SWR/Power/Return Loss monitor – Details of the Software

All of the code for this project was written to run on the Propeller chip, a product of Parallax, Inc. using the free development tools available from Parallax. This chip is designed around a unique architecture using eight independent, identical processors in a single package. These peripheral processors are known as “cogs” and are capable of executing tasks either independently or cooperatively. All cogs are driven from the same clock source and a central “hub” processor cycles through all cogs and synchronizes their actions.

The code for the SWR monitor is written entirely in the Propeller's SPIN language, a high-level language similar to BASIC. Parallax supplies a complete software development package called Propeller Tools, free for the downloading from their web site, www.parallax.com. Also available at this web site is a library of useful self-contained software objects. From this library, the floating point math package, *Float32.spin;* floating point to string conversion utility, *FloatString.spin;* and the serial communication utility, *FullDuplexSerial.spin*. are also used in the code for this instrument.

The complete code for the SWR monitor instrument, ready to be programmed into the Parallax USB Development Board, can be found in the file SWRMON_090816.SPIN . The source code is heavily commented, and is reasonably self-explanatory.

Programming the Propeller Development Board

The USB port and logic-level converter built in to the Propeller USB Development Board makes it easy to load programs using the Propeller Tools. No additional hardware is required other than a USB cable with a mini-B connector. With the USB cable in place, it's only necessary to load the Propeller Tool suite into the host computer and call up the file containing the program to be installed – in this case the SWRMON_090816.SPIN file. Make sure that copies of the three library utilities *FloatString.spin, Float32.spin* and *FullDuplexSerial.spin* have been loaded into the same directory as the SPIN file.

Pressing the F10 key loads the software into the RAM area of the Propeller chip on the board and then executes it. Key F11 does the same thing, but also loads the same software into EEPROM for non-volatile storage. When stored in the EEPROM the program will run automatically whenever the board is powered up or the chip is reset.

Program Flow Diagram

The software for the SWR/Power/Return Loss monitor is made up of a number of software objects housed in several independent cogs. The accompanying software flow chart shows their interaction. Details of the primary software objects are given below, in pseudo-code form.

The MAIN object

The MAIN program object runs first, in its own cog, and starts three other cogs: ADC3CH, PWM and MONITOR. ADC3CH, in turn, starts a new cog, ADC, and MONITOR starts a new cog, PCMON. MAIN also uses library objects FullDuplexSerial.spin and Float32.spin. These last two objects also run in cogs of their own, so that all eight cogs are in use.

Pseudo-code listing – MAIN

```
Set up pin directions
Check out a lock
Start objects in separate cogs:
        AD3CH, PWM, MONITOR, FP (floating-point library)
Enter endless loop:
        Wait until lock is set by the ADC object
        Read data block from AD3CH object:
                forward power, % reflected power, net power and SWR
        Clear lock
        Convert power and SWR data to floating point numbers
        Calculate return loss (db)
        Get state of the DIP switches
        If XLO is enabled and SWR exceeds programmed threshold
                Energize XLO relay and front panel LED
        Get state of the front panel switches
        Write selected data channel to the PWM object
```

The AD3CH object

Pseudo-code – AD3CH

```
Set up multiplexer and LED pin directions
Start A-to-D converter in a new cog
Get initial analog zero level (MUX channel 0)
```

Enter endless loop:
    Wait until lock has been cleared by the MAIN program, then set lock
    Sample forward and reflected voltage amplitudes (MUX channels 1 and 2)
    Correct for zero offset
    Call PEAKHOLD object
    Call LPFILTER object
    At intervals of 10 seconds, take another zero reading and flash LED
    Clear lock

The PEAKHOLD object

This object looks for a peak in the forward power reading and, when it finds one, stores it in memory and starts a software timer. PEAKHOLD is called on each pass of the AD3CH loop, and each time through, the stored peak value is decreased by a small amount, until its value is less than the newest value of forward power. Net power is treated in the same way. If the "Peak Hold" switch on the front panel is ON, these values are used by the MAIN program for display on the panel meter.

The rate of decay is controlled by a program constant (accel) stored in the CON block.

Pseudo-code – PEAKHOLD

IF the newest sample of forward power is less than the value stored in hpkfwd
    Decrease hpkfwd by a fractional amount determined by the constant accel and by the contents of the timer.
ELSE update hpkfwd with newest sample of forward power and start a timer.

The LPFILTER object

This software object implements a single-stage low-pass IIR filter which operates on the data stream generated by the Analog-to-Digital Converter (ADC) running in AD3CH.

The PWM object

The data channel selected by the front-panel rotary switch is converted to analog in the form of a pulse-width modulated (PWM) waveform. The pulse repetition rate is 100 Hz and the duty cycle ranges from 0 to 100%, corresponding to a digital input value of 0 to 4096. The PWM waveform is transmitted from port P5

and is smoothed by a resistor-capacitor low-pass passive filter, which provides dc to drive the analog meter mounted on the front panel.

The MONITOR object

This object handles the interface to an external computer running a terminal display program, similar to HyperTerminal. The same USB port used for programming the Propeller Development Board is also used to communicate with the terminal program. All five channels of data (forward power, Percent reflected power, net power, SWR and Return Loss) are converted to ASCII strings and written out the USB port for display on the terminal. Depending on the command received from the keyboard, this object sends out data records either singly, when the Enter key is pressed, or at regular intervals for logging purposes.

For proper operation, the terminal must be set up for 19200 baud, 8 bits, no parity, and one stop bit. Further details of the data display and the keyboard commands can be found in the accompanying document “Operation and Features”.