INTRODUCTION

This compact instrument is intended to be a digital laboratory tool for hardware and, in some cases, software debugging. It contains four instruments in one unit: logic probe, single channel logic state analyzer, frequency counter and serial code receiver.

The only chip used is a PIC16F84A running at 10 MHz. The display unit is a LCD dot matrix alphanumeric module with 2 rows of 20 characters. The LCD is used as the display device for all functions, except for the logic probe which indicates low, high and pulse logic states on individual LEDs. Mode select, parameter change, function execute and ON/OFF switching is activated by two keys.

The probe tip is the common input for all functions, and the GND cable is used for connection to Vss of the tested circuit.

Although there are a lot of functions integrated in a single chip unit, it did not increase the complexity of hardware, as all functions are implemented in software. This enables a very good price/performance ratio.

The power supply is obtained by four 1.2V/180 mAh or 250 mAh NiCd batteries of LR03 (AAA) size. The instrument also has a battery manager, which supports automatic battery discharging and charging.

The source code is written in MPASM. As it is highly optimized for code space, most of the code could not be written in a modular format. For the same reason, a lot of subroutines have more than one entry point and some of them are terminated by a \texttt{GOTO} instruction instead of using a \texttt{RETURN} instruction.

FEATURES

- Stand-alone hand-held instrument
- Single chip design
- Built-in rechargeable power supply
- Easy to assemble and ready to use, no adjustment needed
- User interface with LCD output and command input by two keys
- TTL or 5V CMOS input, or direct input from RS-232C +/-12V signals

SPECIFIC FEATURES FOR INDIVIDUAL FUNCTIONS

Logic Probe

The low and high logic levels are displayed by LEDs, which are OFF if the probe tip is floating or connected to a hi-impedance (>220k) output. A pulse transition is detected and is indicated by turning on the LED for 80 ms.

Logic State Analyzer

The analyzer fetches 300 single bit samples at a selectable rate (in 16 steps from 40 Hz to 1 MHz). It has a programmable start at High-to-Low or Low-to-High transition at input. Digital waveforms are displayed in a pseudographic mode on the LCD.

Serial Code Receiver

The Serial Code Receiver receives 42 bytes and displays them in both HEX and ASCII. The baud rate is selectable in 8 steps, from 1200 to 115200. The selectable format is 7 or 8 bits with or without a parity bit which is not displayed. Signal polarity is also selectable. Direct signal stealing from an RS-232 or an RS-232C interface is possible.

Frequency Counter

The Frequency Counter counts frequency and displays it in an 8-digit decimal format on the LCD with a refresh rate of 500 ms. There are four ranges, from 5 to 40 MHz, which affect the count resolution (from 4 to 32).

Battery Manager

The Battery Manager provides for discharging with an automatic switch that changes to charge mode at 4V battery voltage, charging with 18 mA of constant current and automatic power off after 14 hours. Any DC source between 10V and 30V, at any polarity, can be used for charging.
SYSTEM FUNCTIONS

User Interface

There are four modes of operation: Analyzer, Serial Code Receiver, Frequency Counter and Battery Manager. The logic probe function is transparent in all modes except in the Frequency Counter.

In all modes, submodes are listed in the lower row of the LCD. The submodes list can be cycled through by pressing the right key, which moves the cursor (blinking block) to the right. The left key activates the selected submode (executes a function or changes the parameter state/value).

The right-most submode (right arrow symbol) acts as a shortcut jump to the next mode. After power-on (by pressing any key), a mode is chosen by pressing the left key, then the submode by the right key, and then the eventual parameter change or command execution by the left key again. The only exception is the Logic Probe function, the only action needed is to switch the instrument on, and the logic probe is ready to use.

In Analyzer and Serial Code Receiver mode, the asterisk (\*) is a special symbol for the "Start" command. When executed (left key pressed while the cursor is on asterisk), it causes the program to wait for a start condition or a start bit.

Although there is a manual Switch Off command (accessible in Battery mode), there is also the automatic power off after approximately 8 minutes of inactivity if no key is pressed. Note that the down counter for automatic power-off is "frozen", while the instrument is waiting for a start condition in analyzer mode and for the start bit in serial code receiver mode. Of course, the same applies to the discharging and charging processes, as another conditions are used to define the end of those processes.

Figure 1 represents the key functions diagram. The dotted line represents actions taken when the right key is pressed, and the solid line is for the left key. The cursor, which is the blinking block on the LCD, is represented as a solid block in Figure 1, but it is moved down on the drawing for clarity.

A variable (named REL) in the assembler source code, defines the position of the cursor on the LCD. If it is a '1' (default), the cursor will be placed on the first character of the command (or parameter). If it is a '0', the code will be assembled so that the cursor will be moved to the preceding location (if it exists) before the command.
FIGURE 1: HIGH LEVEL FUNCTION FLOWCHART

Analyzer
01MHz/01ms 1 →

Sample rate
1MHz/1ms 1 →
1MHz/1ms 1 →
500kHz/2ms 1 →
1MHz/1ms 1 →
250kHz/4ms 1 →
1MHz/1ms 1 →
100kHz/10ms 1 →

Starting condition
L. →

Start

I. →

Display

1 group
1-7

2 group
7-14

3 group
15-21

4 group
22-28

5 group
29-35

6 group
36-42

1MHz/01ms 1 →

Display Group submode (group 3, samples 121-180)

LCD output example while key 1 is depressed

LCD output example while key 1 is released

(group 4, samples 181-240)

Serial
19.2 8 * 1 →

Baud rate
1.200
2.400
4.800
9.600
19.200
38.400
76.800
153.600

Frequency 12345678 →

20MHz/R16

Frequency counter range
range 5 MHz/resolution 4
range 10 MHz/resolution 8
range 20 MHz/resolution 16
range 40 MHz/resolution 32

Battery
Off Disch Charge →

Power
Off Discharge to 4V Charge 14 Hours

41 42 01 1F F3 00 00 19.2 8 * 1 →

1.200 2.400 4.800 9.600 19.200 38.400 76.800 153.600

LCD output example on Break command if no bytes were received

Break
AB ... s4& 19.2 8 * 1 →

LCD output example if Break command was taken after reception of five bytes

41 42 01 1F F3 34 A6 AB ... s4& 19.2 8 * 1 →

41 42 01 1F F3 34 A6 AB ... s4& 19.2 8 * 1 →
Logic Probe

The typical hardware solution for a logic probe is shown in Figure 2. Two inverters, for low and high indication, and two monostables, for pulse detection, are commonly used in most low-cost logic probes. This solution will display an unconnected probe tip as high logic level. There are some better versions which can detect a floating input and turn all LEDs off if it is detected. Figure 3 represents the common solution for such functions, where two analog comparators are employed to detect the low, high and floating inputs.

FIGURE 2: TYPICAL LOGIC PROBE SCHEMATIC

Instead of using such approaches, the logic probe function in this instrument is software aided, and the floating input is detected in a dynamic way instead of a static one. The equivalent hardware schematic diagram of this solution is shown in Figure 4 (Pulse detection circuit not shown). The hardware detail which supports the operation of the logic probe used in this unit is represented by Figure 5. The microcontroller polls the input tip and services LEDs L and H. If a transition is detected, LED P is switched ON and the down counter switches it OFF after 80 ms if no additional transition is detected.

This approach has two disadvantages. Logic state latching at a uniform rate may cause visible interference if the frequency of the monitored signal is near the latching rate. This problem is minimized by adding a self-variable extra delay in software, which makes the latching frequency unstable. This makes the range of critical frequencies much wider, but the interference appears as a very short burst of pauses in LED L or H activity, which is completely avoided by adding an extra debouncer of only 250 microseconds. Although unnoticeable, this delay helps prevent LED level instability while monitoring critical frequencies.
Another disadvantage is related to pulse indication on LED P. In the case of a very short pulse, it is likely that the microcontroller, which polls the input, may omit it between two input reads. Instead of simple polling, the internal counter, TMR0, is used here so that instead of testing the logic state of the input, the state of TMR0 is tested. In this way, pulses as short as 10 ns might be detected. In reality, the minimal pulse width is limited by resistor R6 and the input pin RA4 capacitance. The T0SE bit in the OPTION_REG register is properly updated at each pass, so that the first incoming transition will increment TMR0.

The logic probe software support is integrated in the keyboard routine. LEDs L, H and P are active only while the instrument is idle (doing nothing but waiting for some key to be pressed), which is all the time while the unit is ON, except in frequency counter mode, during battery discharging or charging, or if the START command is issued in analyzer or serial receiver mode and the job (300 samples fetched or all bytes received) is not yet finished.

Pin RB2 is the output which generates square-wave pulses. These pulses are fed through R5 to the probe tip. The resistance is high enough not to affect the tested circuit, except if the tested point is the floating input. However, in that case it will probably make the circuit unstable and thus help in locating the floating input. This pulse stream is also used by software to detect the floating probe tip, and in this case to switch all LEDs off. This saves energy in batteries and helps to detect if the probe tip is validly connected to the point under test.

A simplified flow chart for the logic probe is represented by Figure 6. As this subroutine is an integral part of the key scan routine, the key (debouncers are not shown in detail) and time-out testing (which employs a 16-bit counter, “Time-out Counter”) are also provided. “Up Counter” is the free running counter which enables execution of the second part of the subroutine to be performed at each 256th pass. “Down Counter” is the timing base for the LED Pulse. If the state of this counter is greater than zero and the LED Low or LED High is on, the LED Pulse will be turned on. The program exits only if some key is pressed (flag bit STATUS,C denotes which) or when the time-out counter reaches zero.
FIGURE 6:  LOGIC PROBE FLOWCHART

Logic State Analyzer

The commonly used hardware concept for a logic analyzer design is represented in Figure 7. All those functions are realized in software, which is much easier to implement, but results in a loss of sampling speed. The software solution is briefly represented on the flow chart in Figure 8.

In analyzer mode, a sequence of 300 one-bit fetches is performed. Samples are stored in internal RAM (actually, 304 samples are read, but the last 4 are dummy reads). The upper row of the LCD is used to display the samples. As the LCD (Hitachi's LM032L) has no graphic capabilities (it is not possible to address a single dot), this is simulated by eight special user-defined characters (which are stored in the character generator RAM), each for a group of 3-bit samples. This enables a pseudo-graphic mode which, in this case, looks as if all pixels were individually addressed.

The display shows a window of 60 samples. One of five windows is selected by placing the cursor on the group number and advancing it by pressing the left key. While the key is pressed, the lower row displays the numeric pointers, which help by counting the sample number and calculating the timings in the recorded sequence. When the key is released, the normal row 2 is restored.

A uniform clock, for sample rate, is internally generated. It is selectable to 16 steps. The frequency and period are both displayed. The following is a list of available sample rates:

<table>
<thead>
<tr>
<th>Frequency (Hz)</th>
<th>1 MHz</th>
<th>500 kHz</th>
<th>228 kHz</th>
<th>100 kHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time-out counter = 0?</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Time-out counter = 0?</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

The sampling sequence does not start immediately after the command is issued, but after the selected transition (L to H or H to L) is detected. While waiting for the transition to occur, the RB2 output is continuously held in the state which is opposite of the triggering logic level. This enables application on the wired-or logic, even if it is without pull-ups. If this condition never occurs, it is possible to escape by pressing the right key. In this case, message "Break" is displayed in the LCD upper row.
LED P has an additional function while sampling in analyzer mode. It is turned OFF when the start command is issued, then turned ON when sampling or the receiving condition was met, and then OFF again when all samples are fetched. In slower rates, it is noticeable that LED P blinks while sampling. One blinking period is equal to 32 sampling periods.

FIGURE 7: LOGIC ANALYZER SCHEMATIC
FIGURE 8: LOGIC ANALYZER FLOWCHART

All sample rates are generated by software, and the three highest ones use individual subroutines. The sample rate for 1 MHz, which is at the very beginning of the program, has to fetch and memorize a single bit sample by rotating it into the buffer, change the destination address after every 8 samples and exit the loop after 304 samples. All this while keeping uniform timing of 2 and 3 (alternated, which gives an average of 2.5) instruction cycles for one fetch. That could not be realized in a conventional manner, so it has a location-sensitive structure. Upon exit, it jumps to address 4Dh (which is far from the subroutine itself). If you modify anything in this program, take care not to affect this location.

The analyzer may have some unpredictable delays between an external starting event (rising or falling edge) and the first sample. In all cases, this delay may vary from 0 to 4 microseconds, so it may have some significance only in highest sample rates. One of the reasons for this delay is the time which the microcontroller requires for a key test, which enables the manual break if this event never comes. Also, there is some minor jitter at the 1 MHz analyzer sample rate. In the worst case, it might be 300 ns.
Serial Code Receiver

In this mode, a total of 42 bytes is received and displayed in both HEX and ASCII. The acceptable format is:

- 1 Start Bit / 7 or 8 Data Bits / 0 or 1 Parity Bit / 1 or more Stop Bits.

It is possible to connect the probe tip directly to the RS-232 or RS-232C +/- 12V voltage levels, to RS-422 or RS-485, or to +5V logic.

The available baud rates are:

- 1200 (1.2)
- 2400 (2.4)
- 4800 (4.8)
- 9600 (9.6)
- 19200 (19.2)
- 38400 (38.4)
- 57600 (57.6)
- 115200 (115)

7 or 8 data bits may be selected to adjust for the desired data format.

Parity or no parity bit (suffix "p"). This affects only the proper timing for this bit during reception. It is neither tested for validity nor displayed.

Standard RS232C or inverse polarity. If prefix "i" is displayed, then inverse polarity is active (low start bit, inverted data and optional parity bits and high stop bit). This is useful if the serial message must be fetched before the RS-232C TX drivers and after the RX buffers (which are both inverters).

Received bytes are displayed both in HEX and ASCII in 6 groups of 7 bytes each. ASCII representation is with bit 7 cleared, and the non-printable characters (00h-1Fh) are represented as dots. All other codes are standard ASCII.

The string of received serial codes is 42 bytes long. If the string is shorter, the instrument will wait for next start bit, so it may look like it is stuck without any message. In that case, reception may be stopped by pressing the right key. If no bytes were received, the message "Break" will be displayed, but if at least one byte was received, the received sequence will be displayed with all unreceived bytes represented as zeros.

Similar to the analyzer mode, LED P will be turned ON when the first start bit is detected. This helps to detect sequences of less than 42 bytes in length.

No error test is performed during reception.

Figure 9 represents the flowchart for the serial code receiver.
FIGURE 9: SERIAL CODE RECEIVER FLOWCHART

Serial Code Receiver

Clear buffer and point FSR to buffer

Preset bit counter to 7, 8 or 9

Inverse signal mode?

Clear hi-impedance output

Input = low?

Right key pressed?

Input = high?

Turn on LED "pulse"

Loop 0.5 bit periods

Get input in flag C

Rotate INDF through C

Decrement bit counter

Bit counter = 0?

Display "Break"

Display buffer in HEX and ASCII mode

End of buffer?

Set hi-impedance output

Input = high?

Right key pressed?

Input = low?

Advance FSR

Complement INDF

Inverse signal mode?

Rotate left INDF

Parity bit received?

Rotate right INDF

Any byte received?

Display buffer in HEX and ASCII mode

Turn off LED "pulse"

Return

1

2
Frequency Counter

Figure 10 shows the standard structure of the hardware solution for a frequency counter. All this is substituted by software in the PICmicro® microcontroller (MCU) aided by the existing TMR0. All counters are binary, and the counter state is displayed after a 4-byte binary to 8-digit decimal conversion. The display refresh rate is 2 Hz.

The flow chart for the frequency counter is represented in Figure 11. As this is the real-time function, the existing keyboard subroutine might not be used, but separate key and time-out tests are written. The logic probe function is disabled in this mode.

There is no “Start” command here, as this function is active all the time while the instrument is in Frequency Counter mode. There is only one submode, range select, so pressing the right button is not used for stepping through submodes, but it changes the range immediately.

Internal counter TMR0 is used, and the program expands the width of the counter for an additional two bytes. The fourth byte is added after 500 ms of counting and multiplying the 24-bit counter state by a constant, which depends on which prescaler factor was used.

The prescaler also affects the counter resolution. Here are the counter ranges and corresponding resolutions:

- Range 5 MHz / Resolution 4
- Range 10 MHz / Resolution 8
- Range 20 MHz / Resolution 16
- Range 40 MHz / Resolution 32

The resolution surely affects the reading error of the frequency counter, but this error is still less than the error which is caused by the initial non-accuracy of industrial class quartz crystals.

FIGURE 10: FREQUENCY COUNTER SCHEMATIC

FIGURE 11: FREQUENCY COUNTER FLOWCHART
Battery Manager

The battery manager has three submodes. The first one is manual power-off, although there is also the automatic power off after approximately 8 minutes of inactivity (no key pressed).

The second submode is discharging. It is performed with 100 mA current through the resistors. The voltage monitor informs the PICmicro MCU if battery voltage is lower than 4V. If it is, the mode is automatically switched to charging. It is recommended that an external DC power supply be connected before the discharging command is issued. This will decrease the resulting discharging current to about 80 mA when the instrument is ON, and the DC supply is connected as the charging current flows independently of the mode selected.

Charge submode, when started, displays the time in HH:MM format, starting from 00:00, and switches off the instrument (and charging current also) at 14:00.

It is also possible to charge the NiCd battery even if it is not discharged, but this is not recommended, as unintentional overcharging may affect its capacity and life.

The unit is ready to charge the battery all the time if it is switched ON, even if the command charge is not active. It is enough to connect the external DC supply and to turn the instrument ON.

If the LCD were not counted, more than half of the hardware is used for discharging and charging. Figure 12 explains the structure of the battery manager hardware in a simplified form, where transistors T1, T4 and T5 are replaced by switches, for clarity. The flow chart for the battery discharger and charger is shown in Figure 13.

![Figure 12: Battery Management Schematic](image1)

![Figure 13: Battery Manager Flowchart](image2)
HARDWARE OVERVIEW

The complete schematic diagram is shown in Figure 14. The only IC used is the PIC16F84A-10/P, running at 10 MHz. It controls the intelligent LCD module via pins RB4-RB7, using two additional ports (RB0 and RB1) for the Enable signal and Register Select (Control/Data).

FIGURE 14: PIC16F84A PROBE SCHEMATIC

LCD module LM032L may be controlled in 4-bit parallel mode, which is used in this project. That is why pins 7-10 of the LCD are not used.

LEDs L, H and P share the same outputs with the LCD data bus. The only consequence is that some very minor and short LED flashing is visible while the PICmicro MCU is accessing the LCD (mostly when some key is pressed). RB0 also has two functions. It controls the Enable signal for the LCD module and the transistor T1, which is used for the discharging function. To eliminate current spikes through T1 while accessing the LCD, capacitor C4 is added, which disables activating T1 by short Enable pulses. The main discharging resistors (R20 and R21) are intentionally located at two distant places on the PCB to minimize heat dissipation in a small area.

Inputs RA0 and RA4 are used to read the same signal, which was necessary because RA4 is the only port which may be directed to the TMR0 counter, and RA0 had to be the “edge bit” in the port. The analyzer fetch routine in the highest rate must rotate PORTA bits in a single instruction and perform bit transfers from the input pin to the STATUS, C flag.

Resistors R3 and R6 are used for signal voltage limitation, with internal reverse diodes in the PIC16F84A. This enables connecting the probe tip directly to the RS-232C connector, which has +/- 12V voltage range. Resistor R8 disables input floating while the unit is OFF.

The voltage monitor (transistor T2 with ZD1) acts as a comparator which holds input port RA1 low if the battery voltage is higher than 4V. This is used to automatically detect the end of the discharging process. The value of resistor R13 can be modified to fine trim this cut-off voltage to 4V.

Transistor T5 is the main ON-OFF switch. RB3 controls this transistor through T3. It is also activated by pressing the left or right key. Diodes D3 and D4 disable false activation of T5 through internal diodes in the PICmicro MCU when it is powered OFF.
Transistor T4 is the constant current regulator, which enables the use of any DC supply between 10V and 30V for battery charging. Diodes D9-D12 allow application of any voltage polarity. Zener diode ZD2 is not the voltage stabilizer, but it protects the hardware from overvoltage if the battery contacts are not properly tied while charging.

S3 is the RESET key, which is mounted on the solder side of the PCB, and is accessible from the lower side of the instrument through the small hole at the bottom plane of the package. It is used if the MCU drops into a deadlock state for some reason or when the unit is switched ON for the first time after assembly.

Pin 3 (Vo) on the LCD connector is for LCD driving voltage. The manufacturer recommends the use of a potentiometer (10-20k) for voltage adjustment on this input and to fine trim the LCD contrast, but in all cases the contrast was optimal when the potentiometer was in its lower-most position (Vo shortened to GND). So it was rejected and pin 3 was connected to GND in the final version of this instrument.

Charging current will flow all the time while DC supply is connected and the unit is ON, even if the unit is not in Charge mode. When the unit is OFF (e.g. when the charging 14-hour process is finished), the charging current is stopped.

Both charging and discharging are indicated on individual LEDs.

Note: The LCD module used is Hitachi's LM032L. Type LM032LT may also be used, but it is not recommended, as it is a transflective type and contains the integrated illumination (which may not be used in this case, as it requires high voltage). Do not use modules LM032H or LM032HT as they require a dual voltage supply (+5/-5V).

FIRST SWITCHING ON AFTER ASSEMBLING

The batteries should be connected last, as there is no easy way to disconnect them once they are soldered, also it is not recommended to assemble the hardware while the voltage is present. The best way is to test the instrument with some external 5V power supply, and when it is completely debugged and tested, the batteries may be soldered. Do not connect the external DC supply for charging batteries if the batteries are not safely in their places! Zener diode ZD2 will reduce the voltage to 6.8V, but avoid testing the efficiency of this protection if at all avoidable.

If the NiCd batteries are discharged to the point the PICmicro cannot operate, it will be necessary to keep the left or right key pressed for some time (while the DC supply is connected for charging), as the pressing of any key makes hardware bypass for charging current. After about one minute of such charging, the battery voltage will be sufficient and then the unit will probably need to be reset by pressing S3. The normal charging process should then be used by executing the Charge command in Battery mode.

The contrast on the LCD is voltage-dependent, and as there is no voltage stabilizer, it appears to be a little darker immediately after a full charge. The battery voltage will be slightly over 5V. This will not affect readability. After a few minutes of operation, the battery voltage stabilizes and the LCD appears as normal.

Note: Data EEPROM is used for some lookup tables. This is read-only data and the Data EEPROM must be programmed before the unit is ready to use. The MCU will not affect data EEPROM contents. If your programmer does not support automatic loading of Data EEPROM contents from the HEX file, it must be loaded manually (a total of 61 bytes are used, and the last three bytes are don't care). The following will help in that case (all values are hexadecimal):

| addr 00-07h: | 88 01 01 98 F4 02 8C E4 |
| addr 08-0Fh: | 2C 88 64 0A 88 32 14 D8 |
| addr 10-17h: | 80 1A 89 28 C8 C0 34 |
| addr 18-1Fh: | 88 0A 64 C8 60 68 C8 30 |
| addr 20-27h: | D0 C9 18 A1 80 01 01 14 |
| addr 28-2Fh: | 90 19 00 64 0A 00 28 19 |
| addr 30-37h: | 01 06 09 10 16 2E 30 64 |
| addr 38-3Ch: | CC 05 0E 3B 95 |

MECHANICAL CONSTRUCTION

The components layout is shown in Figure 16. All components are placed on the component side of the PCB, except key S3 (reset), which is on the solder side. So are the NiCd batteries, which are placed in the specially shaped PCB edges and soldered directly to the PCB.

The LCD module is placed on M3 spacers, 7 mm long, which are tightened to 5 mm long spacers at the bottom side of the PCB. This leaves enough room for batteries which are 10 mm in diameter. Key 3 should not be higher than 5 mm, and the recommended height for keys 1 and 2 is about 16 mm. As the keys listed in the parts list are 14.5 mm high, they should be mounted on an extra spacer about 1.5 mm thick, non-conductive material.

The probe tip is fixed using three wire loops soldered to the PCB and to the tip. If it is not possible to get a connector for the DC supply which fits to the PCB pads, it is also possible to cut the PCB (across the dotted line on the components layout) to make enough space for some other type of connector, which may be tightened to the package. Pads for wires, needed in this case, are provided on the PCB. The polarity is not significant.

It is possible to build the package of the same material which is used for printed circuit boards, as it can easily be cut and joints soldered. Figure 15 shows the package detail.
FIGURE 15: CASE/PCB CONSTRUCTION

Upper part of case

Battery

Lower part of case

soldered

FIGURE 16: PCB PARTS LAYOUT
PARTS LIST

1. LCD module type LM032L (Hitachi) ................................................................................................................................. 1
2. PCB.................................................................................................................................................................................... 1
3. Microcontroller PIC16F84A-10/P (Microchip).......................................................................................................................... 1
4. Transistors:
   BC338 (or any small signal silicon NPN in SOT-54, pinning CBE)3
   BC328 (or any small signal silicon PNP in SOT-54, pinning CBE)2
5. Diodes:
   1N4148 (or any small signal silicon diode)12
   ZPD 3V3 (or any low power 3.3V zener diode)1
   ZPD 6V8 (or any low power 6.8V zener diode)1
6. Resistors:
   62R  1/4W axial1
   120R 1/4W axial 2
   330R 1/4W axial 4
   1K5 1/4W axial3
   2K7 1/4W axial3
   5K6  1/4W axial1
   15K  1/4W axial5
   47K  1/4W axial2
   220K 1/4W axial1
7. Capacitors:
   27 pF ceramic2
   100 nF ceramic1
   1 uF tantal1
8. Quartz:
   10 MHz 1
9. LEDs:
   red, 3 mm diameter 2
   green, 3 mm diameter2
   yellow, 3 mm diameter1
10. I.C. socket:
    18-pin 1
11. Keys:
    typ ITT D 6 (raster 5*5 mm, 14.5 mm high) 2
    typ SEL ET 5 (raster 5.5*3) or SEL ET 11 (raster 7.5*5)1
12. Connectors:
    2*7 pins male connector for PCB, raster 2.54 mm
    (100 mils)1
    2*7 pins female connector for PCB, raster 2.54 mm
    (100 mils)1
    cable-end crocodil-grip for GND connection1
    coaxial 3.5 mm female connector typ PG2031
13. Mechanical parts:
    spacer M3, 7 mm high 4
    spacer M3, 5 mm high 4
This is the program for multi-purpose laboratory instrument which consists of logic probe, single-channel logic state analyzer, serial code receiver and frequency counter. As this is single-chip instrument, all functions are supported by software. LCD module used is Hitachi's LM032L with 2 lines of 20 columns.

Note: The code is optimized for code space, and for that reason the most of code could not be written in modular format. For the same reason a lot of subroutines have more than one entry point and some of them are terminated by GOTO instead of RETURN.

I/O port usage (all PORTA bits are inputs, all PORTB bits outputs):

Probe input
Voltage monitor (high if battery voltage < 4 V)
Left key (key 1) (low = key pressed)
Right key (key 2) (low = key pressed)
Probe input
Enable LCD (high=select LCD), discharge (high=on)
Register select LCD (low=instruction, high=data)
Hi-imp output for probe pin
Current hold for MCU supply (low = off)
LCD module D5, LED P (high = on)
LCD module D6, LED H (high = on)
LCD module D7, LED L (high = on)
External Clock Frequency: 10 MHz
Config Bit Settings: CP=OFF, PWRT=ON, WDT=OFF, Oscillator
Program Memory Usage: 1023 words
Data RAM Usage: 68 bytes
Data EEPROM Usage: 61 bytes
Note: This is read-only data, so the Data EEPROM must be programmed before the unit is used. MCU will not affect data EEPROM contents.

Program Memory Usage:

list p=16f84, f=inhx8m, n=0
include "p16f84.inc"
LIST
P16F84.INC Standard Header File, V 2.00 Microchip Technology, Inc.
LIST
LIST
FLAG equ 0ch ; 1 by flag register
RXBITS equ 0dh ; 1 by bit0=parity, bit1=7/8 bits, bit 2=inverse
DJNZ equ 0eh ; 1 by general purpose, e.g. loop counter
SCRATCH equ 0fh ; 1 by general purpose scratchpad
PCOUNT equ 10h ; 1 by timing count for led P (monostable sim)
SUBMODE equ 11h ; 1 by submode (cursor horizontal position)
DEBO1 equ 12h ; 1 by rotor for key 1 debouncing
00000013 00058 DEBO2  equ 13h ; 1 by rotor for key 2 debouncing
00000014 00059 COUNT  equ 14h ; 1 by general purpose counter
00000015 00060 RATE  equ 15h ; 1 by analyzer sample rate, 0...15
00000016 00061 CHARCOU equ 16h ; 1 by char counter for fixed format display
00000017 00062 SHOWCOU equ 17h ; 1 by 1-4, which group of 60 samples is shown
00000018 00063 DELAYH  equ 18h ; 1 by delay for led H on when led L is on
00000019 00064 DELAYL  equ 19h ; 1 by delay for led P on when led H is on
0000001A 00065 PRESC   equ 1ah ; 1 by prescaler rate for frequency counter
0000001B 00066 TIMOUTL equ 1bh ; 1 by timeout counter lo, for auto power off
0000001C 00067 TIMOUTH equ 1ch ; 1 by timeout counter hi, for auto power off
0000001D 00068 RXRATE equ 1dh ; 1 by rx baud rate, 0...7
0000001E 00069 BIN4   equ 1eh ; 4 by arith buf bin value, lo byte first
00000022 00070 CMP4   equ 22h ; 4 by arith buf for comparing, lo byte first
00000026 00071 BUFFER equ 26h ; 42 by 42 by receive buf for analyzer and RX
00000001 00072 REL    equ 1       ; =1 to put cursor on 1st char of command
00000074 00074         ; =0 to put cursor before the command
00000001 00075 ; Bits definitions for FLAG register (bit 0 not used):
00000001 00076 DP     equ 1       ; decimal point in 3-digit bin2dec conv
00000001 00077 PTIP    equ 2       ; prev.state of probe input (for edge detect)
00000001 00078 RIPPLE  equ 3       ; zero blanking bit
00000001 00079 XTOX    equ 4       ; analyzer start at: 1=rising, 0=falling edge
00000001 00080 LEDP    equ 5       ; led Pulse, 1=on
00000001 00081 LEDH    equ 6       ; led High, 1=on
00000001 00082 LEDL    equ 7       ; led Low, 1=on
00000001 00083
00000001 00084 ;*********************************************************************
00000001 00085 ;* Reset vector
00000001 00086 ;*********************************************************************
00000001 00087         org     1               ; this subroutine must start at addr 1
00000001 00088 Get1MHz                         ; 2.5 t read cycle
00000001 00089 ;*********************************************************************
00000001 00090 ;* Get1MHz
00000001 00091 ;* This subroutine fetches 307 samples (last 7 will be ignored) from
00000001 00092 ;* PORTA.0 rotating through CARRY at 1 MHz rate - 2.5 instr. cycles
00000001 00093 ;* for each sample, realized mostly as 2 and 3 cycles alternatively,
00000001 00094 ;* at the following order:
00000001 00095 ;* 4t-2t-2t (not in main loop, executed only once), and then
00000001 00096 ;* 2t-2t-3t-2t-3t-2t-3t-2t-3t-2t-3t-2t-3t-2t-3t-2t-3t-2t-3t-2t-3t-2t-3t-2t-3t-2t-3t-2t-2t-4t  (repeat 19 times)
00000001 00097 ;* Call Common inits loop counter (COUNT) to make 19 cycles before
00000001 00098 ;* exiting (16 samples are fetched at each pass), and FSR to point to
00000001 00099 ;* Call Common inits loop counter (COUNT) to make 19 cycles before
00000001 00100 ;* exiting (16 samples are fetched at each pass), and FSR to point to
00000001 00101 ;* BUFF. It also presets TOSE bit depending on XTOX bit (in FLAG reg)
00000001 00102 ;* to enable proper edge detect, as it will affect TMRO state.
00000001 00103 ;* State of key 2 (Break) is tested while waiting for start condition.
00000001 00104 ;* Write ptr FSR is incremented after every 8 samples. COUNT initial
00000001 00105 ;* value is 01011101, after ANDing 0c0h and subtracting 33h from it,
00000001 00106 ;* makes 0dh, even if COUNT is incremented 18 times. After 19 passes,
00000001 00107 ;* COUNT is incremented to b'10000000', which after AND 0c0h and
00000001 00108 ;* SUB 33h makes 4dh. Those jumps are location sensitive, and it makes
00000001 00109 ;* the whole subroutine unrelocateable.
00000001 00110 ;* Between this subroutine and the instruction goto Finished (below),
00000001 00111 ;* which must be at loc. 4dh, there are 25 free locations. They are
00000001 00112 ;* used for tables DecTab and CurTab, which causes that those tables
00000001 00113 ;* must have the fixed length. If anything relocates here, take care
00000001 00114 ;* not to affect location of instruction goto Finished.
00000001 00115 ;* Input/Output variables: None
00000001 00116 ;*********************************************************************
00000001 00117         org     1               ; this subroutine must start at addr 1
00000001 00118
00000001 00119 Get1MHz                         ; 2.5 t read cycle
00000001 00120 movlw  80h-.19         ; loop end in 19 cyc(38 by=304 smpls)
00000001 00121 call    Common ; initialize COUNT, FSR, hi-imp out...
00000001 00122 ; ...bit XTOX and TOSE bit
00000001 00123 GetEdge

00000001 00124

00000001 00125
btfss PORTA,3 ; test status of key 2 and...
goto Break ; ...jump to Break routine if pressed
movf TMRO,W ; TMRO = logic level edge detector
rff PORTA,F ; ... the first sample is a little earlier
... to compensate starting delay
btfsc STATUS,Z ; test if there was egde...
goto GetEdge ; ...and loop if not
rff INDF,F ; rotate bit into destination byte
rff PORTA,F ; get bit from input to C
rff INDF,F ; rotate bit into destination byte
rff PORTA,F ; get bit from input to C
movwf PCL will jump here at 18 passes
@addr 0100 0000 (40h) - 33h = 0dh
rff INDF,F ; rotate bit into destination byte
incf COUNT,F ; COUNT = loop counter
rff PORTA,F ; get bit from input to C
rff INDF,F ; rotate bit into destination byte
rff PORTA,F ; get bit from input to C
andlw 0c0h ; this will make first 18 jumps to 0dh,
addlw -33h ; this will make first 18 jumps to 0dh,
andlw 0c0h ; this will make first 18 jumps to 0dh,
addlw -33h ; this will make first 18 jumps to 0dh,
andlw 0c0h ; this will make first 18 jumps to 0dh,
AN689

0033 0082 00190 movwf PCL ; jumps to 0dh in first 18 passes
00191 ; jumps to 4dh at 19th pass
00192
00193 ;*********************************************************************
00194 ;* This table is used for bin2ascii (4-byte to 8-digit) conversion
00195 ;*********************************************************************
00196 DecTab
00197 0034 3498 3496 3480 00197 dt 098h,096h,080h ; decimal 10 000 000
00198 0037 340F 3442 3440 00198 dt 00fh,042h,040h ; decimal 1 000 000
00199 003A 3401 3486 34A0 00199 dt 001h,086h,0a0h ; decimal 100 000
00200 0040 3400 3427 3410 00200 dt 000h,027h,010h ; decimal 10 000
00201 0043 3400 3403 34E8 00201 dt 000h,003h,0e8h ; decimal 1 000
00202 0046 3400 3400 340A 00202 dt 000h,000h,00ah ; decimal 10
00203 00204 ;*********************************************************************
00205 ;* Cursor position table for all SUBMODEs in mode 4 (battery manager)
00206 ;*********************************************************************
00207 CurTab4
00208 0049 34D3 34C0 34c4 00208 dt 0d2h+REL,0c0h,0c3h+REL,0cah+REL
00209 004E 3400 3401 3404 00209 dt 0d2h+REL,0c0h,0c3h+REL,0c0h+REL
34CB
00210 ;*********************************************************************
00211 ;* This is exit point for subroutine Get1MHz. Do not move this
00212 ;* instruction, it must be at address 4dh!
00213 ;*********************************************************************
00214 ;*********************************************************************
00215 org 80h-33h ; addr 1000 0000 (80h) - 33h = 4dh
00216 ; Get1MHz jumps here
00217 ;*********************************************************************
00218 Graphs
00219 004E 3400 3401 3404 00219 dt 00h, 01h, 04h, 05h, 10h, 11h, 14h, 15h
00220 004D 2A6A 00220 goto Finished
00221 ;*********************************************************************
00222 ;*********************************************************************
00223 PrescTab
00224 0060 004D 00224 dt 0d2h+REL,0c0h,0c0h+REL,0c0h+REL
00225 0060 0033 00225 ; 0d2h+REL,0c0h,0c0h+REL,0c0h+REL
00226 0060 0034 00226 ; 0d2h+REL,0c0h,0c0h+REL,0c0h+REL
00227 00228 ;*********************************************************************
00229 ;*********************************************************************
00230 ;*********************************************************************
00231 ;*********************************************************************
00232 ;*********************************************************************
00233 ;*********************************************************************
00234 ;*********************************************************************
00235 ;*********************************************************************
00236 ;*********************************************************************
00237 ;*********************************************************************
00238 ;*********************************************************************
00239 ;*********************************************************************
00240 ;*********************************************************************
00241 ;*********************************************************************
00242 ;*********************************************************************
00243 ;*********************************************************************
00244 ;*********************************************************************
00245 ;*********************************************************************
00246 ;*********************************************************************
00247 ;* Range table for max. frequency display in mode 3 (freq counter)
00248 ;********************************************************************
00249 RangeTab
00250 dt .5, .10, .20, .40
00251 ;********************************************************************
00252 ;* Timing constants for serial code receiver
00253 ;********************************************************************
00254 BaudRate
00255 dt .188, .94, .46, .23, .11, .5, .3, .1
00256 ;********************************************************************
00257 ;* Text strings (terminator = last character with bit 7 set)
00258 ;********************************************************************
00259 TxtHz
00260 dt "MHz","/'+80h
00261 DispTxt
00262 dt "Off Disch. Charg","e'+80h
00263 Head1
00264 dt "Analyze","r'+80h
00265 Head2
00266 dt "Serial","l'+80h
00267 Head3
00268 dt "Frequency","y'+80h
00269 Head4
00270 dt "Battery","y'+80h
00271 BrkMes
00272 dt "Break","k'+80h
00273 ;******* START
00274 ;********************************************************************
00275 ;* Power Up sequence:I/O port B defined as all outputs,PORTB,3 set to
00276 ;*  switch power supply on and the internal Data Ram is cleared
00277 ;********************************************************************
00278 ;********************************************************************
00279 ;********************************************************************
00280 Start
00281 bsf STATUS,RP0
00282 clrf TRISB ; portb: all bits outputs, port a: inputs
00283 bcf STATUS,RP0
00284 movlw Och ; start of RAM clr, & PORTB output byte
00285 movwf PORTB ; switch power supply ON (set PORTB,3)
00286 call ClrRam ; clear internal RAM and wait 33.8 ms
00287 ;********************************************************************
00288 ;********************************************************************
00289 ;* LCD module initialization. 4-bit mode selected, display data RAM
00290 ;* cleared cursor set to blink mode, and the pseudographics character
00291 ;* for character set 00h-07h preset from table Graphs.
00292 ;********************************************************************
00293 bcf PORTB,1 ; rs lo (instruction)
00294 movlw 2 ; 4-bit mode
00295 call Nibble ; write 4-bit mode command
00B1 3028 00296  movlw  28h ; func set: 4 bit mode,2 lines,5*7 dots
00B2 23CF 00297  call  WrComL ; write command and wait 130 us
00B3 3006 00298  movlw  06h ; modeset: cursor moves right, no shift
00B4 23CF 00299  call  WrComL ; write command and wait 130 us
00B5 300D 00300  movlw  0dh ; disp on, no cursor,blink cursor pos
00B6 23CF 00301  call  WrComL ; write command and wait 130 us
00B7 3006 00302  movlw  06h ; modeset: cursor moves right, no shift
00B8 23CF 00303  call  WrComL ; write command and wait 130 us
00B9 300D 00304  movlw  0dh ; disp on, no cursor, blink cursor pos
00BC 008F 00305  movwf SCRATCH ; move start address to pointer
00BD 28BF 00306  goto Count ; 8 special characters defined
00BE 28D3 00307  goto FiveRows ; 5 rows are equal
00BF 0A8F 00308  incf SCRATCH,F ; inc ptr
00C0 3015 00309  movlw  15h ; 15h=b'10101'=dot-space-dot-space-dot
00C1 0030A 00310  movlw 40h ; cg ram addr 0
00C2 304E 00311  movlw Graphs ; start addr of graph set for lcd disp
00C3 3005 00312  movlw .5 ; five rows are equal
00C4 3006 00313  movlw .8*.4 ; 8 special characters to define
00C5 00314 00315  movlw .5 ; five rows are equal
00C6 0096 00316  movwf CHARCOU ; loop counter for 8 special characters
00C7 00317 00318  goto GoGraph ; no, loop
00C8 00319 00320  movlw 15h ; 15h=b'10101'=dot-space-dot-space-dot
00C9 00321 00322  call  CharBl ; row 6:all dots set,row 7:all dots clr
00CA 00323 00324  decfsz COUNT,F ; five passes over?
00CB 00325 00326  goto GoGraph ; no, loop
00CC 00327 00328  movlw Head1-1 ; start address of string -1
00CD 00329 00330  movwf COUNT ; counter for 5 rows
00CE 00331 00332  call  Headline ; print "Analyzer"
00C9 00333 00334  goto Farm1 ; avoid "Break" message
00C9 00335 00336  movlw Head1-1 ; start address of string -1
00CA 00337 00338  call  Headline ; print "Analyzer"
00CB 00339 00340  goto Farm1 ; avoid "Break" message
00CC 00341 00342  call  PrintM1 ; print string in line 2
00CC 00343 00344  movlw CurTab1 ; get cursor table addr in analyze mode
00CC 00345 00346  call  CurPosKb ; place cursor on proper position
00CF 00347 00348  decfsz CHARCOU,F ; 8 characters defined?
00CF 00349 00350  btfsc STATUS,C ; test which key was pressed
00D0 00351 00352  goto Key1A ; jump if key 1
00D1 00353 00354  goto Key1A ; jump if key 1
00D2 00355 00356  continue if key 2 pressed
00D3 00357 00358  goto Farm1B ; go wait next key
00D4 00359 00360  Key1A ; key 1 pressed
00D5 00361 00362  goto Farm1B ; go wait next key
00362 ; continue if SUBMODE=1
00363 incf RATE,F ; advance sample rate
00364 bcf RATE,4 ; RATE range 0...15
00365
00366 call ClrRow1 ; prepare line 1 to print sample rate #
00367 incf RATE,W ; readjust RATE from 0...15 to 1...16
00368 call Print255 ; print serial # of sample rate 1...16
00369 goto Farm1 ; go redraw row 2, wait next command
00370
00371 EdgeSet ; Change start cond(L-2-H or H-2-L)
00372 movlw 10h ; bit 4 is flag XTOX
00373 xorwf FLAG,F ; change flag
00374 goto Farm1 ; go redraw row 2, wait next command
00375
00376 btfsc SUBMODE,0 ; bit 0 will be set only if SUBMODE=3
00377 goto Mode1Go ; if SUBMODE=3
00378 btfsc SUBMODE,1 ; bit 1 will be set only if SUBMODE=2
00379 goto EdgeSet ; if SUBMODE=2
00380 btfsc SUBMODE,2 ; bit 2 will be set only if SUBMODE=4
00381 goto Mode1Show ; if SUBMODE=4
00382 ; if SUBMODE=0, program drops to Mode2.
00383
00384 ;********************************************************************
00385 ;* This is home for mode 2 (RS232 receiver): prints text "Serial" and
00386 ;* command line in line 2, cursor placed depended on variable SUBMODE.
00387 ;* BrkRS entry point prints message Break in line 1 and redraws line 2
00388 ;* if 0 bytes are received, display first 7 bytes if any byte received
00389 ;*********************************************************************
00390 Mode2 ; mode 2: Serial receiver
00391 movlw 0c8h ; baud rate position on LCD
00392 call Headline ; print "Serial"
00393 movf FSR,W ; FSR points to write next rcvd byte
00394 xorlw BUFFER ; if FSR=literal BUFF the 0 bytes rcvd
00395 btfss STATUS,Z ; test if FSR = literal BUFFER
00396 goto Show2 ; no -some bytes received,show them
00397 call PrintBrk ; yes -no bytes received, print "Break"
00398 ;*********************************************************************
00399 ;* Prints baud rate in KBaud on LCD
00400 ;* Input variables: RXRATE in range 0...7
00401 ;* Output variables: CHARCOU decremented by num of characters printed
00402 ;*********************************************************************
00403 Farm2
00404 movlw 0c8h ; baud rate position on LCD
00405 call WrComL ; move cursor command
00406
00407 moving 0.15 ; case RXRATE=7:then Baudrate=115 Kbaud
00408 incf RXRATE,F ; move RXRATE from range 0...7 to 1...8
00409 bcf RATE,DP ; no decimal point printing if RATE=0
00410
00411 bcf FLAG,DP ; no decimal point printing if RATE=0
00412 clrf BIN4+1 ; BIN4+1 is high byte for baudrate disp
00413
00414 incf RXRATE,W ; move RXRATE from range 0...7 to 1...8
00415 movwf DJNZ ; DJNZ = RXRATE+1
00416
00417 movlw 0.15 ; case RXRATE=7:then Baudrate=115 Kbaud
00418 incf DJNZ,3 ; test if DJNZ=8 (same as RXRATE=7)
00419 goto Lth256 ; yes, go case 115.2 (RXRATE=7)
00420
00421 bcf FLAG,DP ; for rete 0...6 there is decimal point
00422 bcf BIN4+1,1 ; case RXRATE=6:is hi byte for 57.6
00423 movlw .576-.512 ; case RXRATE=6:is lo byte for 57.6
00424 incf DJNZ,F ; DJNZ=RXRATE+2
00425 bcf DJNZ,3 ; test if DJNZ=8 (same as RXRATE=6)
00426 goto Lth256 ; yes, go case 57.6 (RXRATE=6)
00FC 019F 00428  clrf BIN4+1 ; for rates 0...5 hi byte is zero
00FD 3003 00429  movlw .3 ; constant for rates 0...5
00FE 009E 00430  movwf BIN4 ; will be rotated (mult by 2)
00431  ; RXRATE+2 times to get 1.2 - 2.4 - 4.8
00432  ; - 9.6 - 19.2 - 38.4
00FF 00433  X2Loop
00FF 1003 00434  bcf STATUS,C ; clear bit C to get multiplying by 2
0100 0D9E 00435  rlf BIN4,F ; multiply low byte
0101 0D9F 00436  rlf BIN4+1,F ; multiply hibyte, that is 16-bit rotate
0102 0B8E 00437  decfsz DJNZ,F ; test if RXRATE+2 times multiplied
0103 28FF 00438  goto X2Loop ; no, loop
0104 081E 00439  movf BIN4,W ; yes, get result to print it
0105 00440  Lth256
0105 2372 00441  call PrintBR ; print baud rate, incl. decimal point
0106 23D7 00442  call Blank ; to delete last # from previous rate
00443
00444  ;********************************************************************
00445  ;* Prints num bits to be received (7 or 8), with suffix "p" if parity
00446  ;* bit will be received (not written to RAM!), and with prefix "i" if
00447  ;* inverse input polarity is expected. Input variable RXBITS, bit0 set
00448  ;* if parity bit expected, bit 1 set if 8-bit word and bit 2 set if
00449  ;* inverse polarity (lo start bit, inverse data bits and high stop bit)
00450  ;********************************************************************
0107 30CC 00451  movlw 0cch ; bit# pos (7/8/7p/8p/i7/i8/i7p/i8p) -1
0108 23CF 00452  call WrComL ; write command
0109 3020 00453
010A 190D 00454  movlw ' ' ; space: true polarity
010B 3069 00455  btfsc RXBITS,2 ; let it be space if RXBITS,2 cleared
010C 3037 00456  movlw '7' ; "7": 7 bits
010D 3038 00457  movlw '8' ; "8": 8 bits
010E 3020 00458  movlw ' ' ; space: no parity
010F 3066 00459  btfsc RXBITS,0 ; let it be space if RXBITS,0 cleared
0110 3066 00460  movlw 'p' ; "p": parity bit exists
0111 3020 00461  movlw ' ' ; space: true polarity
0112 3069 00462  btfsc RXBITS,2 ; let it be space if RXBITS,2 cleared
0113 3037 00463  movlw '7' ; "7": 7 bits
0114 3038 00464  movlw '8' ; "8": 8 bits
0115 22A5 00465  call KaoAna ; print rest of line - is the same as
0116 3047 00466  call KaoAna ; on mode 1 (analyzer)
0117 2170 00467  call CurPosKb ; place cursor on proper pos
0118 00468  ; test keys / probe input, service leds
0119 00469  ; return if key press (C: key1, NC: key2)
011A 305B 00470  movlw CurTab2 ; table with cursor positions
011B 2170 00471  call CurPosKb ; place cursor on proper pos
011C 00472  ; test keys / probe input, service leds
011D 00473  ; return if key press (C: key1, NC: key2)
011E 00474  ;********************************************************************
011F 3070 00475  call KaoAna ; print rest of line - is the same as
0120 30CC 00476  call KaoAna ; on mode 1 (analyzer)
0121 00477  ;********************************************************************
0122 22A5 00478  call KaoAna ; print rest of line - is the same as
0123 2170 00479  call CurPosKb ; place cursor on proper pos
0124 00480  ; test keys / probe input, service leds
0125 00481  ; return if key press (C: key1, NC: key2)
0126 00482  ;********************************************************************
0127 30CC 00483  movlw CurTab2 ; table with cursor positions
0128 2170 00484  call CurPosKb ; place cursor on proper pos
0129 00485  ; test keys / probe input, service leds
012A 00486  ; return if key press (C: key1, NC: key2)
012B 00487  ;********************************************************************
012C 1803 00488  btfsc STATUS,C ; test which key was pressed
012D 291C 00489  goto Key1B ; jump if C set, means key 1 pressed
012E 00490  ; key 2 pressed
012F 00491  ;********************************************************************
0130 212A 00492  call Range5 ; increment SUBMODE in range 0...4
0131 28ED 00493  goto Farm2 ; go redraw row2, wait for next command
011C 00494    Key1B ; key 1 pressed
011C 1911    btfs SUBMODE,2 ; bit 2 is set only if SUBMODE = 4
011D 00496    goto Mode2Show ; jump if SUBMODE = 4
00497
011E 1C91    btfs SUBMODE,1 ; bit1 cleared only if SUBMODE = 0 or 1
011F 2925    goto Sub01 ; jump if SUBMODE = 0 or SUBMODE = 1
00500
0120 1811    btfs SUBMODE,0 ; bit 0 is set here only if SUBMODE = 3
0121 2AEA    goto Mode2Go ; jump if SUBMODE = 3
00502
0122 0A8D    incf RXBITS,F ; advance RXBITS (command)
0123 118D    bcf RXBITS,3 ; RXBITS cycle in range = 0...7
0124 28ED    goto Farm2 ; go redraw row2, wait for next command
0125 00500
0125 0A91    incf RXRATE,F ; if SUBMODE = 1 then advance RXRATE
0126 119D    bcf RXRATE,3 ; RXRATE cycle in range 0...7
0127 00504    goto Farm2 ; go redraw row2, wait for next command
0128 00507    Sub01
0128 1C11    btfs SUBMODE,0 ; bit 0 is set here only if SUBMODE = 0
0129 2B28    goto FreqEp ; if SUBMODE =0,goto frequency entry pt
012A 0A8D    incf RXBITS,F ; advance RXBITS (command)
012B 1911    btfsc SUBMODE,2 ; bit 2 is set only if SUBMODE = 4
012C 1C11    btfss SUBMODE,0 ; bit 0 is set here only if SUBMODE = 3
012D 0008    return ; no overflow: return
012E 0191    return ; SUBMODE wrapped to 0, return
00510
012A 00519    Range5 ; increment SUBMODE in range 0...4
012B 0A91    incf SUBMODE,F ; advance SUBMODE
012C 1911    btfs SUBMODE,2 ; if SUBMODE,2 cleared then no overflow
012D 0008    return ; no overflow: return
012E 00511    incf RXRATE,F ; if SUBMODE = 1 then advance RXRATE
012F 00512    bcf RXRATE,3 ; RXRATE cycle in range 0...7
0130 00513    goto Farm2 ; go redraw row2, wait for next command
0131 00514
0132 00515    ;*************************************************************************
0133 00516    ;* This subroutine increments variable SUBMODE,and if the result is >4
0134 00517    ;* it wraps to 0
0135 00518    ;*************************************************************************
0136 00519    Range5 ; increment SUBMODE in range 0...4
0137 00520    incf SUBMODE,F ; advance SUBMODE
0138 1911    btfsc SUBMODE,2 ; bit 2 is set only if SUBMODE = 4
0139 1C11    btfss SUBMODE,0 ; bit 0 is set here only if SUBMODE = 3
0140 2B28    goto FreqEp ; if SUBMODE =0,goto frequency entry pt
0141 00522    return ; no overflow: return
0142 00523    return ; SUBMODE wrapped to 0, return
0143 00524    return ; SUBMODE wrapped to 0, return
0144 00525    ;*************************************************************************
0145 00526    ;* Mode4 is home point for mode 4 (off/discharge/charge): prints text
0146 00527    ;* "Battery" and command line in line2, with cursor placed depended on
0147 00528    ;* variable SUBMODE. Then keyboard routine is called, where it will
0148 00529    ;* wait for key to be pressed
0149 00530    ;* Break4 entry point prints message Break in line1 and readraws line2
014A 00531    ;* ExitDis is the entry point if key 2 is pressed during discharging
014B 00532    ;*************************************************************************
014C 00533    ExitDis ; exit disch entry point,if disch Break
014D 00534    call DisEna30 ; turn off PORTB,0 discharge transistor
014E 00535    call PrintBrk ; print "Break"
014F 00536    call Contm4 ; avoid headline printing
0150 00537    Mode4 ; mode 4: discharge/charge
0151 00538    call Headline2 ; print "Battery"
0152 00539    call Contm4 ; move cursor to line 2
0153 00540    call Row2 ; move cursor to line 2
0154 00541    call CurTab4 ; point to cursor table for mode 4
0155 00542    call Write ; print Off Disch Charge
0156 00543    movlw DisTxt-1 ; point to message -1
0157 00544    call Write ; print Off Disch Charge
0158 00545    call Write ; print Off Disch Charge
0159 00546    movlw CurTab4 ; point to cursor table for mode 4
015A 00547    call CurPosKb ; place cursor @ Off/Disch/Charge/-->
015B 00548    movlw Off/Disch/Charge/--> ; test keys / probe input,service leds
015C 00549    test key press (C:Key1,NC:Key2)
015D 00550    return if key press (C:Key1,NC:Key2)
015E 00551    return if key press (C:Key1,NC:Key2)
015F 00552    ;*************************************************************************
0160 00553    ;* If key1 pressed (C), SUBMODE is advanced (range 0..3,then wrap to 0
0161 00554    ;* If key 2 pressed (NC), program jumps to corresponding routine
0162 00555    ;*************************************************************************
0163 00556    btfs STATUS,C ; test which key was pressed
0164 01803    goto Key1D ; C set: key 1 pressed
0165 01293E    goto Key1D ; C set: key 1 pressed
0166 012958    key 2 pressed
0167 013B 0A91    incf SUBMODE,F ; advance SUBMODE
0168 00559

© 1998 Microchip Technology Inc.
013C 1111 00560  bcf    SUBMODE,2 ; SUBMODE cycle in range 0...3
013D 2937 00561  goto    Farm4 ; go redraw row2, wait for next command
013E 00562 KeyID ; key 1 pressed
013E 1891 00563  btfs    SUBMODE,1 ; bit 1 set only if Chg/Disch. submode
013F 2943 00564  goto    ChargDis ; goto Charge or Discharge process
013F 00565  ; depended on bit 0 in variable SUBMODE
0140 1C11 00566  btfs    SUBMODE,0 ; SUBMODE,0 cleared here if SUBMODE=0
0141 28C9 00567  goto    Mode1 ; shortcut to mode 1 (Analyzer)
00568
0142 29BC 00569  goto    Suicide ; manual power off
00570
00571 ;*******  CHARGE/DISCHARGE
00572 ;********************************************************************
00573 ;* Charge/Disch entry point. If bit SUBMODE,0 set, then go to Charge
00574 ;* SUBMODE,1 is set in this point. (SUBMODE=2 or 3)
00575 ;********************************************************************
0143 00576 ChargDis
0143 22B7 00577  call    ClrRow1 ; in both cases row 1 must be cleared
0144 1811 00578  btfsc    SUBMODE,0 ; test if SUBMODE,0 set, if so...
0145 2952 00579  goto    Charge ; ...jump to Charge (SUBMODE=3)
00580 ; ...else continue to disch (SUBMODE=2)
00581
00582 ;********************************************************************
00583 ;* Discharge starts here (SUBMODE=2)
00584 ;* Cursor moved to line 1 under text "Disch."
00585 ;* Then command 02h (Home Cursor) issued to LCD controller, but this
00586 ;* is dummy command - sense is to freeze it after first nibble, and
00587 ;* thus to leave PORTB,0 (ENA) in high state as long as discharging
00588 ;* lasts. After the discharging termination (if volt monitor detects
00589 ;* <4V or key 2 pressed), the command for LCD controller will be
00590 ;* completed, switching discharging transistor off.
00591 ;* If discharging is broken by key, program returns to user interface
00592 ;* for mode 4, if terminated by voltage monitor, charging takes place
00593 ;********************************************************************
0146 1086 00594  bcf    PORTB,1 ; pull LCD Reg Select low (=instr)
0147 0103 00595  clrw ; high nibble of instruction 02h = 0h
0148 23F1 00596  call    Hinib_B ; output W,4-7 to 4-bit LCD data bus
0149 23ED 00597  call    EnaLCD ; generate En signal (1200us) for LCD
014A 3020 00598  movlw 20h ; command 02h=home cursor(swap nibbles)
014B 23F1 00599  call    Hinib_B ; output W,4-7 to 4-bit LCD data bus
014C 1406 00600  bsf    PORTB,0 ; ENA activated (the command won't be
00601 ; finished until Break or voltage < 4V)
014D 00602 DisLoop
014D 1DB5 00603  btfs    PORTA,3 ; test key 2 status...
014E 2930 00604  goto    ExitDis ; if low,disching manually broke by key
00605
014F 1C85 00606  btfs    PORTA,1 ; test voltage monitor...
0150 294D 00607  goto    DisLoop ; if still >=4V, loop
00608 ; disching terminated (voltage < 4V)
00609
0151 23BD 00610  call    DisEna30 ; switch off discahge transistor
00611
00612 ;********************************************************************
00613 ;* Charging starts here (by command or after successful discharging)
00614 ;* Minute and hour counters are init'd and counting process starts.
00615 ;* Clock(in format HH:MM) is displayed in line 1 under text "Charge".
00616 ;* If charging broken by key, program returns to user interface for
00617 ;* mode 4, if terminated by timeout (14 hours), the unit jumps to
00618 ;* SUICIDE (switches off the unit forcing the output PORTB,3 low).
00619 ;********************************************************************
0152 00620 Charge ; Charge entry point
00621
0152 019B 00622  clrf    TIMOUTL ; initialize minute counter 0...59
0153 019C 00623  clrf    TIMOUTH ; initialize hour counter 0...13
0154 00625 ChLoop
0154 308B 00626  movlw  8bh ; position of digital clock on LCD
0155 23CF 00627  call  WcComL ; cursor to digital clock pos
0156 081C 00628  movf  TIMOUTH,W ; TIMOUTH=hours in binary format
0157 2370 00629  call  Print255 ; print hour in format HH
0158 303A 00630  movlw  ':' ; ':' = separator
0159 21D1 00631  call  PrintTL ; print ':' and minute in format MM
0160 00632  movlw  .228 ; 228 x 263270.4 us = 60 sec
0161 23D2 00633  call  GoLoop ; 1283t (513.2us) inclusive ; 1283t
0162 23D2 00634  call  GoLoop ; total 1026.4us ; 1283t
0163 00635  Min1
0164 0B94 00636  decfsz  COUNT,F ; 1t low byte loop counter
0165 295C 00637  goto  Min1 ; 2t inner pass 1028.4 us
0166 0B8F 00638  decfsz  SCRATCH,F ; high byte loop counter for 1 minute
0167 295C 00639  goto  Min1 ; one pass 263270.4 us
0168 0A9B 00640  incf  TIMOUTL,F ; advance minute counter
0169 081B 00641  movf  TIMOUTH,W ; TIMOUTH = minute up counter
0170 00642  btfss  PORTA,3 ; 2t test status of key 2...
0171 2931 00643  goto  Break4 ; - ...if low,chg manually terminated
0172 00644  decfsz  COUNT,F ; 1t low byte loop counter
0173 295C 00645  goto  Min1 ; 2t inner pass 1028.4 us
0174 00646  goto  Min1 ; one pass 263270.4 us
0175 0A9B 00647  incf  TIMOUTL,F ; advance minute counter
0176 00648  incf  TIMOUTL,F ; advance minute counter
0177 081B 00649  movf  TIMOUTH,W ; TIMOUTH = minute up counter
0178 00650  addlw  -.60 ; test if 60 minutes of charging done..
0179 1C03 00651  btfsz  STATUS,C ; if 60 minutes passed, C should be set
0180 296B 00652  goto  NotHour ; not yet hour advance
0181 019B 00653  clrf  TIMOUTL ; if 60 minutes done, clrf minute cntr
0182 019C 00654  incf  TIMOUTH,F ; ..and advance TIMOUTH=hour up counter
0183 00655  NotHour
0184 00656  movf  TIMOUTH,W ; TIMOUTH = hour up counter
0185 00657  addlw  -.14 ; test if 14 hours of charging
0186 1C03 00658  btfsz  STATUS,C ; if 14 hours passed, C should be set
0187 2954 00659  goto  ChLoop ; ...if not yet 14 hours, loop
0188 00660  goto  Suicide ; charging terminated after 14 h
0189 00661  CurPosKb
0190 00662  CurPosKb
0191 00663  CurPosKb
0192 00664  CurPosKb
0193 00665  CurPosKb
0194 00666  CurPosKb
0195 00667  CurPosKb
0196 00668  CurPosKb
0197 00669  CurPosKb
0198 00670  CurPosKb
0199 00671  CurPosKb
0200 00672  CurPosKb
0201 00673  CurPosKb
0202 00674  CurPosKb
0203 00675  CurPosKb
0204 00676  CurPosKb
0205 00677  CurPosKb
0206 00678  CurPosKb
0207 00679  CurPosKb
0208 00680  CurPosKb
0209 00681  CurPosKb
0210 00682  addwf  SUBMODE,W ; add SUBMODE to lookup table offset
0211 00683  call  PclSub ; get cursor position from table
0212 00684  call  WcComL ; write new cursor position to LCD
0213 00685  clrf  TIMOUTH
0214 00686  CurPosKb
0215 00687  bsf  DEBO1,0 ; set any bit in both debouncers...
0216 00688  bsf  DEBO2,0 ; ...to disable false recognizing of...
0217 00689  low level as falling edge
0218 00690  low level as falling edge
© 1998 Microchip Technology Inc.
GoKbd

Is main keyboard subroutine, in which program loops all the time except in freq counter mode, or while wait for start condition or executing some command. Program exits subrout only if some key is pressed (not if continuously pressed), or when timeout counter (TIMOUTH, TIMOUTL) after 8 min reaches zero. If key 1 was pressed, flag STATUS,C will be reset at exit, if key 2 was pressed, flag STATUS,C will be set. If timeout detected, the routine Suicide is executed (the unit is switched off forcing the output PORTB,3 low).

During keyboard scan, LEDs L, H and P are serviced. Logic state at PORTA,4 affects LEDs L and H directly, and LED P is under control of down counter PCOUNT. This counter is initialized at every logic level transition at PORTA,4, and while counting down, if PCOUNT>0, LED P is switched on.

Loop labeled "Unstable" adds the extra delay which timing is not constant, but changes from 3 to 49t. This mins the interference between the input scan and tested signal frequency.

Counter TMRO is used for detecting of short pulses. At each transition detected, TMRO is cleared, then periodically tested if counter state was incremented. If so, PCOUNT is initialized and LED P turned on.

Register DJNZ is used as a freerunning counter, which divides loop count by 256 and slows down PCOUNT countdown / keys scanning. Keys are debounced and falling edge (pressing moment) detected by rotating registers DEBO1 and DEBO2, and testing them if the key was unpressed at least at 7 passes and then pressed at 1 pass.

Input variables: none

Output variables: Bit STATUS,C reset if key 1 pressed, set if key 2 pressed.

GoKbd

movf DJNZ,W ; 1t to avoid interference, total avg 29t
iorlw 0f0h ; 1t extra time range from -.16 and -1
movwf SCRATCH ; 1t here SCRATCH varies 0f0h to 0ffh
incfsz SCRATCH,F ; 1-17t (avg .9) add extra timing count
goto UnStable ; 2-32t (avg .17) loop loses extra time

movf TMR0,W ; TMR0 = hardware transition detector
btfsc STATUS,Z ; test if transition at PORTA,4...
goto NoIniP ; ...if not, do not affect LED P status
bsf PCOUNT,3 ; initialize PCOUNT for LED P timing
clrf TMR0 ; re-init hardware transition detector
movlw 28h ; bit 4 (TOSE) RESET: L-to-H transition
btfsc PORTA,4 ; if probe tip low, leave TOSE reset...
btfss PORTB,2 ; test hi-impedance output state and...
bsf DELAYL,0 ; init input bit in delay HI rotor
movlw 4 ; value 4 = bit 2 set
xorwf PORTB,F ; change state of PORTB,F (square wave.
goto ContInpX ; jump to skip case 2
0190 00757 InputHi ; entry point for case 2: input high
0190 1906 00758 btfsc PORTB,2 ; test hi-impedance output state and...
0191 1419 00759 bsf DELAYH,0 ; ...turn on led L, PORTB,2 was high
0192 150C 00760 bsf FLAG,PTIP ; set flag to notify that previous...
0193 00761 ; ...state of probe tip was high
0193 138C 00762 ContInpX ; moves LHP leds from FLAG to PORTB
0193 0890 00763 bcf FLAG,LEDL ; init input bit for led L delay rotor
0194 0989 00764 movf DELAYL,F ; test DELAYL status...
0195 1D03 00765 btfss STATUS,Z ; ...and skip if DELAYL=0
0196 178C 00766 bsf FLAG,LEDL ; turn on led L if DELAYL rotor > 0
0197 0D98 00767 rlf DELAYL,F ; propagate bit thru DELAYL rotor
0198 130C 00768 bcf FLAG,LEDH ; init input bit for led H delay rotor
0199 0899 00769 movf DELAYH,F ; test DELAYH status...
019A 1D03 00770 btfss STATUS,Z ; ...and skip if DELAYH=0
019B 170C 00771 bsf FLAG,LEDH ; turn on led H if DELAYH rotor > 0
019C 0D99 00772 rlf DELAYH,F ; propagate bit thru DELAYH rotor
019D 23F0 00773 call MoveLESd ; send leds status flag bits to PORTB
019E 0F8E 00774 incfsz DJNZ,F ; test if this is 256th pass...
019F 00775 goto GoKbd ; ...if not, loop
01A0 128C 00776 bcf FLAG,LEDP ; reset led P flag (set if PCOUNT>0)
01A1 0890 00777 movf PCOUNT,F ; test PCOUNT status...
01A2 1903 00778 btfsc STATUS,Z ; ...and skip jump if PCOUNT>0
01A3 29A9 00779 goto PCount0 ; ...else jump if PCOUNT = 0
01A4 0390 00780 decf PCOUNT,F ; if PCOUNT>0, then decrement it
01A5 0818 00781 movf DELAYL,W ; W>0 if led L is on
01A6 0419 00782 iorwf DELAYH,W ; W>0 if led L or led H is on
01A7 1D03 00783 bsf FLAG,LEDP ; if PCOUNT>0, dec PCOUNT, & set led P
01A8 168C 00784 goto PCount0 ; ...else jump if PCOUNT = 0
01A9 0792 00785 PCOUNT0
01A9 0E05 00786 swapf PORTA,W ; let key 1&2 status move to bits 6&7
01AA 008F 00787 movwf SCRATCH ; SCRATCH,7=key2, SCARTCH,6=key1
01AB 098F 00788 comf SCRATCH,F ; complement to set bit if key pressed
01AC 088F 00789 rlf SCRATCH,F ; set C if key 2 pressed
01AD 0D93 00790 rlf DEBO2,F ; propagate key 2 bit thru rotor
01AE 1003 00791 bcf DEBO2,F ; reset C,notify at exit key 2 pressed
01AF 0313 00792 decf DEBO2,W ; DEBO2 = b'00000001' if just pressed
01B0 1903 00793 bsf STATUS,Z ; ...skip if both L and H leds are off
01B1 0008 00794 return ; *** exit 1: key 2 just pressed (NC)
01B2 0B9B 00795 decfsz TIMOUTH,F ; timeout hi counter...
01B3 2976 00796 goto GoKbd ; ... not yet zero: loop
01B4 0B9C 00797 goto GoKbd ; *** exit 3:cont with timeout process
AN689

00823 ;******************************************************************************
00824 ;* Power Off entry point
00825 ;* Wait until both keys off for 34 ms, and then switch power off.
00826 ;* PORTB,3, when low, switches the unit off.
00827 ;******************************************************************************
01BC 00828 Suicide
01BC 21BF 00829 call KeysOff ; test keys off to avoid re-triggering
01BD 0186 00830 clrf PORTB ; pull PORTB,3 low to switch power off
01BE 29BE 00831 goto $ ; loop until power off
00832
00833 ;******************************************************************************
00834 ;* KeysOff
00835 ;* Loop until both keys off for 34 ms, then exit
00836 ;*
00837 ;* Input variables: none
00838 ;* Output variables: TIMOUTH is cleared to 0
00839 ;******************************************************************************
01BF 00840 KeysOff
01BF 019C 00841 clrf TIMOUTH ; initialize pointer
01C0 00842 BothOff
01C0 1905 00843 btfs PORTA,2 ; skip if key 1 on
01C1 1D85 00844 btfs PORTA,3 ; do not skip if key 2 on
01C2 29BF 00845 goto KeysOff ; reinitialize ptr if any key on
01C3 23D0 00846 call loop130 ; loop 130us
01C4 0B9C 00847 decfsz TIMOUTH,F ; test pointer
01C5 29C0 00848 goto BothOff ; loop 256xs to verify both keys off
01C6 0008 00849 return ; both keys are off for at least 34 ms
00850
00851 ;******* ANALYZER
00852 ;******************************************************************************
00853 ;* Pointer2
00854 ;* Writes 2 measuring points for analyzer reference. 1st=TIMOUTL*10+5, TIMOUTL incremented by 3 (pointer advanced by 30)
00855 ;******************************************************************************
01C7 00860 Pointer2
01C7 21CF 00861 call Pointer1 ; first two digits
01C8 3035 00862 movlw '5' ; third digit is 5 for odd pointer
01C9 0A9B 00863 incf TIMOUTL,F ; each incr advances pointer by 10
01CA 21CD 00864 call AdvToCh ; advance ptr and print "5"
01CB 21CF 00865 call Pointer1 ; first two digits
01CC 304F 00866 movlw '0' ; third digit is 0 for even pointer
01CD 00867 AdvToCh
01CD 0A9B 00868 incf TIMOUTL,F ; each incr advances pointer by 10
01CE 2BD8 00869 goto Char ; print 0 or 5
00870
00871 ;******************************************************************************
00872 ;* Pointer1
00873 ;* Writes blank, then symbol "^" and then converts TIMOUTL and prints
00874 ;* as 2 digits.
00875 ;*
00876 ;* Input variables: TIMOUTL, bin value which prints as 2-digits
00877 ;* Output variables: none
00878 ;******************************************************************************
01CF 00879 Pointer1
01CF 23D7 00880 call Blank ; skip first 3 samples (one character)
01D0 305E 00881 movlw '^' ; "^" (pointing tool)
01D1 00882 PrintTL
01D1 23D8 00883 call Char ; print "^"
01D2 081B 00884 movf TIMOUTL,W ; TIMOUTL is the main pointer for...
01D3 2B70 00885 goto Print255 ; ...first two digits
00886
00887 ;******************************************************************************
00888 ;* ModelShow
00889 ;* Write measuring points at row 2, and wait until keys are released.
00890 ;* Then incr. SHOWCOU in range 0...4 and continue to Draw subroutine
00891 ;* This command, which executes when key 2 is pressed in analyzer mode,
00892 ;* while the cursor is on the number of 60-sample group, advances the
00893 ;* pointer
00894 ;* It continues to Draw routine.
00895 ;********************************************************************************
00896 ModelShow
00897 call Row2 ; pointers are in row 2
00898 bcf STATUS,C ; prepare for multiplying by 2
00899 rlf SHOWCOU,W ; W=SHOWCOU*2
00900 movwf TIMOUTL ; TIMOUTL=SHOWCOU*2
00901 rlf TIMOUTL,F ; TIMOUTL-SHOWCOU*4
00902 addwf TIMOUTL,F ; TIMOUTL-SHOWCOU*6
00903
00904 call Pointer2 ; print 1st and 2nd pointer
00905 call Pointer2 ; print 3rd and 4th pointer
00906
00907 call KeysOff ; wait until key off
00908 call PrintM1 ; restore normal row 2
00909
00910 incf SHOWCOU,F ; advance number of groups displayed
00911 btfsb SHOWCOU,2 ; test if SHOWCOU=5: first test bit...0
00912 goto Draw ; skip wrapping if SHOWCOU<5
00913 clrf SHOWCOU ; SHOWCOU cycle in range 0...4
00914 ;********************************************************************************
00915 ;* Draw
00916 ;* This subroutine writes 20 pseudographic chars in line 1 in analyzer
00917 ;* mode. First, whole buffer is rotated 0/60/120/180/240 bit places to
00918 ;* the right (if SHOWCOU=0/1/2/3/4, respectively) to adj. the sequence
00919 ;* to be displayed to the start of the buffer. Then the string of 20
00920 ;* 3-bit groups is rotated right, and displayed as 20 special chars
00921 ;* (codes 0-7), defined at program setup (at loop labeled GoGraph).
00922 ;* Then the buffer is rotated again, to the total of 304 bit places,
00923 ;* so the buffer contents is unmodified on exit.
00924 ;*
00925 ;* Input variables:
00926 ;* SHOWCOU, denotes which group of 60 samples will be displayed
00927 ;* Output variables: none
00928 ;********************************************************************************
00929 Draw
00930 movf SHOWCOU,W ; prep to rotate buf SHOWCOU*60 times
00931 btfsb STATUS,Z ; avoid rotating if SHOWCOU=0
00932 call Carousel ; rotate buffer SHOWCOU*60 times
00933 call Row1 ; samples must be written in row 1
00934 movlw .20 ; .20 characters to write
00935 movwf CHARCOU ; CHARCOU is the main character counter
00936 call Go20Chars ; register for 3-bit code gen (0...7)
00937 call RRBuf ; bit from buffer in C...
00938 call RRBuf ; bit from buffer in C...
00939 call RRBuf ; bit from buffer in C...
00940 call RRBuf ; bit from buffer in C...
00941 call RRBuf ; bit from buffer in C...
00942 call RRBuf ; bit from buffer in C...
00943 call RRBuf ; bit from buffer in C...
00944 call RRBuf ; bit from buffer in C...
00945 call RRBuf ; bit from buffer in C...
00946 call RRBuf ; bit from buffer in C...
00947 decfsz CHARCOU,F ; 20 characters written?
00948 goto Go20Chars ; no, loop
00949
00950 movf SHOWCOU,W ; prep to rotate to total of 304 bits
00951 sublw .4 ; W=4-SHOWCOU
00952 btfsb STATUS,Z ; avoid rotating if SHOWCOU=0
00953 call Carousel ; rotate buffer again to restore it
01F7 2200 00955 call RRBuf4 ; four more times to get 304 times
01F8 28CD 00956 goto Farm1 ; done, go back to user interface
00957
00958 ;*******************************************************************
00959 ; * Carusel
00960 ; * This subroutine rotates BUFFER right W*60 times
00961 ; * Note: if W=0, rotating will be performed 1024 times
00962 ; *
00963 ; * Input variables: W,how many (*60) x the buf is rotated right (W>0)
00964 ; * Output variables: none
00965 ;*******************************************************************
01F9 00966 Carusel
01F9 008F 00967 movwf SCRATCH ; SCRATCH=W
01FA 0E8F 00968 swapf SCRATCH,F ; SCRATCH=W*16
01FB 028F 00969 subwf SCRATCH,F ; SCRATCH=W*15
01FC 00970 RRLoop
01FC 2200 00971 call RRBuf4 ; 4 rotates in every pass
01FD 0B8F 00972 decfsz SCRATCH,F ; done W*15*4 times?
01FE 29FC 00973 goto RRLoop ; no, continue rotating BUFFER
01FF 0008 00974 return ; done
00975
00976 ;*******************************************************************
00977 ; * RRBuf4 executes RRBuf 4 times
00978 ; * RRBuf rotates buffer (38 bytes=304 bits) right for one bit position.
00979 ; * Bit STATUS,C is first loaded from the first bit in buffer, so
00980 ; * will rotating be completely performed at 304 bits, not through C.
00981 ; *
00982 ; * Input/Output variables: none
00983 ;*********************************************************************
0200 00984 RRBuf4 ; 4 times rotates right 38 bytes
0200 2201 00985 call RRBuf2 ; rotate BUFFER 2* and then again 2*
0201 00986 RRBuf2
0201 2202 00987 call RRBuf ; rotate BUFFER once and then again once
0202 00988 RRBuf ; rotates rt 38 bytes, bit in C at exit
0202 304B 00989 movlw BUFFER+.37 ; start with last byte to be rotated
0204 00990 movwf FSR ; FSR = pointing register for rotating
0204 3026 00991 movlw .38 ; 38 bytes total buffer
0205 00992 movwf COUNT ; byte counter
0205 00993
0206 1003 00994 bcf STATUS,C ; C rotated into BUFFER+.37, so it...
0207 1826 00995 btfsc BUFFER,0 ; ...must be equal to bit BUFFER+0,0...
0208 1403 00996 bsf STATUS,C ; ...to perform non-destruct rotating
0209 00997 ByteLoop
0209 OC80 00998 rrf INDF,F ; byte rotated here
020A 0384 00999 decf FSR,F ; let FSR point to next byte
020B 0BB4 01000 decfsz COUNT,F ; COUNT is byte counter, done?
020C 2A09 01001 goto ByteLoop ; no, loop
020D 0008 01002 return ; here the output bit is in STATUS,C
01003
01004 ;*******************************************************************
01005 ; * Mode1Go
01006 ; * This is entry point for analyzer start command (symbol * ). After
01007 ; * clearing line1 of LCD, the program vectors to routines which handle
01008 ; * different sampling rates. Three highest rates (1 MHz, 500 KHz and
01009 ; * 228 KHz are sampled at individual routines (Get1MHz, Get500KHz and
01010 ; * Get228KHz), and the remaining rates at routine GetSlowClk. All those
01011 ; * routines (except Get1MHz, which is location-sensitive (so it is at
01012 ; * the very beginning of the program), are listed here.
01013 ; *
01014 ; * Input/Output variables: none
01015 ;*********************************************************************
020E 01016 Mode1Go ; *** ept: analyzer start
01017
020E 22B7 01018 call ClrRow1 ; clear LCD line 1 and turn LEDs off
01019
020F 0815 01020 movf RATE,W ; RATE = sample rate in range 0...15
0210 1903 01021 btfs C STATUS,Z ; skip if sample rate>0
0211 2801 01022 goto Get1MHz ; jump to individual routine if RATE=0
01023
01024 ; --- try 500 KHz rate
0212 008F 01025 movwf SCRATCH ; SCRATCH = RATE
0213 0B8F 01026 decfsz SCRATCH,F ; test if RATE=1
0214 2A2B 01027 goto Try228KHz ; if not RATE=1, then try if RATE=2
01028 ; if RATE=1, then drop to Get500KHz
01029
01030 ;******************************************************************************
01031 ;* Get500KHz
01032 ;* This subroutine fetches 304 samples at 2us rate (5 instr cycles
01033 ;* timing).
01034 ;* Call Common initializes loop counter (COUNT) to 38*8=304 samples
01035 ;* and FSR to point to BUFFER. It also presets T0SE bit depended on
01036 ;* XTOX bit (in FLAG register)to enable proper edge detecting, as it
01037 ;* will affect TMRO state.
01038 ;* State of key 2(Break) tested while waiting for starting condition.
01039 ;*
01040 ;* Input/Output variables: none
01041 ;******************************************************************************
0215 01042 Get500KHz ; 5 t read cycle
0215 3026 01043 movlw .38 ; 38 bytes * 8 bits = 304 samples
0216 226F 01044 call Common ; initialize COUNT, FSR, hi-imp out...  
01045 ; ...bit XTOX and T0SE bit
0217 01046 Edge500
0217 1D85 01047 btfss PORTA,3 ; test status of key 2 and...  
0218 28CC 01048 goto Break ; ...jump to Break routine if pressed
01049
0219 0801 01050 movf TMRO,W ; TMRO = logic level edge detector
021A 1903 01051 btfs C STATUS,Z ; test if there was egde...
021B 2A17 01052 goto Edge500 ; ...and loop if not
021C 0384 01053 decf FSR,F ; adj pointer as it will be advanced...
021D 01054 ; ... before data write
021D 01055 Get500Loop
021D OC85 01056 rrf PORTA,F ; <-- move input status to C
021E 0A84 01057 incf FSR,F ; advance write pointer
021F OC80 01058 rrf INDF,F ; write bit C in destination rotor
0220 3006 01059 movlw .6 ; initialize count for 6 bits
0221 008E 01060 movwf DJNZ ; DJNZ = bit counter
0222 01061 Go6Bits
0222 OC85 01062 rrf PORTA,F ; <-- 6* move input status to C
0223 OC80 01063 rrf INDF,F ; write bit C in destination rotor
0224 088E 01064 decfsz DJNZ,F ; DJNZ = bit counter
0225 2A22 01065 goto Go6Bits ; loop if not yet 6 bits fetched
0226 01066
0226 OC85 01067 rrf PORTA,F ; <-- move input status to C
0227 OC80 01068 rrf INDF,F ; write bit C in destination rotor
0228 0B94 01069 decfsz COUNT,F ; COUNT = byte counter
0229 2A1D 01070 goto Get500Loop ; loop if not yet 38 bytes fetched
022A 2A6A 01071 goto Finished ; all bits fetched; go display them
01072
01073 ;******************************************************************************
01074 ;* Test if register SCRATCH reaches 0 after decrementing(this happens
01075 ;* if RATE = 2), if so drop to Get228KHz else go to GetSlowClk
01076 ;******************************************************************************
022B 01077 Try228KHz
022B 0B8F 01078 decfsz SCRATCH,F ; test RATE status (SCRATCH=RATE-1)
022C 2A3F 01079 goto GetSlowClk ; jump if not RATE=2
01080
01081 ;******************************************************************************
01083 ;* This subroutine fetches 304 samples at 4.4us rate (11 instruction
01084 ;* cycles timing).
01085 ;* Call Common304 initializes loop counter (COUNT) to 304 samples
01086 ;* and FSR to point to BUFFER. It also presets T0SE bit depended on
01087 ;* XTOX bit (in FLAG register)to enable proper edge detecting, as it
01088 ;* will affect TMR0 state.
01089 ;* State of key 2(Break) tested while waiting for starting condition.
01090 ;*
01091 ;* Input/Outuput variables: none
01092 ;**************************************************************************
022D 01093 Get228KHz ; 11 t read cycle
022D 226C 01094 call Common304 ; init COUNT(lo), TIMOUTH(hi byte)...  
01095 ;...FSR, hi-imp outbit XTOX & T0SE bit
022E 01096 Edge228
022E 1D85 01097 btfss PORTA,3 ; test status of key 2 and...
022F 28CC 01098 goto Break ; ...jump to Break routine if pressed
01099
0230 0801 01100 movf TMR0,W ; TMR0 = logic level edge detector
0231 1903 01101 btfsc STATUS,Z ; test if there was edge...
0232 2A2E 01102 goto Edge228 ; ...and loop if not
0233
0233 2A34 01103 Go228
0234 01105 Go228B
0234 0C85 01106 rrf PORTA,F ; <---  ; 1     move input bit to C
0235 0C80 01107 rrf INDF,F ; 1     write bitC in destination rotor
01108
0236 0314 01109 decf COUNT,W ; 1     COUNT = bit counter
0237 3907 01110 andlw 7 ; 1     test if 8th pass...
0238 1903 01111 btfsc STATUS,Z ; 1 (2) ...and skip if not
0239 0A84 01112 incf FSR,F ; 1 (0) ...else advance pointer
01113
023A 0B94           01114 decfsz COUNT,F ; 1     COUNT = (lo byte) bit counter
023B 2A33 01115 got0 Go228 ; 2     loop if not yet = 0
01116
023C 0B9C           01117 decfsz TIMOUTH,F ; TIMOUTH = (hi byte) bit counter
023D 2A34 01118 got0 Go228B ; this does not add extra cycles, as it
01119 ;  
023E 2A6A           01120 got0 Finished ; all bits fetched; go display them
01121
01122 ;**************************************************************************
01123 ;* GetSlowClk
01124 ;* This subroutine fetches 304 samples at variable rate, depended on
01125 ;* RATE (SCRATCH=RATE-3). Timing constant is loaded from lookup table
01126 ;* located at DATA EEPROM (locations 30h-3ch).
01127 ;*
01128 ;* Call Common304 initializes 16-bit loop counter to 304 samples
01129 ;* (lo byte: COUNT=.304-.256, hi byte: TIMOUTH=.1)
01130 ;* and FSR to point to BUFFER. It also presets T0SE bit depended on
01131 ;* XTOX bit (in FLAG register) to enable proper edge detecting, as it
01132 ;* will affect TMR0 state.
01133 ;*
01134 ;* Rates 3-11 (100KHz-2.4KHz) have loop period of factor from EEPROM
01135 ;* table multiplied by 5 instruction cycles and adding 20 instruction
01136 ;* cycles (Factor\*5T+20T), and rates 12-15 (1.2KHz-40Hz) multilied the
01137 ;* factor by 417 instruction cycles and adding 415 instruction cycle
01138 ;* (Factor\*417T+415T)
01139 ;*
01140 ;* RATE = 3, factor: 1, T/cycle: 25, Freq: 100 KHz
01141 ;* RATE = 4, factor: 6, T/cycle: 50, Freq: 50 KHz
01142 ;* RATE = 5, factor: 9, T/cycle: 65, Freq: 38.4 KHz
01143 ;* RATE = 6, factor: 16, T/cycle: 100, Freq: 25 KHz
01144 ;* RATE = 7, factor: 22, T/cycle: 130, Freq: 19.2 KHz
01145 ;* RATE = 8, factor: 46, T/cycle: 250, Freq: 10 KHz
01146 ;* RATE = 9, factor: 48, T/cycle: 260, Freq: 9.6 KHz
01147 ;* RATE = 10, factor: 100, T/cycle: 520, Freq: 4.8 KHz
01148 ;* RATE = 11, factor: 204, T/cycle: 1040, Freq: 2.4 KHz
01149 ;* RATE = 12, factor: 5, T/cycle: 2500, Freq: 1 KHz
01150 ;* RATE = 13, factor: 14, T/cycle: 6253, Freq: 400 Hz
01151 ;* RATE = 14, factor: 59, T/cycle: 25018, Freq: 100 Hz
01152 ;* RATE = 15, factor: 149, T/cycle: 62548, Freq: 40 Hz
01153 ;*
01154 ;* State of key 2 (Break) is tested while waiting for start condition.
01155 ;* LED P is turned on while sampling, indicate sample period at slower
01156 ;* rates, in which it appears to be visible.
01157 ;* Input variables: RATE, affects timing factor
01158 ;* Output variables: none
01159 ;*********************************************************************
01160                  GetSlowClk
01161          decf SCRATCH,W       ; W = RATE-3
01162          addlw .48             ; rate table in data eeprom @ addr .48
01163          call    AGet_EE         ; get time const. from dataeeprom table
01164          movwf   SCRATCH         ; time const for rates 3-15 -> SCRATCH
01165          call    Common304       ; init COUNT(lo), TIMOUTH(hi byte)...
01166          ...FSR, hi-imp outbit XTOX & T0SE bit
01167          EdgeSlow
01168          btfss   PORTA,3         ; test status of key 2 and...
01169          goto    Break           ; ...jump to Break routine if pressed
01170          EdgeSlow
01171          movf    TMR0,W          ; TMR0 = logic level edge detector
01172          btfsc   STATUS,Z        ; test if there was edge...
01173          goto    EdgeSlow        ; ...and loop if not
01174          decf    COUNT,W         ; COUNT = bit counter
01175          andlw   7               ; test if 8th pass...
01176          btfsc   STATUS,Z        ; ...and skip if not
01177          GoSlow
01178          bsf     PORTB,5         ; turn led P on, notify sampling on
01179          
01180                  GoSlow
01181          rrf     PORTA,F ; <--- ; 1 move input bit to C
01182          rrf     INDF,F          ; 1 write bitC in destination rotor
01183          Count
01184          decf    COUNT,W         ; 1 COUNT = bit counter
01185          andlw   7               ; 1 test if 8th pass...
01186          btfsc   STATUS,Z        ; 1 (2) ...and skip if not
01187          incf    FSR,F           ; 1 (0) ...else advance pointer
01188          Rate
01189          btfsc   RATE,3          ; 1 1 if bits2 and 3 cleared, that...
01190          btfsc   STATUS,Z        ; 2 1 ...means that rate<12...
01191          goto    Not417          ; - 2 ...if so, jump to short timing
01192          417
01193          movwf    SCRATCH,W       ; 1 SCRATCH = timing constant
01194          movwf   CHARCOU         ; 1 CHARCOU = loop counter
01195          Loop417                         ;   
01196          movlw   .82             ; 1  
01197          movwf   DJNZ            ; 1   
01198          call    Loop7           ; 411  > total cyc here 417*SCRATCH-1
01199          decfsz  COUNT,F         ; 1 2t at exit only
01200          goto    Loop417         ; 2 0t at exit only
01201          
01202          PORTB
01203          movf    PORTB,W         ; 1
01204          andlw   0dfh            ; 1 reset bit 5 (LED P)
01205          btfss   COUNT,4         ; 1 2
01206          iorlw   20h             ; 1 0 set bit 5 (LED P) if COUNT,4 = 0
01207          movwf    PORTB           ; 1 blink LED P while sampling each
01208          16th pass
01209          movlw   .78             ; 1 constant for long timing
01210          goto    SameAs5         ; 2 skip short timing
01211          Not417                         ;       
01212          movf    PORTB,W         ; 1
01213          movwf    SCRATCH,W       ; 1 SCRATCH = timing constant
01214          movwf   DJNZ            ; 1 417 and 5 (RATE 3-15)
01215          call    GoLoop          ; short time: 3T+5T*SCRATCH, long:393T
01216          GoLoop
01217          decfsz   COUNT,F         ; 1 COUNT = (lo byte) bit counter
01218          goto    GoSlow          ; 2 loop if not yet = 0
01220
0268 0B9C  decfsz TIMOUTH,F ; TIMOUTH = (hi byte) bit counter
0269 2A4A  goto GoSlow      ; adds 2T extra 1time, after 48th pass
026A 0197  Finished
026B 29E3  goto Draw        ; all bits fetched; display them
01226
01227 ;*********************************************************************
01228 ;* Common304
01229 ;* This subroutine initializes low byte loop counter (COUNT) to 304
01230 ;* samples (lo byte: COUNT=.304-.256, hi byte: TIMOUTH=.1+1)
01231 ;* and FSR to point to BUFFER. It also presets T0SE bit depended on
01232 ;* XTOX bit (in FLAG register) to enable proper edge detecting, as it
01233 ;* will affect TMR0 state.
01234 ;* Entry point Common allows subroutines Get1MHz and GetSlowClk to
01235 ;* preset COUNT to another values.
01236 ;*
01237 ;* Input variables:
01238 ;* For entry point COMMON, register W is placed in COUNT
01239 ;* Output variables:
01240 ;* COUNT is initialized to # of loop passes (W or low byte of 304)
01241 ;* TMR0 is cleared
01242 ;* T0SE and PORTB,2 are copied from FLAG,XTOX
01243 ;*********************************************************************
01244 Common304
026C 3002  movlw   .2              ; hi byte=2 for reg lo byte value...
01246                                 ; ...plus extra 256 passes
026D 009C  movwf   TIMOUTH         ; hi byte loop counter for 304 passes
026E 3030  movlw   .304-.256       ; lo byte value for 304 passes
026F 0094  movwf   COUNT           ; COUNT = loop counter
026F 0094  movwf   BUFFER          ; first byte of destination
0270 0084  movwf   FSR             ; FSR = destination pointer
01250 ;* COUNT is initialized to # of loop passes (W or low byte of 304)
01251 ;* TMR0 is cleared
01252 ;* T0SE and PORTB,2 are copied from FLAG,XTOX
01253 ;*********************************************************************
01254 026C 3038  movlw   38h             ; initialize T0SE fot L-to-H transition
01255 1106  bcf     PORTB,2         ; clr hi-imp out if expecting rise edge
01256
01257 0272 1A0C  btfsc   FLAG,XTOX       ; test slctd edge for start condition
01258 2A78  goto    ToOption        ; and skip falling edge if rise slctd
01259
01260 0272 3028  movlw   28h             ; initialize T0SE fot H-to-L transition
01261 1106  bcf     PORTB,2         ; clr hi-imp out if expecting fall edge
01262
01263 0272 1863  bsf     STATUS,RP0 ; select bank 1 of registers
01264 0272 0081  movwf   OPTION_REG      ; set/reset T0SE
01265 0272 1283  bsf     STATUS,RP0 ; reselect bank 0
01266 0272 0181  clrf    TMRO          ; initialize TMRO as edge detector
01267 0272 0008  return                  ; finished
01268
01269 ;*********************************************************************
01270 ;* PrintM1
01271 ;* Print string at line 2 in analyzer mode.
01272 ;* At pos 0, rate in format XX[.][X][M][K]Hz/XX[.][X][u|m]s is printed.
01273 ;* Those values picked from table located in Data EEPROM, locations
01274 ;* 0-2Fh. Register CHARCOU is used to fill blanks up to pos 13 in line
01275 ;* 2, to disable phantom characters appearance when changing rate from
01276 ;* some long-string to short-string value.
01277 ;* Symbol for starting or rising edge for starting event is written at
01278 ;* pos 13, symbol "*" for start command at pos 15 and the number of
01279 ;* group displayed at pos. 17.
01280 ;*
01281 ;* Input variables:
01282 ;* RATE will be printed in pos 0, row2
01283 ;* bit FLAG,XTOX affects the printed symbol of rising/falling edge
01284 ;* SHOWCOU (number of group displayed) is printed as numeric (+1)
01285 ;* Output variables: none
027D 01287 PrintM1
027D 23CE 01288 call Row2 ; move cursor to row 2
01289
027E 300D 01290 movlw .13 ; init counter for 13 char fix format
027F 0096 01291 movwf CHARCOU ; CHARCOU = character counter
01292
0280 0815 01293 movf RATE,W ; W = RATE
0281 0715 01294 addwf RATE,W ; W = 2 * RATE
0282 0715 01295 addwf RATE,W ; W = 3 * RATE (ea rate has 3 bytes
01296                                 ;                  in table)
0283 22AC 01297 call AGet_EE ; get 1st byte via table in data eeprom
0284 009B 01298 movwf TIMOUTL ; TIMOUTL = 1st byte from table
0285 0E1B 01299 swapf TIMOUTL,W ; move to bits 0-3, bits 4-7 are freq
01300
0286 22AD 01301 call Get_EE ; get 2nd byte via table in data eeprom
0287 1B1B 01302 btfsc TIMOUTL,6 ; bit 6 = decimal point for frequency
0288 148C 01303 bsf FLAG,DP ; set decimal point bit if bit 6 set
01304
0289 2373 01305 call Print3 ; display sampling frequency
01306
028A 304D 01307 movlw 'M' ; for "MHz" display
028B 0895 01308 movf RATE,F ; if RATE=0...
028C 1D9B 01309 btfss STATUS,Z ; ...then let it be MHz
028D 304B 01310 movlw 'K' ; for "KHz" display
028E 1B9B 01311 btfsc TIMOUTL,7 ; bit 7=KHz or MHz,skip 1st char if clr
028F 23D8 01312 call Char ; print "M" or "K" if bit 7 set
0290 3070 01313 movlw TxtHz-1+1 ; for "Hz" display
0291 23D8 01314 call Write ; print "Hz"
01315
0292 081B 01316 movf TIMOUTL,W ; TIMOUTL = 1st byte from table
01317
0293 22AD 01318 call Get_EE ; get 2nd byte via table in data eeprom
0294 191B 01319 btfsc TIMOUTL,2 ; bit 2 = decimal point for period
0295 1B9B 01320 btfsc TIMOUTL,6 ; bit 6 = decimal point for frequency
0296 2373 01321 call Print3 ; display digits for period
01322
0297 30E4 01323 movlw 0e4h ; Greek "micro"
0298 1D9B 01324 btfss TIMOUTL,3 ; if bit 3 set, let it be "micro"
0299 306D 01325 movlw 'm' ; ...if not, convert to "milli" (m)
029A 23D8 01326 call Char ; print "micro" or "m"
029B 3073 01327 movlw 's' ; s stands for seconds
029C 23D8 01328 call Char ; print "s"
029D 0A17 01329 incf SHOWCOU,W ; increment SHOWCOU, # of samples group
029E 23D9 01330 call CharNCC ; print blank without affecting CHARCOU
029F 0B96 01331 decfsz CHARCOU,F ; CHARCOU=character counter
02A0 2A9D 01332 goto XtraChar ; loop if not yet pos 13
01333
02A1 3001 01334 movlw 1 ; symbol of rising edge
02A2 1EDC 01335 btfss FLAG,XTOX ; let it be rising if XTOX set
02A3 3004 01336 movlw 4 ; symbol of falling edge
02A4 23D6 01337 call CharBl ; print symbol and blank
02A5 01338 goto XtraChar ; adds extra (CHARCOU) blanks
01339
02A6 23D6 0133A movlw ' ' ; blank blank to overprint prev string
02A7 0B96 0133B decfsz CHARCOU,F ; CHARCOU=character counter
02A8 2A9D 0133C goto XtraChar ; loop if not yet pos 13
0133D
02A9 23D6 0133E movlw 7eh ; 7eh-right arrow in LM032L char set
02AB 2BD8 01352 goto Char ; print arrow in rightmost pos
01353
01354 ;****************************************************************************
01355 ;* AGet_EE
01356 ;* Get_EE
01357 ;* This is routine for reading from Data EEPROM. Writing to BIN4+0 and
01358 ;* BIN4+1 is also integrated here, as those variables are used for bin
01359 ;* to decimal conversion.
01360 ;*
01361 ;* Input variables: W, data address at AGet_EE
01362 ;* Output variables: BIN4, binary data of rate display from DATA EEPROM
01363 ;******************************************************************************
02AC 01364 AGet_EE
02AC 0089 01365 movwf EEADR ; initialize eeprom address pointer
02AD 01366 Get_EE
02AD 3903 01367 andlw 3 ; hi byte BIN4 for freq display
01368
02AE 009F 01369 movwf BIN4+1 ; BIN4+1 = hi byte for range 0...999
01369
02AF 1683 01370 bsf STATUS,RP0 ; select bank 1 of registers
02B0 1408 01371 bsf EECON1,RD ; set handcshaking bit for data ee read
02B1 1283 01372 bcf STATUS,RP0 ; reselect bank 0
02B2 0808 01373 movf EEDATA,W ; reading from data eeprom
02B3 0A89 01374 incf EEADR,F ; adv address pointer for future read
02B4 009E 01375 movwf BIN4 ; lo byte BIN4 for freq display
02B5 108C 01376 bcf FLAG,DP ; clear decimal point flag
02B6 0008 01377 return ; finished
01378
01379 ;******************************************************************************
01380 ;* ClrRow1
01381 ;* SameAs20
01382 ;* This subroutine clears line 1 of LCD and switches off LEDs. Entry
01383 ;* point SameAs20 allows clearing some other number of character
01384 ;* positions starting from line 1 of LCD. All LEDs are turned off,
01385 ;* flags for LEDs also Cursor pointer of LCD is restored to first pos
01386 ;* of line 1 at exit of subroutine.
01387 ;*
01388 ;* Input variables:
01389 ;* entry point ClrRow1: none
01390 ;* entry point SameAs20: W=number of characters to be cleared
01391 ;* Output variables:
01392 ;* CHARCOU is decremented by the number of cleared characters
01393 ;* SHOWCOU is cleared to 0
01394 ;******************************************************************************
01395 ;* Row1
01396 ;* This subroutine moves cursor to pos 0 of row 1 on LCD.
01397 ;*
01398 ;* Input variables: none
01399 ;* Output variables: none
01400 ;******************************************************************************
02B7 01401 ClrRow1
02B7 3014 01402 movlw .20 ; 20 spaces (one row) to write
02B8 01403 SameAs20
02B8 0097 01404 movwf SHOWCOU ; SHOWCOU = space counter
02B9 22BF 01405 call Row1 ; move cursor to row 1
02BA 01406 LoopC1D
02BA 23D7 01407 call Blank ; print one space
02BB 0B97 01408 decfshz SHOWCOU,F ; SHOWCOU = space counter
02BC 2ABA 01409 goto LoopC1D ; loop if not yet 20 spaces
01410
02BD 301F 01411 movlw 1fh ; bits 7,6,5 (LED flags) reset
02BE 05BC 01412 andwf FLAG,F ; turn LED flags off
02BF 01413 Row1
02BF 3080 01414 movlw 080h ; command for line 1
02C0 2BCF 01415 goto WrComL ; go write command
01416
01417 ;******* SERIAL CODE RECEIVER
01418 ;*********************************************************************
01419 ;* Mode2Show
01420 ;* This subroutine advances SHOWCOU in range 0...5, and then continues
01421 ;* to subroutine Show2
01422 ;*********************************************************************
01423                Mode2Show
01424                incf SHOWCOU,F ; advance SHOWCOU
01425                btfsc SHOWCOU,2 ; bits 1 & 2 will be set if SHOWCOU...
01426                btfss SHOWCOU,1 ; ...is equal to 5...
01427                goto Show2 ; ...if not, skip clearing
01428                clrf SHOWCOU ; cycle show counter in range 0...5
01429
01430 ;*********************************************************************
01431 ;* Show2
01432 ;* This subroutine prints the 7 bytes of Buffer (+0,+7,+14,+21,+28 or
01433 ;* +35) in hex mode at line 1, and the same bytes in ASCII at line 2.
01434 ;* ASCII representation is with bit 7 reset, and non-printables (<20h)
01435 ;* are printed as dots
01436 ;*
01437 ;* Input variables:
01438 ;* SHOWCOU, denotes which group of 7 bytes will be displayed
01439 ;* Output variables:
01440 ;* CHARCOU is decremented by the number of characters printed
01441 ;*********************************************************************
01442                Show2
01443                movlw BUFFER-1 ; source pointer for reading -1
01444                btfsc SHOWCOU,2 ; if bit 2 of SHOWCOU set...
01445                addlw .28 ; ...then add 28 (4 groups) to pointer
01446                btfsc SHOWCOU,1 ; if bit 1 of SHOWCOU set...
01447                addlw .14 ; ...then add 14 (2 groups) to pointer
01448                btfsc SHOWCOU,0 ; if bit 0 of SHOWCOU set...
01449                addlw .7 ; ...then add 7 to pointer
01450                movwf FSR ; FSR on (1st byte pos)-1 to display
01451
01452                call Row1 ; move cursor to row 1
01453                movlw .7 ; bytes to display
01454                movwf SCRATCH ; SCRATCH = byte display counter
01455                Hex7
01456                incf FSR,F ; adv pointer (it was x-1 at beginning
01457                swapf INDF,W ; move hi nibble to display1. hex digit
01458                call HexDigit ; display 1st digit in hex mode
01459                movf INDF,W ; move lo nibble to disp 2nd hex digit
01460                call HexDigit ; display 2nd digit in hex mode
01461                call Blank ; blank after hex number
01462
01463                decfsz SCRATCH,F ; SCRATCH = byte counter
01464                goto Hex7 ; loop if not yet 7 bytes
01465
01466                movf FSR,W ; FSR = read pointer
01467                addlw -.7 ; restore it for ASCII mode printing
01468                movwf FSR ; FSR on (1st byte pos)-1 to display
01469
01470                call Row2 ; move cursor to row 2
01471                movlw .7 ; bytes to display
01472                movwf SCRATCH ; SCRATCH = byte display counter
01473                Ascii7
01474                incf FSR,F ; adv pointer (it was x-1 at beginning)
01475                movf INDF,W ; read byte
01476                andlw 7fh ; reduce ascii representation to 7 bits
01477
01478                addlw -20h ; test if byte < 20h
01479                addlw 20h ; restore previous value
01480                btfsc STATUS,C ; C is set if byte < 20h
01481                movlw 0a5h ; represent non-printables as dots
01482 ; (0a5h = dot)
01483                call Char ; display ascii char
AN689

02E7 0B8F 01484  decfsz SCRATCH,F ; SCRATCH = byte counter
02EA  2ADF 01485  goto  Ascii7  ; loop if not yet 7 bytes
02E9  28ED 01486  goto  Farm2  ; go back to user interface
02E9  28ED 01487
01488  ;*********************************************************************
01489  ;* Mode2Go
01490  ;* This is entry point for Start command in mode 2 (serial code rcver)
01491  ;* Line 1 and first 7 positions (ASCII chars) of line 2 on LCD is clr.
01492  ;* Here, buffer is cleared and a sequence of 42 bytes are received and
01493  ;* written to buffer. Manual break (key 2) jumps to Break handling.
01494  ;* This protocol is used: High start bit, 7 or 8 data (true) bits, 0 or
01495  ;* 1 parity bit (not written to memory) and 1 low stop bit (not tested
01496  ;* for validity). If RXBITS,2 is set then the input is inverted.
01497  ;* Baud rates 1200-115200 are supported.
01498  ;* Note: no receive errors are detected nor indicated.
01499  ;*
01500  ;* Input variables:
01501  ;*  RXRATE (range 0...7),which affects timing loaded via table BaudRate
01502  ;*  RXBITS, bits 0-2 significant:bit0=parity,bit1=7/8 bits,bit2=inverse
01503  ;*
01504  ;* Output variables:
01505  ;*  Buffer (42 bytes) loaded with bytes received, all unreceived bytes
01506  ;*  represented as 00s.
01507  ;*********************************************************************
02EA 01508       Mode2Go
02EA 1106 01509       bcf  PORTB,2  ; clear hi-imp probe output...
02EC 1506 01510       btfsc  RXBITS,2  ; ...let it be low if polarity bit clr
02EE 302F 01511       movlw .47  ; .47 blanks to clear displayed values
02EF 22B8 01512       call  SameAs20  ; clear displayed HEX and ASCII values
01513
02F0 3026 01514       movlw BUFFER  ; start of buffer...
02F1 0084 01515       movwf FSR  ; ...assigned to destination pointer
01516
02F2 081D 01517       movf RXRATE,W  ; RXRATE = selected rate in range 0...?
02F3 3E68 01518       addlw BaudRate  ; add to timing constant table offset
02F4 23B6 01519       call  PclSub  ; get rate to W
02F5 01520 01521       RX42Bytes
02F5 008E 01522       movwf DJNZ  ; baudrate timing factor to time cnte
01523
02F6 0194 01524       clrf COUNT  ; this is to preset bit counter to 8...
02F7 1594 01525       btfsc COUNT,3  ; ...and not to disturb W
01526
02F8 1C0D 01527       btfs RXBITS,0  ; RXBITS,0 is set if 7 bits selected
02F9 1C8D 01528       incf COUNT,F  ; if not(RXBITS and 3 = 2) then COUNT=9
01529
02FB 188D 01530       btfs RXBITS,1  ; RXBITS,1 is set if parity bit select
02FC 186D 01531       defc COUNT,F  ; ifnot(RXBITS and 3=1)reduce to 8 or 7
01532
02FF 1D0D 01533       btfs RXBITS,2  ; RXBITS,2 set if inverse polar slctd
01534
0300 2B07 01535       goto GetSp2  ; jump to true polarity if RXBITS,2 clr
01536
0300 1E05 01537       btfs PORTA,4  ; test input status...
0301 2B00 01538       goto GetSp1  ; ...and loop if still low
01539
0302 1B05 01540       GetStart1  ; ----- inverse rx
0302 1DB5 01541       btfs PORTA,3  ; test status of key 2...
0303 28EB 01542       goto BrkRS  ; ...and jump to Break if presseed
0304 1A05 01543       btfs PORTA,4  ; test input status...
0305 2B02 01544       goto GetStart1  ; ...and loop if still high
0306 2B0D 01545       goto StartFound  ; falling edge detected: start recept
01550
01551 GetSp2          ; ----- true rx
01552 btfsb PORTA,4  ; test input status...
01553 goto GetSp2    ; ...and loop if still high
01554 GetStart2
01555 btfsb PORTA,3  ; test status of key 2...
01556 goto BrkRS      ; ...and jump to Break if pressed
01557 btfsb PORTA,4  ; test input status...
01558 goto GetStart2  ; ...and loop if still low
01559
01560 StartFound     ; rising edge detected: start reception
01561
01562 HalfBit        ; 2-9 t from starting edge
01563 bsf PORTB,5    ; 1*W led P on
01564 nop            ; 1*W waist one cycle
01565 decfsz DJNZ,F ; 1*W DJNZ = timing constant counter
01566 goto HalfBit    ; 2*W loop if not half bit timing passed
01567
01568 RX8Bits
01569 movwf DJNZ      ; 1 move timing constant to counter
01570 OneBit
01571 call Waist8T    ; 8 8 * W waist 8 t
01572 decfsz DJNZ,F   ; 1 2 * W DJNZ=timing constant counter
01573 goto OneBit     ; 2 - * W loop if not 1 bit time passed
01574
01575 call Waist6T    ; 6 waist 6 t
01576
01577 rrf PORTA,F    ; 1 << move input status to C
01578 rrf INDF,F      ; 1 place C in input rotor
01579
01580 decfsz COUNT,F ; 1 COUNT = bit counter
01581 goto RX8Bits    ; 2 total 22t + (w-1) * 11
01582
01583 btfsb RXBITS,0 ; test if parity bit selected...
01584 rlf INDF,F      ; ...and discard parity bit if received
01585
01586 btfsb RXBITS,2 ; test if inverse polarity selected...
01587 comf INDF,F     ; ...and complement if true parity
01588
01589 bcf STATUS,C   ; clear 8th bit for 7-bit mode
01590 btfsb RXBITS,1  ; test if 7-bit mode selected and...
01591 rrf INDF,F      ; ...rotate 8th(zero) bit if 7bits selected
01592
01593 incf FSR,F      ; advance destination pointer
01594 btfsb FSR,6     ; bits 4 and 6 will both be set if...
01595 btfsb FSR,4     ; ... end of buffer+1 reached
01596 goto RX42Bytes  ; loop if not yet FSR=50h
01597 goto Show2      ; over: go show received bytes
01598
01599 ;****** FREQUENCY COUNTER
01600 ;*********************************************************************
01601 ;* GoPresc
01602 ;* Prescaler factor (variable PRESC, in range 0...3) is advanced
01603 ;* (executes when key 2 is pressed in frequency counter mode)
01604 ;*********************************************************************
01605 GoPresc
01606 incf PRESC,F    ; advance prescaler
01607 bcf PRESC,2     ; and cycle prescaler in range 0...3
01608
01609 ;*********************************************************************
01610 ;* FreqEp
01611 ;* Frequency counter entry point.
01612 ;* Subroutine WrParam does this: Displays message "Frequency" in row 1,
01613 ;* counter range (taken from table RangeTab) & resolution (taken from
01614 ;* table PrescTab) in row2. * LEDs are turned OFF, and the main counter (BIN4, 4 bytes) cleared.
The following is used to count pulses:

State of TMR0 is written to BIN4+0 and sequentially tested, and when bit7 of current value detected as 0 and the previous one was 1, the overflow is considered. In that case, state of BIN4+1 is advanced, extended to BIN4+2. After 500ms, the 32-bit value of BIN4 is shifted left in a total of PRESC+2 times, to get multiply by 4,8,16 or 32. Then BIN4 (4 bytes) is converted to ASCII and printed on LCD.

This routine does not call keyboard routine, as accurate timing of 500 ms must be generated for counting (high count register is CHARCOU to count SHOWCOU). Instead of this, there is the individual routine for key 1 and key 2 test, and also the countdown timer for automatic power-off (registers TIMOUTL, TIMOUTH).

If key 1 is pressed, mode 1 (analyzer) is entered. If key 2 is pressed, prescaler value is advanced.

Note: code from label "Loop500A" to comment "; 500 ms timeout" is real time code. If the # of cycles is changed, then the literals 0f4h+1 and 24h+1 written to CHARCOU and SHOWCOU must be readjusted.

Input variables: PRESC (affects prescaler factor)
Output variables: none

;*************************************************************
0328                FreqEp                          ; mode 3: frequency counter
0328 2364           call    WrParam         ; print "Frequency" and "xxMHz/Rxx"
0329 21BF           call    KeysOff         ; test both keys off for 34 ms and initialize 8 min auto off sequence
032A                Count500
032A 30D3           movlw   0d2h+REL        ; right arrow position
032B 23CF           call    WrComL          ; move cursor on right arrow
032C 019F           clrf    BIN4+1          ; clear next counter byte
032D 01A0           clrf    BIN4+2          ; clear next counter byte
032E 01A1           clrf    BIN4+3          ; clear next counter byte
032F 01B8           clrf    SCRAM               ; initialize TMR0 overflow detector
0330 30F5           movlw   0f4h+1          ; hi byte loop counter for 500 ms
0331 0096           movwf   CHARCOU         ; CHARCOU = hi byte counter
0332 3025           movlw   24h+1           ; 0f424h=.62500 cycles=1250000,T=500 ms
0333 0097           movwf   SHOWCOU         ; SHOWCOU = lo byte counter
0334 081A           movf    PRESC,W         ; PRESC = prescaler factor selected
0335 3E20           addlw   20h             ; for PRESC 0,1,2,3 w=20h,21h,22h,23h
0336 2278           call    ToOption        ; here is clrf TMR0 also
0337                Loop500A
0337 1D85           btfss   PORTA,3         ; test key 2 status and
0338 2B26           goto    GoPresc         ; jump to "prescaler change" if hit
0339                Loop500
0339 0801           movwf   TMR0,W          ; 1 1 1 only place where TMR0 is read
033A 009E           clrf    BIN4               ; rtec --- freq0
033B 0D1E           rlf     BIN4,W          ; 1 1 1 carry ---- TMR0.7
033C 0D8F           rlf     SCRATCH,F       ; 1 1 1 carry
033D 080F           movwf   SCRATCH,W       ; 1 1 1 SCRATCH=TMR0 overflow detect
033E 3903           andlw   3               ; 1 1 1 mask 2 LSbs for edge detect
033F 3A02           xorlw   2               ; 1 1 1 00000000 if 1 <<< 0
0340 1D03           btfsr   PORTA,Z         ; 1 2 2 skip if TMR0 overflow
0341 2B43           goto    NotOvf1         ; 2 -- if nz
0342 0A0F           incf    BIN4+1,F        ; - 11 nsb
0343 1903           NotOvf1 btfsr   STATUS,Z        ; 2 2 1 skip MSB adv if not overflow
0344 0AA0           incf    BIN4+2,F        ; - - 1 nsb
0345 309B           movlw   Head4-1          ; 1 initialize "Battery" message
0346 1D05           btfsr   PORTA,2         ; 2 test key 1 status and...
0347 2933 01682 goto Mode4 ; - got shortcut to mode 4 if pressed
01683
0348 0B97 01684 decfsz SHOWCOU,F ; 1 (2) 1o loop counter
01685 goto Loop500A ; 2 (->) 20T total
01686
034A 0B96 01687 decfsz CHARCOU,F ; (1) hi loop counter
01688 goto Loop500 ; (2) 20T total
01689
01689
01690 goto Suicide ; --- 500 ms timeout here
01691
034C 0A1A 01696 incf PRESC,W ; prepare for x2 multiply: PRESC incr
01697 movwf CHARCOU ; CHARCOU = multiply factor counter
01698 incf CHARCOU,F ; and prescaler constant incr again
01699 ShLoop ; BIN4 = BIN4 * 2 total (PRESC+2) times
034D 0197 01701 decfsz CHARCOU,F ; CHARCOU = multiplier factor counter
01702 goto ShLoop ; loop if not PRESC*2 times multiplied
01703
0350 008F 01704 movlw .59 ; .59 characters to clear
01705 call SameAs20 ; clear all but right arrow
01706 call Print8 ; print the frequency in 8-digit ASCII
01707
0359 0B98 01708 decfsz TIMOUTL,F ; TIMOUTL=lo byte auto power off counter
01709 goto Count500 ; inner loop
01710
01711 goto Suicide ; 8.5 min timeout - go switch power off
01712 ;*********************************************************************
01720 ;* Headline
01721 ;* Clears SHOWCOU (Headline2 skips this), Clears LCD, prints right
01722 ;* arrow 0 last pos of row2, prints message addressed by W+1 at page 0.
01723 ;* Terminator is last character with bit 7 set.
01724 ;* Input variables: W+1 addresses string (on page 0) to be printed
01725 ;* Output variables:
01726 Headline2 ;* CHARCOU is decremented by the number of characters printed
01727 Headline ;*********************************************************************
035F 0197 01730 clrf SHOWCOU ; initialize show group counter
01731 Headline2
0360 008F 01732 movwf SCRATCH ; move input parameter to SCARTCH
01733 movlw .59 ; .59 characters to clear
01734 call SameAs20 ; clear all but right arrow
01735 goto GoWrite ; print headline message on LCD
01736
01737 ;*********************************************************************
01740 ;* Headline
01741 ;* Clears SHOWCOU (Headline2 skips this), Clears LCD, prints right
01742 ;* arrow 0 last pos of row2, prints message addressed by W+1 at page 0.
01743 ;* Terminator is last character with bit 7 set.
01744 ;* Input variables: W+1 addresses string (on page 0) to be printed
01745 ;* Output variables:
01746 Headline2 ;* CHARCOU is decremented by the number of characters printed
01747 ;*********************************************************************
035F 0197 01730 clrf SHOWCOU ; initialize show group counter
01731 Headline2
0360 008F 01732 movwf SCRATCH ; move input parameter to SCARTCH
01733 movlw .59 ; .59 characters to clear
01734 call SameAs20 ; clear all but right arrow
01735 goto GoWrite ; print headline message on LCD
01736
01737 ;*********************************************************************
01740 ;* Headline
01741 ;* Clears SHOWCOU (Headline2 skips this), Clears LCD, prints right
01742 ;* arrow 0 last pos of row2, prints message addressed by W+1 at page 0.
01743 ;* Terminator is last character with bit 7 set.
01744 ;* Input variables: W+1 addresses string (on page 0) to be printed
01745 ;* Output variables:
01746 Headline2 ;* CHARCOU is decremented by the number of characters printed
01747 ;*********************************************************************
035F 0197 01730 clrf SHOWCOU ; initialize show group counter
01731 Headline2
0360 008F 01732 movwf SCRATCH ; move input parameter to SCARTCH
01733 movlw .59 ; .59 characters to clear
01734 call SameAs20 ; clear all but right arrow
01735 goto GoWrite ; print headline message on LCD
01736
01737 ;*********************************************************************
01740 ;* Headline
01741 ;* Clears SHOWCOU (Headline2 skips this), Clears LCD, prints right
01742 ;* arrow 0 last pos of row2, prints message addressed by W+1 at page 0.
01743 ;* Terminator is last character with bit 7 set.
01744 ;* Input variables: W+1 addresses string (on page 0) to be printed
01745 ;* Output variables:
01746 Headline2 ;* CHARCOU is decremented by the number of characters printed
01747 ;*********************************************************************
035F 0197 01730 clrf SHOWCOU ; initialize show group counter
01731 Headline2
0360 008F 01732 movwf SCRATCH ; move input parameter to SCARTCH
01733 movlw .59 ; .59 characters to clear
01734 call SameAs20 ; clear all but right arrow
01735 goto GoWrite ; print headline message on LCD
01736
01737 ;*********************************************************************
01740 ;* Headline
01741 ;* Clears SHOWCOU (Headline2 skips this), Clears LCD, prints right
01742 ;* arrow 0 last pos of row2, prints message addressed by W+1 at page 0.
01743 ;* Terminator is last character with bit 7 set.
01744 ;* Input variables: W+1 addresses string (on page 0) to be printed
01745 ;* Output variables:
01746 Headline2 ;* CHARCOU is decremented by the number of characters printed
01747 ;*********************************************************************
035F 0197 01730 clrf SHOWCOU ; initialize show group counter
01731 Headline2
0360 008F 01732 movwf SCRATCH ; move input parameter to SCARTCH
01733 movlw .59 ; .59 characters to clear
01734 call SameAs20 ; clear all but right arrow
01735 goto GoWrite ; print headline message on LCD
01736
01737 ;*********************************************************************
01740 ;* Headline
01741 ;* Clears SHOWCOU (Headline2 skips this), Clears LCD, prints right
01742 ;* arrow 0 last pos of row2, prints message addressed by W+1 at page 0.
01743 ;* Terminator is last character with bit 7 set.
01744 ;* Input variables: W+1 addresses string (on page 0) to be printed
01745 ;* Output variables:
01746 Headline2 ;* CHARCOU is decremented by the number of characters printed
01747 ;*********************************************************************
035F 0197 01730 clrf SHOWCOU ; initialize show group counter
01731 Headline2
0360 008F 01732 movwf SCRATCH ; move input parameter to SCARTCH
01733 movlw .59 ; .59 characters to clear
01734 call SameAs20 ; clear all but right arrow
01735 goto GoWrite ; print headline message on LCD
01736
01737 ;*********************************************************************
01740 ;* Headline
01741 ;* Clears SHOWCOU (Headline2 skips this), Clears LCD, prints right
01742 ;* arrow 0 last pos of row2, prints message addressed by W+1 at page 0.
01743 ;* Terminator is last character with bit 7 set.
01744 ;* Input variables: W+1 addresses string (on page 0) to be printed
01745 ;* Output variables:
01746 Headline2 ;* CHARCOU is decremented by the number of characters printed
01747 ;*********************************************************************
01748 ;*********************************************************************
01749 * Print255
01750 * Entry point Print255 converts 8-bit binary value (<100) in BIN4 to
01751 * 2-digit ASCII and prints it on LCD, without decimal point. Leading
01752 * zeros are printed.
01753 *
01754 * Input variables: W, binary number (0-99) to be converted and printed
01755 * Output variables:
01756 * CHARCOU is decremented by the number of characters printed
01757 ;*********************************************************************
01758 * Print3
01759 * Entrypoint Print3 converts 16-bit binary value (<1000) in BIN4 to 2-
01760 * or 3-digit ASCII and prints it on LCD. Leading zero is skipped only
01761 * if value is <100. Decimal point is printed between tens and ones if
01762 * FLAG,DP is set, otherwise decimal point is omitted.
01763 *
01764 * Input variables: BIN4 (2 bytes, LSB first, in range 0-999) binary
01765 * number to be converted and printed
01766 * Output variables:
01767 * CHARCOU is decremented by the number of characters printed
01768 ;*********************************************************************
01769 * PrintBR
01770 * Entry point PrintBR does the same as PRINT3, but the low byte value
01771 * is in W instead in BIN4+0. This is used for baud rate display.
01772 *
01773 * Input variables: W, BIN+1 (2 bytes, LSB in W, MSB in BIN+1, in range
01774 * 0-999) binary number to be converted and printed
01775 * Output variables:
01776 * CHARCOU is decremented by the number of characters printed
01777 ;*********************************************************************
0364                01778 WrParam                         ; print xxMHz/Rxx
0364 3092           01779         movlw   Head3-1         ; address of message "Frequency"
0365 2360           01780         call    Headline2       ; print message
0366 23CE           01781         call    Row2            ; move cursor to row 2
0367 3064           01782         movlw   RangeTab        ; offset of max frequency range table
0368 236E           01783         call    Presc255        ; print max frequency range
0369 0185           01784         clrf    BIN4+1          ; clear hi byte (allow range 00-99)
036A                 01785 Presc255
036B                01790 Print255
036C                 01791 PrintBR
036D 01A4           01792         movlw   .100            ; first digit constant
036E                 01793 Print3
036F                 01794 Print
0370                01795 Print3
0371                 01796 Print
0372                01797 Print
0373                01798 Print
0374                01799 Print
0375                01800 Print
0376                01801 Print
0377                01802 Print
0378                01803 Print
0379                01804 Print
037A                01805 Print
037B                01806 Print
037C                01807 Print
037D                01808 Print
037E                01809 Print
037F                01810 Print
0380                01811 Print
0381                01812 Print
0382                01813 Print

037E 23C9 01814  call  Num              ; print digit (tens)
01815
037F 302E 01816  movlw  '.'              ; decimal point
0380 188C 01817  btfsc  FLAG,DP          ; test decimal pnt flag, skip if reset
0381 23D8 01818  call  Char              ; print decimal point if DP set
0382 2B9B 01819  goto  NumBin4           ; print last digit (ones)
01820
01821 ;*********************************************************************
01822 ;* Times
01823 ;* Counts # of times CMP4 (32-bit value) "goes" in BIN4 (32-bit value).
01824 ;* BIN4 is sequentially subtracted by CMP4 and counter COUNT advanced.
01825 ;* When borrow is detected, BIN4 is restored to the last positiv value
01826 ;* (by ADDing CMP4 again), COUNTer decremented and written to W.
01827 ;*
01828 ;* Input variables: CMP4 (32-bit value), BIN4 (32-bit value)
01829 ;* Output variables:
01830 ;*    BIN4 (32-bit value) modified to mod(CMP4)
01831 ;*    W (in range 0...9) = BIN4 (32-bit value) / CMP4 (32-bit value)
01832 ;*********************************************************************
01833 0383  Times
01834 0383 00A2 01835  movwf   CMP4          ; place input param in CMP4 to compare
01836 0384 0194 01837  clrf    COUNT           ; clear result counter
01838 0385                01839 GoTD
01839 0A94 01840  incf    COUNT,F          ; advance result counter
01841 239D 01842  call    Sub4            ; BIN4=BIN4-CMP4  nc if result <0
01843 1803 01844  btfsc   STATUS,C         ; test did it "go"?
01845 2B85 01846  goto    GoTD            ; loop if so
01847 23A5 01848  call    Add4            ; BIN4=BIN4+CMP4  c set if ovf
01849 0389 0314 01850  decf    COUNT,W        ; W=# of times CMP4 goes in BIN4(32bit)
01851 038B 0008 01852  return             ; result in W
01853
01854 ;*********************************************************************
01855 ;* Print8
01856 ;* This subroutine converts 32-bit value in BIN4 (low byte first), to
01857 ;* 8-dig ASCII and prints to LCD. Leading zeros are printed as blanks.
01858 ;* Table DecTab (21 words, must be at page 0 if PCLATH=0) used in conv.
01859 ;*
01860 ;* Input variables: BIN4 (32-bit value, <.100,000,000)
01861 ;* Output variables:
01862 ;*    CHARCOU is decremented by the number of characters printed
01863 ;*********************************************************************
01864 038C  Print8
01865 038C 00A2 01866  movwf   CMP4          ; place input param in CMP4 to compare
01867 038D 0194 01868  clrf    COUNT           ; clear result counter
01869 038E                01870 GoTD
01870 008F 01871  movwf   SCRATCH          ; SCRATCH = tab ptr
01872 118C 01873  bcf     FLAG,RIPPLE     ; zeros initially print as blanks, until
01874 01875  ;...first non-zero appears
01876 038F                01877 Cif7
01877 01A5 01878  clrf    CMP4+3          ; clear CMP4+3, it is =0 in all cases
01879 0390 23B4 01880  call    PclSub2        ; get constant from table
01881 0391 00A4 01882  movwf   CMP4+2          ; load dec. const from table in CMP4+2
01883 0392 23B4 01884  call    PclSub2        ; get constant from table
01885 0393 00A3 01886  movwf   CMP4+1          ; load dec. const from table in CMP4+1
01887 0394 23B4 01888  call    PclSub2        ; get constant from table
01889 0395 23B3 01890  call    Times           ; how many times CMP4 goes in BIN4?
01891 0396 23C3 01892  call    NZNum          ; print if w>0 or RIPPLE=1, else blank
01893 0397 080F 01894  movf    SCRATCH,W       ; SCRATCH = table pointer
01895 0398 3EB9 01896  addiw  .237-DecTab    ; test if end of table
01897 0399 1C03 01898  btfss   STATUS,C        ; C set if end of table
01899 039A 2B8F 01873  goto    Cif7           ; if not end of table loop (will be 7x)
01880 039B 01874  NumBin4
01881 039C 001E 01875  movf    BIN4,W          ; last digit is in BIN4
01882 039D 2BC9 01876  goto    Num            ; last digit must be printed always
01883 01877 039E 01878  ;*********************************************************************
01879 ;* Sub4
01880 ;* Subtract CMP4 (32-byte value) from 32-bit value in BIN4, lo byte 1st
01881 ;* This is performed as adding of negative value of CMP4. Negating is
01882 ;* performed as complementing and incrementing by 1.
01883 ;* Note: Incrementing by 1 is performed on least significant byte only,
01884 ;* without 32-bit extension, for code space saving. This will not cause
01885 ;* error in this case, as the number of all possible values for CMP4+0
01886 ;* is limited and none of them is equal to 0FFh before incrementing
01887 ;* (all possible values are taken from table DecTab, & are: 0ah, 64h,
01888 ;* 0e8h, 10h, 0a0h, 40h and 80h, and their negative values).
01889 ;* However, this is valid if this subroutine is used for decimal
01890 ;* conversion only, and if it is used for some other application,
01891 ;* extension to 32-bit should be added after incrementing.
01892 ;*
01893 ;* Input variables: BIN4 (32-bit value), CMP4 (32-bit value)
01894 ;* Output variables:
01895 ;* BIN4 (32-bit value)
01896 ;* STATUS,C denotes the sign of result: if cleared, output value
01897 ;* is negative (there is borrow)
01898 ;*
01899 ;* Note: Entry points Waist8T and Waist6T are used only by some
01900 ;* real-time routines, in that case the instructions are dummy
01901 ;*********************************************************************

039D 01902 Sub4                            ; 32-bit sub: BIN4 = BIN4 - CMP4
039D 01903                                 ; NC if result<0
039E 01904 call    NegCmp          ; negate CMP (32 bits) first
039E 01905 call    Add4            ; add as negative value
039F 01906 NegCmp
039F 01907 comf    CMP4+0,F        ; complement low byte
03A0 01908 Waist8T
03A0 01909 comf    CMP4+1,F        ; complement next byte
03A1 01910 comf    CMP4+2,F        ; complement next byte
03A2 01911 Waist6T
03A2 01912 comf    CMP4+3,F        ; complement high byte
03A3 01913 incf    CMP4+0,F        ; neg = complement + 1
03A4 01914                                 ; (no need test overflow here, it will
03A4 01915                                 ; ...never reach 0 after incrementing)
03A4 01916 return                  ; finished
03A4 01917 ;*********************************************************************

03A5 01925 Add4                            ; 32-bit add: BIN4 = BIN4 + CMP4
03A5 01926 movf    CMP4,W          ; low byte
03A6 01927 addwf   BIN4,F          ; low byte add
03A6 01928
03A7 01929 movf    CMP4+1,W        ; next byte
03A8 01930 btfsz   STATUS,C        ; skip to simple add if C was reset
03A9 01931 incfsz  CMP4+1,W        ; add C if it was set
03AA 01932 addwf   BIN4+1,F        ; next byte add if NZ
03A3 01933
03AB 01934 movf    CMP4+2,W        ; next byte
03AC 01935 btfsz   STATUS,C        ; skip to simple add if C was reset
03AD 01936 incfsz  CMP4+2,W        ; add C if it was set
03AE 01937 addwf   BIN4+2,F        ; next byte add if NZ
03A3 01938
03AF 01939 movf    CMP4+3,W        ; high byte
03B0 01940 btfsz   STATUS,C        ; skip to simple add if C was reset
03B1 01941 incfsz  CMP4+3,W        ; add C if it was set
03B2 01942 addwf   BIN4+3,F        ; high byte add if NZ
03B3 01943
03B3 01944 return                  ; finished
03B3 01945
01945 ;*********************************************************************
01946 ;* PclSub is used for indirect addressing
01947 ;* PclSub1 uses SCRATCH instead of W as input parameter
01948 ;* PclSub2 advances pointer SCRATCH before executing
01949 ;* Note: PCLATH=0 in all cases. So all tables pointed by this routine
01950 ;* are on page 0
01951 ;*********************************************************************
03B4                01952 PclSub2
03B4 0A8F           01953         incf    SCRATCH,F       ; advance table pointer
03B5                01954 PclSub1
03B5 080F           01955         movf    SCRATCH,W       ; move table pointer to W
03B6                01956 PclSub
03B6 0082           01957         movwf   PCL             ; jump to address pointed by PCLATH,W
01958
01959 ;*********************************************************************
01960 ;* ClrBuf
01961 ;* ClrRam
01962 ;* Subroutine ClrBuf clears BUFFER (42 bytes)
01963 ;* Entry point ClrRam allows some other start point for clearing. It
01964 ;* clears internal RAM from address in W to the location 7Fh.
01965 ;* (locations 50h-7Fh, which do not exist in 16F84, are dummy).
01966 ;* Both entry points continue to disabling Enable signal for LCD and
01967 ;* 33.8 ms timing loop
01968 ;* Input variables:
01969 ;*  W is the start addr if area to be cleared (ClrRam entry point only)
01970 ;* Output variables: none
01971 ;*********************************************************************
03B7                01972 ClrBuf
03B7 3026           01973         movlw   BUFFER          ; get start address of buffer
03B8                01974 ClrRam
03B8 0084           01975         movwf   FSR             ; FSR = dest pointer for clearing
03B9                01976 Zeros
03B9 0180           01977         clrf    INDF            ; clear one byte
03BA 0A84           01978         incf    FSR,F           ; advance dest pointer
03BB 1F84           01979         btfss   FSR,7           ; test if end of RAM...
03BC 2BB9           01980         goto    Zeros           ; ...if not, loop - else move LEDs
01981
01982 ;*********************************************************************
01983 ;*  Entry point DisEna30: Remove enable and discharge signal, and
01984 ;* refresh LEDs. Then loop 33.8 ms
01985 ;* Entry point Wait30: Loop 33.8 ms
01986 ;* Input variables: none
01987 ;* Output variables: none
01988 ;*********************************************************************
03BD                01989 DisEna30
03BD 23EF           01990         call    DisEna          ; disable discharging output signal
03BE                01991 Wait30
03BE 0194           01992         clrf    COUNT           ; COUNT=time loop counter, to wait 33.8ms
03BF                01993 GoWait30
03BF 0084           01994         movwf   COUNT,F         ; COUNT = time loop counter
03C0 23D0           01995         call    loop130         ; wait 130 us
03C0 0B94           01996         decfsz  COUNT,F         ; COUNT = time loop counter
03C1 2BB9           01997         goto    GoWait30        ; loop if not yet 256 passes
03C2 0008           01998         return                  ; timing over
01999
02000 ;*********************************************************************
02001 ;* NZNum
02002 ;* Print numeric value in W (in range 0...9) on LCD. If FLAG,RIPPLE is
02003 ;* cleared, 0 is printed as blank. If non-zero numeric is printed, it
02004 ;* automatically sets FLAG,RIPPLE.
02005 ;* 0 (30h) prints as capital O (4Fh), for improved readability, as 0
02006 ;* may easily be substituted by 8 on LCD. This changing 0 to O is not
02007 ;* performed only in ASCII representation of recorded bytes in serial
02012 ;* code receiver.
02013 ;* Entry point NUM prints numeric unconditionally, independenly of bit
02014 ;* FLAG,RIPPLE.
02015 ;*
02016 ;* Input variables: 
02017 ;* W (0...9), number to be printed at current cursor position of LCD
02018 ;* Output variables: 
02019 ;* CHARCOU is decremented by the number of characters printed 
02020 ;*********************************************************************
03C3 02021 NZNum                ; same aso Num, only blank instead of 0
03C3 198C 02022                 btfscc FLAG,RIPPLE   ; test if RIPPLE bit set...
03C4 2BC9 02023                 goto    Num            ; ...if RIPPLE set, no more blanks
03C5 3E00 02024                 addlw   0               ; set Z flag if W=0
03C6 1903 02025                 btfscc STATUS,Z        ; is it = 0 ?
03C7 2BD7 02026                 goto    Blank           ; if so, jump to space routine
03C8 158C 02027                 bsf     FLAG,RIPPLE     ; if>0,clr RIPPLE bit, no more blanks
03C9 02028 Num                  
03C9 390F 02029                 andlw   0fh             ; isolate low nibble
03CA 1903 02030                 btfsc   STATUS,Z        ; is it = 0 ?
03CB 301F 02031                 movlw   'O'-30h         ; if so, initialize capital O
03CC 3E30 02032                 addlw   30h             ; adjust ASCII for numeric
03CD 2BD8 02033                 goto    Char            ; print digit
02034 ;**************** LCD ROUTINES 
02035 ;*********************************************************************
02036 ;* All these entry ponts of this subroutine are used in program:
02037 ;*
02038 ;* Row2: Issues command to move cursor to row2 of LCD, and loops 130
02039 ;* us, to allow time for LCD controller to execute command
02040 ;* WrComL: Issues command in W to the LCD, and loops 130 us, to allow
02041 ;* time to LCD controller to execute the command
02042 ;* loop130: Loops 130 us including call and return
02043 ;* GoLoop: Loops W*2 us
02044 ;* Loop7: Same as GoLoop, only 2t shorter (for smpl rate routine)
02045 ;*
02046 ;*********************************************************************
03CE 02047 Row2                  
03CE 30C0 02048                 movlw   0c0h            ; command for line 2
02049 
03CF 02050 WrComL                ; issues command in W
03CF 23E5 02051                 call    WrCom           ; write command in LCD
03D0 02052 loop130               ; * waist 130 us
03D0 018E 02053                 clrfl DJNZ          ; this is to init DJNZ to 40h and...
03D1 170E 02054                 bsf     DJNZ,6          ; not to disturb W
03D2 02055 GoLoop                
03D2 2BD3 02056                 goto    $+1             ; 2 waist 2 t
03D3 02057 Loop7                
03D3 0B8E 02058                 dcfsz  DJNZ,F         ; 1 DJNZ - timing loop counter
03D4 2BD2 02059                 goto    GoLoop          ; 2 64x5=320c (128 us)
03D5 0008 02060                
03D6 02061 return               ; 2 finished
02062 ;*********************************************************************
02063 ;* All these entry pons of this subroutine are used in program:
02064 ;*
02065 ;* CharBl: print character in W, blank on LCD and decrement CHARCOU
02066 ;* Blank: print blank (32h) on LCD and decrement CHARCOU
02067 ;* Char: print character in W on LCD and decrement CHARCOU
02068 ;* CharNCC: print character in W on LCD without affecting CHARCOU
02069 ;*
02070 ;* Note: CHARCOU is used to print fixed format message on LCD, as the
02071 ;* calling routine will add N-CHARCOU blanks to fill area N chars long
02072 ;*
02073 ;* Input variables: 
02074 ;* all entry points except Blank: W = character to be printed
02075 ;* Output variables: 
02076 ;* all entry points except CharNCC: CHARCOU is decremented by 1
02077 ;*********************************************************************
03D6  02078 CharBl      ; print char, then blank
03D6 23D8 02079 call Char   ; print char first
03D7  02080 Blank         ; print blank
03D7 3020 02081 movlw ' '  ; print blank
03D8  02082 Char          ; print W
03D8 0396 02083 decf CHARCOU,F ; decrement character counter
03D9  02084 CharNCC        ; print W without affecting CHARCOU
03D9 1486 02085 bsf PORTB,1 ; pull RS hi (data register select)
03DA 2BE6 02086 goto Skr1 ; continue 4-bit mode writing to LCD
02087
02088 ;*********************************************************************
02089 ;* PrintBrk
02090 ;*  Print message "Break" in row 1, pos 0
02091 ;* Input variables: none
02092 ;* Output variables:
02093 ;*    CHARCOU is decremented by the number of characters printed
02094 ;*********************************************************************
02095 ;* Write
02096 ;* Print message addressed by W+1 in line 1
02097 ;* Note: Terminator is last character in string with bit 7 set
02098 ;* Input variables: W points to string (RETLWs) decremented by 1
02099 ;* Output variables:
02100 ;*    CHARCOU is decremented by the number of characters printed
02101 ;*********************************************************************
03DB  02102 PrintBrk
03DB 22BF 02103 call Row1   ; move cursor to row 1
03DC 30A2 02104 movlw BrkMes-1 ; message "Break" address-1
02105
03DD  02106 Write          ; write string addressed by W, end w/ 0
02107
03DE  02108 GoWrite        ; write command in W to LCD, then loop 130 us
02109
03DE 008F 02110 movwf SCRATCH ; SCRATCH = source pointer
02111
03DE 23B4 02112 call PclSub2 ; advance pointer and read pointed byte
02113
03DF 3E80 02114 addlw 80h   ; this is to test if bit 7 was set...
02115
03E0 1803 02116 btfsc STATUS,C ; ...if so, C will be set
02117
03E1 2BD8 02118 goto Char   ; last character was with bit 7 set
02119
03E2 397F 02120 andlw 7fh   ; restore initial character value
02121
03E3 23D8 02122 call Char   ; print one character
02123
03E4 2BDE 02124 goto GoWrite ; loop
02125
02126 ;*********************************************************************
02127 ;* Wrcom
02128 ;*  Write command in W to LCD, then loop 130 us
02129 ;*  Allows data write to LCD, if PORTB,1 is set previously
02130 ;*  Note: all entry ponits are terminated by 130us timing loop, to allow
02131 ;*  LCD controller to execute accepted command/data.
02132 ;* Input variables: command in W
02133 ;* Output variables: none
02134 ;*********************************************************************
03E5  02135 WrCom
03E6  02136 bcf PORTB,1      ; rs lo (command)
03E6 1086 02137 Skr1        ; skr1
03E6 008E 02138 movwf DJNZ ; save W in DJNZ for lo nibble writing
03E7 23F1 02139 call Hini_B  ; outputs w,4-7 to PORTB,4-7
03E8 23ED 02140 call EnaLCD ; generate enable signal for hi nibble
03E9  02141Nibble          ; restore init value of W and swap it
03E9 0E0E 02142 swapf DJNZ,W ; restore init value of W and swap it
03EA 23F1 02143 call Hini_B  ; outputs w,0-3 to PORTB,4-7
03EB 23ED 02144 call EnaLCD ; generate enable signal for low nibble
03EC 2BD0 02145 goto loop130 ; wait 130 for LCD to crunch command
02144 ;*********************************************************************
02145 ;* Generate Enable signal (1200us) for LCD controller, and refresh LEDs.
02146 ;* Entry point DisEna: Remove enable & dischg signal, and refresh LEDs.
02147 ;*********************************************************************
02148
02149 
02150 
02151 
02152 
02153 
02154 
02155 ;*********************************************************************
02156 ;* MoveLEDs
02157 ;* Entry point MoveLEDs: transfer FLAG, LEDP, LEDH and LEDL to PORTB, 5 PORTB, 6 and PORTB, 7 to service LEDs.
02158 ;*
02159 ;*
02160 ;* Input variables:
02161 ;* FLAG, bits LEDP, LEDH, LEDH will affect LED1, LED2, LED3
02162 ;* Output variables: none
02163 ;*********************************************************************
02164 ;* Hinib_B
02165 ;* Entry point Hinib_B: output W, 4-7 to 4-bit LCD data bus
02166 ;*
02167 ;* Input variables:
02168 ;* hi nibble of W is copied to LCD data bus
02169 ;* Output variables: none
02170 ;*********************************************************************
02171 
02172 
02173 
02174 
02175 
02176 
02177 
02178 
02179 
02180 
02181 
02182 ;*********************************************************************
02183 ;* HexDigit
02184 ;* This subroutine prints low nibble of W on LCD as hexadecimal digit.
02185 ;* Zero (30h) is printed as capital O (7Fh)
02186 ;*
02187 ;* Input variables: W in range 0...0fh, hex number to be printed
02188 ;* Output variables: CHARCOU is decremented by two
02189 ;*********************************************************************
02190 
02191 
02192 
02193 
02194 
02195 
02196 
02197 
02198 
02199 
02200 ;****** DATA EEPROM
02201 ;*********************************************************************
02202 ;* This table is located in data eeprom
02203 ;* It contains numerical data for 16 sample frequencies period display
02204 ;* for analyzer. Last 13 bytes are timing constants used by subroutine
02205 ;* GetSlowClk to generate internal timing (three fastest rates need no
02206 ;* constants from the table, as they are treated as special cases)
02207 ;*********************************************************************
02208 ;* ----- TABLE 1 (00h-2Fh): Rate display table for analyzer
02209 ;*
02210 ;* ****First byte: Flags. Bits in this byte have the following functs:
02211 ;* bit 7 = 0: Frequency in Hz
02212 ;*        = 1: Frequency in Mhz or Khz
02213 ;* bit 6 = 0: Frequency does not contain decimal point
02214 ;*        = 1: Frequency contains decimal point before last digit
02215 ;* bits 5,4: Bits 9 and 8 for frequency display, respectively
02216 ;* bit 3 = 0: Period in us (microseconds)
02217 ;*        = 1: Period in ms (milliseconds)
02218 ;* bit 2 = 0: Period does not contain decimal point
02219 ;*        = 1: Period contains decimal point before last digit
02220 ;* bits 1,0: Bits 9 and 8 for period display, respectively
02221 ;* **** Second byte: low significant byte for frequency
02222 ;* **** Third byte: low significant byte for period
02223 ;*********************************************************************
02224 ;* ----- TABLE 2 (30h-3Ch): Timing constant table for analyzer
02225 ;*
02226 ;* Timing constant factors to all sample rates generated by subroutine
02227 ;* GetSlowClk (all except 1MHz, 500KHz and 228KHz)
02228 ;*********************************************************************
02229 ;* Note: This is read-only data, so the Data EEPROM must be programmed
02230 ;* before the unit is ready to use. MCU will not affect data EEPROM
02231 ;* contents. If your programmer does not support automatic loading of
02232 ;* Data EEPROM contents from the HEX file, it must be loaded manually.
02233 ;* This will help in that case (all values are hexadecimal):
02234 ;* 02235 ;* addr 00-07:  88 01 01 98 F4 02 8C E4
02236 ;* addr 08-0f:  2C 88 64 0A 88 32 14 D8
02237 ;* addr 10-17:  80 1A 88 19 28 C8 C0 34
02238 ;* addr 18-1f:  88 0A 64 C8 60 68 C8 30
02239 ;* addr 20-27:  D0 C9 18 A1 80 01 01 14
02240 ;* addr 28-2f:  90 19 00 64 0A 00 28 19
02241 ;* addr 30-37:  01 06 09 10 16 2E 30 64
02242 ;* addr 38-3f:  CC 05 0E 3B 95
02243 ;*
02244 ;* Total bytes used in Data EEPROM: 61 (the last 3 bytes don't care)
02245 ;*********************************************************************
2100 ; org 2100h
02246 ; constant T/sample Hz  s  RATE
02247 ;
02248 ;
2100 0088 0001 0001 02249 de b'10001000', .1, .1 ; -  2.5 1M 1u 0
2103 0098 00F4 0002 02250 de b'10011000', .244, .2 ; -  5  500K 2u 1
2106 008C 00E4 002C 02251 de b'10001100', .228, .44 ; - 11  228K 4.4u 2
2109 0088 0064 000A 02252 de b'10001000', .100, .10 ;  1  100K 10u 3
210C 0088 0032 0014 02253 de b'10001000', .50, .20 ;  6  50K 20u 4
210F 00D8 0080 001A 02254 de b'11011000', .128, .26 ;  9  65  38.4K 26u 5
2112 0088 0019 0028 02255 de b'10001000', .25, .40 ; 16  100  25K 40u 6
2115 00C8 00CC 0034 02256 de b'110101000', .192, .52 ; 22  130  19.2K 52u 7
2118 0088 000A 0064 02257 de b'10001000', .10, .100 ; 46  250  10K 100u 8
211B 00C8 0060 0068 02258 de b'110101000', .96, .104 ; 48  260  9.6K 104u 9
211E 00C8 0030 0000 02259 de b'110011000', .48, .208 ; 100 520  4.8K 208u 10
2121 00C9 0018 00A1 02260 de b'11001001', .24, .161 ; 204 1040  2.4K 161u 11
2124 0080 0001 02261 de b'10000000', .1, .1 ; 5  2500  1K 1m 12
2127 0014 0090 0019 02262 de b'000101000', .144, .25 ; 14  6253  400 2.5m 13
212A 0000 0064 000A 02263 de b'00000000', .100, .10 ; 59  25018  100 10m 14
212D 0000 0028 0019 02264 de b'00000000', .40, .25 ; 149 62548  40 25m 15
02265 ; timing constants table
02266 ;
2130 0001 0006 0009 02267 de .001, .006, .009, .016, .022, .046, .048
0010 0016 002E 0030
02268 0064 00CC 0005 02269 de .100, .204, .005, .014, .059, .149
000E 003B 0095
02270 end

All other memory blocks unused.
Program Memory Words Used: 1023
Program Memory Words Free: 1

Errors : 0
Warnings : 0 reported, 0 suppressed
Messages : 4 reported, 0 suppressed
Note the following details of the code protection feature on PICmicro® MCUs.

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable”.
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip’s products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, FilterLab, KEELOQ, microID, MPLAB, PIC, PICmicro, PICMASTER, PICSTART, PRO MATE, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

dsPIC, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, MXDEV, PICC, PICDEM, PICDEM.net, rPIC, Select Mode and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A.

Serialized Quick Turn Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2002, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

Printed on recycled paper.
AMERICAS
Corporate Office
2335 West Chandler Blvd.
Chandler, AZ  85224-6199
Tel:  480-792-7200  Fax:  480-792-7277
Technical Support: 480-792-7627
Web Address: http://www.microchip.com

Rocky Mountain
2335 West Chandler Blvd.
Chandler, AZ  85224-6199
Tel: 480-792-7966  Fax: 480-792-7456

Atlanta
500 Sugar Mill Road, Suite 200B
Atlanta, GA  30350
Tel: 770-640-0034  Fax: 770-640-0307

Boston
2 Lan Drive, Suite 120
Westford, MA  01886
Tel: 978-692-3848  Fax: 978-692-3821

Chicago
333 Pierce Road, Suite 180
Itasca, IL  60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas
4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7943  Fax: 972-818-2924

Detroit
Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI  48334
Tel: 248-538-7627  Fax: 248-538-7269

Kokomo
2767 S. Albright Road
Kokomo, Indiana  46902
Tel: 765-864-8360 Fax: 765-864-8387

Los Angeles
18201 Von Karman, Suite 1090
Irvine, CA  92614
Tel: 949-263-1888  Fax: 949-263-1338

New York
100 Motor Parkway, Suite 202
Hauppauge, NY  11788
Tel: 631-244-4345 Fax: 631-244-4350

San Jose
Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA  95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto
6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

ASIA/PACIFIC

Australia
Microchip Technology Australia Pty Ltd
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

China - Beijing
Microchip Technology Consulting (Shanghai) Co., Ltd., Beijing Liaison Office
Unit 915
Bei Hai Wan Tai Bldg., No. 6 Choayangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Chengdu
Microchip Technology Consulting (Shanghai) Co., Ltd., Chengdu Liaison Office
Rm. 2401, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-6766200 Fax: 86-28-6766599

China - Fuzhou
Microchip Technology Consulting (Shanghai) Co., Ltd., Fuzhou Liaison Office
Unit 71 Wuluo Road
Fuzhou 350001, China
Tel: 86-591-7503506 Fax: 86-591-7503521

Japan
Microchip Technology Japan K.K.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471-6166 Fax: 81-45-471-6122

Korea
Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Singapore
Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 189890
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan
Microchip Technology Taiwan
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Denmark
Microchip Technology Nordic ApS
Regus Business Centre
Lautrup høj 1-3
Ballrup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France
Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trappu
Batiment A - ier Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany
Microchip Technology GmbH
Gustav-Heinemann Ring 125
D-81739 Munich, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy
Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

United Kingdom
Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5869 Fax: 44-118 921-5820

© 2002 Microchip Technology Inc.