Embedded Software for the CEBAF RF Control Module*

G. Lahti, I. Ashkenazi, C. West, B. Morgan
Continuous Electron Beam Accelerator Facility
12000 Jefferson Avenue
Newport News, VA. 23606

ABSTRACT

The CEBAF accelerator control system employs a distributed computer strategy. As part of this strategy, the RF control sub-system uses 342 RF Control Modules, one for each of four warm section beam forming cavities (*i.e.*, choppers, buncher, capture) and 338 superconducting accelerating cavities. Each control module has its own microprocessor, which provides local intelligence to automatically control over 100 parameters, while keeping the user interface simple. The microprocessor controls analog and digital I/O, including the phase and gradient section, high power amplifier (HPA), and interlocks. Presently, the embedded code is used to commission the 14 RF control modules in the injector. This paper describes the operational experience of this complex real-time control system.

REQUIREMENTS

There are seven major requirements for the embedded software: (1) low power RF control, (2) high power RF control, (3) interlocks, (4) system calibration, (5) module hardware configuration, (6) manual and automatic control, and (7) diagnostics. These requirements are described as follows.

(1) Low Power RF Control

The low power RF control is handled by hardware internal to the RF control module. The phase and gradient loops are handled by this hardware, because of the speed needed. The embedded software's function is to set up the operating point of the hardware. This point can be determined manually (externally by the operator or high level control software) or automatically (internally), or a combination of both. A manually controlled parameter would be the gradient set point. An automatically controlled parameter would be the gradient clamp voltage, that varies with the gradient set point. Measured values are also monitored and sent to the higher level computer. The interface to this computer has been simplified. The transferred values are in convenient units, rather than base units or DAC/ADC bit patterns (*e.g.*, the RF module DACs and ADCs can be changed without changing the computer interface).

Some of the low power control signals are:

Phase
    a. phase set point
    b. measured phase
    c. phase locking phase offset
    d. detector error
    e. amplifier gains and frequency response
    f. modulator bias
    g. reference oscillator power
    h. open/closed loop switch
    i. detuning angle of the cavity

Gradient
    a. gradient set point
    b. measured gradient
    c. clamp modulator drive
    d. detector error
    e. amplifier gains and frequency
    f. offset drive
    g. quench detector
    h. open/closed loop switch

Output
    a. RF attenuator
    b. output switch

(2) High Power RF Control

The high power RF is handled by both the HPA hardware and the RF module hardware. The filament and high voltage switches and some fault flags (bits) are handled by the HPA hardware and are set and measured by the upper level computer. The RF module handles the following HPA signals:

    a. filament voltage set point
    b. filament measured voltage
    c. cathode current
    d. body current (klystron)
    e. mod anode voltage set point
    f. mod anode measured voltage
    g. forward and reflected power monitor
    h. HV series relay to disable HV

(3) Interlocks

Interlocks are needed to keep the hardware in a safe condition. Some interlocks are handled in the hardware. The rest are handled by software, and are split into two groups: fast and slow response. Fast means within 150 μsec, and slow is measured in seconds. Detecting an interlock fault ranges from simply checking a hardware bit to checking a function of one or more analog signals. After an interlock fault is found, there can be various actions to take: (1) turn off RF, (2) turn off high voltage, (3) turn off filaments, or (4) a combination of these but wait at each stage to see if the fault goes away. In addition, some interlocks need the fast shutdown line pulled.

Some of the signals to check for interlock faults are:

a. excessive reflected power
b. excessive cathode current
c. excessive body current
d. arc detector trip
e. watch dog to verify computer activity

(4) Calibration Correction

There are two types of calibrations that affect the RF control module: (1) hardware variations internal to the module and (2) external hardware variations. The calibration coefficients are to be stored in the module's memory. This allows the internal coefficients to move with the module (e.g., from warehouse to accelerator). When a module is moved to a new location in the accelerator, only the external coefficients need to be transferred to the module. Some coefficients are fixed, while others depend on temperature. The use of calibration correction implies the following: (1) a means to determine these coefficients, (2) a means to transfer these coefficients, and (3) a real-time method to use these coefficients efficiently.

External coefficients can be handled by typical measurement techniques (e.g., measure the cable attenuation). For the internal coefficients, a test-stand is needed to exercise the RF module. This requires that the embedded software assist in this operation, which implies the following changes from the normal accelerator mode of operation. (1) Allow direct access to the DACs and ADCs (e.g., phase set X & Y rather than simply the angle), (2) the interface values are voltages rather than usual units (e.g., watts, amps, etc.), and (3) do not modify (i.e., correct) these voltages.

These coefficients are transferred to the module via a download mechanism. This includes standard error checking. In addition, it would be helpful to have an upload feature to see if the downloaded values were loaded properly. As part of the total picture, other features are needed: (1) transfer coefficients from the test-stand to some database such as INGRES as a backup, (2) create download files from INGRES, and (3) create any version of download file (e.g., present values or last month's).

To efficiently use these coefficients, the algorithms should do as much pre-computation as possible, so that the normal run time computations are as small as possible. This is especially important when the coefficients depend on very slow changing signals such as temperatures. When the temperature changes enough to make a difference, then a mechanism is needed to execute the pre-computation procedure during run time while affecting the normal operations as little as possible.

(5) Module Configuration

It is desirable to have only one version of the embedded software for all applications. There are six applications:

a. chopper section
b. buncher section
c. capture section
d. quarter cryo section
e. full cryo section
f. test stand

(6) Manual and Automatic Control

As mentioned above, in normal operation most signals are under automatic control. This is desired so as to simplify the interface to the upper computer (i.e., distributed control strategy). However, during development or debug operation, a manual mode of operation is needed. This manual mode is not simply full manual mode for all signals; that would cause most debugging to be too complex. So, various degrees of manual operation are needed (i.e., allow varying percentages of signals to be under manual control, while the rest are automatic). The switch between automatic and manual modes must be "smooth" so that no glitches are introduced into the hardware.

(7) Diagnostics

There are two types of diagnostics: active all the time and active on command. The first type is basically a display of all important software control words (bit flags). The second type can be activated through normal data flow channels or via a secondary path (such as a RS232 port). As part of this second type, a specialized "peek & poke" could be used to access all major signals and control variables.

IMPLEMENTATION

The total RF control system consists of three supervisor computers (HP835s), one each for the injector, North Linac, and South Linac. These computers are used mainly for display purposes in the control room. However, they do contain some logic that is global to that one subsystem. Under these three computers are the locals, which reside in the service buildings. Next come the RF module microprocessors.

Injector: 1 supervisor,   5 locals (only 2 for RF),
              22 microprocessors.
North Linac: 1 supervisor, 10 locals, 160 microprocessors
South Linac: 1 supervisor, 10 locals, 160 microprocessors

Supervisor to supervisor communication and supervisor to local communication is via LANs. The local to microprocessor communication is via CAMAC. The local can also communicate with other CAMAC devices, ranging from simple DACs/ADCs/switches to more complex devices (e.g., cavity tuner motors).

The local to microprocessor path consists of 32 input channels and 32 output channels for normal communication and various other specialized CAMAC commands. Even with this number, the outputs from the RF module have to be multiplexed because more than 32 channels were needed. For critical signals, error detection is employed.

The software structure is based on a state machine. There are seven basic states: null, test-stand, download, idle, filament, high-voltage, and RF-on. Depending on the state, the various signal algorithms perform specific actions

1291

particular to that state. The signal algorithms are also allowed to request a change to a lower state (*e.g.*, when an interlock fault condition is discovered). This allows more modular construction, since one signal can only affect a major change in another signal via a state change. For example, if the reflected RF power is too high, the state is changed from RF-on to HV; then the RF output switch is opened because the the RF-on state is no longer active. Minor changes are allowed by passing the output of one signal routine to the input of another routine. For example, the cable temperature sensor routine measures the temperature, and the other routines can use this as input to modify their calibration coefficients.

There is very little overlap of responsibilities between the local computer and the RF microprocessor. If the microprocessor has primary responsibility for a particular signal, it handles all aspects: reading the signal from hardware or CAMAC, calibration correction, fault and interlock detection, outputting the signal to CAMAC or hardware, and outputting the fault flag. The local computer will display the signal and fault flag, usually without secondary processing. In other words, the microprocessor will do as much as it can, from signal processing to fault detection.

OPERATIONAL EXPERIENCE

The RF module embedded software is written in the C language, except for the boot code which is in assembly language. The early development was done on one of the supervisor computers in a UNIX environment (HP835 computer). An in-house CAMAC interface emulation set of routines was used to join the local computer TACL (accelerator computer control system) logic with the RF module logic, while running both logics on the one HP835 computer. This allowed us to test both logics with very little (or no) change to either in this emulation environment. Two screens were used, one for the normal TACL display and the other for the RF module debug print statements. At this stage we were able to catch most logic errors.

The next stage was to port the embedded code to the target microprocessor (INTEL 80186). The INTEL development station was used to debug the hardware dependent features. There were not that many errors at this stage.

The final development stage was to transfer the code to the EPROMs and run the module with no development station emulator. This procedure went well. The C language code ported well. The following problems were noted, but they were generally minor.

a. HP and INTEL compilers have some minor differences. The INTEL compiler tended to be more strict in its checks, probably because it is one of the newer ANSI standard versions.

b. Some differences were also due to the different architectures of the processors. The INTEL 80186 uses a segmented memory.

For small changes to the code, we generally skip the UNIX emulation stage. But for new and complex features, we start development on the UNIX system.

The complexity of the software interface to the hardware is reduced by special hardware features. Instead of the software triggering an ADC and waiting for the value, the hardware contains a sequencer to trigger and read all ADCs and put the values into mapped memory. The software simply reads that memory when it needs it. The CAMAC input/output is also memory mapped. This ADC sequencer can also be set up to sample one particular signal at a software selected sampling rate and number of samples, and the hardware maps these sampled values to an array in memory. The software only needs to operate on this array (*e.g.*, digital signal processing).

The time to process one logic cycle in the RF module is about 33 msec (30 Hz). This time is for an RF module that uses the math coprocessor. With no coprocessor, the time increases to about 500 msec.

The time to process the major part of a fast shut down interrupt ranges from 50 to 70 μsec.

CONCLUSION

The RF module software design followed object oriented design guidelines. However, the implementation used the standard C language. Code check out went quite well, and most new features have fitted well into the basic structure.

Our in-house UNIX emulation of the interface between the TACL computer and the RF module microprocessor aided greatly with early design and implementation. We could use the same UNIX tools that we used to develop the TACL system.

The INTEL development system (with microprocessor emulator) was a necessity when it came to finding and fixing hardware/software errors.

The RF module software performs well in reducing the computation burden of the local computers and in modularizing the RF control system.

1292