PERKIN-ELMER

OS/32 SYSTEM SUPPORT
RUN-TIME LIBRARY (RTL)

Reference Manual

48-152 F00 R00

The Perkin-Elmer Corporation, Data Systems Group, 2 Crescent Place, Oceanport, New Jersey 07757

Printed in the United States of America

TABLE OF CONTENTS

PREFACE v

CHAPTERS

1 INTRINSIC ROUTINES

1.1 INTRODUCTION 1-1

1.2 GOAPU Subroutine 1-1

1.3 GOCPU Subroutine 1-2

1.4 GENSIG Subroutine 1-3

1.5 IRTCNT Function 1-4

1.6 RSCHDL Subroutine 1-5

2 RUN-TIME LIBRARY (RTL) SUBROUTINES

2.1 INTRODUCTION 2-1

2.2 APUDABL Subroutine 2-2

2.3 APUENAB Subroutine 2-2

2.4 ASGNQUE Subroutine 2-3

2.5 ASLPU Subroutine 2-4

2.6 CNTLAPU Subroutine 2-5

2.7 HALTAPU Subroutine 2-6

2.8 IDAPU Function 2-7

2.9 LPUPREF Subroutine 2-7

2.10 LPUQUE Subroutine 2-8

2.11 MAPQUE Subroutine 2-9

CHAPTERS (Continued)

2.12	MAPTABL Subroutine	2-10
2.13	QUEOFF Subroutine	2-12
2.14	QUEON Subroutine	2-12
2.15	QUEXON Subroutine	2-13
2.16	RELCNTL Subroutine	2-14
2.17	RELMAP Subroutine	2-15
2.18	REMQUE Subroutine	2-16
2.19	RSCHAPU Subroutine	2-17
2.20	SETQUED Subroutine	2-18
2.21	STAQUE Subroutine	2-19
2.22	STATAPU Subroutine	2-22
2.23	STOPAPU Subroutine	2-24
2.24	STRTAPU Subroutine	2-25
2.25	TRLPU Subroutine	2-26

FIGURES

2-1	Format of Elements in the Array BUFFER Used by the MAPTABL Subroutine	2-11
2-2	Layout of Information Returned by STAQUE to BUFFER	2-21
2-3	Layout of Information Returned by STATAPU to BUFFER	2-23

TABLES

1-1	STATUS CODES FOR THE GENSIG SUBROUTINE	1-4
1-2	STATUS CODES FOR THE IRTCNT FUNCTION	1-4
2-1	STATUS CODES FOR THE APUDABL SUBROUTINE	2-2
2-2	STATUS CODES FOR THE APUENAB SUBROUTINE	2-3
2-3	STATUS CODES FOR THE ASGNQUE SUBROUTINE	2-4
2-4	STATUS CODES FOR THE ASLPU SUBROUTINE	2-5
2-5	STATUS CODES FOR THE CNTLAPU SUBROUTINE	2-6

TABLES (Continued)

2-6	STATUS CODES FOR THE HALTAPU SUBROUTINE	2-7
2-7	STATUS CODES FOR THE LPUPREF SUBROUTINE	2-8
2-8	STATUS CODES FOR THE LPUQUE SUBROUTINE	2-9
2-9	STATUS CODES FOR THE MAPQUE SUBROUTINE	2-10
2-10	STATUS CODES FOR THE MAPTABL SUBROUTINE	2-11
2-11	STATUS CODES FOR THE QUEOFF SUBROUTINE	2-12
2-12	STATUS CODES FOR THE QUEON SUBROUTINE	2-13
2-13	STATUS CODES FOR THE QUEXON SUBROUTINE	2-14
2-14	STATUS CODES FOR THE RELCNTL SUBROUTINE	2-15
2-15	STATUS CODES FOR THE RELMAP SUBROUTINE	2-16
2-16	STATUS CODES FOR THE REMQUE SUBROUTINE	2-17
2-17	STATUS CODES FOR THE RSCHAPU SUBROUTINE	2-18
2-18	STATUS CODES FOR THE SETQUED SUBROUTINE	2-19
2-19	STATUS CODES FOR THE STAQUE SUBROUTINE	2-22
2-20	STATUS CODES FOR THE STATAPU SUBROUTINE	2-24
2-21	STATUS CODES FOR THE STOPAPU SUBROUTINE	2-25
2-22	STATUS CODES FOR THE STRTAPU SUBROUTINE	2-26
2-23	STATUS CODES FOR THE TRLPU SUBROUTINE	2-27

INDEX IND-1

PREFACE

This manual details the OS/32 run-time library (RTL) subroutines and intrinsic routines for the Model 3200MPS System. The Model 3200MPS System features a FORTRAN interface that facilitates the manipulation of system resources. Through applications tasks written in Perkin-Elmer FORTRAN VII language, the RTLs documented in this manual can be used to control these extensive system resources. The capabilities available to application tasks on the Model 3200MPS System are comprehensive. For details on the features of these RTLs and their uses, see the Model 3200MPS System Overview Manual.

Chapter 1 of this manual consists of FORTRAN RTL intrinsic routines designed to measure real-time for the Model 3200MPS System and to allow for task self-control. Chapter 2 consists of FORTRAN VII RTL subroutines that provide for multiprocessor system support.

The information in this manual is applicable to OS/32 R07.2 software release or higher.

For information on the contents of all Perkin-Elmer 32-bit manuals, see the 32-Bit Systems User Documentation Summary.

CHAPTER 1
INTRINSIC ROUTINES

1.1 INTRODUCTION

The following intrinsic routines are recognized by the FORTRAN VII compilers. See the Model 3200MPS System Overview Manual for application examples of the uses of these routines.

WARNING

THE RUN-TIME LIBRARY (RTL) SUBROUTINES FOR THE MODEL 3200MPS SYSTEM, AS DOCUMENTED IN THE FOLLOWING SECTIONS, ARE APPLICABLE ONLY TO USERS OF OS/32 R07.2 SOFTWARE RELEASE OR HIGHER. USERS OF A MODEL 3200MPS SYSTEM RUNNING UNDER OS/32 R07.1 SOFTWARE RELEASE MUST CONSULT THE FORTRAN VII USERS GUIDE FOR DOCUMENTATION OF MODEL 3200MPS RTL SUBROUTINES AND INTRINSICS.

The FORTRAN O and Z compilers translate all calls to these intrinsic routines into their respective instruction sequences. Except when stated otherwise, all of their arguments are integer*4 scalars.

1.2 GOAPU Subroutine

The GOAPU subroutine enables the logical processing unit (LPU) assignment. If the eligibility requirements are satisfied, the calling task is placed on the auxiliary processing unit (APU) execution queue. This queue is specified in the logical processor mapping table (LPMT). Call GOAPU as follows:

Format:

```
CALL GOAPU([APU_ID])
```

Argument:

APU_ID — is an optional integer*4 variable to which GOAPU returns a value as follows:

0	indicates the task is running on the central processing unit (CPU) after invocation of this subroutine.
APU Number	indicates the number on which the task is running after invocation of this subroutine.
-1	indicates that the APU_ID of the processor on which the task is currently running cannot be accessed. This value should only occur in the event of hardware failure.

Functional Detail:

In order to transfer the calling task to an APU, an LPU must be assigned to this task and the APU mapped to the assigned queue.

NOTE

If the calling task is mapped to an execution queue with multiple processors, APU_ID should be used with caution as the task may move to other processors after the GOAPU subroutine is called.

1.3 GOCPU Subroutine

The GOCPU subroutine transfers the calling task to the CPU.

Format:

```
CALL GOCPU
```

1.4 GENSIG Subroutine

The GENSIG subroutine is used to instruct an APU to pulse a specified line of a real-time support module (RTSM). The number of the RTSM line is specified in the input argument. For more information on the RTSM, see the Model 3200MPS Real-Time Support Module (RTSM) Programming Reference Manual.

Format:

```
CALL GENSIG(APU_NO,SIGNAL_NO,APU_ID [,STATUS])
```

Arguments:

APU_NO	is an integer*4 input argument that specifies the APU number.
SIGNAL_NO	is an integer*4 input argument that specifies the output line number.
APU_ID	is an integer*4 output argument in which GENSIG loads the APU_ID number of the processor executing the calling task.
STATUS	is an optional integer*4 variable to which GENSIG sends a status code (see Table 1-1 for possible status codes).

Functional Details:

The hard-wired ID number of the processor (on which the task is running) is read. If this number cannot be determined, then a value of -1 is returned in the APU_ID. If the APU_NO is negative, the signal is sent to the selected pulse line and the APU identification is returned in the APU_ID. If the APU_NO is not negative, the APU_NO is compared with the hard-wired ID number of the processor on which the task is running. If this number does not match the APU_NO (when executing on the APU), a status code of 3 is returned in STATUS and the selected output line is not pulsed (see Table 1-1 for possible status codes), otherwise, the selected output line is pulsed. The SIGNAL_NO must specify a valid pulse output line number from 0 to 7 or else the signal is sent to an unexpected line having a number equal to the modulo-8 of that SIGNAL_NO.

TABLE 1-1 STATUS CODES FOR THE GENSIG SUBROUTINE

CODE	STATUS
0	No error, requested function completed
1	Pulse output device cannot be accessed
2	APU cannot be accessed
3	APU number is illegal

1.5 IRTCNT Function

This integer*4 function returns the value of the real-time counter (RTC) of the CPU or APU in which the calling task is executing. The value is between -2^{31} and $2^{31}-1$ inclusive.

Format:

```
IRTCNT ([STATUS])
```

Argument:

STATUS — is an optional integer*4 variable to which IRTCNT sends a status code.

Functional Detail:

If STATUS is nonzero, the value of the function is zero (see Table 1-2 for possible status codes).

TABLE 1-2 STATUS CODES FOR THE IRTCNT FUNCTION

CODE	STATUS
0	No error, requested function completed
1	The RTC cannot be read

1.6 RSCHDL Subroutine

The RSCHDL subroutine transfers the calling task to the end of the ready queue of the processor in which the task is executing.

Format:

```
CALL RSCHDL
```

WARNING

USE OF THIS SUBROUTINE IN CONJUNCTION WITH PRIORITY OR PRIORITY-ENFORCED QUEUE DISCIPLINE MAY SERIOUSLY DEGRADE SYSTEM THROUGHPUT. USE OF NO PRIORITY IS RECOMMENDED.

CHAPTER 2
RUN-TIME LIBRARY (RTL) SUBROUTINES

2.1 INTRODUCTION

This chapter contains the Model 3200MPS System real-time routines that are included in the FORTRAN VII RTL. For complete descriptions of these RTL's, see the Model 3200MPS System Overview Manual.

WARNING

THE RTL SUBROUTINES FOR THE MODEL 3200MPS SYSTEM, AS DOCUMENTED IN THE FOLLOWING SECTIONS, ARE APPLICABLE ONLY TO USERS OF OS/32 R07.2 SOFTWARE RELEASE OR HIGHER. USERS OF A MODEL 3200MPS SYSTEM RUNNING UNDER OS/32 R07.1 SOFTWARE RELEASE MUST CONSULT THE FORTRAN VII USERS GUIDE FOR DOCUMENTATION OF MODEL 3200MPS RTL SUBROUTINES AND INTRINSICS.

Except where otherwise stated, all parameters of these routines are integer*4 scalars. TASKID is an 8-byte input argument containing the name of the directed task. The task name is left-justified and padded to the right with blanks, if necessary.

Examples:

```
DOUBLE PRECISION TASKID
DATA TASKID /'NAME    '/
CALL TRLPU(TASKID,...)

INTEGER*4 TASKID(2)
DATA TASKID /'NAME    '/
CALL TRLPU(TASKID,...)
CALL TRLPU(8HNAME    ,...)
CALL TRLPU('NAME    ',...)

CHARACTER*8 TASKID
TASKID = 'NAME    '
CALL TRLPU(TASKID,...)
```

2.2 APUDABL Subroutine

The APUDABL subroutine is used to disable the designated auxiliary processing unit (APU).

Format:

```
CALL APUDABL (APU_NO,STATUS)
```

Arguments:

APU_NO	is an integer*4 input argument that specifies the APU that is to be disabled.
STATUS	is an integer*4 variable to which APUDABL sends a status code (see Table 2-1 for possible status codes).

TABLE 2-1 STATUS CODES FOR THE APUDABL SUBROUTINE

CODE	STATUS
0	No errors
4	Errors, task options prevent granting of privilege
5	Task has not been granted privileges required to perform option
6	Invalid APU number
11	APU not enabled (DISABLED state)
16	Cannot disable unless APU is in ENABLED state

2.3 APUENAB Subroutine

The APUENAB subroutine is used to enable the designated APU.

Format:

```
CALL APUENAB (APU_NO,STATUS)
```

Arguments:

APU_NO	is an integer*4 input argument designating the APU that is to be enabled.
STATUS	is an integer*4 variable to which APUENAB sends a status code (see Table 2-2 for possible status codes).

TABLE 2-2 STATUS CODES FOR THE APUENAB SUBROUTINE

CODE	STATUS
0	No errors
4	Error, task options prevent granting of privilege
5	Task has not been granted privileges required to perform option
6	Invalid APU number
14	Cannot enable unless APU is in DISABLED state
15	APU could not pass power-up link check sequence; APU left in DISABLED state

2.4 ASGNQUE Subroutine

The ASGNQUE subroutine is used to assign the specified APU to the designated queue.

Format:

```
CALL ASGNQUE (APU_NO,QUE_NO,STATUS)
```

Arguments:

APU_NO	is an integer*4 input argument specifying the APU number.
QUE_NO	is an integer*4 input argument designating the queue number.

STATUS — is an integer*4 variable to which ASGNQUE sends a status code (see Table 2-3 for possible status codes).

TABLE 2-3 STATUS CODES FOR THE ASGNQUE SUBROUTINE

CODE	STATUS
0	No errors
4	Error, task options prevent granting of privilege
5	Task has not been granted privileges required to perform option
6	Invalid APU number
11	APU not enabled (DISABLED state)
12	Access to QUE could not be obtained
21	Invalid QUE number

2.5 ASLPU Subroutine

The ASLPU subroutine assigns the designated logical processing unit (LPU) to the directed task.

Format:

```
CALL ASLPU (TASKID,LPU_NO,STATUS)
```

Arguments:

TASKID — is an 8-byte input argument containing the task name. If the argument is an integer*4 zero, the call is self-directed.

LPU_NO — is an integer*4 input argument specifying the LPU number.

STATUS — is an integer*4 variable to which ASLPU returns a status code after execution (see Table 2-4 for possible status codes).

TABLE 2-4 STATUS CODES FOR THE ASLPU SUBROUTINE

CODE	STATUS
0	No error, requested function completed
1	Syntax error in TASKID
4	No such task in calling task environment
9	The calling task cannot execute SVC6 control or communication functions
84	LPU number is outside sysgened range
85	Specified task is APU only

2.6 CNTLAPU Subroutine

The CNTLAPU subroutine is used by a task to gain control rights over the designated APU.

Format

```
CALL CNTLAPU (APU_NO,STATUS)
```

Arguments:

APU_NO	is an integer*4 input argument that specifies the APU over which control rights are to be gained by the calling task.
STATUS	is an integer*4 variable to which CNTLAPU sends a status code (see Table 2-5 for possible status codes).

TABLE 2-5 STATUS CODES FOR THE CNTLAPU SUBROUTINE

CODE	STATUS
0	No errors, requested function completed
4	Task errors prevent granting of privilege
6	Invalid APU number
9	Privilege is currently owned by another task

2.7 HALTAPU Subroutine

The HALTAPU subroutine causes the designated APU to stop (as in STOPAPU) and save the power fail image.

Format:

```
CALL HALTAPU (APU_NO,HARDWARE_STATUS,STATUS)
```

Arguments:

APU_NO	is an integer*4 input argument that specifies the APU number.
HARDWARE_STATUS	is an integer*4 variable that indicates the 2-byte APU response status from the APU processor hardware. See the OS/32 Supervisor Call (SVC) Reference Manual for more information.
STATUS	is an integer*4 variable to which HALTAPU sends a status code (see Table 2-6 for possible status codes).

TABLE 2-6 STATUS CODES FOR THE HALTAPU SUBROUTINE

CODE	STATUS
0	No errors
4	Error, task options prevent granting of privilege
5	Task has not been granted privileges required to perform option
6	Invalid APU number
11	APU not enabled (DISABLED state)
18	Task to be preempted, not on specified APU ready queue.

Functional Detail:

This call is similar to STOPAPU except that the save power fail image is performed. Therefore, HALTAPU has a greater impact on the system because of the additional overhead.

2.8 IDAPU Function

This integer*4 function returns the APU_ID number of the processor executing the calling task. This function returns a value of -1 if the device specified by APU_ID cannot be read. Implementation of this function is via the GSIG instruction.

Format:

```
IDAPU ()
```

2.9 LPUPREF Subroutine

The LPUPREF subroutine changes the LPU-directed option of the directed task.

Format:

```
CALL LPUPREF (TASKID,FLAG,STATUS)
```

Arguments:

TASKID	is an 8-byte input argument containing the task name. If this argument is an integer*4 zero, the call is self-directed.
FLAG	is an input LOGICAL*4 argument. If FLAG is .TRUE., the LPU-directed option of the directed task is turned on. If FLAG is .FALSE., the LPU-directed option of the directed task is turned off.
STATUS	is an integer*4 variable to which LPUPREF returns a status code after execution (see Table 2-7 for possible status codes).

TABLE 2-7 STATUS CODES FOR THE LPUPREF SUBROUTINE

CODE	STATUS
0	No error, requested function completed
1	Syntax error in task name field
4	Directed task not present in the calling task
9	The calling task cannot execute SVC6 or communication function

2.10 LPUQUE Subroutine

The LPUQUE subroutine maps the designated LPU to the designated queue.

Format:

```
CALL LPUQUE (LPU_NO,QUE_NO,STATUS)
```

Arguments:

LPU_NO	is an integer*4 input argument that specifies the LPU number.

QUE_NO — is an integer*4 input argument that specifies the queue number.

STATUS — is an integer*4 variable to which LPUQUE sends a status code (see Table 2-8 for possible status codes).

TABLE 2-8 STATUS CODES FOR THE LPUQUE SUBROUTINE

CODE	STATUS
0	No errors
4	Error, task options prevent granting of privilege
5	Task has not been granted privileges required to perform option
7	Invalid LPU number
13	Task has not been granted mapping rights over the APU currently mapped to the specified LPU
21	Invalid QUE number

2.11 MAPQUE Subroutine

The MAPQUE subroutine is used to gain the mapping rights of the designated queue.

Format:

```
CALL MAPQUE (QUE_NO,STATUS)
```

Arguments:

QUE_NO — is an integer*4 input argument designating the queue for which mapping rights are requested.

STATUS — is an integer*4 variable to which MAPQUE sends a status code (see Table 2-9 for possible status codes).

TABLE 2-9 STATUS CODES FOR THE MAPQUE SUBROUTINE

CODE	STATUS
0	No error, requested function completed
4	Task options prevent granting of privilege
5	Task has not been granted privileges required to perform option
9	Privilege is currently owned by another task
21	Invalid QUE number

2.12 MAPTABL Subroutine

The MAPTABL subroutine fetches the maximum LPU number, the maximum APU number, the APU-to-APU queue assignment table and the LPU-to-APU queue mapping table. If there are no errors, this information is stored in the integer*4 array buffer.

Format:

```
CALL MAPTABL (BUFFER,BUFFER_LENGTH,LENGTH_USED,STATUS)
```

Arguments:

BUFFER	is an integer*4 array that is filled using the format in Figure 2-1.
BUFFER_LENGTH	is an input argument indicating the number of elements in the array BUFFER that can be overwritten in this call. If STATUS is nonzero, BUFFER is unchanged.
LENGTH_USED	is an output argument indicating the number of elements in the array BUFFER actually filled by MAPTABL.
STATUS	is an integer*4 variable to which MAPTABL sends a status code (see Table 2-10 for possible status codes).

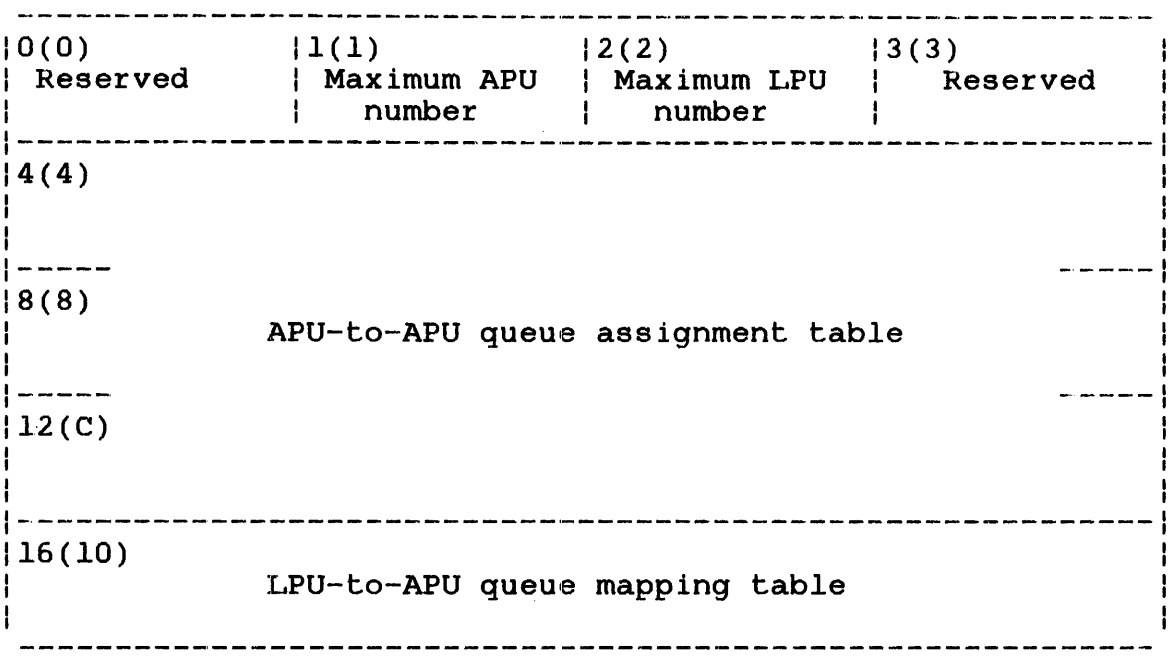

Figure 2-1 Format of Elements in the Array BUFFER Used by the MAPTABL Subroutine

TABLE 2-10 STATUS CODES FOR THE MAPTABL SUBROUTINE

CODE	STATUS
0	No errors, requested function completed
1	Buffer alignment error
2	Buffer not writable
3	Insufficient buffer space

Functional Details:

The APU-to-APU queue assignment table is a 12-byte array (0:m) of 1-byte entries, where m is the maximum APU number. Each number, i, contains the APU queue number to which APU i is mapped, where APU0=CPU. Entries for i > maximum APU number are zero and not meaningful. The CPU entry is always 0.

The LPU-to-APU queue mapping table is a variable length array (0:n) of 1-byte entries, where n is the maximum LPU number. Each entry number, k, contains the APU queue number to which LPU k is mapped. LPU0 is always mapped to queue 0. For more information, see the OS/32 Supervisor Call (SVC) Reference Manual.

2.13 QUEOFF Subroutine

The QUEOFF subroutine marks off the designated queue.

Format:

```
CALL QUEOFF (QUE_NO,STATUS)
```

Arguments:

QUE_NO	is an integer*4 input argument that specifies the number of the queue to be marked off.
STATUS	is an integer*4 variable to which QUEOFF sends a status code (see Table 2-11 for possible status codes).

TABLE 2-11 STATUS CODES FOR THE QUEOFF SUBROUTINE

CODE	STATUS
0	No errors
4	Error, task options prevent granting of privilege
5	Task has not been granted privileges required to perform option
12	Access to QUE could not be obtained
21	Invalid QUE number

2.14 QUEON Subroutine

The QUEON subroutine marks on the designated queue.

Format:

```
CALL QUEON (QUE_NO,STATUS)
```

Arguments:

QUE_NO	is an integer*4 input argument that specifies the queue to be marked on.
STATUS	is an integer*4 variable to which QUEON sends a status code (see Table 2-12 for possible status codes).

TABLE 2-12 STATUS CODES FOR THE QUEON SUBROUTINE

CODE	STATUS
0	No errors
4	Error, task options prevent granting of privilege
5	Task has not been granted privileges required to perform option
21	Invalid QUE number

2.15 QUEXON Subroutine

The QUEXON subroutine marks the designated queue ON-exclusive.

Format:

```
CALL QUEXON (QUE_NO,STATUS,TASKID)
```

Arguments:

QUE_NO	is an integer*4 input argument that specifies the number of the queue to be marked ON-exclusive.

STATUS — is an integer*4 variable to which QUEXON sends a status code (see Table 2-13 for possible status codes).

TASKID — is an 8-byte input argument containing the task name for which the queue is to be marked ON-exclusive. An integer*4 zero in this argument means this call is self-directed.

TABLE 2-13 STATUS CODES FOR THE QUEXON SUBROUTINE

CODE	STATUS
0	No errors
1	Syntax error in TASKID
3	Insufficient buffer space
4	Error, task options prevent granting of privilege
5	Task has not been granted privileges required to perform option
10	QUE cannot be marked ON-exclusive from on state
17	Task could not be found in system, APU not marked ON-exclusive
21	Invalid QUE number
24	The QUE cannot be marked ON-exclusive because more than one APU is currently assigned to it

2.16 RELCNTL Subroutine

The RELCNTL subroutine releases controlling rights over the designated APU.

Format:

```
CALL RELCNTL (APU_NO,STATUS)
```

Arguments:

APU_NO — is an integer*4 input argument that specifies the APU for which controlling rights are to be released.

STATUS — is an integer*4 variable to which RELCNTL sends a status code (see Table 2-14 for possible status codes).

TABLE 2-14 STATUS CODES FOR THE RELCNTL SUBROUTINE

CODE	STATUS
0	No errors
4	Error, task options prevent granting of privilege
5	Task has not been granted privileges required to perform option
6	Invalid APU number

2.17 RELMAP Subroutine

The RELMAP subroutine is used by a task to release the mapping privileges of the specified APU queue.

Format:

```
CALL RELMAP (QUE_NO,STATUS)
```

Arguments:

QUE_NO — is an integer*4 input argument that specifies the queue number.

STATUS — is an integer*4 variable to which RELMAP sends a status code (see Table 2-15 for possible status codes).

TABLE 2-15 STATUS CODES FOR THE RELMAP SUBROUTINE

CODE	STATUS
0	No errors
4	Error, task options prevent granting of privilege
5	Task has not been granted privileges required to perform option.
6	Invalid QUE number

2.18 REMQUE Subroutine

The REMQUE subroutine maps all LPUs mapped to the specified QUE to queue 0.

Format:

```
CALL REMQUE (QUE_NO,STATUS)
```

Arguments:

QUE_NO	is an integer*4 input argument that specifies the queue number.
STATUS	is an integer*4 variable to which REMQUE sends a status code (see Table 2-16 for possible status codes).

TABLE 2-16 STATUS CODES FOR THE REMQUE SUBROUTINE

CODE	STATUS
0	No errors
4	Error, task options prevent granting of privilege
5	Task has not been granted privileges required to perform option
21	Invalid QUE number
22	Specified option is not applicable to QUE to which the mapping function is directed

2.19 RSCHAPU Subroutine

The RSCHAPU subroutine causes the designated APU to place the active task at the end of its ready queue and to replace it with another task in its ready queue.

Format:

```
CALL RSCHAPU (APU_NO,HARDWARE_STATUS,STATUS [,TASKID])
```

Arguments:

APU_NO	is an integer*4 input argument that specifies the APU number.
HARDWARE_STATUS	is an integer*4 variable that indicates the 2-byte APU response status from the APU processor hardware. See the OS/32 Supervisor Call (SVC) Reference Manual for more information.
STATUS	is an integer*4 variable to which RSCHAPU sends a status code (see Table 2-17 for possible status codes).

TASKID	is an optional 8-byte input argument containing the name of the task to be rescheduled. If this parameter is not present, the next active task on the designated APU is rescheduled.

Functional Detail:

If TASKID is currently suspended, this subroutine releases it. This call cannot be self-directed.

TABLE 2-17 STATUS CODES FOR THE RSCHAPU SUBROUTINE

CODE	STATUS
0	No errors
3	Insufficient buffer space for TASKID
4	Error, task option prevent granting of privilege
5	Task has not been granted privileges required to perform option
6	Invalid APU number
11	APU not enabled (DISABLED state)
12	Access to task queue for the specified APU could not be obtained
18	Error in transmission/execution of command
19	Task to be preempted, not on specified APU ready queue

2.20 SETQUED Subroutine

The SETQUED subroutine is used to select a specified queue discipline for the designated queue.

Format:

```
CALL SETQUED (QUE_NO,QUE_DIS,STATUS)
```

Arguments:

QUE_NO — is an integer*4 input argument that designates the queue number.

QUE_DIS — is an integer*4 input argument that specifies one of the following queue discipline numbers:

0 indicates no priority.
1 indicates priority.
2 indicates priority-enforced.

STATUS — is an integer*4 variable to which SETQUED sends a status code (see Table 2-18 for possible status codes).

TABLE 2-18 STATUS CODES FOR THE SETQUED SUBROUTINE

CODE	STATUS
0	No errors
4	Error, task options prevent granting of privilege
5	Task has not been granted privileges required to perform option
12	Access to QUE could not be obtained
21	Invalid QUE number
23	QUE discipline specified is undefined

2.21 STAQUE Subroutine

The STAQUE subroutine returns information about the designated queue.

Format:

```
CALL STAQUE (QUE_NO,BUFFER,BUFFER_LENGTH,LENGTH_USED,
             HARDWARE_STATUS,STATUS)
```

Arguments:

QUE_NO	is an integer*4 input argument specifying the QUE number.
BUFFER	is an integer*4 array that remains unchanged if STATUS is nonzero. Otherwise, BUFFER is filled using the format in Figure 2-2.
BUFFER_LENGTH	is an input integer*4 argument that indicates the number of elements in the array BUFFER that can be overwritten in this call. The minimum size is 24 bytes.
LENGTH_USED	is an integer*4 variable that indicates the number of elements in BUFFER actually filled by STAQUE.
HARDWARE_STATUS	is an integer*4 variable that indicates the 2-byte APU response status from the APU processor hardware. See the OS/32 Supervisor Call (SVC) Manual for more information.
STATUS	is an integer*4 variable to which STAQUE sends a status code (see Table 2-19 for possible status codes).

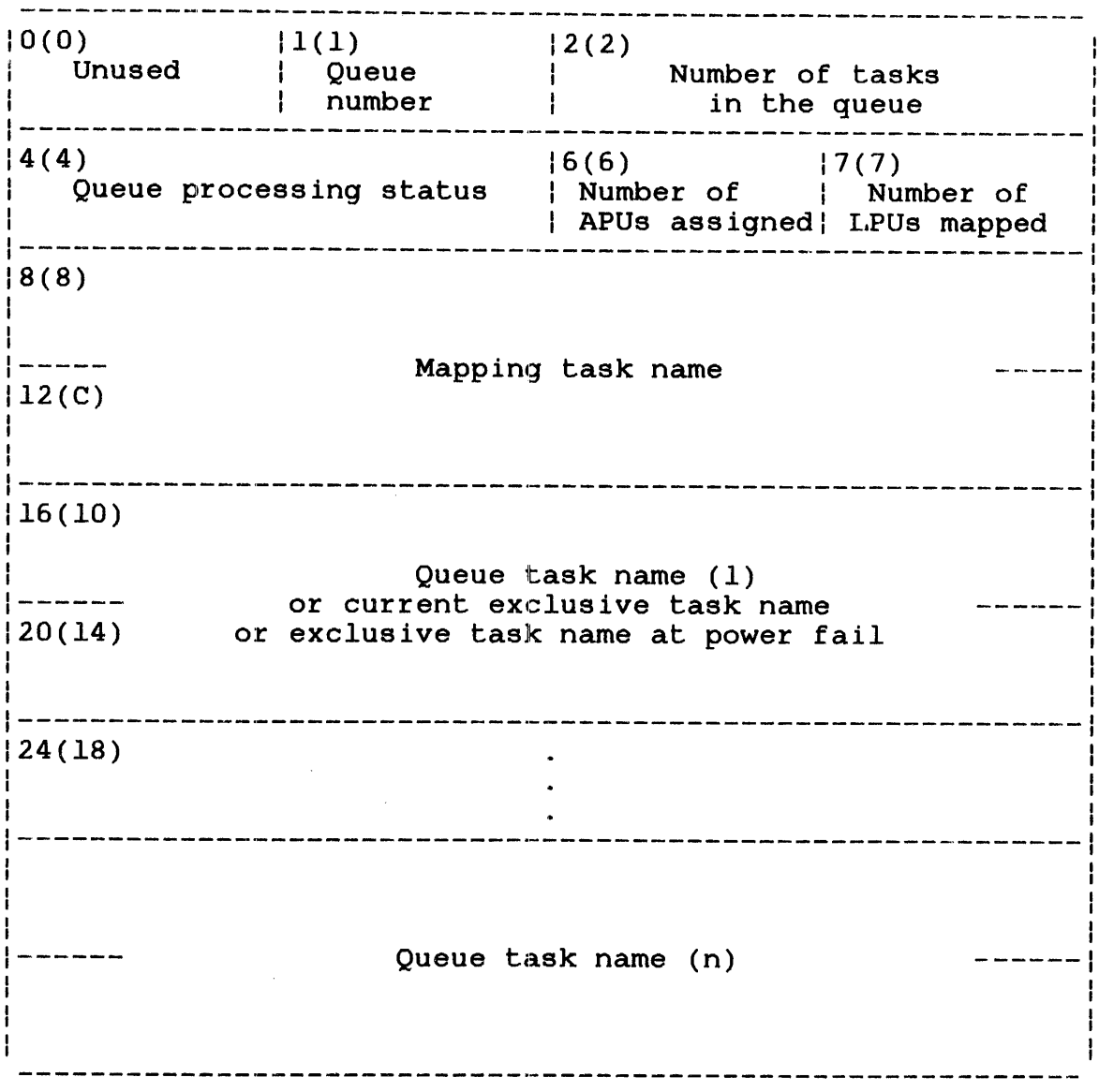

Figure 2-2 Layout of Information Returned by STAQUE to BUFFER

TABLE 2-19 STATUS CODES FOR THE STAQUE SUBROUTINE

CODE	STATUS
0	No errors
1	Buffer alignment errors
2	Buffer not writable
3	Insufficient buffer space
12	Access to QUE could not be obtained
21	Invalid QUE number

2.22 STATAPU Subroutine

The STATAPU subroutine returns information about the designated APU.

Format:

```
CALL STATAPU (APU_NO,BUFFER,BUFFER_LENGTH,LENGTH_USED,
              HARDWARE_STATUS,STATUS)
```

Arguments:

APU_NO	is an integer*4 input argument specifying the APU number.
BUFFER	is an integer*4 array that is unchanged if STATUS is nonzero. Otherwise, BUFFER is filled using the format in Figure 2-3.
BUFFER_LENGTH	is an input integer*4 argument that indicates the number of elements in the array BUFFER that can be overwritten in this call. The minimum size is 40 bytes.
LENGTH_USED	is an integer*4 variable that indicates the number of elements in BUFFER actually filled by STATAPU.

HARDWARE_STATUS is an integer*4 variable that indicates the 2-byte APU response status from the APU processor hardware. See the OS/32 Supervisor Call (SVC) Manual for more information.

STATUS is an integer*4 variable to which STATAPU sends a status code (see Table 2-20 for possible status codes).

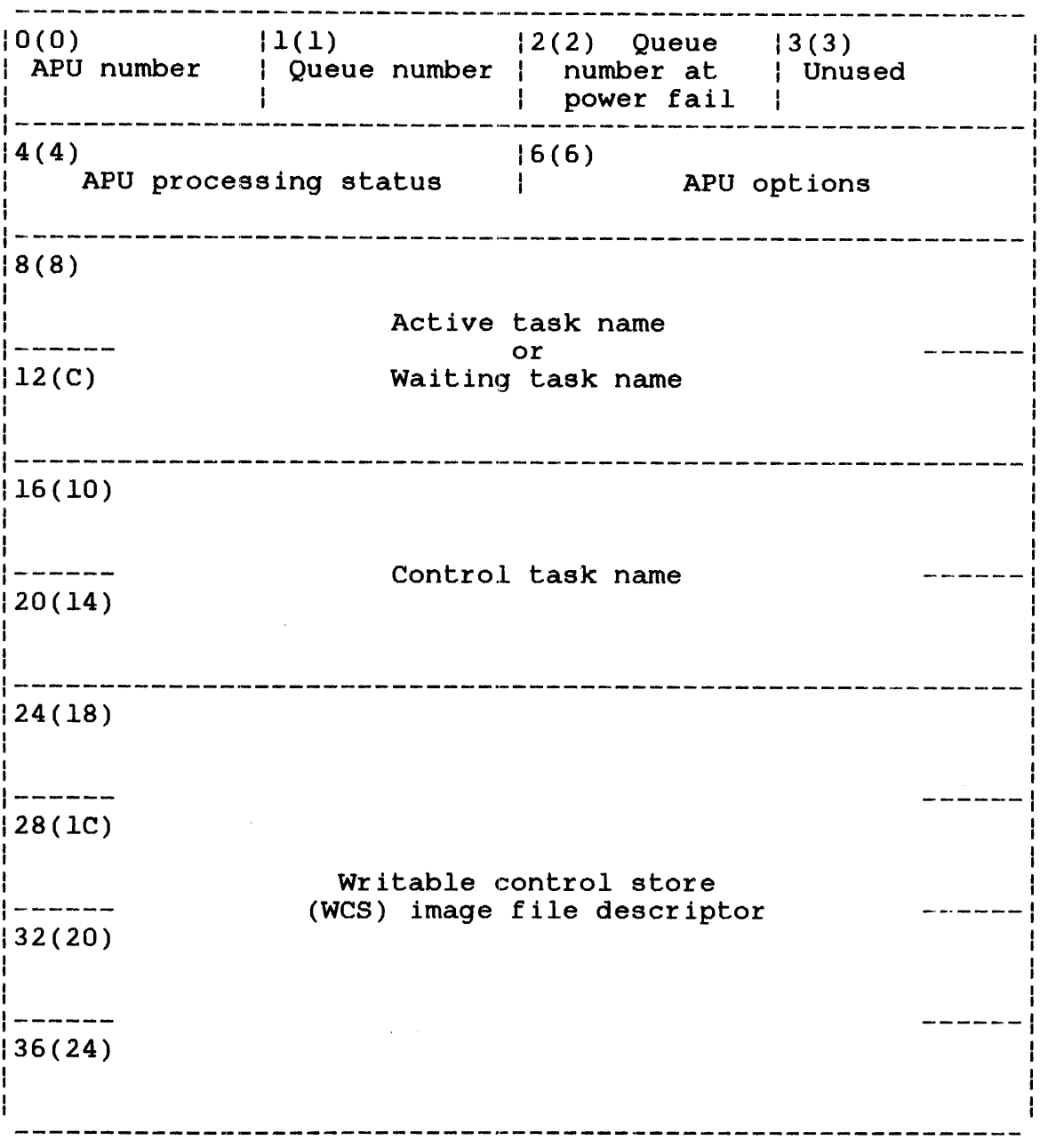

Figure 2-3 Layout of Information Returned by STATAPU to BUFFER

For a complete explanation of the above field names, see the OS/32 Supervisor Call (SVC) Reference Manual.

TABLE 2-20 STATUS CODES FOR THE STATAPU SUBROUTINE

CODE	STATUS
0	No error, requested function completed
1	Buffer alignment error
2	Buffer not writable
3	Insufficient buffer space
6	Invalid APU number

2.23 STOPAPU Subroutine

The STOPAPU subroutine causes the designated APU to stop if it is currently executing.

Format:

```
CALL STOPAPU (APU_NO,HARDWARE_STATUS,STATUS)
```

Arguments:

APU_NO	is an integer*4 input argument that specifies the APU number.
HARDWARE_STATUS	is an integer*4 variable that indicates the 2-byte APU response status from the APU processor hardware. See the OS/32 Supervisor Call (SVC) Reference Manual for more information.
STATUS	is an integer*4 variable to which STOPAPU sends a status code (see Table 2-21 for possible status codes).

TABLE 2-21 STATUS CODES FOR THE STOPAPU SUBROUTINE

CODE	STATUS
0	No errors
4	Error, task options prevent granting of privilege
5	Task has not been granted privileges required to perform option
6	Invalid APU number
11	APU not enabled (DISABLED state)
18	Task to be preempted, not on specified APU ready queue

2.24 STRTAPU Subroutine

The STRTAPU subroutine starts the designated APU for task execution.

Format:

```
CALL STRTAPU (APU_NO,HARDWARE_STATUS,STATUS)
```

Arguments:

APU_NO	is an integer*4 input argument that specifies the APU number.
HARDWARE_STATUS	is an integer*4 variable that indicates the 2-byte APU response status from the APU processor hardware. See the OS/32 Supervisor Call (SVC) Reference Manual for more information.
STATUS	is an integer*4 variable to which STRTAPU sends a status code (see Table 2-22 for possible status codes).

TABLE 2-22 STATUS CODES FOR THE STRTAPU SUBROUTINE

CODE	STATUS
0	No errors
4	Error, task options prevent granting of privilege
5	Task has not been granted privileges required to perform option
6	Invalid APU number
11	APU not enabled (DISABLED state)
18	Task to be preempted, not on specified APU ready queue

2.25 TRLPU Subroutine

The TRLPU subroutine assigns the designated LPU to the directed task and transfers that task to the LPU. If not self-directed, the task is suspended and assigned to its new LPU. The LPU directed option is turned on and the task is released from the suspended state. If the desired task is ready with a higher priority than the current task, it is immediately transferred to the APU queue.

Format:

```
CALL TRLPU (TASKID,LPU_NO,STATUS)
```

Arguments:

TASKID	is an 8-byte input argument containing the task name. If this argument is an integer*4 zero, the call is self-directed.
LPU_NO	is an integer*4 input argument specifing the LPU number.
STATUS	is an integer*4 variable to which TRLPU returns a status code after execution (see Table 2-23 for possible status codes).

TABLE 2-23 STATUS CODES FOR THE TRLPU SUBROUTINE

CODE	STATUS
0	No error, requested function completed
1	Syntax error in TASKID
4	No such task in calling task environment
9	The calling task cannot execute SVC6 control or communication functions
84	LPU number is outside system generation (sysgen) range
85	Specified task is APU only

INDEX

A,B

APUDABL subroutine
  status codes 2-2
APUENAB subroutine 2-2
  status codes 2-3
ASGNQUE subroutine 2-3
  status codes 2-4
ASLPU subroutine 2-4
  status codes 2-5

C,D,E

CNTLAPU subroutine 2-5
  status codes 2-6
Compilers 1-1

F

Functions
  intrinsic (IRTCNT) 1-4
  RTL (IDAPU) 2-7

G

GENSIG subroutine 1-3
  status codes 1-4
GOAPU subroutines 1-1
GOCPU subroutine 1-2

H

HALTAPU subroutine 2-6
  status codes 2-7

I,J,K

IDAPU function 2-7
Intrinsic routines
  arguments 1-1
  O and Z compilers 1-1
Intrinsic subroutines.
 See subroutines, intrinsic. 1-1
IRTCNT function
  status codes 1-4

L

LPUPREF subroutine 2-7
  status code 2-8
LPUQUE subroutine 2-8
  status codes 2-9

M,N,O,P

MAPQUE subroutine 2-9
  status codes 2-10
MAPTABL subroutine 2-10
  status codes 2-11

Q

QUEOFF subroutine
  status codes 2-12
QUEON subroutine 2-12
  status codes 2-13
QUEXON subroutine 2-13
  status codes 2-14

R

RELCNTL subroutine 2-14
  status codes 2-15
RELMAP subroutine 2-15
  status codes 2-16
REMQUE subroutine 2-16
  status codes 2-17
Routines, intrinsic 1-1
RSCHAPU subroutine 2-17
  status codes 2-18
RSCHDL subroutine 1-5
Run-time library (RTL)
 subroutines. See subroutines,
 RTL.

S

SETQUED subroutine 2-18
  status codes 2-19
STAQUE subroutine
  layout of information 2-21
  status codes 2-22
STATAPU subroutine 2-22
  layout of information 2-23
  status codes 2-24
Status codes
  APUDABL 2-2
  APUENAB 2-3
  ASGNQUE 2-4
  ASLPU 2-4
    2-5
  CNTLAPU 2-6
  GENSIG 1-4
  HALTAPU 2-7
  intrinsic 1-4
  IRTCNT
  LPUPREF 2-8
  LPUQUE 2-9
  MAPQUE 2-10
  MAPTABL 2-11
  MENQUE 2-17
  QUEOFF 2-12
  QUEON 2-13

Status codes (Continued)
QUEXON 2-14
RELCNTL 2-15
RELMAP 2-16
RSCHAPU 2-18
STAQUE 2-22
STATAPU 2-24
STOPAPU 2-25
STRTAPU 2-26
TRLPU 2-27
2-19
STOPAPU subroutine 2-24
status codes 2-25
STRTAPU subroutine 2-25
status codes 2-26
Subroutine
RSCHDL 1-5
Subroutines
APUDABL 2-2
APUENAB 2-2
ASGNQUE 2-3
CNTLAPU 2-5
GENSIG 1-3
GOAPU 1-1
GOCPU 1-2
HALTAPU 2-6
LPUPREF 2-7
LPUQUE 2-8
MAPQUE 2-9
MAPTABL 2-10
QUEOFF 2-12
QUEON 2-12
QUEXON 2-13
RELCNTL 2-14
RELMAP 2-15
REMQUE 2-16
RSCHAPU 2-17
SETQUED 2-18
STAQUE 2-19
STATAPU 2-22
STOPAPU 2-24
STRTAPU 2-25
TRLPU 2-26
Subroutines, intrinsic
GENSIG 1-3
GOAPU 1-1
GOCPU 1-2
RSCHDL 1-5
Subroutines, RTL 2-4
APUDABL 2-2
APUENAB 2-2
ASGNQUE 2-3
CNTLAPU 2-5
HALTAPU 2-6
LPUPREF 2-7
LPUQUE 2-8
MAPQUE 2-9
MAPTABL 2-10
parameters 2-1
QUEOFF 2-12
QUEON 2-12
QUEXON 2-13
RELCNTL 2-14
RELMAP 2-15
Subroutines, RTL (Continued)
REMQUE 2-16
RSCHAPU 2-17
SETQUED 2-18
STAQUE 2-19
STATAPU 2-22
STOPAPU 2-24
STRTAPU 2-25
TASKID 2-1
TRLPU 2-26

T-Z

TRLPU subroutine 2-26
status codes 2-27

PERKIN-ELMER

PUBLICATION COMMENT FORM

We try to make our publications easy to understand and free of errors. Our users are an integral source of information for improving future revisions. Please use this postage paid form to send us comments, corrections, suggestions, *etc.*

1. Publication number _____

2. Title of publication _____

3. Describe, providing page numbers, any technical errors you found. Attach additional sheet if neccessary.

   _____

   _____

   _____

4. Was the publication easy to understand? If no, why not?

   _____

5. Were illustrations adequate? _____

   _____

   _____

6. What additions or deletions would you suggest? _____

   _____

   _____

7. Other comments: _____

   _____

   _____

From _____ Date _____

Position/Title _____

Company _____

Address _____

_____

_____

STAPLE

STAPLE

FOLD

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 22 OCEANPORT, N.J.

POSTAGE WILL BE PAID BY ADDRESSEE

PERKIN-ELMER

Data Systems Group
106 Apple Street
Tinton Falls, NJ 07724

ATTN:
TECHNICAL SYSTEMS PUBLICATIONS DEPT.

FOLD

FOLD

STAPLE

STAPLE

6433-1