

IAS Guide to Writing a Command Language Interpreter

Order Number: AA-D120D-TC

This gives programmers detailed instructions on how to write their own CLIs.

Operating System and Version: IAS Version 3.4

May 1990

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

Restricted Rights: Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.


No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

Copyright ©1990 by Digital Equipment Corporation

All Rights Reserved.
Printed in U.S.A.

The postpaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

| | | |
|------------|---------|-------------------------------------------------------------------------------------|
| DDIF | IAS | VAX C |
| DEC | MASSBUS | VAXcluster |
| DEC/CMS | PDP | VAXstation |
| DEC/MMS | PDT | VMS |
| DECnet | RSTS | VR150/160 |
| DECUS | RSX | VT |
| DECwindows | ULTRIX | |
| DECwrite | UNIBUS | |
| DIBOL | VAX |  |

This document was prepared using VAX DOCUMENT, Version 1.2

Contents

PREFACE

ix

CHAPTER 1 INTRODUCTION

1-1

1.1 COMMAND LANGUAGE INTERPRETERS

1-1

1.2 TIMESHARING CONTROL SERVICES

1-1

1.2.1 Subtasks

1-2

1.2.2 Task Events

1-2

1.2.3 Ctl/C Events

1-2

1.2.4 Chaining

1-2

1.2.5 Communication Between Tasks

1-2

1.3 TYPES OF TCS MACROS

1-2

1.3.1 Data Definition Macros (Assembly Time)

1-3

1.3.2 Data Definition Macros (Run Time)

1-3

1.3.3 Imperative Macros

1-3

CHAPTER 2 WRITING A CLI

2-1

2.1 THE REQUIREMENT FOR USER-WRITTEN CLIS

2-1

2.2 TYPES OF CLI

2-1

2.3 SPECIAL FACILITIES AVAILABLE TO A CLI

2-1

2.3.1 System Shutdown

2-2

2.3.2 Limiting Subtasking

2-3

2.4 GUIDELINES FOR WRITING AND RUNNING CLIS

2-4

Contents

CHAPTER 3 THE CREATION AND CONTROL OF SUBTASKS 3-1

| | | |
|---------|-----------------------------------------|-----|
| 3.1 | CONTROL OF SUBTASKS | 3-1 |
| 3.1.1 | Initiation of Subtasks | 3-1 |
| 3.1.1.1 | Reading the Command Line • 3-1 | |
| 3.1.2 | Suspending Subtasks | 3-2 |
| 3.1.3 | Setting the Subtask's Local Event Flags | 3-2 |
| 3.1.4 | Aborting the Subtask | 3-2 |
| 3.1.5 | Checking the State of the Subtask | 3-2 |
| 3.1.6 | Waiting for Task Completion | 3-3 |
| 3.2 | CREATION OF SUBTASKS | 3-3 |
| 3.2.1 | Contents of a Task Descriptor Block | 3-3 |
| 3.2.1.1 | Command Line Section • 3-4 | |
| 3.2.1.2 | Privilege Section • 3-5 | |
| 3.2.1.3 | Attribute Section • 3-5 | |
| 3.2.1.4 | Event Information Section • 3-6 | |
| 3.2.1.5 | Error Status Block Section • 3-7 | |
| 3.2.2 | Internal Fields | 3-7 |
| 3.3 | TASK ACCOUNTING | 3-7 |

CHAPTER 4 COMMUNICATION BETWEEN TASKS 4-1

| | | |
|-------|---------------------------------------------|-----|
| 4.1 | SETTING UP THE MESSAGE | 4-1 |
| 4.1.1 | Contents of a Send/Receive Data Block (SDB) | 4-1 |
| 4.2 | SENDING MESSAGES | 4-2 |
| 4.3 | RECEIVING MESSAGES | 4-3 |

CHAPTER 5 **CTRL/C** EVENTS 5-1

| | | |
|---------------------------------------------------------------|-----------------------------------------------------|------------|
| CHAPTER 6 CHAINING | | 6-1 |
| 6.1 | TCS INTERFACE | 6-1 |
| 6.2 | CONTENTS OF A CHAIN TASK DESCRIPTOR BLOCK | 6-1 |
| CHAPTER 7 USING THE TIMESHARING CONTROL SERVICES (TCS) | | 7-1 |
| 7.1 | INCLUDING TIMESHARING CONTROL SERVICES IN A PROGRAM | 7-1 |
| 7.2 | INITIALIZATION OF TIMESHARING CONTROL SERVICES | 7-1 |
| 7.3 | PRIVILEGE | 7-2 |
| 7.3.1 | Allocation of Privilege | 7-2 |
| 7.3.2 | Privileged Terminals | 7-3 |
| 7.4 | ERROR HANDLING | 7-3 |
| 7.4.1 | Error Handling Subroutines | 7-4 |
| 7.5 | EVENT FLAG USAGE | 7-4 |
| 7.6 | MACRO CHARACTERISTICS | 7-4 |
| 7.6.1 | Imperative Macros | 7-4 |
| 7.6.1.1 | Parameters | 7-4 |
| 7.6.1.2 | Register Usage | 7-5 |
| 7.6.2 | Data Definition Macros (Assembly Time) | 7-5 |
| 7.6.2.1 | Parameters | 7-5 |
| 7.6.2.2 | Offset Definitions | 7-5 |
| 7.6.3 | Data Definition Macros (Run Time) | 7-5 |
| 7.6.3.1 | Register Usage | 7-5 |
| 7.6.3.2 | Error Handling | 7-6 |
| 7.6.3.3 | Parameters | 7-6 |

Contents

| | | |
|------------------|-------------------------------|------------|
| CHAPTER 8 | TCS MACRO DESCRIPTIONS | 8-1 |
| | ABRT\$T | 8-2 |
| | CDBDF\$ | 8-4 |
| | CDDF\$R | 8-5 |
| | CHN\$T | 8-6 |
| | CKEV\$T | 8-8 |
| | CTC\$T | 8-11 |
| | ESBDF\$ | 8-13 |
| | JNOD\$T | 8-14 |
| | RCV\$T | 8-16 |
| | RDEV\$T | 8-18 |
| | RSAS\$T | 8-20 |
| | RSUM\$T | 8-22 |
| | RUN\$T | 8-24 |
| | SDBDF\$ | 8-27 |
| | SDDF\$R | 8-29 |
| | SEND\$T | 8-31 |
| | SETF\$T | 8-33 |
| | SHUT\$T | 8-35 |
| | SPND\$T | 8-37 |
| | TCOFF\$ | 8-39 |
| | TCSMC\$ | 8-40 |
| | TDBDF\$ | 8-41 |
| | TDBD\$T | 8-42 |
| | TDBR\$T | 8-44 |
| | TDCM\$A | 8-46 |
| | TDCM\$R | 8-47 |
| | TDEB\$A | 8-48 |
| | TDOFF\$ | 8-49 |
| | TDPR\$A | 8-50 |
| | TDPR\$R | 8-52 |
| | TDTA\$A | 8-54 |
| | TDTA\$R | 8-56 |
| | TEOFF\$ | 8-58 |
| | TINIT\$ | 8-59 |
| | TKST\$T | 8-61 |
| | TSOFF\$ | 8-63 |

| | | |
|-------------------|--------------------------------------|------------|
| APPENDIX A | DATA BLOCK LAYOUTS | A-1 |
| A.1 | TASK DESCRIPTOR BLOCK | A-1 |
| | A.1.1 T.EVBF—Termination Information | A-2 |
| A.2 | CHAIN TASK DESCRIPTOR BLOCK | A-2 |
| A.3 | SEND/RECEIVE DATA BLOCK | A-3 |

| | | | |
|-------------------|-------------------------------|-------------------------------------|------------|
| | A.3.1 | Message Format _____ | A-3 |
| <hr/> | | | |
| | A.4 | ERROR STATUS BLOCK | A-3 |
| <hr/> | | | |
| APPENDIX B | ERROR CODES | | B-1 |
| <hr/> | | | |
| | B.1 | TASK TERMINATION CODES | B-1 |
| <hr/> | | | |
| | B.2 | RESULT CODES | B-2 |
| <hr/> | | | |
| APPENDIX C | EXAMPLES OF TCS MACROS | | C-1 |
| <hr/> | | | |
| INDEX | | | |
| <hr/> | | | |
| FIGURES | | | |
| | 2-1 | A Typical System Plan _____ | 2-2 |
| | 2-2 | A Simple CLI _____ | 2-3 |
| | 2-3 | A Complex CLI _____ | 2-3 |
| | 5-1 | A Task Hierarchy _____ | 5-1 |
| | A-1 | Task Descriptor Block _____ | A-1 |
| | A-2 | T.EVBF _____ | A-2 |
| | A-3 | Chain Task Descriptor Block _____ | A-2 |
| | A-4 | Send/Receive Data Block _____ | A-3 |
| | A-5 | Message Format _____ | A-3 |
| | A-6 | Error Status Block _____ | A-4 |
| <hr/> | | | |
| TABLES | | | |
| | 7-1 | Error Status Block _____ | 7-3 |
| | B-1 | Task Termination Codes _____ | B-1 |
| | B-2 | Imperative Macro Return Codes _____ | B-2 |

Preface

Purpose of the Manual

The IAS operating system provides two standard Command Language Interpreters (CLIs): the Program Development System (PDS) and the Monitor Console Routine (MCR). This manual is intended for users who require nonstandard CLIs. The manual will enable system programmers to write and run their own nonstandard CLIs on IAS.

In addition, the manual describes the IAS timesharing facilities.

Document Structure

Chapters 1 to 7 are descriptive and should be read sequentially.

- Chapter 1 introduces CLIs and timesharing control services (TCS).
 - Chapter 2 contains guidelines for writing CLIs.
 - Chapters 3 to Chapter 7 describe TCS.
 - Chapter 8 contains TCS macro descriptions and is designed for ease of reference.
 - Appendix A shows the layouts and offset names for the various data blocks you need to write CLIs.
 - Appendix B lists task termination codes and result codes.
 - Appendix C contains examples of TCS.
-

Associated Documents

You should be acquainted with the *IAS System Management Guide*, the *IAS PDS User's Guide*, and the *PDP-11 MACRO-11 Reference Manual*.

All IAS manuals are detailed in the *IAS Master Index and Documentation Directory*.

1

Introduction

1.1 Command Language Interpreters

A Command Language Interpreter (CLI) is a task that enables the outside world to communicate with the IAS system. In reference to commands, CLIs perform the following functions:

- Receive them.
- Interpret their meaning.
- Translate them into statements understandable to the IAS system.

Typically, the commands are part of an overall language that enables users to specify commands and command variations of commands.

A CLI differs from other timesharing tasks in that it is written to communicate with a single input stream (of commands) and is allocated to an input/output device. Typically, this device is a keyboard where a human user can type commands. However, the device can be any logical or physical device that the CLI task is programmed to communicate with.

A CLI built as a multiuser task (see Section 2.4), can be allocated to more than one device (for example, many terminals) at the same time. In this case, when more than one CLI task is active, only one copy of the task's pure area exists for all active CLI tasks and one copy of the impure area exists for each active CLI task. This method is generally more efficient and flexible than writing a single task that services multiple input streams.

A user-written CLI can use many facilities available in the IAS system; for example, it might use the IAS file system, the system directives, and the system utilities. The timesharing control services (TCS) is a facility included in the IAS system specifically for use by CLI tasks.

1.2 Timesharing Control Services

Timesharing control primitives (TCP) are responsible for monitoring and maintaining all timesharing tasks (including CLIs).

For more information on TCP, see the *IAS System Management Guide*. TCS enables a user program to interface with TCP. TCS consists of two parts:

- 1 A set of macros
- 2 A set of subroutines.

The required macros are included in the user source program. These macros generate data areas and calls to the TCS subroutines that are included in the user task by the Task Builder.

Introduction

1.2.1 Subtasks

When a CLI is active, it can use TCS to initiate further timesharing tasks. These tasks are known as subtasks of the CLI. The CLI owns the subtasks and maintains overall control of them. The CLI can abort or suspend the subtask execution, or be notified of subtask events (such as subtask exit). By issuing privileges to subtasks, the CLI controls the operation of a subtask.

If a subtask has sufficient privilege, it can use TCS to initiate further subtasks. This might lead to a hierarchical structure of tasks at a terminal. Subtasks are described in Chapter 3.

1.2.2 Task Events

If a task suspends itself, its owner receives a task event (through TCS) to notify it of the suspension. Another example of a task event is a task exiting or failing to load after a swap. These task events are returned to the task owner (see Section 3.1.5).

1.2.3 `Ctrl/C` Events

A `Ctrl/C` event (or terminal event) occurs when you type `Ctrl/C` on the terminal. Only one task in the possible hierarchy of tasks at the terminal is notified of the event. `Ctrl/C` events are described in Chapter 5.

1.2.4 Chaining

A further indirect method of initiating a task is task chaining. In this case a timesharing task (the chain task) automatically commences execution when an earlier timesharing task exits. This facility is particularly useful in some applications. Chaining is described in Chapter 6.

NOTE: A CLI is cannot chain.

1.2.5 Communication Between Tasks

Related tasks can send messages to each other. This communication might take place between a subtask and its owner or between a task and its chain task. Intertask communication is described in Chapter 4.

1.3 Types of TCS Macros

The three types of TCS macros are as follows:

- 1 Data definition (assembly time). See Section 1.3.1.
- 2 Data definition (run time). See Section 1.3.2.
- 3 Imperative. See Section 1.3.3.

1.3.1 Data Definition Macros (Assembly Time)

To implement the facilities described in Sections 1.2.1 through 1.2.5, TCS uses several different kinds of data structure. Data definition (assembly time) macros are used for assembly time declaration and definition of TCS data structures.

1.3.2 Data Definition Macros (Run Time)

These macros are similar to data definition (assembly time) macros, except that you use them at run time.

1.3.3 Imperative Macros

Tasks use imperative macros to initiate the TCS facilities.

The TCS macros are described in Chapter 8. Special properties of the different macro types are described in Section 7.6.

2 Writing a CLI

This chapter provides guidelines for writing a CLI.

2.1 The Requirement for User-Written CLIs

IAS comes with the Program Development System (PDS) and the Monitor Console Routine (MCR) CLIs. PDS is the normal user/system interface used with IAS. It is designed for use by computer professionals and programmers. With PDS commands, you can use all the program development and execution facilities of the IAS system.

Specialized applications such as text processing or inventory control might benefit if presented as a user-written CLI. The CLI would normally be allocated to a number of dedicated terminals and would only respond to a small number of commands. In this way non-technical people can have available only those facilities that they need. The application can be presented to them in a manner they understand and they need use only a small amount of documentation.

A typical system plan is shown in Figure 2-1.

2.2 Types of CLI

As shown in Figure 2-2, user-written CLIs can be simple (in terms of using system resources); for example, a BASIC interpreter.

As shown in Figure 2-3, CLIs that use TCS can be more complex in terms of system usage. The CLI can accept commands that cause subtasks to be created.

2.3 Special Facilities Available to a CLI

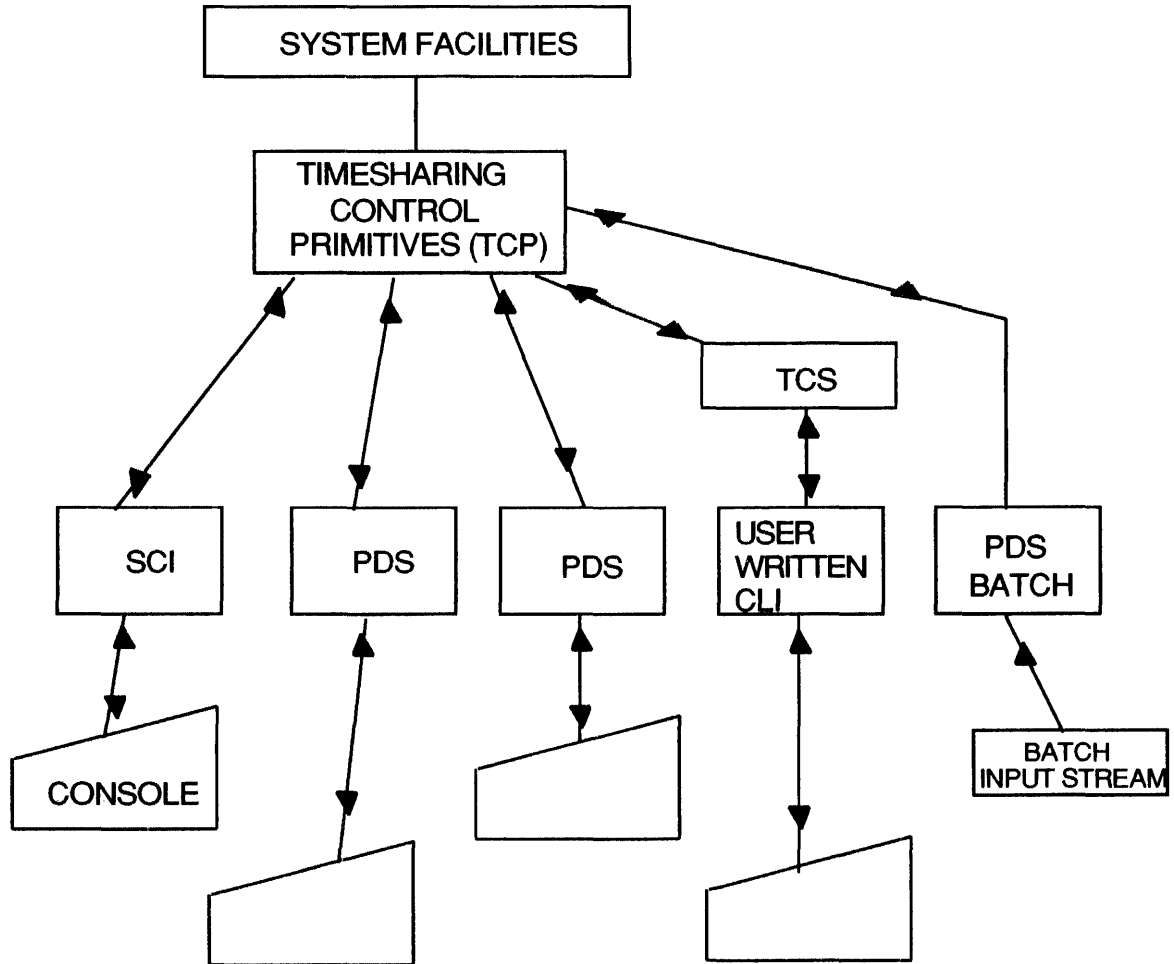
In most respects, any timesharing task can run as a CLI. The main distinguishing feature is that the CLI is the controller (directly or indirectly) of all other tasks running for a particular terminal. Consequently, every active terminal must have a CLI running for it. The functional aspects of the CLI are set at designer discretion (within limits set by the system manager).

Two facilities unavailable to other timesharing tasks are available to CLIs.

- 1 Interface with system shutdown, and
- 2 Dynamic control over the level of multitasking on a particular terminal.

Writing a CLI

Figure 2-1 A Typical System Plan



2.3.1 System Shutdown

A CLI can invoke TCS to check if the system manager has declared shutdown. This is performed using the macro CKEV\$T as follows:

```
CKEV$T SHUT
```

On successful completion, R0 contains -2 if shutdown has been declared and -1 if shutdown has not been declared. In the former case, the CLI can invoke SHUT\$T to interrogate TCS for the number of minutes before shutdown. SHUT\$T returns the number of minutes in R0.

Figure 2-2 A Simple CLI

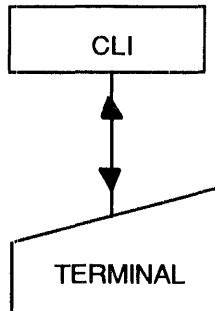
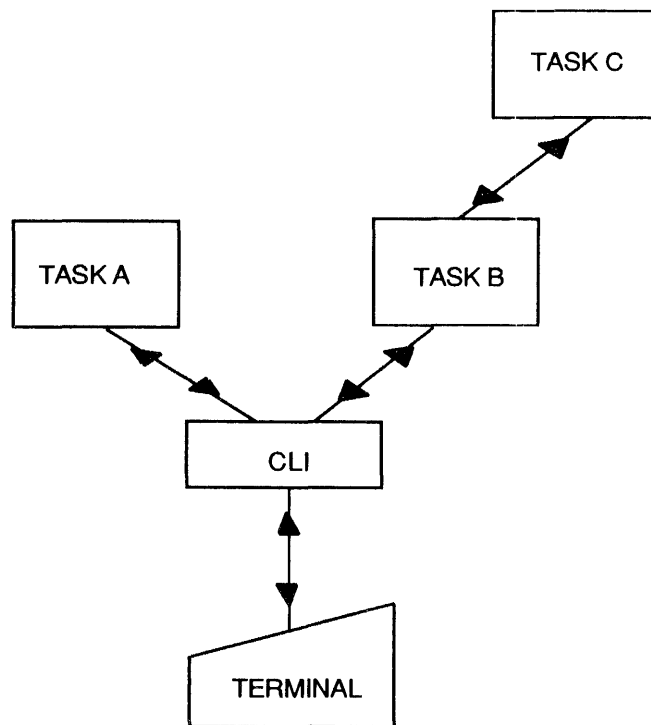


Figure 2-3 A Complex CLI



2.3.2 Limiting Subtasking

The system manager limits the maximum number of timesharing tasks for a terminal when allocating a CLI to that terminal. The default for user-written CLIs is 1. (See the *IAS System Management Guide*.) The CLI itself can use the `JNOD$T` macro dynamically to impose a further limit. The CLI can set the limit to any value between the current usage and the allocation imposed by the system manager.

2.4 Guidelines for Writing and Running CLIs

The following conventions apply to user-written CLIs:

- 1 You can write a CLI in any programming language that IAS supports. For the CLI to use TCS, the language must be MACRO-11 or support the calling of MACRO-11 subroutines (for example, FORTRAN IV-PLUS, COBOL, and CORAL). TCS macros can be called from within MACRO-11 subroutines.
- 2 You can install any suitable task in the system as a CLI. The task must always be built as a multiuser task. (See the *IAS Task Builder Reference Manual*.)
- 3 A CLI must have the appropriate task privilege. (See the *IAS System Management Guide* and Chapters 3 and Chapter 7 of this manual.)
- 4 The CLI must be allocated to each terminal where the CLI is to run. The *IAS System Management Guide* describes how to allocate CLIs.

3 The Creation and Control of Subtasks

A subtask is a timesharing task that has been initiated by another timesharing task. You can run a subtask, for example, to process a number of background jobs in an application.

This chapter describes the relationship between a subtask and the owner task, and the mechanism that TCS uses to support this relationship.

Before a subtask can be initiated, an appropriate task descriptor block (TDB) must exist. The TDB contains all the details concerning the subtask that TCS and the owner task require. The TDB is therefore the interface between the owner task and the subtask. TDBs are described in Section 3.2.

The TCS macros introduced throughout this chapter are fully specified in Chapter 8.

3.1 Control of Subtasks

The subtask owner controls the subtask as follows:

- 1 By initiating the subtask
- 2 By enabling the subtask to read a command line
- 3 By suspending the subtask
- 4 By setting the subtask event flags
- 5 By aborting the subtask
- 6 By checking the subtask status

3.1.1 Initiation of Subtasks

The subtask owner is the task that initiates the subtask. To initiate a subtask, a task uses the `RUN$T` macro. This macro passes the address of a TDB to TCS. For example:

```
RUN$T #TDB1, #BUF1, #LENGTH, #FORM
```

would initiate the subtask described by TDB1. The subtask's name and a command line are contained in the buffer, BUF1. LENGTH is the length of the buffer and FORM is the format of the buffer.

3.1.1.1 Reading the Command Line

A subtask that has been initiated can read its associated command line by using the `GMCR$` directive. This is a system directive and not a TCS macro. (See the *IAS System Directives Reference Manual*.)

The Creation and Control of Subtasks

3.1.2 Suspending Subtasks

A subtask owner can suspend the subtask by means of the SPND\$T macro. For example:

```
SPND$T #TDB1
```

would suspend the subtask TDB1 initiated in Section 3.1.1. Only the owner has the right to suspend a subtask (although a subtask can be automatically suspended by `Ctrl/C`; see Chapter 5).

A suspended subtask can be resumed by means of the RSUM\$T macro. For example:

```
RSUM$T #TDB1
```

would enable a previously suspended subtask to resume executing.

3.1.3 Setting the Subtask's Local Event Flags

A subtask owner can set the subtask's local event flags (see the *IAS Executive Facilities Reference Manual* for a description of event flags). This can be useful for synchronizing the activities of the subtask. For example:

```
SETF$T #TDB1, #2
```

would set local event flag 2 for the subtask initiated.

3.1.4 Aborting the Subtask

A subtask owner can abort the subtask by means of the ABRT\$T macro. For example:

```
ABRT$T #TDB1
```

would abort the subtask indicated (plus any subtasks of that subtask).

The system manager can also abort a subtask by using the SCI command, ABORT/JOB (see the *IAS System Management Guide* for further details). The subtask's job-id (which you can determine by using the SHOW TASKS command) must be specified as a parameter. You can close down a CLI in an orderly way by using the STOP/CLI command; you can abort a CLI by using the ABORT/CLI command.

3.1.5 Checking the State of the Subtask

An event occurs when any of the following things occur:

- Subtask terminates
- Subtask suspends itself
- Subtask sends a message to its owner
- Shutdown is requested
- `Ctrl/C` occurs

The owner can check these events by using the CKEV\$T macro. For example:

```
CKEV$T , #TDB1
```

checks for task related events and sets R0 equal to -1 if no events have occurred, and equal to #TDB1 if an event has occurred.

The task event type can be detected by means of RDEV\$T. For example:

```
RDEV$T
```

causes the type of event that has occurred to be placed in the TDB of the subtask, in addition, the local event flag, which TCS uses to signify the event, is cleared. Section 7.5 details TCS event flag usage.

The status of a subtask can be checked by means of TKST\$T, the information provided is task size and CPU time.

3.1.6 Waiting for Task Completion

On subtask initiation, the owner task can specify that it wants to suspend execution until the subtask causes a task event. The initiation call would be of the following form:

```
RUN$T #TDB1, , , , WAIT
```

or:

```
RUN$T #TDB1, , , , STOP
```

In the first case, the owner task enters a WAITFOR state until the subtask causes a task event, at which point the owner task resumes execution immediately following the RUN\$T. Next, the owner task normally uses RDEV\$T to find the cause of the task event.

In the second case, the owner task enters a STOP state until the subtask causes a task event. The owner task then resumes execution as for the first case.

3.2 Creation of Subtasks

Use the TDBDF\$ macro to reserve the data space required for a TDB. You use the macros TDCM\$A, TDPR\$A, TDTA\$A, and TDEB\$A to insert data into the TDB. These macros define the data at assembly time. The runtime equivalents are TDCM\$R, TDPR\$R and TDTA\$R.

Before a TDB can be used to control a subtask, it must be declared to TCS. TCS then assumes that it can access the TDB whenever TCS is invoked. This feature simplifies the TCS housekeeping procedures. The user program can load data values into the TDB at any time, but cannot initiate a task until its TDB is declared. The TDB declaration is accomplished using the TDBD\$T macro. The TDBR\$T macro is used to release a TDB. TDBR\$T need only be used if the user wishes to prevent TCS accessing the TDB, for example, because TCS is to be invoked from an overlay that is not coresident with the overlay containing the TDB.

3.2.1 Contents of a Task Descriptor Block

The format of a TDB is given in Section A.1. To safeguard user programs against the effect of possible format changes in later versions of TCS, access TDB fields using offset names.

The individual fields are as follows:

- 1 Command Line Section.
- 2 Privilege Section.

The Creation and Control of Subtasks

- 3 Attribute Section.
- 4 Event Information Section.
- 5 Error Status Block Section.
- 6 Internal Fields.

3.2.1.1 Command Line Section

Normally, you set up these fields by using TDCM\$A, TDCM\$R or RUN\$T.

1 T.CMDA

Address of a buffer containing an ASCII command line. When you initiate a subtask, the command line is inspected to find the task (or file) name, and, in some cases, a command line. This field must contain a valid buffer address when you invoke RUN\$T. The command line has four legal formats:

- a. A user file specification preceded by a space; for example:

```
DK0:[200,40]MYTASK.TSK;2
```

The defaults for each field are:

Device—Terminal system device
UFD—Initiating task's UIC
Filename—None, the file name must be specified
Filetype—.TSK
Version—Latest version

The maximum length is 67 characters.

A task invoked in this way is automatically installed (this is also called *auto-installed*) and automatically removed on exit.

- b. A 1- to 3-character name (for example, XYZ) that corresponds to a task installed with the name ...XYZ. This can optionally be followed by a space and a command: for example:

```
PIP FRED.MAC=JIM.MAC
```

The maximum length is 79 characters.

- c. The same format as b. but the installed task name is \$\$\$XYZ.
- d. A 1- to 6-character installed task name.

When a subtask that was initiated using methods b. or c. has a command specified, the subtask must read the command using either GMCR\$ or GCML\$. See either the *IAS System Directives Reference Manual* or the *IAS/RSX-11 I/O Operations Reference Manual*.

2 T.CMDL

Length of the buffer specified in T.CMDA. This field must contain a valid length when invoking RUN\$T.

3 T.TTYP

Used to specify the format of the buffer specified in T.CMDA. When you invoke RUN\$T, this field must contain one of the four values (TS.USE, TS.DOT, TS.DOL, or TS.INS) that correspond to the formats given above.

3.2.1.2 Privilege Section

Normally, you set up these fields by using TDPR\$A or TDPR\$R.

1 T.TP2

Defines which TCP functions the subtask can access. It can contain the logical OR of any of the following:

- PR.RST—Enabled to initiate subtasks.
- PR.CTC—Enabled to be the `[Ctrl/C]` task (see Chapter 5).
- PR.TEV—Enabled to be notified of events.
- PR.CHN—Enabled to chain and send/receive messages.

If the subtask is to have PR.CTC privilege, it should also be given PR.TEV privilege. If a subtask has any of these privileges, it must also be TCP-privileged. Note that a subtask's privileges can never exceed its owner's privileges.

2 T.TP1

Defines four classes of privilege for the subtask. It can contain the logical OR of any of the following:

- JP.PP—Memory management directive privileged.
- JP.PI—TCP privileged.
- JP.PD—Realtime directive privileged.
- JP.PT—Enabled to be an executive privileged task that has been initiated by method a. See the description of T.CMDA in Section 3.2.1.1.

For a definition of these privileges see the *IAS System Management Guide*.

3 T.JNA

Maximum number of related subtasks that can exist simultaneously below the subtask corresponding to this TDB. It can contain any value between 0 and 255.

3.2.1.3 Attribute Section

Normally, you set up these fields by using TDTA\$A or TDTA\$R.

1 T.UIC

UIC under which the subtask is to run. It is stored as a word whose high byte is the group code and whose low byte is the member code. TCS defaults a zero value to the UIC of the initiating task.

2 T.SCHL

Is the scheduling level at which the task is to be initially queued. TCS defaults to scheduling level one.

3 T.FLG1

Can contain flags to give the subtask special attributes. The two possible attributes are as follows:

- FB.BT—The task is to run in the batch scheduling level.
- FB.NC—The task is to be non-auto-suspendable. In other words, the task is not to be automatically suspended when `[Ctrl/C]` is typed on the terminal. This flag has no effect if the owner task itself is auto-suspendable.

The Creation and Control of Subtasks

3.2.1.4 Event Information Section

These fields are described below:

1 T.EVNT

Every time a RDEV\$T is issued for the TDB, this field is loaded with a set of task event indicators which have been collected since the previous RDEV\$T. The field is zeroed by RUN\$T and TDBD\$T. The event indicators are bit flags with the following meanings:

- IF.JS—Successfully terminated.
- IF.JA—Aborted.
- IF.NL—Initiation failure (task aborted).
- IF.SU—Suspended.
- IF.CH—Successfully terminated and successfully chained (possibly more than once).
- IF.SD—Subtask has sent one or more messages.

In addition to containing each of these bit flags singly, T.EVNT can contain combinations.

However, in all cases, if IF.JS, IF.JA or IF.NL is set, the task has terminated. If IF.SU is set the task will be suspended unless it has been explicitly resumed by the owner (using RSUM\$T).

2 T.EVBF

30-byte (decimal) buffer that contains termination information for the subtask. The information is placed into the buffer as soon as TCS has it available (which could be at any time that TCS is invoked). However, the user should not interrogate the buffer until after a RDEV\$T has shown the task has terminated. If RDEV\$T does not show task termination, the contents of the buffer are meaningless.

The following fields exist in the buffer (the offsets are relative to the start of the buffer). The layout is given in Appendix A.

- E.SIZ—Task size in 32-word blocks (decimal).
- E.TIM—Task CPU time (ticks and double-length).
- E.TR—Contains three items:
 - Bit flag EV.ST—Task exited with status
 - Bit flag EV.AB—Task terminated abnormally
 - Low-order byte—Termination code. (These are listed in Appendix B.)
- E.TS—Exit status (only meaningful if EV.ST is set in E.TR).
- E.TPS—Task's PS.
- E.TPC—Task's PC.
- E.TR0—Task's R0.
- E.TR1—Task's R1.
- E.TR2—Task's R2.
- E.TR3—Task's R3.
- E.TR4—Task's R4.
- E.TR5—Task's R5.
- E.TSP—Task's SP.

3.2.1.5 Error Status Block Section

T.TESB

Address of the error status block. It is normally set up using TDEB\$A or TDBD\$T. A zero value means no error codes are to be returned to the user.

3.2.2 Internal Fields

All fields not described in the previous sections are for TCS use only.

3.3 Task Accounting

To account for the subtasks run by a given task, TCS automatically charges the owner for the resources used by its subtasks. TCS assumes that task charging is based on the product of size and CPU time (that is, its total number of core ticks).

If a task of size S1 owns a subtask of size S2, when the subtask terminates the total CPU time is T2. TCS then adds the following quantity to the owner task's current CPU, time: $T2 \times (S2/S1)$ giving a total of $T1 + (T2 \times (S2/S1))$. The owner's total CPU usage calculated from the product of size and time is:

$$\begin{aligned} & S1 \times (T1 + T2 \times (S2/S1)) \\ &= (S1 \times T1) + (S2 \times T2) \end{aligned}$$

This is the total utilization of the owner and its subtask. This enables a CLI to account not only for each of its subtasks as they terminate but also to account automatically for all of their subtasks and descendents.

The description of macro TKST\$T in Chapter 8 describes how to obtain accounting statistics.

4

Communication Between Tasks

In a system of subtasks, when a task requires a set of results from another task the task can transmit this data as a message.

4.1 Setting up the Message

Before a message can be sent and received successfully, the following must occur:

- The characters that compose the message must exist in a message buffer.
- A send data block (SDB) must exist to describe the message.

First, use the `SDBDF$` macro at assembly time to reserve space for the SDB. You can also use this macro to specify the contents of the SDB. You can specify the contents of the SDB at run time by using `SDDF$R` (a data definition macro) or by using `SEND$T` OR `RCV$T` (see Section 4.2).

4.1.1 Contents of a Send/Receive Data Block (SDB)

The layout of an SDB is given in Section A.3. The layout shows fixed positions of fields within an SDB. However, these fields should always be accessed via offset names. This protects the user program against possible changes in later versions of TCS.

The send/receive data block contains the following fields:

1 T.SNDA

This is the address of a send/receive buffer, which comprises a one-word message-length in characters followed by the message itself. A message can be 0-253 (decimal) characters long. When a message is sent, TCS determines the message length from this field. When a message is received, TCS places the number of characters received into this field. The message buffer must be on a word boundary.

2 T.SNDL

This is the total length in characters of the send/receive buffer. This field is used only by `RCV$T`, to prevent buffer overrun. If a received message is too long for the buffer, a warning of `PS.SUC/PS.RBS` will be returned. The buffer should never be less than two bytes. This field is ignored by `SEND$T`.

3 T.SESB

Is the address of the error status block. A zero-value indicates that no error codes are to be returned to the user.

4 T.SNDT

Describes which task is to receive a send message or which user has sent a received message. In the first case, the field should contain one of the following:

- -1—Message to be sent to owner.
- 0—Message to be sent to successor.
- Other—TDB address of a subtask that is to receive the message.

Communication Between Tasks

If the send message is being made via a RUN\$T or CHN\$T, this field is ignored and need not be initialized.

In the case of message receives, RCV\$T will place one of the following into T.SNDT:

- -2—Message sent by an unidentifiable subtask. This occurs because the TDB has been reused or released.
- -1—Message sent by owner.
- 0—Message sent by predecessor.
- Other—TDB address of a subtask that sent the message.

All fields not described above are internal fields for TCS use only.

4.2 Sending Messages

- 1 A message can be sent to a subtask (by its owner) when it is initiated. To do this, include an SDB address as parameter to the RUN\$T macro. For example:

```
RUN$T #TDB1, , , #SDB1
```

would place the subtask described by TDB1 onto one of the scheduler queues. Also, a message described by the SDB would be queued to be sent to the subtask. When the subtask starts to execute, the message becomes available. The name of the recipient is located from the TDB. It is in the command buffer pointed to by T.CMDA. T.SNDT, in the SDB, is ignored. RUN\$T is described in Chapter 8.

- 2 A message can be sent by an owner task to a subtask at any time. To do this, use the SEND\$T macro. For example:

```
SEND$T #SDB1, #TDB1
```

would send a message described by SDB1 to the subtask described by TDB1. SEND\$T is described in Chapter 8.

- 3 A message can be sent to a chain task by its predecessor. To do this, use CHN\$T or SEND\$T. For example:

```
CHN$T #CTDB1, , , #SDB2
```

would declare a chain task described by CTDB1. Also, a message described by SDB2 would be queued to be sent to the chain task.

The name of the recipient is located from the chain descriptor block. It is in the command buffer pointed to by C.CHNA. T.SNDT, in the SDB, is ignored.

```
SEND$T #SDB2, #0
```

would queue a message described by SDB2 to be sent to the chain task of the current task.

- 4 A subtask can send a message to its owner by using SEND\$T. For example:

```
SEND$T #SDB3, #-1
```

would send a message described by SDB3 to the owner of the current task.

4.3 Receiving Messages

Messages are always received by means of the RCV\$T macro. For example:

```
RCV$T #SDB4
```

would attempt to receive a message described by SDB4. RCV\$T is described in Chapter 8.

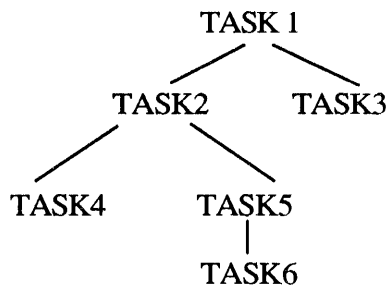
5

Ctrl/C Events

A `Ctrl/C` event occurs when you type `Ctrl/C` at a terminal. TCS notifies the appropriate task when a `Ctrl/C` event occurs. The TCS facility enables tasks to be interrupted by the terminal user.

The task that is notified of `Ctrl/C` events is called the “`Ctrl/C` task” for that terminal. Once the system manager has assigned the CLI to a terminal, it is initially the `Ctrl/C` task. The CLI can then pass this attribute to one of its subtasks, which can then pass the attribute to one of its own subtasks, and so on. When a `Ctrl/C` event occurs, the descendants of the `Ctrl/C` task can be automatically suspended. These tasks can then be resumed by the `Ctrl/C` task. Figure 5–1 shows this task hierarchy for a particular terminal.

Figure 5–1 A Task Hierarchy



As illustrated in Figure 5–1, TASK1 is the CLI for the terminal. The following situation now exists:

- 1 TASK1 has initiated subtasks TASK2 and TASK3.
- 2 TASK2 has initiated subtasks TASK4 and TASK5.
- 3 TASK5 has initiated TASK6.
- 4 TASK2 has claimed the `Ctrl/C` attribute.

When you type `Ctrl/C`, three actions are taken:

- 1 An internal flag is set and TASK2 can check the state of this flag, by means of the `CKEV$T` macro. For example:

```
CKEV$T TERM, , WAIT
```

would stop execution of the task until a `Ctrl/C` event occurred.

- 2 TASK4, TASK5, and TASK6 are automatically suspended unless they are marked “non-auto-suspendable.” This attribute is conferred on a task when it is initiated. The default is auto-suspendable.

Ctrl/C Events

- 3 If TASK2 specified a **Ctrl/C** AST entry point when it claimed the **Ctrl/C** attribute (CTC\$T macro), the AST routine is executed.

When **Ctrl/C** is detected, TASK2 takes any required action; this might include resuming all the auto-suspended tasks using the RSAS\$T macro. RSAS\$T sets the auto-suspended tasks back to the state they were in before **Ctrl/C** occurred (unless some other change of state has occurred (for example, TASK2 aborting TASK4 between **Ctrl/C** being typed and RSAS\$T being invoked). To use RSAS\$T, a task must have the same privileges as for subtasking. In addition:

- 1 **Ctrl/C** is completely transparent to TASK1 and TASK3.
- 2 Checking for **Ctrl/C** automatically resets the **Ctrl/C** flag, which enables further **Ctrl/C** to be detected.
- 3 If TASK2 does not detect a **Ctrl/C** and then another task becomes the **Ctrl/C** task, the original **Ctrl/C** is not detected by the new **Ctrl/C** task.

When a terminal becomes active, the CLI is automatically the **Ctrl/C** task. To designate any other task than the **Ctrl/C** task, invoke the CTC\$T macro with CLAIM as the first parameter.

The conditions for success are as follows:

- 1 The task owner must be the current **Ctrl/C** task.
- 2 The task must be TCP-privileged and have the privilege PR.CTC.

A task relinquishes the **Ctrl/C** attribute by either invoking CTC\$T with RELINQ as the first parameter or by exiting. In either case, the owner reverts to being the **Ctrl/C** task.

If a CLI is the current **Ctrl/C** task and it claims or relinquishes **Ctrl/C**, the effect is a null operation.

To check for **Ctrl/C** events, a task invokes the CKEV\$T macro with TERM as the first parameter. Ignoring task events, this macro returns with R0 set to -1 if **Ctrl/C** has not occurred and R0 set to 0 if **Ctrl/C** has occurred. An additional parameter specifies whether the task wishes to wait or stop until a **Ctrl/C** occurs. The call would be:

```
CKEV$T TERM, , WAIT
```

or

```
CKEV$T TERM, , STOP
```

To use this macro successfully, the task must be TCP-privileged and have the privilege PR.TEV.

6 Chaining

Chaining is the process of passing control to another task (a chain or successor task) from the current task when it terminates. Chaining minimizes the overhead on system resources. A task declares that a chain or successor task is to be initiated when it (the predecessor task) terminates. The chain task reuses the timesharing data structures that were previously being used by the predecessor task and has the same privileges and attributes. Chaining occurs automatically when the predecessor task terminates successfully. An event notifies the owner task that chaining has occurred.

NOTE: A CLI cannot chain.

6.1 TCS Interface

Before a chain task can exist, a chain task descriptor block (CTDB) must be set up. Use the CDBDF\$ macro at assembly time to reserve data space for the CTDB. You can also use the CDBDF\$ macro, the CDDF\$R macro, or CHN\$T to define the contents of the CTDB.

A task uses the CHN\$T macro to declare its chain task to TCS, as follows:

```
CHN$T #CTDB1
```

This would declare a chain task with CTDB1.

To chain successfully, a task must be TCP-privileged and have the privilege PR.CHN.

6.2 Contents of a Chain Task Descriptor Block

The layout of a CTDB is given in Section A.2. Although absolute positions of fields within a CTDB are shown, they are variable. Always access fields via offset names. Normally, you use either CDBDF\$, CDDF\$R, or CHN\$T to set up the fields.

The individual fields are used as follows:

| Field | Description |
|--------|------------------------------------------------------------------------------------------------------|
| T.CHNA | Address of a buffer containing an ASCII command line. |
| T.CHNL | Length of the buffer given in T.CHNA. |
| T.CTYP | Format of the buffer (TS.USE,TS.DOT,TS.DOL or TS.INS). |
| T.CESB | Address of the error status block. A zero value means no error codes are to be returned to the user. |

The first three fields are used in an identical fashion to T.CMDA, T.CMDL, and T.TTYP in TDBs (see Chapter 3). All fields not described above are internal fields for TCS use only.

NOTE: If more than one task is likely to chain to the same successor task concurrently, do not use the TS.USE format of the command line. Instead, pre-INSTALL the successor task and use the TS.DOL,TS.DOT, or TS.INS format. If you do not, the autoinstall for one copy of the successor task might fail if the autoinstall for another copy is in progress.

7

Using the Timesharing Control Services (TCS)

This chapter describes the properties and features of TCS that you should understand before you use the TCS macros.

7.1 Including Timesharing Control Services in a Program

All TCS macros are defined in the system macro library RSXMAC.SML. The TCS subroutines are included in the system object module library SYSLIB.OLB. All symbols used in the macros (except offset names) are defined in SYSLIB.OLB. Both of these files are normally available during assembly and task building of the user program.

You must use the `.MCALL` directive to include each TCS macro to be used in a MACRO program. (See the *IAS/RSX-11 MACRO-11 Reference Manual*. You can use the TCS macro `TCSMC$` to accomplish this. `TCSMC$` causes a `.MCALL` to be issued for each TCS macro (except itself). Consequently, the following sequence is required at the beginning of a MACRO program:

```
.
.
.
.
.MCALL TCSMC$
TCSMC$
.
.
.
```

In general, the LUN required for communication with the Timesharing Control Primitives (TCP) is automatically set up by the task builder when any of the runtime macros are called. However, if all runtime TCS macros are issued in an overlay segment then the SYSLIB module `PILUN` must be included in the root of the task in order for `.PILUN` to be correctly assigned.

For example:

```
.ROOT MAIN-LB:[1,1]SYSLIB/LB:PILUN-*(OVERLAY)
```

7.2 Initialization of Timesharing Control Services

Initialization of TCS in a program involves the clearing of all TCS data structures. This process occurs automatically when the task is built. Consequently, at the start of the task there is no need for explicit initialization. However, when you restart a program, you should initialize TCS. Use the `TINIT$` macro (see Chapter 8) to perform initialization. For example:

```
TINIT$ #EBLCK1, #ERRH
```

would initialize TCS where `EBLCK1` is the name of an error status block and `ERRH` is an error handling routine, see Section 7.3.

Using the Timesharing Control Services (TCS)

7.3 Privilege

Tasks that use TCS need the following privileges:

| Task | Privilege | Explanation |
|-------|----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| JP.PI | TCP privilege. | TCS indirectly uses TCP facilities; therefore, any task that uses TCS must have this privilege. Use of individual TCS facilities is further controlled by additional privileges (prefixed PR.) |
| JP.PD | Real-time directive privilege. | Certain directives can be used only by tasks with this privilege. These directives are defined in the <i>IAS System Directives Reference Manual</i> . |
| JP.PT | Executive privilege. | This is defined in the <i>IAS Executive Facilities Reference Manual</i> . Any task that is auto-installed and built as executive privileged must have this privilege. |
| JP.PP | Memory management directive privilege. | Memory management directives can be used only by tasks with this privilege. |

The following privileges are required for controlling individual TCS facilities.

| Task | Privilege | Explanation |
|--------|---------------------------------|-------------------------------------------------------------------------------------|
| PR.RST | Subtasking. | A task needs this privilege to perform and subtask initiation or control functions. |
| PR.CTC | Ctrl/C task. | A task needs this privilege to be the Ctrl/C task. |
| PR.TEV | Event occurrence. | A task needs this privilege to be notified of events. |
| PR.CHN | Chaining and Sending/Receiving. | A task needs this privilege to declare a chain task or to send or receive messages. |

7.3.1 Allocation of Privilege

A CLI is given privileges at install time (see the *IAS System Management Guide*). The privileges are specified as a 20-bit mask. The correspondence between these privileges and the bits in the privilege mask is as follows:

| | |
|--------|---------|
| JP.PI | 0200000 |
| JP.PD | 0400000 |
| JP.PT | 1000000 |
| JP.PP | 2000000 |
| PR.RST | 0000222 |
| PR.CTC | 0040000 |
| PR.TEV | 0000200 |
| PR.CHN | 0000400 |

A subtask is given privileges by its owner at subtask initiation (see Section 7.3). An owner cannot confer privileges that it does not have itself.

7.3.2 Privileged Terminals

When a CLI begins execution on a terminal, the terminal is marked as nonprivileged, unless the CLI was installed with bit 0 set in its ATTRIBUTES mask. See the *IAS System Management Guide*.

7.4 Error Handling

Errors that occur during the execution of imperative macro expansions are returned from TCS in an error status block (ESB). Runtime data definition macros never return errors. TCS does not deal with errors that might be caused by assembly time macros. They are flagged in the normal way by the MACRO-11 assembler.

All imperative macros have an associated ESB. Macros that use a task descriptor block or a chain task descriptor block use the ESB address contained in the descriptor block. For all other imperative macros, you can specify the ESB as a parameter to the macro call. The format of the error status block is shown in Table 7-1.

Table 7-1 Error Status Block

| Byte Name | Byte Value | C Bit |
|-----------|-----------------|--------|
| T.ERR | Result code | 0 or 1 |
| T.ERR+1 | 0 or -1 | |
| T.ERR+2 | Error qualifier | |

On exit from the code generated by a TCS macro, the following indicate the result:

- 1 The carry bit in the Processor Status word. If the carry bit is set to one an error has occurred during the execution of the macro code. If the carry bit is clear the macro code has completed execution successfully.
- 2 T.ERR in the error status block contains a result code that indicates either that the macro code has completed successfully (PS.SUC) or one of the following error conditions has occurred:
 - Bad parameters, PE.BAD
 - Operation aborted, PE.ABO
 - Privilege violation, PE.PRI
 - Job node pool exhausted, PE.UPN.
- 3 T.ERR+1 contains either zero, which indicates errors for TCP or TCS (in this case a result code will be given in T.ERR), or -1, which indicates an error in system directive usage. (In practice the only error of this type is "handler not resident," which occurs when a task using TCS is run from MCR or in a non-timesharing system.)
- 4 T.ERR+2 might contain an error or success qualifier, depending on the particular macro that has been called. Qualifiers for each macro are given with each macro description in Chapter 8.

Using the Timesharing Control Services (TCS)

7.4.1 Error Handling Subroutines

Each imperative macro call enables an error handling subroutine to be specified. Such a subroutine must be written by the user. If specified, it is automatically entered on exit from the macro code if the carry bit is set to one.

If an error handling subroutine is specified, it has the effect of generating the following code at the end of the macro expansion:

```
BCC .+6
JSR PC,err
```

where `err` is the name given as the TCS macro parameter.

7.5 Event Flag Usage

TCS uses local event flags 2, 3, and 25 (decimal) to signify `Ctrl/C`, shutdown, and task events, respectively. Therefore, user programs that use TCS should not change the state of any of these flags.

7.6 Macro Characteristics

This section describes the conventions and characteristics of each class of macro. The three types of TCS macro are as follows:

- 1 Imperative macros—Used by tasks for performing TCS operations.
- 2 Data Definition (assembly time) macros—Used for assembly time declaration and definition of TCS data structures.
- 3 Data Definition (run time) macros—Similar to data definition (assembly time), but used at run time.

7.6.1 Imperative Macros

Imperative macros are used by tasks for performing TCS operations. Their parameters and register usage are described in this section.

7.6.1.1 Parameters

Unless otherwise stated, parameters are optional. The defaults are as follows:

- 1 For TDBs the default is R0.
- 2 For quantities that can be obtained from the data structure, the default is the existing contents.
- 3 If no error status block is defined, none will be assumed.
- 4 If no error handling subroutine is defined, none will be assumed.
- 5 Unless otherwise stated, all parameters must be of a form that can be directly substituted into the following:
 - a. MOV param,destination
 - b. MOVB param,destination

where param represents the parameter exactly as specified by the caller.

7.6.1.2 Register Usage

All registers except R0 are saved and restored by the macros. R0 is used in one of two ways:

- 1 Those macros that specifically relate to a data structure such as a TDB or SDB always have the address of this structure as the first parameter. If this parameter is omitted, TCS assumes that the address is in R0. In any event, on exit from the macro, R0 will contain the address.
- 2 Some macros need to return a word of information; R0 is always used for this purpose.

7.6.2 Data Definition Macros (Assembly Time)

Data definition macros (assembly time) are used for assembly time declaration and definition of TCS data structures. Their parameters and the use of offset definitions are described in this section.

7.6.2.1 Parameters

Parameters are optional (unless otherwise stated). An omitted parameter means that the corresponding field of the data structure will be untouched. This means that the field will contain nothing unless its contents are (or have been) changed by some other means. All parameters must be of a form that can be directly substituted into the following:

- 1 .WORD param
- 2 .BYTE param

where param represents the parameter exactly as specified by the caller.

7.6.2.2 Offset Definitions

To define the offset names for TCS data structures, use the macros TCOFF\$, TDOFF\$, TSOFF\$, and TEOFF\$. Any TCS data definition macros that need offset names invoke these macros internally to define the offset names as local symbols. Once invoked, the macros redefine themselves to null. The effect of this is that a user program rarely has to invoke these macros explicitly. The exception is when a module tries to access TCS data structures that are defined in another module. Also, if you want to define the offsets as global names, invoke the above offset definition macros with a DEF\$G parameter and before any TCS definition macros are invoked.

7.6.3 Data Definition Macros (Run Time)

Data definition macros (run time) are used for run time declaration and definition of TCS data structures. Their register usage, error handling procedures, and parameters are described in this section.

7.6.3.1 Register Usage

All registers except R0 are saved and restored by the macros. Each macro has, as its first parameter, the address of the data structure where it is to operate. If this parameter is omitted, TCS assumes that the address is in R0. In any event, on exit from the macro, R0 will contain the address.

Using the Timesharing Control Services (TCS)

7.6.3.2 Error Handling

None of these macros causes errors in use. The value of the C-bit is undefined on exit.

7.6.3.3 Parameters

All parameters are optional. If any parameter (other than the first) is omitted, the corresponding field in the data structure is untouched.

All parameters must be of a form that can be directly substituted into the following:

- 1 MOV param,destination
- 2 MOVB param,destination

where param represents the parameter exactly as specified by the caller.

8 TCS Macro Descriptions

This chapter contains a description of each TCS macro. The general format of each description is:

- 1 Macro name
- 2 Type
- 3 Function
- 4 Macro call
- 5 Parameter descriptions
- 6 Implementation notes
- 7 Return data
- 8 Result codes

If any of these sections do not apply to a macro, it is omitted from the description.

ABRT\$T

ABRT\$T

TYPE

Imperative.

FUNCTION

Used by a task to abort a subtask.

MACRO CALL

ABRT\$T *tdb, err*

parameter definitions

tdb

Address of the subtask's task descriptor block. Default: Contents of R0.

err

Address of an error handling subroutine. This is entered if TCS detects an error while executing this macro code. Default: Error handling routine is not entered.

IMPLEMENTATION

NOTES

- 1 On exit from the macro code, use the RDEV\$T macro to obtain information about the aborted task.
- 2 Any subtasks of an aborted subtask are also aborted.
- 3 To use this macro, the task must be TCP-privileged and have the PR.RST TCS privilege.
- 4 The invoking task enters a STOP state until the aborted subtask exits.

RETURN DATA

R0 contains the address of the aborted subtask's task descriptor block.

RESULT CODES

General result codes are given in Appendix B.

The meaning of the following result code relates only to this macro.

| | | | |
|-----------|--------|---------|------------------|
| Bytename: | T.ERR | T.ERR+2 | Meaning |
| Value: | PE.ABO | PE.NTA | Task not active. |

CDBDF\$

CDBDF\$

TYPE

Data definition—assembly time.

FUNCTION

To reserve data space for a chain task descriptor block and optionally to specify the contents of the block.

MACRO CALL

CDBDF\$ *cmda,cmdl,cmdt,esb*

parameter definitions

cmda

Address of a command buffer containing the name of the chain task and a command line. This value is placed into the chain task descriptor block at offset T.CHNA. Default: T.CHNA is left blank.

cmdl

Length, in bytes, of the command buffer specified. This value is placed in the chain task descriptor block at offset T.CHNL. Default: T.CHNL is left blank.

cmdt

Buffer format. This is either TS.USE, TS.DOT, TS.DOL, or TS.INS. Value specified is placed in the chain task descriptor block at offset T.CTYP. Default: T.CTYP is left blank.

esb

Address of an error status block to receive result codes that occur when you use this chain task descriptor block. This value is placed into the chain task descriptor block at offset T.CESB. Default: T.CESB is left blank.

IMPLEMENTATION

NOTES

- 1 Contents of the chain task descriptor block can also be defined by means of CDDF\$R or CHN\$T.
- 2 Layout the block is given in Section A.2.

CDDF\$R

TYPE

Data definition—run time.

FUNCTION

To specify the contents of a chain task descriptor block.

MACRO CALL

CDDF\$R *ctdb,cmda,cmdl,cmdt,esb*

parameter definitions

ctdb

Address of a chain task descriptor block. Default: Contents of R0.

cmda

Address of a buffer that contains the task name and a command line. This address is copied into the chain task descriptor block at offset T.CHNA. Default: Contents of T.CHNA is unchanged.

cmdl

Length, in bytes, of the buffer specified in *cmda*. This value is copied into the chain task descriptor block at offset T.CHNL. Default: Contents of T.CHNL is unchanged.

cmdt

Format of the buffer. This is either TS.USE, TS.DOT, TS.DOL, or TS.INS. Default: Contents of T.CTYP is unchanged.

esb

Address of an error status block to receive result codes that occur when you use this chain task descriptor block. This value is copied into the chain task descriptor block at offset T.CESB. Default: Contents of T.CESB is unchanged.

IMPLEMENTATION

NOTES

- 1 Contents of the chain task descriptor block can also be specified by means of CDBDF\$ and CHN\$T.
- 2 Layout of the block is described in Section A.2.

CHN\$T

CHN\$T

TYPE

Imperative.

FUNCTION

Used by a task to declare a chain (successor) task and, optionally, to send a message to that task.

MACRO CALL

CHN\$T *ctdb,cmda,cmdl,cmdt,sdb,esb,err*

parameter definitions

ctdb

Address of the chain task descriptor block for the chain task. Default: Contents of R0.

cmda

Address of a buffer (the command buffer) that contains the name of the chain task and a command line. This address is copied into the chain task descriptor block at offset T.CHNA. Default: The existing contents of T.CHNA.

cmdl

Length of the command buffer in bytes. This value is copied into the chain task descriptor block at offset T.CHNL. Default: The existing contents of T.CHNL.

cmdt

Value that indicates the format of the command buffer. This is either TS.USE, TS.DOT, TS.DOL, or TS.INS.

sdb

Address of a send/receive data block that describes a message to be sent to the chain task. Default: No message is sent.

esb

Address of an error status block to receive result codes that occur when the task uses this chain task descriptor block. This address is copied into the chain task descriptor block at offset T.CESB. Default: Existing contents of T.CESB.

err

Address of an error handling subroutine. This is entered if errors occur during the macro code execution. Default: None.

IMPLEMENTATION NOTES

- 1 A task can issue this macro an indefinite number of times. Each call supersedes the previous one. However, send messages are not removed from the queue.
- 2 The layout of the chain task descriptor block is given in Section A.2.
- 3 T.SNDT, in the SDB, is ignored. The chain task name is the command buffer.
- 4 Sending data uses SCOM nodes for the data.
- 5 To use this macro, the task must be TCP-privileged and have the TCS privilege PR.CHN.

RETURN DATA

R0 contains the address of the chain task descriptor block.

RESULT CODES

General result codes are given in Appendix B.

The following result codes relate only to this macro.

| Bytename: | T.ERR | T.ERR+2 | Meaning |
|-----------|--------|---------|-------------------------------------------------------------------------------------------------------------------------------------|
| Value: | PE.ABO | PE.TSE | Error in command line; for example, zero-length or space-filled. |
| | PE.ABO | PE.BUF | No command line specified or command too long (>79 characters for system library tasks and >67 characters for user filename tasks). |
| | PE.ABO | PE.ADR | Command line not in user's address space. |
| | PE.ABO | PE.ILL | Illegal command line format type. |
| | PE.ABO | PE.TNI | Nonexistent system or library task. |
| | PE.ABO | PE.UPN | SCOM node pool exhausted. |
| | PE.ABO | PE.IBS | Send message >253 (decimal) characters. |
| | PE.ABO | PE.BUF | Unavailable buffer space. |
| | PE.ABO | PE.ADR | Send message not all in user's address space. |

CKEV\$T

CKEV\$T

TYPE

Imperative.

FUNCTION

- 1 Check for task events or cause the invoking task to wait or stop until a task event occurs.
- 2 Check for a `Ctrl/C` event or cause the invoking task to wait or stop until a `Ctrl/C` event occurs.
- 3 Check for a shutdown event or cause the invoking task to wait or stop until a shutdown event occurs.

MACRO CALL

CKEV\$T *trm, tasks, wt, esb, err*

parameter definitions

trm

Indicates that the invoking task wants to be informed of `Ctrl/C` and/or shutdown events. Only CLIs are informed of a shutdown event.

| Parameter Value | Action Taken |
|-----------------|--------------------------------------------------------------------------------------|
| SHUT | The task is to be informed if a shutdown event occurs. |
| TERM | The task is to be informed if a <code>Ctrl/C</code> (terminal) event occurs. |
| <SHUT,TERM> | The task is to be informed if either a shutdown or <code>Ctrl/C</code> event occurs. |

Default: `Ctrl/C` events and shutdown events are not checked.

tasks

A list of task descriptor block addresses that indicate subtasks that are to be checked for task events. The list is of the form:

<address 1, address2, ..., address n>

Default: Subtask events are not checked.

wt

Indicates whether to stop or wait until an event, as specified in the *trm* and *task* parameters, occurs.

| Parameter Value | Action Taken |
|-----------------|----------------------------------|
| STOP | The host task stops-until-event. |
| WAIT | The host task waits-until-event. |

Default: Task does not stop or wait.

esb

Address of an error status block to be used for storing result codes detected by the code generated by this macro. Default: Result codes are not returned.

err

Address of an error handling subroutine. This is entered if errors occur during the macro code execution. Default: None.

IMPLEMENTATION NOTES

- 1 Subtask addresses must not be specified as R0 but any other register can be used.
- 2 trm and tasks parameters must not both be blank.
- 3 If 1 or 2 are contravened, an assembly error is given.
- 4 Events are checked in the order shutdown, `[Ctrl/C]`, and task.
- 5 Only one event is returned at a time.
- 6 Minutes before shutdown can be obtained by using SHUT\$T.
- 7 Only CLIs are informed of shutdown events. If any other task specifies SHUT in the first parameter, this has no effect.
- 8 To use this macro, the task must be TCP-privileged and have the TCS privilege PR.TEV.

RETURN DATA

R0 contains information as follows:

| Value | Meaning |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| -2 | A shutdown event has been detected. |
| -1 | No events have been detected. |
| 0 | A <code>[Ctrl/C]</code> event has been detected. |
| address | Address of the task descriptor block that relates to a subtask that has caused a task event. The type of task event can be determined by using RDEV\$T. |

RESULT CODES

CKEV\$T

These are given in Appendix B.

CTC\$T

TYPE

Imperative.

FUNCTION

Used by a task either to claim or relinquish the attribute of being the `Ctrl/C` task.

MACRO CALL

CTC\$T *clm, esb, err, ast*

parameter definitions

clm

Indicates whether to claim or relinquish the attribute.

Parameter

| Value | Action Taken |
|--------|-----------------------------------------------|
| CLAIM | The attribute is passed to the invoking task. |
| RELINQ | The attribute is passed to the owner task. |

Default: This parameter is mandatory. The macro causes an assembly error if it is omitted.

esb

The address of an error status block (to receive result codes from the code generated by the macro).

err

The address of an error handling subroutine. This is entered if errors occur during the execution of the macro code. Default: None.

ast

The address of an AST routine within the program that is entered whenever a `Ctrl/C` is typed on the terminal. It is valid only when parameter *clm* is set to CLAIM. Default: Any existing AST entry point is canceled.

IMPLEMENTATION

NOTES

- 1 If a CLI attempts to claim the attribute of being the `Ctrl/C` task while it is already the `Ctrl/C` task, the macro acts only on the *ast* parameter. If the parameter is supplied, the AST entry point is declared. If not, any existing AST entry point is canceled.

CTC\$T

- 2 If the task attempting to relinquish the attribute is a CLI that is the `Ctrl/C` task, the macro code takes no action and exits successfully.
- 3 If the `ast` parameter is specified for a request to relinquish the attribute of being the `Ctrl/C` task, an assembly error results.
- 4 To use this macro, the task must be TCP-privileged and have the TCS privilege `PR.CTC`.

RETURN DATA

None.

RESULT CODES

General result codes are given in Appendix B.

The following result codes relate only to this macro.

| Bytename: | T.ERR | T.ERR+2 | Meaning |
|-----------|--------|---------|------------------------------------------------------------|
| Value: | PE.ABO | PE.CTC | On claim, owner is not the <code>Ctrl/C</code> task. |
| | PE.ABO | PE.ILL | On relinquish, caller is not the <code>Ctrl/C</code> task. |

ESBDF\$

TYPE

Data definition—assembly time.

FUNCTION

Reserve data space for an error status block.

MACRO CALL

ESBDF\$

**parameter
definitions**

No parameters.

IMPLEMENTATION

NOTES

The offset names for error status blocks are automatically defined locally by this macro.

JNOD\$T

JNOD\$T

TYPE

Imperative.

FUNCTION

Used by a CLI to set a limit on the total number of concurrently active timesharing tasks (including the CLI) that can be initiated from the terminal where the CLI is allocated.

MACRO CALL

JNOD\$T *alloc,esb,err*

parameter definitions

alloc

The limit to be set. Default: This parameter is mandatory. If the parameter is not given, the macro causes an assembly error.

esb

Address of an error status block to be used for returning result codes. Default: Result codes are not returned.

err

Address of an error subroutine. This is entered if an error occurs. Default: None.

IMPLEMENTATION

NOTES

- 1 This macro is available only to CLIs.
- 2 The system manager has overall control by setting a limit when allocating a CLI to a terminal (the ALLOCATE/TERMINAL command). The CLI can decrease but not increase this limit.
- 3 A CLI is given an initial allocation of 1 when it begins execution.
- 4 To use this macro, the CLI must be TCP-privileged and have the TCS privilege PR.RST.

RETURN DATA

R0 contains the limit set.

RESULT CODES

General result codes are given in Appendix B.

The following result codes relate only to this macro.

| Bytename: | T.ERR | T.ERR+2 | Meaning |
|-----------|--------|---------|------------------------------------------------------------------------------------------------------------------------------|
| Value: | PS.SUC | PS.JAE | Attempt to set allocation limit above the system-manager-imposed limit. The allocation is set to the system manager's limit. |
| | PS.SUC | PS.NOE | Attempt to set allocation limit to less than current allocation. The allocation limit is unchanged. |
| | PE.ABO | PE.PRI | Attempt by a task that is not a CLI to change its allocation limit. |

RCV\$T

RCV\$T

TYPE

Imperative.

FUNCTION

Used by a task to receive a message from a related task; that is, the owner, a subtask, or a predecessor task.

MACRO CALL

RCV\$T *sdb, buff, length, esb, err*

parameter definitions

sdb

Address of a send/receive data block. Default: Contents of R0.

buff

Address of a buffer where the message is to be placed. This value is copied into the send/receive data block at offset T.SNDA. Default: Existing contents of T.SNDA.

length

Length in bytes of the buffer specified by *buff*. This parameter is copied into the send/receive data block at offset, T.SNDL. Default: The existing contents of T.SNDL.

esb

Address of an error status block to be used for result codes relating to the use of this send/receive block. This parameter is copied into the send/receive data block at offset, T.SESB. Default: Existing contents of T.SESB.

err

Address of an error subroutine. This is entered if the macro fails. Default: None.

IMPLEMENTATION

NOTES

- 1 The message to be received could have been sent by a subtask, the owner task, or the predecessor task. The sender can be identified by the contents of the send/receive data block at offset T.SNDT as follows:

- 2 A subtask whose TDB has been released or reused and cannot, therefore, be identified.
 - 1 The owner task.
 - 0 The predecessor task.
 - address Address of the task descriptor block of the sending subtask.
- 2 Form of send/receive data blocks is given in Section A.3.
 - 3 To use this macro, the task must be TCP-privileged and have the TCS privileges PR.CHN and PR.TEV.

RETURN DATA

As above.

RESULT CODES

General result codes are given in Appendix B.

The following result codes relate only to this macro.

| Bytename: | T.ERR | T.ERR+2 | Meaning |
|-----------|--------|---------|----------------------------------------------------|
| Value: | PE.ABO | PE.ILL | Buffer length >2 bytes. |
| | PE.ABO | PE.ADR | Buffer not all in user's space. |
| | PE.ABO | PE.NOD | No messages queued. |
| | PS.SUC | PS.RBS | Buffer too short—message truncated. |
| | PS.SUC | PS.TDB | Sending subtask's TDB has been released or reused. |

RDEV\$T

RDEV\$T

TYPE

Imperative.

FUNCTION

Used by a task when one of its subtasks has caused a task event to read information concerning that task event.

MACRO CALL

RDEV\$T *tdb, err*

parameter definitions

tdb

Address of the task descriptor block for the subtask that has caused the task event. Default: Contents of R0.

err

Address of an error subroutine. This is entered if the macro code fails. Default: None.

IMPLEMENTATION

NOTES

- 1 Event information is returned in the subtask's task descriptor block.
- 2 Normally, RDEV\$T is used immediately after CKEV\$T when R0 was set up with the tdb address. Consequently, the tdb parameter need not be supplied.
- 3 To use this macro, the task must be TCP-privileged and have the TCS privilege PR.TEV.

RETURN DATA

- 1 Byte T.EVNT of the task descriptor block indicates which task events have occurred since the previous call of RDEV\$T. The following table indicates the meaning of bits set in T.EVNT.

| Bit Name | Meaning |
|----------|---------------------------------------------------|
| IF.JS | Successfully terminated. |
| IF.JA | Aborted. |
| IF.NL | Load failure. |
| IF.SU | Suspended itself. |
| IF.CH | Successfully terminated and successfully chained. |
| IF.SD | Task has sent a message. |

- 2 A buffer in the task descriptor block starting at byte T.EVBF contains task termination information. This is described in Section 3.2.

RESULT CODES

General result codes are given in Appendix B.

The following result code relates only to this macro.

| | | | |
|-----------|--------|---------|-------------------------------------|
| Bytename: | T.ERR | T.ERR+2 | Meaning |
| Value: | PS.SUC | PS.NOE | The task has not caused any events. |

RSAS\$T

RSAS\$T

TYPE

Imperative.

FUNCTION

Used by a task to resume its descendants that were automatically suspended on the occurrence of `Ctrl/C`.

MACRO CALL

RSAS\$T *esb, err*

parameter definitions

esb

Address of an error status block to be used for returning result codes. Default: None.

err

Address of an error handling subroutine. This is entered if errors occur during execution of the macro code. Default: None.

IMPLEMENTATION NOTES

To use this macro, the task must be TCP-privileged and have the TCS privilege PR.RST.

RETURN DATA

None.

RESULT CODES

General result codes are given in Appendix B.

The following result code relates only to this macro.

Bytename: T.ERR T.ERR+2 Meaning

Value: PE.ABO PE.NTA The invoking task has no active subtasks.

RSUM\$T

RSUM\$T

TYPE

Imperative.

FUNCTION

Used by a task to resume a suspended subtask.

MACRO CALL

RSUM\$T *tdb, err, subtsk*

parameter definitions

tdb

Address of the subtask's task descriptor block. Default: Contents of R0.

err

Address of an error subroutine. This is entered if the macro code fails. Default: None.

subtsk

Indicates whether the subtask alone or the subtask and its descendants are to be resumed.

Parameter

| Value | Action Taken |
|--------|--------------------------------------|
| SUBTSK | Subtask and descendants are resumed. |

Default: Only the subtask is resumed.

IMPLEMENTATION

NOTES

- 1 If the macro is used to try to resume a subtask that is already running, the macro has no effect and indicates successful completion.
- 2 To use this macro, the task must be TCP-privileged and have the TCS privilege PR.RST.

RETURN DATA

R0 contains the address of the subtask's task descriptor block.

RESULT CODES

General result codes are given in Appendix B.

The following result codes relate only to this macro:

| Bytename: | T.ERR | T.ERR+2 | Meaning |
|-----------|--------|---------|-------------------------------------------------------------------------|
| Value: | PE.ABO | PE.NTA | The subtask is not active. |
| | PE.ABO | PE.ILL | The subtask is not owned by the task that issued the RSUM\$T directive. |

RUN\$T

RUN\$T

TYPE

Imperative.

FUNCTION

Used by a task to place one of its subtasks on one of the queues maintained by the timesharing scheduler. The scheduler is described in the *IAS System Management Guide*.

Additionally, a message can be sent to the subtask that would be available to the subtask at startup time. The invoking task can also wait or stop until a task event from the subtask occurs.

MACRO CALL

RUN\$T *tdb,cmda,cmdl,cmdt,sdb,wt,err*

parameter definitions

tdb

Address of the subtask's task descriptor block. Default: Contents of R0.

cmda

Address of a command buffer that contains the name of the subtask and a command line. This address is copied into the task descriptor block for the subtask at offset T.CMDA. Default: Existing contents of T.CMDA.

cmdl

Length, in bytes, of the buffer specified by cmda. This value is copied into the task descriptor block for the subtask at offset T.CMDL. Default: Existing contents of T.CMDL.

cmdt

Format of the command buffer. This can take the following values: TS.USE, TS.DOT, TS.DOL, or TS.INS. The specified value is copied into the task descriptor block at offset T.TTYP. Default: Existing contents of the TDB.

sdb

Address of a send/receive data block that describes a message to be sent to the subtask. Default: No message is sent.

wt

Indicates whether to stop or wait until the subtask causes a task event.

| Parameter Value | Action Taken |
|-----------------|-----------------------------------|
| STOP | Host task stops-until-task-event. |
| WAIT | Host task waits-until-task-event. |

Default: The invoking task continues execution in parallel with the subtask.

err

Address of an error handling subroutine. This is entered if errors occur during execution of the macro code. Default: None.

IMPLEMENTATION NOTES

- 1 Name of the subtask to run is contained in the command buffer specified in the TDB. T.SNDR, in the SDB, is ignored.
- 2 Parameter wt must be either blank, WAIT, or STOP.
- 3 If the subtask is to be autoinstalled, using SETF\$T to attempt to set a flag will return PE.ABO/PE.INS until the task is installed and active.
- 4 To use this macro, the task must be TCP-privileged and have the TCS privilege PR.RST (and the TCS privilege PR.CHN if sending messages).
- 5 The macro clears the event indicators (T.EVNT) in the TDB.

RETURN DATA

R0 contains the address of the subtask's task descriptor block.

RESULT CODES

General result codes are given in Appendix B.

The following result codes relate only to this macro.

| Bytename: | T.ERR | T.ERR+2 | Meaning |
|-----------|--------|---------|----------------------------------------------------------------------------------------------------------------------------------------|
| Value: | PE.ABO | PE.MJA | Attempt to exceed the maximum number of active tasks for this terminal. |
| | PE.ABO | PE.TDB | TDB not declared or already being used for an active task. |
| | PE.ABO | PE.TSE | Error in command line—for example, zero length or space-filled. |
| | PE.ABO | 0 | Error in command line—invalid file name or task name. |
| | PE.ABO | PE.BUF | No command line specified or greater than 79 characters (system or library tasks) or greater than 67 characters (user filename tasks). |
| | PE.ABO | PE.ADR | Command line not in user's address space. |

RUN\$T

| | | |
|--------|--------|----------------------------------------------------------------------|
| PE.ABO | PE.ILL | Illegal command line format-type. |
| PE.ABO | PE.TNI | Nonexistent system task or library task. |
| PE.ABO | PE.MJN | Attempt to exceed the maximum number of active tasks for the system. |
| PE.ABO | PE.NOB | Attempt to run the task as a batch task when no batch level exists. |
| PE.ABO | PE.UPN | SCOM node pool exhausted. |
| PE.ABO | PE.IBS | Send message >253 (decimal) characters. |
| PE.ABO | PE.BUF | Unavailable buffer space for send message. |
| PE.ABO | PE.ADR | Send message not all in user's address space. |
| PE.BAD | 0 | Invalid scheduling level. |
| PE.BAD | PE.PRI | Attempt to confer privileges that are greater than the caller's. |
| PE.ABO | PE.TAA | Subtask is already active. |

SDBDF\$

TYPE

Data definition—assembly time.

FUNCTION

Reserve data space for a send/receive data block and define its contents.

MACRO CALL

SDBDF\$ *mess,length,tdb,esb*

parameter definitions

mess

Address of a message buffer. The first word of the buffer must contain the message length; the remainder of the buffer contains the message itself. This parameter value is placed in word T.SNDA of the send/receive data block. The buffer must begin on a word boundary. Default: T.SNDA is left blank.

length

Length of the message in bytes. This value is placed in word T.SNDL of the send/receive data block. Default: T.SNDL is left blank.

tdb

Indicates the destination for the message. The parameter value is placed in word T.SNDT of the send/receive data block.

| Parameter Value | Meaning |
|-----------------|-----------------------------------------------------------------------------------------------|
| -1 | The message is to be sent to the owner task. |
| 0 | The message is to be sent to the successor task. |
| address | The message is to be sent to a subtask whose task descriptor block has the address specified. |

Default: T.SNDT is left blank.

esb

Address of an error status block to be used for result codes that relate to the use of this send/receive data block. This is placed into word T.SESB of the send/receive block. Default: T.SESB is left blank.

SDBDF\$

IMPLEMENTATION NOTES

- 1 If the contents of the send/receive data block are not defined by this macro, a block is created but the contents are blank.
- 2 The contents of a send/receive data block can also be set up by the macros SDDF\$R, SEND\$T, and RCV\$T.
- 3 The macro defines local offsets for SDBs and ESBs.

SDDF\$R

TYPE

Data definition—run time.

FUNCTION

Define the contents of a send/receive data block.

MACRO CALL

SDDF\$R *sdb,mess,length,tdb,esb*

parameter definitions

sdb

Address of the send/receive data block. Default: Contents of R0.

mess

Address of a message buffer. This value is copied into the send/receive data block at offset T.SNDA. Default: Existing contents of T.SNDA.

length

Length in bytes of the buffer specified by *mess*. This value is copied into the send/receive data block at offset T.SNDL. Default: Existing contents of T.SNDL.

tdb

Indicates the destination of a message to be sent.

Parameter

| Value | Action Taken |
|---------|----------------------------------------------------------------------------------------|
| -1 | Message is to be sent to the owner task. |
| 0 | Message is to be sent to successor task. |
| address | Message is to be sent to the subtask whose task descriptor block address is specified. |

This parameter value is copied into the send/receive data block at offset T.SNDT. Default: Existing contents of T.SNDT.

esb

Address of an error status block to be used for result codes that relate to the use of this send/receive data block. This value is copied into the send/receive data block at offset T.SESB. Default: Existing contents of T.SESB.

SDDF\$R

IMPLEMENTATION NOTES

- 1 Contents of a send/receive data block can also be defined by means of SDBDF\$, SEND\$T, and RCV\$T.
- 2 The format of the send/receive data block is given in Section A.3.

SEND\$T

TYPE

Imperative.

FUNCTION

Used by a task to send a message to a related task (either a subtask, the task's owner, or a chained successor task).

MACRO CALL

SEND\$T *sdb,dest,mess,esb,err*

parameter definitions

sdb

Address of the send/receive data block that describes the message to be sent. Default: Contents of R0.

dest

Indicates the destination of the message.

| Parameter Value | Action Taken |
|-----------------|------------------------------------------------------------------------------|
| -1 | Message is sent to the task's owner. |
| 0 | Message is sent to the chained successor task. |
| address | Message is sent to the subtask whose task descriptor block address is given. |

The value specified is copied into the send/receive data block at offset T.SNDDT.

Default: Existing contents of T.SNDDT.

mess

Address of the message buffer. This is copied into the send/receive data block at offset T.SNDA. Default: Existing contents of T.SNDA.

esb

Address of an error status block. This is copied into the send/receive data block at offset T.SESB. Default: Existing contents of T.SESB.

err

Address of an error handling subroutine. Default: None.

SEND\$T

IMPLEMENTATION NOTES

- 1 When data is sent, it is queued for the receiving task. The data is intermediately stored in the SCOM node pool.
- 2 To use this macro, the task must be TCP-privileged and have the TCS privileges PR.CHN and PR.TEV.

RETURN DATA

R0 contains the address of the send/receive data block.

RESULT CODES

General result codes are given in Appendix B.

The following result codes relate only to this macro.

| Bytename: | T.ERR | T.ERR+2 | Meaning |
|-----------|--------|---------|-------------------------------------------|
| Value: | PE.ABO | PE.NTA | Receiver not active. |
| | PE.ABO | PE.BUF | Failed to claim a buffer. |
| | PE.ABO | PE.ADR | Message not all in user's space. |
| | PE.ABO | PE.IBS | Message length >253 (decimal) characters. |
| | PE.ABO | PE.ILL | CLI sending to owner. |

SETF\$T

TYPE

Imperative.

FUNCTION

Used by a task to set a subtask's local event flag.

MACRO CALL

SETF\$T *tdb,ef,err*

parameter definitions

tdb

Address of the task descriptor block for the subtask. Default: Contents of R0.

ef

Number of a local event flag. Default: No default. This parameter is mandatory.

err

Address of an error handling subroutine. Default: None.

IMPLEMENTATION

NOTES

- 1 This macro cannot be used to set a global event flag (the system directive SETF\$ can be used for this. See the *IAS System Directives Reference Manual*).
 - 2 To use this macro, the task must be TCP-privileged and have the TCS privilege PR.RST.
-

RETURN DATA

R0 contains the address of the subtask's task descriptor block.

RESULT CODES

General result codes are given in Appendix B.

SETF\$T

The following result codes relate only to this macro.

| Bytename: | T.ERR | T.ERR+2 | Meaning |
|-----------|--------|---------|--------------------------------------------------------------|
| Value: | PE.ABO | PE.NTA | Subtask not active. |
| | PE.ABO | PE.GEF | Global event flag specified. |
| | PE.ABO | PE.INS | User filename task not yeat installed. |
| | PE.ABO | PE.ILL | Task issuing the SETF\$T directive does not own the subtask. |

SHUT\$T

TYPE

Imperative.

FUNCTION

Used by a CLI to obtain the number of minutes before shutdown. An error indication is returned if shutdown has not been declared at the SCI console.

MACRO CALL

SHUT\$T *esb,err*

parameter definitions

esb

Address of an error status block to be used by TCS to return error codes. Default: None.

err

Address of an error handling subroutine. This is entered if errors occur during execution of the macro code. Default: None.

IMPLEMENTATION

NOTES

- 1 This macro is available only to CLIs.
 - 2 To use this macro, the task must be TCP-privileged and have the TCS privilege PR.TEV.
-

RETURN DATA

R0 contains the number of minutes before shutdown.

RESULT CODES

General result codes are given in Appendix B.

SHUT\$T

The following result code relates only to this macro.

| Bytename: | T.ERR | T.ERR+2 | Meaning |
|-----------|--------|---------|--------------------------|
| Value: | PE.ABO | PE.NOS | No shutdown outstanding. |

SPND\$T

TYPE

Imperative.

FUNCTION

Used by a task to suspend one of its subtasks.

SPND\$T

tdb, err

parameter definitions

tdb

Address of the task descriptor block for the subtask to be suspended. Default: Contents of R0.

err

Address of an error handling subroutine. Default: None.

IMPLEMENTATION

NOTES

- 1 If the subtask specified has already been suspended, no action is taken and the macro code exits normally.
 - 2 An event is not caused by the suspension.
 - 3 To use this macro, the task must be TCP-privileged and have the TCS privilege PR.RST.
-

RETURN DATA

R0 contains the address of the subtask's task descriptor block.

RESULT CODES

General result codes are given in Appendix B.

SPND\$T

The following result codes relate only to this macro.

| Bytename: | T.ERR | T.ERR+2 | Meaning |
|-----------|--------|---------|--------------------------------------------------------------|
| Value: | PE.ABO | PE.NTA | Subtask not active. |
| | PE.ABO | PE.ILL | Task issuing the SPND\$T directive does not own the subtask. |

TCOFF\$

TYPE

Data definition—assembly time.

FUNCTION

Define all offset names for chain task descriptor blocks. They can be defined to be either local or global.

MACRO CALL

TCOFF\$ *offset*

parameter definitions

offset

Indicates whether the offsets are local or global.

Parameter

| Value | Action Taken |
|--------|-------------------------------------|
| DEF\$L | Offset names are defined as local. |
| DEF\$G | Offset names are defined as global. |

Default: DEF\$L.

IMPLEMENTATION NOTES

- 1 TCOFF\$ is called by the macros TDOFF\$ and CDBDF\$ to define local offset names.
- 2 After one invocation, the macro redefines itself to null.

TCSMC\$

TCSMC\$

TYPE

Not applicable.

FUNCTION

Used by a task to MCALL all the other TCS macros.

MACRO CALL
TCSMC\$

parameter
definitions

This call has no parameters.

IMPLEMENTATION
NOTES

1 If this macro is not used, a program must individually MCALL each TCS macro it invokes.

TDBDF\$

TYPE

Data definition—assembly time.

FUNCTION

Reserve data space for a task descriptor block.

MACRO CALL

TDBDF\$

parameter definitions

This call has no parameters.

IMPLEMENTATION

NOTES

- 1 The contents of a task descriptor block can be set up by means of the following macros: TDCM\$, TDPR\$, TDTA\$, TDEB\$, TDCM\$, TDPR\$, TDTA\$, RUN\$, and TDBD\$.
- 2 This macro invokes TDOFF\$ to define local offset names.

TDBD\$T

TDBD\$T

TYPE

Imperative.

FUNCTION

Used by a task to declare a task descriptor block.

MACRO CALL

TDBD\$T *tdb,esb,err*

parameter definitions

tdb

Address of the task descriptor block. Default: Contents of R0.

esb

Address of an error status block. This parameter is copied into the task descriptor block at offset T.TESB. Default: Existing contents of T.TESB.

err

Address of an error handling subroutine. Default: None.

IMPLEMENTATION

NOTES

- 1 A subtask cannot be run until it has a declared task descriptor block associated with it.
 - 2 The error status block associated with a task descriptor block is used for all reporting of errors concerning the task.
 - 3 The macro clears the task event indicators (T.EVNT) in the TDB.
-

RETURN DATA

R0 contains the address of the task descriptor block.

RESULT CODES

General result codes are given in Appendix B.

The following result codes relate only to this macro.

| Bytename: | T.ERR | T.ERR+2 | Meaning |
|-----------|-------|---------|---------|
|-----------|-------|---------|---------|

| | | | |
|--------|--------|--------|-----------------------------------------|
| Value: | PE.ABO | PE.TDB | Task descriptor block already declared. |
|--------|--------|--------|-----------------------------------------|

TDBR\$T

TDBR\$T

TYPE

Imperative.

FUNCTION

Release a task descriptor block.

MACRO CALL

TDBR\$T *tdb, err*

parameter definitions

tdb

Address of the task descriptor block that is no longer required. Default: Contents of R0.

err

Address of an error handling subroutine. This is entered if the macro code fails. Default: None.

IMPLEMENTATION

NOTES

- 1 When the task descriptor block is released, it is ready for declaration.
 - 2 To use this macro, the task must be TCP-privileged. Depending on the current state of the task, TCS might have to perform some event-related housekeeping activities. For this reason, the task must also have the TCS privilege PR.TEV.
-

RETURN DATA

R0 contains the address of the task descriptor block.

RESULT CODES

| Bytename: | T.ERR | T.ERR+2 | Meaning |
|-----------|--------|---------|--------------------------------------------------|
| Value: | PE.ABO | PE.TDB | Task descriptor block not declared. |
| Value: | PE.ABO | PE.TAA | Task descriptor block represents an active task. |

TDCM\$A

TDCM\$A

TYPE

Data definition—assembly time.

FUNCTION

Specify the name of the task to which a task descriptor block relates. Also, in the case of system and library tasks, specify a command line. See Section 3.2.

MACRO CALL

TDCM\$A *cmda,cmdl,cmdt*

parameter definitions

cmda

Address of a command buffer that contains the name of the subtask and, optionally, a command line. This address is placed in the task descriptor block for the subtask at offset T.CMDA. Default: T.CMDA is left blank.

cmdl

Length, in bytes, of the buffer specified by *cmda*. This value is copied into the task descriptor block at offset T.CMDL. Default: T.CMDL is left blank.

cmdt

Format of the buffer. This can take the following values: TS.USE, TS.DOT, TS.DOL, or TS.INS. The specified value is placed in the task descriptor block at offset T.TTYP. Default: T.TTYP is left blank.

IMPLEMENTATION

NOTES

- 1 This macro relates to the msot recently defined task descriptor block.
- 2 The macro can be used at any point in a program.
- 3 The layout of the task descriptor block is given in Section A.1.
- 4 The macro invokes the TDOFF\$ macro to define local offset names.

TDCM\$R

TYPE

Data definition–runtime.

FUNCTION

Define (in a task descriptor block) the task name and, in the case of system tasks and library tasks, a command line.

MACRO CALL

TDCM\$R *tdb,cmda,cmdl,cmdt*

parameter definitions

tdb

Address of a task descriptor block. Default: Contents of R0.

cmda

Address of a buffer that contains a taskname and, optionally, a command line. This value is copied into the task descriptor block at offset T.CMDA. Default: Existing contents of T.CMDA.

cmdl

Length, in bytes, of the specified buffer. This value is copied into the task descriptor block at offset T.CMDL. Default: Existing contents of T.CMDL.

cmdt

Format of the specified buffer. This can take the following values: TS.USE, TS.DOT, TS.DOL, or TS.INS. The specified value is copied into the task descriptor block at offset T.TTYP. Default: Current contents of T.TTYP.

IMPLEMENTATION

NOTES

Layout of the task descriptor block is described in Section A.1.

TDEB\$A

TDEB\$A

TYPE

Data definition—assembly time.

FUNCTION

Set a pointer in a task descriptor block to an error status block.

MACRO CALL

TDEB\$A *esb*

parameter definitions

esb

Address of an error status block. This is written into the task descriptor block at offset T.TESB.
Default: T.TESB is left blank.

IMPLEMENTATION

NOTES

- 1 This macro relates to the most recently defined task descriptor block.
- 2 The macro can be used at any point within a program.
- 3 The macro invokes the TDOFF\$ macro to define local offset names.

TDOFF\$

TYPE

Data definition—assembly time.

FUNCTION

Define all offset names for task descriptor blocks, chain task descriptor blocks, send/receive data blocks, and error status blocks.

MACRO CALL

TDOFF\$ *offset*

parameter definitions

offset

Indicates whether the offset names are to be defined as global or local.

| Parameter Value | Action Taken |
|-----------------|-------------------------------------|
| DEF\$L | Offset names are defined as local. |
| DEF\$G | Offset names are defined as global. |

Default: DEF\$L.

IMPLEMENTATION NOTES

- 1 TDOFF\$ is called by TDCM\$, TDPR\$, TDTA\$, and TDEB\$ to define local names.
- 2 The macro redefines itself to null after it has been invoked.

TDPR\$A

TDPR\$A

TYPE

Data definition—assembly time.

FUNCTION

Define whether or not a task has the following privileges:

- 1 Auto-installed task with executive privilege.
- 2 Real-time directive privilege.
- 3 TCP privilege.
- 4 Memory management privilege.
- 5 Maximum number of descendants of this task that can be active simultaneously.

MACRO CALL

TDPR\$A *tp1, tp2, jnod*

parameter definitions

tp1

A set of bit flags that specifies task privileges as follows:

| | |
|-------|------------------------------------------------------------------------------------------------------------------------------------------|
| JP.PI | Privilege to issue TCP requests. |
| JP.PD | Privilege to issue real-time directive privileged directives. |
| JP.PT | Privilege to be an executive-privileged task that is auto-installed. |
| JP.PP | Privilege to issue memory management directives. JP.PP must be set for tasks that use resident overlaid libraries (for example, RMSRES). |

The parameter can take the value JP.PI, JP.PD, JP.PT, or JP.PP or the logical OR of any of these values. For example, JP.PI!JP.PD!JP.PT!JP.PP specifies all the task privileges.

The parameter value is copied into the task descriptor block at offset T.TP1. Default: The existing value of T.TP1.

tp2

A set of bit flags that specifies which TCS facilities can be used by the task. The flags are as follows:

| | |
|--------|------------------------------------------------------------------------------------------------|
| PR.RST | Privilege to initiate and control subtasks. |
| PR.CTC | Privilege to be the Ctrl/C task. |

PR.TEV Privilege to be notified of the occurrence of task events.

PR.CHN Privilege to chain and send or receive messages.

The parameter can take the value PR.RST, PR.CTC, PR.TEV, or PR.CHN or the logical OR of any of these values. For example, PR.RST!PR.CTC!PR.TEV!PR.CHN as parameter value specifies that the task can use all the TCS facilities.

The parameter value is copied into the task descriptor block at offset T.TP2. Default: Existing contents of T.TP2.

jnod

Maximum number of descendants of this task that can be simultaneously active. This parameter value is copied into the task descriptor block at offset T.JNA. Default: Existing contents of T.JNA.

IMPLEMENTATION

NOTES

This macro invokes TDOFF\$ to define local offset names.

TDPR\$R

TDPR\$R

TYPE

Data definition—runtime.

FUNCTION

To define whether or not a task has the following privileges:

- 1 Executive-privilege that is auto-installed.
 - 2 Real-time directive privilege.
 - 3 TCP privilege.
 - 4 Memory management privilege.
 - 5 Maximum number of descendants of this task that can be simultaneously active.
-

MACRO CALL

TDPR\$R *tdb,tp1,tp2,jnod*

parameter definitions

tdb

Default: Contents of R0.

tp1

A set of bit flags that specifies task privileges as follows:

| | |
|-------|------------------------------------------------------------------------------------------------------------------------------------------|
| JP.PI | Privilege to issue TCP requests. |
| JP.PD | Privilege to issue real-time directive privileged directives. |
| JP.PT | Privilege to be an executive-privileged task that is auto-installed. |
| JP.PP | Privilege to issue memory management directives. JP.PP must be set for tasks that use resident overlaid libraries (for example, RMSRES). |

The parameter can take the value JP.PI, JP.PD, JP.PT, or JP.PP or the logical OR of any of these values. For example, JP.PI!JP.PD!JP.PT!JP.PP specifies all the task privileges.

The parameter value is copied into the task descriptor block at offset T.TP1. Default: The existing value of T.TP1.

tp2

A set of bit flags that specifies which TCS facilities can be used by the task. The flags are as follows:

PR.RST Privilege to initiate and control subtasks.
PR.CTC Privilege to be the `Ctrl/C` task.
PR.TEV Privilege to be notified of the occurrence of task events.
PR.CHN Privilege to chain and send or receive messages.

The parameter can take the value PR.RST, PR.CTC, PR.TEV, or PR.CHN or the logical OR of any of these values. For example, PR.RST!PR.CTC!PR.TEV!PR.CHN as parameter value specifies that the task can use all the TCS facilities.

The parameter value is copied into the task descriptor block at offset T.TP2. Default: Existing contents of T.TP2.

jnod

Maximum number of descendants of this task that can be simultaneously active. This parameter value is copied into the task descriptor block at offset T.JNA. Default: Existing contents of T.JNA.

IMPLEMENTATION

NOTES

None.

TDTA\$A

TDTA\$A

TYPE

Data definition—assembly time.

FUNCTION

To define task attributes.

MACRO CALL

TDTA\$A *level,attrib*

parameter definitions

level

Scheduling level. This value is assembled into the most recently defined task descriptor block at offset T.SCHL. The permissible values are 0 (which defaults to 1) or 1 to n, where n is the number of scheduling levels in the system. Default: None.

attrib

This parameter indicates whether or not the task is to be run in the batch scheduling level and whether or not the task is to be automatically suspended when `[Ctrl/C]` occurs. (See Section 3.2.1.3.)

Parameter

| Value | Action Taken |
|-------------|--------------------------------------------------------------------------------------------------------------|
| FB.BT | Run task in the batch scheduling level. In this case, the value specified in level is ignored. |
| FB.NC | Do not suspend task automatically when <code>[Ctrl/C]</code> occurs. |
| FB.BTIFB.NC | Run task at the batch scheduling level; do not suspend task automatically when <code>[Ctrl/C]</code> occurs. |

The value specified is written into the most recently defined task descriptor block at offset T.FLG1. Default: The task is automatically suspended on `[Ctrl/C]` and is not run in the batch scheduling level.

IMPLEMENTATION NOTES

- 1 This macro call can be written at any point in the program. It always relates to the most recently defined task descriptor block.
- 2 Scheduling levels are described in the *IAS System Management Guide*.
- 3 The macro invokes the TDOFF\$ macro to define local offsets.

- 4 Tasks defined with this macro are run under the same UIC as the requestor. For PDS terminals, the UIC is the user's logged-in UIC. For terminals running with other CLIs, the UIC is that specified when the CLI is allocated (with the ALLOCATE/TERMINAL command). Default is [1,1].

TDTA\$R

TDTA\$R

TYPE

Data definition—runtime.

FUNCTION

Defines the following task attributes:

- 1 Initial scheduling level.
- 2 Whether or not it is to be run at the batch level.
- 3 Whether or not it is to be automatically suspended when **Ctrl/C** occurs.

MACRO CALL

TDTA\$R *tdb,level,attrib*

parameter definitions

tdb

Address of a task descriptor block. Default: Contents of R0.

level

Scheduling level. This has the value 0 (which defaults to 1) or 1 to n where n is the number of scheduling levels in the system. It is copied into the task descriptor block at offset T.SCHL. Default: Existing contents of T.SCHL.

attrib

This parameter indicates the following:

- 1 Whether or not the task is to be run in the batch scheduling level.
- 2 Whether or not the task is to be automatically suspended on **Ctrl/C** occurrence. See also Section 3.2.1.3.

| Parameter Value | Action Taken |
|-----------------|--------------|
|-----------------|--------------|

| | |
|-------|----------------------------------------------|
| FB.BT | Task is to be run in batch scheduling level. |
|-------|----------------------------------------------|

| | |
|-------|------------------------------------------------------------------------|
| FB.NC | Task is not to be suspended automatically on Ctrl/C occurrence. |
|-------|------------------------------------------------------------------------|

| | |
|-------------|--------------------------|
| FB.BTIFB.NC | Both of the above apply. |
|-------------|--------------------------|

Parameter value specified is copied into the task descriptor block offset T.FLG1. Default: Existing contents of T.FLG1.

IMPLEMENTATION NOTES

- 1 Scheduling levels are described in the *IAS System Management Guide*.
- 2 Tasks defined with this macro are run under the same UIC as the requestor's. For PDS terminals, the UIS is the user's logged-in UIC. For terminals that run with other CLIs, the UIC is that specified when the CLI is allocated (with the ALLOCATE/TERMINAL command). Default is [1,1].

TEOFF\$

TEOFF\$

TYPE

Data definition—assembly time.

FUNCTION

Define all offset names for error status blocks.

MACRO CALL

TEOFF\$ *offset*

parameter definitions

offset

Indicates whether the offset names are to be defined as local or global.

| Parameter Value | Action Taken |
|--------------------|---------------------------------|
| DEF\$L | Offset names defined as local. |
| DEF\$G | Offset names defined as global. |

Default: DEF\$L.

IMPLEMENTATION NOTES

- 1 TEOFF\$ is used by TDOFF\$, CDBDF\$, SDBDF\$, and ESBDF\$ to define local names.
- 2 After invocation, the macro redefines itself to null.

TINIT\$

TYPE

Imperative.

FUNCTION

Initialize the timesharing control services.

MACRO CALL

TINIT\$ *esb, err*

parameter definitions

esb

Address of an error status block used by the macro to return result codes. Default: No result codes returned.

err

Address of an error handling subroutine. Default: None.

IMPLEMENTATION

NOTES

- 1 This macro initializes TCS and releases all task descriptor blocks. It should be used if it is necessary for a program to restart without being reloaded.
 - 2 This macro is not required for the initial start of a program (that is, immediately after loading).
 - 3 To use this macro, a task must be TCP-privileged. Depending on the current state of the task, TCS might have to perform some event-related housekeeping activities. For this reason, the task must also have the TCS privilege PR.TEV.
-

RETURN DATA

None.

TINIT\$

RESULT CODES

General result codes are given in Appendix B.

The following result codes relate only to this macro.

| Bytename: | T.ERR | T.ERR+2 | Meaning |
|-----------|--------|---------|------------------|
| Value: | PE.ABO | PE.TAA | Active subtasks. |

TKST\$T

TYPE

Imperative.

FUNCTION

Used by a task to obtain the size and CPU time of one of its subtasks. The size and CPU time information is returned at offsets **E.SIZ** and **E.TIM** in the subtask's TDB.

MACRO CALL

TKST\$T *tdb, err*

parameter definitions

tdb

Address of the task descriptor block of the subtask for which information is required. Default: Contents of R0.

err

Address of an error handling subroutine. Default: None.

IMPLEMENTATION

NOTES

To use this macro, the task must be TCP-privileged and have the TCS privilege **PR.RST**.

RETURN DATA

R0 contains the address of the subtask's task descriptor block.

RESULT CODES

General result codes are given in Appendix B.

The following result codes relate only to this macro.

Bytename: T.ERR T.ERR+2 Meaning

TKST\$T

Value: PE.ABO PE.NTA Subtask is not active.

Value: PE.ABO PE.ILL Task issuing the TKST\$T directive does not own the subtask.

TSOFF\$

TYPE

Data definition—assembly time.

FUNCTION

Define all offset names for send/receive data blocks.

MACRO CALL

TSOFF\$ *offset*

parameter definitions

offset

Indicates whether the offset names are to be defined as local or global.

Parameter

| Value | Action Taken |
|--------|-----------------------------------------|
| DEF\$L | The offset names are defined as local. |
| DEF\$G | The offset names are defined as global. |

Default: DEF\$L.

IMPLEMENTATION

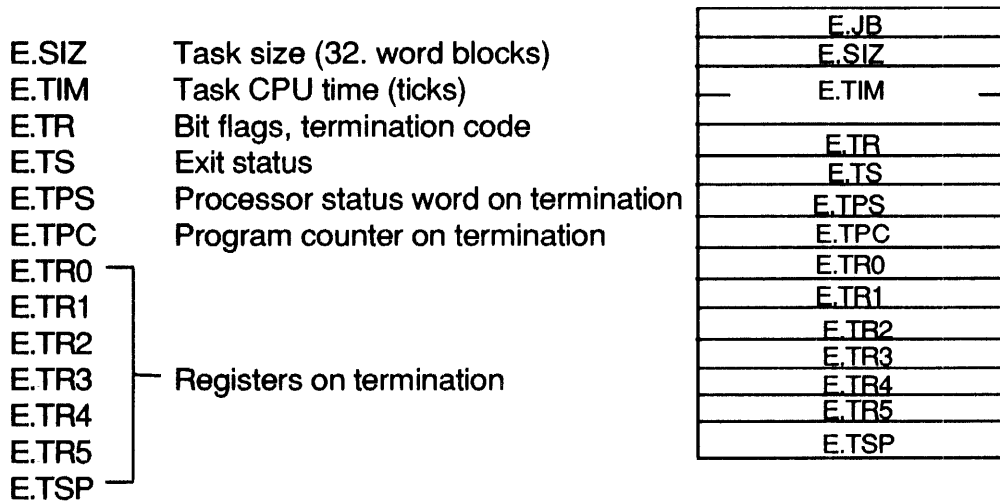
NOTES

- 1 TDOFF\$ and SDBDF\$ use TSOFF\$ to define local names.
- 2 After invocation, the macro redefines itself to null.

Data Block Layouts

A.1.1 T.EVBF—Termination Information

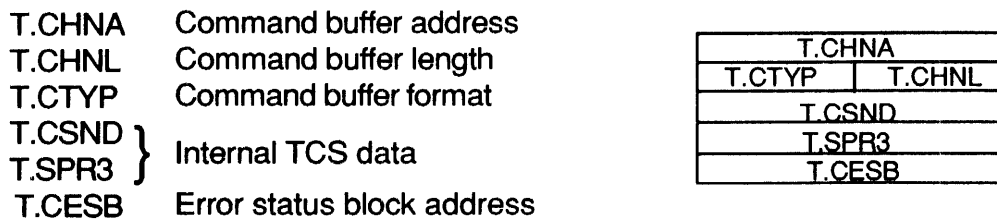
Figure A-2 T.EVBF



Task descriptor blocks are fully described in Section 3.2.1.

A.2 Chain Task Descriptor Block

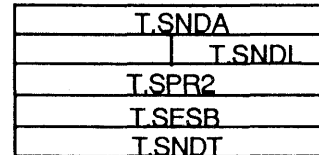
Figure A-3 Chain Task Descriptor Block



A.3 Send/Receive Data Block

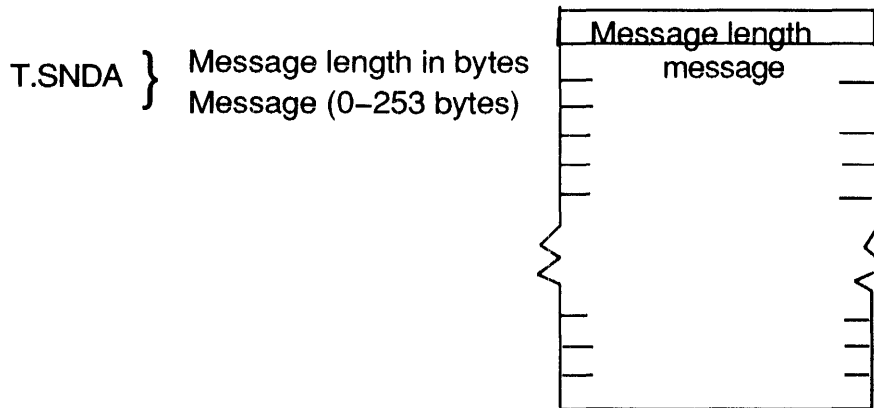
Figure A-4 Send/Receive Data Block

T.SNDA Message buffer address
 T.SNDL Message buffer length
 T.SPR2 Internal TCS data
 T.SESB Error status block address
 T.SNDT Sender or recipient name



A.3.1 Message Format

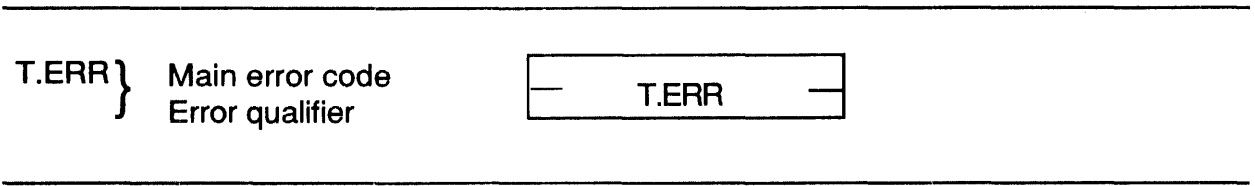
Figure A-5 Message Format



A.4 Error Status Block

Data Block Layouts

Figure A-6 Error Status Block



B

Error Codes

B.1 Task Termination Codes

As shown in Table B-1, if the event indicator EV.AB is set in the high order byte of E.TR, one of the following codes is returned in the low byte of E.TR in the task descriptor block.

Table B-1 Task Termination Codes

| Code | Meaning |
|------|----------------------------------------------|
| 0 | Odd address or other trap 4 |
| 2 | Segment fault |
| 4 | Execution of "BPT" or T-bit trap |
| 6 | Execution of "IOT" |
| 10 | Execution of reserved instruction |
| 12 | Execution of non IAS "EMT" instruction |
| 14 | Execution of "TRAP" |
| 16 | 11/45 FPP exception |
| 20 | Reserved |
| 22 | Reserved |
| 24 | Reserved |
| 26 | Reserved |
| 30 | SST aborted (bad stack) |
| 32 | AST aborted (bad stack) |
| 34 | Task aborted |
| 36 | Unused |
| 40 | Task load failure |
| 42 | Task swap failure |
| 44 | IOSB fault |
| 46 | Task failed to install |
| 50 | Task not loaded because handler not resident |
| 52 | Task too large |
| 54 | Executive-privileged task but not allowed |
| 56 | Non-multiuser task already active |
| 60 | No swap space available |
| 62 | Unavailable pool node for ATL |
| 64 | Task fixed |
| 66 | Task disabled |

Error Codes

B.2 Result Codes

As shown in Table B-2, the following codes can be returned from any imperative macro.

Table B-2 Imperative Macro Return Codes

| Byte Name | T.ERR | T.ERR+1 | T.ERR+2 | Meaning |
|-----------|--------|---------|---------|-------------------------------------------------------------------------------------|
| | PS.SUC | 0 | 0 | Complete success. |
| | PE.PRI | 0 | 0 | Caller is not TCP-privileged. |
| Value | PE.PRI | 0 | PE.ILL | Caller is not privileged enough to perform this function. |
| | PE.PRI | 0 | PE.NTS | Caller is not a timesharing task. |
| | PE.ABO | 0 | PE.BUF | Unavailable buffer for event reporting. |
| | PE.ABO | 0 | PE.MJN | Calling task failed to initiate because system maximum task limit has been reached. |

C

Examples of TCS Macros

```
User BOB UIC [200,140] BA00: 13:58:49 26-MAY-78

$!
$ON ERROR CONTINUE
$!
$!
$! THIS BATCH JOB DEMONSTRATES THE USE OF SOME TCS MACROS.
$! THE EXAMPLES ARE COMPLETELY ARTIFICIAL, THEIR ONLY GOAL
$! BEING TO SHOW AS MANY MACROS AS POSSIBLE IN AS SHORT
$! SPACE AS POSSIBLE. THE JOB HAS TWO SEPARATE EXAMPLES:
$! 1. SUBTASKING
$! 2. CHAINING AND SENDING/RECEIVING.
$!
$! THIS BATCH STREAM WAS ALLOCATED USING:
$! SCI> ALLOCATE/TERMINAL BA0N PDS (RUN BATCH MAX:3)
$!
$! USER 'BOB' HAS A UPF ENTRY CONTAINING (INTER ALIA):
$! TP1 = 1
$! TP2 = 622
$! MTS = 3
$!
$!
$! S U B T A S K I N G
$! - - - - -
$!
$! IN THIS EXAMPLE, 'EXAMPLE1' INITIATES A SUBTASK CALLED 'SUBTASK1'.
$! 'SUBTASK1' FIRST SUSPENDS ITSELF THEN WAITS FOR EVENT FLAG #1
$! BEFORE EXITING. 'EXAMPLE1' DETECTS THE SUSPENSION, RESUMES
$! 'SUBTASK1' AND SETS ITS EVENT FLAG #1. IT THEN WAITS FOR THE
$! EXIT EVENT.
$!
$!
$! FIRST ASSEMBLE AND BUILD THE TASKS
$!
$MAC/LI/SW:(/LI:TTM/NL:BIN) EXAMPLE1
13:59:15 Size: 27K CPU: 10.22 Status: SUCCESS
$MAC/LI/SW:(/LI:TTM/NL:BIN) SUBTASK1
13:59:24 Size: 27K CPU: 2.00 Status: SUCCESS
$LINK EXAMPLE1
13:59:44 Size: 20K CPU: 11.12 Status: SUCCESS
$LINK SUBTASK1
13:59:57 Size: 20K CPU: 7.84 Status: SUCCESS
$!
$!
$! NOW RUN EXAMPLE1
$!
$RUN EXAMPLE1
13:59:57
SUBTASK SUSPENDED
SUBTASK RESUMED
SUBTASK TERMINATED
13:59:59 Size: 1K CPU: 0.04
$!
$!
$!
```

Examples of TCS Macros

```
$!  
$!      C H A I N I N G      &      S E N D I N G / R E C E I V I N G  
$!      - - - - -  
$!  
$!      IN THIS EXAMPLE, 'EXAMPLE2' INITIATES A SUBTASK CALLED 'SUBTASK2'  
$!      AND DECLARES A CHAIN TASK CALLED 'CHAIN'. 'EXAMPLE2' SENDS  
$!      MESSAGES TO BOTH OTHER TASKS. 'SUBTASK2' SENDS THE MESSAGE BACK.  
$!  
$!  
$! FIRST ASSEMBLE AND BUILD THE TASKS  
$!  
$MAC/LI/SW:(/LI:TTM/NL:BIN) EXAMPLE2  
14:00:27 Size: 27K CPU: 12.78 Status: SUCCESS  
$MAC/LI/SW:(/LI:TTM/NL:BIN) SUBTASK2  
14:00:45 Size: 27K CPU: 6.10 Status: SUCCESS  
$MAC/LI/SW:(/LI:TTM/NL:BIN) CHAIN  
14:01:02 Size: 27K CPU: 5.88 Status: SUCCESS  
$LINK EXAMPLE2  
14:01:22 Size: 20K CPU: 11.50 Status: SUCCESS  
$LINK SUBTASK2  
14:01:39 Size: 20K CPU: 10.40 Status: SUCCESS  
$LINK CHAIN  
14:01:57 Size: 20K CPU: 10.56 Status: SUCCESS  
$!  
$!  
$! NOW RUN EXAMPLE2  
$!  
$RUN EXAMPLE2  
14:01:57  
SENDING "FGHI" TO CHAIN TASK  
RUNNING SUBTASK -- SENDING "ABCDE"  
SUBTASK EXITING  
MESSAGE RECEIVED FROM SUBTASK = "ABCDE"  
EXAMPLE2 CHAINING  
14:01:59 Size: 1K CPU: 0.10 (Chaining)  
CHAIN MESSAGE RECEIVED = "FGHI"  
14:02:00 Size: 1K CPU: 0.02  
$!  
$!  
$EOJ  
User BOB UIC [200,140] BA00: 14:02:00 26-MAY-78  
CONNECT TIME 04 M SYSTEM UTILIZATION 101 MCTS  
EXAMPLE1          MACRO D1110 26-MAY-78 13:58 PAGE 1
```


Examples of TCS Macros

```

1          .TITLE  EXAMPLE1
2          ;
3          ;          E X A M P L E 1
4          ;          - - - - -
5          ;
6          .MCALL  TCSMC$,QIOW$$,EXIT$$
7 000000    TCSMC$          ; MCALL ALL TCS MACROS
8          ;
9 000000    TDB:    TDBDF$
10 000074   TDCM$A    BUFF,LENGTH,TS.USE
11          ; THIS AUTOMATICALLY REFERS
12          ; TO TDB
13          ;
14 000074   BUFF:    .ASCII  / SUBTASK1/      ; THIS IS THE SUBTASK FILENAME
15          LENGTH=-.BUFF
16          .EVEN
17          ;
18 000106   MES1:    .ASCII  <15><12>/SUBTASK SUSPENDED/
19          MES1L=-.MES1
20          .EVEN
21          ;
22 000132   MES2:    .ASCII  <15><12>/SUBTASK TERMINATED/
23          MES2L=-.MES2
24          .EVEN
25          ;
26 000156   MES3:    .ASCII  <15><12>/ERROR/
27          MES3L=-.MES3
28          .EVEN
29          ;
30          ;
31 000166   START:
32 000166       TDBD$T  #TDB          ; DECLARE THE TDB
33 000176       BCS     ERR
34 000200       RUN$T   , , , , , WAIT    ; INITIATE THE SUBTASK...
35          ; RO WAS SET UP BY TDBD$T...
36          ; WAIT FOR SUBTASK EVENT
37 000216       BCS     ERR
38 000220       RDEV$T          ; GET THE EVENT...
39          ; NO NEED TO CKEV$T BECAUSE...
40          ; RUN$T WITH WAIT GUARANTEES...
41          ; AN EVENT HAS HAPPENED
42 000224       BCS     ERR
43 000226       BITB    #IF.SU,T.EVNT(RO) ; IS IT A SUSPENSION?
44 000234       BEQ     ERR          ; ERROR IF NOT
45 000236       QIOW$$  #IO.WLB,#5,#1, , , , <#MES1,#MES1L>
46 000276       BCS     ERR
47 000300       RSUM$T  #TDB          ; RESUME THE SUBTASK...
48          ; STRICTLY SPEAKING #TDB...
49          ; NEED NOT BE SPECIFIED...
50          ; BECAUSE RO IS STILL SET UP
51 000310       BCS     ERR
52 000312       SETF$T  , #1          ; SET ITS EVENT FLAG
53 000324       BCS     ERR
54 000326       CKEV$T  , #TDB,WAIT   ; WAIT FOR TASK EVENT...
55          ; IN THIS CASE #TDB HAS
56          ; TO BE SPECIFIED BECAUSE...
57          ; USE OF RO IS PROHIBITED
EXAMPLE1    MACRO D1110 26-MAY-78 13:58 PAGE 1-1
58 000370       BCS     ERR
59 000372       RDEV$T          ; READ THE EVENT...
60          ; BUT ONCE AGAIN RO IS SET...
61          ; BECAUSE THIS IS THE ONLY...
62          ; EVENT THAT WE ARE WAITING FOR

```

Examples of TCS Macros

```

63 000376          BCS      ERR
64 000400          BITB     #IF.JS,T.EVNT(R0) ; SUCCESSFUL TERMINATION?
65 000406          BEQ      ERR
66 000410          QIOW$$   #IO.WLB,#5,#1,,,,<#MES2,#MES2L>
67 000450          EXIT$$
68                ;
69                ;
70 000456  ERR:
71 000456          QIOW$$   #IO.WLB,#5,#1,,,,<#MES3,#MES3L>
72 000516          EXIT$$
73                ;
74                .END     START

```

EXAMPLES OF TCS MACROS Page A-6

EXAMPLE1 MACRO D1110 26-MAY-78 13:58 PAGE 1-2

SYMBOL TABLE

| | | | | | |
|---------|----------|---------|--------|------------|---------|
| BUFF | 000074R | T.CESB= | 000010 | T.SPR1= | 000032 |
| ERR | 000456R | T.CHNA= | 000000 | T.SPR2= | 000003 |
| IF.JS = | ***** GX | T.CHNL= | 000002 | T.SPR3= | 000006 |
| IF.SU = | ***** GX | T.CMDA= | 000002 | T.TESB= | 000020 |
| IO.WLB= | ***** GX | T.CMDL= | 000004 | T.TP1 = | 000010 |
| LENGTH= | 000011 | T.CSND= | 000004 | T.TP2 = | 000006 |
| MES1 | 000106R | T.CTYP= | 000003 | T.TSND= | 000016 |
| MES1L = | 000023 | T.EFN = | 000031 | T.TTYP= | 000005 |
| MES2 | 000132R | T.ERF = | 000030 | T.UIC = | 000012 |
| MES2L = | 000024 | T.ERR = | 000000 | \$\$\$ARG= | 000002 |
| MES3 | 000156R | T.EVBF= | 000036 | \$\$\$M = | 000006 |
| MES3L = | 000007 | T.EVNT= | 000035 | \$\$\$TYP= | 000027 |
| START | 000166R | T.EVRS= | 000034 | \$\$\$T1 = | 000010 |
| S.CTDB= | 000012 | T.FLG1= | 000015 | .CKEV = | ***** G |
| S.ESB = | 000004 | T.FLG2= | 000026 | .RDEV = | ***** G |
| S.SDB = | 000012 | T.JID = | 000024 | .RSUM = | ***** G |
| S.TDB= | 000074 | T.JNA = | 000011 | .RUN = | ***** G |
| TDB | 000000R | T.LINK= | 000000 | .SETF = | ***** G |
| TD.WT = | ***** GX | T.SCHL= | 000014 | .TDBD = | ***** G |
| TS.USE= | ***** GX | T.SESB= | 000006 | ...PC1= | 000000R |
| TW.TAS= | ***** GX | T.SNDA= | 000000 | ...PC2= | 000074R |
| TW.WT = | ***** GX | T.SNDL= | 000002 | ...TPC= | 000074 |
| T.AST = | 000022 | T.SNDT= | 000010 | | |

. ABS. 000000 000
 000524 001

ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 5729 WORDS (23 PAGES)
DYNAMIC MEMORY: 16016 WORDS (61 PAGES)
ELAPSED TIME: 00:00:22
SY0:[200,140]EXAMPLE1,SPO:[1,4]EXAMPLE1/LI:TTM/NL:BIN=SY0:[200,140]EXAMPLE1
SUBTASK1 MACRO D1110 26-MAY-78 13:59 PAGE 1

Examples of TCS Macros

```

1          .TITLE  SUBTASK1
2          ;
3          ;
4          ;           S U B T A S K 1
5          ;           - - - - -
6          ;
7          .MCALL  SPND$$,WTSE$$,QIOW$$,EXIT$$,CLEF$$
8 000000 MES:    .ASCII <15><12>/SUBTASK RESUMED/
9          MESL=-MES
10         .EVEN
11         ;
12 000022 START:
13 000022 CLEF$$ #1          ;ENSURE FLAG IS CLEAR
14 000034 SPND$$          ;SUSPEND
15 000042 QIOW$$ #IO.WLB,#5,#4,,, <#MES,#MESL>
16 000102 WTSE$$ #1          ;WAIT UNTIL FLAG IS SET
17 000114 EXIT$$
18         ;
19         .END    START
SUBTASK1  MACRO D1110 26-MAY-78 13:59 PAGE 1-1
SYMBOL TABLE
IO.WLB= ***** GX          MESL = 000021          $$$ARG= 000002
MES      000000R          START  000022R          $$$T1 = 000010
. ABS. 000000          000
        000122          001
ERRORS DETECTED:  0
VIRTUAL MEMORY USED:  1223 WORDS  ( 5 PAGES)
DYNAMIC MEMORY:  16016 WORDS  ( 61 PAGES)
ELAPSED TIME:  00:00:05
SY0:[200,140]SUBTASK1,SP0:[1,4]SUBTASK1/LI:TTM/NL:BIN=SY0:[200,140]SUBTASK1
EXAMPLE2  MACRO D1110 26-MAY-78 14:00 PAGE 1

1
2          .TITLE  EXAMPLE2
3          ;
4          ;           E X A M P L E 2
5          ;           - - - - -
6          ;
7          .MCALL  TCSMC$,EXIT$$,QIOW$$
8 000000 TCSMC$
9          ;
10 000000 CTDB:  CDBDF$  BUFF1,LEN1,TS.USE ; DEFINE THE CTDB
11         ;
12 000012 TDB:   TDBDF$
13 000106 TDCM$A  BUFF2,LEN2,TS.USE
14         ;
15 000106 SDB1:  SDBDF$          ; DEFINE AN SDB. THIS...
16         ; WILL BE FILLED IN...
17         ; DURING SEND$T
18 000120 SDB2:  SDBDF$  MESS2          ; USED FOR SENDING BY RUN$T...
19         ; NO DESTINATION NEEDED
20 000132 SDB3:  SDBDF$  MESS3,MESS3L  ; USED FOR RECEIVING...
21         ; BUFFER LENGTH MUST BE...
22         ; SPECIFIED
23         ;
24 000144 ESB:   ESBDF$          ; DEFINE ERROR STATUS BLOCK
25         ;
26         ;
27 000150 BUFF1: .ASCII  / CHAIN.TSK/  ; FILENAME OF CHAIN TASK
28         LEN1=-BUFF1
29         ;

```

Examples of TCS Macros

```

30 000162  BUFF2:  .ASCII  / SUBTASK2/      ; FILENAME OF SUBTASK
31          LEN2=.-BUFF2
32          .EVEN
33          ;
34 000174  MESS1:  .WORD   4                ; LENGTH OF MESSAGE
35 000176          .ASCII  /FGHI/
36          .EVEN
37          ;
38 000202  MESS2:  .WORD   5                ; LENGTH OF MESSAGE
39 000204          .ASCII  /ABCDE/
40          .EVEN
41          ;
42 000212  MESS4:  .ASCII  <15><12>/MESSAGE RECEIVED FROM SUBTASK  =/
43 000254  MESS3:  .ASCII  /              /
44          MESS3L=.-MESS3
45          MESS4L=.-MESS4
46          .EVEN
47          ;
48 000264  MESS5:  .ASCII  <15><12>/RUNNING SUBTASK -- SENDING "ABCDE"/
49          MESS5L=.-MESS5
50 000330  MESS6:  .ASCII  <15><12>/EXAMPLE2 CHAINING/
51          MESS6L=.-MESS6
52          .EVEN
53          ;
54 000354  MESS7:  .ASCII  <15><12>/SENDING "FGHI" TO CHAIN TASK/
55          MESS7L=.-MESS7
56          .EVEN
57          ;
EXAMPLE2  MACRO D1110  26-MAY-78 14:00  PAGE 1-1

58 000412  START:
59 000412          QIOW$$  #IO.WLB,#5,#1,,,,<#MESS7,#MESS7L>
60 000452          SEND$$T  #SDB1,#0,#MESS1 ; SEND TO CHAIN...
61                                     ; DEFINING SDB CONTENTS...
62                                     ; AT SAME TIME
63 000474          BCS      ERR1
64 000476          CHN$$T  #CTDB          ; DECLARE CHAIN TASK
65 000512          BCS      ERR1
66          ;
67 000514          TDPR$$R  #TDB,#JP.PI,#PR.CHN ; SET UP PRIVILEGE...
68                                     ; FOR SUBTASK TO SEND...
69                                     ; AND RECEIVE. NOTE USE...
70                                     ; OF RUN-TIME MACRO.
71 000534          TDBD$$T  ,#ESB          ; DECLARE THE TDB...
72                                     ; AND SET ERROR STATUS...
73                                     ; BLOCK
74 000546          BCS      ERR2
75 000550          QIOW$$S  #IO.WLB,#5,#1,,,,<#MESS5,#MESS5L>
76 000610          RUN$$T   ,,,,#SDB2      ; INITIATE AND SEND
77 000622          BCS      ERR2
78          ;
79 000624          CKEV$$T  ,#TDB,WAIT     ; WAIT FOR EVENT
80 000666          BCS      ERR3
81 000670          RDEV$$T          ; WHAT WAS IT?
82 000674          BCS      ERR2
83 000676          BITB     #IF.SD,T.EVNT(RO) ; SHOULD BE SEND
84 000704          BEQ      ERR4
85 000706          RCV$$T   #SDB3          ; GET THE MESSAGE
86 000716          BCS      ERR5
87 000720          CMP      MESS3,MESS2     ; CHECK MESSAGE LENGTH
88 000726          BNE      ERR4
89 000730          CMP      T.SNDT(RO),#TDB ; CHECK MESSAGE SENDER
90 000736          BNE      ERR4

```

Examples of TCS Macros

```

91 000740      MOV      MESS3,R1          ; MESSAGE LENGTH
92 000744      MOV      #" ",MESS3
93 000752      MOVB     #' ",MESS3+2(R1)
94 000760      QIOW$$   #IO.WLB,#5,#1,,,,<#MESS4,#MESS4L>
95 001020      MOV      #TDB,R1
96 001024      BITB     #IF.JS!IF.JA!IF.NL,T.EVNT(R1)
97
98
99
100
101
102 001032      BNE      EXIT            ; BR IF TASK ALREADY EXITED
103 001034      CKEV$$   ,<R1>,WAIT     ; ELSE WAIT FOR IT...
104
105 001066      BCS      ERR2           ; NOTE USE OF R1 AS PARAM
106
107 001070      ERR1:
108 001070      ERR2:
109 001070      ERR3:
110 001070      ERR4:
111 001070      ERR5:
112 001070      HALT
113 001072      EXIT:
114 001072      QIOW$$   #IO.WLB,#5,#1,,,,<#MESS6,#MESS6L>
EXAMPLE2      MACRO D1110 26-MAY-78 14:00 PAGE 1-2

```

```

115 001132      EXIT$$
116
117
EXAMPLE2      .END      START
EXAMPLE2      MACRO D1110 26-MAY-78 14:00 PAGE 1-3
SYMBOL TABLE

```

| | | | | | |
|---------|----------|---------|----------|------------|---------|
| BUFF1 | 000150R | SDB1 | 000106R | T.LINK= | 000000 |
| BUFF2 | 000162R | SDB2 | 000120R | T.SCHL= | 000014 |
| CTDB | 000000R | SDB3 | 000132R | T.SESB= | 000006 |
| ERR1 | 001070R | START | 000412R | T.SNDA= | 000000 |
| ERR2 | 001070R | S.CTDB= | 000012 | T.SNDL= | 000002 |
| ERR3 | 001070R | S.ESB = | 000004 | T.SNDT= | 000010 |
| ERR4 | 001070R | S.SDB = | 000012 | T.SPR1= | 000032 |
| ERR5 | 001070R | S.TDB = | 000074 | T.SPR2= | 000003 |
| ESB | 000144R | TDB | 000012R | T.SPR3= | 000006 |
| EXIT | 001072R | TD.WT = | ***** GX | T.TESB= | 000020 |
| IF.JA = | ***** GX | TS.USE= | ***** GX | T.TP1 = | 000010 |
| IF.JS = | ***** GX | TW.TAS= | ***** GX | T.TP2 = | 000006 |
| IF.NL = | ***** GX | TW.WT = | ***** GX | T.TSND= | 000016 |
| IF.SD = | ***** GX | T.AST = | 000022 | T.TTYP= | 000005 |
| IO.WLB= | ***** GX | T.CESB= | 000010 | T.UIC = | 000012 |
| JP.PI = | ***** GX | T.CHNA= | 000000 | \$\$\$ARG= | 000002 |
| LEN1 = | 000012 | T.CHNL= | 000002 | \$\$\$M = | 000006 |
| LEN2 = | 000011 | T.CMDA= | 000002 | \$\$\$TYP= | 000001 |
| MESS1 | 000174R | T.CMDL= | 000004 | \$\$\$T1 = | 000010 |
| MESS2 | 000202R | T.CSND= | 000004 | .CHN = | ***** G |
| MESS3 | 000254R | T.CTYP= | 000003 | .CKEV = | ***** G |
| MESS3L= | 000010 | T.EFN = | 000031 | .RCV = | ***** G |
| MESS4 | 000212R | T.ERF = | 000030 | .RDEV = | ***** G |
| MESS4L= | 000052 | T.ERR = | 000000 | .RUN = | ***** G |
| MESS5 | 000264R | T.EVBF= | 000036 | .SEND = | ***** G |
| MESS5L= | 000044 | T.EVNT= | 000035 | .TDBD = | ***** G |
| MESS6 | 000330R | T.EVRS= | 000034 | ...PC1= | 000012R |
| MESS6L= | 000023 | T.FLG1= | 000015 | ...PC2= | 000132R |
| MESS7 | 000354R | T.FLG2= | 000026 | ...PC3= | 000012R |
| MESS7L= | 000036 | T.JID = | 000024 | ...TPC= | 000074 |
| PR.CHN= | ***** GX | T.JNA = | 000011 | | |

Examples of TCS Macros

```

. ABS. 000000      000
        001140      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  5923 WORDS  ( 24 PAGES)
DYNAMIC MEMORY:  16016 WORDS  ( 61 PAGES)
ELAPSED TIME:  00:00:25
SY0:[200,140]EXAMPLE2,SP0:[1,4]EXAMPLE2/LI:TTM/NL:BIN=SY0:[200,140]EXAMPLE2
SUBTASK2          MACRO D1110  26-MAY-78 14:00  PAGE 1

      1              .TITLE  SUBTASK2
      2              ;
      3              ;
      4              ;           S U B T A S K  2
      5              ;           - - - - -
      6              ;
      7              .MCALL  TCSMC$,EXIT$$,QIOW$$
      8 000000      TCSMC$
      9 000000      SDB:   SDBDF$  BUFF,LENGTH
     10              ;
     11 000012      BUFF:   .BLKW   20.
     12              LENGTH=.-BUFF
     13              ;
     14 000062      MESS1:  .ASCII  <15><12>/SUBTASK EXITING/
     15              MESS1L=.-MESS1
     16              ;
     17 000103      MESS2:  .ASCII  <15><12>/SUBTASK ERROR/
     18              MESS2L=.-MESS2
     19              .EVEN
     20              ;
     21 000122      START:
     22 000122      RCV$T   #SDB           ; GET THE MESSAGE
     23 000132      BCS     ERR
     24 000134      CMP     T.SNDT(R0),#-1 ; SHOW IT CAME FROM OWNER
     25 000142      BNE     ERR
     26 000144      SEND$T           ; SEND IT BACK...
     27              ;           ; NOTE THAT ALL FIELDS...
     28              ;           ; WILL HAVE BEEN CORRECTLY...
     29              ;           ; SET UP BY RCV$T
     30 000150      BCS     ERR
     31 000152      QIOW$$  #IO.WLB,#5,#1,,,,<#MESS1,#MESS1L>
     32 000212      EXIT$$
     33 000220      ERR:
     34 000220      QIOW$$  #IO.WLB,#5,#1,,,,<#MESS2,#MESS2L>
     35 000260      EXIT$$
     36              ;
     37              .END    START
SUBTASK2          MACRO D1110  26-MAY-78 14:00  PAGE 1-1
SYMBOL TABLE
BUFF      000012R      START  000122R      $$$ARG= 000002
ERR       000220R      S.ESB = 000004      $$$TYP= 000027
IO.WLB=   ***** GX  S.SDB = 000012      $$$T1 = 000010
LENGTH=   000050      T.ERR = 000000      .RCV = ***** G
MESS1     000062R      T.SESB= 000006      .SEND = ***** G
MESS1L=   000021      T.SNDA= 000000      ...PC1= 000000
MESS2     000103R      T.SNDL= 000002      ...PC2= 000000R
MESS2L=   000017      T.SNDT= 000010      ...PC3= 000000
SDB       000000R      T.SPR2= 000003      ...TPC= 000004

. ABS. 000000      000
        000266      001
ERRORS DETECTED:  0

```

Examples of TCS Macros

```

VIRTUAL MEMORY USED: 5663 WORDS ( 23 PAGES)
DYNAMIC MEMORY: 16016 WORDS ( 61 PAGES)
ELAPSED TIME: 00:00:15
SY0:[200,140]SUBTASK2,SP0:[1,4]SUBTASK2/LI:TTM/NL:BIN=SY0:[200,140]SUBTASK2
CHAIN MACRO D1110 26-MAY-78 14:00 PAGE 1

```

```

1          .TITLE  CHAIN
2          ;
3          ;                               C H A I N
4          ;                               - - - - -
5          ;
6          .MCALL  TCSMC$,EXIT$$,QIOW$$
7 000000    TCSMC$
8          ;
9 000000    SDB:   SDBDF$  BUFF,LENGTH
10         ;
11 000012    MESS:  .ASCII  <15><12>/CHAIN MESSAGE RECEIVED =/
12 000044    BUFF:  .ASCII  \      \
13          LENGTH=-.BUFF
14          MESSL=-.MESS
15          .EVEN
16         ;
17 000054    ERROR: .ASCII  <15><12>/ERROR IN CHAIN TASK/
18          ERRORL=-.ERROR
19          .EVEN
20         ;
21 000102    START:
22 000102    RCV$T   #SDB           ; GET A MESSAGE
23 000112    BCS     ERR
24 000114    MOV     BUFF,R1       ; GET ITS LENGTH
25 000120    MOV     #" ",BUFF
26 000126    MOVEB  #' ",BUFF+2(R1)
27 000134    QIOW$$ #IO.WLB,#5,#1,,,,<#MESS,#MESSL>
28 000174    EXIT$$
29         ;
30 000202    ERR:
31 000202    QIOW$$ #IO.WLB,#5,#1,,,,<#ERROR,#ERRORL>
32 000242    EXIT$$
33         ;
34         .END    START

```

```

CHAIN MACRO D1110 26-MAY-78 14:00 PAGE 1-1
SYMBOL TABLE

```

| | | | | | |
|---------|----------|---------|---------|------------|---------|
| BUFF | 000044R | START | 000102R | \$\$\$ARG= | 000002 |
| ERR | 000202R | S.ESB = | 000004 | \$\$\$TYP= | 000027 |
| ERROR | 000054R | S.SDB = | 000012 | \$\$\$T1 = | 000010 |
| ERRORL= | 000025 | T.ERR = | 000000 | .RCV = | ***** G |
| IO.WLB= | ***** GX | T.SESB= | 000006 | ...PC1= | 000000 |
| LENGTH= | 000010 | T.SNDA= | 000000 | ...PC2= | 000000R |
| MESS | 000012R | T.SNDL= | 000002 | ...PC3= | 000000 |
| MESSL = | 000042 | T.SNDT= | 000010 | ...TPC= | 000004 |
| SDB | 000000R | T.SPR2= | 000003 | | |

```

. ABS. 000000 000
        000250 001
ERRORS DETECTED: 0

```

```

VIRTUAL MEMORY USED: 5703 WORDS ( 23 PAGES)
DYNAMIC MEMORY: 16016 WORDS ( 61 PAGES)
ELAPSED TIME: 00:00:15
SY0:[200,140]CHAIN,SP0:[1,4]CHAIN/LI:TTM/NL:BIN=SY0:[200,140]CHAIN

```

Index

A

- Aborting subtasks
 - by owner • 3-2
 - by system manager • 3-2
 - ABRT\$T • 8-1
 - Assembly time
 - declaration • 1-3
 - definition • 1-3
 - Assembly time declaration of TCS data structures • 7-5
 - Auto-install • 3-4
 - Auto-suspendable
 - as default • 5-1
-

C

- CDBDF\$ macro • 6-1, 8-4
 - CDDF\$R macro • 8-5
 - Chain task
 - attributes of • 6-1
 - privileges • 6-1
 - Chain task descriptor block (CTDB) • 6-1
 - CHN\$T macro • 8-6
 - CKEV\$T macro • 8-8
 - CLI • 1-1
 - as controller • 2-1
 - facilities available to • 2-1
 - functional aspects of • 2-1
 - guidelines for programming language • 2-4
 - CLI characteristics • 1-1
 - CLI privileges
 - at installation • 7-2
 - CLI subtasks • 1-2
 - CTC\$T macro • 8-11
 - CLAIM parameter • 5-2
 - Ctrl/C** task • 5-1
-

D

- Data block fields
 - addressing • A-1
-

- Data definition TCS macro
 - assembly time • 1-2
 - run time • 1-2
-

E

- Error status block
 - address of • 4-1
 - ESBDF\$ macro • 8-13
 - Events • 3-2
-

F

- Fields in the buffer • 3-6
-

I

- Imperative TCS macro • 1-2
 - Interface between owner and subtask • 3-1
-

J

- JNOD\$T macro • 8-14
-

M

- MACRO-11 assembler
 - error-flagging • 7-3
 - .MCALL directive • 7-1
 - Message buffer • 4-1
 - Message length • 4-1
-

O

- Offset names
 - for accessing fields • 6-1
-

Index

P

Parameters
 optional • 7-4
Passing control from task to task • 6-1
Privileges for controlling TCS facilities • 7-2
Processor status word
 carry bit • 7-3

R

RCV\$T macro • 4-3, 8-16
RDEV\$T macro • 8-18
Registers
 saved by macros • 7-5
 saving by macros • 7-5
Required macros • 1-1
Reserving data space • 3-3
RSAS\$T macro • 8-20
RSUM\$T macro • 8-22
RUN\$T macro • 4-2, 8-24

S

Safeguarding user programs • 3-3
Scheduler queue • 4-2
Scheduling level • 3-5
SDBDF\$ macro • 8-27
 use of • 4-1
SDDF\$R macro • 4-1, 8-29
SEND\$T macro • 8-31
Send/receive buffer length • 4-1
Send/receive data block
 fields in • 4-1
Send data block (SDB) • 4-1
SETF\$T macro • 8-33
SHUT\$T macro • 8-35
Specialized applications • 2-1
SPND\$T macro • 8-37
Subtask
 privileges • 3-5
Subtask owner • 3-1
Subtasks
 accounting for • 3-7
 relationship to owner tasks • 3-1
System resources

System resources (Cont.)
 minimizing • 6-1

T

TASK2
 action • 5-2
Tasks that use TCS
 privileges of • 7-2
TCOFF\$ macro • 8-39
TCP • 1-1
TCS • 1-1, 7-1
TCS contents • 1-1
TCS facilities
 initiating • 1-3
TCS facility • 5-1
TCS initialization • 7-1
 automatic • 7-1
TCS macro
 types of • 7-4
TCS macro description
 format • 8-1
TCS macros • 7-1
 CDBDF\$ • 8-4
 CDDF\$R • 8-5
 CHN\$T • 8-6
 CKEV\$T • 8-8
 CTC\$T • 8-11
 ESBDF\$ • 8-13
 JNOD\$T • 8-14
 RCV\$T • 8-16
 RDEV\$T • 8-18
 RSAS\$T • 8-20
 RSUM\$T • 8-22
 RUN\$T • 8-24
 SDBDF\$ • 8-27
 SDDF\$R • 8-29
 SEND\$T • 8-31
 SETF\$T • 8-33
 SHUT\$T • 8-35
 SPND\$T • 8-37
 TCOFF\$ • 8-39
 TCSMC\$ • 8-40
 TDBD\$T • 8-42
 TDBDF\$ • 8-41
 TDBR\$T • 8-44
 TDCM\$A • 8-46
 TDCM\$R • 8-47
 TDEB\$A • 8-48
 TDOFF\$ • 8-49

TCS macros (Cont.)

TDPR\$A • 8-50
 TDPR\$R • 8-52
 TDTA\$A • 8-54
 TDTA\$R • 8-56
 TEOFF\$ • 8-58
 TINIT\$ • 8-59
 TKST\$T • 8-61
 TSOFF\$ • 8-63
 TCS Macro section • 8-1
 TCSMC\$ macro • 8-40
 TDBD\$T macro • 8-42
 TDBDF\$ macro • 8-41
 TDBR\$T macro • 8-44
 TDCM\$A macro • 8-46
 TDCM\$R macro • 8-47
 TDEB\$A macro • 8-48
 TDOFF\$ macro • 8-49
 TDPR\$A macro • 8-50
 TDPR\$R macro • 8-52
 TDTA\$A macro • 8-54
 TDTA\$R macro • 8-56

TEOFF\$ macro • 8-58
 Timesharing control services • 1-1
 Timesharing tasks
 CLIs • 1-1
 maximum number of • 2-3
 TINIT\$ macro • 8-59
 TKST\$T macro • 8-61
 TSOFF\$ macro • 8-63

U

UIC
 for running subtask • 3-5
 Undefined C-bit value • 7-6
 User/system interface
 MCR • 2-1
 PDS • 2-1
 User subroutines
 error-handling • 7-4

Reader's Comments

This form is for document comments only. Digital will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent:

- Assembly language programmer
- Higher-level language programmer
- Occasional programmer (experienced)
- User with little programming experience
- Student programmer
- Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code _____

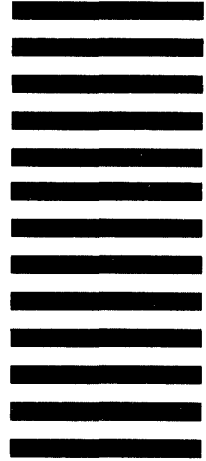
or Country

Do Not Tear - Fold Here and Tape

digital™



No Postage
Necessary
if Mailed in the
United States



BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO 33 MAYNARD MASS

POSTAGE WILL BE PAID BY ADDRESSEE

IAS Engineering/Documentation
Digital Equipment Corporation
5 Wentworth Drive GSF/L20
Hudson, NH 03051-4929



Do Not Tear - Fold Here