OPERATIONAL SOFTWARE FOR MULTICYCLING THE CERN SUPER PROTON SYNCHROTRON

R.Bailey, J.Ulander, I.Wilkie
SPS Division, CERN
1211 Geneva 23, Switzerland

For commissioning and subsequent operation of LEP, the SPS is required to perform as an injector to the lepton machine while continuing to provide proton beams for the physics program. This has to be achieved by handling the different beams in a repetitive mode.

In order that the control system can meet this demand, we have developed generalised high level applications to allow independent control of the different SPS beams. The development has been driven by the operational requirements with the result that the new software imposes to a large extent the way in which the SPS is operated.

We discuss our development philosophy, the important functionality of the applications software and the implications for operating the accelerator.

The Software Development Approach

Requirements Specification

In mid-1985 there already existed a large body of applications software for controlling the SPS. Although these applications had become very complex and were largely undocumented they worked, and so had to be performing the necessary functions in some way or another. We therefore took this software as the starting point in our requirements specification.

We analysed the main features of the existing software and extracted the essential processes that were involved, removing details that were due purely to the implementation on our software environment. We then formed a "template" system that was a super-set of all the features present in the existing software. After ensuring that all the existing SPS systems could be controlled by such a system we extended it to cope with the problems imposed by a multicycling machine.

We produced in this way a model of the major functionality of the new system. This model was then developed in great detail, following a top-down decomposition in order to control the complexity while keeping an integrated system. We used Structured Analysis (SA) to describe the processes involved and the data flows between them, and all data structures were rigorously defined [1].

Among the main features of this model were :

- The update of the SPS from an essentially single cycle machine to a multicycling machine.
- The ability to re-use individual cycles in many different supercycles.
- The rationalisation of the best features of the existing software.
- A common operator interface for all the traditional SPS systems.
- The ability to restore the SPS to a known state "at the touch of a button".
- The ability to change SPS supercycle in a very short space of time (one or two SPS supercycles).

Design and Implementation

An important stage in the transition from analysis to design is to partition the model into different tasks that can then be developed independently of each other. We divided our model into the major functions performed on the accelerator;

- Supercycle generation
- Supercycle management
- Send settings to hardware
- Trim settings
- Measure beam and hardware parameters

These different packages are quite general, being data driven to accommodate all systems of the accelerator.

Having a clear division of the tasks enabled us to concentrate on the parts of the system that were needed early. Generating supercycle settings was such a task, which we were able to specify sufficiently well for a small team of people to develop it separately [2]. Being able to send settings to the hardware and subsequently be able to trim them were also among the first things required of the new software, and we undertook to develop these.

We used Structured Design to produce a series of connected structure charts. All modules were described in pseudo-code and all data couples between modules were specified.

Producing code from these structure charts then proved to be straightforward. Modules could be coded more or less on an individual basis, since all analysis and design considerations had been taken care of. Tested modules were built together into more and more complex programs in a controlled way.

Functionality of the Software

Supercycles and Segments

A supercycle consists of several individual cycles. Since considerable time is spent trimming the machine to achieve the best beam conditions, it is desirable to be able to re-use an individual cycle, say for Fixed Target Proton work in both a LEP filling supercycle and a completely different supercycle.

In order to meet this requirement we have divided the individual cycles time-wise into 2 segments:

Beam Segments cover the time during which all systems must act together to maintain beam in the SPS.

Preparation Segments cover the time during which each individual system moves as fast as possible to a stable state ready for the next injection of beam.

A beam segment then has an existence independent of any specific supercycle. Beam segments can be trimmed and adjusted and these changes will be reflected wherever the segment is used.

Preparation segments, however, exist only to join beam segments together in particular supercycle combinations. They have no other independent

existence. Preparation segments are thus not re-usable in any way, but this is not a significant loss since little or no time is usually invested in optimising them.

Modelling and Trims

The existing software for individual SPS systems often adopted the approach described below in controlling a machine system. In some cases the ideas were heavily obscured by implementation details but this scheme was the logical basis of the control.

Consider the example of the "Main Power Supplies" (MPS). This system is responsible for control of the main bending magnets and the main focussing quadrupoles. In the past an MPS "cycle" was generated by taking input parameters such as the beam injection energy, the extraction energy, the required betatron tune and limitations of the power supplies. From this starting point a set of functions was generated that described the required currents in the magnet systems as a function of time.

It was known that these generated functions were a good approximation to the necessary values but that they would have to be adjusted by measuring the response of the particle beam itself. In the case of the main quadrupoles the corresponding beam quantities are the horizontal and vertical betatron tunes. Trims would be made, by the operator, so that the measured betatron tunes of the beam would agree with the desired tunes (as had been specified to the software that originally calculated the functions).

These trims were made in units that the operator was familiar with (in this case tune values) and converted by the trimming software into units suitable for the hardware (in this case Amperes).

Sometimes, however, trims were made for a different reason than the matching of the beam response with the desired parameters. Rather the trim was done because of a change of the desired parameter itself. A typical example was the raising of the horizontal betatron tune during the time that the beam was extracted. Trims made for this reason were indistinguishable, in the existing software, from those made for the reason first described.

There was also a third category of adjustment made. Sometimes power supplies will not deliver to the magnets chains the current or voltage demanded of them. This is because of intrinsic shortcomings in the power supplies and magnet chains. In these cases the demanded value would be adjusted before being given to the supply so that the values produced by supply was what had been originally requested.

The Data Hierarchy

We decided to rationalise the chain of data described above and for each system introduce a hierarchy of functions. The hierarchy consists of a series of levels running from demanded physics parameter at the top, down to the data used to drive the accelerator hardware at the bottom. At each level the function is derived from the function above it in the hierarchy.

The levels are as follows :

The Segment Descriptor (SD) defines the desired parameter of the beam. An example is the horizontal betatron tune.

The Control Variable (CV) enables the trims to b made to make the beam achieve the desired Segmen Descriptor Parameter. Note that this can either be i "Physics" units or "Hardware" units for any give system depending on the state of the models and also o operational preferences.

The Hardware Function (HW) is for example th current in a magnet that should achieve the desire parameter. It is derived from the Segment Descripto by means of a model or via empirical trimming of th control variable.

The Compensated Hardware Function is a correcte function with the defects of the power supplie included. This ensures that the supplies deliver th values that are required for the beam.

Because it is often the case that one hardwar element (such as one of the two main quadrupole sets actually affects more than one desired parameter of th beam (in this case both the horizontal and vertica tune), we found it necessary to make groupings of thes levels of functions and introduce both Segmen Descriptor and Control Variable Sets.

Implications for Trimming

We allow trimming at each level in the hierarchy. This suggests that the connection between each level o the data is known. That is to say, ultimately al hardware functions can be derived from the segmen descriptors. In the existing software this is no always the case. Hardware functions are sometime constructed and trimmed "by hand" until the bea reaches its desired state.

In such cases we intend to keep the data structur and use the segment descriptors simply as a referenc for the operator or for surveillance software. Usin the data structure we describe also means that a software is written that can calculate the necessar functions, it can quickly be applied to the data.

For systems where the derivations can be calculate by software, this is used to calculate all the origina functions. In addition this software calculates series of gradients that describe the relation betwee the SD and the CV and between the CV and the H functions. These gradients can be picked up b generalised trimming software that can then perform al the necessary function adjustments without bein written specifically for each system. In a complet hierarchy for a given Segment Descriptor Set then would be gradients linking each Segment Descriptor t each Control Variable and linking each Control Variabl to each Hardware Function. Some of these gradient might well be unity.

For systems where the relationship between level cannot be adequately defined by gradients, we hav provided the facility to "plug in" specific modules t calculate the lower levels in the hierarchy.

Finally, all trims made on a beam segment ar stored and time stamped. This allows the possibility t go back to any previous state of any system at an level in the hierarchy. A combination of suc operations will enable the whole machine to be returne to some previous state.

Trims in a Supercycle

A major problem with the idea of re-using bea segments is that there will be effects seen in the bea of a particular beam segment due to its position withi a specific supercycle. These are eddy current an

remanent field effects due to the magnetic behaviour of previous segments in the supercycle. We have therefore added a second class of trims to keep adjustments that are made to compensate for these effects separate from normal trims. We call these special trims "fixit" trims. Such trims are associated with a particular beam segment and a particular position in a specific supercycle. Keeping these trims separate allows them to be applied to the machine only when it is appropriate.

Rapid Supercycle Changes

A loaded supercycle is actually run by events issued by a Fast Broadcast Message Timing System [3]. When a supercycle is loaded the software prepares timing tables for this timing system to use. Any new supercycle can be prepared and loaded asynchronously into spare memory in the hardware . The action of making the new supercycle run then consists only of instructing the timing system to send out messages from a different event table. This ensures that switching between supercycles can be very fast.

Current Status of the Software

The system is only partially implemented at the moment. The needs of machine developments sessions and the running of the SPS as before but with the new control hardware have meant that some temporary software has been written.

However, the base level of the system is in place and several major parts have been implemented :

- Supercycle Generation for the settings of main systems
- Implementation of the Data Stores.
- Function trimming and recall of past trim states.
- Most of the Hardware Control Functions necessary for running a single supercycle are in place.

At present the software can control the following SPS systems :

- Main Bending and Focussing magnets
- Radio Frequency system functions
- Chromaticity Sextupoles
- Skew Quadrupoles
- Harmonic Correctors

Implications for Operations

One of the first things that we noticed as the software came into operational use was that the users did not fully grasp the significance of our data hierarchy. Understanding this is an essential prerequisite to exploiting the software functionality to the full. Consider the commissioning of a new accelerator system; settings are generated and loaded and measurements are made. The order in which the various trims are then applied is important.

The first trims to make are Hardware Compensation trims, to ensure that the measured hardware values are equal to the demanded hardware references. One can then be sure that when one looks at the beam, any discrepancy between measured beam parameters and desired physics values does not come from inadequacies in the power supply. In fact, any such discrepancies must come from weaknesses in the model used to generate the settings in the first place, and these should be corrected using Control Variable trims. Segment Descriptor trims are only ever needed to cause a change in the desired physics parameters of this system.

While applying trims in the above way, one has to separate trims that are particular to that segment from fixit trims that are to compensate for remanent effects from other parts of the supercycle. This separation is in fact difficult to make. The only way to be sure is to trim a segment that is in a simple supercycle consisting of just this segment and a preparation segment. If this trimmed segment does not produce the same beam conditions when used in a more complex supercycle, the new trims required are of the fixit kind; they must be to compensate for effects particular to the supercycle makeup. While it may not be possible to create the simple supercycle mentioned above, it is possible to arrange for the segment to be trimmed in a place free of large remanent effects. One can then be reasonably sure that the trims being made really do apply to that segment in their own right.

These conceptual difficulties aside, we have received a fair amount of feedback on some of the features of the software.

The fact that the trims are stored means that it is possible to go back to any previous machine state. We have provided the means to go back either a certain number of steps or to a particular trim. This facility in fact makes a separate archive facility unnecessary, although this puts a heavy responsibility on the integrity of the data stores.

It has proved very useful to be able to copy settings from one part of a supercycle to another. Copying of this kind can be either for a whole segment, a single segment descriptor set, or the trims of a segment descriptor set. It is in this way that we have effectively re-used segments in the same supercycle.

All accelerator systems running under the new software do so through a single interface. Furthermore, introducing a completely new accelerator system does not require new code except for the cycle generation part. The rest of the software simply needs the necessary data to drive the new system and it will work in the same way as all the other systems do.

On the negative side the most serious problem is one of speed. It currently takes too long to perform somewhat routine operations on the accelerator; a trim to the machine chromaticity throughout a whole segment can take as long as one minute to get into the hardware. Some of the slowness undoubtably lies in the fact that we have produced generalised software, but there are also problems in this respect with the environment on which our software runs. We hope that by improving certain aspects of the environment that things can be considerably improved.

References

[1] R.Bailey, J.Ulander, I.Wilkie "Experience with SASD for production of Accelerator Control Software" Proceedings of the Europhysics Conference on Control Systems for Experimental Physics, Villars-sur-Ollon, Switzerland, September 28-October 2 1987

[2] K.Cornelis, W.Herr, J.Reid, R.Schmidt, "The Generation of the SPS Supercycle" Proceedings of the Europhysics Conference on Control Systems for Experimental Physics, Villars-sur-Ollon, Switzerland, September 28 - October 2 1987

[3] C.G.Beetham, R.J.Lauckner, C.Saltmarsh "Overview of the SPS/LEP Fast Broadcast Message Timing System" Proceedings of the 1987 Particle Accelerator Conference, Washington D.C., March 16-19, 1987