PS/CO-Note 92-20

J.Cuperus, G.Daems, F.Di Maio, Cl.H.Sicard

30 September 1992

# The Reference for Operational Values

## Historical Overview

### Introduction

A control system can be seen as composed of a framework which is fairly fixed and operational data which can be changed at any time by the operators. When something is corrupted, the fixed part can be reconstructed from source files but this is not true of the operational data. When the operator changes a value, this change is done in the data table (DT) of an equipment module (EM). This is in volatile memory and, when something goes wrong, or when you stop the system, these data are lost unless they have been saved on a more permanent medium. Also, if the operator changes a value, he cannot come back to a previous value, or even know what it was.

Several strategies were invented over the last 14 years to make the operational data recoverable without reaching that goal entirely and now, when we want to invent still a new system, it is important to make a review of what worked or not in the past. We will examine the 4 main strategies: save and back, data table block-save , selective data-table save, and archives.

### Save and Back

For each control store (CCV) there are associated REF and BUF stores. There are also properties for transferring a value from one store to another and back. This enables e.g. to save a set of values, do some experimenting with CCV values and then to come back to the original values. This works well if the operator saves all values before he changes them and nobody else saves them while he is away. In practice, the operator is never sure what will happen if he does the 'back' operation and this facility is rarely used, if at all.

### Data-Table Block-Save

The whole ACC DT is saved on a disk at night. This is used after an EM reload or crash to recover with fairly recent data (refs. [2], [4], [5]).

### Selective Data-Table Save

The DSC DT is saved every 10 minutes on a local disc together with structure information, so that it can be used when an EM is reloaded with the DT structure changed (e.g. a member or a column added). In addition, the DT is saved each night on a central disk to be able to recover from corruptions which have already propagated to the local copy.

Some EMs have also a couple of Nodal files which save the operational parameters to a file and restore them on demand (ref. [7]).

### *Archives*

An interesting setting of a sub-system of the accelerator is saved to an archive file. The idea is to be able later to re-set the accelerator in the same way by re-loading the archive (refs. [1] and [3]). There are some difficulties when the configuration of the system has changed in the meantime (e.g. new parameters added) but this can be overcome. A more serious difficulty is the cross-coupling between different users of the accelerator which share settings for a sub-system (more about this later). Also, the operator does not really know what is in the archive and what will happen when he loads it and, when he loads a whole group of data, he is not sure whether everything has been correctly transmitted to the hardware. These difficulties and some proposed remedies (none of them entirely satisfactory) are discussed in refs. [6] and [8] . . [13].

## *Virtual Accelerators and Beam Synthesizers*

A supercycle is a sequence of cycles and each cycle is associated with a PLS telegram. This telegram has groups, with, in each group a number of lines of which one and only one is active in any given cycle. Many pieces of equipment can change values each cycle (Pulse to Pulse Modulation or PPM). If the equipment is linked to a particular group, then there is a value of the parameter for each line of the group and one of the values (the one corresponding to the active line in the group) is used to set the hardware for that cycle.

In the PLS telegram there is a user group (UG), identifying the nature of the operation going on in the cycle, and several elementary groups (EG), identifying the possible settings for a sub-system of the accelerator.

One possibility is to link all equipment to the UG. This gives maximum flexibility because each user has the impression to control a virtual accelerator independently of any other user. Also, with 24 possible users in the new telegram, we do not have to re-use the same line for different users as is the case at present for only 8 possible users. The price you have to pay for this is a lot of duplicate information. Suppose, for instance, that there are only two different settings for an ejection: HIGH and LOW and we have optimized the LOW operation for one user. We must now copy these setting to all users which can profit from this improved setting. Instead of just 2 independent settings, you have now 24.

There is an alternative: we can see the accelerator as a succession of sub-systems: injection, acceleration, transition, ejection, ... Each of these sub-systems has only a few useful different settings, each linked to a line in an EG. The number of independent parameters has now been reduced drastically and we can now see the accelerator as a beam synthesizer. Setting up a beam for a new user can now be as simple as combining a new set of couplings between the lines of the UG and those of the EG of the different subsystems (setting up a new user matrix). The price to pay for this is coupling between different users which share a line in an EG. This can be especially disturbing in case we want to try out a new mode of operation without disturbing other users.

Experience, and interminable discussions, have shown that the operators are attracted to the idea of virtual accelerators but do not want to give up the advantages of beam synthesizers. The new PLS telegram with more lines per group enables us to make better compromises:

- More elementary lines so that we can make finer distinctions between operations.
- A few user lines dedicated to machine development (e.g. MD1 . . MD4). The user matrix would couple these users to reserved lines in each EG (e.g. EJ1 . . EJ4 for ejection). The operator would be able to copy whole sets of parameters from one elementary line to another, first to have a set of initial values for the development session, and afterwards to transfer useful new settings to a specific elementary line.

## *Operational Requirements*

What the operators want is summarized in [14]. This is reproduced literally below:

- For the values of CCV and AQN there must be the possibility to save references individually (by element), by working set and globally for a given Cycle (Cycle in the sense of USER per machine). These references are unique per Cycle. It is necessary to have an indication of the last update of the references. It is not necessary to have this time indication individually but only for the entire set of references of one machine/Cycle.
- The reloading of references must as well be possible individually, by working set and globally.
- It is necessary to be able to reload the references after a crash (i.e. they should be non-volatile).
- It is necessary to have tolerances (absolute or relative) for the CCV and AQN associated individually to each element in order to create a varilog which is based on the above mentioned references (i.e. two varilogs: one for the CCV and one for the AQN values).

## *Additional Requirements*

The operational requirements listed above are minimum requirements for implementing the individual references for a simple machine like the lead linac. We would like to propose, from the beginning, a system which can be implemented on all our machines and which is well integrated with the rest of our system. This leads to the following additional requirements:

- The system should be able to handle PPM parameters, both for user groups and for elementary groups. This may be implied in the operational requirements but is not explicitly stated.
- The set of parameters saved should be complete: not only the analog values but also the complete state of the equipment must be saved. It must be possible to restore the data table to a functional state after a reload of the EM.
- The system should be integrated with the rest of our control system such as knobs and working set displays.

## *Implementation of the Requirements*

We want the storage of the reference values to remain valid when equipment is added or removed and when the list of parameters to be saved changes. We can distinguish between different categories of information:

1. ***A list of what parameters to store for each EM and the attributes of these parameters.*** A similar list exists already for the knob and working set display facilities (ref.[17]) and this information should be moved to Oracle and slightly extended to cover our new requirements. For details, see Appendix 1: Property Tables.
2. ***A list of the pieces of equipment for which the parameters must be stored and their attributes.*** Most of this information exists already in Oracle tables EQUIPMENT and INSTVAL. For details see Appendix 2: Equipment List.
3. ***A way of grouping the equipment in sets: individual, per DSC, per working set ...*** This is either covered by the working set mechanism or by selection criteria applied to Oracle table EQUIPMENT.
4. ***The relation between user lines and elementary lines in the PLS telegram.*** When you want to save reference data, you must first read the actual values from the EM interface. You do this for a user line. For equipment not connected to the user group but to an elementary group, we must find the name of the corresponding elementary line because the values are stored that way in the reference . Note that lines and groups have names in the reference store while they have numbers in the EM call. Translation between names and numbers and between user lines and elementary lines is done with the tgm package (see man tgm for documentation).
5. ***The values of parameters and the environment in which they were stored.*** It is certainly possible to store the values in a spreadsheet, and this would be the simplest solution if only a rather short and fixed list of parameters had to be stored. For a large and flexible system, however, the facilities of a spreadsheet are no match for those of a modern relational database and, to make the whole system work smoothly together, it is only logical to store the reference values also also in Oracle, in tables described in the next section.

All this information should not be of direct concern to the operator. He should have access to the reference values through user friendly application programs which we will discuss later in this note.

## Reference Value Tables

In table REFVAL, a record is foreseen for each parameter. It contains the identifier and value of the parameter and the conditions under which the parameter was acquired:

Table REFVAL = {

| | |
|---|---|
| CPID + | integer, identifying class+property |
| MBNO + | the member number in the class |
| PLSLINE + | the relevant elementary or user PLS line (or null) |
| FIXFLAG + | 'Y' when control value is protected (see below) |
| UPDATED + | the date when VALUE was last changed |
| VALUE | parameter value or ARRID identifier in case of array |

}

Only a single value is foreseen per parameter. If we want to store arrays of values then we must store them in a second table:

Table REFARR = {

| | |
|---|---|
| ARRID + | integer, identifying the array |
| SEQNO + | number for ordering the array (array index) |
| VALUE | value of the parameter |

}

Notes:

- Meaningless identifiers such as CPID and ARRID are automatically supplied by the system and are usually not seen by the users.
- CPID is a number which refers to a description of the parameter which contains the EM class, the property for acquiring the value and some information about the nature of the parameter and the way of displaying it. These details can be found in Appendix 1.
- Arrays of values are needed for GFAs and for the acquisitions of beam monitoring data or digitized scope signals.
- The time of last update of each parameter is not required by the operators. On the other hand, indicating only when any change took place at all in the whole reference store is not very informative. We may indicate the update per accelerator and per PLS line but for this we need still another table. The disk space needed for storing UPDATE for each parameter is reasonable and this is very easy to implement, so we propose this solution.
- When FIXFLAG='Y', the parameter value cannot be changed by downloading the actual value from the EM, but only with an Oracle Form. This is useful for parameters which are rarely, if ever, changed.
- In the remainder of this note, when we mention table REFVAL, we mean the combination of this table and table REFARR.

## *Access to Reference Values*

With Oracle Forms we can search, read, update, and compare the reference values but we need also a more conventional access method for application programs. This can be done with a normal EM single or array call. This is done for a PROPERTY which is in fact a symbol with a value between 1 and 999 (e.g. AQN is a property symbol with symbolic value 257). We introduce now two new symbols:

```
#define VREF  10000;

#define VDAT  20000;
```

An EM call with property AQN + VREF will not return the actual value in the DSC but the corresponding reference value in the database. For NODAL this will not work and a pseudo-property with syntax AQN_VREF will be implemented. Property AQN is used here as an example but this is valid for any property in REFVAL.

An EM call with property AQN + VDAT will return the date in a long integer (in POSIX format) when the corresponding reference value was last updated. In NODAL this will be: AQN_VDAT.

When the call is made in a workstation, these calls are intercepted by the dispatcher and sent to the database instead of to the DSC. To the user this looks like a normal EM call, except

that the error returns may be different in case of malfunctions. ***The reference values can not be accessed when the call is made in the DSC itself.***

## Application Programs

The updating of all Oracle tables, except table REFVAL, can be done with a set of SQL*Forms, conveniently grouped in menus.

Storing parameters in REFVAL and writing them back to the EM can be integrated in the knob and working-set user interfaces. It should be possible to view the reference values and select some of them before sending them to the EM. Facilities should exist for copying values between PLS lines.

Browsing through the reference store, or updating it manually (if this is desired), is easily done with an Oracle Form.

Various reports can easily be produced with the report facilities of Oracle.

## Backups of REFVAL

Like all data, the reference store can get corrupted by writing bad data to it. Then we must either correct it manually or get an old copy back from a database export on previous Friday evening. It would be more practical to have these backups in the database itself. The size of REFVAL, for 5000 pieces of equipment, 1 pair of PPM parameters per equipment, and 8 PLS lines, is about 2 Mbyte. Storing GFAs and some instrumentation readings will double this and indexing the tables will double this again to, say, 8 Mbyte. Given the present price of disk space, we can afford to have several backups, e.g. for:

day 1, day2, day3, week1, week2, week3,

month1, month2, mont3, end-of previous-run . . .

With Oracle Forms, we can then compare the backups with REFVAL and, if necessary, restore subsets of the data. This should be seen as a maintenance tool and not as an operational facility.

## Machine Development

We can have a few machine development references (MDVAL). These MDVAL would store the parameters for a single user line. We can initialize MDVAL by copying portions of REFVAL for various user lines and elementary lines. We can then edit MDVAL manually and store it in one of the special user lines MD1 . . MD4 and try it out. If we are satisfied, we can copy MDVAL back to the REFVAL for one of the regular user lines. Here comes the delicate part: for equipment hooked to an elementary group, the userline will be converted to the corresponding elementary line and parameters for other users may be modified. To get a better idea of the consequences, we may want to view first all these modifications before going on. Database Forms are suitable for acquiring and displaying this information in any way desired by the user.

Particularly interesting MD sets can be saved as archives, on disk or on tape. This comes close to meeting the requirements for archives, discussed in ref. [18].

We mention this only as a possible future extension. It is not part of this proposal for implementing the reference values.

## Comparison with Existing Mechanisms

It is useful to compare REFVAL the mechanisms in the Historical Overview:

- ***Save and Back*** comes close to REFVAL but there are essential differences: REFVAL is on disk instead of in volatile memory and we have now the chance to view and select the data before sending them to the EM.
- ***Data-Table Block-Save*** (for ACCs) and ***Selective Data-Table Save*** (for DSCs) are still useful mechanism for quickly restoring data after a crash and should be kept.
- ***Archives*** are not really part of this proposal but we have shown that archives can be a logical development of the REFVAL mechanism.

## Conclusions

What we propose in this note is first of all a very general way of storing reference values which adapts almost automatically to new equipment and new properties. Applications for using these reference values should be easy to use and well integrated with the existing data-driven operator interface but it is clear that this subject must be developed further. The proposed mechanism fills all the operational requirements and leaves room for further development.

## References


1. Working Set and Archives; P.Heymans, M.Lelaizant, C.Serre; Sep. 1979 (a description of the original implementation of the archives).
2. Programs for Saving and Restoring the ACC-Data Tables; P.Skarek; A.P. Appendix A.32; Mar. 1982
3. Description du Logiciel de l'Arbre "Starting Up"; L.Merard; PS/CO/Note 83-20; Jul. 1983 (a formal description of the archives in the PS control system).
4. How Save is DAT-SEG-SAVE ?; P.Skarek; PS/CO/WP 84-020; Feb 1984 (doubts about the safety of data-table save and restore).
5. Coherence of the Datatables and related Problems; E.Malandain, J.P.Potier; PS/CO/WP 84-025; March 1984 (guidelines for save and restore of data-tables).
6. Meeting on the Archive System; PS/CO/Min 86-094; Nov. 1986 (discusses difficulties with the archives and first suggestion to use Oracle).
7. Selectif Restore des EM Data; PS/CO/Min 87-32; Jul. 1987 (meeting on storage of EM data in files during the night).
8. Remarks on the Performance of the Archive System; Cl.H.Sicard; PS/CO/WP 88-014; Jun. 1988 (proposes to improve the archives with a better data structure).
9. Towards a systematic Management of the PS Machine Parameter Data; T.Risselada; Jun.1988 (a proposal to implement pure virtual accelerators with only user PLS lines).
10. Compte Rendu de la Reunion No 100 d'Aspects Operationels; PS/OP/Min 88-23; Jul. 1988 (discussions about the requirements of a new archive system).
11. Accelerator-Operation Archives; J.Cuperus, M.Lelaizant; Oct. 1988 (a proposal for archives based on Oracle and archive files).
12. Meeting on Archive System; PS/CO/Min 88-033; Oct. 1988 (discussion of previous paper).
13. Archivage des Donnees Machine; PS/OP/MB AWG11; Jun. 1990 (a summary of ideas by operation about archives).
14. Individual References; Memorandum PS/OP/jf; Apr. 1992 (short summary of operator requirements).
15. Proposal for a Reference Value Set on Oracle; J.Cuperus; Apr. 1992 (proposal to store sets of reference values in Oracle tables).
16. LIN92: References individuelles pour chaque Parametre: Proposition de Realisation; PS/CO/Min 92-21; May 1992 (some considerations towards the practical realisation of the reference set).
17. Property Tables; F.Di Maio; Oct. 1991 (tables indicating what to display or hook to a knob).
18. Compte rendu du N2OAS-11, PS-OP MIN 82-21, Aug 1992 (discussions about archives).

## *APPENDIX 1: Property Tables*

There are a lot of equipment modules and, for each EM, a lot of properties. A general purpose data-driven program needs some guidance to the set of EMs it has to handle and what the interesting properties are. A general display program, for instance, needs, for each EM, a list of interesting parameters to display and indications on how to display them. Such lists exist already for the display and knob facilities of our new control system [17]. It was already foreseen to move this information to Oracle. With some slight additions, this will also indicate which parameters to store in a reference set, and how. The proposed structure of the table is:

Table PROPLIST = {

| | |
|---|---|
| CLASSNAME + | name of the equipment module class |
| PROPERTY + | name of the property |
| CPID + | integer identifying the record |
| CA + | 'C'=control, 'A'=acquisition |
| ACPROPERTY + | corresponding acquisition or control property |
| PARKIND + | 'S'=Main-Status, 'V'=Main-Value, 'A'=Auxiliary |
| INCREMENT + | 'C'=Continuous, 'D'=Discrete |
| PARTYPE + | 'I'=Integer, 'R'=Real, 'P'=pattern |
| TITLE + | title on a display (max. 8 char) |
| FORMAT + | C format for converting value to string |
| CHECKMODE + | 'A'=Absolute, 'R'=Relative, 'E'=Equal |
| PATTERN + | 'Y' if value can be seen as a bit pattern |
| PPMFLAG + | 'Y' if property may be PPM |
| KNOBFLAG + | 'Y' if parameter is controlled by knob |
| DISPOS+ | Position, if in working-set display, else 0 |
| LOGPOS + | Position, if in varilog display, else 0 |
| REFPOS + | Position, if in REF display, else 0 |
| | |
| TOLPROP + | property for acquiring tolerance for checking |
| TRMPROP + | property for acquiring treatment code |
| DIMPROP + | property for acquiring dimension if compliant array |
| MINPROP + | property for acquiring minimum value |
| MAXPROP + | property for acquiring maximum value |
| UNITPROP | property for acquiring units string |

}

Notes:

- Only one pair of records per class can have PARKIND='S' (e.g. for properties CCSACT and STAQ) and only one pair can be of kind 'V' (e.g. for properties CCV and AQN) but several (pairs of) records per class can be Auxiliary.
- DISPOS .. REFPOS are of format 'rc' where r is the row number [1..9] in the display and c is the column number [1..9].
- TOLPROP . . UNITPROP may be virtual properties, implemented on the level of the dispatcher. This makes no difference for the application programs. They may be null if the information does not apply.

For each TRMPROP that is not null in table PROPLIST, there must be one record in table TRMCODES for each possible value of the treatment code:

Table TRMCODES = {

| | |
|---|---|
| CPID + | integer, identifying class+property |
| TRMCODE + | treatment code |
| PROPFLAG + | 'Y' if property is implemented |
| LABELNOS | list of LABELNOs, separated by commas, which are implemented for this treatment code. |

}

For each PARTYPE='P' in table PROPLIST there must be one or more records with information about the possible bitpatterns:

Table BITPATTERNS = {

| | |
|---|---|
| CPID + | integer, identifying class+property |
| LABCAT + | label category [1..n] |
| MASK + | HEX mask to be applied to parameter bitpattern |
| LABVAL + | possible HEX value of the masked bitpattern |
| LABEL + | corresponding label string for display (max. 8 char) |
| COLOR | color for display |

}

Notes:

- For LABVAL=-1 any acquired value is a match and VALUE $\wedge$ MASK is a real value and not a pattern.

## *Real-Time Database*

Read access to most of this information is possible through the real-time database functions (see /usr/local/include/rtdb.h for details). Relevant functions are:

- DbrtDispPos Return relevant properties and corresponding display positions for a given class and system.

- DbrtClassProp Return CPID and other information for a given class and property.
- DbrtTrmInfo Return treatment information for given CPID and code number.
- DbrtBitPattern Return meaning of bit patterns for a given CPID.

This gives no access to the reference set itself. Application programs will get this access through a library of database access routines, so that they do not have to use a pre-compiler for Oracle access.

## *APPENDIX 2: Equipment List*

The Property-Tables described above give information common to an equipment class. Some static parameters can be different for each piece of equipment.. The more general parameters are in table EQUIPMENT:

Table EQUIPMENT = {

| | |
|---|---|
| EQNAME + | equipment name |
| CLASSNAME + | name of the equipment class |
| MBNO + | member number in the class |
| ACCELERATOR + | accelerator name ['PSB'\|'CPS'\|'LIL'\|'EPA' ... ] |
| SUBSYSTEM + | relevant part of the accelerator ['RF'\|'INJ'\|'EJ' ... ] |
| DSCNAME + | name of DSC |
| EQDESCRIP + | short description of the purpose of the equipment |
| UNITS + | unit string for parameter with PARKIND='V' |
| PLSMACHINE + | PLS telegram name: ['PSB'\|'CPS'\|'LPI'] |
| PLSGROUP + | group for PPM, if any |
| . . . | other information, not relevant here |

}

Static parameters which are only relevant to a subset of the EMs are in table INSTVAL:

Table INSTVAL = {

| | |
|---|---|
| CLASSNAME + | name of the equipment class |
| MBNO + | member number in the class |
| VARNAME + | name of the Read-Only variable |
| VALUE | value of the RO variable |

}

This information can be read by application programs through EM properties or through the real-time database function DbrtEqMembInfo.

## GROUPE PS/CONTROLE

NAME
ADORNI Valerio
ARRUAT Michel
ASSOR Jean-Luc
BENETTON Maurice
BENINCASA Gianpaolo
BERLIN Fridtjof
BRESSANI Ginevra
BOBBIO Piero
BOUCHE Jean-Marc
CLOYE Jean Jacques
CROTEAU Pascal
CUISINIER Gérard
CUPERUS Jan Hendrik
DAEMS Gilbert
DAVIDENKO Yuri
DEHAVAY Claude
DELOOSE Yvan
DE METZ-NOBLAT Nicolas
DI MAIO Franck
GAGNAIRE Alain
GAYRAUD Christine
GIOVANNINI Fernando
GIUDICI François
HEINZE Wolfgang
IOURPALOV Vladimir
IZGARSHEV Serguei
KIRK Michael
KNOTT Gisèle
KUIPER Berend
LELAIZANT Monique
LEROY Christine
LEWIS Julian
LUSTIG Hans-Dieter
MERARD Lucette
MIKHEEV Mikhail
PACE Alberto
PEARSON Toby
PERRIOLLAT Fabien
PETTERSSON Thomas
PHILIPPE Jean
POTDEVIN Philippe
RAICH Ulrich
REDARD Jacques
RISSO Alessandro
SERRE Christian
SICARD Claude-Henri
SIKOLENKO Vitaly
SKAREK Paul
TROFIMOV Nikol
ZELEPOUKINE Serguei

15.09.92

GROUPE OP

G. ADRIAN
D. ALLEN
M. ARRUAT
G. AZZONI
S. BAIRD
N. BLAZIANU
J. BOILLOT
X. BRUNEL
J.C. CENDRE
E. CHERIX
E. CHEVALLAY
J.J. CLOYE
G. CYVOCT
D. DAGAN
I. DELOOSE
B. DUPUY
J. DURAN-LOPEZ
J.M. ELYN
T. ERIKSSON
P. FERNIER
A. FINDLAY
B. FRAMMERY
V. GARRIC
D. GUEUGNON
G.-H. HEMELSOET
R. HOH
G. JUBIN
J. KUCZEROWSKI
F. LENARDON
G. LEO
B. L'HUILLIER
T. MALMEDAL
B. MANGEOT
D. MANGLUNKI
J.L. MARY
G. MERCER
G. METRAL
H. MULDER
A. NICOUD
J.M. NONGLATON
J. OTTAVIANI
E. OVALLE
S. PASINELLI
M. PERFETTI
K. PRIESTNALL
Y. RENAUD
M. ROCHE
M. RUETTE
C. SAULNIER
J.-L. SANCHEZ-ALVAREZ
Ch. STEINBACH
G. TRANQUILLE
B. VANDORPE
V. VICENTE
E. WILDNER