CWVM -.-. .-- …- -- A CW VoltMeter

A station voltage monitoring meter for the blind/visually impaired

WA7ZVY

CWVM is a station monitoring voltmeter that outputs the voltage reading in Morse code.

The CWVM uses a PIC12HV615 microcontroller that contains a four channel 10 bit A/D converter, 1K of code space, 64 bytes of register RAM, three timers, along with a built in shunt regulator to keep the power supply simple.

The firmware was written using Microchip's MPLAB toolset.

The CWVM circuit board, speaker, and push-button power switch are installed in a small ABS plastic box.

The CWVM operates from +7 volts to +25 volts. Pushing, and holding, the power button results in a Morse code voltage report similar to: "13.80V" followed by a 2 second delay. The report repeats (with a new A/D sample each time) as long as the button is pushed.

Since each CPU has a slightly different shunt regulator voltage, which I've used for the PIC's internal analog to digital converter sub-circuit supply (a PIC programmed configuration setting), the resulting voltage measurement could require some fine tuning. My attempts at using the high precision internal reference voltage in the PIC to do an auto-compensation routine to null out variations from device to device never did work out as well as I wanted. So, I finally gave up and added a 10 turn potentiometer to do the fine tune voltage measurement calibration. The code for the auto-compensation is still in the PIC code source files and can be enabled with a #define, if someone wants to attempt to make that feature work (or re-write it!).

To make the CW speed user settable, the latest version (Rev A04) also includes a 10K potentiometer feeding a second analog to digital converter channel in the PIC12HV615. The potentiometer allows the speed to be set between 5 WPM and 45 WPM.

For power supply voltages greater than 16V for the CWVM, R1 needs to be changed to a higher wattage resistor (or to a higher resistance value) if you are planning to have continuous duty cycle operation. Otherwise, for brief reading sessions (15 seconds or so), the 0.25 Watt resistor is adequate.

Circuit Description:

J1 is the power supply and voltmeter input connector (circuit board pads). A push button switch is used to turn on the CWVM and perform a voltage reading. The switch's output feeds J1 pin 1. A 9V battery could be used to power the CWVM, or one can use the same voltage that you are measuring (as long as it is a reasonably low impedance power source). As long as power is applied, the CWVM outputs Morse code voltage readings, thus the reason for the push button switch. Push and hold while doing a voltage

reading. The voltage input attenuator is not a high impedance, so be aware that it does put a load on whatever you are measuring (up to 4mA with a 25 volt input signal, less at lower signal voltage levels).

R1 is the load resistor for the PIC CPU's internal shunt regulator. See the PIC12HV615 data sheet for information on the value of this resistor. C1 and C3 provide power supply noise filtering. D2 provides protection for a reversed power supply connection.

R12, R3 and R4 form a 5 to 1 voltage attenuator to scale the input voltage down to a level that the PIC can measure. The maximum input voltage allowed into the PIC's analog to digital converter is +5V; therefore with the 5:1 attenuation, the maximum voltage that the CWVM can read is +25.00V. D1 protects the input from a reverse voltage connection and C4 helps to eliminate noise on the analog to digital converter input signal.

U1 is the PIC CPU and contains 1K words of instruction memory. U1 contains the analog to digital converters used to measure the voltage level and to also read the R10 potentiometer voltage output level, which sets the Morse code sending speed. The Morse code sending speed is set to 5 WPM when the potentiometer output is at +5V and 45 WPM when it is at ground potential. There are 16 different Morse code speeds between 5 WPM and 45 WPM inclusive. The speed values are arrived at by using a voltage to speed lookup table in the firmware code.

Q1 is the speaker driver for the Morse code sound generation. The parallel resistors, R6 and R8, set the speaker drive/sound level. And, as with the other diodes, D3 provides some level of protection from an accidental reverse power connection. J2 (circuit board pads) connects to a small 8 ohm speaker.

J3 is the in-circuit programming header connector for the development environment. R2 provides a level of isolation between the programming hardware's power source and the PIC's internal shunt regulator as a protective measure.

Firmware Description:

cwvm_main.asm contains the reset vector, interrupt vector, Morse code character elements (dits and dahs) table, Morse code speed table, and the main code loop. Upon code startup (when power is applied), the code jumps through the reset vector to the main code entry point. First it initializes the I/O ports of the PIC processor to be analog or digital compatible pins, sets their signal direction as input or output, and then initializes some memory variable locations. The PIC analog to digital converter then reads the voltage value from the Morse code speed setting potentiometer. A Morse code speed table lookup is performed to get the value to put into the PIC's Timer 2. Timer 2 is used to time each Morse code character element (in dit time lengths). Next the PIC's internal precision 0.6V reference voltage is read by the A/D converter to arrive at any necessary auto-compensation value to add or subtract from the actual CWVM voltage reading (the result of which is currently disabled from affecting the output reading with the "#define ZERO_CORRECTION TRUE" statement in the code). As of now, we are using the CAL potentiometer, R12, to fine tune the voltage calibration of the CWVM. Now the easy part happens, which is the actual reading of the CWVM's voltage input signal. The A/D converter reads the voltage value, the code adds or subtracts the auto-compensation value and then converts it from binary

to decimal nibbles. The code then turns on Timer 0's interrupt. Timer 0 is used to create the actual tone of the Morse code audio by changing the state of the speaker output drive pin on each Timer 0 interrupting event. For each decimal nibble of the CWVM's voltage reading, the *send_morse_char* routine is called and that Morse character is sent out the speaker. Finally, a two second delay is done. As long as the push button is pressed, the CWVM continues to do readings and sending of the voltage value in Morse code.

comm.asm contains the routines used to actually send the Morse code characters. *delay_dit_times, send_morse_element,* and *send_morse_character*.

hardware.asm contains the code, *init_hardware*, to initialize the PIC's I/O ports and the code, *timer0_irq_service*, to handle the Timer 0 interrupts that are used to change the state of the speaker output pin to form the actual audio tone (650 Hz).

utilities.asm contains the code *binary_to_decimal*, *divide_by_n*, *multiply_by_n*, *delay_100msec* and *adc_sample_delay*. These routines are all pretty much self explanatory. The *divide_by_n* and *multiply_by_n* routines perform very simplistic 16 bit operations. The *adc_sample_delay* is a small delay used to allow the analog to digital converter's voltage sampling capacitor (internal to the PIC) to stabilize before the actual analog to digital conversion is started.