SAX data analysis software

Rules and recommendations for Fortran programs

A proposal to the
SAX data analysis working group
by

L.Chiappetti (IFCTR)

Draft - September 1986

This document is file RULES.SAX

1 : Introduction

This document is the second in a collection of notes for discussion within the SAX data analysis working group. The final aim of this group should be the production of a set of rules, which should enable the different software modules written at the various SAX hardware institutes to be a) easily integrated in a common SAX data analysis package at the Scientific Data Centre (CDS); b) easily exchanged between the hardware institutes (and the scientific community at large ?).

A driving consideration has been so far (August 1986) the fact that the hardware institutes presently have different computers, and that this situation is not likely to change on a short term. Hence there is no immediate compatibility between software packages running at the different institutes. The goal of a machine-independent software is worthwhile to be mantained even if a transition occurs towards a situation of hardware compatibility between the institutes of the SAX Consortium, as it is currently under discussion for the Italian Institutes.

As a preliminary step I have prepared the document "SAX data analysis software - A study about Fortran compatibility", which has been distributed to the members of the Data Analysis Working Group in draft form in mid-August. The latter, and the draft document by Morini quoted therein, will constitute the reference documents for the present work.

Additionally, the documents listed here have also been freely consulted. We thank A.Spizzichino of TESRE for having collected them and kindly provided them to the Data Analysis Working Group.

Rules for software programming in Fortran language (Hipparcos FAST Consortium, Dec. 1983, Pieplu, J.L. and Huc C.)

File access rules for operational software programming in Fortran language (Hipparcos FAST Consortium, Jan. 1984, Huc C.)

Norme per la documentazione di programmi (d.n.a, Fini L.)

Fortran-77 and Portability (Conv.naz. Astronet 1984/85, Fini L.)

Starlink Application Programming Standards (June 1981, Wallace P.T)

ESO Programming and Documentation Guidelines (d.n.a, a.n.a)

GSSS-TM 83/8 (ST) Software Standards (Apr. 1983, a.n.a.)

Document for which no computer readable source exists any longer

Partial scan of original hardcopy (available on request) supplied

The above documents have been used only when applicable, because of different environment and requirements.

The present document will not deal with the items of graphics and documentation (except in a limited way for on-line comments). Specific documents should be probably dedicated to the above items (not excluding further preliminary investigations).

The present document will not develop a proposal about the functions and the commands of the SAX data analysis environment, or the file structures. However some hints thereabout may be incidentally given.

The layout and content of the present document is intended as a proposal for the final document to be produced by the data analysis working group.

2 : Compatibility Standards

Many possible ways exist to enforce software compatibility on different machines. They are listed below : we note that they are not mutually exclusive, but can (and possibly) will co-exist concurrently.

a) The simplest and safest (but not always possible) way, is to limit oneself to a **subset of the language common to all machines.** This will be adopted as the main criterion in the set of rules.

b) If some functions can be provided in a very efficient way on some systems (and emulated via users' software on other systems), or if it is not possible or desirable to renounce to some system-dependent feature, they could be **encapsulated in a common interface subroutine.** The main program will be portable, only a different system-dependent library will be linked on each machine (on some machine the routines will just be no-operation perhaps).
A case for this will be sometimes indicated in the set of rules.

c) For some repetitive features, which are provided in standard different ways on different system, or which can be converted in a standard way, the **use of a pre-processor** to convert code running on a machine to a different one may be of some help.

d) Another way would be to provide **parallel pieces of code** for different machines in the same program, all commented out. Usage of a particular piece of code will be enabled by removing the C or * in column 1. This could be the case of some special statements or directives which must be present on some machines.
The case for this will also be indicated in the set of rules.

Actually a detailed proposal for solution d) is also presented here (this could also be used in conjunction with pre-processor or edit macros) :

d1) Single lines shall be preceded by a comment like **Csys1**, where sys is a three letter code like IBM, VAX, HP-. To preserve alignment and possible insertion of continuation lines, column 6 must be left free from comments, also no labels are allowed. To enable a line, just replace **Csys1** with 5 blanks.

d2) Blocks of code (including labels in columns 2-5 and continuation in column 6) shall be commented out with a C in column 1. The block will be preceded by a a card containing **Csysn** (where n is the number of lines in the block, to help any pre-processor, or compare utility), and terminated by a card containing **Csys***. To enable a block just replace the C with a single blank in all lines between but not including a **Csysn** and a **Csys***.

The methods described above will be referred as **solutions a-d** in the rest of the document.

In the set of rules given below, I have indicated a number of severity levels (attributed in a tentative way only). All rules are identified by a monotonically increasing sequence number, irrespective of the severity level (to allow easy re-classification of a rule). The levels are indicate by a two-letter code:

Ru:n A mandatory rule, which shall always be obeyed. These are mainly driven by hard compatibility requirements.

Re:n A strong recommendation. Possibly we should request a written justification to deviate from this, or consider case by case.

St:n A style recommendation, included mostly in order to keep a good and uniform programming style. May not be adhered to, if a self-consistent, different style is used throughout the program.

A decision about the actual severity level for each rule will be left to the data analysis working group. However I have tried to keep the number of Ru:n to a minimum (wherever required by examination of the Fortran Compatibility document), and would not be so strict to ban individual programming styles. Experienced programmers may go on "almost" as they were used to. On the other hand beginners will be strongly encouraged to follow Re:n and St:n too.

2 : Rules and Recommendations

0000-0999 General : language definitions

Ru:0001 The language to be used is ANSI Fortran 77 (reference and applicable documents TBD) with the common extension supported by all machines (Reference document could be the final version of my first document, applicable documents TBD).

Ru:0002 Only the character A-Z (upper case), 0-9, ( ) = + - * / ' . , " : and $ shall be used in program code.

Re:0003 Use of lower case in code (on machines which allow it) is discouraged, since it does not improve clarity of presentation. A pre-processor shall be used to convert any lower case code.

Re:0004 Any other printable character may be freely used in **comments** and in text to be output via FORMAT statements.

St:0005 Usage of lower case in comments and text to be output is actually recommended, since it improves the clarity of presentation.

Ru:0006 Use of tabs and non-printable characters is forbidden. Non-printable characters for e.g. terminal control is allowed in system-dependent routines (solution b).

Ru:0007 Variable names shall be a maximum of 6 alphanumeric characters, starting with A-Z. Use of underscore is forbidden.

Re:0008 Use of long meaningful names is preferred to short generic names (A, AQ, B) except for work areas (I,J,K and so on).

Re:0009 Do not use lower bounds for arrays (which is equivalent to bounds 1:n), unless it is required for clarity (and then propagate correctly through all subroutines !).

Ru:0010 Subscripts shall be INTEGER (use INTEGER tout court), or anyhow evaluate to valid machine-dependent INTEGER.

Re:0011 Dimensions as ARRAY(*) are preferred to ARRAY(1) for adjustable arrays in subroutines. Be careful with multi-dimensional arrays (it is probably safer to allocate the maximum range, not to use adjustable arrays).

Ru:0012 Use of intrinsic functions is limited to TBD list.

Re:7014 All references to subroutines which are called only by one specific program shall be resolved internally, i.e. the source must be in the same file as the main. All other references must be kept in libraries. No individual relocatables are allowed for subprograms.

Ru:7015 A maximum of 8 libraries shall be allowed to be searched for each main program.

Re:7016 A way to handle a single source file for a library should be sought.

Re:7017 Should deletion and renaming of files inside a program be allowed (if at all possible) ?

Re:7018 How does one handle the scheduling of other programs (very system dependent) ?

Ru:7019 Passage of parameters from the run string is anyhow a desirable feature (solution b).

Re:7020 An agreement shall be reached whether program must be menu-driven, command-driven, command-file-driven, fully interactive, etc.

Re:7021 Terminal screen management is a desirable feature (solution b)

Re:7022 Also form-filling mode may be a desirable feature (solution b)

Ru:7023 Help files must possibly use the local (system-dependent) facilities, and a pre-processor to convert the help file text from one site to the other.

Ru:7024 A set of system-dependent routines with common interface (solution b) is desirable for :

current date and time
elapsed or CPU time measurement
random number generator
move-word utility
bit-field handling
number of bytes actually transferred in i/o transaction

Re:7025 A common solution towards error handling must be sought. Statements shall be protected towards mathematical library errors (check if in range etc.). All I/O statement shall use the ERR clause (although not all errors are trapped ..).
All input values must be verified if valid and if in the allowed range. All error return codes from subroutine must be checked.

Re:7026 Commonly used mathematical functions must be unified (look for source libraries available "on the market")

Re:7027 Also (home-grown) for main functions used for general astronomy or X-ray astronomy (e.g. spectral forms).

Re:7028 Usage of assembler routines shall be reduced to a minimum, even under solution b. Usage of (system) Fortran-callable routines is allowed under solution b.

Re:7029 File structures non sunt multiplicanda praeter necessitatem.