

AID (BS2000/OSD) V3.2 Advanced Interactive Debugger

Issue June 2006

Pages 4

AID is a powerful debugging system for program monitoring, error diagnosis and temporary correction of program errors.

Product Characteristics

AID supports symbolic debugging of COBOL, Fortran, C/C++, C, Assembler and PL/I programs, and non-symbolic debugging of any BS2000 programs at machine code level. During symbolic debugging, the source code symbols of the programming languages mentioned can be used to address memory elements and stop labels. Non-symbolic debugging at machine code level can be used wherever symbolic debugging is insufficient or not possible, e.g. because the symbol information is not available.

The following basic functions can be called via special AID commands:

- for runtime monitoring
 - specific source statement types
 - selected events in the program run
 - defined program addresses
- for accessing data fields and modifying field contents
- for management of
 - AID output files and libraries
 - for controlling
 - output datasets
 - AID output media
- for specifying global declarations and providing information services

To enhance ease of use, a HELP function is provided:

- for all AID commands and operands
- to explain the meaning of and possible responses to AID messages.

The user can direct AID to interrupt program execution at specified addresses or when selected statement types are executed or defined events occur, and then to execute subcommands. A subcommand is a single command or a sequence of AID and BS2000 commands. It is defined as an operand of an AID command. Under AID V2.0 or higher, execution of subcommands can be made dependent on conditions. This permits dynamic monitoring of program status or variable values, etc.

A command is available to enable the user to view the level of the call hierarchy at which the program was interrupted, and which modules are in the CALL sequence.

AID can be used to process an executing program or to diagnose a dump in a disk file. It is possible to toggle freely between these two options in the same debugging session,

e.g. in order to compare data in the currently executing program with the data in a dump file.

No additional compiler or linkage runs are required in order to debug a program with AID. There is also no need to reload the program, even for "symbolic" debugging. If the symbolic information is not currently loaded, AID can load it dynamically from a specified (PLAM) module library. When no debugging activities are taking place, the test object runs with no loss of performance.

Functional Description

AID is a test and diagnostic tool for debugging application programs at source language level (high-level language debugger). But it can also be used for debugging any BS2000 programs at machine code level. AID allows the different levels (source code symbols and machine code) to be mixed when memory references are defined, thus creating transitions from symbolic debugging to debugging at machine code level and vice versa.

Enhancements in AID (BS2000/OSD) V3.2 compared to the previous version V3.1:

AID V3.2 supports the Unicode data type NATIONAL implemented with COBOL2000 V1.4.

Besides, Version 3.2 contains low-level upgrades for Unicode assistance and supports CCS names (code character set) of input and output media:

- enhanced %AID command: %AID EBCDIC=.. for adjustment of a EBCDIC table for the conversion between Unicode- and C-strings;
- new data type %UTF16, which corresponds with NATIONAL of COBOL2000
- also supports as CCS names of input and output media UTFE
- shows with %SHOW %CCSN the current valid CCS names

Basic functions of the AID symbolic debugger:

- Read, modify and output registers and user-defined data: Data elements defined in the user program can be addressed and modified interactively using their symbolic names. The uniqueness and qualification rules of the relevant programming language apply during this process. The data is converted and edited in accordance with the data type specified in the user program.

- Set test points

Stop points can be defined using symbolic program labels (logical line numbers, paragraphs, etc.), virtual addresses or memory references. When these points are reached, the program is temporarily halted and, provided a definable condition is met, a predefined command sequence is executed (e.g. output data, resume or terminate the program).

- Control via event classes

The program is likewise interrupted when certain user-definable events occur, e.g. specific errors or execution of particular statement types. A test is then made to verify whether a predefined condition has been met and, depending on the result, a sequence of subcommands is/is not executed.

- Tracing at statement level

AID supports dynamic tracing controlled by means of statement classifications (e.g. procedure trace, assignment trace, control flow trace). AID outputs the source references of the executed statements that correspond to the statement classification.

- Symbolic dump

All data or selected data of programs within the dynamic nesting structure can be edited and output in accordance with this structure.

- Environment control

AID permits debugging of live objects (programs) as well as analysis of dumped objects (dump files), e.g. for comparison purposes. Switching between the different program environments is possible anytime during a session.

- Default information

Output of information on current AID defaults, as well on events/test points set and the actions defined for them.

Program Description

The AID debugging system comprises two components:

- the AID user interface and
- the AIDSYS system interface.

AID is a standalone tool, independent of the various BS2000 system versions. All the necessary system functions are handled via AIDSYS.

AID and AIDSYS are loaded as subsystems by the system administrator. The debugging system is then available to all users. All AID functions are invoked using AID commands. These start with a % character followed by the command name.

AID commands are input and AID messages output via system files. Memory contents can be output to system files or to cataloged files.

When data is output, AID edits the field contents in accordance with the data definition in the source program, i.e. data field length, data type and structures are taken into account and output along with the symbolic names.

When data elements are modified, AID performs the necessary conversions, carries out the transfer according to the data type of the destination field and truncates or pads the source field to correspond to the length of the destination field. Data groups are treated as alphanumeric fields.

To enable the user to reference program names, variable names and source statement lines during symbolic debugging, the required information must be available in the * file or in a library at object compilation time.

This supplementary information consists of two parts:

- LSD records (list for symbolic debugging): list of symbolic names and statements defined in the module
- ESD records (external symbol dictionary): directory of symbolic external references of a module.

Technical data	
TECHNICAL REQUIREMENTS HARDWARE	<p>Business servers of the S series (/390 architecture) Business servers of the SX series (SPARC RISC architecture) Memory requirements: on S-systems: approx. 1.5 MB static, on SX-systems: approx. 3.0 MB static Dynamic: approx. 1.0 MB, depending on the number of debugged CSECTs, for symbolic debugging, several MB, depending on the number of programs with symbolic information (LSD) and the size of the symbolic information.</p>
TECHNICAL REQUIREMENTS SOFTWARE	<p>BS2000/OSD-BC V5.0 or higher for S series servers OSD/XC V1.0 or higher for SX series servers Other subsystems required for AID production operation: LLMAID (supplied as part of AID package) SMI (component of the respective operating system) ANITA (component of the respective operating system) AIDSYS (component of the operating system) AIDSYSA (component of the operating system) In order to analyze dump files generated in a system running under another operating system version, it is always necessary to use the version of ANITA in which the dump file was generated. Optional for symbolic debugging: ■ COBOL85 (BS2000) V2.3 or higher ■ COBOL2000 (BS2000/OSD) V1.2 or higher ■ FOR1 (BS2000) V2.2 or higher ■ PLI1 (BS2000) V4.2 or higher ■ ASSEMBH (BS2000) V1.2 or higher ■ C/C++ (BS2000) V3.1 or higher The LSD information generated by the compilers is a prerequisite for symbolic debugging with AID. The full-featured product variant of the compiler is therefore required for the COBOL85 and COBOL2000 compilers.</p>
OPERATING MODE	Batch and interactive
IMPLEMENTATION LANGUAGE	SPL, assembler
USER INTERFACE	<p>Commands: English Message texts optionally English or German</p>
INSTALLATION	Refer to the relevant Release Notice.
DOCUMENTATION	<p>Documentation in English and German: Debugging of C/C++-Programs (C/C++ V3.0 or higher) Debugging at Machine Code Level Core Manual Debugging under POSIX Debugging of COBOL Programs Tabellenheft (ready reference, in German only) AID Supplement Debugging of PL/I Programs Debugging of ASSEMBH Programs Debugging of FORTRAN Programs</p>
USER REQUIREMENTS	Knowledge of BS2000/OSD

TRAINING	See course offer at: http://www.fujitsu-Siemens.com/training
CONDITIONS	This software product can be purchased by the customer against a single payment or leased in accordance with our conditions for the use of software products.
WARRANTY	Class: A Delivery format: Machine language
ORDERING AND DELIVERY	This software product may be obtained from your local Fujitsu Siemens Computers regional office.