occurred.

The Control Command Interpreter checks the ASM control command parameters for errors. When there is an error in the parameter, an error message indicating the error is printed, followed by the printing of S : at the beginning of the next line. The user may now input the correct ASM command. An example sequence could be :

S : ASM
FILE NAME MISSING
S : ASM /S

or

S : ASM <name>

The source module residing in the temporary file is now read and assembled.
Errors in the source program are detected by the Assembler. It is not possible to correct errors during processing. In case of a fatal error during processing e.g. table overflow, core overflow, IDENT missing, END missing, the source module is read until an : EOF mark is encountered. At that moment the following message is printed :

FATAL ERROR HAS OCCURRED NO OBJECT CODE PRODUCED

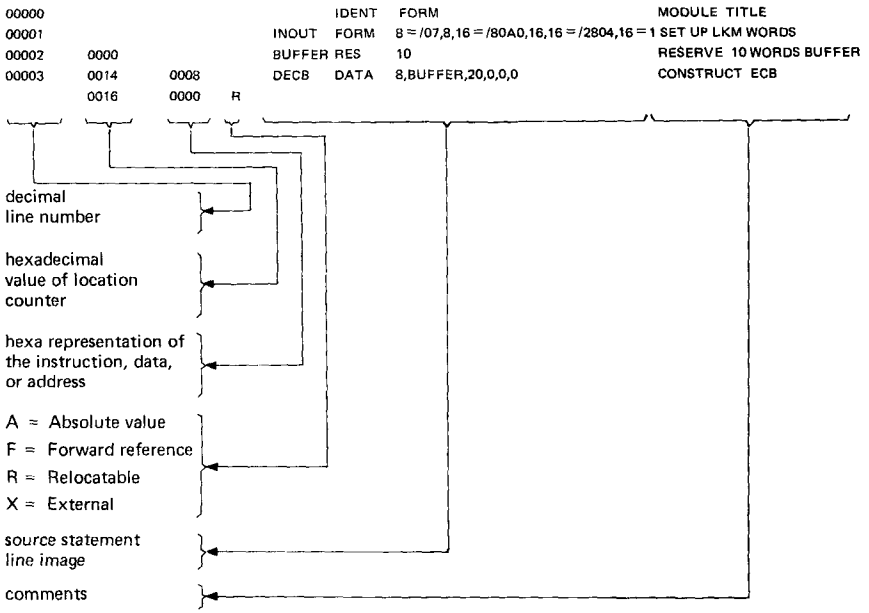
The object file, on which the output of the Assembler was written, is deleted.
If an I/O error occurs, processing is terminated immediately.

2.3. Output

The standard object output file for the Assembler is the temporary/O file. If this file does not yet exist an assignment is made for it. When it does exist the object output is written after the information already existing in this file unless it has been closed by an EOF record. In that case a new/O file is created and the old one is deleted.

2.3.1. Assembly Listing

The Assembly listing is output on the listing device if the NL option is omitted. The format of the printout is shown in the example below.



When a non-fatal error has occurred during assembly the processing continues but the place where the error occurred is indicated by an error code (see Section 2.3.3) following the line in which the error occurred. An asterisk is printed underneath the place where the error was detected.

An error counter is updated every time an error occurs. The number of errors is given after the printing of the symbol table, by :

ASS. ERR. 5 decimal digits (see example in Symbol Table, below).

2.3.2. Symbol Table

Each label which appeared in the label field of an instruction is given an address relative to the beginning of the module.
The symbol table consists of a list of labels for each name defined in or referred to within the module. This table is always printed out even if the user did not ask for a listing.
The symbol table has the following format :

label value type

where label is the name that appeared in the label field of an instruction,
value is either the address of the label relative to the beginning of the module or an absolute value (see type below),
type is the type of label. It can be one of four single digit codes : A, F, R, or X.

A = an absolute value. The label has been assigned a value with an EQUate directive.

F = a forward reference. The label is defined later in the module.

R = a relocatable reference.

X = an externally defined label (EXTRN)

The symbol table is printed with three labels across the width of the paper. The total number of errors encountered during Assembly is printed at the end of the symbol table. An example is shown below.

M:A00	0000	R	M:B00	005C	R	MPYMOD		X
ADDMOD		X	DSUMOD		X	SYSAB		X
M:A01	0006	R	T:SOPC	0136	R	M:A02	0018	R
M:A03	0010	R	T:SVR	0152	R	M:A04	0020	R
M:B01	008C	R	M:B00A	0074	R	M:B00B	0082	R
M:B01A	00A2	R	M:B02	00AA	R	M:B03	00CA	R
M:B04	00DC	R	M:B04A	00EA	R	M:B07	0100	R
ENDSVR	024E	R						

ASS. ERR. 00000

2.3.3. *Error Messages*

The following error messages can be output by the Assembler :

CODE	MEANING	DESCRIPTION
*C	Illegal constant	– "Constant" overflow – A constant with hexadecimal value must begin with either / or X. In the latter case, the value must be enclosed by quote marks, e.g. X 'ZF'. – A constant should not have been written here. – Hexadecimal constant written either X" or /".
*E	Not an even address	– The specified start address is not even. – The specified AORG or RORG operand is not even.
*F	Illegal FORM or XFORM directive	– An XFORM declared symbol must be linked to a FORM defined pseudo whose name is the first parameter of the operand. – More than 16 fields specified. – Negative field length. – The length of this field cannot be contained in 16 bits. It is too long.

CODE	MEANING	DESCRIPTION
		– A displacement value is not allowed when the predefinition concerns an external reference name. – The : predefinition is only allowed for a 16 bit field. – Invalid predefined value of a field (overdisplacement or negative value for a less than 16-bit field). – The division of the current word of an XFORM declaration is not the same as the corresponding word of the linked FORM symbol. – The predefinition of the fields of the current word of an XFORM declaration is not the same as the corresponding word of the linked FORM symbol. – More than 8 words described by a FORM declaration. – More than the number of words described by the linked FORM symbol described by an XFORM declaration. – The field number specified in the syntax definition line is invalid. – The same field specified twice in the syntax definition line.
*I	Illegal identifier	The first character of a symbol must be a letter.
*L	Illegal label	– The label has been defined previously as : – a symbol name – an external reference name – an entry point name – A label has been given where it was not allowed. – A label must be specified.
*M	Unknown mnemonic	– Unknown mnemonic – Unknown condition mnemonic
*O	Overdisplacement	– Displacement value of parameter too large.
*P	Illegal parameter	– Too many parameters specified in the operand of an instruction, in the pseudo-instruction or directive. – Not enough parameters specified in the operand of an instruction, defined pseudo-instruction or directive. – A parameter in the STAB directive must not be an entry point name, a COMMON name or a forward reference. – The operand in a DATA directive must not give more than 16 code words. – " is not a character string – Illegal use of a register name in a standard instruction operand.

CODE	MEANING	DESCRIPTION
*R	Illegal relocation	– Either a predefined expression or predefined relocatable section has been input. – Too many relocatable symbols are added to or subtracted from each other. – The expression is equal to the result of a subtraction of a relocatable part from an absolute part. – If an external reference is specified the displacement value must be absolute. – The instruction code operation defined by an EQU directive must be absolute.
*S	Illegal statement	– The ENTRY or EXTRN or COMN directive is no longer acceptable. – The directive does not need an operand. – The directive needs an operand. – Invalid character. – Invalid indirect addressing – Invalid condition specification. – The label is not followed by an operation code. – ' ('is not followed by')' – The operand value of a RES directive makes the instruction counter value negative. – GEN cannot produce any code as either any error occurred or the code word has already been produced. .
*X	Illegal expression	– More than two symbols defined – More than three terms in the expression. – An external reference and a forward reference have been specified in the same expression. – An external reference is preceded by a minus sign. – A plus or minus sign is not followed by a term. – A forward reference or external reference is specified in a requested predefined expression. – A register expression must not contain more than one term.
******O	Core overflow	– Fatal error. Too many symbols or forward references used.
******E	End missing	– Fatal error. The END statement is missing.
******I	IDENT missing	– Fatal error. The IDENT statement is missing.

2.3.4 *Saving the Assembled Module*

After a successful assembly the source module can be saved by using the command :

KPF␣ /S[, (file-name / module name)]

If (file-name / module name) is not given, the module is catalogued with the name given in its IDENT statement.

3. TOSS SYSTEM START

3.1. General

System start is the initialisation process which prepares a PTS 6000 Terminal Computer for application program running. It comprises the following steps :

- Load the TOSS Monitor into memory.
- Load the application program into memory.
- Load the application program into memory.
- Set up the required Monitor tables (optional Monitor configuration)
- Set up the required CREDIT tables (if a CREDIT program is being used (see the CREDIT Programmer's Reference Manual, M04)).
- Activate the application task. If there is more than one task to be activated, this must be done by the application itself (only the 1st application task, or DEBUG, is started by the Monitor).

3.2. System Start Procedure

The Monitor, Monitor configuration data, and application can be loaded from cassette, disk, or flexible disk. Loading can be controlled from the full panel (if present) or from the SOP only. All the relevant procedures are described below.

3.2.1. From Cassette

It is possible to load the system from one, or two cassettes depending upon which option was taken during system Generation (see $PCAS, DOS 6810 System Software PRM–M11). The first option, everything on one cassette, is :

- Monitor
- Application
- [• Monitor configuration data]

The second option, two cassettes, is :

- Monitor and application on the first cassette
- Monitor configuration data on the second cassette

The generation of the cassettes is described briefly in section 3.3 but a detailed description is given in DOS 6810 System Software PRM, M11.
The procedure for loading the system is :

1. Ensure that the power is switched on at the Terminal Computer and at each peripheral device.
2. Insert the Monitor cassette in a cassette drive.

3a. To load from the full panel, press the following buttons in sequence :

   1) Reset (RST)
   2) Master Clear (MC)
   3) Initial Program Load (IPL)

3b. To load from the SOP only, press the SOP IPL Switch.

4. Select the cassette drive containing the Monitor cassette by pressing SOP switch 1 (for the left-hand cassette drive) or SOP switch 2 (for right-hand cassette drive).
5. The Monitor and application program will now be read into memory.
6. If a second cassette containing the Monitor configuration data is to be loaded, mount this cassette in a cassette drive and press the appropriate SOP switch (1 or 2) when the first cassette has been read.
   The Monitor configuration data will then be read into memory and the required Monitor tables will be set up.
7. The application will now be executed unless DEBUG has been included. DEBUG has a higher priority than the application program (because it is part of Monitor) and it will stop as soon as it is loaded, waiting for the operator's command (see Chapter 4).

During loading, lamps 1, 2 or 3 may light up :

Lamp 1 indicates that an application is being loaded
Lamp 2 indicates an input error
Lamp 3 indicates that the application is too large for the memory

3.2.2. *From disk*

The Monitor and application must be on the same disk, either cartridge or fixed disk.
The procedure for loading the system is :

1. Ensure that the power is switched on at the Terminal Computer and at each peripheral device.
2. Insert the disk on a disk drive, press the START button and wait for the READY lamp to light up.
   If loading is to be done from the fixed disk, a disk cartridge must still be loaded.

3a. To load from the full panel, press the following buttons in sequence :

   1) Reset (RST)
   2) Master Clear (MC)
   3) Initial Program Load (IPL)

3b. To load from the SOP only, press the SOP IPL switch.

4. Select the disk containing the Monitor by pressing SOP switch 3 (for the cartridge disk) or SOP switch 4 (for the fixed disk) Monitor will then be loaded.
5. If there is only one application program on the disk, lamp 1 will light up and the application will be loaded automatically.
6. If there is more than one application program on the disk, all the lamps will light up. Choose the application program to be loaded by pressing the appropriate SOP switch.
7. The application will now be executed unless DEBUG has been included. DEBUG has a higher priority than the application program (because it is part of Monitor) and it will stop as soon as it is loaded, waiting for the operator's command (see Chapter 4).

During loading, lamp 1, 2 or 3 may light up :

Lamp 1 indicates that an application is being loaded
Lamp 2 indicates that the application chosen by the SOP switch does not exist.
Lamp 3 indicates that the application is too large for the memory.

3.2.3. *From Flexible disk*

The Monitor and application program must be present on the same flexible disk. The disk is produced by using $PFLEX during System Generation (see $PFLEX DOS 6810 System Software PRM M11).
The procedure for loading the system is :

1. Ensure that the power is switched on at the Terminal Computer and at each peripheral device.
2. Put the diskette into a drive and shut the door.

3a. To load from the full panel, press the following buttons in sequence :

   1) Reset (RST)
   2) Master Clear (MC)
   3) Initial Program Load (IPL)

3b. To load from the SOP only, press the SOP IPL switch.

4. Select the flexible disk drive containing the Monitor and application by pressing SOP Switch 5, 6, 7 or 8.

switch 5 = flexible disk 1, multiplex channel
switch 6 = flexible disk 2, multiplex channel
switch 7 = flexible disk 1, programmed channel
switch 8 = flexible disk 2, programmed channel

Monitor will then be loaded.

5. If there is only one application on the disk, lamp 1 will hight up and the application will be loaded automatically.

6. If there is more than one application on the disk, all the lamps will light up. Choose the application program to be loaded by pressing the appropriate SOP switch.

7. The application will now be executed unless DEBUG has been included. DEBUG has a higher priority than the application program (because it is part of Monitor) and it will stop as soon as it is loaded, waiting for the operator's command (see Chapter 4).

During loading, lamp 1, 2 or 3 may light up :

Lamp 1 indicates that an application is being loaded.
Lamp 2 indicates that the application chosen by the SOP switch does not exist.
Lamp 3 indicates that the application is too large for the memory.

3.3. Deferred binding of Monitor Configuration Data

3.3.1. General

The TOSS for a particular PTS 6000 Terminal System must be generated by the utility SYSGEN. This utility generates the TOSS Monitor on disk. The Monitor can be copied to cassette or flexible disk for the system start procedure. This copying is done by the catalogued procedure $PCAS (for cassette) or $PFLEX (for flexible disk).
If the programmer wants to test his program with Monitor configuration data supplied at run time (deferred binding) he can exclude the Monitor configuration data from the Monitor cassette at system generation time. The Monitor will be generated without configuration data but with the Monitor Configuration Program (MONCON) which will read the configuration data at run time. In this case the Monitor and application program must be on the same cassette. The instructions to generate a Monitor configuration data cassette are given below in paragraph 3.
MONCON will read the Monitor configuration data from the cassette and generate Monitor tables at the end of memory (higher addresses).
Only the drivers and tables which are included at system generation time may be configured by MONCON.
The use of the Linkage Editor, $PCAS, and $PFLEX is described in the DOS 6810 System Software PRM (M11).

3.3.2. Monitor Configuration Cassette

When the Monitor and application load module have been read into memory, control is passed to MONCON.
This program will read the Monitor configuration data and generate the necessary Monitor tables related to the hardware configuration.
To enable MONCON to set up the correct Monitor tables, the following data must be supplied as Monitor configuration data on the cassette :

- Terminal class identifier
- Number of work positions in the terminal class
- Priority level of the task
- Terminal device class
- Connection on the local/remote channel unit
- Special device classes

3.3.2.1. Generating the Monitor Configuration Cassette

The cassette containing Monitor configuration data is generated at the console typewriter with the aid of DOS 6810 control commands.
The procedure for generating a Monitor configuration data cassette is as follows.

First insert a cassette in one of the cassette drives, then key in the following sequence.

(i) ASG␣/E1, TY10
(ii) REW␣/03
(iii) RDA␣/0A
(iv) Monitor configuration data
(v) WEF␣/03
(vi) PCH␣/0A
(vii) WEF␣/03
(viii) REW␣3
(ix) ULD␣3

Explanation :

(i) Assign file code/E1 (source input) to the console typewriter.
(ii) Rewind the left-hand cassette drive.
(iii) Read data from the source input file (typewriter) and transfer to temporary disk file /A.
(iv) The format of the Monitor configuration data is described below.
(v) Write an end of file mark on the cassette.
(vi) Write the contents of temporary disk file /A to the cassette, an end of file mark is written automatically.
(vii) Write one end of file mark on the cassette.
(viii) Rewind the cassette.
(ix) Unload the cassette.

3.3.2.2. Format of Monitor Configuration Data

The Monitor configuration data must be keyed-in in the following sequence :

- Terminal class identifier (2 characters) followed by the number of work positions in the terminal class.
- Priority level
- Terminal device class followed by the connection on the local/remote channel unit.
- Special device class.
- EN End of task definition

a. The terminal class identifier and number of work positions have the following format :

terminal class identifier : decimal-integer CR LF

< terminal class identifier > is specified in the main module of the application program.
< decimal–integer> is the number of work positions in this terminal class.

b. Priority level has the following format :

decimal–integer CR LF

< decimal–integer > consists of two digits specifying the priority level of the task, which is normally 60.

A number lower than 60, means that the task has higher priority than the one with level 60.

c. Terminal device class and connection have the following format :

T decimal-digit, decimal-digit L CR LF
(local channel unit)

T decimal-digit, decimal-digit R CR LF
(remote channel unit)

< T decimal-digit >, is the terminal device class which is specified during system generation.

< decimal-digit L > or < decimal-digit R > is the number of the connector on the local or remote channel unit, respectively to which the work position is connected.

d. Special device class has the following format :

S decimal-digit CR LF

< S decimal-digit > is the special device class which is specified during system generation.

e. EN CR LF terminates the parameters for the terminal class concerned and the same parameters as mentioned above may be specified now for another terminal class.
When closing with ENEN, the data that follows will be considered as common device information.
An example of Monitor configuration data is shown below for a configuration which consists of 3 work positions configured in the same way and one special work position with a different configuration.
Common devices are used by both terminal classes.
The first terminal class is T0 and the second is F0.
Monitor configuration data :

```
T0 : 3    (Three tasks with identifiers T0, T1 and T2 will be generated)
60
T1, 1L    (Terminal device class 1 on channel unit line 1, 2, 3)
T2, 1L    (Terminal device class 2 on channel unit line 1, 2, 3)
S1        (Special device class)
EN
F0
T3, 8L
EN
EN
S3        Common devices
S4
EN
```

3.3.2.3. *Errors During Configuration*

During Monitor configuration any errors will be indicated on the System Operator's Panel :

Lamp 1 — Monitor configuration data reading
Lamp 2 — Cassette input error
Lamp 3 — Format error
Lamp 4 — Memory overflow

4. ASSEMBLER DEBUGGING PROGRAM

During the production of a program, the programmer requires a method investigating errors that may have occurred. The Assembly Debugging Program (DEBUG) is provided to assist the programmer in monitoring and controlling the execution of this program in order to find errors.

This chapter contains :

- a description of the Assembly Debugging Program
- definitions of the control parameters
- instructions for running the program
- interpretations of the result.

4.1. Introduction

The Assembly Debugging Program (DEBUG) is an interactive diagnostic task which runs under the control of the TOSS Monitor. It runs in parallel with the Assembly application program being tested.

DEBUG may be used to control the execution of the application program in the following ways :

- the contents of registers or memory may be examined or modified;
- the application program may be stopped by inserting breakpoints at selected places, and started again;
- parts of the program may be conditionally executed, once or a number of times (looping);
- calculations on addresses may be performed.

DEBUG operates in either one of two modes :

- Debug mode – the programmer has control over the execution of the user program;
- User mode – the user program is executed normally.

Readers of Chapter 4 should be familiar with the following DOS 6810 System Software concepts :

- linkage editor
- control command
- user library
- TOSS system generation

These concepts are explained in the DOS 6810 System Software PRM (M11).

Before using DEBUG, the specialist reader should read Appendix A, Memory Organization.

4.2. Using DEBUG

DEBUG is specified at SYSGEN time and is part of the TOSS Monitor.
If DEBUG is not required (e.g. for production versions of the application program) it must be excluded at SYSGEN time.

DEBUG runs on any level greater than or equal to 8 and always on a higher level than the application program being tested. The default value is 50 (/32). All Monitor tasks, except data management can use DEBUG. Control is handed to DEBUG immediately after the TOSS Monitor and application program are loaded into memory (system start).

The User communicates with DEBUG via the system operator's console (CTW) using the commands described in Section 4.

DEBUG maintains two sets of registers-relocation registers for addressing purposes and "pseudo registers" for processing purposes.
The use of relocation registers is described in Section 4.3.3.

The pseudo registers correspond to CPU registers P, A1 to A15 (inclusive).

These are loaded when DEBUG is entered from either a halt command or a trap.

The user may then examine and modify these contents, and before the user program is restarted, the pseudo registers are copied into the real CPU registers. From the user's point of view these are indistinguishable from the real registers.

4.2.1. Breakpoints

The user can stop his program at a specific point by a command to DEBUG. The point at which the program stops is called a 'breakpoint'.

DEBUG replaces the instruction at the breakpoint with an illegal code, causing a call to the trap call interpreter (/6000). When the program reaches the call, control is given to DEBUG. The user may then examine and modify registers and memory before continuing.

DEBUG also provides the possibility to loop through a breakpoint a specified number of times and to execute breakpoints conditionally depending on register or memory values. This can be useful when debugging program, loops etc.

It is possible to maintain up to sixteen breakpoints simultaneously.
However, only one may be looped or executed conditionally at a time.
This is called the 'active breakpoint'. The last breakpoint given or looped on is active.

When looping, DEBUG uses an internal register called the loop counter.

4.2.2. DEBUG Start

During execution, DEBUG is entered after a breakpoint, illegal instruction, or if the user issues a halt command from the console.

When DEBUG is started, relocation register E contains the first address of the application program.

DEBUG can be restarted from address /92.

4.3. DEBUG Input

4.3.1. General

DEBUG provides the programmer with a variety of commands to control the testing process. The commands are briefly :

Command	Mnemonic
calculate	=
open memory word	/
go	G
halt	H
interrupt	I
loop	L
print memory location	M
proceed from breakpoint	P
print location registers	Q
print CPU registers	R
open program status word	S
set breakpoint	T
verification	V
remove breakpoint	Y

These commands are described in full below :

4.3.2. DEBUG Modes

DEBUG operates in one of two modes known as B (debugging) mode, and U (user) mode. In B mode DEBUG is running and waiting for an operator's command. B mode is selected whenever the application program stops. The mode is selected by the system

A stop occurs when :

- a halt (H) command is keyed in
- a breakpoint is encountered in the application program
- a verification halt condition is encountered
- an illegal operation code is detected.

U mode is selected when one of the following commands is keyed in :

- proceed from breakpoint (P)
- loop through breakpoint (L)
- Go (G)

Commands other than H, P, L and G will not result in a change of mode .
The current mode is indicated by the letter B or U printed at the left of each line of output.
Immediately after system start DEBUG is in B mode.

4.3.3. Relocation Registers

DEBUG maintains 16 relocation registers. These registers are numbered 0 to F and may be used in commands as indexes when refering to memory locations. The contents of relocation registers may be examined and modified using the Q command.

The following description illustrates the way in which the relocation registers are normally used. The start address of the module currently being tested is loaded into a relocation register. Commands then refer to locations within this module by quoting the address relative to the start of the module (listed by the assembler as the second field from the left) together with the number of the relocation register.

Relocation registers may also be used to save constants.

When starting an application program, relocation register E contains the LOAD address of the program – 8. Register F contains – 8 which can be used to obtain module start addresses from the Linkage Editor Map.

4.3.4. Addressing

The following commands contain references to memory addresses :

G I L M P Q R S T Y = , / .

Relative, absolute or indirect addresses may be used in these commands.
All address values are given as hexadecimal numbers.

The relative address is calculated as a displacement from a named relocation register. The address in the relocation register could be the start address of the module or some other reference address (the start of an array for example). The relative address from the start of the module is shown in the second field from the left on the Assembler listing. The relocation register may also contain an absolute address.

The absolute address is that which is given in the program counter and is the displacement from word zero of memory. If the address refers to the start of an array, a displacement value must be given to specify the word within the array (counting the first word as 1).

Indirect addresses are used to reference :

- CPU registers (R)
- control registers (I)
- relocation registers (Q)
- memory words (=, I, V)

If an address is indirect it must be prefixed by an asterisk (e.g. * 10). In this case the indirect address will point to a memory word that contains the absolute address to be referenced. If the resulting address is odd, the next lower even address will be used.

4.3.5. Command Syntax

Commands are keyed in immediately after the B or U prompt which is printed at the left of each line by DEBUG.

Commands have the following Syntax :

$$\left\{\begin{matrix} B \\ U \end{matrix}\right\} [\ \text{parameter 1}\ ][\ ;\text{parameter 2}\ ]\ \text{command}$$

"B" or "U" is printed by the system at the beginning of the line. "Command" is one of the single character commands listed above in section 4.3.1.

$$\text{parameter} ::= \left\{\begin{matrix} \text{TERM} \\ \text{parameter}\ (\pm)\ \text{parameter} \end{matrix}\right\}$$

$$\text{TERM}::= \left\{\begin{matrix} \text{hexadecimal integer} \\ *\ \text{term} \\ \text{term}\ R \\ \text{term}\ Q \\ \bullet\ \text{term} \end{matrix}\right\}$$

where :– "hexadecimal integer" is a four digit hexadecimal number

"* term" is an address of a memory word that contains the address of the required information.

"term R" is the sum of "term" and the contents of CPU register R.

"term Q" is the sum of "term" and the contents of relocation register Q.

"• term" is a register that contains the address of the required information.

An open and modify command has the following syntax :

$$\left\{\begin{matrix} B \\ U \end{matrix}\right\}\ /nnnn\ [\ \text{parameter}\ ]\ C$$

where "/nnnn" is the content that "parameter" optionally replaces,

"C" is a terminating character which can be :

CR	modify and close
LF	modify and open the variable at the next higher address
@	modify the variable and use the new value as the address of the next variable to be opened (i.e. indirect addressing) This next variable is automatically opened.

Any other character will result in the current variable being closed without modification and without the next variable being opened.
The description of each command will specify which terminating characters can be used.

Notes : An empty or non-existent parameter may have a special meaning to the command, see the description of each command.

Parameters are left-shifted so that only the last four character keyed in are taken as significant, the others are ignored. If an undefined command is keyed in, a question mark is printed and no further action is taken. If an illegal command is keyed in, it is rejected and 'NO' is typed out; no further action is taken.

4.3.6. *Calculate Command (=)*

Format : parameter =

Description : "Parameter" is calculated and displayed. This command is useful for calculating hexadecimal addresses. "Parameter" can consist of a string of terms because the calculation is performed as each term is input.
The result is printed after the = has been keyed in.

Examples :
```
U 262A+2A4 = 28CE
U 23+45+45 = 00AD
  |   parameter  |  result
mode          command
```

4.3.7. *Open Memory Word Command (/)*

Format 1 : parameter /

Description : Address "parameter" is opened for modification.

Format 2 : /

Description : The last used address is re-opened.

Example :
```
B      1Q/ 262A
mode    |  |  result
        |  command
        parameter
```

4.3.8 *Go Command (G)*

Format 1 : parameter G

Description : Start user program at "parameter".

Format 2 : G

Description : Start user program at the address in the program counter.

Example :
```
B      0,1 G
mode    |  command
        parameter
```

4.3.9. *Halt Command (H)*

Format : H

Description : The application program is stopped.

4.3.10. *Open Interrupt Control Register (I)*

Format : parameter I

Description : Open interrupt control register specified by "parameter".
The registers are defined in the table below :

Code	register
0	Priority level of DEBUG is displayed
1	DEBUG interrupt mode is displayed 0 = inhibit mode (INM) non-zero = enable mode (ENB)
2	DEBUG interrupt control, 0 = None
3	Real time clock control, 0 = Off

Examples :

```
B          0!/0032
|          |  \
mode       |   result
           |
           command
           |
           parameter
```

```
B          1!/0000
|          |  \
mode       |   result
           |
           command
           |
           parameter
```

```
U          2!/0001
|          |  \
mode       |   result
           |
           command
           |
           parameter
```

```
U          3!/FFFF
|          |  \
mode       |   result
           |
           command
           |
           parameter
```

4.3.11. Loop (L)

Format 1 : *parameter* L

Description : Loops "parameter" times through the current breakpoint.
The program will run without stopping at that breakpoint until it comes to it (parameter + 1) times.

Format 2 : L

Description : Loops only once through the current breakpoint.

Example :

```
B          5 L
|          |
mode       command
           |
           parameter
```

4.3.12. *Print Memory Location Command (M)*

Format 1 : parameter 1; parameter 2 M

Description : Prints all memory locations from "parameter 1" to "parameter 2" (inclusive).

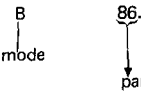
Format 2 : parameter M

Description : Prints eight words from "parameter".

Format 3 : M

Description : Prints eight words from the last address specified.

Examples :



The result is shown below

```
0086    0000  0236  23DC  FFFE  0000  5710  5714  1CAC    ...6#.....W.W...
0096    2276  01F7  0000  00F1  0000  0000  8120  0246    "............. .F
00A6    5710  85A0  0090  86A0  FFF8  96C0  0094  8120    W..............
```

4.3.13. *Proceed from Breakpoint (P)*

Format 1 : parameter 1; parameter 2 P

Description : Removes the current breakpoint, sets a new breakpoint "parameter 1", proceeds, and loops "parameter 2" times through the new breakpoint.

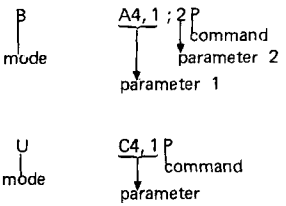
Format 2 : parameter P

Description : Sets a new breakpoint "parameter" and proceeds from there.

Format 3 : P

Description : Removes current breakpoint and proceeds.

Examples :



4.3.14. Print Relocation Registers (Q)

Format 1 : parameter Q

Description : Opens and prints the content of relocation register Q.

Format 2 : Q

Description : Prints all relocation registers

Examples :

```
B          1Q/262A
|            result
mode       command
           |
           parameter
```

```
B          Q
|          |
mode       command
```

The result is shown below.

```
0000  0000  0000  0000  0000  0000  0000  0000
0000  0000  0000  0334  0000  0000  1CA4  FFF8
```

4.3.15 Print CPU Registers (R)

Format : parameter R

Description : Opens and prints the content of the CPU register "parameter"

OR = program counter.

Format 2 : R

Description : Prints all CPU registers.

Examples :

```
B          2R/25A3
|            result
mode       command
           |
           parameter
```

```
U          R
|          |
mode       command
```

The result is shown below :

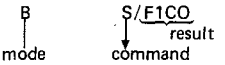
```
0238  10EA  8F20  873F  85BF  023A  85A0  813F
5002  0A10  1E80  813F  0100  0216  87A0  0236
```

4.3.16. Open Program Status Word (S)

Format : S

Description : Opens and prints the program status word (PSW). For a description of the PSW see Appendix A.

Examples :



4.3.17. Set Breakpoint (T)

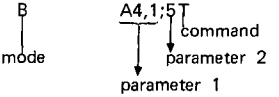
Format 1 : parameter 1; parameter 2 T

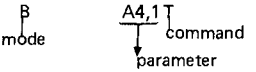
Description : Sets a breakpoint (i.e. a subroutine call) at the address specified by "parameter 1" and the loop counter to a value "parameter 2".

Format 2 : parameter T

Description : Sets a breakpoint at address specified by "parameter" and the loop counter to zero.

Examples :





4.3.18 Verification (V)

Format 1 : parameter 1; parameter 2 V

Description : Stops at the next breakpoint encountered when the condition specified by xx is true, where xx can be :

RE	(register parameter 1) is equal to parameter 2.
RN	(register parameter 1) is not equal to parameter 2.
ME	(address parameter 1) is equal to parameter 2.
MN	(address parameter 1) is not equal to parameter 2.

Before the comparison is made, parameter 1 is masked with the verification mask register. If register 0 is specified the Program Status Word is used.

For the contents and use of memory, see Appendix A, Memory Organization.

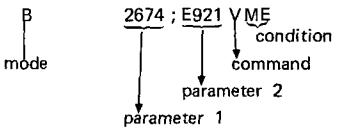
Format 1 : OV

Description : Open verification mask register

Format 2 : V

Description : Turn off verification

Example : B 2674 ; E921 V ME



4.3.19. Remove Breakpoint (Y)

Format 1 : parameter Y

Description : Removes the breakpoint at address "parameter"

Format 2 : Y

Description : Removes all breakpoints

4.4. Running DEBUG

4.4.1. DEBUG Output

Each time DEBUG is entered, the following message is printed :

C – PC = LC, REL – PSW

where

C is used to indicate the way DEBUG was entered.
It may have the following values :

C	Meaning
S	start or restart
T	breakpoint
I	illegal instruction
H	halt command

PC — program counter is the absolute address where the program was interrupted

LC, REL — is the relative address LC is indexed by the relocation register REL. The REL with the nearest corresponding LC value is chosen.

PSW — CPU program status word at interrupt, see Appendix A

Output from the commands is shown in the paragraph describing each command in Section 4.3.

4.4.2. Loading Segments for Debugging

It is not possible to set a trap in a segment until it has been loaded. This means that it is necessary to set the trap in the load task just after the segment has been loaded, then find out in what partition the segment is placed and set a trap in the appropriate position. At the end of the load task an exit is made (LKM, DATA 3). The LKM is a useable breakpoint.

It is also possible to use the verify function (see Section 4.3.18) after setting a call in the load task. The segment pointer (address in the LSET to the segment) is found in the register A3 at exit of the load task.

4.4.3. Interrupt Control

When DEBUG is entered from a breakpoint, it is set to run at level 50 (= /32), in the mode ENB. If in INH mode, it is set to the same as the interrupted user program.

This is, in most cases, just what the user wants. INH sequences will not be interrupted, Monitor tasks will interrupt DEBUG etc.

However, there might be reasons to have DEBUG working in another way. For example:, when debugging synchronous devices, interrupt routines in ENB mode, time out functions, etc.

Therefore DEBUG has the following features :

for commands see Section 4.3.

1. The level may be changed. Since the console uses level 7, the level should not be set to less than 8.
2. The interrupt mode may be changed to either ENB or INH.
3. The function that DEBUG changes mode at breakpoints to the user mode may be excluded.
4. The real time clock may be handled by DEBUG.

4.4.3.1. Interrupt Button

Pressing the interrupt button (INT) on the full panel forces DEBUG immediately to command mode. This may be used to stop a long memory dump or to get the console on-line after pressing master clear (MC)–

The INT button is only available on computers with a full panel.

4.4.3.2. Power Fail

The power on/off interrupt is just switched through DEBUG after having set the console on-line.

4.4.3.3. Real Time Clock

The real time clock may be operated in two modes :

1. The RTC interrupt is switched through DEBUG;
2. DEBUG handles the RTC interrupt

Mode 1 is default.

APPENDIX A : MEMORY ORGANIZATION

The information in this Appendix will not normally be required by most Assemble Programmer's but has been included for specialist programmers.

A.1 Memory Layout

AREA	ADDRESS
Hardware Interrupt Locations	/0 /7C
Pointer to Subroutine Call Interpreter	/7E
System Halts	/80 /84
Communication Vector Table	/86 /98
Stack Overflow Area	/9A /100
Stack	variable
Monitor	variable
User Program Area	variable end of memory

Figure A1 Memory Layout

- Locations /0 to /7C are hardware interrupt locations. They are hard-wired to internal and external lines. Each location contains the address of the interrupt routine required to service the interrupt connected to that location. The interrupt connected to location /0 has the highest priority (level 0).
- Location /7E contains the address of the subroutine call interpreter which handles simulation of certain instructions not included in the hardware.
- Locations /80 to /84 contain the system halts.
- Locations /86–/98 are the Communication Vector Table. This is a System table. This table has the following layout :

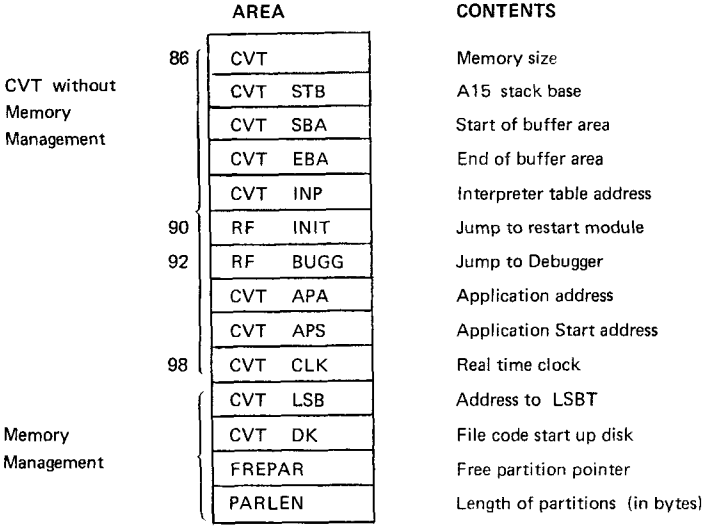


Figure A2 Communication Vector Table

Notes

1. If Memory Management is used, the CVT extends into the stack overflow area.
2. CVT SBA holds the address after the root.
3. CVT APA is the address of the beginning of the root.

– The area occupied by the stack is defined at system generation time. When an interrupt occurs, P-register, PSW and a number of registers are stored here. The number of registers stored depends on whether the interrupt routine servicing the interrupt runs in inhibit mode (anywhere from 0 to 15 registers) or in enable mode and branches to the dispatcher (always 8 registers).

  The A15 register always points to the next free location in the stack (where all information is stored towards the lower memory address).
  When A15 reaches the value /100 or becomes lower a stack overflow interrupt is given.

– The area after the Monitor area is the user area.

A.2. Interrupt System

When working in interrupt mode each interrupt program may be connected to an interrupt level. As the actioning of an interrupt involves the direct accessing of the interrupt level's start address from its hardware interrupt location, the contents of this location must have been previously loaded with the correct address.

The start addresses loaded in these locations are not fixed and must be defined by the programmer at SYSGEN time.

interrupt level	hardware interrupt location
0 to 63	/0000 to /007C

where level 0 has the highest priority and 63 the lowest. The levels are defined at SYSGEN time.

PROGRAM STATUS WORD

The program status word (PSW) contains data relating to the status of that program. It is maintained by the System Software.

These fields can be modified by the user	0 — 5	priority level
	6 — 7	condition register
These fields cannot be modified by the user except during Debugging	8	run indicator
	9	interrupt enabled indicator
	10	control panel interrupt
	11	power failure
	12	real-time clock (RTC)
	14, 15	not used