MOPS – A DATA STRUCTURE MANAGEMENT SYSTEM FOR THE LEP/SPS CONTROL SYSTEM

W. Herr
SPS Division, CERN 1211 Geneva 23, Switzerland

Abstract

The MOPS data structure management system has been developed for the LEP/SPS control system and is designed to support the programmers in organising their data in a structured fashion. MOPS data structures can be accessed from FORTRAN 77 as well as from "C" programs and therefore allow the easy exchange of data between programs written in different languages. The data structure contains a full description of all data elements and their logical relations and simplifies the transfer across networks with different computer systems and data representations. A description of the data structure and the basic philosophy are presented.

INTRODUCTION

The design and operation of an accelerator requires a large set of programs with substantially different structure and functionality (e.g. simulations, calculation of Twiss parameters, closed orbit corrections, beam diagnostics, etc.) and which are often written in different languages. The portability of the programs as well as the portability of the data becomes a very important issue and the integration of these programs into a coherent system requires a standard procedure for the exchange of data between the program modules. The standard high level programming languages (e.g. FORTRAN 77, C or PASCAL) provide very few tools and the data organisation is entirely left to the programmer. The data may be passed through many modules of a program and its structure must be designed with great care. In addition, the memory space and the data structures are allocated at compilation time resulting in a static data organisation. For the analysis of high energy physics experiments several data management systems have been designed [1, 2] and are widely used with FORTRAN 77.

In order to achieve an efficient and fast communication and to minimise the system resources needed we have studied the requirements for a standard data organisation for the SPS control system and we have defined the following properties for a new data structure:

- The data must be portable between different high – level languages (e.g. "C" and FORTRAN 77).
- It should be possible to mix data types in the data structure (e.g. integer and floating point numbers, strings and characters).
- Data objects should be accessible using keywords rather than through variable names or pointers.
- The data storage should be efficient, i.e. binary.
- The data transfer across heterogeneous networks should be simplified by the use of the data structure.
- The implementation must be portable to a large range of computers from mainframes to VME systems (OS9).
- It must be easy to use also for inexperienced programmers.
- It should simplify the archival of data on mass storage devices.

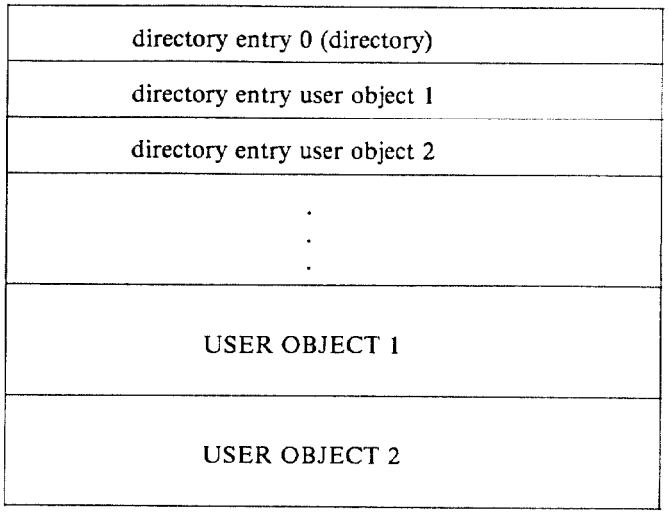
Based on these requirements a data structure management system MOPS (Multiple Object Partitioned Structure) has been designed which introduces concepts and conventions and provides utilities to express complicated data structures in a very simple way. All programs which conform to this standard automatically become modules of a common framework for software in accelerator physics. At CERN MOPS data structures are used in e.g. optics calculation, orbit correction, tracking and programs used in accelerator control systems [3].

BASIC CONCEPTS

The physical storage: The physical location of the data structure I shall call the "dynamic store" which is a contiguous part of the memory and it can be allocated either at compilation time or at execution time if this is permitted by the operating system. In FORTRAN 77 the dynamic store may be allocated by defining an array which can be of any type. A MOPS data structure is mapped onto this dynamic store. The implementation allows the simultaneous use of several such data structures stored in different dynamic stores. from the same program.

The basic unit of the data structure is an entity I shall call an "object" which contains a number of data elements grouped together according to logical affinity (e.g. β functions) and stored in *contiguous* storage words in the dynamic store. A data element can be a single data item (floating point or integer number etc.), a "C" structure or a collection of data items like a matrix. In particular, another MOPS data structure can be stored as an object in a data structure.

The data structure is arranged as a partitioned data set with a "directory" at the beginning, which may be considered as a system object, followed by one or more user objects which are described in the directory. A simple example for such a data structure is illustrated in Fig.1.



CH2669-0/89/0000-1713$01.00©1989 IEEE

The figure shows a data structure with two user objects and a directory in front of them. The directory contains all information necessary to access the user objects and in this example the directory has three entries, one for each user object and one for the directory itself. The service routines and tools provided are used to manipulate and access the different data objects and the directory.

The directory: In the memory the directory appears as an array of structures at the beginning of the data structure. A complete description of the data contained in the structure is held in the directory. For every object an entry in this directory is created by the MOPS routines using a specification provided by the user as parameters to the routines. The information stored in the directory for one object is:

- Data pointer, i.e. offset to the user object in bytes (calculated and filled in by the system).
- Keyword or object name which can be up to 80 characters.
- Element type (valid types are all standard FORTRAN 77 and "C" data types, and more complex data types such as "structures").
- Element code (an integer number, derived from element type by the system).
- Number of elements for this object (e.g. number of integer variables etc.).
- Size of one element in bytes (e.g. 4 for "INTEGER" or "REAL*4"). For certain "C" data types this depends on the computer.
- Timestamps for the creation and last write of the data structure.

The directory is normally invisible and not directly accessible to the user and can only be manipulated with MOPS system calls.

User objects: All the elements grouped together in one object must be of the same type, i.e. they must be all integer numbers, floating point variables or structures of the same type. The different user objects can hold data of different types. A special exception is when an object holds another MOPS data structure which itself can contain several objects with different types. The data elements are stored in a way which is independent of the language, for example character strings are always stored in the same way, no matter whether the application is written in "C" or FORTRAN 77 and on which computer it is run (the representation of strings in FORTRAN 77 strongly depends on the compiler used).

Apart from these concepts the way of accessing data via a keyword rather than through variable or array names was one of the basic aims to simplify its use and the portability of the data between different high-level languages. In all MOPS routines the objects are referred to with these keywords. This causes an inevitable (but small) overhead at execution time but any program written using this data structures is easy to understand and to modify without having to care about side effects of the changes and without having to recompile the entire program or a group of programs. Only when the definition of this object (i.e. the keyword) is changed a recompilation is necessary.

FUNCTIONALITY OF MOPS

Before I give a short overview how to work with the data structure I would like to point out what the system provides to the user.

- All user data is accessible using keywords rather than variable names.
- Data can be entered into a data structure or read back from it.
- Manipulate objects at runtime: objects can be changed in size, they can be dropped from the data structure and new objects can be added at any time. This makes the data organisation dynamic.
- The data structure can be handled as an entity, e.g. for input and output on an external medium.
- Attributes of user objects can be made available at runtime to the user program, e.g. number of data elements etc.
- Logical relationships can be set up between objects.
- Since the data structure itself contains a complete description of the data items it allows a transfer accross networks.
- Facilities exist for debugging the data structure and the programs.

MOPS USER INTERFACE

MOPS data structures can be accessed from FORTRAN 77 and "C" programs and the corresponding packages of the service routines are embedded in these languages, i.e. they are FORTRAN and "C" callable subroutines and are written mostly in standard FORTRAN and "C". Non-standard features of the languages have been avoided wherever possible. In this chapter I shall briefly outline the manner in which MOPS data structures are used. In a few places I shall illustrate the usage by giving the calling sequence for FORTRAN 77 as an example.

Memory management: The dynamic store must be provided by the programmer and it can be defined as a FORTRAN or "C" array or it can be allocated with the MOPS storage allocator where this is possible [4]. On UNIX SYSTEM V operating systems the dynamic store can be a shared memory segment to allow the concurrent access to the data structure from several separate processes. A protection scheme with semaphores is used to avoid contention when several processes have to write into the same data structure. The MOPS support routines can handle an arbitrary number of data structures mapped into different dynamic stores and the start address of the dynamic store (array name in FORTRAN) is passed as an argument to the routines.

Before the data structure can be used it must be *initialised.* This sets up the directory structure and reserves the necessary space. The FORTRAN calling sequence looks like:

```
REAL*4     Q(100000)
CALL SDINI('TWISS', 25, Q)
```

The name of the data structure, the maximum number of user objects and the address of the dynamic store must be provided as arguments. Time stamps associated with the creation of the data structure are filled in during this initialisation process.

After the data structure has been initialised user objects can be *booked* within the data structure with a call to the appropriate booking routine [4]. The information which must be provided are:

- Number of data elements.
- Type of the data elements, e.g. CHARACTER, REAL*8, INTEGER, float etc.
- Keyword or object name for this user object.

The booking routine will reserve the necessary space for the user object in the data structure, fill the directory entry for this object and calculate the pointer to the beginning of the data. For example the call

```
CALL SDBOOK(216, 'BETA-HORIZ', 'REAL*4', Q)
```

would reserve space for an object called 'BETA-HORIZ' with 216 floating point numbers. After a user object has been booked, the objects can be manipulated using the object name and data can be entered into or read back from the object using the appropriate MOPS routines.

Input/output of data structures: One of the most important features of the MOPS data structure management system is that utilities are provided to handle the transfer of a data structure as an entity to and from an external medium such as a disk or tape. The user does not need to call any (usually system dependent) I/O statement explicitly. These utilities maintain the validity and integrity of the data structure during this transfer. To write the entire data structure from "Q" into the file "TWISS.DATA" the call

```
CALL SDFIL('TWISS.DATA', Q)
```

would be used in FORTRAN 77. A similar call would read the data structure back into the dynamic store from a file. The representation of the MOPS data structures on the external medium is usually sequential and can be either strictly binary or in a special MOPS ASCII format. Utilities exist for the transformation between the two formats. In heterogeneous networks where computers have different number representations this ASCII format can be used to transfer an entire data structure from one computer to another and therefore allows the exchange of data between a large variety of computers. Since all type information, i.e. the I/O characteristic, is available in the data structure all data items are generated in the proper format. For standard FORTRAN 77 or "C" types the I/O characteristic is generated at the time when the objects are booked. Data types which are not standard FORTRAN 77 or "C" data types and complicated user defined types such as "C" structures must be indicated to the MOPS system by the programmer. This is performed by a routine which carries out a lexical analysis on a format descriptor, supplied by the user in a fashion similar to [1]. This allows the definition of complicated structures and types by the user.

For the fast transfer across networks we have developed a standard exchange format [5] which allows the binary transfer between computers with incompatible architectures. The I/O routines to handle this format make the proper conversions to and from this exchange format. This has been installed for the use in the SPS control system and works on all UNIX installations which run standard TCP/IP.

REFERENCES

[1] R. Brun et al; "ZEBRA user guide", CERN/DD/EE/85-06 (1986).

[2] J. Zoll et al; "HYDRA topical manuals" (1981-1984).

[3] Many different applications, for example:
W. Herr et al. "A new Closed Orbit Correction Procedure for the CERN SPS"; CERN/SPS/88-12 (AMS) 1988.
K. Cornelis et al. Multicycling at the CERN SPS - Supercycle generation and first Experience with this Mode of Operation"; CERN/SPS/88-18 (AMS) 1988.
W. Herr et al. "Application Software for the SPS Master Timing Generator"; CERN/SPS/AMS/Note 87-06 (1987).

[4] W. Herr, "MOPS User Guide for C programs"; CERN/SPS/88-43 (AMS) (1988).
W. Herr and R.Schmidt, "MOPS-Fortran 77 User Guide"; CERN/SPS/88-44 (AMS) 1988.

[5] W. Herr and G. Morpurgo, "Binary Transfer of MOPS Data Structures "; CERN/SPS/ACC/89-01 (1989).