

## **BUILD A TI-59 PROGRAMMABLE CALCULATOR**

The TI-59 Programmable calculator from Texas Instruments along with the HP-65 from Hewlett Packard was considered at one time to be the equivalent of a pocket PC in power and functionality during the late 70's and early 80's. The feature that made this calculator so popular was the Algebraic Operating System (AOS) that allowed the user to enter mathematical expressions including parentheses. Other features were the programmability that included being able to store up to 999 steps and 100 registers. Instructions included subroutine calls, conditional and flag tests, and looping capabilities. For the article I will submit a design for a DIY TI-59 Programmable Calculator that includes two keypads, pushbuttons, an LED display and an LCD display. In addition I will provide the firmware for the dsPIC30F6014 to emulate a subset of the TI-59 functions including the AOS system and some programming capabilities. The reader, using my article entry as a starting point could complete the remaining TI-59 functions and features. In addition it opens up the possibility of emulating modern TI and HP graphing calculators.

User input and numeric key sequences are entered by means of the keypad or by initializing the Program Memory with an AOS expression or program, or through the serial port when it's configured for debugging. The TI-59 Programmable Calculator processes the selected operations by feeding data to the embedded dsPIC30F6014 micro-controller for evaluation, taking advantage of the double precision floating point math library for processing 64-Bit floating point operations, including arithmetic, trigonometric, scientific, and statistical and conversion functions. The answers are then converted from the internal IEEE floating point formats to ASCII values and are sent to the desired output device which includes the LED display connected via the SPI interface, an external serial LCD display, or the serial port connected to a host PC or Laptop. The two keypads are scanned and the keystrokes are de-bounced with a simple algorithm using the weak pull-up resistors. In addition, push-button switches are used to provide the missing keys for the complete TI-59 keypad matrix.

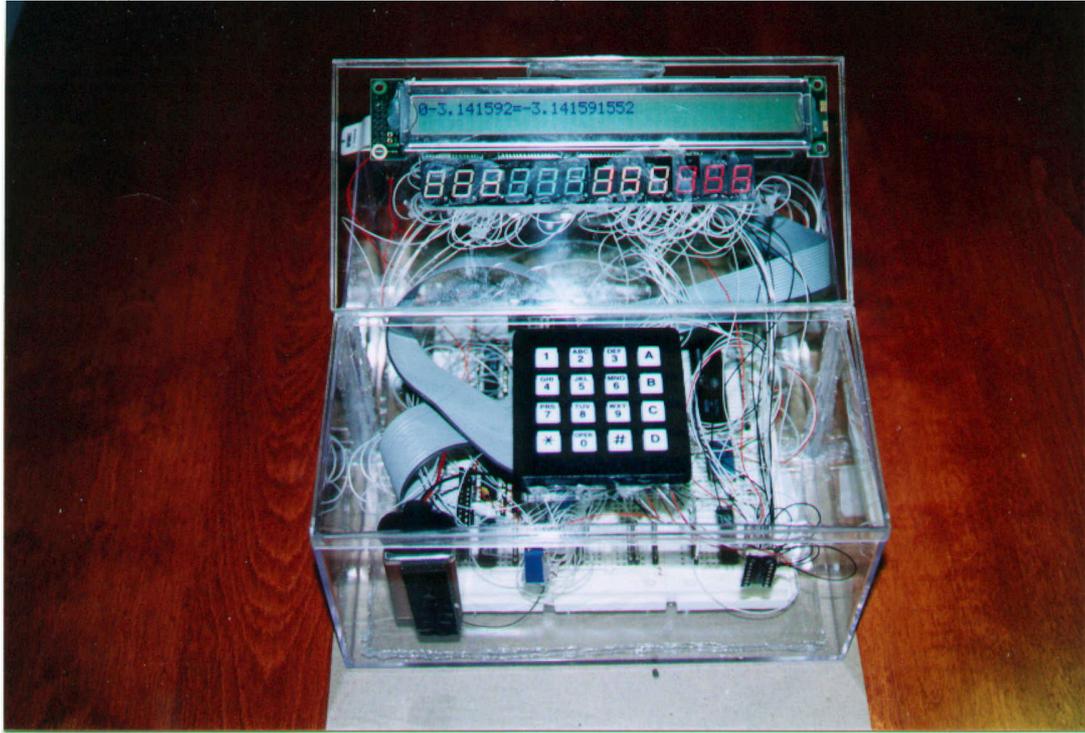
The UART1 peripheral is used for serial I/O to debug the application by sending/receiving serial output to a Windows Hyperterminal on a PC or laptop setup for 115200 baud, 8 data bits and 1 stop bit. In addition the SPI1 peripheral is used to send data to the LED Display module consisting of two MAX7219 LED Driver IC(s). The SCI peripheral in turn is used to send data to the on-board LCD Display.

This is a tale of two calculators, one made by Hewlett Packard, and the other made by Texas Instruments. The project that I selected for the Microchip dsPIC Article is a combination of two innovative Programmable Calculators, the HP-65 and the TI-59 shown in Photo 1 that paved the way to modern PDA's such as the Pocket PC and the PALM. In a sense they are also the progenitors to PCs and laptops and as such they can be used as simpler models to learn from. I Selected the TI-59 as the model for this article because of its immense historical significance in the advancement of personal calculators and computers. I also have a design for the HP-65 Programmable Calculator using RPN notation which could be the subject of another article. Comparisons between these two classic calculators have been the subject of much debate over the years as to which system was better for evaluating expressions, so comparisons between the TI-59 and the HP-65 will be apparent, although I loved both these calculators equally so I will stay out of this debate.

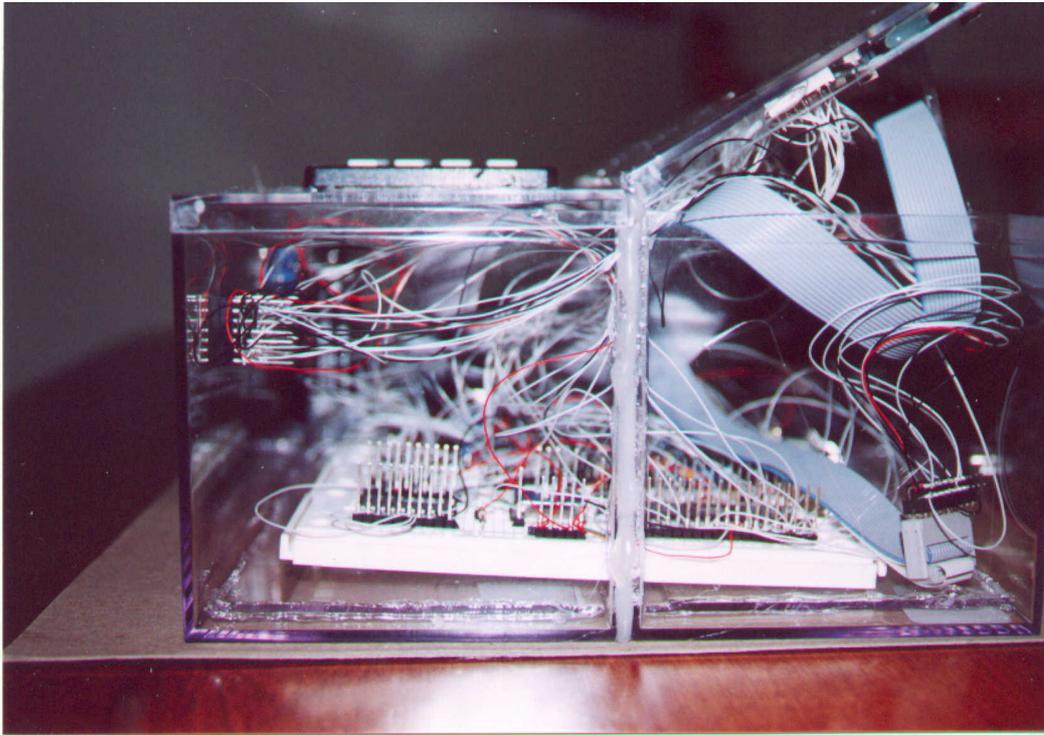


Photo 1 - Shows the original TI-59 Calculator.

Photo 2a – Is a photo of a prototype HP-65 calculator with the colorful LED display that I recycled for use with the TI-59 Programmable Calculator.



**Photo 2b– You can see the wire-wrap construction very clearly using the clear plastic enclosure for this calculator.**



Modern day hand held calculators were the brainchild of Jack St. Clair Kilby (1923- ), an American scientist from TI, who needed a vehicle to demonstrate his other invention, the Integrated Circuit (IC) which he co-invented with Bob Noyce and Gordon Moore of Fairchild Semiconductor. The invention of the Integrated Circuit is claimed as “the invention that changed the world”. These two inventions were also credited with starting the modern information age of the Personal Computer (PC). More information on Jack is available from the TI WEB site: [www.ti.com/corp/docs/kilbyctr/jackbuilt.shtml](http://www.ti.com/corp/docs/kilbyctr/jackbuilt.shtml). There are many web sites dedicated to the TI-59, SR-52 and the HP-65 Programmable Calculators, including many applications that run on them. It is amazing what people will patent. Designs for the first TI hand held calculator and for the HP-calculators stored in the vaults of the patent office at the site previously mentioned. They are readily accessible in text and TIFF format on the WEB. The intricate hand drawn schematics are wonder to look at, and must have taken draftsmen many long hours to draw since schematic capture software tools (CAD and CAE) were not available then.

Texas Instruments along with having invented the original handheld calculator, (Patent # 3819921) in 1967, pioneered the Algebraic Operating System (AOS), (Patent # 4208720) that was first used by TI in their SR-52 Calculator late in the 1970's. AOS is simply a way of evaluating expressions using the order of precedence of operators, which we all learned in grade school as prefix notation (also known as My Dear Aunt Sally (MDAS)). In their patent disclosure they describe in great detail how the AOS system works. This calculator boasted the capability to handle complex expressions including up to 9 levels of parentheses 224 program steps and 20 memory registers, and also included a

card reader /writer similar to the HP-65 calculator. Check out the following link for more information:

<http://www.ti.com/corp/docs/kilbyctr/jackbuilt.shtml>

TI-59 soon calculator followed It also was programmable with up 960 steps or 100 memory registers steps and allowed complex expressions to be entered with parentheses and first to use ROM Solid State modules (Patent #4153937) containing various applications and also included a printer attachment. I had a TI-59 as a freshman in college and it was very useful in my physics and math classes. I also programmed it for TIC-TAC-TOE and space war for entertainment, instead of wasting quarters at the arcades. I remember buying a TI-59 for college. I programmed it for Tic-Tac-Toe and Space War as a cheap substitute to the coin operated arcade games that gobbled up so much of my lunch money.

The other method for entering expressions is prefix notation also known as Polish Notation in honor of Jan Lukasiewicz who developed the notation to specify arithmetic expressions without the need to use parentheses in the early 1920's. HP modified the format slightly for use on their calculators by swapping the order of the operator (function) from the front to the back and renamed the notation RPN.

<http://www.hpmuseum.org/rpn.htm>.

If the reader is interested in HP calculators the virtual HP museum on the NET has galleries of photos, articles etc., relating to HP calculator products. It is located at

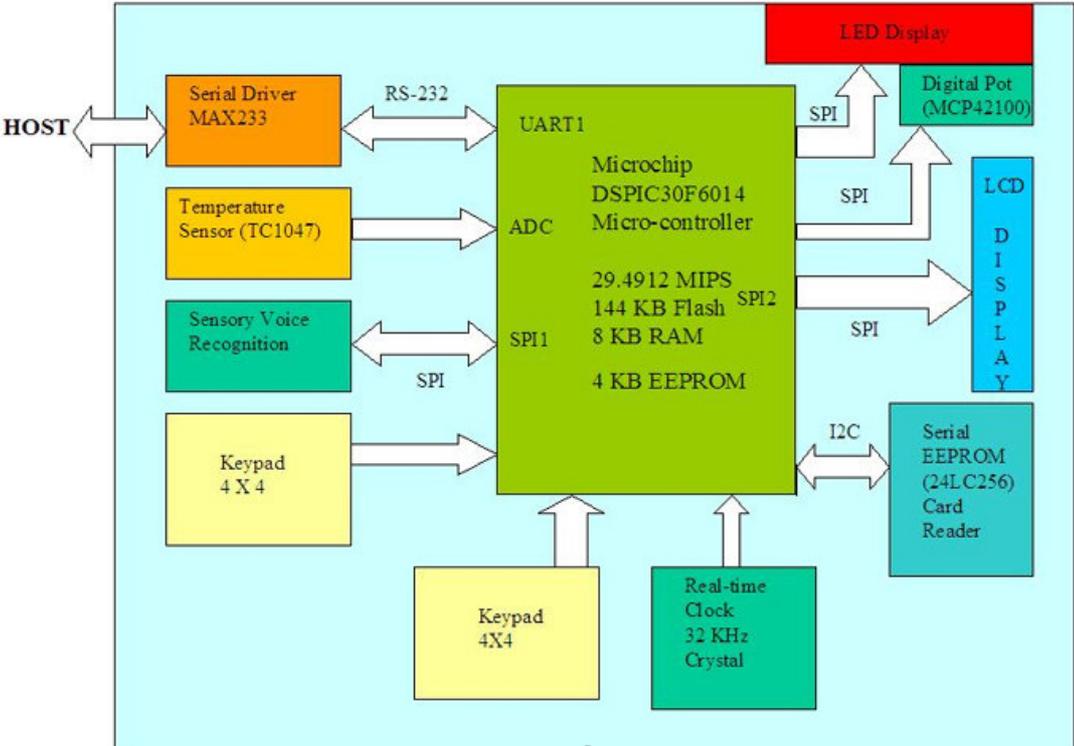
<http://www.hpmuseum.org/>

There are many ways that a modern Programmable Calculator can be built including using a micro-code emulator to run the actual HP or TI microcode instructions but I decided against this due to the fact that the microcode is patented and copyrighted by HP and TI. Another method is to use VLSI techniques to emulate the Programmable Calculator hardware but again I did not choose this route because of the overall complexity and expensive tools required. Instead I chose to use the Microchip DSPIC30F6014 as the Programmable Calculator's Central Processing Unit (CPU) by utilizing the dsPIC 1.1 Development Board (which I will refer to as dsPIC from now on) as the Programmable Calculator mother board, and the DSPIC30 C math and peripheral libraries for implementation of the emulated TI-59 functions, including using the [IEEE Standard 754 Floating-Point](#) double precision floating point for all the emulated TI-59 arithmetic functions. With the dsPIC I get the LCD Display (which the original HP-65 and SR-52 did not have) but newer models such as the HP-49G and TI-89 Graphing calculators do. I mention this fact only because for the current application to emulate an HP-49G requires minor modifications to the hardware and firmware as I will point out later on.

Because of personal time constraints I was not able to emulate and test all the TI-59 functions and additional features. Instead, I chose to implement and test only a subset of the TI-59 arithmetic and programming functions since the remaining functions follow the same patterns and leaves the reader the option of completing them if so desired. The TI-59 AOS calculator functions that I completed work well as shown in the examples provided with this article. I was also able to get most of the TI-59 programming capabilities emulated and

working as shown in the TI-59 test program that I provide to calculate the Statistical Mean and Standard Deviation functions. I also took some liberties with the encoding of the keystrokes with regards to the “Gold” functions that dedicated TI-59 users may have issues with. The knowledgeable reader could easily resolve these issues by investing some time and making a few changes to the firmware. A block diagram for my version of a Programmable Calculator is shown Figure 1.

**Figure 1 – Is a block diagram of the TI-59 Programmable Calculator.**



A harder task for the reader is to actually emulate the SR-52 or TI-59 Programmable Calculator with its pioneering Algebraic Operating System (AOS) used to evaluate parenthesized arithmetic expressions. The big hurdle here is to develop a parser to convert expressions from AOS to RPN and just use the RPN interpreter, which I already provide to evaluate them. The keypad layouts are suitably changed to match the SR-52 or TI-59 keypads, and the function key coding is changed accordingly. Writing a parser for AOS is beyond the scope of this article but many sources and examples exist using the popular Unix YACC and LEX tools that are used for compiler development. The TI AOS notation is used to handle complicated parenthesized expressions by using the grade school MDAS order of operations algorithm:

- 1) Parentheses and functions
- 2) Multiplication
- 3) Division
- 4) Addition
- 5) Subtraction

Hewlett Packard calculators avoid this difficulty by using the Reverse Polish Notation (RPN) to handle complicated parenthesized expressions. Using RPN for evaluating expressions requires a new paradigm from the original My Dear Aunt Sally Rule. To use it simply start at the innermost parentheses and evaluate, pushing the intermediate results to the RPN stack and proceed to work out to the outermost parentheses, all the while using the HP-65 RPN stack manipulation keys: Enter, Roll Down, Roll Up, Exchange X and Y. Another advantage to choosing RPN for the calculator notation is the minimal hardware needed for the implementation, and must have been very important to HP engineers during the development cycle of the HP-65.

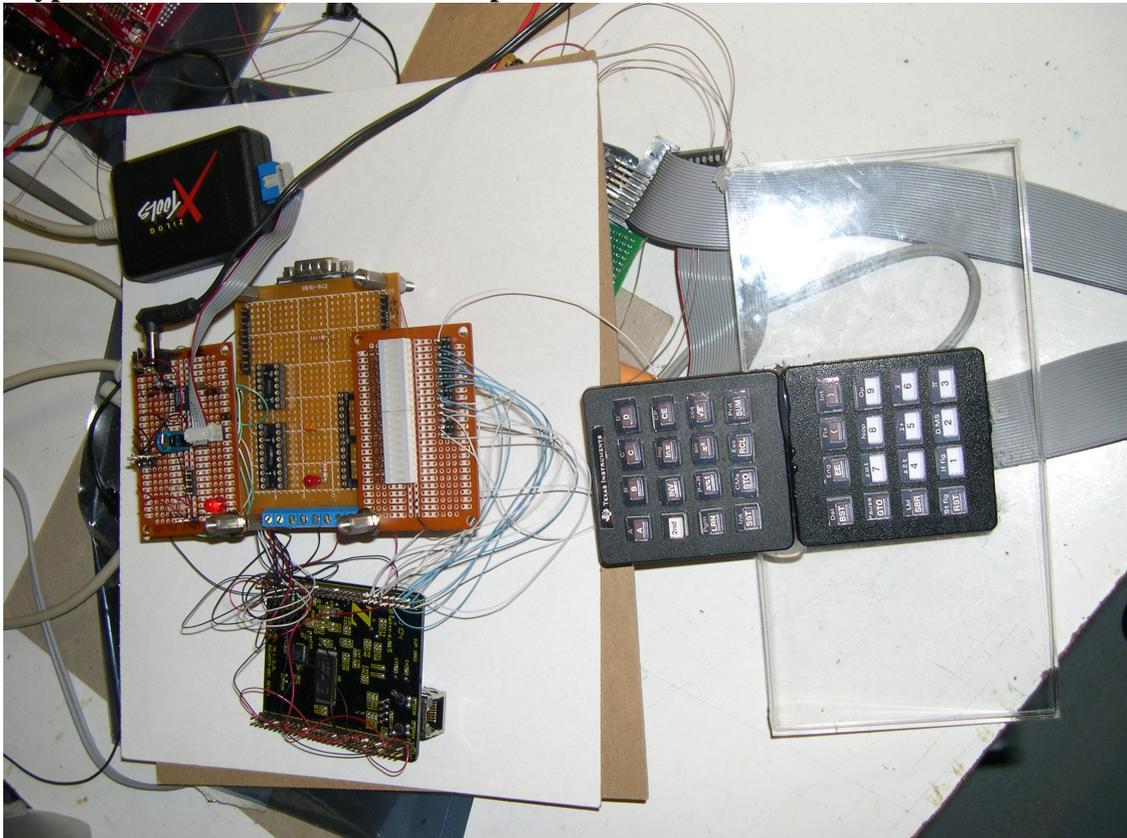
Although emulating an antique Programmable Calculator as a project does not fit the “cutting edge” my hope is that the reader new to the dsPIC architecture will learn from this article project how to work with the following hardware including: a numeric LED displays, an LCD display, two hex keypads, switches, pushbuttons, and the DSPIC30F6014 microcontroller. In addition to developing software for Interrupt Service Routines (ISRs), and peripherals such as the SPI, UART and low level LED Display drivers, LCD Display drivers and keypad drivers, the software covered in this article will also provide an introduction to various calculator languages, such as My Dear Aunt Sally (MDAS), Reverse Polish Notation (RPN), Algebraic Operating System (AOS) and FORTH as well as some of the tools needed to create lexical analyzers and parsers for them. By learning how the Programmable Calculator works, the reader will be able to better understand modern digital computers and how the input devices, output devices, computer languages and operating systems work, since an advanced calculator is very similar to a small laptop or palmtop computer.

Because of my lack of SMT skills, and time constraints I chose to develop a “large scale prototype” of the TI-59 to house my project instead of a compact calculator size model, but there is no reason why this design shown in Figure 2, could not be constructed on a small 2 inch square PCB using SMT technology by those skilled in this new technology and having it

manufactured by a PCB FAB house since the dsPIC30F6014 microcontroller is no larger than a US Postage Stamp, so that it could be integrated into a real pocket calculator. Think of the look your boss or co-workers will give you when you tell them you made this device from scratch. TI-59 owners and collectors may want to bring new life to their broken calculators by performing some microsurgery on them and installing a TI-59 emulator mother board, although the card reader would still be a problem. The reader building the board from scratch could just use the dsPIC schematic shown in Figure 3 as a starting point for the compact motherboard.

Instead, I used the dsPIC 1.1 Development Board as the basic design for the Programmable Calculator mother board using the on-board LCD display for debug and I/O and connecting the LED display, two keypads, and buttons to the board wire-wrap construction and pin headers that I soldered directly to the board prototype area as shown in Photo 3a-b.

**Photo 3a - Shows the prototype TI-59 Programmable Calculator including the custom keypad. Not shown are the thirteen pushbutton switches.**



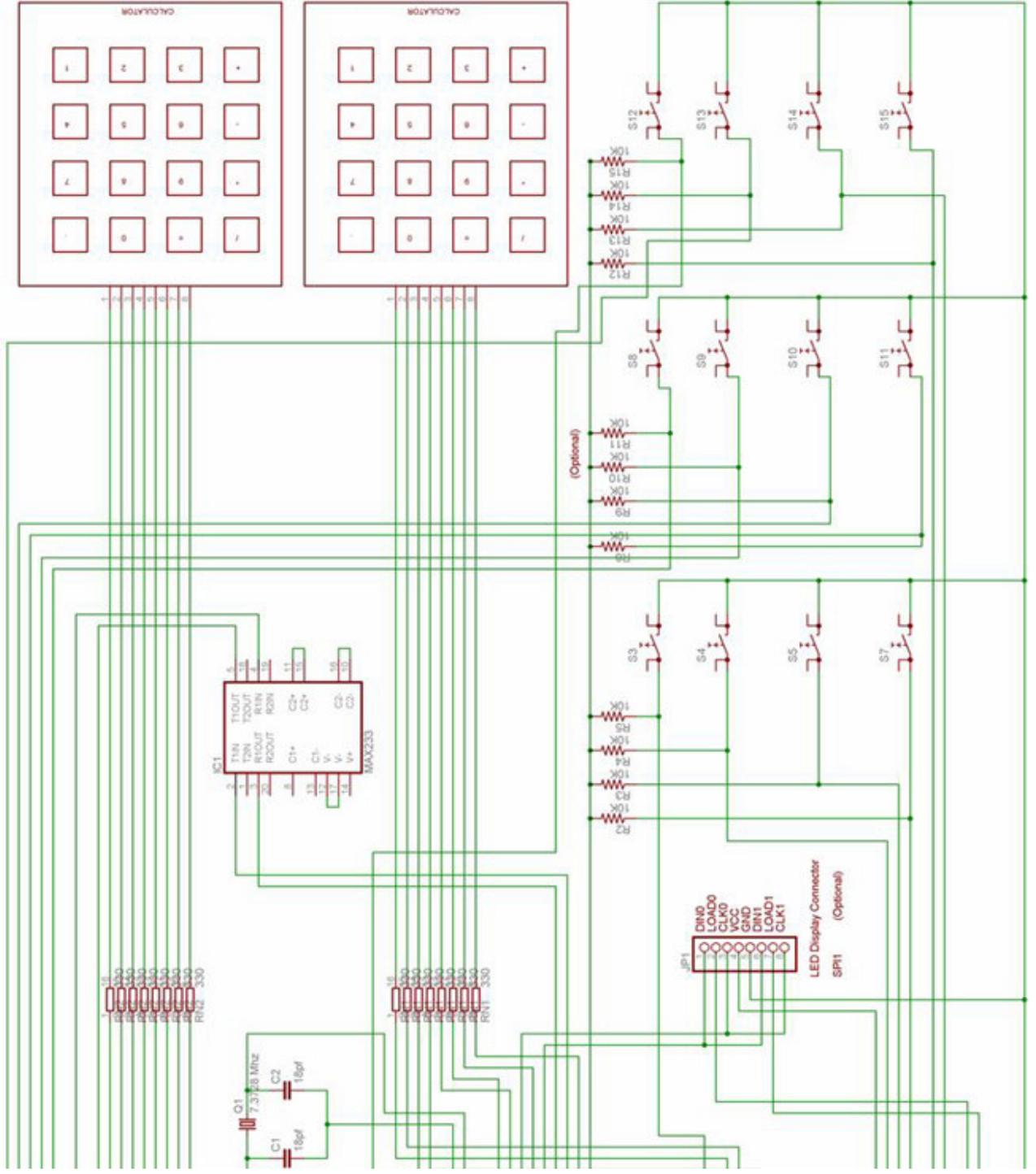
**Photo 3b - Shows the prototype TI-59 Programmable Calculator including the custom keypad with TI-59 function key labels glued to each keypad key. Not shown are the thirteen pushbutton switches.**



Readers are encouraged to modify this design as desired: For example, larger keypads or the addition of a sixteen segment alphanumeric LED display may be used, and financial, vector, matrix, trigonometric and scientific functions (FFT and Digital Filters) may be added if so desired. Applications for this project abound, for instance a portable data analyzer using the DSPIC ADC to collect the data and utilizing the FFT for generating the spectrum and the Digital Filters to signal condition the data collected. While other microcontrollers such as the 68HC12 from Motorola and the Microchip dsPIC30F6014 from MICROCHIP could have been used for this application I chose to enter the dsPIC article and use the dsPIC30F6014 because I felt it was ideal for handling the necessary double precision floating math processing for the emulation of the TI-59 math functions.

**Figure 2 – Complete schematic for the TI-59 Programmable Calculator**

# BLE CALCULATOR





## PROGRAMMABLE CALCULATOR EMULATOR HARDWARE

The complete TI-59 Programmable Calculator (designated as Calculator from now on) was shown in the block diagram. The boxes in dark forest green represent functions that I have not yet completed because of time constraints, but are part of the design. The calculator hardware configuration shown the block diagram was the easiest for me to build in time for the article, although with additional hardware, this design could be the basis for building the complete TI-59-based AOS or an advanced HP-41C based RPN calculator. I include thirteen extra push button switches used to fill out the missing row and column of function keys available on an TI-59. These pushbutton switches are connected to available microcontroller port I/O lines using 10K pull-up resistors. One terminal of each switch is pulled-up with a 10K resistor and connected to the available digital I/O port, while the other is tied to ground. In addition I used two 4 X 4 keypads to complete the TI-59 keypad matrix. Figure 3 shows the LED Display driver schematic to be used for driving Common Cathode LED digits.

## HOW IT WORKS

The Calculator's internal workings are similar to those of commercial programmable calculators and computers. User input and numeric key sequences are entered by means of the keypad or through the serial port when it's used as a numeric co-processor unit. The Calculator processes the selected operations by feeding data to the embedded microcontroller for evaluation. It then performs the calculations using standard floating-point arithmetic as shown in Listing 1. Answers are converted from the internal IEEE floating point formats to ASCII values and are sent to the desired output device, which includes the LED display, the LCD display, or the serial port connected to a host PC or Laptop. The co-processor mode uses the serial interface exclusively for its I/O. A unique feature of this calculator is that answers may be displayed in any combination of the three display methods mentioned above, as shown in Photos.

### Listing 1 – Shows the simple recursive descent algorithm used to evaluate mathematical expressions using double precision floating point.

```

/*****
/* Expression - Evaluate a floating point expression using recursive descent.
*****/
double Expression(double ExprValue)
{
    Token_Key_T Addop;
    double NextTermVal;
    // Process the next term in the expression
    ExprValue = Term(ExprValue);
    while(in(Token, AddingOps))
    {
        Addop = Token;
        // Get next character
        ReadChar();
        // Process the next factor in the expression
        NextTermVal = Term(NextTermVal);
        switch(Addop)
        {
            case PLUS:
            {
                ExprValue = ExprValue + NextTermVal;
                break;
            }
            case MINUS:

```

```

    {
ExprValue = ExprValue - NextTermVal;
break;
    }
}
//return ExprValue;
}
/*****
/* Term - Evaluate a floating point term using recursive descent.
*****/
double Term(double TermValue)
{
Token_Key_T Mulop;
double NextFacVal;
// Process the next factor in the expression
TermValue = Factor(TermValue);
while(in(Token, MultOps))
{
Mulop = Token;
// Get next character
ReadChar();
// Process the next factor in the expression
NextFacVal = Factor(NextFacVal);
switch(Mulop)
{
case TIMES:
{
TermValue = TermValue * NextFacVal;
break;
}
case DIVIDE:
{
if (NextFacVal != 0.0)
{
TermValue = TermValue / NextFacVal;
}
else
{
// Report division error ...
ReportError();
}
break;
}
}
}
return TermValue;
}
/*****
/* Factor - Evaluate a floating point factor using recursive descent.
*****/
double Factor(double FactorValue)
{
Literal_Key_T The_Literal; // Holds the ascii literal for selected
numeric key
int The_Digit = 0; // The numeric value of the digit
// Convert the token representing a digit to a numeric value
strcpy(The_Literal, (char *) Get_Literal(Token));
#ifdef DEBUG
printf("Token = !%s! \r\n", The_Literal);
#endif
sscanf(The_Literal, "%d", &The_Digit);
if ((in(Token, Digits) || (in(Token, Functions) || (Token == PREVIOUS) ||
(Token == LEFT_PARENTHESSES))
{
// Process Digits 0..9
if (in(Token, Digits))
{
FactorValue = ReadNumber(FactorValue);
}
else
// Process "P"
if (Token == PREVIOUS)
{
// Get next character

```

```

ReadChar();
FactorValue = The_Previous_Result;
}
else
// Process "("
if (Token == LEFT_PARENTHESSES)
{
// Get next character
ReadChar();
FactorValue = Expression(FactorValue);
if (Token == RIGHT_PARENTHESSES)
{
// Get next character
ReadChar();
}
}
else
{
// Report error ...
ReportError();
}
}
else
{
// Report error ...
ReportError();
FactorValue = 0.0;
}
}
return FactorValue;
}

```

## **BUILDING THE PROGRAMMABLE CALCULATOR**

In addition to using the dsPIC 1.1 Development Board as the main platform (mother board) for the Calculator, I also included designs for the remaining hardware that may be assembled from electronic components found at Radio Shack, Digikey or Jameco. In this way we can recapture something of Pascal and Babbage's spirit of invention by building a calculator piece by piece in the same way that those men did. Its inner workings may differ somewhat from commercial calculators, but it is functionally comparable and may be customized by the reader to include more features. During the course of this article, I learned how all the calculator subsystems, such as the keypad, the display, and the DSPIC30F6014 microcontroller work together to make this device. I ended up using wire-wrap, point to point and soldering techniques to build the calculator electronics but the reader may use any method of construction desired.

## **BUILDING THE PROGRAMMABLE CALCULATOR CONTROLLER BOARD**

The schematic shown in Figure 2 can be used to build a Calculator controller board from scratch. Just use the power and decoupling along with the schematics for the UART and other features that you might want, that are provided with the dsPIC to complete the design. Point-to-point and PC board construction may also be used for this project. For the article I just interfaced the external hardware directly to the

The keypad is connected to the Calculator Controller board with either an 8 or 14 pin ribbon. Mark pin 1 on the ribbon if there is no marking for it already, and connect it to the keypad. The 330-Ohm resistor network is used to prevent ESD damage to the sensitive Calculator Controller electronics. Use a Digital Multi-meter to check all the connections between the keypad and the controller.

At this point, it's time to check the circuit for shorts or open lines by using the Digital Multi-meter to check continuity on all power, ground, and logic signals. The board may be inspected with a magnifying glass.

## **THE PROGRAMMABLE CALCULATOR BRAIN**

The use of a microcontroller for this application is unavoidable unless EPLD, FPGA or VLSI techniques are used. I chose the DSPIC30F6014 microcontroller, because I felt that, at around \$17.00 on E-Bay, it was more economical than a BASICX or ATOM microcontroller costing approximately \$50.00 that also supports double precision IEEE floating point.

The DSPIC has 144 Kbytes Flash, 4 Kbytes of EEPROM and 8Kbytes bytes of on-chip RAM capacity, most of which I used for the calculator firmware development. I selected a PLL active 16X for 29.4912 MIPS, using the internal 7.3728 MHz crystal for the oscillator settings for speedier calculations. The new Flash-based DSPIC30F6014, now available from Microchip uses the using ICSP interface for programming the on-chip FLASH and also takes advantage of low cost In Circuit debugging capability via ICD2 debugger.

I used the on-chip UART1 (RS-232) interface to communicate with a PC or laptop host via Hyperterminal Calculator for debugging and testing and can also be a substitute display for the LED or LCD display.

To emulate the TI-59card reader I planned on using the on-chip Microchip EEPROM, used as the primary programmable calculator storage device, can store up to 4 KB to be used for storing multiple 100 step programs. It may also be used for storing intermediate calculation results. Another possible function would be for its use by the firmware as a stack for evaluation of expressions or as a Forth or Basic interpreter. The capacity can easily be increased to 512 KB by dropping in a 24LC512 KB X 8 serial EEPROM device.

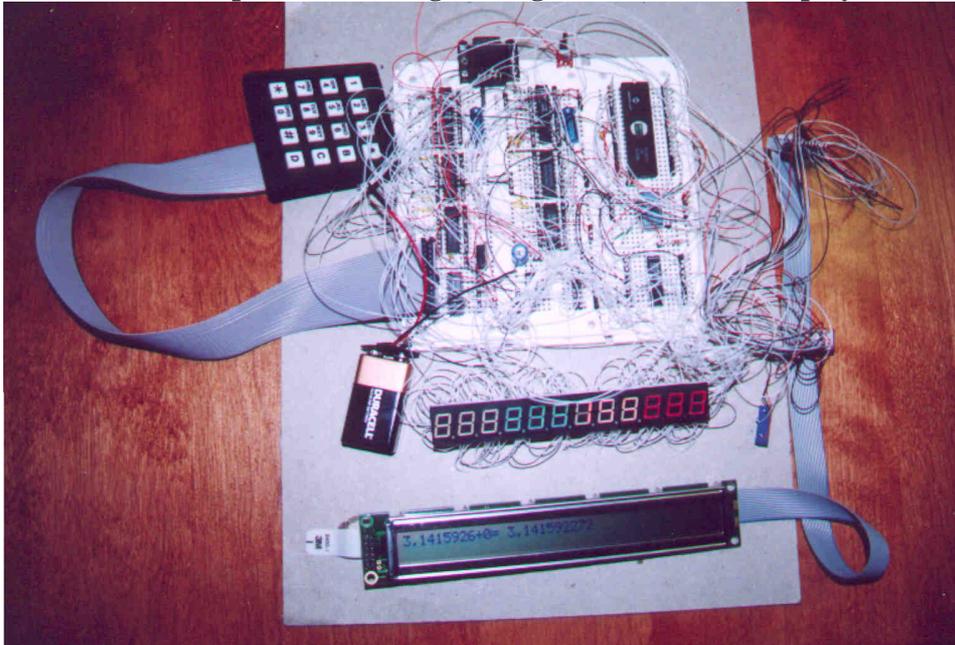
## **MAKING THE NUMERIC LED CALCULATOR DISPLAY**

In order to make an optional LED display easier to read under various lighting conditions, I decided to use the large digit LEDs to build the 12-digit numeric display as shown in Photo 4. The LED display module is assembled from individual LED digits, stacked together using Crazy Glue. To keep the digits aligned, I used a heavy bookend with a straight edge, and carefully glued one digit at a time. I let each set for a few minutes to dry before attaching the next digit. Note: Crazy Glue can glue fingers together, so it's important to avoid contact. If it gets on the skin, it should be wiped it off immediately.

Once the LED digits were glued, it was time to wire-wrap them as shown in Figure 3. The LED display BUS is made by connecting each of the leads in a daisy-chain fashion. This had to be done very carefully since the LED digit leads are delicate and can only support one level

1 wrap. It was while wrapping the display that I broke one of the leads and had to solder it. To avoid this problem, use the lead to wrap a special 2 level wrap with a manual wire-wrap tool. Only use ½ " inch from each end of the wire-wrap wire, rather than the usual 1 inch to accommodate the second wrap, and proceed to carefully wrap it on each LED pin with the hand-wrap tool.

**Photo 4 – Is a snapshot of the large 12-digit numeric LED Display Driver Board.**



## THE NUMERIC LED DISPLAY MAX7219 DRIVER BOARDS

The complete LED display board schematic is shown in Figure 3 and Photo 5. The MAX7219 LED digit display driver IC is very flexible. It is only needed if the optional LED Display is going to be used instead of the LCD or UART for displaying the numeric output. Although, it's one of the more expensive components at \$9.00, it's well worth the expense. It drives the LED digits' segments using a multiplexing scheme that keeps them from drawing too much power and also keeps them at the correct level of brightness. A single MAX7219 can drive up to 8 LED digits or 64 individual LED(s), although it can be cascaded to drive groups of 8 digits. It can also control their brightness using a PWM and a current limiting resistor (10K). The MAX7219 converts the binary values to BCD to drive the correct LED segments.

The DSPIC30F6014 sends data to the MAX7219 display driver using a 3-wire SPI interface using the SPI1, since SPI2 is reserved for the LCD Display. The configuration data, consisting of level of brightness, number of digits, and data format (BCD or Binary), are sent to the MAX7219. There is also a test mode that lights up all the segments of each LED digit connected.

Before populating the board with the IC(s), power the board by connecting a 6 Volt battery to the (+) and (-) power terminals, and check to see if the Power LED lights up. If it does, check for +5V at the VDD pin of each IC and V at the VSS pins. Once this has been done, the Board may be populated with the IC(s) and fired up for the first time.

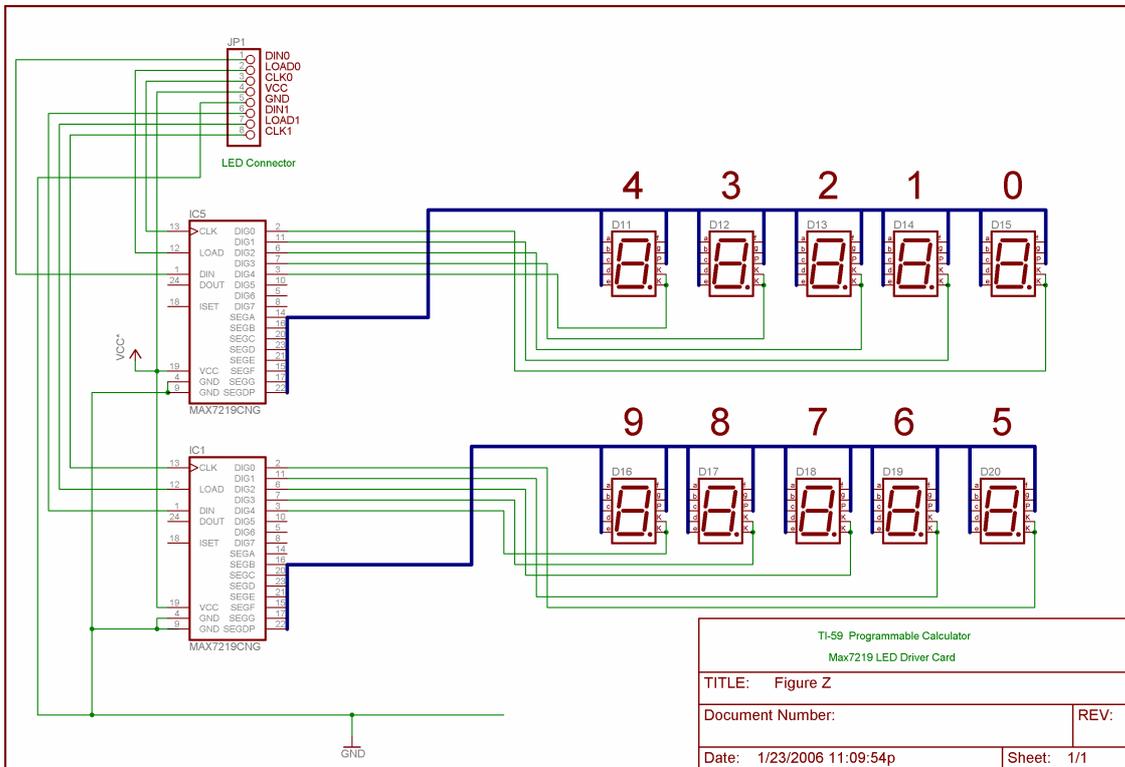
An interesting feature found on most cars these days is the ability to automatically adjust the dashboard instrument lighting. A similar feature could be built into the Calculator by using a photo-transistor or CDS cell to measure the ambient light and use it to adjust the LED digit brightness. The brightness may also be set via the MAX7219, using the "brite xx" command, where xx ranges from 0..15.

The MAX7219 can also be configured to drive popular matrix LED(s), although that would involve changes to the firmware to store the character set bitmaps. These bitmaps could be stored in EPROM, and the bits corresponding to a given character could be sent to the MAX7219. The advantage of using matrix LED(s) is that alphanumeric values could be displayed in a similar manner to the LCD displays. The firmware for the MAX7219 is available with the contest materials in the file max7219.c

**Photo 5 – Shows the colorful LED display that I recycled for use with the TI-59 Programmable Calculator.**



**Figure 3 - The complete LED display board schematic is shown in this schematic using the MAX7219 LED digit display driver IC.**



**Photo 5 – Shows the MAX7219 LED Display Driver Board.**

## USING THE BUILT-IN LCD DISPLAY

The multi-line LCD PG1223D-1 8-Bit display that is available on the dsPIC is put to good use for debugging and for easy to read I/O from the calculator. The HP-41C was one of the first calculators to use this type of display and if you happen to prefer LCD displays or if you don't want to take the time to make the LED display then all I/O can be redirected to the LCD display only by removing the LED DRIVER support flag from the flags.h file. The LCD Driver is a separate PIC18F242 Programmed to control the LCD via the SPI2 interface.

Advantages to the LCD display include low power consumption necessary for independent battery operation of a calculator, alphanumeric capability 4 lines X 20 characters or graphics consisting of 120 X 32 pixels using a standard SED1520 LCD controller, that could be put to use for emulating one of the latest HP or TI graphic calculators, and finally its simple SPI interface. Most of the LCD capabilities are available for other applications using the functions in the lcd.c.

## **SERIAL INTERFACE**

The Calculator's serial user interface (UI) uses the dsPIC's UART1 to communicate with a Host PC or Laptop using Windows Hyperterminal. The UI interface was initially developed for debugging and testing calculator functions, including the floating point I/O routines. The UI consists of a simple prompt ">" that accepts text input from the operator and displays the answers to the terminal in text form, and is always available by default to the Calculator. The user enters calculator keystrokes and operands to evaluate floating-point expressions and the results are displayed on the PC or Laptop's screen. Otherwise the TI-59 Calculator keypad and LCD/LED display provide all the I/O required allowing the user to enter expressions and display the results.

## **USING THE BUILT-IN LCD DISPLAY**

The multi-line LCD PG1223D-1 8-Bit display that is available on the dsPIC is put to good use for debugging and for easy to read I/O from the calculator. The HP-41C was one of the first calculators to use this type of display and If you happen to prefer LCD displays or if you don't want to take the time to make the LED display then all I/O can be redirected to the LCD display only by removing the LED DRIVER support flag from the flags.h file. The LCD Driver is a separate PIC18F242 Programmed to control the LCD via the SPI2 interface.

Advantages to the LCD display include low power consumption necessary for independent battery operation of a calculator, alphanumeric capability 4 lines X 20 characters or graphics consisting of 120 X 32 pixels using a standard SED1520 LCD controller, that could be put to use for emulating one of the latest HP or TI graphic calculators, and finally its simple SPI interface. Most of the LCD capabilities are available for other applications using the functions in the lcd.c.

## **PROGRAMMABLE CALCULATOR KEYPAD**

The custom TI-59 keypad is made from two standard 4 X 4 keypads along with four push button switches, two slide switches to for a keypad with enough keys for an TI-59 calculator. The keys are mapped to functions as shown in Table 1, using a keypad encoding scheme that I made up to map the row: column key codes along with the state of the special "gold" function keys found on TI-59 Programmable calculators. For convenience, these codes map linearly to individual tokens representing all possible key strokes.

The on-board pushbutton switches are interfaced via the Change on Notice (CN) interrupts using CN interrupt service routines (ISRs) to obtain their current states . I could have used the CN interrupts for scanning the keypads PORT B status interrupt, but I chose not to in order to keep the firmware simpler by minimizing the number of ISRs.

**Table 1 - Keypad Layouts for the TI-59**

TI-59 Default key labels layout table:

"A	","B	","C	","D	","E	","
"2nd	","INV	","lnx	","CE	","CLR	","
"LRN	","x<>t	","x^2	","SQRT	","1/x	","
"SST	","STO	","RCL	","SUM	","Y^x	","
"BST	","EE	","(	",")	","/"	","
"GTO	","7	","8	","9	","X	","
"SBR	","4	","5	","6	","-	","
"RST	","1	","2	","3	","+	","
"R/S	","0	",".	","+/-	","="	","

TI-59 "2nd" Gold key labels layout table:

"A*	","B*	","C*	","D*	","E*	","
"	","	","log	","CP	","	","
"Pgm	","P->R	","sin	","cos	","tan	","
"Ins	","CMs	","Exc	","Prd	","Ind	","
"Del	","Eng	","Fix	","Int	"," x	","
"Pause	","x=t	","Nop	","Op	","Deg	","
"Lbl	","x>=t	","Sigma+	","x_bar	","Rad	","
"St flg	","If flg	","D.MS	","Pi	","Grad	","
"Write	","Dsz	","Adv	","Prt	","List	","

## DEBOUNCING THE CALCULATOR KEYS

Depending on the mechanical and electrical characteristics of the selected keypad, "extra" keystrokes may be encountered each time a key is pressed. This anomaly can be taken eliminated by de-bouncing the keypad keys -- usually by placing a 20-millisecond delay in a loop where the key is being polled.

I wrote my version of de-bounce in DSPIC C30 C from information provided in Myke Predko's book [7] . Myke's de-bounce routine in DSPIC assembly was re-hosted to C so that I could include it for my Calculator. The algorithm worked flawlessly, filtering multiple keystrokes. In addition it debounces the three extra pushbutton switches required for the missing calculator keys. The source code provided debounces two 4 X 4 keypads. Perhaps others may benefit from using the C version please refer to the C source files keypad.c and keypad.h, which I provide with the article materials for more information.

## LED DISPLAY +5 VOLT POWER SUPPLY

A good power supply is a must for the LED display. Even though the LED digits are multiplexed, they can still draw quite a lot of power when all the segments are turned ON, for example, when displaying the number 8. Earlier, I mentioned the problem of only being able to display a total of 10 large LED digits, which I believe could be due to excessive current drawn by each LED. The power supply must be able to handle this load along with the power required for the DSPIC30F6014 and all the other IC(s) used in the calculator electronics. A +5 Volt supply sufficient for all these power consumption requirements is shown in Figure 4. The project may be powered by a 9 Volt Alkaline or NICAD rechargeable battery, or even a 9 to 12 Volt wall transformer, as shown in the figure. Depending upon the number and type of devices connected to the Calculator Controller board, a heat sink may be needed by the 7805 IC if it gets too hot during normal use or shuts down, although I didn't require one for my



The financial, trigonometric, and scientific functions are left as a future exercise for those interested in numerical methods.

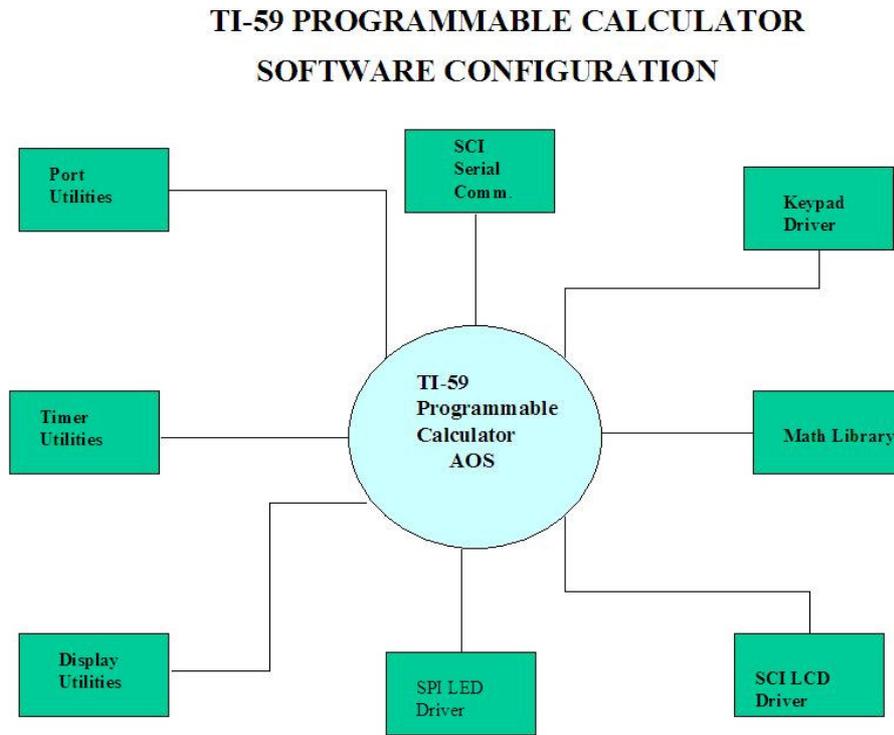
## **CALCULATOR FIRMWARE**

Firmware for the Calculator application, `ti59.c`, was developed from requirements shown in Figure 5, based on using the previously-mentioned historical information and my personal experience using programmable calculators. I didn't attempt a complete emulation, since that would require extra hardware and development. Instead, I opted to emulate most of the basic features, while leaving out the most complex. As such, the calculator is useful as a demo or learning tool, but not for work where more precision and accuracy are required.

The complete application is ready to be programmed into a `dsPIC30F6014` using the files provided with the article materials. For those interested in customizing the software, I divided the firmware for various hardware functions into separate C files to facilitate development, debugging, and maintenance. These modules are shown in Figure 6, which is a high-level block diagram of the Calculator firmware. In my opinion, this is the best way to organize a large application such as this, since in this way, it may be successfully tested and integrated into the hardware one module at a time. There are better methods for organizing the firmware, using the Object Oriented Programming methodology with a language such as C++ or Java. Although I would like to see a C++ or Java compiler for the DSPIC, it would require a very high-end dsPIC with lots of on-chip RAM and Flash.

Values entered into the TI-59 registers are sent to the LCD, LED, or serial port, depending on which driver flags have been defined in the file `flags.h`. If the keypad driver flag hasn't been defined, input is redirected from the keypad to the serial port. Otherwise, input comes directly from the keypad, which means the application may hang if the keypad isn't connected to the Calculator hardware.

Figure 5 - Shows the TI-59 Programmable Calculator firmware structure



## PROGRAMMABLE CALCULATOR MODES

There are three modes of operation available to the user. The Calculate Mode is the default mode of operation. In addition there is a Program Mode and a Run Mode. The program mode also provides rudimentary editing capabilities when entering an AOS program. Not all functions from these modes are available since I was not able to complete them in time for the article.

The AOS Calculator mode is the default mode when the unit is powered up. See Diagram\_5, for a list of currently available Calculator functions. In this mode, the firmware works by simply reading keystrokes from the serial port for user input, converting from ASCII to IEEE floating point operands, and using them to build simple arithmetic expressions with the floating point numbers and the basic operators "+", "-", "X" and "/", "E" as delimiters. The "E" character is used for exponential notation when entering floating point numbers when using UART1 for I/O.

Even though I haven't completed this feature of the Calculator, it's possible to use the DSPIC's internal EEPROM to write AOS programs to EEPROM and read them back in such a fashion to emulate the TI-59 Card Reader for non-volatile storage capabilities using the 4 Kbytes of EEPROM. This capability can be extended by using external serial I2C EEPROM with up to 512 Kbytes if a Microchip 24LC512 is interfaced to the calculator.

## **ADVANCED PROGRAMMABLE CALCULATOR SOFTWARE**

I used algorithms provided in Jack Crenshaw's column in the Embedded Systems Programming journal and in his new book titled "MATH Toolkit for REAL-TIME Programming" [2] for this article for the development of some missing arithmetic functions.

I did not have time to complete the financial functions and statistical functions that were available on the original HP-65. While developing these missing functions I referred to the "Numerical Recipes in C" [3], which is another excellent book for the hobbyist's bookshelf. The book is a classic that has been available for years. It includes reliable algorithms for advanced mathematics plus algorithms in both Fortran and C for: least square fit, rms, mean, standard deviation, FFT, interpolation, integration, etc.

Advanced expression evaluation that either completely eliminates parentheses or allows the use of parentheses in a mathematical expression may be explored with calculator languages such as RPN, AOS, Forth, and Basic. Special unit conversion functions, such as those used in Physics, Astronomy, Electronics, and Mechanics are also readily added. In addition, modern calculators now have advanced graphing capabilities with an LCD display. These capabilities could be added to the Calculator with additional software.

## **GOING FURTHER**

Some simple, easy-to-do improvements to the Calculator could include adding larger numeric LED digits for the display or the ability to drive large Matrix LED displays. The LED display could also be improved by incorporating the new MAX6954 Display Driver that drives the 14/16 segment displays. It would allow the TI-59 to display short alphanumeric messages similar to the HP-67 and HP-41C programmable calculators. The MAX6954 uses an SPI interface similar to the interface used by the MAX7219 LED Driver IC, so changes to the firmware would be minimal for this improvement.

Information provided in this article, can be the starting point for a complete TI-59 AOS-based Programmable Calculator emulator. A simple recursive descent calculator algorithm written in Pascal, as described in Niklaus Wirth's classic book "Algorithms + Data Structures = Programs" [1], may be found at your local technical library or on Amazon is a good starting point.

## **CONCLUSION**

For this article I described in great detail all the firmware required to make the Programmable AOS Calculator perform useful mathematical functions including trigonometric, scientific and conversion functions. In addition I discussed calculator languages used on popular commercial calculators from HP, TI and Casio.

This calculator although educational, is not really useful for everyday use since it is still missing some functions, and has not been thoroughly tested for accuracy and correctness.

Instead think of it as a training tool for learning embedded microcontroller and DSP development using Microchip tools and the dsPIC.

## **REFERENCES**

books:

[1] Wirth, Niklaus, "Algorithms + Data Structures = Programs", Prentice-Hall Series In Automatic Computation.

[2] Crenshaw, Jack W., "MATH Toolkit for REAL-TIME Programming", CMP Books, CMP Media Inc., 2000

[3] Press, William, Flannery, Brian, Teukolsky, Saul, Vettering, William, "Numerical Recipes in C", Cambridge University Press, 1993

[4] Holub, Allen I., "Compiler Design in C", Prentice Hall Software Series, 1990

[7] Predko, Michael, "Programming & Customizing DSPICmicro Microcontrollers", McGraw-Hill/TAB Electronics; ISBN: 0071361723; 2nd edition (December 4, 2000)

datasheets:

[11] Maxim, "Serially Interfaced, 8-Digit LED Display Drivers/MAX7219", 1997.

sources:

Microchip Technology Inc 1-800-344-4539, [www.microchip.com](http://www.microchip.com)

## **BIOGRAPHY**

Daniel Ramirez is currently a Senior Software Engineer with over 10 years of experience working on Real-Time Embedded Systems. Hobbies are Travel, Golf, Treasure Hunting, and Robotics.