CONF-930401--8

# EMBEDDED COMPUTER SYSTEMS FOR CONTROL APPLICATIONS IN EBR-II

ANL/IFR/CP--77128

DE93 010114

Reed B. Carlson, Steven E. Start
Integral Fast Reactor Operations Division
Argonne National Laboratory
P.O. Box 2528
Idaho Falls, Idaho 83403-2528

## ABSTRACT

The purpose of this paper is to describe the embedded computer systems approach taken at Experimental Breeder Reactor II (EBR-II) for non-safety related systems. The hardware and software structures for typical embedded systems are presented. The embedded systems development process is described. Three examples are given which illustrate typical embedded computer applications in EBR-II.

## INTRODUCTION

An embedded computer system in EBR-II is typically a general purpose microprocessor based computer system that runs firmware based software. They are installed as part of plant systems and provide real-time control, monitoring, data conversion, and input/output (I/O) front end functions. Embedded systems of the type described in this paper have been designed, installed and used in EBR-II for more than a decade. The embedded systems approach taken uses a standardized hardware bus structure, software structure and software engineering approach and allows incorporation of current technologies.

The embedded computer systems in EBR-II are STD bus based and consist of a card cage, a CPU board and a complement of I/O boards. The CPU board has a CISC microprocessor which runs the application software. The I/O boards provide the interface between the plant sensors and actuators and the application software. The systems do not use hard or floppy disks. To date, none of the installed systems use operating systems or executives. Each system may or may not have an attached video terminal as the normal operator interface. The computer systems are usually installed in the equipment cabinets of particular plant systems and are not readily accessible.

The embedded system application software consists of real-time data acquisition and control functions,

* Work supported by the U.S. Department of Energy, Reactor Research Technology under Contract No. W-31-109-ENG-38.

serial communications handlers, and operator and maintenance interfaces. The software is comprised of real-time foreground functions and non-real-time background functions. The foreground functions take care of real-time control and interrupt handling and the background functions take care of the operator/maintenance video terminal interface.

## HARDWARE STRUCTURE

### Description of an STD bus.

STD bus is an internationally recognized computer bus architecture. The bus utilizes eight data lines and sixteen address lines. The card cages are approximately five inches high, seven inches deep and are available in 19 inch rack mount sizes suitable for industrial environments. Presently, there are over 2000 manufacturers of STD bus products. Systems are available which can either run without an operating system or can use any one of several popular operating systems such as UNIX, DOS, OS/2, VRTX and QNX. Several STD bus manufacturers offer IBM PC compatible STD bus systems.

### CPU and I/O Board Descriptions

The CPU boards used for embedded computer applications in EBR-II have some common features. Each CPU board has at least two RS-232C compatible serial ports and three hardware counter/timers. The serial ports allow data communications with other computers, terminals or printers. The counter/timers provide the clock for serial port baud rates and are used to generate periodic interrupts for real-time control and I/O tasks. System memory consists of RAM and EPROM which all resides on the CPU board. The systems operate without moving disks and some CPU boards support a floating point math coprocessor.

Two types of CPU boards have been used in EBR-II, a Z80 based board and a V53 based board. The Z80 board used in EBR-II has a CPU clock of 6 MHz. This board has three on-board counter timer units, two serial ports and can support up to 64 kbytes of memory consisting of both RAM and EPROM. The V53 board has a CPU clock of 16 MHz. The on-board V53 microprocessor is instruction set compatible with the Intel 80286. This board supports three on-board counter timer units, three serial ports and can support up to 1 Mbyte of RAM and another 1 Mbyte of EPROM. The board has an 8259 compatible interrupt control unit and optionally supports an 80287 math coprocessor.

STD bus I/O boards can consist of about any number, combination and type of analog and/or digital I/O boards. Intelligent boards are available which can perform automatic engineering unit conversions apart from the CPU.

A watchdog timer board is included in each system to provide fault detection capability independent of the CPU board. The watchdog timer requires periodic digital pulses from a digital output board that is driven from a real-time software function. These pulses indicate that the real-time control and I/O tasks are running normally. If these pulses are not received, the watchdog board shuts the systems down by removing power to the system. Outputs in the system are configured to assume a fail safe configuration in their powered down state.

### Flexibility, Accuracy, Reliability, Maintainability and Cost

The STD bus embedded system approach has proven to be very flexible, accurate and reliable. It is a general purpose computer approach and, as such, is not limited to a specific language, a specific CPU or I/O board set, or operating system. Many vendors make boards for the STD bus so a wide variety of CPU and I/O boards are available and there are multiple sources for boards of similar capability. The I/O boards can be purchased with varying conversion resolutions and speeds. Analog I/O boards are available with from 8 to 16 bit with +/- 1 bit resolutions and conversion times varying from 25 microseconds to 30 milliseconds. The CPU and I/O boards have proven to be very reliable. Some EBR-II embedded systems have had no failures since installation and have been running for several years. On one particular CPU board the manufacturers MTBF is quoted at 25 years.

The STD bus boards are usually repaired by EBR-II plant technicians. Since the I/O boards are small and come with manuals that typically include schematics, technicians are able to troubleshoot and repair the boards. The CPU boards are often complex enough that, depending on the problem, they are sent back to the manufacturer for repair. Experience has shown that CPU boards are either delivered with problems or that problems occur shortly after the boards are first installed in a system. At the embedded system level, video terminal screens greatly simplify maintenance and troubleshooting by providing channel calibration and I/O board status information.

The STD bus boards are relatively inexpensive. For the embedded systems at EBR-II, CPU board prices have ranged from $285 to $1080, depending on microprocessor and speed; analog I/O board prices have ranged from $400 to $600 for 16 to 32 analog input channels or 4 to 12 analog output channels; digital I/O board prices have ranged from $250 to $350 for 56 to 112 I/O channels; and STD bus card cage prices, with power supply, range from $500 to $600. The STD bus hardware cost for a typical embedded system with 96 analog and 112 digital I/O points and a Z80 based CPU board is approximately $4000.

## SOFTWARE STRUCTURE

The software is divided into two main areas, foreground or real-time processes and background or non-real-time processes. The foreground software consists of all of the functions that handle interrupts, perform real-time acquisition and control, and, in some cases, handle serial communications. The background software consists of all of the functions that pertain to operator interface, maintenance and troubleshooting video terminal screens. The foreground and background functions exchange data through shared memory. See Figure 1.

The foreground software runs as large interrupt handlers. The interrupts are generated either by the counter/timer hardware or the communications hardware. When an interrupt occurs, the CPU temporarily suspends background process execution, saves the processor registers and begins executing the appropriate foreground functions. When the real-time processing is complete the CPU register contents are restored and the interrupted background process resumes execution.

The screen handling software is designed to write to VT100 or VT340 type video terminals and to read keyboard or trackball input. VT100 or VT340 screen and cursor control commands are used to create either interactive alpha-numeric or color-graphic screens. When the embedded system does not have a dedicated video terminal based operator interface, screen handling functions are still incorporated to provide maintenance and troubleshooting screens. When maintenance or troubleshooting is to be performed, a video terminal is plugged into a specific CPU board serial port. When the activity is complete the terminal is removed.

The operation and maintenance screens are menu driven and provide specific screens for plant system operation, monitoring, I/O calibration, and troubleshooting. The monitoring and maintenance screens display I/O information in real-time. To assist maintenance technicians during I/O calibration and

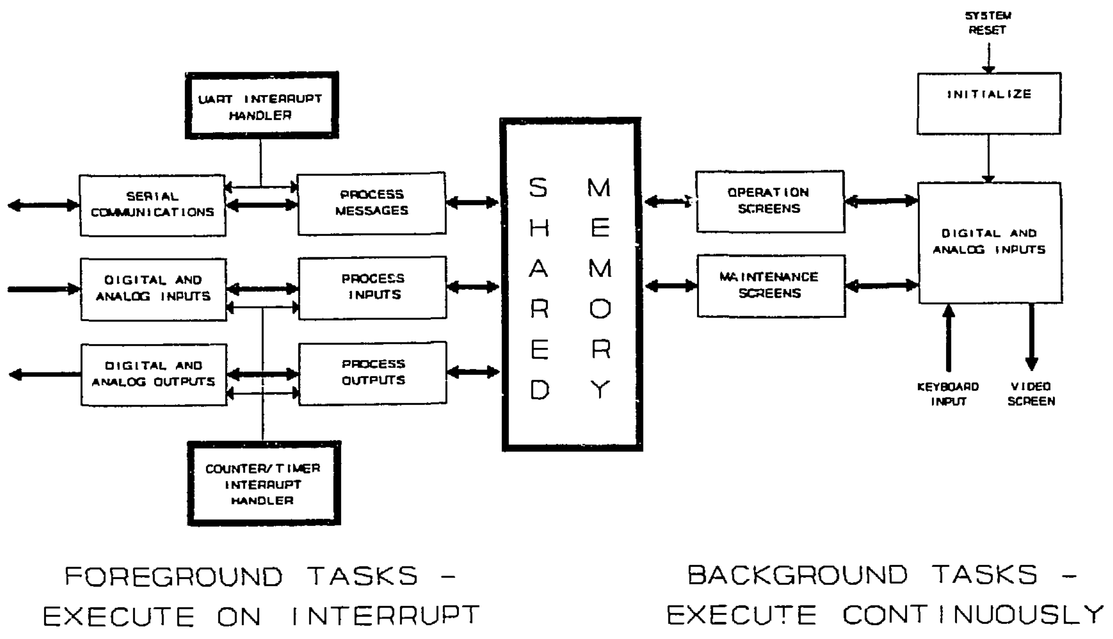

SYSTEM RESET
UART INTERRUPT HANDLER
INITIALIZE
SERIAL COMMUNICATIONS
PROCESS MESSAGES
SHARED MEMORY
OPERATION SCREENS
DIGITAL AND ANALOG INPUTS
PROCESS INPUTS
MAINTENANCE SCREENS
DIGITAL AND ANALOG INPUTS
DIGITAL AND ANALOG OUTPUTS
PROCESS OUTPUTS
KEYBOARD INPUT
VIDEO SCREEN
COUNTER/TIMER INTERRUPT HANDLER
FOREGROUND TASKS - EXECUTE ON INTERRUPT
BACKGROUND TASKS - EXECUTE CONTINUOUSLY

**FIGURE 1:** General Software Structure Diagram

troubleshooting, help information and prompting is provided on screen. Data entry and displayed values are in engineering units. Screens are normally provided that display all I/O input and output values and, often times, the capability is provided to drive specific outputs with keyboard entered values.

## EMBEDDED SYSTEM DEVELOPMENT

### Embedded System Design

Engineering embedded systems in EBR-II presents unique challenges compared with other engineering modifications because software and hardware are both involved. Successful implementation of plant modifications involving embedded computer systems requires that several documents be developed at appropriate phases in the engineering modification process. Independent review boards are convened at the end of each design phase before continuing to the next phase. The standard EBR-II approach to engineering plant modifications involves scoping, conceptual, preliminary design, final design, installation and testing phases followed by as-building and operational and maintenance phases. The following paragraphs describe the phases in more detail.

During the scoping phase, system boundaries are defined, major safety and performance criteria are established and project feasibility and scheduling are determined. In this phase, emphasis is placed on what the design is to do and not how it is to do it.

In the conceptual phase, major system functional objectives, and hardware and software functional requirements are defined. The software functional requirements are developed and a high level software criteria document is prepared which describes, in English language, what the software will do.

In the preliminary design phase, hardware and software designs are developed in greater detail. The hardware and software design requirements documents are finalized in this phase. It is common to prototype certain portions of the hardware and software to determine the direction the designs should take. Software prototyping is done for several reasons. Before completing the software design, it is often necessary to experiment with new hardware or new software algorithms or methods to determine the feasibility or adequacy of a design idea. Prototyping is also done to examine different approaches to a problem in order to allow some of the software design details to be established. The goal of prototyping is not to develop code to be a part of the final project software, but rather to determine the most reasonable solution to a given problem.

In the final design phase, hardware and software designs are finalized. The complete software design document will contain several items: a high level English language description of the software; a document describing module details; and, a tree structure diagram which shows module hierarchy.

**Coding, Simulation, and Test**

Hardware and software activities proceed in parallel in the building or implementation phase. Embedded system software is primarily written in C. Assembly language is used for initialization and interrupt handling routines and for speed critical functions as necessary. In-line and header software documentation is done during software coding. Software maintenance and installation instructions are developed. Final operational descriptions are written. A formal system test plan is completed once all of the hardware for the system has been obtained. At this point hardware and software are close to their final configurations. A software and test plan review is conducted to review the adequacy of the implementation and documentation. The code is reviewed for understandability and maintainability but not for implementation details.

The inherent portability of the C language allows software development to be done on PC's running DOS or on larger UNIX computers. Development on a UNIX platform allows aspects of the system to be simulated during development. Interrupt driven real-time portions of the software can be setup to run on a periodic basis and video terminal screens can be adapted to run off a UNIX system terminal or through a serial port to the embedded system video terminal.

Final target system testing and debugging methods vary with the CPU board type. In the case of the Z80 CPU board, a cross-compiler and assembler are used to build the Z80 instructions that are either downloaded to an in-circuit emulator or burned into EPROM. The in-circuit emulator allows testing and debugging of application software before it is burned into EPROM. In the case of the V53 CPU board, Turbo C++ is used to make the native code for the target system. The program may be downloaded to the target V53 system with a development PROM via a high speed serial link or burned into EPROM and installed on the board directly. With the development PROM, the V53 system allows remote debugging from a PC environment. As the final system development step, the development PROMs are removed and the final code EPROMs are installed.

Testing consists of informal and formal testing phases. In the informal phase, unit and integration testing is done by the software developers. In the formal phase, integration and validation bench testing are done. During testing, all of the I/O points are exercised and preliminary operational tests are performed. Following system installation, more integration and validation testing follow as the actual system hardware and software are integrated. When this is complete, end-user and system acceptance testing are performed.

At the completion of system acceptance testing, the hardware and software documentation are "as-built" to reflect any changes that were made during the final installation and testing phases and the system is ready for operation. The software documentation with it's design descriptions and commented source code listings are placed under configuration control. Any software maintenance or modifications to the embedded system use this documentation. When modifications are made, all of the affected design descriptions and module header and in-line comments are updated to reflect the changes.

## EXAMPLES

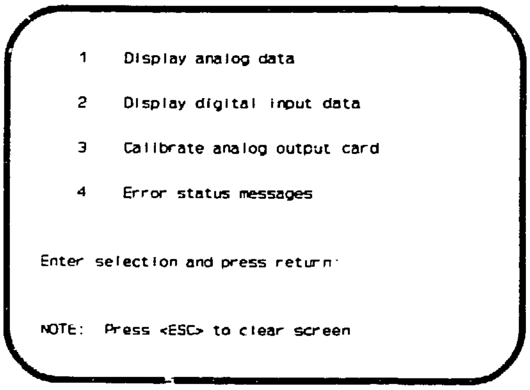
The following examples illustrate typical embedded systems that have been installed in EBR-II. These range from simple, non-terminal based to higher performance, color graphics based embedded systems.

### Example 1: Primary Tank Heater Control System

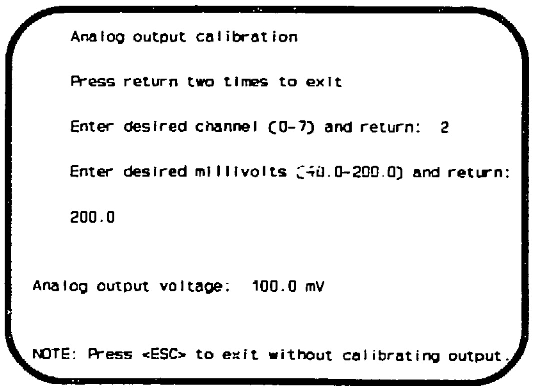
The primary tank heater control system controls six 112 kW heaters that maintain reactor primary tank sodium temperature when the reactor is shutdown. The system is monitored and controlled via a switch/digital display panel. The operators can raise or lower the output of each heater and can view heater current and power and total system phase current and power. A video terminal can be attached to the system for maintenance and troubleshooting.

The real-time control software runs in response to a one second interrupt and the analog data acquisition portion of the software runs in response to a 4.167 millisecond interrupt. When the 4.167 millisecond interrupt occurs the heater and phase current input data is read. When the one-second interrupt occurs heater and phase current input data is averaged, operator input switches are scanned, power is calculated, amps or power is displayed, and heater outputs are written in response to any raising or lowering operator input.

When a video terminal is attached to a specified serial port on the CPU board, a main menu can be brought up from which various calibration and maintenance screens can be viewed. Figure 2 shows the main menu and one of the calibration screens.



```
      1     Display analog data

      2     Display digital input data

      3     Calibrate analog output card

      4     Error status messages


Enter selection and press return


NOTE:  Press <ESC> to clear screen
```



```
    Analog output calibration

    Press return two times to exit

    Enter desired channel (0-7) and return:  2

    Enter desired millivolts (-40.0-200.0) and return:

    200.0


Analog output voltage:  100.0 mV


NOTE: Press <ESC> to exit without calibrating output.
```

Figure 2:  Main Menu and Calibration Screens

**Example 2: Hydrogen Meter Leak Detection System**

The HMLD system functions to detect steam leaks in the secondary sodium system that could occur in the sodium-to-steam heat exchangers. The Hydrogen Meter Leak Detector (HMLD) control system was upgraded a few years ago from a system that consisted of several individual controllers and relays to a new system that utilizes two embedded STD bus computers.

The STD bus based system provides automated heatup and steady state control of a system consisting of approximately 60 heaters, 24 of which use PID control. Two STD bus computers each provide independent control and monitoring functions for one of half of the system. There are approximately 260 I/O points in the system which consist of thermocouples, current sensors, calibration signals, solid state relays and mechanical relays. This system has a dedicated video terminal which is used as both an operator interface and for on-line maintenance and troubleshooting.

The real-time control software runs in response to a one second interrupt handler. Functions such as PID control calculations, alarm checking and analog signal averaging and engineering unit conversion are done by this interrupt handler. Analog data acquisition and PID control output are done in response to a higher priority 16.667 millisecond interrupt handler.

**Example 3: Secondary Sodium Loop Plugging Temperature Indicator**

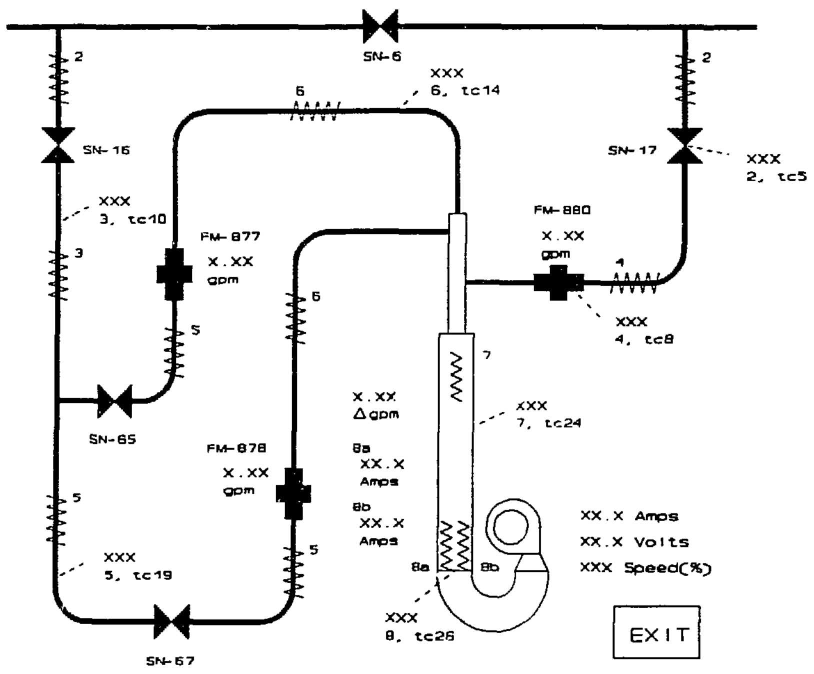


Figure 3: System Status Screen

The plugging temperature indicator is a system that determines the level of impurities in the EBR-II secondary sodium coolant loop by finding the temperature at which the liquid sodium will solidify and plug an orifice. The control system controls nine heaters and a variable speed fan, and monitors three flowmeters, heater currents, fan speed inputs, and approximately thirty thermocouples. The system is monitored and controlled via a VT340 color graphics video terminal. Operator input is through a trackball and keyboard.

The real-time software runs in response to a one second interrupt. When the interrupt occurs heater currents, fan speed inputs, thermocouples, and flowmeters are read; raw analog input values are converted to engineering units; inputs are checked against alarm limits; on-off and PID heater control algorithms are run; and new control outputs are written to the heater and fan outputs.

The color graphics video terminal allows an operator, through a menu driven set of screens, to change the system operating mode, to monitor the operation of the plugging temperature system, to change alarm limits, and to change PID tuning factors. In addition to operating screens, maintenance screens are provided for troubleshooting and I/O board calibration. Figure 3 gives a representation of one of the system status screens. The actual screen would appear in color with different colors for on, off, labels, real-time data, components, and the screen background.

## CONCLUSION

The embedded systems approach described in this paper has proven to work quite well for EBR-II. It has allowed EBR-II to implement computerized upgrades without incurring the expense and schedule normally associated with non-embedded computer solutions. The flexibility attained by using the STD bus approach allows a wide range of applications to be developed using the same basic hardware and techniques. It has provided a very flexible, reliable, maintainable, and cost effective solution for new control systems and control system upgrades in the plant.