A MAGNET CURRENT MONITOR FOR GYROTRON MAGNET POWER SUPPLIES*

N. Greenough[1], J. Lohr[2]
[1]Princeton Plasma Physics Laboratory, Princeton, NJ 08543-0451
[2]General Atomics, San Diego, CA 92121
ngreenou@pppl.gov.

Abstract— **Gyrotrons are widely used in Electron-Cyclotron Heating systems (ECH) to heat electrons in plasmas. Modern gyrotrons require a precise magnetic field for proper operation which is supplied by an electromagnet with multiple windings. The current for the electromagnet is provided by multiple high-current power supplies. The loss of any one of the power supplies during operation can cause significant and expensive damage to the gyrotron. An independent power supply monitoring system can prevent damage should a magnet power supply fail.**

This paper presents a magnet current monitor system capable of measuring and monitoring five magnet currents simultaneously. It provides limit tests, an interlock output for power supply failures and a status display for the operator. A self-test feature is included and the measurements and status can be examined through an Ethernet connection.

The rapid advance of inexpensive microcontroller technology has led to the development of small microcontroller platforms and associated code libraries that have dramatically decreased the costs and development time required for such projects. This paper describes the use of a popular open-source software, open-hardware platform for the design and fabrication of the magnet current monitor.

I. INTRODUCTION

Megawatt-level millimeter-wave RF power sources use gyrotrons for RF power generation. Gyrotrons require a precise magnetic field for proper operation which is supplied by an electromagnet with multiple windings. The current for the electromagnet is provided by multiple high-current power supplies. The loss of any one of the power supplies during operation can cause significant and expensive damage to the gyrotron. An independent power supply monitoring system can prevent damage should a magnet power supply fail or be powered off inadvertently. This paper describes an independent, non-contacting current monitor for the magnet power supplies fabricated from inexpensive and widely-available microcontroller components.

The typical gyrotron magnetic system requires as many as 6 or more independent power supplies. The superconducting magnets for the gyrotron installation at General Atomics require five power supplies plus an AC sweeping magnet for the gyrotron collector. The failure of any one of these power supplies can cause electron beam impingement and melting of internal components of the gyrotron. Prompt removal of the high voltage is required should a magnet power supply current fall outside a specified set of limits.

II. FEATURES AND SPECIFICATIONS

The magnet current monitor system being prototyped for General Atomics has the following features:

- Five independent DC magnet current measuring channels using non-contacting 50 ampere Hall current sensors, appropriately filtered for superconducting magnets
- One or two sweep coil current measuring channels using 50 ampere non-contacting Hall current sensors
- Resolution of approximately 0.1 ampere on all channels
- Dual limit test on all DC current monitor channels- high and low tolerance limits for information; high and low operating limits for fault signaling
- AC and DC test for the sweep current monitor channels- high and low limits on average sweep coil current; high and low limits on AC sweep coil current
- Interlock outputs for magnet and sweep power supplies "OK"
- Intelligent status output over Ethernet or USB using an SCPI-like command set
- A local LCD display for magnet power supply status, limit and device setup
- Simple 5-button, menu based user interface for setup

III. INSTALLATION

The magnet current monitor is intended to be installed adjacent to the magnet power supplies. The current sensors are installed on the magnet leads near the power supplies. An existing magnet lead interconnect panel mounted above the power supplies presently provides a convenient location for the current sensors. The measurement chassis can be mounted in a cabinet nearby for operator convenience. Figure 1 shows a physical layout of the measurement chassis and sensors.

*This work supported by the US DOE Contract No. DE-AC02-09CH11466.

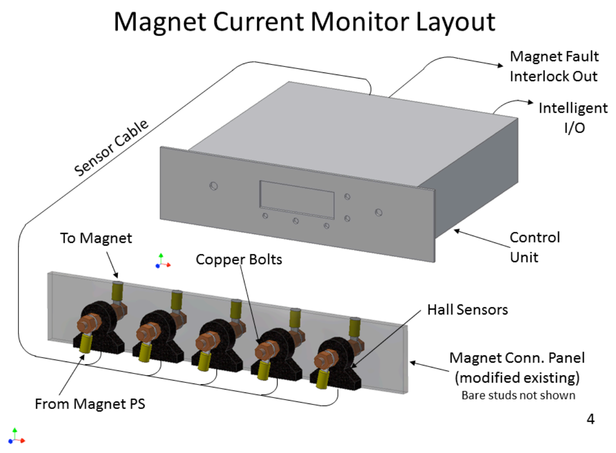

Fig. 1. Magnet Current Monitor Physical Layout

The requirements of this device are well within the capabilities of a mid-range dedicated microcontroller from a number of different manufacturers. Internal multichannel A/D converters, sufficient I/O pins, watchdog timers and on-chip serial communications hardware of several types are common. Some important selection considerations are:

- Sufficient on-chip peripherals such that external circuitry is minimized
- On-chip flash program, data and non-volatile memory
- Support for programming languages appropriate for the application
- Availability of pre-tested library code and application samples simplifying development
- Availability of a user-friendly programming environment eases less-experienced programmers
- Footprint suitable for low-volume production, no fine-lead-pitch package assembly challenges
- In-field software upgradeability via a common interface; no expensive programming devices needed
- Well-established user base assuring future availability and upgrade paths

The "Arduino" microcontroller board series, associated "C" compiler and software libraries meet the needs of this project. The hardware is widely available and second-sourced, the compiler and libraries are licensed under the GPL non-proprietary software license which assures future availability, and the entire source code for the compiler and libraries is freely available for examination and modification [1].

The Arduino system is based on the AVR brand of microcontrollers and offers several choices suitable for this application. The system is PC board based, with the microcontroller, 5-volt regulator, USB interface and I/O connectors on the board. It is small and can be readily daughter-boarded to a larger custom PC board in cases where additional circuitry is required.

The Arduino boards are typically programmed in C using the well-known GCC cross-compiler running under a development editor/uploader. The boards attach to the development computer with a simple USB cable which also allows rudimentary debugging and monitoring.

Two Arduino boards were chosen for evaluation; the "UNO" containing an ATMEGA28p processor and the MEGA2560 containing the ATMEGA2560 processor. The UNO is the smaller of the two, with 32K of program memory and 2K of data memory. The MEGA2560 offers 256K program memory and 8K data memory. The UNO offers a socketed processor allowing program upgrades by processor swap. The MEGA series has more I/O pins and larger memory spaces for more complex processing if needed.

An initial DC magnet current demonstration was developed using the UNO board. The entire button-and-menu-based control code, the analog measurement code and the limit testing code readily fit in about 1/3rd of the available program memory. The user interacts with the demo using 5 buttons to control menus, make choices and set parameters.

Figure 2 shows a demonstration version of the magnet current monitor Most all of its software features are in place except the remote interface. The potentiometer is a stand-in for one of the Hall-effect current sensors.

Array spaces, however, taxed available data memory on the UNO when AC sweep coil measurements were added. Development was therefore moved on to the larger MEGA2560 board for further work.

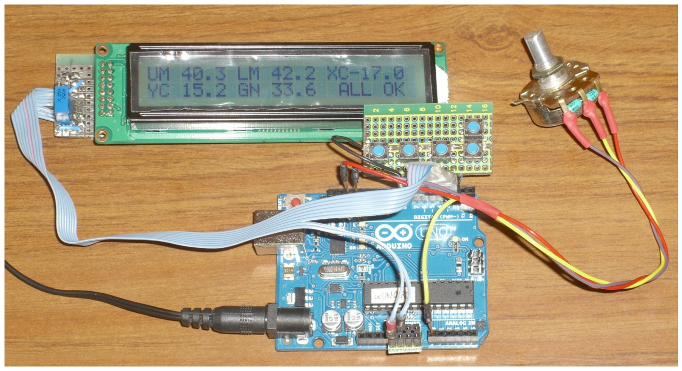

Fig. 2. Magnet Current Monitor Physical Layout

Figure 3 shows a functional block diagram of the Magnet Current Monitor. It consists of six or more Hall-effect current transducers, analog input processing circuitry, the microcontroller with its on-board processing program, an LCD display, user buttons, interlock and status outputs, and self-test circuitry.

Passive style Hall transducers were chosen for this project due to their increased ruggedness in hostile environments over active types that include feedback coils. The loss in accuracy is slight in modern models. They have a single-ended Hall bridge output and require a small DC bias current. The output swing is about +/- 2 volts for 50 ampere versions.

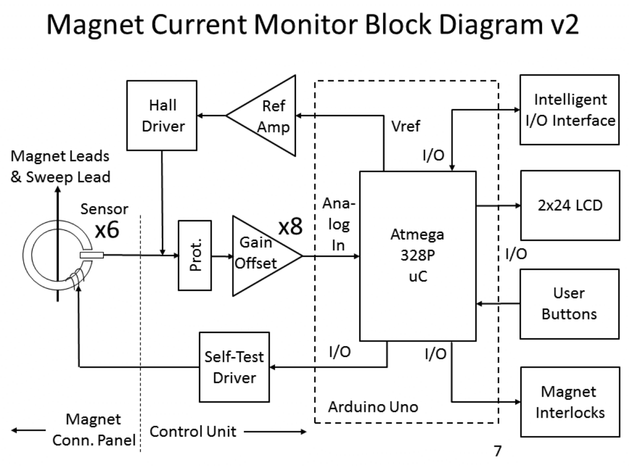

Fig. 3. Magnet Current Monitor block diagram

Some analog input processing circuitry is required to interface the Hall sensor to the 0 to 5-volt range of the on-chip A/D converter of the microcontroller. Filtering, transient protection, zeroing, scaling and offsetting are required.

The microcontroller takes care of most all of the tasks required for the magnet current monitor. It acquires data from the on-chip A/D representing magnet currents, checks against user-settable limits for each magnet, and issues interlock outputs based on the limit tests. The speed requirements are minimal as the superconducting magnet time constants are several seconds. The limits and other operational parameters are stored in on-chip non-volatile storage so that reprogramming is not required. The limits and parameters are recovered automatically at power up.

The intelligent interface is an input/output from the microcontroller that sends more specific status and control information to the gyrotron control system. It consists of text-based high level commands and responses similar to the SCPI instrument control language. The gyrotron control system can request information such as individual magnet currents and their limit test responses. Optionally it can allow altering the limits of the tests. This allows a remote mimic display, for example, of the measured magnet currents to be provided for the gyrotron system operators.

The self-test feature is included to proof the magnet monitor itself either upon demand or periodically. It operates by inducing a significant current into the Hall sensors with a secondary winding and monitoring the measurements produced for the correct responses. The interlock outputs will be forced to the failed state during the self-test routine to prove their efficacy to the gyrotron control system.

IV. System Code

Microcontrollers differ from higher-level computer systems in that they do not have a background operating system. The programmer codes all of the tasks to be accomplished either with custom-written code or by invoking pre-written library code with the appropriate parameters. The benefit of this architecture is increased reliability due to the very limited number of processes running simultaneously, often only two to four. Compare this to the hundreds of simultaneous processes that a typical operating system such as MacOS, Windows or Linux invokes. The programmer can be very confident about just what his code is doing at most every instant of time. There is considerably less chance of inter-process conflict.

The code for the magnet monitor is written in a simplified C using the GPL-licensed GCC compiler and Arduino editor. The Arduino editor and compiler provide background code to initialize the processor and get it ready to run user code at startup. It also provides a library of higher-level system calls such as analogRead(analogChannel) to obtain A/D data and serial.print() to output data from the on-chip serial port.

The measurement of DC magnet coil current is straightforward. Accept a value from the on-chip A/D converter and format it appropriately, then filter it with the equivalent of a low-pass filter. The filtering can be performed by emulating an R-C network in code. Assume that an output value from a previous cycle has been saved. Take a fraction of the difference between the new value and the previous output value and add it to the previous output value to create the new output value for this pass. The "fraction taken" controls the time constant of the filter which can be adjusted to meet real-world noise rejection requirements.

Measurement of AC sweep coil parameters is more interesting. Ideally code can watch for the maximums and the minimums of the typical sawtooth sweep coil current. Doing so with a fixed sample rate of the A/D converter leads to large errors. A beat frequency appears in the data caused by the difference between the sample rate of the A/D clock and the sweep coil sawtooth frequency. There are a number of possible solutions to this; incoherent sampling and a Fast Fourier Transform (FFT) are two of many possibilities.

Incoherent sampling relies on the fact that the statistics of a set of fixed-time-increment samples taken in order; a set of fixed-time-increment samples in random order; and a set of samples taken at random times are identical given sufficient samples. This technique was first made known to the author by the Hewlett-Packard (now Agilent) 3406 Sampling RF Voltmeter circa 1966. Its manual has a clear explanation of how the technique works and how randomness can be simulated by a frequency-sweeping A/D sample clock [2].

An approximation of incoherent sampling is relatively simple to execute in software due to the looping nature of microcontroller code. Simply introduce a random delay to the A/D sample loop timing. Tests of this technique showed peak and minimum detection jitter of about .5 to 1% over 1024 samples of the typical sweep coil waveform. This is well within the accuracy required for the application. Averaging to obtain the DC component of the sweep was done by summing all 1024 samples and dividing by 1024. The averaging algorithm was less successful with about 2% jitter.

The Fast Fourier Transform technique may provide superior sweep coil parameter detection. An excellent integer FFT code is available for the Arduino and processes 256-

sample FFTs in about 8 milliseconds [3]. A standard FFT processes N samples at T sample rate into N/2 sample bins at a frequency spacing of T/N. Theoretically Bin #0 represents the DC component of the sweep coil waveform, the bin with the largest value represents the sweep coil frequency, and the sum of all bins except #0 indicates the total sweep coil AC energy. There are intricacies with this method, however. The windowing function used sets the available accuracy due to its influence on the binning. Obtaining a fixed-rate sample clock for an A/D converter in a higher-level language such as C can prove difficult due to its isolation from the hardware. Luckily, the internal peripheral control register names are known to the GCC compiler. Information and code examples for obtaining a fixed-rate interrupt on the Arduino can be found on the Internet [4]. A description of the FFT mathematics and windowing functions can be found on many Internet sites [5]. The FFT technique has not yet been tested.

There is little point in including volumes of C code in this paper. However, there are some interesting techniques worth examining that proved useful in coding the magnet monitor.

Calling a function in C involves the compiler copying all of the parameters to a "heap" (a scratch area of volatile memory) then jumping to the code of the function. When the function ends the code returns to its previous position in the program and the heap is discarded. Only one parameter can be returned by the function, such as a calculated variable or the address of some data. This causes difficulties for microcontrollers with very limited volatile memory if the function's parameters involve arrays or other large structures of data. There is a simple solution to the dilemma: pass the address of the data structure to the function and have it operate on the data structures in-place without making copies. This has the added benefit of being able to write one function and have it operate on multiple data structures. It is a simple concept but the syntax to code it is obscure. Examples of this technique can be found on the Internet and programming books with varying levels of clarity. The following shows one complete example coded for Arduino [6]:

```
// Set up a simple structure with three variables in it
struct mydata {
  int item1, item2, item3;
};
// Now declare a couple of instances of type mydata...
  struct mydata astruct, anotherstruct;
  int X, Y; // And a few scratch variables
// The "setup" routine runs once when you press reset:
void setup() { // Nothing needed here
}
// The "loop" routine runs over and over again forever:
void loop() {
/* Initialize one of the variables in the structure using dot
notation */
  astruct.item1 = 10;
  anotherstruct.item1 = 6;
/* Now call a function to do something to the calling structure.
Call it with the address of the structure. Do something that
modifies astruct */
  afunction(&astruct);
//Now do the same "something" to anotherstruct
afunction(&anotherstruct);
/* This allows us to do the SAME thing to DIFFERENT data
sets */
/* Now look at the modified structures using dot notation */
  X = astruct.item2; /* X gets assigned 50 here, as
modified by afunction */
  Y = anotherstruct.item2; /* Y gets 30 from
afunction. */
// The same "afunction" thing is done to both structures.
}
/* Now the function. Reference the structure from the call by a
pointer "*"  */
void afunction(struct mydata *d) {
/*"d" can be any name. Access elements of the structure by
name using "->" or "(*  )." notation */
  d->item2 = 5 * d->item1; // Dummy action
  (*d).item3 = 9 * (*d).item1; // More dummy
} // Either form works.
```

V. Conclusion

The Gyrotron Magnet Current Monitor is a protection device designed to prevent operation of high-power gyrotrons in case of incorrect magnetic fields. The design is in the prototype implementation stage, and is general-purpose enough to be applicable for other current-monitoring systems. It is implemented with a simple microcontroller with minimal external circuitry and can be reproduced at moderate cost. Its hardware, internal programming, programming environment and software tools are all open-source and royalty-free. It can be programmed or re-programmed in the field with an ordinary computer of most any type with nothing more than a USB cable.

VI. References

[1] See www.arduino.cc main page.

[2] "Model 3406A Broadband Sampling Voltmeter Operating and Service Manual", Hewlett-Packard Corporation, 1966, p.4.1 on; can be located at www.agilent.com, search on "3406A".

[3] See http://wiki.openmusiclabs.com/wiki/ArduinoFFT for a fast integer FFT library compiled for Arduino.

[4] http://hobbytronics.co.uk/arduino-timer-interrupts provides one of the clearer examples of the technique.

[5] One of the more entertaining references is at www.katjaas.nl

[6] Patterned after "New C Primer Plus", Waite Group, 1990 p. 559.