VLBA CORRELATOR: PROGRAMMING STANDARDS

T. J. Pearson

February 6, 1985

INTRODUCTION

The Correlator Design Group has been asked to present its views on standards for software Specification, Documentation, and Coding. The Correlator Group recognises the importance of such standards and intends to define standards for its own use. It will make every effort to ensure that these standards will be compatible with those adopted elsewhere in the project. The following discussion does not attempt to define the standards, but only to list some of the areas that should be standardised.

SPECIFICATION AND DOCUMENTATION STANDARDS

There will be a number of Documents produced as part of the VLBA Correlator Software Subsystem. Here we describe only the Detailed Design Document which will be produced for each program unit as part of the design phase (and refined during implementation and coding). The list of contents is intended as a suggestion for discussion.

Section 1 – Introduction

1.1 Purpose [the place and function of this unit in the Correlator software subsystem].
1.2 Scope [defines the program unit defined in this document, and the level of this document (*i.e.*, design/as built/as delivered)].
1.3 Design Overview [overall description and top-level diagram].

Section 2 – Related Documents

2.1 Applicable References [the documents from which this design was derived].
2.2 Development Documents [which this supersedes].

Section 3 – Functional Description

[Operations performed by the software, sequence of operations, decision points, *etc.*]

Section 4 – Module Design [one section for each module]

4.1 Purpose of Module [name, configurational control identification number, abstract, language used; major interfaces].
4.2 Functional Description [organisational structure of the the module, data flow, control, external interfaces and interfaces with other modules].
4.3 Procedures [one subsection for each procedure] [this will be the same as the prologue comments within the procedure code: see below].

Section 5 – Data Bases

[Includes a detailed definition of the content and storage locations of each file, table, and item within each table that is incorporated in the database; length and format of file; access methods.]

Section 6 – System Resource Allocation

[Memory usage, use of temporary disk files, *etc.* Detailed timing analysis. Requirements for other limited resources, *e.g.*, printers, displays.]

Section 7 – Required Utilities

[Utility subroutines, such as graphics, that are required by this unit but are not provided as part of it.]

Section 8 – Notes

Appendix A – Acronyms and Abbreviations

CODING STANDARDS

It is planned that the Correlator Control System will be a multiprocessing system programmed in a mixture of languages (probably C and FORTRAN). It is important to standardize and document the interfaces between different modules, procedures, and processors. It is also desirable to adopt coding standards within procedures to enhance their readability and maintainability.

This includes standards and conventions for languages, prohibited coding practices, required coding practices, and recommended coding practices.

A *module* is a single procedure or group of related procedures intended to perform some specific task.

A *procedure* is a single routine (a Subroutine or Function in FORTRAN) which performs a defined task. The same procedure should be used for the same task wherever it is performed in the system.

Standards should be adopted for:

- **Code dialect.** We might suggest that FORTRAN procedures be coded in ANSI standard FORTRAN-77, possibly with some restrictions (*e.g.*, no use of the Arithmetic IF statement, the ASSIGN statement, the EQUIVALENCE statement), and possibly with a limited number of extensions (*e.g.*, the IMPLICIT NONE statement and the INCLUDE statement); and that C procedures be coded according to the standards defined in Kernighan and Ritchie's *The C Programming Language*. We should aim to use "industry-wide" standards rather than adopting our own; we should also try to adopt standards which can be easily enforced by machine (*e.g.*, by use of the /STANDARD option of the VAX FORTRAN and C compilers.)
- **Code layout and formatting.** This is a controversial subject, but we hope that we can arrive at a consensus. Examples of areas which might be standardized: indentation in FORTRAN IF—THEN—ELSE—ENDIF blocks and DO loops and in {...} groups in C; placement of comments (example: start FORTRAN comments in column 17 and offset with blank lines before and after). Again, it is helpful if the standards adopted can be enforced my machine.
- **Prefatory comments.** Each module and procedure should have a preface at the start of the code which documents its function and how it is to be used. The preface should be laid out in a standard form that enables it to be extracted from the code by machine as a first stage in building a program manual. The preface should include the following:
  Title
  Version
  Facility (*i.e.*, which part of the system it belongs to)
  Abstract
  Environment (any restrictions on where in the system this module may be used)
  Author/date
  Modification history
  Functional description/processing method
  Limitations
  Calling sequence(s) (particularly important if the procedure may be called from more than one language). Any special initialization procedures should be noted.
  Parameters (name, read/write/modify, data type, meaning)
  Implicit inputs (does the procedure take inputs from files, common areas, operating system [*e.g.*, logical names], *etc.*?)
  Implicit outputs
  Return value (for procedures which return a value, *e.g.*, FORTRAN functions)
  Side effects (changes to memory allocations, files, *etc.*)
  Error conditions
  Subroutines required
  Size and timing estimates

- **Naming rules and conventions.** Every name that has *global* significance (*e.g.*, module names, procedure names, FORTRAN common blocks; these are called *external* variables in some languages) must be standardised to avoid conflict and to improve readability. Long names are helpful here as one can include a "facility prefix" as in the VMS standards (*e.g.*, LIB$GET_INPUT is better than ZGTIA0). Unfortunately FORTRAN-77 limits names to 6 characters: can we relax this requirement? File names should likewise be standardised.
- **Error reporting.** A single centralised mechanism for reporting errors must be adopted. Programs should not be allowed to write messages directly. In fact all I/O should be confined to a small number of routines.
- **Use of resources.** A single mechanism for allocation of dynamic memory should be used. Other system-wide "resources" (*e.g.*, FORTRAN unit numbers) should be allocated by a single mechanism (*e.g.*, specify in advance how each unit number is to be used, or allocate numbers from a system-wide pool as needed).
- **Order and type of procedure parameters.** *e.g.*, list parameters in order read, read-write, write. Some data types supported by the language may not be appropriate for use as procedure parameters if they cannot be understood in another language.

CONFIGURATION MANAGEMENT

Procedures must be adopted to ensure the following:

1. When a section of code is updated, all parts of the system that use that code-section (*e.g.*, by explicit INCLUDE statement or referencing it as a subroutine) are updated to to the new code, and fully tested.
2. Code must be certified that it follows the adopted standards and performs as specified before it is incorporated into a baseline software version.
3. Each baseline version is fully tested and documented.
4. Each baseline version is archived and can be reinstated if necessary.
5. All changes are approved and documented before introduction.
6. When more than one project member works on the same file at the same time, none of the modifications made are lost.
7. A record is kept of all changes introduced in all baseline software versions.

The VLBA Correlator Group expects to develop or purchase software to assist in the management of the software development. Available commercial products should be studied to determine their cost-effectiveness in the VLBA project. Available products for VMS include:

- **DEC/MMS.** This automates the updating of a software system when any component is changed. This is a fairly straightforward procedure which could be done using locally-developed command procedures. MMS merely provides a standardised way of doing this, integrated with VMS.
- **DEC/CMS.** This is a tool for coordinating program development by a team of programmers and for maintaining baseline systems. Its functionality includes that of the AIPS Checkout scheme, but goes considerably beyond it.
- **DEC/Test Manager.** This automates the organisation, execution and review of tests and allows several developers to use one set of tests at the same time. [This is perhaps not very applicable to a real-time system like the correlator, where the same test is not expected to give exactly the same results every time it is run.]