FREEMASTER QUICKSTART: DEVELOPMENT AND DEBUG MADE EASY

Mike Cao 曹学余
GC AUTOMOTIVE FAE
XUEYU.CAO@NXP.COM
+86 18616552690
AGENDA

• What is FreeMASTER
• How to get FreeMASTER
• FreeMASTER features
  – as a Real-Time Monitor
  – as a Control GUI
  – vs. a Debugger
• FREEMASTER Window Description
What is FreeMASTER?

FreeMASTER is a user-friendly **real-time debug monitor and data visualization** tool that can be used for application development and information management.

- Supports **non-intrusive monitoring of variables** on a running system.
- Display multiple variables changing over time on an **oscilloscope-like display**, or view **data in text form**.
- Supports **additional capabilities** and targets **with an on-target driver** for transmitting data from the target to the host computer.

What do we do with FreeMASTER?

- **Connect**: to target MCU over UART, CAN, BDM, JTAG etc
- **Monitor**: read & show variables in run-time
- **Control**: set variables, send commands
- **Share**: enable Excel, Matlab or a script engine to add hardware to the control loop
What is **Freemaster**

**Application control and monitor**

Live graphs, variable watches, and graphical control page

**Real-time operation monitor**
What is FreeMASTER - Connection options

- Supported Devices
  - S08
  - DSC
  - ARM Cortex-M (Kinetis/S32)
  - S12/S12X/S12Z (MagniV),
  - MPC56xx, MPC57xx
  - ColdFire V1/V2

- Supported Interfaces
  - BDM
  - JTAG (Segger, PE, CMSIS DAP, etc)
  - Serial
  - CAN
  - LIN
  - USB
What is **FREEmASTER** - Supported Devices & Interfaces

<table>
<thead>
<tr>
<th>MCU Families</th>
<th>No Driver</th>
<th>Target Driver Required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BDM /JTAG</td>
<td>Packet-Driven BDM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Serial</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CAN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LIN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>USB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MQX IO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>eOnce /JTAG</td>
</tr>
<tr>
<td>S12 Magni® Mixed Signal, S12 and S12X MCUs</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>S32 MCUs based on ARM Cortex-M</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MPC86xx MCUs based on Power Architecture</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MPC57xx based on Power Architecture</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Kinetis MCUs based on ARM Cortex-M</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>STM32 MCUs</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>DSC</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ColdFire MCUs</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

**NOTE:**
If it is desired to run the debugger and **FREEmASTER** concurrently, the target driver option is required!

**System Requirements**
- Host side operating system: Windows XP to through Windows 10 32/64bit
- Required software: Internet Explorer 8 or higher installed beforehand.
- Hard drive space: 50 MB
- Other hardware requirements: Serial RS-232 port for local control or USB-to Serial converter.
What is **FREEMASTER** - Features and Usage

**Real Time Monitor**
- Watching on-board variables or memory locations in various formats
- Text (name, value, min, max, enumerated labels...)
- Real-time waveform (real-time oscilloscope)
- High-speed recorded data (on-board memory oscilloscope)
- User-defined dashboard for data visualization

**Control Panel**
- Direct setting of the variable value from the variable watch
- Time-table stimulation of the variable value
- User command/message control
- Visual Basic script or JScript-powered HTML Forms (with push buttons, indicators and sliders) or custom HTML5 gauges
- By external application like Excel, Matlab or other which support ActiveX embedding

**Demonstration Platform**
- You can both describe and demonstrate your embedded application by HTML pages that contain pictures, sounds, video sequences, links or any web content
- Display simultaneous real-time data monitoring
- Browse through the functional blocks of the embedded application

**Easy Project Deployment**
- Entire project saved to a single file
- All resources/files packed in the project file
- "Demo mode" with password protection available
- New in v2.0: Project files embedded in target MCU Flash memory
What is **FreeMASTER** - FreeMASTER vs. IDE/Debugger

- Write source code
- Compile
- Flash code to MCU
- Debug code
- Logging data to file
- Graphs & Visualization
- Control Pane
- Tune parameters
- Remote control
- Plugins, custom communications, scripting

**GHS, DIAB, CW10x, S32DS...**

**limited functionality**

**FreeMASTER**
Tuning application constants with help of Freemaaster

- The most challenging task for the developer is the setting of the application constants, sometimes trial-error method must be used when the system (drive) parameters are difficult to identify:
  - $P$ and $I$ constants of the regulators
  - Filter constants
  - Constants of the position estimation algorithms
  - Tuning the merging process when switching from the open loop start-up to full sensorless mode
FreeMASTER Highlights

• FreeMASTER helps developers to debug or tune their applications

• Replaces debugger in situations when the processor core can not be simply stopped (e.g. motor control)

• Recorder may be used to visualize transitions in near 10-us resolution

• No EXTRA code is required on the embedded side to interface to FreeMASTER via BDM/OpenSDA/OSJTAG plug in modules.
How to get FreeMASTER

FreeMASTER is a FREE download from www.nxp.com/freemaster
01.
FreeMASTER Overview
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.
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**
FreeMASTER as a Real-Time Monitor

- **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

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

Variable Stimulus
- Stimulator tables to run
FreeMASTER as a Real-Time Monitor

- Establish a Data Trace on Target
  - Set up buffer (up to 64KB), sampling rate and trigger
FreeMASTER as a Real-Time Monitor

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

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

Steps to integrate FreeMASTER in your Application

• Include the files under the FreeMASTER Serial Communication Vxx\src_common in your application code project with no changes.

• One file changed in FreeMASTER Serial Communication V1.6\src_platforms\MPC56xx directory:
  - renamed freemaster_cfg.h.example to freemaster_cgh.h
  - Configure freemaster by changing macro definitions

• Addition to main.c
  - Add function call FMSTR_Init() after system init
  - Add function call FMSTR_Poll(); in main loop

• 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
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).
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
FreeMASTER as a Control GUI

• Target-in-loop Simulations
  - FreeMASTER invisible ActiveX object is accessible also by external standalone applications
  - Standard C++ or VB applications
  - Excel & Visual Basic for Applications
  - Matlab, Simulink

• Target-in-loop Simulation
  - Matlab or Simulink engine lets embedded application to perform calculations
  - SFIO toolbox for direct HW-in-loop simulations and algorithm testing
FreeMASTER Replacing Custom GUI Applications

FreeMASTER instead of Custom GUIs

- Comparing FreeMASTER with custom GUI approach
- Typical use cases
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
FreeMASTER Internal Application Structure

Basic FreeMASTER Communication Diagram

FreeMASTER Executable

---

Communication DLL

- RS232
- Plug-ins

---

Target Board

- SCI
- phy iface

---

Custom connection
(CAN, BDM, JTAG, ...)

---

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

FreeMASTER Executable

Internet Explorer running embedded in FM

ActiveX Interface

HTML page

JScript or VBScript

In-process calls

Out-of-process calls (exe-to-exe)

Communication DLL

RS232

Plug-ins

Custom connection (CAN, BDM, JTAG, ...)

Custom Plug-in DLL

Target Board

Target Board

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.

In-process calls

RS232

SCI or phy iface
FreeMASTER Internal Application Structure

Internet Explorer Running Separately (no difference)

FreeMASTER Executable

<table>
<thead>
<tr>
<th>ActiveX Interface</th>
<th>Internet Explorer running separately</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML page</td>
<td>JScript or VBScript</td>
</tr>
</tbody>
</table>

In-process calls

Communication DLL

<table>
<thead>
<tr>
<th>RS232</th>
<th>Plug-ins</th>
</tr>
</thead>
</table>

Out-of-process calls (exe-to-exe)

RS232

Custom Plug-in DLL

In-process calls

Custom connection (CAN, BDM, JTAG, ...)

SCI

or

phy.iface

Target Board

Target Board

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

- FreeMASTER Executable
  - In-process calls
  - Communication DLL
    - RS232
    - Plug-ins
    - Custom Plug-in DLL
    - SCI
    - phy iface
    - Target Board
    - or
    - Target Board

- Microsoft Excel
  - VB macros
  - Out-of-process calls (exe-to-exe)
  - Excel/VBA uses the same ActiveX interface like IE HTML/JScript as shown on previous slide.
02.
FREEMASTER Window Description
Application Windows

**Project Tree**
- Variable Stimulus / App Commands

**Variable Watch**
- Watch
- Control

**Detail View**
- Scope / Recorder
- Description / Control

---

**Welcome to FreeMASTER**

- What's New in Version 2.0?
  - A description of the most important changes and improvements implemented in the application version. Connection wizard, Active Content and graphics fetched from target memory, improved GUI and more.

- Visit FreeMASTER home page
  - Visit the application home page at www.freescale.com.

- Get MCU Communication Drivers
  - The latest version of FreeMASTER drivers for UART, Linux, FlexCAN, mcCAN and other interfaces is available for download at the official download page. The driver documentation is available here...

**Create New Project**

- Continue with this Project
  - Use the welcome message to start working with the empty project. You can disable the welcome message for subsequent start-ups in the Option Settings dialog.

- Open an Existing Project...
  - Browse your disk folders and find the project file you want to open in the FreeMASTER.

---

**FreeMASTER**

- [NXP Semiconductors](http://www.nxp.com)
Variable Watch

Variable Transformations
- Variable value can be transformed to a custom unit
- Variable transformations may reference other variable values
- Values are transformed back when writing a new value to the variable

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

Variable update rate
- Choose rate of update based on type of variable
- Rate selectable from 100ms to 10s or as fast as possible (limited by communication interface)
# Explanation of Variables

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Variable Update Rate</th>
<th>Display Format</th>
<th>Display Value and optionally display min/max values</th>
<th>Bit Field Manipulation</th>
<th>Enumerate values to text</th>
</tr>
</thead>
<tbody>
<tr>
<td>counter</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address/Type/Size and Format</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>counter</td>
<td></td>
<td>REAL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unsigned int</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fixed point number</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Variable Name**: The name of the variable being defined.
- **Variable Update Rate**: The rate at which the variable is updated.
- **Display Format**: The format in which the variable value is displayed.
- **Display Value and optionally display min/max values**: Whether to display minimum and maximum values along with the current value.
- **Bit Field Manipulation**: Options for manipulating bit fields.
- **Enumerate values to text**: Option to enumerate variable values to text.
Explanation of Variables

- **Variable Protection**
  - Define when variable is actually written
  - Limit variable modification as Min/Max

- **Limit variable by pre-defined values**
- **Define appearance of variable editing**
- **Write**
  - Write immediately after value changes
  - Write after ENTER or cell-lose focus
  - Finish edit mode after ENTER

- **Limit values**
  - Min (optional):
  - Max (optional):

- **Edit style**
  - Simple edit box
  - Edit box with spin control (change by step)
  - Combo box with pre-defined values
  - Hide edit interface at inactive cells

- **Pre-defined values**
  - All numbers from min to max (by step)
    - Stop:
  - Text enumeration (see transformation setup)
  - Other:
    - (sep. ;)
Variable Watch Customization

- Variable Color and Size
- Variable Stimulus
- Display Order
Multiple Scope or Recorders can be configured

- Select between different Scope views
- Can do time graphing or X-Y graphing
- Scope can display multiple variables and/or multiple axis
- Can set up a left and right y-axis on same graph
NXP Automotive FAE interface for TW customer

Mike Cao  曹学余
Xueyu.cao@nxp.com
Control&Network, Automotive Customer Application Solution and Support
NXP (China) Management Co., Ltd
Mobile: +86 186 1655 2690
Tel: +86 21 2205 2745
地址: 中国上海市裕通路100号宝矿洲际商务中心20层  邮编: 200070
Add: 20F, BM InterContinental Business Center, 100 Yu Tong Road, Shanghai, P.R.C 200070