# A proposal for XAS Software Specifications

Final draft 0.5 - January 5, 1990

Prepared by: L.Chiappetti - IFCTR, M.Morini - IFCAI

"Dieses Buch wird vielleicht nur verstehen, der die Gedanken die darin ausgedrueckte sind, oder gern aehnliche Gedanken, schon selbst einmal gedacht hat. (...) Sein Zweck waere erreicht, wenn es einem, der es mit Verstandnis liesst, Vergnuegen bereitete."

L. Wittgenstein, Logisch-philosophische Abhandlung, Vorwort

"This book will perhaps be understood only by somebody, who already had the thoughts expressed herein, or quite similar thoughts. (...) Its purpose will be reached, if it will bring pleasure to an understanding reader."

L.Wittgenstein, Tractatus Logico-philosophicus, Foreword

## Table of Contents

Table of Contents ....... I-1
Amendment history ....... II-1
1. Context ....... 1-1
1.1 Reference documentation ....... 1-2
2. Introduction ....... 2-1
2.1 Essentials of XAS ....... 2-2
2.1.1 Organization of the document and preliminary definitions ....... 2-2
2.1.2 Summary of main features of XAS ....... 2-3
2.1.3 A sample session ....... 2-4
3. VOS Specifications ....... 3-1
3.1 Program execution ....... 3-3
3.2 Program communication ....... 3-6
3.3 File System Management ....... 3-8
3.4 Other I/O routines ....... 3-16
3.5 Miscellaneous routines ....... 3-17
3.6 VOS Programmer Library ....... 3-18
4. VD Specifications ....... 4-1
4.1 Virtual terminal device ....... 4-4
4.1.1 Definition of the virtual terminal device ....... 4-4
4.1.2 Virtual terminal interface routines (Level 1) ....... 4-5
4.1.3 Virtual terminal interface routines (Level 2) ....... 4-8
4.2 Virtual graphical device ....... 4-10
4.2.1 Definition of the virtual graphical device ....... 4-10
4.2.2 Introduction to the virtual graphical device interface ....... 4-11
4.2.3 Virtual graphical device interface output routines ....... 4-15
4.2.4 Virtual graphical device interface input routines ....... 4-20
4.3 Virtual pictorial device ....... 4-21
4.3.1 Definition of the XAS virtual pictorial device ....... 4-21
4.3.3 Virtual pictorial device interface routines ....... 4-22
4.4 Virtual array processor ....... 4-25
5. User Interface Specifications ....... 5-1
5.1 XAS User Interface routines ....... 5-5
6. Disk Space Management Rules ....... 6-1
7. Data Structure (Layout) Specifications ....... 7-1
7.1 Data flow ....... 7-1
7.2 Data structures ....... 7-4
7.2.1 RRD Archive ....... 7-8
7.2.2 Telemetry files ....... 7-9
7.2.3 Reduced data files ....... 7-10
7.2.3.1 Lists of photons ....... 7-11
7.2.3.2 Images ....... 7-13
7.2.3.3 Spectra ....... 7-16
7.2.3.4 Light curves ....... 7-20
7.2.3.5 Response matrices ....... 7-22
7.2.3.6 Ancillary files ....... 7-24
7.2.4 Printout files ....... 7-24
7.2.5 Table files ....... 7-25
7.2.6 Calibration files ....... 7-26
7.2.7 Result data base ....... 7-27
7.2.8 Data archiving and exchange ....... 7-29

8. XAS Monitor Specifications ....... 8-1
8.1 Introduction to the XAS Monitor ....... 8-1
8.2 Accessing the XAS Monitor ....... 8-3
8.3 Processing commands ....... 8-4
8.4 XAS Monitor internal commands and functionalities ....... 8-6
9. XAS Command Specifications ....... 9-1
9.1 Commands replacing system commands ....... 9-1
9.2 Context definition commands ....... 9-9
9.3 RRD Archive Management commands ....... 9-11
9.4 Filing and related commands ....... 9-12
9.5 Data accumulation commands ....... 9-15
9.6 Reduced data display and reduction commands ....... 9-20
9.6.1 Display of file content in alphanumeric form ....... 9-20
9.6.2 Display of files in graphical form ....... 9-21
9.7 Data analysis commands ....... 9-33
9.7.1 Energy spectra analysis commands ....... 9-33
9.7.2 Timing analysis commands ....... 9-40
9.7.3 Image analysis commands ....... 9-44
9.7.4 Auxiliary commands ....... 9-46
9.8 Experiment simulation commands ....... 9-47
9.9 Reduced data archiving and format conversion commands ....... 9-48
9.10 Automatic analysis and result data base ....... 9-50
10. Mail and Networking area ....... 10-1
11. XAS Distribution Policy ....... 11-1
12. Program Development in XAS ....... 12-1
12.1 Tools for program development ....... 12-2
12.2 Rules for program development ....... 12-4
13. Rules for XAS Documentation ....... 13-1
13.1 Users' Documentation ....... 13-1
13.1.1 On-line documentation ....... 13-1
13.1.2 Users' Manual ....... 13-2
13.2 Programmers' Documentation ....... 13-2
13.3 Documentation format and configuration control ....... 13-3

## Amendment history

| | | |
|---|---|---|
| **Preliminary Draft (0.0)** | 19 August 1988 | First edition of a preliminary draft prepared by L.Chiappetti |
| Draft 0.1 | 26 October 1988 | VOS and VDI specifications expanded by M.Morini. Other minor changes. |
| Draft 0.2 | 15 February 1989 | User Interface specifications expanded by M.Morini. Networking section by L.Chiappetti inserted. Other minor changes. |
| Draft 0.3 | 8 March 1989 | XAS Monitor specifications expanded by M.Morini. XAS command specification partially expanded by M.Morini. Data structure and other XAS command specifications inserted from RD 2. Other minor changes. Presented to the DAWG. |
| Draft 0.4 | 31 July 1989 | Sections 1-3 include comments by L.Chiappetti. Sections 3-5 include changes derived from the development of the sperimental version of the VOS interface for the VMS and of the User Interface by M.Morini. |
| Final draft 0.5 | 5 January 1990 | Title modified: All sections revised and sections 7 to 13 completed by L.Chiappetti. All sections are now mantained as separate document, which will be updated independently. For release to the DAWG. |

# 1. Context

The functions and activities of the SAX "Centro Dati Scientifici" (SDC) have originally been outlined in the document "A Proposal for the Organization of the Activities concerning the SAX Data Analysis, Calibration, Distribution and Archive (the "Centro Dati Scientifici")", prepared by the SAX Data Analysis Working Group (DAWG) and dated December 15th, 1987 (RD 1), which was approved by the SAX Coordination Group (SCG) at its meeting of January 28th, 1988. At the same meeting a definition team (DT, composed by L.Chiappetti, acting as coordinator, M.Morini, acting as deputy coordinator, D.Dal Fiume, P.Ubertini, plus M.B.Negri as observer for PSN and M.Manca as observer for Telespazio) was set up (formalized at the SCG meeting of June 30th, 1988.

An extensive set of preliminary elements for discussion (mainly about the SDC software) have been put forward in the document "Elements for a discussion concerning the SAX CDS software components" dated March 31th, 1988 (RD 2), compiled by the coordinators and discussed at the DT meeting of April 27th, 1988. As some elements, mainly concerning management or anyhow non-software items, were felt to be missing, a further document was prepared after the plenary DAWG meeting of June 6th, 1988. This is the document "Considerazioni del Data Analysis Working Group sulla realizzazione del CDS (SAX/CDS/02)" dated June 17th, 1988 (RD 3), which was submitted to the SAX Program Manager by the Program Scientist in charge at the time.

The hardware components of the SDC, in terms of requirements, are described in the document "Computer System Requirements for SAX Scientific Data Analysis" dated September 8th, 1987 (RD 4), compiled by a subgroup of the SAX Ground Segment Working Group (GSWG). This description pertains to a SAX Computer System (SCS), which includes the Local Centres (LC) at the Institutes and the SDC proper (note that, for what concerns the software to be developed , the SDC is to be intended as comprehensive of the Local Centres). A revision of such document (RD 4b), limited to the selected starting configuration for the LCs has been prepared in June 1989 as requested by ASI for the contract for the Bridging Phase of the SAX Ground Segment. The activities for the procurement of the hardware at the Local Centres were started within a dedicated Technical Committee nominated by the PSN on December 1st, 1987, which defined a hardware configuration based on a DEC VAX machine running under the VMS operating system, described in the document "Definizione dei sistemi di calcolo locali per il programma SAX" dated February 29th, 1988 (RD 5).

This document is intended to expand RD 2 and to give the full detail about the X-ray Astronomy analysis System. The format of this document is that of specifications; the reader should however consider that the development of a system of this kind is a multi-step process in which programs are stepwise refined and modified.

The ideas expressed herein remain intellectual property of the authors, which reserve to themselves the right of using such ideas, jointly or individually, in the context of other research projects.

At the same time of the first public release (0.3) of the present document, extensive discussion has occurred with ASI about the nature and status of the SDC. As such it has been agreed that the SDC would be developed by ASI within the framework of the Ground Segment contract, or anyhow related to it, and that part of the activities of the SDC could be fulfilled within an industrial contract. Such Mission Support activities are therefore described in a requirement document prtepared for ASI ("SAX Scientific Data Centre Requirements; SAX/GS/SDCUR, June 1989, RD 6).

## 2. Introduction

XAS is the X-ray Astronomy analysis System which is proposed to be be developed for the SAX mission. However, while it should be capable to cope with all aspects of the SAX data reduction and analysis, XAS aims to be a general purpose package for X-ray astronomy, which could be used with minor additions to analyse X-ray astronomy data from any mission.

XAS will be developed within the framework of the SAX Project, therefore it has to run immediately at the principal sites in charge of SAX data analysis, namely in the order described in the section "XAS distribution policy" (11) below.

XAS software and documentation will be produced by a team composed by people residing at the SAX SDC in conjunction with people in the Consortium Institutes. XAS will be maintained by SDC staff. SDC will also be responsible for XAS distribution.

XAS will be a collection of software modules, based on a (Reduced Instruction Set) Virtual Operating System (VOS) concept. All calls to operating system services will be encapsulated in dedicated VOS interface routines. The commitment of the SDC is to produce a VOS interface library for DEC VMS operating system. However, in the definition of the specification for the VOS, described in the section "VOS Specifications" (3) below, consideration shall be taken about the functionality of UNIX for future developments and of previous experience with other systems by people in the XAS team. The minimum requirement for the operating systems to which the VOS could be interfaced is multitasking and hierarchical file system. This requirement could, in line of principle, be relaxed. The question is academic, as both VMS and UNIX correspond to those requirements, and we cannot realistically think of other systems so far.

However the requirement of multitasking derives only from the fact that the monitor shall schedule the application commands (we have no strict requests of background jobs, nor of concurrent communicating processes), and under other OS's different solutions (overlaying) could be feasible. The "environment" (global variables) are anyway best kept either in a system (virtual memory) area, or in a file, and not in the core by the XAS monitor (this will make easier to mix XAS with other astronomical packages, if anybody desires to do so... the converse is not necessrily true, but that is not our fault).

Also the requirement of hierarchical file system is not a strict need (we are interested in defining different areas for different kinds of data, but there is not a definite hierarchy implied in the data organization itself).

XAS will also exploit a Virtual Device (VD) interface concept. Actions on special peripheral devices (e.g. graphic-pictorial units and array processors) will be encapsulated in dedicated VD interface routines. VD routines will ultimately interface the operating system via VOS calls. The commitment of the SDC is to produce a set of VD interface libraries for all devices officially supported in the SCS. See section "VD Specifications" (4) below.

## 2.1 Essentials of XAS

### 2.1.1 Organization of the document and preliminary definitions

The XAS environment is essentially a set of commands acting upon a set of data structures. A monitor program or shell will be used to coordinate the various commands. The XAS monitor and the commands will interact with the user via an User Interface. Data structures will be mostly disk files, and as such there is a need for rules for disk space management in XAS. Therefore sections will be dedicated below to:

VOS Specifications (3)
VD Specifications (4)
User Interface Specifications (5)
Disk Space Management Rules (6)
Data Structure (Layout) Specifications (7)
XAS Monitor Specifications (8)
XAS Command Specifications (9)

As a further topic, of primary concern for both use and development of XAS, sections will be added covering:

Rules for Program Development in XAS (12)
Rules for XAS Documentation (13)

We give at this point some definitions of terms, as they will be used in this document, which may have a different interpretation in the slang used by different hardware vendors or in the context of different different operating systems.

By EXECUTIVE we define those programs which integrate the operating system and are permanently loaded in memory. It consists of the control program or monitor itself (operating system executive, resident monitor or kernel in different systems), of the peripheral drivers, the system services, and other programs depending on the actual operating system.

By SYSTEM SERVICES we define those routine provided by the operating system that can be called by user programs.

By PROCESS we define the basic entity scheduled by the system software that provides the context in which a program executes. A process may execute a sequence of programs under the control of a Command Interpreter, or a single program and terminate at program termination (as in the case of the login; in UNIX all commands are individuals processes, and the Command Interpreter itself is a process).

By PROGRAM we define both what in high level language terms is a MAIN PROGRAM (is source or object form, as a counterpart to SUBPROGRAMS possibly stored in a LIBRARY) and the result of its link-editing as an autonomous EXECUTABLE MODULE.

By COMMAND we define an instruction typed by the user at a terminal or included in a command procedure that request to perform a well defined activity.

By COMMAND INTERPRETER we define the entity (either a program running in an individual process as the UNIX shells, or a system code running in the context of a process as in VMS) that receive, check the syntax of, parse commands, and activates the corresponding (system or user) programs for execution. Among Command Interpreters we define SYSTEM INTERPRETER the one that processes system commands (CLI in the VAX/VMS, CMS in the IBM VM, the Bourne and C shells in UNIX, etc.) and ENVIRONMENT INTERPRETER a command interpreter layered upon a system interpreter to perform some user specific actions.

By SYSTEM PROCEDURE we define a sequence of system commands, including program invocations, which are processed by a system interpreter. Similarly a PROCEDURE, without the "system" attribute,

# 3. VOS Specifications

The rationale for the VOS (Virtual Operating System) concept is described in the following section. We go here beyond the general motivation for portability (both in space, i.e. among different sites, and, perhaps more important, in time, i.e. with the aim of saving the efforts done with the possibility of re-using the code on new systems introduced in the future), which should be obvious. It has to be noted that the requirement for portability must not affect efficiency (particularly in the sense of introducing too many calling overheads). On the other hand the VOS routines will be explicitly designed to improve the i/o speed (therefore any calling overhead should be compensated by such improvement).

However in some cases the degree of complexity and insight into the system necessary to attain a given (may be slight) improvement on a given (may be seldom used) function may be large (unusual logic, use of assembler routines, etc.). Although we probably could not do better in term of efficiency than professional compiler writers, in particular cases the way a compiler handles i/o is notably inefficient, with respect to other possibilities provided by the system, may be already in the form of library routines, which can just be "hidden" within our VOS interface routines.

The guidelines are therefore as follows:

We will define a list of functions as general and complete as possible, and as "natural" as possible (however bearing in mind particularly but not only the VMS and UNIX peculiarities).

However it is likely that only a subset of such functions will actually be required within XAS. Therefore the not-required functions will not be implemented (at least in the first step).

One should not be misled by the name of VOS to think this is an over-ambitious approach. In fact the approach proposed here is far less ambitious then other approaches (e.g. the one followed in IRAF, where an entire new language and a dedicated programming environment have been set-up, together with a maximal approach to supply all system calls under any operating system), and could appropriately be called a Reduced Instruction set VOS (RI-VOS). Moreover (see section 12) it is our intention to have the code in standard Fortran.

Some required functions may be implemented using existing high level system routines ("masked" by the VOSI), or even by using plain Fortran calls (even if this results in a slow down). The latter Fortran solution may be available immediately. It should be regarded as an "interim VOS" interface. We will therefore have (at some time in the future and with an appropriate timeline):

| | |
|---|---|
| a VOSI-library for VMS | (baseline requirement) |
| a VOSI-library for UNIX | (optional add-on) |
| an interim VOSI-library | (by-product) |

The advantage will be that we could start writing programs using the interim library for some functions, and reload the programs with the full library as improved routines become ready. On the other hand the interim library will be available to third parties as a skeleton by which not-SAX sites could have a (degraded performance) operation of XAS, and an example of what they should do to write their own interface.

The following is the original list of functions (prepared by M. Morini for version 0.4) to be incorporated in VOS. Part of those have been implemented as an experimental set by him, and the technical details are documented in an extensive technical note (RD 8). The approach followed is sometimes slightly maximal, therefore comments by L.Chiappetti, particularly indicating a ranking of priorities, are added (indicated with a bar on the left hand side), leaving however the original text unchanged.

The idea is that each of this function will correspond to a Fortran-callable routine. Were it necessary, these routines may also be non-Fortran (assembler or C), or depart from some of the programming rules imposed below on Fortran programs. The specifications for the VOS interface routines will be public, and the calling sequence shall be kept as simple as possible, to enable anybody to build an own VOS Interface library (the commitment of SDC being only so far to provide a VMS VOS Interface).

For each VOS Interface routine a six character meaningful name is defined, accordingly to standard Fortran 77 naming conventions. However since most compiler (at least under VAX VMS, SunOs UNIX, IBM FORTVS, etc.) support longer names with underscores, we will define a longer format and more readable name for each routine. Long format names for the VOS Interface routines will have the form "aa_xxxxx" or "aa_xxxxx_yyyyy", where "aa" is a prefix (with a number n of characters equal to one or two) identifying the library (in this case "Z"), "xxxxx" is replaced by a verb and "yyyyy" by a noun meaning the routine action. Short format names will be generated deleting all underscores, and abbreviating the verb "xxxxx" to its first four characters (or leaving the verb as it is if shorter) if no noun follows, or to the first three ones if a noun "yyyyy" follows, which is abbreviated to its first (3-n) characters. Long format names will be chosen to produce unique short forms. A preprocessor converting long to short name formats could be easily implemented.

# 4. VD Specifications

The following is the original list of functions (prepared by M. Morini for version 0.4) to be incorporated in VD routines. The approach followed is sometimes slightly maximal, therefore comments by L.Chiappetti, particularly indicating a ranking of priorities, are added (indicated with a bar on the left hand side), leaving however the original text unchanged.

The VD interface for a particular device will not exhaust the capabilities of that device, it will better represent the lowest level of performances for that class of devices. Such interface routine(s) will construct a device-dependent escape sequence or binary command or ASCII command string and send it to the device, generally via the lowest level VOS basic i/o call.

Normally the basic commands will be buffered within each VD interface routine call to obtain greater VOS i/o efficiency. The buffer will be flushed at the end of the execution of each VD interface routine to mask the buffering mechanism to the higher level routines.

All programs accessing a device in a class will generally support all devices in that class which are available at the specific site. The distribution of the call to the selected device in the class will be handled within the corresponding VD interface routine. A database will be maintained at each site with all available devices, their "file" names, class, physical type, description and physical location. This database will be accessed by the VOS Z_OPEN routine when opening a specified device to make known to the system the specific device-dependent control language into which to translate all subsequent VD interface routine calls.

The following is a list of VD functions for various classes of devices. As for VOS functions the specifications for VD interface routine will be public, and the calling sequence will be kept as simple as possible, to enable anyone to build an own VD interface library. VD interface routine will be defined for a virtual terminal device, a virtual graphical device, a virtual pictorial device (including the capabilities of the virtual graphical device), and a virtual array processor device. The commitment of SDC will be so far to provide a DEC VT100 (compatible with VT220 and VT330) virtual terminal interface, a DEC ReGIS Graphics Library (compatible with DEC VT125, VT240 and VT340) virtual graphical device interface (VDIs for Tektronix 401x, HP Graphics Language and PostScript are TBV), a TBD virtual pictorial device interface and, depending on the adopted solution for the hardware of the SCS, a TBD virtual array processor interface. VD interface routine will have names prefixed by "Yx", where x refers to the specific type of defice: T for terminal, G for graphical device, P for pictorial device, A for array processor. See in section 3 above the rules for generating short and long routine names. All parameters in the following VD interface routine lists are of Integer type unless otherwise specified.

It has to be noted, that, at the time these notes were originally written, the underlying concept was that of a central CPU (mini), with a cluster of terminals and other devices (pictorial, plotter etc.). The introduction of workstations providing window management system does not radically alter this view, as, in the XAS usage, programs will still see a "tty window" as a terminal, a "graphics window" as a graphics device, and dedicated windows as pictorial devices. The tools needed to set up the windows (most likely semi-permanent processes detached from XAS, which will capture the output directed to a particular device) are considered not as part of XAS (therefore they may make use of whatewer window management environment, e.g. X-Windows etc., is available on the particular system).

Any user interface making use of menu/button interfaces (usually available under such windowing systems) is not considered at this stage, as it could not be supported on standard terminals (which even in the futurewill be still necessary in some cases, e.g. remote access).

VD interface routine could incur in errors during execution. A general VD interface error routine is defined as follows:

**Y_ERROR**(errcode) **YERRO**

Return a code for the error status of the last-called y-routine. Normally error occurrence causes the routine to have no effect, a positive integer error code is generated and program execution continues at the next statement. A zero value code means correct execution. The user is left responsibility to check errors and take appropriate actions.

Priority rank: essential.

A few routine for the device database management are listed:

**Y_ADD**(name, class, type, description, location) **YADD**

Character*(*) name, class, type, description, location

Add a device to the database. Name is the unique file name with which the device will be identified by the system. Class is "T" for terminals, "G" for graphical devices, "P" for pictorial devices, "A" for array processors. Type identifies the device-specific control language into which the VD interface routine calls must be translated. Description and location are character string giving a description of the device, and its physical location, for user reference.

| | | |
|---|---|---|
| Error codes: | 1 | Incorrect name |
| | 2 | Incorrect class |
| | 3 | Incorrect type |

Priority rank: TBD as the unit database management is a system administration task and need not necessarily be performed within XAS (certainly not by normal users; an exception could be represented by automatic setup of virtual devices in a window environment).

**Y_REMOVE**(name) **YREMO**

Character*(*) name

Remove a device from the database.

| | | |
|---|---|---|
| Error code: | 4 | Non-defined device |

Priority rank: TBD as the unit database management is a system administration task and need not necessarily be performed within XAS .

**Y_LIST**(class, name, new) **YLIST**

| | |
|---|---|
| Character*(*) | class, name |
| Logical | new |

## 4.4 Virtual array processor

The need for an attached array processor was justified at the time of writing RD4 , the "Computer System Requirement for SAX Scientific Data Analysis" (September 1987) by the requirement of floating point capabilities at the 1-2 Mflops level. This could not be achieved at that time by means of minicomputers CPU's of the 200 MLit cost class.

The situation is different when writing this document (February 1989), since complete minicomputers with CPU's capable of more than 10 Mflops performances are now available for less than 200 MLit, as well as microcomputer coprocessors with 1 Mflops performances at 3 MLit cost. The use of a Fortran compiler capable of a vectorization analysis and to generate the appropriate code depending on the available hardware options is also to be considered instead of direct programming of vectorized operations. Fortran compilers of this kind are now available on some machines capable of vectorial operations.

The definition of the virtual array processor device and the specifications of the virtual array processor interface are left TBD at this stage.

## 5. User Interface Specifications

The essential user interface used by XAS application programs corresponds to the approach proposed in section 4.1 of the RD2 document, and is partly based on the interface originally developed for the EXOSAT LE system at ESOC and extensively used in the EXOSAT system in Milano.

Within the XAS monitor (see sections 2 and 8) the user enters commands in response to a XAS monitor prompt which has the form

```
xxx XAS >
```

where xxx is the number (left-filled with points (.) if necessary) of the command in a sequence starting from 1, and a blank follows the > sign. The XAS monitor always prompts at the start of the line following the cursor position on the terminal. Scrolling will be performed if necessary.

In the most general format, a command string has the syntax:

```
command ⊗ commandfile ⊗ parameters ⊗ (options
```

where the command is the name of either a system primitive command, a system procedure, or a program.

A command string may occupy more than one line. The following proposal might be implemented with lower priority if long lines without intervening carriage return cannot be handled by some terminal types. A line to be continued must be terminated by a - sign before the carriage return: the XAS monitor responds with a > prompt followed by a blank at the start of the next line (scrolling if necessary). If this is not ncessary a command line can simply be a TBD number of characters before a carriage return.

The commandfile name must be clearly identifiable, but should be possible to omit it (in which case the command file is the terminal). This means the simplest approach (the command file name being the first positional parameter) is most likely not viable. Other approaches could be to have the command file name appear prefixed by a < sign (UNIX style, although this is NOT Unix i/o redirection !), or attached to the command name with a slash followed by COMMAND= (VMS style), or to set up the command file name with a separate XAS command.

The ⊗ symbol indicates a separator, which is a (non-null) string containing any combination of blanks, (eventually also tabs), and at most one comma. Parameters are separated by a separator.

An option list might follow the parameter list, separated from it by a right parenthesis (there shall be only one right parenthesis in the list). Options are separated by separators as for parameters. Options are in the form [NO|OPTION, and a default value is assumed for each option not explicitely specified. If no option is specified the right parenthesis following the parameter list can be omitted. Options shall be used only by those programs which really need them, and could be implemented in a later time.

The user interface is not case sensitive unless when passing as a parameter a character string enclosed within double quotas (not currently foreseen). Separators may be included in parameter strings delimited by double quotas. A \ character in a command string forces to accept the following character with no special action. The latter two clauses may be included with lower priority.

# 6. Disk Space Management Rules

This proposal is based on the assumption the operating system (like VMS and UNIX) allows a (sub)directory tree arrangement. However the hierarchical file system is not to be considered a must (see also section 2, Introduction).

The first scheme shown here is a representative arrangement for the overall disk storage, which is just of logical nature (that is, for example, not necessarily implying the SAX and XAS disks are directly subordinated to the root directory). The exact implementation implies some form of (strict) coordination among SDC and Local Centres, as an uniform arrangement is desirable.

```
ROOT -+--- SYSTEM  -+---...            System directories with usual
      I             I                  access privilege/restrictions
      I             +---...
      +--- user 1
      I
      .     ......                     User private (full access)
      I
      |--- user n
      I
      +--- non-SAX -+---...            Site-dependent
      I             I
      I             +---...
      I
      +---  XAS   ---+--- CURRENT      Read and Execute world access
      I              I
      I              +--- development  TBD restricted access
      I              I
      I              +--- TABLES       Read world access
      I              I
      I              +--- SCRATCH      Full world access
      I
      +--- CALIB ------- ...           Site dependent
      I
      +--- other SAX                   Site dependent
```

The above scheme refers to SDC and Local Centres. The details of the arrangement of the system area (consistent between SDC and LCs) is TBD.

User areas will be set up locally, together with the privileges of each user. However at least one GUEST account (more at SDC) will be set up with minimal privileges (alternatively temporary accounts for visitors may be set up time to time).

Non-SAX areas are site-dependent (not present at SDC), as well as the arrangement of the CALIB area (intended for calibration analysis work, both "unofficial" or "reserved" software and intermediate results, not for final public calibration results. Other SAX areas are mainly intended at SDC for functions not related with the Interactive Analysis (Mission Planning, Automatic Analysis etc.).

In what follows, consistently with the usage in the remainder of the document, and with the VOS approach described in section 3, the directory hierarchy is indicated with the (Unix-like) notation /dir1/dir2/... Rules to map VOS (system-independent) pathnames to sytstem-dependent (e.g. VMS) pathnames will be part of the definition of the specific VOS Interface.

The /XAS/CURRENT and /XAS/development areas are reserved for program sources and executables. Their respective arrangement in subdirectories is sketched in

the "Program Development Rules" section. Only the Software Librarian will have write/delete access to the CURRENT area. Selected users will have access to the development areas as needed for program development.

The /XAS/TABLES area is intended for permanent data files (tables, calibration results etc.) of common use. Only the Software Librarian will have write/delete access. A detailed description will be given in the future.

The /XAS/SCRATCH area is intended to hold all sort of user data. It is subject to periodic automatic cleanup, like e.g. a daily deletion of all files unused by more than TBD (2 weeks to 3 months according to filetype) or automatic deletion of data belonging to guests after their departure. Its organization is sketched below and referred to in the "Data Structure (Layout) Specifications" and "XAS Command Specification" sections below.

Once again this is just an indicative proposal, and alternative approaches may be considered (moreover this choice is of operative nature, and has no binding implication on the definition of the software architecture). The basic exigence pointed out here is the one of keeping logically separate areas for different kind of data (e.g. the ones pertaining to different targets or pointings of the same targets or experiments etc.). As there is no definite hierarchy , different arrangements may be considered (or actually changed at later stage as experience better suggests).

```
/XAS/SCRATCH -+- user1 -+- target1 -+- period1 -+- exp1 -+- TELEMETRY
              I         I           I           I        I
              I         I           I           I        +- REDUCED.
              I         I           I           I        I
              I         I           I           I        +- PRINT
              I         I           I           .
              I         I           I           +- expn -+- ...
              I         I           .
              I         I           +- periodn -+- ...
              I         .
              I         +- targetn -+- ...
              I         I
              I         +- other
              .
              +- usern -+- ...
```

Only terminal subdirectories are allowed to hold data files (other than lower level subdirectories).

/XAS/SCRATCH/usern will define a default root directory to which all data directories shall refer. Typically usern will be the user name, but could be any name (e.g. to enable data sharing among more users) and the access privileges will be determined at set up.

/XAS/SCRATCH/usern/targetn will define a subdirectory for all data related typically to a particular celestial target.

/XAS/SCRATCH/usern/other is care of the user. Hold any data not foreseen elsewhere (e.g also user temporary commands?).

/XAS/SCRATCH/usern./targetn/periodn will define a subdirectory for all data relevant to a given observing period (FOT)

`/XAS/SCRATCH/usern/targetn/periodn/expn` (where `expn` will be chosen among a limited list like CS, LECS, HPGSPC, PDS, WFC) will define a subdirectory for all data relevant to a given experiment.

The TELEMETRY subdirectory will contain all data files filed from a FOT.

The REDUCED subdirectory will contain all reduced data files created by XAS commands (may be subordinated directly to `periodn` or higher directory?). The PRINT subdirectory will contain output files of XAS commands, log files, etc.

# 7. Data Structure (Layout) Specifications

This chapter is concerned with the identification of the minimum number of data structures which are requested by (and really specific of) an X-ray astronomy analysis environment. The guidelines in the definition of such data structures are the following :

a) the data structures shall be appropriate to X-ray astronomy (i.e. spectra and light curves shall allow carrying along errors and propagating them appropriately).

b) the data structures shall be as natural as possible consistently with the optimal i/o efficiency (e.g. storing an nxm image as a direct access file with m records each one n words long is natural, but may not necessarily be optimum for i/o speed under certain environments).

c) Ockham's razor : data structures non sunt multiplicanda praeter necessitatem. The number of different data structures shall be kept to a minimum (and also the number of layouts for a given data structure shall be kept to a minimum, that is one). Support of "foreign" data structures (e.g. as used by other astronomical packages) is obtained by use of simple format conversion utilities.

The material in this chapter has been originally taken from RD 2, and expanded by L.Chiappetti in version 0.5.

## 7.1 Data flow

Typically data from an X-ray astronomical experiment originate on-board as a multiplicity of data streams. Such data streams can be variously scrambled and rearranged in the various stages (virtual channeling, telemetry packet generation, intermediate storage at Ground Station and OCC) before they are deposited on the final archive medium (FOT or RRD optical disk) and reach the end user. Here they need to be reconstituted in their original nature.

The original, logical, structure (irrespective of any transmission protocol, virtual channel grouping, telemetry frame organization etc.) is depicted in the figure below in a representative example applicable to the case of SAX. In the figure the horizontal axis represents time, the vertical axis different data streams, while thick bars define "observing period boundaries" (i.e. different target pointings) and thin bars "observation boundaries" (i.e. change in the experiment configuration).

ESIS project, and we suggest to suspend any action awaiting such conclusions. Otherwise it is necessary to resort to tricks like an hexadecimal dump packed into 80-byte records (just a factor of 2 space wastage), or to a compressed-ASCII packing (like the one used by ULDA), but this implies both the sending and the receiving end have a copy of the packing/unpacking software.

# 8. XAS Monitor Specifications

## 8.1 Introduction to the XAS Monitor

As already said (cfr. sections 2 and 5) XAS commands are divided in the following categories:

XAS internal commands
XAS external application programs
System commands
XAS procedures, alias (parametric TBD) sequences
System procedures

The first three cases are elementary commands.

**Internal commands** are command strings received by the XAS monitor and executed within it (typically to do something immediate, or to change the value of a memory variable).

**External application programs** are in line of principle any executable module in the system. This includes system utilities, application programs developed for XAS and any (user) application program. See however section 9.2 below for "context level switching" where the XAS monitor can call one program, in turn dispatching the call to other programs.

**System commands** are all commands which may be issued normally to the system interpreter. It is useful to be able to call them from within XAS monitor transparently (with a system-escape facility, i.e. prefixing them with the **SYSTEM** keyword). However equivalent programs for some elementary system functionalities will be defined in XAS (see 9.1) using the VOS Interface routines defined in Section 3; the latter commands should be used within XAS procedures to allow portability.

The other two types of commands are command files combining sequences of elementary commands. As such they are not generally part of the basic XAS system, but are written by user. We are not concerned so far with providing a set of procedures, but just with providing the possibility of using them.

**XAS procedures** will be kept in files (for instance of the type XPR), which are essentially command files for XAS (XAS Monitor will take input from the file instead than from the keyboard, eventually substituting some parameter values, i.e. using them as parametric sentences).

**System procedures** are normal files written in the system command interpreter language (COM files written in DCL for VMS systems; shell scripts under Unix, etc.).

# 9. XAS Command Specifications

The specifications given here are only for external commands. Internal commands, interface to system commands, interface to XAS procedures and system procedures are specified under "XAS Monitor specifications" in section 8.

System commands do not require further specifications. XAS and system procedures are considered second-priority commands, whose building blocks are mainly external commands, and their definition will be mostly left to the final user.

All external commands officially supported within XAS shall use the User Interface specified above. It will however be possible to access any (non-XAS) executable program with the modalities foreseen by such program.

All external commands officially supported within XAS shall use the VOS and VD interfaces specified above in section 3 and 4 and comply to the programming rules specified below in section 12.

A tentative list of individual commands (or of actions which may be assigned to a single command or included as part of other commands) follows (provisional names are included for each command). The final specification shall occur in the form of data sheets as described in section 13.

The best effort is put to ensure that the list be as complete as possible; for each group of commands (corresponding to a subsection heading), and also for individual commands, a priority cathegorization is attempted, indicating which commands are essential (therefore such programs shall be developed in any case), which commands are highly desirable (but could be developed at a later time), which commands are just desirable but optional (therefore they could be developed only if time and manpower allows).

## 9.1 Commands replacing system commands

Within the XAS monitor system commands will be called transparently with the system dependent syntax, making use of the escape-to-system mechanism. This will guarantee access to all system commands to an user on a given system, while, on the other hand, portability of procedures including such system dependent commands is excluded. However a limited set of equivalent programs (corresponding to some essential commands common to all relevant operating systems) will be defined in XAS, based on the VOS Interface routines defined in Section 3, to allow portability of XAS procedures.

Some commands could allow the use of wildcard characters to specify patterns which file names must match (a single asterisk can be substituted by any non-null character string, a single question mark can be substituted by any single character, a backslash forces to accept the following character with no interpretation). The use of wildcard characters in file specifications will be valid when explicitely specified in the description of individual commands.

This last feature is to be considered a design goal and not an essential requirement.

The list of commands in section 9.1 is due to Marco Morini, and is left at the level in which it was written in Version 0.4. A comment by L.Chiappetti indicating the

priority (essential, high, medium, low, very low) is indicated, and noted by a vertical bar on the left hand side.

The entire set of commands in this section could be considered in any case not essential (even if some commands will be designated high priority), as the corresponding functions will anyhow be accessible in a system dependent way via the escape-to-system facility embedded in the XAS monitor from the very beginning.

**APPEND** `input-file output-file`

Add the content of a specified input file to the end of the specified output file. Wildcard characters are not allowed in the input file specification. When an asterisk wildcard character is used in any field of the output file specification, the APPEND command uses the corresponding field from the input file. If the input file does not exist the output file is unchanged. If the output file does not exist a copy of the input file is created. If neither the input nor the output file exist, an empty file with the output file name is created. Defaults for the input and output files are the files associated with the 'stin' and 'stout' global variables, respectively (input from the terminal is terminated at the occurrence of a <ctrl>Z character - ASCII 26).

This command (to be used essentially with ASCII files, therefore not operating on the primary data structures) could be considered medium or low priority.

**CANCEL** `job` **(queue**

Remove a job from a printing queue. The default for the job specification is the last enqueued job by the current process. If the job is printing, print is aborted. If the queue is not specified, the default print queue is assumed.

This function is of administrative nature, and can be quite dependent on the way spooling is implemented under the different operating systems (therefore it is best left to escape-to-system mechanism). It could be considered as having a very low priority.

**CHDIR** `dirname`

Establish a directory as the default directory for file specifications. The default for `dirname` is the current default directory.

Note that the default directory (or working directory in Unix terminology) will not be used by the majority of XAS analysis commands (which will refer implicitly to pathnames assumed by the disk space management criteria described in section 6, and selected via the commands described in section 9.2 below). Therefore this command (of very simple implementation) is considered of medium to high priority only.

**COPY** `input-file output-file`

Copy a group of input files into a group of output files. Wildcard characters can be used in the input file specification to copy more than one file. At least one field must be specified in the output file specification. If the directory is not specified the current directory (for non-XAS files; or a directory determined by context - cfr. 9.2 below - for XAS data files) is used. Other missing fields are replaced with the corresponding fields of the input file. Alternatively an asterisk wildcard character can be used instead of the missing field in the output file specification. Defaults for the input and output files (in the case of non-XAS files) are the files associated with the 'stin' and 'stout' global

# 11. XAS Distribution Policy

This section can be considered partially decoupled from the rest of the document, as it concerns the policy for the distribution of the XAS software. A proposal is made here for a fully open distribution, however more restrictive solutions may be considered, and the eventual decision will have no impact on the software architectural choices. However public distribution may have an impact on the way the software is developed and documented (in the sense it may pose further requirements of clarity and good documentation : but any solution, even if for use only at SAX sites, will anyhow require a large degree of documentation work).

The following classes of sites entitled to receive and use XAS have been identified in order of priority:

a at the SAX Scientific Data Centre (SDC)
b at the four SAX Local Centres in Italy (LCs/I)
c at the Dutch SAX Local Centres (DLC)
d at other sites associated to SAX in Italy and Holland (universities etc.)
e at any other site in the astronomical community interested in SAX data analysis or in applying XAS to other missions

The degree of conformity of XAS to the official maintained version (see below) at the five classes of sites will be as follows:

a/b there will be absolute conformity between class a and b sites, which together make up the SAX Computer System (SCS), that is the identical version of XAS shall run without any change at SDC and LCs/I. This means all software modules running at SDC will be able to run at any LC; even those which are not normally required outside SDC (e,g. some archive maintenance or calibration module) shall be available and able to run at an LC for backup purposes or if the need for it arises for any reason). The conformity will be ensured by:

b1 identical hardware being present at SDC and LCs, namely CPUs of the same manufacturer (upgrades to upward-compatible models is allowed) and an identical set of officially supported peripherals. Acquisition of other makes of peripherals (either as the case it is officially approved for testing) and development of related interfaces will be possible under class d/e rules;

b2 the same version of the same operating system;

b3 an identical set-up, including system generation, disk space handling conventions (rules for file and directory naming etc ), data management procedures. (Site-dependent extensions will be allowed as far as they do not conflict with the common kernel setup)

b4 identical XAS software modules being available in source, relocatable and executable form. Distribution of updates (via remote file copy or download) shall be automatic.

b/c It is highly desirable that the DLCs (at Utrecht and ESTEC ?) have the same standing of the LCs/I within the SCS. This implies acquisition of compatible hardware and submission to all other class b rules. If for any reason that is not possible, at least class d rules shall apply.

d/b Here we consider mainly university institutes in Italy where members of the SAX Consortium are present (IOA, Milano, Palermo, Ferrara). Then other institutes where people involved in the Observational Program Working Group (OPWG) and/or in the Core Program shold be present could be considered. It is not clear the situation for other Dutch institutes with respect to Utrecht and SSD, nor the possible position of MPE.

These sites are considered as far as they are already equipped with hardware compatible with the current DEC/VAX choice. As they are not integral part of the SCS, the class b rules may not be made binding for them. It is their care to arrange (rule b1) to be conformant to CPU and peripheral compatibility as closely as possible, to upgrade the operating system when necessary (rule b2) and to set up (as subset of) their system according to rule b3, or to take responsibility in applying any necessary modification in the interfaces. Concerning XAS software distribution, they will have at least class e privileges, but possibly receive automatically a notification of any software updates via network, after which they can decide to issue a request to receive/copy/upload the updated module (all handled via network).

In a word they will share all rights of class b sites, which are compatible with sharing none of the obligation of class b sites.

e any other site will be assumed as non-conformant. In the most favourable case such sites will be in a situation similar to class d sites (having quasi-compatibility at hardware level) but in line of principle they can be assumed to be equipped with any brand of hardware.

Most of XAS software will be available for public distribution, however we reserve the right not to disclose software modules of specific SCS usage.

Software will be distributed to class e sites in source format only, as any adaptation before re-linking (even for hardware and operating system compatible sites) is their responsibility.

The fact that XAS will be layered upon a Virtual Operating System (VOS) and use a set of Virtual Device (VDI) interface libraries to handle peripherals will make the conversion to different machines or operating system easier.

However such conversion, i.e. the writing of system specific VOS and VD interfaces, and any other system-dependent change, will be exclusive responsibility of the receiving site. All necessary documentation (user's and programmer's) will be made available.

No further support will be given by SDC and/or LCs, nor by their sponsoring organizations. No guarantee will be made by the same that the software is error-free or the algorithm are accurate. Software will be dispatched to requesting sites under a disclaimer (as it is common usage for public domain software developed in academic environments) such as:

"The XAS software ... was developed under sponsorship of ASI. Neither the ... (list of organizations or institutes) ... make any warranty, express or implied, or assume any liability or responsibility for accuracy, completeness or usefulness of any XAS software module ... etc."

In particular while the SAX Consortium retains the right to use and disseminate XAS for any purpose whatsoever, it is requested that any site receiving XAS does not further distribute it. This could even be formalized as a copyright (as an example, one may consider the policies for the distribution of MIDAS and/or IRAF). It is also requested that any substantial usage of XAS to produce scientific results be acknowledged, as well as any re-use of substantial parts of the XAS code for other purposes.

The first distribution of XAS to a requesting site will be made via magnetic tape or similar medium (possibly supplied by requestor, or at a nominal charge). Two distribution formats may be devised (one being VMS Backup for VAX sites, and the other one a general purpose format for other sites, e.g. similar to the FOT format with a tape directory and fixed-blocked source files).

Further updates will in line of principle be distributed via network. An exception is made in case of complete new releases, which will be delivered again on tape. Distribution of updates will not be automatic. Update notices will be sent to all sites registered on a distribution list, however only sites explicitly requesting a particular update will receive it. A log of updates sent will be kept at SDC together with the register file.

All distribution of XAS shall be handled by SDC, through a semiautomatic system. Requests sent to LCs shall be routed to SDC. The software tools necessary at SDC to handle the distribution are not of concern of the present document, and the relevant requirements are contained in RD 6. In fact the distribution of any data or software related to SAX is part of the Mission Support functions of SDC. XAS software is just another "item" which is distributed within the Mission Support framework (even if it is not developed within such framework). The purpose of this section is just to give the rules under which XAS, the object of the present document, will be distributed by SDC using the Mission Support tools.

## 12. Program Development in XAS

Program development of XAS software shall essentially occur at SDC, where a dedicated team shall reside. Contributions developed at other sites shall be inserted in the official XAS software care of the staff at SDC.

The following way of working is foreseen at SDC. It is important that all scientific members in the team (that is, the data analysis experts and the experiment support scientists) share a good knowledge of the entire XAS system. It shall be avoided the fact that a single person knows the details of a particular program or program group (this is a potential single point failure). At the same time all modules should be integrated in an homogeneous system. For this purpose, even if a particular person will be in charge of an individual program, or subroutine, or aspect (e.g. fitting, or timing, or WFC data reduction, or PDS data reduction), knowledge about all other modules shall be shared by all. For this purpose it is planned to held get-together meetings of the team (at least with weekly frequency), where the details of programs in development will be presented and discussed, and any change to the general rules are agreed.

These meetings could be attended also by system programming people, which should therefore get an idea of the overall purpose of XAS.

Two other roles are important in the XAS s/w handling. The first one is that of the coordinator (and its deputy), whose primary task will be that of inspecting any piece of code (program, subroutine or procedure) newly developed or updated, and verify the compliance with the general rule. The coordinators will have the authority to reject non-compliant software and have it modified by the authors. They will also supervise the XAS documentation in the same way.

Once a piece of software has been approved, it will be passed to the software librarian, which will insert it into the system, and assign version numbers. He will also keep track of any modification and of the overall history of the system. He will represent the interface between the developers of XAS and the Mission Support components of SDC.

The production of the XAS documentation in the final typesetting, the distribution of the documentation, the distribution of the XAS updates to Local Centres and other sites will be part of the SDC Mission Support activities (see also section 11).

### 7000-7999 miscellanea

Ru: 7000 Never assume a value for uninitialized variables.

Re: 7004 All files foreseen for lineprinter output shall start in column 2. Column 1 shall be left blank (except for 1's in column 1 to mark page throws). Other carriage control codes and in particular overstrike codes are forbidden.

Ru: 7008 Unreachable code, unreferenced variables and unused labels shall be avoided.

# 13. Rules for XAS Documentation

Two kind of essential documentation has to be provided: Users' Documentation and Programmers' Documentation. Both of these are for public release. Further internal documentation may be necessary (e.g.history of updates). The different types of documentation are briefly described below, together with the rules for the generation and maintenance of documentation.

Documentation authors are expected to supply plain ASCII files, plus instruction for typesetting. Final typesetting should be handled centrally at SDC. Also the distribution of the documentation shall be handled by the SDC Mission Support components.

## 13.1 Users' Documentation

The Users' Documentation will comprise on-line documentation (help files) and an Users' Manual.

### 13.1.1 On-line documentation

On-line documentation is a tricky business, as the way it is implemented under different systems may be quite different. The proposal here is not to develop a dedicated system-independent HELP facility under the VOS, but to have an HELP command within XAS calling the system dependent HELP facility. The source of the help files will be kept in a series of plain ASCII files, and a system-dependent utility will "compile" such help file sources in the format necessary to the particular system.

The consideration above follow from the fact different systems use radically different approaches (see discussion below), while on the other hand the generation of a system independent HELP cannot be regarded as a high priority task.

Under different systems the HELP facility is handled in a variety of ways :

A simple non-hierarchical one, in which ASCII help files for individual commands are collectged are indexed by keywords into a special compiled file. The HELP command will locate the help information for the specific command, and display it to the terminal as typed.

A case with very limited hierarchy, but with some formatting control, is given by the Unix man command, which uses dedicated help files residing in special directories, an external page formatting utility (troff) and some terminal control to allow paging through the file (stop at page full, and display one page at a time).

More hierarchy is available under the VMS topic-subtopic-subsubtopic tree arrangement. Full screen capabilities are however not fully exploited.

A different kind of hierarchy is obtained (for instance with the IBM CMS HELP) using menu files (which are displayed in full screen to some particular terminals), where individual help entries can be pointed, and selected for display. The structure can be nested at more levels. Some page formatting control shall be provided in the file. It is possible to page through the help file (or view selected sections), using functional keys and full screen control.

Another type of user interface, allowing the text of the help file to appear in a separate window is sometimes used by dedicated applications running in a window environment, but its usage is not widespread.