

STROBE MVS

STROBE Interface Feature

Release 3.0



Please direct questions about STROBE MVS
or comments on this document to:

STROBE MVS Technical Support

Compuware Corporation
124 Mount Auburn Street
Cambridge MA 02138-5758
1-800-585-2802 or
1-617-661-3020
1-617-498-4010 (fax)
strobe-sup@compuware.com

Outside the USA and Canada, please contact
your local Compuware office or agent.

This document and the product referenced in it are subject to the following legends:

Copyright 2002 Compuware Corporation. All rights reserved. Unpublished rights reserved under the Copyright Laws of the United States.

U.S. GOVERNMENT RIGHTS-Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in Compuware Corporation license agreement and as provided in DFARS 227.7202-1(a) and 227.7202-3(a) (1995), DFARS 252.227-7013(c)(1)(ii) (OCT 1988), FAR 12.212(a) (1995), FAR 52.227-19, or FAR 52.227-14 (ALT III), as applicable. Compuware Corporation.

This product contains confidential information and trade secrets of Compuware Corporation. Use, disclosure, or reproduction is prohibited without the prior express written permission of Compuware Corporation. Access is limited to authorized users. Use of this product is subject to the terms and conditions of the user's License Agreement with Compuware Corporation.

STROBE, iSTROBE, and APMpower are trademarks or registered trademarks of Compuware Corporation.

AD/Cycle, BookManager, CICS, DB2, IBM, IMS/ESA, Language Environment, MQSeries, OS/2, Software Mall, VisualGen, and VTAM are trademarks of International Business Machines Corporation. Microsoft is a registered trademark of Microsoft Corporation. Windows, Windows NT, and Windows 98 are trademarks of Microsoft Corporation. Attachmate and EXTRA! are registered trademarks of Attachmate Corporation. RUMBA is a registered trademark of Wall Data Inc. WRQ and Reflection are registered trademarks of WRQ, Inc.

Adobe ® Acrobat ® Reader copyright © 1987-2001 Adobe Systems Incorporated. All rights reserved. Adobe and Acrobat are trademarks of Adobe Systems Incorporated.

All other company and product names are trademarks or registered trademarks of their respective owners.

Contents

Summary of Changes	v
Changes to the STROBE Interface Feature	v
Changes to this Manual	v
Introduction	vii
How This Manual Is Organized	vii
How to Use This Manual	vii
The STROBE Library	vii
STROBE Feature Manuals	viii
Online Documentation	viii
Online Help	viii
Other Compuware Application Performance Management Products	viii
iSTROBE	ix
SQL Analysis Feature	ix
APMpower	ix
Compuware APM Technical Support	ix
Compuware APM Training	ix
Compuware APM Service Offerings	x
APM Installation Assurance	x
Application Performance Management Consulting	x
Application Performance Assessment	x
Chapter 1. Overview	1-1
Data Collector Programs	1-1
Identifying Modules	1-1
Identifying Transactions	1-1
Other Uses	1-2
Chapter 2. Data Collector Programs	2-1
Functions of Data Collector Programs	2-1
Identifying Transactions	2-1
Identifying Active Load Modules	2-1
Supplementing STROBE Features	2-1
Other Data Collector Functions	2-2
Structure of Data Collector Programs	2-2
Parameter List	2-2
DCCPCST	2-2
DCCLIST	2-3
Register Usage	2-3
Data Collector Communications Area	2-3
DCILCPSW	2-4
DCILCTCB	2-4
DCILCBAS	2-4
DCILCSIZ	2-4
DCILCNMA	2-4
DCILCTRA	2-4
DCILCOLA	2-4
DCILCIDA	2-4
Supplementing Measurement Data	2-4
Identifying Active Load Modules	2-5
Identifying Transactions	2-5

Operational Considerations	2-5
Link Editing a Data Collector	2-6
Invoking a Data Collector	2-6
AMODE/RMODE Considerations	2-6
Programming Considerations	2-7
Appendix A. Examples of Data Collector Programs	A-1
Data Collector Program Showing Module Identification.	A-1
Data Collector Program Showing 4GL Attribution.	A-4
Index	I-1

Summary of Changes

This section lists the changes to the Interface Feature for STROBE MVS for Sysplex Release 3.0.

Changes to the STROBE Interface Feature

The technical content of the Interface Feature has not changed since STROBE MVS for Sysplex Release 2.5.

Changes to this Manual

No changes have been made to this manual for STROBE MVS for Sysplex Release 3.0.

Introduction

This manual describes the STROBE Interface Feature, a product that extends the capabilities of the STROBE MVS Application Performance Measurement System by providing an interface between STROBE and user-written data collector programs. This manual explains how to write data collector programs to gather supplemental performance data.

How This Manual Is Organized

Chapter 1, “Overview” presents an overview of the STROBE Interface Feature.

Chapter 2, “Data Collector Programs” describes data collector programs.

Appendix A, “Examples of Data Collector Programs” contains examples of data collector programs.

How to Use This Manual

Read Chapter 1, “Overview” for an overview and definition of data collector programs. Read Chapter 2, “Data Collector Programs” to learn how to write a data collector program. Appendix A, “Examples of Data Collector Programs” provides two sample data collector programs that you can use as bases for your own programs.

You need experience with IBM assembler and job control language (JCL) to write your programs.

The STROBE Library

The STROBE base product manuals include:

- *STROBE MVS Concepts and Facilities*, document number CWSTGX3A
STROBE MVS Concepts and Facilities explains how to decide which programs and online regions to measure, when to measure them, and how to interpret the reports in the STROBE Performance Profile.
- *STROBE MVS Messages*, document number CWSTXM3A
STROBE MVS Messages lists all messages and abnormal termination (ABEND) codes, describes how to interpret them, and in many cases suggests a corrective action.
- *STROBE MVS System Programmer's Guide*, document number CWSTXI3A
The *STROBE MVS System Programmer's Guide* explains how to install and maintain STROBE.
- *STROBE MVS User's Guide*, document number CWSTUX3A and the *STROBE MVS User's Guide with Advanced Session Management*, document number CWSTUA3A
The *STROBE MVS User's Guide* explains how to use STROBE to measure application performance. The *STROBE MVS User's Guide with Advanced Session Management* explains how to use STROBE with the STROBE Advanced Session Management Feature to

measure application performance. Users who have the STROBE Advanced Session Management Feature will use this manual rather than the *STROBE MVS User's Guide*.

- *STROBE MVS Application Performance Measurement System Quick Reference*

The *STROBE MVS Application Performance Measurement System Quick Reference* is a convenient reference for how to use STROBE and for interpreting the STROBE Performance Profile.

STROBE Feature Manuals

These manuals describe the optional features of the STROBE MVS Application Performance Measurement System. Each manual describes measurement concepts applicable to and specific data made available by the feature.

- *STROBE MVS User's Guide with Advanced Session Management*, document number CWSTUA3A
- *STROBE ADABAS/NATURAL Feature*, document number CWSTUN3A
- *STROBE CA-IDMS Feature*, document number CWSTUR3A
- *STROBE CICS Feature*, document number CWSTUC3A
- *STROBE COOL:Gen Feature*, document number CWSTUG3A
- *STROBE CSP Feature*, document number CWSTUP3A
- *STROBE DB2 Feature*, document number CWSTUD3A
- *STROBE IMS Feature*, document number CWSTUI3A
- *STROBE Interface Feature*, document number CWSTUF3A
- *STROBE Java Feature*, document number CWSTUJ3A
- *STROBE MQSeries Feature*, document number CWSTUM3A
- *STROBE UNIX System Services Feature*, document number CWSTUU3A

Online Documentation

STROBE manuals are available in HTML, Adobe Acrobat PDF format, and IBM BookManager format, on CD-ROM and at Compuware's technical support Web site at <http://frontline.compuware.com>.

Online Help

STROBE products provide the following online information:

- STROBE/ISPF Online Tutorials, Option T from the STROBE/ISPF STROBE OPTIONS menu
- STROBE/ISPF Online Message Facility, Option M from the STROBE/ISPF STROBE OPTIONS menu

Other Compuware Application Performance Management Products

The following products and features work in conjunction with the STROBE MVS Application Performance Measurement System. These tools extend the benefits of application performance management (APM).

iSTROBE

iSTROBE enables you to view and analyze STROBE Performance Profile data on a workstation using a standard Web browser. Easy to install and easy to use, iSTROBE guides you through the performance analysis process and offers recommendations for improving performance. iSTROBE simplifies the performance analysis of applications that you measure with STROBE. For more information on iSTROBE, see the *iSTROBE Getting Started Guide*.

SQL Analysis Feature

The SQL Analysis Feature works in conjunction with STROBE and iSTROBE or APMpower to supply access path analyses and database and SQL coding recommendations for DB2 applications measured by STROBE. The SQL Analysis Feature pinpoints the most resource-consumptive static or dynamic SQL statements, explains why these statements might be inefficient, and provides recommendations to improve the performance of the DB2 application. For more information on the SQL Analysis Feature, see the *STROBE MVS User's Guide* or the *STROBE MVS User's Guide with Advanced Session Management*.

APMpower

The APMpower Application Performance Analysis System extends the benefits of STROBE to application developers who use workstations to develop, test, and maintain MVS applications. Developers employ the APMpower graphical user interface and advanced analytical aids to navigate the Performance Profile, analyze and improve application performance, and share performance knowledge across the IS organization. For more information about APMpower, see the APMpower documentation.

Compuware APM Technical Support

For North American customers, for technical support, please contact the Technical Support department by telephone at (800) 585-2802 or (617) 661-3020, by fax at (617) 498-4010, or by e-mail at strobe-sup@compuware.com.

To access online technical support, visit Compuware's FrontLine page on the World Wide Web at <http://frontline.compuware.com> and select the product "STROBE and APMpower."

For other international customers, please contact your local Compuware office or STROBE supplier.

Compuware APM Training

Compuware's Education Resources Group offers a range of training options for organizations that use STROBE, iSTROBE, and APMpower. To arrange Application Performance Management training, please contact Compuware at 1-800-835-3190 or visit Compuware's Education Resources Group at <http://www.compuware.com/training>

For other international customers, please contact your local Compuware office or STROBE supplier for a complete list of APM Training offerings.

Compuware APM Service Offerings

For North American customers, for information about current service offerings, please contact your local Compuware sales office or call Compuware Corporate Headquarters at 1-800-COMPUWARE (266-7892) or visit Compuware's APM Product page on the World Wide Web at <http://www.compuware.com/products/strobe>.

For other international customers, please contact your local Compuware office or STROBE supplier for a complete list of Services offerings.

APM Installation Assurance

The APM Installation Assurance service assists you in planning for, installing, customizing and using APM products. The service will help you maximize the value and benefits derived from the APM product family.

Consulting engineers work closely with your IT personnel to understand your operating environment and your organization's APM goals. The engineer will assist you in developing a customization and installation plan for STROBE, iSTROBE, and APMpower. The engineer will oversee the installation process and verify product readiness. The engineer will also help set up measurement request schedules, request groups, history records, AutoSTROBE measurement requests, and will verify the installation of the SQL Analysis Feature.

With APM Installation Assurance services, your organization can immediately maximize the value received from your investment in the APM product family. You will also benefit from a fully customized installation that will enhance the product functionality and increase the automation aspects of your APM initiatives.

Application Performance Management Consulting

The Application Performance Management (APM) Consulting services assist you in identifying and resolving specific performance problems in your OS/390 business-critical applications.

Using STROBE, iSTROBE, and APMpower, consulting engineers work closely with your IT personnel to measure an application's performance, identify performance improvement opportunities and make recommendations for implementing solutions.

With APM Consulting services, your organization cannot only resolve problems quickly and effectively, but also gain the skills necessary to prevent application performance degradation in the future.

Application Performance Assessment

The Application Performance Assessment (APA) service assists you in achieving a higher level of performance for your OS/390 business-critical applications.

Using STROBE, iSTROBE, and APMpower, consulting engineers work closely with your IT personnel to evaluate the efficiency of business-critical applications, identify opportunities for improving performance and document the potential savings that can result from implementing recommended solutions.

With APA services, you cannot only improve application performance quickly and effectively, but also gain the knowledge and skills necessary to implement and sustain a process-oriented application performance management (APM) program.

Chapter 1.

Overview

The STROBE Interface Feature allows you to extend STROBE capabilities by providing interface support for user-written data collector programs.

You can use data collector programs to:

- identify load modules that the system's contents management facility does not load
- supplement the transaction or module-identifying information that STROBE supplies
- gather other types of subsystem-specific data when STROBE has no Feature for that subsystem.

Data Collector Programs

A data collector program is a user-written program that STROBE invokes to supplement performance data collected by STROBE and its features or collect performance data for environments STROBE does not support.

Identifying Modules

If you want STROBE Performance Profile information on any modules not loaded by the contents management facility, you need a data collector program to gather identifying information on these modules. For example, if your application program does not load modules with link, load, attach, or xctl (which employ the contents management facility), you need to write a data collector program to gather information on these modules. STROBE uses the module identifiers supplied by the data collector program as the basis for STROBE execution and wait time reports.

You can also use a data collector program to identify the name of the executing module when neither STROBE nor a STROBE Feature has been able to identify it.

Identifying Transactions

You can write a data collector to supplement transaction information gathered by the STROBE Features. For example, you can write a data collector program to identify the transaction that initiates execution of a transaction-processing module in a transaction-driven subsystem. This transaction identifier forms the basis for reporting in the STROBE Transaction Usage Summary and Transaction Usage by Control Section reports.

Also, many data communications subsystems invoke just one transaction for what may be many different "subtransactions." You can write a data collector program that looks at activity beneath the transaction level to identify these "subtransactions."

Other Uses

Besides module and transaction information, you can use a data collector program to gather other types of performance information. Because of the fixed format of the STROBE reports, the additional information you collect must fit into the eight-character transaction field and the eight-character module name field. This information will override the transaction or module information collected by STROBE. The column titles on the reports, however, may not match the type of information now in the report.

A data collector program can also collect information to:

- identify the calling region for multiple user regions
- identify function names in application development systems
- identify 4GL statements or calling programs responsible for system overhead
- bolster the Performance Profile contents with any other system-specific information (within size constraints).

Chapter 2.

Data Collector Programs

This chapter describes how data collector programs work and explains what information is required in a data collector program. A data collector works very closely with STROBE. Each time it samples the target job step, STROBE attempts to identify the name, load address, and size of the module in which execution or wait is occurring. STROBE then passes control to the appropriate data collector programs. If more than one task is executing in a given sample, STROBE gives control to a data collector for each active task.

Functions of Data Collector Programs

You can use data collector programs to augment the data provided in the STROBE Performance Profile. This section lists some of those uses.

Identifying Transactions

In transaction-driven subsystems, a data collector program can identify the transaction in control. The data collector program supplies transaction names to STROBE, which uses them to produce the Transaction Usage Summary and Transaction Usage by Control Section reports of the Performance Profile.

When STROBE cannot assign activity to a single transaction, a data collector program can generate one or more pseudo-transaction names. Pseudo-transaction names can, for example, denote specific overhead functions.

Identifying Active Load Modules

As STROBE samples, it attempts to identify the module that is currently executing or waiting. That module may be the one in which program execution began; it may be an operating system support module (such as an SVC module or an I/O module); or it may be a module that has been dynamically loaded during program execution.

If your application program has not loaded modules with the system contents management facilities employed by the link, load, attach, or xctl macro instructions, then STROBE cannot identify the module in which activity takes place. You can supply a data collector program to report on modules that STROBE cannot identify so the Performance Profile provides more complete information about the application.

Supplementing STROBE Features

You can write data collector programs to supplement information gathered by STROBE Features. STROBE calls your data collector program after it calls any Features.

For example, STROBE identifies a transaction, which it may label as a true user transaction or as system overhead. (A beginning period "." identifies the name of an overhead transaction.) Your data collector program can change the pointer to the transaction name to identify a subtransaction or to define overhead functions further.

When STROBE cannot identify the module in which execution or wait is occurring, your data collector program may be able to do so.

For each transaction name, STROBE produces a detail line in the Transaction Usage Summary and a report subsection in the Transaction Usage by Control Section report of the Performance Profile.

Other Data Collector Functions

For subsystems that are not transaction-driven, you can write a data collector program that groups functionally related activities under a name that you specify. The data collector program supplies this name to STROBE. STROBE reports activity associated with this name in pseudo-control sections. The name appears in a detail line in the Program Section Usage Summary report and as a header of a subreport in the Program Usage by Procedure report.

Structure of Data Collector Programs

To write a data collector that STROBE can use requires you to follow certain conventions. A data collector program must begin with the following structure:

```

STRBxxxx  CSECT
          USING      *,15
BASE      B          BEGIN
          DC          AL1(FILL-*-1)          length of identifier
          DC          CL17'STROBE COLLECTOR '
          DC          CL1' '
          DC          CL8'STRBxxxx'
IDSIZE    EQU       *-BASE
FILL      DC        XL(64-IDSIZE)'00'
ADDRLIST  DS         OF
          DC          A(0)                  not used
          DC          A(0)                  not used
DCCSCPA   DC        A(0)                  not used
DCCPCST   DC        A(0)                  pseudo-csect table
DCCSW     DC        AL1(0)                not used
DCCLIST   DC        AL3(PROGRAMS)         list of programs supported
BEGIN     DS         OH

```

Parameter List

You need to supply STROBE with some information so it can combine what your data collector provides with the rest of the measurement data in the Performance Profile. The parameter list contains the following fields:

DCCPCST

DCCPCST can contain the address of a table supplying the pseudo-control section names that STROBE uses to condense its reports. STROBE reports activity in functionally related system control program modules (whose names share a common three-character prefix) as occurring in a single pseudo-control section. The name of the control section begins with "." and suggests its function. For example, activity reported in the pseudo-control section .COBLIB comprises activity measured in modules whose names share the common prefix ILB, which identifies some COBOL library modules.

To condense the reports for your modules, create a table that holds the definitions for the pseudo-control sections and give the address of the table in the DCCPCST parameter of the data collector program. Code each entry in the table by specifying a pseudo-control section name and its identifying real module prefix.

```

DC  CL7'AAAAAAA'
DC  CL3'ABC'

```

This code will cause STROBE to report in the Program Section Usage Summary report and the Program Usage by Procedure report the processing of all modules beginning with the characters ABC under the module .SYSTEM and the section .AAAAAAA.

You can specify more than one prefix for a pseudo-control section. For example, you can also specify:

```
DC CL7'AAAAAAA'
DC CL3'DEF'
```

You can supply as many as 25 pseudo-control section entries. Terminate the list with X'FF'.

DCCLIST

DCCLIST contains the address of a list specifying the names of programs supported by the data collector. This list ensures that a data collector program is invoked only for the application program or online subsystem for which it is written. STROBE scans the list, matching each entry against the name of the subject program and the names contained in the MAPPROGRAM parameter list supplied by the user when the user submits a measurement request. (See Chapters 2 and 3 of the *STROBE MVS User's Guide*.) STROBE calls the data collector only when it finds a match. If you explicitly name your data collector program when you submit a measurement request, STROBE does not refer to this list. (See Invoking a Data Collector.)

You can supply any number of programs. Terminate the list with X'80'. To allow STROBE to call your data collector program each time it measures, code a field of asterisks (CL8'*****') in the list.

Register Usage

A data collector program uses the standard IBM linkage conventions. When STROBE calls the data collector, the following registers are set:

R1	address of the data collector communications area (described below)
R13	address of a save area provided by STROBE
R14	return address of STROBE
R15	base address for the data collector program.

The data collector returns control to STROBE through register 14. All registers must be saved and restored.

Data Collector Communications Area

The data collector communications area provides for communication between STROBE and the data collector program. The format is as follows:

DCILCOMM	DSECT		data collector communications area
DCILCPSW	DS	F	current user task PSW instruction address
DCILCTCB	DS	F	address of the current user TCB
DCILCBAS	DS	F	load address of the module in control
DCILCSIZ	DS	F	size of the module in control
DCILCNMA	DS	F	address of the name of module in control
DCILCTRA	DS	F	address of the name of transaction in control
DCILCOLA	DS	F	data collector work address
DCILCIDA	DS	F	address of the data collector ID field

When STROBE calls a data collector, STROBE provides the following information in the data collector communications area.

DCILCPSW

DCILCPSW contains the instruction address from the program status word (PSW) for the task that is active at the time STROBE takes the sample. This field is provided for information only. STROBE does not act on modifications to it.

DCILCTCB

DCILCTCB contains the address of the task control block (TCB) of the user task in control at the time the sample is taken. This field is provided for information only. STROBE does not act on modifications to it.

DCILCBAS

When STROBE has identified the module that is executing or waiting, DCILCBAS contains the load address of the module.

DCILCSIZ

When STROBE has identified the module that is executing or waiting, DCILCSIZ contains the size of the module in bytes.

DCILCNMA

When STROBE has identified the module that is executing or waiting, DCILCNMA provides the address of an eight-byte field containing the name of the module in control. The field must be left-justified and padded with blanks. STROBE sets DCILCNMA to zero if neither STROBE nor any other data collectors have identified the module in control.

DCILCTRA

When a data collector has identified a transaction, DCILCTRA provides the address of an eight-byte field containing the name of the transaction. The field must be left-justified and padded with blanks.

DCILCOLA

DCILCOLA is a four-byte field that is available to your data collector program for any purpose you choose. For example, you can use it to anchor storage acquired via GETMAIN.

DCILCIDA

DCILCIDA contains the address of an eight-byte field supplying information about measured programs or subsystems that is printed in the Measurement Parameters column under the field labeled SUBSYSTEM in the Measurement Session Data report. STROBE initializes this field to zero on each invocation of the data collector.

Supplementing Measurement Data

Your data collector program can supplement data that STROBE collects. All changes caused by the data collector must be indicated in the data collector communications area. Your data collector should not change the PSW instruction address or the TCB address. The data collector program can change:

- the address of the name of a transaction in control at the time of the sample
- the address of the name of a module waiting or executing
- the load address of the module
- the reported size of the module
- the address in the information field.

Identifying Active Load Modules

A data collector can examine the DCILCNMA field to determine if STROBE has identified the module that is executing or waiting. (The DCILCBAS and DCILCSIZ fields are not reliable indicators; they may contain zeros when STROBE identifies activity in an SVC module.) When a data collector program identifies the module, it places the address of the module name in DCILCNMA. The data collector also places the module base in DCILCBAS, and the module size in DCILCSIZ.

STROBE verifies that the PSW instruction address supplied in the data collector communications area fits within the size and base supplied by the data collector program. If the address does not fit, STROBE does not record the module data that the data collector supplies.

If the identified module is also one defined as a pseudo-control section (see “DCCPCST,” above), a data collector can force STROBE to accept the module by:

- setting the value of DCILCBAS to zero
- setting the size value to X'7FFFFFFF'.

If neither STROBE nor any of the active data collector programs can identify the active load module, STROBE attributes execution or wait to a pseudo-control section corresponding to the location of the current PSW (for example, to .PRIVATE if the PSW is within the private area of the address space).

Identifying Transactions

In transaction-driven subsystems, STROBE may not be able to identify the transaction in control. If a data collector program can identify the transaction, it places the address of the name of the transaction in the DCILCTRA field. The name must be in an eight-byte field, left-justified and padded with blanks.

Your data collector program can also attribute activity to pseudo-transactions that may identify overhead functions. If a data collector program cannot otherwise identify a transaction, it should supply a pseudo-transaction name so that STROBE can account for all activity in the transaction reports.

When a data collector program identifies a transaction, STROBE attributes subsequent CPU time to the transaction until the data collector either identifies another transaction or supplies a null transaction identifier (binary zeros) in the field addressed by DCILCTRA.

Operational Considerations

Besides the program structure and reporting requirements you must meet for a data collector, several operation factors should be taken into account and are described in this section.

Link Editing a Data Collector

STROBE loads a data collector program whenever it finds within a load module library a program with the name STRBxxxx, where xxxx is the unique four-character prefix of the name of the target program. To allow STROBE to automatically load your data collector program for the appropriate programs, link edit your data collector with the associated name. If the data collector supports more than one program, supply a STRBxxxx alias for each program whose name begins with a unique four-character prefix. For example:

- to link edit a data collector that STROBE automatically invokes whenever it measures programs MYPROG, MYPROG1, and MYPR1234, code the NAME statement for the link editor input:

```
NAME STRBMYPR
```

- to link edit a data collector that STROBE automatically invokes whenever it measures programs MYPROG, OURPROG, and YOURPROG, code:

```
ALIAS STRBOURP
ALIAS STRBYOUR
NAME STRBMYPR
```

To make your data collector program available to any job step that you want to measure, link edit your data collector program into an authorized link list library. If you do not use an authorized link list library for your data collector, you can specify the name of an authorized library when you add a measurement request through the STROBE command language. (See Invoking a Data Collector.)

Invoking a Data Collector

If you have used the naming conventions described above, STROBE automatically invokes your data collector when it encounters a program with the appropriate four-character prefix, provided that it meets the verification requirements described for the DCCLIST parameter.

Alternatively, if you use the STROBE command language to submit your measurement request, you can specify the name of your data collector and the data set name of the authorized library in which it resides with the DCC and DCCLIB operands with the ADD or CHANGE commands. If you use STROBE/ISPF to submit the measurement request, you can specify the name of your data collector on the STROBE - DATA COLLECTORS panel.

When you explicitly specify a data collector name, STROBE loads and calls the data collector program, checking only for the data collector's identifying constant "STROBE COLLECTOR". (See the previous section called Structure of Data Collector Programs.) When you specify more than one data collector, STROBE executes them in the order you have specified.

AMODE/RMODE Considerations

When STROBE measures programs, it executes and calls a data collector program in 31-bit addressing mode (AMODE 31), supervisor state, and in the protect key of the target program. If your data collector cannot operate in AMODE 31, you can change it to operate in 24-bit mode (AMODE 24). The program must, however, return control to STROBE in 31-bit mode.

The following code is suggested to change from 31-bit mode to 24-bit mode:

```

L      R1,16          load CVT address
TM     116(R1),X'80'  is this XA
BZ     *+10          no
LA     R1,*+6        get next instruction address
BSM    0,R1          go to 24-bit mode
This code is suggested to change from 24-bit mode to 31-bit mode:
L      R1,16          load CVT address
TM     116(R1),X'80'  is this XA
BZ     *+14          no
LA     R1,*+10       get next instruction address
O      R1,=XL4'80000000' set bit for 31-bit mode
BSM    0,R1          go to 31-bit mode

```

Link edit your data collector in RMODE 24.

Programming Considerations

Because STROBE calls the data collector each time it takes a measurement sample, code it to perform efficiently.

Your data collector program *must* be re-entrant.

Your data collector program must determine if the target program is one that it can measure. If not, it must return control immediately to STROBE.

STROBE calls user-written data collectors in user key supervisor state, which is the key of the measured address space.

To obtain storage, specify the storage protection key on the GETMAIN or STORAGE OBTAIN macro. For example:

```

IPK     R2          determine our key
LA      R0,STORAGE_SIZE indicate storage size
STORAGE OBTAIN,ADDR=(8),SIZE=(0),KEY(2),LOC(ANY,ANY),COND=YES
LTR     R15,R15    work ok?
BNZ     ERROR      bif no

```

Because the Supervisor State allows privileged instructions and authorized services to be issued from the user-written data collector, review all customer-written data collectors to ensure that there are no integrity exposures.

To facilitate the process, we have provided a 256-byte workarea in the correct key. When STROBE calls the user-written data, Register 13 points to a 72-byte save area. Following the savearea, there is a 256-byte workarea in the correct key. The workarea is not persistent; its contents will not be preserved across calls to the data collector. You may use the following code to access the workarea

```

STM     14,12,12(13) save caller's registers
LA      R2,72(,R13)  get workarea address
ST      R02,8(R13)   forward chain pointer
ST      R13,4(R02)   backward chain pointer
LR      R13,R02      R13 -> workarea
USING WORKAREA,R13
WORKAREA DSECT

SAVEAREA DS 18F
FREESTORAGE DS 46F

```


Appendix A.

Examples of Data Collector Programs

This appendix contains two data collector programs that you can use as examples to help you write your own programs as explained in Chapter 2, "Data Collector Programs".

Data Collector Program Showing Module Identification

```

*****
*
*   STRBTSMD - SAMPLE DATA COLLECTOR FOR STROBE
*
*   THE PURPOSE OF THIS DATA COLLECTOR IS TO GIVE AN EXAMPLE OF
*   A DATA COLLECTOR THAT IDENTIFIES THE NAME OF A MODULE, WITH
*   ACTIVITY/WAIT, IN A SYSTEM THAT PERFORMS ITS OWN CONTENTS
*   SUPERVISION.
*
*   IF THE MODULE NAME HAS NOT BEEN IDENTIFIED BY STROBE, THEN
*   THIS DATA COLLECTOR WILL RUN THE CONTROL BLOCK STRUCTURES,
*   ASSOCIATED WITH THIS SUBSYSTEM, THAT REPRESENT WHERE PROGRAMS
*   WERE LOADED INTO STORAGE AND ATTEMPT TO IDENTIFY THE PSW
*   THAT WAS PASSED IN THE PARAMETER LIST.
*
*   FOR STROBE REL. 2.2.1 AND HIGHER, THIS EXIT MUST BE CODED
*   RE-ENTERANT, AND THE LOAD MODULE MUST BE MARKED WITH THE
*   RENT (RE-ENTERANT) ATTRIBUTE. BE SURE TO SPECIFY OPTION
*   'RENT' IN THE LINKAGE EDITOR STEP FOR THIS MODULE.
*
*   REVISION HISTORY:
*
*   REV  DATE      CHANGE                                BY
*   ---  -
*   00   27NOV95   NEW MODULE                                XXX
*
*****
STRBTSMD CSECT
STRBTSMD AMODE 31
STRBTSMD RMODE 24
R0      EQU 0      BRANCH WORK REGISTER
R1      EQU 1      BRANCH WORK REGISTER
R2      EQU 2      WORK REGISTER
R3      EQU 3      WORK REGISTER
R4      EQU 4      WORK REGISTER
R5      EQU 5      WORK REGISTER
R6      EQU 6      WORK REGISTER
R7      EQU 7      WORK REGISTER
R8      EQU 8      WORK REGISTER
R9      EQU 9      WORK REGISTER
R10     EQU 10     WORK REGISTER
R11     EQU 11     A(DCILCOMM)
R12     EQU 12     BASE REGISTER
R13     EQU 13     A(INPUT SAVEAREA)
R14     EQU 14     BRANCH WORK REGISTER
R15     EQU 15     BRANCH WORK REGISTER
SPACE
B       BEGIN-STRBTSMD(R15)    BRANCH AROUND HEADER
DC     AL1(FILL-*-1)          LENGTH OF IDENTIFIER
DC     CL17'STROBE COLLECTOR '

```

```

                DC    CL1' '
                DC    CL8'STRBTSMD'
                DC    CL1' '
GENID          DC    CL4'G-00'
IDSIZE        EQU    *-STRBTSMD                CALC SIZE OF HEADER
FILL          DC    XL(64-IDSIZE)'00'          FILL TO EP+64
*
* THE FOLLOWING TABLE MUST BEGIN AT ENTRY POINT +64 BYTES.
*
ADDRLIST      DS    0F
                DC    3A(0)                    UNUSED
DCCPCST       DC    A(PCSECTBL)                A(PSEUDO-CSECT TABLE)
                DC    AL1(0)                    UNUSED
DCCLIST       DC    AL3(PROGRAMS)              LIST OF PROGRAMS SUPPORTED
*
* PSEUDO-CSECT TABLE
*
PCSECTBL      DS    0F
                DC    CL7'TSMSYST'             USE PSEUDO NAME TMSYST
                DC    CL3'TSM'                 FOR ALL MODS TSM*
                DC    X'FF'                    MARK END OF TABLE
*
* LIST OF SUPPORTED PROGRAMS
*
PROGRAMS      DS    0F
                DC    CL8'TSMDRIVR'           RUN WHENEVER INVOKED
                DC    X'80'                    MARK END OF TABLE
*
* START OF COLLECTOR CODE
*
BEGIN         DS    0H
                STM   R14,R12,12(R13)         SAVE REGS
                LR    R12,R15                 SET NEW BASE
                USING STRBTSMD,R12           ESTABLISH ADDRESSABILITY
                LR    R11,R1                  GET A(DCILCOMM)
                USING DCILCOMM,R11           ESTABLISH ADDRESSABILITY
*
* CHECK TO SEE IF THE MODULE HAS ALREADY BEEN IDENTIFIED.
*
                L     R3,DCILCNMA             GET A(MODULE NAME)
                LTR   R3,R3                    IS IT THERE?
                BZ    IDMOD                     NO, TRY TO ID MODULE
                CLI   0(R3),X'00'             MODULE NAME SUPPLIED?
                BNE   EXIT                      YES, NOTHING TO DO
*
* ATTEMPT TO ID THE MODULE. IN ORDER TO DO THIS WE MUST FIRST GET
* ADDRESSABILITY TO TSMANCH, THE TSM SYSTEM COMMON ANCHOR CONTROL
* BLOCK, WHICH SHOULD BE POINTED TO VIA THE R7 SAVED IN THE CURRENT
* TCB.
*
IDMOD         DS    0H
                L     R3,DCILCTCB             @(CURRENT TCB)
                SLL   R3,8                     CLEAN FOR A
                SRL   R3,8                     24-BIT ADDRESS
                L     R3,TCBGRS7-TCB(R3)       GET SAVED R7
                USING TSMANCH,R3              ESTABLISH ADDRESSABILITY
                CLC   TSMYEC,=CL8'TSMANCH'     IS IT THE ANCHOR CB?
                BNE   EXIT                      NO, NOT INTERESTED
*
* GET THE ADDRESS OF THE FIRST TSMPROG, TSM SYSTEM LOADED PROGRAM
* CONTROL BLOCK, AND RUN THE CHAIN ATTEMPTING TO IDENTIFY THE PSW
* ADDRESS PASSED IN THE PARAMETER LIST.
*
                L     R4,TSMPROGL             GET A(1ST TSMPROG CB)
                LTR   R4,R4                    IS IT THERE?
                BZ    EXIT                      NO, CAN'T ID PSW
                DROP  R3                       DROP ADDRESS - TSMANCH
                USING TSMPROG,R4              ESTABLISH ADDRESSABILITY

```

```

                L    R1,DCILCPSW          GET PSW FROM PARMLIST
                LA   R1,0(R1)             AND CLEAR HIGH-ORDER BIT
                B    TSMPLLOOP+4         ENTER LOOP
TSMPLLOOP DS   0H
                L    R4,TPROGNXT         @(NEXT TSMPROG CB)
                LTR  R4,R4               ADDRESS FOUND?
                BZ   EXIT                NO, CAN'T ID PSW
                L    R2,TPROGADD         GET A(LOADED PROGRAM)
                LA   R2,0(R2)           AND CLEAR HIGH-ORDER BIT
                CR   R1,R2               PSW ABOVE PROGRAM START
                BL   TSMPLLOOP          NO, TRY NEXT TSMPROG CB
                L    R3,TPROGLN         GET LENGTH OF LOADED PROG
                LA   R4,0(R2,R3)       CALC END OF LOADED PROG
                CR   R1,R4               PSW IN LOADED PROGRAM?
                BNL  TSMPLLOOP          NO, TRY NEXT TSMPROG CB
*
* THE CURRENT TSMPROG CONTROL BLOCK REPRESENTS THE PROGRAM WITH
* OBSERVED ACTIVITY/WAIT. SAVE THE ADDRESS OF THE PROGRAM NAME, THE
* BASE ADDRESS OF THE PROGRAM, AND THE PROGRAM LENGTH IN THE
* PARAMETER LIST.
*
                LA   R15,TPROGNAM       GET A(LOADED PROGRAM NAME)
                ST   R15,DCILCNMA       AND SAVE IN PARMLIST
                ST   R2,DCILCBAS       SAVE BASE ADDR IN PARMLIST
                ST   R3,DCILCSIZ       SAVE SIZE IN PARMLIST
                DROP R3                 DROP ADDRESS - TSMPROG
*
* PROGRAM EXIT
*
EXIT          DS   0H
                LM   R14,R12,12(R13)   RESTORE REGS
                BR   R14               RETURN TO CALLER
                DROP R11               DROP ADDRESS - DCILCOMM
                DROP R12               DROP ADDRESS - BASE REG
                EJECT
*
DC2_LEN EQU   *-STRBTSMD              LENGTH OF CSECT
*
* DATA COLLECTOR COMMUNICATION AREA DSECT
*
DCILCOMM DSECT
DCILCPSW DS   F                      CURRENT TASK PSW ADDR
DCILCTCB DS   F                      A(CURRENT TASK TCB)
DCILCBAS DS   F                      A(LOAD MOD IN CONTROL)
DCILCSIZ DS   F                      SIZE OF LOAD MOD IN CNTRL
DCILCNMA DS   F                      A(LOAD MOD IN CTL NAME)
DCILCTRA DS   F                      A(TRANSACTION IN CONTROL)
DCILCOLA DS   F                      A(WORK AREA)
DCILCIDA DS   F                      A(COLLECTOR ID FIELD)
                EJECT
*
* MVS DSECTS
*
                PRINT NOGEN
                IKJTCB                 TCB
                EJECT
*
* TSM SYSTEM CONTROL BLOCKS
*
                PRINT GEN
                TSMANCH                TSM SYSTEM ANCHOR CB
                TSMPROG                TSM SYSTEM LOADED PROG CB
                END

```



```

                DC    CL1' '
                DC    CL8'STRBDC1'
                DC    CL1' '
GENID          DC    CL4'G-00'
IDSIZE        EQU    *-STRBDC1                CALC SIZE OF HEADER
FILL          DC    XL(64-IDSIZE)'00'        FILL TO EP+64
*
* THE FOLLOWING TABLE MUST BEGIN AT ENTRY POINT +64 BYTES.
*
ADDRLIST      DS    0F
                DC    3A(0)                    UNUSED
DCCPCST       DC    A(PCSECTBL)                A(PSEUDO-CSECT TABLE)
                DC    AL1(0)                    UNUSED
DCCLIST       DC    AL3(PROGRAMS)              LIST OF PROGRAMS SUPPORTED
*
* PSEUDO-CSECT TABLE
*
PCSECTBL      DS    0F
                DC    X'FF'                    MARK END OF TABLE
*
* LIST OF SUPPORTED PROGRAMS
*
PROGRAMS      DS    0F
                DC    CL8'*****'              RUN WHENEVER INVOKED
                DC    X'80'                    MARK END OF TABLE
*
* START OF COLLECTOR CODE
*
BEGIN         DS    0H
                STM   R14,R12,12(R13)          SAVE REGS
                LR    R12,R15                  SET NEW BASE
                USING STRBDC1,R12              ESTABLISH ADDRESSABILITY
                LR    R11,R1                    GET A(DCILCOMM)
                USING DCILCOMM,R11             ESTABLISH ADDRESSABILITY
*
* GET THE ADDRESS OF THE LOCAL WORK AREA. IF IT HASN'T BEEN
* ALLOCATED THEN ALLOCATE AND INITIALIZE IT.
*
                USING LWA,R10                  ESTABLISH ADDRESSABILITY
                L     R10,DCILCOLA              GET A(LWA)
                LTR   R10,R10                  UNALLOCATED?
                BZ    ALLOCLWA                  YES, GO ALLOCATE IT
                CLC   LWA(4),DC1EYEC           OUR EYECATCH THERE?
                BE    GOT_LWA                  YUP, WE ALREADY HAVE IT
ALLOCLWA      DS    0H
                LA    R2,0                     CLEAR REGISTER 2
                IPK   0                         GET PSW KEY INTO R2
                LA    R3,LWALEN                LENGTH LWA STORAGE
                STORAGE OBTAIN,                ACQUIRE LWA STORAGE          X
                    LENGTH=(R3),              LENGTH                      X
                    LOC=(ANY),                ABOVE THE LINE                X
                    SP=130,                   SUBPOOL 130                    X
                    KEY=(R2),                 IN USER'S KEY                  X
                    COND=NO                   MUST GET IT
                LTR   R15,R15                  DID WE GET IT?
                BNZ   EXITNCHN                 NO, GO EXIT
                LR    R10,R1                    @(ACQUIRED STORAGE)
                LR    R4,R10                    CLEAR
                LA    R5,LWALEN                THE
                LA    R6,DC1EYEC                LOCAL
                LA    R7,L'DC1EYEC             WORK
                MVCL  R4,R6                     AREA
GOT_LWA       DS    0H
                ST    R13,SAVEAREA+4           CHAIN
                LA    R13,SAVEAREA             SAVEAREA
*
* MAKE SURE THE ACTIVITY/WAIT IN TINTERPT, THE 4GL INTERPRETER.
*

```

```

L      R1,DCILCPSW          GET THE CURRENT PSW
LA     R1,0(R1)             AND CLEAN IT UP
BAL    R14,CHKFORTI        GO CHECK FOR TINTERPT
LTR    R15,R15             IS THIS IT?
BNZ    EXIT                NO, WE AREN'T INTERESTED
*
* GET R8 FROM THE CURRENT TCB AND VERIFY THAT ITS IT TMAINCB, THE
* INTERPRETER ANCHOR CONTROL BLOCK.
*
L      R3,DCILCTCB          @(CURRENT TCB)
SLL    R3,8                CLEAN FOR A
SRL    R3,8                24-BIT ADDRESS
L      R3,TCBGRS8-TCB(R3)  GET SAVED R8
CLC    0(8,R3),=CL8'TMAINCB' IS IT TMAINCB?
BNE    EXIT                NO, NOT INTERESTED
*
* NAVIGATE FROM TMAINCB TO THE CURRENT TASK CONTROL BLOCK, TCURTASK,
* AND THEN TO THE CURRENT TASK'S PROGRAM CONTROL BLOCK, TCURPROG.
* TCURPROG CONTAINS THE NAME OF THE PROGRAM AND STATEMENT NUMBER
* CURRENTLY BEING PROCESSED BY THE INTERPRETER.
*
L      R3,X'138'(R3)        @(TCURTASK)
LTR    R3,R3               ADDRESS FOUND?
BZ     EXIT                NO, NOT INTERESTED
L      R3,X'24'(R3)        @(TCURPROG)
LTR    R3,R3               ADDRESS FOUND?
BZ     EXIT                NO, NOT INTERESTED
*
* R3 CURRENTLY POINTS AT THE TCURPROG CONTROL BLOCK.  EXTRACT THE
* ADDRESS OF THE PROGRAM NAME AND STATEMENT NUMBER BEING PROCESSED BY
* THE INTERPRETER AND SAVE THE INFORMATION INTO DCILCOMM.
*
LA     R15,X'18'(R3)        GET A(8-CHAR PROG NAME)
ST     R15,DCILCTRA        AND SAVE AS TRAN NAME
MVC   LWASTMTN(2),=CL2' '  SET 1ST 2-CHARS TO SPACES
MVC   LWASTMTN+2(6),X'20'(R3) GET 6-CHAR STMT #
LA     R15,LWASTMTN        GET A(8-CHAR STMT NUMBER)
ST     R15,DCILCTRA        AND SAVE AS PROG NAME
*
* PROGRAM EXIT
*
EXIT   DS      0H
LA     R13,4(R13)          GET A(ENTRY SAVEAREA)
EXITNCHN DS  0H
LM     R14,R12,12(R13)    RESTORE REGS
BR     R14                RETURN TO CALLER
EJECT

*****
*
* SEARCH THE JPQ TO VERIFY THAT THE PSW ADDRESS PASSED IN R1
* REPRESENTS ACTIVIY/WAIT IN TINTERPT, THE TEST INTERPRETER.
*
* REGS ON ENTRY:
* R1 - ADDRESS TO BE IDENTIFIED. THIS ADDRESS IS ASSUMED TO
* BE ALREADY 'CLEANED' FOR EITHER A 24- OR 31-BIT ADDRESS
* PRIOR TO INVOKING THIS ROUTINE
* R11 - A(DCILCOMM)
* R12 - BASE REG
* R13 - A(INPUT SAVEAREA)
* R14 - RETURN ADDRESS
*
* EXIT REGS: (ONLY CHANGED REGS)
* R15 - 0, IF THE PSW ADDRESS PASSED IN R1 REPRESENTS ACTIVITY/
* WAIT IN TINTERPT.
* R15 - 4, IF THE PSW ADDRESS PASSED IN R1 DOES NOT REPRESENT
* ACTIVITY/WAIT IN TINTERPT.
*

```

```

*****
CHKFORTI DS    OH
          STM   R14,R12,12(R13)          SAVE REGS
          LTR   R1,R1                    ADDRESS PASSED?
          BZ    CFTIRC4                  NO, RETURN W/RC=4
          L     R15,DCILCTCB             @(CURRENT TCB)
          L     R15,TCBJSTCB-TCB(R15)    @(JSTCB)
          L     R15,TCBJPQ-TCB(R15)     @(1ST CDE)
          B     CFTILOOP+4              ENTER LOOP
CFTILOOP DS    OH
          L     R15,CDCHAIN-CDENTRY(R15) @(NEXT CDE)
          LTR   R15,R15                  IS IT THERE?
          BZ    CFTIRC4                  NO, RETURN W/RC=4
          TM    CDATTR2-CDENTRY(R15),CDXLE  XTLST BUILT?
          BZ    CFTILOOP                  NO, TRY NEXT CDE
          TM    CDATTR-CDENTRY(R15),CDMIN  MINOR CDE?
          BO    CFTILOOP                  YES, TRY NEXT CDE
          TM    CDATTRB-CDENTRY(R15),CDIDENTY ALIAS?
          BO    CFTILOOP                  YES, TRY NEXT CDE
          L     R3,CDXLMJP-CDENTRY(R15)   @(XTLST)
          LTR   R3,R3                    IS IT THERE?
          BZ    CFTILOOP                  NO, TRY NEXT CDE
          L     R2,XTLMSBAD-XTLST(R3)     GET A(LOAD MODULE) AND
          LA    R2,0(R2)                  TURN OFF HIGH-ORDER BIT
          CR    R1,R2                    ADDR BEFORE A(LOAD MOD)?
          BL    CFTILOOP                  YUP, TRY THE NEXT CDE
          L     R3,XTLMSBLA-XTLST(R3)     GET LOAD MODULE LENGTH
          SLL   R3,8                      AND CLEAN OFF THE
          SRL   R3,8                      FIRST BYTE
          LA    R2,0(R3,R2)              CALC END ADDR OF LOAD MOD
          CR    R1,R2                    ADDRESS IN LOAD MOD?
          BNL   CFTILOOP                  NO, TRY NEXT CDE
          CLC   CDNAME-CDENTRY(8,R15),=CL8'TINTERPT' IS IT TINTERPT?
          BNE   CFTIRC4                  NO, RETURN W/RC=4
          LA    R15,0                     YES, SET RC=0
          B     CFTIEXIT                  AND GO RETURN
CFTIRC4  DS    OH
          LA    R15,4                     SET RC TO NOT TINTERPT
CFTIEXIT DS    OH
          L     R14,12(R13)              RESTORE R14
          LM    R0,R12,20(R13)          R0-R12
          BR    R14                      RETURN TO CALLER
          DROP  R10                      DROP ADDRESS - LWA
          DROP  R11                      DROP ADDRESS - DCILCOMM
          DROP  R12                      DROP ADDRESS - BASE REG
*
*   CONSTANTS
*
DC1EYEC  DC    CL4'DC1 '                LWA EYECATCH
*
DCOM_LEN EQU   *-STRBDC1                LENGTH OF CSECT
*
*   DATA COLLECTOR COMMUNICATION AREA DSECT
*
DCILCOMM DSECT
DCILCPSW DS    F                        CURRENT TASK PSW ADDR
DCILCTCB DS    F                        A(CURRENT TASK TCB)
DCILCBAS DS    F                        A(LOAD MOD IN CONTROL)
DCILCSIZ DS    F                        SIZE OF LOAD MOD IN CNTR0L
DCILCNMA DS    F                        A(LOAD MOD IN CTL NAME)
DCILCTRA DS    F                        A(TRANSACTION IN CONTROL)
DCILCOLA DS    F                        A(WORK AREA)
DCILCIDA DS    F                        A(COLLECTOR ID FIELD)
          EJECT
*
*   LOCAL WORK AREA DSECT
*
LWA      DSECT

```

```
SAVEAREA DS 18F
LWASTMTN DS CL8
LWALEN EQU *-LWA
EJECT
```

```
*
* MVS DSECTS
*
```

```
PRINT NOGEN
IKJTCB
IHARB
IHACDE
IHAXTLST
END
```

```
SAVEAREA 18 FULLWORDS
ID'D MODULE NAME
```

```
TCB
RB
CDE
XTLST
```

Index

Special Characters

.COBLIB, 2-2
.PRIVATE, 2-5

A

ADD command
 specifying data collector name, 2-6
addressing mode, 2-6
AMODE, 2-6

C

calling sequence, 2-1
CHANGE command, 2-6
combining data collectors, 2-1
controlling module
 load address, 2-4
 name of, 2-4
 size of, 2-4
controlling transaction, name of, 2-4

D

data collector, 1-1, 2-1, 2-7
 addressing mode of, 2-6
 base address of, 2-3
 calling sequence, 2-1
 communications area, 2-3–2-4
 address of, 2-3
 examples
 identifying transactions, A-1
 supplementing CICS data, A-4
 forcing STROBE to load, 2-3
 identifying
 transactions, 2-5
 invoking, 2-6
 modifying sample data with, 2-4
 non-transaction-driven subsystems, 2-2
 operational considerations, 2-7
 order of invocation, 2-6
 programming considerations, 2-7
 register usage, 2-3
 specifying
 the library name, 2-6
 structure, 2-4
 supplementing, 2-1
 transferring control to user-written programs, 2-1
 uses of, 1-1–1-2

DCC operand, 2-6
DCCLIB operand, 2-6
DCCLIST, 2-3
DCCPCST, 2-2
DCILCBAS, 2-3–2-4
DCILCIDA, 2-3
DCILCNMA, 2-3–2-5
DCILCOLA, 2-3–2-4
DCILCOMM, 2-3
DCILCPSW, 2-3–2-4
DCILCSIZ, 2-3–2-4
DCILCTCB, 2-3–2-4
DCILCTRA, 2-3–2-5

I

identifying
 subsystem-specific data, 1-2
information field, 2-5
invoking
 a data collector, 2-6
 order of, 2-6

L

library name, specifying for a data collector, 2-6
link editing
 a data collector, 2-6
load module, identifying, 2-1, 2-5

M

MAPPROGRAM
 parameter, 2-3
modifying
 sample data, 2-4

O

operational considerations for a data collector, 2-7

P

parameter list, 2-2
Performance Profile
 condensing reports, 2-2
 data collectors, 2-1
Program Section Usage Summary report, 2-2–2-3
Program Usage by Procedure report, 2-2–2-3
programming considerations, 2-7
pseudo-control sections, 2-2, 2-5
pseudo-transactions, 2-1
 supplying a name for, 2-5
PSW instruction address, 2-4–2-5

R

register usage, 2-3
RMODE, 2-7

S

sample data, modifying, 2-4
save area, address of, 2-3
STROBE/ISPF
 specifying data collector name and library, 2-6
subsystem-specific
 fields, identifying, 1-2
supplementing data collectors, 2-1
system contents management facilities, 2-1

T

task control block (TCB)
 address of, 2-4
Transaction Usage by Control Section report, 1-1, 2-1
Transaction Usage Summary report, 1-1, 2-1
transactions
 identifying, 1-1, 2-1, 2-4-2-5