SLAC-PUB-8936
July, 2001

LIBXSIF, A Standalone Library for Parsing the Standard Input Format [1]

P. TENENBAUM
Stanford Linear Accelerator Center, Stanford University, Stanford, CA

Abstract

The Standard Input Format for the description of accelerator beamlines has achieved limited acceptance due to the complexity of the parser required. We describe a standalone library of Fortran-90 routines which can be used to parse a superset of the Standard Input format in use at SLAC, named Extended Standard Input Format (XSIF). This library provides authors of new simultion codes with a simple means of adding XSIF compatibility to their programs, and also permits users to add their own features to the parser with relative ease. As examples we describe the manner in which the linear accelerator code LIAR was modified to use LIBXSIF, and changes made to DIMAD to switch from its internal Standard Input Format parser to use of the external XSIF parser. URL's for the source code, documentation, and ready-to-use libraries are provided.

Contributed to *2001 Particle Accelerator Conference,* Chicago, Illinois, June 18 – June 22, 2001

[1]Work supported by the Department of Energy, Contract DE-AC03-76SF00515.

LIBXSIF, A STANDALONE LIBRARY FOR PARSING THE STANDARD INPUT FORMAT

P. Tenenbaum, SLAC[†]

Abstract

The Standard Input Format for the description of accelerator beamlines has achieved limited acceptance due to the complexity of the parser required. We describe a standalone library of Fortran-90 routines which can be used to parse a superset of the Standard Input format in use at SLAC, named Extended Standard Input Format (XSIF). This library provides authors of new simultion codes with a simple means of adding XSIF compatibility to their programs, and also permits users to add their own features to the parser with relative ease. As examples we describe the manner in which the linear accelerator code LIAR was modified to use LIBXSIF, and changes made to DIMAD to switch from its internal Standard Input Format parser to use of the external XSIF parser. URL's for the source code, documentation, and ready-to-use libraries are provided.

1 INTRODUCTION

The Standard Input Format (SIF) for accelerator beamline description was created in 1984 to respond to the need for a more user-friendly *lingua franca* for the accelerator world [1]. SIF had several notable improvements over previous languages (such as TRANSPORT format [2]):

- SIF permitted highly repetitive elements, from magnets to beamline symmetry units, to be defined once and reused repeatedly
- SIF permitted the beam optics to be defined independently of the beam energy, by declaring magnet strengths in quantities normalized to the energy (for example, defining the strength of a bend magnet by its total bend angle rather than its integrated magnetic field)
- SIF permitted many beamlines to be defined in a single file, and allowed the user to select which one was to be simulated
- SIF permitted the user to define named parameters, and allowed beamline element parameters to be defined in terms of arithmetic relationships between constants, named parameters, and parameters of other elements
- SIF permitted decks to be spread over multiple files, so that the actual beamline description and commands for the simulation program could be kept separate; this in turn permitted a single deck to be used without modification by many users, each of whom performed different simulations with different command files
- SIF permitted a more relaxed and intuitive syntax than the existing beamline description languages.

The standard input format was immediately adopted by the CERN simulation program MAD (Methodical Accelerator Design) [3], and was later added to the programs DIMAD [4], TRANSPORT [5], TURTLE [6], and COMFORT [7]. Nonetheless, the widespread adoption of SIF has been impeded by the absence of a single, standalone version of the deck parser that can easily be added to any existing simulation program. In addition, the absence of linear accelerator elements in SIF has delayed its usage in programs in which such elements are critical, such as LIAR [8].

2 LIBXSIF

LIBXSIF is a standalone library for the parsing of Extended Standard Input Format (XSIF) decks. Software engineers need add only a few calls to routines in this library to allow their programs to read and manage decks in the XSIF language. The standard is "extended" from SIF in the following ways:

- Linear accelerator elements, using the DIMAD keyword LCAVITY, are permitted
- Elements have an APERTURE attribute
- A handful of non-SIF elements, such as GKICKs, are included.

LIBXSIF is written in Fortran-90, and the latest version (v1.2) is available for Solaris, NT, and VMS-Alpha platforms; a Linux-i86 version will be released later this year. The present version of LIAR (v2.3) uses LIBXSIF, as does the present version of NLC-DIMAD (v2.8). The use of a common library to perform all XSIF parsing will greatly ease maintenance and expansion/extension of the code.

2.1 Calling LIBXSIF Routines from Existing Programs

LIBXSIF contains 76 object files, but only a few of these need to be called by the main program. In addition to the executables, LIBXSIF stores global data in Fortran-90 modules, which are accessed by USE statements in subroutines that require access to the data. The modules are XSIF_SIZE_PARS (global parameters for setting the sizes of statically-declared arrays), XSIF_INOUT (data related to input and output), XSIF_ELEM_PARS (fixed parameters

[†] Work supported by the U.S. Department of Energy, Contract DE-AC03-76SF00515.

related to element defintions, such as dictionary arrays for each element type), XSIF_ELEMENTS (parsed element data), XSIF_CONSTANTS, XSIF_INTERFACES (explicit interfaces for a handful of procedures).

Historically, programmers have been of two minds on the desired "look and feel" of Standard Input Format parsers in their simulation programs. Some programs, such as MAD and DIMAD, have incorporated the parser into their command input stream, so that beamline definitions and simulation commands co-exist in a single file. Other programs, such as LIAR, keep the beamline and command file separate: a command in the command input stream directs the command parser to open, parse, and close a separate file full of beamline definitions. In addition, many programs add capabilities over and above simple beamline parsing to their Standard Input parsers (DIMAD is a prime example of this). We have configured LIBXSIF to accomodate all of these preferences.

XSIF_IO_SETUP This is the top-level routine for applications in which beamline information is not to be stored in the command input stream. XSIF_IO_SETUP takes arguments which indicate the filenames of the deck file, the XSIF output stream, and the XSIF error stream, as well as logical i/o unit numbers for each. XSIF_IO_SETUP will open each of these files with the appropriate status, and return a good status if all file-open operations were successful, or an error otherwise. XSIF_IO_SETUP will also call RDINIT, CLEAR, and INTRAC (see below).

RDINIT Subroutine RDINIT initializes various i/o variables and writes the LIBXSIF header, with version information, to the screen, the standard output, and the error stream.

CLEAR Initializes variables related to the parser (for example, number of elements parsed so far).

INTRAC A dummy function that tells LIBXSIF that non-interactive operation (the only supported mode at this time) is selected.

XSIF_CMD_LOOP This is the master command loop which reads new lines from the deck file and takes appropriate actions. XSIF_CMD_LOOP will continue to parse the requested file, and/or any files which are OPENed and CALLed from the requested one, until one of the following: a fatal read occurs, in which case all routines in the call chain will abort execution as quickly as possible; a function or subroutine in the call chain requests a halt to parsing (for example, if the STOP command is encountered), in which case all routines in the calling chain will complete execution normally; or exhaustion of all data in the file(s) of interest. In all cases, XSIF_CMD_LOOP will ultimately return control to its calling routine (i.e., it will not abort execution of the program), with a status or error message.

One of the arguments of XSIF_CMD_LOOP is an optional function name. If a function name is provided, XSIF_CMD_LOOP will permit that function to attempt to manage the most recent line of input before the standard command loop; in this way, users can add their own extensions to LIBXSIF without ever touching the existing codebase.

XSIF_IO_CLOSE Closes files opened by XSIF_IO_SETUP.

XUSE2 Permits the calling routine to specify the name of a beamline to be expanded for simulation. This is identical to specifying a beamline with a USE statement in the deck, but it permits the calling program to pass a name rather than relying on the deck file to provide it.

3 EXAMPLES

Two programs which use LIBXSIF to parse beamlines are LIAR and NLC-DIMAD version 2.8. LIAR's "look and feel" dictate a separate beamline and command stream, while DIMAD's traditional interface combines the two.

3.1 LIAR

The LIAR command READ_XSIF specifies a beamline file for parsing:

```
read_xsif,
file= ‘‘main_linac.xsif’’,
line = ‘‘linac’’
```

indicates that the file "main_linac.xsif" is to be opened and parsed, and that a beamline named "linac" is to be expanded for simulation purposes at the end of parsing. The subroutine READ_XSIF in LIAR calls XSIF_IO_SETUP, which opens the desired file; it then calls XSIF_CMD_LOOP to perform parsing and XSIF_IO_CLOSE to close all files that were opened. Finally, XUSE2 is called with "linac" as an argument; this causes the expansion of the requested beamline in a manner identical with a `use, linac` statement at the end of the deck file.

3.2 DIMAD

The traditional DIMAD "look and feel" incorporates beamline information and simulation commands into a single input stream; in addition, DIMAD's version of the Standard Input parser has several commands, including DIMAT (which signals the beginning of simulation) which are not part of traditional Standard Input. To preserve these features, DIMAD does not use XSIF_IO_SETUP (which would automatically open a file separate from the command stream for beamline information), and the subroutines RDINIT, CLEAR, and INTRAC are called by DIMAD itself. The extra commands in DIMAD's version of the parser are handled in subroutine DI-

MAD_XSIF_EXTRA, which is passed as an argument to XSIF_CMD_LOOP.

4 WHERE TO GET IT

LIBXSIF was primarily constructed for use by the Next Linear Collider project at SLAC, but with some consideration of its potential utility to other accelerator designers. All of the potentially-useful bits and pieces (binaries, source, documentation, and LIAR and DIMAD examples) is freely available over the World Wide Web. The parent URL for all information is:
`http://www.slac.stanford.edu/accel/nlc/local/AccelPhysics/codes`.

4.1 Source Code

Folder `xsif/src` contains up-to-date source code for LIBXSIF, including a TAR of the entire source.

4.2 Binaries

At the present time, binary versions of LIBXSIF are available via WWW for Solaris (in `xsif/bin`) and for Windows running on Intel compatible CPUs (in `xsif/binnt`). A version for IBM's AIX operating system is available, but this version is not up-to-date due to the elimination of AIX support at SLAC. While LIBXSIF will compile without errors on VMS/Alpha as well, web access to this version is not available at this time. A version for Linux will be made available in the second half of 2001.

Note that, since LIBXSIF is a Fortran-90 library, it is necessary to acquire the "module" files (binary versions of INCLUDE files) in order to link against LIBXSIF.

4.3 Documentation

A detailed guide to LIBXSIF is available in `xsif/doc`, in PostScript or Portable Document Format.

4.4 Examples

Sometimes the best documentation is an example. The relevant portions of LIAR's connection to LIBXSIF can be seen in the file `read_xsif.f`, in the `liar/src` folder. DIMAD's connections to LIBXSIF are in files `madin_new.f` and `dimatd.f`, in `dimad/source/v2.8`. Note that these files also show examples of how to strip beamline information out of LIBXSIF's structures.

5 REFERENCES

[1] D.C. Carey and F.C. Iselin, "A Standard Input Language for Particle Beam and Accelerator Computer Programs," Proceedings of the 1984 Summer Study on the Design and Utilization of the Superconducting Super Collider, Snowmass, Colorado (1984).

[2] K.L. Brown *et al*, "TRANSPORT: A Computer Program for Designing Charged Particle Beam Transport Systems," SLAC-Report-91 Rev. 2 (1977).

[3] H. Grote, "The MAD Program User's Reference Manual," CERN/SL/90-13 (AP) Rev. 5 (1996). See also CERN-LEP-TH notes 83-30, 85-15, 85-38, and 87-33.

[4] R.V. Servranckx *et al*, "User's Guide to the Program DIMAD," SLAC-Report-285 (1990).

[5] D.C. Carey *et al*, "Third-Order TRANSPORT with MAD Input," SLAC-Report-530 (1998).

[6] D.C. Carey *et al*, "TURTLE with MAD Input," SLAC-Report-544 (1999).

[7] C. Hawkes and M.J. Lee, "Recent Upgrading of the Modeling Program COMFORT," SLAC-CN-342 (1986).

[8] R. Assmann *et al*, "LIAR: A Computer Program for the Modeling and Simulation of High Performance Linacs," SLAC-AP-103 (1997).