User's Manual
V4.1

Micriμm®
Press

Weston, FL 33326

Micriμm
1290 Weston Road, Suite 306
Weston, FL 33326
USA

www.micrium.com

600-uC-Probe-008

Micriμm® Press

Table of Contents

Chapter

1

Introduction

µC/Probe is a Windows application designed to read and write the memory of any embedded target processor during run-time. Memory locations are mapped to a set of virtual controls and indicators placed on a dashboard. Figure 1-1 shows an overview of the system and data flow.

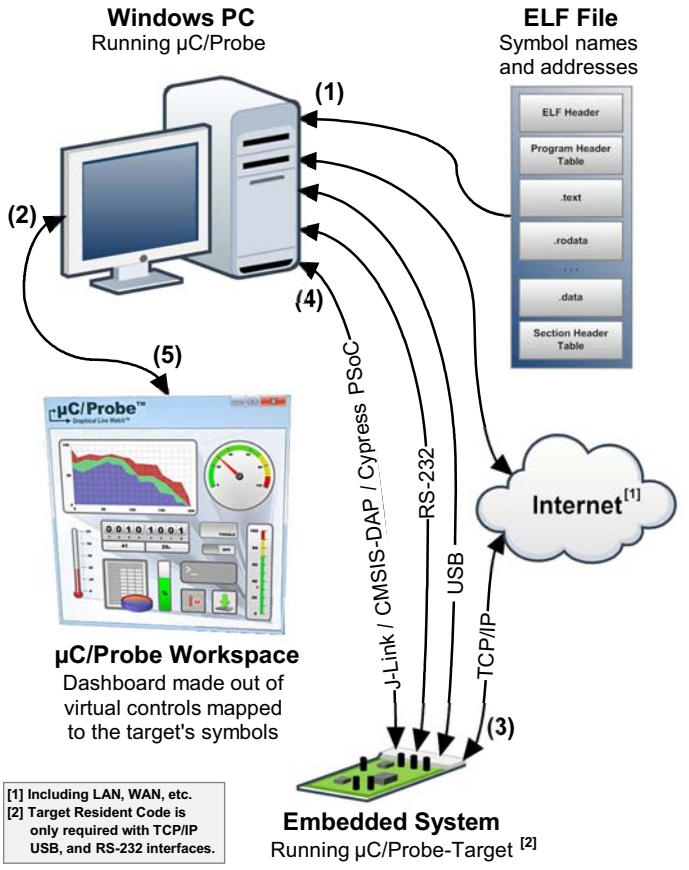


Figure 1-1 **µC/Probe Data Flow Diagram**

F1-1(1) You have to provide µC/Probe with an ELF file with DWARF-2, -3 or -4 debugging information. The ELF file is generated by your toolchain's linker. µC/Probe parses the ELF file and reads the addresses of each of the embedded target's symbols (i.e. global variables) and creates a catalog known as *symbol browser*, which will be used by you during design-time to select the symbols you want to display on your dashboard. Refer to the document µC/Probe Target Manual for more information on installing the µC/Probe Target C files and building the ELF file.

Alternatively, you can also provide a chip definition file that contains the chip's peripheral register addresses or provide your own custom XML based symbol file for those cases when your toolchain cannot generate one of the supported ELF formats.

F1-1(2) During design-time, you create a µC/Probe workspace using a Windows PC and µC/Probe. You design your own dashboard by dragging and dropping virtual controls and indicators onto a *data screen*. Each virtual control and indicator needs to be mapped to an embedded target's symbol by selecting it from the symbol browser. This document aims at providing more information on creating your own dashboard with µC/Probe.

F1-1(3) Before proceeding to the run-time stage, µC/Probe needs to be configured to use one of the following communication interfaces: J-Link, CMSIS-DAP, Cypress PSoC Prog, USB, RS-232 or TCP/IP. In order to start the run-time stage, you click the *Run* button and µC/Probe starts making requests to read the value of all the memory locations associated with each virtual control and indicator (i.e. buttons and gauges respectively). At the same time, µC/Probe sends commands to write the memory locations associated with each virtual control (i.e. buttons on a click event).

F1-1(4) In the case of a reading request, the embedded target responds with the latest value. In the case of a write command, the embedded target responds with an acknowledgement. Refer to the document µC/Probe Target Manual for more information on all you need in regards to the firmware that implements the communication interface that runs on the embedded target.

F1-1(5) µC/Probe parses the responses from the embedded target and updates the virtual controls and indicators.

In case the communication of your choice is USB, RS-232 or TCP/IP, refer to the document µC/Probe Target Manual for more information about the firmware that resides on the Embedded System.

This document only provides information about the Windows PC side of the system.

Chapter

2

µC/Probe System Overview

This section provides an overview of the µC/Probe Windows Application.

Whenever you start µC/Probe in your Windows PC, three different modules are started: µC/Probe Automatic Updates and Licensing System, µC/Probe Data Client and µC/Probe Generic Target Communications Module as illustrated in Figure 2-1:

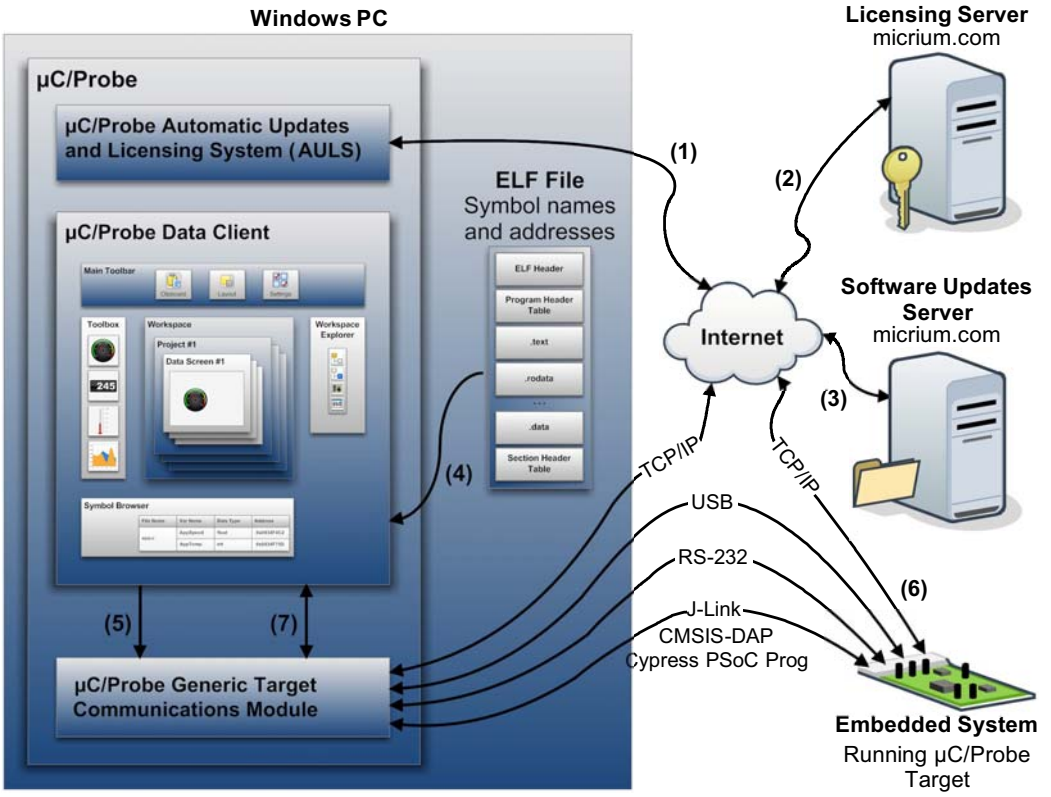


Figure 2-1 **µC/Probe System Overview**

F2-1(1) The Automatic Updates and Licensing System (AULS) is the part of the application that allows you to install and keep your µC/Probe application up to date.

F2-1(2) The Educational Edition of µC/Probe is deployed with some of the Professional Edition's features and do not require internet access to activate the software application. The Professional Edition of µC/Probe requires internet access to validate the license key provided by your Micriμm's sales representative.

For more information on µC/Probe Licensing see Appendix L, "Licensing" on page 174.

F2-1(3) All Editions of µC/Probe are self-updating and every time you start the application, if internet access is available, the µC/Probe AULS module checks for newer versions of µC/Probe from the Micriμm website and as they become available, the µC/Probe AULS module, automatically replaces any updated files.

F2-1(4) The µC/Probe Data Client is the part of the application that allows you to design your dashboard (design-time mode) and run it (run-time mode).

The next section in this document provides more information in regards to using the µC/Probe Data Client during design-time and run-time.

F2-1(5) The µC/Probe Generic Target Communications Module is the part of the application that connects directly with the Embedded Target and responds to the requests from the Data Client.

When the µC/Probe run-time mode gets started, the Data Client sends requests to the Generic Target Communications Module. The requests contain not only the embedded target communication settings but also all the symbol's memory address required by your dashboard design.

F2-1(6) The Generic Target Communications Module takes the request from the µC/Probe Data Client and initiates a communication with the embedded target through the configured communication interface.

F2-1(7) The µC/Probe Data Client exchanges requests to read and write the memory locations required by the current view of your dashboard's design with the embedded target through the Generic Target Communications Module.

2-1 µC/PROBE DATA CLIENT

The µC/Probe Data Client is illustrated in more detail in Figure 2-2:

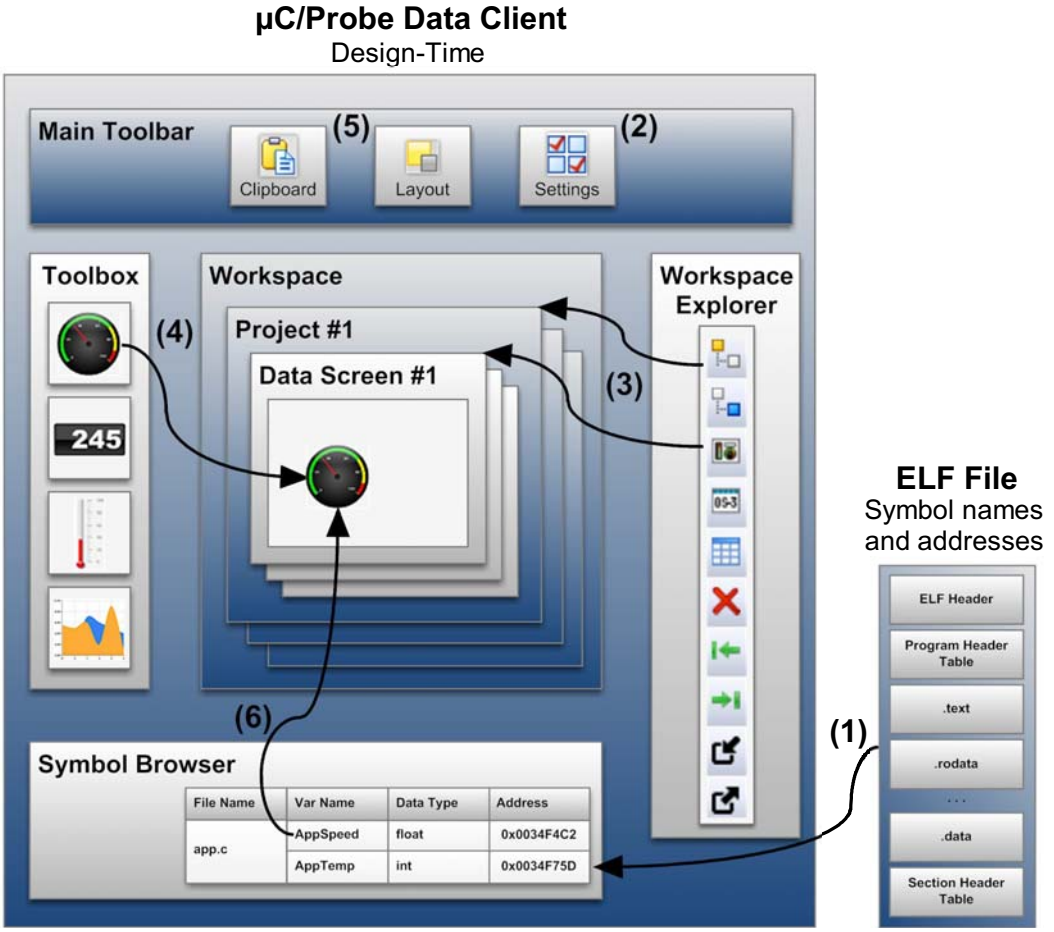


Figure 2-2 **µC/Probe Data Client: Design Time**

F2-2(1) The µC/Probe Data Client is the part of the application that during design-time takes the ELF file with DWARF-2, -3 or -4 debugging information. As previously discussed, the ELF file is generated by your toolchain's linker. The µC/Probe Data Client parses the ELF file and reads the addresses of each of the embedded target's symbols (i.e. global variables) and creates a catalog known as symbol browser, which will be used by you during design-time to select the symbols you want to display on your dashboard. Refer to the document µC/Probe Target Manual for more information on installing the µC/Probe Target C files and building the ELF file.

For more information, see Chapter 3, "µC/Probe Symbol Browser" on page 16.

F2-2(2) During design-time it is necessary to adjust the communication and other general settings. See Chapter 4, "µC/Probe Settings" on page 31 for more information on configuring µC/Probe.

F2-2(3) The Workspace Explorer in the µC/Probe Data Client allows you to add or delete Projects and Data Screens among other things.

For more information, see Chapter 5, "µC/Probe Workspace Explorer" on page 50.

F2-2(4) The µC/Probe Toolbox displays icons for the virtual controls and indicators that you can add to your Data Screens. Each toolbox icon can be dragged and dropped onto the Data Screen to build your own dashboard.

For more information, see Chapter 6, "µC/Probe Toolbox" on page 53.

F2-2(5) The µC/Probe Layout Design Tools help you arrange the virtual controls and indicators on your data screen by speeding up the creation of your dashboard and making it look great.

For more information, see Chapter 7, "µC/Probe Layout Design Tools" on page 60.

F2-2(6) The last step during design-time is to map each virtual control and indicator in your Data Screen with an Embedded Target's memory location. The symbol browser allows you to quickly find the variable you want to display and then all you have to do is drag the variable from the symbol browser and drop it onto the virtual control or indicator of your choice.

See Chapter 8, "Associating Symbols to Virtual Controls and Indicators" on page 63 for more information on using the symbol browser to map virtual controls and indicators to the embedded target's memory locations.

The actual µC/Probe windows application is shown in Figure 2-3:

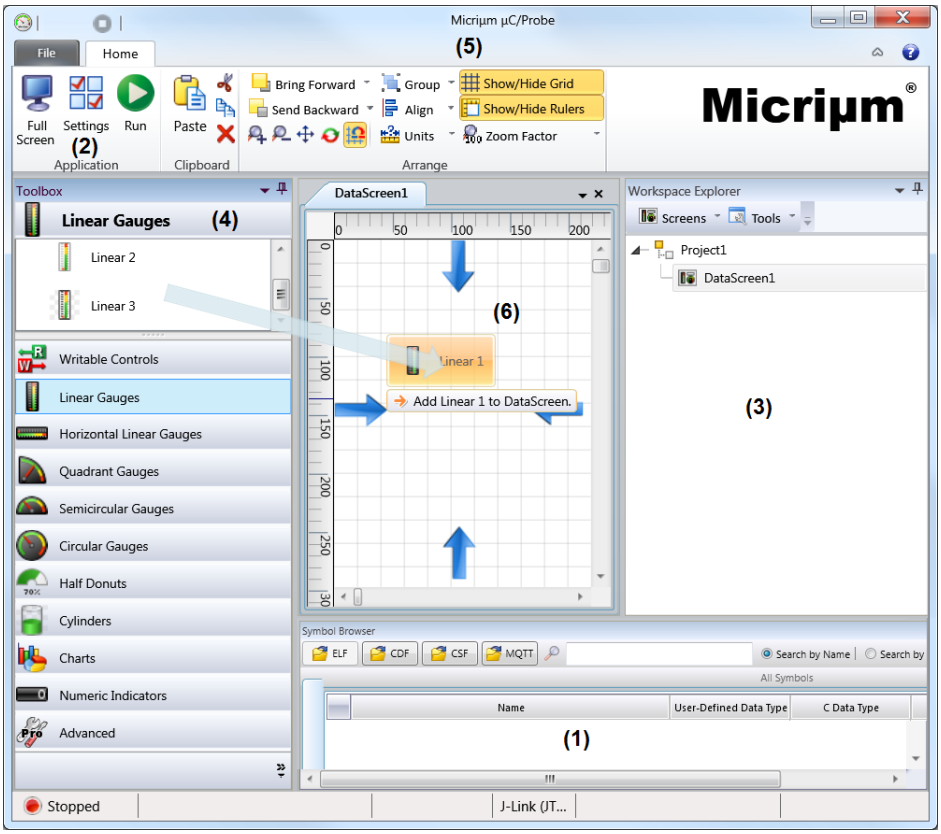


Figure 2-3 **µC/Probe Windows Application**

F2-3(1) Chapter 3, "µC/Probe Symbol Browser" on page 16.

F2-3(2) Chapter 4, “µC/Probe Settings” on page 31.

F2-3(3) Chapter 5, “µC/Probe Workspace Explorer” on page 50.

F2-3(4) Chapter 6, “µC/Probe Toolbox” on page 53.

F2-3(5) Chapter 7, “µC/Probe Layout Design Tools” on page 60.

F2-3(6) Chapter 8, “Associating Symbols to Virtual Controls and Indicators” on page 63.

Chapter

3

µC/Probe Symbol Browser

The µC/Probe's Symbol Browser is a list of your embedded target's symbols that helps you quickly find the symbol you want to use in your data screen. The symbol browser is available during design-mode and it is located at the bottom of the application window.

There are four types of symbol files supported by µC/Probe's parser:

- ELF (Executable and Linkable Format).
- CDF (Chip Definition File).
- CSF (Custom Symbol File).
- MQTT Configuration File (Message Queueing Telemetry Transport).

To load any of these files, you start by clicking one or more of the buttons indicated in Figure 3-1:

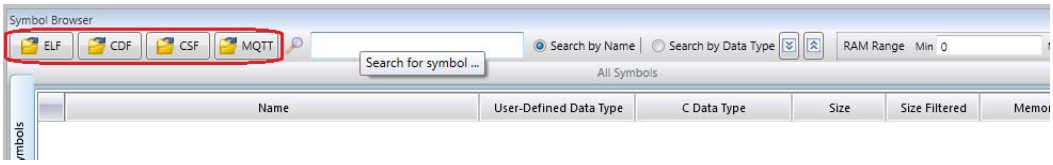


Figure 3-1 **µC/Probe Symbol Browser: Loading a symbols file**

The following sections will describe each of the four types of symbol files supported by µC/Probe. Keep in mind that you only require to load at least one of the following files.

3-1 ELF FILE

This file is the output from your compiling and linking process. It contains the name, data type and address of all your global variables.

3-1-1 LOADING AN ELF FILE

In order to provide µC/Probe with the path of the embedded target's output file (ELF file), locate and click the button labeled *ELF File* indicated in Figure 3-1.

µC/Probe will prompt for the ELF file path by using a standard open file dialog. Locate the output file in your PC. The path is usually configured from your toolchain's linker configuration.

The ELF file needs to have symbolic information for debugging purposes in the DWARF-2, -3 or -4 format.

3-1-2 BROWSING THE ELF FILE

µC/Probe parses the ELF file and creates a catalog that you can browse to search for the symbol you need. The symbol browser is a five-column tree list of symbols grouped by the C file name where the variable was declared as shown in Figure 3-2:

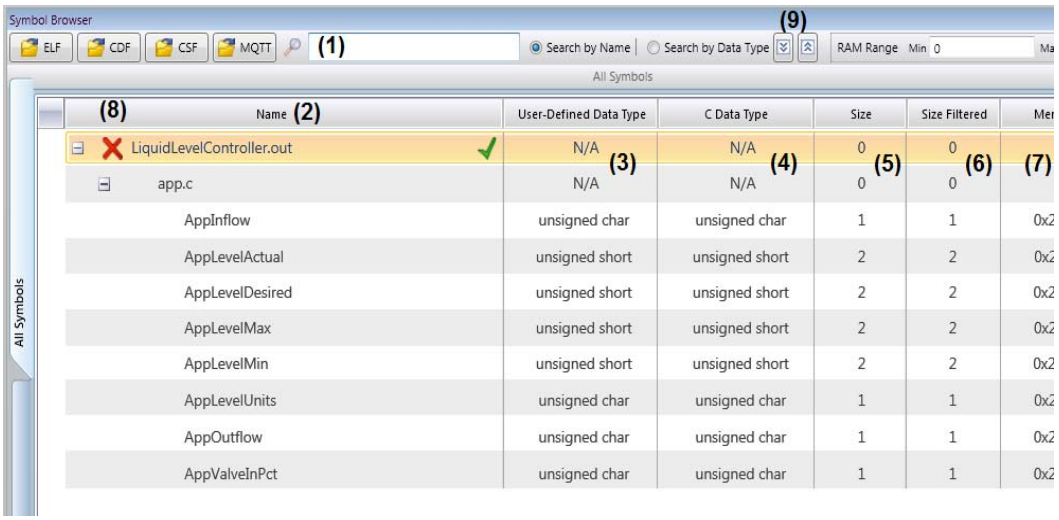


Figure 3-2 **µC/Probe Symbol Browser: Symbols grouped by C file**

F3-2(1) The symbol browser allows you to quickly find the symbol you want. Click on the symbol browser headers row to sort the list by the column you want. You can also expand and collapse tree nodes to focus on a particular C file, or you can use the search box and search by symbol name or data type.

F3-2(2) The **Name** column shows the name of the symbol as declared in your C file.

F3-2(3) The **User-Defined Data Type** column displays the name of the symbol's data type as declared by your code.

F3-2(4) The **C Data Type** column displays the symbol's C data type.

F3-2(5) The **Size** column displays the size in bytes of the symbol.

F3-2(6) The **Size Filtered** column displays the size in bytes of the symbol as seen by the data footprint calculator.

F3-2(7) The **Memory Address** column displays the symbol's location in the embedded target's memory.

F3-2(8) Click on the red **X** next to the name of the ELF file, to remove a symbol file from the symbol browser.

F3-2(9) The *expand all* and *collapse all* buttons allow you to browse more efficiently throughout the symbol browser tree.

Be aware that the symbol browser in µC/Probe will detect if the ELF file has been re-compiled and will refresh all addresses. However, in case you move the ELF file to a different location in your file system, µC/Probe cannot update the addresses automatically. Instead, you can update the symbol browser with the new path by first removing the ELF file (red X next to the filename) and then opening the new ELF file.

3-2 CDF FILE

The CDF (Chip Definition File) contains the name, data type and address of all your device's I/O registers. µC/Probe installs in your PC with a very large catalog of chip definition files that includes chips from semiconductors such as Silicon Labs, Infineon, Analog Devices, Atmel, Cypress, NXP, Renesas, ST, Texas Instruments and Xilinx among others.

3-2-1 LOADING A CDF FILE

You can browse the CDF catalog and select your platform's chip definition file by clicking the button *CDF File* and using the CDF browser shown in Figure 3-3:

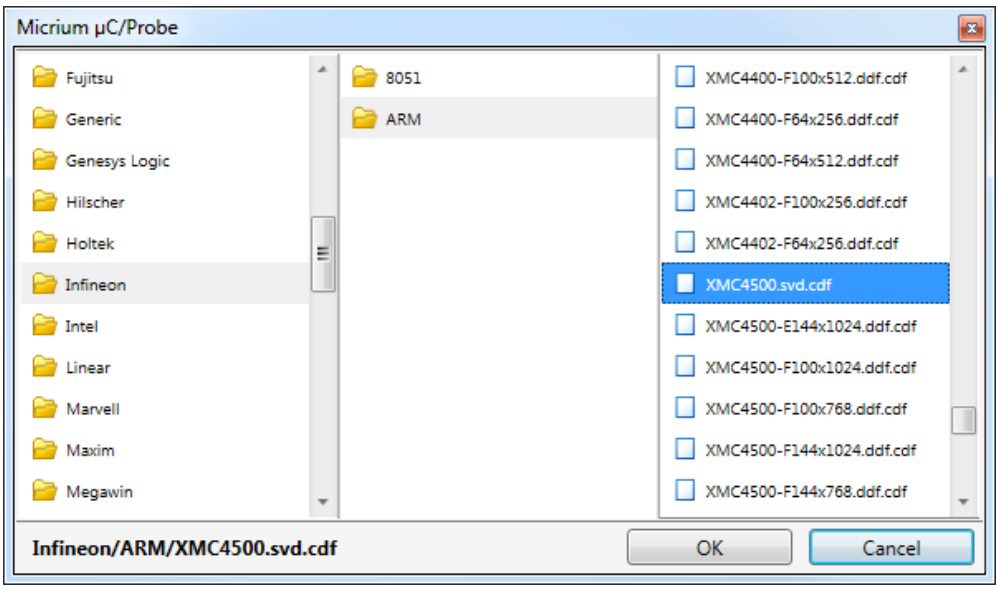


Figure 3-3 **Chip Definition Files Browser**

The CDF files provide the chip's peripheral I/O register names and addresses which you can use to associate with controls such as the *Bit Control* described in section A-2-8 "Bit Control Properties Editor" on page 90.

3-3 CSF FILE

µC/Probe is capable of parsing XML-based Custom Symbol Files (CSF), which is very useful for those cases where your toolchain is incapable of generating one of the ELF file formats supported by µC/Probe.

3-3-1 CREATING A CSF FILE

The best way to create a CSF file is by modifying the template located in your µC/Probe installation directory at:

```
$\Micrium\uC-Probe\Templates\uC-Probe-CSF-Template.csf
```

The template is associated with an XSD document that defines the XML schema for CSF files supported by µC/Probe. We recommend using an XML editor capable of providing *IntelliSense* features such as Visual Studio, shown in Figure 3-4:

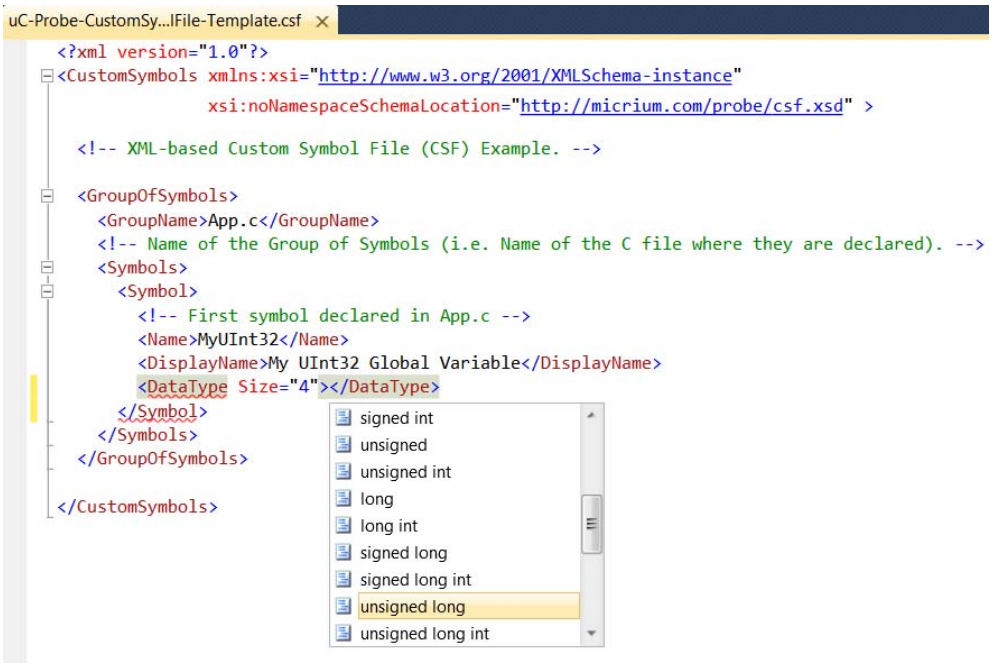


Figure 3-4 **Creating a CSF in Visual Studio: Drop-Down Lists**

Visual Studio makes editing your CSF file easier by filling required XML syntax for you. For example, after a schema is associated with your CSF, you get a drop-down list of expected elements any time you type "<".

When you type *SPACE* from inside a start tag, you also get a drop-down list showing all attributes that can be added to the current element.

Likewise, when you type "=" for an attribute value, or the opening quote for the value, you also get a list of possible values for that attribute.

Moreover, *ToolTips* appear on these IntelliSense lists giving you a description of each element as illustrated in Figure 3-5:

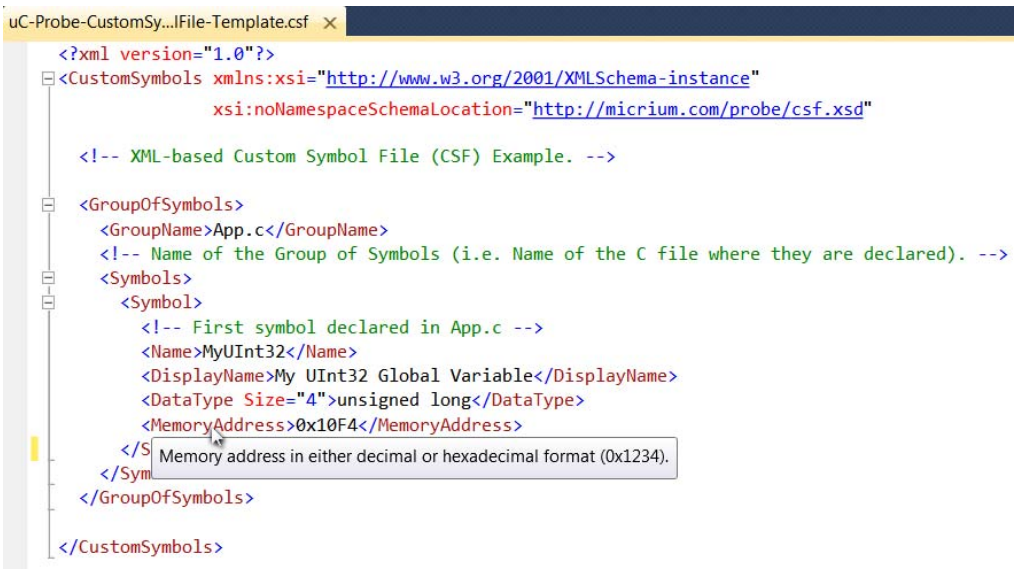


Figure 3-5 **Creating a CSF in Visual Studio: Tool Tips**

Custom Symbol Files need to have the extension `.csf` for µC/Probe to recognize them as such. Listing 3-1 shows an example of a CSF file that declares one integer, one array and one data structure.

```
<?xml version="1.0"?>
<CustomSymbols xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:noNamespaceSchemaLocation="http://micrium.com/probe/csf.xsd" >          (1)
  <GroupOfSymbols>                                                                        (2)
    <GroupName>App.c</GroupName>                                                          (3)
    <Symbols>
      <Symbol>
        <Name>MyUInt32</Name>                                                             (4)
        <DisplayName>My UInt32 Global Variable</DisplayName>                              (5)
        <DataType Size="4">unsigned long</DataType>                                       (6)
        <MemoryAddress>0x10F4</MemoryAddress>                                             (7)
      </Symbol>
      <Symbol>
        <Name>MyStruct</Name>
        <DataType Size="50">struct</DataType>
        <MemoryAddress>0x50AC</MemoryAddress>
        <DataMembers>                                                                     (8)
          <Symbol>
            <Name>MyArray</Name>
            <DataType Size="16" IsArray="true" ArrayLength="4">int</DataType>             (9)
            <MemoryAddress>0x50AC</MemoryAddress>
          </Symbol>
          <Symbol>
            <Name>MyCharPointer</Name>
            <DataType Size="4" IsPointer="true">char</DataType>                          (10)
            <MemoryAddress>1025</MemoryAddress>
          </Symbol>
        </DataMembers>
      </Symbol>
    </Symbols>
  </GroupOfSymbols>
</CustomSymbols>
```

Listing 3-1 **XML-based Custom Symbol File Example**

L3-1(1) The root element is called `<CustomSymbols>`. It includes the XSD schema reference to help you editing in an XML editor such as Visual Studio.

L3-1(2) Each symbol or group of symbols are within the element tag `<GroupOfSymbols>`. In this case it is used with the purpose of creating a group of symbols declared in one single C file.

L3-1(3) The name of the group of symbols is `App.c`.

L3-1(4) The name of the first symbol in the group as declared in C.

L3-1(5) You can also specify an alias for the group name for display purposes.

L3-1(6) The element tag `<DataType>` is the ANSI C data type of the variable including the size in bytes as an attribute.

L3-1(7) The element tag `<MemoryAddress>` is the variable's memory address in either decimal or hexadecimal format (0x1234).

L3-1(8) For more complex symbols such as data structures, there is a tag called `<DataMembers>` that allows you to specify a group of symbols that make part of a data structure.

L3-1(9) In order to declare an array, you need to specify three attributes: A boolean flag that indicates that the symbol is an array, the total number of bytes and the number of elements in the array.

L3-1(10) Finally, any data type can be declared as a pointer by using the data type boolean attribute `IsPointer.` In this case, it is the intention to specify a symbol declared as `char *.`

In order to verify your CSF file, you can use the Symbol Browser from within µC/Probe as shown in Figure 3-6. Notice the relationship between the XML tags and the tree nodes in the Symbol Browser.:

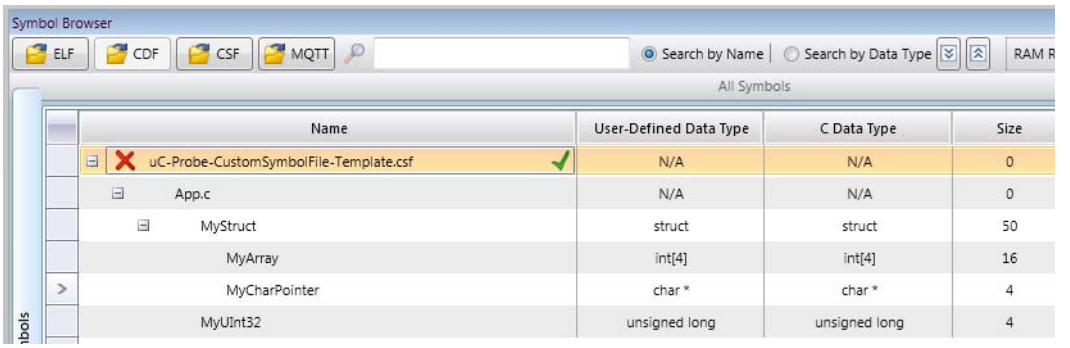


Figure 3-6 **XML-based Custom Symbol File Example as seen from µC/Probe's Symbol Browser**

3-4 MQTT CONFIGURATION FILE

The MQTT (Message Queueing Telemetry Transport) protocol is becoming the de facto standard for IoT (Internet of Things) applications.

µC/Probe can be used to build an MQTT Client. If you have an embedded system that is MQTT-ready, you can monitor and control your embedded system remotely by using any of the virtual controls and indicators in µC/Probe's toolbox.

Let's take for example an embedded systems-based *Weather Station* that remotely publishes via MQTT its readings to an MQTT broker as illustrated in Figure 3-7.

The MQTT broker stores the readings and then you can use µC/Probe to create a GUI that displays the readings using some of the virtual indicators (e.g. thermometer to display temperature) and configures the Weather Station with some of the virtual controls (e.g. slider control to configure the sampling rate).

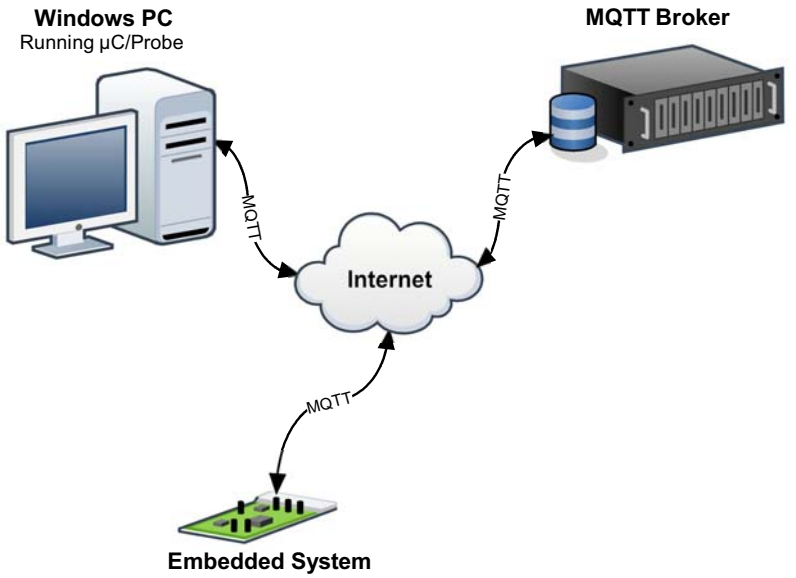


Figure 3-7 **µC/Probe as an MQTT gateway**

The details on how to setup an MQTT broker and how to make an embedded systems MQTT-ready are beyond the scope of this document.

However, a good place to start is looking at 2lemetry as an MQTT broker and Micriμm's μC/TCP-IP and μC/MQTTc as the firmware you need to make your embedded systems MQTT-ready.

The next section will explain how to create an MQTT configuration file assuming that you already have an account and configuration setup at the MQTT broker of your choice.

3-4-1 CREATING AN MQTT CONFIGURATION FILE

Similar to the Custom Symbol File (CSF) in section 3-3 "CSF File" on page 20, the best way to create an MQTT configuration file is by modifying the template located in your μC/Probe installation directory at:

```
$\Micrium\uC-Probe\Templates\uC-Probe-MQTT-CfgFile-Template.mqtt
```

The template is associated with an XSD document that defines the XML schema for MQTT configuration files supported by µC/Probe. We recommend using an XML editor capable of providing *IntelliSense* features such as Visual Studio, shown in Figure 3-4:

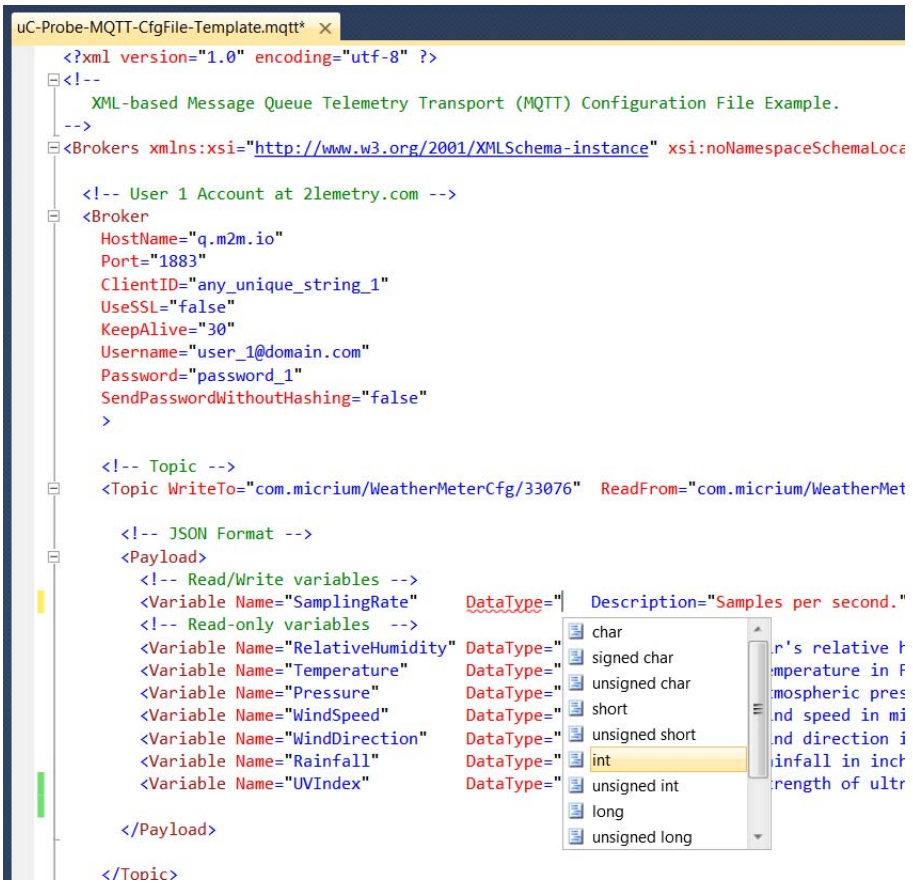


Figure 3-8 **Creating an MQTT Configuration File in Visual Studio: Drop-Down Lists**

Visual Studio makes editing your MQTT configuration file easier by filling required XML syntax for you. For example, after a schema is associated with your MQTT configuration, you get a drop-down list of expected elements any time you type "<".

When you type *SPACE* from inside a start tag, you also get a drop-down list showing all attributes that can be added to the current element.

Likewise, when you type "=" for an attribute value, or the opening quote for the value, you also get a list of possible values for that attribute.

Moreover, *ToolTips* appear on these IntelliSense lists giving you a description of each element as illustrated in Figure 3-5 when the user hovers the mouse over the Topic's **WriteTo** property:

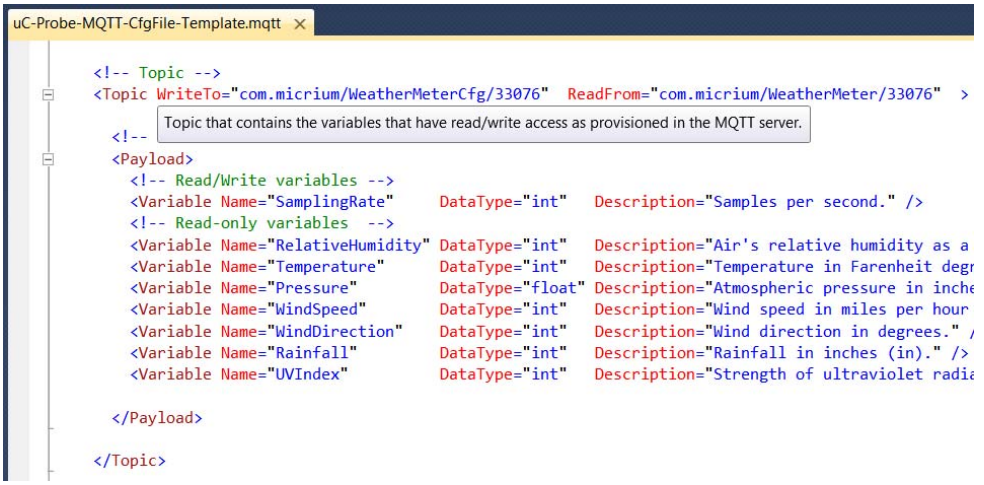


Figure 3-9 **Creating an MQTT Configuration File in Visual Studio: Tool Tips**

MQTT Configuration Files need to have the extension `.mqtt` for µC/Probe to recognize them as such. Listing 3-1 shows an example of an MQTT Configuration File that declares a few variables.

```
<?xml version="1.0" encoding="utf-8" ?>
<!--
   XML-based Message Queue Telemetry Transport (MQTT) Configuration File Example.
-->
<Brokers xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="http://micrium.com/probe/mqtt.xsd">                  (1)
  <!-- User 1 Account at 2lemetry.com -->
  <Broker
    HostName="q.m2m.io"
    Port="1883"
    ClientID="any_unique_string_1"
    UseSSL="false"
    KeepAlive="30"
    Username="user_1@domain.com"
    Password="password_1"
    SendPasswordWithoutHashing="false">                                                     (2)
    <!-- Topic -->
    <Topic WriteTo="com.micrium/WeatherMeterCfg/33076"
           ReadFrom="com.micrium/WeatherMeter/33076"  >                                     (3)
      <!-- JSON Format -->
      <Payload>                                                                            (4)
        <!-- Read/Write variables -->
        <Variable Name="SamplingRate"
                  DataType="int"
                  Description="Samples per second." />                                    (5)
        <!-- Read-only variables  -->
        <Variable Name="RelativeHumidity"
                  DataType="int"
                  Description="Air's relative humidity as a percentage (%)." />
        <Variable Name="Temperature"
                  DataType="int"
                  Description="Temperature in Fahrenheit degrees (F)." />
      </Payload>
    </Topic>
  </Broker>
</Brokers>
```

Listing 3-2 **XML-based MQTT Configuration File Example**

L3-2(1) The root element is called `<Brokers>`. Inside this tag, you can have more than one MQTT broker in case your data comes from various sources. The tag includes the XSD schema reference to help you editing in an XML editor such as Visual Studio.

L3-2(2) Each MQTT broker needs to be configured with the access credentials and other settings specific to the account you have with your MQTT broker.

L3-2(3) MQTT brokers are usually provisioned with two topics per device (thing). In this example, the weather meter at zip code 33326 is provisioned with two topics; One that contains the Read/Write access variables and the other one that contains the Read-only access variables. And if you want added security you can even create a separate account for Read/Write access.

L3-2(4) The data to be sent needs to be in JSON format. The tag `<Payload>` specifies the JSON format expected by both the MQTT broker and µC/Probe.

L3-2(5) The tag `<Variable>` allows you to specify the name, data type and an optional description for each variable within a topic.

In order to verify your MQTT configuration file, you can use the Symbol Browser from within µC/Probe as shown in Figure 3-10. Notice the relationship between the XML tags and the tree nodes in the Symbol Browser.:

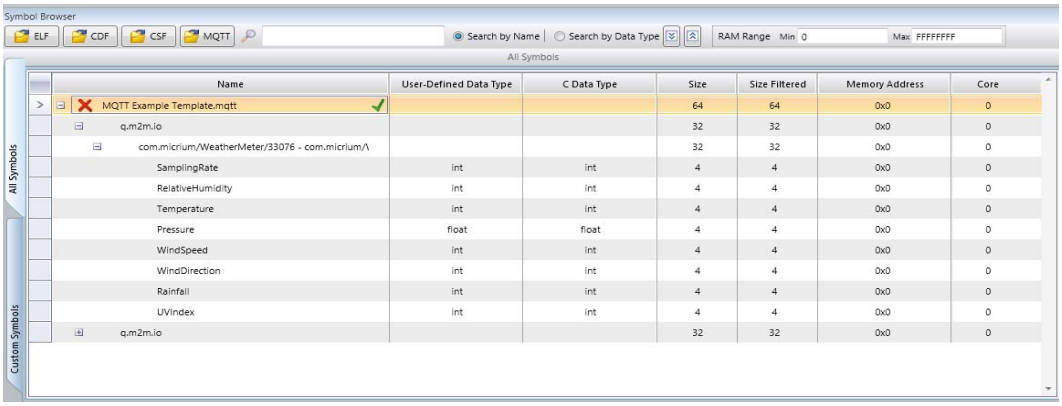


Figure 3-10 **XML-based MQTT Configuration File Example as seen from µC/Probe's Symbol Browser**

3-5 DATA FOOTPRINT CALCULATOR

The Data Footprint Calculator sums the space in RAM taken by your embedded target symbols. You simply go to the Symbol Browser and enter the address range as configured in your linker settings and µC/Probe will calculate the totals in the column labeled as **Size Filtered** as illustrated in Figure 3-11. In other words, any symbol that falls into this address range will be used in the calculation.

The reason why the column labeled as **Size** shows footprint amounts larger than those in the column **Size Filtered** is because some of the symbols in your ELF file do not reside in RAM. That is the case with symbols declared as constants which your linker places in ROM (code space).

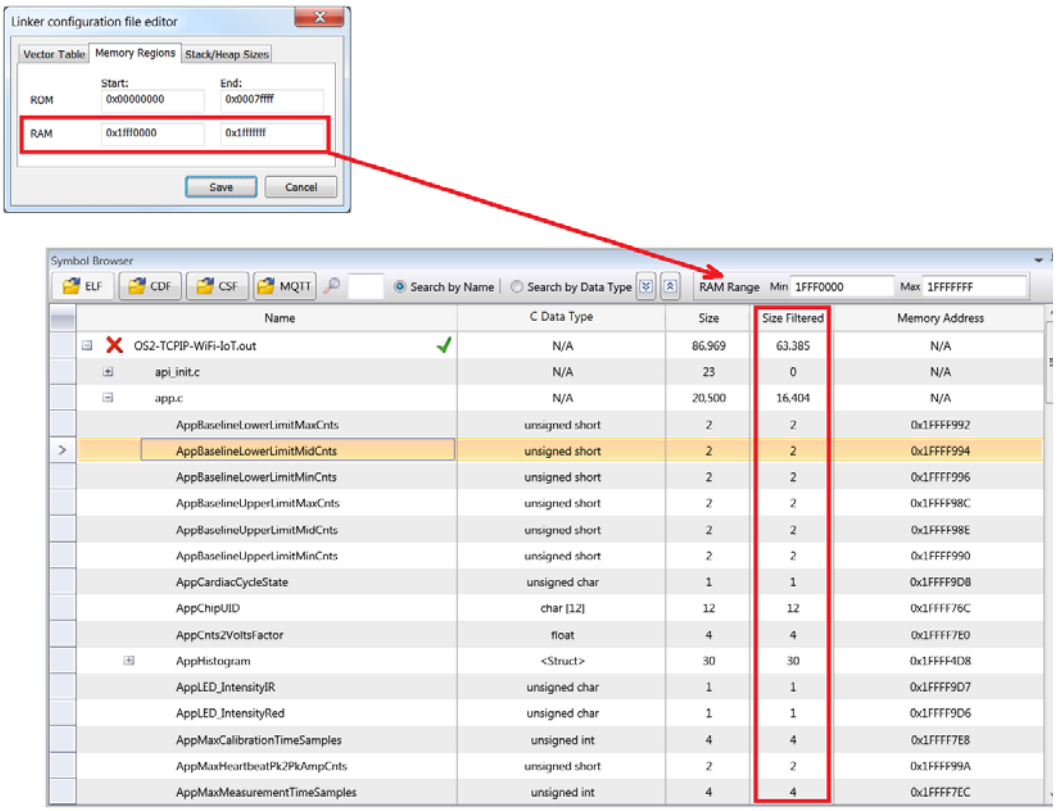


Figure 3-11 **Data Footprint Calculator**

Chapter

4

µC/Probe Settings

The µC/Probe application's tool bar is located at the top of the application window.

The µC/Probe Settings window is opened by making click on the **Settings** button in the application's tool bar as indicated in Figure 4-1:

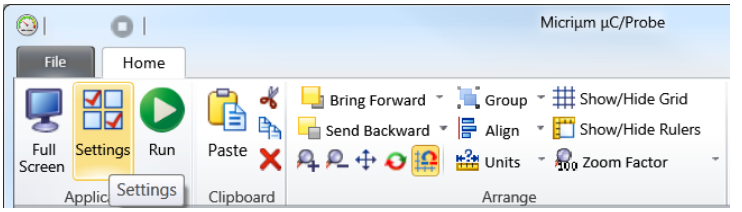


Figure 4-1 **µC/Probe Toolbar: Settings**

The settings window is divided in the following categories:

- Host Application:

  Here is where you get to configure the settings related to the Windows PC host application including your proxy server, automatic software updates, statistics and graphics hardware acceleration.

- Embedded Target:

  Here is where you get to configure the settings related to the embedded target including the endianness type and the communication interface settings.

4-1 HOST APPLICATION SETTINGS

The Host Application Tab allows you to configure the settings for the µC/Probe Windows Application.

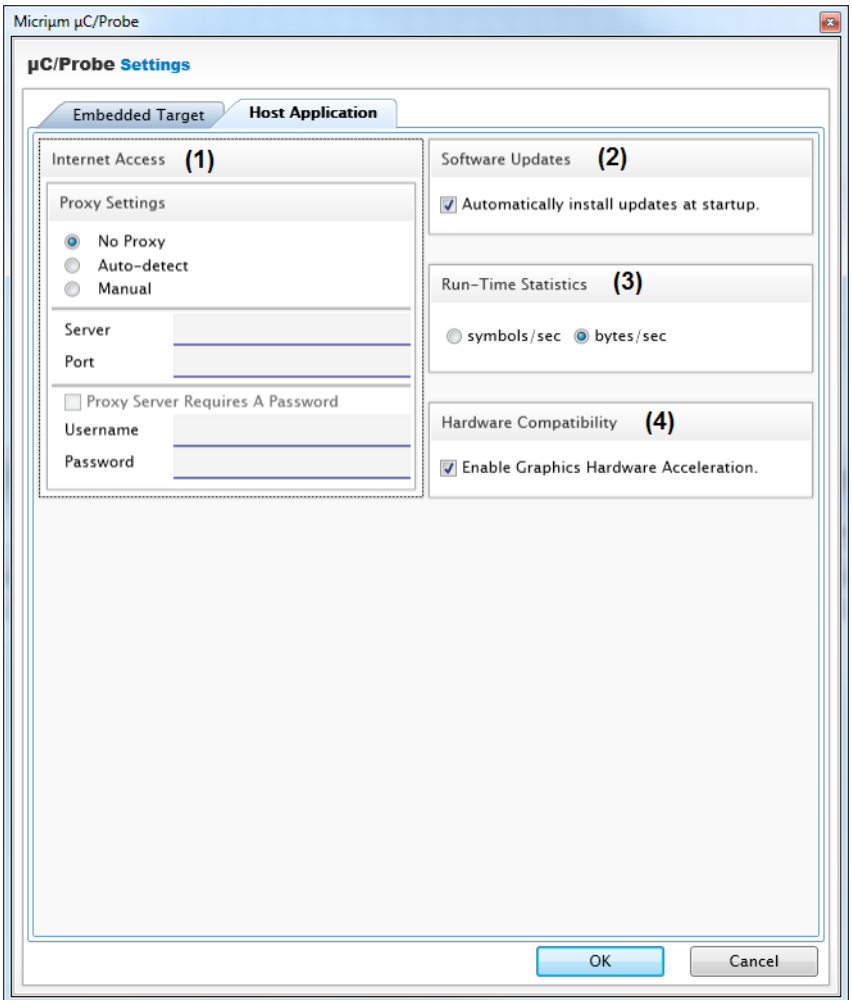


Figure 4-2 **µC/Probe Communication Settings - Host Application**

F4-2(1) Internet Access is required once to activate your license key and optionally, every time you launch the windows application to check for any software updates. Depending on your network setup, you may require a proxy server to access the Internet. Here is where you can configure the proxy server.

F4-2(2) At startup, µC/Probe checks for any available software updates through the Internet. You can disable this function from here.

F4-2(3) During run-time, µC/Probe calculates the speed of your connection and displays the value in either symbols per second or bytes per second.

F4-2(4) µC/Probe's chart libraries use your Windows PC graphics hardware acceleration features. Sometimes however, compatibility issues may arise depending on the type of your Windows PC's graphics hardware. Here, you can disable the use of hardware acceleration in case your charts are not displayed properly.

The following sections describe how to configure each communication interface. Refer to the document µC/Probe-Target Manual for more information on the communication interface supported by the embedded target.

4-2 EMBEDDED TARGET SETTINGS

µC/Probe supports a variety of communication interfaces that can be classified in four groups:

- Debugger-based Interfaces
- Peripheral-based Interfaces
- 3rd. Party Plugins
- MQTT Interface

The following sections will describe each of the groups.

4-2-1 DEBUGGER-BASED INTERFACES

µC/Probe can interface with your embedded target via standard debugging tools such as:

- J-Link
- CMSIS-DAP

- Cypress PSoC Prog
- ARM Keil µVision
- Analog Devices Cross Core Embedded Studio
- OpenSDA

When using one of these interfaces, µC/Probe reads and writes your global variables in a non-intrusive way and without the need for any target resident code. These are the ideal interface options if you require to maintain the real-time behavior of your application.

Follow the decisions tree diagram in Figure 4-3 to see if you can use one of these debugger-based interfaces with your platform.

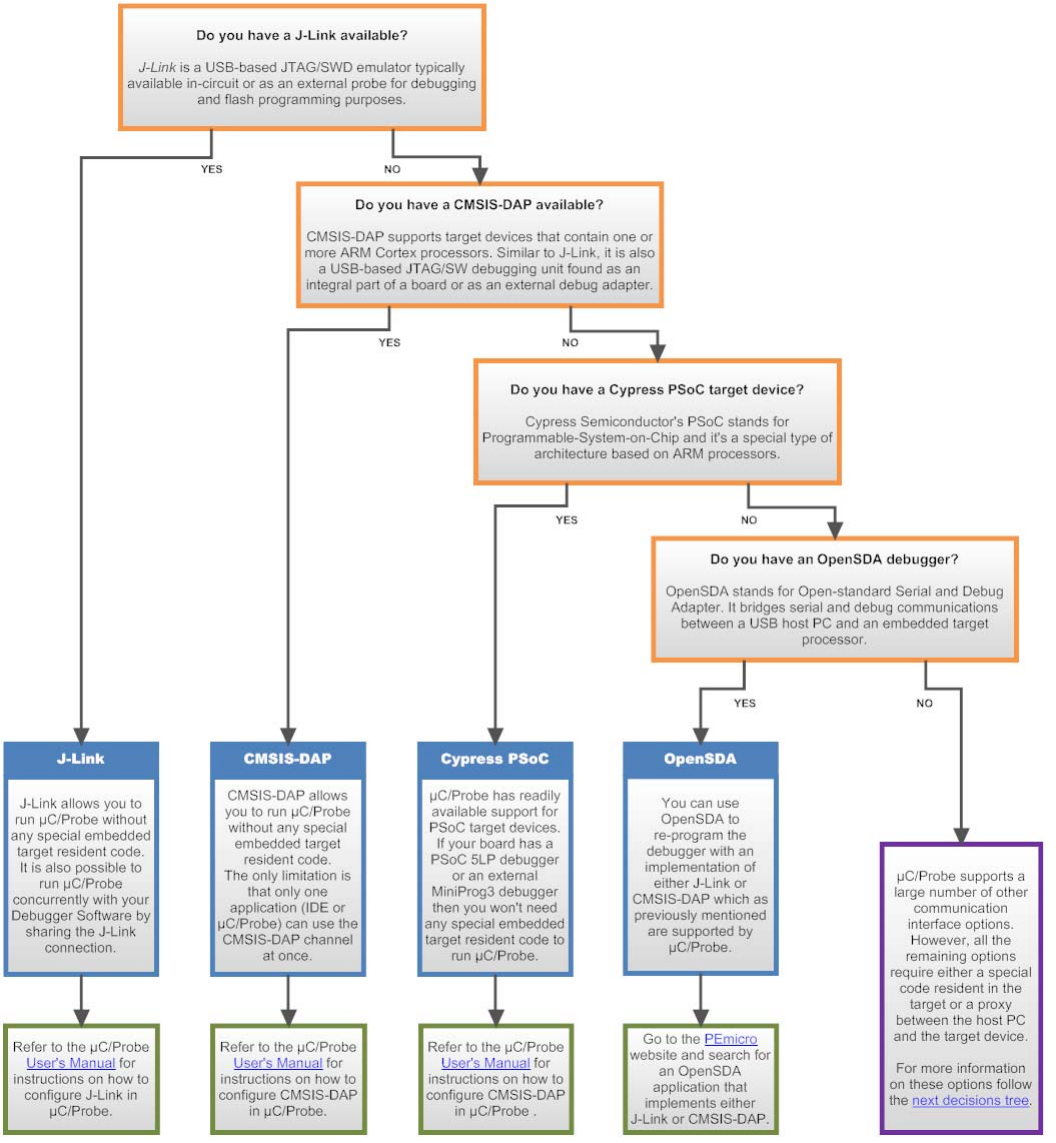


Figure 4-3 **Debugger-based Interfaces**

4-2-2 PERIPHERAL-BASED INTERFACES

µC/Probe can interface via the following communication modules:

- USB
- TCP/IP
- RS-232

These interfaces however, require a special target resident code that implements the µC/Probe protocol.

Micriμm has RS-232 drivers for various platforms available for free and if you are a µC/USBD or µC/TCP-IP licensee then µC/Probe is ready to run on these stacks.

Follow the decisions tree diagram in Figure 4-4 to see if you can use one of these interfaces with your platform.

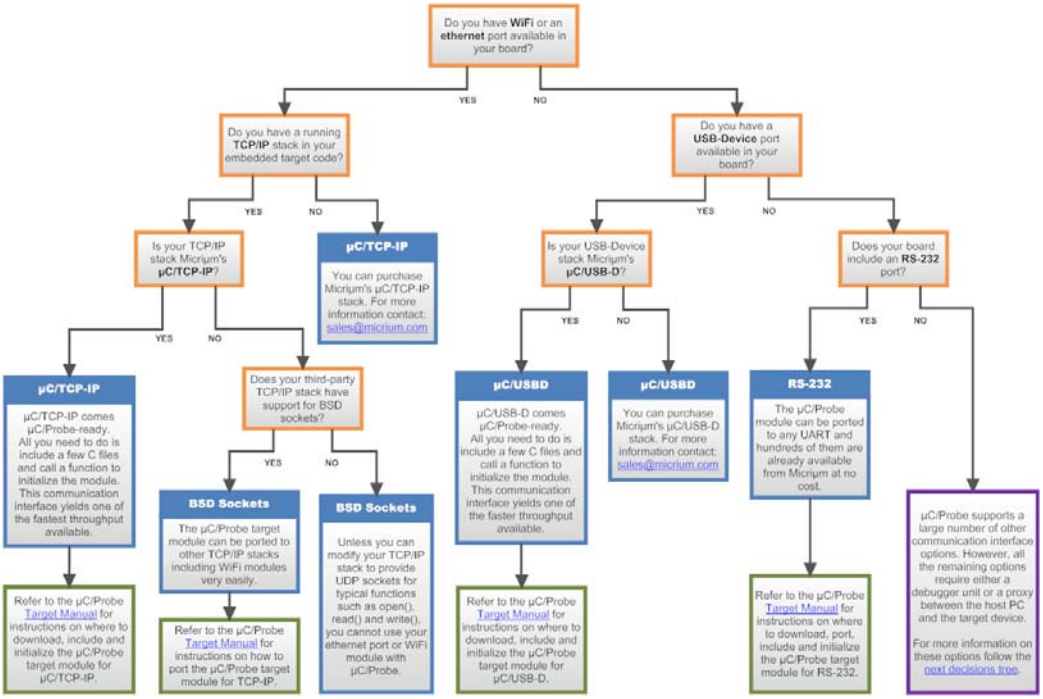


Figure 4-4 **Peripheral-based Interfaces**

4-2-3 THIRD PARTY PLUGINS

µC/Probe can interface with the embedded target via your IDE thanks to a public API that enables third party software applications to establish a TCP/IP bridge between the IDE and µC/Probe.

The TCP/IP bridge is created in the form of a plugin and gives µC/Probe access to as many platforms as the IDE supports.

Examples of these plugins are:

- IAR Systems C-SPY plugin for µC/Probe
- Eclipse plugin for µC/Probe

Follow the decisions tree diagram in Figure 4-5 to see if you can use one of these plugins with your platform.

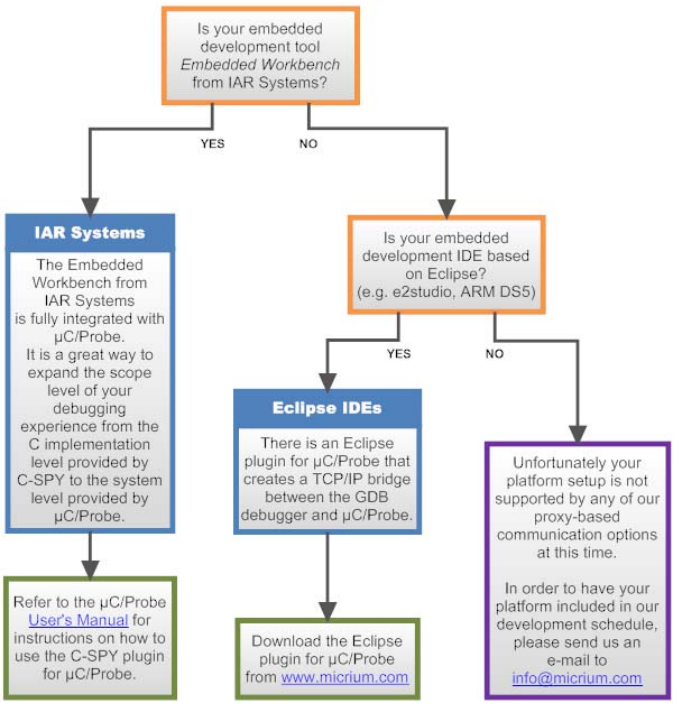


Figure 4-5 **Third Party Plugins**

4-2-4 MQTT INTERFACE

µC/Probe can also interface remotely to your embedded target via MQTT.

MQTT is becoming the de facto protocol for IoT applications and µC/Probe can be configured to be an MQTT client for those embedded systems that do not have neither TCP/IP connectivity nor intelligence that implements the MQTT protocol.

4-3 COMMUNICATION SETTINGS WINDOW

Once you have chosen the appropriate communication interface for your platform, you can configure µC/Probe from the communication settings window shown in Figure 4-6:

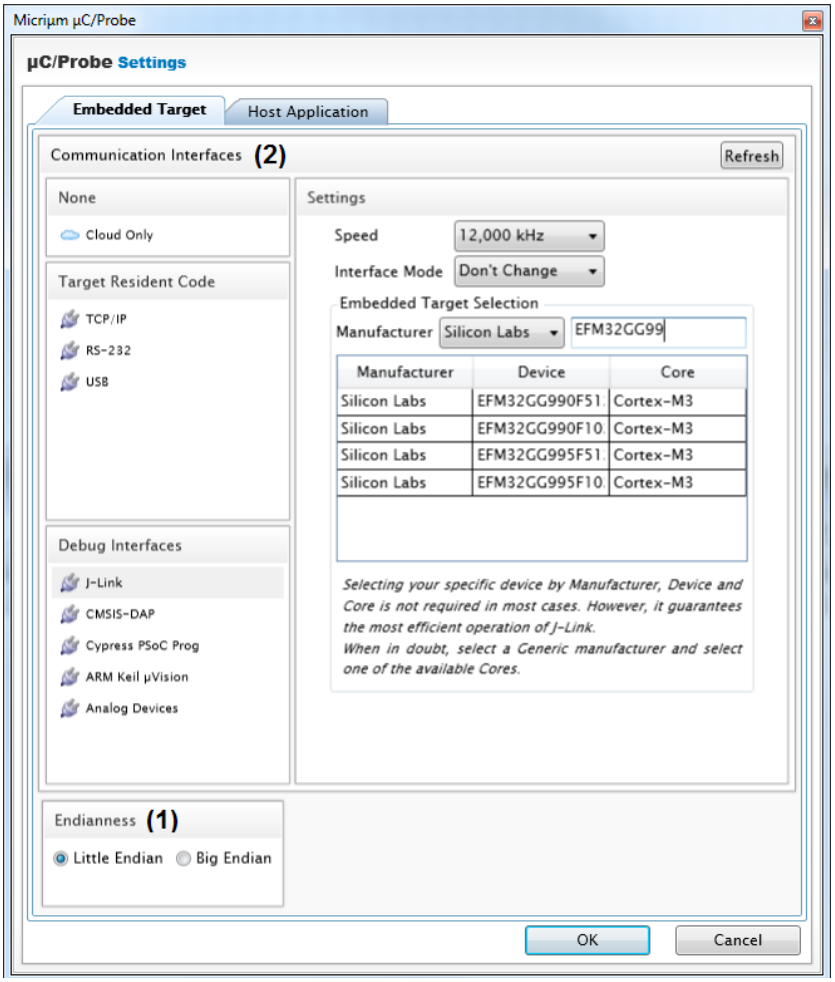


Figure 4-6 **µC/Probe Communication Settings - Embedded Target**

F4-6(1) µC/Probe can be configured to interpret the byte order in either little or big endian, select the option that corresponds to your target's endianness type.

F4-6(2) µC/Probe supports MQTT, J-Link, CMSIS-DAP, Cypress PSoC Prog, ARM Keil µVision, Analog Devices Cross Core Embedded Studio, USB, TCP/IP and RS-232. Select the interface that your target supports, and configure the settings corresponding to the interface.

4-3-1 SEGGER J-LINK

J-Link is a USB powered JTAG emulator designed by Segger. In order to install the windows drivers for J-Link (J-Link DLL) go to Segger's website at www.segger.com and download the J-Link software pack for Windows.

J-Link is the most popular emulator for ARM cores and it does not require any special code resident in the embedded target to connect with µC/Probe.

If using J-Link, you can interface µC/Probe even with a bare-metal application running no kernel at all, as shown in Figure 4-7:

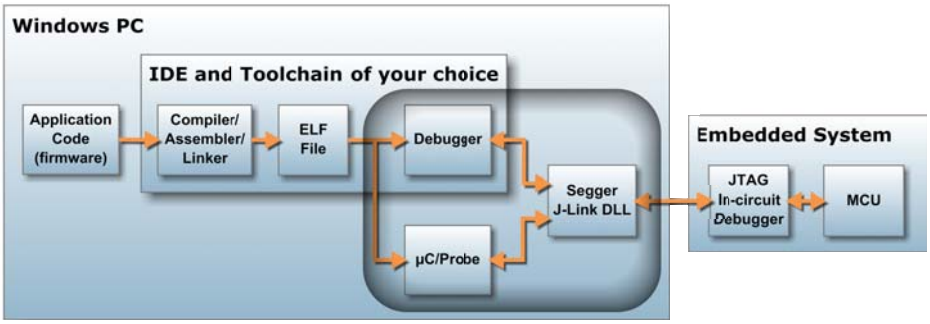


Figure 4-7 **µC/Probe via J-Link**

µC/Probe supports two types of J-link interface modes; JTAG and SWD. Select the interface mode from the radio buttons, configure the J-Link speed from the horizontal slider and select the embedded target device as shown in Figure 4-8. µC/Probe will negotiate the speed you configured and if your device does not support it, then it will select the maximum possible.

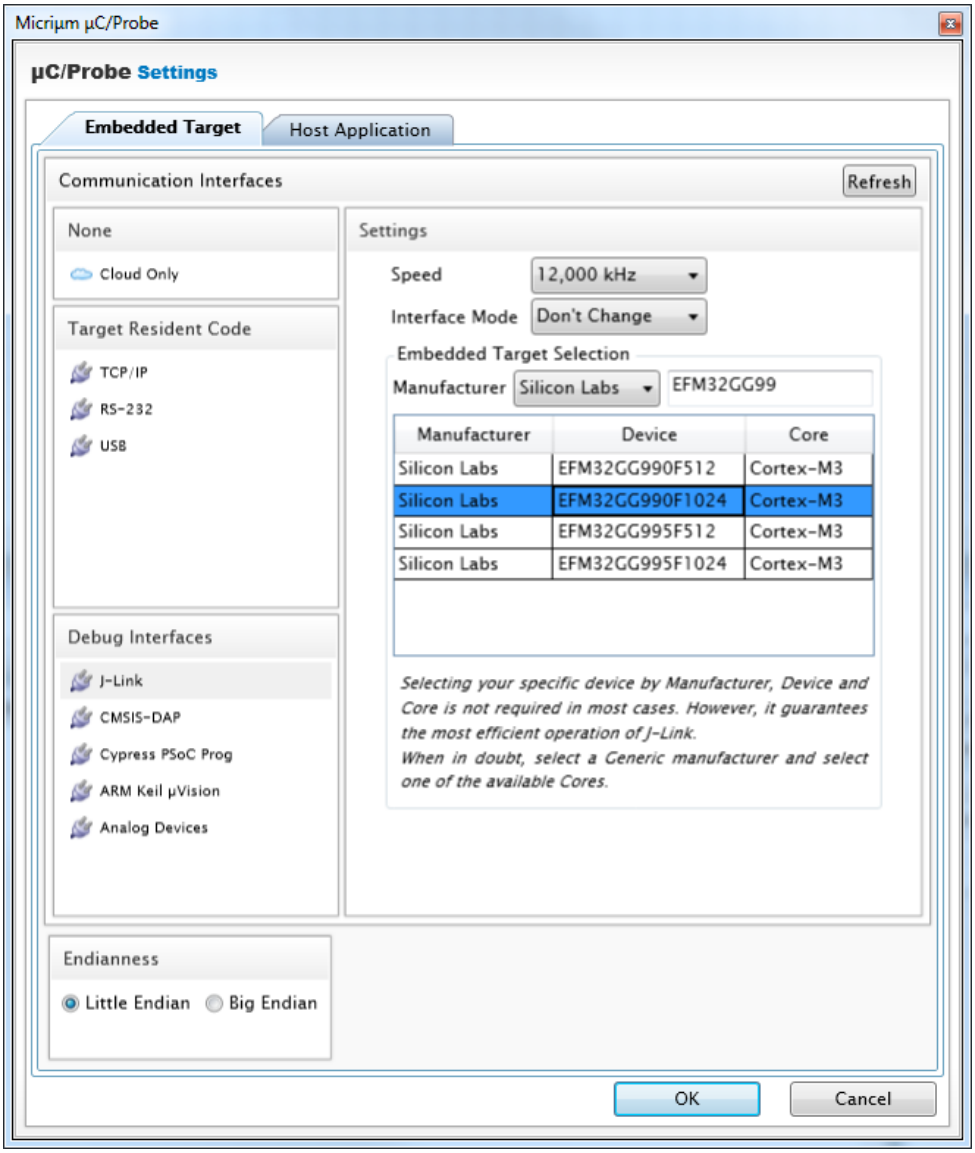


Figure 4-8 **µC/Probe Communication Settings: J-Link**

4-3-2 CMSIS-DAP

CMSIS-DAP is the interface firmware for an ARM Cortex processor's Debug Unit that connects the Debug Port to USB. µC/Probe which executes on a host computer, connects via USB to the Debug Unit and to the device that runs the application software. The Debug Unit connects via JTAG or SW to the target device.

The CMSIS-DAP interface does not require you to install any drivers, simply connect a USB cable between your board debug port and your Windows PC and then select the CMSIS-DAP interface from the Settings window as shown in Figure 4-9:

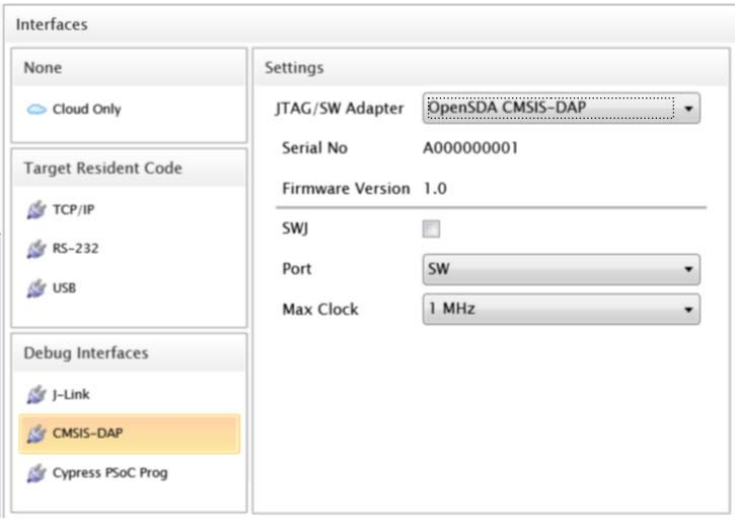


Figure 4-9 **µC/Probe Communication Settings: CMSIS-DAP**

Keep in mind that the CMSIS-DAP interface supports the communication of one single client at a time. In other words, you cannot run µC/Probe and your debugger software at the same time.

4-3-3 CYPRESS PSOC PROG

Cypress have their own debugging interface for their PSoC devices called PSoC Programmer. The interface allows you to not only program and configure the PSoC device but also to debug it. µC/Probe is tightly integrated with Cypress PSoC 5LP devices through this interface.

To use this interface with your Cypress PSoC 5LP device you need to download and install PSoC Programmer from the Cypress website at: `http://www.cypress.com`

Then you simply connect the board and select the PSoC Prog interface from the Settings menu as shown in Figure 4-10:

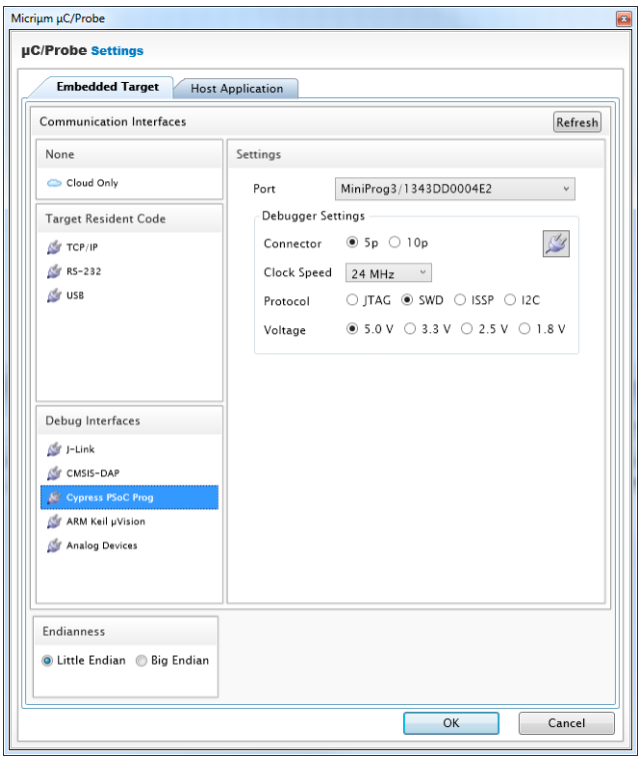


Figure 4-10 **µC/Probe Communication Settings: Cypress PSoC Prog**

Keep in mind that the Cypress PSoC Prog interface supports the communication of one single client at a time. In other words, you cannot run µC/Probe and your debugger software at the same time.

4-3-4 ARM KEIL µVISION

µC/Probe is tightly integrated with ARM Keil µVision and µC/Probe can read and write memory locations from and to embedded targets on a debug session via µVision. This integration means that µC/Probe can interface with any embedded target supported by Keil µVision without requiring any special target resident code and without any embedded target's CPU intervention.

To configure this interface you simply browse to the installation folder of your µVision and select the executable as shown in

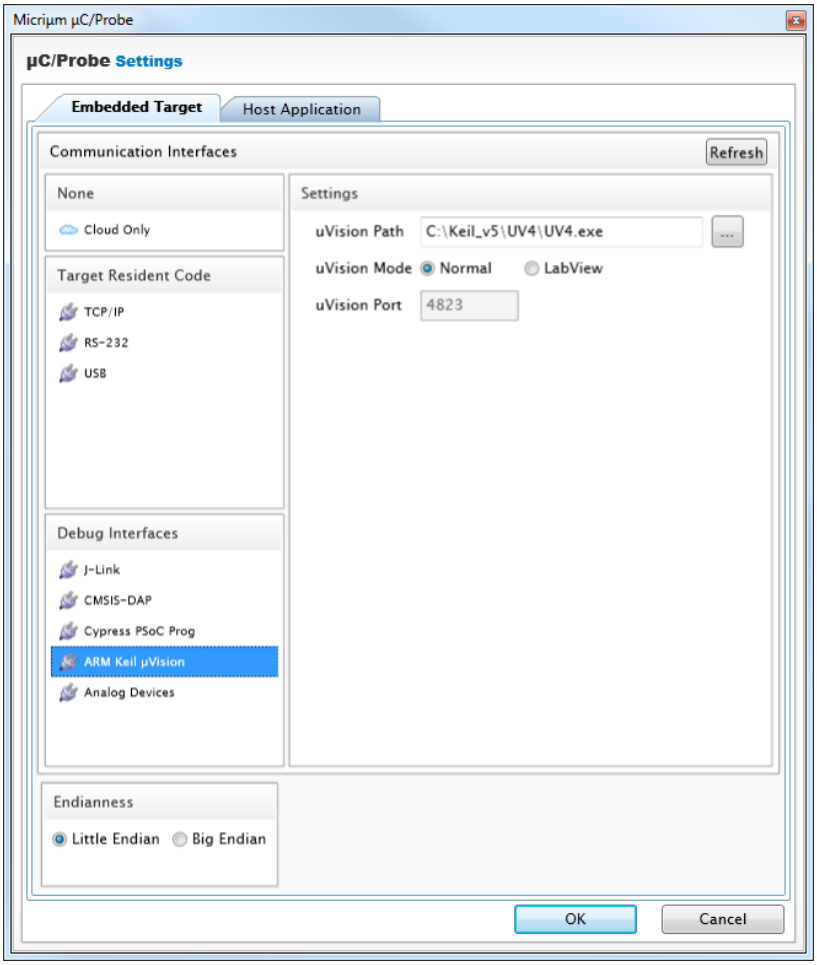


Figure 4-11 **µC/Probe Communication Settings: ARM Keil µVision**

4-3-5 ANALOG DEVICES CROSS CORE EMBEDDED STUDIO

Cross Core Embedded Studio is the IDE from Analog Devices. It includes a toolchain and debugger and µC/Probe is integrated to it via a named pipe server, meaning that µC/Probe can interface with any embedded target supported by Cross Core Embedded Studio without requiring any special target resident code and without any embedded target's CPU intervention.

To configure this interface you simply need to select it from the settings window and µC/Probe will confirm with a big green checkbox that your version of Cross Core Embedded Studio supports the pipe server as shown in Figure 4-15:

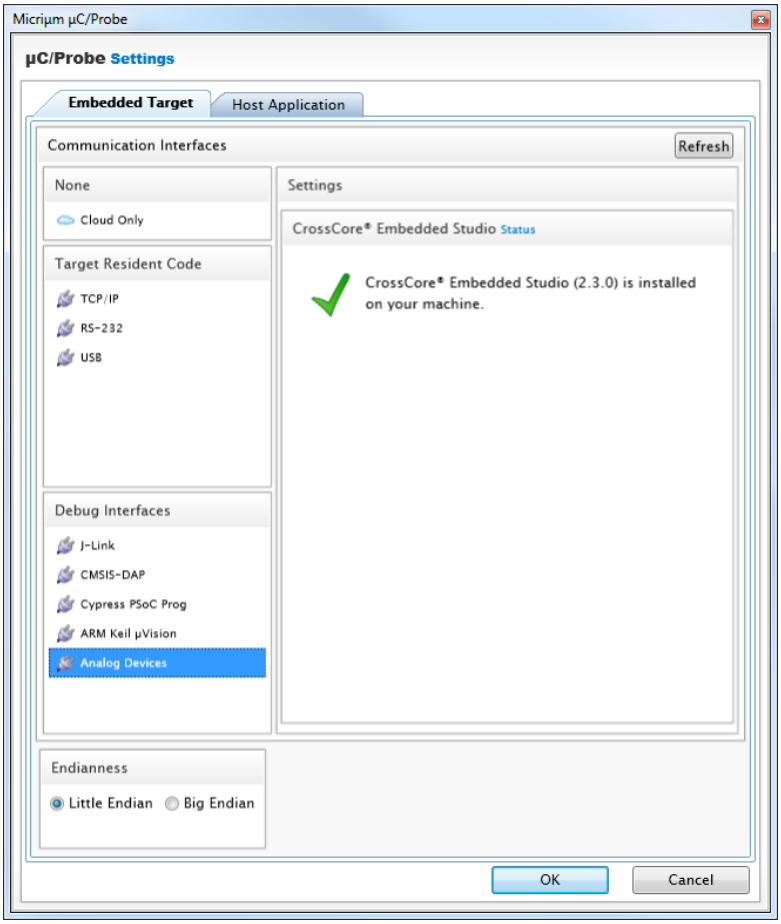


Figure 4-12 **µC/Probe Communication Settings: Analog Devices CCES**

4-3-6 USB

µC/Probe supports a USB interface over the µC/USB Device stack by Micriµm. This USB interface requires µC/Probe-Target code resident in your embedded system and because of the nature of USB, it also requires a kernel. Micriµm supports many cores and most likely this code is available from Micriµm. Contact Micriµm to find out if resident code for your particular setup is available.

Once your embedded system is running µC/Probe-Target as described in the document µC/Probe-Target Manual, the device should be ready to connect after plugging in. The Windows computer will enumerate the device and will display it as one of the available devices in the communication settings window as shown in Figure 4-13:

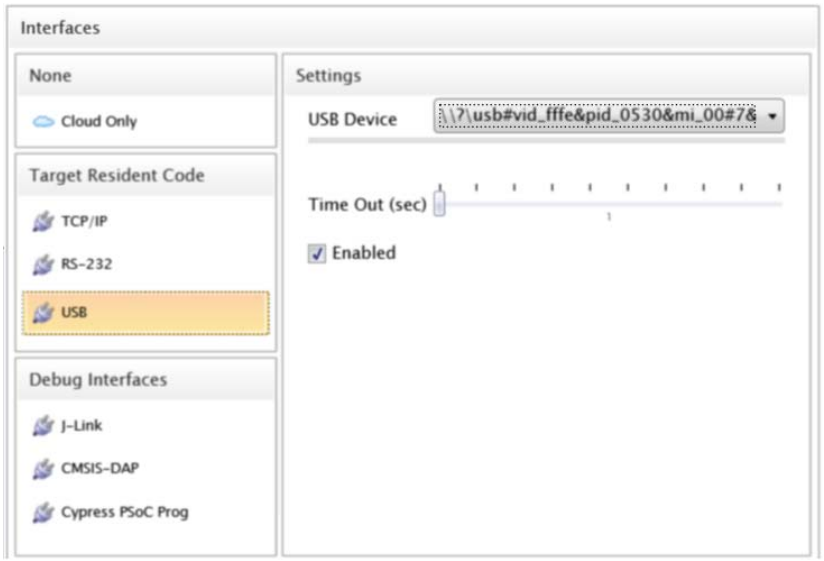


Figure 4-13 **µC/Probe Communication Settings: USB**

You can specify an optional timeout in seconds, which is the time µC/Probe is willing to wait for the target to respond before presenting an error message.

4-3-7 TCP/IP

µC/Probe supports a TCP/IP interface over the UDP protocol. The target requires a TCP/IP stack that provides a BSD sockets interface. Regardless of the TCP/IP stack being used, this interface requires µC/Probe-Target code resident in your embedded system and because of the nature of TCP/IP, it also requires a kernel. Micriμm supports many cores and most likely this code is available from Micriμm. Contact Micriμm to find out if resident code for your particular setup is available.

Assuming your embedded system is running µC/Probe-Target as described in the document µC/Probe-Target Manual, enter the IP address and port number of your embedded system in the text boxes shown in Figure 4-14:

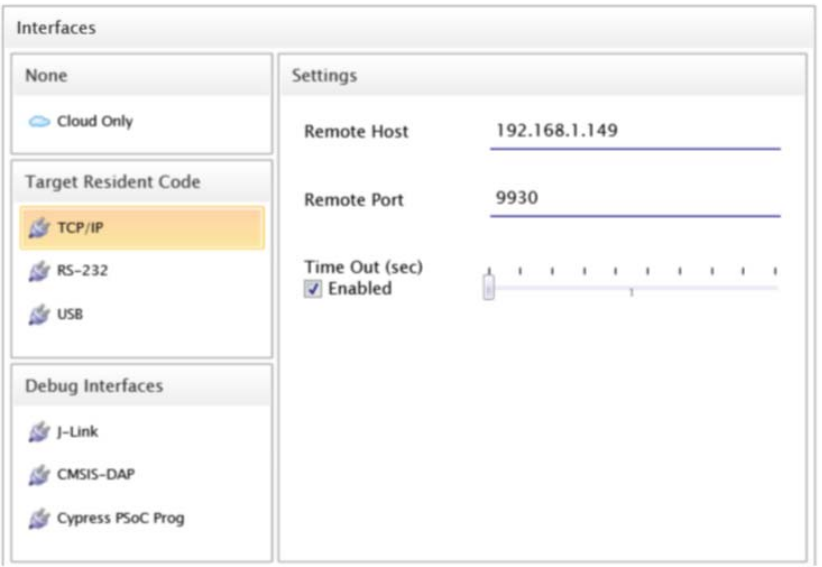


Figure 4-14 **µC/Probe Communication Settings: TCP/IP**

You can specify an optional timeout in seconds, which is the time µC/Probe is willing to wait for the target to respond before presenting an error message.

The TCP/IP interface is also used to interface through third-party plugin proxies such as the IAR Systems C-SPY plugin for µC/Probe as described in section 9-4 “IAR Systems C-SPY Plugin for µC/Probe” on page 71.

4-3-8 RS-232

µC/Probe supports a Serial RS-232 interface. This serial interface requires µC/Probe-Target code resident in your embedded system. Micriμm supports many UARTs and most likely this code is available from Micriμm. Contact Micriμm to find out if resident code for your particular setup is available.

Assuming your embedded system is running µC/Probe-Target as described in the document µC/Probe-Target Manual, enter the serial COM port number that your embedded target is attached to and select the baud rate from the drop downs shown in Figure 4-15:

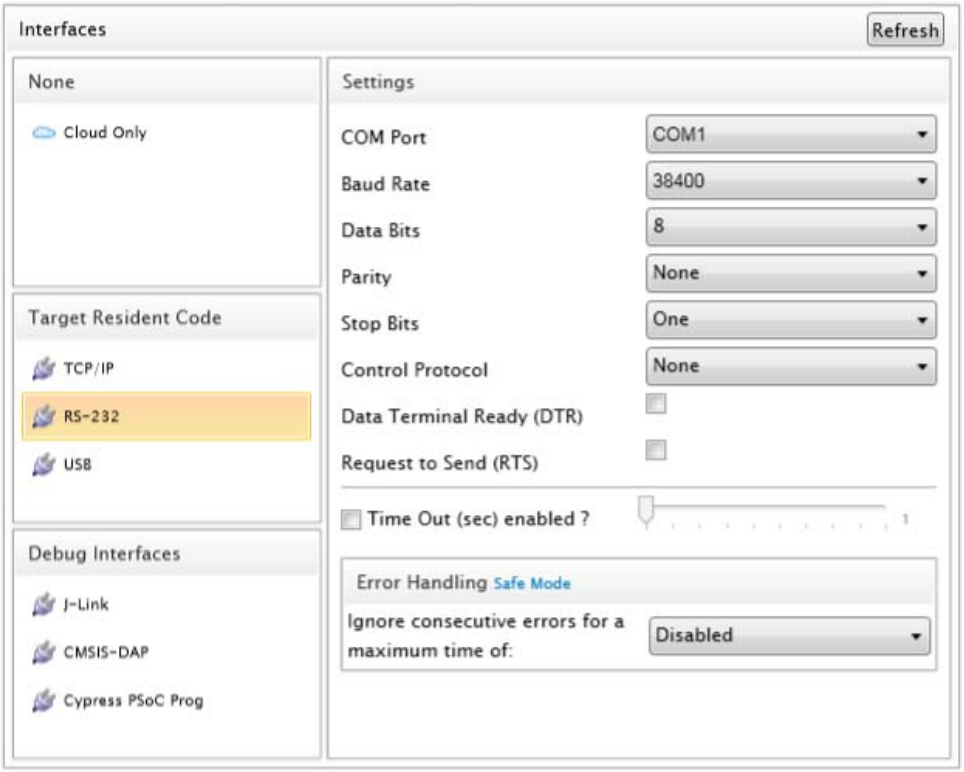


Figure 4-15 **µC/Probe Communication Settings: RS-232**

Chapter

5

µC/Probe Workspace Explorer

The µC/Probe Workspace Explorer is located on the right side of the application window and it is shown in Figure 5-1:

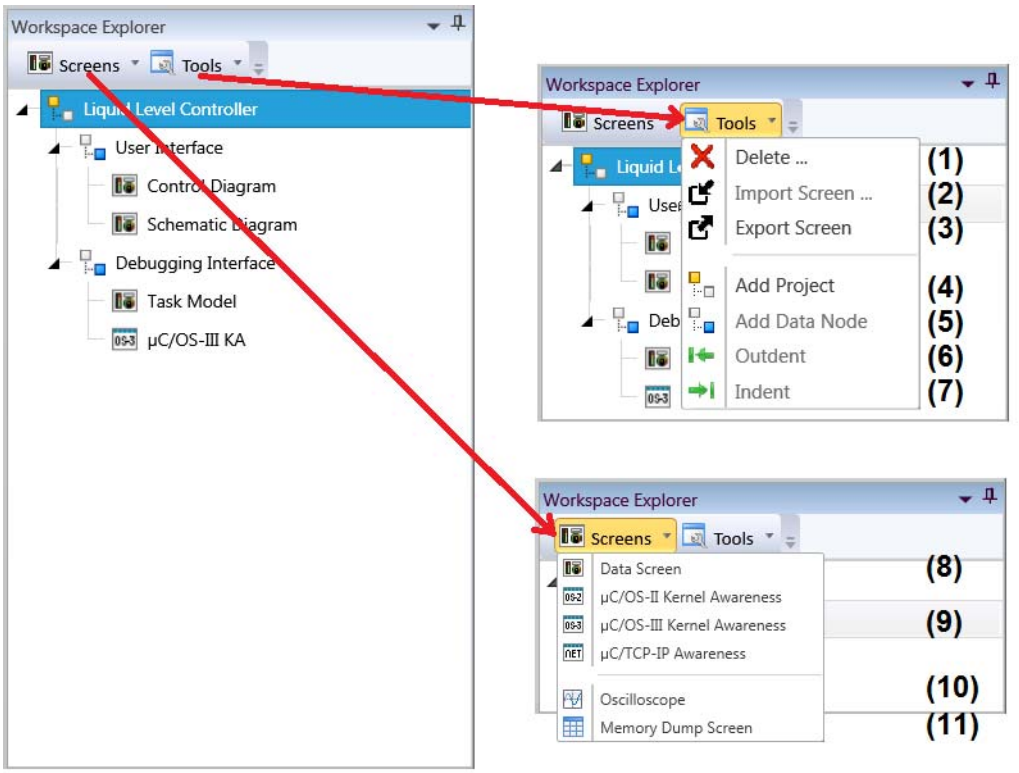


Figure 5-1 **µC/Probe Workspace Explorer**

F5-1(1) Use the Delete button to delete an item from the workspace explorer, including Projects, Data Nodes, Data Screens and Kernel Awareness Screens.

F5-1(2) Use the Import Screen button to import a previously exported screen.

F5-1(3) Use the Export Screen button to export the screen currently in focus to a file.

F5-1(4) µC/Probe allows you to create a dashboard or user interface in a matter of minutes. The data screen is where you drag and drop the virtual controls and indicators. Your data screen's appearance and layout are very important. You can use Projects as a means to separate complex dashboards into multiple regions. Each project can contain multiple data screens and the workspace explorer allows you to navigate through this hierarchy.

F5-1(5) Similar to Projects, Data Nodes are just another level of hierarchy that allows you to group sets of virtual controls and indicators together into categories you define.

F5-1(6) Use the outdent button to push out an item in the Workspace Explorer tree. The items you can outdent include Projects, Data Nodes, Data Screens and Kernel Awareness Screens.

F5-1(7) Use the indent button to push in an item in the Workspace Explorer tree. The items you can adjust the level of indentation include Projects, Data Nodes, Data Screens and Kernel Awareness Screens.

F5-1(8) Data Screens are the screens where you drag and drop the virtual controls and indicators. You can add as many data screens as you want.

F5-1(9) The Kernel Awareness and TCP/IP Screens are pre-configured Data Screens with all the symbols related to µC/OS-III, µC/OS-II and µC/TCP-IP respectively. See Appendix B, "Micriµm Software Awareness Screens" on page 110 for more information about this.

F5-1(10) The Oscilloscope control is documented in Appendix I, "Oscilloscope Control" on page 155.

F5-1(11) The Memory Dump screen allows you to read a range of the embedded target's memory.

In order to organize your workspace tree you can also use your mouse to drag and drop items and rename items by invoking the context menu with a right-click.

Figure 5-2 shows an example of using projects and data nodes to better present a control panel for a liquid level control system:

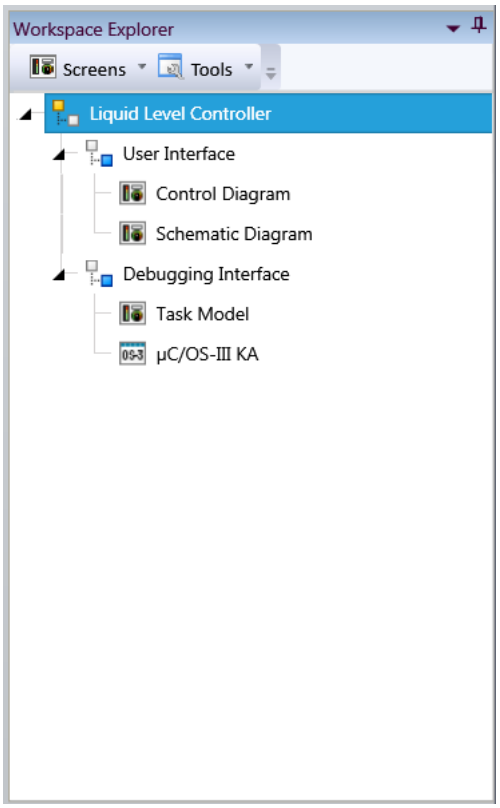


Figure 5-2 **Workspace Explorer for a Liquid Level Control System**

Chapter

6

µC/Probe Toolbox

The µC/Probe Toolbox is located on the left side of the application window and it is shown in Figure 6-1:

Figure 6-1 **µC/Probe Toolbox**

Depending on the µC/Probe Edition you purchased, the available tools will vary. This document describes all the features found in the Professional Edition of µC/Probe. For more information on which features you have, see Appendix L, "µC/Probe Editions Comparison Table" on page 176.

The items in the toolbox are contained in an accordion type of panel. You click on each button to display the items that belong to a category.

If you are running the Professional Edition of µC/Probe and have the automatic updates enabled, each category in this toolbox will expand with more virtual controls and indicators as software updates become available.

The following sections present a brief introduction to each of the toolbox categories. For more information on configuring each type of virtual control or indicator, see Appendix A, "Configuring Virtual Controls and Indicators" on page 74.

6-1 WRITABLE CONTROLS

The writable controls shown in Figure 6-2 include buttons, check boxes, sliders, radio buttons, a bit control, an RGB LED color picker, a numeric up/down and a textbox. Use these controls to read and modify the value of symbols from the embedded target. For more information configuring the properties of writable controls see Appendix A, "Virtual Controls" on page 82.

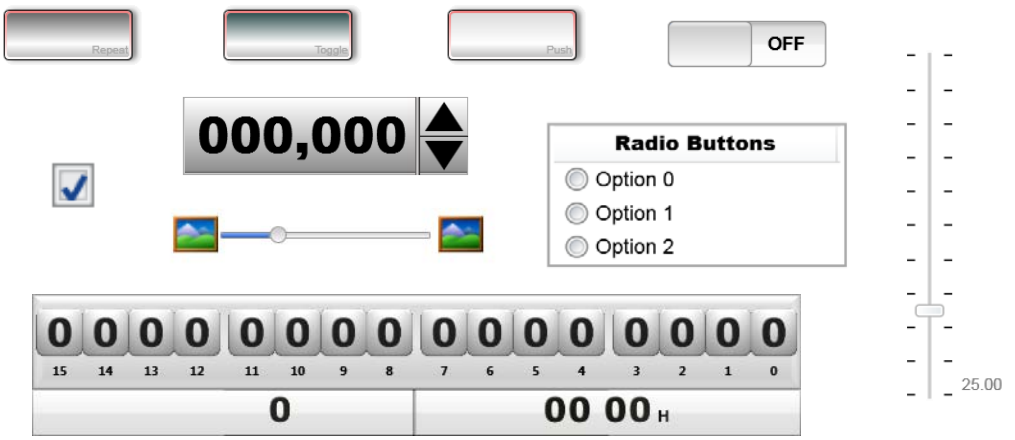


Figure 6-2 **µC/Probe Toolbox: Writable Controls**

6-2 LINEAR GAUGES

Use the linear gauges shown in Figure 6-3 to display numeric data in a tri-color linear scale, in terms of percentage or degrees of temperature. For more information configuring the properties of linear gauges see Appendix A, “Virtual Indicators” on page 75.

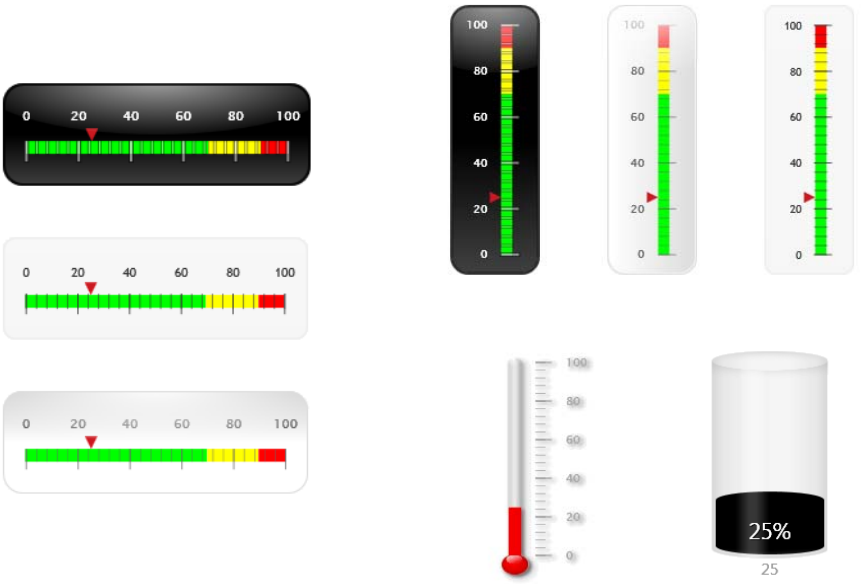


Figure 6-3 **µC/Probe Toolbox: Linear Gauges**

6-3 ANGULAR GAUGES

Use the angular gauges shown in Figure 6-4 to display numeric data in a tri-color angular scale. For more information configuring the properties of angular gauges see Appendix A, “Virtual Indicators” on page 75.

Figure 6-4 **µC/Probe Toolbox: Angular Gauges**

6-4 CHARTS

Use the charts shown in Figure 6-5 to display numeric data including arrays in a marker, line, area or scatter x-y chart. For more information configuring the properties of charts see Appendix A, “Sample-based line Charts” on page 98.

Figure 6-5 **µC/Probe Toolbox: Charts**

6-5 NUMERIC INDICATORS

Use the numeric indicators shown in Figure 6-6 to display numeric data in text. For more information configuring the properties of numeric indicators see Appendix A, “Formatting Properties Editor” on page 75 and Appendix A, “Numeric Indicator Properties Editor” on page 77.

Figure 6-6 **µC/Probe Toolbox: Numeric Indicators**

6-6 ADVANCED

The advanced category of the toolbox includes other miscellaneous indicators such as a text box, terminal window, scripting control, spreadsheet control, µC/Trace trigger control, data logger, HID control and an image container capable of displaying an indexed array of images. For more information configuring the properties of these advanced controls see Appendix A, “Virtual Indicators” on page 75, Appendix E, “Spreadsheet Control” on page 129 and Appendix F, “Scripting Control” on page 139.

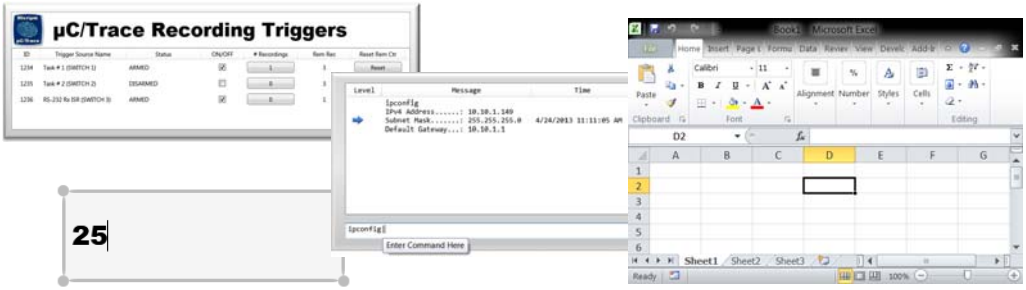


Figure 6-7 **µC/Probe Toolbox: Advanced**

6-7 MISCELLANEOUS

All the rest of virtual indicators and controls are located in the *Miscellaneous* category. The image below shows some of them.

Use the LEDs shown in Figure 6-8 to display not only boolean data types but also any numeric variable that represents a color code in ARGB format. For more information configuring the properties of LEDs see Appendix A, “Formatting Properties Editor” on page 75 and Appendix A, “LED Properties Editor” on page 78.

Use the Compass indicator to display azimuth data in degrees; where 0 represents North and it increments clockwise up to 359.

Figure 6-8 **µC/Probe Toolbox: Miscellaneous**

Chapter

7

µC/Probe Layout Design Tools

The Layout Design Tools are located on the Main Toolbar at the top of the application's window. They include tools to arrange the virtual controls and indicators on your data screen as shown in Figure 7-1:

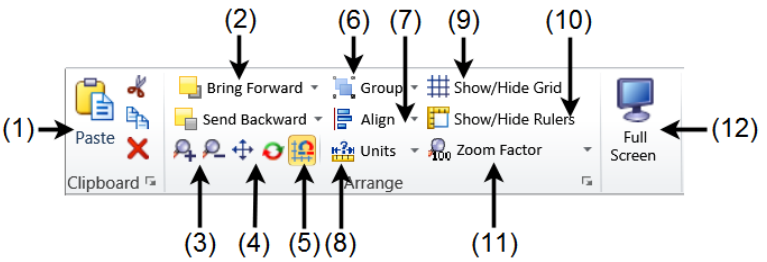


Figure 7-1 **µC/Probe Layout Design Tools**

F7-1(1) µC/Probe allows you to copy, cut and paste a single or multiple virtual controls or indicators. Everything you select will be stored temporarily into µC/Probe's clipboard during your session.

F7-1(2) µC/Probe allows you to drag and drop virtual controls and indicators onto the data screen and have them overlap one another. In some cases may be necessary to move some of them to the front of the data screen. Select the virtual control and indicator you want to move and click the **Bring Forward** or **Send Backward** button to move it to the front or to the back respectively.

F7-1(3) µC/Probe includes accessibility features that make the software more user friendly. You can control the zoom level of your µC/Probe Data Screen during both design-time and run-time. Click the buttons with the magnifier glass to zoom in and out, or use the keyboard shortcuts **Ctrl+** to zoom-in and **Ctrl-** to zoom-out.

F7-1(4) The panning tool in µC/Probe makes it easy to move a data screen around while your are zoomed in. The **Pan** button is a toggle button, clicking the button toggles the panning mode on and off.

F7-1(5) When you drag and drop a virtual control or indicator onto the data screen, the next thing you usually do is resize or move the object around the data screen. Turn the Snap-to-Grid mode on in order to align the virtual control or indicator to the nearest intersection of grid lines. The **Snap to Grid** button is a toggle button, clicking the button toggles the snap-to-grid mode on and off.

F7-1(6) You can combine multiple virtual controls and indicators so you can work with them as though they were a single object. You can resize, move, copy and paste all virtual controls and indicators in a group as a single unit.

After you have grouped virtual controls and indicators, you can still select any single object within the group without un grouping by first selecting the group, and then clicking on the object you want to select.

F7-1(7) µC/Probe allows you to easily align virtual controls and indicators by first selecting the group of objects you want to align and then clicking on one of the following alignment options:

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

All the objects are aligned with respect to the first selected item.

F7-1(8) Use the **Units** button to select the grid and ruler's metric system.

F7-1(9) Use the **Show/Hide Grid** button to show and hide the grid lines on the data screen. The snap-to-grid mode still works even if the grid is not visible.

F7-1(10) Use the **Show/Hide Rulers** button to show or hide the ruler. The Show/Hide Rulers button is a toggle button, clicking the button turns the rulers on and off.

F7-1(11) Every time you click the magnifier glass buttons to zoom in and out, µC/Probe zooms in and out by certain zooming factor. Click the **Zoom Factor** button to select a different zooming factor.

F7-1(12) Click the **Full Screen** mode button to hide all the tools except the data screen. The Full Screen button is a toggle button, clicking the button turns the full screen mode on and off.

7-1 µC/PROBE EXAMPLE

In order to demonstrate the previous layout design tools, Figure 7-2 shows an example of a power plant's diagram used as a background to create a control panel with µC/Probe:

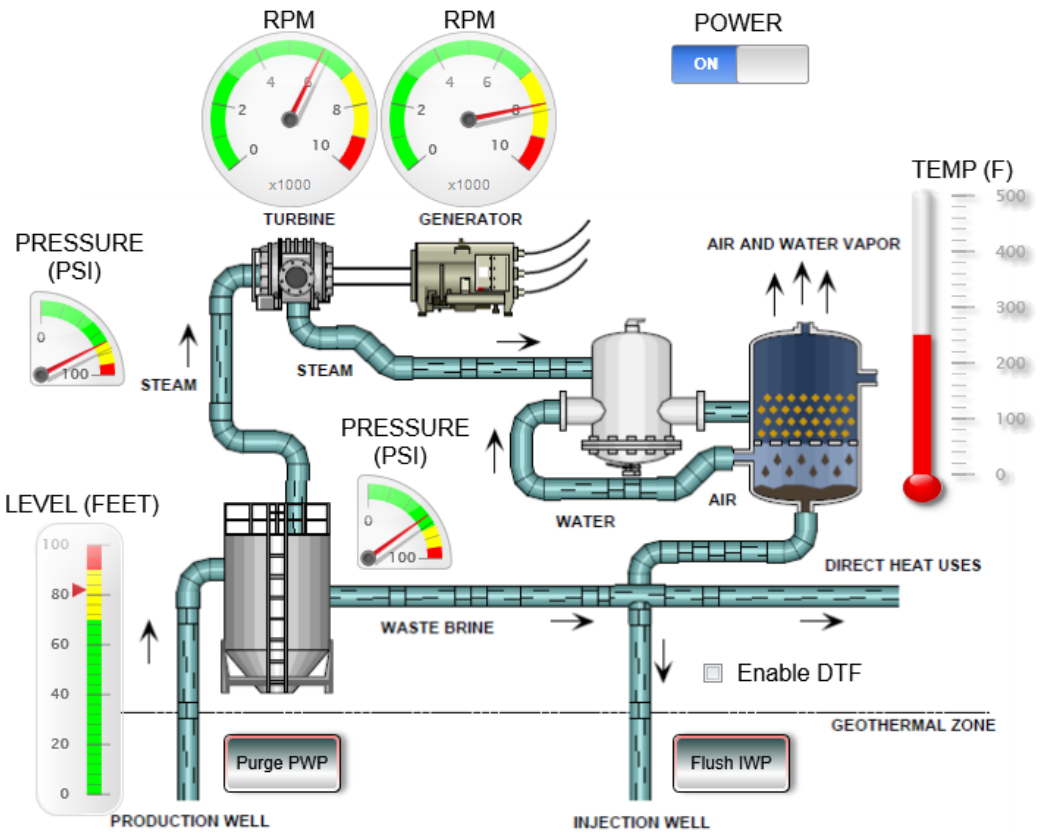


Figure 7-2 **µC/Probe Example of a Power Plant**

Chapter

8

Associating Symbols to Virtual Controls and Indicators

During design-time, use the µC/Probe symbol browser discussed in Chapter 3, “µC/Probe Symbol Browser” on page 16, to search and select the embedded target variables you want to associate to each of the virtual controls and indicators you placed on your data screen.

Once you find the symbol you want to associate, drag and drop the symbol over the virtual control or indicator you want, as shown in Figure 8-1:

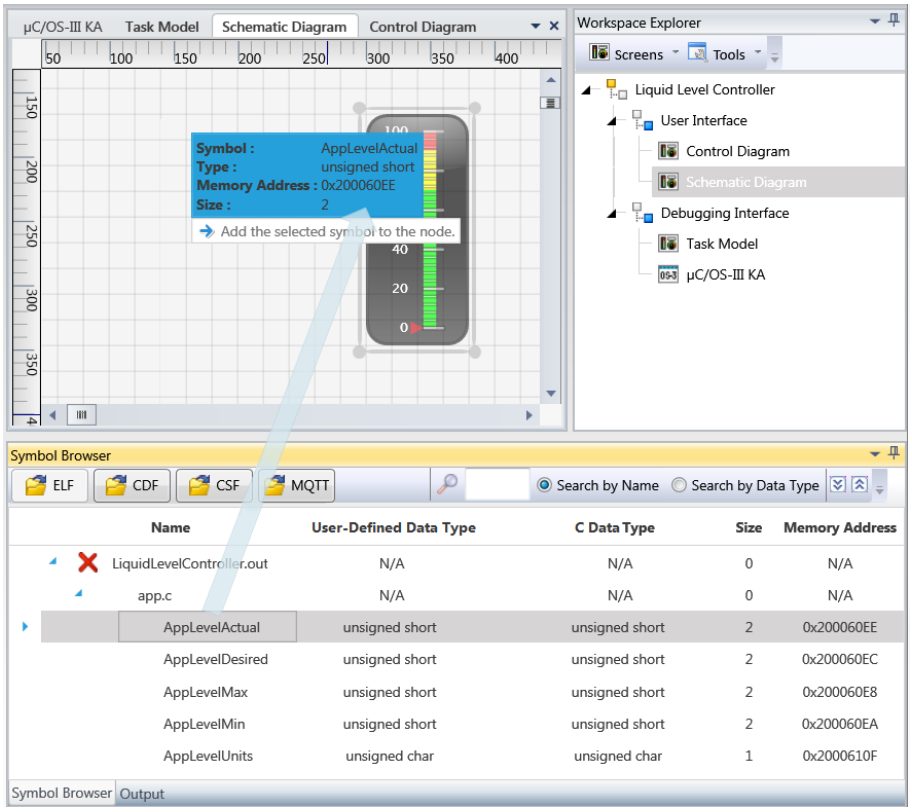


Figure 8-1 **Associating Symbols to Virtual Controls and Indicators**

Repeat the same process for each of the virtual controls and indicators placed on your data screen and µC/Probe will be ready to go into Run Mode unless you want to further configure other optional settings.

8-1 CUSTOM SYMBOLS

µC/Probe supports the association of more types of symbols such as the following:

- Full Array
- Elements of an Array
- Full Structure
- Member of a Structure
- The combinations of the above (up to 4 levels)

We call the above types of symbols: *Custom Symbols* and you can use the *Symbol Settings* panel on the right hand side of the *Symbol Browser* to declare these type of Custom Symbols.

The Symbol Settings panel changes its context every time you select a different symbol from the Symbol Browser.

To illustrate the way to use this Symbol Settings panel, imagine you want to associate the following structure member (Baudrate) to one of the virtual controls or indicators:

AppCommInterfaces	<struct>	0x20016B58	COMM_INTERFACES
RS232	<array>	0x20016B58	COMM_INTERFACE_RS232[2]
[0]	<struct>	0x20016B58	COMM_INTERFACE_RS232
[1]	<struct>	0x20016CE0	COMM_INTERFACE_RS232
Cfg	<struct>	0x20016CE0	COMM_INTERFACE_RS232_CFG
SerialLine	'\0' (0x00)	0x20016CE0	CPU_INT08U
Baudrate	115200	0x20016CE4	CPU_INT32U
Data	<struct>	0x20016CE8	COMM_INTERFACE_RS232_DATA
TCPIP	<struct>	0x20016E68	COMM_INTERFACE_TCPIP

Figure 8-2 **AppCommInterfaces.RS232[1].Cfg.Baudrate**

The first step is to go to the Symbol Browser and search for the symbol's parent as shown in Figure 8-3:

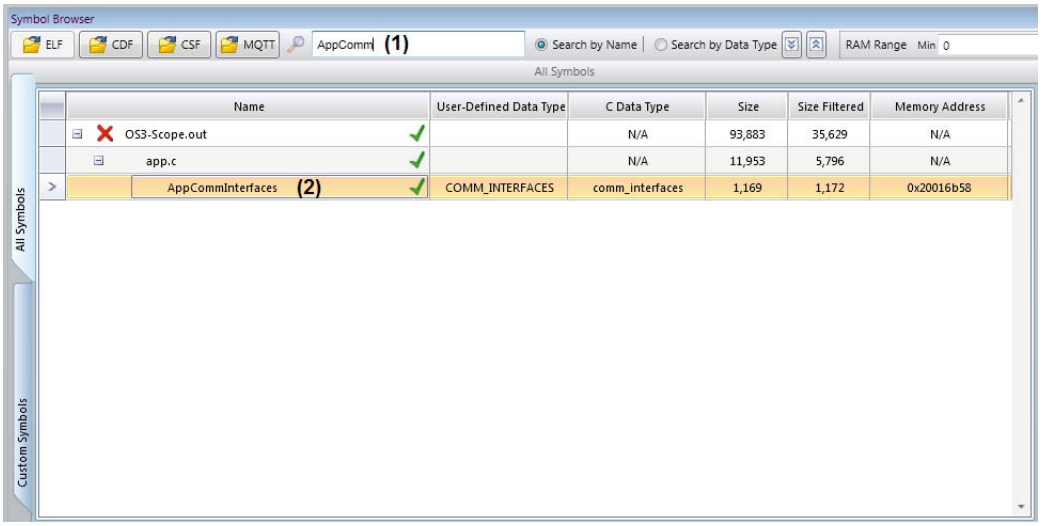


Figure 8-3 **Symbol Browser Search Box**

F8-3(1) Type in the name of the symbol's parent in the search box. e.g. 'AppCom'. Alternatively, you can use the tree navigator to search for the symbol by expanding the appropriate nodes.

F8-3(2) Select the symbol's parent from the Symbol Browser. Notice how the Symbol Settings panel on the right hand side, changes its context as you select a different symbol.

The next step is to configure the Custom Symbol by navigating down to the structure member of interest (i.e. Baudrate).

To illustrate the navigation process, see Figure 8-4 where we show you a series of screen captures of the different screens you will be presented while navigating.

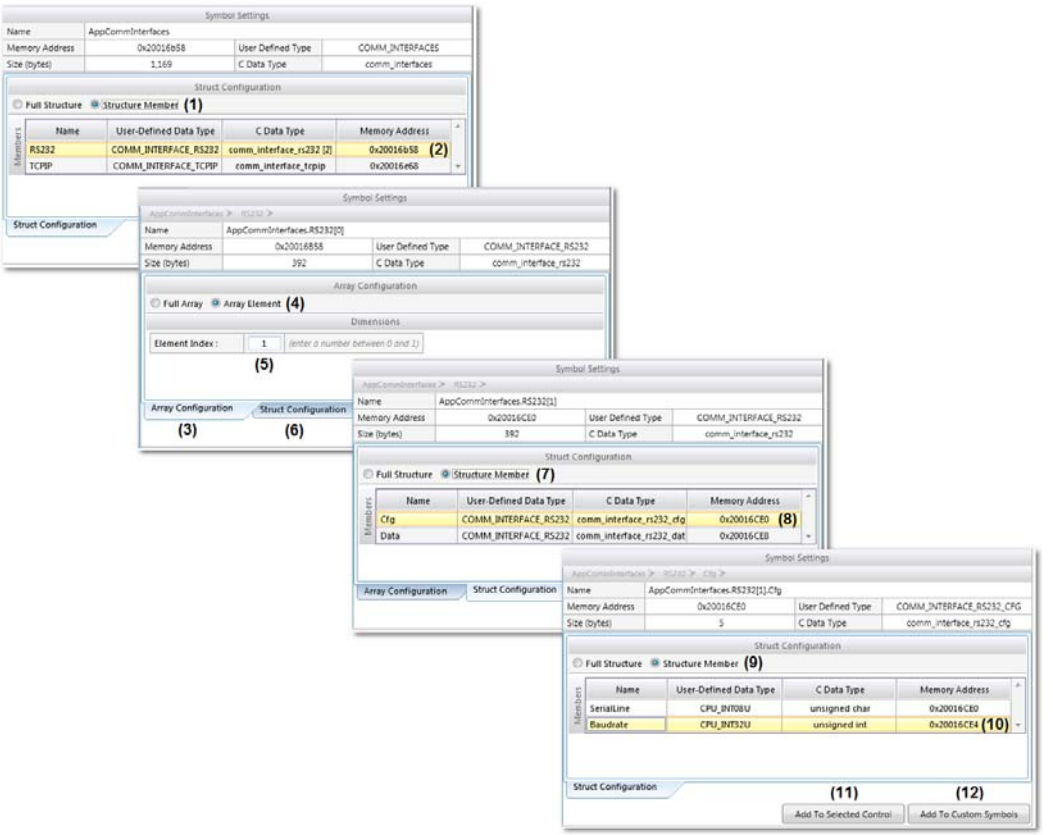


Figure 8-4 **Symbols Settings Panel Navigation**

F8-4(1) Select the option *Structure Member* from the radio buttons. You can also select the option *Full Structure* if you want to associate the entire structure (a feature that is supported only by the Tree Control and the Spreadsheet Control).

F8-4(2) Select the structure member you intend to use. e.g. RS232.

F8-4(3) Because the selected structure member is an array, select the tab *Array Configuration*. You can also choose the *Struct Configuration* tab if you intend to associate the entire structure down this level.

F8-4(4) Select the option *Array Element* from the radio buttons. You can also select the option *Full Array* if you want to associate the entire array down this level.

F8-4(5) Type in the index of the array element you want to associate. e.g. 1

F8-4(6) Select the tab *Struct Configuration* to keep navigating.

F8-4(7) Select the option *Structure Member* from the radio buttons. You can also select the option *Full Structure* if you want to associate the entire structure.

F8-4(8) Select the structure member you intend to use. e.g. `Cfg`.

F8-4(9) Select the option *Structure Member* from the radio buttons. You can also select the option *Full Structure* if you want to associate the entire structure down this level.

F8-4(10) Select the structure member you intend to associate. e.g. `Baudrate`.

F8-4(11) Now that you have navigated down to the structure member you wanted, you can add the selected symbol `AppCommInterfaces.RS232[1].Cfg.Baudrate` by first selecting the virtual control or indicator and then pressing the button *Associate to selected control*.

F8-4(12) Alternatively, you can create a whole new Custom Symbol that you can always go back and associate to different controls and indicators by pressing the button *Add to custom symbols*.

The list of custom symbols that you have created is displayed in the Symbol Browser's Custom Symbols tab as indicated in Figure 8-5:

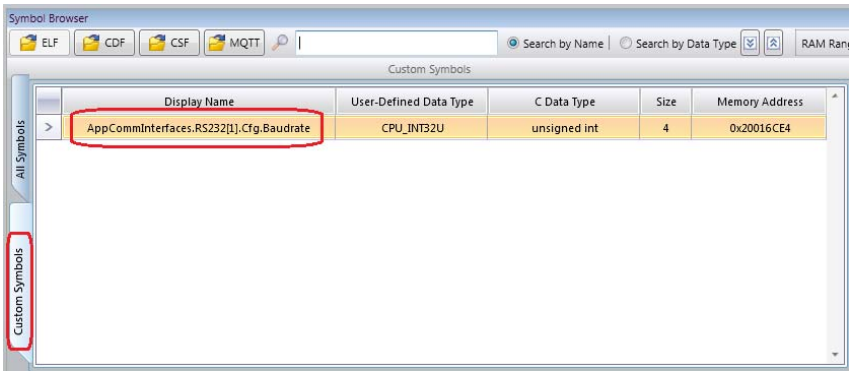


Figure 8-5 **Custom Symbols**

Chapter

9

Run-Time Mode

9-1 RUN-TIME CHECKLIST

Before setting µC/Probe in Run-Time mode you should verify each of the following items:

Item #	Description	Reference
1	The embedded target has been programmed with an output file (ELF file) with debug information in the DWARF-2, -3 or -4 format or with an XML-based Custom-Symbol File or Chip-Definition File.	µC/Probe Target Manual: Chapter 5, on page 21, Appendix C, on page 55 and Appendix D, on page 61
2	The embedded target is running and connected to the Windows PC through the communication interface of your choice.	µC/Probe Target Manual: Chapter 3, on page 14
3	µC/Probe has been configured with the latest symbol file (ELF, CDF, CSF or MQTT) that the embedded target is actually running.	µC/Probe User's Manual: Chapter 3, on page 16
4	µC/Probe has been configured with the proper communication interface and settings.	µC/Probe User's Manual: Chapter 4, on page 31
5	µC/Probe contains at least one virtual control or indicator on the data screen.	µC/Probe User's Manual: Chapter 7, on page 60
6	µC/Probe has been configured to associate the virtual control or indicator with one of the embedded target's variables displayed in the symbol browser.	µC/Probe User's Manual: Chapter 8, on page 63
7	In case you are running the Infineon Edition of µC/Probe known as µC/Probe XMC™, make sure that it has been configured to communicate with the correct XMC™ family of microcontrollers.	µC/Probe User's Manual: Appendix K on page 172

Table 9-1 **Run-Time Mode Checklist**

In order to set µC/Probe in Run-Time mode, click on the **run** button indicated in Figure 9-1:

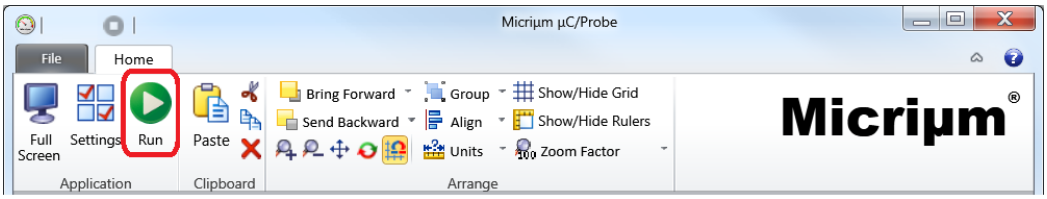


Figure 9-1 **Setting µC/Probe in Run-Time mode**

µC/Probe should start updating your data screens immediately and the application displays all kinds of status information in the status bar at the bottom of the µC/Probe window as shown in Figure 9-2:

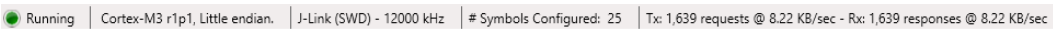


Figure 9-2 **µC/Probe Status Bar**

9-2 CUSTOMIZING DATA SCREENS LAYOUT

In µC/Probe you can customize the position, size and behavior of data screens to create window layouts that work best for your workspace. When you customize the layout, µC/Probe remembers it on a workspace-by-workspace basis.

By default, all data screens that you create, product awareness screens and the oscilloscope control window are arranged as *tabbed windows*. This means that only one window can be visible and updated at a time. However, all these windows can also be *docked*, so that they have position and size within the µC/Probe workspace area, or floating as separate windows independent of µC/Probe.

For example, you could display your own data screen, the kernel awareness screen and the oscilloscope control in three different monitors and µC/Probe will update them all at the same time. Notice that when you do this, update rates are affected as more controls need to be updated.

You can select between the options **Tabbed** and **Floating** by right-clicking the title bar of the window to be arranged.

9-3 RUNNING µC/PROBE AND YOUR DEBUGGING SOFTWARE AT THE SAME TIME

Your debugging software for embedded applications usually comes integrated with your IDE and at a minimum, allows you to step through the code, set breakpoints, display register and memory windows, display call stack information, and monitor variables and expressions. Examples of debugging software include IAR's C-SPY and GNU's GDB.

You can also use µC/Probe to extend the capabilities of your debugging software by running both at the same time. µC/Probe allows you to have instant control over your global variables in a real-time and non-intrusive way. From you debugger software, you can set breakpoints at locations of particular interest in the application being debugged and µC/Probe will stop updating the virtual controls and indicators at the same time.

This feature is accomplished by sharing the connection between the Windows PC and the Embedded Target being debugged. Whether the debugger of your choice is IAR's C-SPY, GNU's GDB or any other debugging software that supports J-Link, Figure 9-3 illustrates and example of running µC/Probe and the debugger of your choice at the same time:

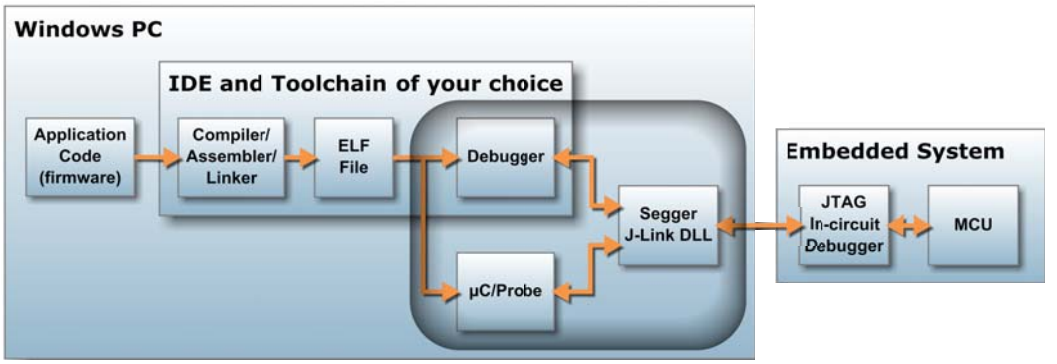


Figure 9-3 **Running µC/Probe and your debugging software at the same time**

Notice how µC/Probe and the Debugger Software not only share the same ELF file but also the same logic and physical interface through the Segger's J-Link DLL and JTAG in-circuit debugger respectively.

9-4 IAR SYSTEMS C-SPY PLUGIN FOR µC/PROBE

µC/Probe is tightly integrated with IAR Embedded Workbench® thanks to a TCP/IP bridge between C-SPY® and µC/Probe. This bridge gives µC/Probe access to not only its native supported platforms but also all the devices and processor architectures supported by IAR Systems without the need to write any target resident code in the form of communication routines, because C-SPY® handles all communication needed as illustrated in the following Figure 9-4:

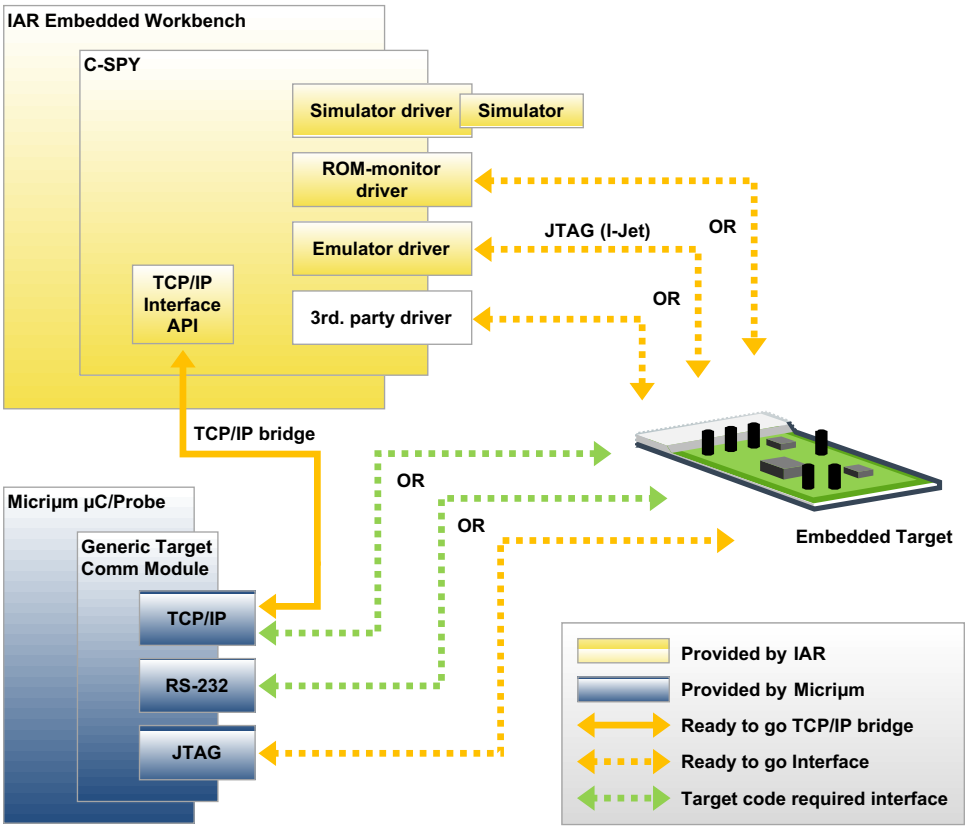


Figure 9-4 **Communication Interfaces Supported by IAR Systems C-SPY and µC/Probe**

In regards to the communication interface options illustrated in Figure 9-4, some of them, such as the ones based on JTAG are ready-to-go assuming your embedded target has either an in-circuit debugger or an external JTAG probe (i.e. IAR's I-Jet). Other communication interfaces require some resident code running in the embedded target, which is available by Micriµm for most platforms.

9-4-1 CONFIGURING THE TCP/IP BRIDGE BETWEEN IAR C-SPY AND µC/PROBE

The TCP/IP bridge between C-SPY® and µC/Probe is built in the form of a plugin module delivered with the Embedded Workbench product installation.

In order to configure Embedded Workbench to load the plugin, you open your project's debugger options and select the plugin from the list of available plugins as shown in Figure 9-5:

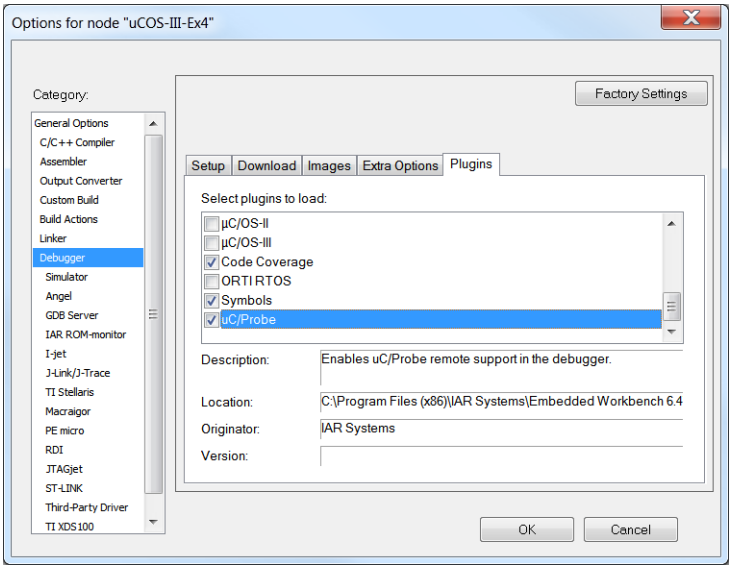


Figure 9-5 **Configuring the IAR Systems Embedded Workbench**

The plugin module gets loaded during a debug session and opens a TCP/IP socket on localhost to listen for µC/Probe requests.

Depending on your network security settings, the first time you launch a debug session you may be asked to allow Embedded Workbench to open a TCP/IP connection.

At the same time, µC/Probe needs to be configured to connect through its TCP/IP interface to localhost on port 9930 as described in section 4-3-7 “TCP/IP” on page 48 and as shown in Figure 9-6:

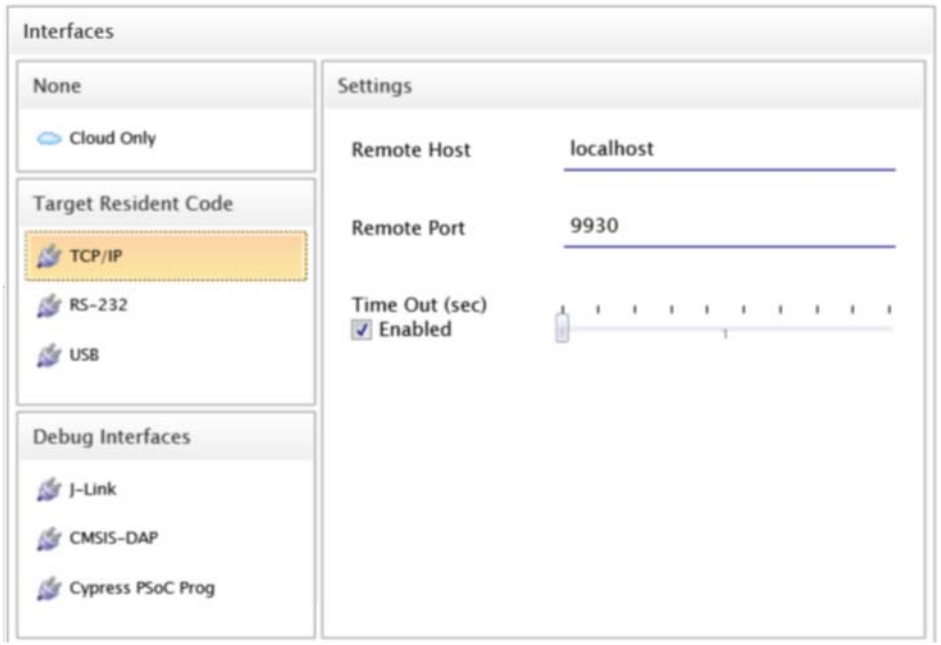


Figure 9-6 **Configuring µC/Probe to Interface with IAR Systems C-SPY plugin for µC/Probe**

Appendix

A

Configuring Virtual Controls and Indicators

Once you drag and drop one of the virtual controls or indicators onto the data screen and associate it with one of the embedded target's symbols from the symbol browser, you can access the properties tool bar by moving the mouse over the virtual control or indicator.

The tool bar shown in Figure A-1 appears for you to select between one of the three configuration categories:

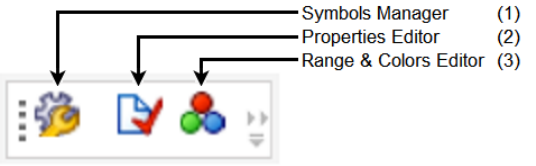


Figure A-1 **Virtual Controls and Indicators Toolbar**

FA-1(1) The Symbols Manager is common for all virtual controls and indicators, see Chapter 8, "Associating Symbols to Virtual Controls and Indicators" on page 63 for more information on the Symbols Manager.

FA-1(2) The Properties Editor is similar among most of the virtual controls and indicators and the next sections will describe how to use the Properties Editor for just a few of the most representative virtual controls and indicators.

FA-1(3) The Range and Colors Editor is only available to those virtual indicators that feature a multi-color scale. The next sections will describe how to use the Range and Colors Editor for a few of the most representative virtual indicators.

A-1 VIRTUAL INDICATORS

A-1-1 FORMATTING PROPERTIES EDITOR

The virtual indicators formatting category applies to linear gauges, half donuts, cylinders, numeric indicators, thermometers, graphs and any virtual indicator capable of showing the symbol's value in a graphical or text format. Figure A-2 shows the formatting category of a linear gauge:

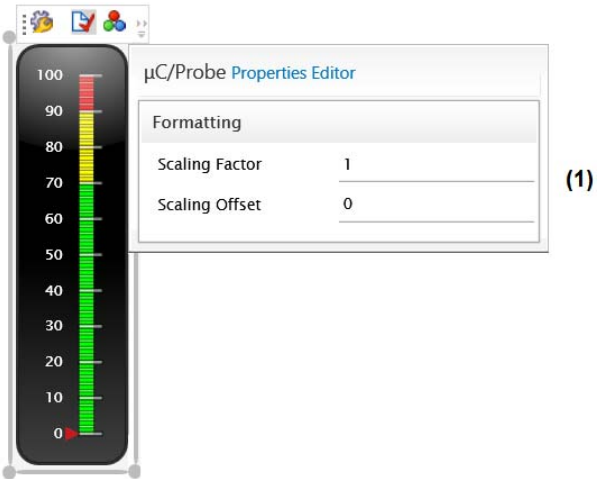


Figure A-2 **Formatting Properties Editor**

FA-2(1) In case you need to convert the value to Engineering Units (EU) before displaying in the virtual indicator, you can use the scaling factor and offset to specify the parameters of a linear conversion function. For example, if the embedded target's symbol you need to display is a 4-20mA value, you can implement the standard linear equation y = mx + b where m is the scaling factor, x is the 4-20mA value, b is the offset and y is the resulting Engineering Units (EU) value to display.

A-1-2 RANGE AND COLORS EDITOR

The Range and Colors Editor applies to linear gauges, half donuts, cylinders and any virtual indicator capable of displaying the symbol's value in a graphical format along a multi-color scale. Figure A-3 shows the Range and Colors Editor for a linear gauge:

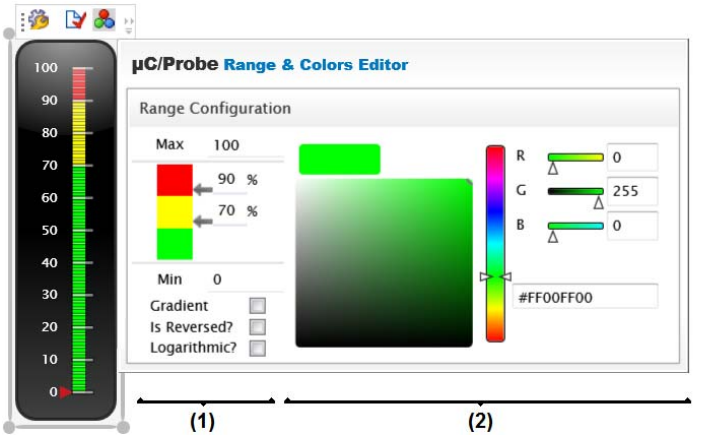


Figure A-3 **Range and Colors Editor**

FA-3(1) Start by setting the **Max** limit, then click on the text boxes next to the percentage signs and enter the thresholds in terms of percentage.

At the bottom of this section, the checkboxes allow you to configure the type of scale and it's appearance.

FA-3(2) Each time you click on one of the text boxes to set the percentages, the color picker allows you to choose the color for that gauge band. You can enter the color you want in hex format or by selecting a color from the vertical slider and then fine tuning with the palette.

A-1-3 NUMERIC INDICATOR PROPERTIES EDITOR

The Numeric Indicator category from the Properties Editor only applies to numeric indicators. Figure A-4 shows the numeric indicator's properties. Font styles, alignment and the thousand separator, they all apply to the number 0 shown in white:

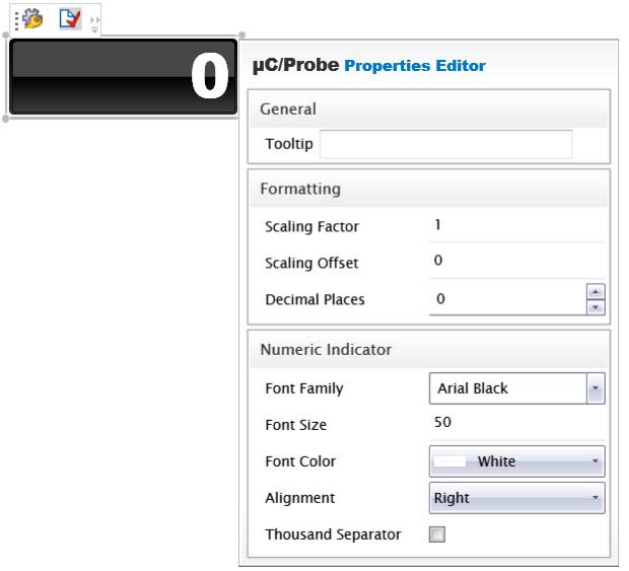


Figure A-4 **Numeric Indicator Properties Editor**

A-1-4 LED PROPERTIES EDITOR

The LED control allows you to display a circle, triangle or rectangle with its color mapped to the value of one of your embedded target's global variable. This control works as a virtual LED as illustrated in the example in Figure A-5 where the LED control is configured to turn bright red when the value is 1 and dark red when the value is 0.

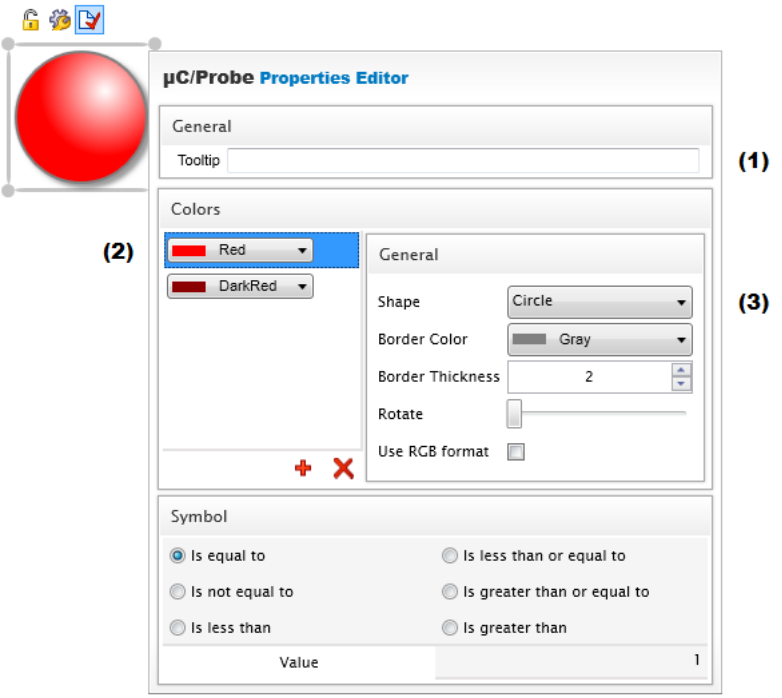


Figure A-5 **LED Properties Editor**

FA-5(1) You can include an optional **Tooltip** message that will appear with any information you want to display about the item being hovered over.

FA-5(2) The **Mappings** section is where you get to make the association between the LED's color and the value of the symbol.

FA-5(3) The **Style** section allows you to configure the LED's appearance by modifying the shape and border.

The checkbox labeled as **Use RGB Format** can be used to associate this control to a global variable that stores a color code encoded in the ARGB additive color model format. The ARGB format is a 32-bit value where each byte is a channel representing the intensity of the channels Alpha, Red, Green and Blue. For example, if the variable's value is 0xFFFFFF00 then the LED would turn Yellow.

A-1-5 BITMAP ANIMATION PROPERTIES EDITOR

The bitmap animations are part of the toolbox's advanced group. They are one of the most powerful virtual indicators because you have the freedom to customize it however you want by providing your own images.

Imagine you want to display the state of a valve to be either open or closed in a graphical way by using the bitmap images shown in Figure A-6 and an embedded target's application variable named `AppValveOutPct` that stores the state of the outflow valve (0%:open and 100%:closed).

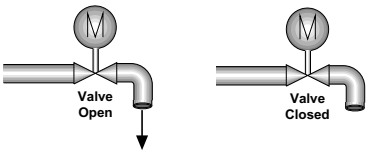


Figure A-6 **Bitmaps to Animate**

Figure A-7 shows the properties editor for the bitmap animation:

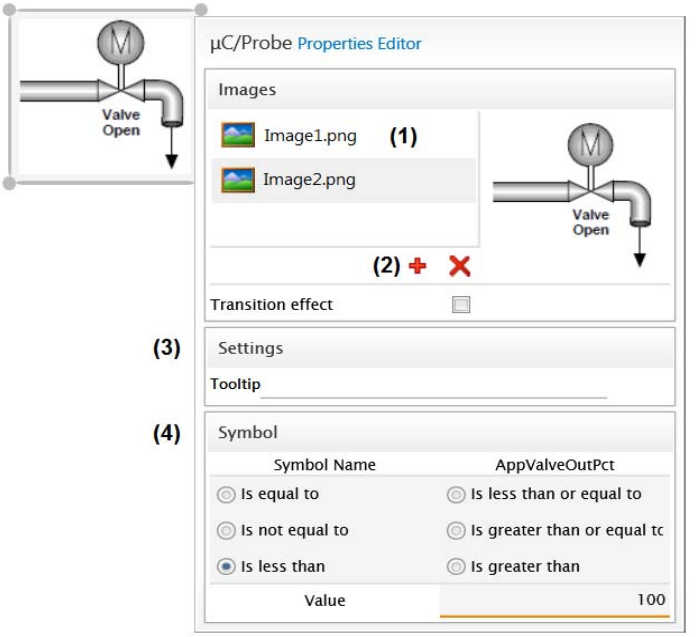


Figure A-7 **Bitmap Animation Properties Editor**

FA-7(1) The list of images available to display. You select the image you want to work with and a preview of the image is shown on the right side.

FA-7(2) You can add or delete more images into the list by making click on the red **+** or **x** buttons respectively.

Select the transition effect check box if you want to add a fade-in and fade-out effect between image transitions.

FA-7(3) You can include an optional **Tooltip** message that will appear with any information you want to display about the item being hovered over.

FA-7(4) Specify the rules. In this case, if `AppValveOutPct < 100` then the valve is open, and if `AppValveOutPct = 100` then the valve is all the way closed.

A-1-6 AZIMUTH INDICATOR

The Azimuth Indicator or Compass shown in Figure A-8 is the only virtual indicator without a properties editor; that's because you simply map it to a symbol whose value is within 0 and 359, where 0 represents *North*, 90 *East* and so on as illustrated in Figure A-8.

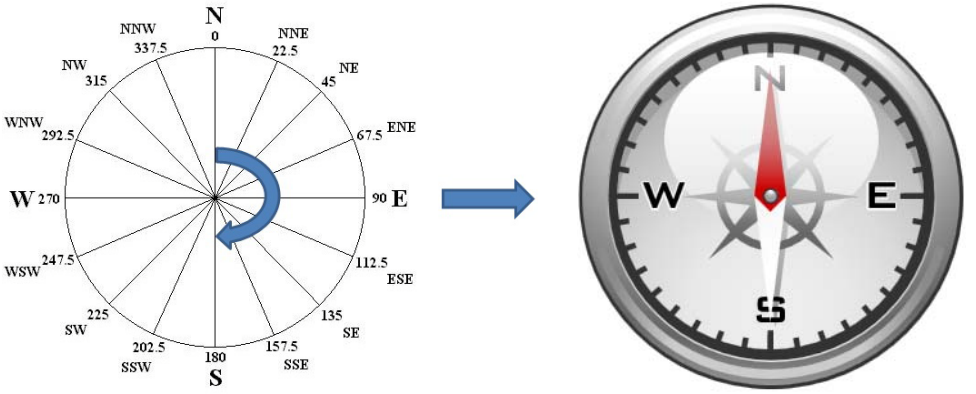


Figure A-8 **Compass**

A-2 VIRTUAL CONTROLS

A-2-1 SLIDER CONTROL PROPERTIES EDITOR

The slider control is one of the few writable controls. It allows you to gradually modify an adjustable embedded target symbol's value. The user gets to select from a range of values by moving a value indicator up and down a track. For example, you typically create a volume control by using a slider control.

Figure A-9 shows the slider control properties editor:

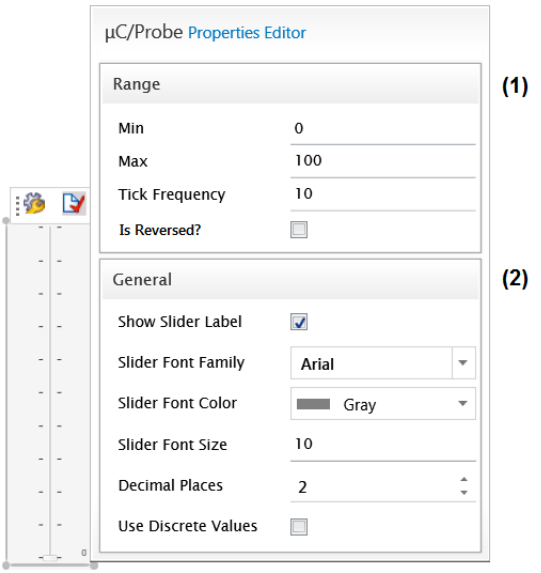


Figure A-9 **Slider Control Properties Editor**

FA-9(1) The slider control has a minimum, a maximum, and an increment value. The **Tick Frequency** not only determines the increment value but also the number of tick marks along the track. You can reverse the scale by ticking the checkbox.

FA-9(2) The **General** category includes the formatting properties that affect the slider's tick labels. Select the check box **Use Discrete Values** if you want the slider to adjust the associated symbol by making discrete increments.

A-2-2 CUSTOM SLIDER PROPERTIES EDITOR

The custom slider is similar to the one from section A-2-1, except that it also allows you to include two images to the left and right side of the slider's track as shown in Figure A-10:

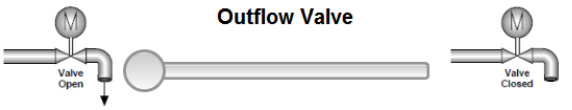


Figure A-10 **Custom Slider Example**

Imagine you want to control the state of a valve to be either open or closed by modifying an embedded target's application variable named `AppValveOutPct` that stores the state of the outflow valve (0%:open and 100%:closed). Figure A-11 shows the custom slider properties editor for such example:.

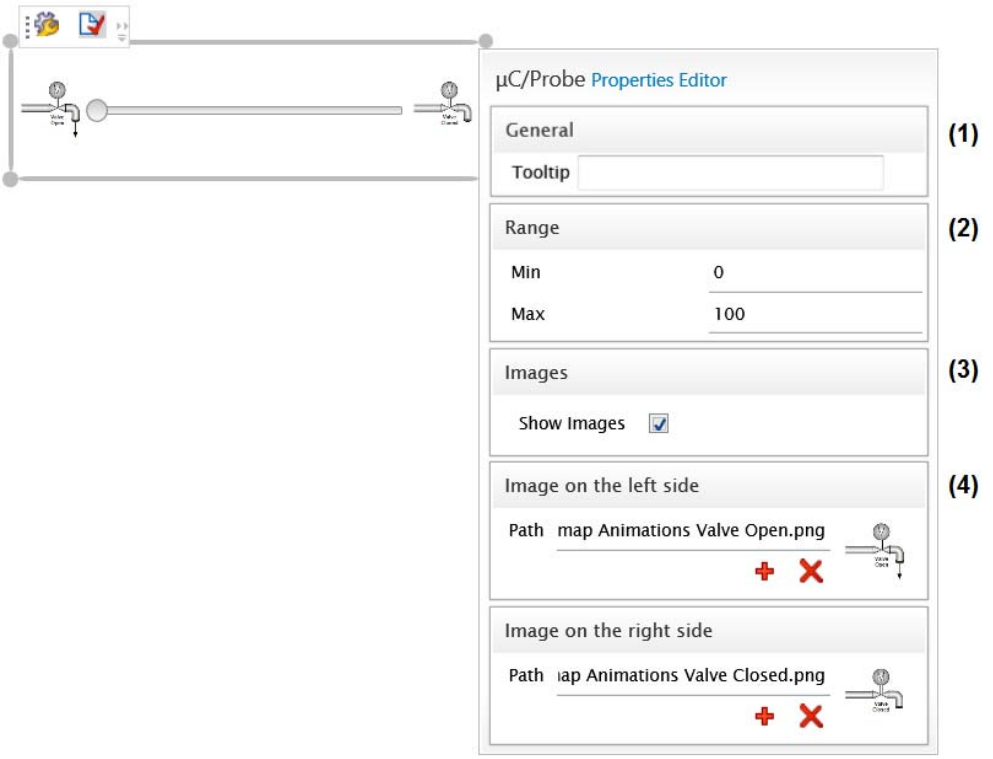


Figure A-11 **Custom Slider Properties Editor**

FA-11(1) You can include an optional **Tooltip** message that will appear with any information you want to display about the item being hovered over.

FA-11(2) The custom slider control has a minimum and maximum range.

FA-11(3) You can turn on and off the images.

FA-11(4) Specify the path or browse to the bitmap you want to be placed on the left and right sides of the track.

A-2-3 CUSTOM SWITCH PROPERTIES EDITOR

The custom switch control is a two state button. You can modify the value of an embedded target's symbol by specifying the values you want to write when the button is switched between the **On** and **Off** states as shown in Figure A-12:

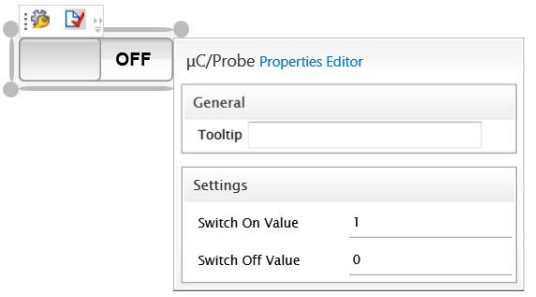


Figure A-12 **Custom Switch Properties Editor**

A-2-4 CHECKBOX PROPERTIES EDITOR

The checkbox control is similar to the custom switch but it also allows you to specify a label to display when the checkbox is selected and not selected. In the example shown in Figure A-13 such labels are set to Enabled during the On state and Disabled during the Off state.

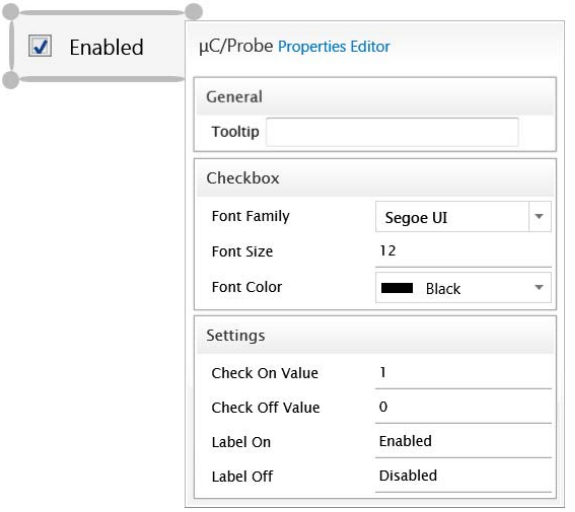


Figure A-13 **Checkbox Properties Editor**

A-2-5 PUSH BUTTON PROPERTIES EDITOR

The push button control is a momentary switch that switches between the states **On** while held down and **Off** when released. The properties window is shown in Figure A-14.

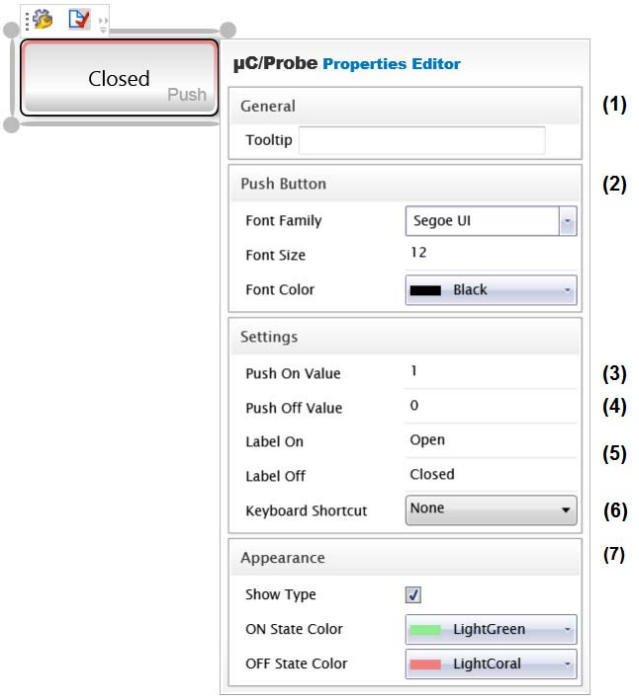


Figure A-14 **Push Button Properties Editor**

FA-14(1) You can include an optional **Tooltip** message that will appear with any information you want to display about the item being hovered over.

FA-14(2) You can specify the labels to display during the **On** and **Off** states. These properties affect the label's font family, font size and font color properties.

FA-14(3) During run-time, when the user holds down the push button, µC/Probe writes the **Push On Value** one single time to the embedded target.

FA-14(4) When the user releases the button, µC/Probe writes the **Push Off Value** one single time to the embedded target.

FA-14(5) Here you specify the labels you want to display during the **On** and **Off** states.

FA-14(6) The keyboard shortcut is a sequence or combination of keystrokes on the keyboard which will invoke the Click event on the Push Button.

FA-14(7) You can also specify the border colors you want to display during the **On** and **Off** states and whether or not you want to show the **Push** label on the corner.

A-2-6 TOGGLE BUTTON PROPERTIES EDITOR

The toggle button control is a button that switches between the states **On** and **Off** when clicked. The properties window for a relay's toggle button is shown in Figure A-15:

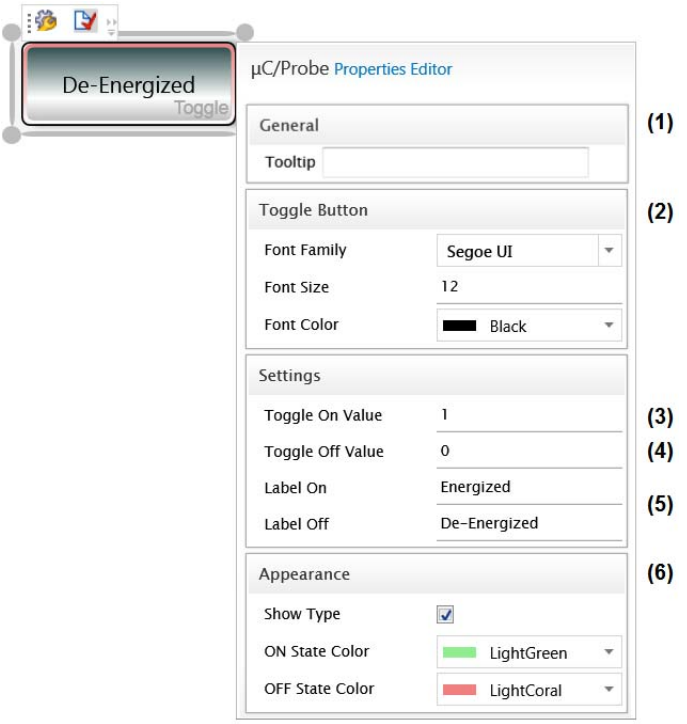


Figure A-15 **Toggle Button Properties Editor**

FA-15(1) You can include an optional **Tooltip** message that will appear with any information you want to display about the item being hovered over.

FA-15(2) You can specify the labels to display during the **On** and **Off** states. These properties affect the label's font family, font size and font color properties.

FA-15(3) During run-time, when the user clicks and releases the button, µC/Probe writes the **Toggle On Value** one single time to the embedded target.

FA-15(4) When the user clicks and releases the button again, µC/Probe writes the **Toggle Off Value** one single time to the embedded target.

FA-15(5) Here you specify the labels you want to display during the **On** and **Off** states.

FA-15(6) You can specify the border colors you want to display during the **On** and **Off** states and whether or not you want to show the **Toggle** label on the corner.

A-2-7 REPEAT BUTTON PROPERTIES EDITOR

The repeat button control is a button that switches between the states **On** while held down and **Off** when released. The properties window is shown in Figure A-16:

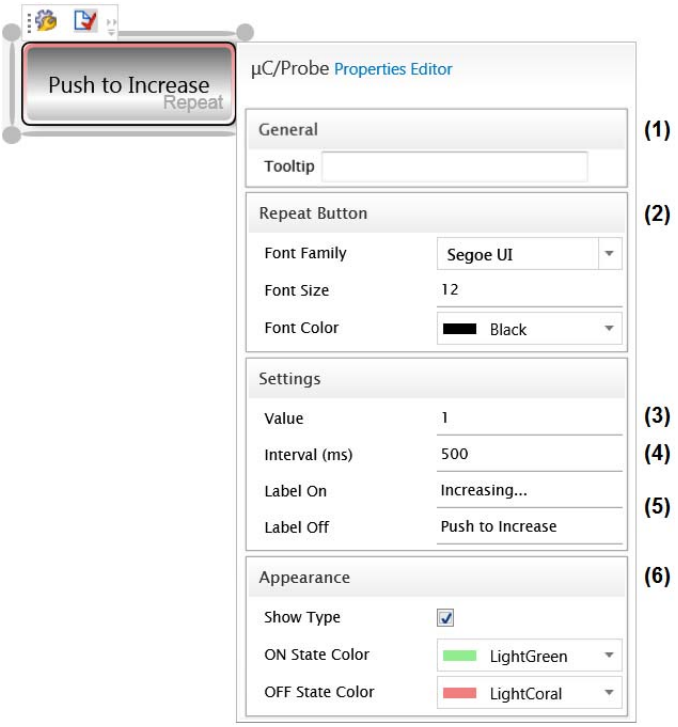


Figure A-16 **Repeat Button Properties Editor**

FA-16(1) You can include an optional **Tooltip** message that will appear with any information you want to display about the item being hovered over.

FA-16(2) You can specify the labels to display during the **On** and **Off** states. These properties affect the label's font family, font size and font color properties.

FA-16(3) During run-time, when the user holds down the repeat button, µC/Probe writes the **Value** multiple times to the embedded target until the button is released.

FA-16(4) The value gets written to the embedded target multiple times at the specified interval in milliseconds.

FA-16(5) Here you specify the labels you want to display during the **On** and **Off** states.

FA-16(6) You can specify the border colors you want to display during the **On** and **Off** states and whether or not you want to show the **Repeat** label on the corner.

A-2-8 BIT CONTROL PROPERTIES EDITOR

The bit control is part of the writable controls category in µC/Probe's toolbox. It allows you to read and write to a symbol by either toggling its bits on and off or entering the value in either decimal or hexadecimal format.

This control is perfect to represent peripheral I/O registers and the properties window is shown in Figure A-17:

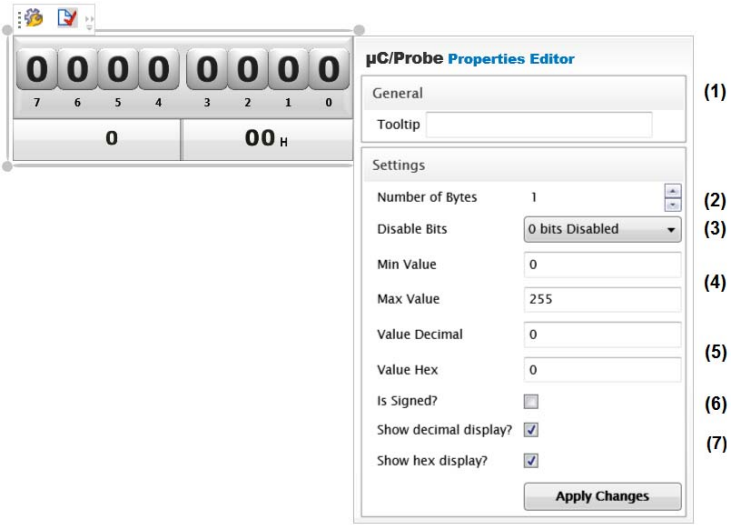


Figure A-17 **Bit Control Properties Editor**

FA-17(1) You can include an optional **Tooltip** message that will appear with any information you want to display about the item being hovered over.

FA-17(2) You can specify the number of bytes in case you want to override the original data size.

FA-17(3) You can also disable bits, in those cases where some of the bits are reserved which is typically the case with peripheral I/O registers. This drop-down gives you a list of all the bits along with a checkbox to enable and disable each bit.

FA-17(4) You can specify the minimum and maximum value range allowed.

FA-17(5) You can specify an initial value in either decimal or hexadecimal format.

FA-17(6) You can override the data type and specify whether it is signed or unsigned.

FA-17(7) Finally, you can hide the decimal and hexadecimal displays in case you want to show the bit buttons only.

A-2-9 NUMERIC UP/DOWN CONTROL PROPERTIES EDITOR

The numeric up/down control is part of the writable controls category in µC/Probe's toolbox. It allows you to write a number to the embedded target. It consists of a single line input text field and a pair of arrow buttons for stepping up or down as shown in Figure A-18:

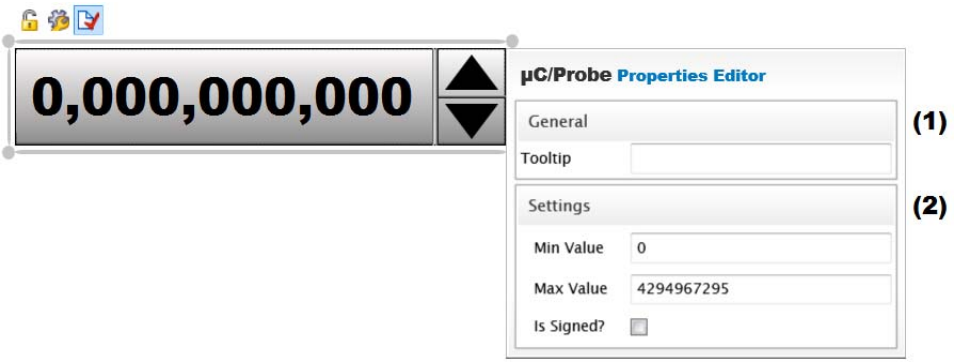


Figure A-18 **Numeric Up/Down Control Properties Editor**

FA-18(1) You can include an optional **Tooltip** message that will appear with any information you want to display about the item being hovered over.

FA-18(2) You can specify the maximum and minimum range.

A-2-10 TEXT BOX CONTROL PROPERTIES EDITOR

The text box control is part of the writable controls category in µC/Probe's toolbox. It allows you to write a number to the embedded target. It consists of a single line input text field as shown in Figure A-18:

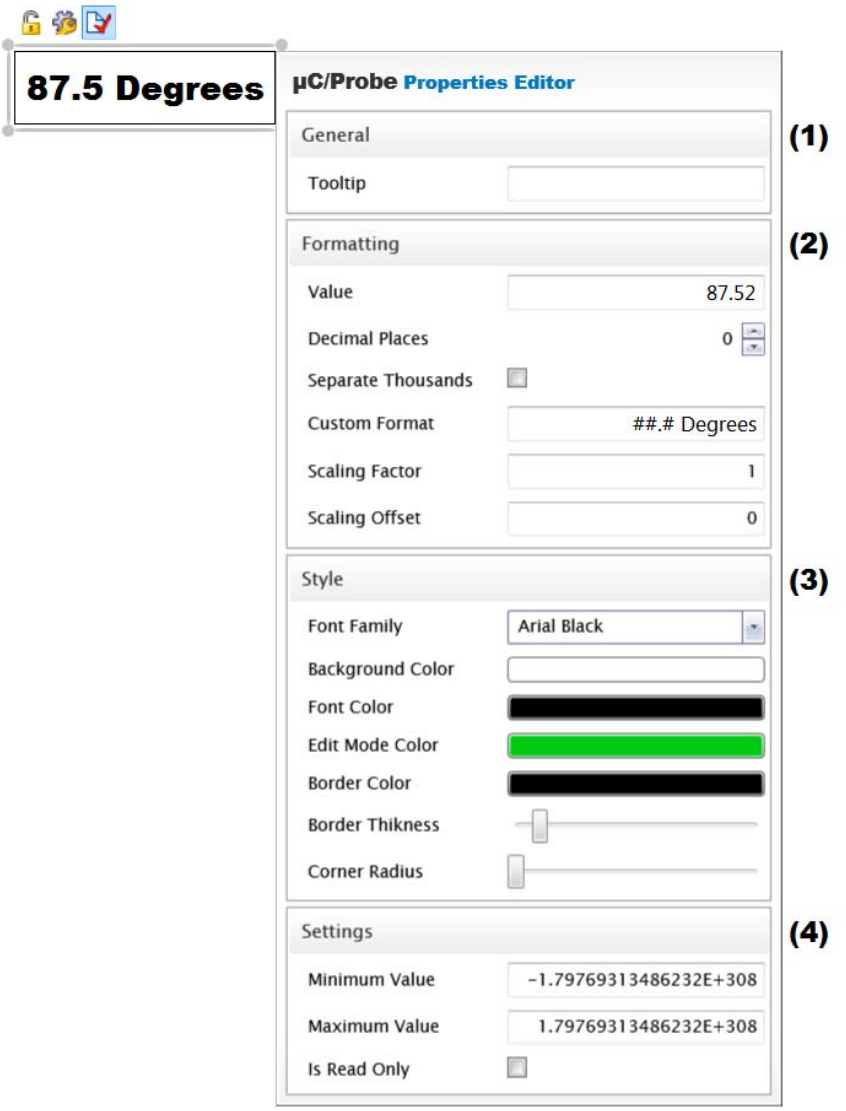


Figure A-19 **Text Box Control Properties Editor**

FA-19(1) You can include an optional **Tooltip** message that will appear with any information you want to display about the item being hovered over.

FA-19(2) You can specify an initial value, scaling options and a custom format. Standard numeric format strings are used to format common numeric types. A standard numeric format string takes the form Axx, where:

A is a single alphabetic character called the format specifier. Any numeric format string that contains more than one alphabetic character, including white space, is interpreted as a custom numeric format string.

xx is an optional integer called the precision specifier. The precision specifier ranges from 0 to 99 and affects the number of digits in the result. Note that the precision specifier controls the number of digits in the string representation of a number. It does not round the number itself.

Here is some of the most typical examples of custom format strings:

The value 123.456 with a custom format string of "C" results in "$123.46".

The value 1052.0329112756 with a custom format string of "E" results in "1.052033E+003".

The value 255 with a custom format string of "X" results in "FF".

Or, as shown in the example: The value 87.52 with a custom format string of "##.# Degrees" results in "87.5 Degrees".

For more information on custom format strings search the Microsoft documentation on *Custom Format Strings*.

FA-19(3) You can configure the style by modifying font, colors and borders.

FA-19(4) You can also specify the maximum and minimum ranges and whether or not you want to restrict access to read-only.

A-2-11 RGB COLOR PALETTE PROPERTIES EDITOR

This control is a color picker that allows you to select a color from a palette. When you select a color, the color is encoded in the ARGB additive color model format and written to the associated global variable in the embedded target.

The ARGB format is a 32-bit value where each of the 4 channels (Alpha, Red, Green and Blue) is a number between 0 and 255 that specifies the intensity of each color in the mixture, from fully-off (0) to fully-on (255). The alpha channel represents the opacity of the entire mixture of colors.

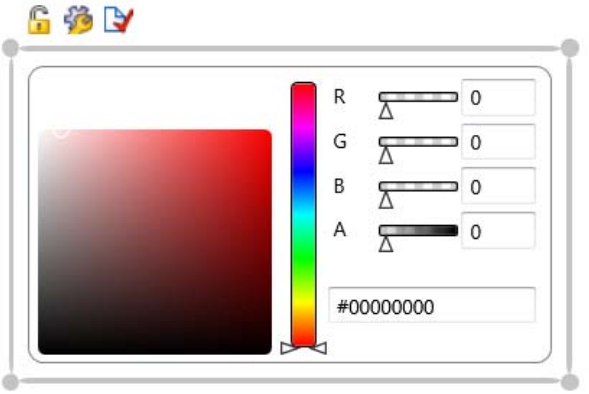


Figure A-20 **RGB Color Palette**

A-2-12 TREE VIEW PROPERTIES EDITOR

This control is probably one of the most flexible controls to read and write. It allows you to display a data structure in a tree view as shown in Figure A-21:

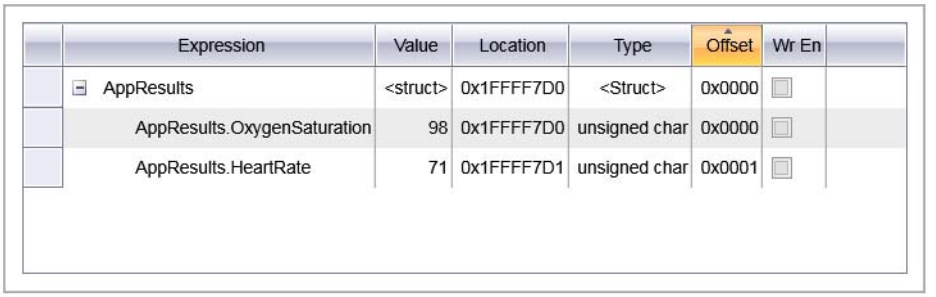


Figure A-21 **Tree View Control**

The properties editor in the tree view control is shown below in Figure A-22:

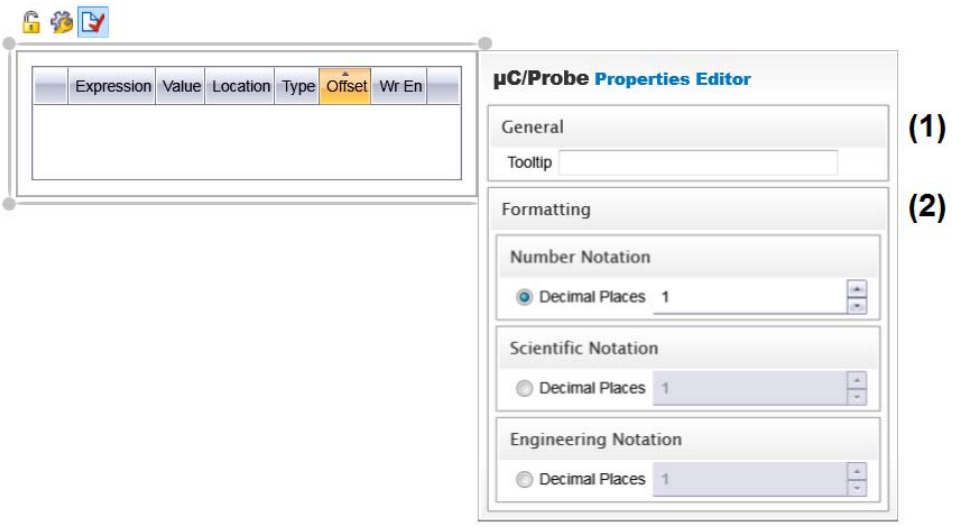


Figure A-22 **Tree View Properties Editor**

FA-22(1) You can include an optional **Tooltip** message that will appear with any information you want to display about the item being hovered over.

FA-22(2) During run-time you can configure the way each node is displayed. You can select between a decimal and hexadecimal representation. The properties editor in the tree view control allow you to specify the number of decimal places and notation in case the symbol is a float or double.

A-2-13 RADIO BUTTONS PROPERTIES EDITOR

The radio buttons control is a group of buttons that allow you to select one option from a set. You should use it for optional sets that are mutually exclusive. The radio buttons properties editor is shown in Figure A-23:

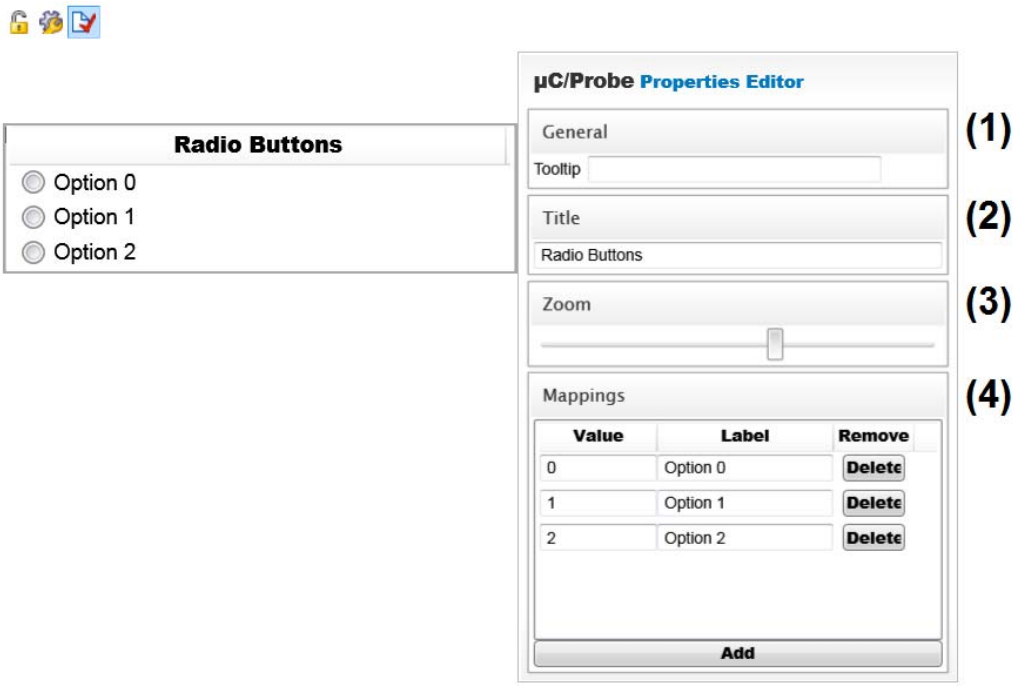


Figure A-23 **Radio Buttons Properties Editor**

FA-23(1) You can include an optional **Tooltip** message that will appear with any information you want to display about the item being hovered over.

FA-23(2) Here is where you specify the title that appears at the top of your set of options.

FA-23(3) The zoom level controls the font size.

FA-23(4) Here is where you not only provide the name for each option but also the value. Every time the user selects one of the options, the value configured in this section will be written to the associated embedded target symbol.

A-3 CHARTS

µC/Probe supports three types of charts; sample-based line charts, time-based line charts and scatter x-y charts.

A-3-1 SAMPLE-BASED LINE CHARTS

Sample-based line charts are those whose sample numbers are displayed in the horizontal axis. Figure A-24 shows the three types of sample-based line charts supported by µC/Probe:

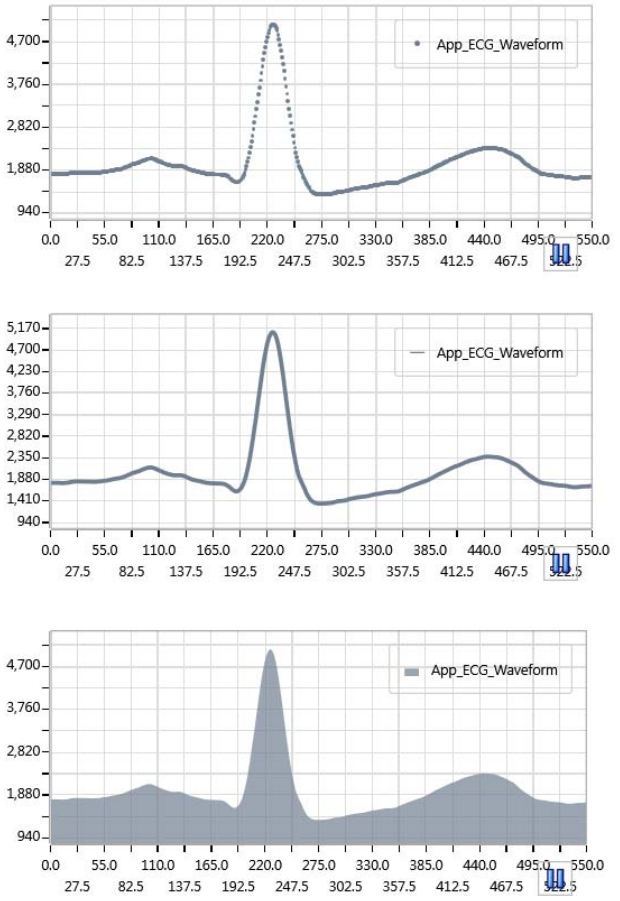


Figure A-24 **Marker, Line and Area Charts**

SAMPLE-BASED LINE CHART PROPERTIES EDITOR

All three sample-based line chart types share the same properties editor as shown in Figure A-25:

Figure A-25 **Charts Properties Editor**

FA-25(1) You can include an optional **Tooltip** message that will appear with any information you want to display about the item being hovered over.

FA-25(2) In order to change the number at which the vertical axis value starts or ends, enter a different number in the **Min** box or the **Max** box.

In order to change the interval of tick marks and chart grid lines, enter a different number in the **Tick Marks Step** box.

FA-25(3) In order to change the number at which the horizontal axis value starts or ends, enter a different number in the **Offset** box or the **Max Samples** box.

FA-25(4) µC/Probe updates charts in one of three modes:

- **Strip** Mode: similar to a chart recorder device that prints over a paper strip, µC/Probe first plots points from the left to the right side of the chart. From there, any new points are plotted at the rightmost side of the chart by shifting old points to the left.

- **Scope** Mode: similar to an oscilloscope, µC/Probe first plots points from the left to the right side of the chart. Once the plot reaches the right side of the chart, it erases the plot and begins plotting again from the left side of the chart.

- **Burst** Mode: this update mode was made for high performance applications where you want to plot array data quickly by plotting the entire array in one sweep. µC/Probe will not shift any points over the plotting area. Instead, it will erase the plot and will plot the same array again, assuming that the array is being updated by the embedded target.

FA-25(5) µC/Probe supports charts with multiple data series. That means that you can associate multiple symbols from your embedded target into one single chart.

In order to tell which trace represents a symbol in your chart, a color-coded legend with the name of the symbol is displayed over the chart. Select the legend position that better suits your needs.

FA-25(6) In case you need to convert the value points to Engineering Units (EU) before plotting in the chart, you can use the scaling factor and offset to specify the parameters of a linear conversion function. For example, if the embedded target's symbol you need to display is a 4-20mA value, you can implement the standard linear equation y = mx + b where m is the scaling factor, x is the 4-20mA value, b is the offset and y is the resulting Engineering Units (EU) value to be plotted.

SAMPLE-BASED LINE CHART SERIES EDITOR

The charts series editor allows you to configure each trace in the plotting area. The series editor is shared by all three types of charts as shown in Figure A-26:

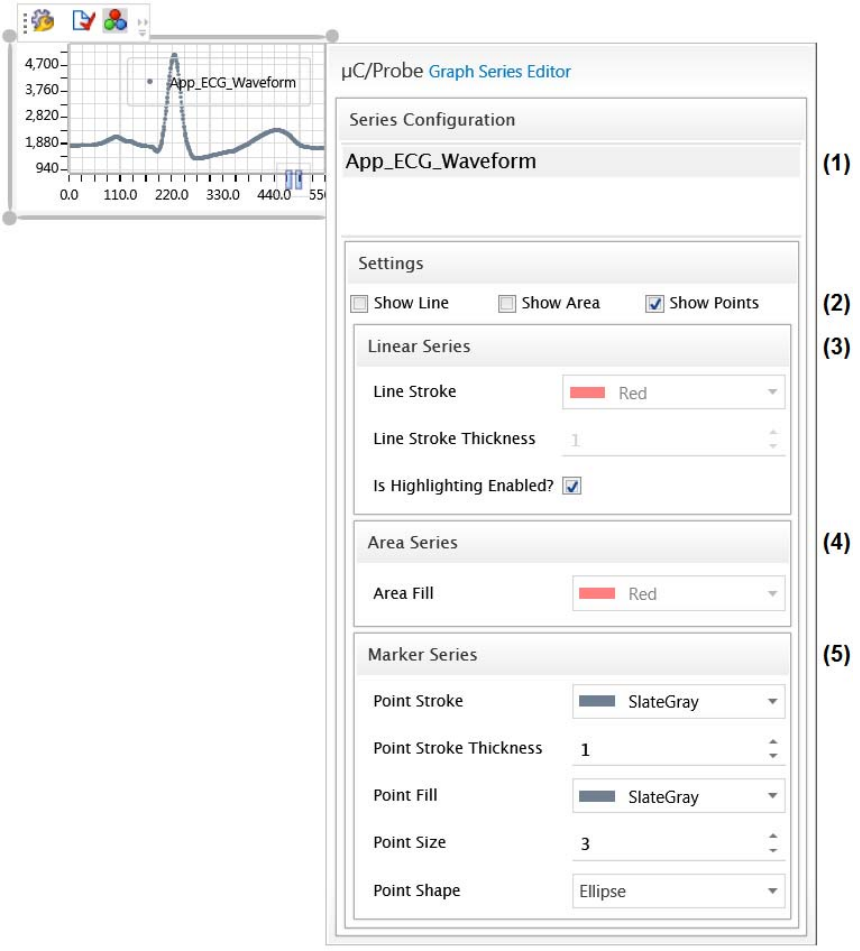


Figure A-26 **Charts Series Editor**

FA-26(1) Select the data series you want to work with. By default, the name of the data series is the same of the symbol as declared in your embedded target C files.

FA-26(2) µC/Probe supports three types of charts: marker, line and area charts. You can mix and match chart types in one single chart by selecting the corresponding check box.

FA-26(3) If the **Show Line** check box is selected, this section allows you to configure the color and thickness of the line.

FA-26(4) If the **Show Area** check box is selected, this section allows you to configure the color of the area.

FA-26(5) If the **Show Points** check box is selected, this section allows you to configure the color, thickness and shape of the points.

A-3-2 TIME-BASED LINE CHARTS

All the previous charts are designed to display samples over a horizontal axis that shows number of samples only. This section describes the time-based line charts, which are those whose samples are considered events in time and the horizontal axis represents the time.

Because the sampling rate is limited to 20 samples per second, waveforms at an operating frequency of less than 4 Hz are a good candidate to be displayed with this control. If you want to capture waveforms of higher frequency then you should use the Oscilloscope Control (Appendix I, "Oscilloscope Control" on page 155).

Contrary to the oscilloscope control, the time-based line charts do not require any target resident code. You simply drag-and-drop an instance of the time-based line chart control onto the data screen and configure it as shown in Figure A-27:

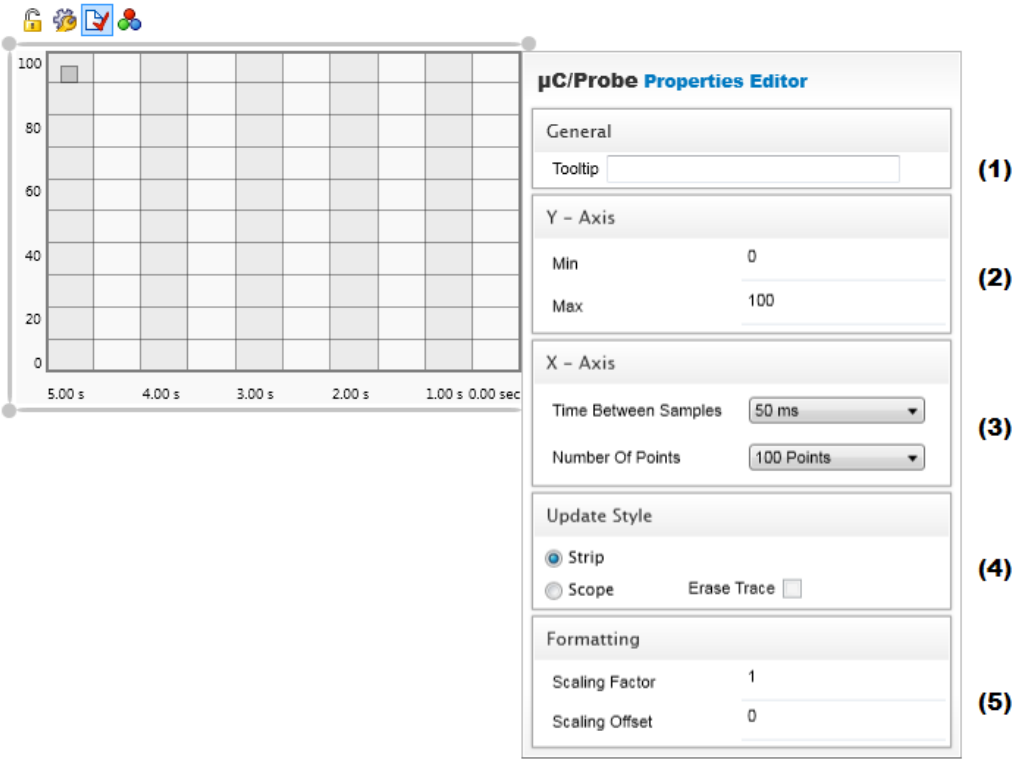


Figure A-27 **Time-based Line Chart**

FA-27(1) You can include an optional **Tooltip** message that will appear with any information you want to display about the item being hovered over.

FA-27(2) In order to change the number at which the vertical axis value starts or ends, enter a different number in the **Min** box or the **Max** box.

FA-27(3) You can configure the sampling rate and time base by specifying the amount of time between each sample acquisition and by specifying the maximum number of samples to display.

FA-27(4) µC/Probe updates this chart in one of two modes:

- **Strip** Mode: similar to a chart recorder device that prints over a paper strip, µC/Probe first plots points from the left to the right side of the chart. From there, any new points are plotted at the rightmost side of the chart by shifting old points to the left.

- **Scope** Mode: similar to an oscilloscope, µC/Probe first plots points from the left to the right side of the chart. Once the plot reaches the right side of the chart, it erases the plot and begins plotting again from the left side of the chart.

FA-27(5) In case you need to convert the value points to Engineering Units (EU) before plotting in the chart, you can use the scaling factor and offset to specify the parameters of a linear conversion function. For example, if the embedded target's symbol you need to display is a 4-20mA value, you can implement the standard linear equation y = mx + b where m is the scaling factor, x is the 4-20mA value, b is the offset and y is the resulting Engineering Units (EU) value to be plotted

A-3-3 SCATTER X-Y CHARTS

Contrary to time-based line charts where the horizontal axis is a representation of time, the scatter x-y charts allow you to use two different arrays to specify the x-y coordinates of each data point.

Use a scatter x-y chart if the data you want to plot includes pairs of values and you want to compare data points without regard of time.

The scatter x-y chart supports two modes of operation; burst and plot mode.

SCATTER X-Y CHART IN BURST MODE

Use the *burst mode* to compare sets of values stored in two different arrays.

For example, imagine a blood pressure monitor that stores the diastolic pressure in mmHg for each patient. You can use the scatter x-y chart in burst mode to compare the diastolic pressure among the patients age groups as follows.

Code Listing A-1 shows the two arrays that contain the data points.

```
static  const  CPU_INT08U  AppAgeTbl[20] =
{
    20, 28, 32, 45, 26, 64, 23, 54, 32, 54, 23, 33, 44, 21, 43, 52, 56, 62, 23, 45
};

static  const  CPU_INT08U  AppDiastolicPressureTbl[20] =
{
    64, 67, 72, 76, 65, 100, 64, 94, 80, 90, 70, 73, 81, 62, 87, 89, 91, 98, 68, 87
};
```

Listing A-1 **Two Arrays**

First you drag-and-drop an instance of the scatter x-y chart into a data screen. Then you open the symbols file (i.e. ELF file), search for the two arrays `AppAgeTbl[]` and `AppDiastolicPressure[]`, and drag-and-drop the arrays over the scatter x-y chart.

If you open the properties of the chart you will see something similar to Figure A-28:

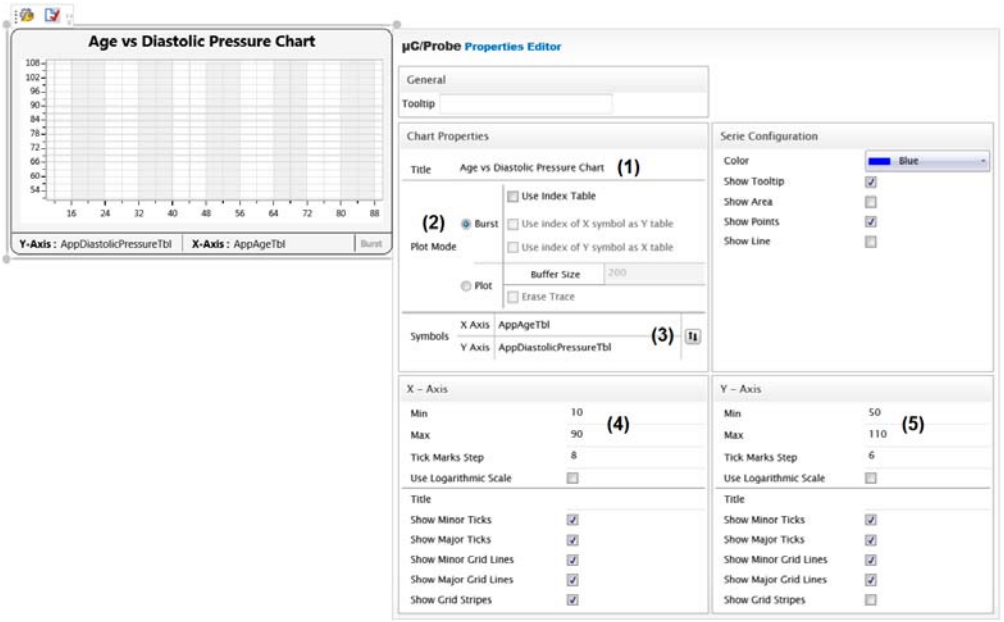


Figure A-28 **Scatter X-Y Chart Properties**

FA-28(1) Enter the title for your scatter x-y chart.

FA-28(2) µC/Probe will automatically set your plot mode to *Burst Mode* in case the axis have been mapped to arrays.

In case you only have one single array and want to use the array indexes as one of the axis, you can enable the checkbox *Use Index Table* and select which axis you want the index to be.

FA-28(3) You can use this button to swap the axis.

FA-28(4) Set the range of your data points over the horizontal axis.

FA-28(5) Set the range of your data points over the vertical axis.

During run-time, µC/Probe and the scatter X-Y chart should plot all the data points at once and will keep refreshing the plot as the values of the arrays change. You should see a scatter chart similar to the one in Figure A-29:

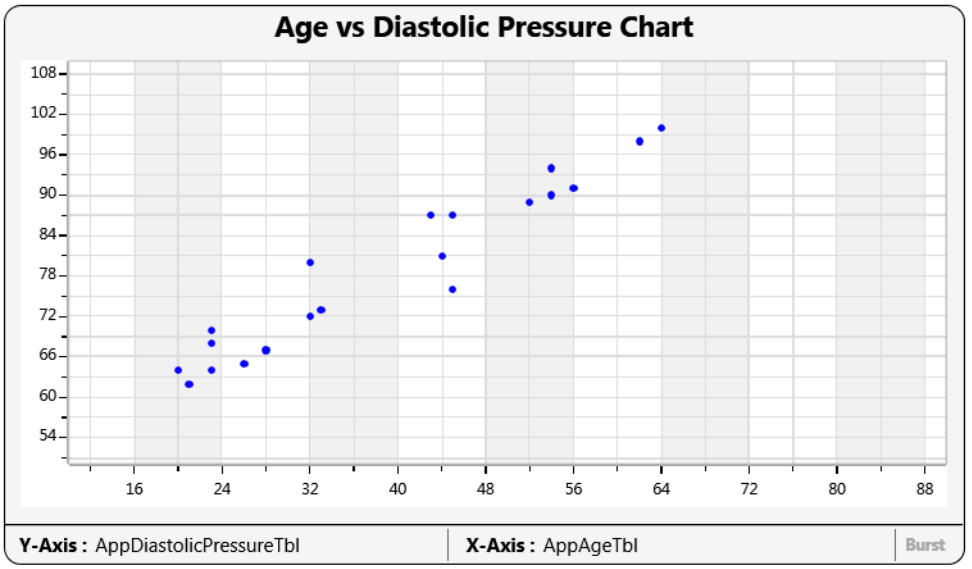


Figure A-29 **Scatter X-Y Chart in Burst Mode**

SCATTER X-Y CHART IN PLOT MODE

For the *plot mode* of the X-Y scatter chart, imagine for instance you want to monitor in real-time the power curve of a wind turbine. Your embedded application does not buffer any data and instead, it only keeps the current wind speed (meters/sec) and current power (watts) stored in the global variables `AppWindSpd` and `AppPwr` respectively.

Similar to the previous example, you drag-and-drop an instance of the scatter x-y chart into a data screen. Then, you open the symbols file (i.e. ELF file), search for the two variables `AppWindSpd` and `AppPwr`, and drag-and-drop the them over the scatter x-y chart.

If you open the properties of the chart you will see something similar to Figure A-30:

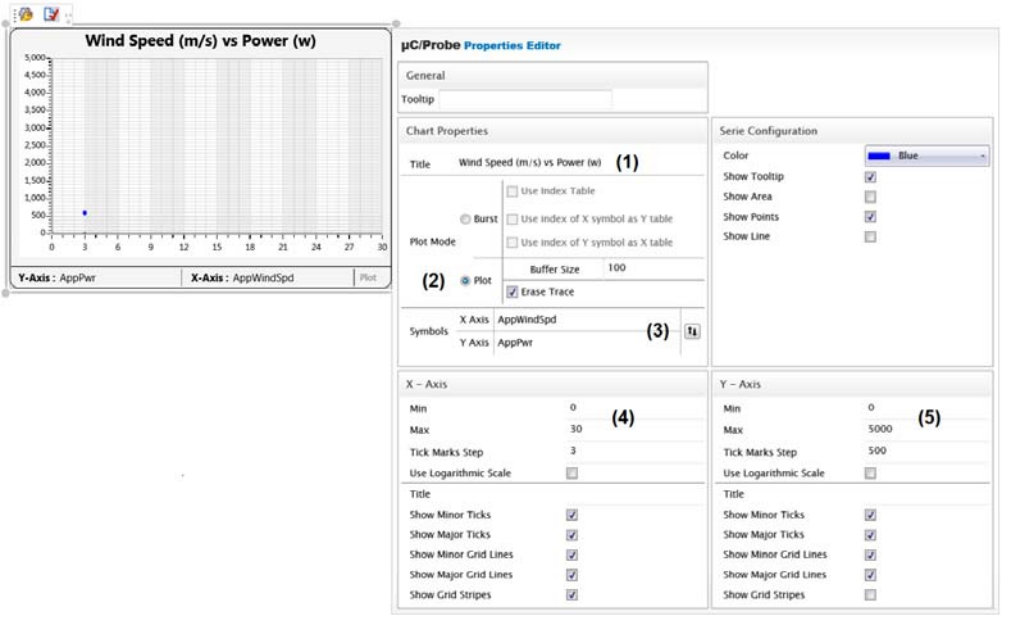


Figure A-30 **Scatter X-Y Chart Properties**

FA-30(1) Enter the title for your scatter x-y chart.

FA-30(2) µC/Probe will automatically set your chart to *Plot Mode* in case the axis have been mapped to a pair of non-array variables.

In this case, you need to specify the size of the host-side buffer that will hold the pair of values and whether or not you want the entire plot to be erased once the buffer is full.

FA-30(3) You can use this button to swap the axis.

FA-30(4) Set the range of your data points over the horizontal axis.

FA-30(5) Set the range of your data points over the vertical axis.

During run-time, the scatter chart should start to fill the plot area with one data point at a time and at the coordinate (`AppWindSpd`, `AppPwr`). Depending on the data change rate and the data collection rate, eventually the chart should look similar to the one in Figure A-31:

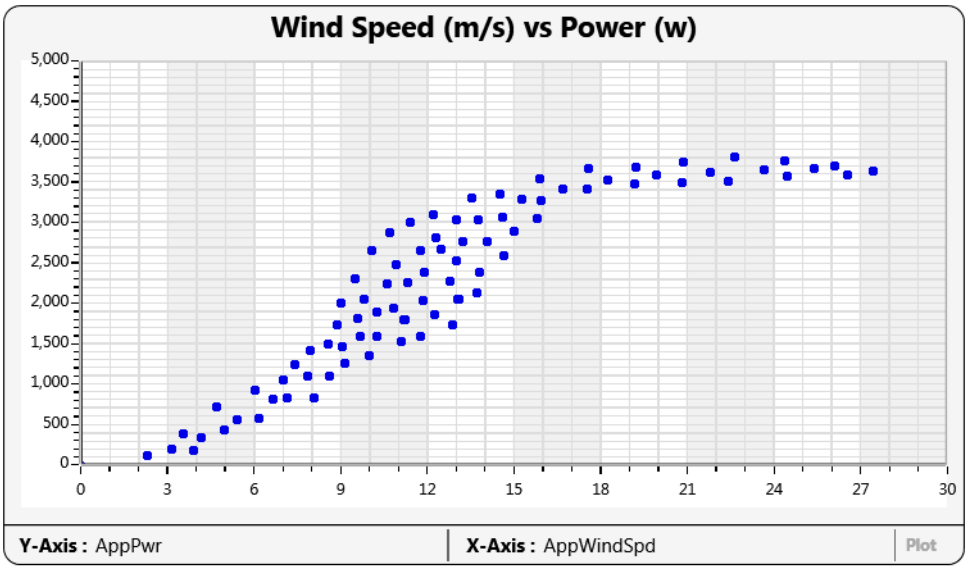


Figure A-31 **Scatter X-Y Chart in Plot Mode**

Appendix

B

Micriμm Software Awareness Screens

μC/Probe allows you to add pre-configured data screens that display some of the most popular Micriμm's software modules such as μC/OS-II, μC/OS-III and the μC/TCP-IP stack.

B-1 KERNEL AWARENESS SCREENS

The following images, show examples of the Kernel Awareness Screens for μC/OS-II and μC/OS-III. The Screens display each kernel's internal data structures in a convenient series of windows. This provides you with information about each of the active tasks in the embedded target application among other kernel objects such as semaphores, message queues, event flags, mutexes, etc.

B-1-1 μC/OS-II REQUIREMENTS

In order to display the μC/OS-II Awareness Screens you have to configure and initialize a few things in your C project:

- Make sure **`OS_DEBUG_EN`** is set to 1u
- Make sure **`OS_TASK_STAT_EN`** is set to 1u
- Make sure **`CPU_CFG_INT_DIS_MEAS_EN`** is set to 1u
- Call **`CPU_IntDisMeasMaxCurReset()`** in the first task that starts (after **`BSP_Init()`** and **`CPU_Init()`**)
- Call **`OSStatInit()`** in the first task that starts (after **`BSP_Init()`** and **`CPU_Init()`**)

- Make sure optimization is set to LOW because the tools will remove unused variables which could be needed for debug.

B-1-2 µC/OS-II KERNEL AWARENESS SCREENS

The µC/OS-II Kernel Awareness Screens include the following:

- Task List
- Semaphores
- Mailboxes
- Mutexes
- Message Queues
- Memory Partitions
- Timers
- Event Flags
- Configuration Constants

The following images show some of them.

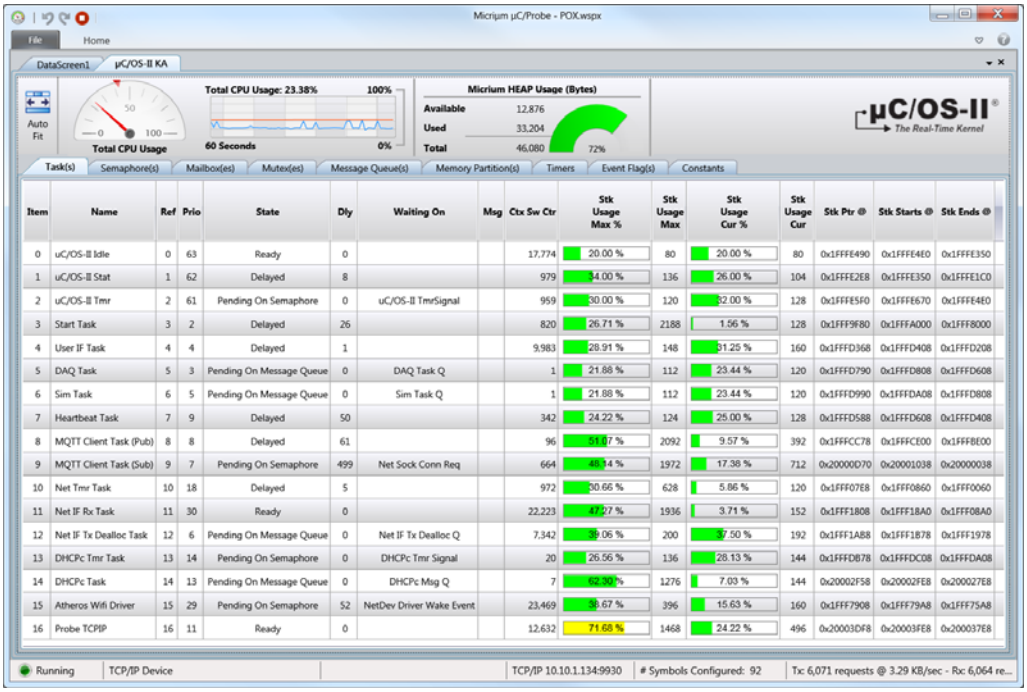


Figure B-1 **µC/OS-II Kernel Awareness Screen: Task List**

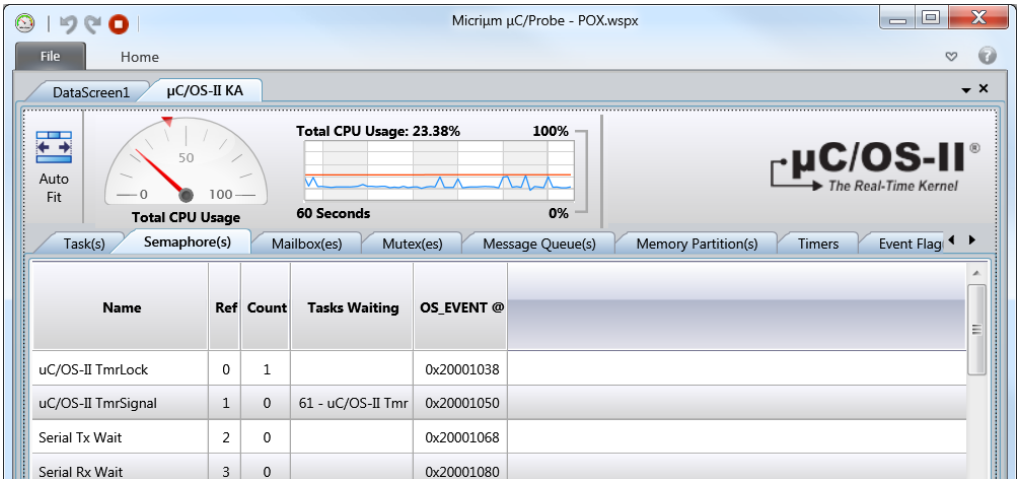


Figure B-2 **µC/OS-II Kernel Awareness Screen: Semaphores List**

B-1-3 µC/OS-III REQUIREMENTS

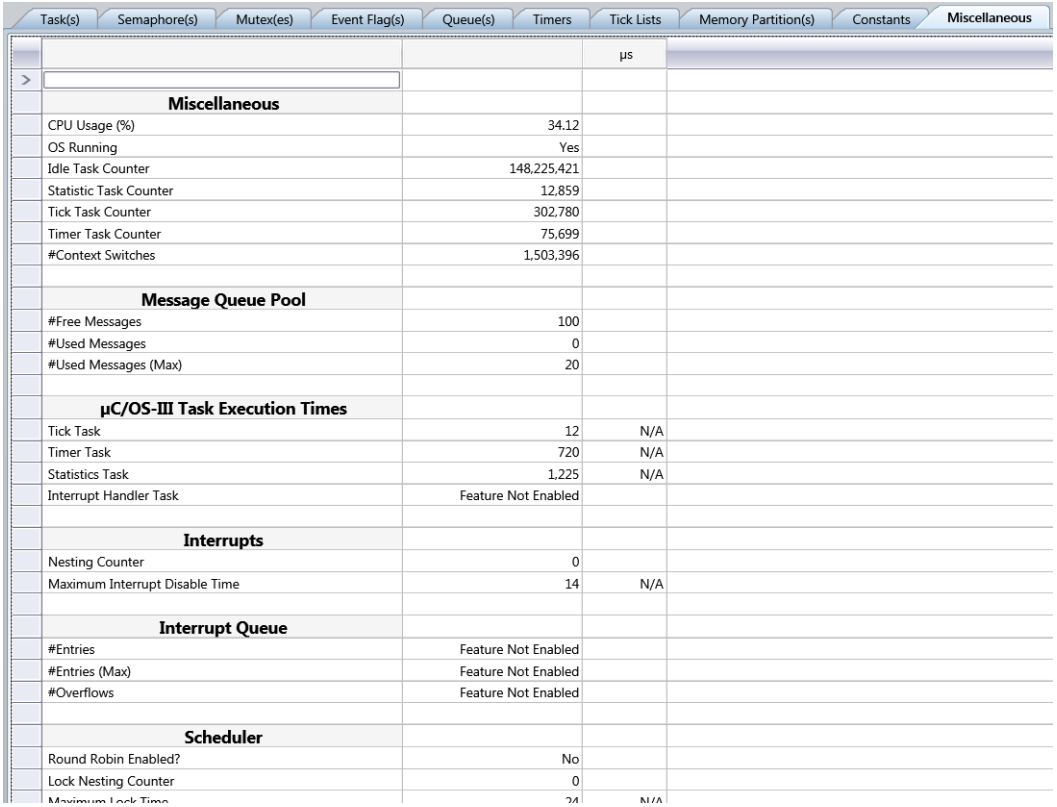
Similar to µC/OS-II, in order to display the µC/OS-III Awareness Screens you have to configure and initialize a few things in your C project:

- Make sure **OS_CFG_DBG_EN** is set to **DEF_ENABLED**
- Make sure **OS_CFG_STAT_TASK_EN** is set to **DEF_ENABLED**
- Make sure **OS_CFG_TASK_PROFILE_EN** is set to **DEF_ENABLED**
- Make sure **CPU_CFG_INT_DIS_MEAS_EN** is set to **DEF_ENABLED**
- You need to provide a *timestamp* timer which is used to measure delta time between events such as task execution time, CPU usage on a per-task basis, etc.

  For ARM Cortex-M, we use the *cycles counter* and for Renesas RX, we typically use **CMT1**. Ideally, they should be an 'UP' timer as fast as the CPU cycles but in some cases could be in the few MHz range. Implementation details are typically found in **CPU_TS_TmrInit()** and **BSP_TS_TmrInit()**.
- Call **OSStatTaskCPUUsageInit()** in the first task that starts (after **BSP_Init()** and **CPU_Init()**)
- Call **CPU_IntDisMeasMaxCurReset()** in the first task that starts (after **BSP_Init()** and **CPU_Init()**)
- Call **OSStatReset()** in the first task that starts (after **BSP_Init()** and **CPU_Init()**)
- Make sure optimization is set to LOW because the tools will remove unused variables which could be needed for debug.

B-1-4 µC/OS-III KERNEL AWARENESS SCREENS

The 10 screens are organized in a tabular view as shown in Figure B-3:



		µs
Miscellaneous		
CPU Usage (%)	34.12	
OS Running	Yes	
Idle Task Counter	148,225,421	
Statistic Task Counter	12,859	
Tick Task Counter	302,780	
Timer Task Counter	75,699	
#Context Switches	1,503,396	
Message Queue Pool		
#Free Messages	100	
#Used Messages	0	
#Used Messages (Max)	20	
µC/OS-III Task Execution Times		
Tick Task	12	N/A
Timer Task	720	N/A
Statistics Task	1,225	N/A
Interrupt Handler Task	Feature Not Enabled	
Interrupts		
Nesting Counter	0	
Maximum Interrupt Disable Time	14	N/A
Interrupt Queue		
#Entries	Feature Not Enabled	
#Entries (Max)	Feature Not Enabled	
#Overflows	Feature Not Enabled	
Scheduler		
Round Robin Enabled?	No	
Lock Nesting Counter	0	
Maximum Lock Time	24	N/A

Figure B-3 **Kernel Awareness Screen: Miscellaneous**

Figure B-4 shows the information displayed for each task. The columns can be sorted and the tri-color bar graphs highlight those tasks reaching their maximum stack space, which is a typical bug when developing embedded systems:

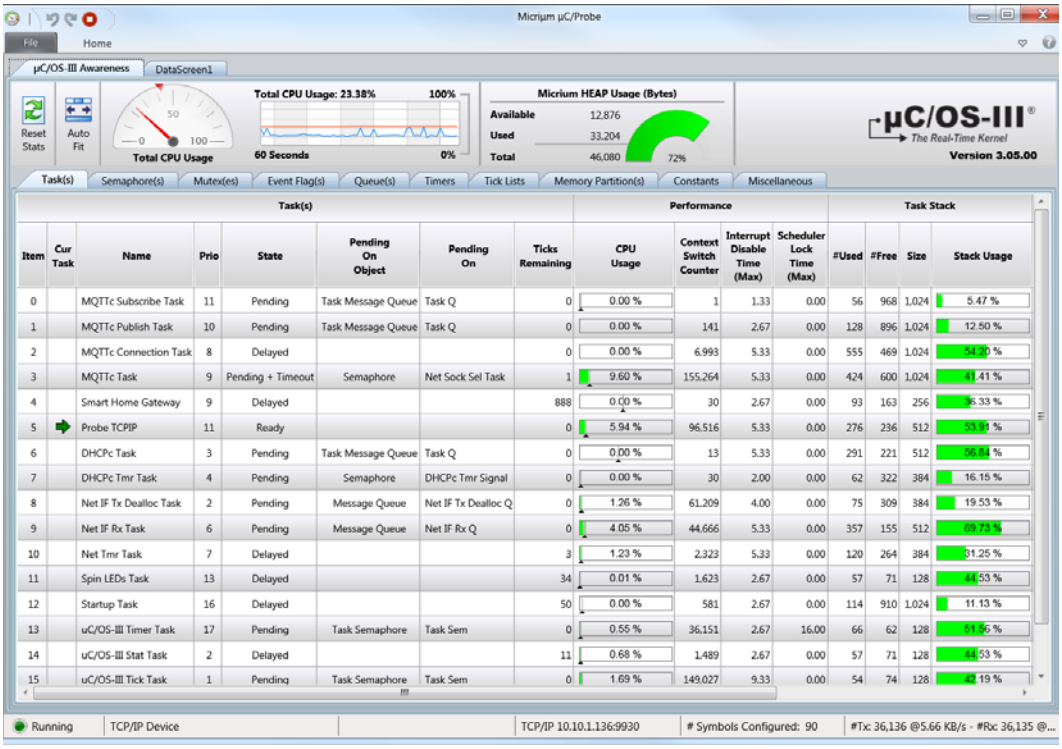


Figure B-4 **Kernel Awareness Screen: Task List**

B-1-5 ADDING A KERNEL AWARENESS SCREEN

To include a kernel awareness screen in your µC/Probe workspace, first make sure you load the ELF file as described in Chapter 3, “Loading an ELF file” on page 17. Then you can go to the **Workspace Explorer** on the right hand side and click on the button **Screens** and select either **µC/OS-II** or **µC/OS-III** kernel awareness screen as illustrated in Figure B-5:

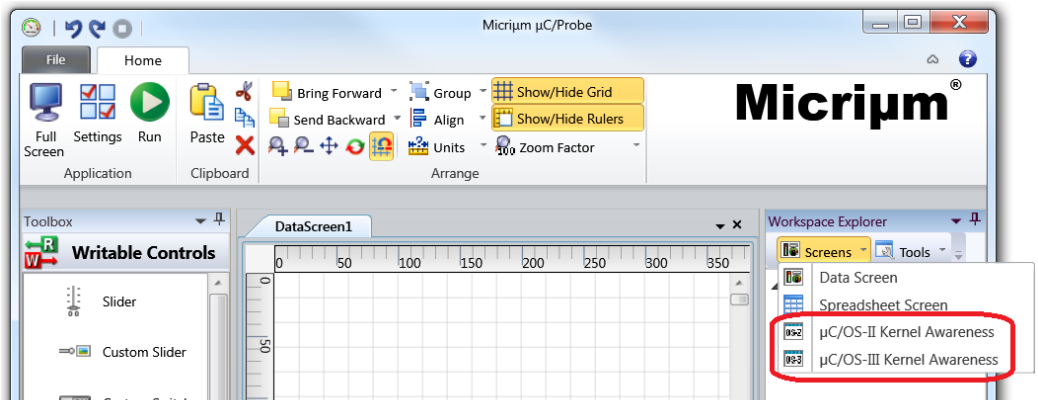


Figure B-5 **Kernel Awareness Screens**

B-2 µC/TCP-IP AWARENESS SCREENS

This includes a series of screens that display the run-time state of the µC/TCP-IP stack including:

- Stack Layers Overview
- TCP Resources and Statistics
- UDP Resources and Statistics
- ARP Resources and Statistics
- IP Statistics
- Socket Resources and Status List
- Connection Resources and Statistics
- Timer Resources and Statistics
- Interface Resources and Statistics

The information in these screens allows you to optimize the performance and footprint of your µC/TCP-IP stack. Here are some screen captures:

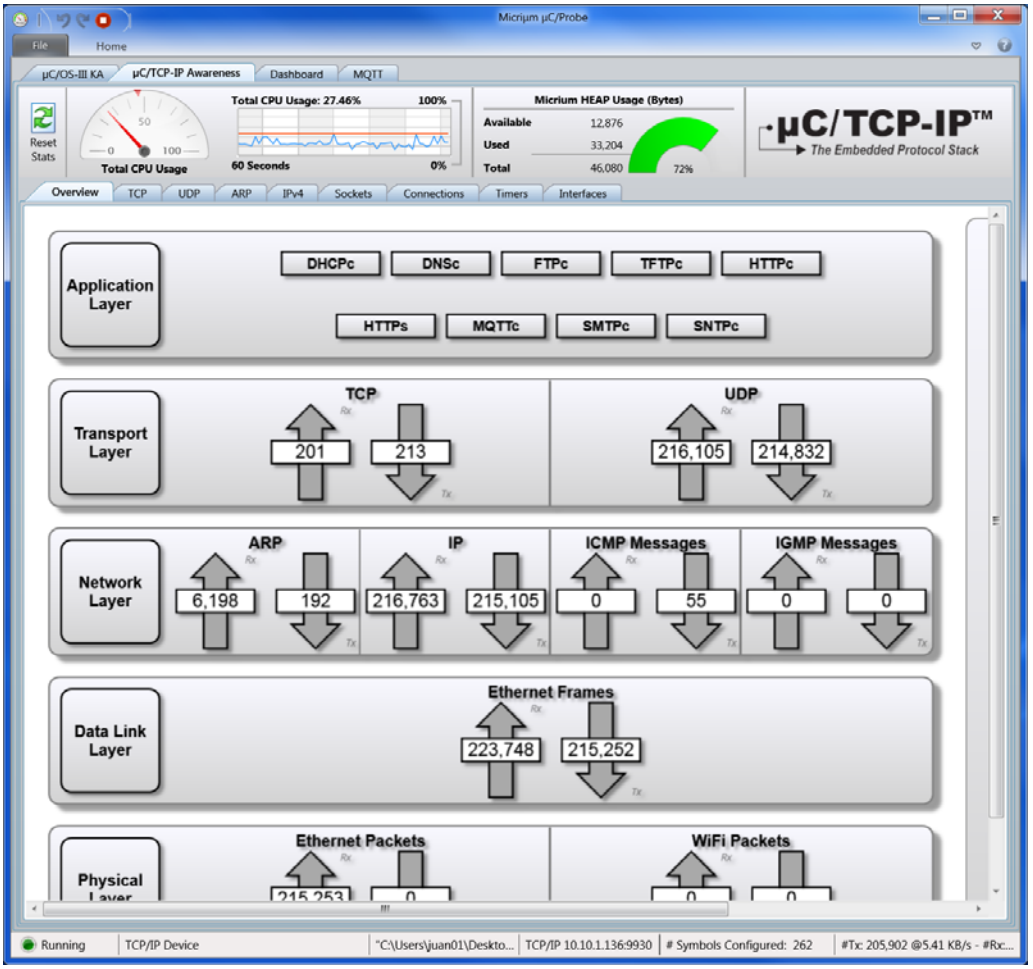


Figure B-6 **µC/TCP-IP Awareness Screens: Overview**

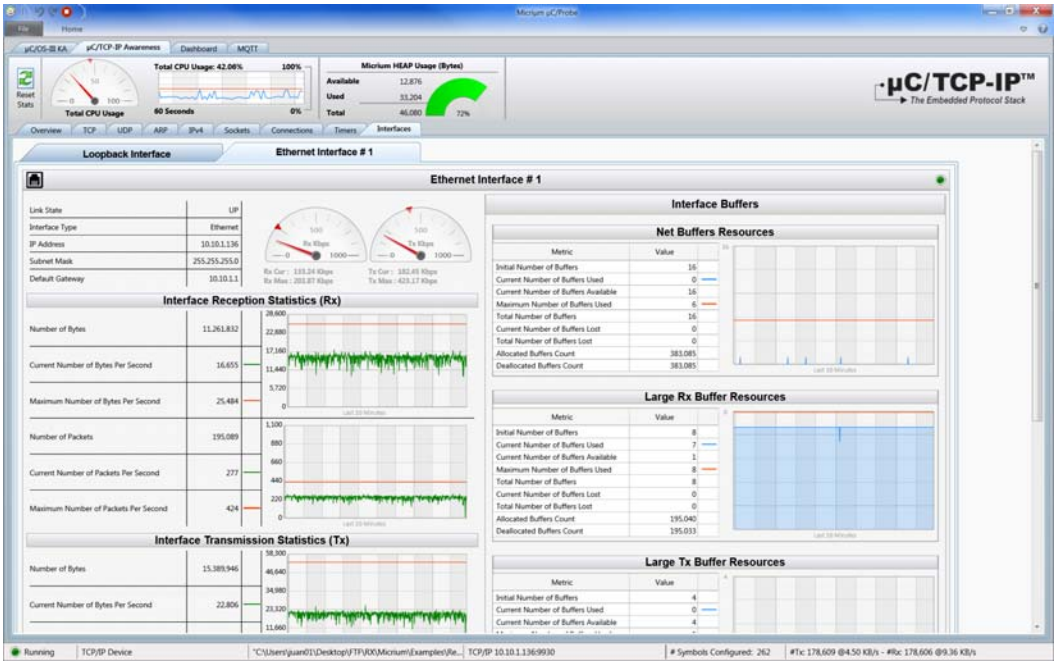


Figure B-7 **µC/TCP-IP Awareness Screens: Interfaces**

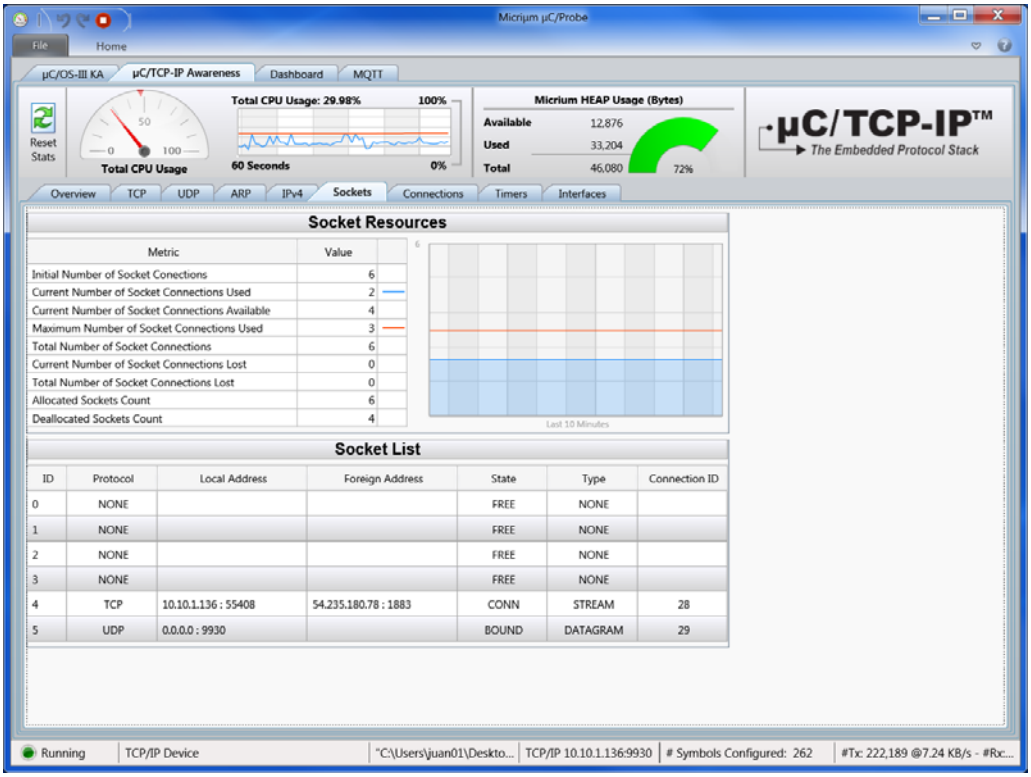


Figure B-8 **µC/TCP-IP Awareness Screens: Sockets**

B-2-1 ADDING A µC/TCP-IP AWARENESS SCREEN

To include a µC/TCP-IP awareness screen in your µC/Probe workspace, first make sure you enable the statistics and error counters in your µC/TCP-IP configuration declared in **net_cfg.h** as shown in Listing B-1:

```
#define  NET_CTR_CFG_STAT_EN                    DEF_ENABLED

#define  NET_CTR_CFG_ERR_EN                     DEF_ENABLED
```

Listing B-1 **µC/TCP-IP Configuration File: net_cfg.h**

Then make sure you load the ELF file as described in Chapter 3, “Loading an ELF file” on page 17. Then you can go to the **Workspace Explorer** on the right hand side and click on the button **Screens** and select **µC/TCP-IP** Awareness Screen as illustrated in Figure B-9:

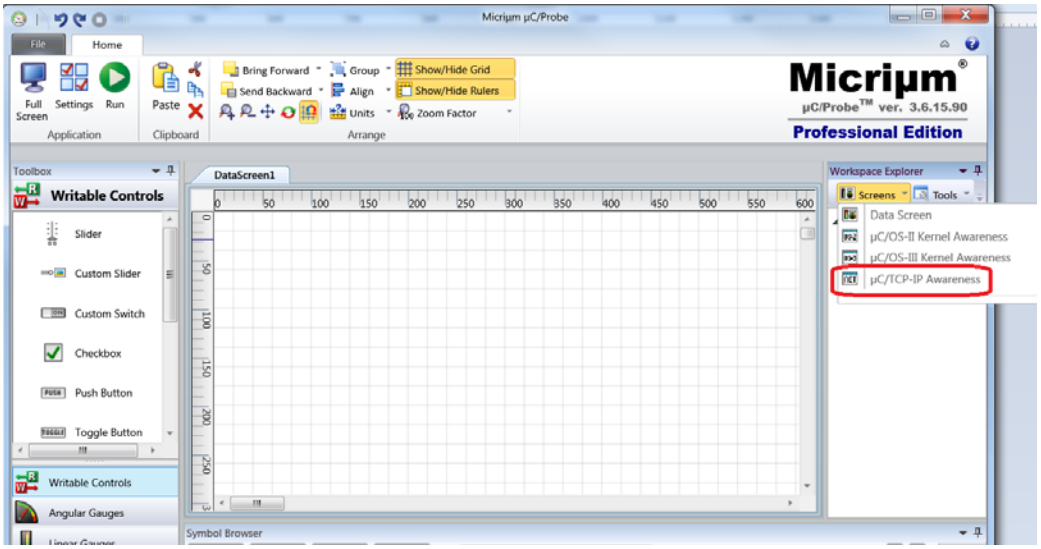


Figure B-9 **µC/TCP-IP Awareness Screens: Adding a Screen**

Appendix

C

Terminal Window Control

µC/Probe provides an option to enable debug traces to output the embedded target's activity via any of the communication interfaces supported by µC/Probe. A trace message is displayed in a terminal window control in µC/Probe, by calling a function `ProbeTermTrcPrint()` from your embedded application as illustrated in Figure C-1. Additionally, you can prefix the messages with special tags that µC/Probe will replace with icons that you get to choose.

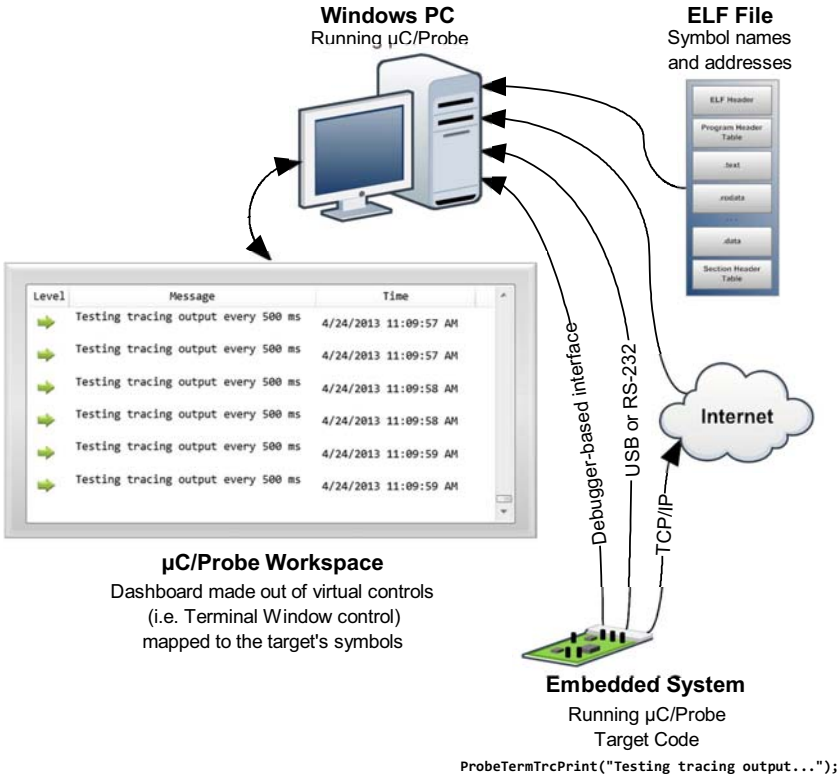


Figure C-1 **Terminal Window Control - Trace Interface**

At the same time, µC/Probe provides the option to enable a command-line interface to the embedded target. A command-line interface allows the user to issue a command to the target from a terminal window control in µC/Probe. Examples of command lines are `ipconfig`, `dir` or whatever command the programmer wants to implement in the embedded target.

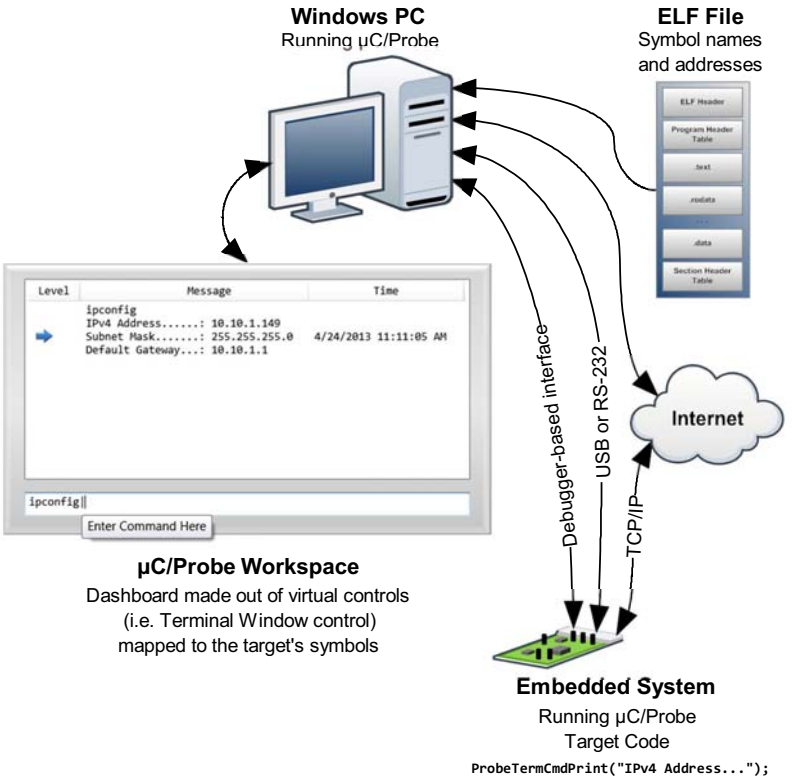


Figure C-2 **Terminal Window Control - Command Line Interface**

This appendix will introduce you to the debug trace and command-line tools available in µC/Probe. It will show you how to configure the control in µC/Probe. More information on the Terminal Window control such as how to include it in your embedded target code and make use of it are in the target version of the µC/Probe manual.

C-1 TERMINAL WINDOW CONTROL CONFIGURATION

The terminal window control is found in the miscellaneous category of µC/Probe's toolbox. Once you drag and drop an instance of this control onto the data screen, you do not need to associate it with any of the embedded target's symbols from the symbol browser, as this is done automatically, assuming you have included the required target code to support terminal window as described in the target version of the µC/Probe manual.

You can access the properties tool bar by moving the mouse over the terminal window control. The tool bar shown in Figure C-3 appears for you to select between one of the two configuration categories:

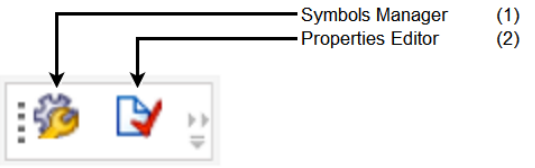


Figure C-3 **Terminal Window Control Toolbar**

FC-3(1) The Symbols Manager is common for all virtual controls and indicators, see Chapter 8, "Associating Symbols to Virtual Controls and Indicators" on page 37 for more information on the Symbols Manager.

FC-3(2) The Properties Editor is similar among most of the virtual controls and indicators and the next sections will describe how to use the Properties Editor.

C-2 PROPERTIES EDITOR

The properties editor for the terminal window control is shown in Figure C-4:

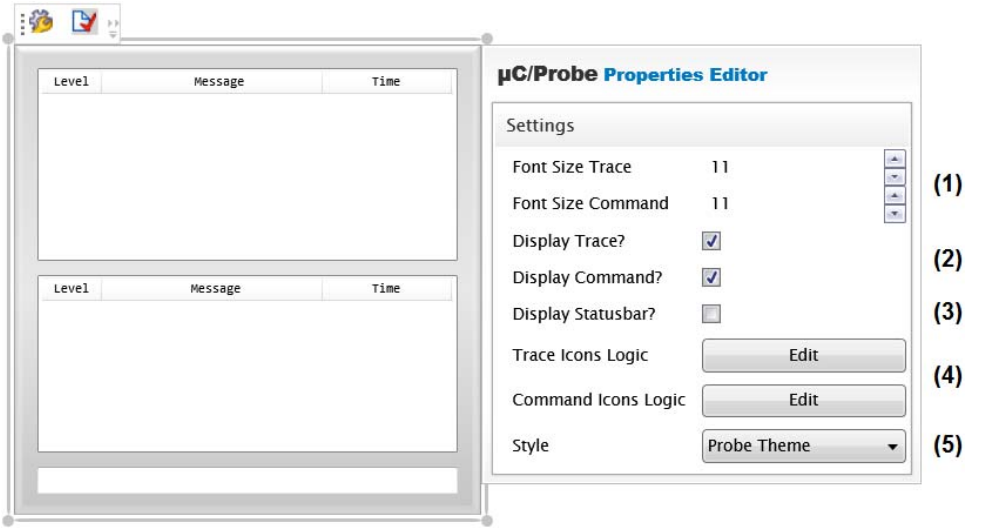


Figure C-4 **Terminal Window Control Properties Editor**

FC-4(1) You can change the font size for each panel.

FC-4(2) You can hide one of the interfaces.

FC-4(3) The terminal window control has a status bar that allows you to see the communication status.

FC-4(4) The icons mapping interface allows you to configure an icon to be displayed whenever a keyword is found in the message back from the target. This is useful if you want to bring more attention to a set of messages, such as warnings or errors.

FC-4(5) The terminal window control is also available in a high contrast theme.

Appendix

D

µC/Trace Triggers Control

µC/Trace is a runtime diagnostics tool for embedded software systems based on µC/OS-III. µC/Trace gives developers an unprecedented insight into the runtime behavior, which allows for reduced troubleshooting time and improved software quality, performance and reliability. Complex software problems which otherwise may require many hours or days to solve, can with µC/Trace be understood quickly, often in a tenth of the time otherwise required. This saves you many hours of troubleshooting time. Moreover, the increased software quality resulting from using µC/Trace can reduce the risk of defective software releases, causing damaged customer relations.

The insight provided by µC/Trace also allows you to find opportunities for optimizing your software. You might have unnecessary resource conflicts in your software, which are "low hanging fruit" for optimization and where a minor change can give a significant improvement in real-time responsiveness and user-perceived performance. By using µC/Trace, software developers can reduce their troubleshooting time and thereby get more time for developing new valuable features. This means a general increase in development efficiency and a better ability to deliver high-quality embedded software within budget.

µC/Trace provides more than 20 interconnected views of the runtime behavior, including task scheduling and timing, interrupts, interaction between tasks, as well as user events generated from your application as shown in Figure D-1. µC/Trace can be used side-by-side with a traditional debugger and complements the debugger view with a higher level perspective, ideal for understanding the complex errors where a debugger's perspective is too narrow.

µC/Trace is more than just a viewer. It contains several advanced analyses developed since 2004, that helps you faster comprehend the trace data. For instance, it connects related events, which allows you to follow messages between tasks and to find the event that triggers a particular task instance. Moreover, it provides various higher level views such as the Communication Flow graph and the CPU load graph, which make it easier to find anomalies in a trace.

µC/Trace does not depend on additional trace hardware, which means that it can be used in deployed systems to capture rare errors which otherwise are hard to reproduce.

Figure D-1 **µC/Trace Analyzer Windows**

The µC/Trace solution consists of three parts:

- The PC application (µC/Trace), used to analyze the recordings as shown in Figure D-1.
- A trace recorder library that integrates with µC/OS-III, provided in C source code.
- Optionally, µC/Probe can be used for the target system connection.

The PC application µC/Trace has been developed for Microsoft Windows.

The trace recorder library stores the event data in a RAM buffer, which is uploaded on request to the host PC using your existing debugger connection or µC/Probe.

And finally, you can use µC/Probe and a special control designed for µC/Trace called µC/Trace Trigger Control, to trigger a recording and launch the µC/Trace analyzer. The µC/Trace Trigger Control is shown in Figure D-2:

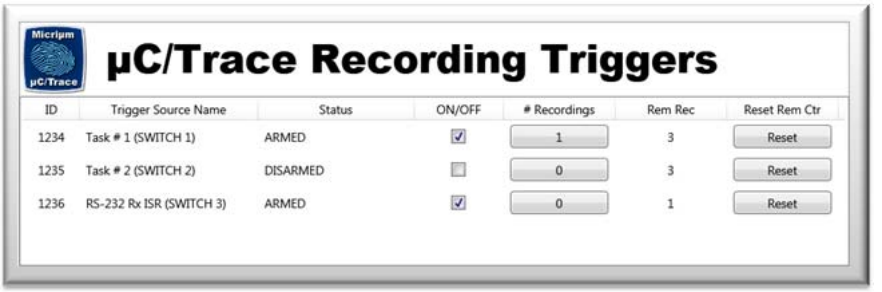


Figure D-2 **µC/Trace Trigger Control**

For more information on µC/Trace, go online to http://micrium.com/tools/uctrace

Appendix E

Spreadsheet Control

Microsoft® Excel® is a very widely used spreadsheet application developed by the Microsoft® Corporation. It features calculation and graphing tools among other features that now you can embed in your µC/Probe data screens thanks to a technology offered by Microsoft® called *Automation*.

Automation to Excel® allows you to programmatically perform actions such as creating a new workbook, adding data to the workbook, or creating charts. Virtually, all of the actions that you can perform manually through the user interface can also be performed programmatically by using automation.

µC/Probe makes use of automation to allow you to create a workbook, map the spreadsheet cells to your embedded target symbols and make use of all the features Excel® has to offer.

This appendix will show you how to drag-and-drop an instance of Excel® into a µC/Probe data screen and how to associate your embedded target symbols to the spreadsheet's cells.

Microsoft®, Excel® and Windows® are either registered trademarks or trademarks of Microsoft® Corporation in the United States and/or other countries. The use of Microsoft® Excel® automation features by µC/Probe does not imply that Micriμm and/or µC/Probe have any Microsoft® affiliation, sponsorship, endorsement, certification, or approval.

E-1 ADDING AN INSTANCE OF THE SPREADSHEET CONTROL

The Spreadsheet Control is only available on the Professional Edition of µC/Probe. Additionally, you need to have Microsoft® Excel® version 2003 or newer installed on your Windows® PC.

In order to add an instance of Excel® into a Data Screen, go to the µC/Probe toolbox in the *Advanced* controls category and drag-and-drop the icon labeled *Spreadsheet* into a data screen as illustrated in Figure E-1:

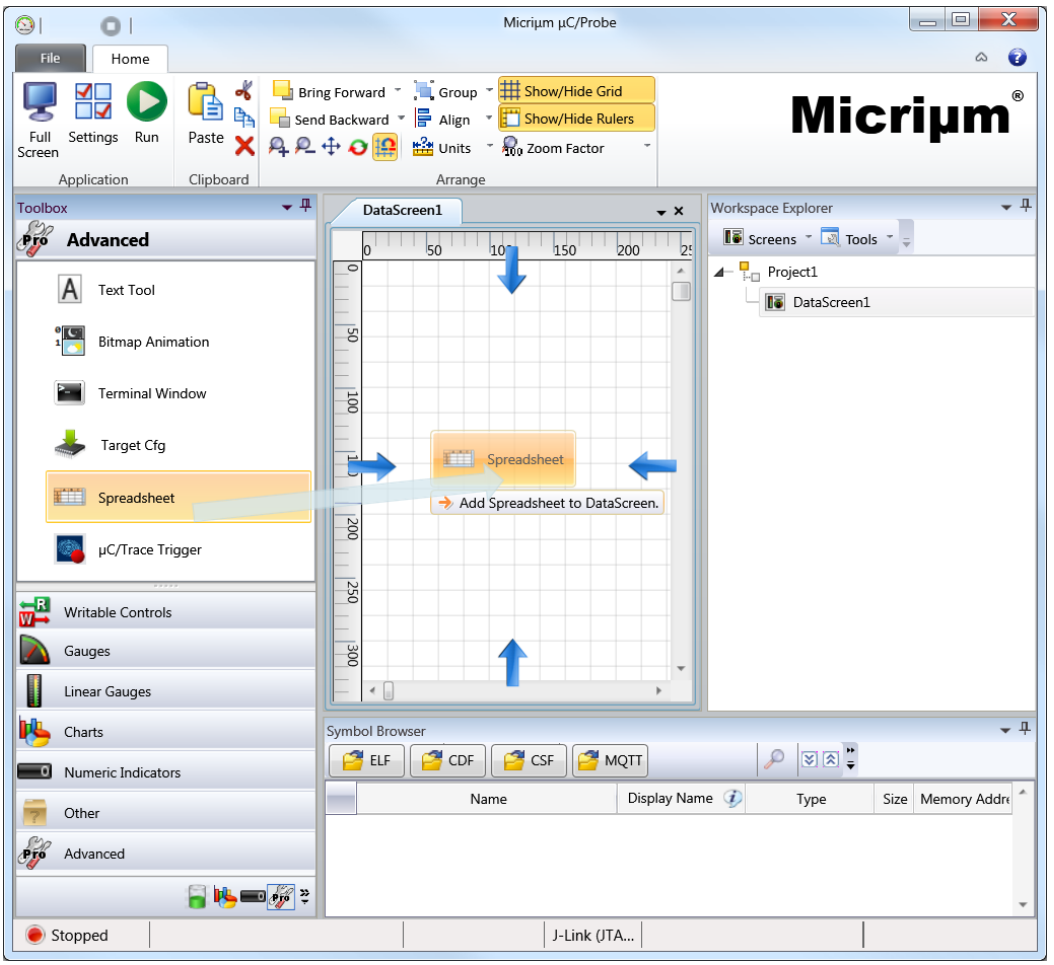


Figure E-1 **Adding an Instance of the Spreadsheet Control**

E-2 THE SPREADSHEET CONTROL PANEL

Once you have successfully added an instance of Excel® into a data screen, not only a new Microsoft Excel workbook will be launched but also the *Spreadsheet Control Panel* within µC/Probe shown in Figure E-2 will appear. This control panel will allow you to perform the functions indicated in Figure E-2:

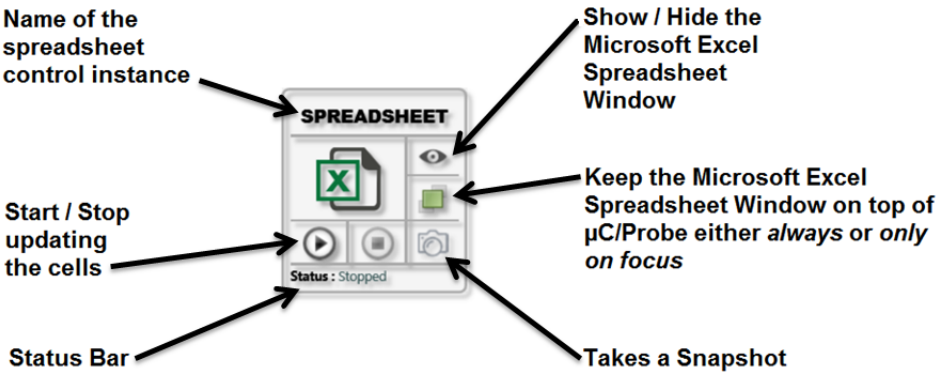


Figure E-2 **Spreadsheet Control Panel**

E-3 CONFIGURING THE SPREADSHEET

The next step is to configure the newly created Microsoft Excel Spreadsheet. In essence, it is time to associate your embedded target symbols from the *Symbol Browser* to any cell in the spreadsheet as illustrated in Figure E-3:

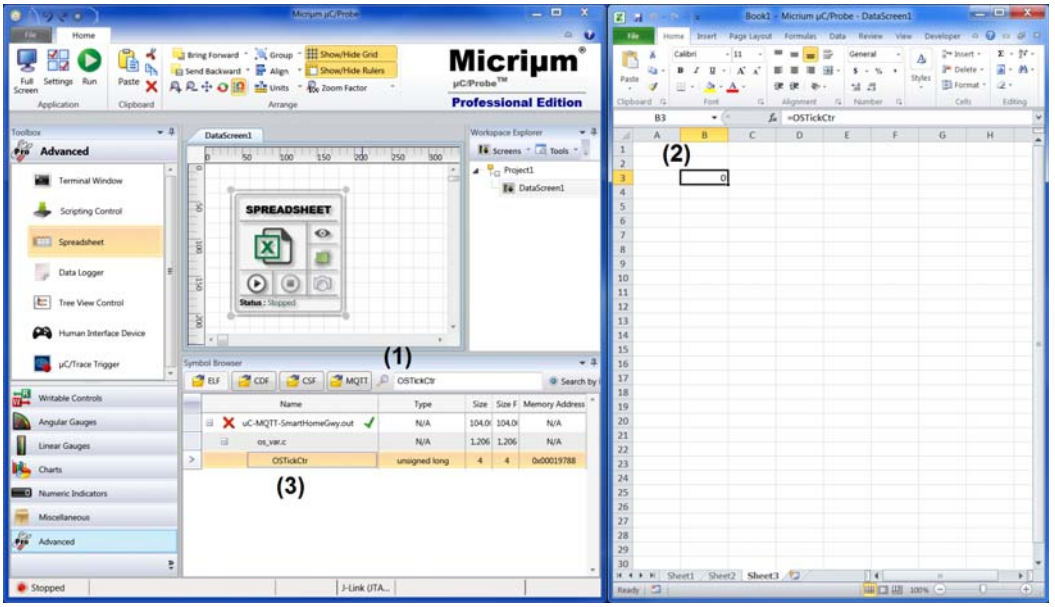


Figure E-3 **Configuring the Spreadsheet**

FE-3(1) You open a symbols file (i.e. ELF file) and browse through it as described in Chapter 3, “µC/Probe Symbol Browser” on page 16.

FE-3(2) Select the Cell in Microsoft Excel that you want to configure.

FE-3(3) Go back to the *Symbol Browser* and double-click over the symbol you want to configure.

E-3-1 CONFIGURING THE SPREADSHEET WITH MORE COMPLEX DATA TYPES

The previous section showed how to configure the spreadsheet by first selecting the cell, then selecting the symbol from the symbol browser and finally double-clicking over the symbol. If said symbol is one of the basic data types such as an `int`, `float` or `char` then there is really not much more to it. The cell will display the value of said symbol during run-time period.

However, if the symbol you select is rather complex such as an array or data structure, then an additional dialog window will be displayed when you double-click over the symbol. The following sections show you examples of adding an array and a data structure.

ARRAYS

If you associate a cell with an array the dialog shown in Figure E-4 will be presented:

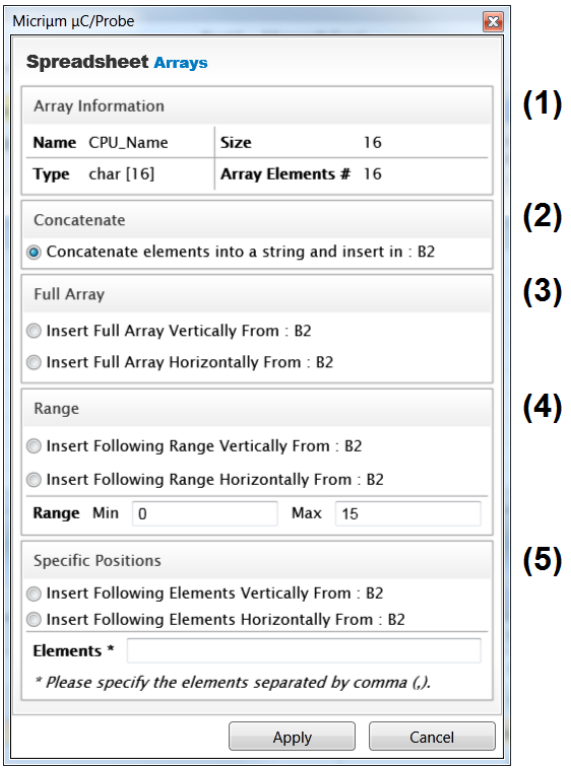


Figure E-4 **Spreadsheet: Arrays**

FE-4(1) The section at the top present information about the array (i.e. Name, Data Type and Size).

FE-4(2) This is the default mode when the data type is string. It tells µC/Probe to include the array as a representation of a string. Each element in the arry will be concatenated and placed inside one single cell.

FE-4(3) The Full Array mode, means that each element in the array will occupy a cell. The first element will be placed in the cell you selected and the rest of element will be added in the next cells either horizontally or vertically.

FE-4(4) The Range mode is similar to the previous one, except that you also get the opportunity to specify a range of elements in the array.

FE-4(5) The last mode is even more flexible as you can specify which elements go into each cell. You simply configure the array indexes you want included in a comma separated list. For example if you just want to include 4 elements: 1,3,6,9.

DATA STRUCTURES

When you add a data structure you will not be presented with any special dialog windows. µC/Probe will simply create a table in Excel as illustrated in Figure E-5:

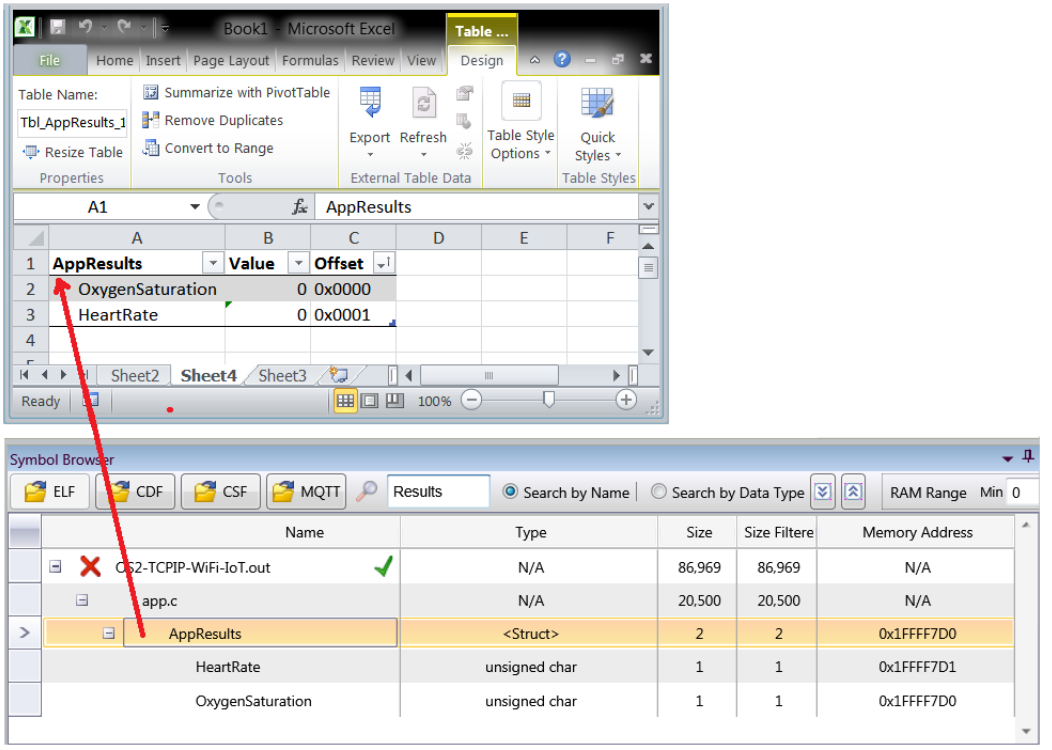


Figure E-5 **Spreadsheets: Data Structures**

E-4 OTHER FEATURES

E-4-1 INTELLISENSE

The previous section E-3 “Configuring the Spreadsheet” on page 132 showed you how to configure the spreadsheet by adding one embedded target symbol at a time. As another option you can add all the symbols declared in a single C file at once. You simply go to the symbol browser and instead of selecting a symbol, you select the parent node which is the name of the C file and double click on it. That in turn will include all the symbols in the

spreadsheet's list of names. This way, you can type in a few letters of the symbol you want to add in a cell and Excel will show you a list of options. For example Figure E-6 shows the Excel spreadsheet after doing a double-click over the file `app.c` in µC/Probe's Symbol Browser. Notice that the list of options when you type in `=App` is made with all the variables declared in `app.c` that start with `App`. With this feature, you can use the variable names even as part of Excel formulas.

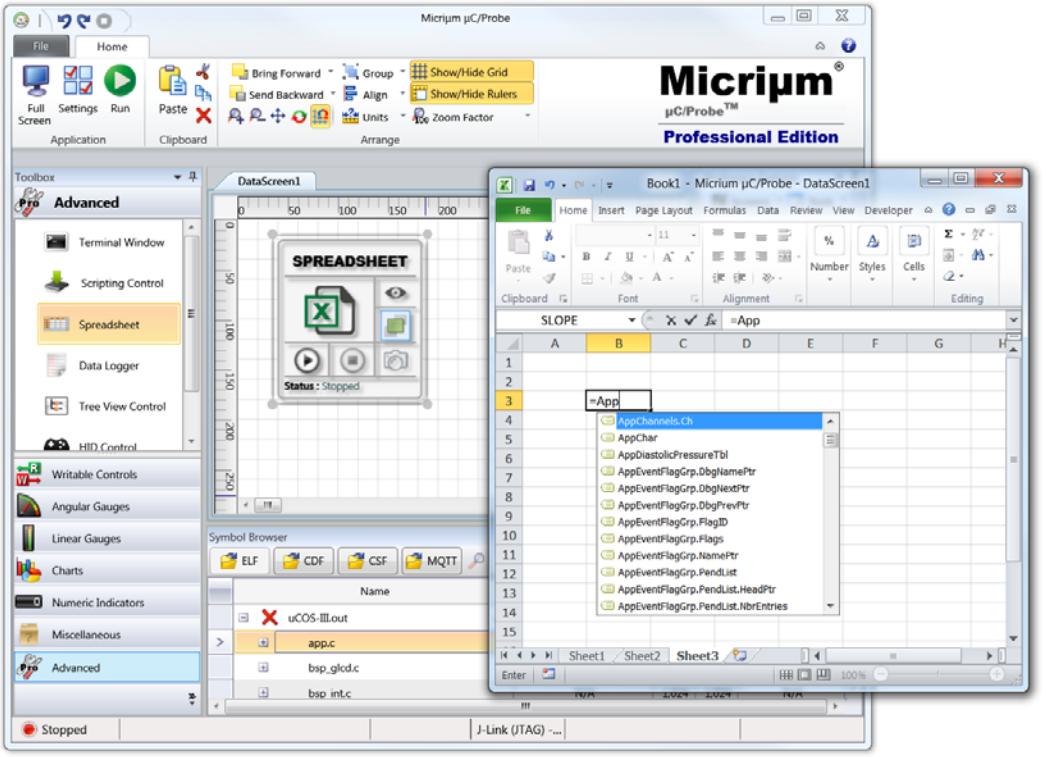


Figure E-6 **Intellisense**

E-4-2 FORMULAS

Figure E-7 shows you other features:

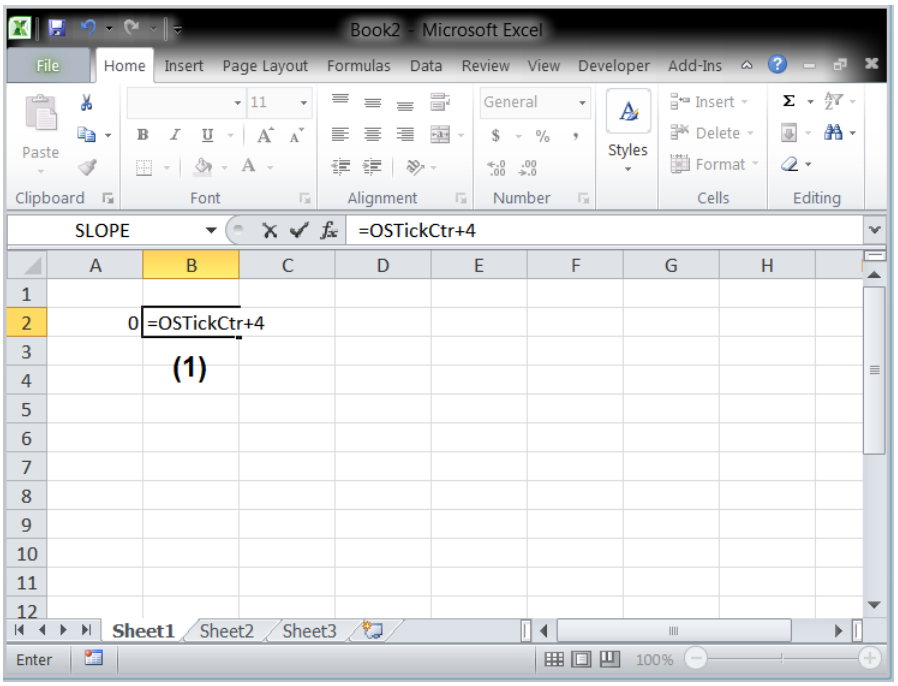


Figure E-7 **Other Features**

FE-7(1) Excel® formulas start by typing in the equal sign in the cell where you want the formula result to appear. As soon as you type in the equal sign, start entering the first few letters of the name of the symbol you want to use in the expression, and for your convenience, Excel® will show you a list of the symbol names you have previously added to the spreadsheet.

When you click on a cell containing a formula in Microsoft® Excel®, the formula always appears in the formula bar located above the column letters.

FE-7(2) You can import and export the spreadsheet along with the embedded target symbols mapping in the form of a regular Microsoft® Excel® file (xls).

FE-7(3) Press the button labeled *Delete Spreadsheet* to remove the spreadsheet from the data screen.

E-5 APPLICATION EXAMPLE

A great way to use µC/Probe and the Spreadsheet Control is to test and calibrate an onboard accelerometer. Imagine your embedded application is reading an accelerometer's X and Y axis and storing them in the global variables `AppAccelX` and `AppAccelY` respectively as shown in Figure E-8:

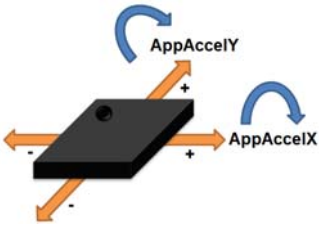


Figure E-8 **Accelerometer**

Very quickly you can add an instance of the spreadsheet control, map two cells to the variables `AppAccelX` and `AppAccelY`, insert a bubble chart from Excel® and there you have a bubble level as shown in Figure E-9:

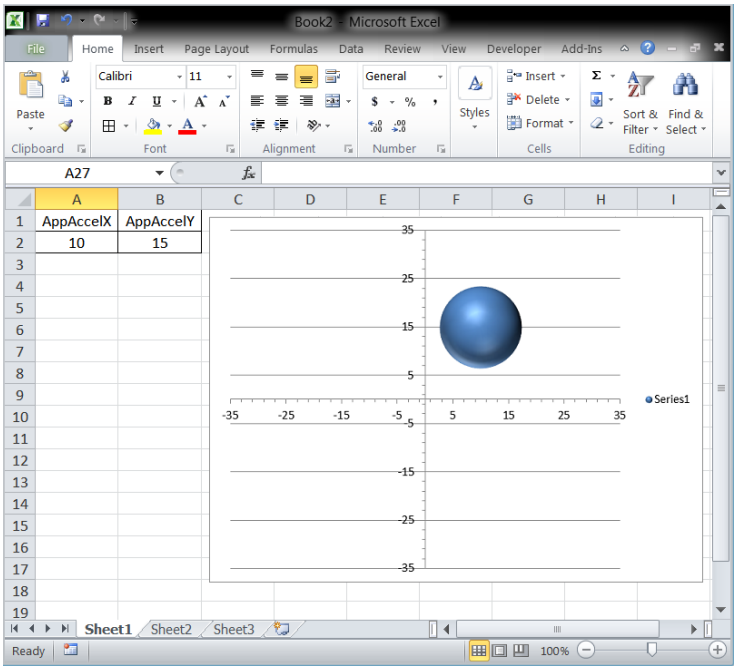


Figure E-9 **Bubble Level Chart**

Appendix

F

Scripting Control

The Professional Edition of µC/Probe comes with a control that helps you programmatically read and write your embedded target symbols on the fly.

You write a script in a C style language to read and write your embedded target symbols and this control will execute your script whenever you want.

Use the µC/Probe Scripting Control to create configuration files, test scripts and to automate the execution of any other common tasks which could alternatively be executed manually by a human operator.

F-1 WRITING A SCRIPT

The µC/Probe Scripting Control supports a programming language that is very similar to C. You write the script as a text file to manipulate variables in two different scopes; the script-side and the embedded target side as described in the following sections.

F-1-1 SCRIPT-SIDE VARIABLES

You declare a variable on the script-side (host PC) by prefixing it with the keyword `var` or with the data type as shown in Code Listing F-1:

```
var max_ch = 3;                     /* Declare a script-side variable by using the keyword  */
                                    /* 'var' and initialize it before using it.             */

string lcd_part_nbr = "";           /* Alternatively, you can declare script-side variables */
                                    /* by specifying its data type (char, string, short,    */
                                    /* int, uint, long, ulong, float, double, bool, etc.).  */

int    gain = 10;

string filter_type;                 /* Furthermore, when the data type is specified, you do */
                                    /* not require to initialize the variable.              */

if (max_ch == 8) {
    ...
} else {
    ...
}
```

Listing F-1 **Script-Side Variables**

F-1-2 EMBEDDED TARGET SIDE VARIABLES

The variables on the embedded target side are referenced by prefixing the name of the variable with the dollar sign `$` as shown in the following Code Listing F-2:

```
$AppTimeout   = 100;                /* Target-side variables are prefixed with the $ sign   */
$AppFrequency =  50;                /* and they must be present in the target's ELF file.   */

for (int i = 0; i < max_ch; i++) {
    $AppChannels.Ch[i].Gain = gain;
}
```

Listing F-2 **Embedded Target Side Variables**

F-1-3 FLOW CONTROL STATEMENTS

The µC/Probe Scripting Control provides two styles of flow control; branching and looping.

BRANCHING

- `if` statement
- `switch/case` statement
- `?` operator

LOOPING

- `while` loop
- `for` loop
- `do/while` loop

It is your responsibility to ensure the loops have a terminating condition and that the terminating condition can be met. Code Listing F-3 shows an example of using some of these flow control statements:

```
if (max_ch == 8) {
    ...
} else {
    ...
}

switch ($AppSwitches) {
    case 1:
         ...
         break;
}

while ($AppStateCalibrating == true) {
    Sleep("Calibrating", 1000);
}
```

Listing F-3 **Flow Control Example**

F-1-4 BUILT-IN INSTRUCTIONS

Besides the standard flow control statements previously mentioned, the µC/Probe Scripting Control provides a set of built-in instructions to help you create a better user interface.

Code Listing F-4 shows you how to use the `Sleep()` and `Pause()` built-in instructions.

```
for (int i = 0; i < max_ch; i++) {
    $AppChannels.Ch[i].Gain = 0;
                                        /* The Sleep instruction allows you to delay the script */
                                        /* execution (milliseconds) with a custom message.      */
    Sleep("Reaching Steady-State...", 200);
}

$AppSelfTestStart = true;

Pause("Self-Test in progress...");    /* The Pause instruction allows you to pause the script */
                                        /* execution until you press a button in µC/Probe.      */
if ($AppSelfTestResult < 0) {
    Abort();                           /* The Abort instruction allows you to stop the script  */
                                        /* execution.                                           */
} else {
    ...
}
```

Listing F-4 **Built-in Instructions**

F-1-5 INCLUDING OTHER SCRIPT FILES

Similar to the directive `#include` in the C language, the µC/Probe Scripting Control allows you to include other script files to help you organize your scripts in modules.

When a script file is included, the code it contains inherits the variable scope of the line on which the include occurs. Any variables available at that line in the calling file will be available within the called file, from that point forward. However, all functions defined in the included file have the global scope.

F-2 ADDING AN INSTANCE OF THE SCRIPTING CONTROL

To add an instance of the µC/Probe Scripting Control go to the µC/Probe toolbox in the *Advanced* Controls category and drag-and-drop the icon labeled *Scripting* into a data screen as shown in Figure F-1:

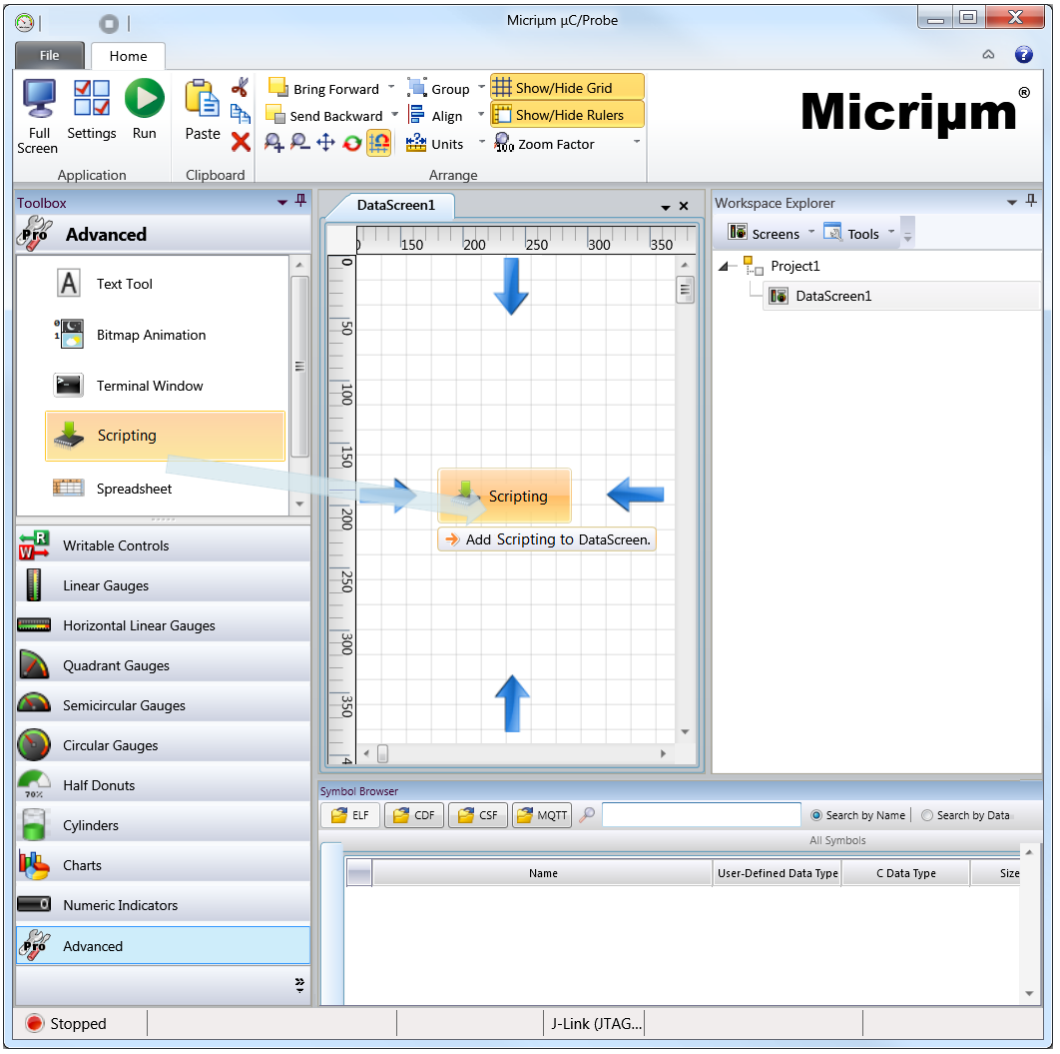


Figure F-1 **Adding an Instance of the Scripting Control**

F-3 CONFIGURING THE SCRIPTING CONTROL

Once you have successfully added an instance of the Scripting Control into a data screen, you have to specify the path to the script file by making click on the properties editor button, which in turn opens a file dialog for you to specify the location of your script file as shown in Figure F-2:

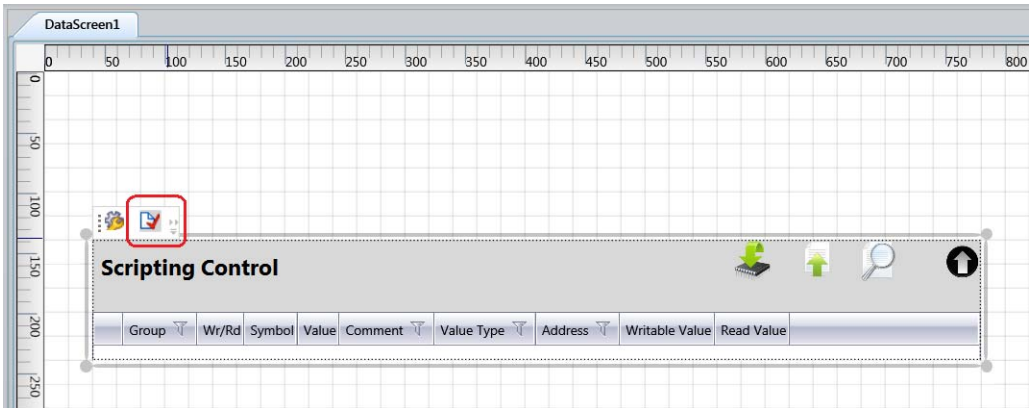


Figure F-2 **Configuring the Scripting Control**

F-4 EXECUTING THE SCRIPT

Suppose you create a configuration script that not only sets the gain of each channel in your data acquisition system, but also activates a system self test and then evaluates the results.

Code Listing F-5 shows the script for such an example.

```
                                    /* Configure the system.                                */
var max_ch = 3;
int gain   = 10;

for (int i = 0; i < max_ch; i++) {
    $AppChannels.Ch[i].Gain = gain;
}

$AppSelfTestStart = true;           /* Gets the system self test started.                   */

                                    /* The Sleep instruction allows you to delay the script */
                                    /* execution (milliseconds) with a custom message.      */
Sleep("Self-Test in progress...", 5000);

                                    /* Evaluate the self test results.                      */
if ($AppSelfTestErr == 0) {
                                    /* The Pause() instruction allows you to suspend        */
                                    /* execution until you press a button in µC/Probe.      */
    Pause("System Configured OK.");
} else {
    Pause("Unable to Configure the System.");
}
```

Listing F-5 **Configuration Script Example**

Once you configure the Scripting Control with the script in Listing F-5, the control will parse the code and will display each of the instructions the control will execute for you as shown in Figure F-3:

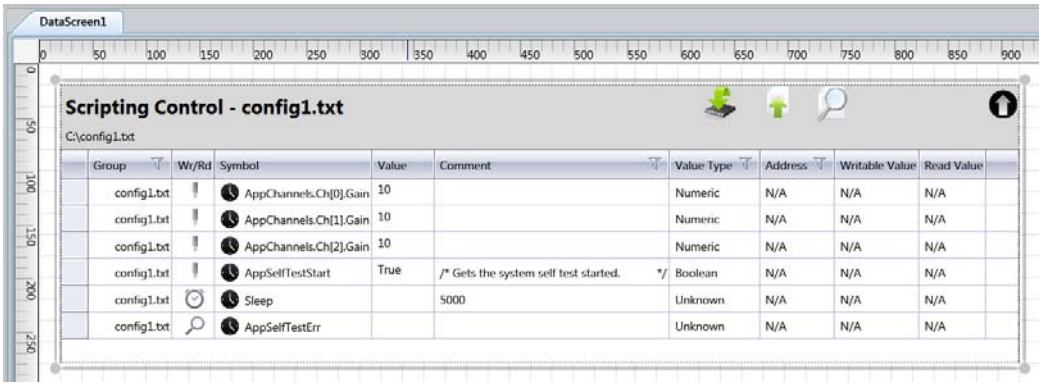


Figure F-3 **Configuration Script Example - Design Time**

When you run µC/Probe, the control will execute your script when you click the button with the chip icon and will display the status of each instruction's execution as shown in Figure F-4:

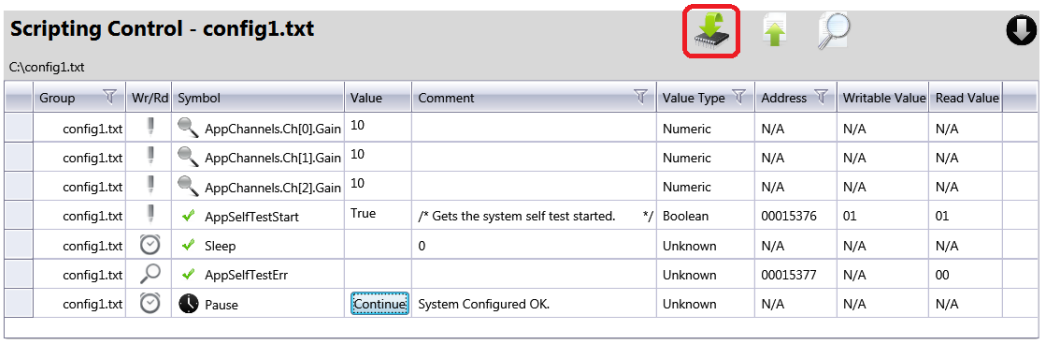


Figure F-4 **Configuration Script Example - Run Time**

Appendix

G

Data Logging Control

µC/Probe provides an option to log the values of any variable(s) in your symbols browser to a CSV file. The Data Log Control is part of the **Advanced** category of µC/Probe's toolbox.

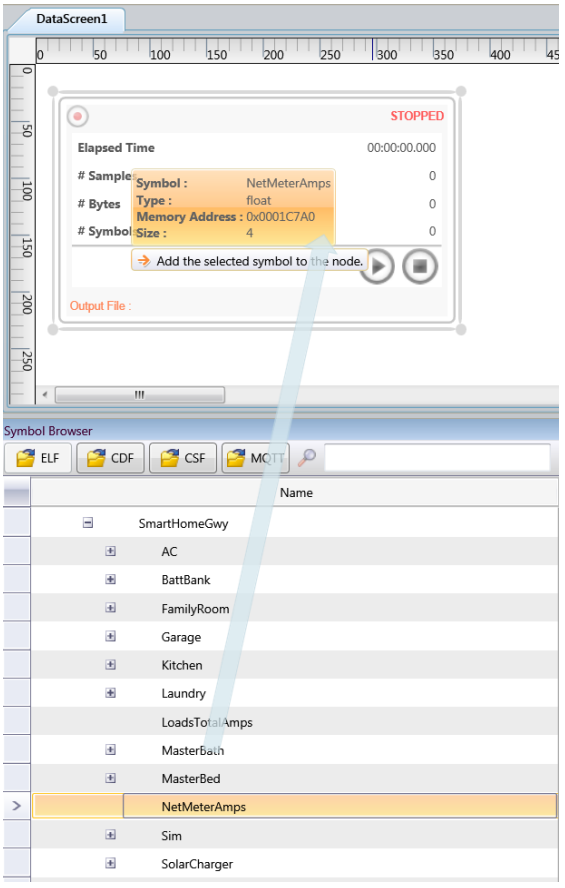


Figure G-1 **Data Log Control**

Figure G-1 illustrates how to log a variable called `NetMeterAmps`. All you have to do is drag and drop an instance of the Data Log Control from the Advanced category in your toolbox and then search the variable you want to data log in the Symbol Browser. You add the variable to the data log by dragging and dropping it onto the Data Log control. You can add as many variables as you want and they will be stored in a CSV file, one column per variable.

You can configure the Data Log Control from the Properties Editor as shown in Figure G-2:

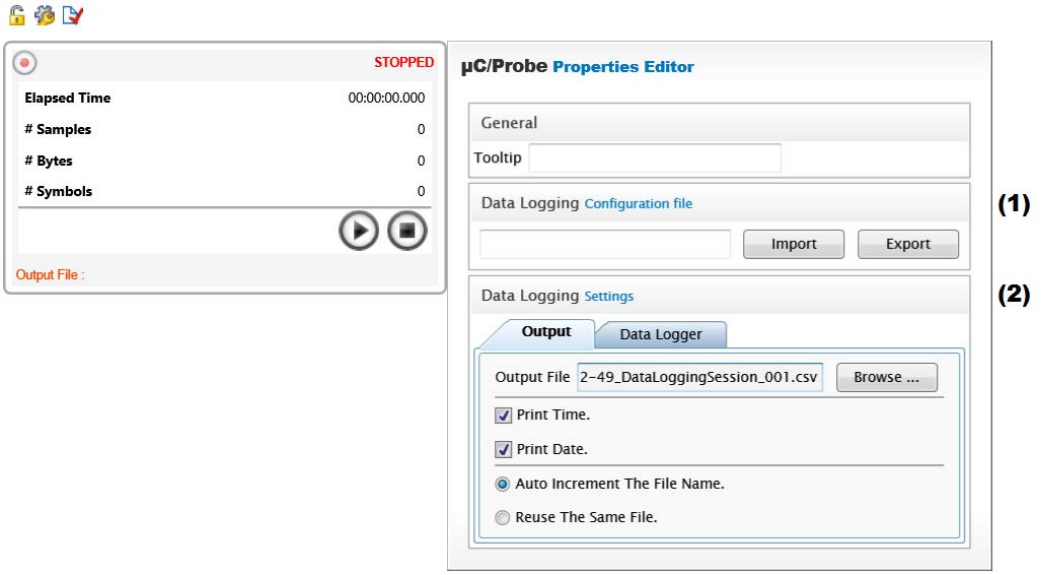
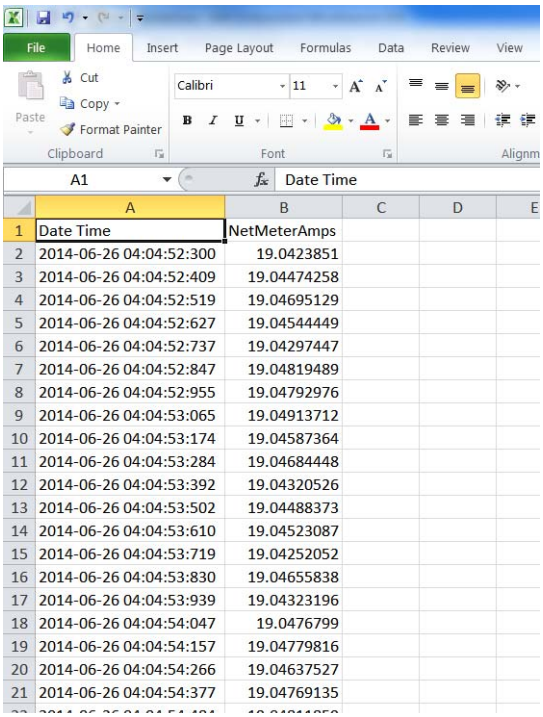


Figure G-2 **Data Log Control Properties Editor**

FG-2(1) You can either **Import** a Data Log Configuration file from a past logging session or you can **Export** your current configuration to a file.

FG-2(2) The Data Logging Settings panel consists of two tabs; The **Output** tab and the **Data Logger** tab. From the **Output** tab you can configure the output file. You can specify the path and name of the output file along with other settings such as the timestamps and whether or not reusing the same file or create a new one by using an auto-increment number as part of the file name. By default the output files are stored in the µC/Probe installation directory in the subfolder `DataLogging`.

During run-time, the value of `NetMeterAmps` will be logged in the CSV file as shown in Figure G-3:



	A	B	C	D	E
1	Date Time	NetMeterAmps			
2	2014-06-26 04:04:52:300	19.0423851			
3	2014-06-26 04:04:52:409	19.04474258			
4	2014-06-26 04:04:52:519	19.04695129			
5	2014-06-26 04:04:52:627	19.04544449			
6	2014-06-26 04:04:52:737	19.04297447			
7	2014-06-26 04:04:52:847	19.04819489			
8	2014-06-26 04:04:52:955	19.04792976			
9	2014-06-26 04:04:53:065	19.04913712			
10	2014-06-26 04:04:53:174	19.04587364			
11	2014-06-26 04:04:53:284	19.04684448			
12	2014-06-26 04:04:53:392	19.04320526			
13	2014-06-26 04:04:53:502	19.04488373			
14	2014-06-26 04:04:53:610	19.04523087			
15	2014-06-26 04:04:53:719	19.04252052			
16	2014-06-26 04:04:53:830	19.04655838			
17	2014-06-26 04:04:53:939	19.04323196			
18	2014-06-26 04:04:54:047	19.0476799			
19	2014-06-26 04:04:54:157	19.04779816			
20	2014-06-26 04:04:54:266	19.04637527			
21	2014-06-26 04:04:54:377	19.04769135			

Figure G-3 **Data Log Output File (CSV format)**

You can suspend or stop the data logging process by using the buttons in the status screen shown in Figure G-4

Figure G-4 **Data Log Control During Run-Time**

Additionally, you can configure **Start** and **Stop** conditions to trigger the data logging process. For example, Figure G-5 illustrates how to configure the Data Logger to Start logging when the value of `NetMeterAmps` is between 20 and 50 amps.

The sampling period can also be configured from the same screen in terms of hours, minutes, seconds and milliseconds

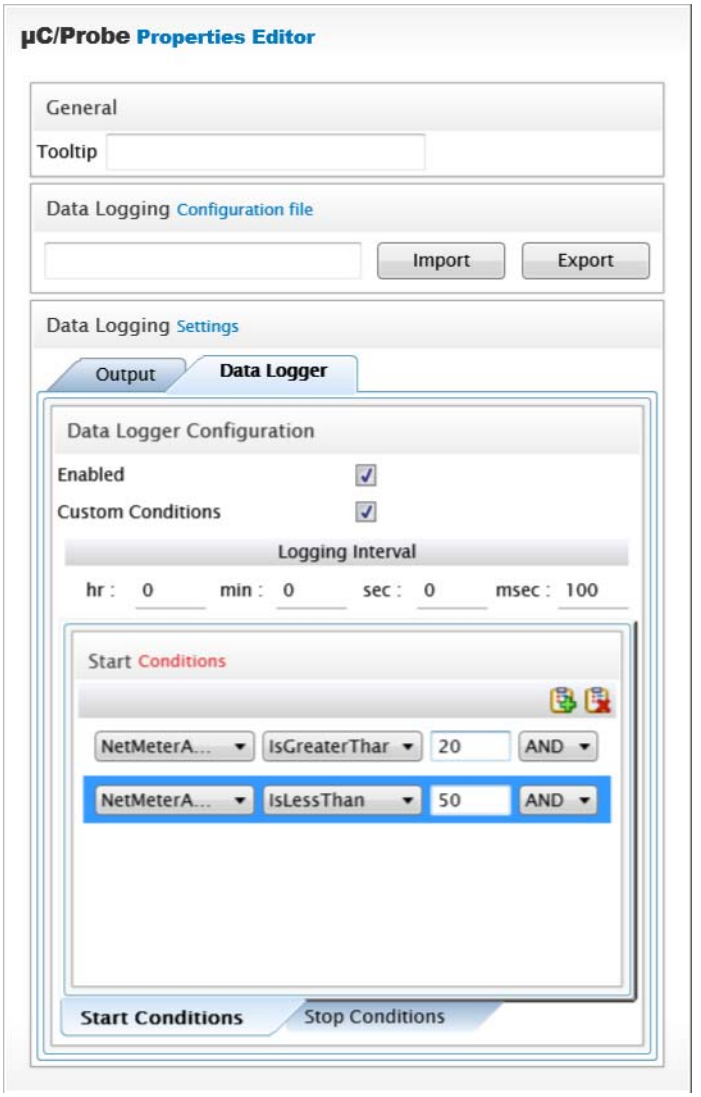


Figure G-5 **Data Logger Start and Stop Conditions**

Appendix

H

Human Machine Interface (HID) Control

µC/Probe allows you to use a USB HID control such as a gamepad, joystick or steering wheel to control your embedded target. The HID Control is part of the **Advanced** category of µC/Probe's toolbox.

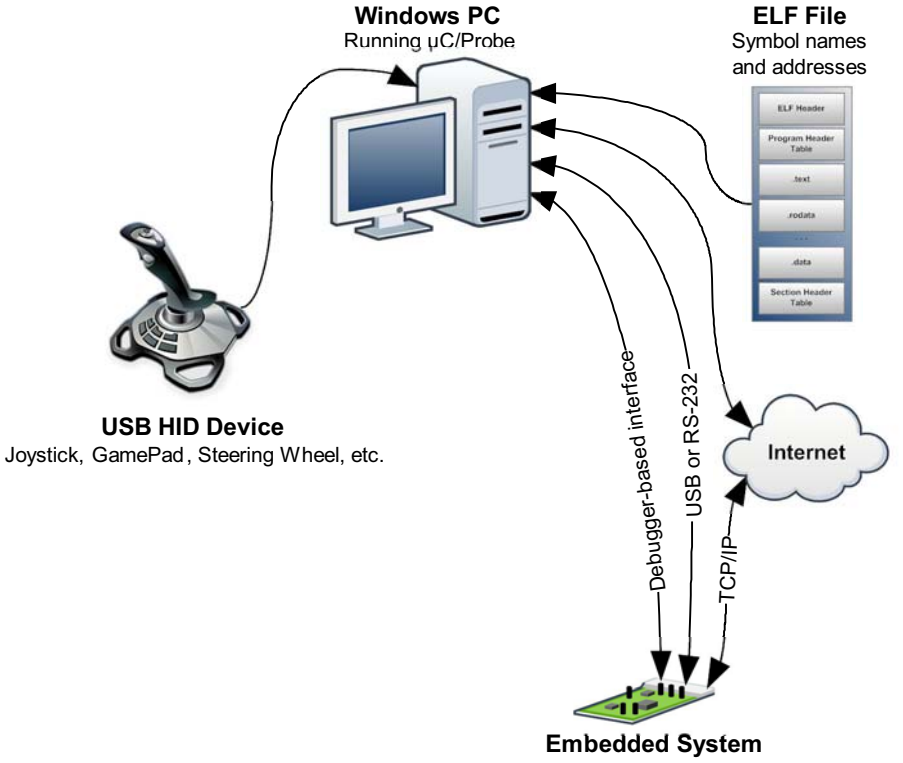


Figure H-1 **HID Control Block Diagram**

You simply declare in your embedded target the variables that will store the output from your HID device.

For example, if your HID device is a Joystick, then you would declare variables such as:

- AppJoystickButton1
- AppJoystickButton2
- AppJoystickButton3
- AppJoystickButtonX
- AppJoystickButtonY

Then you drag and drop an instance of the HID control onto your Data Screen and start associating the embedded variables to the control as shown in Figure H-2:

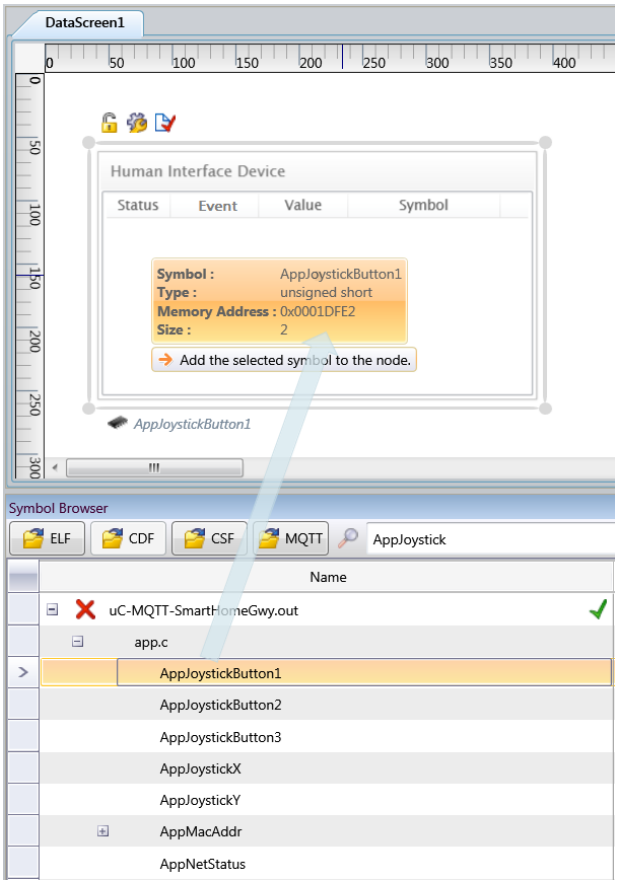


Figure H-2 **Symbol Browser and HID Control**

The next step is to configure each possible event from the HID device and map it to a unique embedded target symbol. You first open the HID Control's Properties Editor and select your HID device as shown in Figure H-3:

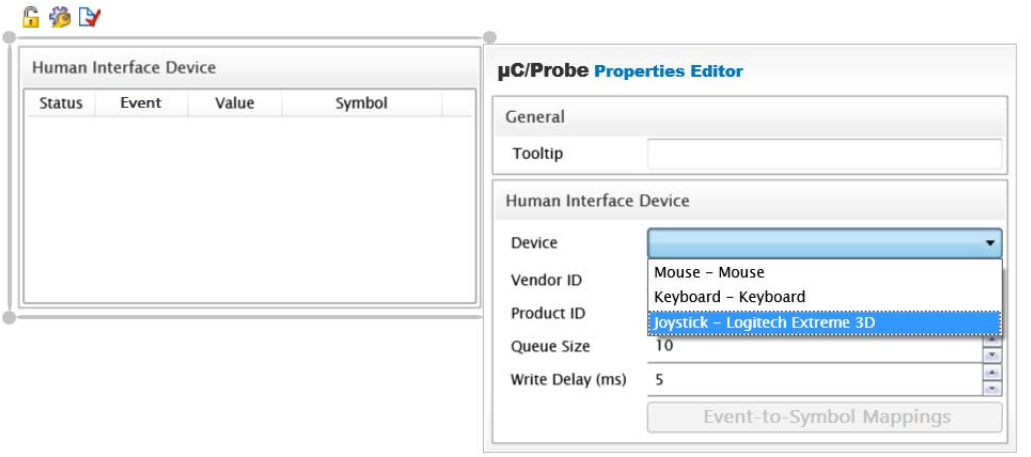


Figure H-3 **HID Control Properties Editor**

Once you select your HID Device, the button labeled as **Event-to-Symbol Mappings** will be enabled and you can proceed to map each event to an embedded target symbol. As you move the joystick, µC/Probe will catch all the possible events and create a new option to configure a mapping as shown in Figure H-4:

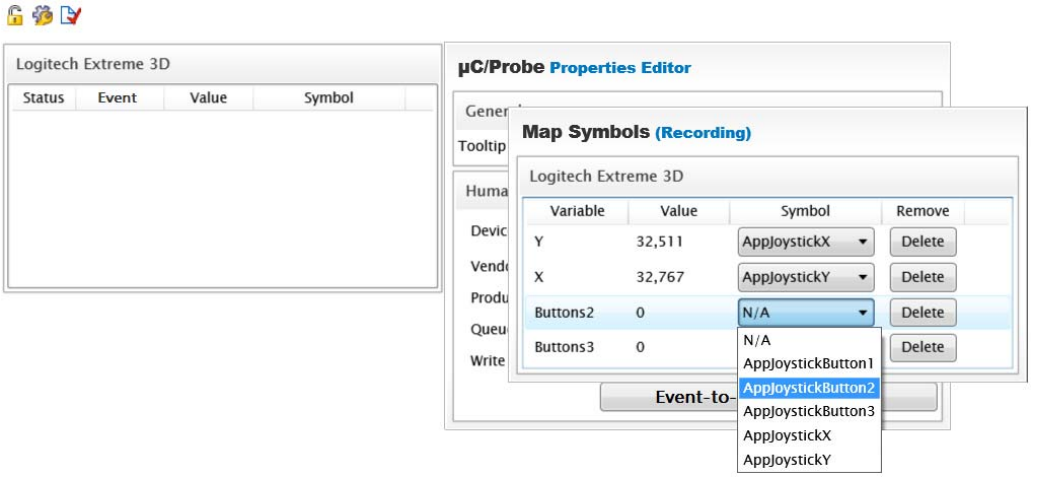


Figure H-4 **HID Control Event-to-Symbol Mappings**

During run-time µC/Probe displays the status of your HID Device with the table shown in Figure H-5. The LED remains green during idle time and turns red once the event is caught. At each event the value is written to the associated target symbol.

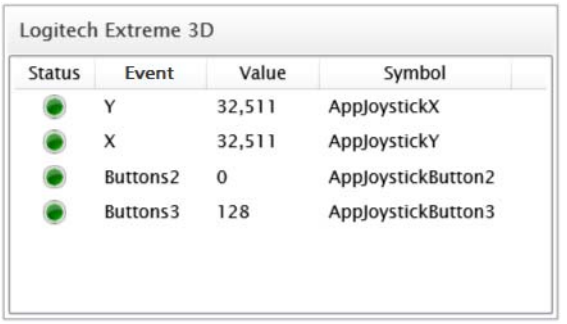


Figure H-5 **HID Control Status**

Appendix

Oscilloscope Control

µC/Probe allows you to analyze data in real-time by showing the value of multiple memory addresses in a screen akin to an oscilloscope. Similar to other controls in µC/Probe, you simply select the variables you want to plot from the **Symbol Browser**. The oscilloscope control can display up to 8 channels in either a single vertical scale or multiple scales.

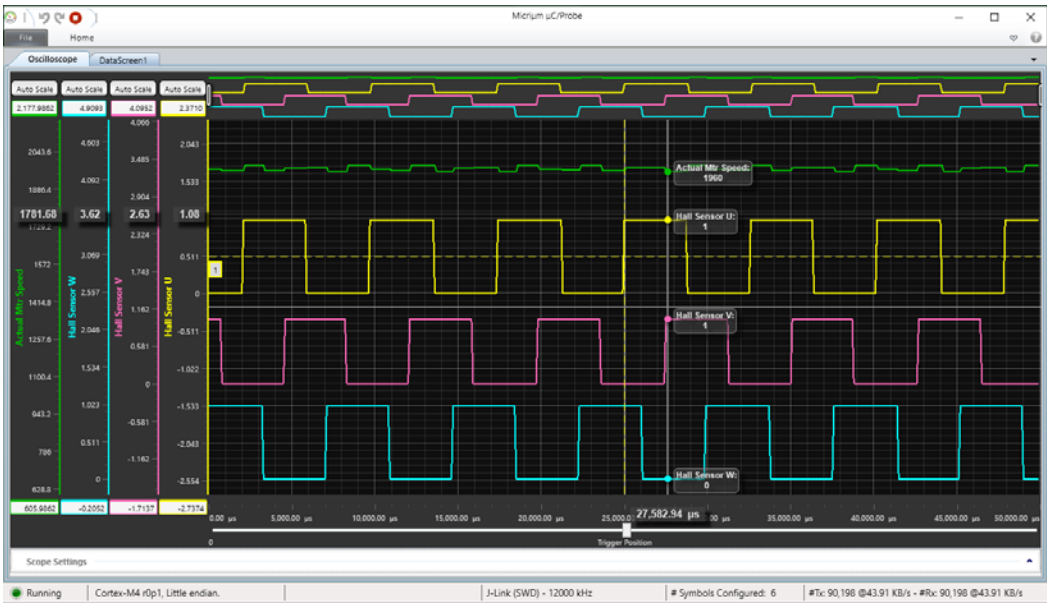


Figure I-1 **Oscilloscope Control**

This oscilloscope control requires target resident code in the form of three simple C files that you would have to include in your existent project. They implement the algorithm for triggering and samples acquisition into a size configurable buffer.

This document describes the oscilloscope control from the Windows Application point of view. For more information on how to include and configure the embedded target resident code to support it, refer to the document **µC/Probe Target Manual**.

I-1 PREREQUISITES

The first thing you need to do to support the oscilloscope control is to include the embedded target resident code as described in a separate document titled **µC/Probe Target Manual**.

Once you have built your embedded project you can open the ELF file from the **Symbol Browser** as described in Chapter 3, “ELF File” on page 17.

At the same time, you can also include a Chip Definition File (CDF) in case you want to plot a chip I/O register on the oscilloscope. For more information on CDFs refer to Chapter 3, “CDF File” on page 19.

I-2 CREATING AN INSTANCE OF THE OSCILLOSCOPE CONTROL

To create an instance of the oscilloscope control, go to the **Workspace Explorer** panel, select your **Project**, right-click to display the context menu and select the option **Add Oscilloscope** as shown in Figure I-2:

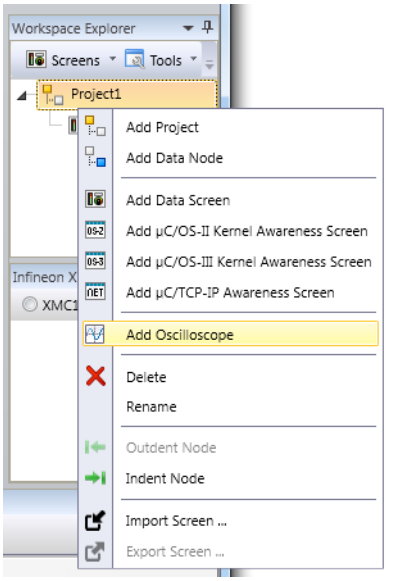


Figure I-2 **Creating an Instance of the Oscilloscope Control**

I-3 CONFIGURING THE SYMBOLS TO PLOT

The next step is to configure the symbols or variables that you want to display in the oscilloscope. You simply select a symbol from the **Symbol Browser** and either double-click on it or, drag-and-drop it over the oscilloscope control.

You can configure the oscilloscope with as many variables as you want, but you can only display a maximum of 8 channels at once.

The maximum number of channels is configured at compile time and from the embedded target resident code as described in the **µC/Probe Target Manual**.

As you select each symbol from the **Symbol Browser** the list of channels gets populated in the same order as shown in Figure I-3:

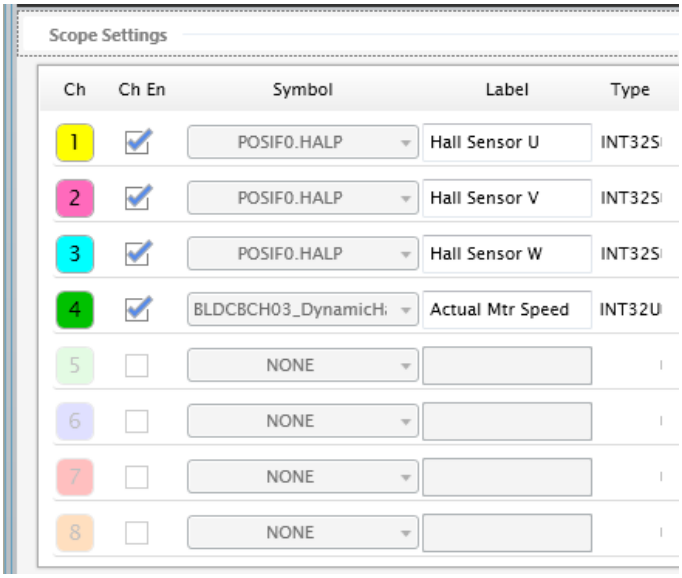


Figure I-3 **Oscilloscope Control: Symbols Configuration**

I-4 CHANNEL CONFIGURATION

The list of input channels shown in Figure I-4 is part of the Oscilloscope Control's Settings window located below the plot area. You can show and hide the entire section with either a double-click of the window's title bar or by a click of the icon with an arrow pointing downwards on the right top corner of the window.

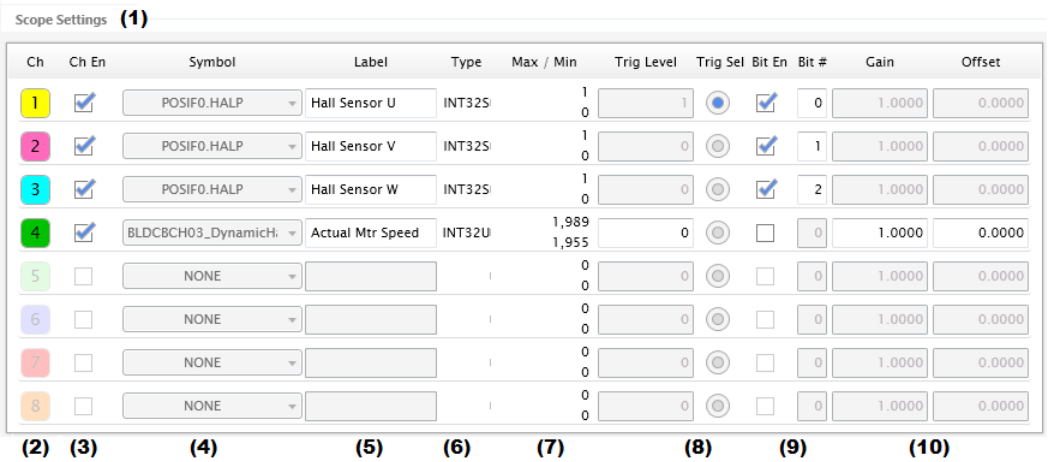


Figure I-4 **Oscilloscope Control: Channel Configuration**

FI-4(1) Double-click the Scope Settings window's title bar to show and hide the entire section.

FI-4(2) All input channels employ the same color scheme whether they are enabled or disabled. The colors are used not only for the waveform traces in the plot area but also to identify other properties such as the vertical axes and trigger level.

FI-4(3) Each input channel may be individually enabled or disabled during both; design and run time. Disabling a channel turns off its data capture on the embedded target side. It is a good idea to disable channels you are not using because it will free up the target's CPU.

FI-4(4) The symbol name associated to each input channel can be changed during design-time.

FI-4(5) You can specify an alias for each input channel in those cases where the symbol's name is either ambiguous or simply not very friendly. You can also use this label to show the engineering units.

FI-4(6) The symbol's data type as declared in the embedded target application.

FI-4(7) The maximum and minimum running statistics for each input channel. Very useful to determine the signal's dynamic range.

FI-4(8) When the scope is configured in **Triggering Mode**, here you can specify not only which input channel is the trigger source but also the trigger level.

FI-4(9) You can extract and plot bit information from a symbol by enabling the **Bit Mode** and specifying the bit number to extract and plot. When Bit Mode is enabled the trace will be a square wave signal between 0 and 1. This is very useful when you want to plot a chip's I/O bit-addressable register or if the data you want to plot and analyze is stored in a bit-addressable variable.

FI-4(10) You can specify a linear scaling function in the form `y = gain * x + offset` in case you want to convert a raw value to engineering units.

I-5 OSCILLOSCOPE CONTROL MODES

During run-time, you can select one of the four oscilloscope modes shown in Figure I-5:

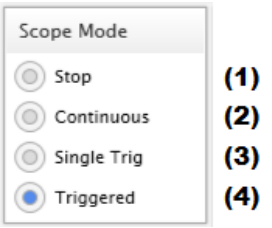


Figure I-5 **Oscilloscope Control: Modes**

FI-5(1) **Stop Mode:** data capture in all channels gets disabled on the embedded target side.

FI-5(2) **Continuous Mode:** data capture in all channels is done continuously without any triggering mechanism. Continuous mode makes it easier to see waveforms before you have established the trigger condition or when configuring a trigger condition is not possible.

FI-5(3) **Single Trigger Mode:** data is captured in all channels from the same trigger point, which is defined as the point in time when the signal met the trigger condition (level and type of transition). The data is captured only once and then the oscilloscope stops capturing.

FI-5(4) **Triggered Mode:** similar to the previous Single Trigger mode except that after a trigger condition has been met and the corresponding data has been captured, the triggering mechanism is rearmed again after some hold off period of time to capture more data indefinitely, updating the plot area in real-time until stopped.

I-6 TRIGGERING MODES

The trigger condition is configured by specifying the trigger source and level as previously described in FI-4(8) and by selecting the type of signal transition as shown in Figure I-6.

You can select the transition to be a **Positive Slope** (rising edge) or **Negative Slope** (falling edge).

Optionally, if the oscilloscope mode is set to **Triggered Mode** then you can specify a hold off period of time in terms of number of samples.

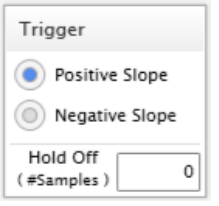


Figure I-6 **Oscilloscope Control: Trigger Configuration**

I-7 TIMEBASE CONFIGURATION

All input channels use the same timebase parameters so they remain time synchronized.

You specify during compile-time the frequency of the **Sampling Clock** in hertz. Then, during run-time and from the host PC you can configure a **Clock Divider** to further decimate the data. The Sampling Clock divided by the Clock Divider becomes the effective data acquisition **Sampling Rate**.

Other parameters defined at compile time on the embedded target code modify the number of samples per channel and the total time of each data capture frame. For more information on how to configure the sampling clock and the maximum number of samples per channel refer to the **µC/Probe Target Manual**:

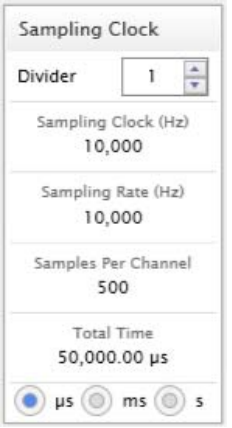


Figure I-7 **Oscilloscope Control: Timebase Configuration**

I-8 OSCILLOSCOPE CONTROL STATUS

Once you get µC/Probe running the first thing to look at is the status panel to make sure that things are working properly. It has a set of two LEDs and one progress bar to indicate the moment when the trigger condition has been met, the progress as the data buffers in the target are being filled with samples and the moment when the data is ready to plot as shown in Figure I-8:

Figure I-8 **Oscilloscope Control: Status**

I-9 OSCILLOSCOPE CONTROL TOOLS

The **Tools** panel provides a set of buttons to execute some of the most common operations on the oscilloscope control:

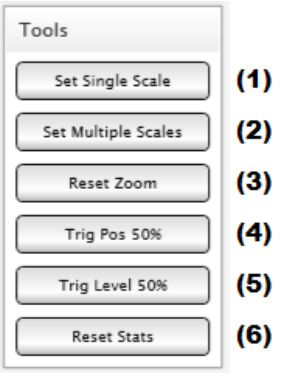


Figure I-9 **Oscilloscope Control: Tools**

FI-9(1) This configures the vertical axes to have the same range. The range is calculated by taking the minimum and maximum values among all input channels.

FI-9(2) When the values in the plot area vary widely from channel to channel, or when you have mixed types of data (e.g. motor speed and temperature), you can plot the waveforms on different vertical axes (multiple scales).

FI-9(3) There is a zoom and pan tool above the plot area. This button resets the tool to display all the data.

FI-9(4) There is a slider control right below the plot area that allows you to move the trigger point along the time axis. This button positions the trigger point right in the middle, giving you a waveform that occurred before and after the trigger condition (i.e. 50% pre-trigger).

FI-9(5) This button configures the trigger level on the selected trigger source to be a 50% of the observed range.

FI-9(6) This button resets the minimum and maximum calculations for each channel.

I-10 OSCILLOSCOPE CONTROL PLOT AREA

The Oscilloscope Control's Plot Area has a dark background similar to the ones found in any other oscilloscope. It has however, other features only found in some of the most sophisticated digital oscilloscopes out there:

Figure I-10 **Oscilloscope Control: Plot Area**

FI-10(1) The zoom and pan tool bar allows you to interact with the plot area and inspect your data. You can zoom in and out by dragging left and right the vertical bars on each side of the tool bar. Then you can drag left and right any area inside the tool bar to move quickly around the zoomed in view.

FI-10(2) The dashed line indicates the trigger level. It has the same color of the selected trigger source channel. You can modify the trigger level from the channels list below the plot area or by dragging up and down the square on the left side of this dashed line.

FI-10(3) As you move the mouse around the plot area, legends will pop up displaying the input channel's name and value.

FI-10(4) There is a slider control right below the plot area that allows you to move the trigger point along the time axis. This allows you to specify the size of the pre and post trigger buffer.

FI-10(5) The text boxes at the top and bottom of each vertical axis allow you to enter their maximum and minimum range.

FI-10(6) Each vertical scale has the same color of its corresponding input channel. You can modify the scale's range by either entering the values in the text boxes at the top and bottom of the axes or, you can use your mouse to execute the following operations:

- **Single Channel Zoom**: to zoom in and out each individual waveform, press and hold the **Ctrl** key, click the mouse's **left button** and drag the corresponding vertical axis up and down to the desired zoom level.
- **Single Channel Offset**: to move a single waveform up and down, simply click the mouse's **left button** and drag the corresponding vertical axis up and down to the desired offset.
- **All Channels Zoom**: to zoom in and out all waveforms at the same time, press and hold the **Ctrl+Shift** keys, and spin the **mouse wheel** up and down over the waveforms to the desired zoom level.
- **All Channels Offset**: to move all the waveforms up and down at the same time, press and hold the **Shift** key, and spin the **mouse wheel** up and down over the waveforms to the desired offset level.

FI-10(7) Each vertical axis provides a button to calculate the scale's range automatically. It uses the running statistics to configure the maximum and minimum range.

I-11 OSCILLOSCOPE SPECIFICATIONS

I-11-1 BANDWIDTH

The oscilloscope's bandwidth depends on the embedded target's CPU. For example, an ARM Cortex-M4 at 80 MHz, with compiler optimization set to -O3 is capable of capturing samples from one channel at a sampling rate of as high as 1.33 MHz without any triggering mechanism. In order to ensure that this bandwidth is adequate for your application, it is necessary to ensure that the maximum sampling rate is higher than the waveform's operating frequency. As a rule of thumb, five times higher, in which case, at 1.33 MHz sampling rate the scope can measure a 266 kHz waveform.

For more information on performance results in other embedded devices and under other conditions see "Performance Results" on page 169.

I-11-2 NUMBER OF CAPTURE CHANNELS

The number of channels is configurable at compile time. The maximum number is 8. Notice that the oscilloscope's bandwidth is inversely proportional to the number of channels.

Refer to the document µC/Probe Target Manual for more information on how to configure the maximum number of channels.

I-11-3 INPUT RANGE

The input range for each channel depends on the associated symbol's data type. If support for 32-bit has been enabled at compile-time, then the maximum ranges are:

Data Type	Minimum	Maximum
unsigned int	0	4294967295
signed int	-2,147,483,647	+2,147,483,647
float	1.17549×10^{-38}	3.40282×10^{38}

I-11-4 CAPTURE BUFFER SIZE

The size of the capture buffer is also configurable at compile-time and it can be as high as the resources in your embedded target platform permit.

I-11-5 TIME BASE RANGE

The time base range depends on the sampling rate and the size of the capture buffer. For example, a sampling rate of 10kHz and a buffer size of 500 samples will give you a time base of 20 ms. If you want to increase the time base then you need to either increase the buffer size or decrease the sampling rate at compile-time. Alternatively, you can configure a clock divider at run-time from µC/Probe to decrease the sampling rate.

I-11-6 COMMUNICATION INTERFACES

The oscilloscope control works through any of the supported communication interfaces by µC/Probe (see Chapter 4, “Embedded Target Settings” on page 33). However, using one of the faster interfaces such as J-Link, TCP/IP and USB is recommended.

I-11-7 OSCILLOSCOPE'S PERFORMANCE

The oscilloscope's performance is presented in terms of dead-time or blind-time, which is the time right after each data acquisition during which the data is transferred to µC/Probe and displayed on the screen. The dead-time depends a lot not only on the Embedded Target's CPU, but also on the Windows PC performance and the Capture Buffer Size. During the oscilloscope's dead-time, any signal activity that may be occurring will be missed as illustrated in the following images that show the acquisition-time and dead-time for an ARM-Cortex-M0 and an ARM Cortex-M4 respectively:

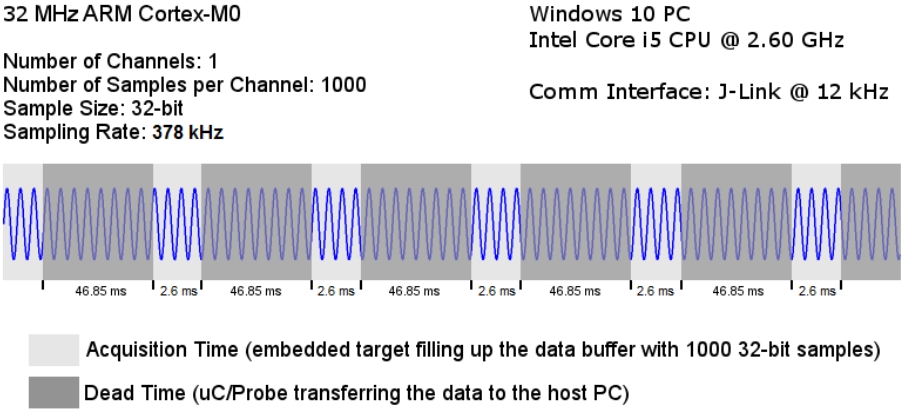


Figure I-11 **Scope's Performance in an ARM Cortex-M0**

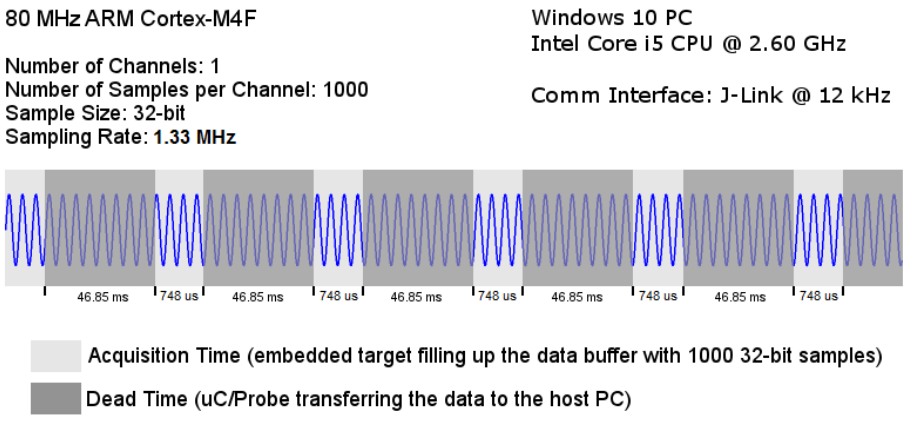


Figure I-12 **Scope's Performance in an ARM Cortex-M4**

From the previous images, it can also be figured that the time for the embedded target to acquire one sample is the inverse of the maximum sampling rate. For example, in the case of the Cortex-M4 a maximum sampling rate of 1.33 MHz means that the function `ProbeScope_Sampling()` takes around 748.5 ns.

The table below summarizes all the results:

ARM Device and Clock Frequency	Scope Mode	Sample Size	Code Optimization Level	Maximum Sampling Rate	ProbeScope_Sampling() execution time
Cortex-M0 @ 32 MHz	Continuous	32-bit	None	157 kHz	6.34 µs
Cortex-M0 @ 32 MHz	Triggered	32-bit	None	47 kHz	21.25 µs
Cortex-M0 @ 32 MHz	Continuous	32-bit	-O3	378 kHz	2.64 µs
Cortex-M0 @ 32 MHz	Triggered	32-bit	-O3	76 kHz	13.12 µs
Cortex-M4F @ 80 MHz	Continuous	32-bit	None	534 kHz	1.87 µs
Cortex-M4F @ 80 MHz	Triggered	32-bit	None	142 kHz	7.00 µs
Cortex-M4F @ 80 MHz	Continuous	32-bit	-O3	1.33 MHz	748.5 ns
Cortex-M4F @ 80 MHz	Triggered	32-bit	-O3	254 kHz	3.93 µs

Table I-1 **Performance Results**

Appendix

J

Memory Dump Control

µC/Probe allows you to analyze data in real-time by showing the contents of an area of your embedded target's memory. Similar to your favorite IDE's memory window, the memory dump control in µC/Probe displays the contents of memory at address level as shown Figure J-1:

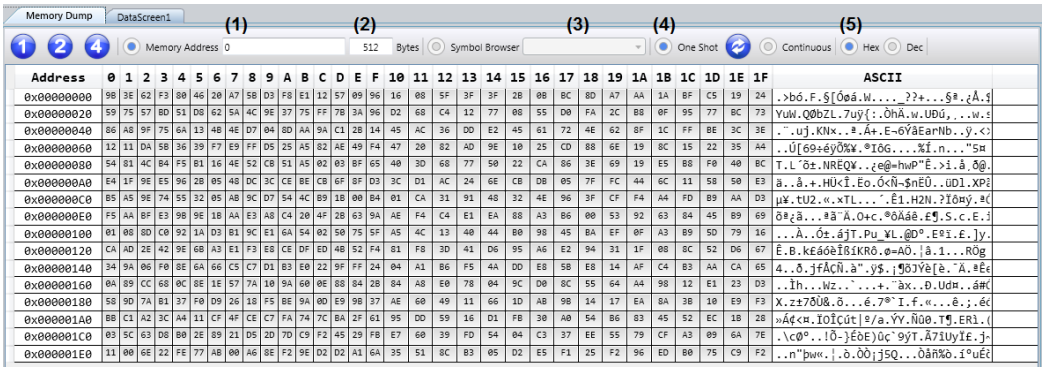


Figure J-1 **Memory Dump Control**

FJ-1(1) You can type in the base address or the beginning of the memory area you want to watch.

FJ-1(2) You can type in the number of bytes to watch starting from the base address.

FJ-1(3) For your convenience, you can also select a symbol and µC/Probe will use its memory address as the base address to watch.

FJ-1(4) You can configure µC/Probe to update the contents of the memory area in one single shot or continuously.

FJ-1(5) You can configure the memory window to display the values in either Hexadecimal or Decimal format.

J-1 CREATING AN INSTANCE OF THE MEMORY DUMP CONTROL

To create an instance of the memory dump control, go to the **Workspace Explorer** panel, select your **Project**, right-click to display the context menu and select the option **Add Memory Dump Screen** as shown in Figure J-2:

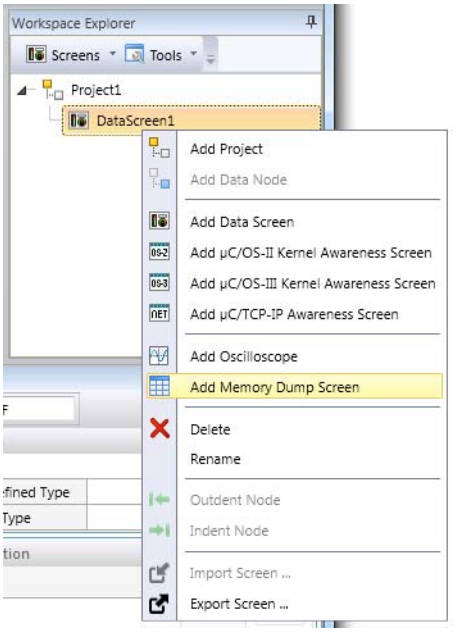


Figure J-2 **Creating an Instance of the Memory Dump Control**

J-2 CONFIGURING THE MEMORY RANGE TO WATCH

The next step is to configure the symbols or variables that you want to display in the Memory Window. You simply select a symbol from the **Symbol Browser** and either double-click on it or, drag-and-drop it over the Memory Window. That will configure the base address to watch. Alternatively, you can simply type in the base address and the number of bytes to watch.

Appendix

K

Infineon Edition of µC/Probe: µC/Probe XMC™

The Infineon Edition of µC/Probe, known as µC/Probe XMC™ is essentially the same as the µC/Probe Professional Edition, except that its usage is limited to Infineon XMC™ microcontrollers only and the fact that it does not require any license activation.

To operate µC/Probe XMC™ the user has to select the correct XMC™ target family, XMC1000 or XMC4000 in the Infineon XMC™ Family pane underneath the Workspace Explorer pane as shown in Figure K-1.

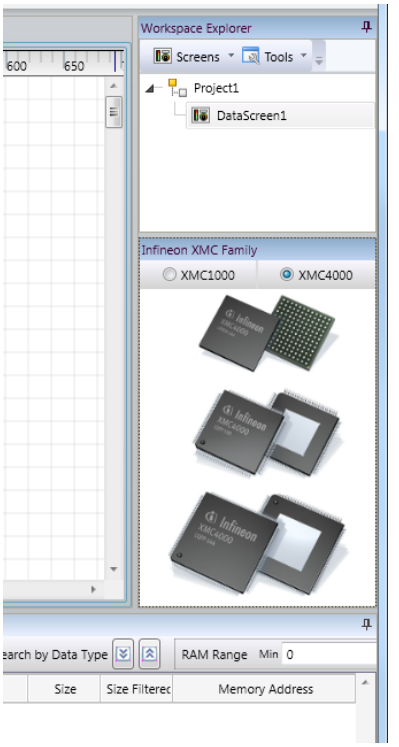


Figure K-1 **Infineon XMC Family Selection**

The XMC1000 family provides the option of a single pin debug interface (SPD). This protocol is transparently supported by a few J-Link debuggers, like the OBD debugger integrated to the kits and the XMC Link. In case a used XMC1000 target MCU is set to SPD mode µC/Probe XMC™ can only be used in parallel with a debugger: Before running µC/Probe XMC™ a debugger (e.g. the one integrated in DAVE v4, IAR or Keil MDK) has to be connected to the target MCU and put in run state, then µC/Probe XMC™ can be connected and operated as usual.

Appendix L

Licensing

The Educational Edition of µC/Probe is freely available for students, teachers, and academic organizations. It has a limited number of features as shown in Appendix L, "µC/Probe Editions Comparison Table" on page 176.

The Professional Edition of µC/Probe, comes with all the features and requires the purchase of a license.

The Infineon Edition of µC/Probe, known as µC/Probe XMC™ is essentially the same as the µC/Probe Professional Edition, except that its usage is limited to Infineon XMC™ microcontrollers and the fact that it does not require any license activation.

You can purchase the µC/Probe Professional Edition for your personally owned computer or those owned by your employer by a subscription based license as follows:

- Yearly Subscription License

  Our e-commerce partner FastSpring will bill you automatically each year and you may cancel the subscription at any point, no questions asked.

Note: Keep in mind that all µC/Probe sales are final and non-refundable. The Educational Edition of µC/Probe is available for free to enable you to "try before you buy".

For example, if you purchase a yearly subscription and wish to cancel the subscription after a couple of months, we will cancel your subscription and any future billing. However, we will not be able to make any partial refunds and you would still be charged for the first year's subscription.

L-1 ORDERING

If you decide to purchase a license of µC/Probe Professional you can go to our online store at: http://micrium.com/tools/ucprobe/buy

We also accept company Purchase Orders by contacting us directly at sales@micrium.com or via the phone at 954-217-2036.

The table below summarizes the main differences among the different editions of µC/Probe:

Feature	**Educational Edition**	**Professional Edition**
Design Mode	x	x
Run-Time Mode (timeout in minutes)	x (1)	x (no-timeout)
Maximum Number of Data Screens	1	unlimited
Maximum Number of Gauge Styles	5	unlimited
Maximum Number of Numeric Indicator Styles	3	unlimited
Thermometer	x	x
Cylinder Indicators	x	x
Button Controls	x	x
Slider Controls	x	x
Bit Control	x	x
Marker Chart	x	x
Line Chart	x	x
Area Chart	x	x
Scatter X-Y Chart		x
µC/OS-II and µC/OS-III Kernel and µC/TCP-IP Awareness	x	x
Terminal Window Control		x
Scripting Control		x
Microsoft® Excel® Bridge		x
µC/Trace Trigger Control		x
Import/Export Data Screens		x
Numeric Up/Down Control		x
Textbox Control		x
Data Log Control		x
Tree View Control		x
Radio Buttons Control		x
HID Control		x
Oscilloscope Control		x
Memory Dump Control		x

Table L-1 **µC/Probe Editions Comparison Table**

L-2 ACTIVATING

This section only applies to the Professional Edition of µC/Probe, which requires the purchase of a license. If you are running the Infineon Edition of µC/Probe known as µC/Probe XMC™ then you do not need to purchase any license.

Once you place an order on the Professional Edition of µC/Probe in our online store you will receive an e-mail message with your license key.

To activate your copy of µC/Probe you need Internet access to validate your license key. If your internal network uses a proxy server to connect to the Internet, then you will need to configure µC/Probe to use your proxy server. To do so, click *File -> Settings -> Host Internet Access* and provide your Proxy server settings in a window similar to the one shown below:

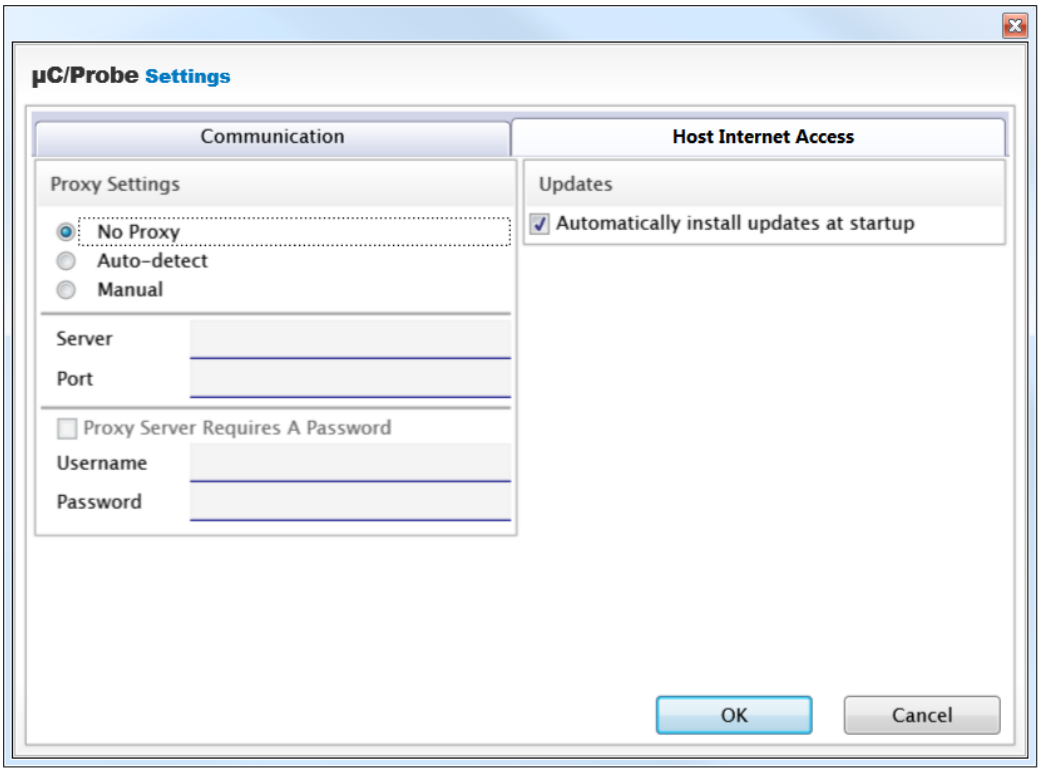


Figure L-1 **Proxy Server Settings**

Once you have verified that your Windows PC has Internet access, click *File -> Activation* and a window similar to the one shown below will be displayed:

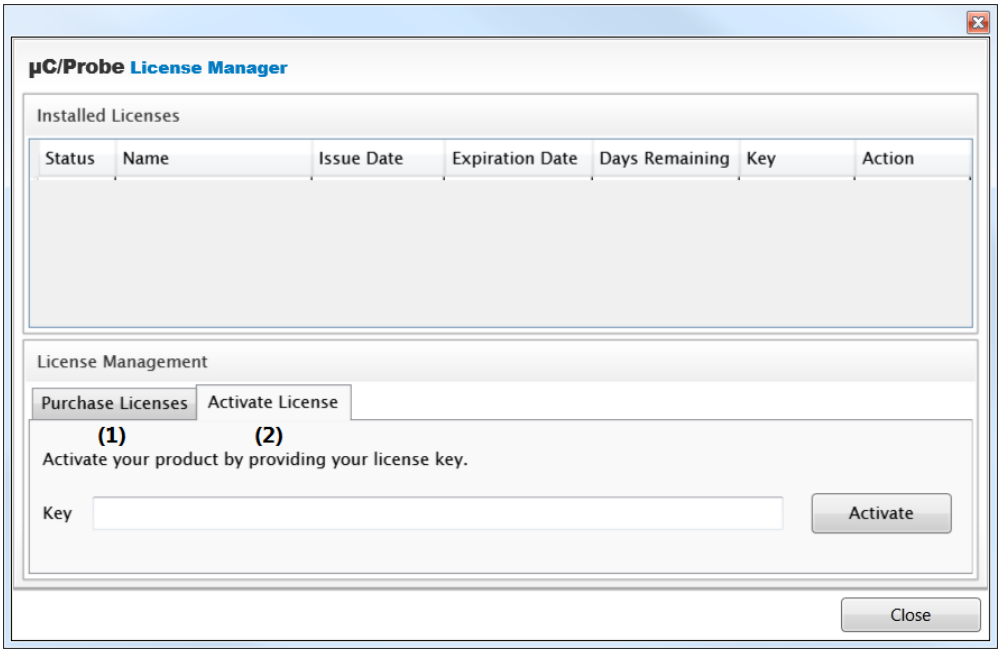


Figure L-2 **License Manager**

FL-2(1) Use this tab to go online and purchase a license of µC/Probe.

FL-2(2) Use this tab to activate a license key and node-lock it to your computer. All you need to do is copy and paste the license key from the e-mail message you received and press the button *Activate*.

Please contact us for further information about pricing, ordering options, license activation or cancelling a subscription at:

Micriμm
1290 Weston Road, Suite 306
Weston, FL 33326

+1 954 217 2036
+1 954 217 2037 (FAX)

E-Mail : sales@micrium.com
Website : www.micrium.com

Appendix

M

Bibliography

- Labrosse Jean. *µC/OS-II The Real-Time Kernel*. R&D Technical Books, ISBN 1-57820-103-9, 2002.

- Labrosse Jean. *µC/OS-III The Real-Time Kernel*. Micriµm Press, ISBN 978-0-98223375-3-0, 2009.

- Légaré Christian. *µC/TCP-IP The Embedded Protocol Stack*. Micriµm Press, 2011.

Index

Q

R

S

T

U

V

W

X

Z