BNL-105206-2014-TECH

Booster Technical Note No. 162;BNL-105206-2014-IR

PROGRAMMING INTERFACE WITH THE BOOSTER DATABASE, EXAMPLES

E. H. Auerbach

April 1990

Collider Accelerator Department

Brookhaven National Laboratory

U.S. Department of Energy

USDOE Office of Science (SC)

DISCLAIMER

PROGRAMMING INTERFACE WITH THE BOOSTER DATABASE, EXAMPLES

BOOSTER TECHNICAL NOTE
NO. 162

E.H. AUERBACH and A. LUCCIO

APRIL 6, 1990

ALTERNATING GRADIENT SYNCHROTRON DEPARTMENT
BROOKHAVEN NATIONAL LABORATORY
UPTON, NEW YORK 11973

PROGRAMMING INTERFACE WITH THE BOOSTER DATABASE. EXAMPLES.

E.H.AUERBACH and A.LUCCIO

6 April 1990

1. The Booster Database

All data for the AGS Booster are contained in a relational database (DB), which is an extension of the AGS Database[1] with some variations; tools to manage the database belong to the commercial package Interbase. Normal access to the DB to write to and to read from is conveniently done by use of the Query Language Interpreter (QLI).

The data contained in the DB are mapped to the outer world (the console) in the convenient form of tables of data, called "relations". Each row of a relation is a "record" and each column is a "field". The name of a record, contained in field one, is the name of a physical element of the booster, like a magnet, a beam position monitor, etc, chosen according to an accepted nomenclature[2]. The most important property of the DB is that once an entry in a field is changed, the changement will show in all relations.

The relations presently defined for the Booster DB are shown in Appendix A; some relations contain informations on the geometry and geography of the Booster and on the properties of machine elements, others contain also machine (Twiss) parameters at the element locations. The latter relations are most interesting for use with application programs. Twiss parameters are calculated by a model, e.g. **MAD**[3] and written to the DB; when the real machine will be available, measured values will be also entered to the DB.

---

[1]T.Clifford, "A C Programmer's View of a Relational Database", Proc. Int. Conf. on Accelerator and Large Experimental Systems, Vancouver, BC, Canada Oct 30- Nov 3, 1989

E.H. Auerbach, "A Database for the Booster (From the Viewpoint of Modeling)", AGS Note, Draft rev. 5/19, May 15, 1989

[2]K.Reece, "Device Names for the AGS Complex", AGS/AD/Tech. Note No. 317, March 6, 1989, and modifications thereof

[3]Iselin, F. Ch, and Niederer, J., "The MAD Program (Methodical Accelerator Design, Version 6, User's Reference Manual", Rept. CERN/LEP-TH/87-33, Geneva, April 1987

All application programs for the model based control of the Booster should use the DB. Since access times to the DB are unacceptably long for real time use, the best strategy is to write ordinary Apollo files from the DB, use these files at all times, and write back results to the DB only at specific times (typically "at the end of the run"). This procedure, schematically shown in Figure 1, has the additional advantage to keep the DB protected from the risk of being corrupted as a results of errors or program malfunctions.

Program calls to read DB records and entire relations and to write back record to the DB have been written[4]. In this report we will discuss the use of these calls to write files from the DB, we will describe two examples of use of the files and we will show how the results of calculations can be stored back in the DB. The described procedures are meant to become the blueprint for the usage of the DB in connection with all model based control programs.

2. Read and Write to the Booster Database

2.1. Read

To read a record of a relation from the DB and write to a file, the needed calls[4] are:

set_db_server ;
get_relation;
print_db_record.

These calls are implemented in the program module **relation_import.c**, shown in Appendix B. Since the form of the directly obtained output file is not too convenient, we reformat it in a way that the file will resemble the DB relation as it appears on the console screen with the QLI "print" command. The files thus produced are the "lookout tables" that can be inspected at all times during accelerator operation.

Each record in the file has the record (element) name in a field of 22 character, followed by the other entries in field of 12 characters each. Reformatting is performed by

---

[4]C.Griffiths, "Database Routine Descriptions" and "Additional Routines", Database Info (Apollo), 02/14/90 and 02/15/90

the program **file_reformat.ftn**. This program also orders the machine elements according to increasing values of the longitudinal coordinate s.

The two programs above are run interactively tby the Aegis[5] script **DB_IMPORT**, shown in Appendix C, that sequentially asks for the relation to be read and written to a file. The resulting files are given the same name as the corresponding DB relations, and are normally stored in an archive with the name of the "machine" [here, "machine" denotes a subsystem, like LTB (Linac to Booster), HEBT, BOOSTER, etc. For the LTB line we use the archive name LTB_archive].

To invoke **DB_IMPORT** at the Aegis prompt $, type

$ DB_IMPORT

and select the machine to be read.

2.2. Write

Since programs will not be used in general to enter in the DB variations of the geometry of the Booster or of the properties of the elements, for the time being we limit ourselves to write Twiss parameters to the DB. We have chosen to write entries to the DB through the relation "machine_model_parameters". This is done in two steps: first with a Fortran program that reads the output of a modelling code and then with the program **record_export.c** (Appendix D), that makes use of the calls[4]

set_db_server;
get_record;
put_db_record.

The Fortran programs reading from the model is taylored to the output of the specific model. Using **MAD** with the "tape" option, the program is **mad_export.ftn** (Appendix E), that reads the "TWISS" file created by **MAD**, and writes an input script for **record_export.c**. An example of such a script is shown in Appendix F.

---

[5]Aegis is the Apollo Shell Command language. See: "Domain System Call Reference", Apollo Computers Inc., Order No. 002547, Rev. 04, 1987.

Both programs are run through an interactive script **DB_EXPORT** (Appendix G), that gives one the choice to update the Twiss parameters at the location of a given element, or at all elements in the machine . At the prompt just type

$ DB_EXPORT

3.EXAMPLE: **MAD** run for the LBT

To run **MAD**, an input file "mad.in" must be prepared. A Fortran program **madwrite.ftn**[6] has been written that reads the files "machine_element_layouts" and "magnet_properties" and accordingly updates an input file "mad.write". The arrangement of the elements to form machine "lines", the addition of a title and of starting Twiss parameters if the machine is an "open line", is done by editing the input file mad.write (afterwards renamed mad.in). Editing can be directly performed or can be done automatically with the interactive Aegis script **DB_MAD**, shown in Appendix H. An example of input file mad.in for the LTB (Linac to Booster) line obtained in this way is shown in Appendix I.

Once a **MAD** run is successfully completed, **DB_EXPORT** can be used to read its output and transfer the values of the Twiss parameters to the DB via the relation machine_model_parameters, as discussed in Section 2.2. Appendix J shows an example of a machine_model_parameters file after the loop DB-**MAD**-DB was completed.

4. EXAMPLE: PLOT the layout of the LTB

To plot the layout of a boster machine (in the present example: LTB) from the database, a **CPLOT**[7] plot file is created with the Fortran program **plot_import.ftn.** Input to this program is the file "element_layouts" created from the DB with

[6] E.H. Auerbach, "madwrite.ftn", program and notes, March 1990

[7] A.Luccio,"CPLOT: an Apollo Plotting Program using CALCOMP and GPR", AGS Booster Tech. Note No. 156, January 9, 1990

DB_IMPORT as described in Sec. 2.1. The plotting procedure is controlled by the Aegis scripts **LAYOUT_PLOT** and **PLOT**, shown in Appendix K.

The simple graphic convention used in the plots is shown in Figure 2. This convention is temporary: all magnets are represented by rectangles, all monitors by lines; the longitudinal extent of a rectangle is to scale. The names of the elements, as read in the DB, are automatically shown on the plot. Figure 3 shows a plot for the LTB line.

The entire procedure is controlled by the scripts **LAYOUT_PLOT** and **PLOT**, shown in the Appendices J and K. At the Aegis script just invoke:

$ LAYOUT_PLOT

and a plot of the machine that has been previously selected in DB_IMPORT will appear on the screen.

Appendix A Relations for the Booster Database[8]

(Names in < > are mnemonic and do not correspond exactly to names in the relations)

RELATION NAME	DESCRIPTION	FIELDS
(1) element_layouts	location and basic geometrical properties of the elements of the booster	machine_element_name machine s_coord element_length section x_, y_, s_offset pitch, yaw, roll
(2) machine_elements	functional description of the machine elements	machine_element_name machine_element_type subsystem_name ser_no, control_device_name usable_flag
(A) machine_list	"view", from relations (1) and (2) above	machine_element_name machine, section s_coord element_length machine_element_type subsystem_name control_device_name usable_flag
(3) magnet_properties	physical magnets	ser_no prototype_name, data_type mag_length <entrance, exit angles> magnet_model_name <relative field errors>

[8]These are the relations in the first version of the Booster DB. An upgrading is in progress. See: E.H.Auerbach, "A New Iteration for the Booster Model Database", informal note 6 April 1990

(4) magnet_prototypes	design prototypes description	prototype_name magnet_class mag_length, phys_length core_length apert_type, apert_x, apert_y gap_height, pole width pole_tip_radius resistance, inductance no_magnets comments
(5) magnet_models	magnet parameters for use in modelling	<machine element name> momentum <multipole strengths> K0..K5
(6) machine_model_parameters	Twiss parameters at element location from modelling	machine_element_name operating_regime s_coord beta_x, beta_y ⌉ alfa_x, alfa_y upstream eta_x, eta_y downstream mu_x, mu_y x_c, y_c, xpr_x, ypr_y ⌋
(7) <instruments>		machine_element_name ser_no <x, y offset> <calibration constants>
(8) magnet_strings	properties of aggregates of magnets controlled together	control_device_name resistance, inductance max_current, max_voltage <time constants> no_magnets B_I_transfer function

Appendix B   relation_import.c

```
#include <stdio.h>
#include "/users/source/dg_tools.ins.c"
#include "/users/source/mailbox_tools.ins.c"
#include "/users/source/database_status.ins.c"
#include "/users/source/database_server.ins.c"
#include "/users/source/database_client.ins.c"

main()
{
  int               relation_size,size_of_relation(),count,i;
  char              *buffer,*data_ptr,filename[50];
  status_enum_t     status,get_relation(),get_record_names();
  relation_enum_t relation_name;
  char              relation[50];
  FILE              *fd;
  FILE              *fopen();

  scanf             ("%s",relation);                {read from script
  relation_name     = parse_relation(relation);
                    set_db_server("booster_server",0);
  status            = get_relation(relation_name,&buffer,&count);
  scanf             ("%s",filename);                {read from script
  fd                = fopen(filename, "w");
  relation_size     = size_of_relation(relation_name);

                  if (count != 0)
                  {
                    data_ptr = buffer;
                    for (i=0;i< count;i++)
                    {
                              print_db_record(fd,relation_name,data_ptr);
                      data_ptr = data_ptr + relation_size;
                    } /*end for count*/
                  } /* if count is not 0 */
                  free(buffer);
                  fclose(fd);
}
```

Appendix C DB_IMPORT.

Aegis script to transfer machine element data from the Booster DB to files. It runs two programs: **relation_import.c** (App.B) and **file_reformat.ftn.**

```
eon

      relation_no := 1
while (( ^relation_no <= 8 )) do                                                  #
                                                        machine := ALL            #
  select ^relation_no                                                             #
    case 1 relation := element_layouts              ; no_of_fields := 11          #
                                                        machine := CHOICE         #
    case 2 relation := instrument_calibration       ; no_of_fields := 35          #
    case 3 relation := machine_elements             ; no_of_fields := 8           #
    case 4 relation := machine_element_parameters ; no_of_fields := 13            #
    case 5 relation := machine_list                 ; no_of_fields := 17          #
                                                        machine := CHOICE         #
    case 6 relation := machine_model_parameters   ; no_of_fields := 0             #
    case 7 relation := magnet_properties            ; no_of_fields := 17          #
    case 8 args "no more relations"  ; xdmc dq                                    #
  endselect                                                                       #

    read -p (( " -> choose relation (" + ^relation_no +"): " @
                           + ^relation + " [y/n/q]? " )) choose
    if (( ^choose = q )) then xdmc dq endif
    if (( ^choose = y )) then
      if (( ^relation = instrument_calibration   or @
            ^relation = machine_model_parameters )) then
      args " ====== not implemented ======"
      else exit
      endif
    endif
      relation_no := ^relation_no +1
enddo

if existf ^relation then
  read -p " -> file ^relation exists. overwrite [y/n]? " delete
  if (( ^delete = y )) then
    dlf ^relation
    args (( " relation: " + ^relation )) >> ^relation
  else
    DB_IMPORT
  endif
endif

relation_import<<{
^relation
junk_junk
{

if (( ^machine = CHOICE )) then
  read -p " -> machine (LTB,BTA,BOOSTER,ALL): " machine
endif

file_reformat<<{
junk_junk
^no_of_fields
^machine
temp_file
{

  catf temp_file >> ^relation
```

Appendix D record_export.c

```
#include <stdio.h>
#include "/users/source/dg_tools.ins.c"
#include "/users/source/mailbox_tools.ins.c"
#include "/users/source/database_status.ins.c"
#include "/users/source/database_client.ins.c"
#include "/users/development/dsee/source/database_server.ins.c"
#include "/users/development/dsee/database/inter.ins.c"
#ifndef i_booster_model_ins_c
#include "/users/development/dsee/database/booster_model.ins.c"
#endif

char *status_to_text(status)
status_enum_t status;

main()
{
  char             field_id[50];
  char             buffer[mbx_buffer_size],record_id[50];
  status_enum_t    status,get_record(),put_db_record();
  relation_enum_t relation_name,element_layouts;
  char             relation[50];
  i_machine_element_parameter_record_t   *machine_element_parameter_record;
  float            new_value;
                   set_db_server("booster_server",0);
  scanf            ("%s",relation);             /*read from script*/
  relation_name    = parse_relation(relation);
  scanf            ("%s",field_id) ;            /*read from script*/

  while (scanf("%s%f",record_id,&new_value) !=EOF)  /*read from script*/
  {
    status         = get_record(relation_name,record_id,buffer);
    machine_element_parameter_record = (i_machine_element_parameter_record_t *) buffer;
    switch         (field_id[0])
    {
      case  'A' :
                   machine_element_parameter_record->alpha_x = new_value; break;
      case  'B' :
                   machine_element_parameter_record->beta_x = new_value;  break;
      case  'M' :
                   machine_element_parameter_record->mu_x = new_value;    break;
      case  'E' :
                   machine_element_parameter_record->eta_x = new_value;   break;
      case  'a' :
                   machine_element_parameter_record->alpha_y = new_value; break;
      case  'b' :
                   machine_element_parameter_record->beta_y = new_value;  break;
      case  'm' :
                   machine_element_parameter_record->mu_y = new_value;    break;
      case  'e' :
                   machine_element_parameter_record->eta_y = new_value;   break;
      case  'x' :
                   machine_element_parameter_record->x_c = new_value;     break;
      case  'u' :
                   machine_element_parameter_record->xpr_c = new_value;   break;
      case  'y' :
                   machine_element_parameter_record->y_c = new_value;     break;
      case  'v' :
                   machine_element_parameter_record->ypr_c = new_value;   break;
      default   :
                   printf ("field_name unknown\n");                       break;
    }
    status         = put_db_record(relation_name,buffer);
    printf         ("%s\n", status_to_text(status) );
  }
}
```

Appendix E mad_export.ftn

```
      character*80 record,relation,element,field,vname(15)*8,f_keyword*15
      real        v(15)
      logical     all
*
* Each field name is represented by a letter (keyword)
* in the export script file
      data        vname/
     &    'alpha_x ','beta_x  ','mu_x    ','eta_x   ','        ',{ABME
     &    'alpha_y ','beta_y  ','mu_y    ','eta_y   ','        ',{abme
     &    'x_c     ','xpr_c   ','y_c     ','ypr_c   ','s_coord '/{xuyvs
      data f_keyword/'ABME abme xuyvs'/,all/.false./

      open        (10,file='mad.twiss')    {mad.twiss="tape" from MAD
      read        (10,*)
      read        (10,*)
      open        (20,file='EXPORT_SCRIPT')
      write       (20,'(''record_export<<%'')')

        read      (*,'(a)') relation          {read from script: relation name
        write     (20,'(a)') relation
        read      (*,'(a)') field             {read from script: fieldname
      do       i = 1,24                       {return with length of fname
        if (field(i:i).eq.' ') then
          lf = i-1
          goto 100
        endif
      end do
*
* Write keyword corresponding to field name in the export script file
100   do       iv = 1,15
        if (vname(iv)(1:lf).eq.field(1:lf)) then
          write   (20,'(a)') f_keyword(iv:iv)//' : '//field
          goto 110
        endif
      enddo
*
* Either read machine element from terminal or from file "relation"
110     read      (*,'(a)') element           {read from script: element name
        if (element(1:3).ne.'ALL') goto 210
        all       = .true.
        open      (11,file=relation)
        read      (11,*)
        read      (11,*)
        read      (11,*)
200     read      (11,'(a)',end=800) element  {read from file: element name
        if (element(5:5).eq.' ') goto 200
210   do       i = 1,24                       {length of elname
        if (element(i:i).eq.' ') then
          le = i-1
          goto 120
        endif
      end do
      if (le.le.4) goto 200
*
* Read element name in  mad.twiss. Read corresponding parameters
120     rewind    (10)
130     read      (10,'(a)',end=200) record   {if end read next element
*
* Compare element name with reading in "mad.twiss" (record)
      if (record(9:8+le-4).eq.element(5:le)) then
        read      (10,*)
        read      (10,'(5e16.0)') (v(i), i=1,15)
        goto 140
      endif
        goto 130
140   write       (20,'(a,5x,e14.7)') element(1:le),v(iv)
      if (all) goto 200

800   write       (20,'(''%'')')
      end
```

Appendix F

Example of Aegis input script to **record_export.c.** This constitutes, in Aegis lingo an "here document"[9]

```
record_export<<%
machine_element_parameters
B             beta_x
BLI.KR1        0.325
BLI.DH015      4.826
BLI.QH1        5.126
BLI.DV018      5.619
BLI.BPM019     6.127
BLI.QV2        6.427
BLI.QH3        7.727
BLI.BPM026     8.384
BLI.DV026      8.728
BLI.QV4        9.028
BLI.QH5       10.328
BLI.MW035     11.328
BLI.DH1       12.770
BLI.DH2       14.370
BLI.QH6       15.028
BLI.DH3       16.570
BLI.QH7       17.228
BLI.BPM066    20.951
BLI.DH4       22.193
BLI.DH075     24.044
BLI.BPM078    24.552
BLI.QH8       24.852
BLI.DV082     25.853
BLI.QV9       26.153
BLI.DH088     27.646
BLI.BPM090    28.154
BLI.QH10      28.454
BLI.DV095     29.455
BLI.QV11      29.755
BLI.BPM102    31.755
BLI.QH12      32.055
BLI.MW107     33.347
BLI.BPM109    33.855
BLI.QV13      34.155
BLI.DV112      0.000
%
```

[9] Apollo Computers, "DOMAIN System User's Guide", Order No. 005488, Rev. 02, January 1987, p. 9-9

Appendix G DB_EXPORT.

Aegis script to transfer Twiss parameters to the Booster DB. It runs two programs: **mad_export.ftn** (App.E) which produces the here document EXPORT_SCRIPT (An example in App. F), and **record_export.c.**

```
eon
if existf TWISS then chn TWISS mad.twiss -l endif
relation := machine_element_parameters
read -p " --> enter machine element (or ALL): " element

for field in "beta_x beta_y mu_x mu_y alpha_x alpha_y eta_x eta_y" by word
args (( " === variable being updated : " + ^field + " === " ))
mad_export<<%
^relation
^field
^element
%
EXPORT_SCRIPT
endfor

dlf EXPORT_SCRIPT
```

Appendix H **DB_MAD**

Aegis script to prepare the input to MAD (mad.in) using the program **madwrite.ftn** and run MAD. The final product are the three files mad.in, mad.out and mad.twiss (renamed from TWISS, created with the "tape" option in **MAD**)

```
eon
# Extract DB files from archive
read -p " Enter machine (LTB,BOOSTER..) --> " machine
arcf -x (( ^machine + "_archive" )) element_layouts magnet_properties
if existf element_layouts magnet_properties then
  cpf element_layouts   temp1 -r
  mvf element_layouts   temp2 -r
  cpf magnet_properties temp3 -r
  mvf magnet_properties temp4 -r
else
  args " either file element_layouts or magnet_properties does not exist "
  xdmc dq
endif

# Strip first three lines from element_layouts and magnet_properties
file := temp
for i := 1 to 2
ed -n ^file <<%
1
1,3D
w
q
%
file := temp4
endfor

# Read and put title in "element_layouts"
readln -p " --> enter title: " title
args (( "TITLE! " + ^title )) >>element_layouts
catf temp2    >>element_layouts
if (( ^machine = LTB )) then    # append init. Twiss parameters to element_layouts
  read -p " --> beta_x = 3.649.  Accept [y] or enter new value: beta_x= " btx
    if (( ^btx = y )) then btx := (( "3.649" )) endif
  read -p " --> beta_y = 10.024. Accept [y] or enter new value: beta_y= " bty
   if (( ^bty = y )) then bty := (( "10.024" )) endif
  read -p " --> alfa_x = -0.341. Accept [y] or enter new value: alfa_x= " afx
   if (( ^afx = y )) then afx := (( "-0.341" )) endif
  read -p " --> alfa_y =  2.163. Accept [y] or enter new value: alfa_y= " afy
   if (( ^afy = y )) then afy :=  (( "2.163" )) endif
args ** ((BETX=^btx)) ((ALFX=^afx)) ((BETY=^bty)) ((ALFY=^afy)) >>element_layouts
endif

xdmc ce element_layouts ; read -p "continue [y/n]? " c
cpf  temp4      magnet_properties

# Write input to MAD
madwrite
cpf mad.write mad.in -r
cpf temp1 element_layouts   -r
cpf temp2 magnet_properties -r

# Run MAD
/reality/luccio/model/mad.run7
if existf TWISS then mvf TWISS mad.twiss -r
else args "file TWISS does not exist!" ; catf mad.out
endif
```

Appendix I File mad.in, input to **MAD**, for the LTB (Linac to Booster) Line

```
TITLE,   LTB Line. Test.
KR1:     SBEND,   L = 0.344000, ANGLE= -0.1309, E1=-.032725, E2=-.032725
DH015:   HKICK,   L = 0.344000, KICK = 0.
QH1:     QUAD,    L = 0.300000,   K1 = 1.400
DV018:   VKICK,   L = 0.344000, KICK = 0.
BPM019:  DRIFT,   L = 0.508000
QV2:     QUAD,    L = 0.300000,   K1 =-1.600
QH3:     QUAD,    L = 0.300000,   K1 = 0.600
BPM026:  DRIFT,   L = 0.508000
DV026:   VKICK,   L = 0.344000, KICK = 0.
QV4:     QUAD,    L = 0.300000,   K1 =-1.200
QH5:     QUAD,    L = 0.300000,   K1 = 1.600
MW035:   MARKER,  L = 0.
DH1:     SBEND,   L = 1.183000, ANGLE=-0.550499, E1=-.2752495,E2=-.2752495
DH2:     SBEND,   L = 1.183000, ANGLE=-0.550499, E1=-.2752495,E2=-.2752495
QH6:     QUAD,    L = 0.300000,   K1 = 1.466
DH3:     SBEND,   L = 1.183000, ANGLE=-0.550499, E1=-.2752495,E2=-.2752495
QH7:     QUAD,    L = 0.300000,   K1 = 0.960
BPM066:  DRIFT,   L = 0.508000
DH4:     SBEND,   L = 1.183000, ANGLE=-0.550499, E1=-.2752495,E2=-.2752495
DH075:   HKICK,   L = 0.344000, KICK = 0.
BPM078:  DRIFT,   L = 0.508000
QH8:     QUAD,    L = 0.300000,   K1 = 1.107
DV082:   VKICK,   L = 0.344000, KICK = 0.
QV9:     QUAD,    L = 0.300000,   K1 =-1.165
DH088:   HKICK,   L = 0.344000, KICK = 0.
BPM090:  DRIFT,   L = 0.508000
QH10:    QUAD,    L = 0.300000,   K1 = 0.870
DV095:   VKICK,   L = 0.344000, KICK = 0.
QV11:    QUAD,    L = 0.300000,   K1 =-0.813
BPM102:  DRIFT,   L = 0.508000
QH12:    QUAD,    L = 0.300000,   K1 = 1.600
MW107:   MARKER,  L = 0.
BPM109:  DRIFT,   L = 0.508000
QV13:    QUAD,    L = 0.300000,   K1 =-1.580
DV112:   MARKER,  L = 0.

DRF01:   DRIFT,   L = 4.1475
DRF02:   DRIFT,   L = 0.1490
DRF03:   DRIFT,   L = 1.0000
DRF04:   DRIFT,   L = 0.2590
DRF05:   DRIFT,   L = 0.4170
DRF06:   DRIFT,   L = 0.3580
DRF07:   DRIFT,   L = 0.3590
DRF08:   DRIFT,   L = 0.357999
DRF09:   DRIFT,   L = 3.215002
DRF10:   DRIFT,   L = 0.058999
DRF11:   DRIFT,   L = 1.5070
DRF12:   DRIFT,   L = 0.6570
DRF13:   DRIFT,   L = 1.149001
DRF14:   DRIFT,   L = 0.000001
DRF15:   DRIFT,   L = 1.491999
DRF16:   DRIFT,   L = 0.000002
DRF17:   DRIFT,   L = 1.291999
DRF18:   DRIFT,   L = 0.000002
DRF19:   DRIFT,   L = 0.844999

LIN001: LINE=(KR1,DRF01,DH015,QH1,DRF02,DV018,BPM019,QV2,DRF03,          &
          QH3,DRF02,BPM026,DV026,QV4,DRF03,QH5,DRF03,MW035,DRF04,        &
          DH1,DRF05,DH2,DRF06,QH6,DRF07,DH3,DRF08,QH7,DRF09,BPM066,      &
          DRF10,DH4,DRF11,DH075,BPM078,QH8,DRF12,DV082,QV9,DRF13,        &
          DH088,BPM090,DRF14,QH10,DRF12,DV095,QV11,DRF15,BPM102,         &
          DRF16,QH12,DRF17,MW107,DRF18,BPM109,QV13,DRF19,DV112)

 USE, (LIN001)
PRINT, LIN001

TWISS, TAPE, DELTAP=0., BETX=3.649, ALFX=-0.341, BETY=10.024,            &
          ALFY=2.163

STOP !
```

Appendix J File machine_element_parameters. The values of the Twiss parameters have been calculated by **MAD** and written to the DB with **DB_EXPORT.**

relation: machine_element_parameters

machine element name	beta x	beta y	alpha x	alpha y	eta x	eta y	mu x	mu y	x_c	y_c	xpr_c	ypr_c
BLI.KR1	0.325000	8.603000	3.868000	2.191000	-0.021000	0.000000	0.014000	0.006000	0.000000	0.000000	0.000000	0.000000
BLI.MW107	33.347000	12.362000	1.701000	-3.017000	-0.605000	-0.004000	1.063000	0.919000	0.000000	0.000000	0.000000	0.000000
BLI.QH3	7.727000	10.361000	6.569000	-1.262000	-0.594000	-0.001000	0.190000	0.366000	0.000000	0.000000	0.000000	0.000000
BLI.DH2	14.370000	7.268000	1.519000	3.018000	-1.537000	-0.001000	0.494000	0.465000	0.000000	0.000000	0.000000	0.000000
BLI.QH6	15.028000	4.493000	2.061000	0.226000	-2.031000	0.000000	0.552000	0.484000	0.000000	0.000000	0.000000	0.000000
BLI.DH4	22.193001	12.282000	9.181000	0.890000	-2.448000	0.004000	0.842000	0.731000	0.000000	0.000000	0.000000	0.000000
BLI.BPM109	33.855000	15.638000	0.808000	-3.432000	-0.089000	-0.005000	1.134000	0.925000	0.000000	0.000000	0.000000	0.000000
BLI.EXIT	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
BLI.DH015	4.826000	2.539000	12.667000	-0.843000	-0.610000	0.000000	0.125000	0.299000	0.000000	0.000000	0.000000	0.000000
BLI.DV018	5.619000	6.373000	8.826000	-3.422000	-0.548000	0.000000	0.136000	0.332000	0.000000	0.000000	0.000000	0.000000
BLI.BPM066	20.951000	10.649000	6.759000	-1.879000	-2.301000	0.004000	0.818000	0.714000	0.000000	0.000000	0.000000	0.000000
BLI.DV082	25.853001	15.338000	9.255000	-3.324000	-2.637000	0.001000	0.890000	0.786000	0.000000	0.000000	0.000000	0.000000
BLI.BPM090	28.153999	8.638000	11.426000	1.419000	-2.901000	-0.001000	0.927000	0.817000	0.000000	0.000000	0.000000	0.000000
BLI.DH3	16.570000	3.106000	2.954000	0.749000	-2.476000	0.001000	0.655000	0.549000	0.000000	0.000000	0.000000	0.000000
BLI.QV9	26.153000	15.711000	8.653000	2.116000	-2.554000	0.001000	0.895000	0.789000	0.000000	0.000000	0.000000	0.000000
BLI.QH1	5.126000	3.484000	12.062000	-2.439000	-0.611000	0.000000	0.129000	0.315000	0.000000	0.000000	0.000000	0.000000
BLI.DH075	24.044001	9.485000	14.223000	0.620000	-3.204000	0.002000	0.868000	0.759000	0.000000	0.000000	0.000000	0.000000
BLI.BPM026	8.384000	12.128000	5.978000	-1.427000	-0.590000	-0.001000	0.206000	0.375000	0.000000	0.000000	0.000000	0.000000
BLI.DV026	8.728000	13.139000	5.735000	-1.513000	-0.588000	-0.002000	0.216000	0.379000	0.000000	0.000000	0.000000	0.000000
BLI.QV11	29.754999	10.221000	6.971000	1.458000	-2.165000	-0.002000	0.956000	0.844000	0.000000	0.000000	0.000000	0.000000
BLI.MW035	11.328000	11.273000	5.812000	-2.043000	-0.611000	-0.002000	0.269000	0.422000	0.000000	0.000000	0.000000	0.000000
BLI.QH7	17.228001	2.555000	3.746000	-0.295000	-2.797000	0.001000	0.685000	0.588000	0.000000	0.000000	0.000000	0.000000
BLI.BPM078	24.552000	8.893000	15.904000	0.546000	-3.411000	0.002000	0.873000	0.767000	0.000000	0.000000	0.000000	0.000000
BLI.DV112	0.000000	15.448000	0.751000	4.033000	0.216000	-0.004000	1.198000	0.928000	0.000000	0.000000	0.000000	0.000000
BLI.BPM019	6.127000	10.364000	6.069000	-4.435000	-0.483000	-0.001000	0.147000	0.342000	0.000000	0.000000	0.000000	0.000000
BLI.DV095	29.455000	10.349000	7.376000	-1.040000	-2.258000	-0.002000	0.949000	0.839000	0.000000	0.000000	0.000000	0.000000
BLI.BPM102	31.754999	5.611000	8.410000	0.847000	-2.077000	-0.003000	0.998000	0.886000	0.000000	0.000000	0.000000	0.000000
BLI.QV2	6.427000	11.528000	5.492000	0.747000	-0.479000	-0.001000	0.156000	0.346000	0.000000	0.000000	0.000000	0.000000
BLI.QV4	9.028000	12.734000	6.142000	2.820000	-0.616000	-0.002000	0.224000	0.383000	0.000000	0.000000	0.000000	0.000000
BLI.DH088	27.646000	10.169000	10.609000	1.596000	-2.813000	0.000000	0.920000	0.808000	0.000000	0.000000	0.000000	0.000000
BLI.QH10	28.454000	8.469000	11.046000	-0.839000	-2.839000	-0.001000	0.931000	0.822000	0.000000	0.000000	0.000000	0.000000
BLI.QH5	10.328000	7.646000	10.181000	-1.584000	-0.810000	-0.001000	0.249000	0.405000	0.000000	0.000000	0.000000	0.000000
BLI.QV13	34.154999	15.448000	0.751000	4.033000	0.216000	-0.004000	1.198000	0.928000	0.000000	0.000000	0.000000	0.000000
BLI.DH1	12.770000	13.856000	2.190000	1.095000	-0.651000	-0.002000	0.333000	0.440000	0.000000	0.000000	0.000000	0.000000
BLI.QH8	24.851999	9.471000	15.359000	-2.537000	-3.363000	0.001000	0.876000	0.773000	0.000000	0.000000	0.000000	0.000000
BLI.QH12	32.055000	5.930000	7.545000	-1.961000	-1.916000	-0.003000	1.004000	0.895000	0.000000	0.000000	0.000000	0.000000

Appendix K LAYOUT_PLOT and PLOT

Aegis scripts to create a **CPLOT** file from element_layouts and plot it.

```
eon

file    := layout.ppar
machine := LTB
s_begin := -5
s_end   := 40
y_min   := -20
y_max   := 20
y       := n

if existf ^file then
  read -p (( " ->file " + ^file + " exists. Overwrite [y/n]? " )) y
endif
if (( ^y = y )) then
  dlf ^file -l
  args " $plotp" >> ^file
  args (( "  title='" + ^machine + "$'" ))  >> ^file
  args    "  xx=9."                         >> ^file
  args    "  yy=4."                         >> ^file
  args (( "  xlabel='s (m)$' "))            >> ^file
  args (( "  ylabel=' $' " ))               >> ^file
  args (( "  xpmin=" + ^s_begin ))          >> ^file
  args (( "  xpmax=" + ^s_end ))            >> ^file
  args (( "  ypmin=" + ^y_min ))            >> ^file
  args (( "  ypmax=" + ^y_max ))            >> ^file
  args    "  n_data_max=1000"               >> ^file
  args    "  max_plot=512"                  >> ^file
  args    "  window=0,100,900,650"          >> ^file
  args    " $"                              >> ^file
endif

plot_import.bin<<%
element_layouts
layout.pdat
%

args " layout.pdat created "
if existf layout.pin then dlf layout.pin endif
args "temp" >> layout.pin
args "layout.pdat" >> layout.pin

PLOT layout.ppar layout.pin 0 0
```

```
eon
read -prompt "edit plot_parameters [y/n]? "e
if
  ((^e = y))
then
  xdmc ce ^1
  read -prompt "continue [y]? "c
endif
//acn40d01/reality/luccio/com/ctime ^1 temp ^3 ^4
cpf ^2 cplot.in -r
//acn40d01/reality/luccio/graf/cplot
```

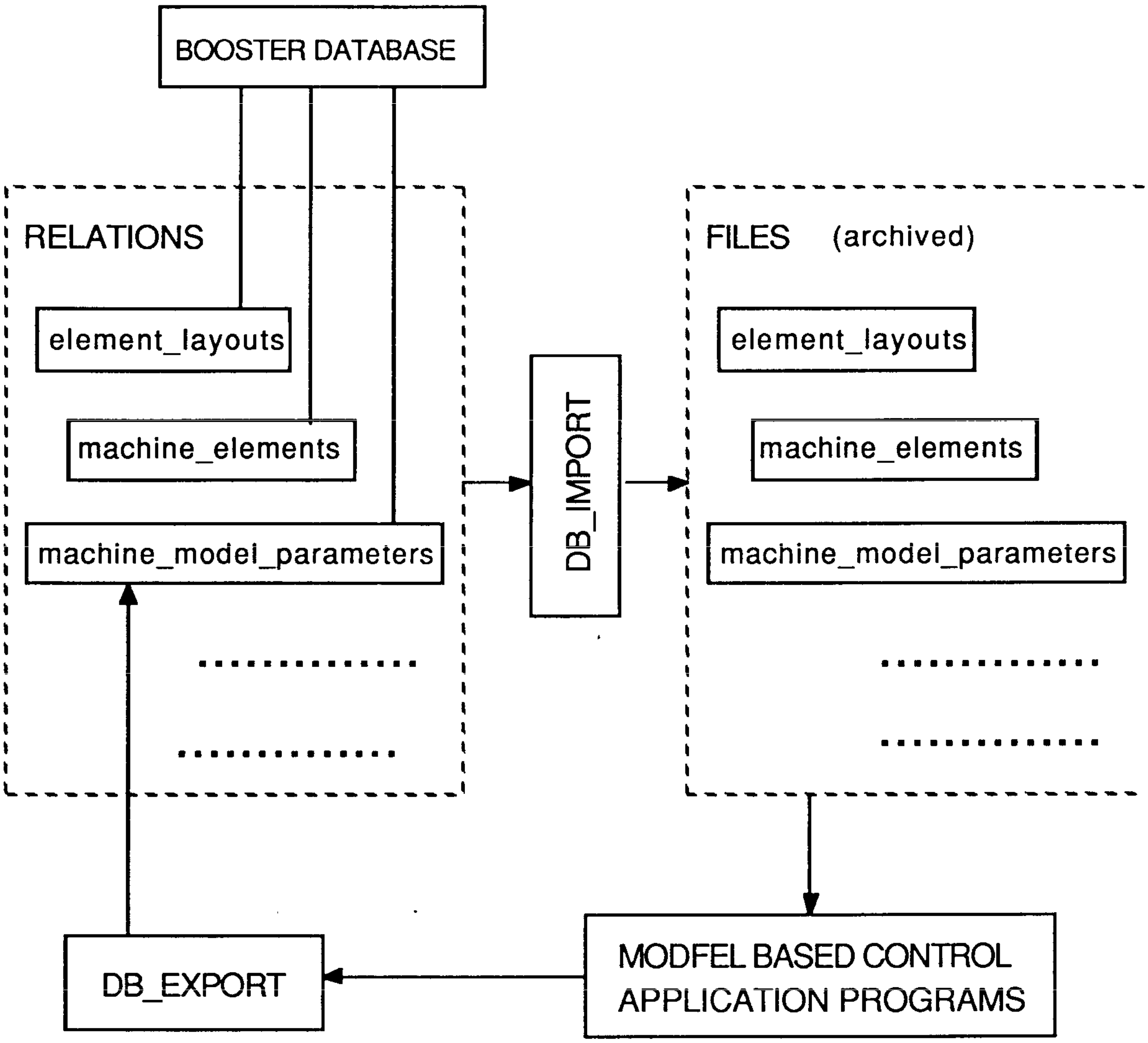

Fig. 1. Transfer from and to the Database.

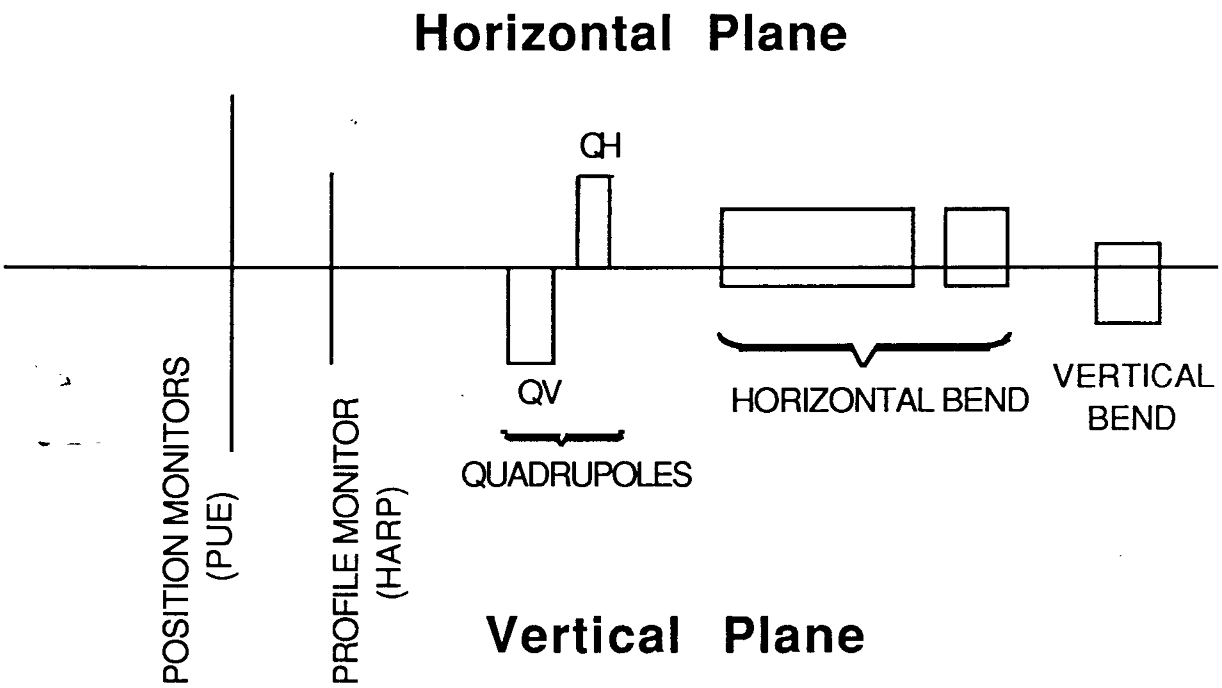

Fig. 2. Graphic conventions for PLOT.

Fig. 3. Layout Plot for the Linac to Booster Line (LTB)