8888 University Drive
Burnaby, BC
Canada V5A 1S6

info@mclabs.com
www.mfclabs.com

small innovations
for big solutions™

micro fuel cell labs

March 14, 2005
Mr. Lucky One
School of Engineering Science
Simon Fraser University
Burnaby, British Columbia
V5A 1S6

Re: ENSC 440 Design Specifications for a Micro Fuel Cell Testbench

Dear Mr. One:

Please find attached the document, *Design Specifications for a Micro Fuel Cell Testbench,* outlining the design solutions to meet the functionality of our product for ENSC 440. The purpose of the product is to allow researchers to easily perform rigorous testing on their design of a novel micro fuel cell. The testbench shall perform real-time data acquisition of the micro fuel cell operating parameters and shall be capable of operating as a stand-alone system which can be used to demonstrate the micro fuel cell for consumers and potential investors.

The attached document presents the design of the system including hardware, firmware, user interface software, and test plans to be executed during the development of the MFC Testbench.

Our team includes four talented individuals with expertise in various technical fields. The members of our team are Arash Jamshidi, Sarang Toosi, Olha Lui, and Shirin Farrahi. Please feel free to contact us at by e-mail at info@mfclabs.com for more information regarding our project.

Sincerely,

Shirin Farrahi

Shirin Farrahi
Chief Executive Officer (CEO)
Micro Fuel Cell Labs (MFC Labs)

Enclosure: *Design Specifications for a Micro Fuel Cell Testbench*

micro fuel cell labs

Design Specifications for a Micro Fuel Cell Testbench

Project Team
Arash Jamshidi
Olha Lui
Sarang Toosi
Shirin Farrahi

Contact person
Shirin Farrahi
sfarrahi@sfu.ca

Submitted to
Lucky One – ENSC 440
Mike Sjoerdsma – ENSC 305
School of Engineering Science
Simon Fraser University

Date: March 14, 2005

Executive Summary

MFC Labs recognizes the increasing need for convenient, long-lasting power for portable electronics equipment. Our company believes that micro fuel cell technology shows great potential for solving many of the problems with current portable power solutions including weight, recharging requirements, and lifetime. Our product, the MFC Testbench, is meant to address fuel cell researchers' need for a cheap and portable device for testing micro fuel cells. The MFC Testbench will allow for accurate and exhaustive characterization of a micro fuel cell's characteristics while also being a useful tool for demonstrating the fuel cell's operation to potential investors at tradeshows. The prototype of our product will be completed by the end of April 2005 and will immediately be put into use for testing the micro fuel cell developed by the National Research Council of Canada's (NRC) Institute for Fuel Cell Innovation (IFCI).

The main function of the MFC Testbench is to provide a stand-alone, portable, durable, accurate and easy-to-use measurement system that can help researchers demonstrate their micro fuel cell in tradeshows. In addition, the system can be used to characterize the voltage, current, temperature, and humidity operating parameters of a DMFC for many hours. The MFC Testbench will satisfy this functionality by providing a sensoring block with voltage, current, temperature and humidity sensing capabilities. These sensors will interface with a microcontroller which will display values on an LCD and will be capable of sending them to a computer with a graphical user interface. The Testbench will also have a onboard manual load and a load which can be accurately controlled electronically. The entire circuitry will be capable of being powered by a battery or using an AC source.

The MFC Testbench can be broken down into four different functional blocks: hardware, firmware, software, and mechanical. The MFC Labs Design Specifications document outlines the detailed design of each block including the material and parts that will be used for each, how they will be implemented, and how they will be tested. In addition, we outline which functionalities are satisfied by each functional block. Schematics, block diagrams, and tables are used to present detailed design information.

Table of Contents

List of Tables

List of Figures

Glossary

1. A/D – Analog to digital
2. ADC – analog to digital converter
3. DMFC – Direct methanol fuel cell
4. DUT - Device under test
5. EMC – Electro Magnetic Compatibility
6. GUI - Graphical user interface
7. IC – Integrated Circuit
8. LCD – liquid Crystal Display
9. MFC - Micro Fuel Cell
10. PC – Personal Computer
11. RISC – Reduced Instruction Set Computer
12. RTC – Real-Time Clock
13. SCK – Serial Clock
14. SPDT – Single Pole Double Throw
15. UART – Universal asynchronous receiver-transmitter

1. Introduction

The Micro Fuel Cell Testbench is a product in development that is intended for use in the laboratory environment for rigorous testing of novel direct methanol fuel cells (DMFCs). Additionally, the MFC Testbench will be able to operate as a stand-alone unit during tradeshows. The main testing operations of the device will include monitoring the operating voltage and current of the DMFC in response to variable load. The unit will also monitor the ambient temperature and humidity while the fuel cell is operating. The prototype of the MFC Testbench is currently under development with the intent of being tested in a laboratory environment by April 2005.

1.1 Scope

This document presents the design specifications and solutions that would meet all required functional specifications for the prototype of the MFC Testbench. One optional feature such as fuel supply and fluidics connections are addressed in this document since their implementation will allow for better testing of the prototype. The rest of the optional features as outlines in MFC Testbench Functional Specifications are not implemented in the prototype and are not presented in this document.

1.2 Objective

The requirements listed in this document are intended to outline the functional specifications of the MFC Testbench to the potential users, as well as to guide the design of the testbench.

1.3 Intended Audience

This document is intended to provide guidance for design engineering when implementing prototype of MFC Testbench.

The project manager will use this document to monitor the progress of the project and verify that design meets the functional specifications.

Marketing personnel will use this document to develop the promotional material.

2. System Overview

The main function of the MFC Testbench is to provide a stand-alone, portable, durable, accurate and easy-to-use measurement system that can help researchers demonstrate their micro fuel cell in tradeshows. In addition, the system can be used to characterize the voltage, current, temperature, and humidity operating parameters of a DMFC for many hours. Figure 1 shows a graphical representation of the MFC Testbench functional blocks.

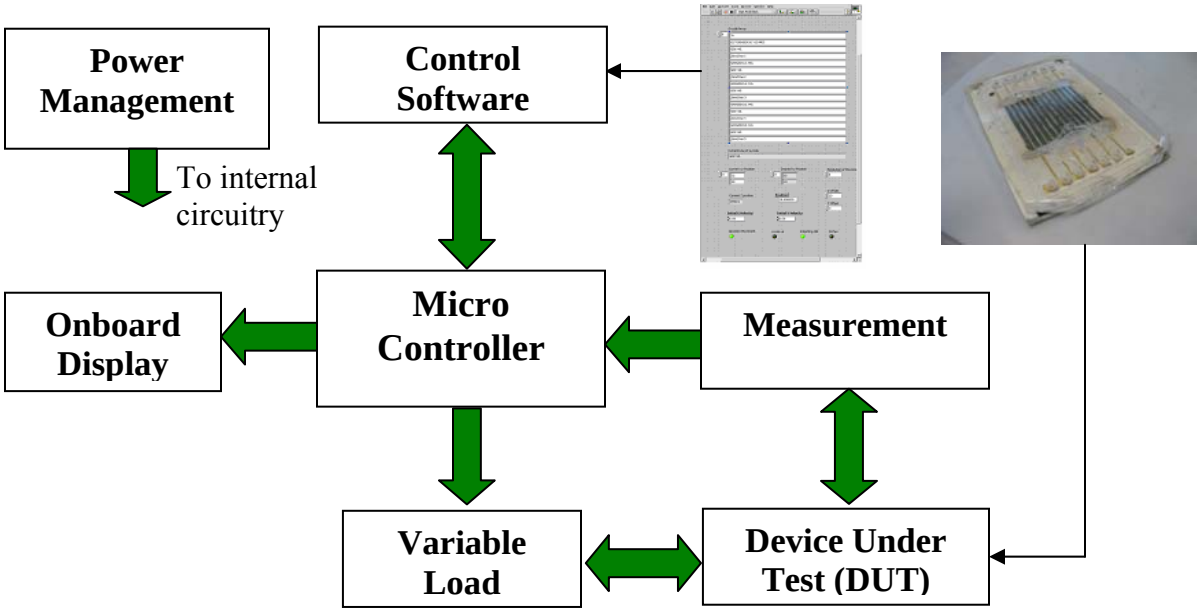


Figure 1: Graphical Representation of MFC Testbench System Blocks [1]

The micro fuel cell developed by Kevin Stanley's group at the Institute for Fuel Cell Innovation is shown in Figure 1. This fuel cell will be connected to the MFC Testbench and supplied with a 5% solution of methanol fuel. The MFC Testbench can be controlled by a graphical user interface (GUI) which can both log test data for the fuel cell and control the value of the variable load. The micro fuel cell operating data will also be shown on the testbench Liquid Crystal Display (LCD) as it is acquired through the measurement block. In addition, there will be a variable load connected to the DUT which will allow researchers to test the DMFC under various load conditions.

3. System hardware

This section describes the design solutions and considerations for the hardware of the MFC Testbench and external system components.

3.1 Block diagram of hardware

Figure 2 presents the block diagram of hardware for the MFC testbench. The main components include the sensoring block, microcontroller, buttons and knobs, LCD, power management block, DUT interface, and manual load.

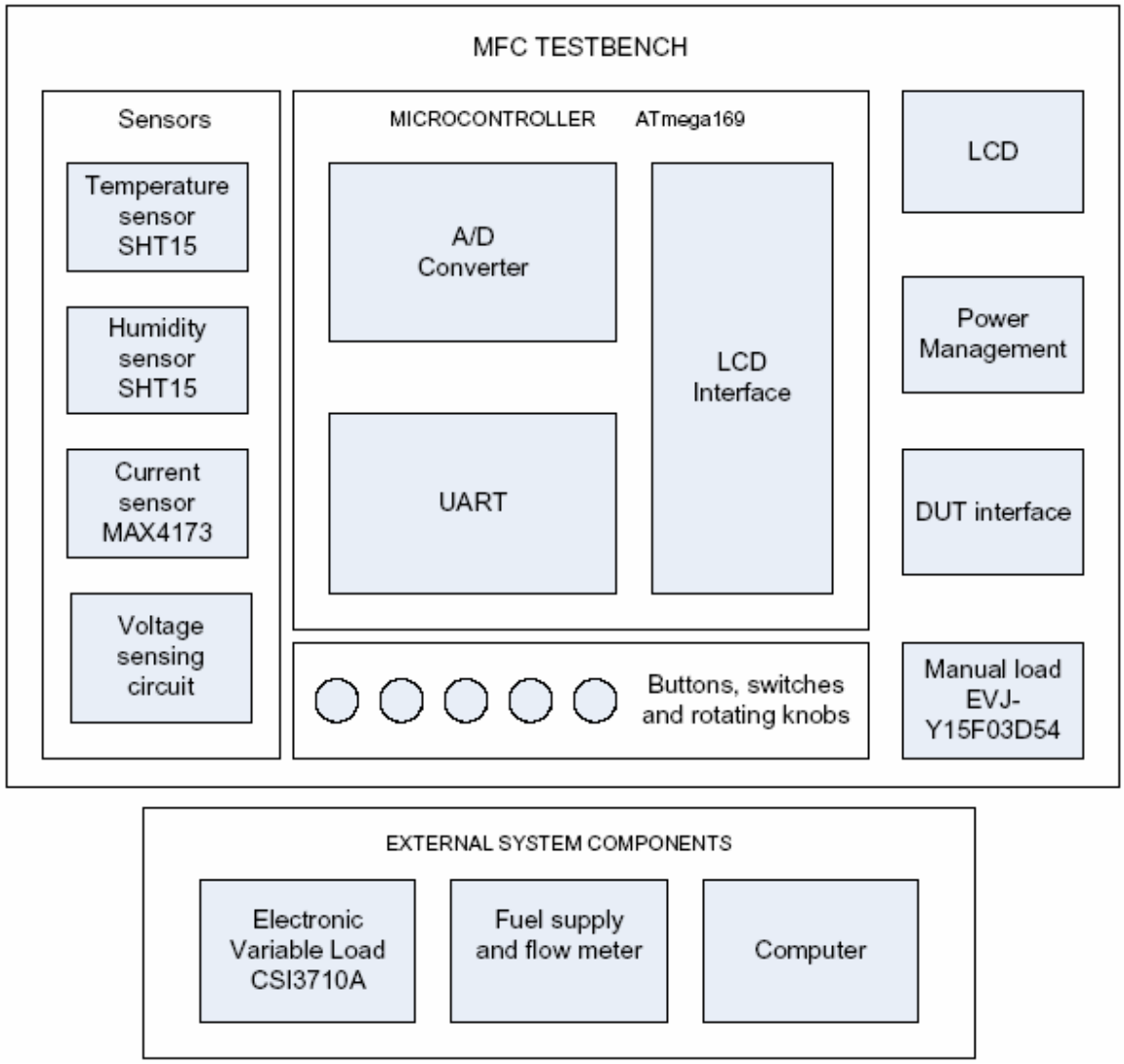


Figure 2. Block diagram of main system components

3.2 Sensoring Block

The sensoring block is made up of four sensors which sense the DUT operating voltage, current, temperature, and humidity as outlined in [6]. The sensor data is sent to the microprocessor which performs the necessary calculations to interpret the sensor data as outlined in section 4.2. The microprocessor then displays these values on the LCD. If the Testbench is in the data logging mode, the values are sent to the GUI program for data logging and display on the computer interface.

3.2.1 Temperature and Humidity

The temperature sensor used to sense the ambient operating temperature and humidity of the DUT is a single-chip multi-sensor board from Sensirion (SHT11). The chip consists of a capacitive polymer sensing element for relative humidity and a bandgap temperature sensor integrated with a 14-bit analog to digital converter and serial interface circuit as shown in Figure 3. This sensor was chosen because of its wide range of temperature and humidity sensing, its accuracy, its ease of interfacing with the microcontroller, its reasonable price, and its power requirement of between 2.4 V to 5.5 V. The chip interface consists of four pins shown in Figure 3 below, which will be connected to the microcontroller as detailed in Appendix II.

.

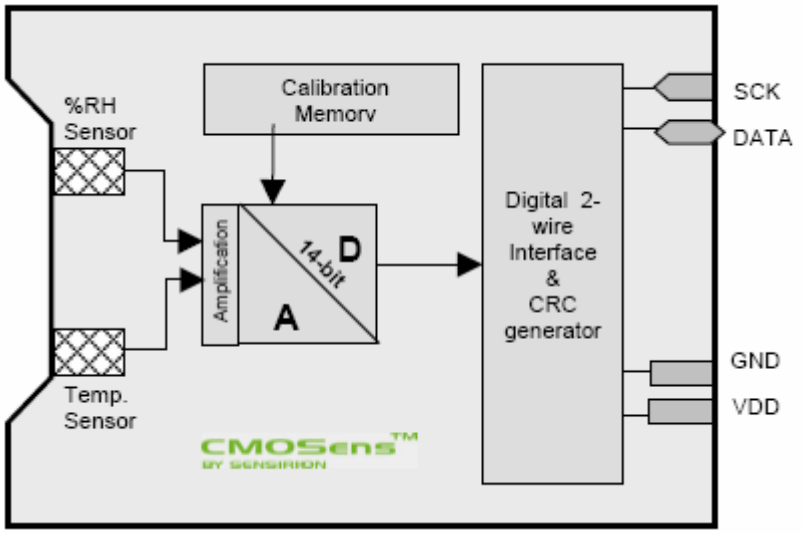


Figure 3: Block Diagram of SHT15 Board [7]

The ranges and accuracies of measurement provided by the SHT11 are given in Table 1. The accuracy of the temperature reading is given at 25 ºC.

Table 1: Ranges and Accuracies of Measurements for SHT11 Board [7]

	Range of Measurement	Accuracy
Temperature (ºC)	-40 – 100	± 0.4
Relative Humidity (% RH)	0 – 100	± 2

3.2.1.1 SHT11 Microcontroller Interface

Communication between the SHT11 and the microcontroller is timed as shown in Figure 4 below. Transmission is signaled to start when the Data line is held low and the serial clock (SCK) is pulsed low by the microcontroller. The SHT11 acknowledges proper receipt of a command by pulling the Data pin low after the falling edge of the 9th SCK clock pulse. The command from the microcontroller to the SHT11 consists of eight bits with the top three bits being the address of the SHT11 (only "000" available). The last five bits determine the type of measurement being made. For example, "00101" indicates a humidity measurement. The SHT11 indicates the end of a measurement by pulling the Data line low after which point two bytes of measurement data and one byte of checksum will be transmitted to the microcontroller.

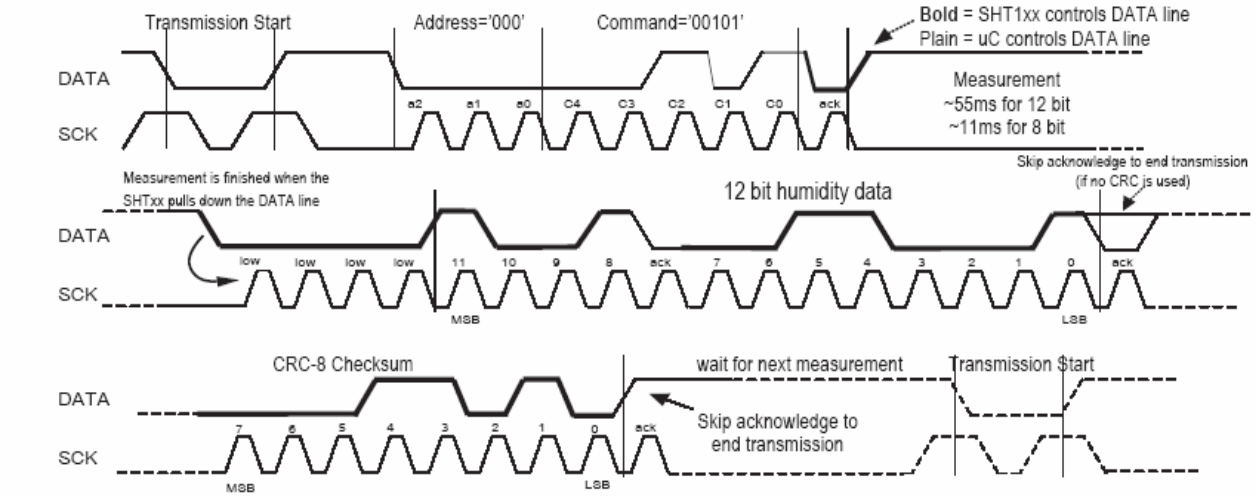


Figure 4: Timing Diagram for SHT11 Board [7]

The timing of communication between the microcontroller and the SHT11 will be encapsulated in a function provided by Atmel Corporation as outlined in section 4.1.

3.2.2 Current sensor

The current sensor must be placed in series with the electronic load and DUT as shown in Figure 5.

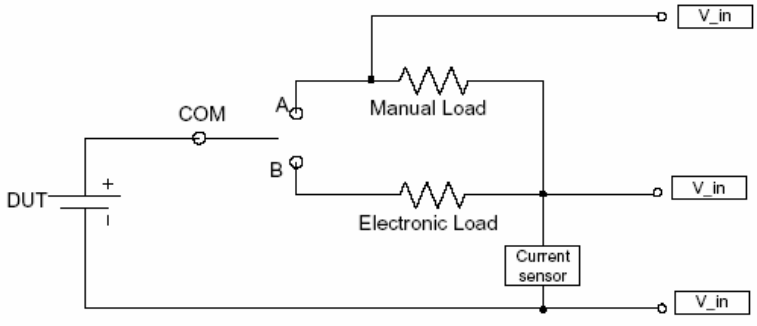


Figure 5: Schematic of DUT, Load and Sensors

Since the minimum load applied to the fuel cell is 1 Ω, the current sensor used must have an internal resistance much lower than this value. Therefore, we chose the MAX4173 chip whose internal block diagram is shown in Figure 6 below. The MAX4173 will measure the voltage across an accurate, sensing resistor R_{SENSE} = 0.1 Ω placed in series with the DUT [8]. The sensing resistor used will be the 3W 0.1 Ω 13FR100 from Ohmite. The MAX4173 chip will provide a useful scaling gain value for connecting to the ADC of our microcontroller. We will choose a gain value of 20 V/V (MAX4173T) to provide a full-scale V_{out} reading of 2V to the ADC of the microcontroller for a current, I_{LOAD}, of 1A, which satisfies the maximum current specified in [6]. The voltage supplied to the MAX4173 chip, V_{CC}, will be 3V.

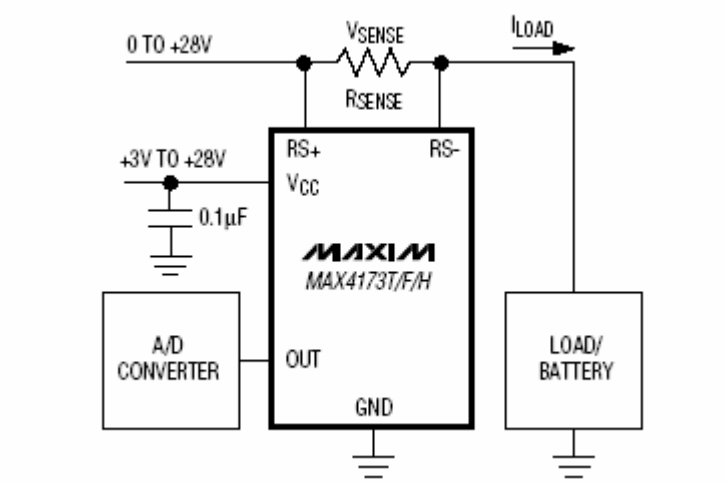


Figure 6: Block diagram of MAX4173 Chip [8]

The accuracy of the current reading by the MAX4173 is determined by three parameters: the tolerance of R_{SENSE}, the total error of the MAX4173 V_{OUT}, and the output resistance of the MAX4173. The tolerance of R_{SENSE} is ±1% [13]. The typical error of the MAX4173 V_{OUT} is ±0.5% [8]. The analog input resistance (R_{AIN}) of the ATmega169 is 100MΩ. Therefore, connecting the MAX4173 to the ADC by V_{OUT} shown in Figure 6 above, we get the percentage error due to the 12kΩ output resistance of the current sensing chip given by the following equation [8]

$$\%Error = 100\left(\frac{R_{LOAD}}{12k\Omega + R_{LOAD}} - 1\right). \quad (1)$$

Substituting R_{AIN} for R_{LOAD} in Equation 1, we get a %Error of roughly 0.01%.

Since a gain of 20V/V is being used, using the expression in Equation 5, we get the %Error in V_{OUT} due to the tolerance of R_{SENSE} to be 1%. Therefore, combining these errors, we get the total %Error of the V_{OUT} reading to be less than 2%. Since this voltage is divided by the gain and R_{SENSE} to get the current being sensed, this translates to an error in current measurement of less than 1%.

3.2.3 Voltage sensing circuit

As shown in Figure 5, the operating voltage of the DUT will be sensed across the entire load including the current sensing circuit. As specified in [6], the voltage across the load will be between 0 – 2 V. This analog voltage value will be read by the ADC pins of the microcontroller. The details of converting the analog voltage to a digital value that can be displayed on the LCD and logged by the computer are explained in section 4.2.3.

The maximum voltage reading at V_in is 3V, and since our ADC performs 10-bit conversions, the accuracy of our voltage reading is $3V/(2^{10})$ or roughly 3mV, which satisfies the requirements outlined in [6].

Figure 7 shows the circuit used by the AVR Butterfly to sense the voltage at the V_in pin, which is connected to the MFC Testbench circuitry as shown in Figure 5. To reduce loading by the circuit shown in Figure 7, we will be connecting a voltage buffer between the MFC Testbench circuitry and the V_in pin. An opamp will be used whose common-mode range and voltage swing include the range 0-2V when powered with a single supply.

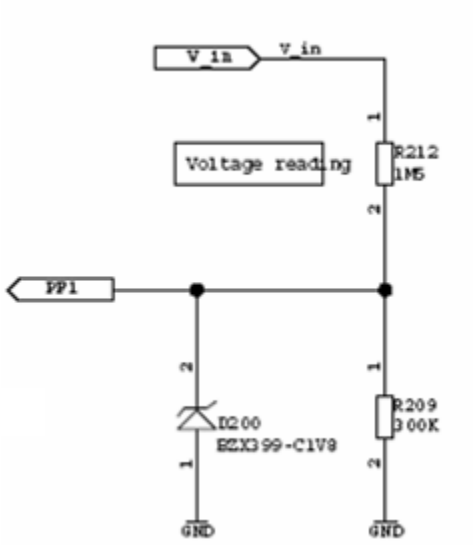


Figure 7: Schematic of Connection Between Voltage Sensing Wires and Microcontroller [2]

3.3 AVR Butterfly

The AVR Butterfly is an evaluation board for the ATmega169 microcontroller that contains extra circuitry for different sections of the microcontroller. In this section, we explain the features of the AVR Butterfly board that are being used by the MFC Labs Testbench.

3.3.1 Microcontroller

The on-board microcontroller is the ATmega169 which is an 8-bit microcontroller based on the AVR enhanced RISC architecture. Several features of the microcontroller are used in the development of the MFC Labs Testbench. The microcontroller has 64 pins that are connected to other modules as shown in the schematics provided in Appendix I. Some of the features of the microcontroller used for development of the MFC Labs Testbench include:

1) 16 kB of In-System Programmable flash memory which is the program memory space section and is arranged into 8k * 16 words. The program memory space is divided into the application program section and the boot program section.
2) 53 general purpose I/Os are used for several purposes including analog and digital communication with the microcontroller and interrupts.
3) 32 general purpose working registers that provide flexibility for programming the microcontroller.
4) A complete on-chip LCD controller with internal step-up voltage used for controlling the on-board LCD module.
5) Three flexible Timer/Counters with compare modes used for implementing the Real Time Clock (RTC), periodic updating of A/D values, and serial communication through the USART.
6) An 8-channel, 10-bit ADC used for measurement of voltage and current.
7) An SPI serial port used for programming and communicating with the microcontroller.
8) Internal and external interrupts used for detection of different events such timer/counter overflows, buttons, etc.
9) On-chip Debugging support and programming used for programming and debugging different sections of the microcontroller.

Please refer to Appendix I for complete schematics of different sections of the ATmegat169 and Appendix II for the pin assignment of the microcontroller for the MFC Testbench.

3.3.2 LCD Display and Toggle Switch

The AVR Butterfly is equipped with firmware for the menu and joystick for the user to browse the menu. The user is able to scroll through the menu by pushing the joystick

switch up, down, left, right, and downwards. The miniature joystick button is provided with the AVR Butterfly. The operations of the joystick are presented in Figure 8.

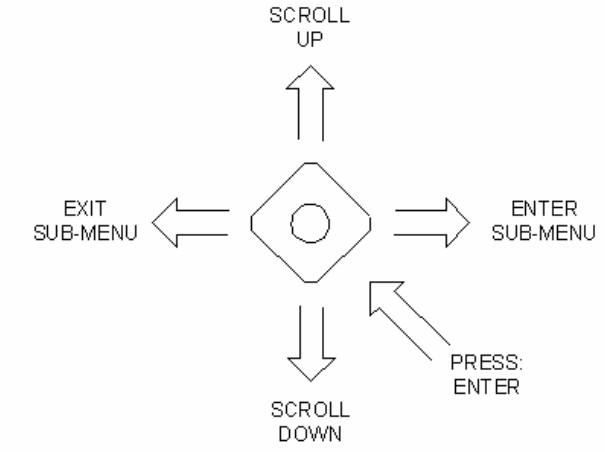


Figure 8: Operation of the Joystick to Control the menu [2]

The operation of the menu is presented in the flowchart in Figure 13.

3.3.3 Crystal Oscillator

The on-board 32.768 KHz crystal oscillator is connected to the microcontroller (pins XTAL1 and XTAL2) and is used by timer/counter2 to implement the RTC feature.

3.3.4 RS232 Level Shifter

The UART section of the microcontroller is used to communicate with the PC (LabView program) to provide the real-timer data logging feature. An on-board level shifter is provided on the AVR Butterfly board for communication using the RS232 cable. The schematic for the level shifter is provided in Appendix I.

3.3.5 Voltage Sensor Circuitry

The AVR Butterfly Port F pin 1 is configured for voltage measurement as described in sections 3.2.3 and 4.2.3.

3.4 Load Block

3.4.1 Manual load

The on-board manually controlled load will vary between 1Ω and 50KΩ and is connected to the MFC Testbench as shown in Figure 5. The potentiometer used will be the EVJC manufactured by Panasonic as shown in Figure 9.

Figure 9: EVJC Vertical Mounting potentiometer [11]

The EVJC has a 12mm square Two-in-One shaft which can handle 80N maximum push and pull strength applied to the shaft. The EVJC will be mounted vertically with respect to the enclosure box. The power rating of the EVJC is 0.5W for 0°C to 50°C with 100MΩ insulation resistance at 250Vdc.

The potentiometer will be calibrated and labeled for different resistance values by connecting to an external ohmmeter and labeling significant values.

3.4.2 Electronic variable load

For rigorous testing of the DUT, the Testbench setup will include the variable electronic load 3710A by Array Electronics Co. shown in Figure 10. This load was chosen because it is the only one in its price range that offers the variable load features required by the MFC Testbench.

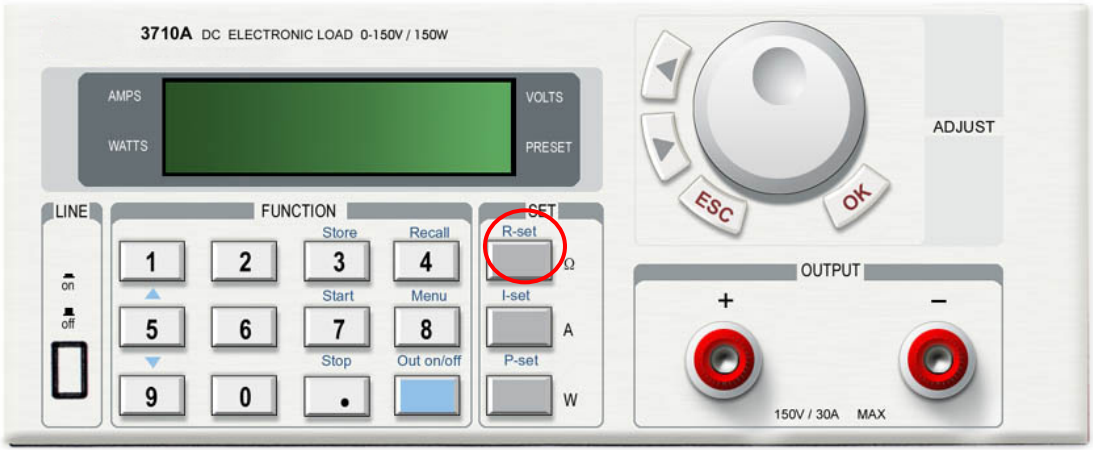


Figure 10. The front panel of the electronic load 3710A [10]

The 3710A is equipped with RS232 communication and can be programmed via LabVIEW to automate the loading process. The device can handle voltages ranging from 0-360V and currents ranging from 0-30A.

To operate the device in the constant resistance mode the user has to press the R-set button (in red circle on the Figure 10). The value of the desired resistance can be entered using key pads or by rotating the “Adjust” knob.

The load will be connected via an RS232 port to the ATmega169 on the MFC Testbench through the RS232 outlet. The positive and negative outputs of the load will be connected to the negative and positive terminals on the MFC Testbench by two banana plugs.

3.4.3 Switching circuit

As specified in [6], the user will be provided with choice of testing the DUT with a manually adjusted resistive load or an electronic variable load. The selection between the two loads will be achieved via the manual switch onboard the MFC Testbench. A generic 6A rated SPDT "center off" switch will be installed with COM pin connected to the DUT (or a battery) and A and B pins connected to the leads of the manual load and electronic loads respectively, as shown in Figure 5 .

3.5 Power Management

The MFC Testbench will be powered up using both AC and battery power, independently. A protective diode will be used to achieve plug polarity reversibility. A status LED will indicate the presence of power to the circuit.

3.5.1 AC Mode

Power will be introduced to the MFC Testbench using a 9V, 200 mA, AC wall-wart. Using a 9V wall-wart will allow us to upgrade the battery to a rechargeable battery in the next design iteration, and it will also provide stability for the DC regulator. The AC wall-wart will be connected to the MFC Testbench using a 2.1mm power jack and receptor mounted on the PCB. The delivered voltage will be stepped down through a DC converter circuit.

3.5.2 Power supply

Power will be regulated by 3V3, 100 mA low power regulator where capacitors maintain the stability of the regulator. Also power will be switched between the battery and the AC outlet automatically by detecting the higher voltage presented to the circuit. A 2A toggle switch will turn on and off the MFC Testbench. The location of the switch and power supply are shown in a schematic in Appendix I. The block diagram of the power supply is shown in Figure 11.

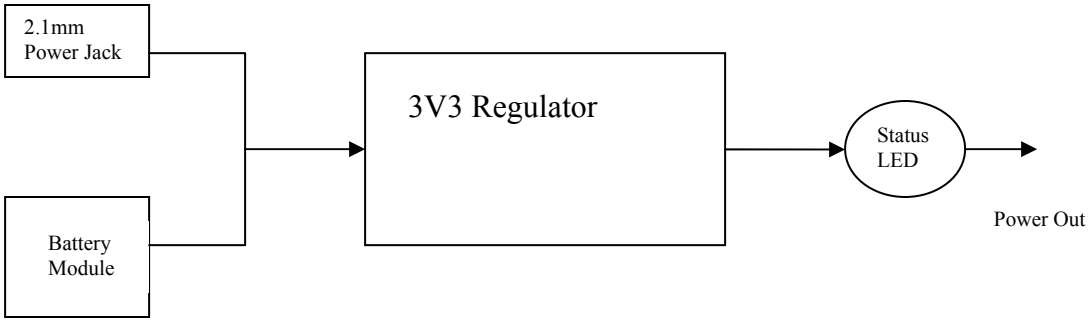


Figure 11: Power Block Diagram

3.5.3 Battery Mode

A 9V alkaline battery will power the testbench in the absence of the AC wall-wart. The battery output will be regulated by a precision regulator for constant power output.

3.5.4 Voltage Reference

A 3V3 precision DC-DC switching regulator will be introduced to output a 3V3 reference voltage for V_{REF} of the ATmega169 ADC. The MAX872 chip will be used for this purpose.

3.6 Fuel supply and Flow Meter

The fuel supply and flow meter are optional features of the MFC Testbench design, and their implementation will enhance our ability to test the prototype version of the DMFC. This section describes the experimental setup of the fuel supply and flow meter interface which will be implemented for testing of the prototype of the MFC testbench. The 5 % methanol-water solution has a low chemical reactivity and closely resembles water by its viscosity and flow characteristics. Hence, water will be used for testing of the implementation of the prototype's fuel supply and flow meter. Water will be stored in the plastic jug/bottle with fluidic tubing secured to the cap of the bottle to allow for the fuel flow. The bottle will be elevated if necessary to allow the flow of the water under the force of gravity. The user will be provided with a fluid flow meter with direct read scale capable of reading the flow rate of 10 to 120 mL/min with accuracy of ± 5% of fixed scale. The flow meter will also be able to operate under testing conditions of increased temperature and pressure. The flow meter C-32461-32 by Cole-Parmer is similar to that presented in Figure 12. This flow meter can operate in maximum temperature up to 65 C, and pressure of 100 psi meeting the functional specifications. The fluid touching materials are acrylic, brass, Buna N and stainless still, which are chemically rated as A-Excellent resistant to weak water-methanol solutions [4].

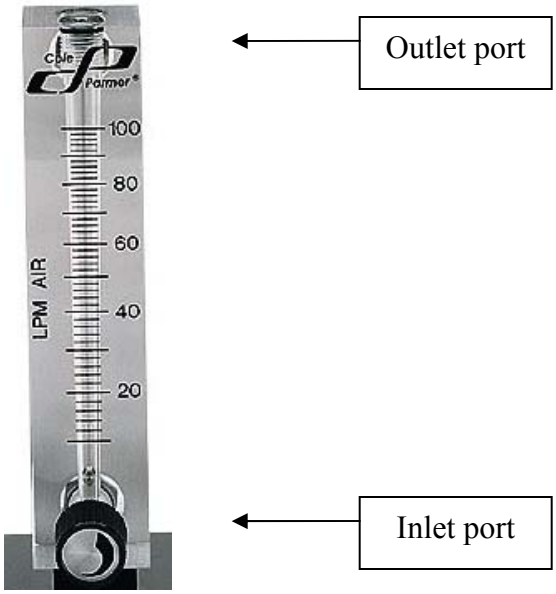


Figure 12. Direct read flow meter with flow control valve [5]

This acrylic flow meter is equipped with two extended inlet and outlet ports and two mounting holes. The flow meter will be securely mounted to an external panel/stand. The flow of the methanol-water solution will be directed through the plastic tubing attached to the bottle cap and into the inlet port of the flow meter and out of the outlet port of the flow meter to the tubing connector from the DUT. The flow meter will register the flow rate by the stainless steel float balancing at the corresponding mark on a metric-reading scale.

3.7 DUT Interface

The user will provide the DUT, which is a direct methanol fuel cell, in its appropriate enclosure for voltage and current measurements. The MFC Testbench will be equipped with two banana jacks and two insulated wires with banana plugs on one end and alligator clip on the other end for easy assembly to take measurements. The fluidic tubing will be connected to the DUT's fluidic ports by tubing union connectors.

4 Firmware Design

4.1 High level design

The firmware is structured as a state machine as shown in Figure 13 below.

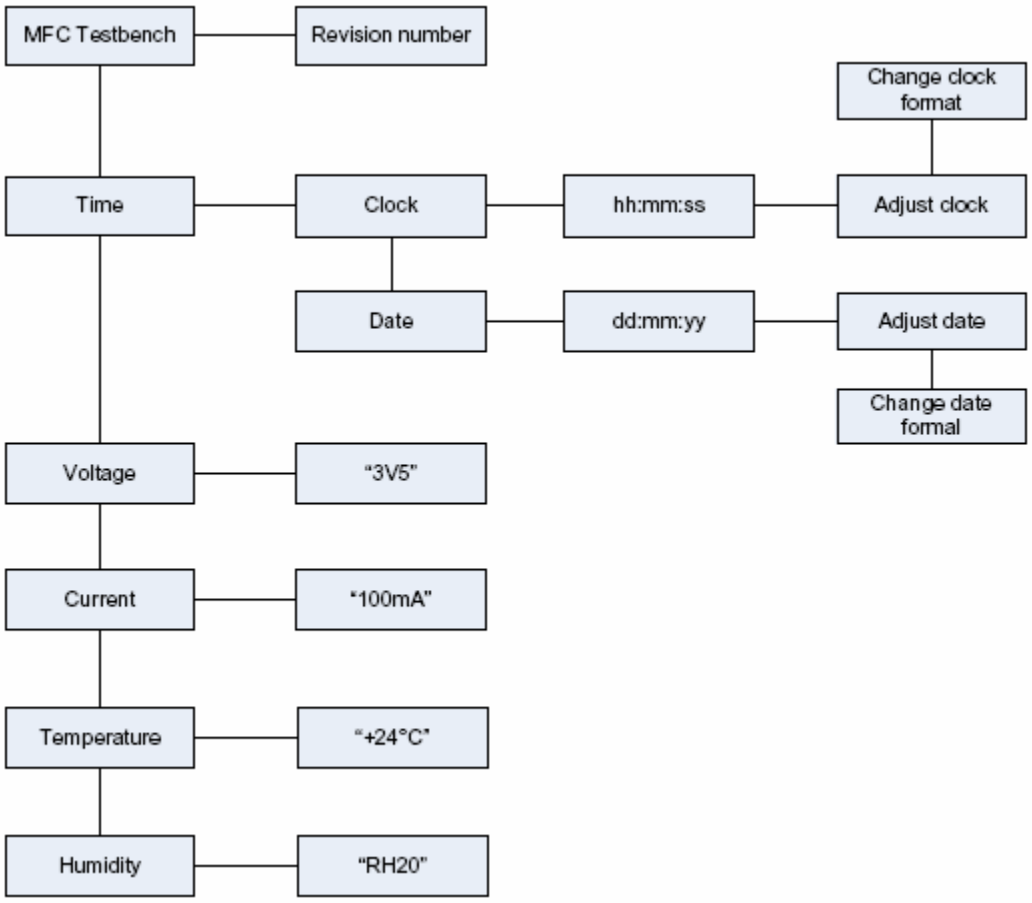


Figure 13. Menu flowchart for the LCD display

To shift between states in the most left column the user has to press joystick up or down. To select options in a submenu, such as "Clock" or "Date" the user has to press right or right and down respectively. When "Adjust clock"/"date" appears on the screen, the user has to press the joystick down to set the desired time/date.

The algorithm used to switch between different states is done according to the flowchart shown in Figure 14.

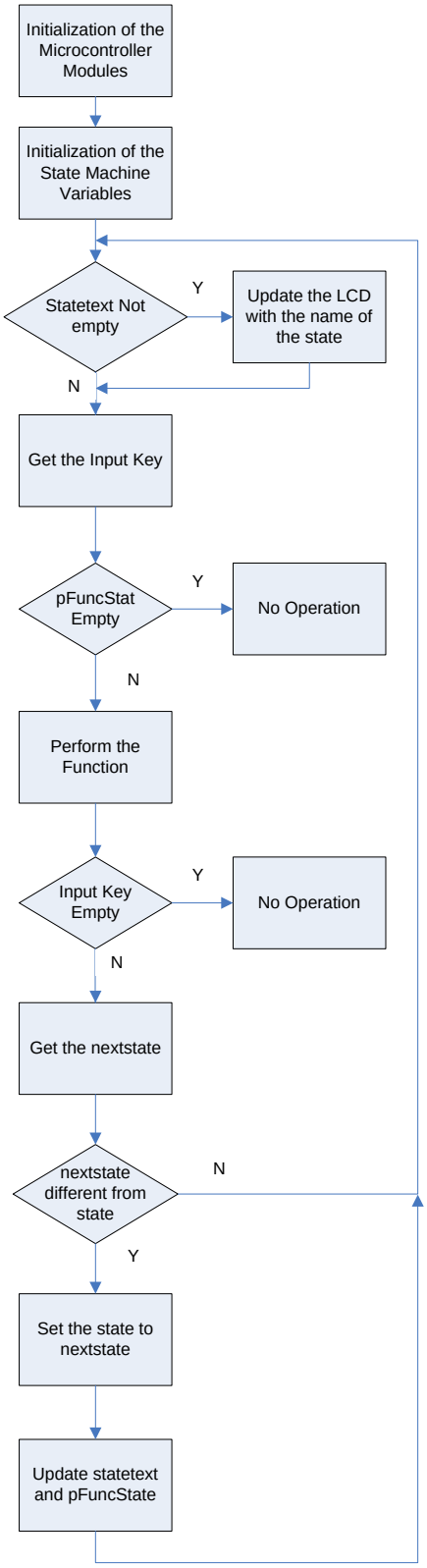


Figure 14: Flowchart of State Selection Logic in Firmware

Each state has four different attributes:

1) State: a number that determines the state.
2) Statetext: determines the name of the state to be displayed on the LCD.
3) pStateFunc(): a function that determines what needs to be done in a particular state.

The variable nextstate holds the value of the next state determined by the toggle switch interrupts. Some states do not perform any specific function but only display the name of the state. Other states perform functions such as measuring the temperature, humidity, and voltage. Table 2 shows the name and function of all the states.

In the initialization section before entering the infinite state machine loop, we initialize different sections of the ATmega169 microcontroller that are used, namely, the internal and external oscillator, timer/counter0, timer/counter1, timer/counter2, the buttons, the UART functionality, and the LCD. In addition, the state attributes are initialized to ST_AVRBF which is the welcome state of the device.

Table 2. Names and Functions of states

State	Statetext	State Function
ST_AVRBF	MFCLABS TESTBENCH	Display the Name
ST_AVRBF_REV	No Name	Display the software revision
ST_TIME	Time	Display the Name
ST_TIME_CLOCK	Clock	Display the Name
ST_TIME_CLOCK_FUNC	No Name	Display the clock
ST_TIME_CLOCK_ADJUST	Adjust Clock	Display the Name
ST_TIME_CLOCK_ADJUST_FUNC	No Name	Adjusting the clock value
ST_TIME_CLOCKFORMAT_ADJUST	Change Clock Format	Display the Name
ST_TIME_CLOCKFORMAT_ADJUST_FUNC	No Name	Adjusting the clock format
ST_TIME_DATE	Date	Display the Name
ST_TIME_DATE_FUNC	No Name	Display the date
ST_TIME_DATE_ADJUST	Adjust Date	Display the Name
ST_TIME_DATE_ADJUST_FUNC	No Name	Adjusting the date
ST_TIME_DATEFORMAT_ADJUST	Change Date Format	Display the Name
ST_TIME_DATEFORMAT_ADJUST_FUNC	No Name	Adjusting the date format
ST_HUMIDITY_SHT11	Humidity SHT11	Display the Name
ST_HUMIDITY_SHT11_FUNC	No Name	Measure and display the humidity value
ST_TEMPERATURE_SHT11	Temperature SHT11	Display the Name

State	Statetext	State Function
ST_TEMPERATURE_SHT11 _FUNC	No Name	Measure and display the temperature value
ST_VOLTAGE	Voltage	Display the Name
ST_VOLTAGE_FUNC	No Name	Measure and display the voltage value
ST_CURRENT	Current	Display the Name
ST_CURRENT_FUNC	No Name	Measure and display the current value
ST_DATALOG	Datalog	Display the Name
ST_DATALOG_FUNC	No Name	Send the voltage, current, humidity, and temperature to the RS232 port

4.2 Low level design

The details of implementing the states shown in Figure 14 are explained as low-level design below.

4.2.1 Clock and Date

The 8-bit Timer/Counter2 of the ATmega169 is used to implement the RTC feature of the device. Timer/Counter2 is connected to an external crystal 32.768 KHz oscillator. A prescaler of 128 is used to reduce the clock value to 256 Hz. The counter value increments every 1/256 seconds. Every time the 8-bit counter generates an overflow, an interrupt is fired, indicating the passage of 1 sec.

4.2.2 Measuring Humidity and Temperature

To measure the DUT operating humidity and temperature using the SHT11 IC, the clock line of the SHT11 is connected to the microcontroller as described in Appendix II.

When the measurement data from the SHT11 has been read by the microcontroller, the following calculation is required to interpret the humidity data, where 12 bits are used. SO_{RH} represents the SHT11 serial output for a relative humidity measurement and c_1, c_2, and c_3 are constants.

$$RH_{linear} = c_1 + c_2 \bullet SO_{RH} + c_3 \bullet SO_{RH}^{\ 2} \quad (2)$$

SO_{RH}	c_1	c_2	c_3
12 bit	-4	0.0405	$-2.8 * 10^{-6}$
8 bit	-4	0.648	$-7.2 * 10^{-4}$

To interpret temperature data, the following calculation is required by the microcontroller where 14 bits are used. SO_T represents the SHT11 serial output when a temperature measurement is taken and d_1 and d_2 are constants.

$$\text{Temperature} = d_1 + d_2 \bullet SO_T \tag{3}$$

VDD	d_1 [°C]	d_1 [°F]
5V	-40.00	-40.00
4V	-39.75	-39.50
3.5V	-39.66	-39.35
3V	-39.60	-39.28
2.5V	-39.55	-39.23

SO_T	d_2 [°C]	d_2 [°F]
14bit	0.01	0.018
12bit	0.04	0.072

4.2.3 Measuring Voltage and Current

The voltage and current are measured using the internal 10-bit ADC of the ATmega169 microcontroller. The pFuncState of the voltage measurement initializes the A/D to measure the voltage by specifying the analog input pin to the A/D that is connected to the incoming voltage to be measured (refer to section 3.2.3). After the voltage reading by the ADC is complete, the voltage is calculated using the reference voltage value (V_{ref}) and the following formula:

$$V_{IN} = \frac{ADC}{1024} \times V_{ref} \tag{4}$$

The current measurement process is similar to voltage measurement except that Equation 5 is used to interpret the voltage read by the ADC from the MAX4173 V_{OUT}. GAIN is 20V/V for the MAX4173T current sensor, and $R_{SENSE} = 0.1\Omega$ for the MFC Testbench [8].

$$I_{LOAD} = \frac{V_{OUT}}{GAIN * R_{SENSE}} \tag{5}$$

Timer/counter0 is used to update the values of the voltage and current after a specified period of time.

4.2.4 Data Logging

The data logging function uses Timer/Counter1 to control the frequency of updating the data to be sent. When the user enters the data logging mode, the Timer/Counter1 overflow interrupt is enabled, and the voltage, current, humidity, and temperature are sent using the UART every time an interrupt is received. The UART is configured to send the information with a baud rate of 9600 bps. Using Timer/Counter1, we can control the frequency of sending a new stream of data.

The SHT11 humidity and temperature sensing board must perform no more than two measurements per second to prevent overheating. Therefore, by specifying the Timer/Counter1 value, we ensure that the data is set with at least 1 sec delay between consecutive data streams.

To communicate with the PC, a protocol has been developed which is in the format of "**s**aVaabAbbhhtt**r**" where the character "s" specifies the start of a new stream of data, "r" specifies the end of a new stream of data, "aVaa" is the voltage value, "bAbb" is the current value, "hh" is the humidity value in RH, and "tt" is temperature value in degrees centigrade.

4.2.5 LCD Control

The testbench is equipped with an onboard display to allow the users to see the time and date as well as the operating parameters of the DUT. When the MFC Testbench is turned on, the LCD is activated and the user has the option of scrolling through the menu by pressing the joystick button to display the measurement value of interest.

The on-board display is provided by a seven digit LCD and ATmega169 LCD controller with integrated LCD drivers. Since ATmega169 can drive 100 segments, some of the segments on the LCD display are not connected as shown in Figure 15 [1]. For the MFC Testbench, we are using alphanumerical symbols only.

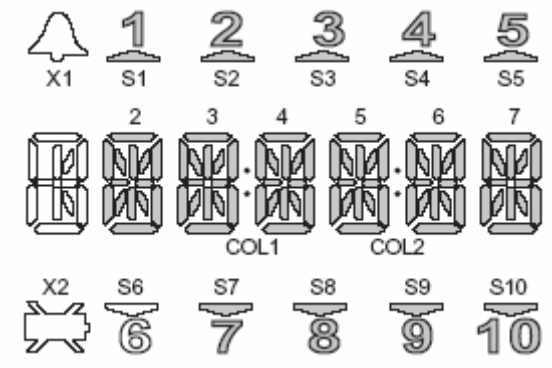


Figure 15. LCD module with active segments

The LCD segments are individually controlled and are arranged in digits as shown in Figure 16.

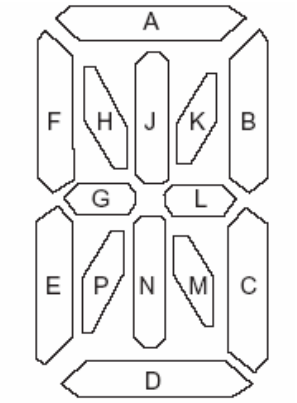


Figure 16. LCD digit with segments and reference letters. [1]

The 16 segment registers which control individual segments of the LCD are defined as (LCDDR19:0) and are activated by setting the corresponding bit.

The control of the LCD is done through the AVR LCD drivers. The main LCD Driver functions are summarized in Table 1.

Table 3. LCD driver functions [1]

Function Name	Arguments	Return	Description
LCD_Init (global)	void	void	Initialize the LCD display Data Buffer, which is used to buffer the LCD Data Registers. The LCD frame rate and contrast is selected. The segment lines and common lines are configured.
LCD_WriteDigit (global)	unsigned char c, unsigned char digit	void	The ASCII character passed in the "c" argument is converted to a LCD Segment Control Code (SCC). The SCC is the data that maps the ASCII into LCD Segment Symbols. The SCC is copied into the LCD Display Data Buffer. (The actual update is handled by the LCD_SOF_interrupt routine.) Digit is the number of the digit that is desired accessed. The value of digit is in this implementation limited from 2 to 7.
LCD_AllSegments (global)	unsigned char input (used as bool)	void	Clear or sets all the segments of the LCD (updating the LCD Display Data Buffer only – the actual update is handled by the LCD_SOF_interrupt routine).
LCD_SOF_interrupt (local, interrupt service routine)	void	void	Latches the LCD Display Data Buffer to the LCD Data Registers. The latching is depending on the LCD_timer variable and the LCD_status.updateRequired variable.

5. Application Software Design

5.1 Introduction

The MFC Testbench application software is a graphical user interface (GUI) program which allows the user to log the operating data of the DUT including date, time, voltage, current, temperature, and relative humidity into a Data Log File. The GUI program also controls the value of the variable load by allowing the user to enter a Load Cycle Document which lists the discrete load values and the time for which each is activated. In addition, the program displays the most recent data instances on the computer screen for the user to view.

LabVIEW 6 will be used to implement the GUI program due to its ease of communicating with instruments and its support of sophisticated GUI functions.

5.2 High-Level Design

As shown in Figure 17, the LabVIEW front panel will present the user with buttons and two listbox displays. The first display presents the current value of load being applied to the DUT and the time in seconds for which it will be applied. The second listbox displays the most recent test data. The buttons allow the user to start or end test cycles.

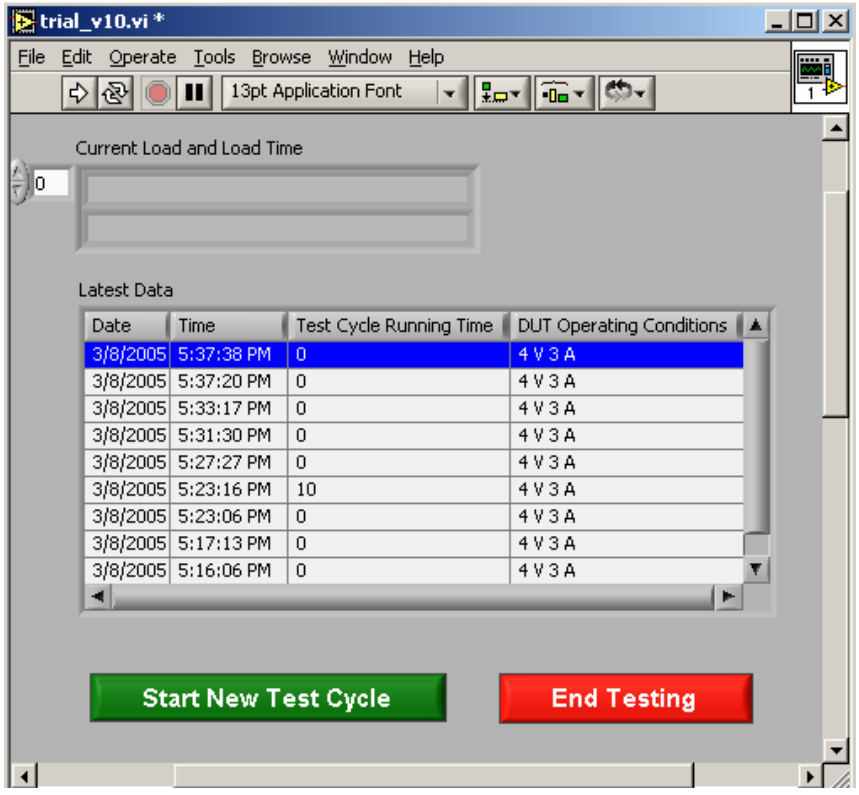


Figure 17: Prototype Front Panel of GUI Program

5.2.1 State Diagram

The GUI program is designed as a state diagram with six states, as shown in Figure 18. Aside from the initial (Idle) and final (Exit) states, the "Data Log File Entry" state waits until the user enters a Data Log File then moves on to the "Load Cycle File Entry" state where the user may or may not enter a file which specifies which loads are applied to the DUT and the length of time for which it should be applied. If a Load Cycle File is entered, the program varies between the "Load Control" and "Data Logging" states until the test cycle has completed or the user presses the "End" button. If a Load Cycle File is not entered, the program enters the "Data Logging" state until the user presses the "End" button. In the "Load Control" state, the program reads the Load Cycle File, determines which load is desired and the time for which it should be set, and communicates with the variable load to set the desired load. In the "Data Logging" state, the program uploads the DUT operating data from the microcontroller through the serial port, displays this data on the front panel, and sends the data to the specified Data Log File.

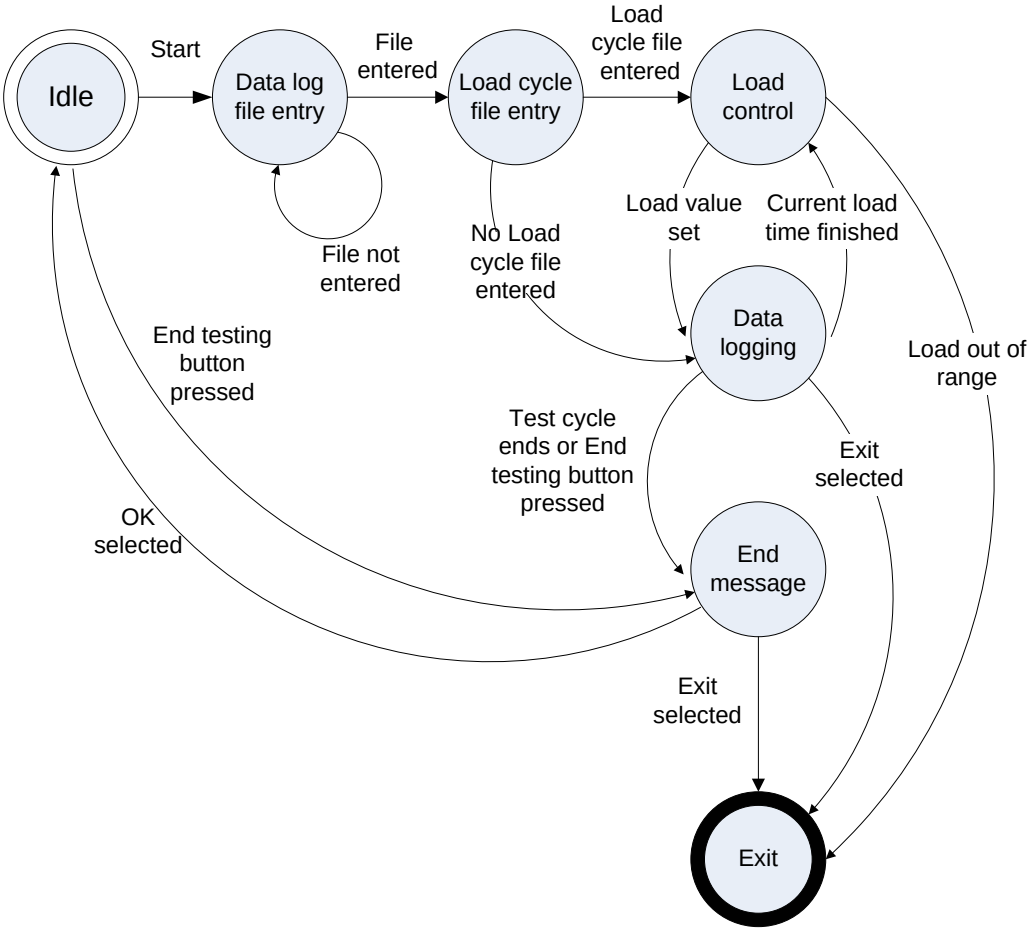


Figure 18: State Diagram of GUI Program

When the user presses the "End Test Cycle" button, the program will terminate all communication with peripheral devices and close the Load Cycle and Data Log files.

The program will then display a message indicating that the test cycle was aborted and give the user the option of doing further testing or exiting the program.

5.2.1.1 Load Cycle File Format

The user will be asked whether or not he wants to enter a Load Cycle File after pressing the "Start" button. If the user says yes, the Load Cycle File entered should have the format shown in Table 4 where each entry specifies a load value and a time for which that load should be applied. The two columns must be separated by tab delimiters.

Table 4: Sample Load Cycle File Format

Load Value (Ω)	Time (s)
300	120
20	10000
…	….

5.2.1.2 Data Log File Format

When the GUI program is in the idle state, after pressing the "Start" button, the user will be prompted to enter a Data Log File by the appearance of a window allowing the user to browse the computer system's files as shown in Figure 19 below. This Data Log File can be an existing text file or can be created by the GUI program. If the user refuses to enter a Data Log File by pressing "Cancel", the program will terminate.

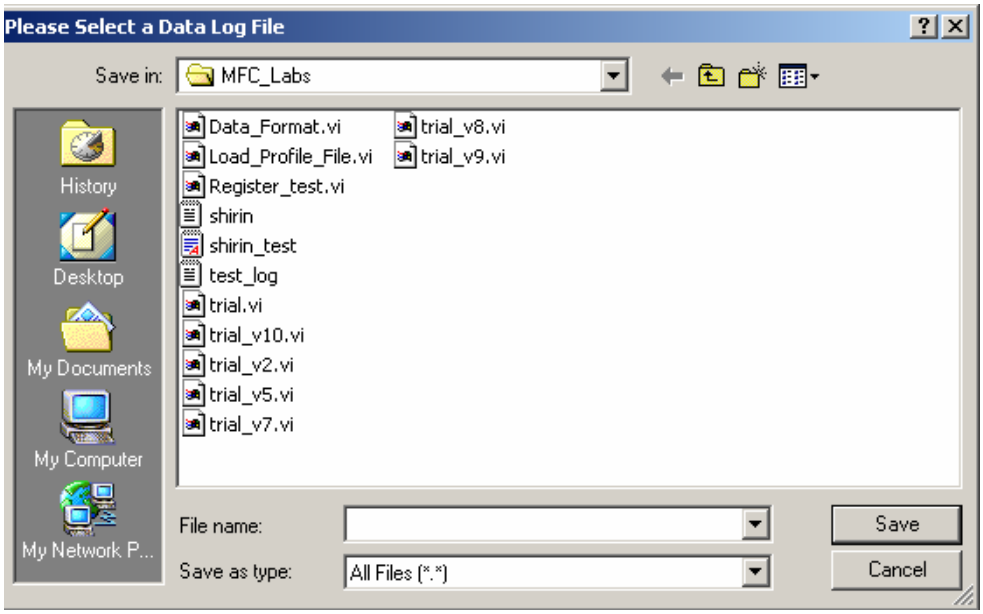


Figure 19: File Browsing Window

The Data Log File will have the format shown in Table 5 and as described in [6]. Each entry in the table specifies the operating conditions of the DUT. The columns will be

separated by tab delimiters to allow the user to open the file using a spreadsheet program for easy plotting and reporting of test data.

Table 5: Sample Data Log File Format

Date	Time	Total Test Time (s)	Voltage (V)	Current (mA)	T (ºC)	RH (%)
3/6/2005	4:57:03 PM	0	2.00	100	25	33
3/6/2005	4:57:13 PM	10	2.12	95	25	35
3/6/2005	4:57:23 PM	20	2.01	100	25	34

5.2.1.3 Implementation of "End Test Cycle" Button

In order to terminate communication with all peripheral devices before ending a test cycle, the program will have to wait for the most recent data upload to complete before the test cycle can be completed. Therefore, the maximum response time of the program to the "End Test Cycle" button is 10 s. If the user wants to terminate the program more rapidly, the "Stop program" button can be used as provided by LabVIEW to end the program instantaneously.

In LabVIEW's dataflow environment, in order to ensure that the program can respond to the pressing of the "End Test Cycle" button at various points of its operation, local variables will be declared for this button in addition to its control button.

5.2.2 Layers of Programming

The GUI program controls the flow of states as shown in the state diagram of Figure 18. Some of the program's tasks will be encapsulated in sub-programs as shown below in Figure 20.

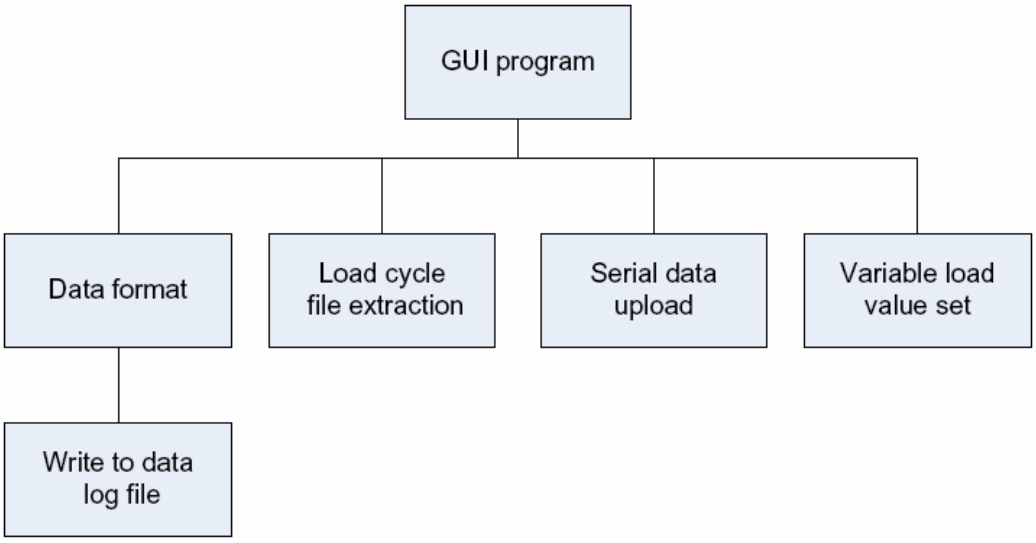


Figure 20: Program Hierarchy for GUI Program

The Data Format sub-program will be called in the "Data Logging" state and will format the data for displaying on the GUI program front panel and will write the data to the Data Log File using the format described in Tables 6 to 10. Since this sub-program writes to the Data Log File, the Write_File.vi provided by LabVIEW will be used as a sub-program to handle data writing. The algorithm for the operation of the Data Format sub-program is given in Figure 21.

The Load Cycle File Extraction sub-program will be called in the "Load Control" state and will read the entire contents of the user-entered Load Cycle File, store it into an array, and send it to the GUI program.

Table 6: Data Format Sub-program summary

Data Format	
Inputs	**Input Type**
Input Data String	String
Input Data Log File Path	File Path
Input Beginning Time	Float
Outputs	**Output Type**
Output Spreadsheet String	String
Test Cycle Time (s)	Integer
Test Cycle Time (min)	Integer
Output Array	String Array
Sub-Programs	Write Characters to File
Other	See Fig. X for flowchart

Table 7: Load Cycle File Extraction Sub-Program Summary

Load Cycle File Extraction	
Inputs	
None	
Outputs	**Output Type**
File Array	String Array
Size of File Array	Integer
Sub-Programs	Read Characters from File
Other	

Table 8: VISA Configure Serial Port Sub-Program Summary

VISA Configure Serial Port	
Inputs	**Input Type**
VISA resource name	String
Baud rate	Integer
Data bits	
Parity	
Error in	Error Cluster
Outputs	**Output Type**
Duplicate VISA resource name	String
Error out	Error Cluster
Sub-Programs	None
Other	Provided by LabVIEW

Table 9: VISA Read Sub-program summary

VISA Read	
Inputs	**Input Type**
VISA resource name	String
Byte count	Integer
Data bits	
Outputs	**Output Type**
Duplicate VISA resource name	
Read buffer	
Read count	Integer
Error out	Error Cluster
Sub-Programs	None (?)
Other	Provided by LabVIEW

Table 10: Variable Load Value Set Sub-Program Summary

Variable Load Value Set	
Inputs	**Input Type**
Load Value Serial Port Address	
Outputs	**Output Type**
None	
Sub-Programs	None
Other	Provided by Array Electronic Co.

Figure 21 presents a flowchart describing the sequence of operations used to implement the Data Format sub-program.

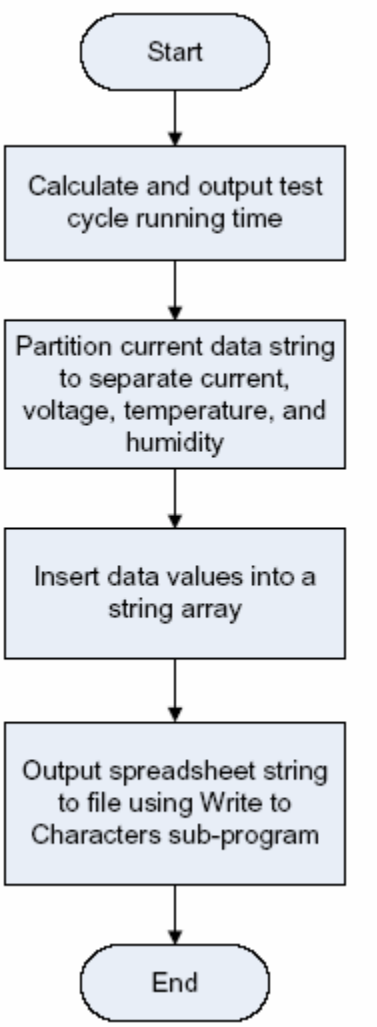


Figure 21: Flowchart of Data Format Sub-Program Operation

5.3 RS-232 Communication with Microcontroller

The GUI program will communicate with the microcontroller using RS-232 serial communication. RS-232 was chosen because of its availability on most computers and its availability on the microcontroller by the SPI interface.

The number of data uploads will be determined by reading the second column of the Load Cycle File which gives the time for which a load is applied in seconds. This value will be divided by the period of data uploading used (10 s) to determine the number of data uploads that will occur. If the second column of the Load Cycle File is not a multiple of 10 s, the quotient will be taken as the number of data uploads, ignoring the remainder term.

For information on the protocol used by the microcontroller for sending DUT operating conditions data, refer to section 4.2.4. The GUI program will store the DUT operating conditions data in string format in the Data Format sub-program. After inputting the data, the data will be stored into in a string array. The string array will then be turned into the spreadsheet string format before being saved to the Data Log File and output on the front panel display. The Data Format program will also output the string array to the main GUI program which will use this as a dummy array for updating the front panel list of Latest Data. In LabVIEW's dataflow environment, a tunnel will have to be used to link data between consecutive data upload sequences to keep this front panel display up to date.

6 Mechanical Design

6.1 Electromechanical enclosure

All electronic components will be housed in the desktop electronic instrument box KEU-7 series by Plastic Electronics as shown in Figure 22.

Figure 22. Desktop electronic instrument box KEU-7 series [3].

Fabricated from ABS plastic and aluminum, the box is safe and easy to machine. The sloped position of the front recessed panel allows for better view of the device and easy access to operational parts. The front panel is attached by four screws and is easy to remove for installation of the internal components. The back of the box also has an aluminum panel for mounting parts. The bottom of the box is also removable with four screws and will allow the user to access the internal parts if needed for service or repair.

6.2 Electromechanical user interface

This section addresses the design of the user interface for control and operation of the MFC Testbench and for the connection/communication interface between the testbench and other electronic components of the system, such as the computer, DUT, and electronic variable load. The dimensions of the front panel are 14.9 cm x 10.5 cm allowing sufficient room for placement of the LCD display, control buttons, switches and other components. The dimensions of the back aluminum panel are 12 cm x 3.5 cm, allowing sufficient room for mounting 4 banana jacks, RS-232 jack, and power supply plug. A 9 V battery will be housed inside the box with access from the back or side of the box. The necessary interface components and their placement are summarized in Table 11.

Table 11. Components for the User Electromechanical Interface

User interface/functionality	Component	Placement
Turning the device on/ff	On/Off toggle switch or push button	Front panel
Visual display of the measurements and operational state	LCD	Front panel
Menu operations	Joystick	Front panel
Selection of the power supply option (AC, DC) and battery recharge	3 position switch	Front panel
Selection between manual and electronic load	SPDT "center-off" switch	Front panel
Manually varying the load	Rotary potentiometer knob	Front panel
Interfacing with DUT for measuring voltage and current (negative and positive terminals)	2 color coded banana jacks (red for positive terminal, black for negative terminal); 2 isolated wires with color matched banana plugs and alligator clips.	Back of the box
Interfacing between testbench and variable electronic load	1 black banana jack; 1 isolated wire with black banana plugs.	Back of the box
Communication between computer and testbench	RS232 jack	Back of the box
Power supply connection	Power supply connector	Back of the box
Access to 9 V non-rechargeable battery	9 V battery compartment	Back/side of the box
Sensing ambient humidity and temperature	Humidity/Temperature sensor	Back of the box

7 Testplans

7.1 Hardware Testplan

To facilitate testing of the MFC Testbench, first each module presented in Figure 2 will be tested separately, then as functioning in the device.

7.1.1 Component and Connection Verifications

Each module will be checked for the following items before having the power connected in the order specified.

1. Check the parts numbers with its corresponding location and numbers on the schematic and PCB in Appendix I.
2. Check the location of all color-coded wires as presented on the schematics.
3. Check for positive voltage and ground connections.

7.1.2 Power Supply Testplan

The power supply testing will be performed for proper operation under Thermal, Electrical, and Vibrational stresses. Also the circuit powered by the battery will be left to run for 8 hours to test the performance of the MFC Testbench in battery operated mode.

Thermal and Electrical Variation

The power supply will be exposed to up to 50°C while the output voltage will be closely monitored for variation. The same procedure will be expected for 0°C. The temperature will be monitored using an external thermometer. The auxiliary AC power supply will be exposed to the same extreme temperatures while it is connected to the power supply circuit.

The input voltage of the regulators will be altered to their maximum, minimum and nominal voltage rating to determine any possible fluctuation in the power output.

a) Vibration Effects

The power jack and battery connector will be shaken by connecting and reconnecting the connectors. While power is applied to the circuit, the test engineer will tap the power module and voltage routings with a light hammer to ensure the connectivity of traces.

7.1.3. Temperature and Humidity Sensors

Sensor wires will be examined for EMC and thermal variation while they are connected to the microcontroller, and the reading will be confirmed with an external temperature and humidity sensors.

7.2 Firmware Testplan

The ATmega169 daughter board from Atmel has sample codes for testing different sections of the ATmega169 microcontroller. Using the STK 500 Evaluation board, the microcontroller should be programmed to test the Interrupts, Timers, and A/D conversion independently. The generic Atmel LCD driver will be loaded to the EPROM of the daughter board which tests the LCD for possible malfunctions. Debugging will be done during the programming.

7.3 Software Testplan

When testing the GUI program software, we will test all normal operating conditions according to the state diagram in Figure 18 and some additional conditions that may take place when the user runs the program. The test cases will include the following steps.

1) The user starts the program and selects the "Start Testing" Button. If no Data Log file is selected, the program will wait at that point until the file is selected.
2) After the user has entered the Data Log File, the request for load cycle file entry will appear. The user enters the file or does not enter the file, which results in the program starting the load changing – data logging cycle or simply entering data logging.
3) If the test cycle ends (the end of the load cycle file) or the "End Testing" button is pressed, the program will end with an end message.
4) If user presses the OK button after the end message appears, the program goes into the idle state and waits for the user to press Start again.
5) If the user presses "Exit", the program terminates.
6) When in the idle state and user presses the "End of testing" button, the program should stop and display the end message.
7) If the user presses the Exit button when the program is in the data logging process, the program will exit.
8) If the load specified by the user is out of range, the program displays an error message and terminates.
9) If the data during data logging is out of range, the program displays an error message and terminates.

When debugging, we will consider the following scenarios:

1) What happens if the user presses "Start" and "End" buttons at the same time?
2) We will try to stop the program at various points of operation and check that such actions do not corrupt the Load Cycle and Data Log Files.
3) We will run the software overnight to check its performance over long periods of time.
4) We will test the sub-programs described in Figure 20 by
    a) verifying that the data writing format is preserved at various points of operation;
    b) verifying that the load cycle file name is extracted properly;

c) verifying that serial data from the serial port uploads properly by checking the values using a DMM and hygrometer against the data written in the Data Log File;
d) verifying the variable load value set and communication through serial port by varying the values and observing the data written in the Data Log File.

7.4 Fuel supply and flow meter Testplan

1. The complete setup for fuel supply including plastic bottle, tubing and flow meter will be tested with water.
2. The plastic bottle will be filled with water and elevated to generate flow through the fluidic path under the force of gravity. The flow path will end with the empty plastic bottle or laboratory glassware with mL scale where the water will collect.
3. The flow process will be timed and the water level at the container will be measured to determine the experimental flow rate.
4. The test will be deemed successful if both conditions are true:
    a) The flow of the water is achieved
    b) The experimental flow rate of the water is within 10 % of the reading on the flow meter.

References

[1] Atmel Corporation Products, 2004, "AVR065: LCD Driver for the STK502 and AVR Butterfly." Accessed on March 13, 2005 at <http://www.atmel.com/dyn/resources/prod_documents/doc2530.pdf>

[2] Atmel Corporation Products, 2003, "AVR Butterfly evaluation kit. User guide." Accessed on March 13, 2005 at <http://www.atmel.com/dyn/resources/prod_documents/doc4271.pdf>

[3] PacTec Divison of LaFrance Corp., 2005, "KEU-7." Accessed on March 13, 2005 at <http://www.pactecenclosures.com/Plastic-Enclosures/KEU-7.html>

[4] Cole Parmer Instrument Company, 2000, "Chemical Compatibility." Accessed on March 13, 2005 at <http://www.coleparmer.com/techinfo/chemcomp.asp> and conversation with application support on Wednesday March 9, 2005.

[5] Cole Parmer Instrument Company, 2003, "Valved Acrylic flow meter." Accessed on March 13, 2005 at <http://www.coleparmer.com/catalog/0304_pdf/A-0466.pdf>

[6] MFC Labs. *Functional Specifications for a Micro Fuel Cell Testbench.* February 2005.

[7] Sensirion: the Sensor Company, July 2004, "SHT1x/SHT7x Humidity Temperature Sensor." Accessed on March 13, 2005 at <http://www.sensirion.com/en/pdf/Datasheet_SHT1x_SHT7x.pdf>

[8] Maxim Integrated Products, 2004, "MAX4173F-MAX4173T Datasheet." Accessed on March 13, 2005 at <http://pdfserv.maxim-ic.com/en/ds/MAX4173F-MAX4173T.pdf>

[9] Atmel Corporation Products, 2005, "8-bit AVR Microcontroller with 16K Bytes In-System Programmable Flash." Accessed on March 13, 2005 at <http://www.atmel.com/dyn/resources/prod_documents/doc2514.pdf>

[10] Circuit Specialists Inc., 2005, "Programmable DC Electronic Load (CSI3710A)." Accessed on March 13, 2005 at < http://www.web-tronics.com/3710aprogdce.html>

[11] Panasonic Canada Inc., 2000, "Rotary Potentiometers/EVJC/EVJY." Accessed on March 13, 2005 at <http://www.panasonic.com/industrial/components/pdf/aok0000ce9.pdf>

[12] ECROS Technology, 2004, “AVR Butterfly Carrier.” Accessed on March 13, 2005 at <http://www.ecrostech.com/Products/Butterfly/Intro.htm>

[13] Ohmite Mfg. Co., 2000, “10 Series Lo-Mite Molded Silicone Axial Lead Wire Element Resistors.” Accessed on March 13, 2005 at http://www.ohmite.com/catalog/pdf/10_series.pdf

[14] Maxim Integrated Products, 1997, “MAX872 Voltage Reference.” Accessed on March 13, 2005 at <http://pdfserv.maxim-ic.com/en/ds/MAX872-MAX874.pdf>ppendix I – Schematics

Appendix I - Schematics

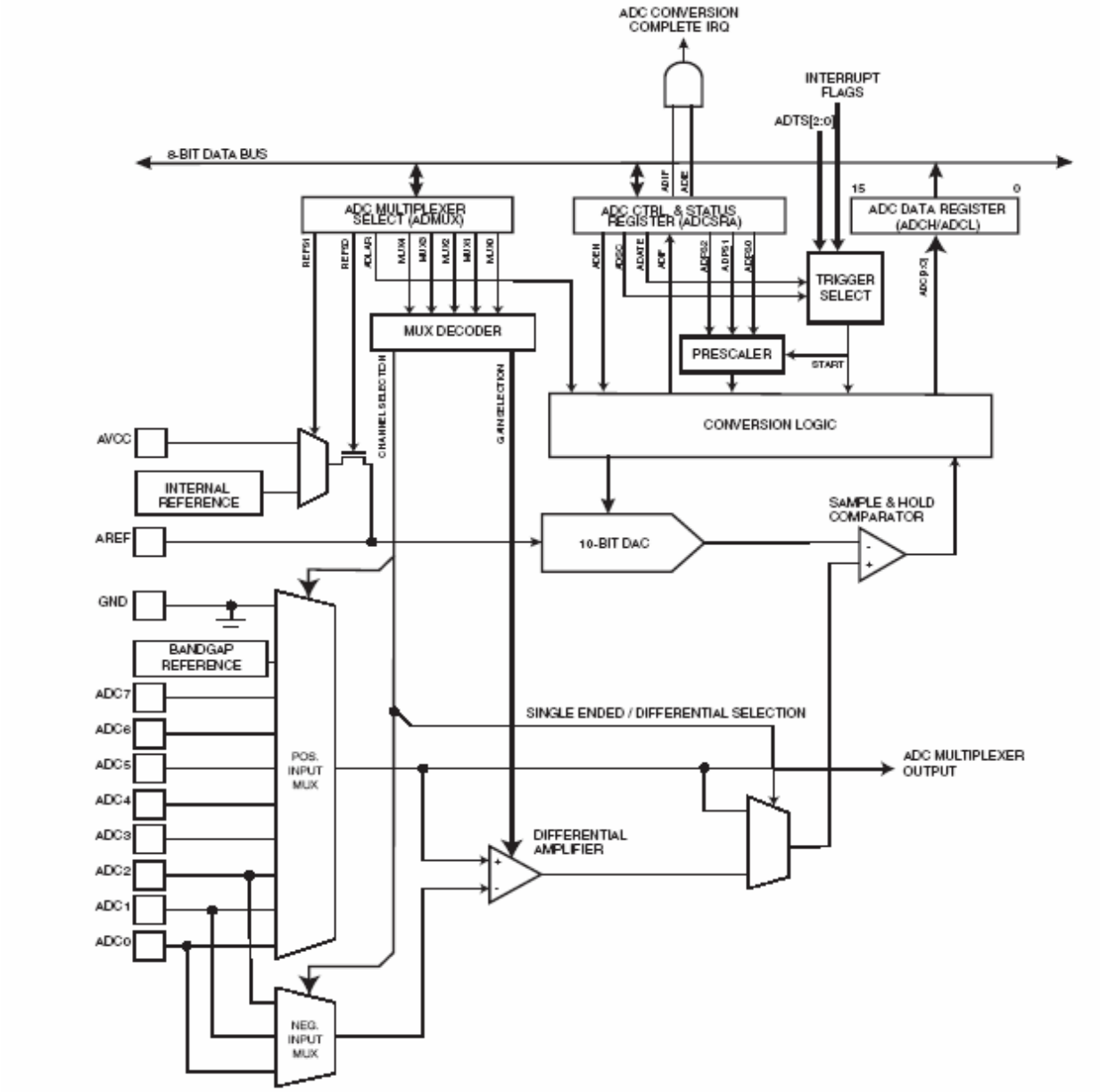


Figure 23: Block Diagram of ATmega169 ADC [9]

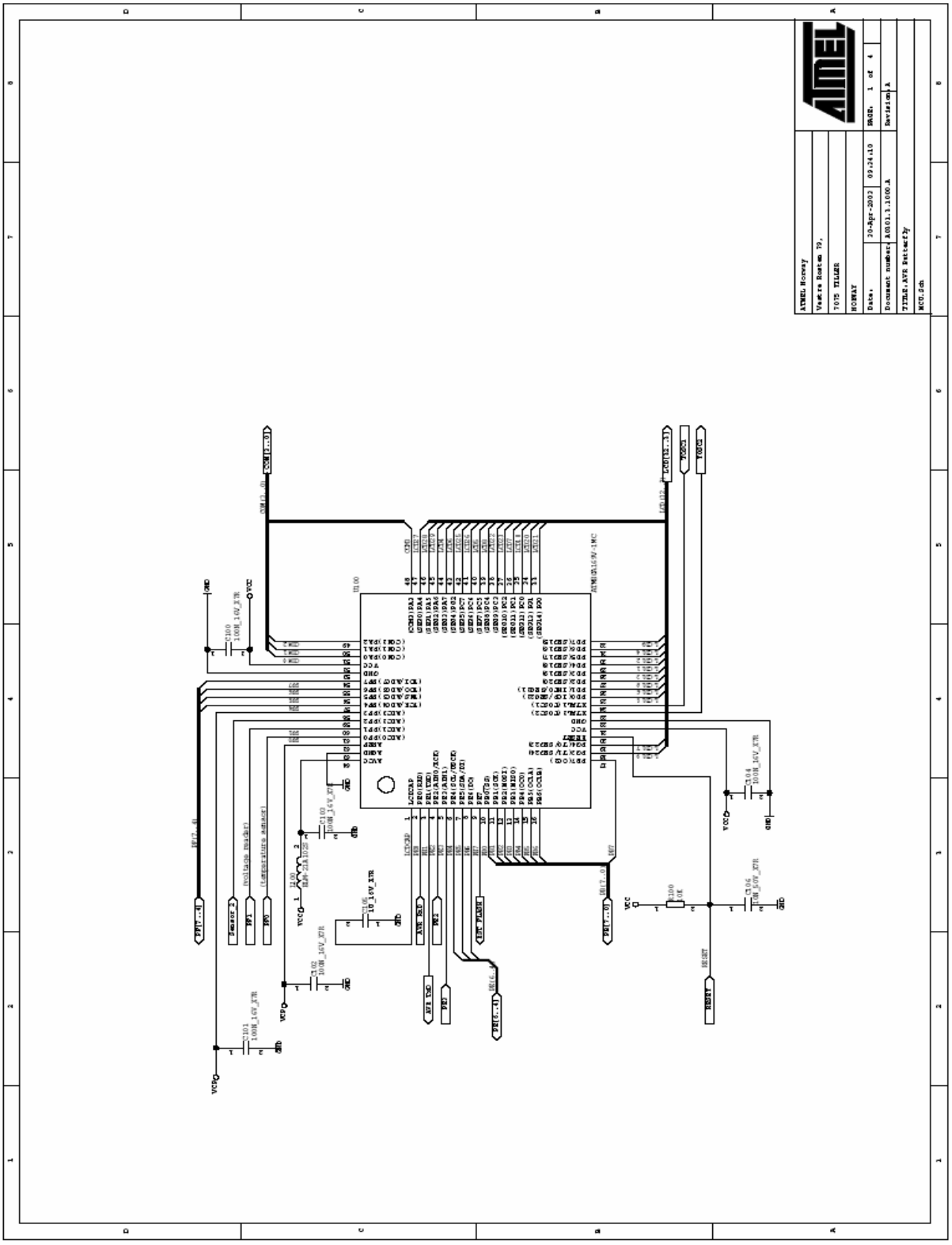


Figure 24: ATmega169 Schematic on AVR Butterly [2]

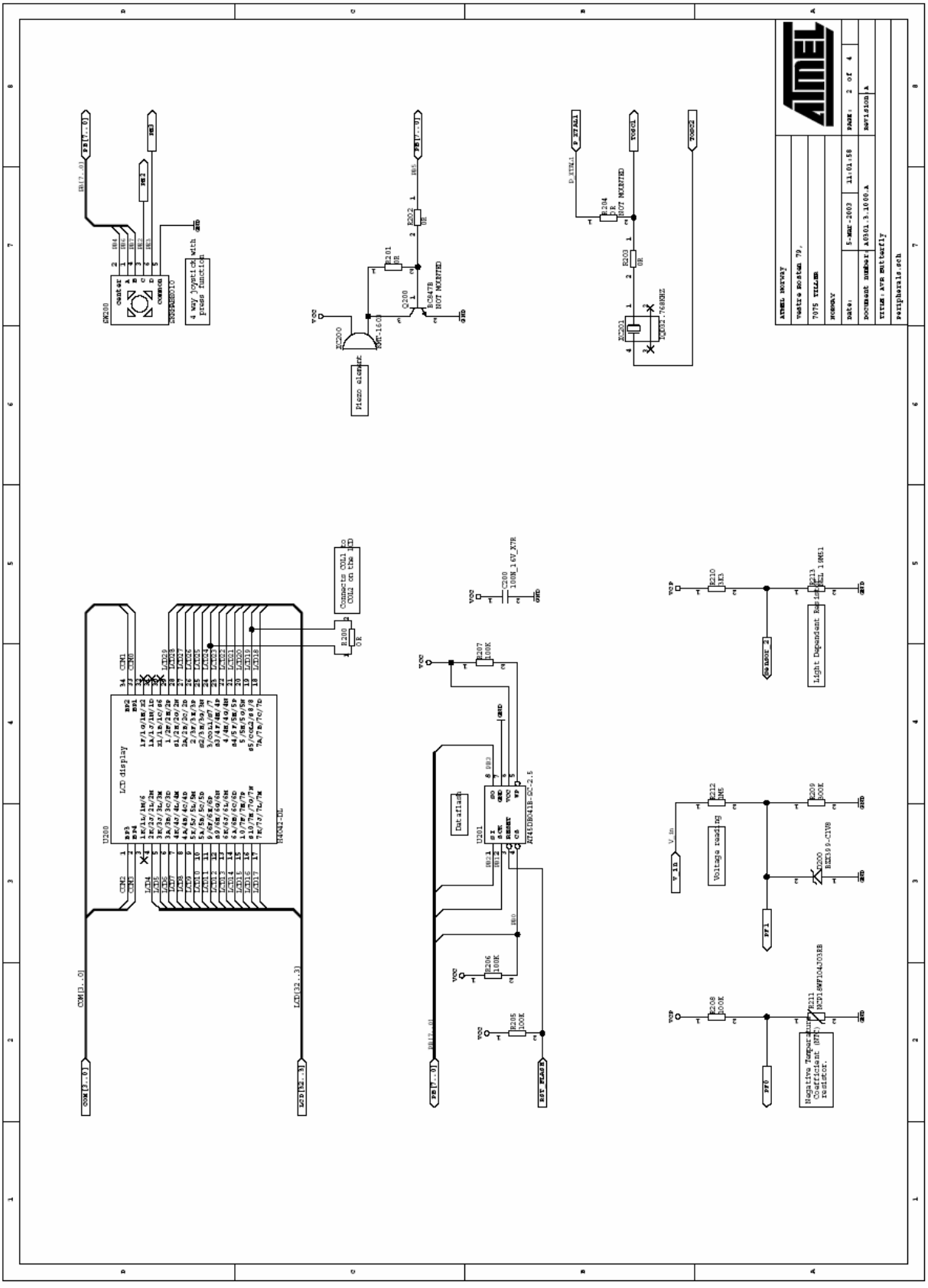


Figure 25: ATmega169 Peripherals on AVR Butterfly [2]

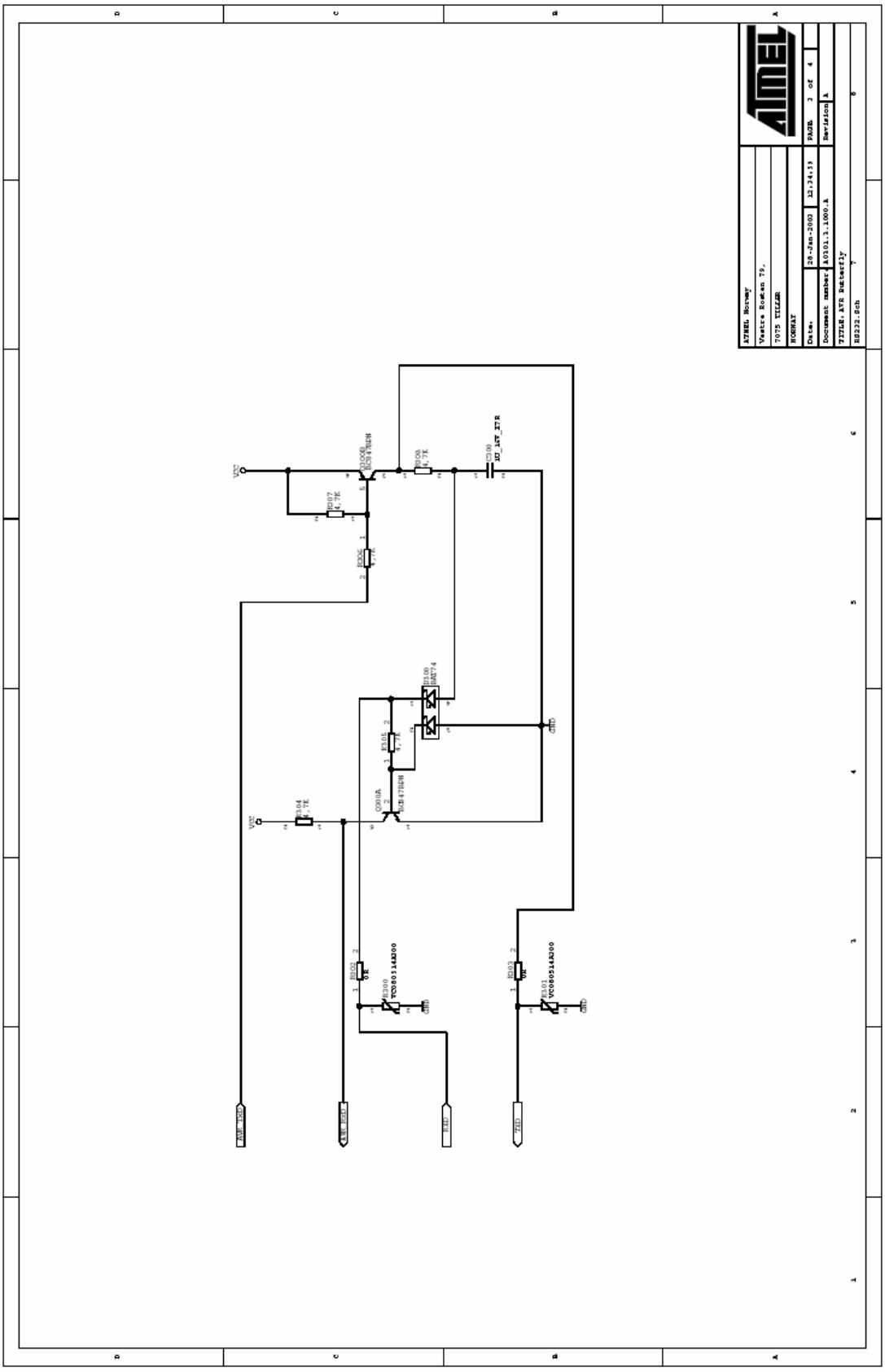


Figure 26: RS-232 Connection of AVR Butterfly [2]

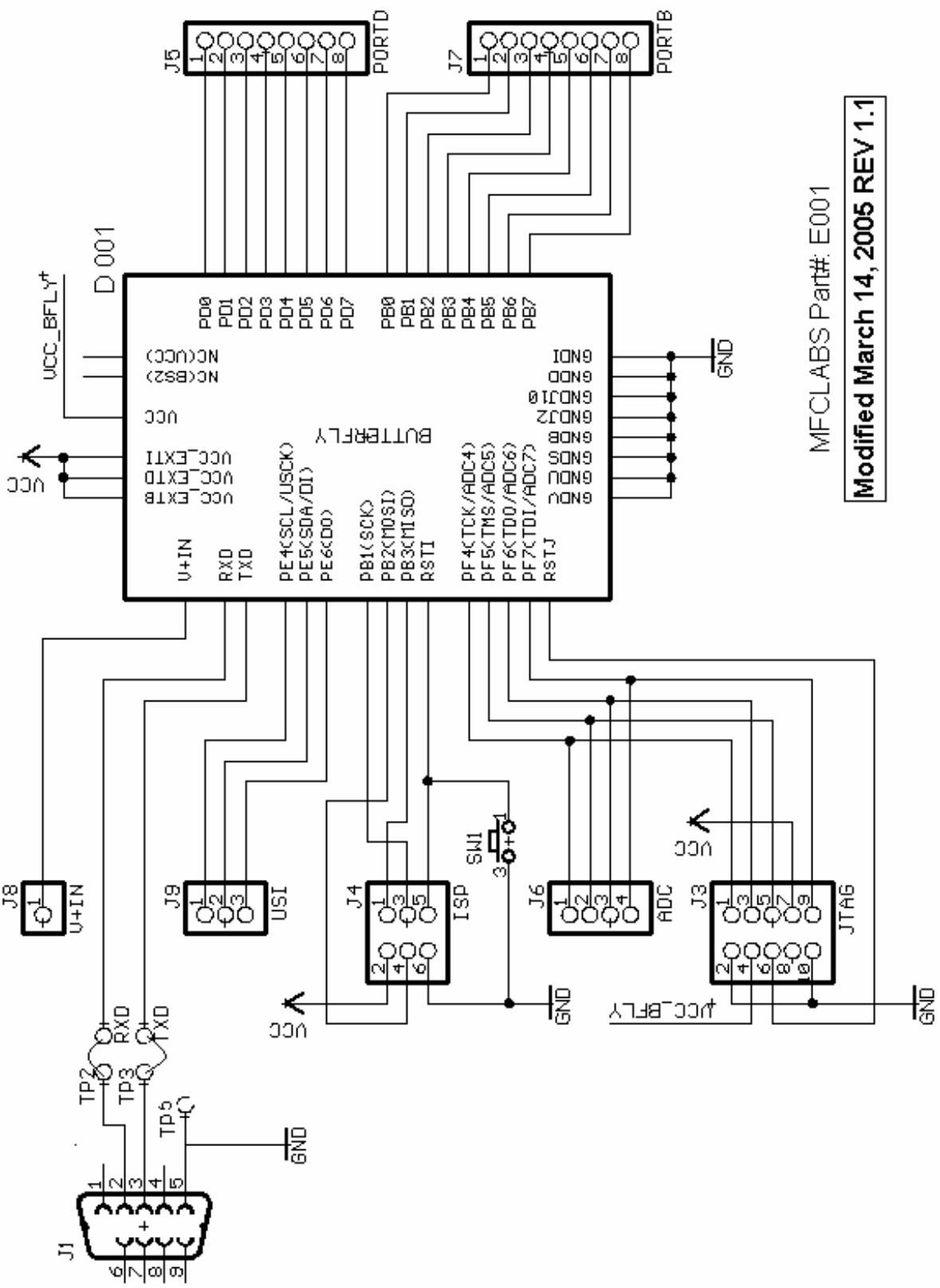


Figure 27: ATmega169 Main Connections Schematic

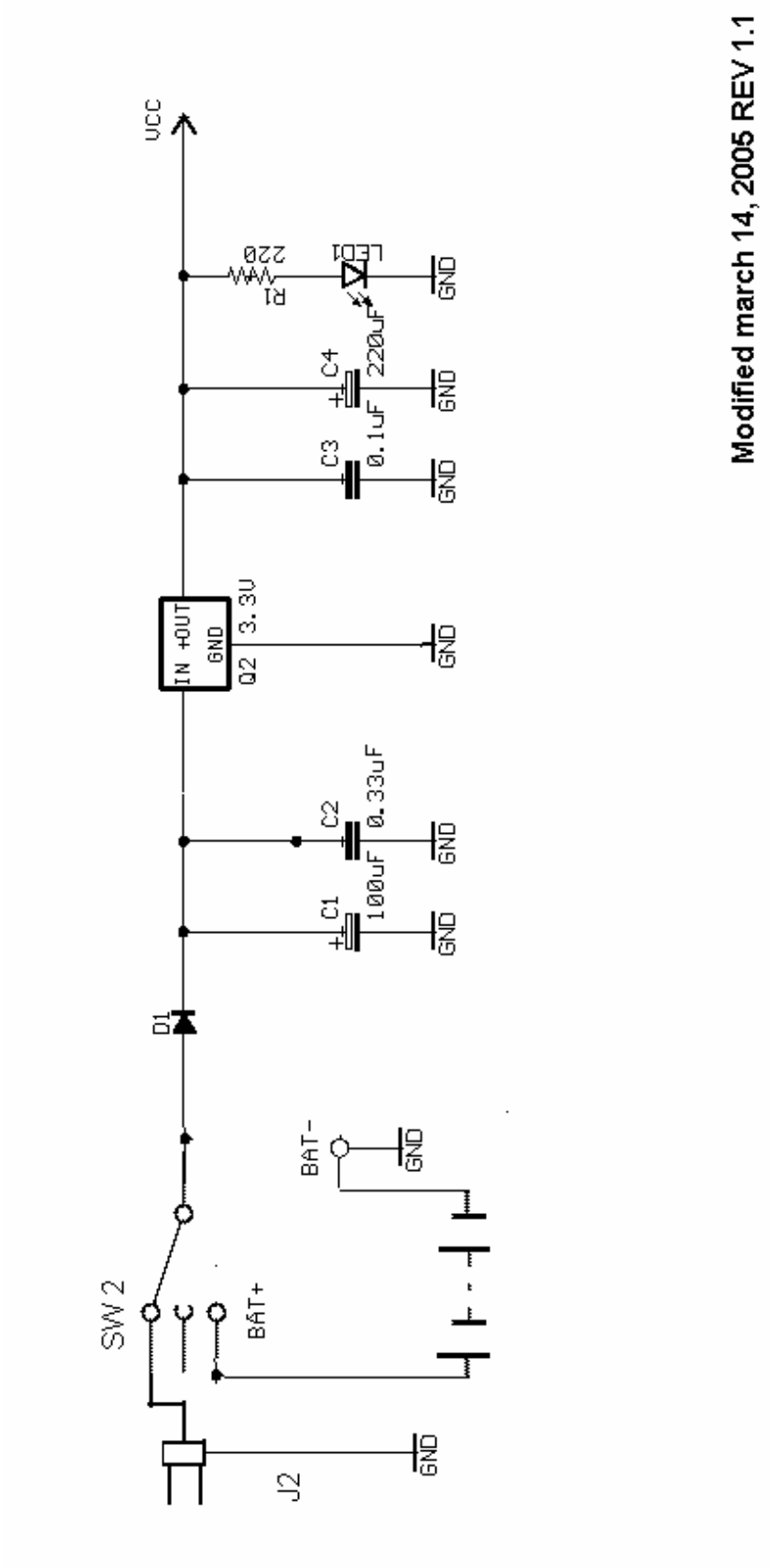
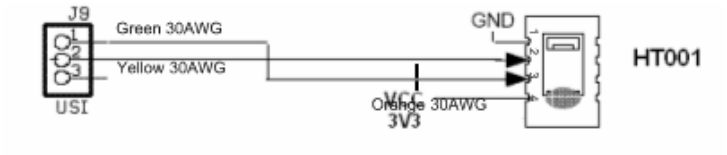


Figure 28: MFC Testbench Power Schematic



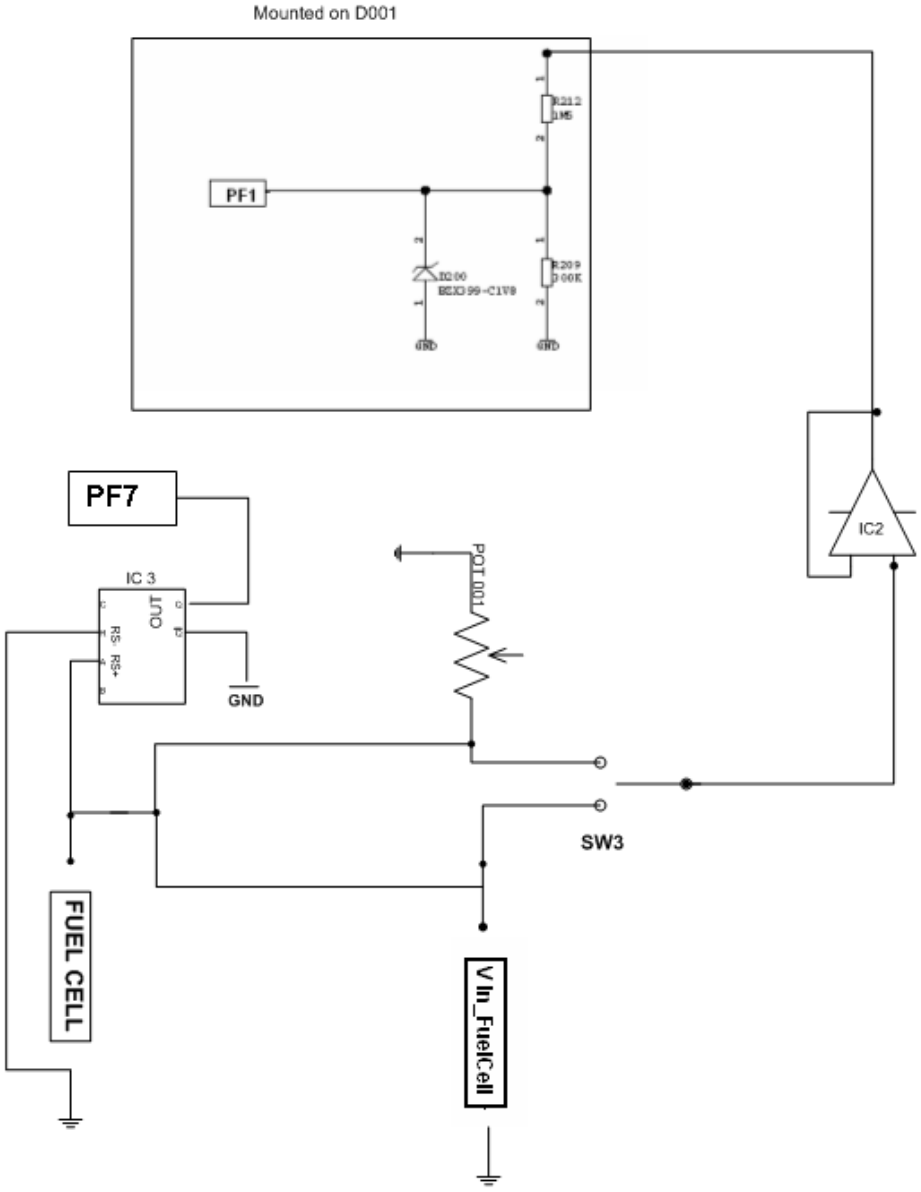


Figure 29: MFC Testbench DUT and Measurement Interface Circuit

Appendix II – Microcontroller Pin Assignments

Table 12: ATmega169 Pin Assignments

	Legend
	LCD Functions
	A/D Section
	Power/GND/RESET lines
	RS232 Communication
	Switch

Microcontroller Pins	Function	Comments
PA0	LCD (COM0)	
PA1	LCD (COM1)	
PA2	LCD (COM2)	
PA3	LCD (COM3)	
PA4	LCD27 (SEG0)	
PA5	LCD28 (SEG1)	
PA6	LCD29 (SEG2)	
PA7	LCD4 (SEG3)	
PB0	NC	
PB1	SCK (external flash clock)	Also connected to the SPI connector clock
PB2	SI (external flash input)	Also connected to the SPI connector input
PB3	SO (external flash output)	Also connected to the SPI connector output
PB4	Switch	
PB5	Piezo Element	
PB6	Switch	
PB7	Switch	
PC0	LCD10 (SEG12)	
PC1	LCD7 (SEG11)	
PC2	LCD23 (SEG10)	
PC3	LCD22 (SEG9)	
PC4	LCD8 (SEG8)	
PC5	LCD5 (SEG7)	
PC6	LCD26 (SEG6)	
PC7	LCD25 (SEG5)	
PD0	LCD15 (SEG22)	
PD1	LCD16 (SEG21)	
PD2	LCD18 (SEG20)	
PD3	LCD13 (SEG19)	
PD4	LCD11 (SEG18)	
PD5	LCD12 (SEG17)	
PD6	LCD14 (SEG16)	
PD7	LCD9 (SEG15)	
PE0	AVR_RXD	

PE1	AVR_TXD	
PE2	Switch	
PE3	Switch	
PE4	Temp/Humid Sensor Clock line	
PE5	Temp/Humid Data line	
PE6	NC	
PE7	RST_FLASH (external flash reset)	Also connected to the SPI connector reset line
PF0	Temperature Sensor	
PF1	Voltage Sensor	
PF2	Light Sensor	
PF3	VCP	
PF4	NC	
PF5	NC	
PF6	NC	
PF7	NC	
PG0	LCD21 (SEG14)	
PG1	LCD21 (SEG13)	
PG2	LCD21 (SEG4)	
PG3	LCD19 (SEG24)	
PG4	LCD17 (SEG23)	
RESETn	External Reset	
XTAL1	External Oscillator	
XTAL2	External Oscillator	
AVCC	VCC	
LCDCAP	External Capacitor	
AREF	VCP	
AGND	GND	
VCC	VCC	
GND	GND	