Hands-On Workshop: How to Use Freescale's **FreeMASTER Tool for Development and Debugging**

AMF-ACC-T1244

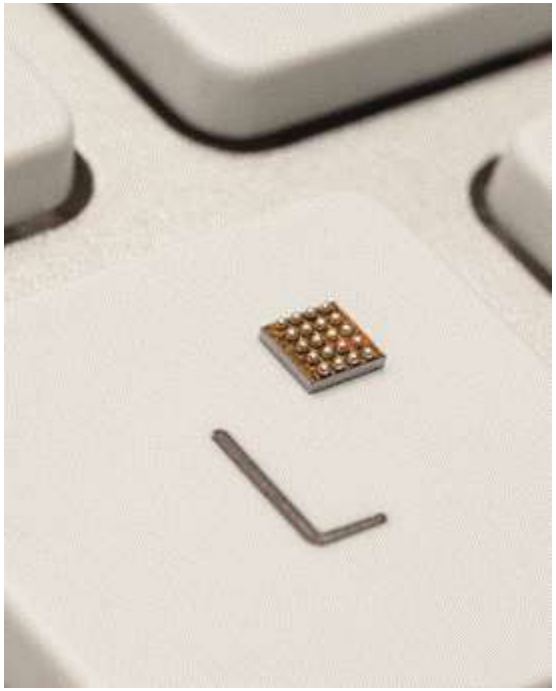
Michal Hanak | Application Engineer
John H. Floros | Field Application Engineer

SEPT.2015

freescale™

External Use

Introduction: MOTIVATION FOR FREEMASTER



- FreeMASTER was created as an internal development tool by our Motor Control team - “by the engineers for the engineers” in year 2000. Today it is maintained in the SW Libs team in Freescale.

- The original motivation was to be able to visualize and tune parameters without stopping the MCU core in the debugger. Breakpoints in code are often a luxury which you cannot afford in real time applications.

- As it matured a customizable and scriptable HTML rendering engine was added to enable custom GUI pages to be created and used to control, demonstrate or sell embedded applications.

FreeMASTER Overview
A Real-Time Monitor for Your Freescale MCU

Agenda

- What is FreeMASTER?
- FreeMASTER as a Real-Time Monitor
- FreeMASTER as a Control GUI
- FreeMASTER vs. Debugger
- FreeMASTER Replacing Custom GUI Applications
- FreeMASTER Internal Application Structure
- Summary

Objectives

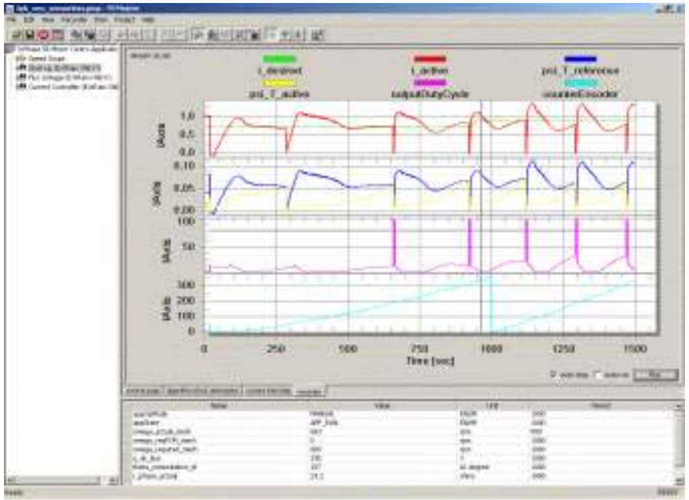
After this FreeMASTER Overview, you will know:

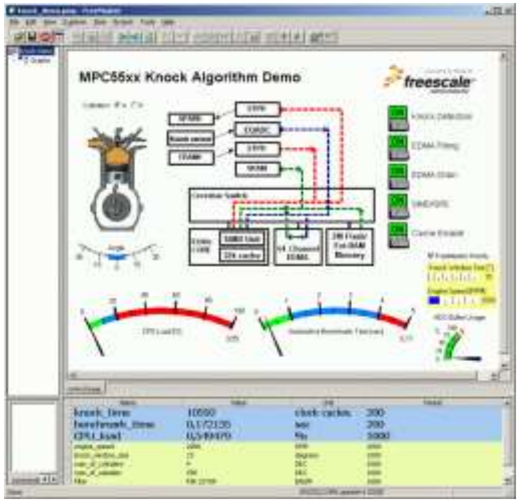
- ✓ What FreeMASTER is and what features it contains for real time monitoring of your application on the MCU
- ✓ How to configure some of the features available in the FreeMASTER user interface
- ✓ The steps required to enable FreeMASTER in your application at a high level

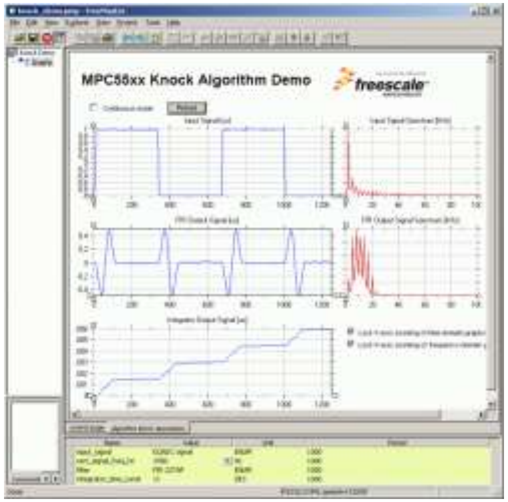
What is FreeMASTER?

- Real-time Monitor
- Graphical Control Panel
- Demonstration Platform & Selling Tool

FOR YOUR EMBEDDED APPLICATION

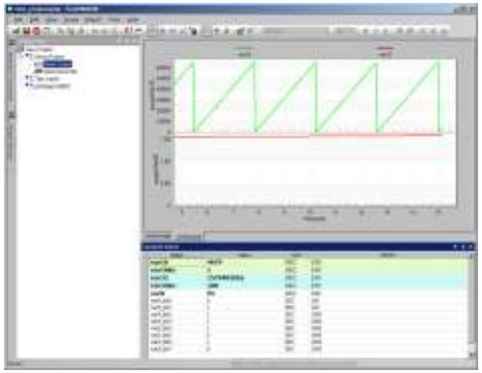


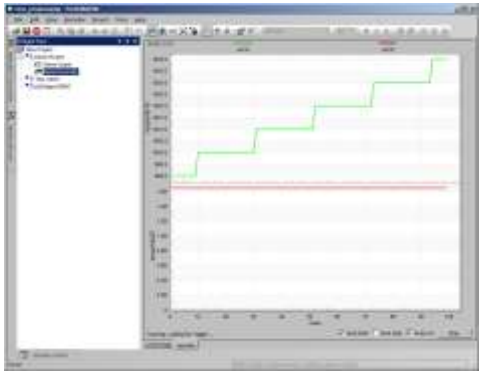


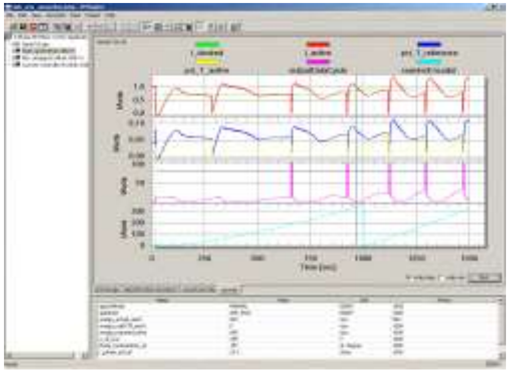


FreeMASTER as a Real-Time Monitor

- **FreeMASTER can Real-time Monitor**
  - **internal variables**
  - **processes & algorithms**
  - **application states**



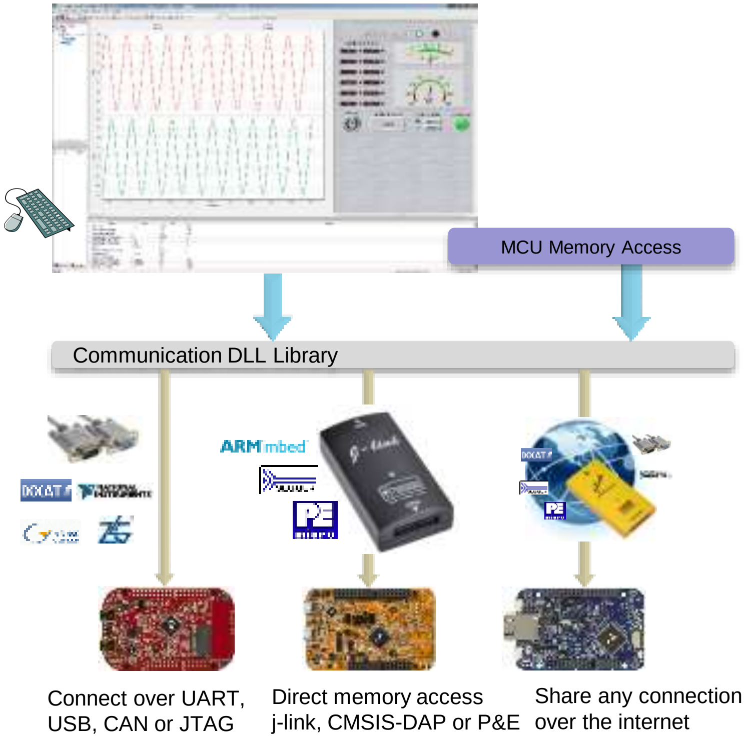




FreeMASTER as a Real-Time Monitor

- **PC Host Connects to an embedded application over unified DLL library**
  - **SCI, UART**
  - **USB**-CDC - Kinetis, ColdFire V2
  - **CAN** - msCAN, FlexCAN with PC interface from IXXAT, Vector, NI, Glinker, ZLG
  - **JTAG**/EOnCE (56F8xxx only)
  - **BDM** - Kinetis, PowerPC, ColdFire, HCS with Segger, P&E Micro, CMSIS-DAP, iSystem, ...
  - Any of the above remotely over the IP network

- **Enables access to application memory**
  - Parses ELF application executable file
  - Parses DWARF debugging information in the ELF file
  - Knows addresses of global and static C-variables
  - Knows variable sizes, structure types, array dimensions etc.



MCU Memory Access

Communication DLL Library

Connect over UART, USB, CAN or JTAG

Direct memory access j-link, CMSIS-DAP or P&E

Share any connection over the internet

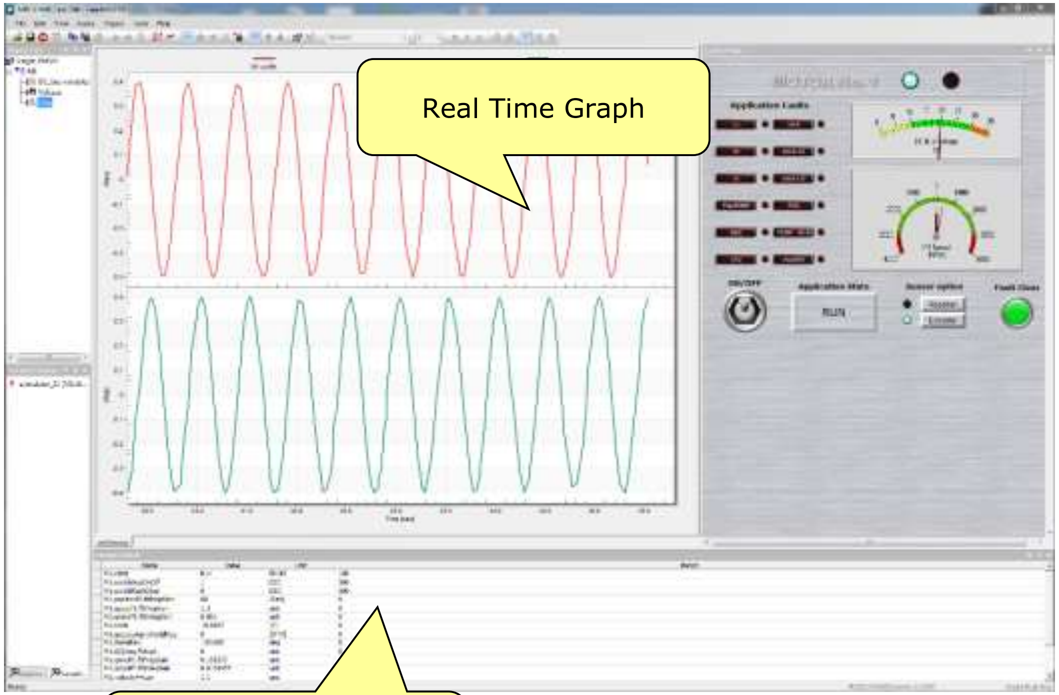
FreeMASTER Features

- Graphical environment
- Easy to understand navigation
- Real-time access to embedded-side C variables
- Visualization of real-time data in the Scope window
- Acquisition of fast data changes using the on-target Recorder
- Built-in support for standard variable types (integer, floating point, bit-fields)
- Demo mode with password protection support
- HTML-based description or navigation pages
- ActiveX interface to enable VBScript or JScript control over embedded applications
- Remote Communication Server enabling a connection to target board over a network, including the Internet

FreeMASTER as a Real-Time Monitor

Display the variable values in various formats:

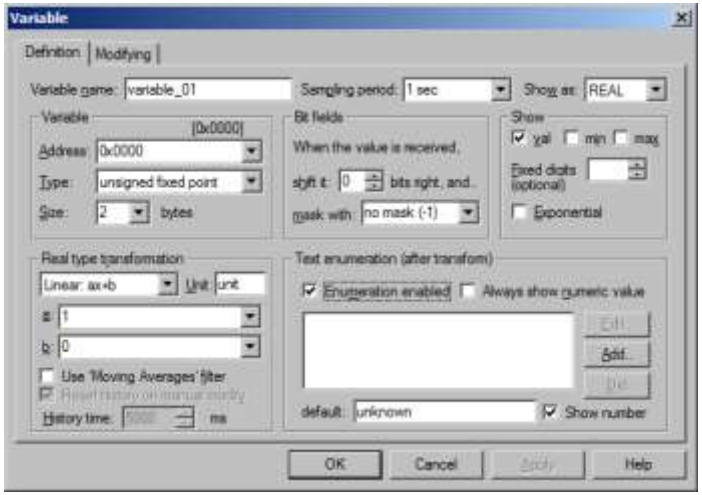
- **Text**, tabular grid
  - variable name
  - numeric value
  - peak detector
  - number-to-text enumeration
- **Real-time waveforms**
  - up to 8 variables simultaneously in an oscilloscope-like graph
- **High-speed recorded data**
  - up to 8 variables in on-board memory **transient recorder**



Real Time Graph

Variable Watch (one per project block)

FreeMASTER as a Real-Time Monitor


Variable

Definition | Modifying

Variable name: variable_01 | Sampling period: 1 sec | Show as: REAL

Variable [0x0000]
Address: 0x0000
Type: unsigned fixed point
Size: 2 bytes

Bit fields
When the value is received,
shift it: 0 bits right, and
mask with: no mask (-1)

Show
☑ val ☐ min ☐ max
Fixed digits (optional):
☐ Exponential

Real type transformation
Linear: ax+b Unit: unit
a: 1
b: 0
☐ Use 'Moving Averages' filter
☑ Reset history on manual modify
History time: 5000 ms

Text enumeration (after transform)
☑ Enumeration enabled ☐ Always show numeric value
default: unknown ☑ Show number

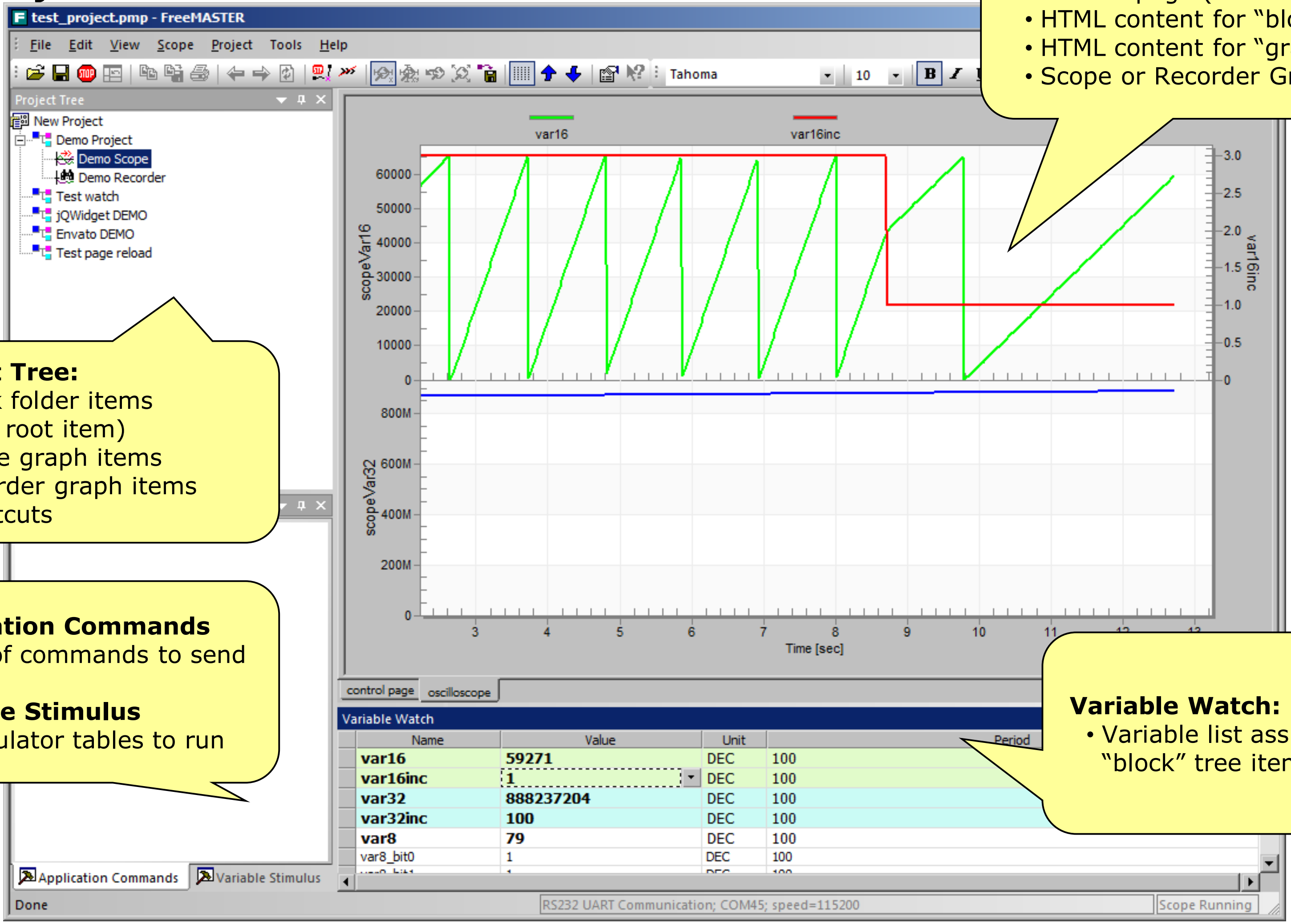
- **Variable Transformations**
  - Value can be transformed to custom units
  - Transformations may reference other variable values
  - Inverse-transformation applied when writing a new value to the variable

- **Ability to protect memory regions (TSA)**
  - Describing variables visible to FreeMASTER
  - Declaring variables as read-write to read-only for FreeMASTER - the access is guarded by the embedded-side driver

- **Application Commands**
  - Command code and parameters are delivered to an application for arbitrary processing
  - After processed (asynchronously to a command delivery) the command result code is returned to the PC
  - Legacy feature, not used in today’s applications (requires target-side driver)

FreeMASTER as a Real-Time Monitor

Anatomy of the main window



- New Project
  - Demo Project
    - Demo Scope
    - Demo Recorder
  - Test watch
  - jQWidget DEMO
  - Envato DEMO
  - Test page reload

Name	Value	Unit	Period
var16	**59271**	DEC	100
var16inc	**1**	DEC	100
var32	**888237204**	DEC	100
var32inc	**100**	DEC	100
var8	**79**	DEC	100
var8_bit0	1	DEC	100

Done | RS232 UART Communication; COM45; speed=115200 | Scope Running

Main pane:
- Control page (if not floating)
- HTML content for “block” items
- HTML content for “graph” items
- Scope or Recorder Graphs

Project Tree:
- Block folder items (also root item)
- Scope graph items
- Recorder graph items
- Shortcuts

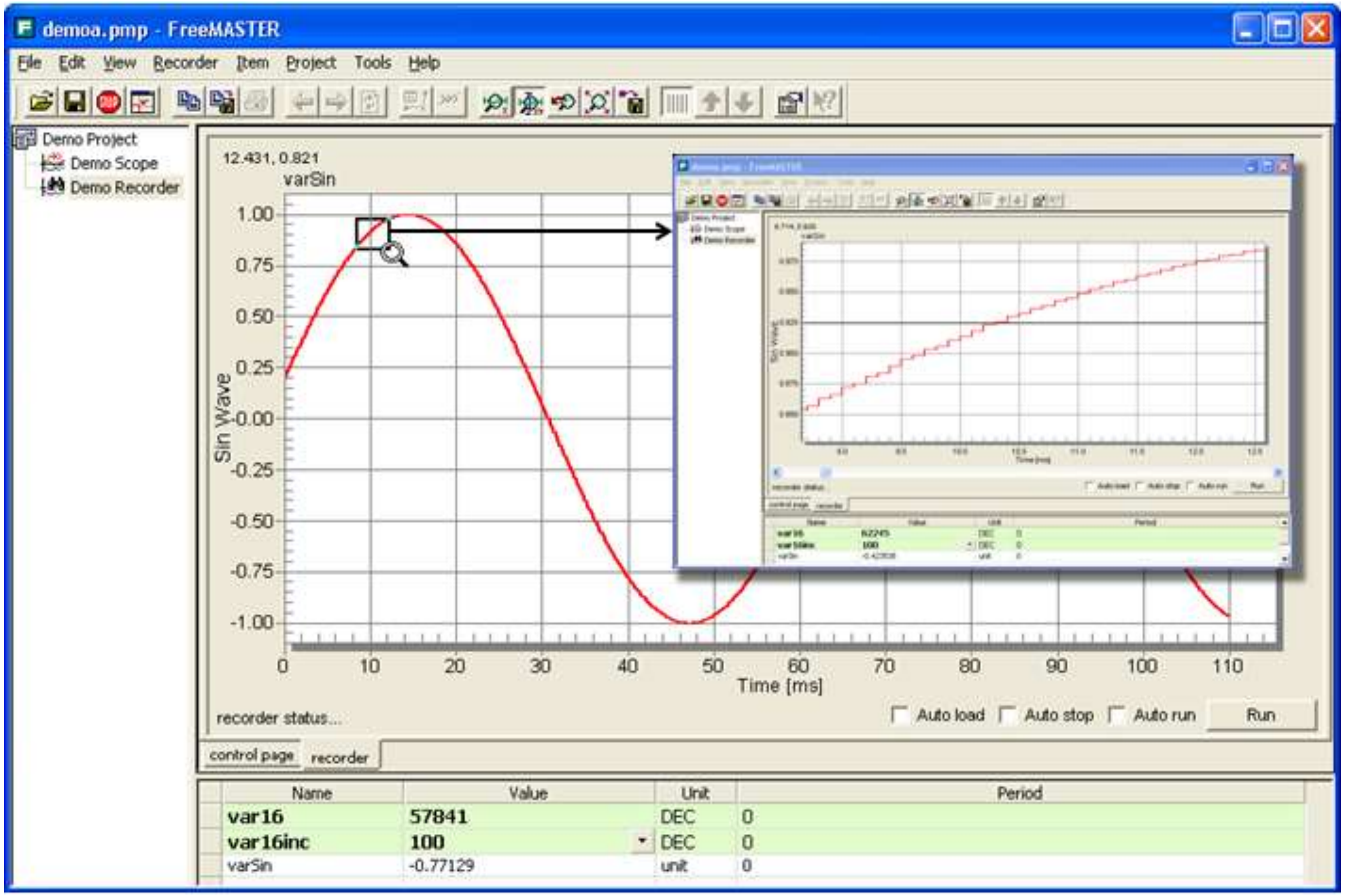
Application Commands
- List of commands to send

Variable Stimulus
- Stimulator tables to run

Variable Watch:
- Variable list assigned to “block” tree item

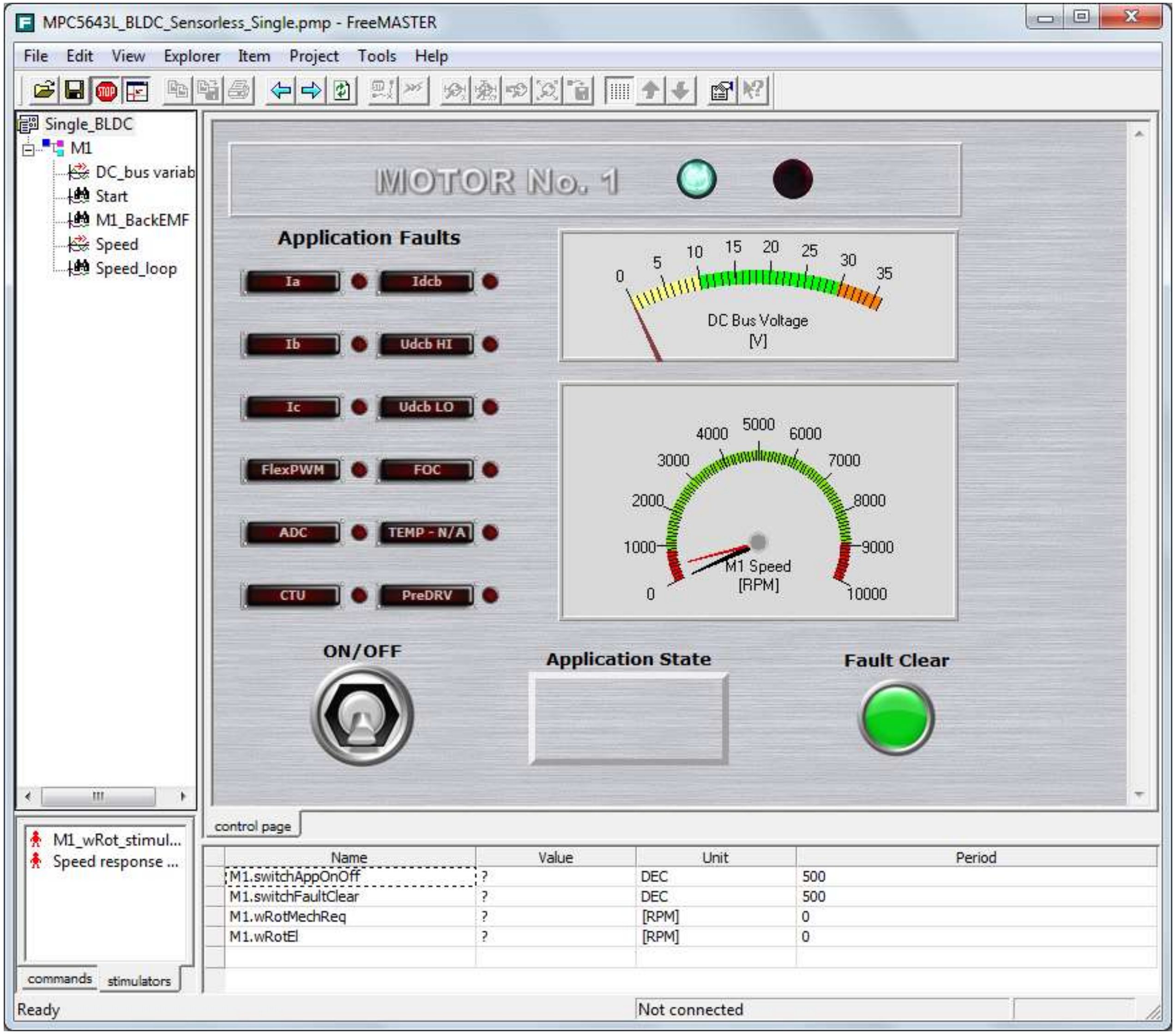
FreeMASTER as a Real-Time Monitor

- **Establish a Data Trace on Target**
  - Set up buffer (up to 64KB), sampling rate and trigger



Name	Value	Unit	Period
var16	57841	DEC	0
var16inc	100	DEC	0
varSin	-0.77129	unit	0

FreeMASTER as a Real-Time Monitor

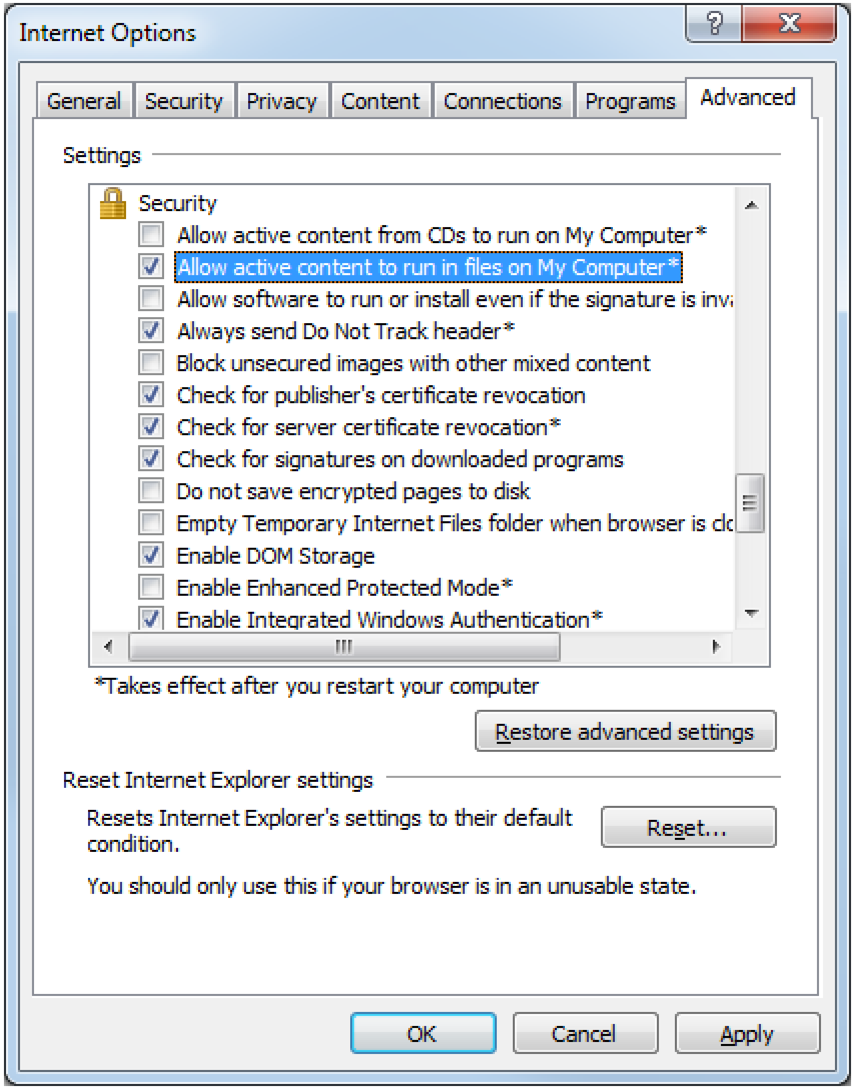


Name	Value	Unit	Period
M1.switchAppOnOff	?	DEC	500
M1.switchFaultClear	?	DEC	500
M1.wRotMechReq	?	[RPM]	0
M1.wRotEl	?	[RPM]	0

The HTML-based data visualization area. The user can provide any collection of ActiveX-based instrumentation to create custom visual dashboards as complex or elegant as desired.

The data visualization area may also be used to display arbitrary information, such as presentations, help files and fact sheets.

FreeMASTER as a Real-Time Monitor



Internet Options

Settings

Security

- ☐ Allow active content from CDs to run on My Computer*
- ☑ Allow active content to run in files on My Computer*
- ☐ Allow software to run or install even if the signature is inva
- ☑ Always send Do Not Track header*
- ☐ Block unsecured images with other mixed content
- ☑ Check for publisher's certificate revocation
- ☑ Check for server certificate revocation*
- ☑ Check for signatures on downloaded programs
- ☐ Do not save encrypted pages to disk
- ☐ Empty Temporary Internet Files folder when browser is clc
- ☑ Enable DOM Storage
- ☐ Enable Enhanced Protected Mode*
- ☑ Enable Integrated Windows Authentication*

*Takes effect after you restart your computer

Reset Internet Explorer settings

Resets Internet Explorer's settings to their default condition.

You should only use this if your browser is in an unusable state.

- In order to allow the ActiveX – based instrumentation to run it may be necessary to set you Internet options to allow the active content to run.

FreeMASTER as a Real-Time Monitor

Demo Mode

- To prevent modification, the project's author can lock the project against changes by switching it into the *Demo Mode.*
- An important part of the FreeMASTER's capabilities is the demonstration and description of the target board application. It is essential that the demonstration project, once prepared, is not accidentally modified.
- In the Demo Mode, the user cannot change the *Project Tree item properties, cannot add or remove the tree items, and cannot change any project options.*

FreeMASTER as a Real-Time Monitor
FreeMaster Communication Driver

- Go to www.freescale.com/freemaster
  - Go to the “downloads” tab and look for “FreeMASTER Communication Driver”
  - In the CodeWarrior project window, paste the FreeMASTER folder into the “Project_Headers” folder of your project
  - Once the package is installed, there are several options to interface with the target device, using CAN, SCI, or JTAG

For additional information, refer to Freescale’s Application Note AN4752

FreeMASTER as a Real-Time Monitor
Adding the FreeMaster Communication Driver

- The corresponding header and C files from the unpacked folder are added to the project header files.

- The paths containing the FreeMaster files must be incorporated into the project:
  - From the CW menu bar, go to Project > Properties
  - Go to “C/C++ Build”> “Settings”
  - Look for the item “Access Paths” under S12Z Compiler
  - Add the following paths under: ”Search User Paths”:

    "${ProjDirPath}\Project_Headers\FreeMASTER“

    "${ProjDirPath}\Project_Headers\FreeMASTER\S12ZVM“

    "${ProjDirPath}\Project_Headers\FreeMASTER\src_common"

FreeMASTER as a Real-Time Monitor Using the FreeMaster Serial Driver

- At the top of your project, we have included the freemaster header file:

```
#include "freemaster.h‟
```

- The “main” routine now includes a FreeMaster initialization (must be always after the comms initialization; in this case, the SCI):

```
FMSTR_Init();
```

- The infinite for loop now includes a function that continuously sends the variable values to FreeMaster

```
FMSTR_Poll();
```

FreeMASTER as a Real-Time Monitor
Steps to integrate FreeMASTER in your Application

1. Include the files under the **FreeMASTER Serial Communication Vxx\src_common** in your application code project with no changes.
2. One file changed in **FreeMASTER Serial Communication V1.6\src_platforms\MPC56xx** directory:
   a) renamed freemaster_cfg.h.example to freemaster_cgh.h
   b) Configure FreeMASTER by changing macro definitions
3. Addition to main.c
   a) Add function call FMSTR_Init() after system init
   b) Add function call FMSTR_Poll(); in main loop
4. To build with FreeMASTER support for MPC56xx, include all files under **FreeMASTER Serial Communication V1.6\src_platforms\MPC56xx** and **FreeMASTER Serial Communication V1.6\src_platforms\MPC56xx** directories.

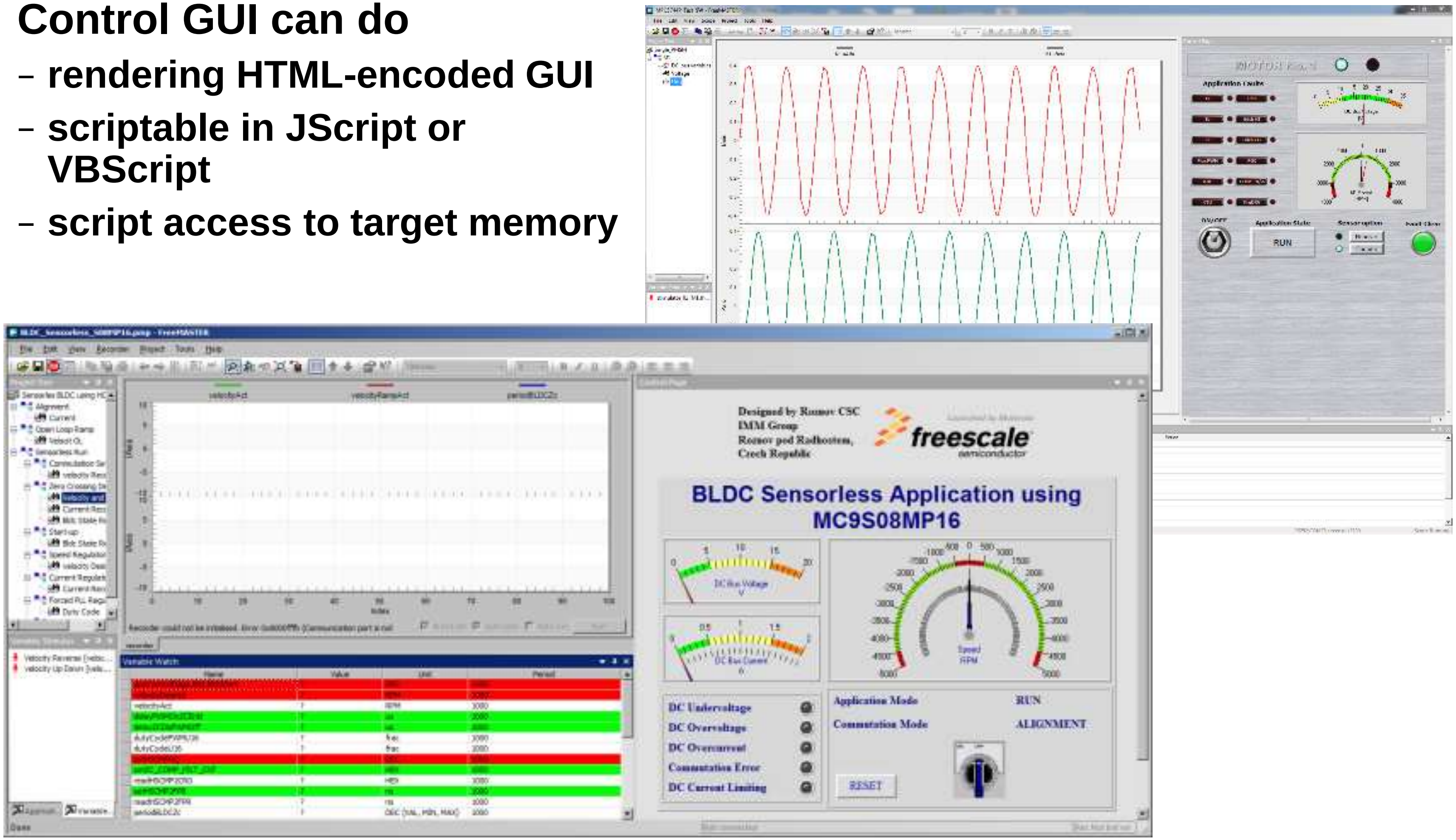
FreeMASTER as a Real-Time Monitor

Highlights

- Access to target variables, symbols and data types
- Safe access over UART, CAN or USB with target-side driver
- Transparent access over BDM (no target-side driver needed)
- Addresses parsed from ELF file or provided by target (TSA)
- Fine tuning parameters or direct control via variable modifications
- Scope graphs with real time data in [ms] resolution
- Recorder visualization transitions close to 10[us] resolution

FreeMASTER as a Control GUI

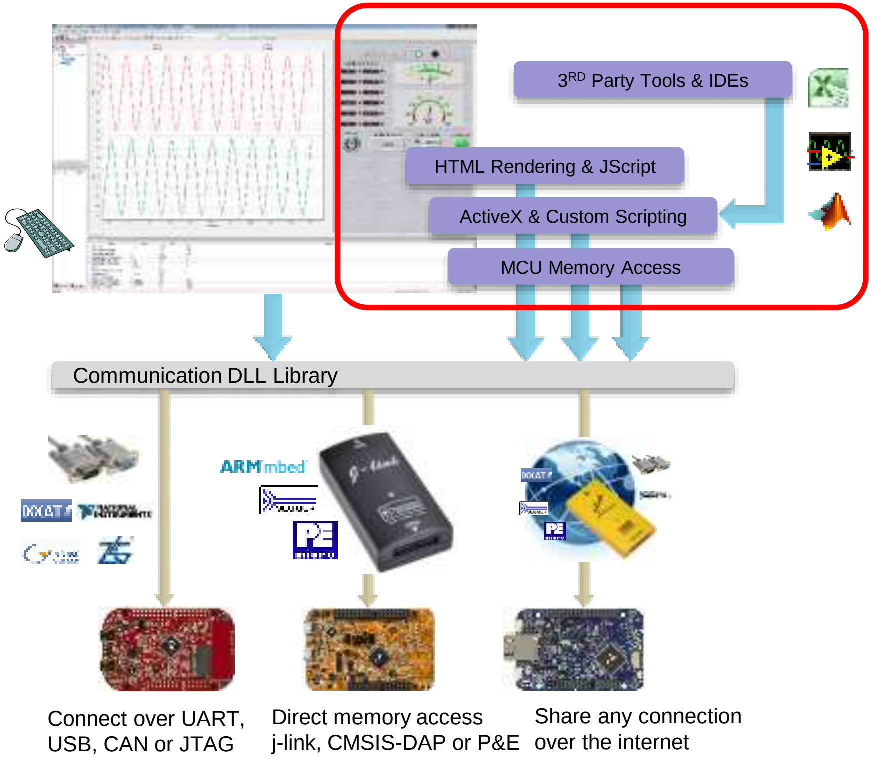
- **What the FreeMASTER Control GUI can do**
  - **rendering HTML-encoded GUI**
  - **scriptable in JScript or VBScript**
  - **script access to target memory**



FreeMASTER as a Control GUI

- **Variable access and modification**
  - Manually in the Watch pane
  - Time-tables & stimuli modification
  - Script-based read/write directly from GUI
    - mouse-clicks and keyboard control
    - push buttons and forms
    - sliders, gauges or other ActiveX/HTML5 widgets
    - custom intelligence and control algorithms
  - ActiveX clients external to FreeMASTER
    - Excel or Matlab – typical programmable clients
    - FreeMASTER enables HW-in-loop simulations
  - Works over all communication interfaces

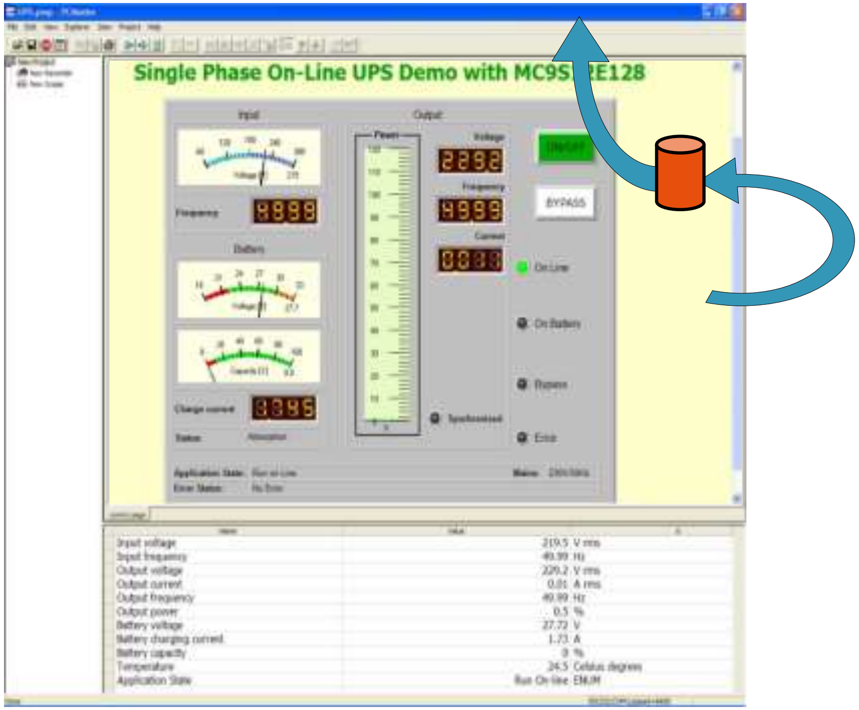
- **Sending Application Commands**
  - "Traditional" control approach
  - Not recommended as it is limited to systems with target-side agents (UART & CAN).


3RD Party Tools & IDEs
HTML Rendering & JScript
ActiveX & Custom Scripting
MCU Memory Access
Communication DLL Library
ARM mbed
IXXAT
P&E
Connect over UART, USB, CAN or JTAG
Direct memory access j-link, CMSIS-DAP or P&E
Share any connection over the internet

FreeMASTER as a Control GUI

- **Scripting in FreeMASTER**
  - HTML pages are displayed directly in the FreeMASTER window
  - InternetExplorer v10 used as the rendering engine
  - HTML may contain scripts and ActiveX objects

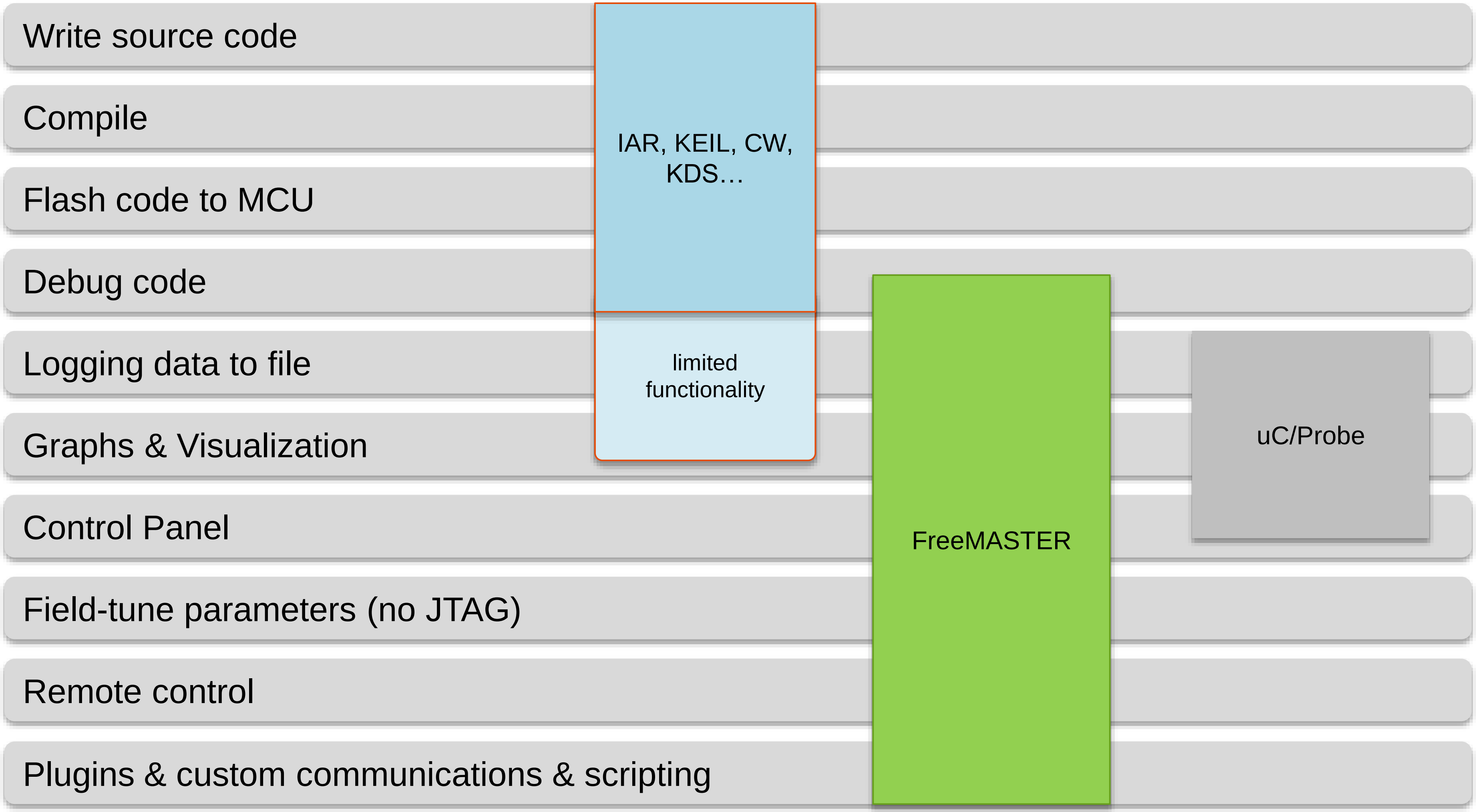
- **FreeMASTER invisible ActiveX object**
  - Script accesses the FreeMASTER functionality through this object
  - Variable access
  - Direct memory access
  - Stimulator access
  - Application Commands
  - Recorder Data
  - Symbol and data type information



Application State: Run on Line
Error Status: No Error
Mains: 230V/50Hz

Input voltage	219.5	V rms
Input frequency	49.99	Hz
Output voltage	229.2	V rms
Output current	0.01	A rms
Output frequency	49.99	Hz
Output power	0.5	%
Battery voltage	27.72	V
Battery charging current	1.73	A
Battery capacity	0	%
Temperature	24.5	Celsius degrees
Application State	Run On-line	ENUM

FreeMASTER vs. Debugger



Write source code

Compile

Flash code to MCU

Debug code

Logging data to file

Graphs & Visualization

Control Panel

Field-tune parameters (no JTAG)

Remote control

Plugins & custom communications & scripting

IAR, KEIL, CW, KDS…

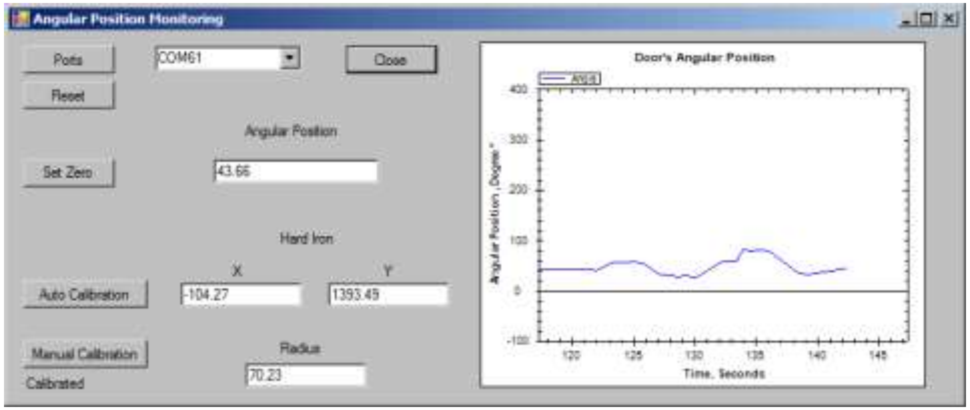
limited functionality

FreeMASTER

uC/Probe

FreeMASTER Replacing Custom GUI Applications

- **FreeMASTER instead of Custom GUIs**
  - **comparing FreeMASTER with custom GUI approach**
  - **typical use cases**



COM61

Angular Position 43.66

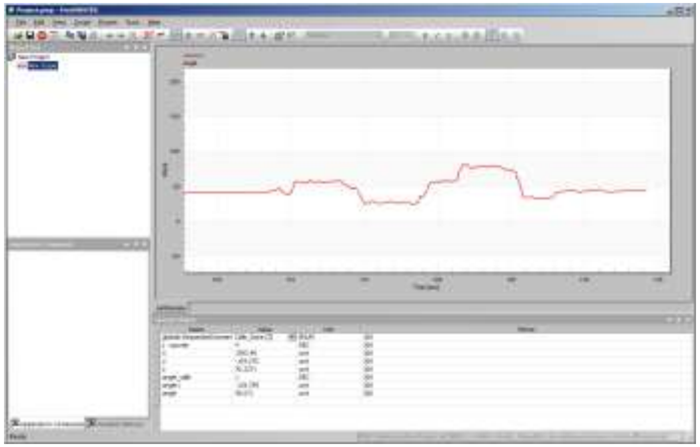
Hard Iron

X -104.27

Y 1393.49

Radius 70.23

Calibrated



From Custom GUI to FreeMASTER

- **Typical pitfalls of using custom GUI**
  - Requires PC Host programming tools and skills
  - Never enough communication interfaces, communication issues over and over again
  - Time to develop a robust PC Host application
  - Deploying GUI to host PC
  - Using custom GUI with modified user application

- **Benefits of FreeMASTER**
  - uniform approach – application control by variable modification
  - works over UART/CAN but also over non-intrusive BDM
  - one tool used with variety of GUIs
  - GUI easily extended by multimedia content (charts, documentation) local, online or embedded
  - Can be used with user-modified content too. User able to mix “our” data with “his” data in common charts.
  - GUI project can be extended by user to cover more functionality

From Custom GUI to FreeMASTER

- **Typical custom GUI approach**

  communication driven data collection, custom protocol

  - PC sends request, Target processes and replies with data
    - pro: under full control of developer, may be shielded from the rest of application logic
    - con: communication development just for sake of GUI, typically not used for any other purpose
    - con: migration to different communication media is typically hard
    - con: user modifications of firmware makes the GUI to stop working

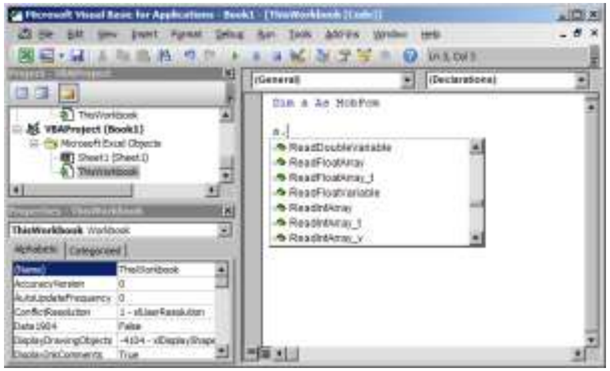
- **FreeMASTER approach**

  control by modifying variables

  - use either artificial variables dedicated for GUI control
  - or modify state variables used also by the general application algorithm
    - con: typically requires to change existing applications with custom GUI
    - pro: works over standardized protocol or with BDM direct memory access
    - pro: easy to protect or restrict functionality
    - pro: easy to integrate this approach with additional user modifications to firmware
    - pro: the TSA feature – self-describing and automatic board discovery (FreeMASTER 2.0 in 2015)

FreeMASTER Internal Application Structure

- **Inside FreeMASTER**
  - **how to get maximum out of FreeMASTER**
  - **linking with other executables**
  - **reusing communication layer**

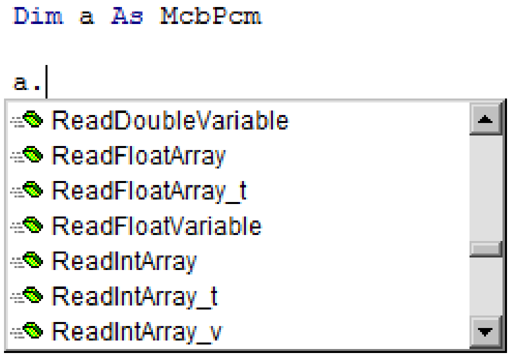


(Name)	ThisWorkbook
AccuracyVersion	0
AutoUpdateFrequency	0
ConflictResolution	1 - xlUserResolution
Date1904	False
DisplayDrawingObjects	-4104 - xlDisplayShape
DisplayInkComments	True

```
Dim a As McbPcm

a.
```

ReadDoubleVariable
ReadFloatArray
ReadFloatArray_t
ReadFloatVariable
ReadIntArray
ReadIntArray_t
ReadIntArray_v



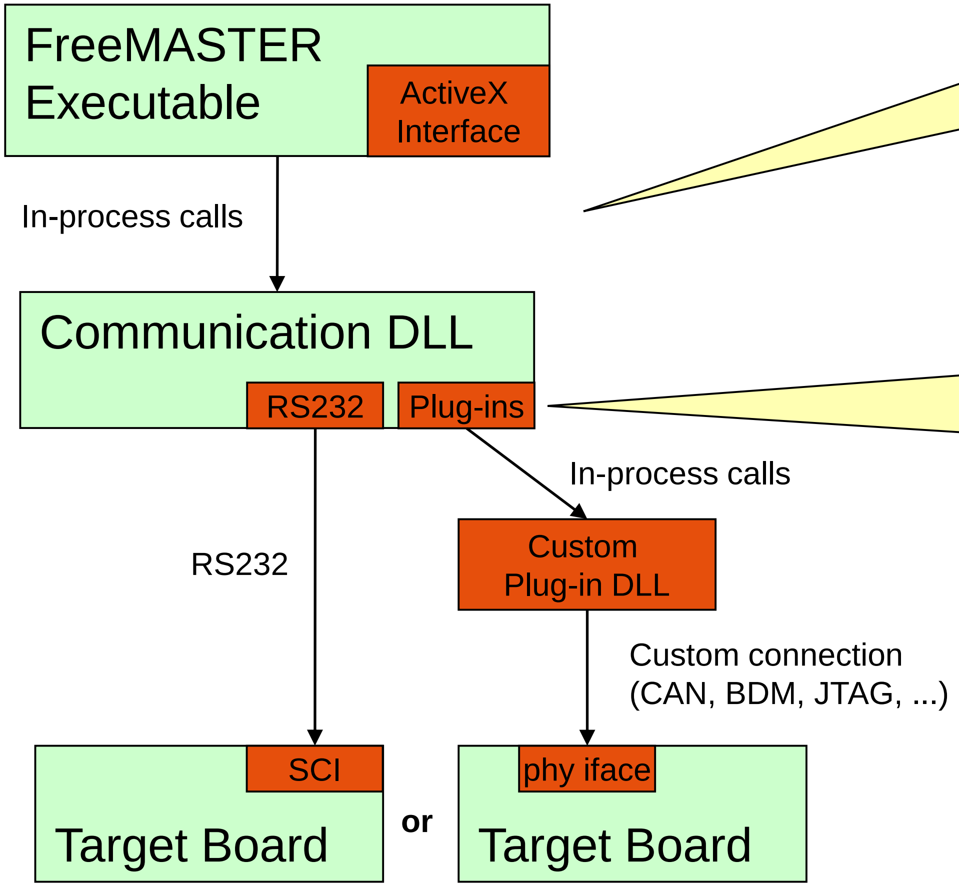
```
Dim a As McbPcm

a.
```

ReadDoubleVariable
ReadFloatArray
ReadFloatArray_t
ReadFloatVariable
ReadIntArray
ReadIntArray_t
ReadIntArray_v

FreeMASTER Internal Application Structure

Basic FreeMASTER Communication Diagram



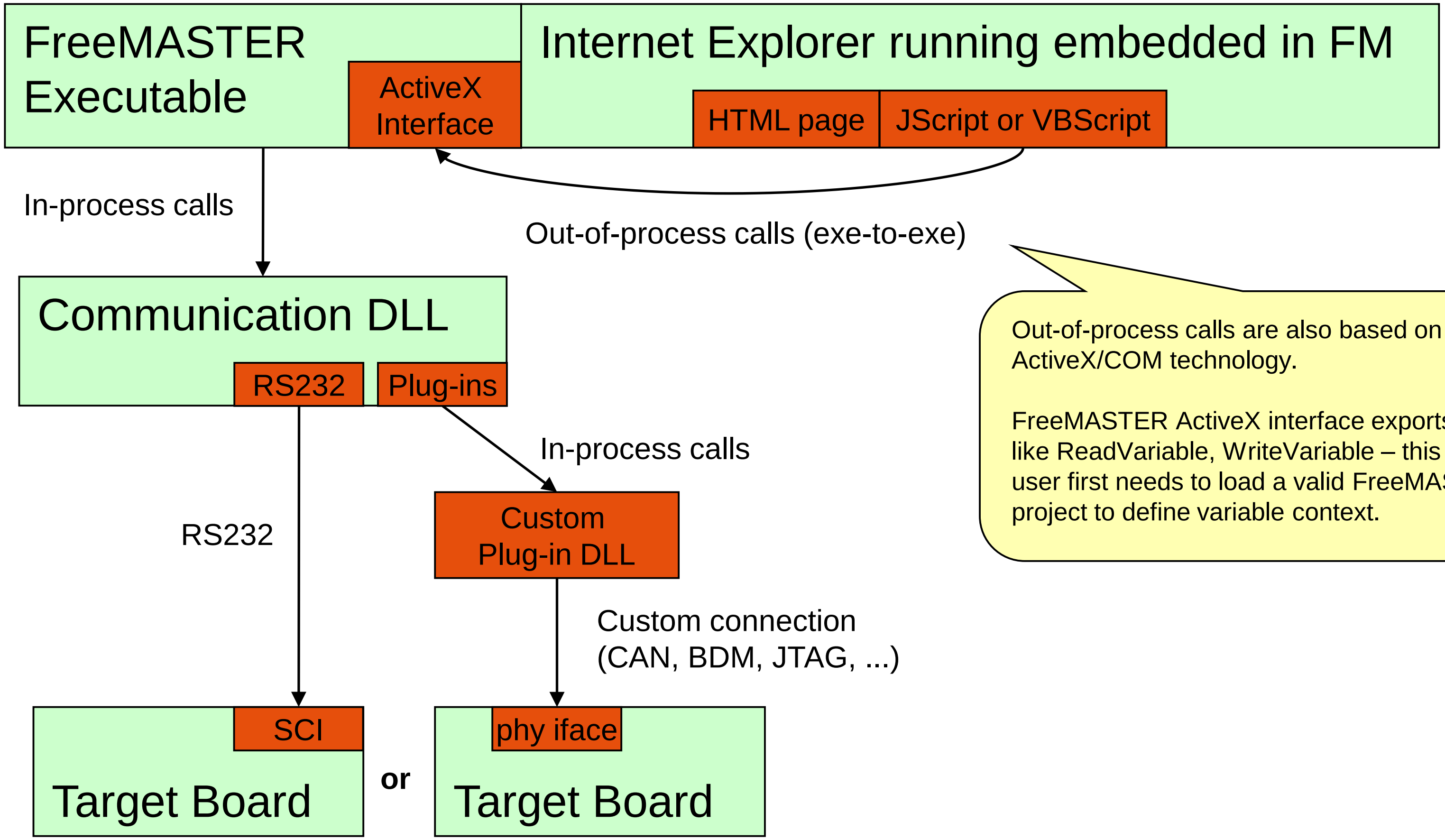
So-called in-process calls are the simple calls to functions located in dynamically loaded libraries (DLLs). The calls are fast just like if the function would be located inside the executable itself.

FreeMASTER enables to use custom plug-in modules to implement the communication layer.

The plug-ins use Microsoft COM (ActiveX) procedure call standard. Typically the fast in-process plug-in DLLs are used.

FreeMASTER Internal Application Structure

FreeMASTER Communication with HTML/JScript Pages

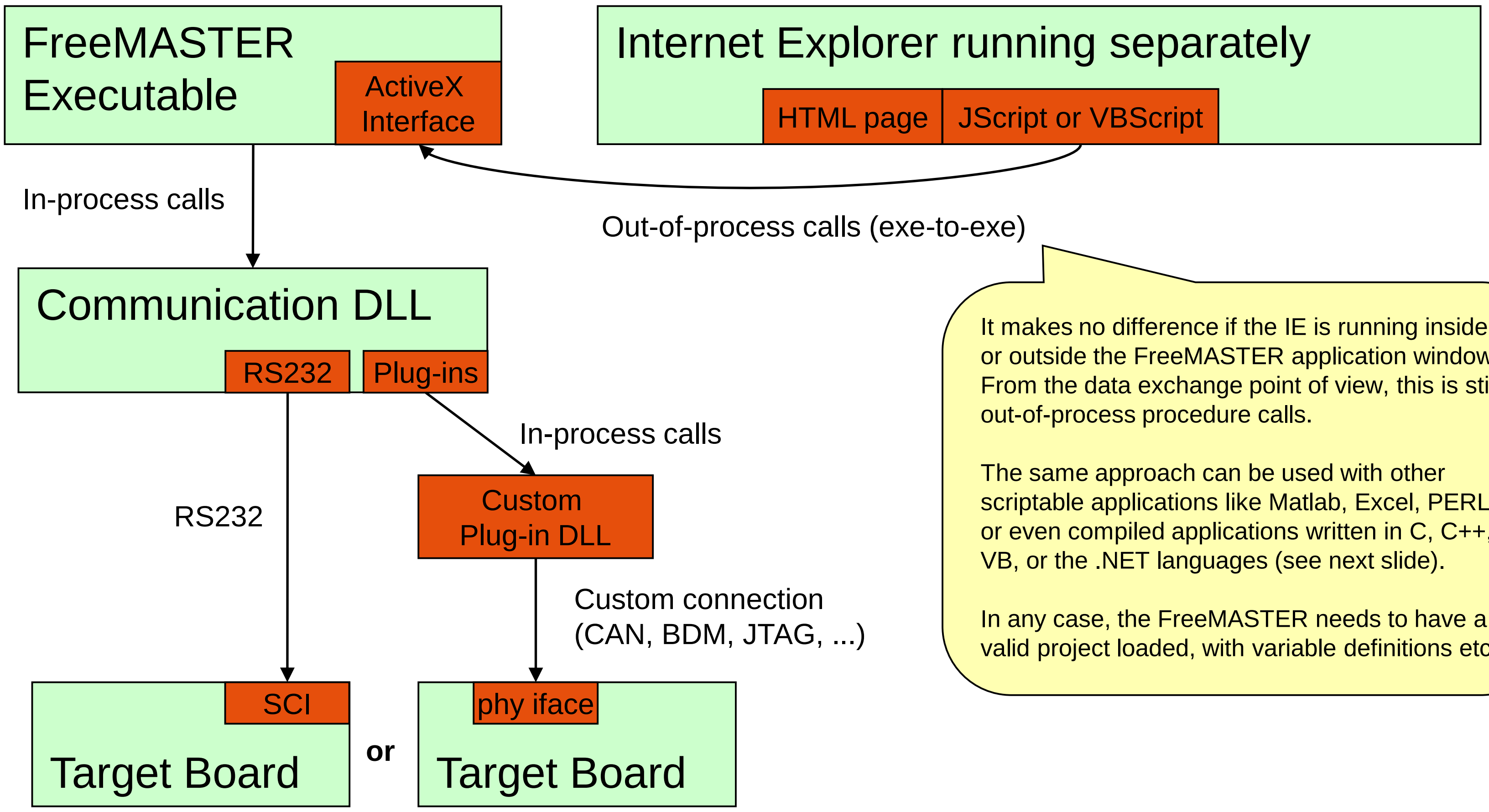


Out-of-process calls are also based on Microsoft ActiveX/COM technology.

FreeMASTER ActiveX interface exports methods like ReadVariable, WriteVariable – this means the user first needs to load a valid FreeMASTER project to define variable context.

FreeMASTER Internal Application Structure

Internet Explorer Running Separately (no difference)



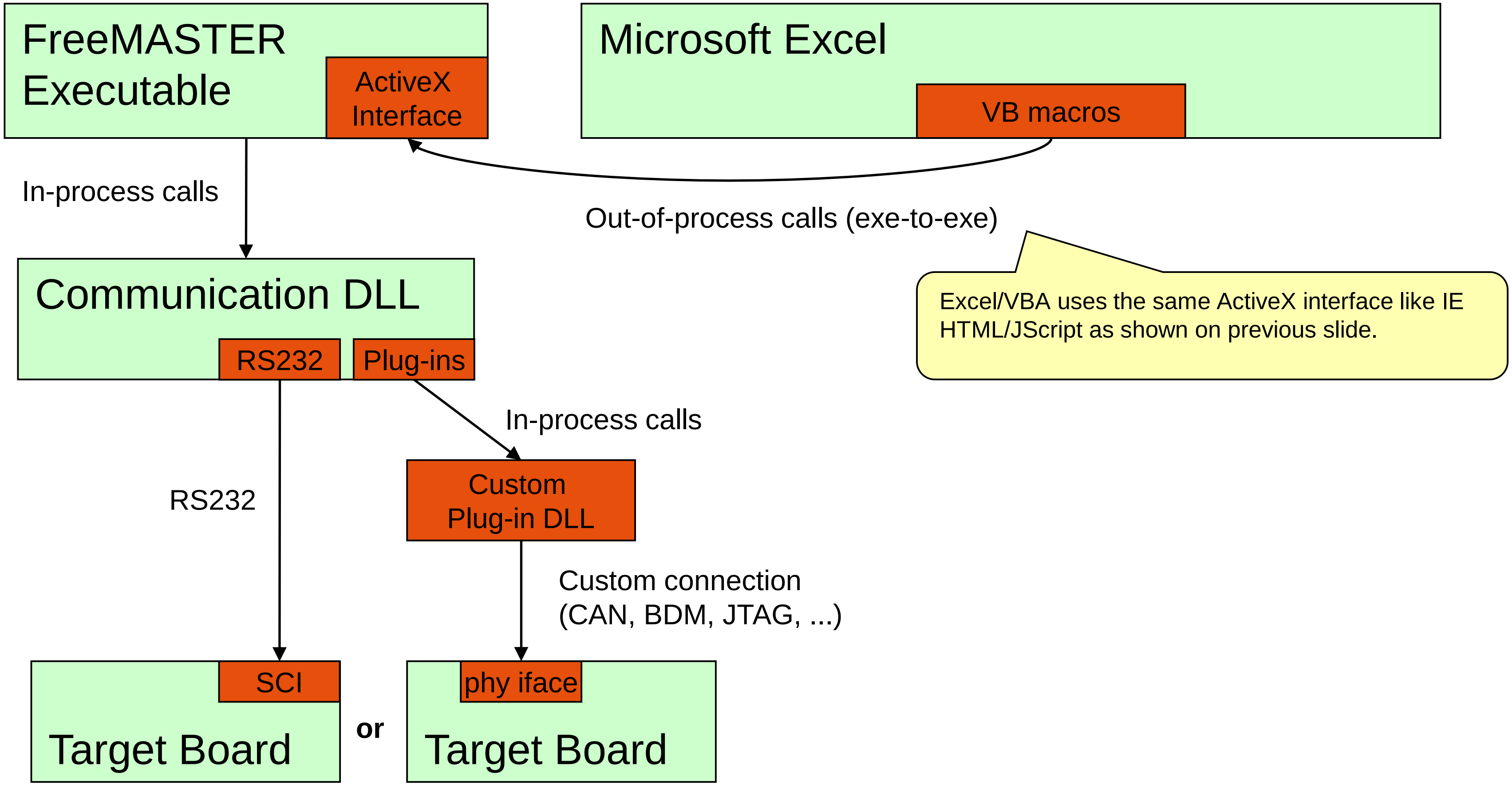
It makes no difference if the IE is running inside or outside the FreeMASTER application window. From the data exchange point of view, this is still out-of-process procedure calls.

The same approach can be used with other scriptable applications like Matlab, Excel, PERL, or even compiled applications written in C, C++, VB, or the .NET languages (see next slide).

In any case, the FreeMASTER needs to have a valid project loaded, with variable definitions etc.

FreeMASTER Internal Application Structure

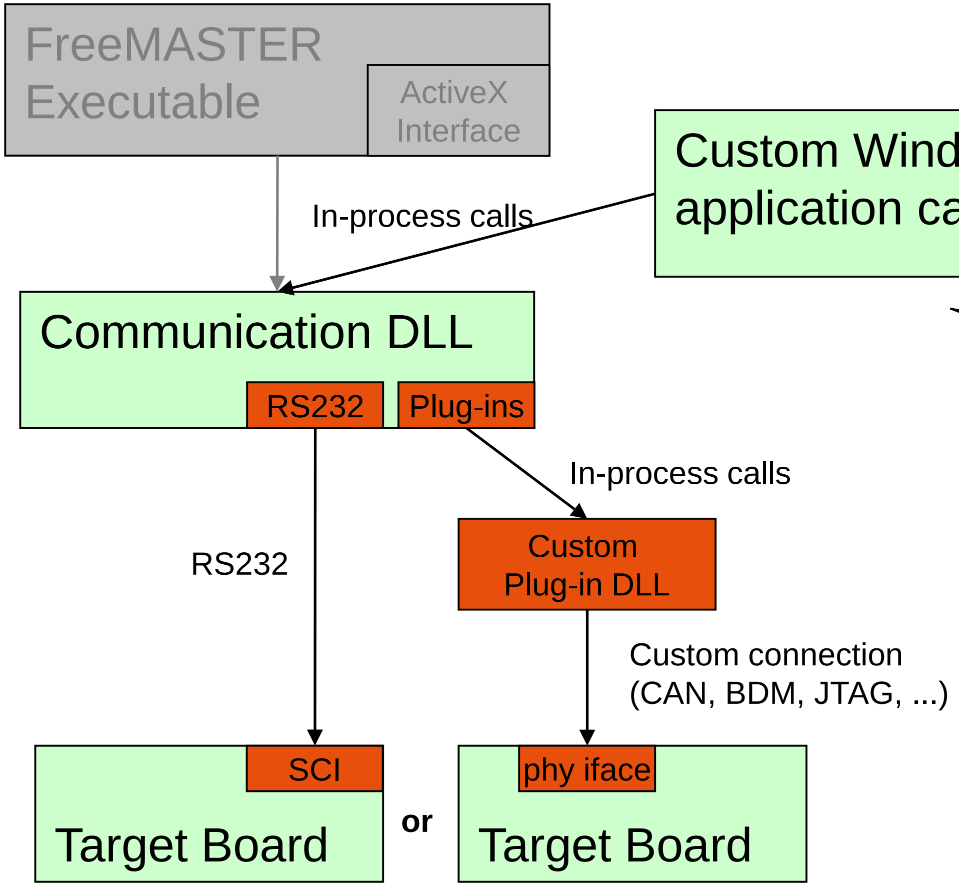
Excel (or other application) accessing FM ActiveX



Excel/VBA uses the same ActiveX interface like IE HTML/JScript as shown on previous slide.

FreeMASTER Internal Application Structure

Other Ways to Access Target Microprocessor: C, C++



Any Windows-based application which is capable of direct C-like calls into the DLL may reuse the communication DLL to make use of the FreeMASTER protocol.

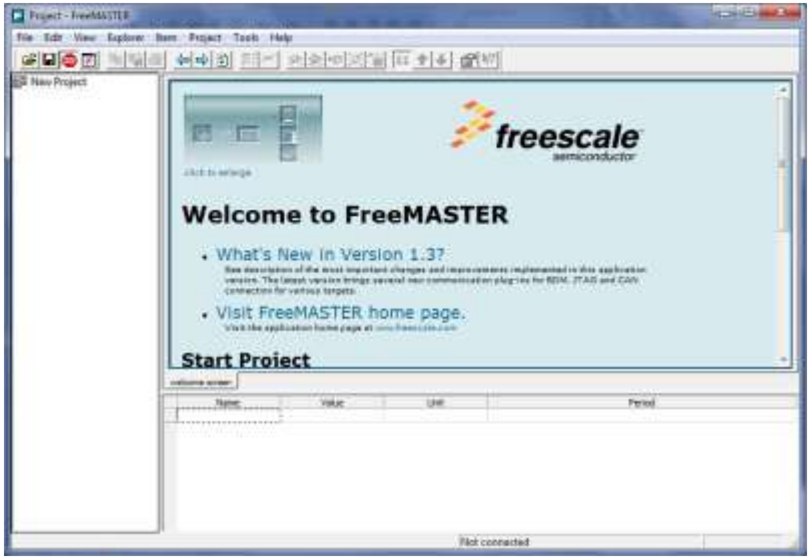
The DLL exports calls like OpenPort, ClosePort, ReadMemory, WriteMemory, SetupRecorder, ...

As FreeMASTER is out of the game here, the term “variable” makes no more sense in this scenario. User application needs to use numeric memory addresses and sizes when accessing the board (see FM protocol for more details)

Start FreeMASTER Interface

- From the Start Menu in Windows, go to
  - Start > All Programs > FreeMaster 1.4

- The FreeMASTER tool will start
  - ignore all the warnings and error messages, they are most probably caused by incorrectly assigned serial port)



Welcome to FreeMASTER

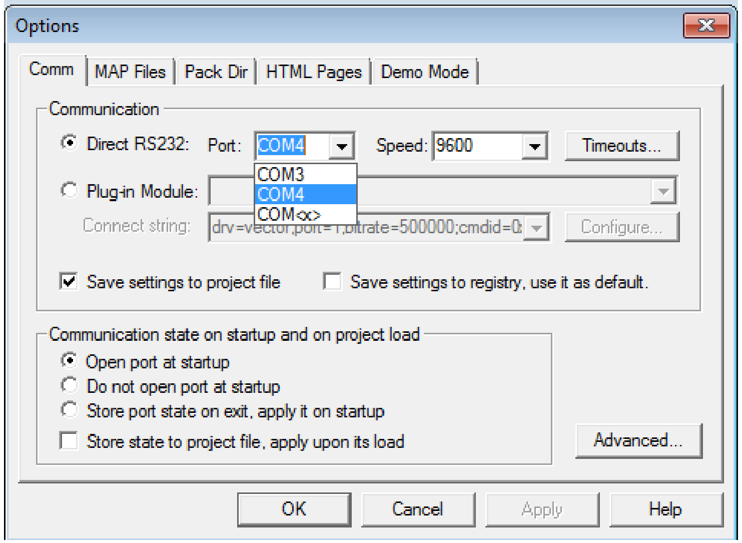
- What's New in Version 1.3?
  See description of the most important changes and improvements implemented in this application version. The latest version brings several new communication plug-ins for BDM, JTAG and CAN connection for various targets.
- Visit FreeMASTER home page.
  Visit the application home page at www.freescale.com

Start Project

FreeMaster – Configuring the Serial Port

- On the menu bar, go to Project > Options

- Select the correct COMM port, with a speed setting of 9600 (this is the value we used in the SCI initialization)



Communication

● Direct RS232: Port: COM4 Speed: 9600

COM3
COM4
COM<x>

○ Plug-in Module:

Connect string: drv=vector;port=1;bitrate=500000;cmdid=0

☑ Save settings to project file ☐ Save settings to registry, use it as default.

Communication state on startup and on project load

● Open port at startup
○ Do not open port at startup
○ Store port state on exit, apply it on startup
☐ Store state to project file, apply upon its load

FreeMaster – Loading the MAP file

- From the options window, go to tab “MAP Files”

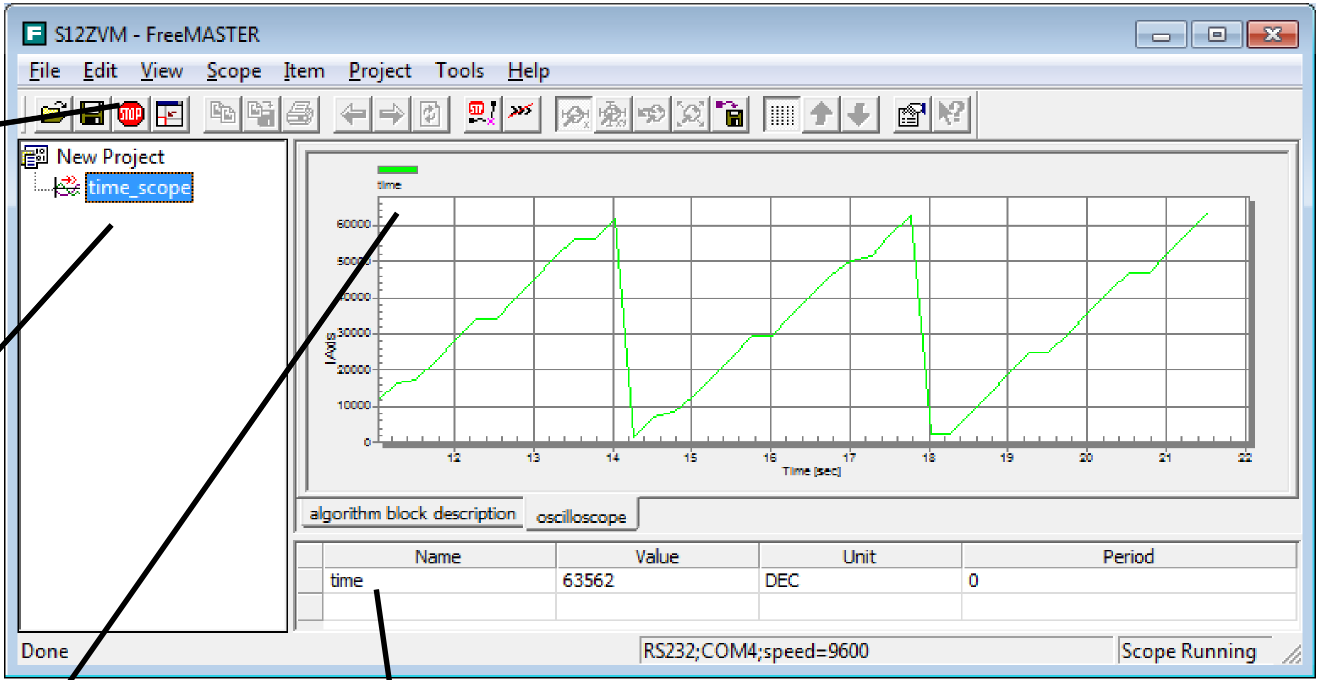
- Select the default symbol file:
  - Click on “…” and browse to the location where the ELF file is stored (C:\BLDC_workshop\ S12ZVM_Lab2\FLASH\)
  - Select the file “S12ZVM_Lab2.elf”

- Select the file format:
  - Binary ELF with DWARF1 or DWARF2 dbg format

- Click OK

FreeMASTER Interface

- In the FreeMASTER interface for “Empty Project” variable time is watched. This variable is also added to scope interface in order to be monitored in graphical representation.



Name	Value	Unit	Period
time	63562	DEC	0

Done

RS232;COM4;speed=9600

Scope Running

Start/Stop serial communication with target

Variable “time” in watch window

Visualization of variable “time” in scope

Scope selector

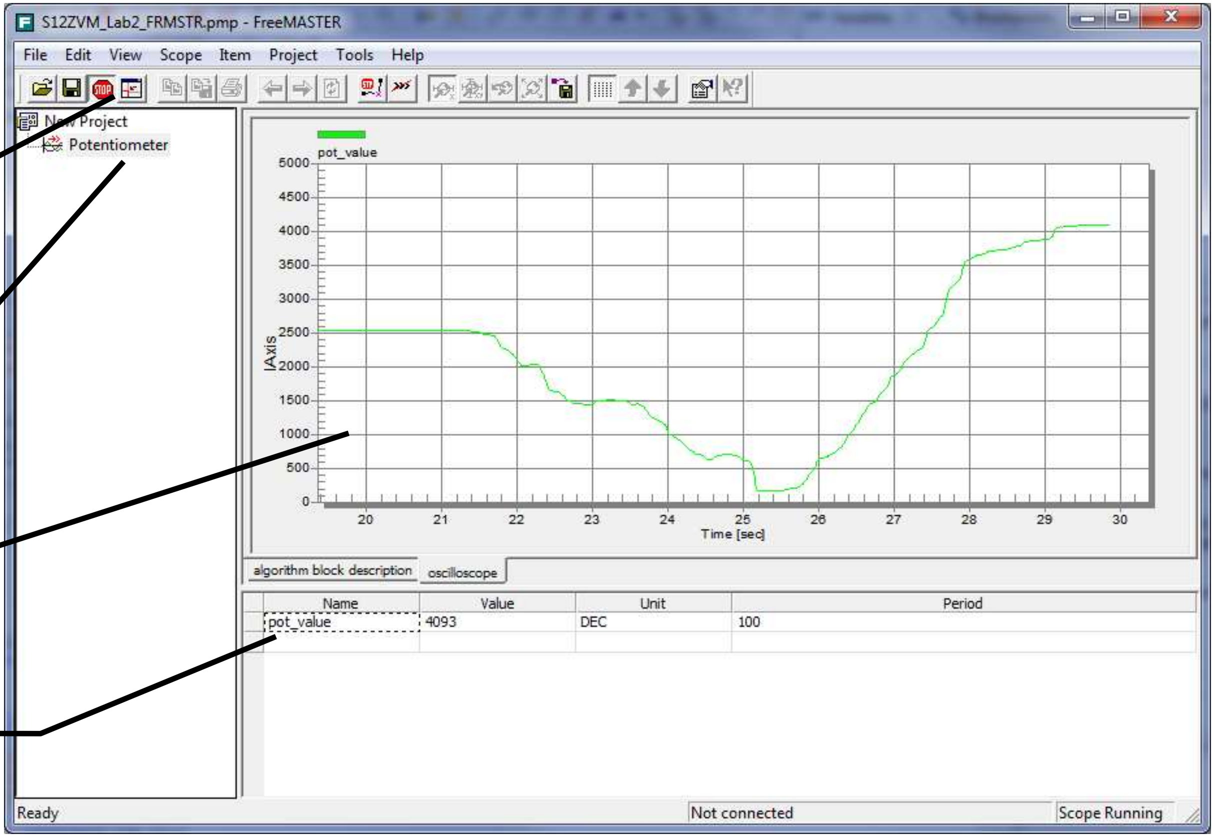
FreeMASTER Interface

Start/Stop serial communication with target

Scope selector

Visualization of variable in scope

Variable in watch window

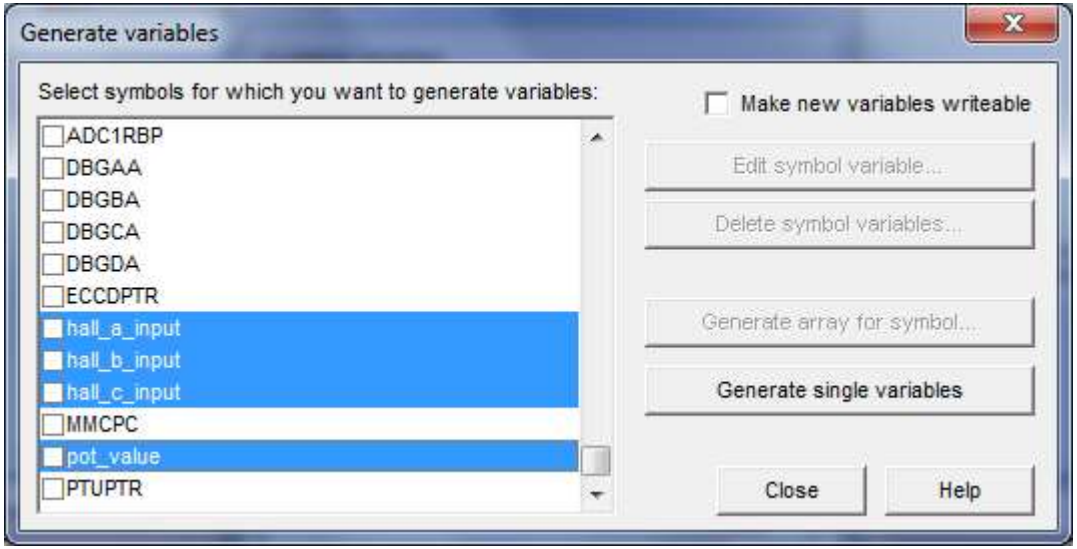


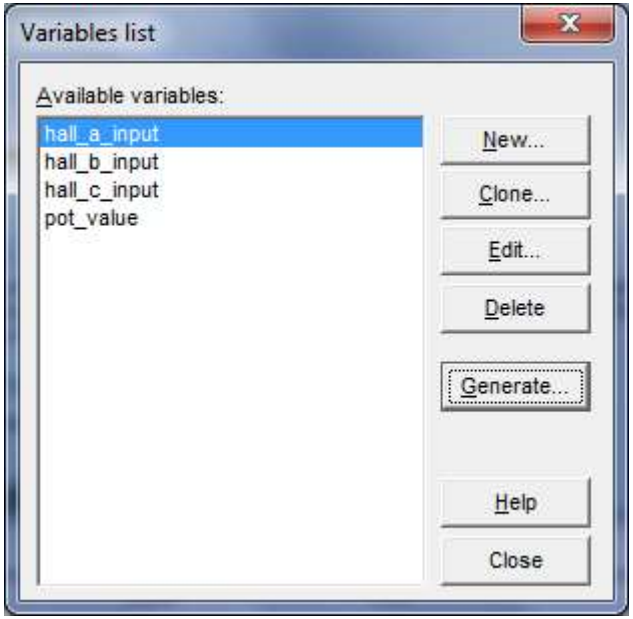
Name	Value	Unit	Period
pot_value	4093	DEC	100

Ready | Not connected | Scope Running

Adding Variables

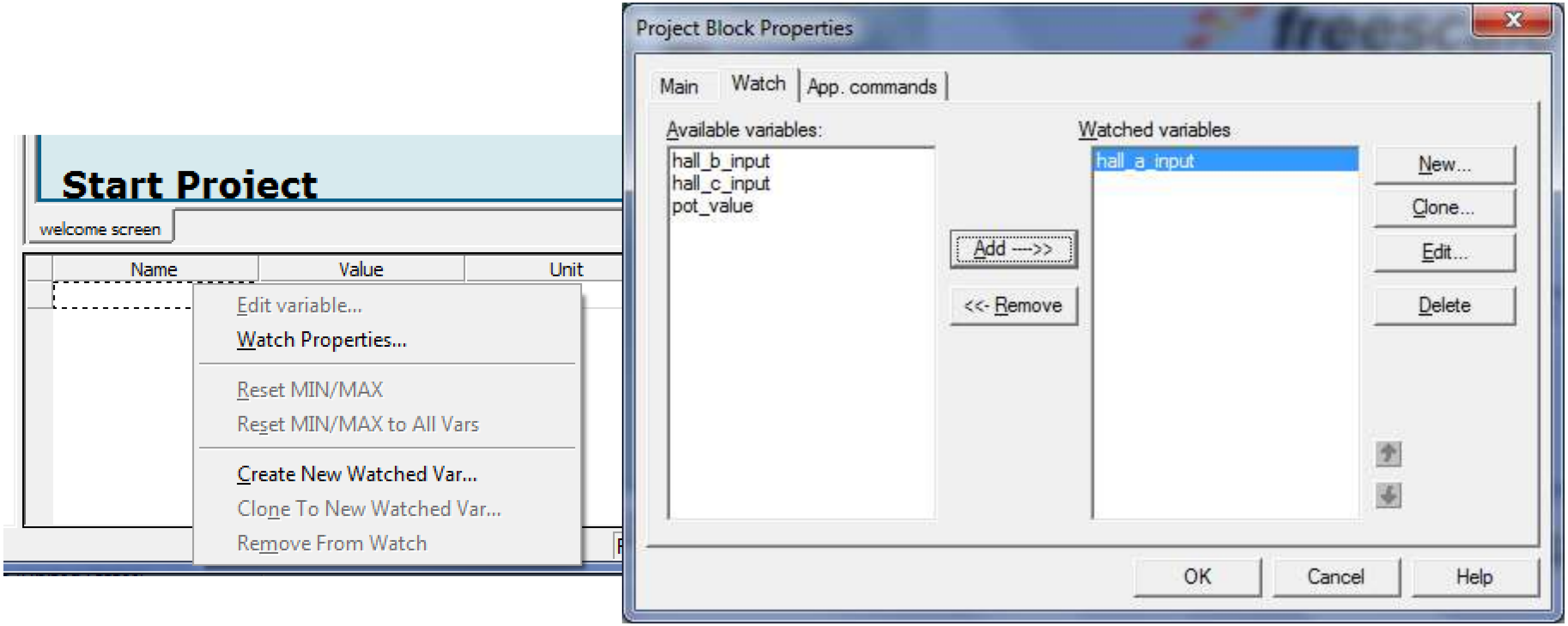
- On the menu bar, go to Project > Variables
- When the window appears, select “Generate”
- Scroll down the list of variables and find the global variables that will be monitored
- Click on “Generate single variables”, then “Close”
- Close the “Variables list” window


Generate variables
Select symbols for which you want to generate variables:
□ ADC1RBP
□ DBGAA
□ DBGBA
□ DBGCA
□ DBGDA
□ ECCDPTR
□ hall_a_input
□ hall_b_input
□ hall_c_input
□ MMCPC
□ pot_value
□ PTUPTR
□ Make new variables writeable
Edit symbol variable...
Delete symbol variables...
Generate array for symbol...
Generate single variables
Close
Help



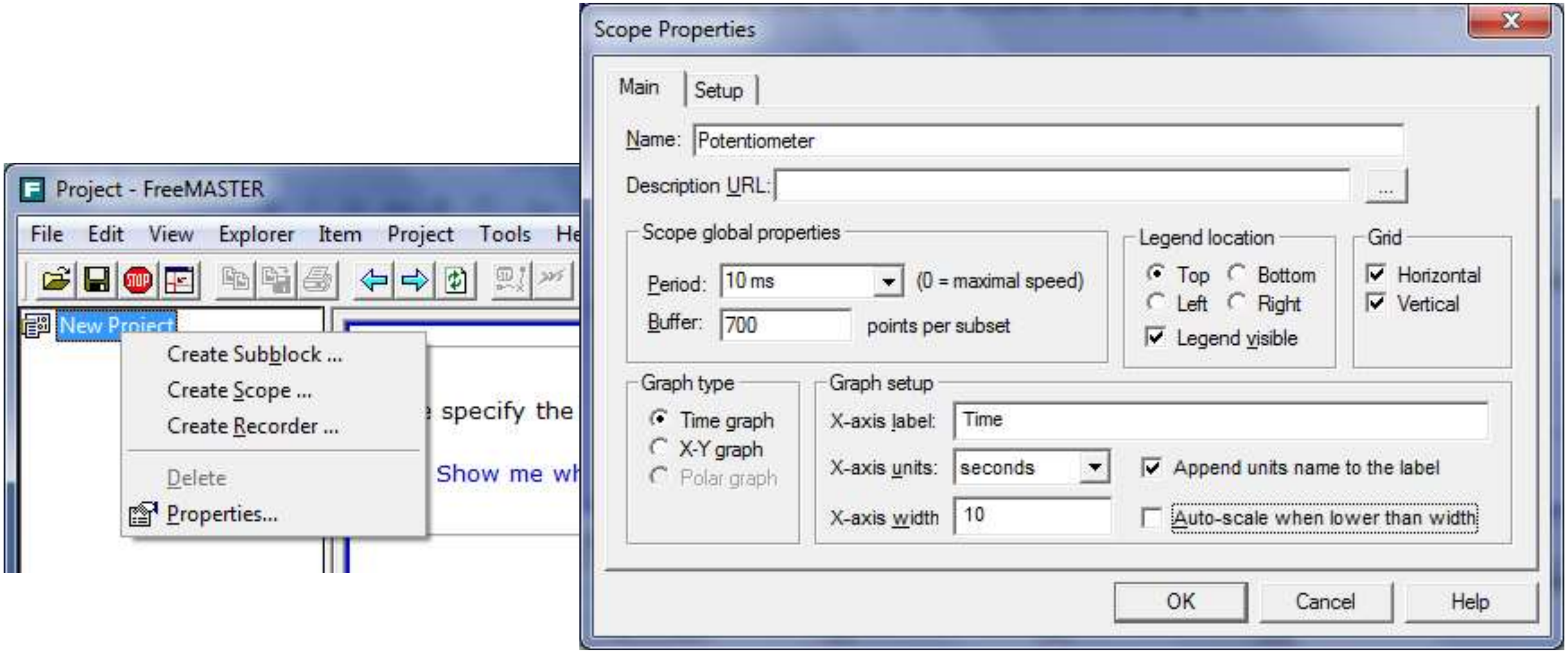
Adding variables to the Watch List

- Right click into “watch” area and select “Watch Properties”
- Switch to tab “Watch” in Project Block Properties
- Select the variables to watch and click on “Add”



Adding a Scope

- Right-click on New Project and select the option “Create Scope”
- Define a name for the scope
- Change Period to 10ms, and Buffer to 700 points per subset



Name: Potentiometer

Description URL:

Scope global properties

Period: 10 ms (0 = maximal speed)

Buffer: 700 points per subset

Legend location: ● Top ○ Bottom ○ Left ○ Right

☑ Legend visible

Grid: ☑ Horizontal ☑ Vertical

Graph type: ● Time graph ○ X-Y graph ○ Polar graph

Graph setup

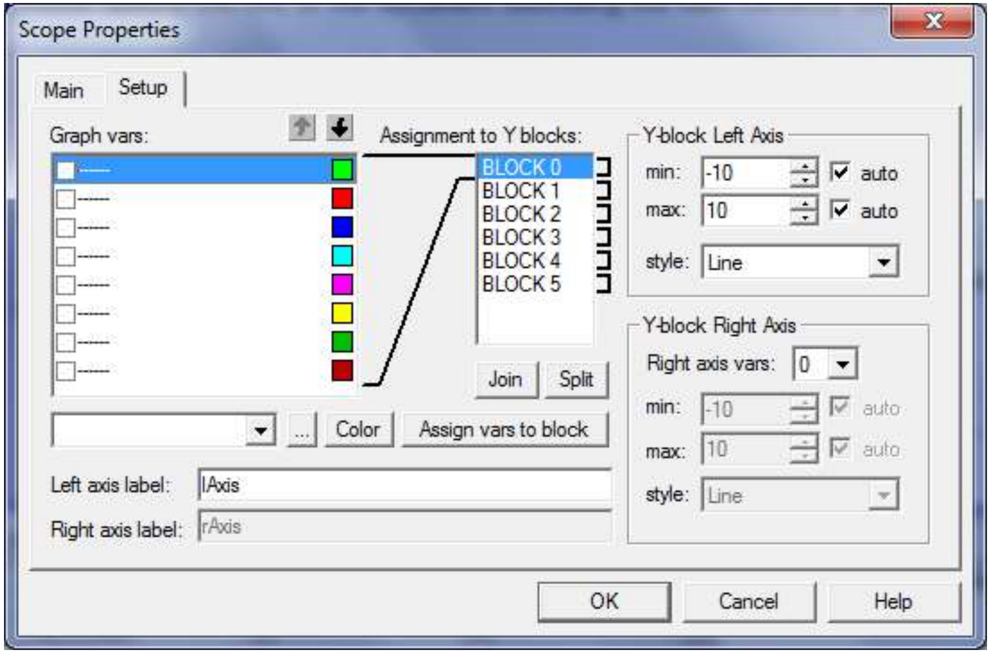
X-axis label: Time

X-axis units: seconds ☑ Append units name to the label

X-axis width: 10 □ Auto-scale when lower than width

Setup a variable in the scope

1. Select the first unassigned variable slot
2. Select the variable pot_value from the dropdown list
3. With BLOCK 0 selected, click on “Assign vars to block”
4. Set the Y-block left axis min value to 0, max value to 5000.



Scope Properties

Graph vars:

Assignment to Y blocks: BLOCK 0, BLOCK 1, BLOCK 2, BLOCK 3, BLOCK 4, BLOCK 5

Y-block Left Axis

min: -10 ☑ auto

max: 10 ☑ auto

style: Line

Y-block Right Axis

Right axis vars: 0

min: -10 ☑ auto

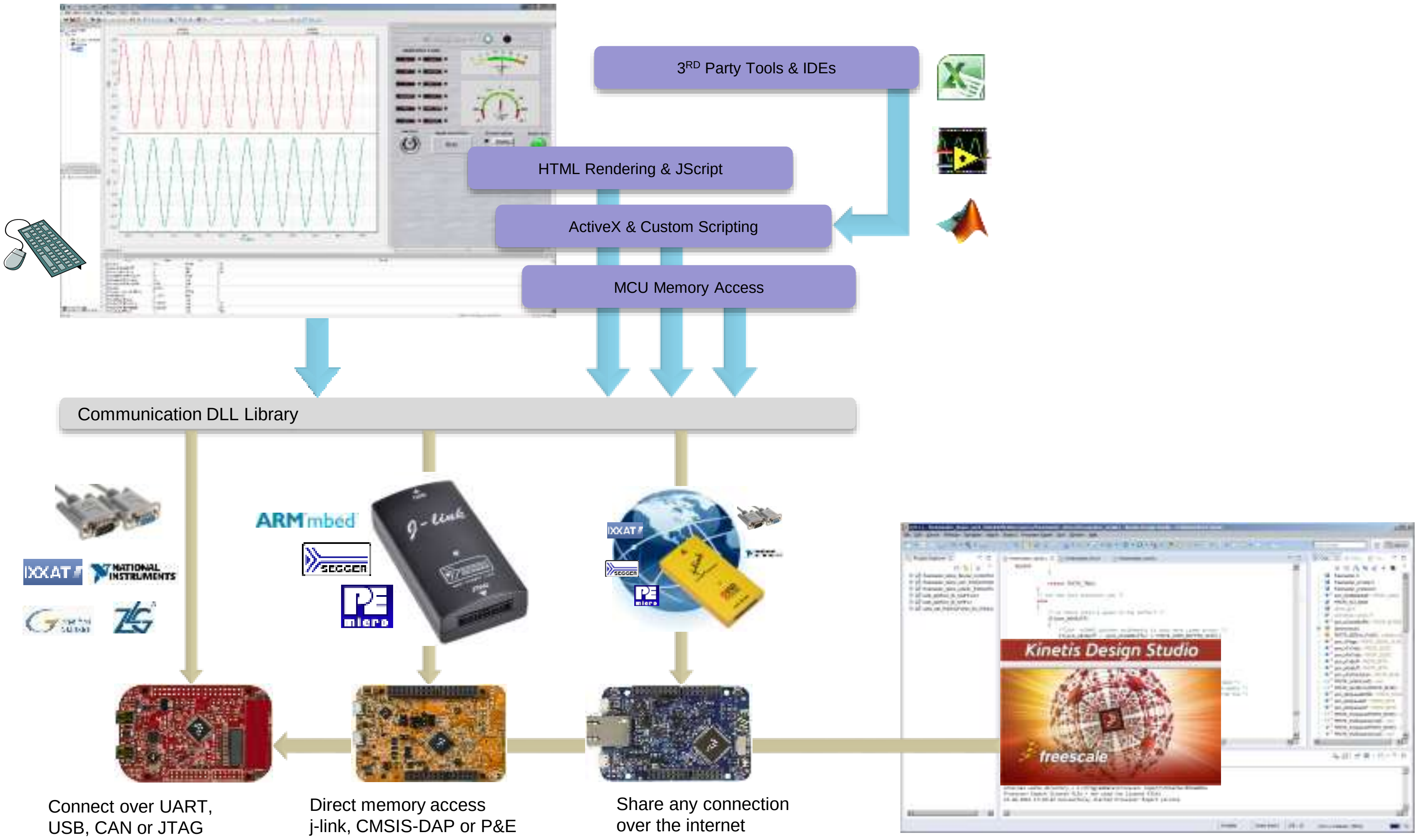
max: 10 ☑ auto

style: Line

Left axis label: lAxis

Right axis label: rAxis

Thank you - Questions?



3RD Party Tools & IDEs

HTML Rendering & JScript

ActiveX & Custom Scripting

MCU Memory Access

Communication DLL Library

Connect over UART, USB, CAN or JTAG

Direct memory access j-link, CMSIS-DAP or P&E

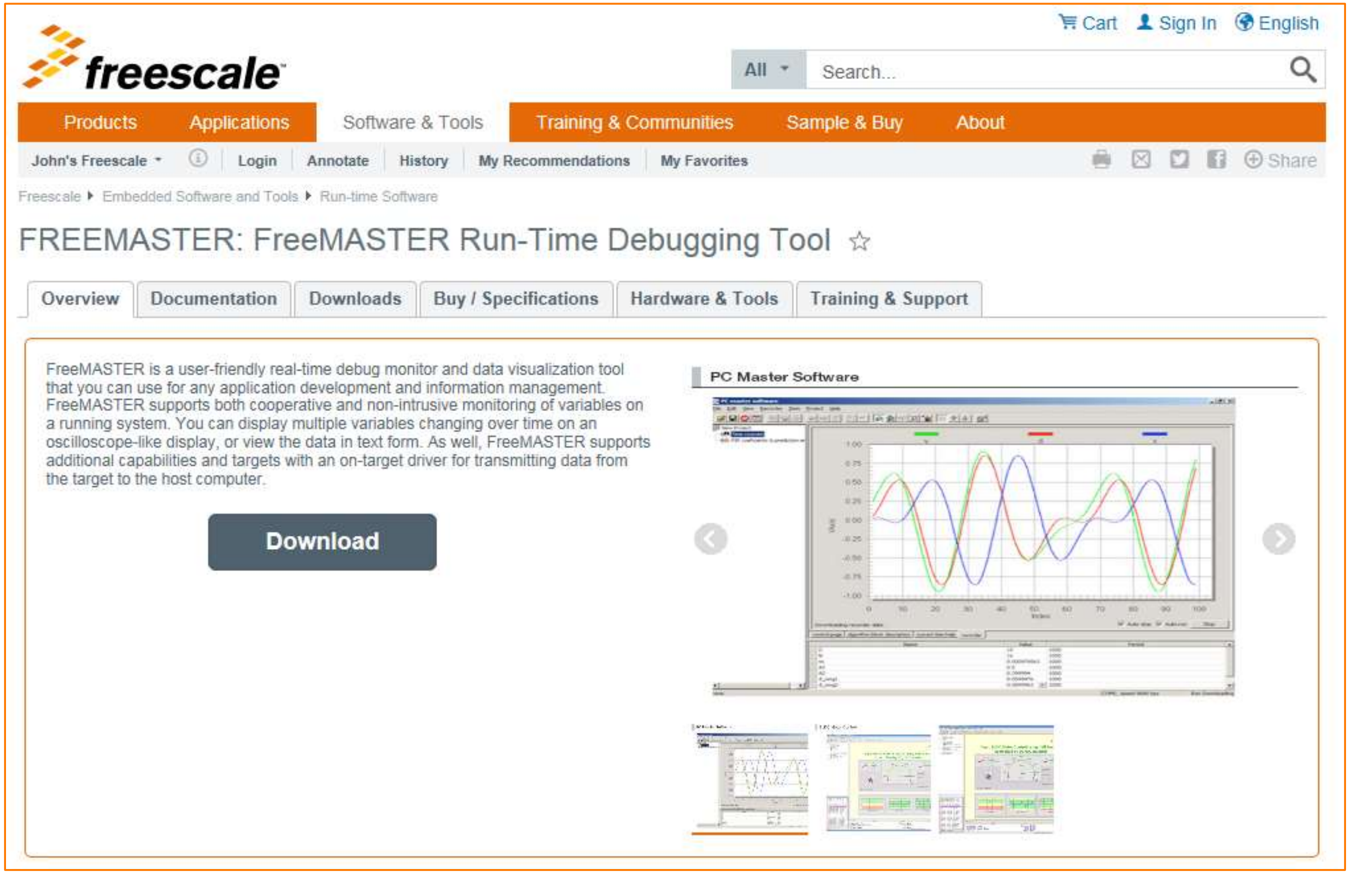
Share any connection over the internet

Any Questions?

Summary: More Information

For FreeMASTER, visit the Freescale website:

www.freescale.com/Freemaster



FREEMASTER: FreeMASTER Run-Time Debugging Tool

FreeMASTER is a user-friendly real-time debug monitor and data visualization tool that you can use for any application development and information management. FreeMASTER supports both cooperative and non-intrusive monitoring of variables on a running system. You can display multiple variables changing over time on an oscilloscope-like display, or view the data in text form. As well, FreeMASTER supports additional capabilities and targets with an on-target driver for transmitting data from the target to the host computer.

Summary

- Any Questions?
- Please Fill Out Your Surveys
- Thank you for your time

freescale™

www.Freescale.com