Seeing Inside your Target at Run-Time with µC/Probe

Jean J. Labrosse, Founder, President and CEO
Jean J. Labrosse

- Master’s Degree in Electrical Engineering
- Wrote books (\texttt{\mu C/OS-xxx} and \texttt{ESBB})
- Wrote many papers for magazines
  - Embedded Systems Programming
  - Electronic Design
  - C/C++ User’s Journal
  - ASME
- Regular lecturer at ESC
- Designed Embedded Systems for over 30 years
- President of \texttt{Micrium}
  - Provider of Embedded Software Components (Middleware)
Renesas Technology & Solution Portfolio
Micrium at a Glance

- Founded in 1999 by Jean Labrosse, developer of $\mu$C/OS, $\mu$C/OS-II, and $\mu$C/OS-III
- Headquarters in Weston, FL (USA), with a second office in Montréal, QC (Canada)
- Mission is to provide high-quality, well-documented software to the embedded community
  - Micrium is truly “for the way engineers work”
  - With Micrium’s products, embedded software developers have a clear time-to-market advantage
Agenda

- Introduction
- Visualizing Run-Time Systems
- μC/Probe the Ideal Companion to RX
- μC/Probe Lab 1
- μC/Probe How Does it Work
- μC/Probe – The Ribbon
- μC/Probe – Advanced Objects
- μC/Probe – Lab 2
- μC/Probe – Lab 3
- When Should You Use μC/Probe
- μC/Probe – Lab 4
- Free Evaluation
- Q & A
Micrium’s Software Components

- **μC/OS-II and μC/OS-III**
  - Real-Time Kernels
  - μC/TimeSpaceOS and μC/OS-MPU
    - Kernel add-ons

- **μC/FS**
  - File system module

- **μC/GUI**
  - Graphical software for embedded systems

- **μC/FL**
  - Easy-to-use bootloader

- **μC/BUILDING BLOCKS**
  - μC/Shell
  - μC/CRC
  - μC/CIC
  - μC/LCD

- **μC/TCP-IP**
  - Highly dependable TCP/IP stack

- **μC/USB Device and μC/USB Host**
  - Versatile USB stacks

- **μC/CAN**
  - Robust CAN stack

- **μC/Modbus**
  - High-quality implementation of the Modbus protocol
Visualizing Run-Time Systems ... during Development
You can’t always ‘single step’ through code

- Industrial Engine
- Chemical reaction
- Printing Press
- Agricultural Equipment
- Food Processing
- Flight Controls
- Etc.
Using LEDs (annunciators)

- Adds recurring costs
- You have to write code ... typically simple
- Good at displaying go/no-go state
- LEDs are assigned to specific ‘things’
  - Health indicator
  - Running indicator
  - Power indicator
  - Detection of error conditions
  - Etc.
- Blink rate & color can provide addition
7-Segment Displays (LED or LCD-based)

- Popular in process control equipment
  - Temperature and PID controllers
- Adds recurring costs
  - Necessary for final product
  - Cost increases based on #digits
- You have to write (simple) code
  - Integer to 7-segment
  - Limited ‘ASCII’ characters
- You are limited to what you can ‘see’ and change
  - Needs simple user interface to change the monitored ‘value’ and change setpoints
- Blinking can provide additional feedback
Bargraphs

- Popular in process control equipment
  - Percentages
  - Temperatures
  - Pressures
  - Levels
  - Etc.
- Adds recurring costs
  - Necessary for final product
  - Cost increases based on #segments
  - Monitoring many values can get expensive
- You have to write (simple) code
  - Integer to N-bars
- You are limited to what you can ‘see’
printf() Statements

- Assumes you dedicate a serial port
- You have to write code
  - Insert `printf()` statements at strategic points
  - Move `printf()` statements to see different things
  - You have to ‘format’ the data (text, values)
  - `printf()` is known to require
    - a lot of code space
- Changes the run-time behavior
  - `printf()` are known to consume
    - a lot of CPU cycles
- Not very good at displaying trends/plots
- Data ‘flies off’ the screen quickly
Character-based LCD modules

- You need to add extra hardware
  - Extra recurring cost
  - Requires extra I/Os
  - Which Display Module?
    - 1x16, 2x16, 2x20, 4x16, 4x20, 2x40

- You have to write code
  - Formatting / positioning
  - User Interface

- Can only display a few values
- Limited bargraphs
- Limited plots
Full Color Graphics Display

- You need to add extra hardware
  - Extra recurring cost (expensive)
    - End product may not need a graphics display
  - Requires extra I/Os
  - Which Display Module?
    - B&W, Grayscale or Color?
    - What size and resolution?

- You have to write complex code
  - The GUI code itself
  - Formatting / positioning
  - User Interface

- Uses a LOT of code ... > 10s of Kbytes!
- Uses a LOT of CPU time ... 10-50%!
Debuggers

- Typically provide access to variables through a watch window
- Variables are displayed as text
- Updates occur only at breakpoints
- Heisenberg effect is often significant

```java
static void AppTempCtrl (void)
{
    AppTempErr = AppTempActual - AppTempStop;
    AppTempHeatRate = ((CPU_FP32)AppTempHeatRate) / (CPU_FP32)1000.0;
    AppTempCoolRate = ((CPU_FP32)AppTempCoolRate) / (CPU_FP32)1000.0;
    if (AppTempActual > (AppTempStop + AppTempHyst)) |
        AppTempState = 3; /* Above Stop + Hyst */
    else if (AppTempActual < (AppTempStop - AppTempHyst)) |
        AppTempState = 1; /* Below Stop - Hyst */
    else |
        AppTempState = 2; /* Between Stop + Hyst and Stop - Hyst */

    if (AppTempCtrlEn == DEF_ENABLED) |
        BSP_LED_Toggle(2); /* See if controller is turned on */
    if (AppTempSelHeater == DEF_ON) |
        AppTempAC_Ctrl = DEF_OFF; /* See if heater is selected */
    switch (AppTempState) |
    case 1:
        AppTempHeatCtrl = DEF_ON;
        AppTempActual = AppTempHeatRate;
        BSP_LED_On(3);
        BSP_LED_OFF(1);
        break;
    case 2:
        if (AppTempHeatCtrl) |
            AppTempActual = AppTempHeatRate;
        else |
            AppTempActual = (CPU_FP32)0.0005; /* Cool the room at natural rate */
        break;
    case 3:
        AppTempHeatCtrl = DEF_OFF;
        AppTempActual = (CPU_FP32)0.0005; /* Cool the room at natural rate */
        BSP_LED_OFF(3);
        BSP_LED_OFF(1);
        break;
    else |
        AppTempHeatCtrl = DEF_OFF;
    switch (AppTempState) |
    ```
Why not ‘all of the above’?

- Offload to a PC
- Leverage Windows reusable components
  - Meters, Gauges, Bargraphs
  - Sliders, Switches
  - Spreadsheets and Tables
  - Numeric indicators
  - Graphs and Plots
  - Virtual LEDs
  - Etc.
- Use available communication interfaces
  - J-Tag, RS232C, TCP/IP, USB, other
- Avoid writing custom code just to ‘see inside’ your target
  - Don’t burden the application code
- ...

Micrium
µC/Probe ... The Ideal Companion for the RX
What is µC/Probe?

- **µC/Probe** is a universal embedded system monitoring tool at run-time.
- Windows application to display & change target data at run-time:
  - Any variable
  - Any memory location
  - Any I/O port
- Works with **ANY** processor
  - 8-, 16-, 32-, 64-bit or DSP
- Works with **ANY** compiler
  - Compiler/linker needs to generate an ELF/DWARF file
- Target doesn’t need an RTOS
  - Works with or without an RTOS
μC/Probe Architecture

Your Code

RTOS
Or No-RTOS

Libraries

e²Studio
IAR
Compiler
Assembler
Linker

μC/Probe

.JELF

RTOS
Or
No-RTOS

Libraries

μC/Probe

JTAG
RS232C
Ethernet
USB
Etc.

Micrium

µC/Probe V3 - Thermostat Demo

<table>
<thead>
<tr>
<th>Task</th>
<th>Name</th>
<th>Period</th>
<th>Priority</th>
<th>Preempt</th>
<th>Elapsed Time</th>
<th>Memory Used</th>
<th>Code Size</th>
<th>Data Size</th>
<th>Task Priority</th>
<th>Task State</th>
<th>Task Flags</th>
<th>Task Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task1</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task2</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task3</td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task4</td>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task5</td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2012 Renesas Electronics America Inc. All rights reserved.
Multiple Interface Supported
µC/Probe Views

- **µC/Probe** users can switch between one of two different views

- **Design View** facilitates the development of a user interface
  - Graphical components can be dragged and dropped onto data screens

- In **Run-Time View**, the interface becomes active
  - **µC/Probe** updates components with values gathered from a running embedded system
μC/Probe – Design View

‘Ribbon’ containing options

‘Toolbox’ Containing Graphical objects

‘Data Screen’ where objects are placed

‘Workspace’ Listing Data Screens In Projects

‘Symbol Browser’ contain your application variables
µC/Probe – Design View

(1) – Load Symbol file (.ELF)

(2) – Drag and Drop Object

(3) – Assign the Object to a Symbol (OSTickCtr)

(4) – Select the Communication I/F

(5) – Run
μC/Probe – Run View

3647543
µC/Probe – Lab #1
Lab 1

- Board Setup

- Start µC/Probe and load the lab’s .ELF file

- Select J-LINK interface – 12000 KHz

- Assign OSTickCtr to Numeric Indicator
  - Run

- Assign a variable to a slider and a gauge
How does µC/Probe work?
How does µC/Probe work?

- Windows-based application with all the ‘smarts’
- Target code with as little ‘smarts’ as possible

Windows-based PC

- Reads .ELF from compiler/linker
- Rich set of visual objects

Embedded System

- J-Link or RS-232C or Ethernet

J-Link (Doesn’t require ANY target code)
RS-232C: Requires simple target agent
TCP/IP: Requires UDP/IP stack and simple UDP server
μC/Probe user assigns an object to a variable

- OSStatTaskCPUUsage is a 2-byte variable assigned RAM address 0x2000A31C by the linker.
**µC/Probe sends a ‘memory read’ to the target**

- In ‘Run’ mode, µC/Probe sends a command to read 2 bytes from location 0x2000A31C using the selected communication interface.
- The target responds with 720 but µC/Probe scales it by 0.01

“What’s the 2-byte value at address 0x2000A31C?”

“720″
µC/Probe – The Ribbon
The Ribbon

- Left Edges
- Horizontal Centers
- Right Edges
- Top Edges
- Vertical Centers
- Bottom Edges

- Bring Forward
- Bring To Front

- Send Backward
- Send To Back

- Group
- Ungroup

- Paste
- Clipboard

- Show/Hide Grid
- Show/Hide Rulers

- Full Screen
- Layout

- Units
- Zoom Factor

- Communication
- Endianness
- Little Endian
- Big Endian

- Interfaces
- J-Link

- Remote Host
- Remote Port

- General

- OK
- Cancel
The Ribbon

- Zoom In
- Zoom Out
- Pan
- Reset Panning
- Snap to Grid
μC/Probe – Advanced Objects
Text Tool

- Used to add ‘label’ to other objects
- Overlay on top or behind other objects like a gauge
Text Tool

- Used to add ‘numeric’ value
- Can be overlaid on top of another object like a gauge
Bitmap Animation

- Assigns bitmaps to numeric values
  - Change a picture based on an integer value
  - This is a great feature when used with state machines
- Can be overlaid on top or behind other objects
- For example:

<table>
<thead>
<tr>
<th>Value</th>
<th>Image</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(All OFF Test)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(State Machine Values)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>(All ON Test)</td>
<td></td>
</tr>
</tbody>
</table>
Kernel Awareness

- Built-in µC/OS-III awareness
  - Always present, even in FREE version
- Displays run-time information about your tasks:
Kernel Awareness ... continued

- More task related data
μC/Probe – Lab #2
Lab 2 – Advanced Controls

- Text Object
- Bitmap Animation
- Writable controls
  - Sliders
  - Push Button Switches
- µC/OS-III Kernel Awareness
μC/Probe – Lab #3
Lab 3 – Graphs

- Graph types
  - Marker chart
  - Line chart
  - Area chart

- Properties
  - X-Y Axis
  - Multiple symbols per-graph

- Stripchart mode

- Burst Mode
  - Sinewave array
µC/Probe - When would you use it?
Debugging

- **µC/Probe** can be used to supplement a conventional debugger
  - Especially helpful for detecting stack overflows – at run-time
- Kernel awareness aids in debugging **µC/OS-III**-based applications
  - Many per-task statistics: CPU usage, Stack usage, Interrupt Disable time, Scheduler lock time, context switch counter, etc.

<table>
<thead>
<tr>
<th>CPU Usage</th>
<th>CtxSwCtr</th>
<th>Interrupt Disable Time (Max)</th>
<th>Scheduler Lock Time (Max)</th>
<th>#Used</th>
<th>#Free</th>
<th>Size</th>
<th>Stack Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01 %</td>
<td>81</td>
<td>4.9</td>
<td>0.0</td>
<td>31</td>
<td>49</td>
<td>80</td>
<td>8.75 %</td>
</tr>
<tr>
<td>0.18 %</td>
<td>137</td>
<td>7.7</td>
<td>0.0</td>
<td>38</td>
<td>62</td>
<td>100</td>
<td>8.00 %</td>
</tr>
<tr>
<td>2.38 %</td>
<td>1,283</td>
<td>9.8</td>
<td>0.0</td>
<td>242</td>
<td>58</td>
<td>300</td>
<td>80.67 %</td>
</tr>
<tr>
<td>0.42 %</td>
<td>665</td>
<td>8.6</td>
<td>0.0</td>
<td>54</td>
<td>46</td>
<td>100</td>
<td>54.40 %</td>
</tr>
<tr>
<td>1.37 %</td>
<td>998</td>
<td>9.8</td>
<td>0.0</td>
<td>160</td>
<td>140</td>
<td>300</td>
<td>53.33 %</td>
</tr>
<tr>
<td>0.03 %</td>
<td>85</td>
<td>4.3</td>
<td>0.0</td>
<td>74</td>
<td>76</td>
<td>150</td>
<td>49.33 %</td>
</tr>
<tr>
<td>0.00 %</td>
<td>48</td>
<td>4.8</td>
<td>0.0</td>
<td>107</td>
<td>43</td>
<td>150</td>
<td>71.33 %</td>
</tr>
<tr>
<td>0.03 %</td>
<td>86</td>
<td>4.4</td>
<td>6.8</td>
<td>36</td>
<td>44</td>
<td>80</td>
<td>45.00 %</td>
</tr>
<tr>
<td>0.29 %</td>
<td>87</td>
<td>4.8</td>
<td>0.0</td>
<td>38</td>
<td>62</td>
<td>100</td>
<td>8.00 %</td>
</tr>
<tr>
<td>2.81 %</td>
<td>8,644</td>
<td>6.1</td>
<td>0.0</td>
<td>28</td>
<td>52</td>
<td>80</td>
<td>85.00 %</td>
</tr>
<tr>
<td>93.45 %</td>
<td>7,879</td>
<td>4.3</td>
<td>0.0</td>
<td>18</td>
<td>32</td>
<td>50</td>
<td>36.00 %</td>
</tr>
</tbody>
</table>
Demonstrations

- **µC/Probe** facilitates the development of eye-catching presentations

- No on-board display hardware is needed for a **µC/Probe**-based presentation

- Presentations are highly portable
In the Field

- µC/Probe is not solely a development tool

- Technicians can use µC/Probe to gather status information from a product

- Product performance is usually not affected by target-resident code
μC/Probe – Lab #4
Lab 4 – Save / Restore Workspaces + More Graphs

- Save / Restore Workspaces
- Linear Graph in Scope mode
µC/Probe - Free Evaluation – Contact Micriµm
Free Evaluation

- Two versions of µC/Probe are currently available from http://micrium.com/page/products/tools/probe
  - 5-symbols (variables), 5-objects with no time limit
  - Unlimited symbols and objects (licensed)

- The 5 symbols version allows the use of µC/OS-III kernel awareness

- The full version of µC/Probe is only $99 USD
Contact Information

- **Sales questions:**
  Robert Langley, Sales Representative
  Phone: +1 954 217 2036 Ext. 104
  E-mail: robert.langley@micrium.com

- **Other inquiries:**
  Matt Gordon, Sr. Applications Engineer
  Phone: +1 954 217 2036 Ext. 102
  E-mail: matt.gordon@micrium.com
Questions?
Please Provide Your Feedback...

- Please utilize the ‘Guidebook’ application to leave feedback

or

- Ask me for the paper feedback form for you to use...