Tufts University
Electrical and Computer Engineering Department

Senior Design Project: NerdGirls
Instrumentation Panel Group

Final Report

Gladys Magh and Megan Schwartz

May 12, 2003

TABLE OF CONTENTS

PROJECT REQUIREMENTS

The formal project requirement for the Instrumentation Group was to develop a design, build, and test an instrumentation panel for a solar vehicle. As solar cars are run off of solar energy as opposed to fossil fuels, the driver of a solar vehicle has much different needs than the driver of a gas-powered car. Instead of an engine, a solar car collects energy from the solar cells positioned on the vehicle, stores it in batteries, and sends the electricity to motors. The need for an instrumentation panel arises in the fact that instead of a gas gauge, there needs to be other displays letting the driver know different values that are relevant for a safe drive. The parameters mostly involve the batteries and the power they contain, as well as how quickly the batteries are using the energy stored in them.

Certain overall parameters for the vehicle have been defined. A solar array is connected to two banks of batteries, each consisting of four, 12V batteries. The batteries connect to two separate motors (one per rear wheel) to power the vehicle.

The project requirements included designing, building and testing a prototype of the instrumentation panel, which would display the following parameters:

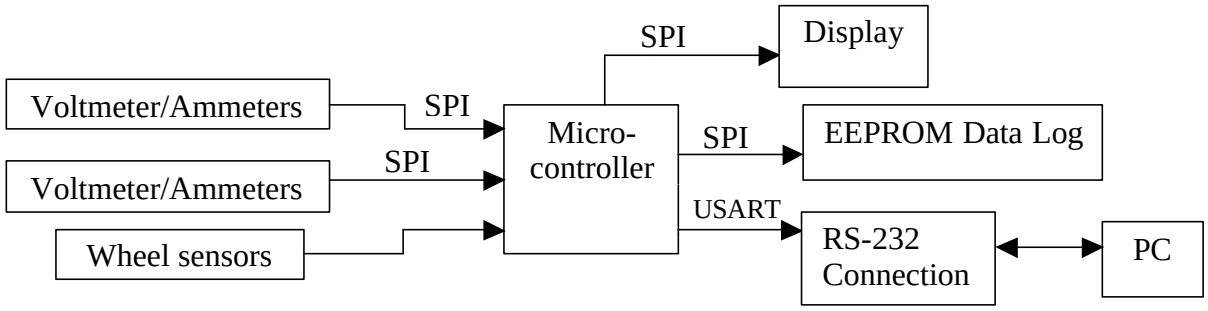
- Voltage stored in each battery bank
- Amperes stored in each battery bank
- Amp-hours of each battery bank
- Speed of the Vehicle (Speedometer)
- Total distance traveled of the vehicle (Odometer)

The instrument panel design will contain an EEPROM, which should be used to store the device information periodically, as a data log. An RS-232 connection should be included to allow Nerdgirls members to pull the data log from the EEPROM for data analysis. The main component is a micro-controller or FPGA, which will poll sensing devices at a regular interval.

DESIGN

The design of the instrumentation panel included the selection of the sensing devices, the role of the micro-controller in the display, and the coordination between the components through code.

The overall design was determined as seen below in a basic diagram:



The components selected to fulfill the roles of each block in the diagram that require explanation.

Voltmeter/Ammeter

There is 96 volts total that must be measured in order to appropriately inform the driver of the energy available as the vehicle is driven. There are two concerns with this aspect: first, any component used must be able to handle measuring such a large amount of voltage, and second, the component must be able to measure an increase as well as a decrease in voltage/amperes.

As it was investigated, a product used for electrically powered systems was discovered. Cruising System's Link 10 Meter measures not only voltage and amperes, but amp-hours, hours remaining before the batteries are fully drained. The product is designed to measure the characteristics of a solar array for a home, so it is fully capable of measuring an increase/decrease of power. As for the large number of batteries within the vehicle, the Link 10 Meter is able to measure up to 48V with prescaling modifications. As there are two banks of batteries, the design will allow for two meters to measure the voltage/amperes of the batteries.

Wheel Sensor

The design requirements state that a speedometer and an odometer must be incorporated into the system and displayed onto the instrument panel. There was an option to purchase a component that would evaluate both of these parameters, but instead, a custom component was developed. The custom part uses a sensor is put onto the wheel of the vehicle that will sense a revolution of the wheel. Using this, the micro-controller will perform the calculations required to evaluate the speed of the vehicle and the total distance traveled.

The selection of the sensor is notable. While there are many types of sensors available, the IC Hall Effect sensor was the most feasible in this application. The bi-directional sensor will pulse high when a magnet passes by, small distances away. The Schmitt trigger included on the part takes the analog value and digitizes it, which makes it possible to directly drive TTL and MOS IC. The motor synchronization group is using the same sensors for measuring the revolutions of the wheel, so it is sensible to use the same type of sensor.

LCD Display

The decision to build a custom speedometer/odometer meant that there needed to be a method of displaying these two values. An LCD display is necessary as a separate entity, which will alternate in displaying the speed and the distance traveled. The LCD is controlled exclusively through the micro-controller software. The only concern regarding the LCD is compatibility with the micro-controller.

Micro-controller

The micro-controller is necessary to comply with design requirements, as well as to unify all of the components of the instrumentation system. The micro-controller plays the role of coordination in this application, as it must be able to input and output to all of the external components, as well as evaluate the mathematics required to determine speed and distance. All three Nerdgirls groups chose to use a PIC micro-controller for its ease of use, wide availability, and wide range of features. The features needed for the instrumentation application are the large number of I/O pins, as well as the large amount of Flash memory available. As the design requires, a time must drive the entire system. The PIC micro-controller connects to an external

timer that is mounted on the demo board. The PICDEM demo board and development kit were also important for the development of the software.

Software Design

There are multiple parts to the software design, as the micro-controller plays many parts in the coordination of the instrumentation panel.

The first role that the software must include is a sensor input counter, which counts the number of sensor pulses that arrive into the PIC during a set time interval. The calculations module in the code then translates the number of revolutions per time period mathematically. Counting pulses requires the code to perform two functions: to measure a set time interval, and to increment a counter for every pulse on an input.

The next step for the code is the calculations for the speed and distance traveled. This is basic unit analysis, as revolutions per second are translated into miles per hour. This

Another function the code must perform is to write all of the data to the attached EEPROM, which must be able to be accessed by an RS-232 connection. The design here is to connect the EEPROM to the PIC, and use the on-board RS-232 connector for the PC data log downloads. This involves not only outputting the speed/distance values calculated, but also inputting the information from the Link 10 Meters and redirecting it to memory.

Hardware Design

The hardware design consists of different parts that work together in order to attain information from the E-meter and read into the micro-controller. A Maxim3110E chip communicates between the E-meter and the microchip. The Maxim3110E chip contains the RS232 and UART ports where it switches back and forth from TTL to RS232 levels. Then it is read into the micro-controller to be logged into the I2C-EEPROM where information from the E-meter is stored in there and read out from there if necessary.

Bill of Materials

Product	Manufacturer	Part #	Vendor	Amt
1. Link 10 Electric-Meter	Crusing Equipment Co.	Link 10 E-meter	http://www.homepower.com/files/hp52-30.pdf	2
1a. Prescalars	Maxim Dallas Semiconductor	Prescalers	Maxim-ic.com	4
1b. Twisted Wire for connection	Maxim Dallas Semiconductor		Maxim-ic.com	10 feet long
2. Sensors: Meet w/ Motor Group	(see motor group)			
3. Microchip's PIC18F452	Microchip Co.	PIC18F452	http://www.microchip.com/1010/pline/picmicro/category/embctrl/14kbytes/devices/18f452/index.htm	1
4. LCD Display	Matrix Orbital	LCD2041	http://www.matrixorbital.com/products/lcd2041.htm	1
5. EEPROM	SGS-Thomson Microelectronics	EEPROM 64K M2764A-2FI	http://www.uib.es/depart/dfs/GTE/staff//jfont/InstrETT/M2764a.pdf	1
6. Max3110E/Max3111E device	Maxim Dallas Semiconductor	Max3110E	http://www.maxim-ic.com/quick_view2.cfm/qv_pk/2052	4

Strengths and Weaknesses in Design

There are both strengths and weaknesses in the design. The first strength is that by utilizing more software, it allows for more leeway in the design. The coordination of more hardware seemed to be a daunting task, especially when software can manipulate inputs and outputs using a micro-controller so easily. An added strength is the luck of finding a component, the Link 10 Meter, whose function is the same application (electric systems) and taking advantage of the find. Using this component took out a lot of the hardware work necessary to evaluate the current state of the battery banks in the vehicle.

The main weakness in order for the design to work is the compatibility of the separate components, and the ability to pull together parts to work as one. This challenge has been met by doing extensive research before ordering specific parts, as well as an investigation into communication protocols that are available for use between components. These include I2C and SPI, as well as UART and RS-232 connections between components on the hardware level. The second weakness in the design is the self-imposed design requirement to build a custom speedometer. The accuracy of a custom part may not have the accuracy of an off-the-shelf part, especially at high speeds where the wheel is revolving very quickly.

Component	Use
2 Link 10 Electric Meters	Read in Volts and Amps
2 Max3110E/Max3111E chips	Connect to 2 E-meters
2 Max3110E/Max3111E chips	Connect to 2 external sensors
64K I2C-EEPROM	Log data from E-meter
LCD display	Displays speed, volts, amps
2 Tachometer Sensors	Measures speed

IMPLEMENTATION

The implementation follows the design in the software and hardware. Details of both implementations are discussed below.

Software Implementation

Implementing the software for this design project consisted of many components. The design shows that the system has a timed polling of sensors, which will then display values and send the data to an EEPROM for data logging. The different sections of the code are described below, and can be found in Appendix A1-A4. There are four files required to run the code; the program was modeled using Microchip's PICDEM2 Plus Demo Code, written in their own MPASM assembly code. The files are named the following:

- instrument.asm: This is the file containing the main code to run the program. This program calls the functions in the rest of the files.
- p18LCD.asm: This file contains the low-level functions necessary to initialize the LCD and write characters to the display.
- p18MATH.asm: The multiply and divide functions are in this file.
- p2plsp18.lkr: This is the linker file necessary to connect the files in the compiler.

instrument.asm

The instrument.asm file has all of the custom components developed by the group. As there are line-by-line comments in the code, the following is an explanation of each basic section of the code, with how it was implemented and the level of functionality.

The first section of the code is setup for the actual program. The initialization shows that the program is intended to run on a PIC 18F452, and the associated configuration bits are set to adjust the oscillator, the watchdog timer, and code protection. As all of the functions called in the main program are not included in the instrument.asm file, the EXTERN assembler directives tell the compiler that the following functions are located in external files. There is a block of variables declared using the RES directive. The directive reserves the specified number of addresses in memory for the variables to be stored. Variables here include those needed for calculations, timing, and binary-to-BCD conversion.

The beginning of the code is indicated by the org statement, showing that the start of the main program should begin at the very first address in memory. The interrupt service routine, isr_tach, is located in memory that is reserved for high-priority interrupt vectors. The last data initialization is declared in a stan_table to be accessed by the stan_char functions, which send data to the LCD. The data declared here is constant, so only output information that is static may be entered in the table. Each data entry in the table is sixteen bits wide, as that is the maximum width of the LCD display.

The start of the program begins with the LCD initialization, calling the LCDInit function in the p18LCD.asm file. After the LCD is prepared to output information, the program initializes the USART for receiving data by enabling the serial port as well as the continuous receive enable bit. The TXSTA, SPBRG, and RCSTA registers are assigned the appropriate bit values as specified in the commented code. This is included for testing, as further along, the USART will be used to send a data log from the EEPROM to the connected PC laptop. Timer2 is also initialized, as the main timing in the main routine. The initialization occurs by setting the appropriate bits in the T2CON control register.

Before the rest of the code is discussed, a description of how the timing and polling execute is necessary. The design calls for a loop that will count ten seconds, and during those ten seconds, poll the PORTB external pins to count the number of pulses from the wheel sensors. The number of revolutions per ten seconds is used to calculate RPM, MPH, and distance traveled. Setting a timer for such a long period as a second is not feasible, so a 1 second timer is used ten times. During that ten second period, polling the ports is implemented through an interrupt service routine that is triggered every time a change on the pins of PORTB is detected. Thus, before the timing begins, the interrupt control registers must be set with values to configure them properly.

Also necessary to explain is the 1-second timing loop. The loop's pseudo-code steps are as follows:

- Set timer for one second with the timing control registers
- Begin loop
- Continually test timer:
    - if expired, exit loop and move to next 1 second timer
    - else, stay in loop and test again
- [Run this 1 second timer and loop ten times]

The code continues at the MainLoop label, which exists purely for testing. The next section of code is to test sending output to the LCD through the table, using the stan_char_1 routine and stan_char_2 routine in the file. Two lines outputting "Nerdgirls" on the first line and "JUMBOS 2003" on the second line are displayed. Three seconds of a delay follow this.

The setup for enabling the interrupt is next. The registers that require setup include the RCON register (Reset Control), the INTCON register, (Interrupt Control), and INTCON2. As seen in the attached code, RCON bits are set to enable priority interrupts. The INTCON register enables high priority, and disables low priority interrupts. It also enables the PORTB change interrupt, and disables all other special interrupts. INTCON2 sets the priority of the PORTB change interrupt to high.

The timer has yet to be implemented. The intent was to use Timer0 and set it to a prescale value that would use the 4Mhz clock running to create a 1 second timer. Once the timer expires, the TMR0 overflow interrupt flag will set. This bit, RBIF in the INTCON register, is the bit that the expiration loop tests.

After the 10 seconds has passed, the program disables interrupts by clearing the main interrupt bit in the RCON register, and then begins the calculations for RPM. The rev_count register contains a count of revolutions per ten seconds, and multiplying by a decimal value of six will result in a 16 bit RPM value. The routine call is to the UM0808L function located in the p18MATH.asm file.

*There is one assumption made: the diameter of the vehicle wheel. We have chosen to assign the diameter to 1 foot, so that the radius would be .5 feet.
In order to get from RPM to MPH, the following equation shows the mathematics necessary:

rev/min * 60 min/hour * (2p*.5)/rev * 1 mile/5280 feet = miles / hour

This can be implemented currently only partway. The 16bit x 16bit multiply necessary to calculate feet/hour exists in the routine call mult_16_16 within the instrument.asm file. While there is a 16 bit x 8 bit divide available in the p18MATH.asm file, there is no math function implemented that can divide a 32 bit value. . The isr_tach routine is the only remaining custom code for this implementation. The following areas of the code were not addressed in the software currently:

- calculation of distance traveled
- display of speed and distance on LCD
- read input of the two Link10 Meters
- write both the Link10 information as well as the calculated values to the EEPROM
- clear the rev_count value, begin again at timing module.

The list of routines called are now listed in the code. All of these were pulled from the p18demo.asm original file, or the 18F452 Datasheet. They include the stan_char_1 and stan_char_2, which display a line of data to the LCD on the first or second line, respectively. Delay_1s executes enough cycles to delay for one second, and mult_16_16 executes a 16 bit by 16 bit multiply. Bin_bcd and bin16_bcd translate binary to BCD for 8 bits or 16 bits. The last routine is the isr_tach interrupt service routine.

The isr_tach function performs very little, but is essential to the timing/polling capabilities of the program to evaluate speed and distance. Every time any of the four PORTB pins change value, the isr_tach is triggered. The isr_tach routine reads PORTB and clears the

interrupt flag in the INTCON register to disable the interrupt. The value of rev_count is incremented, and the LCD displays two lines of information: “INTERRUPTED!” and “isr_tach is running.” A three second delay is implemented, and then the ISR returns to the main program by popping the stack and returning the PC to the correct address in code.

p18LCD.asm

The routines included in this file are all called by the main program, or they call each other. The most important function to understand is the LCDInit routine. This function begins setting up the pins for output to the LCD. PORTA pins 1:3 are used as control pins for output to the LCD. The upper four bits of the PORTD register are used to send data to the display. The A/D conversion control register ADCON1 is set to allow all bits on PORTA to be digital, except for AN0, which is an analog value. This exists only because AN0 is hardwired to the potentiometer on the demo board.

The rest of the subroutine is a series of delay loops and commands sent directly to the microcontroller embedded in the LCD display. The commands are the “control sequences” that are numbered in the comments. The LCD display requires three unlock commands, then the bus width is set to 16 bits, and several other commands that turn the display on and clear the screen. All of these steps are necessary to explain to the LCD that it will be outputting information.

The remaining routines inside the p18LCD.asm file are all very low-level functions that the LCDInit file calls for initialization, or that the main program is able to use the d_write to send variable values to the LCD display.

p18MATH.asm

The math file contains two main routines that are used to calculate multiplication and division. The first routine, UMUL0808L, calculates a 8 bit by 8 bit unsigned multiply. The second routine, UDIV1608L, performs a 16 bit by 8 bit unsigned divide. Both are used in the calculation section of the main program.

Hardware Implementation

Link 10 Electric Meters

Many different types of components are needed to implement the hardware for the instrument panel which include the 2 Link 10 Electric Meters, 4 Max3110E chips that contain the UART and RS232 terminals, a EEPROM, LCD Display, and 2 tachometer sensors.

The two Link10 Electric Meter was chosen because of the strict requirements from the Tour de Sol race competition and thus used in the hardware. Each E-meter reads from the 2 48-battery banks the volts and amps used from the solar car. Diagram 1 shows the schematics for connecting the E-meter terminal strip to a power supply. For the laboratory testing purposes, a 12-volt power supply was connected to the E-meter which is a lesser voltage supply than a regular car battery. The E-meter will display 50 mAmps and 12 volts for testing. The light bulb is for testing purposes to see if current was going through the circuit. The serial port on the E-meter transfers data between the E-meter, through the Max3110E chips, and to the microchip. From Diagram 4, the pins from the serial port show where it should go to the Max3110E chip.

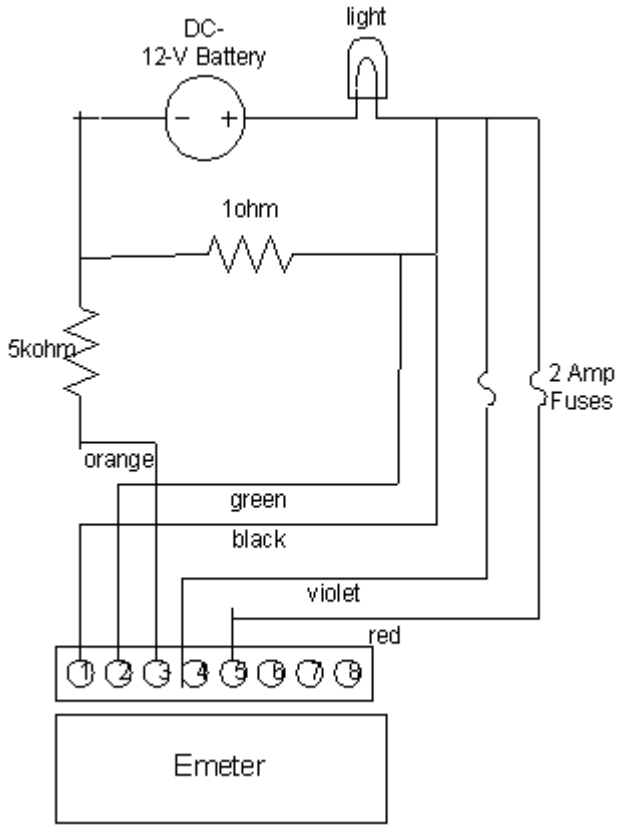


Diagram 1- E-meter Schematics

Max3110E/Max3111E chip

The Max3110E/Max3111E chip contains the UART terminal which addresses the microchip and the RS232 terminal that addresses to the E-meter. The Functional Diagram of the Max3110E chip (see Diagram 3) shows the actual TTL and RS232 levels that are needed in order to translate from incoming data and convert them into either RS232 level outputs or UART level outputs. Diagram 2 shows the actual pins from the chip and what each pin represents. Within the chip, however, there specific requirements when connecting pins from a RS232 level to a UART level. This is necessary because of the sensitivity of data that is being transferred between the microchip and the E-meter.

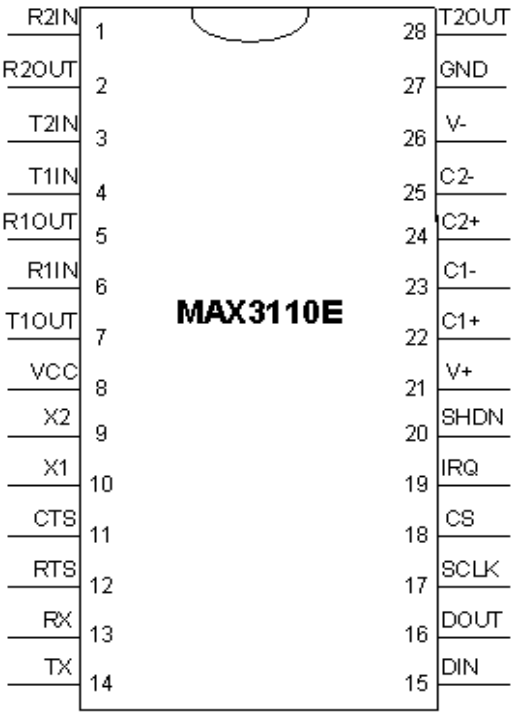


Diagram 2 – Maxim3110E chip

Diagram 3 represents the internals of the Max3110E chip. Each output or input is connected to a certain pin number or else it is outputted to the microchip such as in the case of the lower right hand side of the diagram. IRQ, DOUT, SCLK, CS, and DIN are all in TTL level that are connected to the Microchip 18F452. The upper right hand corner of Diagram 3 represents the pins to the DB9 connector on the E-meter.

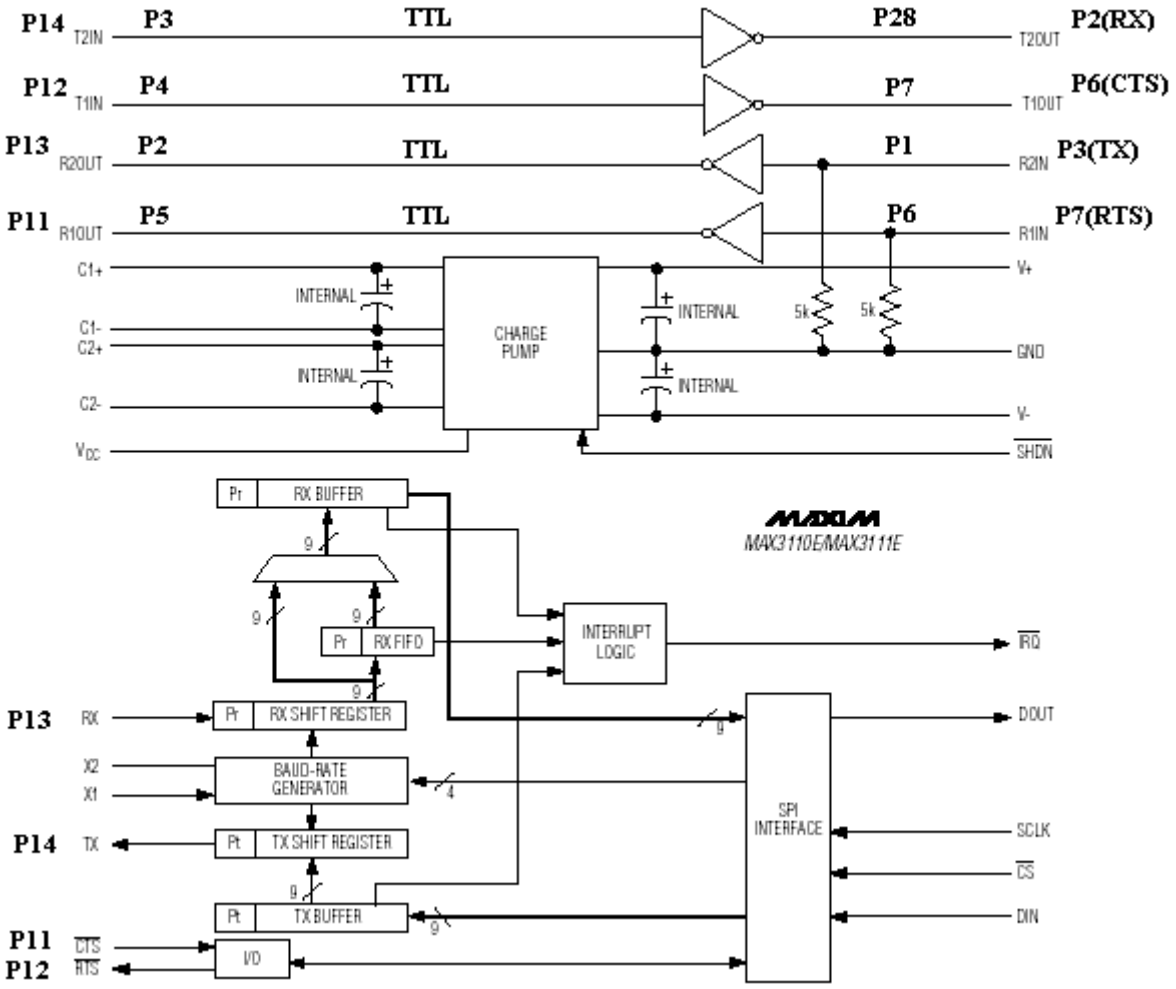


Diagram 3 – Functional Diagram of Max3110E

Microchip 18F452

Microchip 18F452 is used to receive data from the E-meters, process that information, then output the data into an EEPROM. Pins 2 through 5 are connected to the tachometer sensors for reading. Pin18 and Pin23 are needed to download information into the EEPROM. Pin26 receives information from the Max3110E chips and Pin25 transfers out information to the Max3110E chip and eventually the E-meters. Data is transferred between the Port B and Port D pins. Port B pins include Pin 33-40 which are 8 bits wide. Port D pins include Pin 19-22, 27-30 which are also 8 bits wide.

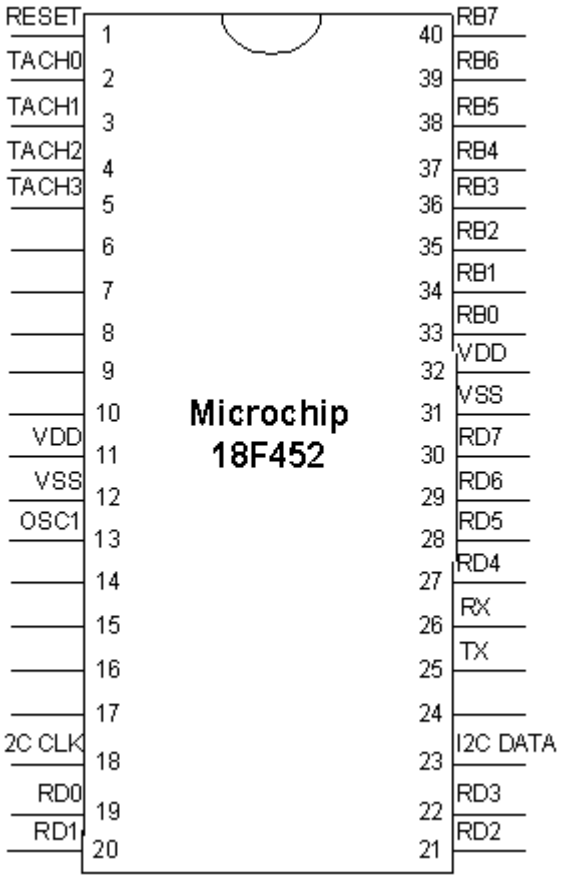


Diagram 4 – Microchip 18F452

Diagram 5 represents the actual pin connection between the Microchip 18F452, Max3110E chip, and the DB9 connection on the E-meter. When wiring the components together, it is necessary to remember that when one chip is receiving, the connection on the other end is transferring. So therefore, careful attention must be made when wiring pins together because it is not always the case where the pin with the same name should be connected together but should be the complement of that pin. For example, a RX pin on one chip should be connected to a TX pin on the other chip.

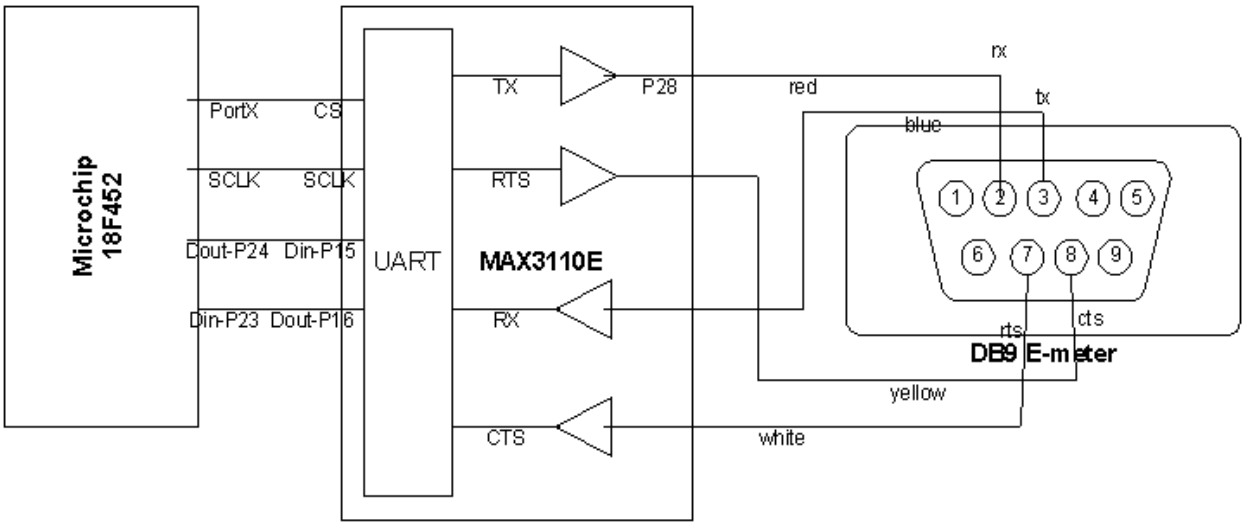


Diagram 5 – connection between microchip, max3110E, DB9 of E-meter

ASSESSMENT

Software

Due to time constraints on the project, much of the software that was implemented does not function as expected. The LCD correctly displays the static data output, and the three second delay functions correctly. The interrupt control registers are initialized properly and the correct bits are set.

The interrupt service routine does not function as expected. The output during a program run goes as follows: The "Nerdgirls/ JUMBOS 2003" message displays for three seconds, then immediately the "INTERRUPTED/isr_tach is running" message for the interrupt displays for an indefinite period. The program stops once there is a change of the pins on PORTB. Thus, the program recognizes the interrupt, just not at the correct time.

The non-functional parts of the code that was implemented include the one second timer and the calculations. The timer code was designed but not successful due to the inability to set the register value that would result in a 1 second timer. The calculations failed due to not being able to divide a 32 bit value.

Overall, the design of the software is complete in regards to polling, timing, and interrupts. The calculations simply need a routine to handle larger-bit numbers. As the designs for the intended display of calculations, reading from the Link 10 Meter, and the writing to the EEPROM are not implemented, the software design fails overall to obtain the requested objectives as stated in the design requirements. If substantial work went into the software, the program as designed should execute and display the information as requested.

Hardware

As the implementation proceeded, and due to time constraints on the project, the EEPROM that was assumed compatible with the microchip was deemed unusable due to the application used in the software implementation. Also, an EEPROM with SPI was needed due to the UARTS being SPI as well. There was no available SPI EEPROM and therefore not implemented in hardware.

The Microchip was discovered to have only one set of pins for data in and data out transfer. Since there were 2 Max3110E chips used, a second set of pins for data in and data out are needed to transfer data. Not enough careful assessment was made in determining the number of pins needed for data transfer was made.

CONCLUSION

In hindsight, it was not realized how software-heavy the design was until it was too late. As the hardware experienced much less issue than the software, much more effort should have gone into software. Learning the assembly code used in the program and careful preparation before implementation would get future groups onto a solid footing before coding. Selecting and purchasing components should have happened much earlier, because the PIC micro-controller arrived only a month before completion of the project. Much more time was needed to develop the more complex functions necessary to the program, such as the timing and the interrupt.

A more closer look into the exact amount of pins from the PIC was needed in order to connect the second E-meter to the PIC. The EEPROM also needed more research to determine what was the correct way to implement it with the PIC.

FUTURE WORK

Software

The software requires much more development before being fully functional, but the design has the potential to be successful. Future plans should include developing the interrupt code so that for every PORTB pin change, the interrupt actually runs. Timing needs to be developed, as well as the correct routines necessary to complete the MPH calculation.

Displaying the speed and distance traveled can be modeled after the variable output as seen in the original p18demo.asm file in the Voltmeter section. There, each character of the output is sent to the LCD independently, after being translated from binary into BCD. This means that each BCD value needs to be separated into each decimal place, and then written to the LCD.

Reading the Link 10 Meter information may be more difficult. The issue of how much space each read will take in the EEPROM is just an estimate, and the compatibility of the hardware with the Meter may become more complex. Writing to the EEPROM and the LCD through the same medium is a potential point of concern, but using the chip selects on the UART/RS-232 chips should remove all possibility for bus contention.

Hardware

Decisions that include buying hardware with enough pins for memory and data transfer are necessary when there are numerous sensors and meters needed to read from. Since there is only one set of Data In and Data Out on the Microchip, for 2 E-meters, another Microchip is needed to accommodate the second E-meter reading.

A PCI-EEPROM was necessary to transfer data between E-meter and the PIC. If the amount of time for this project was extended, a better implementation in the software would provide a use for the PCI-EEPROM.

REFERENCES

Microchip PIC 18FXX2 Datasheet. Microchip Technology, Inc., 2002.

MPASM User's Guide with MPLINK and MPLIB. Microchip Technology Inc., 1999.

Panasonic DN6849/SE/S Hall ICs Datasheet. Panasonic Corporation, 2001.

Peatman, John B. *Design with PIC Microcontrollers.* Prentice Hall, Upper Saddle River, New Jersey: 1997.

ACKNOWLEDGEMENTS

- Matthew Heller, Principal Consultant
- Mike Quaglia, Mechanical Consultant
- Rick Colombo, Electrical and Schematic Consultant
- Dr. Karen Panetta, Head Advisor
- Dr. Steven Morrison, Advising
- Larisa Schelkin, Managing
- Warren Gagosian, Electrial Expert

APPENDIX CONTENTS

Appendix A1	instrument.asm
Appendix A2	P18LCD.asm
Appendix A3	P18MATH.asm
Appendix A4	p2plsp18.lkr

APPENDIX A1: instrument.asm

APPENDIX A2: p18LCD.asm

APPENDIX A3: p18MATH.asm

APPENDIX A4: p2plsp18.lkr