BNL ACCESS ROUTINES FOR THE SSP

USER'S GUIDE

Version 5.10

R. Hackenburg

Brookhaven National Laboratory

MPS Group

September 27, 1989

Table of Contents

Section 1

The SSP Access Routine: SSPAR

SSPAR permits a number of operations on and with the Slac Scanner Processor (SSP), most of which are very general, and a few which assume that the SSP-resident error handler performs certain simple tasks. SSPAR also serves as a general purpose interactive FASTBUS tool. SSPAR exists in three flavors, corresponding to the three interface types available for the BNL SSP software (two versions of the CERN CFI, qv ref. 2, and the BNL simulator, which is not really an interface; qv section 6). To run SSPAR, first set up the necessary VAX/VMS symbols and commands with the DCL command:

```
$ @BNL$SSP:BNLSSPINT
```

Next, select an interface type with a command of the form:

```
$ SELxxx
```

where "xxx" is a three character abreviation of the interface type. Currently, "SELxxx" may be one of:

SELUFI, SELSIM or SELCFI

(abbreviations are acceptable.)

NOTE

> Throughout this document, the term "CFI" is used to mean the CAMAC version of the CFI, and "UFI" is used to mean the UNIBUS/QBUS version of the CFI. qv ref. 2.

To show which interface is currently selected, use the DCL command:

```
$ SHOINT
```

To run SSPAR, use the DCL command:

$ SSPAR

The message "SSPAR Version 5.10" and a one-line interface-type identification should appear, followed by the prompt "Command:". SSPAR will then attempt to open the file SSPAR_BEG.COM in the current default area and execute it, as if it were a set of commands typed on the terminal. If SSPAR_BEG.COM does not exist, a message will be typed out to that effect.

Section 1.1

SSPAR Commands

Except for the special single character commands (see below), the command syntax used by SSPAR is that of a command word followed by one or more optional command modifiers and numerical or symbolic parameters:

com/mod1/mod2.../modn par1 par2 ... parm

SSPAR commands may be abbreviated; if an insufficient number of characters are specifed for a command to be unambiguous, an error message is displayed.

Command modifiers, or options, are preceded with a "/", and follow immediately after the command; no space or other character may separate the command and its modifier. Command modifiers may be used only in conjunction with the command itself; command modifiers may not appear on any parameter. The valid command modifiers are listed after each command's description, but a full description of each of these modifiers is defered until the section 1.2. All commands take the /PRINT modifier, which only works with a VT-220 type terminal with a printer attached to its print-port.

Numerical parameters are interpreted in radix 16 (hexadecimal), unless the parameter is prefixed by a single-character radix modifier; no space, tab, comma, etc. may separate the radix modifier from the parameter, and that modifier affects only that one parameter. The exception to this is the (CAMAC-via-FBD) command FCNA, which accepts decimal arguments by default. Radix modifiers may be used with FCNA.

Some of the SSPAR commands expect a BNLSYS-linked SSP program to be resident in the SSP (qv ref. 1). Any command which accesses a slave which is not in the interface's crate requires all SSPs along the route to that slave to be loaded with a BNLSYS-linked program. qv Ref. 1, section 5.

Special Single Character Commands and Modifiers:

1. @file-spec

Opens the file specifed and interprets its contents as commands. The default file-type is COM, and command files may be nested to a level of 2 (three files openned at once).

2. !

Any characters following a "!" are ignored. Thus, comments may be included in command files.

3. <RETURN>

Repeats last command for RPM, RDM, RBPM, RBDM and RBCSR with arguments appropriately incremented; simply repeats the command for GO, and does nothing if the last command was anything else. Command modifiers that were specifed in the previous command remain in effect.

4. /<RETURN>

Same as return, but decrements arguments instead of incrementing them.

Radix Modifiers:

1. $

Decimal radix. A decimal point is optional.

2. %

Binary radix. No decimal point allowed.

3. O or "

Octal radix. No decimal point allowed.

4. H or X

Hexadecimal radix. No decimal point allowed.

Command Descriptions:

1. ALLOCATE

Allocates the VAX device which is the Fastbus/VAX interface. In the case of the UFI, the logical UFIDEV: is allocated. UFIDEV: is defined to be UUA0: in BNLSSPINT.COM; the user may redefine UFIDEV: if needed. In the case of the CFI, this actually causes the "CERN system (CAMAC) crate" to be "booked" for exclusive use by SSPAR. qv ref. 2. Other users are prevented from accessing the interface after this command is executed, except in the case of the CFI when recovering from a CFI error, during which time the "CERN system crate" is briefly "unbooked". Use the DEALLOCATE command to deallocate the device.

2. ARBITRATE level

   Sets the arbitration level of the interface to "level". The arbitration level defaults to 1.

3. ASSEMBLE p [assembly-code-line]

   This does a simple single-line (no relocatable symbols!) assembly of the assembly-code-line included on the command line, and puts the resulting machine instruction in PM location "p", with "p" specified in bytes. If "p" is zero, and a PM range has been specified (with the "RANGE" command), the assembly will start at the begining of the range (the /NOBASE modifier overrides this feature). If no assembly-code-line is specified, SSPAR goes into single-line assembler mode. In this mode, the user is prompted with the PM address (byte-wise, followed by the actual word-wise PM address in parenthesis) at which the assembled instruction will go, as well as the instruction currently residing in that location, deassembled. The first such prompt is simply "p", followed by "p"+4, etc. If "p" is blank, "p" is taken to be zero. Each line enterred by the user after this prompt is interpretted as an assembly-code-line for the single-line assembler, until the END command (qv) is given or control-z is typed. The single-line assembler/deassembler is explained in more detail in section 1.3.

   Modifiers: /WORD (Affects only the "p" argument.)
   /NOBASE

4. BASE a

   Specifies a base address, "a", which is then added to DM

   addresses specified in future commands before those commands are executed.

   Modifiers: /WORD

5. BRC a

   Does a broadcast-class read from CSR space, function "a", and displays the read data. The upper 3 bytes (if non-zero) of the current primary address, set with the SLOT command, are used as a BSI route. The low byte of the current primary address is ignored.

6. BRD a

   Does a broadcast-class read from data space, function "a", and displays the read data. The upper 3 bytes (if non-zero) of the current primary address, set with the SLOT command, are used as a BSI route. The low byte of the current primary address is ignored.

7. BREAK n,[p or module-name] Requires BNLSYS

Insert breakpoint number "n" at PM location "p" ("p" in bytes), or at the beginning of module "module-name" if the /MODULE modifier is specified. Up to four breakpoints may be specified at any one time - "n" must be a number from 1 to 4. If breakpoint number "n" already is defined when the BREAK command is issued, the previous breakpoint is removed and replaced with the new one, at "p", unless the SSP is currently stopped at that breakpoint. If "p" is zero or blank, and "n" is a number from 1 to 4, breakpoint number "n" is removed. If an attempt is made to insert a breakpoint at a branch-and-link instruction, the existing breakpoint number "n" will be removed, and an error message will be displayed - breakpoints cannot be inserted at branch-and-link instructions.

Modifiers: /WORD
/LINE
/MODULE

NOTE

Breakpoints are implemented as unsupported op-code instructions inserted at the location of the break. This causes the error handler in BNLSYS to be invoked, and also saves the condition code bits in the SSP PSW. Earlier versions of SSPAR did not implement breakpoints in this manner, and hence, lost the condition

code bits. The error handler checks a list of breakpoints, and quickly identifies the upsupported op-code error as a breakpoint. The "broken" instruction is moved to a breakpoint handler in BNLSYS and is executed just before the SSP halts, or breaks. Because the instruction is moved, branch-and-link instructions may not be broken (ie, a breakpoint may not be set there). Similarly, Fastbus instructions may not be broken, since bus mastership is lost whenever a non-Fastbus instruction is executed. See the SINGLE command.

8. BUFFER size

Changes the internal values corresponding to the Fastbus interface's buffer sizes, expressed in 32-bit words. For indirect-mode transfers, the input and output buffer sizes default to 1984 (decimal), which is appropriate for 16K CFI/UFIs. For 64K CFI/UFIs, a good number is 16000 (decimal). By default, SSPAR (and the BSI software in general) does transfers in direct mode, which does not use the CFI/UFI buffer at all, but is still limitted by VAX mapping register availability. The direct mode buffer size limit is 1F800h bytes (126K, decimal, corresponding to 254 VAX mapping registers). qv ref. 2.

Modifiers: /DIRECT
/INPUT
/OUTPUT

9. BWC a

Does a broadcast-class write to CSR space, function "a". The upper 3 bytes (if non-zero) of the current primary address, set with the SLOT command, are used as a BSI route. The low byte of the current primary address is ignored.

10. BWD a

Does a broadcast-class write to data space, function "a". The upper 3 bytes (if non-zero) of the current primary address, set with the SLOT command, are used as a BSI route. The low byte of the current primary address is ignored.

11. BZ

Issues a CAMAC BZ through a Struck-320 FBD, which must previously have been selected with the SLOT command.

12. CABLE Requires BNLSYS

Tells the host that it communicates through a cable segment to an SSP. This will generally not be the situation, but it can be used to "trick" the BSI routines into communicating via the SSP with modules in the same crate segment as the host/Fastbus interface. For example, if the interface's response to a non-zero SS code from a slave is suspected of being in error, issue the CABLE command, then specify a primary address (with the SLOT command), from the BSI default routing table, that communicates with the same slave, but with an SSP doing the actual communication with the slave, not the host/Fastbus interface. See section 5 of the BNL Cross Assembler for the SSP User's Guide for details on the BSI default routing table. See the CRATE command.

13. CRATE Requires BNLSYS

Tells the host that it communicates through a crate segment to an SSP. This is the initialized default. CRATE is used to cancel a CABLE command (qv).

14. DEALLOCATE

Reverses the action of the ALLOCATE command.

15. DEBUG module-name

Same as MODULE and LISTING (qv) combined, using "module-name" as the listing-filename. Valid only if the listing filename is the same as the module name, and if the listing is in the current default directory. Markers must be downloaded. See also the EXAMINE and

DEPOSIT commands.

16. DEPOSIT symbol,value

Invokes the SSP symbolic debugger to deposit "value" in location "symbol" in the previously defined module, using the previously defined VAX-11 MACRO listing file. Markers must be downloaded. See the MODULE, LISTING, DEBUG and EXAMINE commands.

17. DOWN filename

Download a file. If "filename" is not specified, the user is prompted for the file name, of default file type "SSP". The SSP's PSW and register contents are unaffected by the DOWN command. Use the /VER0 modifier when downloading (restoring) files created with the DUMP command.

Modifiers: /NOMARKER
/VER0
/VER1

18. DREG Requires BNLSYS

Dump all registers onto the screen. Register values displayed are not necessarily current, but those when the SSP executed the last "DMPREG" macro-instruction.

19. DUMP filename

Dump the SSP's DM and PM contents into a file. If "filename" is not specified, the user is prompted for the file name, of default file type "EXT". Files created with the DUMP command may be restored to the SSP with the DOWN/VER0 command.

20. END or End-of-File

Terminate single line assembler mode (see the ASSMEBLE command).

21. EXAMINE symbol

Invokes the SSP symbolic debugger to examine the contents of location "symbol" in the previously defined module, using the previously defined VAX-11 MACRO listing file. Markers must be downloaded. See the MODULE, LISTING, DEBUG and DEPOSIT commands.

22. EXIT

Exit SSPAR.

23. FCNA function,crate,slot,subaddress

Issues a CAMAC FCNA through a Struck-320 FBD, which must previously have been selected with the SLOT command. The FCNA command accepts decimal values by default, unlike the other SSPAR commands.

Modifiers: /NOQ

/NOX

24. GO Requires BNLSYS or BNLBAS

Start the SSP executing instructions at the PM location whose address is stored in the SSP's DM location 0. (DM 0 gets loaded into the SSP's PSW.) The SSP's halt code (BNLSYS location $H_CODE, or DM_P_H_CODE, as defined in the FORTRAN INCLUDE files - see section 3.1) is zeroed.

CAUTION

Do not use the "GO" command with SSP programs that have been linked with SSP systems other that BNLSYS or BNLBAS, since other systems may use location $H_CODE for purposes other than a halt-code.

25. HELP

Types a brief synopsis of the SSPAR commands.

26. INITIALIZE Requires BNLSYS

Initialize the SSP - (hex) 8080 is written to DM 0, the BNLSYS error and interrupt counters are zeroed and the BSI error retry counters are zeroed.

27. INTERFACE

Initializes the Fastbus interface. In the case of the CFI, the CERN "system crate" is first initialized with a CAMAC-Z. qv ref. 2.

28. IO

Causes an I/O request by the SSP, if such a request has been made, to be processed. I/O requests, of the sort referred to here, will generally occur only in SSP (experimental) FORTRAN programs or subroutines, and the IO command is intended mainly to be a debugging aid for these I/O requests.

29. LISTING filename

Defines the VAX-11 MACRO listing file to be used in symbolic debugging. See the MODULE, DEBUG, DEPOSIT and EXAMINE commands.

30. LOG filename

Opens a file and begins logging interface and BSI errors in it. The default filetype is "ELG". Because of the BSI error recovery software, some errors may be occuring with the Fastbus-host interface, but which will produce no error messages unless the retry count is exceeded.

Error logging forces each error recovery attempt to make an entry in the log-file. To observe these errors on the screen, specify TT: for the file name.

31. MODULE module-name

Defines the module (SUBROUTINE, PROGRAM, SYSTEM or FIXED_DATA) to be symbolically debugged. Sets the DM and PM base addresses to the beginning of the named module, if present in the SSP. Sets the DM and PM ranges to cover the named module. Markers must be downloaded. See the LISTING, DEBUG, DEPOSIT and EXAMINE commands.

32. NOLOG

Closes the error-logging file openned by the "LOG" command and ceases error-logging.

33. PSW psw[,mar]

Analyzes "psw" as an SSP PSW, displaying an English message describing the PSW. In cases where the MAR is needed, an optional MAR may be specifed.

34. QUIT

Exits SSPAR. Same as EXIT, but may be abbreviated by a single letter, Q.

35. RANGE low,high

Specifies the search range ("low" to "high") for DM or PM (DM is the default), and sets the DM or PM base address to "low". See the SEARCH command.

```
Modifiers: /DM     (default)
           /PM
           /WORD
```

36. RBCSR a,n

Read a block of CSR from the SSP and display it in hex on the screen. "a" is the starting CSR address (in words). "n" is number of CSR words to be read.

37. RBDM a,n

Read a block of DM from the SSP and display it in hex on the screen. "a" is the starting DM address. "n" is number of DM words to be read.

```
Modifiers: /NOBASE
           /WORD
```

38. RBPM a,n

Read a block of PM from the SSP and display it in hex on the screen. "a" is the starting PM address (in bytes, as it would appear in a VAX-11 MACRO listing, NOT in 32-bit words, as PM locations are in fact represented internally, unless the /WORD modifier is used). "n" is the number of PM words to be read. If "a" is zero, and a PM range has been specified (either explicitly, with the "RANGE" command, or implicitly, with the DEBUG or MODULE command), PM locations will be displayed starting at the begining of the range (the /NOBASE modifier overrides this feature). The resulting display consists of four columns:

1. The byte-wise address of a PM location, as would be seen on a VAX-11 MACRO listing.

2. The word-address of the PM location, as the SSP would refer to it (in parenthesis).

3. The hexadecimal contents of that PM location.

4. The deassembled instruction in the PM location. See section 1.3 for more details on the deassembler/reassembler.

Modifiers: /NOBASE
/WORD

39. RCSR c

Read CSR, address "c".

40. RDM a

Read DM location "a".

Modifiers: /NOBASE
/WORD (Affects only repeat-reads: <RETURN> or /)

41. RESET

Issues a Fastbus RB; resets Fastbus. If a primary address is specified (with the SLOT command) which does not reside in the same crate as the host-interface, then the segment in which that primary address is a member will be reset (see "The BNL Cross Assembler for the SSP User's Guide" section 5, the "Buffered Segment Interconnect").

Modifiers: /GLOBAL

42. RETRY retry-count-limit

Sets the retry-on-error count limit.

43. RPM a

Read PM location "a" (address in bytes). If "a" is zero, and a PM range has been specified (either explicitly, with the "RANGE" command, or implicitly, with the DEBUG or MODULE command), the PM location at the begining of the range will be displayed (the /NOBASE modifier overrides this feature). The display is the same as with the RBPM command.

Modifiers: /NOBASE
/WORD

44. SCAN

Scans each slot in the Fastbus crate and identifies any modules that reside there. If a "slot" is specified which does not reside in the same crate as the host-interface, then the segment in which that "slot" is a member will be scanned (see "The BNL Cross Assembler for the SSP User's Guide" section 5, the "Buffered Segment Interconnect", and the "Default Routing Table"). The identifier-bits portion of CSR0 is displayed if a module is found in a slot. If a slot is empty, or the module in a slot does not respond properly, nothing is typed out. If the module in a slot responds with an SS code other than 0 or 2, the SS code is displayed and anything read from CSR0 is disregarded; no module identification is made. The slot number is shown in decimal and in hex, while the ID bits are in hex. The route to the scanned segment is described in a series of "English" lines. See the TREE command.

45. SEARCH h

Search through the contents of the SSP's DM or PM for

the specified hex number; the argument must be an eight-digit hexadecimal number. The wild-card character, "*" may be substituted for any or all of the nibbles (each digit in a hexadecimal number is a nibble, being only half a byte). For example, the command SEARCH/PM 50**C*** searches through all the SSP's PM for ST (store, opcode 50h) instructions that use register 12 (Ch) as a base register. Each find is displayed, along with the address in either DM or PM of the find. If a range has been specified (either explicitly, with the "RANGE" command, or implicitly, with the DEBUG or MODULE command), the search is confined to that range.

Modifiers: /DM
/NOBASE
/PM

46. SHOW feature

Where "feature" is one of (and may be abbreviated):

1. ALL Requires BNLSYS

   Displays all the SHOW features at once.

2. BREAKPOINTS Requires BNLSYS

   Displays all the current SSP breakpoints, and states at which breakpoint, if any, the SSP is halted at. If the SSP is not halted at a breakpoint, the reason that the SSP is halted will be stated, if possible.

3. CALLS Requires BNLSYS or BNLBAS

Lists the sequence of subroutine calls starting with the subroutine in which program control existed at the time the SSP was halted. The PM address of each call is given, as well as the DM base address of each of the listed subroutines. The call trace uses both markers and the subroutines' register save areas to accomplish the trace, and if markers are not downloaded into the SSP or if the last subroutine call was cross assembled with the cross assembler control symbol $SUBTR (qv Ref. 1, section 2.4) set to zero, the trace will not function; this, however, is not the usual mode of operation, requiring deliberate action by the programmer. SHOW CALLS will not work on SSP programs cross assembled and linked in versions earlier than 4.00.

4. ERRORS Requires BNLSYS

Displays the BNLSYS error-counters, as well as the most recent error code (BNLSYS location $ERROR, or DM_P_ERROR). The error code is the value returned by the last trap to the error handler, and identifies the type of error that occured. The first seven possible values (1 through 7) for this number are simply the SS code which caused the error. An English description of the most recent error is also given.

5. INTERFACE

Displays information about the FASTBUS interface.

6. MARKERS Requires BNLSYS

Displays the marker blocks currently resident in the SSP, if possible. If the marker blocks (PSECT MARKER) were not downloaded into the SSP, then the marker blocks from the most recent downloading are displayed. Marker blocks are read in by subroutine DOWN during downloading, whether or not they are actually downloaded into the SSP, and are preserved until SSPAR is exitted, or until another downloading occurs. Each main program and subroutine automatically generates two marker blocks: one giving the DM base address, the other giving the PM starting address of that routine. A third marker block for the "virtual" DM, or VM, base address is generated if the cross assembler control symbol $VMFLG is set by the programmer to a non-zero value. qv Ref. 1, section 2.4.

7. PSW Requires BNLSYS

Types the current contents of $INTPSW (or DM_P_INTPSW; see section 3.1) and a one-line "English" message which assumes that $INTPSW contains an SSP PSW (say, from an error or interrupt) and attempts to explain its meaning.

8. RETRIES

Lists the current retry counters and their limits. There are two retry counters: the NTA retry counter, which has to do with recovering from bad slave-NTA reads, and the interface/BSI error retry counter, which covers all other errors. Such things as non-zero SS codes are not considered to be errors, and no recovery is attempted from these.

9. SYSTEM (default feature) Requires BNLSYS

   Displays BNLSYS key-locations, such as halt-code, error-code, first free DM and PM location, etc.

10. VARIABLE

    Displays the internal BSI variables; see section 2.4.

Modifiers: /DM (Applies to MARKER only)
/PM (Applies to MARKER only)

The SHOW command modifiers must follow the SHOW command itself, not the feature.

47. SINGLE Requires BNLSYS

    Single-steps the SSP through its code, if stopped at a breakpoint. Single-stepping correctly handles conditional branching, but processes Fastbus instructions all at once, stopping at the first non-Fastbus instruction. Single-stepping will not progress past a Branch-and-Link type instruction. See the BREAK command. SINGLE may be repeated by pressing the Return key.

48. SLOT p

    Specify the primary address, "p", of the Fastbus slave with which future communications will be made. Be careful; unless the "$" decimal radix modifier precedes the primary address specified, it will be interpreted as hex. The primary address is usually the Fastbus slot number in which the slave resides, but it may also use the default routing table to specify Fastbus slaves not in the same segment with the host/Fastbus interface. See section 5 of the BNL Cross Assembler for the SSP User's Guide for details on the default routing table. For values of "p" beyond eight bits, BNLSYS is required.

49. TMODE tmode

    Sets the TMODE parameter in the CERN FMODE routine.

50. TREE [filename or "*"]

    Displays a FASTBUS "tree". Similar to the SCAN command (qv), but then continues to scan along the opposing segment of any SSPs encountered, up to a maximum of three SSPs along a given route. In addition, if an FBD

(Struck 320, FASTBUS Branch Driver) is found, a "scan" is performed to see which, if any, CAMAC crates are online. If multiple paths to a given module exist, TREE will find that module once for each possible path. If an optional file name is specified, each SSP encountered is first downloaded with the SSP program specified as "filename", or, if "*" is specified as the file name, the SSP program BNL$SSP:ARBTST.SSP is downloaded.

51. WCSR c,d

    Write "d" into CSR address "c".

52. WDM a,d

    Write "d" into DM location "a".

    Modifiers: /NOBASE

53. WPM a,d

    Write "d" into PM location "a". If "a" is zero, and a PM range has been specified (either explicitly, with the "RANGE" command, or implicitly, with the DEBUG or MODULE command), the PM location at the begining of the range will be written to (the /NOBASE modifier overrides this feature).

    Modifiers: /NOBASE
    /WORD

Hex arithmetic commands:

1. ADD d1,d2

   Add d1 to d2, output in hex and decimal.

2. BITMAP d

   Displays a bit-map of the value "d", broken up into nibbles (4-bits), with each bit numbered. This is useful for interpreting CSR0 and the PSW, among other things.

3. CONVERT d1

   Convert and display the hex, octal, decimal and ASCII representation of the value "d1". To input "d1" values in other than hex, use the radix modifiers $, %, " or O.

4. DIVIDE d1,d2

   Divide d1 by d2, output in hex and decimal.

5. MULTIPLY d1,d2

Multiply d1 by d2, output in hex and decimal.

6. SUBTRACT d1,d2

   Subtract d2 from d1, output in hex and decimal.

Section 1.2

SSPAR Command Modifiers

Modifiers may be abbreviated; an error message will be displayed if the modifier as specified is ambiguous. Since many of the SSPAR command modifiers are common to several commands, and have the same (or similar) effect on each command to which they apply, the modifiers are described in detail here, and are only refered to in a list of valid modifiers following each command description.

1. /DM

   Specifies that the secondary address given as one of the arguments in the command is a DM address.

2. /DIRECT

   (BUFFER command only) Modifies the unbuffered (direct-transfer) buffer size.

3. /GLOBAL

   (RESET command only) Resets each segment along the route specified, instead of only the last segment.

4. /INPUT

   (BUFFER command only) Modifies the input (buffered-transfer) buffer size.

5. /LINE

   (BREAK command only) Causes "p" in the BREAK command to be interpretted as a FORTRAN line number.

6. /MODULE

   (BREAK command only) Causes "p" in the BREAK command to be interpretted as a module name. Sets a breakpoint at the entry point of the specified module.

7. /NOBASE

   Prevents any previously specified base (DM or PM) address from being added to the secondary address given in this command.

8. /NOQ

   (FCNA command only) Sets the FBD Q-mode to not set SS on no-Q.

9. /NOX

   (FCNA command only) Sets the FBD X-mode to not set SS on no-X.

10. /NOMARKER

   (DOWN command only) Causes marker blocks (PSECT MARKER) not to be downloaded into the SSP.

11. /OUTPUT

   (BUFFER command only) Modifies the output (buffered-transfer) buffer size.

12. /PM

   Specifies that the secondary address given as one of the arguments in the command is a PM address.

13. /PRINT

   Requires a VT-220 type terminal with a printer attached to its print-port. All commands may take the /PRINT modifier. The /PRINT modifier causes all output to the terminal to be printed, as well. This modifier must be specified with each command, if printout is desired, unless the <RETURN> auto-repeat command is used, in which case it remains in effect.

14. /VER0

   (DOWN command only) Used to indicate that the file being downloaded is either a product of the BNL Cross Assembler for the SSP, Version 0, or is a file that was

   created with the DUMP command.

15. /VER1

   (DOWN command only) Used to indicate that the file being downloaded is a version 1.00, 1.01 or 1.02 SSP image file.

16. /VM

   (SHOW MARKERS command only) Causes only VM markers to be displayed.

17. /WORD

   Causes secondary addresses specified as command parameters to be interpreted as word addresses instead of as byte addresses (the default). Note that the /WORD modifier must follow the command, not the argument. Commands which take a PM or DM address as an argument all accept the /WORD modifier. If the /WORD modifier is used with these commands, the argument which is the address is interpreted by SSPAR as being a word address; if /WORD is not specified, the address is interpreted as a byte address. This has a somewhat different effect when used with commands that expect DM addresses than those that expect PM addresses, since the SSP actually addresses PM wordwise, but DM byte-wise.

Although PM locations are addressed by the word internally in the SSP, VAX-11 MACRO listings show byte-wise addressing for instructions. The byte-wise PM address is the actual PM address, times 4. Conversely, the actual PM address is the byte-wise PM address with the lower two bits ignored (divided by 4). Anytime SSPAR lists a PM address, the byte-wise PM address is shown, together with the actual, internal PM address in parenthesis, following the byte-wise address. When used with PM addresses, the /WORD modifier merely causes the specified address to NOT be divided by four; the output remains the same, and the repeat function (RETURN key) is unaffected.

When used with DM addresses, the /WORD modifier affects only the output; the address given as a command argument is not divided by four (with or without the /WORD modifier). However, if a block is read, the address on each successive word is shown to be incremented only by one if the /WORD modifier is used, instead of by four, and the repeat function will start reading with an NTA calculated by adding the number of words just read, instead of four times the number of

words just read.

The /WORD modifier thus is used for different reasons when used with PM addresses than with DM addresses. With PM addresses, use the /WORD modifier to avoid multiplying hex addresses by four; with DM addresses, use it when trying to read Fastbus slaves other than an SSP.

Section 1.3

The Single-Line Assembler/Deassembler

SSPAR has the ability to generate machine instructions for single PM locations from ASCII character strings expressed in a simple assembler language, as well as the ability to reverse this process. Through the use of command files, simple assembly programs can be written for use with the single-line assembler.

Assembler

The single-line assembler accepts lines in the form of an opcode mneumonic followed by one or more operands, expressed in the usual IBM-like fashion:

1. R1,R2 RR instructions

If R2 is omitted, it will be taken as zero. For instructions not using the R2 field, R2 will be ignored.

2. R1,D2(X2,B2) RX instructions

If the (X2,B2) part is omitted, X2 and B2 will be taken as zero. D2 must not be omitted. Those Fastbus instructions which do not initiate a primary address cycle, and the WCSR instruction as well, do not have the "R" field as do their equivalents from the full Cross Assembler; do not specify this operand on these instructions, as it will NOT be ignored.

3. R1,R3,D2(B2) RS instructions

For instructions not using the R3 field (the shift instructions), the R3 field must be omitted. D2 must not be omitted.

4. D1(B1),I2 SI instructions

For instructions not using the I2 field, I2 is ignored (SSM and LPSW instructions). D1 must not be omitted.

Labels may not be specified on single-line assembler lines; hence, numerals only may be used as operands. Displacements must always be specified as the SSP expects them. There is no implied base register capability, and displacements in branch instructions (PM addresses) do not get divided by 4, as they do in the BNL Cross Assembler. The single line assembler, unlike the BNL Cross Assembler, assumes the hexadecimal radix. Decimal numbers must be so-indicated with the $ radix modifier. The permitted opcode-mneumonics are those specified in the BNL Cross Assembler for the SSP User's Guide, Appendix G, including all the extended mneumonics. The HALT instruction may also be used, but this generates WCSR0 20(0,0) and expects the (hex) value 40000 to be in DM location 20, otherwise it won't actually halt the SSP. In the event of a single-line assembler error, the message "REASSM-E0-reassembler error." is typed out.

Deassembler

The deassembler, which is invoked when the ASSEMBLE, RPM or RBPM command is issued to SSPAR, translates the data in the PM address specified in the command into a crude assembly-code type ASCII string. This string has an opcode-mneumonic followed by operands, expressed as they are in the single-line assembler. All operands shown are hexadecimal numbers. The deassembler is the only exception to the rule that SSPAR always lists PM addresses byte-wise; displacements in branch instructions, which refer to PM addresses, are shown as they appear in the SSP itself. In the case of conditional branch instructions, the byte-wise PM displacement operand is listed right after the opcode-mneumonic, and is enclosed in square brackets; this number is not itself an operand, but is a translation of the displacement operand which follows (the number immediately preceding the left parentheses). Because some of the IBM extended mneumonics are degenerate, only a subset of the standard IBM extended mneumonics are used by the deassembler; those generally used after a "compare" instruction will work (that is BH, BL, BE, BNH, BNL, BNE; not BP, BM, BZ, BNP, BNM, BNZ, which mean exactly the same things. BO and BNO are allowed). If a conditional branch instruction is encountered for which this is no extended mneumonic, it is displayed as "BC cond.", where "cond." is the condition code on which the branch will occur (first

operand in the machine instruction itself). If a "6E000020" instruction is encountered (WCSR0 20(0,0)), the string "HALT" is typed, since this is a HALT instruction. If a non-existent opcode is found, the string "Unsup. opcode" is typed.

It is advisable to cross-check any work done with the single-line assembler by reading back the SSP's program memory locations that were modified, and verifying that the de-assembled line is the same as the line given to the single-line assembler. A misunderstanding of the operand format of an instruction can result in an incorrectly assembled line. By mimicing the output from the de-assembler when inputting lines to the single-line assembler, one can be sure that the instruction inserted into the SSP's program memory by the single-line assembler will be the intended one (bear in mind, however, that some of the extended mneumonics will be deassembled into their equivalents rather than as they had been specified).

Section 2

User-Callable Subroutines

An extensive library of subroutines is available and, except for the interface-defining routine SSPINTDEF, may be found in the file BNL$SSP:SSPBSI.OLB. See section 4 for a description on linking user-written software with the BNL SSP software. Some routines in SSPBSI.OLB, intended only for internal use in constructing other SSPBSI routines, are undocumented here; such routines are, as are all routines described here, self-documented with comments in the source code.

Section 2.1

General Purpose FASTBUS Access Subroutines

A number of subroutines have been written to facilitate communication between the VAX and the SSP, or any FASTBUS slave, via a FASTBUS interface. Unless otherwise mentioned, these subroutines are defined in the file BNL$SSP:SSPBSI.FOR. Most of these subroutines are based on the "equivalent" CERN CFI routines, such as SSPWD, which is basically the same as the CFI CERN routine FWD. (qv ref. 2.) Such routines (in which the "F" is replaced by an "SSP") differ from the CERN equivalents in the following ways:

1. Initialization - Initialization is done with a single call to INTINI (qv), which takes care of all the particular initialization needs of the interface in use. The CERN "crate", "branch" and "slot" numbers (qv ref. 2.) are set by the routine SSPINTDEF, called by INTINI on initialization and defined in one of the files SSPCFIDEF.FOR, SSPUFIDEF.FOR, or SSPSIMDEF.FOR, depending on the interface type. In general, if alternates to the values provided in these files are wanted, a new version of SSPINTDEF should be created in a separate file. SSPINTDEF is an extremely simple routine, consisting of several FORTRAN arithmetic assignment statements.

2. Number of arguments - the SSP routines define the CERN CFI arrays STATUS and CNTRL (qv ref. 2.) internally, in COMMON /CFIVAR/, defined in BNL$SSP:CFIVAR.INC, so that

STATUS and CNTRL are not passed as arguments. This is transparent to the programmer, who need not worry about STATUS and CNTRL. STATUS may be read by calling the routine INTERR. CNTRL is, generally, not available to the programmer.

3. The CERN CFI block transfer routines use a byte-count argument (qv ref. 2.); the equivalent SSP routines use a (4-byte) word-count instead.

4. For block transfers larger than the interface buffer size, with the CERN CFI routines the programmer must break the transfer up into smaller pieces and keep track of or read the slave's NTA between pieces of the transfer. With the BSI routines, breaking the transfer up into pieces is done internally and transparently. While the BSI software initializes to a sensible and generally safe default NTA strategy, the programmer has some options as to how a slave's NTA is used in such piecemeal transfers, such as whether NTAs are calculated or read directly from the slave, and if they are calculated, whether they should be calculated byte-wise or four-byte-word-wise. The safest method is to read them from the slave (the initialized default), but this results in extra overhead (5% to 10% reduction in speed) which may be eliminated for transactions to a specific slave by setting internal BSI variables NTA_CALC to .TRUE., and NTA_FOURS to .TRUE. if the slave is byte-wise addressable, and .FALSE. otherwise. See section 2.4 for more information on accessing the BSI variables.

5. The CERN routines come in two "flavors" - data space access routines and CSR space access routines. The equivalent SSP routines have three such "flavors", namely data space (DM), CSR space, and program space

(PM). PM space is CSR space in reality, but with PM location 0 having a CSR space address of 40000000h. As the CERN routines use the letters "D" or "C" in the name of the subroutine to indicate the type of space that the subroutine accesses, the SSP routines use the letters "D", "C" and "P".

6. The CERN CFI routines either assume the presence of an SI (Segment Interconnect, qv ref. 3, sect. 10) in the CFI's crate, or else they can communicate only with modules in that same crate. The equivalent SSP subroutines will communicate with other crates connected via cable & crate segments and SSPs that have been downloaded with the BSI software in BNLSYS in a manner transparent to the calling routine, as if the equivalent CFI routine had been called in a system with SIs. The

user does not have any ability to specify routing tables and timeouts, but a fairly versatile "default routing table", described in the BNL Cross Assembler for the SSP User's Guide, section 5, exists as part of the BSI software, and which permits communications to Fastbus slaves as far away from the host/Fastbus interface as 3 segments, not counting the interface's own segment.

7. Most of the SSP routines are FUNCTION subroutines which return the value of the BSI status word, BSI_STATUS, which may also be accessed with the routines VARCOM, VARGET and VARPUT. Also, INTERR may be called to get the last value of BSI_STATUS. These BSI status codes are described in "The BNL Cross Assembler for the SSP User's Guide", Appendix A, under the headings "Error Codes" and "BSI Return Codes". For the purpose of minimizing the work necessary to adapt programs which currently use the CERN CFI routines to using the BSI routines, in the event of a strictly BSI error (one that does not involve the interface) the BSI routines force the CFI status word STATUS(2) to be -1, which is not going to give the correct reason for the error, but will cause the calling routine to be aware of an error without checking anything besides STATUS(2). See the INTERR subroutine description.

8. The SSP routines will attempt to recover from Fastbus/host interface errors by performing various interface-dependent tasks, including issuing a CAMAC Z (CAMAC CFI version only), reinitializing the interface, and retrying the operation up to a user-specified number of times, if the retry-count (INT_FAILS_MAX, accessed with VARCOM, VARGET and VARPUT) is set to be positive. INT_FAILS_MAX defaults to 20. A counter, INT_FAILS, also accessed with VARCOM, VARGET and VARPUT, is incremented on each retry. In addition to interface errors, error recovery is also implemented for garbled NTA registers in slaves. The routine INTPCM, defined in the file BNL$SSP:SSPINT.FOR and which does piecemeal buffer transfers between the host and a Fastbus slave, checks each NTA it reads (if NTA_CALC is .FALSE.) from a slave before it transfers the next piece of the buffer to verify that the NTA makes sense, and, if it doesn't, goes back and retries the last piecemeal transfer,

logging the error by both incrementing the location NTA_FAILS, accessed with VARCOM, VARGET and VARPUT, and making an entry in the error log-file, if logging is enabled by setting BSI_ERR_LOG to .TRUE. with VARCOM or VARPUT, and by setting OUTERR (again, with VARCOM or VARPUT) to the FORTRAN LUN on which error logging is to occur (standard ASCII text, 130-character long records). Unless the location MAX_NTA_FAILS is changed by the

calling routine, also set with VARPUT, the routine will retry up to 100 times, each time it is called, before failing. If it fails 101 times, it returns BSI_STATUS = BS_P_NTA and STATUS(2) = -1. INTPCM is not actually called directly by user-written routines, but is called by the BSI routines whenever a block transfer is requested between a Fastbus slave and the host. INTPCM in turn calls the appropriate interface routines.

9. For 64Kbyte versions of the CFI, the user may increase the buffer size of the CFI that the software uses by setting INT_BUFSIZ_IN and INT_BUFSIZ_OUT, with VARCOM or VARPUT, to 16,000 (words). INT_BUFSIZ_IN and INT_BUFSIZ_OUT default to 1984. Note that the buffer in the CFI available for transfers is less than half the CFI's full memory size (qv ref. 2). To change the "mode" of the CFI (buffered input, direct output, etc.; the "TMODE" argument of the CERN subroutine FMODE - qv ref. 2), a new version of the subroutine SSPINTDEF must be created (a trivial operation; SSPINTDEF contains only a few direct assignment statements), with TMODE set to the desired value. Note that INT_UNBUFMOD, set with VARCOM or VARPUT, if .TRUE. (the default), causes the SSP software to adjust TMODE before and after block transfers, so that all transfers are direct by default. Setting INT_UNBUFMOD to .FALSE. overrides this.

All of these routines actually use the CERN CFI routines to do I/O (or the simulator to fake I/O into a large data array). Currently, the SSP routines that exist and correspond to CERN CFI routines are: SSPR*, SSPW*, SSPR*B, SSPW*B, SSPR*M and SSPW*M with the "*" being either "D", "C" or "P", where the "P" case is actually the same as the "C" case, with an offset of (hex) 40000000 being added to the secondary address. (Program space is CSR space, with a base address of 40000000.) Additionally, the CERN CFI routine FWRB has its equivalent in SSPWRB, with SSPWRBG being a global version of SSPWRB (qv). All of these routines use an internal location, INT_TYPE, set by a call to INTINI(0) or INTINI(2), to decide which interface is being used. The choice of interface type is made at link-time, by linking the appropriate versions of interface or simulator routines and the subroutine SSPINTDEF (called only by INTINI) which sets INT_TYPE. Additionally, if the user wants to allocate the Fastbus interface for his exclusive use, the location INT_ALLOC (with VARCOM or VARPUT) should be set to .TRUE.; INT_ALLOC defaults to .FALSE. . In the CAMAC version of the CFI, "booking" occurs whenever INT_ALLOC is .TRUE. and INTINI(0), INTINI(1) or INTINI(2) is called. The BSI routines which correspond to CERN CFI routines return as their FUNCTION value BSI_STATUS, and use one or more of the following as arguments:

1. PA The primary address of the Fastbus slave.

2. SA The secondary address in the Fastbus slave's data or CSR space.

3. BUFFER A one or more word data buffer, which is either the source or destination of the transfer.

4. NWORDS The number of 32-bit words to be transfered.

The SSP (and BSI) subroutines are:

1. INTEGER FUNCTION INTERR(INTSTAT,NELMTS,C80,IERR)

   In file BNL$SSP:SSPINT.FOR, INTERR returns the interface-dependent error message in C80, according to the current status of the interface and augmented with the BSI error messages. Returns the first NELMTS elements of the interface dependent status array in INTSTAT, a user-supplied 32-bit-integer array, and the interface-dependent error code in IERR. BSI_STATUS is returned as the FUNCTION value.

2. INTEGER FUNCTION INTINI(LEVEL)

   In file BNL$SSP:SSPINT.FOR, this routine performs an initialization of level or type "LEVEL" on the interface.

   LEVEL -1:
   Special level used to initialize the CERN "system crate" with a CAMAC "Z". INTINI(-1) does nothing if the interface is not a CAMAC version of the CFI.

   LEVEL 0:
   Full software (internal variables) and power-up interface initialization, beginning with an arbitration cycle in the Fastbus interface. INTINI(0) returns 0 if successful, -1 if not.

   LEVEL 1:
   This routine resets the interface after a Fastbus error (e.g., the CERN CFI array element STATUS(2) .NE. 0). The first 8 words of the internal array STATUS are preserved intact. STATUS(1 through 8) is accessible with the INTERR routine. INTINI returns the SS code, if non-zero, as the BSI status word BSI_STATUS (in hex format):

   2xx - where "xx" is the SS code, for non-zero SS codes.
   101 - for zero SS codes, indicating an interface error. (The details will be found in STATUS).
   -1 - for general interface intialization error.

LEVEL 2:
Full software (internal variables) initialization. No I/O is done; FASTBUS is not accessed, the interface is not initialized, and no arbitration is done. The hardware initialization occurs automatically on the first actual FASTBUS transaction attempted. INTINI(2) returns 0 if successful, -1 if not.

3. INTEGER FUNCTION OUT_INTERR

   Displays the BSI error/status message from the most recent transaction on BSI device OUTCOM (normally the terminal - see section 2.4) and on device ERRLOG (see section 2.4), if logging errors. OUT_INTERR returns equal to BSI_STATUS.

4. SUBROUTINE SSPINTDEF

   This routine is called only by INTINI, and defines the interface type. At present, there are three versions of SSPINTDEF, contained in the files:

   SSPCFIDEF.FOR - CAMAC version of the CFI.
   SSPSIMDEF.FOR - Simulator "interface/SSP"
   SSPUFIDEF.FOR - UNIBUS/QBUS version of the CFI ("UFI").

   SSPINTDEF initializes values such as the CERN "branch", "crate" and "slot" numbers, the CERN "interface select" number, the CERN subroutine FMODE argument "TMODE", the input and output buffer sizes, and the interface type. Some of these values are unused for some interface types. For example, the simulator doesn't use the branch, crate, slot or "interface select" numbers. Note that if TMODE is changed, the input and output buffer sizes may need to be changed as well.

5. INTEGER FUNCTION SSPRC(PA,SA,BUFFER)
   Reads a single word from CSR space.

6. INTEGER FUNCTION SSPRCB(PA,SA,BUFFER,NWORDS)
   Reads a block of words from CSR space.

7. INTEGER FUNCTION SSPRCM(PA,BUFFER)
   Reads a single word from CSR space, broadcast class (multi-listener), function code PA. If non-zero, the upper three bytes of PA (the function code) are

   interpretted as a BSI route; only the low byte of PA is ever actually used as a function code. See section 5 of "The BNL Cross Assembler for the SSP User's Guide" for a description of the BSI.

8. INTEGER FUNCTION SSPRD(PA,SA,BUFFER)
   Reads a single word from data space.

9. INTEGER FUNCTION SSPRDB(PA,SA,BUFFER,NWORDS)
   Reads a block of words from data space.

10. INTEGER FUNCTION SSPRDM(PA,BUFFER)
    Reads a single word from data space, broadcast class (multi-listener), function code PA. If non-zero, the

upper three bytes of PA (the function code) are interpretted as a BSI route; only the low byte of PA is ever actually used as a function code. See section 5 of "The BNL Cross Assembler for the SSP User's Guide" for a description of the BSI.

11. INTEGER FUNCTION SSPRP(PA,SA,BUFFER)
    Reads a single word from SSP program space.

12. INTEGER FUNCTION SSPRPB(PA,SA,BUFFER,NWORDS)
    Reads a block of words from SSP program space.

13. INTEGER FUNCTION SSPWC(PA,SA,BUFFER)
    Writes a single word to CSR space.

14. INTEGER FUNCTION SSPWCB(PA,SA,BUFFER,NWORDS)
    Writes a block of words to CSR space.

15. INTEGER FUNCTION SSPWCM(PA,BUFFER)
    Writes function code PA to CSR space, broadcast class (multi-listener). If non-zero, the upper three bytes of PA (the function code) are interpretted as a BSI route; only the low byte of PA is ever actually used as a function code. See section 5 of "The BNL Cross Assembler for the SSP User's Guide" for a description of the BSI.

16. INTEGER FUNCTION SSPWD(PA,SA,BUFFER)
    Writes a single word to data space.

17. INTEGER FUNCTION SSPWDB(PA,SA,BUFFER,NWORDS)
    Writes a block of words to data space.

18. INTEGER FUNCTION SSPWDM(PA,BUFFER)
    Writes function code PA to data space, broadcast class (multi-listener). If non-zero, the upper three bytes of PA (the function code) are interpretted as a BSI route; only the low byte of PA is ever actually used as a

function code. See section 5 of "The BNL Cross Assembler for the SSP User's Guide" for a description of the BSI.

19. INTEGER FUNCTION SSPWP(PA,SA,BUFFER)
    Writes a single word to SSP program space.

20. INTEGER FUNCTION SSPWPB(PA,SA,BUFFER,NWORDS)
    Writes a block of words to SSP program space.

21. INTEGER FUNCTION SSPWRB(PA)

    Issues an RB on the segment in which the slave whose primary address is PA is a member. Calling SSPWRB(PA) with PA being in the range 0 to 255 will issue an RB on the interface's segment. BSI_STATUS is returned as the FUNCTION value.

22. INTEGER FUNCTION SSPWRBG(PA)

    Issues an RB (Globally) on all segments along the BSI route leading up to and including the segment in which the slave whose primary address is PA is a member. Calling SSPWRBG(PA) with PA being in the range 0 to 255 will issue an RB on the interface's segment. BSI_STATUS is returned as the FUNCTION value.

Section 2.2

Special Purpose SSP Subroutines

Various special functions peculiar to the SSP are implemented with the subroutines described in this section. These special functions include downloading, single-stepping, debugging, initializing programs, and a whole host of others.

1. INTEGER FUNCTION BREAK(PA,BN,PMLOC)

   Found in file BNL$SSP:BREAK.FOR. Insert "breakpoint" #BN in the SSP whose primary address is PA at PM location PMLOC. The SSP will execute code until it arrives at the breakpoint, then stop until told to resume. A total of 4 breakpoints are allowed at one time. SSPGO is used to resume execution after a

   breakpoint. (The SSPAR "GO" command.) If PMLOC is 0, then the current breakpoint is removed, but no new breakpoint is added. BREAK returns the value of BSI_STATUS.

2. SUBROUTINE CAM_TREE(PA,LU,INDENT)

   Found in file BNL$SSP:SSP_TREE.FOR. On FORTRAN Logical Unit LU, display a CAMAC system tree, showing all on-line CAMAC crates which respond to F(24) N(30) A(9) with CAMAC-X (qv CAM_TREE). An FBD (FASTBUS BRANCH Driver, Struck 320) is assumed to reside in slot or route PA, and INDENT is an integer specifying the indentation (column number) that the messages should start with. qv SSP_TREE.

3. INTEGER FUNCTION DOLD(PA,UNIT)

   Found in file BNL$SSP:SSPDWN.FOR, this simply calls DOWNV0 (qv) and it exists only for the purpose of compatibility with earlier releases.

4. INTEGER FUNCTION DOWN(PA,UNIT,MK_SEL)

   Found in file BNL$SSP:SSPDWN.FOR. Used to download a file, openned on logical unit "UNIT", into the SSP. The file is expected to be an SSP-executable image file, created with a command file similar to BNL$SSP:SLINK.COM or BLINK.COM; DOWN expects a version 2 or later SSP image file (file type SSP). The file needs to be openned with the FORTRAN OPEN qualifiers: READONLY, STATUS='OLD', FORM='UNFORMATTED', RECORDTYPE='FIXED', RECL=128. MK_SEL determines whether the markers generated by the cross assembler are downloaded into the SSP or not. If MK_SEL is 1, the markers are downloaded. If MK_SEL is 0, they are not downloaded. DOWN returns the values:

 0 for success,
-1 for an interface (Fastbus) error,
 1 for an SSP error,
 2 for a file error.

5. INTEGER FUNCTION DOWNV0(PA,UNIT)

   Found in file BNL$SSP:SSPDWN.FOR. Similar to DOWN, but for version 0 SSP image files, or for files produced by subroutine DUMP. See DOWN for a list of return values.

6. INTEGER FUNCTION DOWNV1(PA,UNIT)

   Found in file BNL$SSP:SSPDWN.FOR. DOWNV1 is nearly

   identical to DOWN, but works on version 1 SSP image files. Since there was no marker capability in version 1, the argument MK_SEL does not exist. See DOWN for a list of return values.

7. INTEGER FUNCTION DUMP(PA,UNIT)

   Found in file BNL$SSP:SSPDWN.FOR. Dumps the contents of the SSP into a file, openned on logical unit "UNIT" with the same qualifiers as used with subroutine DOWN. The file has the same structure as a version 0 SSP-executable image file. Returns the values:
   DUMP = 0 for success,
   DUMP = -1 for interface (FASTBUS) error.

8. SUBROUTINE FNDPAT(PATERN,BUFFER,BUF0,BUFL)

   Found in file BNL$SSP:DBGSUB.FOR. Find the hexadecimal pattern PATERN, CHARACTER*8, in BUFFER, starting at BUF0 and ending at BUFL. Any nibbles in PATERN which "don't matter" are indicated by asterisks "*". Matches are displayed on OUTCOM (normally the terminal - see section 2.4). See the SSPAR "SEARCH" command, sect. 1.1.

9. SUBROUTINE GETMRK(MARKER,MPT,MRKWDS,SYSVER,LOC,PID,
   INDENT,TITLE,TMAX,NBYTES,IERR)

   Found in file BNL$SSP:DBGSUB.FOR. Get the information present in a marker entry, starting at MARKER(MPT). MARKER, MPT, MRKWDS and SYSVER are supplied by the calling routine. MARKER is an INTEGER*4 array containing the markers. MPT is incremented to point at the next element in MARKER after one marker has been "extracted" from it. MARKER must first have been loaded, for example, by LODMRK. MRKWDS is the number of marker words contained in MARKER, and should have been set at the time when MARKER was loaded. The BNLSYS system version number is supplied by the calling routine in SYSVER, as a REAL*4 number. LOC is returned by GETMRK, and is the PM or DM address refered to by the marker (markers refer to addresses in SSP memory). PID is the PSECT number of LOC, and determines whether LOC is a PM (PID=PM_P_PID) or a DM (PID=DM_P_PID) address. The marker's character string, length NBYTES, is returned in TITLE, which has at most TMAX characters. TMAX is supplied by the calling routine, and NBYTES is returned by GETMRK. INDENT is the marker's indentation, as stored in the SSP (see "The BNL Cross Assembler for the SSP User's Guide" for a description of markers). IERR is returned as 0 for success, -1 for error.

10. INTEGER FUNCTION IOHAND(PA)

    Found in file BNL$SSP:IOHAND.FOR. "Transparently" handles I/O requests from the SSP, enabling the SSP to use essentially all the I/O devices on the VAX as if they were the SSP's own. This is not fully implemented, and is intended mainly for use with the experimental SSP FORTRAN compiler. IOHAND returns BSI_STATUS.

11. SUBROUTINE LODMRK(DM,MARKER,MARK0,MRKWDS,MRKMAX)

    Found in file BNL$SSP:DBGSUB.FOR. Extract the markers from SSP DM, already present in the array DM. Put the markers in array MARKER. Set MARK0 to be the starting address in DM (bytewise address, not actually an array index), and MRKWDS to be the number of marker words found. MRKMAX, supplied by the calling routine, is the maximum value for MRKWDS (ie, the size of array MARKER) without array overflow.

12. SUBROUTINE SETMDL(MARKER,MRKWDS,NAME,DM0,DML,VM0,VML,PM0,PML,IERR)

    Found in file BNL$SSP:DBGSUB.FOR. Set the "module" to routine NAME (set DM0, VM0 and PM0 to be the base addresses of DM, VM and PM of routine NAME and set DML, VML and PML to be the upper limits). NAME is a 1 to 31 character string (CHARACTER type). Markers (MARKER) are used for this purpose. MRKWDS is the number of marker words contained in array MARKER. IERR returns as zero if successful, -1 if failed.

13. INTEGER FUNCTION SHOW(PA,COM,DMBASE)

    Found in file BNL$SSP:SHOW.FOR. "Show" various items concerning the current status/contents/etc. of the SSP whose primary address is PA. COM is CHARACTER type and contains an ASCII qualifier describing what is to be shown. COM has already been converted to all caps, and the ASCII qualifier begins in COM(1).

    Qualifiers are:
    ALL BREAK CALLS ERRORS MARKERS PSW
    RETRIES SYSTEM VARIABLES
    Default is SYSTEM.
    Returns the value of BSI_STATUS.

14. INTEGER FUNCTION SINGLE(PA)

Found in file BNL$SSP:SINGLE.FOR. Single-step the SSP whose primary address is PA.
SINGLE returns the value of BSI_STATUS.

15. SUBROUTINE SSP_TREE(LU,FU)

Found in file BNL$SSP:SSP_TREE.FOR. On FORTRAN Logical Unit LU, display a FASTBUS tree, consisting of a list of all FASTBUS modules and their locations and identities, and all on-line CAMAC crates which respond to F(24) N(30) A(9) with CAMAC-X (qv CAM_TREE). FU, if greater than zero, is a FORTRAN Logical Unit of an SSP file, openned before calling SSPTREE, and to be downloaded into each SSP as it is encountered. If FU is zero or negative, any SSPs encountered are assumed to be downloaded with BSI software and are left undisturbed after being used as BSIs.

16. INTEGER FUNCTION SSPCMP(DM,DMCHK,PM,PMCHK)

Found in file BNL$SSP:SSPDWN.FOR, this routine compares two pairs of arrays, DM with DMCHK and PM with PMCHK, and alerts the calling routine to any differences, outputting messages on device OUTCOM (qv section 2.4). DM and PM (and DMCHK and PMCHK) are expected to have the sizes DMMAX, accessed with VARCOM, VARGET and VARPUT (qv section 2.4), and SIZ_P_PM, in BNL$SSP:SSPSYM.INC. SSPCMP returns:

0 If DM is identical to DMCHK and PM is identical to PMCHK.
-1 If DM differs from DMCHK or PM differs from PMCHK.

17. SUBROUTINE SSPDTM(DM)

Insert the date and time of loading into array DM, which is an SSP DM image, to be downloaded directly into an SSP with SSPPUT (qv).

18. INTEGER FUNCTION SSPFMT(FUNIT,DM,PM,MM,MRKWDS,MK_SEL)

Found in file BNL$SSP:SSPDWN.FOR, this program will format a file to be downloaded to an SSP, from a file expected to be openned on unit FUNIT when this routine is called, to the arrays DM, PM and MM. MM is expected to have size MRKWDS. DM and PM are expected to have the sizes DMMAX, accessed with VARCOM, VARGET and VARPUT, and SIZ_P_PM, in BNL$SSP:SSPSYM.INC. MRKWDS must be set to the size of the array MM by the calling routine. SSPFMT sets MRKWDS to the number of marker words actually used when SSPFMT is done. MK_SEL determines whether the markers generated by the cross assembler are incorporated into the array DM or not. If MK_SEL is 1, the markers are incorporated into DM. If MK_SEL is 0, they are omitted from DM. Returns:

SSPFMT = 0 for success,
SSPFMT = 1 for file error (eg, file is not an SSP

image).

19. INTEGER FUNCTION SSPGET(PA,DM,PM)

    Found in file BNL$SSP:SSPDWN.FOR, this routine gets (reads) the DM and PM from the SSP whose primary address is PA. DM and PM are expected to have the sizes DMMAX, accessed with VARCOM, VARGET and VARPUT, and SIZ_P_PM, in BNL$SSP:SSPSYM.INC. BSI_STATUS is returned as the FUNCTION value.

20. INTEGER FUNCTION SSPGO(PA)

    Found in file BNL$SSP:SSPBSI.FOR. This routine starts the SSP by writing a bit pattern to its CSR0. SSPGO also clears SSP DM location DM_P_H_CODE, defined in the FORTRAN include file SSPSYM.INC, which is used by the SSP-resident error and interrupt handler to indicate the reason that the SSP is halted; the system routines always update this location. The SSPAR command "SHOW BREAKPOINT" will examine this location and display a short English message, explaining why the SSP is halted. SSP programmers should note this point: always update system location $H_CODE (in the SSP code - this is the same as DM_P_H_CODE in the FORTRAN include file, SSPSYM.INC) before a HALT instruction if it is desired to make use of this feature. BSI_STATUS is returned as the FUNCTION value.

21. INTEGER FUNCTION SSPINI(PA)

    Found in file BNL$SSP:SSPBSI.FOR. Routine to "initialize" the SSP's error counters and restore DM 0 to have the initial PM starting address, as if the code were just downloaded. The interface error retry counters are also zeroed. Any DM locations that may have been changed by user-written SSP programs since downloading will not, of course, be affected by SSPINI. BSI_STATUS is returned as the FUNCTION value, unless no SSP resides at PA, in which case SSPINI returns as 0.

22. SUBROUTINE SSPPSW(PSW,MAR,DMMAX,NXTPSW,ERROR,MESSAGE)

    Found in file BNL$SSP:SSPBSI.FOR. This routine, when given an SSP PSW in "PSW" will return the equivalent BNLSYS error code in "ERROR" and a one-line CHARACTER*80 English message in "MESSAGE". The SSP's MAR is required in "MAR" only for correct identification of DM wrap-around errors. DMMAX specifies the maximum DM address, and is used only with MAR. Both MAR and DMMAX

    may be specified as zero, but this will make it impossible to distinguish between data memory wrap around errors and divide by zero errors. If the PSW corresponds to an interrupt, then "ERROR" will be returned as negative one (-1). If the PSW does not correspond to either an interrupt or an error, "ERROR" will be returned as zero. The PSW of the next instruction that would be executed is also calculated, in "NXTPSW". Note that the undisturbed PSW in an SSP containing BNLSYS, following an error or interrupt, is stored by BNLSYS in location DM_P_INTPSW; BNLSYS overwrites DM 0 as part of the error handling procedure, so it is important that the contents of DM location DM_P_INTPSW be given to SSPPSW and not that of DM 0.

23. INTEGER FUNCTION SSPPUT(PA,DM,PM)

Found in file BNL$SSP:SSPDWN.FOR, this routine puts (writes) the arrays DM and PM to the SSP whose primary address is PA. DM and PM are expected to have the sizes DMMAX, accessed with VARCOM, VARGET and VARPUT, and SIZ_P_PM, in BNL$SSP:SSPSYM.INC. BSI_STATUS is returned as the FUNCTION value.

24. INTEGER FUNCTION SSPSIZ(PA,SIZE,NK)

Found in file BNL$SSP:SSPDWN.FOR, this routine returns the size, in 32-bit words, of the DM of the SSP whose primary address is PA. Only two possible sizes exist: 32K and 128K. If an error occurs, SIZE is returned as zero, and the details may be found in BSI_STATUS and STATUS. NK is an integer, and is returned as the "number" of "K" words (ie, 32 or 128). BSI_STATUS is returned as the FUNCTION value.

25. INTEGER FUNCTION SYMFND (PA,FLUN,SYMREQ,VALUE,PID,DMBASE,VMBASE,PMBASE)

Found in file BNL$SSP:DBGSUB.FOR. Find a symbol in a VAX-11 MACRO symbol table. Type out the value of the symbol, or contents of a location, if the symbol is an address. PA is the primary address of the SSP in question. FLUN is the FORTRAN LUN of the openned VAX-11 MACRO listing file. SYMREQ is a 1 to 31 character symbol name, requested for a lookup. VALUE is the value (or address) of the symbol).

Return with:
    SYMFND = 0 for success
    SYMFND = -1 if symbol not found, or external.

26. SUBROUTINE SYSDAT(LODAT,CD,YR,MON,DAY,LOTIM,CT,HR,MIN,SEC,TICK)

Found in file BNL$SSP:DBGSUB.FOR. Get the date and time of the last down-loading, according to the BNLSYS locations $LODAT and $LOTIM, supplied by the calling routine in LODAT and LOTIM. CD is a CHARACTER*11 variable returned with an ASCII representation of the date, YR, MON and DAY are integers returned with the year, month, and day. CT is a CHARACTER*11 variable returned with an ASCII representation of the time, HR, MIN, SEC and TICK are integers returned with the hour, minute, second, and CPU clock-ticks.

27. SUBROUTINE SYSGET(NAME,NAMAX,MARKER,MRKWDS,ERRFLG)

Found in file BNL$SSP:DBGSUB.FOR. Get the name of the system, according to the current set of markers. NAME is CHARACTER type, NAMAX is its size, MARKER must have been filled with SSP markers, for example, by LODMRK.

MRKWDS is the number of elements in MARKER that are used (set by LODMRK), and ERRFLG is an error flag, returned as 0 for success, -1 for failure.

Section 2.3

CAMAC-Via-FBD Routines

These routines are defined in the file SSPFBD.FOR and are contained in the library SSPBSI.OLB. They permit transparent access to CAMAC via an FBD (Struck 320, Fastbus Branch Driver). Note that if any of these routines is used, a call to FBD_I must be issued first.

1. INTEGER FUNCTION FBD_BZ(FBD_PA)

   Issues a CAMAC BZ via a Struck 320 (FBD), at Fastbus primary address FBD_PA.

2. INTEGER FUNCTION FBD_I(FBD_PA)

   Initializes a Struck 320 (FBD), at Fastbus primary address FBD_PA. This call must preceed all other FBD calls.

3. INTEGER FUNCTION FBD_N(FBD_PA,F,C,N,A,Q,X)

   FBD Non-data-transfer
   Performs a CAMAC NDT FCNA via a Struck 320 (FBD), at Fastbus primary address FBD_PA. Q and X are the logical values of CAMAC Q and X following or accompanying the operation, if not being ignored. Returns as the value of BSI_STATUS.

4. INTEGER FUNCTION FBD_R(FBD_PA,F,C,N,A,Q,X,DATA)

   FBD Read
   Performs a CAMAC read FCNA via a Struck 320 (FBD), at Fastbus primary address FBD_PA. Q and X are the logical values of CAMAC Q and X following or accompanying the operation, if not being ignored. Returns as the value of BSI_STATUS.

5. INTEGER FUNCTION FBD_RA(FBD_PA,F,C,N,A,Q,X,B,N)

   FBD Read block, Auto-increment mode
   Performs a CAMAC read FCNA via a Struck 320 (FBD), at Fastbus primary address FBD_PA. INTEGER*4 Buffer B is filled with N 4-byte words; the lower three bytes are the CAMAC 24-bit data, the upper byte is FBD-supplied. Q and X are the logical values of CAMAC Q and X following or accompanying the operation, if not being ignored. Returns as the value of BSI_STATUS.

6. INTEGER FUNCTION FBD_RB(FBD_PA,F,C,N,A,Q,X,B,N)

FBD Read Block
Performs a CAMAC read FCNA via a Struck 320 (FBD), at Fastbus primary address FBD_PA. INTEGER*4 Buffer B is filled with N 4-byte words, the lower three bytes are the CAMAC 24-bit data, the upper byte is FBD-supplied. Q and X are the logical values of CAMAC Q and X following or accompanying the operation, if not being ignored. Returns as the value of BSI_STATUS.

7. INTEGER FUNCTION FBD_W(FBD_PA,F,C,N,A,Q,X,DATA)

FBD Write
Performs a CAMAC write FCNA via a Struck 320 (FBD), at Fastbus primary address FBD_PA. Q and X are the logical values of CAMAC Q and X following or accompanying the operation, if not being ignored. Returns as the value of BSI_STATUS.

8. INTEGER FUNCTION FBD_WA(FBD_PA,F,C,N,A,Q,X,B,N)

FBD Write block, Auto-increment mode.

Performs a CAMAC write FCNA via a Struck 320 (FBD), at Fastbus primary address FBD_PA. N 4-byte words are writen to CAMAC from INTEGER*4 Buffer B, with the lower three bytes being the CAMAC 24-bit data and the upper byte being ignored. Q and X are the logical values of CAMAC Q and X following or accompanying the operation, if not being ignored. Returns as the value of BSI_STATUS.

9. INTEGER FUNCTION FBD_WB(FBD_PA,F,C,N,A,Q,X,B,N)

FBD Write Block.
Performs a CAMAC write FCNA via a Struck 320 (FBD), at Fastbus primary address FBD_PA. N 4-byte words are writen to CAMAC from INTEGER*4 Buffer B, with the lower three bytes being the CAMAC 24-bit data and the upper byte being ignored. Q and X are the logical values of CAMAC Q and X following or accompanying the operation, if not being ignored. Returns as the value of BSI_STATUS.

10. SUBROUTINE FBD_Q

Disables Q-ignore. This is the default, following an FBD_I call.

11. SUBROUTINE FBD_QI

Enables Q-ignore.

12. SUBROUTINE FBD_X

Disables X-ignore. This is the default, following an FBD_I call.

13. SUBROUTINE FBD_XI

Enables X-ignore.

Section 2.4

Internal Data and Constants Access Subroutines

Internal data such as BSI_STATUS, and constants such as INT_TYPE, are accessed by the user with the routines VARGET and VARPUT by specifying the "ID" of the location to which access is desired (the IDs are defined in the INCLUDE file SSPSYM_SV.INC, or SSPSYM.INC). VARCOM permits a similar access to internal variables, but instead of an ID, VARCOM accepts an ASCII string which is the name of the desired variable, and a read flag, which VARCOM uses to decide whether a value is being put or gotten. VARGET and VARPUT automatically take care of the data-type of these data, be it logical or integer; VARCOM returns a data type argument. While VARGET and VARPUT serve mainly as programming tools to access the internal variables, VARCOM can be used by a dialog routine to access the internal variables interactively. Additionally, the routine VARDSP displays all of the accessible internal variables and their names on a specified FORTRAN LUN, so that the exact operating conditions of the BSI software can be monitored and logged, if needed. Using VARCOM, VARGET and VARPUT to read and write these locations should simplify the programmer's job, especially when upgrading software to conform to new releases. While in a new release a variable may be moved from one COMMON block to another, or from one INCLUDE file to another, the ID and symbol names will remain constant. The allowed IDs are defined in the INCLUDE file BNL$SSP:SSPSYM_SV.INC (which is itself included in SSPSYM.INC), described in section 3.1, and are FORTRAN PARAMETERs which have names of the form:

SSPVAC_P_symbol

where "symbol" is one of those listed below. VARCOM uses "symbol" directly as an argument, as an ASCII string. These (symbols and IDs) will not change as new releases are put out, unless there is some fundamental requirement for eliminating the variable altogether. The locations to which the programmer has access via VARCOM, VARGET and VARPUT are listed below. Not all SSPBSI variables can be accessed this way.

NOTE

> Interface-specific locations, such as the CERN branch, crate and slot numbers of the CFI (qv ref. 2), are not accessed with VARCOM, VARGET and VARPUT. These values are set once at the first INTINI(0) call, and remain constant thereafter. To change

this type of internal location, create a new SSPINTDEF subroutine (see the description of SSPINTDEF in section 2.1) with these values set appropriately; SSPINTDEF is a very simple and self-explanatory subroutine. The BNL SSP software has no convenient scheme for modifying such values after initialization.

Some variables, such as the TMODE argument of the CERN FMODE subroutine, are continually being modified by the BSI routines, depending on the mode of access (ie, read or write), unless a corresponding flag is set to prevent this continual modification (for TMODE, the corresponding flag is INT_UNBUFMOD; qv below). Any redefinition of such a variable in an SSPINTDEF would be meaningless without proper care taken of the corresponding flag.

Some inaccessible variables are inaccessible because they are obsolete, and may be erroneously lurking about in existing code. For the purpose of flushing out such calls, VARCOM, VARGET and VARPUT output a message on BSI logical unit OUTCOM (OUTCOM is itself accessible with VARCOM, VARGET and VARPUT!) if an obsolete variable is accessed. Obsolete variables eventually become nonexistant variables.

1. INTEGER FUNCTION VARCOM(C32,READ_FLAG,TYPE,ID,DATUM)

   Found in file BNL$SSP:SSPVAC.FOR. Routine to give access to some of the SSP/BSI variables interactively. The calling routine specifies a variable with a text string in C32, the string being the same as the variable name. The variable name should not include the "SSPVAC_P_" FORTRAN PARAMETER access prefix as would be used with VARGET and VARPUT. If "SSPVAC_P_" is prefixed, which is incorrect, VARCOM strips it off (a copy of) the string. If READ_FLAG is .TRUE., VARCOM gets the value of the named variable, and sets DATUM to be that value. If READ_FLAG is .FALSE., VARCOM sets the named variable to the value contained in DATUM. VARCOM also returns the ID, as would be used by VARGET and VARPUT, of the variable. VARCOM returns the data type in TYPE, with:

   TYPE = 1 internal variable is INTEGER
   TYPE = 2 internal variable is LOGICAL

   VARCOM returns with a FUNCTION value of 0 for successful access. VARCOM returns -1 if it could not give access to the item named in the string C32.

2. SUBROUTINE VARDSP(DL,QL)

   Found in file BNL$SSP:SSPVAC.FOR. Displays the current values of the internal variables on the device or file openned on unit DL. If QL is non-zero, it is assumed to be the terminal, and scrolling is synchronized at every 22 lines with a prompt to DL followed by a read from QL.

3. INTEGER FUNCTION VARGET(ID,DATUM)

Found in file BNL$SSP:SSPVAC.FOR. Gets the 32-bit value of the internal datum whose "id" is ID (the IDs are defined in the INCLUDE file SSPSYM_SV.INC). If ID is not known, VARGET returns as -1; otherwise, VARGET returns as 0, together with the value in DATUM.

4. INTEGER FUNCTION VARPUT(ID,DATUM)

Found in file BNL$SSP:SSPVAC.FOR. Modifies the 32-bit value of the internal datum whose "id" is ID (the IDs are defined in the INCLUDE file SSPSYM_SV.INC). If ID is not known, VARPUT returns as -1; otherwise, VARPUT returns as 0, and the internal datum is modified to be to value of DATUM.

Symbols Accessed by VARCOM, VARGET and VARPUT:

BSI_DMMAX See below - "Latecomers".

1. BSI_ERR_LOG A logical value, which, if set to .TRUE., causes all interface errors which would initiate a software recovery attempt to be logged on FORTRAN LUN "OUTERR".

2. BSI_FAILS Counter of the number of times a BSI was busy.

3. BSI_FAILS_MAX Software time-out "limit", used to determine how many times the BSI software will retry communicating with a busy (SS = 1) SSP before giving up.

4. BSI_SS Segment select. The normal mode is for BSI_SS to be -1, and the Fastbus/host-computer interface is assumed to be in a crate segment. The first SSP in a chain of BSIs will assume that its cable segment is the far-side from the interface. If BSI_SS is set to +1, the BSI software will behave as if the interface is on a cable segment. The first SSP in a chain of BSIs will assume that its crate segment is the far-side from the interface.

5. BSI_STATUS The status of the last Fastbus transaction.

BSI_V501 See below - "Latecomers".

6. BSI_WC The number of words successfully transfered in the last transaction.

7. DEFRAD Obsolete. This variable is no longer in use.

8. DMMAX The size, in words, of the SSP currently selected for downloading, dumping, debugging, etc.

9. INCOM The FORTRAN LUN for command input.

This is set to 5 by default.

10. INDBG — The FORTRAN LUN for debug input. This is not currently in use.

11. INT_ALLOC — Logical value which determines whether the Fastbus interface device is allocated (or "booked", for the CFI; qv ref. 2) for exclusive use.

12. INT_ARB — The interface arbitration level, set after each error recovery.

    INT_AUTOBUFRED — See below - "Latecomers".

    INT_BUFSIZ_DIRECT — See below - "Latecomers".

13. INT_BUFSIZ_IN — Specifies the available input buffer size in the interface, expressed in 32-bit words.

14. INT_BUFSIZ_OUT — Specifies the available output buffer size in the interface, expressed in 32-bit words.

15. INT_FAILS — The number of interface errors since last zeroed by the programmer.

16. INT_FAILS_MAX — Number of interface error retries that will be attempted before giving up.

17. INT_TYPE — Specifies which interface type is in use.

    INT_UNBUFMOD — See below - "Latecomers".

    NTA_CALC — See below - "Latecomers".

18. NTA_FAILS — The total number of times since last zeroed by the programmer (or since start-up) that an attempt to recover from an NTA error has been made.

19. NTA_FAILS_MAX — The maximum number of times that a bad NTA read will initiate a recovery attempt.

    NTA_FOURS — See below - "Latecomers".

20. OUTCOM — The FORTRAN LUN for command or information output. This is set to 6 by default.

21. OUTDBG — The FORTRAN LUN for debug output. This is set to 6 by default, and should not be found to be in use except in commented-out WRITE statements.

22. OUTERR — The FORTRAN LUN on which interface errors are logged if the logical value BSI_ERR_LOG is set to .TRUE.

23. SSPPA — The physical address of the SSP currently selected for downloading, dumping, debugging, etc. SSPPA is not, strictly speaking, an internal constant, since it is not used by any of the access routines. SSPPA does, however, permit user-defined routines to communicate a primary address to one another

without setting up additional COMMON blocks.

Latecomers: (user-accessable variables added after the initial 23)

24. BSI_DMMAX The size, in words, of the SSP currently in use as a BSI by the BSI routines.

25. INT_UNBUFMOD Logical value which, if .TRUE., causes the interface routines to block-transfer data to and from Fastbus without buffering them in the interface.

26. INT_AUTOBUFRED Logical value which, if .TRUE., causes the buffer size, used in block transfers in unbuffered mode only, to be automatically reduced on certain types of errors, thereby permitting recovery in cases when the buffer size is too large. This logical has no effect if INT_UNBUFMOD is .FALSE. .

27. INT_BUFSIZ_DIRECT Buffer size, in words, of unbuffered (direct) block transfers through the Fastbus interface.

28. NTA_CALC Logical value which, if .TRUE., causes NTAs to be calculated, not read, from slaves during piecemeal transfers. See section 2.1, item 4.

29. NTA_FOURS Logical value which, if .TRUE., causes NTAs to be calculated assuming the slave is byte-wise addressable (each word read from the slave would correspond to an NTA increment of four instead of one), NTA_FOURS has no effect if NTA_CALC is not .TRUE. . See section 2.1, item 4.

30. BSI_V501 Logical value which, if .TRUE., informs the BSI software that the SSPs are loaded with software linked with a BNLSYS of version later than (and not including) 5.00. There has been a slight improvement in the BSI method, which reduces overhead and increases transfer speeds by about 5% to 10% or more, depending on the size of the transfer. Back-compatability has been retained through the introduction of this flag. Currently, BSI_V501 initializes to .FALSE., but in later releases it will initialize to .TRUE. Under version 5.10, to make full use of the improvements, the programmer must include a VARPUT call to set BSI_V501 to .TRUE. . SSP programs linked with BNLSYS version 5.1 may be run with BSI_V501 set either to .TRUE. or .FALSE. , or used with version 5.0 VAX-resident SSP software. SSP programs linked with BNLSYS version 5.0 (ie, SSP resident software) may not be run with BSI_V501 set to .TRUE. ; these SSP programs may be upgraded simply by re-linking with the SLINK (or similar) command. qv ref. 1, section 3.

String Manipulation Routines

Several special-purpose character-string manipulation routines are used by SSPAR, as well as by some of the other routines. These string manipulation routines are defined in the file BNL$SSP:STRPKG.FOR, and are described below. The arguments "S" and "COM" are FORTRAN CHARACTER constants or arrays.

1. INTEGER FUNCTION ACODE(S,BEG,END,AINT,AREAL,REQRAD)

   Decode the character string in S, from S(BEG) to S(END), as a number. Put the result in AREAL and AINT, if possible. BEG and END are untouched.

   Return ACODE = :

0	a real number was decoded in AREAL, and exists in integer form, too, in AINT.
-1	a real number was decoded in AREAL, but the result was too big to be converted into integer form.
-2	error - nothing was decoded.

   The default conversion radix is determined by setting REQRAD to 2, 8, 10, or 16. Numbers of radices other than the REQRAD-determined default are allowed, with the radix being indicated by prefixing the ASCII representation of the number with a '%' for radix 2, '"' or 'O' for radix 8, '$' for radix 10 and 'H' or 'X' for radix 16. The prefix and the number may not be separated by any character, such as a space.

2. SUBROUTINE ASCDAT(ASCDATE,ASCWEEKDAY,ASCTIME)

   Return 9 character ASCII date, 11 character ASCII time, and 9 character ASCII weekday.

3. SUBROUTINE BLNKLR(S,EPT,SMAX)

   Routine to annihilate trailing blanks (blank killer) in S and to append a null to S after the last nonblank character, if before SMAX. EPT is set to point at the last non-blank character or else EPT=SMAX. Also remove trailing TABs.

4. SUBROUTINE CNVCAP(C1,BPT,EPT)

   Convert all lower case characters to capitals, in string C1, from C1(BPT) to C1(EPT).

5. INTEGER FUNCTION COMSI(COM,CPT,CLST,MODE,OLST)

   Search the list of defined commands in CLST for the command in COM, starting at COM(CPT). Commands defined in CLST are separated from one another by single nulls, with the last command defined in CLST being followed by a double null. COMSI is equal to the ordinal number of the found command in CLST, if found. If an option (command modifier) search is requested (by setting MODE=1; MODE=0 otherwise), the list of defined options in OLST is searched for the option in COM (if any) and the bit in OID corresponding to that ordinal number is set. OLST is defined like CLST. OID is a bit-mask INTEGER*4 array in COMMON/OID/, with the first word in COMMON/OID/ being OMAX, the size of array OID. eg: COMMON/OID/OMAX,OID(3). For 32 or fewer options in OLST, OID may be a single INTEGER*4 word, otherwise, OID

must be an array with one element for every 32 options. OID is intended for use with TSTBIT, SETBIT, CLRBIT and TOGBIT, the extended bit-mask manipulation routines (qv below). OID is set to 0 if no options are present in COM. COMSI is set to less than or equal to zero to indicate an error:

```
 0 - unknown command
-1 - ambiguous command
-2 - unknown option
-3 - ambiguous option
-4 - insufficient space in OID; determined by OMAX.
```

CPT is set to point after the separator or at the null at the end of the command, if found, in COM.

6. SUBROUTINE CPU_T0

   Initialize CPU elapsed time counter.

7. SUBROUTINE CPUCNV(CPU_TIME,CPU_ASC)

   Convert CPU_TIME (INTEGER*4) to 17 character ASCII days, hours, minutes, seconds and ticks.

8. SUBROUTINE CPUTIM(TCPU_TIME,DCPU_TIME,TCPU_ASC,DCPU_ASC)

   Return absolute CPU time (since last CPU_T0 call) and delta CPU time, both as INTEGER*4 values (TCPU_TIME and DCPU_TIME) and as ASCII CHARACTER*17 strings (TPCU_ASC and DCPU_ASC).

9. SUBROUTINE CPUTXT(TEXT,LUN)

   TEXT is a short (CHARACTER*40) line of text, supplied by the calling routine. LUN is the FORTRAN LUN on which the output appears. LUN, if not the terminal, must be openned with CARRIAGECONTROL='LIST' Same as DATEID, except CPU time is displayed instead of weekday.

10. SUBROUTINE DATEID(TEXT,LUN)

    TEXT is a short (CHARACTER*40) line of text, supplied by the calling routine. LUN is the FORTRAN LUN on which the output appears. LUN, if not the terminal, must be openned with CARRIAGECONTROL='LIST' The date, time and day of the week are output to LUN, preceeded by TEXT.

11. SUBROUTINE DECA(S,SPT,SMAX,VALUE)

Convert VALUE (INTEGER*4) to a character string in S, starting at SPT. Append a null & set SPT to point at

it. VALUE is non-negative.

12. SUBROUTINE ELA_T0(TELA_TIME)

    Initialize elapsed time counter. The time uses 64 bit system time format.

13. SUBROUTINE ELATIM(TELA_TIME,DELA_TIME)

    Return absolute (since last ELA_T0 call) and delta elapsed times. The time uses 64 bit system time format.

14. SUBROUTINE FPARSE(CIN,COUT,DEFTYP)

    Parse out a file name from CIN and put it in COUT. If no file type is specified, append the default file type, DEFTYP. CIN and DEFTYP should be CHARACTER constants or arrays and terminated with nulls. CIN should be no more that 80 (less the length of DEFTYP) characters long, determined by the position of the null. COUT should be a CHARACTER constant or array at least 80 characters long. COUT will be terminated with a null, ready for use in an OPEN statement.

15. SUBROUTINE GETCHR(CHR,LONG,SEL)

    Get an 8-bit character out of LONG, a 32 bit word. The character is the SELth byte (1, 2, 3 or 4) of LONG, and goes into CHR.

16. SUBROUTINE INSERT(SOURCE,START,END,DEST,DPT,DMAX)

    Inserts characters from SOURCE (from START to END or null) to DEST (from DPT to .LE. DMAX). Sets DPT to point after the last character inserted. No null is appended; START is not modified.

17. SUBROUTINE LKUP(STRING,START,END,LIST,ID)

    LIST=multi-character array of knowns, separated by nulls and terminated with two nulls. STRING=multi-character array of unknowns; must not contain nulls (except the first char.). ID=identification of STRING(from START to END) as found in LIST. Return ID=0 if not found, ID=-1 if ambiguous.

18. SUBROUTINE MOVE(SOURCE,START,SMAX,DEST,DPT,DMAX)

    Moves characters from SOURCE (from START to SMAX or null) to DEST (from DPT to .LE. DMAX). Appends DEST with a null. Leaves DPT pointing at the appended null, or equal to DMAX.

19. SUBROUTINE MOVE0(SOURCE,START,SMAX,DEST,DPT,DMAX)

    Set DPT=1, then: Moves characters from SOURCE (from

START to SMAX or null) to DEST (from DPT to .LE. DMAX). Appends DEST with a null. Leaves DPT pointing at the appended null, or equal to DMAX.

20. INTEGER FUNCTION MSRCH(CHAR,LIST)

Search LIST to see if CHAR is in it; LIST should end with a null. If not found, MSRCH=0. If found, MSRCH points to the matched character in LIST.

21. LOGICAL FUNCTION NPARSE(S,SPT,SMAX,D,DMAX)

Parse the next argument out of S, starting at SPT, and into D. Return .TRUE. if a next argument exists, .FALSE. otherwise. Leave SPT pointing at the space after the argument parsed, or at SMAX.

22. SUBROUTINE NULKLR(C,L)

"Kill" a trailing null and all that follows, by blanking out all characters in C starting at the null and proceeding to the Lth character.

23. LOGICAL FUNCTION QUERYL(Q80,D70,A70,INLUN,OUTLUN)

Long query/response. Ask the operator, found on INLUN and OUTLUN, for the answer to the 80 character question Q80. The default 70 character answer is specified by the calling program in D70. The answer is returned in A70. If OUTLUN is zero, input is taken from INLUN without a prompt, and if OUTLUN is negative, input is taken from INLUN without a prompt, and then displayed, in the form of a prompt, on -OUTLUN.

24. LOGICAL FUNCTION QUERYM(Q80,D32,A32,INLUN,OUTLUN)

Medium query/response. Ask the operator, found on INLUN and OUTLUN, for the answer to the 80 character question Q80. The default 32 character answer is specified by the calling program in D32. The answer is returned in A32. If OUTLUN is zero, input is taken from INLUN without a prompt, and if OUTLUN is negative, input is taken from INLUN without a prompt, and then displayed, in the form of a prompt, on -OUTLUN.

25. LOGICAL FUNCTION QUERYS(Q40,D16,A16,INLUN,OUTLUN)

Short query/response. Ask the operator, found on INLUN and OUTLUN, for the answer to the 40 character question

Q40. The default 16 character answer is specified by the calling program in D16. The answer is returned in A16. If OUTLUN is zero, input is taken from INLUN without a prompt, and if OUTLUN is negative, input is taken from INLUN without a prompt, and then displayed, in the form of a prompt, on -OUTLUN.

26. SUBROUTINE SEARCH(STRING,START,END,CHAR)

Search the character string STRING from START to END or null for the character CHAR and set START to point at it. Return START=0 if the search is unsuccessful.

27. SUBROUTINE STRIP(S,SPT,SMAX)

Routine to strip off comments (and trailing blanks) at the end of "command" lines, where comments are denoted by the usual exclaimation mark (!).

28. SUBROUTINE TIMASC(TIME,ASC)

Convert TIME to 17 character ASCII days, hours, minutes, seconds and ticks. TIME is a 2-word INTEGER*4 array, and uses 64-bit system time format. ASC is CHARACTER*17.

29. SUBROUTINE TIMCNV(TIME,DAYS,HOURS,MINS,SECS,TICKS)

Convert TIME to days, hours, minutes, seconds and ticks. Ticks are milliseconds. TIME is a 2-word INTEGER*4 array, and uses 64-bit system time format. DAYS, HOURS, MINS, SECS and TICKS are one-word INTEGER*4.

30. LOGICAL FUNCTION TSTBIT(ARRAY,BIT)

Tests the BITth bit in ARRAY and return its value (.TRUE. or .FALSE.) ARRAY is an INTEGER*4 array and is assmumed to have BIT/32 elements; for BIT less than or equal to 32, ARRAY may be a single word.

31. SUBROUTINE TOGBIT(ARRAY,BIT,AMAX)

Toggles the BITth bit in ARRAY. ARRAY is an INTEGER*4 array and has AMAX elements; for AMAX equal to 1, ARRAY may be a single word.

32. SUBROUTINE CLRBIT(ARRAY,BIT,AMAX)

Clears the BITth bit in ARRAY. ARRAY is an INTEGER*4 array and has AMAX elements; for AMAX equal to 1, ARRAY may be a single word.

33. SUBROUTINE SETBIT(ARRAY,BIT,AMAX)

Sets the BITth bit in ARRAY. ARRAY is an INTEGER*4 array and has AMAX elements; for AMAX equal to 1, ARRAY may be a single word.

34. INTEGER FUNCTION UNCODE(COM,CPT,UNINT,UNREAL,REQRAD)

==> New version, which uses subroutine arguments instead of COMMON blocks.
Decodes the character string in COM, starting at COM(CPT), as a number. Put the result in UNREAL and UNINT, if possible. Return UNCODE = :

-1	error - nothing was decoded.
0	a real number was decoded in UNREAL, but the result was too big to be converted into integer form.
1	a real number was decoded in UNREAL, and exists in integer form, too, in UNINT.

Leaves CPT pointing at the first character after the separator following this parameter or at the null, if this is the last parameter on the line.

The default conversion radix is determined by setting REQRAD to 2, 8, 10, or 16. Numbers of radices other than the REQRAD-determined default are allowed, with the radix being indicated by prefixing the ASCII representation of the number with a '%' for radix 2, '"' or 'O' for radix 8, '$' for radix 10 and 'H' or 'X' for radix 16. The prefix and the number may not be separated by any character, such as a space.

35. SUBROUTINE WIDEN(S,BEG,WIDTH,SMAX)

    Move everything in S, from and including BEG, up by WIDTH. Effectively widen or create a gap in S. WIDTH is not negative. BEG is left to point at the first position of the new gap (ie, BEG is unchanged).

36. INTEGER FUNCTION ZCODE(S,SPT,SMAX,LENGTH,INT)

    Encode an integer, INT, of LENGTH decimal digits, into S, starting at S(SPT) and not going past S(SMAX), right justified and left-padded with zeroes. Append a null on S. Return ZCODE = -1 for failure, ZCODE = 0 for success.

Section 3

FORTRAN INCLUDE Files.

Section 3.1

SSP Symbols - the INCLUDE File SSPSYM.INC

The INCLUDE file SSPSYM.INC contains all the various parameters and COMMON blocks used by the SSP access routines, and has not changed in content very much from version 4.00. SSPSYM.INC consists of several, smaller INCLUDE files. By judiciously selecting which of these smaller files to include, a programmer can greatly reduce the size of his listings; however, if that is not a concern, including SSPSYM.INC itself simplifies the programmer's task somewhat. The smaller files contained in SSPSYM.INC are:

SSPSYM_MS.INC !Miscellenaous definitions.

SSPSYM_DM.INC !BNLSYS DM locations

SSPSYM_PM.INC !BNLSYS PM locations.

SSPSYM_HC.INC !HALT code definitions.

SSPSYM_EC.INC !BNLSYS and BSI Error codes.

```
SSPSYM_SV.INC  !SSP-variable access IDs (VARGET and VARPUT)

SSPSYM_FD.INC  !FUNCTION-type declarations.
```

All INCLUDE files except SSPSYM_SV.INC are subject to change without notice. Internal locations should therefore be accessed with the routines VARCOM, VARGET and VARPUT; code written that accesses internal constants with these routines will be as near as possible to release/version-independent code as possible.

An INCLUDE file which contains everything in SSPSYM.INC but the FUNCTION type declarations (ie, SSPSYM_FD.INC) is available as SSPSYM_NF.INC.

SSPSYM_MS.INC

```
*
*      Miscelleneous definitions.
*
*      Interface types:
*
INTEGER INT_P_TYPE_SIM,INT_P_TYPE_CFI,INT_P_TYPE_UFI
PARAMETER (INT_P_TYPE_SIM=-1)
PARAMETER (INT_P_TYPE_CFI=0)
PARAMETER (INT_P_TYPE_UFI=1)
*
*      DM and PM sizes and CSR offsets to DM and PM:
*
INTEGER SIZ_P_DMS,SIZ_P_DMB,SIZ_P_PM,CSR_P_DM,CSR_P_PM
PARAMETER (SIZ_P_DMS=32768)  !Small (32K) SSP DM size, in words.
PARAMETER (SIZ_P_DMB=4*32768) !Big (128K) SSP DM size, in words.
PARAMETER (SIZ_P_PM=4096) !SSP PM size, in words.
PARAMETER (CSR_P_DM='80000000'X) !CSR space access to DM.
PARAMETER (CSR_P_PM='40000000'X) !CSR space access to PM.
*
*      PSECT ID values  "PID":
INTEGER PID_P_DM,PID_P_PM,PID_P_VM,PID_P_ABS,PID_P_MM,PID_P_CM
PARAMETER (PID_P_ABS=0) !.ABS.
PARAMETER (PID_P_DM=1) !$DATA
PARAMETER (PID_P_VM=2) !$VIRTUAL
PARAMETER (PID_P_PM=3) !$CODE
PARAMETER (PID_P_MM=4) !$MARKER
PARAMETER (PID_P_CM=5) !COMMONs. (PID .GE. PID_P_CM)
*
*      I/O request codes:
INTEGER IO_P_GET,IO_P_PUT
PARAMETER (IO_P_GET='01000000'X) !Get request.
PARAMETER (IO_P_PUT='02000000'X) !Put request.
*
*      BSI commands:
INTEGER BSI_P_WD,BSI_P_RD,BSI_P_WC,BSI_P_RC,BSI_P_WRB
INTEGER BSI_P_WDM,BSI_P_RDM,BSI_P_WCM,BSI_P_RCM
INTEGER BSI_P_WDP,BSI_P_RDP
PARAMETER (BSI_P_WD=1) !Command to write to data space.
PARAMETER (BSI_P_RD=2) !Command to read from data space.
PARAMETER (BSI_P_WC=3) !Write CSR.
PARAMETER (BSI_P_RC=4) !Read CSR.
PARAMETER (BSI_P_WRB=5) !Write RB.
PARAMETER (BSI_P_WDM=6) !Write to data space, Multi-listener.
PARAMETER (BSI_P_RDM=7) !Read from data space, Multi-listener.
```

```
      PARAMETER (BSI_P_WCM=8) !Write CSR, Multi-listener.
      PARAMETER (BSI_P_RCM=9) !Read CSR, Multi-listener.
      PARAMETER (BSI_P_WDP=10) !Write to data space, pipeline (not used).
      PARAMETER (BSI_P_RDP=11) !Read from data space, pipeline (not used).
*
*       The parameter block size and offsets:
                                                                Page 54


      INTEGER PB_P_SIZE,PB_P_CM,PB_P_SS,PB_P_FF,PB_P_HP
      INTEGER PB_P_HS,PB_P_RC,PB_P_PA,PB_P_SA,PB_P_WC
      INTEGER PB_P_BA,PB_P_BF
      INTEGER PB_P_RA,PB_P_RH
      PARAMETER (PB_P_SIZE=12) !Parameter block size (words).
      PARAMETER (PB_P_CM=0) !Command word.
      PARAMETER (PB_P_SS=4) !Segment select;  1 for crate, -1 for cable.
      PARAMETER (PB_P_FF=8) !Final-transfer flag. -1 if final, 0 if more.
      PARAMETER (PB_P_HP=12) !PA of initiator - none if it's the host.
      PARAMETER (PB_P_HS=16) !SA of initiator - none if it's the host.
      PARAMETER (PB_P_RC=20) !Return code from BSI. (Like BSI_STATUS).
      PARAMETER (PB_P_PA=24) !Primary address.
      PARAMETER (PB_P_SA=28) !Secondary address.
      PARAMETER (PB_P_WC=32) !Word count.
      PARAMETER (PB_P_BA=36) !Buffer address.
*old:  PARAMETER (PB_P_BF=40) !Buffer start.
      PARAMETER (PB_P_RA=40) !Restart address (new - Version 5.1).
      PARAMETER (PB_P_RH=44) !Restart halt code (new - Version 5.1).
      PARAMETER (PB_P_BF=48) !Buffer start.
*
*       Some CFI STATUS codes:
      INTEGER CFI_P_NOROOM,CFI_P_TIMOUT,CFI_P_SBUSY
      INTEGER CFI_P_SEMTY,CFI_P_SERR
      INTEGER CFI2_P_ERROR,CFI2_P_EMASK,CFI2_P_DK
      PARAMETER (CFI_P_NOROOM='C0000000'X) !Buffer full.
      PARAMETER (CFI_P_TIMOUT='A0000000'X) !Time out.
      PARAMETER (CFI_P_SBUSY='8A'X) !Slave busy.
      PARAMETER (CFI_P_SEMTY='90'X) !Slave empty.
      PARAMETER (CFI_P_SERR='92'X) !Slave data error.
      PARAMETER (CFI2_P_ERROR='FFFFFFFF'X) !General-purpose "error" flag.
      PARAMETER (CFI2_P_EMASK='F8008000'X) !Mask this onto STATUS(2).
      PARAMETER (CFI2_P_DK='100'X) !The "bad SS response with DK" bit.
                                                                Page 55


                       SSPSYM_DM.INC

!
!       The BNLSYS DM locations:
!
! Define all the fixed memory locations for the SSP DM, as per
! the BNL resident system for the SSP.
!
! Addresses in the SSP DM space are prefixed with:
!                              DM_P_
!
! A more complete description of these locations can be found
! in the document "The BNL Cross Assembler for the SSP", in
! section 4, "The BNL System for the SSP - BNLSYS".
!
!DM 0
!
      INTEGER DM_P_PSW0,DM_P_MAR,DM_P_INTPSW,DM_P_INT_RET
      PARAMETER (DM_P_PSW0='000'X) !Contains SSP PM starting address.
```

```
      PARAMETER (DM_P_MAR='004'X) !SSP Fastbus memory address register.
      PARAMETER (DM_P_INTPSW='008'X) !Contains interrupt or error PSW.
      PARAMETER (DM_P_INT_RET='00C'X)
      !
      !DM 10:
      !
      INTEGER DM_P_ERR_RET,DM_P_DM_MAIN,DM_P_ERROR,DM_P_H_CODE
      PARAMETER (DM_P_ERR_RET='010'X) !User-spec'd error-return addr.
      PARAMETER (DM_P_DM_MAIN='014'X) !PM start addr. of MAIN routine.
      PARAMETER (DM_P_ERROR='018'X) !The error code.
      PARAMETER (DM_P_H_CODE='01C'X) !The "Halt code" word.
      !
      !DM 20:
      !
      INTEGER DM_P_HALT,DM_P_ERR_CTL,DM_P_PAUSE,DM_P_TRACE
      INTEGER DM_P_SYSPM,DM_P_SMASK,DM_P_SSPSIZ
      PARAMETER (DM_P_HALT='020'X) !Contains data for the HALT macro.
      PARAMETER (DM_P_ERR_CTL='024'X)
      PARAMETER (DM_P_PAUSE='028'X) !Contains data for the PAUSE macro.
      PARAMETER (DM_P_TRACE='030'X) !Contents of current R13.
      PARAMETER (DM_P_SYSPM='034'X) !Maximum system PM address.
      PARAMETER (DM_P_SMASK='038'X) !Contains data for INT_ON macro.
      PARAMETER (DM_P_SSPSIZ='03C'X) !The DM size in bytes
      !
      !DM 40:
      !
      INTEGER DM_P_RDUMP
      PARAMETER (DM_P_RDUMP='040'X) !Register dump area.
      !
      !DM 80:                Error counters:
      !
      !
      INTEGER DM_P_ECT_TOT,DM_P_ECT_DZ,DM_P_ECT_OF,DM_P_ECT_UO
```


```
      INTEGER DM_P_ECT_BCE,DM_P_ECT_ARB,DM_P_ECT_ADD,DM_P_ECT_IOD
      INTEGER DM_P_ECT_WRP,DM_P_ECT_UTO,DM_P_ECT_BSY,DM_P_ECT_EOB
      INTEGER DM_P_ECT_SSX,DM_P_ECT_SSE,DM_P_ECT_UFB,DM_P_INT_TOT
      !
      PARAMETER (DM_P_ECT_TOT='080'X) !Total errors.
      PARAMETER (DM_P_ECT_DZ='084'X) !Divide-by-zero errors.
      PARAMETER (DM_P_ECT_OF='088'X) !Total overflows.
      PARAMETER (DM_P_ECT_UO='08C'X) !Total unsup. opcodes.
      PARAMETER (DM_P_ECT_BCE='090'X) !Block-count-exhausteds.
      PARAMETER (DM_P_ECT_ARB='094'X) !Arbitration errors
      PARAMETER (DM_P_ECT_ADD='098'X) !I/O Address errors
      PARAMETER (DM_P_ECT_IOD='09C'X) !I/O data errors
      PARAMETER (DM_P_ECT_WRP='0A0'X) !SSP DM wrap-arounds.
      PARAMETER (DM_P_ECT_UTO='0A4'X) !Unknown Timeouts.
      PARAMETER (DM_P_ECT_BSY='0A8'X) !SS=1 errors.
      PARAMETER (DM_P_ECT_EOB='0AC'X) !SS=2 errors.
      PARAMETER (DM_P_ECT_SSX='0B0'X) !SS=3,4,5 errors.
      PARAMETER (DM_P_ECT_SSE='0B4'X) !SS=6,7 errors.
      PARAMETER (DM_P_ECT_UFB='0B8'X) !Unknow FB errors.
      PARAMETER (DM_P_INT_TOT='0BC'X) !Total interrupts.
      !
      !DM C0:
      !
      INTEGER DM_P_IPL_TOT,DM_P_LOC_LH,DM_P_PF0_INI,DM_P_PFREE0
      PARAMETER (DM_P_IPL_TOT='0C0'X) !Total starts at PM 80.
      PARAMETER (DM_P_LOC_LH='0C4'X)
      PARAMETER (DM_P_PF0_INI='0C8'X) !Initial 1st free PM loc. (at load)
```

```
      PARAMETER (DM_P_PFREE0='0CC'X) !Current 1st free PM loc.
      !
      !DM D0:
      !
      INTEGER DM_P_F0_INI,DM_P_FREE0,DM_P_FREEL,DM_P_BRK_LOC
      PARAMETER (DM_P_F0_INI='0D0'X) !Initial 1st free DM loc. (at load)
      PARAMETER (DM_P_FREE0='0D4'X) !Current 1st free DM loc.
      PARAMETER (DM_P_FREEL='0D8'X) !Last free DM location.
      PARAMETER (DM_P_BRK_LOC='0DC'X)
      !
      !DM E0:
      INTEGER DM_P_BRKID,DM_P_FORLIN,DM_P_LODAT,DM_P_LOTIM
      PARAMETER (DM_P_BRKID='E0'X) !Breakpoint ID number.
      PARAMETER (DM_P_FORLIN='E4'X) !"FORTRAN" line number.
      PARAMETER (DM_P_LODAT='E8'X) !Load-date.
      PARAMETER (DM_P_LOTIM='EC'X) !Load-time.
      !
      !DM F0:
      INTEGER DM_P_PM_BSI
      PARAMETER (DM_P_PM_BSI='0F4'X) !System startup location for BSI.
      !
      !DM 10C:
      INTEGER DM_P_ECT_CLR,DM_P_ECT_IOF,DM_P_ECT_FOF
      INTEGER DM_P_ECT_FUF,DM_P_ECT_FRO,DM_P_ECT_FDZ
      INTEGER DM_P_ECT_BRK
```

```
      INTEGER DM_P_BRK_1,DM_P_BRK_2,DM_P_BRK_3,DM_P_BRK_4
      INTEGER DM_P_BRK_S,DM_P_BRK_B
      PARAMETER (DM_P_ECT_CLR='10C'X) !PSW-clear-on-entry-to-handler errors.
      PARAMETER (DM_P_ECT_IOF='110'X) !Fixed overflow errors.
      PARAMETER (DM_P_ECT_FOF='114'X) !Floating overflow errors.
      PARAMETER (DM_P_ECT_FUF='118'X) !Floating underflow errors.
      PARAMETER (DM_P_ECT_FRO='11C'X) !Floating reserved-operand errors.
      PARAMETER (DM_P_ECT_FDZ='120'X) !Floating divide-by-zero errors.
      PARAMETER (DM_P_ECT_BRK='124'X) !Breakpoint handler error.
      PARAMETER (DM_P_BRK_1='128'X) !Breakpoint 1 address is stored here.
      PARAMETER (DM_P_BRK_2='12C'X) !Breakpoint 2 address is stored here.
      PARAMETER (DM_P_BRK_3='130'X) !Breakpoint 3 address is stored here.
      PARAMETER (DM_P_BRK_4='134'X) !Breakpoint 4 address is stored here.
      PARAMETER (DM_P_BRK_S='138'X) !Breakpoint S address is stored here.
      PARAMETER (DM_P_BRK_B='13C'X) !Breakpoint B address is stored here.
      !
      !DM 140:
      INTEGER DM_P_INT_CB,DM_P_INT_MB,DM_P_CRATE_PA,DM_P_CABLE_PA
      INTEGER DM_P_IOREQ,DM_P_IOLU,DM_P_IOBA,DM_P_IOWC,DM_P_IOWT
      INTEGER DM_P_IOFM,DM_P_IOFS,DM_P_BSI_SS,DM_P_BSI_WC
      INTEGER DM_P_MARK0,DM_P_PCMSK,DM_P_CCMSK
      PARAMETER (DM_P_INT_CB='140'X)
      PARAMETER (DM_P_INT_MB='180'X)
      PARAMETER (DM_P_CRATE_PA='1C0'X)
      PARAMETER (DM_P_CABLE_PA='1C4'X)
      PARAMETER (DM_P_IOREQ='1C8'X)
      PARAMETER (DM_P_IOLU='1CC'X)
      PARAMETER (DM_P_IOBA='1D0'X)
      PARAMETER (DM_P_IOWC='1D4'X)
      PARAMETER (DM_P_IOWT='1D8'X)
      PARAMETER (DM_P_IOFM='1DC'X)
      PARAMETER (DM_P_IOFS='1E0'X)
      PARAMETER (DM_P_BSI_SS='1E4'X) !BSI segment select (1=crate, -1=cable)
.
      PARAMETER (DM_P_BSI_WC='1E8'X) !BSI count of successfully xferred word
```

```
s.
          PARAMETER (DM_P_MARK0='1F4'X) !Marker-words pointer-location.
          PARAMETER (DM_P_PCMSK='1F8'X) !PC-mask, for PSW.
          PARAMETER (DM_P_CCMSK='1FC'X) !CC-bits-mask, for PSW.
          !
          !DM 200:               User defined DM locations start here.
```

Page 58

SSPSYM_PM.INC

```
          !
          !       The BNLSYS PM locations:
          !
          ! Define all the fixed memory locations for the SSP PM, as per
          ! the BNL resident system for the SSP.
          !
          ! Addresses in the SSP PM space are prefixed with:
          !                               PM_P_
          !
          ! A more complete description of these locations can be found
          ! in the document "The BNL Cross Assembler for the SSP", in
          ! section 4, "The BNL System for the SSP - BNLSYS".
          !
          !       PM locations - breakpoint handler:
          INTEGER PM_P_BRKADD_1,PM_P_BRKADD_2,PM_P_BRKADD_3,PM_P_BRKADD_4
          INTEGER PM_P_BRKADD_S,PM_P_BRKADD_B
          PARAMETER (PM_P_BRKADD_1='A4'X)
          PARAMETER (PM_P_BRKADD_2='A6'X)
          PARAMETER (PM_P_BRKADD_3='A8'X)
          PARAMETER (PM_P_BRKADD_4='AA'X)
          PARAMETER (PM_P_BRKADD_S='AC'X)
          PARAMETER (PM_P_BRKADD_B='AE'X)
          !
          !       Location of "Resume after HALT" for all breakpoints:
          INTEGER PM_P_BRKHLT
          PARAMETER (PM_P_BRKHLT='96'X)
          !
          !       Location of "$BRKERR" (Breakpoint error) for all breakpoints:
          INTEGER PM_P_BRKERR
          PARAMETER (PM_P_BRKERR='B0'X)
```

Page 59

SSPSYM_HC.INC

```
          !
          !       HALT codes definitions:
          !
          INTEGER HC_P_INT,HC_P_UOP,HC_P_BRK,HC_P_BDN,HC_P_BRP,HC_P_NRM
          INTEGER HC_P_DED,HC_P_IOR,HC_P_BER,HC_P_CLR,HC_P_NYS,HC_P_BWT
          PARAMETER (HC_P_INT='FF000000'X) !Interrupt.
          PARAMETER (HC_P_UOP='FE000000'X) !Unsupported opcode (fatal) error.
          PARAMETER (HC_P_BRK='FD000000'X) !Breakpoint (system subroutine BREAK)
.
          PARAMETER (HC_P_BDN='FC000000'X) !BSI done.
          PARAMETER (HC_P_BRP='FB000000'X) !Reserved.
          PARAMETER (HC_P_NRM='FA000000'X) !SSP halted after normal completion.
          PARAMETER (HC_P_DED='F9000000'X) !SSP halted by "CALL DEAD"
          PARAMETER (HC_P_IOR='F8000000'X) !SSP halted by internal I/O request.
          PARAMETER (HC_P_BER='F7000000'X) !Breakpoint (fatal) error.
          PARAMETER (HC_P_CLR='F6000000'X) !PSW "clear" on error handler entry.
          PARAMETER (HC_P_NYS='F5000000'X) !SSP not yet started.
```

```
      PARAMETER (HC_P_BWT='F4000000'X) !BSI waiting for HBSI.
```

SSPSYM_EC.INC

```
!
!       BNLSYS and BSI Error codes:
!
!       Error codes numbered 1 through 7 are the SS values themselves.
INTEGER ECD_P_DZ,ECD_P_OF,ECD_P_UO,ECD_P_BCE,ECD_P_ARB
INTEGER ECD_P_ADD,ECD_P_IOD,ECD_P_WRP,ECD_P_UTO,ECD_P_UFB
INTEGER ECD_P_CLR,ECD_P_IOF,ECD_P_FOF
INTEGER ECD_P_FUF,ECD_P_FRO,ECD_P_FDZ,ECD_P_BRK
PARAMETER (ECD_P_DZ=8) !Divide-by-zero.
PARAMETER (ECD_P_OF=9) !Arithmetic overflow.
PARAMETER (ECD_P_UO=10) !Unsupported opcode.
PARAMETER (ECD_P_BCE=11) !Block-count exhausted.
PARAMETER (ECD_P_ARB=12) !Arbitration cycle time-outs.
PARAMETER (ECD_P_ADD=13) !Address cycle (AS/AK) time-outs.
PARAMETER (ECD_P_IOD=14) !Data cycle (DS/DK) time-outs.
PARAMETER (ECD_P_WRP=15) !DM wrap-arounds.
PARAMETER (ECD_P_UTO=16) !Unknown time-outs.
PARAMETER (ECD_P_UFB=17) !Unknown Fastbus errors.
PARAMETER (ECD_P_CLR=18) !PSW-clear-on-entry-to-error-handler errors.
PARAMETER (ECD_P_IOF=19) !Fixed overflow errs. (software-implemented).
PARAMETER (ECD_P_FOF=20) !Flt. overflow errs. (software-implemented).
PARAMETER (ECD_P_FUF=21) !Flt. underflow errs. (software-implemented).
PARAMETER (ECD_P_FRO=22) !Flt. reserved-operand errs. (soft-impl).
PARAMETER (ECD_P_FDZ=23) !Flt. divide-by-zero errs. (software-impl).
PARAMETER (ECD_P_BRK=24) !Break-point handler error.
```



```
*       BSI_STATUS values:
*
* 0     Normal completion.
* 0xx (hex) xx is BSI's $ERROR.
* 020 (hex) BSI data buffer is too small.
* 021 (hex) Software timeout.
* 022 (hex) Invalid primary address specified.
* 023 (hex) BSI in path is not an SSP.
* 024 (hex) Invalid BSI command.
* 025 (hex) BSI in path is busy.
* 026 (hex) BSI in path does not contain BSI software.
* 027 (hex) NTA read error during piecemeal transfer.
* 028 (hex) Invalid interface operation.
* 1xx (hex) Interface error.
* 101 (hex) Failure during normal non-BSI operation (Interface trouble
).
* 102 (hex) Power might be off.
* 2xx (hex) Halt code is not HC_P_BDN.  xx is halt code.
* -x    (Sign bit set) The negative of words-actually-transfered.
*
INTEGER BS_P_MSK,BS_P_ECM,BS_P_ECB
PARAMETER (BS_P_MSK='F00'X) !The code-mask.
PARAMETER (BS_P_ECM='0FF'X) !Error code mask.
PARAMETER (BS_P_ECB='000'X) !Error code bit.
* Errors from 1 to 7 are SS codes.
* Errors from 8 to 20 (hex) are SSP hard-errors.
```

```
INTEGER BS_P_BUF,BS_P_STO,BS_P_IPA,BS_P_NSP,BS_P_IVC
INTEGER BS_P_BSY,BS_P_BNR,BS_P_NTA,BS_P_IIO,BS_P_ICM
INTEGER BS_P_ICB,BS_P_INT
INTEGER BS_P_OFF,BS_P_UIT,BS_P_CBK
INTEGER BS_P_NDA,BS_P_AAU
INTEGER BS_P_HCM,BS_P_HCB
PARAMETER (BS_P_BUF='020'X)
PARAMETER (BS_P_STO='021'X)
PARAMETER (BS_P_IPA='022'X)
PARAMETER (BS_P_NSP='023'X)
PARAMETER (BS_P_IVC='024'X)
PARAMETER (BS_P_BSY='025'X)
PARAMETER (BS_P_BNR='026'X)
PARAMETER (BS_P_NTA='027'X)
PARAMETER (BS_P_IIO='028'X) !Invalid interface operation.
PARAMETER (BS_P_ICM='1FF'X) !Interface code mask.
PARAMETER (BS_P_ICB='100'X) !Interface code bit.
PARAMETER (BS_P_INT='101'X)
PARAMETER (BS_P_OFF='102'X) !Power might be off.
PARAMETER (BS_P_UIT='103'X) !Unknown interface type.
PARAMETER (BS_P_CBK='104'X) !CERN crate booked by another user.
PARAMETER (BS_P_NDA='105'X) !Not successfully deallocated.
PARAMETER (BS_P_AAU='106'X) !Allocated by another user.
PARAMETER (BS_P_HCM='2FF'X) !Halt code mask.
PARAMETER (BS_P_HCB='200'X) !Halt code bit.
```


SSPSYM_SV.INC

```
*
*       SSP-variable access IDs (VARGET and VARPUT) and
*       CHARACTER names of those symbols:
*
*       The variables for which VARGET and VARPUT have access.
*       The parameter is the variable name, with the prefix:
*       SSPVAC_P_
*
INTEGER SSPVAC_P_BSI_ERR_LOG
INTEGER SSPVAC_P_BSI_FAILS,SSPVAC_P_BSI_FAILS_MAX
INTEGER SSPVAC_P_BSI_SS,SSPVAC_P_BSI_STATUS
INTEGER SSPVAC_P_BSI_WC
INTEGER SSPVAC_P_DEFRAD
INTEGER SSPVAC_P_DMMAX
INTEGER SSPVAC_P_INCOM,SSPVAC_P_INDBG
INTEGER SSPVAC_P_INT_ALLOC,SSPVAC_P_INT_ARB
INTEGER SSPVAC_P_INT_BUFSIZ_IN,SSPVAC_P_INT_BUFSIZ_OUT
INTEGER SSPVAC_P_INT_FAILS,SSPVAC_P_INT_FAILS_MAX
INTEGER SSPVAC_P_INT_TYPE
INTEGER SSPVAC_P_NTA_FAILS,SSPVAC_P_NTA_FAILS_MAX
INTEGER SSPVAC_P_OUTCOM,SSPVAC_P_OUTDBG,SSPVAC_P_OUTERR
INTEGER SSPVAC_P_SSPPA,SSPVAC_P_BSI_DMMAX
INTEGER SSPVAC_P_INT_UNBUFMOD
INTEGER SSPVAC_P_AUTOBUFRED
INTEGER SSPVAC_P_INT_BUFSIZ_DIRECT
INTEGER SSPVAC_P_NTA_CALC,SSPVAC_P_NTA_FOURS
INTEGER SSPVAC_P_BSI_V501
PARAMETER (SSPVAC_P_BSI_ERR_LOG=1)
PARAMETER (SSPVAC_P_BSI_FAILS=2)
PARAMETER (SSPVAC_P_BSI_FAILS_MAX=3)
PARAMETER (SSPVAC_P_BSI_SS=4)
PARAMETER (SSPVAC_P_BSI_STATUS=5)
PARAMETER (SSPVAC_P_BSI_WC=6)
```

```
      PARAMETER (SSPVAC_P_DEFRAD=7) !Obsolete.
      PARAMETER (SSPVAC_P_DMMAX=8)
      PARAMETER (SSPVAC_P_INCOM=9)
      PARAMETER (SSPVAC_P_INDBG=10)
      PARAMETER (SSPVAC_P_INT_ALLOC=11)
      PARAMETER (SSPVAC_P_INT_ARB=12)
      PARAMETER (SSPVAC_P_INT_BUFSIZ_IN=13)
      PARAMETER (SSPVAC_P_INT_BUFSIZ_OUT=14)
      PARAMETER (SSPVAC_P_INT_FAILS=15)
      PARAMETER (SSPVAC_P_INT_FAILS_MAX=16)
      PARAMETER (SSPVAC_P_INT_TYPE=17)
      PARAMETER (SSPVAC_P_NTA_FAILS=18)
      PARAMETER (SSPVAC_P_NTA_FAILS_MAX=19)
      PARAMETER (SSPVAC_P_OUTCOM=20)
      PARAMETER (SSPVAC_P_OUTDBG=21)
      PARAMETER (SSPVAC_P_OUTERR=22)
      PARAMETER (SSPVAC_P_SSPPA=23)
```



```
      PARAMETER (SSPVAC_P_BSI_DMMAX=24)
      PARAMETER (SSPVAC_P_INT_UNBUFMOD=25)
      PARAMETER (SSPVAC_P_AUTOBUFRED=26)
      PARAMETER (SSPVAC_P_INT_BUFSIZ_DIRECT=27)
      PARAMETER (SSPVAC_P_NTA_CALC=28)
      PARAMETER (SSPVAC_P_NTA_FOURS=29)
      PARAMETER (SSPVAC_P_BSI_V501=30)

      INTEGER SSPVAC_NMAX !This is equal to the highest valued SSPVAC_P_x.
      PARAMETER (SSPVAC_NMAX=30)


*      The variable types (used by VARCOM):

      INTEGER SSPVAC_TYPE(SSPVAC_NMAX) !TYPE=1 for INTEGER, 2 for LOGICAL.
      DATA SSPVAC_TYPE/
      1 2, !BSI_ERR_LOG
      2 1, !BSI_FAILS
      3 1, !BSI_FAILS_MAX
      4 1, !BSI_SS
      5 1, !BSI_STATUS
      6 1, !BSI_WC
      7 0, !DEFRAD   0 ==> Obsolete.
      8 1, !DMMAX
      9 1, !INCOM
      1 1, !INDBG
      1 2, !INT_ALLOC
      2 1, !INT_ARB
      3 1, !INT_BUFSIZ_IN
      4 1, !INT_BUFSIZ_OUT
      5 1, !INT_FAILS
      6 1, !INT_FAILS_MAX
      7 1, !INT_TYPE
      8 1, !NTA_FAILS
      9 1, !NTA_FAILS_MAX
      2 1, !OUTCOM
      1 1, !OUTDBG
      2 1, !OUTERR
      3 1, !SSPPA
      4 1, !BSI_DMMAX
      5 2, !INT_UNBUFMOD
      6 2, !AUTOBUFRED
      7 1, !INT_BUFSIZ_DIRECT
```

```
     8 2, !NTA_CALC
     9 2, !NTA_FOURS
     3 2/ !BSI_V501


     *      The variable names (used by VARCOM):

     CHARACTER*32 SSPVAC_NAME(SSPVAC_NMAX)
     DATA SSPVAC_NAME/
```

Page 63

```
     1 'BSI_ERR_LOG',
     2 'BSI_FAILS',
     3 'BSI_FAILS_MAX',
     4 'BSI_SS',
     5 'BSI_STATUS',
     6 'BSI_WC',
     7 'DEFRAD',
     8 'DMMAX',
     9 'INCOM',
     1 'INDBG',
     1 'INT_ALLOC',
     2 'INT_ARB',
     3 'INT_BUFSIZ_IN',
     4 'INT_BUFSIZ_OUT',
     5 'INT_FAILS',
     6 'INT_FAILS_MAX',
     7 'INT_TYPE',
     8 'NTA_FAILS',
     9 'NTA_FAILS_MAX',
     2 'OUTCOM',
     1 'OUTDBG',
     2 'OUTERR',
     3 'SSPPA',
     4 'BSI_DMMAX',
     5 'INT_UNBUFMOD',
     6 'AUTOBUFRED',
     7 'INT_BUFSIZ_DIRECT',
     8 'NTA_CALC',
     9 'NTA_FOURS',
     3 'BSI_V501'/
```

SSPSYM_FD.INC

```
     !
     !      FUNCTION-type declarations.
     !
     INTEGER SSPRCM,SSPRDM,SSPWCM,SSPWDM,SSPGO,SSPINI
     INTEGER SSPRC,SSPRCB,SSPRD,SSPRDB,SSPRP,SSPRPB
     INTEGER SSPWC,SSPWCB,SSPWD,SSPWDB,SSPWP,SSPWPB
     INTEGER SSPWRB,SSPWRBG
     INTEGER INTERR,INTINI,INTRTY
     INTEGER OUT_INTERR
     INTEGER DOWN,DOWNV0,DOWNV1,DUMP
     INTEGER SSPCMP,SSPFMT,SSPGET,SSPPUT,SSPSIZ
     INTEGER VARGET,VARPUT,VARCOM
     INTEGER BREAK,IOHAND,SHOW,SINGLE,SYMFND
     INTEGER ACODE,COMSI,MSRCH,TSTBIT,UNCODE,ZCODE
     LOGICAL QUERYS,QUERYL
     INTEGER FBD_BZ,FBD_I,FBD_N,FBD_R,FBD_RA,FBD_RB
```

```
      INTEGER FBD_W,FBD_WA,FBD_WB
```


Section 3.2

FASTBUS CSR0 Module ID - The INCLUDE File MPSIDS.INC

To facilitate FASTBUS module identification, especially in interactive environments, an INCLUDE file has been written which consists of an INTEGER*4 array, MPSID_N, which contains the FASTBUS CSR0 module IDs of several (NUM_IDS_P) FASTBUS modules, and a CHARACTER*48 array, MPSID_C, which contains the names and/or model numbers of these modules.

MPSIDS.INC

```
      INTEGER NUM_IDS_P
      PARAMETER(NUM_IDS_P=52)
      INTEGER MPSID_N(NUM_IDS_P)
      CHARACTER*48 MPSID_C(NUM_IDS_P)
*
      DATA MPSID_N(1)/'0011'X/
      DATA MPSID_C(1)
     1 /'SLAC Primo memory...............................'/
      DATA MPSID_N(2)/'0012'X/
      DATA MPSID_C(2)
     1 /'SLAC sequencer BT...............................'/
      DATA MPSID_N(3)/'0014'X/
      DATA MPSID_C(3)
     1 /'SLAC crate GAC (*)..............................'/
      DATA MPSID_N(4)/'0016'X/
      DATA MPSID_C(4)
     1 /'SSP  (SLAC Scanner Processor, new ID)...........'/
      DATA MPSID_N(5)/'0022'X/
      DATA MPSID_C(5)
     1 /'TRANSIAC DVM....................................'/
      DATA MPSID_N(6)/'0033'X/
      DATA MPSID_C(6)
     1 /'de/dx control...................................'/
      DATA MPSID_N(7)/'0040'X/
      DATA MPSID_C(7)
     1 /'TRIUMF IORFI (faked)............................'/
      DATA MPSID_N(8)/'0041'X/
      DATA MPSID_C(8)
     1 /'TRIUMF 32 channel discriminator.................'/
      DATA MPSID_N(9)/'0042'X/
      DATA MPSID_C(9)
     1 /'TRIUMF ADI test jig.............................'/
      DATA MPSID_N(10)/'0043'X/
      DATA MPSID_C(10)
     1 /'TRIUMF memory...................................'/
```



```
      DATA MPSID_N(11)/'0044'X/
      DATA MPSID_C(11)
     1 /'TRIUMF ADC......................................'/
      DATA MPSID_N(12)/'0045'X/
```

```
      DATA MPSID_C(12)
      1 /'TRIUMF PCL demo.  .............................'/
      DATA MPSID_N(13)/'00A2'X/
      DATA MPSID_C(13)
      1 /'BiRa/FNAL memory................................'/
      DATA MPSID_N(14)/'00A6'X/
      DATA MPSID_C(14)
      1 /'BiRa/FNAL display module........................'/
      DATA MPSID_N(15)/'00AA'X/
      DATA MPSID_C(15)
      1 /'BiRa/FNAL dual display..........................'/
      DATA MPSID_N(16)/'0104'X/
      DATA MPSID_C(16)
      1 /'SLAC crate GAC .................................'/
      DATA MPSID_N(17)/'0106'X/
      DATA MPSID_C(17)
      1 /'SSP  (SLAC Scanner Processor)...................'/
      DATA MPSID_N(18)/'01C0'X/
      DATA MPSID_C(18)
      1 /'Ames 32 channel FADC............................'/
      DATA MPSID_N(19)/'0400'X/
      DATA MPSID_C(19)
      1 /'Old TRIUMF IORFI (faked)........................'/
      DATA MPSID_N(20)/'0401'X/
      DATA MPSID_C(20)
      1 /'Old TRIUMF 32 channel discriminator.............'/
      DATA MPSID_N(21)/'0402'X/
      DATA MPSID_C(21)
      1 /'Old TRIUMF ADI test jig.........................'/
      DATA MPSID_N(22)/'0403'X/
      DATA MPSID_C(22)
      1 /'Old TRIUMF memory...............................'/
      DATA MPSID_N(23)/'0404'X/
      DATA MPSID_C(23)
      1 /'Old TRIUMF ADC..................................'/
      DATA MPSID_N(24)/'0405'X/
      DATA MPSID_C(24)
      1 /'Old TRIUMF PCL demo.............................'/
      DATA MPSID_N(25)/'1001'X/
      DATA MPSID_C(25)
      1 /'GAC card, crate seg.............................'/
      DATA MPSID_N(26)/'1002'X/
      DATA MPSID_C(26)
      1 /'GAC card, cable seg.............................'/
      DATA MPSID_N(27)/'1007'X/
      DATA MPSID_C(27)
      1 /'Kinetics F290 display...........................'/
```



```
      DATA MPSID_N(28)/'1015'X/
      DATA MPSID_C(28)
      1 /'Kinetics F820 Scanner Processor.................'/
      DATA MPSID_N(29)/'1020'X/
      DATA MPSID_C(29)
      1 /'GAC card, crate seg.............................'/
      DATA MPSID_N(30)/'1032'X/
      DATA MPSID_C(30)
      1 /'LeCroy 1878 TDC.................................'/
      DATA MPSID_N(31)/'1033'X/
      DATA MPSID_C(31)
      1 /'LeCroy 1879 TDC.................................'/
```

```
      DATA MPSID_N(32)/'1035'X/
      DATA MPSID_C(32)
     1 /'LeCroy 1879 TDC.................................'/
      DATA MPSID_N(33)/'1038'X/
      DATA MPSID_C(33)
     1 /'LeCroy 1810 CAT.................................'/
      DATA MPSID_N(34)/'1039'X/
      DATA MPSID_C(34)
     1 /'LeCroy 1810 CAT.................................'/
      DATA MPSID_N(35)/'1040'X/
      DATA MPSID_C(35)
     1 /'LeCroy 1882N ADC................................'/
      DATA MPSID_N(36)/'1041'X/
      DATA MPSID_C(36)
     1 /'LeCroy 1885N ADC................................'/
      DATA MPSID_N(37)/'1042'X/
      DATA MPSID_C(37)
     1 /'LeCroy 1882P ADC................................'/
      DATA MPSID_N(38)/'1043'X/
      DATA MPSID_C(38)
     1 /'LeCroy 1885P ADC................................'/
      DATA MPSID_N(39)/'1044'X/
      DATA MPSID_C(39)
     1 /'LeCroy 1885(unk. versn.) ADC....................'/
      DATA MPSID_N(40)/'1046'X/
      DATA MPSID_C(40)
     1 /'LeCroy 1891 memory..............................'/
      DATA MPSID_N(41)/'1063'X/
      DATA MPSID_C(41)
     1 /'LeCroy 1892 memory (old)........................'/
      DATA MPSID_N(42)/'1065'X/
      DATA MPSID_C(42)
     1 /'LeCroy 1892 memory..............................'/
      DATA MPSID_N(43)/'6103'X/
      DATA MPSID_C(43)
     1 /'STR136 ECL I/O latch............................'/
      DATA MPSID_N(44)/'6105'X/
      DATA MPSID_C(44)
     1 /'STR146 ECL input register.......................'/
```



```
      DATA MPSID_N(45)/'6806'X/
      DATA MPSID_C(45)
     1 /'Struck 161? (FDM)...............................'/
      DATA MPSID_N(46)/'6817'X/
      DATA MPSID_C(46)
     1 /'Struck xxx GAC, crate segment...................'/
      DATA MPSID_N(47)/'681D'X/
      DATA MPSID_C(47)
     1 /'Struck xxx GAC, crate segment...................'/
      DATA MPSID_N(48)/'6823'X/
      DATA MPSID_C(48)
     1 /'STR200 scaler...................................'/
      DATA MPSID_N(49)/'6841'X/
      DATA MPSID_C(49)
     1 /'Struck 320 CAMAC Branch Driver (FBD)............'/
      DATA MPSID_N(50)/'6842'X/
      DATA MPSID_C(50)
     1 /'Struck 402 GAC, cable segment...................'/
      DATA MPSID_N(51)/'C000'X/
      DATA MPSID_C(51)
```

```
1 /'de/dx FADC.....................................'/
DATA MPSID_N(52)/'F0F0'X/
DATA MPSID_C(52)
1 /'BiRa IORFI II..................................'/
```


Section 3.3

Internal Usage INCLUDE Files

For reference purposes, the contents of the several internal usage INCLUDE files is listed here. It is cautioned that VARCOM, VARGET and VARPUT be used to access the variables defined herein, rather than the more direct approach of directly including these files in sources and modifying these variables with FORTRAN assignment statements. In particular, direct-modification code will not be version-independent, and may also conflict with internal useage by modifying variables that are not really available for modification, producing unpredictable results.

CFIVAR.INC
(qv ref. 2.)

```
*      Fastbus interface variables and Fastbus status and control
*      arrays:
PARAMETER P_STALEN=32
PARAMETER P_CNTRLEN=1000
*
INTEGER BR,CR,SN,INTF,TMODE,STATUS,CNTRL
COMMON/CFIVAR/BR,CR,SN,INTF,TMODE
COMMON/CFIVAR/STATUS(P_STALEN),CNTRL(P_CNTRLEN)
```

INTCTL.INC

```
*
*      Interface control COMMON.
*
INTEGER INT_TYPE,INT_ARB
INTEGER INT_BUFSIZ_IN,INT_BUFSIZ_OUT,INT_BUFSIZ_DIRECT
INTEGER INTINI_CALLED_CHECK
LOGICAL INTINI_SUCCESS,INT_ALLOC,AUTOBUFRED,INT_UNBUFMOD
LOGICAL INTRTY_FLAG,NTA_CALC,NTA_FOURS
COMMON/INTCTL/INT_TYPE,INT_ARB
COMMON/INTCTL/INT_BUFSIZ_IN,INT_BUFSIZ_OUT,INT_BUFSIZ_DIRECT
COMMON/INTCTL/INTINI_CALLED_CHECK
COMMON/INTCTL/INTINI_SUCCESS,INT_ALLOC
COMMON/INTCTL/AUTOBUFRED,INT_UNBUFMOD,INTRTY_FLAG
COMMON/INTCTL/NTA_CALC,NTA_FOURS
*
INTEGER INTINI_CALLED_P
PARAMETER (INTINI_CALLED_P='AFAFAFAF'X)
```


SSPBSI.INC

```
INTEGER BSI_STATUS,BSI_PA,BSI_BUFAD,BSI_RESTART,BSI_HCODE
```

```
      INTEGER BSI_FAILS,BSI_FAILS_MAX,BSI_START,BSI_FREE,BSI_SS
      INTEGER BSI_FREE0,BSI_FREEL,BSI_WC,INT_FAILS,INT_FAILS_MAX
      INTEGER NTA_FAILS,NTA_FAILS_MAX,BSI_INT_WC,BSI_DMMAX
      INTEGER BSI_OLDPA
      LOGICAL BSI_PIPE,BSI_ERR_LOG,BSI_V501
      COMMON/SSPBSI/BSI_STATUS,BSI_PA,BSI_BUFAD,BSI_WC,BSI_RESTART
      COMMON/SSPBSI/BSI_HCODE,BSI_START
      COMMON/SSPBSI/BSI_FREE(2),BSI_SS,BSI_PIPE
      COMMON/SSPBSI/BSI_FAILS,BSI_FAILS_MAX
      COMMON/SSPBSI/INT_FAILS,INT_FAILS_MAX,NTA_FAILS,NTA_FAILS_MAX
      COMMON/SSPBSI/BSI_INT_WC,BSI_ERR_LOG,BSI_DMMAX,BSI_OLDPA
      COMMON/SSPBSI/BSI_V501
      *
      EQUIVALENCE (BSI_FREE(1),BSI_FREE0),(BSI_FREE(2),BSI_FREEL)
      *
      *      BSI_SS is the segment-select word.  If it's 1, then the
      *      BSI will talk to its slave via the crate segment.  If it's -1,
      *      then the BSI will use the cable segment.  Each lower level
      *      BSI gets a segment select word that is the complement of what
      *      it passes along.
      *
      INTEGER PBLOCK(PB_P_SIZE+1)
```

SSPCTL.INC

```
      *
      *      SSP control COMMON:
      *      (The DMMAX values are in words, not bytes.)
      INTEGER SSPPA,PMMAX,DMMAX
      COMMON/SSPCTL/SSPPA,PMMAX,DMMAX
```

SSPFBD.INC

```
      *The FBD routines need to communicate some information with one anothe
r:
      LOGICAL FBD_IGNORE_Q,FBD_IGNORE_X
      COMMON/SSPFBD/FBD_IGNORE_Q,FBD_IGNORE_X
```


SSPLUN.INC

```
      C
      C      Logical Units for the SSP Access Routines:
      INTEGER INCOM,OUTCOM,INDBG,OUTDBG,OUTERR
      COMMON/SSPLUN/INCOM,OUTCOM,INDBG,OUTDBG,OUTERR
```

SSPMEM.INC

```
      *      Contains a duplicate SSP memory in a COMMON block.
```

```
*       This is needed by the downloading routines.
INTEGER PM,DM,CSR0,CSR8
COMMON/SSPMEM/PM(SIZ_P_PM),DM(SIZ_P_DMB),CSR0,CSR8
*
INTEGER PAR_MRKMAX,MRKWDS,MARKER
PARAMETER (PAR_MRKMAX=4000)
COMMON/SSPMRK/MRKWDS,MARKER(PAR_MRKMAX)
```

SSPSIM.INC

```
C       Contains a duplicate SSP memory in a COMMON block.
C       This is needed by the simulation routines.
INTEGER SIM_PM,SIM_DM,SIM_CSR0,SIM_CSR8,SIM_CNTA,SIM_DNTA
COMMON/SSPSIM/SIM_PM(SIZ_P_PM),SIM_DM(SIZ_P_DMB)
COMMON/SSPSIM/SIM_CSR0,SIM_CSR8,SIM_CNTA,SIM_DNTA
```


Section 4

Linking User Written Routines with the SSP Routines

DCL option files are provided to simplify the linking process. It is at link-time that the interface-type with which a given program will work is selected, by linking in the appropriate modules; no values or constants should be set by the user at run time for the purpose of selecting the interface type. Three option files exist, one for each interface type:

BNL$SSP:SSPBSIUFI.OPT (UNIBUS/QBUS CFI)

BNL$SSP:SSPBSISIM.OPT (simulator)

BNL$SSP:SSPBSICFI.OPT (CAMAC CFI - no longer tested)

In addition, the equivalent VAX-DEBUG libraries (compilied with the VAX FORTRAN /DEBUG option) can be linked by using these option files instead:

BNL$SSP:SSPBSIUFI_DBG.OPT (UNIBUS/QBUS CFI)

BNL$SSP:SSPBSISIM_DBG.OPT (simulator)

BNL$SSP:SSPBSICFI_DBG.OPT (CAMAC CFI - no longer tested)

SSPBSIUFI.OPT consists of:

```
BNL$SSP:SSPBSI/LIB,-
BNL$SSPINT:FBERRS,FBLHAND,FCFIPROC,-
BNL$SSPUFISUB:,- !Select the UFI version of CERN routines.
BNL$SSP:SSPUFIDEF,- !Select the UFI version.
BNL$SSPINT:FLIBVAX/LIB,VAXLIB/LIB,FDUMLIB/LIB
```

SSPBSISIM.OPT consists of:

```
BNL$SSPSIMSUB:,- !The simulator, of the CERN CFI routines.
BNL$SSP:SSPBSI/LIB,-
BNL$SSP:SSPSIMDEF !Select the simulator.
```


SSPBSICFI.OPT consists of:

```
BNL$SSP:SSPBSI/LIB,-
BNL$SSPINT:FBERRS,FBLHAND,FCFIPROC,-
BNL$SSPCFISUB:,- !Select the CFI version of CERN routines.
BNL$SSP:SSPCFIDEF,- !Select the CFI version.
BNL$SSPINT:FLIBVAX/LIB,VAXLIB/LIB,FDUMLIB/LIB
```

The file SSPBSI.OLB contains the SSP software. The general-purpose CERN software (qv ref. 2) is contained in the files, in BNL$SSPINT: FBERRS.OBJ, FBLHAND.OBJ, FCFIPROC.OBJ, FLIBVAX.OLB, VAXLIB.OLB and FDUMLIB.OLB, and the interface-dependent CERN software is contained in two files, assigned to two logicals:

BNL$SSPUFISUB: "UFI" - UNIBUS/QBUS version.

BNL$SSPCFISUB: "CFI" - CAMAC (CERN system crate) version.

The simulator is contained in a file assigned to a logical:

BNL$SSPSIMSUB: assigned to BNL$SSP:SIMINT.OBJ (simulator).

The files SSPUFIDEF.OBJ, SSPSIMDEF.OBJ and SSPCFIDEF.OBJ are described in section 2.1, and define the interface-selection module SSPINTDEF.

Section 5

The SSP Diagnostic Routine - CHECK

CHECK is a program which performs a variety of SSP diagnostics in sequence, checking that each diagnostic performs correctly. Some diagnostics are executed by the SSP; these diagnostics must be written using the BNL Cross Assembler DIAGNOSTIC control statement (see "The BNL Cross Assembler for the SSP User's Guide"). CHECK is run with the command (defined by executing the command file BNL$SSP:BNLSSPINT.COM):

```
$ CHECK
```

This will run one of the files BNL$SSP:CHECK_U.EXE, BNL$SSP:CHECK_S.EXE or BNL$SSP:CHECK_C.EXE, depending on the interface type currently selected (see section 1). The simulator version of CHECK is useful mainly to list all the possible error messages which the SSP-executed diagnostics produce; a properly written SSP-executed diagnostic will always fail with the simulator version of CHECK. A memory diagnostic will always pass on the simulator.

The operator is prompted for several items, including the primary address of the SSP to be checked, the number of times diagnostics should be repeated and which diagnostics should be performed. CHECK may be run on an SSP not directly accessible to the host/FASTBUS interface by specifying a BSI route for the SSP's primary address or slot (qv ref. 1, sect. 5). The results are output both to the terminal, and to the file CHECK.ELG.

CHECK performs the following diagnostics on the SSP:

1. CSR0 module ID test - The ID should be 0106h.

2. Memory tests

    1. DM size determination - SSPs can have 128 Kb or 512 Kb DM.

    2. Memory address uniqueness - Each word in DM and PM has its own address written and read back, to determine that each location is addressed once and only once.

    3. All bits on, all bits off - Zeros are written to each DM and PM location and read back, to determine that no memory bits are stuck on. Ones are then written to each DM and PM location and read back, to determine that all memory bits may be set on.

    4. Memory alternate bit patterns - A pattern of all even bits set and all odd bits cleared is written to both DM and PM to test for unusual memory problems. This test is repeated with the complementary pattern.

3. CPU instruction diagnostics - written with the DIAGNOSTIC statement, and from the file BNL$SSP:DIAG.SSP.

    1. Load and store (L and ST) registers over zeroes test

    2. Load and store (L and ST) registers over ones test

    3. Store-multiple (STM) test

    4. Load-multiple (LM) test

    5. Subtract-register (SR) test

    6. Load address (LA) test

    7. Branch and link register (BALR) test

    8. Load positive register (LPR) test

    9. Load negative register (LNR) test

10. Load and test register (LTR) test

11. Load complement register (LCR) test

12. Branch and link (BAL) test

13. Load PSW and branch on condition (LPSW and BC) test

14. Subtract register and set CC bits test

15. And register (NR) test

16. Compare logical register (CLR) test

17. Or register (OR) test

18. Load register (LR) test

19. Compare register (CR) test

20. Add register (AR) test

21. Multiply register (MR) test

22. Divide register (DR) test

23. Add logical register (ALR) test

24. Subtract logical register (SLR) test

25. Compare Logical Immediate (CLI) test

26. Pipeline Register Contention (PRC) test

27. SSP Condition Code Bits check

4. Single-step/Breakpoint Condition Code Bits test.

5. Cable segment arbitration test.

6. Crate segment arbitration test.

Section 6

The SSP Simulator

The simulator consists of a set of fake CERN CFI routines (qv ref. 2), most of which do nothing, and some of which read and write into large data arrays set up to look like SSP DM and PM, as well as CSR0 and CSR8. These routines will actually return the usual SS=2 or SS=6 errors if areas outside these arrays are accessed. No attempt is made to use the primary address, so that an SSPAR "SCAN" command, for example, shows a Fastbus crate full of SSPs (all the same "SSP"). The simulator routines are defined in the FORTRAN file BNL$SSP:SIMINT.FOR. See "The BNL Cross

Assembler for the SSP User's Guide", section 8 "Debugging Techniques" for an example of one use of the simulator.

Brief List of the Simulator Routines

These routines do absolutely nothing, except to zero STATUS(2) if the array STATUS is one of the arguments, and return the value zero for an error code. They are called in exactly the same way as the real CERN routines:

CAMQIO	CTSTAT
CCBOO	DLOUT
CCCD	DLOUTX
CCCI	DLYENO
CCICL	FARB
CCRLS	FBOOK
CDREG	FCAM
CDSET	FMINI
CFSA	FMODE
CGINF	FREQ
CSSA	MESLOG
CSUBC	

These routines "simulate" an SSP which responds to all primary addresses (8-bit range of addressing - the BSI default routing table won't work, since a real, working SSP isn't simulated) and has 4K words of PM and 128K words of DM. Illegal addresses produce SS=6 responses, and block read/writes past valid address space produce SS=2 responses.

FERDEC	FRDSA
FRC	FWC
FRCB	FWCB
FRCM	FWCM
FRCSA	FWD
FRD	FWDB
FRDB	FWDM
FRDM	FWRB

References

1. The BNL Cross Assembler for the SSP User's Guide, Version 5.10
R. Hackenburg, BNL MPS group
Unpublished disk-file manual, (system) logical device and filename:
BNL$SSP:SSP_PROGRAMMER.MEM.
Extant on PHYSNET VAX nodes: BNLMPS, BNLA00, BNL14A (and others)

2. CERN FASTBUS Software, Version: August 1, 1984.
E. M. Rimmer, CERN D.D. division

3. FASTBUS - Modular High Speed Data Acquisition and Control System for High Energy Physics and Other Applications
U.S. NIM Committee, December 1983, DOE/ER-0189