FreeMASTER & S32 Design Studio

Iulian Stan
Software Engineer, AP System Tools
APRIL 2020
FreeMASTER 3.0

• Communication library & plug-ins
• Windows native charts, table views
• Integrated IE 11 and Chromium
• Light weight service with JSON-RPC API

Embedded application
FreeMASTER Communication Driver (minimal changes to application)

- UART / USB-CDC
  Plain serial line or on board USB port
- USB to CAN
- USB to CAN converter
- JTAG / SWD
  Debug probes
NXP TOOLS ECOSYSTEM

DEVELOP

Mode Based Design Toolbox

Code Generation From Model Based Design Toolbox

MCUXpresso / S32 Design Studio

Generate Simple Models Based on SDK

DEPLOY

EDGES / AP Evaluation Kits

S32K144

General Purpose Arm Cortex-M based MCUs

VISUALIZE

FreeMASTER

Load FreeMASTER project for demo, fine tuning or validation of the embedded application
**NXP TOOLS ECOSYSTEM (AUTOMOTIVE PROCESSING)**

**DEVELOP**

- Mode Based Design Toolbox
  - Code Generation
  - From Model Based Design Toolbox

- MCUXpresso / S32 Design Studio
  - Generate Simple Models
  - Based on SDK

**DEPLOY**

- EDGE / AP Evaluation Kits
  - S32K144
  - General Purpose Arm Cortex-M based MCUs

**VISUALIZE**

- FreeMASTER
  - Load FreeMASTER project for demo, fine tuning or validation of the embedded application
1. Create S32DS Project from Examples
   ✓ Ready to build C application
   ✓ FreeMASTER project included

2. Add FreeMASTER Driver to an Existing Project via SDK Configuration
   ✓ Automatically adds driver source files and configures target project structure
   ⚠ Requires manual communication & library initialization
CREATE FREEMASTER PROJECT FROM EXAMPLE

STEP 1

File → New → S32DS Project from Example

Filter: freemaster
CREATE FREEMASTER PROJECT FROM EXAMPLE

STEP 2

Open: freemaster_s32k144.pmp

FreeMASTER is launched automatically based on file extension
ADDING FREEMASTER SDK TO EXISTING PROJECT

**STEP 1**

Project→Properties

Select: FreeMASTER_S32xx→Press: Attach/Detach
**Freemaster Driver Implementation & Run-Time**

### Run-Time Phases

- **Communication**
  - Sends and receives commands from host PC

- **Execution**
  - Decodes and executes received commands

### Implementation

- **FMSTR_Isr**
  - Is assigned and called by a system interrupt

- **FMSTR_Poll**
  - Is called from user code

### Short interrupt

- **FMSTR_Isr**
  - Communication

- **FMSTR_Poll**
  - Execution

### Long interrupt

- **FMSTR_Isr**
  - Communication

- **FMSTR_Poll**
  - Execution

### Polling

- **FMSTR_Isr**
  - Communication

- **FMSTR_Poll**
  - Execution
ADDING FREEMASTER SDK TO EXISTING PROJECT

STEP2

1. #include "freemaster.h" ← include single header file

2. LPUART_DRV_Init(INST_LPUART1, &lpuart1_State, &lpuart1_InitConfig0);
   INT_SYS_InstallHandler(LPUART1_RxTx_IRQn, FMSTR_Isr, NULL); ← initialize communication interface and attach FreeMASTER handler (FMSTR_Isr)

3. FMSTR_Init(); ← initialize FreeMASTER driver (see freemaster_cfg.h)

4. FMSTR_Poll(); ← call polling function whenever target board is free to or should process FreeMASTER commands (in polling mode)
SECURE CONNECTIONS
FOR A SMARTER WORLD
Tips for Enhancing Embedded Applications with FreeMASTER UI from MATLAB/Simulink

Daniel Popa
MBDT Product Manager & Architect

APRIL 30, 2020
Contents

- Model-Based Design General Concepts
- Benefits of FreeMASTER Simulink Blocks
- Embedded Application Examples
  - Simulink Modelling and Automatic Code Generation for Applications with FreeMASTER Simulink Blocks
  - Data Logger in MATLAB with FreeMASTER ActiveX Controls
Model-Based Design
“at a glance”
Model-Based Design – Concept

**Model-Based Design (MBD)** is a mathematical and visual method of addressing problems associated with designing complex control, signal processing and communication systems. It is used in many motion control, industrial equipment, aerospace, and automotive applications.

- **RESEARCH**
  - Environmental Models
  - Physical Components
  - Algorithms

- **IMPLEMENTATION**
  - C, C++
  - VHDL, Verilog
  - Structured Text
  - MCU, DSP, FPGA, ASIC, PLC

- **TEST AND VERIFICATION**
How Is That Possible?

Step 1 – System Requirements:
Model-in-the-Loop
- Software requirements
- Control system requirements
- Overall application control strategy

Step 2 – Modeling/Simulation:
Software-in-the-Loop
- Control algorithm design
- Code generation preparation
- Control system design
- Start testing implementation approach

Step 3 – Rapid Prototype:
Processor-in-the-Loop
- Controller code generation
- Determine execution time on MCU
- Verify algorithm on MCU
- See memory/stack usage on MCU

Step 4 – Target MCU Implementation
MCU Final Application
- Validation/verification phase
- Controller code generation
- Test system in target environment using tools for data logging and parameter tuning

FreeMASTER
Collection of Tools & Libraries designed to **Assist** customers with prototyping and accelerate algorithm development on NXP MCUs

- MCU Peripherals **Initialization & Configuration** through GUI from a Model-Based Design environment like Simulink®

- Supported **Platforms** for automatic Code Generation:
  - Arm®-based S32K
  - E200®-based MPC57xx/MPC56xx
  - MagniV S12ZVMx/S12ZVC

- Customer **Support and Training**: [https://community.nxp.com/community/mbdt](https://community.nxp.com/community/mbdt)
Benefits of FreeMASTER Integration with MATLAB/Simulink

1. No need to write any c-code: just **drag&drop** FreeMASTER Simulink blocks anywhere in the Model

2. Real Time debugging of Simulink Models with minimal intrusiveness

3. Use FreeMASTER as “a bridge” to upload/download data from/to embedded MCU from MATLAB
Embedded Applications Examples
Example#1: Modelling & Verification with Simulink & FreeMASTER

**Simulink:**
- ✓ Modelling & Simulations
- ✓ FreeMASTER configuration
- ✓ Automatic Code Generation
- ✓ Automatic Target Deployment

**S32K144 EVB:**
- ✓ Embedded Application Execution

**FreeMASTER 3.0.2**
- ✓ Real-Time data inspection
- ✓ Variable Watch
- ✓ Oscilloscope monitoring
Example#2: MATLAB Data Logger with FreeMASTER

**S32K144 EVB:**
- ✓ Embedded Application Execution

**FreeMASTER 3.0.2**
- ✓ Real-Time data inspection
- ✓ Variable Watch
- ✓ Oscilloscope monitoring

**MATLAB:**
- ✓ Read/Write Data via ACTXSERVER
- ✓ Plot Data
- ✓ Manipulate Data
Bonus Example: Optimize Data Throughput

• Use Simulink FreeMASTER Configuration Block to optimize the data exchange throughput
  ✓ Change the communication protocol
  ✓ Change ISR priorities
GETTING HELP

MBDT Online Community
Examples & Help

MBDT home page
www.nxp.com/mbdt
SHARE YOUR FREEMASTER DASHBOARD DESIGNS WITH THE NXP COMMUNITY
GET A FREE BOARD!

Why?

To build a robust community of support for FreeMASTER with idea share.

How to participate?

1. Submit your idea through June 19, 2020 to the NXP Community, request your board of choice (one of the following: i.MX RT1020 EVK, LPC55S28 development board and S32K144EVB), available on first come, first served basis until quantities are depleted.

2. Once you’ve created your code example, post a brief description and a screenshot of your dashboard along with a ZIPped code to the original blog comment thread.

Click here for complete details!
HOW TO CONTROL AND VISUALIZE DATA FROM YOUR EMBEDDED APPLICATION WITH FREEMASTER | A FOUR-PART WEBINAR SERIES

• Part 1: Now Available On-Demand | Watch Now >
  Get to Know the Easy-to-Use FreeMASTER Runtime Debugging Tool – Now Part of MCUXpresso SDK

• Part 2: Today
  Tips for Enhancing Embedded Applications with FreeMASTER UI from Various Development Environments like S32DS and Matlab/Simulink

• Part 3: Tuesday, May 5 | 10 AM CDT | Register Here >>
  Introduction to FreeMASTER Dashboard Coding Using HTML, JavaScript, ActiveX and JSON-RPC

• Part 4: Tuesday, May 12 | 10 AM CDT | Register Here >>
  Getting Started with FreeMASTER Lite and JSON-RPC Protocol: From Scripting to Visual Dashboards with Python and JavaScript
SECURE CONNECTIONS FOR A SMARER WORLD