PS/HP/Note 97-01
17 January, 1997

PSTOOL

a High Level Interface to CERN/PS Accelerator Control System

Reference Manual

Version 1.0

András Ster

Foreword

PSTOOL is a Fortran callable program interface to the CERN/PS Accelerator Control System. It is based on the existing PS Equipment Access (EQP) and Pulse to Pulse Modulation (PPM) libraries, written in C++ language, which contain the equipment access and event synchronization functions, respectively. The package provides simplified high level procedures for accelerator control operations described in this manual. Due to the programming language, application programs can use it together with CERNLIB library facilities, like advanced graphics functions, as well.

Table of Contents

Preliminary remarks

Apart from the references listed in this manual further information concerning the PS Accelerator Control System are available on the WWW under the divisional home page.

This package is available from the PS Program Library. When compiling application programs on the PS platform IBM AIX/6000 use the compiler option **-qextname** to be compatible with this and other libraries.

In this document Fortran program code segments appear in `Courier` typographical form. Where a routine or term is defined its name is in **bold**.

1. Introduction

1.1 Motivation

Years ago CERN/PS developed a special language, NODAL, with the intention of creating a high level programmation tool and environment for equipment control. Nowadays, except for special cases, its use is less preferred and several efforts have been made to replace it with more advanced methods. The present idea was conceived on the basis of the Linac 2 performance optimization initiative [1]. Accelerator physicists have required a program that is flexible enough to be quickly adapted to the continually changing conditions and other supported software packages can be used with it. Hence the choice of the Fortran program development environment [2]. The first version of the interface that provides some simple Fortran callable routines for basic accelerator controls has been implemented.

1.2 Conventions

1.2.1 Terms

The term **equipment** in this manual means any accelerator element that is controlled via the PS Equipment Access Library [3] of the PS Accelerator Control System.

The event synchronization is controlled through the Program Line Sequencer (PLS) of the Pulse to Pulse Modulation Library [4]. The frequently used term **PLS line** incorporates all the conditions that define a particular beam cycle.

1.2.2 Parameter types

The parameter types of the PSTOOL routines obey the standard Fortran rules, i.e., if not stated explicitly they are `INTEGER` if the initial letter is between I and N inclusively and `REAL` otherwise. Character variables are always indicated by the key `STRING` that has a variable length unless it is indicated explicitly. The length can not exceed 80 characters.

1.2.3 Parameter identifiers

In the context the parameter MACHINE identifies an accelerator line and it may have one of the three predefined values that follow the CERN/PS control accelerator number convention:

0 :	CPS	(Cern Proton Synchrotron)
1 :	PSB	(Proton Synchrotron Booster)
2 :	LPI	(Lep Injector Linacs)

The argument EQP_PROPERTY identifies an equipment parameter. Consult the WWW pages for more details about properties. In this library 4 different parameter type codes have been implemented according to the following cast:

0 :	AQN	(1st acquisition value)
1 :	AQN1	(2nd acquisition value)
2 :	CCV	(1st current control value)
3 :	CCV1	(2nd current control value)

In case of radio frequency the 1st parameters refer to the phase and the 2nd ones refer to the power supply amplitude. When quadrupoles are tuned the power supplies are controlled via the 1st parameters. To measure beam intensities the beam transformers must be read with the option AQN1.

1.3 Remarks

In case of errors the routines automatically send messages to the screen with a brief explanation of the problem occurred. The present error function implemented here returns a code that can be either -1 if fatal or 1 if non-fatal error occurs and 0 if there is no problem.

2. Control routines

2.1 PLSLNAME Return the line names in a PLS group

```
FUNCTION PLSLNAME(MACHINE, GROUP_NAME, LINE_NAMES, LINES_NUM)
```

Action:

This function returns the line names of the PLS group for the requested machine. The function value returns the index of the first name in the string array. The number of line names in the name array is also returned. The index of a name is the line number.

Input parameters:

MACHINE	Machine identifier
GROUP_NAME	PLS group name (STRING)

Output parameters:

LINE_NAMES	Array of PLS line names (STRING)
LINES_NUM	Number of line names
PLSLNAME	Line number of the 1st name in the string array (INTEGER)

Note:

If any problem occurs the function returns 0.

Example:

```
N = 0
I1 = PLSLNAME(PSB, "USER", LINE_NAMES, N)
DO I = I1, I1 + N - 1
  PRINT *, I, LINE_NAMES(I)
ENDDO
```

2.2 PLSUSER Return the number of the current active PLS user line

```
FUNCTION PLSUSER(MACHINE, CYCLE, SUPER)
```

Action:

This function returns the line number of the current active PLS user line for the requested machine. The function returns the current cycle number in the supercycle and the current supercycle number, too.

Input parameter:

MACHINE	Machine identifier

Output parameters:

CYCLE	The current cycle number in supercycle (INTEGER)
SUPER	The current supercycle number (INTEGER)
PLSUSER	The current user line number (INTEGER)

Note:

If any problem occurs the function returns 0.

Example:

```
CYCLE = 0
SUPER = 0
LINE_ACTIVE = PLSUSER(1, CYCLE, SUPER) !Current active line in PSB
```

2.3 PLSUSERS **Return the names of the current active user lines**

```
FUNCTION PLSUSERS(MACHINE, LINE_NAMES)
```

Action:

This function returns the names of the current active user lines for the requested machine. The routine returns all the lines either in waiting or in operating state which always depend on the current user requests.

Input parameters:

MACHINE	Machine identifier

Output parameters:

LINE_NAMES	Array of PLS line names (STRING)
PLSUSERS	Number of names returned (INTEGER)

Note:

If any problem occurs the function returns 0.

Example:

```
N = PLSUSERS(1, CYCLE_NAMES)
DO I = 1, N
  PRINT *, I, CYCLE_NAMES(I)
ENDDO
```

2.4 XEVTCB **Initializes-terminates event processing**

```
SUBROUTINE XEVTCB(ID, MACHINE, PULSE, PLSLINE, CB)
```

Action:

This routine initializes or clears event-related internal variables. Its name and form comes from the corresponding NODAL function for traditional reasons only and it does not have any relation with Xwindows at all. The routine sets the current PLS line as default for the requested machine and pulse if initialization is requested. After termination request the PLS line condition becomes undefined. A pulse value equal to 0 allows for PLS line events. It redefines the CTRL-C keyboard event handling allowing to test this button action.

Input parameters:

`ID`	Process identifier
`MACHINE`	Machine identifier
`PULSE`	Pulse number `(INTEGER)`
`PLSLINE`	PLS line number `(INTEGER)`
`CB`	Command flag. 0 value means initialization, non-zero value means termination of event processing `(INTEGER)`

Note:

`ID` can be any identifier number. Don't use any pulse number other then 0 in the current version. A value of -1 for plsline ignores PLS line condition and a default line is set.

Example:

```
MACHINE = PSB
CALL XEVTCB(1, MACHINE, 0, -1, 0)    !Init and Ignore PLS line
```

2.5 `RFLIN` Read a Linac2 radio frequency parameter

```
FUNCTION RFLIN(EQP_NUMBER, EQP_PROPERTY, PLSLINE, COCO)
```

Action:

This function reads the corresponding Linac2 radio frequency parameter defined by the equipment, the property and the PLS line. It also returns the current error code.

Input parameters:

`EQP_NUMBER`	Equipment identifier `(INTEGER)`
`EQP_PROPERTY`	Equipment parameter type `(INTEGER)`
`PLSLINE`	PLS line number `(INTEGER)`

Output parameters:

COCO	Function completion code (INTEGER)
RFLIN	Current equipment RF parameter value

Note:

To get the equipment number call the routine EQPDATA with the standard name of the equipment. The controllable parameters are those of the radio frequency phase (1) and power supply (2).

Example:

```
!Read buncher1 phase of Linac2 MEBT

PLSLINE = NUM_OF_SFTPRO
PHASE1 = RFLIN(2002, 0, PLSLINE, COCO)
```

2.6 RFLINS **Set a Linac2 radio frequency parameter**

```
SUBROUTINE RFLINS(EQP_NUMBER, EQP_PROPERTY, PLSLINE, COCO, PARAM)
```

Action:

This routine sets the corresponding Linac2 radio frequency parameter defined by the equipment, the property and the PLS line. It also returns the current error code.

Input parameters:

EQP_NUMBER	Equipment identifier (INTEGER)
EQP_PROPERTY	Equipment parameter type (INTEGER)
PLSLINE	PLS line number (INTEGER)
PARAM	New equipment RF parameter value

Output parameter:

COCO	Function completion code (INTEGER)

Note:

To get the equipment number call the routine EQPDATA with the standard name of the equipment. The controllable parameters are those of the radio frequency phase (1) and power supply (2).

Example:

```
!Set new buncher1 phase in MEBT of Linac2

PHASE1 = RFLIN(ID_BUNCH1, CCV, PLSLINE, COCO)
CALL RFLINS(ID_BUNCH1, CCV, PLSLINE, COCO, PHASE1 * 1.05)
```

2.7 POW Read an equipment power supply parameter

```
FUNCTION POW(EQP_NUMBER, EQP_PROPERTY, PLSLINE, COCO)
```

Action:

This function reads the corresponding equipment power supply parameter defined by the equipment, the property and the PLS line. It also returns the current error code.

Input parameters:

EQP_NUMBER	Equipment identifier (INTEGER)
EQP_PROPERTY	Equipment parameter type (INTEGER)
PLSLINE	PLS line number (INTEGER)

Output parameters:

COCO	Function completion code (INTEGER)
POW	Current equipment power supply parameter value

Note:

Call the routine EQPDATA with the standard name of the equipment to get the EQP_NUMBER identifier. Power supplies are controlled via the 1st equipment parameter.

Example:

```
!Read 1st QUAD power in Linac2

CALL EQPDATA("LA1.QFN02", EQM_NAME, EQM_NUMBER, LA1_QFN02, S, L)
POWER1 = POW(LA1_QFN02, 0, PLSLINE, COCO)
```

2.8 POWS Set an equipment power supply parameter

```
SUBROUTINE POWS(EQP_NUMBER, EQP_PROPERTY, PLSLINE, COCO, PARAM)
```

Action:

This routine sets the corresponding equipment power supply parameter defined by the equipment, the property and the PLS line. It also returns the current error code.

Input parameters:

EQP_NUMBER	Equipment identifier (INTEGER)
EQP_PROPERTY	Equipment parameter type (INTEGER)
PLSLINE	PLS line number (INTEGER)
PARAM	New equipment power supply parameter value

Output parameter:

COCO	Function completion code (INTEGER)

Note:

Call the routine EQPDATA with the standard name of the equipment to get the EQP_NUMBER identifier. Power supplies are controlled via the 1st equipment parameter.

Example:

```
!Set 1st QUAD power in Linac2

CALL EQPDATA("LA1.QFN02", EQM_NAME, EQM_NUMBER, LA1_QFN02, S, L)
POWER1 = POW(LA1_QFN02, CCV, PLSLINE, COCO)
CALL POWS(LA1_QFN02, CCV, PLSLINE, COCO, POWER1 * 1.05)
```

2.9 TRAFO **Read a beam intensity transformer**

```
FUNCTION TRAFO(EQP_NUMBER, EQP_PROPERTY, PLSLINE, COCO)
```

Action:

This function reads the beam intensity value from a beam transformer according to the specified equipment, property and PLS line. It also returns the current error code.

Input parameters:

EQP_NUMBER	Equipment identifier (INTEGER)
EQP_PROPERTY	Equipment parameter type (INTEGER)
PLSLINE	PLS line number (INTEGER)

Output parameters:

COCO	Function completion code (INTEGER)
TRAFO	Current beam intensity

Note:

Call the routine EQPDATA with the standard name of the equipment to get the EQP_NUMBER identifier. Beam intensities are expressed in mA and are transferred through the 2nd equipment parameter.

Example:

```
!Beam intensity at the entrance of Linac2
```

```
INTEN1 = TRAFO(2002, AQN1, PLSLINE, COCO)     ! Read TRA06
```

2.10 EQPDATA **Return information about an equipment**

```
SUBROUTINE EQPDATA(EQP_NAME, EQM_NAME, EQM_NUMBER, EQP_NUMBER, S, L)
```

Action:

This routine returns information about the equipment defined by its standard name. It provides practical equipment identifiers for later use by the other interface routines. The last two parameters are auxiliary ones and are intended to be filled in with extra information for future version only.

Input parameter:

EQP_NAME	Equipment name (STRING)

Output parameters:

EQM_NAME	Equipment module name (STRING)
EQM_NUMBER	Equipment module number (INTEGER)
EQP_NUMBER	Equipment number (INTEGER)
S	Equipment special info (STRING)
L	Equipment special info (INTEGER)

Note:

An equipment name is a character string according to the rules and conventions of the accelerator control system (see an example below). Currently the only meaningful parameter is EQP_NUMBER, however the parameters EQM_NAME and EQM_NUMBER are also correctly returned but only experts can make use of them.

Example:

```
!Get the identifier of 1st Linac2 quadrupole and pass it further

CALL EQPDATA("LA1.QFN02", EQM_NAME, EQM_NUMBER, LA1_QFN02, S, L)
PLSLINE = MEAPSB
AQN_POWER1 = POW(LA1_QFN02, AQN, PLSLINE, COCO)
```

2.11 ERROR **Return the latest error code**

```
FUNTION ERROR()
```

Action:

This function returns the code of the latest error that occurred during a call to an interface routine. Currently it indicates fatal or non-fatal errors by the codes -1 and 1, respectively. It returns zero if there is no problem.

Output parameter:

`ERROR`	Latest error code `(INTEGER)`

Example:

```
IF(ERROR() .EQ. -1) STOP
```

3. Utilities

3.1 `WAIT_TIME` Wait time

```
SUBROUTINE WAIT_TIME(TIME)
```

Action:

This routine waits the specified time then returns. In particular cases the program must wait for the completion of an operation.

Input parameter:

`TIME`	Time in seconds

3.2 `KEY_CTRL_C` Test termination request

```
FUNCTION KEY_CTRL_C()
```

Action:

This function returns the status of the termination flag that show whether CTRL-C button has been pressed since the last call of this routine.

Output parameter:

`KEY_CTRL_C`	Status of pressed CTL-C button `(LOGICAL)`

3.3 `UCASE` Uppercase string converter

```
SUBROUTINE UCASE(STR)
```

Action:

Uppercase converter for FORTRAN character strings. A space character in the string regarded as the text termination.

Input-Output parameter:

STR	Character array to change the case of (STRING)

3.4 SCREEN_CLEAR **Clear screen**

```
SUBROUTINE SCREEN_CLEAR()
```

Action:

This routine clears the screen and moves the cursor to the lefttop corner.

3.5 SCREEN_HOME **Move cursor home**

```
SUBROUTINE SCREEN_HOME()
```

Action:

This routine moves the cursor to the lefttop corner of the screen.

Acknowledgments

Many thanks to Franck Di Maio of PS/CO who has introduced me into the secrets of the PS Accelerator Control System and has provided me with all the right information, documents and codes, very helpfully.

I also appreciate the technical help and advises of my colleagues Eugène Tanke and Alessandra Lombardi during the test phase of the interface.

A special thank goes to my section leader Werner Pirkl who has encouraged me to carry out this job.

References

1. A. Ster, *Linac 2 performance optimization,* CERN/PS 97-05 (HP) (1997).

2. A. Nathaniel, *Interfacing Fortran and C,* CERN Computer Newsletters, 217 July - September (1994).

3. F. Di Maio and A. Risso, *The CERN-PS Equipment Access Library, Software specifications, version 3,* CERN/PS/CO/Note 93-87 (1993).

4. Y. Pujante, *PPM Classes: User Guide,* PS/CO internal note, 3 December (1995).