

Cavity Conditioning

Local application for new Linac
Thu Sep 24, 1992

Introduction

Conditioning the new Linac rf cavities will take hours or days for each module. This local application is an implementation designed to automate the procedure so that it can run unattended. The main idea is to slowly ramp the rf peak forward power from the klystron toward a target value, as long as the radiation remains below a threshold, taking care to watch the vacuum pressure and the spark rate, reducing the forward power if either exceeds specified levels, and resetting any rf system trips. The implementation of the program is the subject of this note. The user interface is provided by the Parameter Page.

Hardware signals (klystron system 3)

<i>Description</i>	<i>Name</i>	<i>Units</i>
1. rf peak forward power A/D reading	K3WG1P	MW
2. rf peak forward power D/A setting	"	MW
3. vacuum pressure A/D reading (2)	V3VP1 , V3VP2	V
4. spark digital status bit reading		
5. rf "on" digital status bit reading		
6. rf interlocks reset digital control bit pulse		
7. rf system reset digital control bit pulse		
8. clock event status bits		

Software parameters

<i>Description</i>	<i>Name</i>	<i>Units</i>
1. enable status/control bit for this application		
2. rf peak forward power target value	K3TRGP	MW
3. rf peak forward power delta	K3DLTP	MW
4. time interval delta	K3DLTT	SEC
5. spark rate threshold	K3SPKT	%
6. vacuum pressure threshold	K3VACT	V
7. rf peak power back-off percent	K3PPBK	%
8. maximum #resets of trips	K3MAXT	
9. spark rate output value	K3SPKR	%
10. delay after back-off due to vacuum	K3VACD	SEC
11. maximum #sparks to compute spark rate	K3MAXS	
12. status bit for program state 0. Waiting for recovery.		
13. control bit to clear spark-counting statistics counters		

Local application support

As a local application, the code is called as a Pascal procedure by a special entry in the Data Access Table of the local station. This entry causes each local application residing in the system Local Application Table be called by name. The 4-character name is used to search the CODES table of programs that have been previously downloaded by name into non-volatile memory. The first argument of the call is a byte whose value identifies the type of call:

0: Initialization call. Allocate and initialize static memory used during the time the local application is enabled.

1: Termination call. Free static memory allocated by Initialization call.

2: (not used)

3: Cycle call. Process with new data in data pool.

4: Network call. Message received for this application.

The second argument is a pointer to the 12-word parameter area of the Local Application Table entry. The first *longword* of this area provides storage for the pointer to the static memory allocated during the Initialization call. The next *word* is the enable Bit#, and the remaining words are used for additional parameters, usually specified as Channel#s and Bit#s. (See the layout for this application in a later section.) When the enable bit is set, the application is enabled. When it is clear, the application is disabled. The system notices changes in the state of this enable bit and schedules Initialization and Termination calls accordingly. When there is no change in the enable bit, *and the bit is set*, the application receives a Cycle call. Special logic is included that provides for automatic replacement to a new program version as soon as it is downloaded, if the application is enabled. A local application is downloaded into non-volatile memory but executes out of on-board ram. A checksum is kept for the downloaded version that is verified each time an application's enable bit changes from a 0 to a 1 and its code is copied into allocated ram for execution.

State flow

Local applications of the closed loop style are typically implemented with state logic. In this case of the cavity conditioning application, there are two states: 0 and 1. When the application is first enabled, state 0 is asserted. While in state 0, the application looks for a valid set of readings (both hardware and software) and also for the rf system to be "on". Constants in the program are used to assess whether the values of the hardware and software parameters are within "reasonable" ranges. Once these conditions are satisfied, the program switches to state 1.

In state 1, the time delta value is used as a period over which to determine a maximum value of the rf peak forward power readings. At the end of the time interval, the maximum is compared with the target value to determine whether an adjustment of the peak power delta can bring the power closer to the target value. Independent of this time delta interval, the maximum #sparks parameter is used to form a spark period interval over which the spark rate is calculated. The spark rate is checked against the threshold value to decide whether to back off. Vacuum is always checked against the vacuum threshold to decide whether to back off. And the rf "on" status is always checked to detect rf system trips.

Statistics are maintained about the relationship between sparks that occur and the relevant Tevatron clock event signals that indicate what kind of accelerating cycle was active. Counts of events, sparks occurring on those events, counts of sparks that occur during which Booster batches, and a histogram of spark occurrences throughout the supercycle. A self-clearing control bit is provided that, when set, clears the spark counting statistics.

All software input parameters (except the maximum #trips) can be modified during normal state 1 operation to take effect after the current time interval. Also, the peak forward power can be changed manually by knob control even while the conditioning program is regulating the peak power, as the changes made by the algorithm are always applied incrementally from the current setting.

A "state 0" status bit is provided as an output so that it can be monitored by the alarm system to announce when the application is no longer regulating.

Logic details

The response to an rf system trip is to first back off the forward power, reset the rf interlocks and subsequently to reset the rf system itself. Such resets of rf trips are limited to repeat no more often than 10 seconds, a program constant.

There are two vacuum readings. The one with the worst reading is used in the algorithm because only one vacuum pump may be required to be running, and the reading of a pump which is off appears as “excellent” vacuum. The response to poor vacuum compared to the vacuum threshold value is to back off the forward power and delay for the vacuum delay time before allowing another back-off due to vacuum. This gives the vacuum system time to approach equilibrium under operation at a reduced peak forward power level.

The spark rate is computed over the number of 15 Hz cycles required to accumulate the number of sparks specified by the maximum #sparks parameter. Expressed as a percentage, it is compared against the spark threshold.

When there have been more rf system trips than that given by the maximum #trips parameter, the application reverts to state 0. Manual recovery of the rf system will allow a return to state 1 processing with an additional maximum #trips permitted—assuming nothing else is wrong.

The application was developed using MPW Pascal on the Macintosh to take advantage of its support for the floating point 68881 chip, as most of the logic in the program is based upon engineering units values. Its current 400 lines of source code run in less than 3K bytes.

Parameters layout

The layout of the parameters area of the Local Application entry as viewed by the Local Application Parameter Page is as follows:

```

E LOC APPL PARAMS 09/24/92 1711
NODE<0623>  NTRY< 5>
NAME=COND  CNTR=00F8
TITL"LGAL RF CAV CONDITIONING"
SVAR=00000000
ENABLE  B<02A0> COND ENABLE
SPARKS  B<019E> REFLECTED POWER
RFONST  B<0198> RF ON IS ENABLED
RFINTLK B<015C> INTERLOCK RESET
PPWR    C<0480> K3WG1P      MW
VACUUM  C<049E> V3VP1      V
        <0000>
RFRESET B<0325> SYSTEM RESET
EVENTS  B<0230> EVENT 18
OTHERS  C<0490> K3TRGP      MW

```

The first parameter word specifies the enable Bit# that must be set to enable the local application program to run. Setting it to a “1” enables calls to be made to the application, beginning with the Initialization call. Setting it to a “0” schedules a Termination call before releasing the program’s execution memory.

This application uses enable Bit#+1 for the “state 0” output bit and enable Bit#+2 for

the control bit that clears the spark counters.

The lo byte of the CNTR word is merely a diagnostic count of the number of times the application code is called. It serves to show evidence of obvious activity when viewing this entry on a memory page display. The hi byte of the same word shows the elapsed time required by the last 15 Hz program invocation.

The ptr to the application's static memory, established as a result of the standard Pascal New procedure used for dynamic memory allocation, is stored during Initialization call processing for use during subsequent Cycle call processing. One can gain some diagnostic insight of the application's activity by observing the contents of the memory block pointed to by this address with the Memory Dump Page application. The first 8 bytes are used for a standard memory block header. The rest of the block can be matched with the declaration of the static variable data structure of the application.

The last 9 words are used for hardware Chan and Bit#s. The final word is a base Chan# of the sequence of Chan#s used for the software application parameters.