A Modulated X-ray Source Controller for the In-Flight Calibration of X-Ray Astronomy Payloads

G. Davidoff, K. Pletcher, M. Schmidt, and A. Whitcombe

July 30, 2010

Abstract

We designed and built a controller for a modulated X-ray source (MXS) that can be used to calibrate X-ray detecting instruments. This source has three adjustable inputs: the voltage supplied to an ultraviolet light-emitting diode (UV LED), a high voltage supplied to an electron multiplier, and a high voltage to accelerate electrons towards an elemental target. The controller provides both a graphical user interface (GUI) through which one can set these voltages, as well as the supporting circuitry to accomplish this. The system also displays and saves relevant housekeeping information, including the voltages, currents, and temperatures at critical points in the system.
CONTENTS

I Introduction ................................. 3
   I-A Specifications ........................................... 4

II System Overview ............................... 4
   II-A Major Goals ............................................. 4
   II-B Software ................................................. 6
   II-C Firmware ............................................... 7
   II-D Hardware ............................................... 7

III Testing results ............................... 8
   III-A Background noise characterization ......................... 9

IV Future Directions ............................. 10

V Acknowledgements ............................. 10

Appendix A: PIC Firmware ............................. 11

Appendix B: Circuit Schematic ..................... 15
I. INTRODUCTION

We designed and constructed a control system for an X-ray source used to calibrate X-ray astronomy payloads—specifically, a modulated X-ray source (MXS) to be used in the X-ray Advanced Concepts Testbed (XACT) sounding rocket.

X-ray astronomy is a particularly significant area of study because it provides information about high-energy phenomena throughout the universe. Our team’s NASA mentor, Keith Gendreau, is particularly interested in the polarization of galactic X-ray sources. One method of conducting research on X-rays (which is best conducted outside of the Earth’s atmosphere) uses relatively small and reusable sounding rockets. The XACT sounding rocket which uses the MXS that we worked with will be used to send X-ray astronomy experiments into space quickly and inexpensively.

Accurate studies of X-rays require well-calibrated X-ray detectors. To accomplish this, X-rays of known energies are generated from a source and provided to the detector, thereby providing a reference point for subsequent measurements. Most present methods of in-flight X-ray detector tuning use radioactive sources to generate these known X-rays. However, because the radioactive source constantly produces X-rays, this calibration method greatly reduces the sensitivity of the X-ray detector, generating a large amount of background noise that can hinder the detection of actual X-rays. The MXS allows an X-ray detector to both remain properly calibrated and retain its sensitivity by producing X-rays at a controllable rate.

To do this, the MXS uses ultraviolet light to generate photoelectrons which are accelerated into a high voltage target that emits X-rays. A UV LED shines onto the X-ray source through a small window where the high-frequency emitted photons cause electron emission from a photocathode by means of the photoelectric effect. These electrons pass through an electron multiplier, and are then accelerated onto an elemental target. The energy of these accelerated electrons colliding into the target causes X-rays characteristic of the target element to be emitted. This phenomenon is illustrated in Figure 1.

![Modulated X-ray source (MXS) overview](image)

**Fig. 1. Modulated X-ray source (MXS) overview.** This diagram represents the modulated X-ray source. Relatively high-energy photons emitted from a UV LED are shone onto a photocathode. These electrons are multiplied by a cascading process of secondary emission, and then accelerated toward a high voltage target. The target material is an element that emits X-rays of a known energy which can be used to calibrate X-ray payloads.

The goal of our project is to design and implement a Mac-based controller and housekeeping measurement system for a modulated X-ray source. The controller should output two user-specified high voltages (0-3kV and 0-10kV), as well as a signal necessary to achieve a desired relative flux output from an ultraviolet LED. The housekeeping measurement system will constantly display relevant measurements to the user such as the current, voltage, and temperature of the LED as well as the current and voltage input to the two DC-DC converters.
Controller responsibilities. This figure illustrates the voltages that the MXS is responsible for controlling. It must be able to drive the UV LED with a user-defined input, control the voltage going into the 3kV DC-DC converter used to power the electron multiplier, and control the voltage going into the 10kV DC-DC converter used to accelerate electrons towards the elemental target. At the same time, it must display housekeeping data such as the temperature of the LED and the current and voltage supplied to the various system components.

A. Specifications

The system for our MXS controller has three main components: software to interface with the user, hardware to interact with the physical world, and firmware to connect the two. The main components of our system are illustrated in Figure 3. Its main purpose is to allow a user to control X-ray output by controlling the voltages which are input to the MXS.

The software is a graphical user interface (GUI) with supporting code created in Python 2.6 which allows a user to both read desired housekeeping information—LED temperature and currents and voltages going into various components—and set voltage inputs. To set the LED output, the user has the option to upload a list of relative flux inputs in ASCII format and specifying a low and high voltage to drive the LED, or to set high and low voltages and the pulse width and frequency of the signal.

To read and write voltages, the MXS controller uses analog-to-digital and digital-to-analog conversion to allow a PIC microcontroller to interface with the GUI input. The hardware also includes differential amplifiers that magnify voltage drops across a small known resistance so that current can be calculated from the amplifier’s output voltage. A sensor whose voltage output correlates with temperature allows for temperature monitoring of the UV LED.

In order to process and send information from the USB peripheral, the firmware on the PIC microcontrollers are configured for high-speed USB communication. To set LED voltages, which are sent in ASCII format, the PIC that drives the LED stores the list of desired voltages in its memory, and then loops through them at a rate determined by a specified frequency. The other PIC sets voltages to the DC-DC converters and reads housekeeping voltages when requested by the USB peripheral. Both PICs are enabled with Serial Peripheral Interface (SPI), allowing them to communicate desired information to and from the ADC and DACs.

II. SYSTEM OVERVIEW

A. Major Goals

Our objective for this project is to build a system that allows a user to monitor and control the MXS. In order to accomplish this task, we set several goals for ourselves specific to the software, firmware, and hardware subsystems.

The main goals for our software are: to provide a user-friendly way to set voltages and LED flux, to display the current state of the housekeeping information, and to communicate with the PIC microcontroller in order to perform these tasks.
For the firmware, we have two main goals. Our first goal is for the PIC to be able to cycle through an input of a 250-line ASCII file, which can be used to simulate the behavior of a pulsar. The second goal is for the control system to operate in a timely manner, with a time resolution on the order of several microseconds. This is critical because any significant time delay caused by the PIC has the potential to alter the LED flux signal—a factor that is ideally known for every instant in time.
Finally, our principal goal for the hardware subsystem is to design a USB peripheral device that can set voltages and monitor housekeeping data. This device’s hardware consists of a printed circuit board housed in a sturdy box.

B. Software

The software for the MXS controller consists of a GUI that accomplishes two tasks, controlling the inputs and interpreting the outputs of the MXS.

The control aspect of the GUI has two parts. It can be used to determine the high voltages which will be output by the DC-DC converters—powering the target and multiplier on the MXS—and to control the flux output of the LED. The high voltages are changed by editing the correct line in the GUI and pressing enter. The desired output voltage of the DC-DC converter is stored in the program from the line edit. This value is then converted to the voltage which the PIC will send to the converters using Equation [1] determined from the DC-DC converter datasheet.

\[ V_{\text{toconverter}} = 4.3 \frac{V_{\text{out}}}{V_{\text{max}}} + 0.7 \]  

In Equation [1], \( \frac{V_{\text{out}}}{V_{\text{max}}} \) is the ratio of the desired output of the DC-DC converter to its maximum output. \( V_{\text{max}} \) is equal to three for the 3kV converter (multiplier) and ten for the 10 kV converter (target).

The LED flux has two methods of manipulation. The user can either specify the “discrete flux”—amplitude, low voltage, frequency and width of a desired pulse pattern—or load an ASCII file (see Figure [2]) which lists desired voltages relative to the low and high voltages specified in the same tab. The discrete flux generates a file from the variables specified at a file length that maximizes time efficiency. This is done by finding the file length according to Equation [2], where \( P \) is the pulse width and \( GCD \) is the greatest common denominator of 100 and \( 100 \cdot P \).

\[ \text{filelength} = \frac{100}{GCD} \]  

The high voltage is then a fraction of the file length specified by the pulse width while the low voltage is \( 100 \cdot (1 - \text{pulsewidth}) \) percent of the file. The ASCII file allows the controller to simulate such events as pulsars by outputting a voltage to the UV LED that is some percentage of the range specified by the user such that a “1” in the ASCII file corresponds to the high voltage and “0” corresponds to the low voltage.

This file, either generated in the “discrete flux” tab or loaded from an ASCII file, is sent to the PIC through a USB connection along with the high and low voltages and the desired frequency. These are used to generate a pattern of voltages which then drives the LED as detailed in the discussion on firmware. The upper and lower bounds of the frequency at which the voltages can be sent to the PIC is determined by the relationship shown in Equation [3] where \( d_{\text{input}} \) is the input delay—0 for the minimum frequency and 255 for the maximum frequency—and \( c \) is the number of lines in list of desired voltages.

\[ \frac{1 \cdot 10^6}{(3.5d_{\text{input}} + 30)c} \]  

The delay, implemented as a while loop on the PIC, has a time resolution of 3.5 \( \mu s \) (determined experimentally), and PIC takes 30 \( \mu s \) to change a voltage (also determined experimentally).

The secondary function of the GUI is to save and store “housekeeping data,” including the currents and voltages going into the high voltage target and multiplier as well as the LED temperature and maximum and average LED current. This information is updated visually in the GUI and is appended to a housekeeping file saved as the current date.
C. Firmware

The PIC firmware performs several key functions. One role of the PIC is to read voltage inputs and set outputs to the MXS controller system. Another task is to relay housekeeping information from the circuit to the GUI. To read and write voltages, the PIC has an enabled serial peripheral interface (SPI) bus allowing for data transmission between the digital-to-analog converter (DAC) and the PIC (to set voltages on the circuit) as well as transmission between the analog-to-digital converter (ADC) and PIC (to read voltages from the housekeeping sensors).

Beyond simply setting two fixed voltages for the DC-DC converters and reading housekeeping information, the PIC must be able to cycle through voltages input by the user to drive the LED. One possible implementation of this cycling involves constantly sending commands to the PIC from the computer after delaying a time correlating with the user’s desired frequency. However, this approach greatly strains the USB data transfer speeds, limiting the system to a time resolution of 3 ms. As the stand-alone PIC is capable of carrying out instructions at a much faster rate, a more efficient approach is to store the data on an array on the PIC and then loop through the voltage values stored there after delaying an amount of time corresponding to the user’s desired frequency. This approach improves the time resolution of the system by two orders of magnitude, to 30 \( \mu s \). In order to ensure that the process of setting voltages to drive the LED is not interrupted with housekeeping data controls, and to ensure that memory storage on the PIC was optimized, two PICs are used in our system—one to drive the LED and another to read housekeeping information and set DC-DC converter voltages (which are held constant). The difference between these two potential approaches to implementing this programmable LED flux function is illustrated in Figure 5.

For further details on the PIC code, see Appendix A.

D. Hardware

The system’s circuitry uses a USB interface to set voltages and read voltages to record housekeeping information. In this project, we used two PIC 18F2455 microprocessors—one to read housekeeping data and set voltages to the two EMCO Q-series DC-DC converters, and one to control the UV LED voltage—two DACs, and one ADC to give the user the ability to specify characteristics of the system through the GUI.

This system uses USB communication to transfer a user’s request from a computer to a PIC 18F2455 microcontroller. To set voltages, it uses a 12-bit MCP4922 DAC which uses an SPI interface to read information from the PIC. The DAC converts the
digital signal from the microcontroller into an analog value from 0 to 5V with a resolution of $\frac{1}{2^{12}} = 0.00024\text{V}$. Conversely, the 12-bit MCP3302 ADC transforms an analog voltage measurement from the current and temperature sensors into a digital signal that the PIC can read. It has the same resolution (12-bit) as the DAC, and also uses SPI to interface with the PIC.

To sense current, AD626 differential amplifiers are used to amplify a small voltage drop across a high-precision $0.25\ \Omega$ resistor. By calculating the actual voltage drop from the amplified value, the current going into a component can be calculated using the known resistance across the amplifier and the formula $I = \frac{V}{R_G}$, where $V$ is the measured voltage and $G$ is the gain. To achieve the maximum output of around 4 V during normal operation, the amplifiers were configured for a gain of around 40 given an input of 100 mV to the sensor. This gain was adjustable in the circuit using a $1\ k\Omega$ potentiometer in series with an $80.6\ \Omega$ resistor. To smooth the signal, the AD626 includes a built-in low-pass filter.

The system also includes two sets of operational amplifiers: the TLV274IN quad single-supply op-amp, used to buffer the output of the current sensor as supplied to the ADC, and the high current output L2722 op-amp, used to allow the DC-DC converters and LED to source current that the output from the DACs cannot provide. Both operational amplifiers are configured as simple voltage followers. For the full circuit schematic, see Appendix B.

III. TESTING RESULTS

Testing our controller with the MXS allowed us to validate its predicted behavior. With the MXS, one would expect to see an X-ray output dependent upon the flux output from the LED, which is controlled by the voltage supplied to the LED. Moreover, X-ray emission also depends upon the appropriate target voltage accelerating the electrons, as well as the voltage supplied to the electron multiplier which (influences the number of electrons emitted). In preliminary tests, we confirmed that the X-ray output lessened when the electron multiplier supply voltage was turned low, and saw that the output ceased when the electron target voltage was turned down too low.
To observe how the X-ray output varied with the LED flux output, the LED was driven with a simple triangle wave at voltages ranging through the minimum and maximum ratings of the LED with which we were testing at a frequency low enough to allow the MCA we were using to read data. The results of this test are shown in Figure 7.

At the top is the cumulative readings of X-ray counts, taken directly from the MCA (compensating for the capabilities of the 16-bit counter used by the MCA). Ultimately, we found that when driving the LED with a triangle wave within the range of possible supply voltages, the resulting X-ray counts produced a shape similar to the input voltage. Two issues keeping the measured X-ray flux output from matching the relative desired input were observed, however. First, the nonlinearity of the voltage-current relationship in the particular UV LED we used caused the resulting X-ray output to be nonlinear, correlating to the known flux output of the LED. Also, thermal issues pertaining to the LED heating with time and producing less flux were evident.

A. Background noise characterization

To confirm that the output of the MXS consisted of X-rays, and not simply extra UV radiation, we tested the system with only the UV LED–and not the target and multiplier voltages–on. If the avalanche photodiode used to detect X-rays was detecting UV photons from the LED instead of X-rays produced from the target, the readings would still be high even when the portions of the system meant to convert the UV to X-ray radiation were turned off. When we tested this, we found that the number of registered counts on the photodiode was negligible in comparison to the number of counts that were observed as X-rays.
Fig. 7. Testing results. At the center is the X-ray output read from the MCA in response to an LED driven with a triangle pulse ranging from 3.1 to 3.6 V, the operational range of the LED. The X-ray output correlates rather well with the voltage input, with the nonlinear relationship between LED flux and input voltage noticeable at low voltage inputs, and the noticeable decay of LED flux with time as the LED heats. Note that the bottom graph is on a different time scale than the other two.

IV. FUTURE DIRECTIONS

An opportunity for future research could be integrating our system with the MCA designed by a previous NASA/Olin summer research team. This would involve integrating the both sets of hardware and software into one package. There would be several obstacles associated with this process. First, the MCA code would need to be debugged, as the code for the flux monitor does not always behave predictably. Second, the MCA seems overly sensitive to vibrations and movements during data collection. If we were to continue with this project, we would need to determine why this is, and make the hardware more stable before integrating with the MXS Controller hardware. Finally, we would need to integrate the two GUIs into one software package. This is certainly possible, and has the potential to make concurrent use of both devices a much more user-friendly process.

Further capabilities that could be added into the system include an automatic shutdown feature, in which the voltage supplied to the DC-DC converters or the LED could be set to zero when the current or temperature readings are too high. Also, the system does not presently compensate for the nonlinearity of the relationship between LED voltage and flux (see Figure 8). Future revisions to the system could account for this by determining an equation to describe voltage versus flux for each individual LED used.

V. ACKNOWLEDGEMENTS

Many thanks to Olin College professors Steve Holt, Brad Minch, and Siddhartan Govindasamy for their help throughout this project. We’d also like to thank our mentor, Keith Gendreau, at NASA’s Goddard Space Flight Center in Greenbelt, Maryland.
Fig. 8. **Performance characteristics from UV LED datasheet.** While the flux-current relationship in the LED is approximately linear, the voltage-current relationship in the LED is not linear for low input voltages. Presently, our system assumes a direct correlation between flux and voltage, causing an unpredictable flux output when driving the LED at low voltages.

**APPENDIX A**

**PIC Firmware**

We would like to thank Professor Brad Minch for letting us use his code, created for a low-cost data acquisition module, as a foundation basis for our project. This served as the basis for our use of the DACs and ADC, as well as the PIC to USB communication framework. For the final firmware for both PICs used in our controller, see the documentation section of our website. Here are the portions of the code that played the most significant role in our system.

```c
unsigned char ReadADC(unsigned char ch) {
    unsigned char temp;
    word64 result;

    ch = ch + 1;
    _CS_ADC = 0;
    SSPBUF = 0x00000000; // reset the ADC
    while ( (SSPSTAT) & (1<<0) ) {} // make sure we are completely reset
    temp = SSPBUF;
    SSPBUF = (ch<<00000000) ? 0x00000000 : 0x00000000;
    while ( (SSPSTAT) & (1<<0) ) {} // make sure we are completely reset
    result = SSPBUF;
    SSPBUF = 0;
    while ( (SSPSTAT) & (1<<0) ) {} // reset the ADC
    return result;
}
```

Fig. 9. **ReadADC.** This function returns the voltage measurement on a desired pin of the ADC. First, it creates a temporary variable temp and string result. It begins communication to the ADC by setting the _CS_ADC pin low, and stores data received through serial communication (SSPBUF). It makes the high and low bytes of result correspond with SSPBUF, and then returns result.

The vendor requests determine what information the PIC will send or read from the computer by connecting specific data addresses corresponding to information sent through USB communication (the input to endpoint 0 on the USB, BD0I) to commands that correlate with the voltage output on the DAC or ADC. The vendor requests set up for the housekeeping and DC-DC converter PIC are shown in Fig. 11 and the vendor requests for the LED PIC are found in Figure 12.

In the housekeeping PIC, the GET_DAC case retrieves the voltage measurement output on each VOUT pin of the DAC and sends it to the computer. To determine which of the four DAC inputs to read out, the code takes in a specific index (referred to as wIndex in the usb_control_transfer, for information on USB transfers please refer to [USB Made Simple][1]), correlating with

Fig. 10. WriteADC. This function sets a desired voltage on a specified pin of the DAC. Depending upon the channel specified by the user, the code will send information to one of two DACs. To write information, it first writes the low value stored in the buffer representing the user’s input voltage, and then writes the high value stored in the buffer.

Each of the VOUT pins on the two MCP4922 dual DACs used. Then, it assigns high and low bytes of the DAC reading to the high and low bytes of information sent to the USB through BD0I.address. The SET_DAC case outputs a voltage specified by the computer on a desired VOUT pin of the DAC. To accomplish this, it assigns the variable DAC[specified index ranging from 0-3][b0 or 1 for a high or low byte] to the high or low byte of endpoint 0 of the USB communication. The specific voltage is dependent upon the user input (USB_buffer_data[wValue]). Then, the WriteDAC function is executed to actually write the desired voltage output to the DAC. The LDAC pin on the DAC is set low to “transfer the input latch registers to the DAC registers (output latches)” (DAC datasheet, p. 15). The GET_ADC case sends the voltage read on a specified pin of the ADC to the computer. The ReadADC function is used to find the voltage of the specified pin, which is then sent to endpoint 0 of the USB with BD0I.address.
Fig. 11. Housekeeping, DC-DC converter PIC vendor requests.
void VendorRequests(void) {
    unsigned char temp, i; // Variables for set_name and get_name function
    switch (USB_buffer_data[1][0]) {
    case STOP: // STOP vendor request: ends loop, sets LED voltage to zero
        initvec = 0; // End loop by setting initvec to 0
        p = 0; // Reset SET_LED index to 0
        DAC[1][0] = 0; // Save 0 to low byte of DAC input
        DAC[1][1] = 0; // Save 0 to high byte of DAC input
        WriteDAC(1); // Set DAC output using WriteDAC
        LBAC = 0; // Set LBAC pin low
        _LBAC = 1; // Set LBAC pin high
        MSG1.bytecount = 0; // No bytes are being sent to USB
        MSG1.status = 0x00; // Send packet as DATA, set 0x00 bit
        break;
    case START: // START vendor request: starts cycling through LED voltages
        initvec = 6; // Sets initvec > 5
        break;
    case SET_FREQ: // Set frequency by defining time between pulses
        signal_period[USB_buffer_data[1][0]] = USB_buffer_data[1][1]; // Save signal_period as the low byte of incoming data
        break;
    case SET_LED: // Stores incoming values from USB into arrays to cycle through voltages
        voltagelow[p] = USB_buffer_data[1][0]; // Save low byte into an array
        voltagelow[p] = USB_buffer_data[1][1]; // Save high byte into an array
        p++; // Increment p (index counter)
        break;
    case SET_NAME:
        EXCONBits.EXPD0 = 0;
        EXCONBits.CFGS = 0;
        EXAD = 0;
        EXCONBits.RD = 0;
        temp = 0;
        if (USB_buffer_data[1][0] == 0)
            temp = USB_buffer_data[1][1];
        for (i = 0; i < temp; i++) {
            EXAD++; // EXCONBits.RD = i;
            MSG1.bytecount = 1;
            MSG1.status = 0x00;
            break;
        }
    case SET_NAME:
        USB_dev_cfg = SET_NAME;
        USB_packet_length = USB_buffer_data[1][0];
        break;
    default:
        USB_error_flag = 0x01; // set Request Error Flag
    }

Fig. 12. LED PIC vendor requests.
APPENDIX B

CIRCUIT SCHEMATIC

[Diagram of Circuit Schematic]