FreeMASTER & S32 Design Studio

Iulian Stan
Software Engineer, AP System Tools

APRIL 2020

SECURE CONNECTIONS FOR A SMARTER WORLD

EXTERNAL

EASY TO USE AND FLEXIBLE

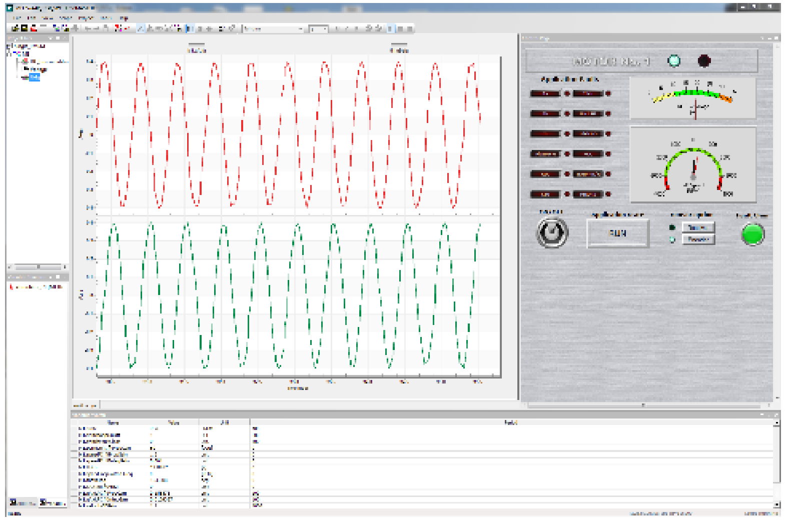
HOST PC

COMMUNICATION LAYER

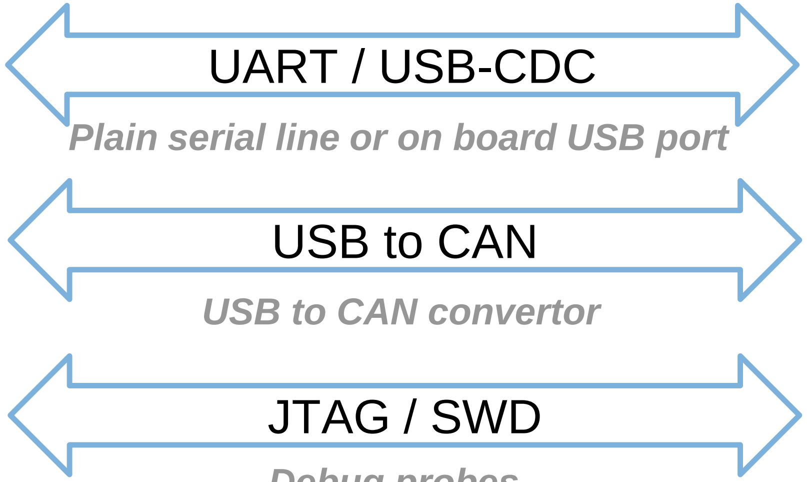
NXP OR CUSTOMER BOARD

FreeMASTER 3.0

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



IXXAT · 吉陽光電 GLINKER · ZLG · PEAK System · NATIONAL INSTRUMENTS · KVASER · ARM mbed · SEGGER · PE micro



UART / USB-CDC

Plain serial line or on board USB port

USB to CAN

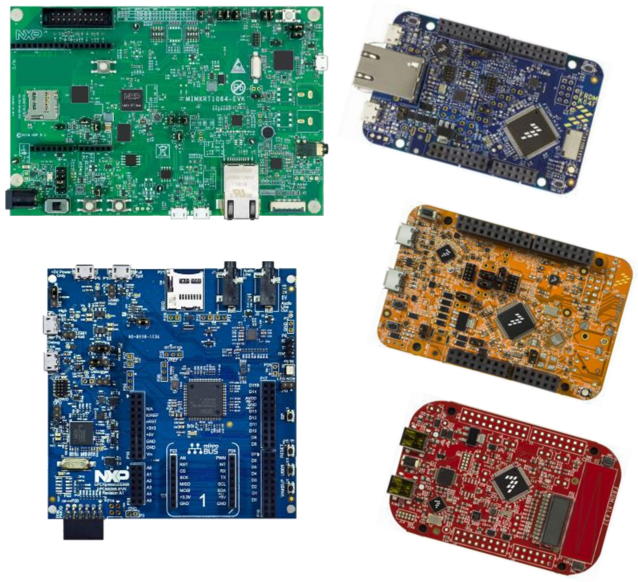
USB to CAN convertor

JTAG / SWD

Debug probes

Embedded application

FreeMASTER Communication Driver (minimal changes to application)



NXP TOOLS ECOSYSTEM

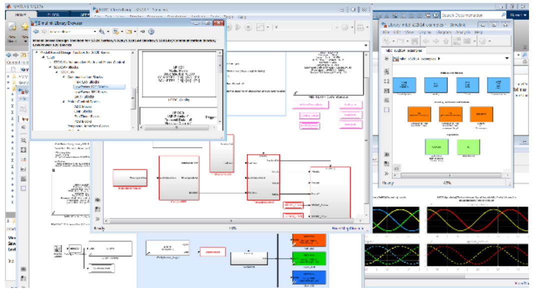
DEVELOP

DEPLOY

VISUALIZE

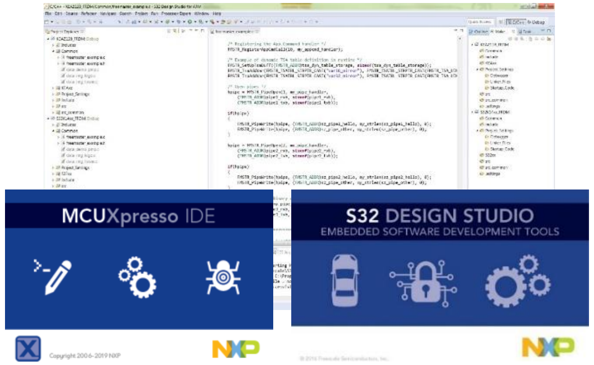
Mode Based Design Toolbox

Code Generation From Model Based Design Toolbox



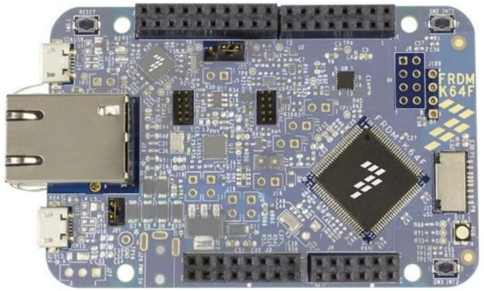
MCUXpresso / S32 Design Studio

Generate Simple Models Based on SDK



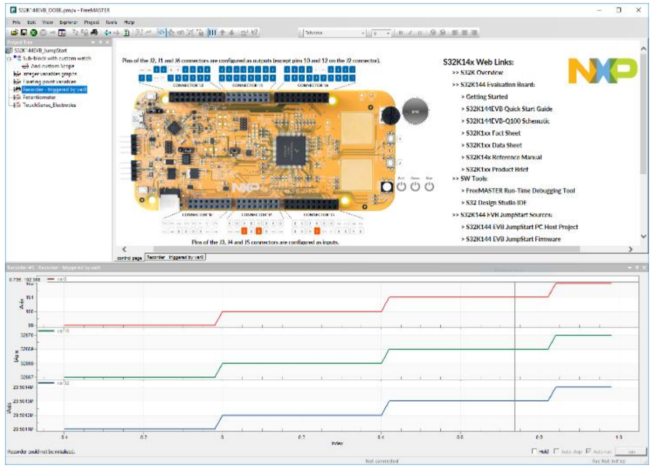
EDGE / AP Evaluation Kits

S32K144



General Purpose Arm Cortex-M based MCUs

FreeMASTER



Load FreeMASTER project for demo, fine tuning or validation of the embedded application

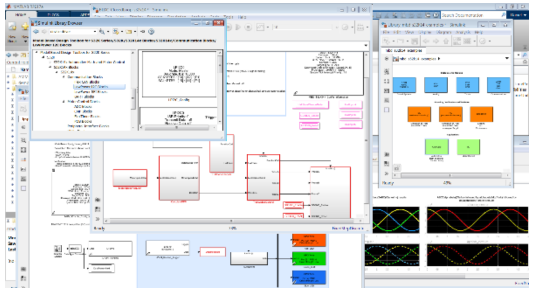
NXP TOOLS ECOSYSTEM (AUTOMOTIVE PROCESSING)

DEVELOP

DEPLOY

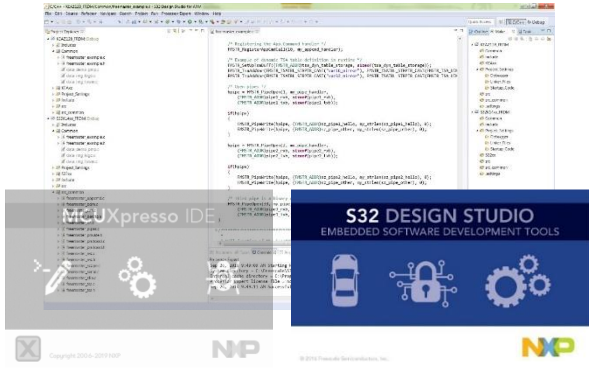
VISUALIZE

Mode Based Design Toolbox



Code Generation From Model Based Design Toolbox

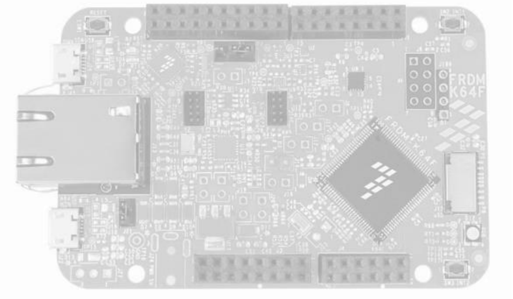
MCUXpresso / S32 Design Studio



Generate Simple Models Based on SDK

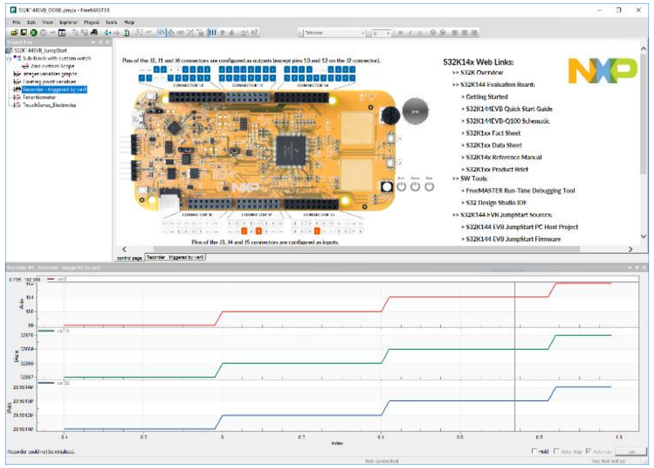
EDGE / AP Evaluation Kits

S32K144



General Purpose Arm Cortex-M based MCUs

FreeMASTER



Pins of the J2, J1 and J6 connectors are configured as outputs (except pins 10 and 12 on the J2 connector).

Pins of the J3, J4 and J5 connectors are configured as inputs.

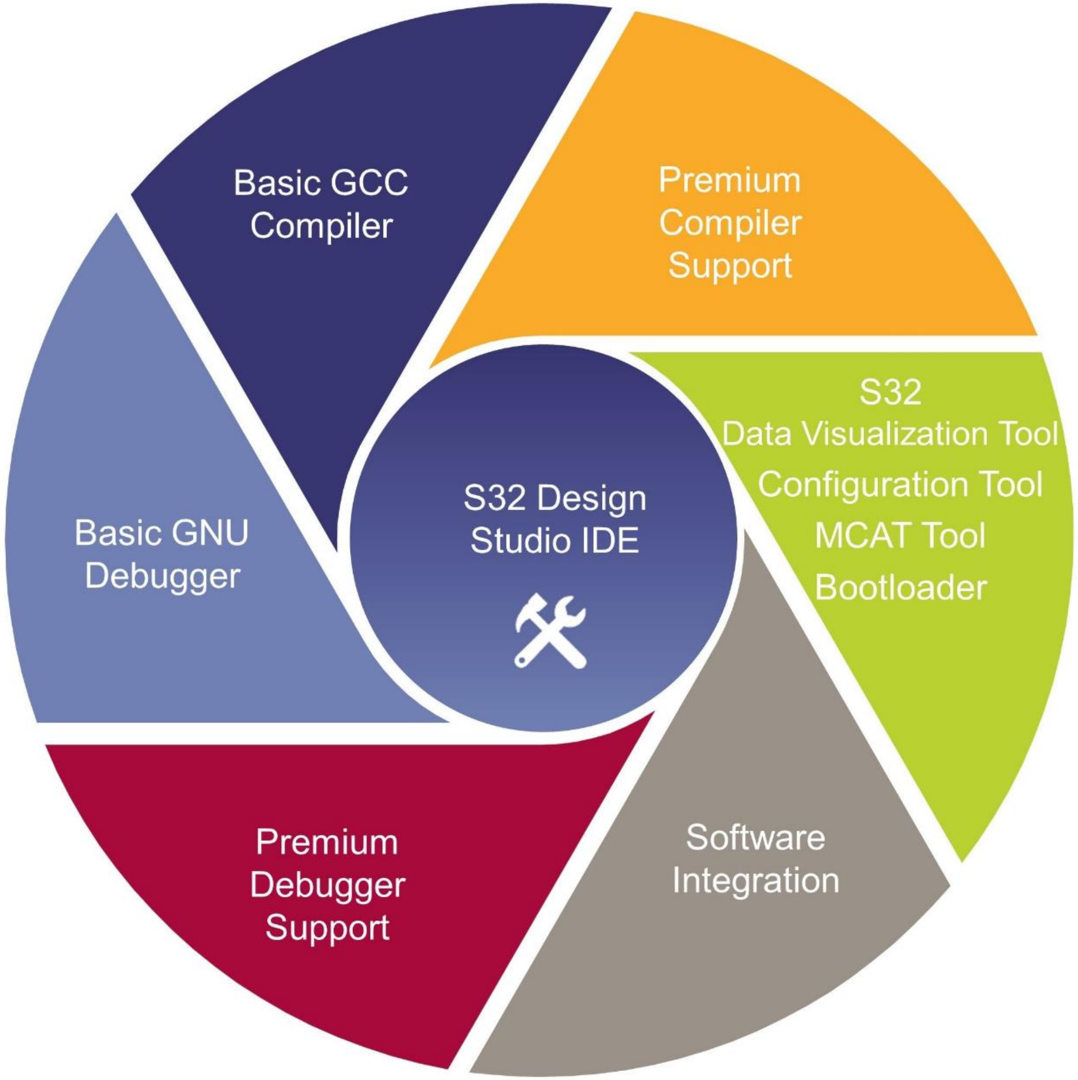
S32K14x Web Links:
>> S32K Overview
>> S32K144 Evaluation Board:
> Getting Started
> S32K144EVB Quick Start Guide
> S32K144EVB-Q100 Schematic
> S32K1xx Fact Sheet
> S32K1xx Data Sheet
> S32K14x Reference Manual
> S32K1xx Product Brief
>> SW Tools:
> FreeMASTER Run-Time Debugging Tool
> S32 Design Studio IDE
>> S32K144 EVB JumpStart Sources:
> S32K144 EVB JumpStart PC Host Project
> S32K144 EVB JumpStart Firmware

Load FreeMASTER project for demo, fine tuning or validation of the embedded application

FREEMASTER & S32DS INTEGRATION OPTIONS

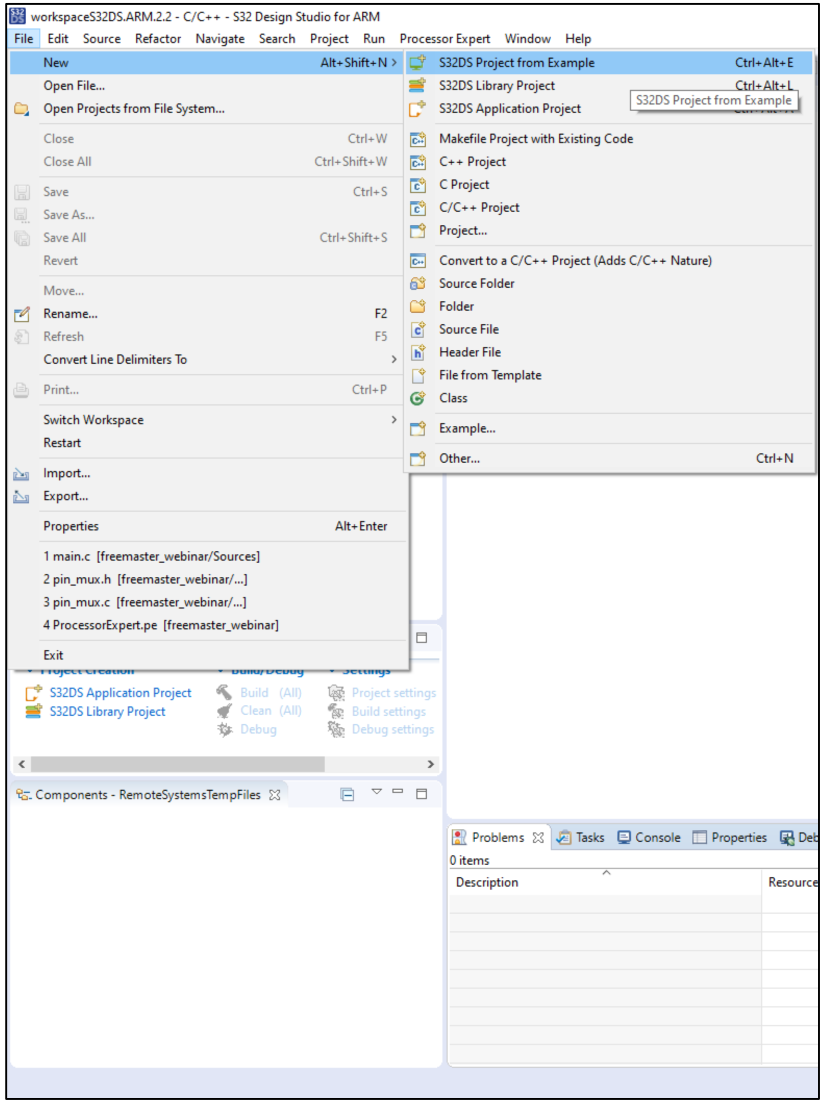
1. Create S32DS Project from Examples
   - ✓ Ready to build C application
   - ✓ FreeMASTER project included

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

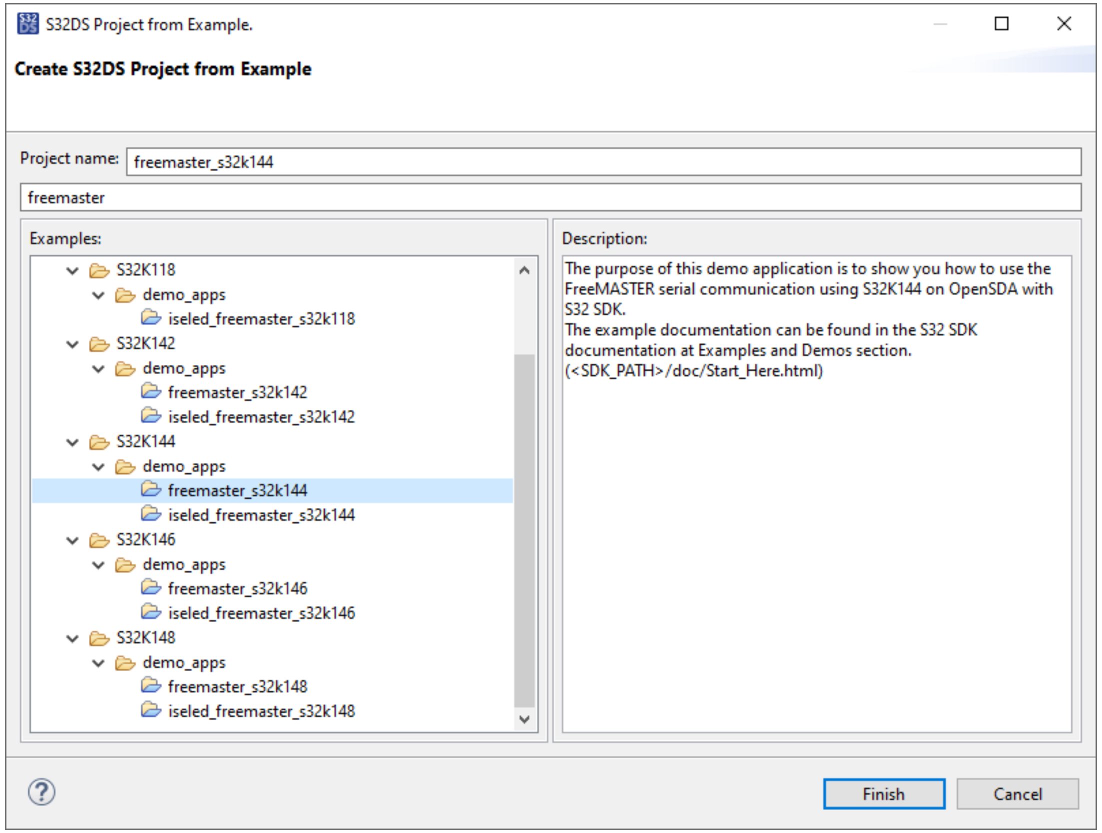


CREATE FREEMASTER PROJECT FROM EXAMPLE STEP1

File→New →S32DS Project from Example


workspaceS32DS.ARM.2.2 - C/C++ - S32 Design Studio for ARM
File Edit Source Refactor Navigate Search Project Run Processor Expert Window Help
New Alt+Shift+N >
Open File...
Open Projects from File System...
Close Ctrl+W
Close All Ctrl+Shift+W
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
Revert
Move...
Rename... F2
Refresh F5
Convert Line Delimiters To >
Print... Ctrl+P
Switch Workspace >
Restart
Import...
Export...
Properties Alt+Enter
1 main.c [freemaster_webinar/Sources]
2 pin_mux.h [freemaster_webinar/...]
3 pin_mux.c [freemaster_webinar/...]
4 ProcessorExpert.pe [freemaster_webinar]
Exit
S32DS Project from Example Ctrl+Alt+E
S32DS Library Project Ctrl+Alt+L
S32DS Application Project
S32DS Project from Example
Makefile Project with Existing Code
C++ Project
C Project
C/C++ Project
Project...
Convert to a C/C++ Project (Adds C/C++ Nature)
Source Folder
Folder
Source File
Header File
File from Template
Class
Example...
Other... Ctrl+N
S32DS Application Project
S32DS Library Project
Build (All)
Clean (All)
Debug
Project settings
Build settings
Debug settings
Components - RemoteSystemsTempFiles
Problems Tasks Console Properties Deb
0 items
Description
Resource

Filter: freemaster

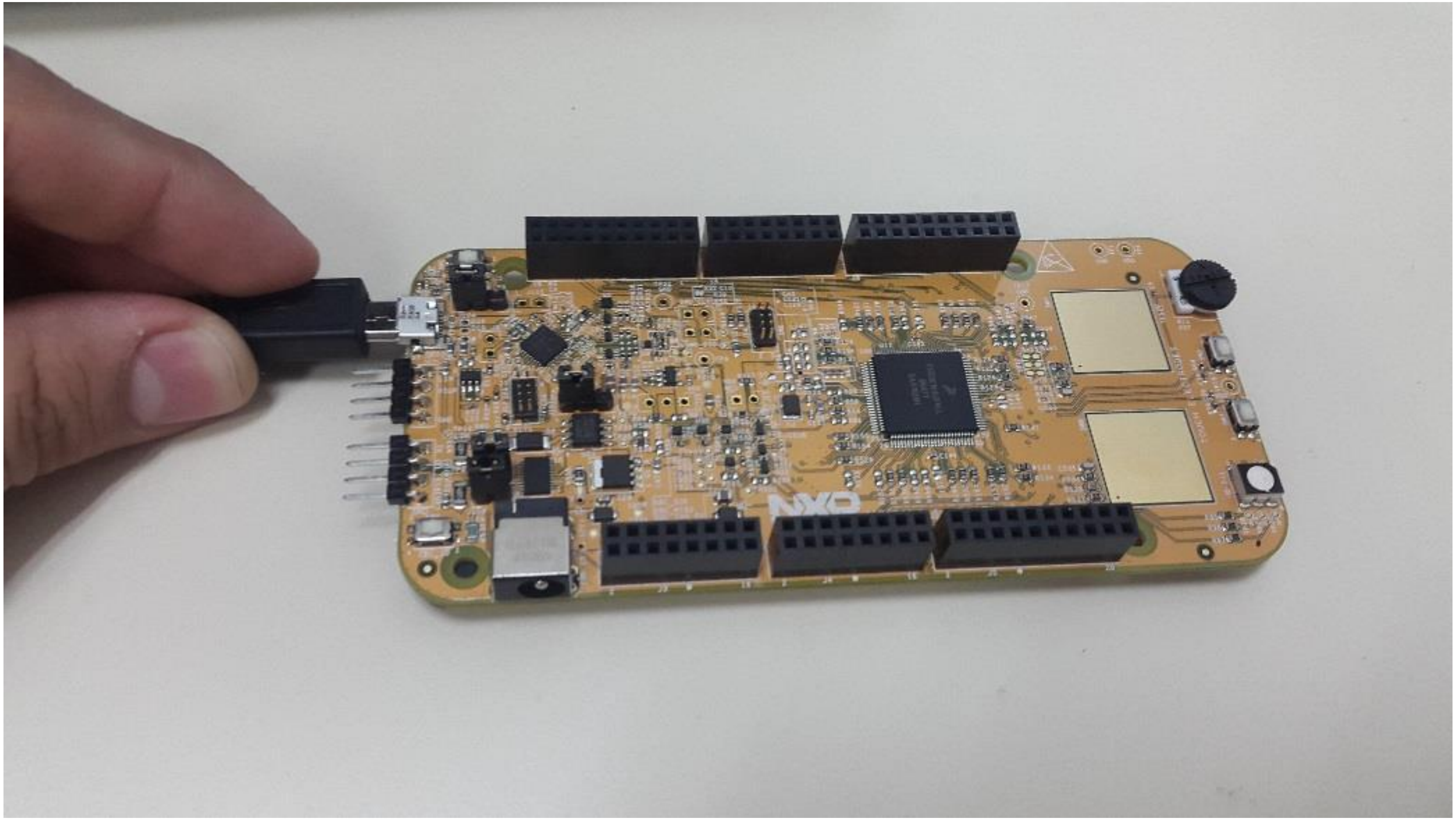

S32DS Project from Example.
Create S32DS Project from Example
Examples:
S32K118
demo_apps
iseled_freemaster_s32k118
S32K142
demo_apps
freemaster_s32k142
iseled_freemaster_s32k142
S32K144
demo_apps
freemaster_s32k144
iseled_freemaster_s32k144
S32K146
demo_apps
freemaster_s32k146
iseled_freemaster_s32k146
S32K148
demo_apps
freemaster_s32k148
iseled_freemaster_s32k148
Finish
Cancel

Project name: freemaster_s32k144

freemaster

Description:

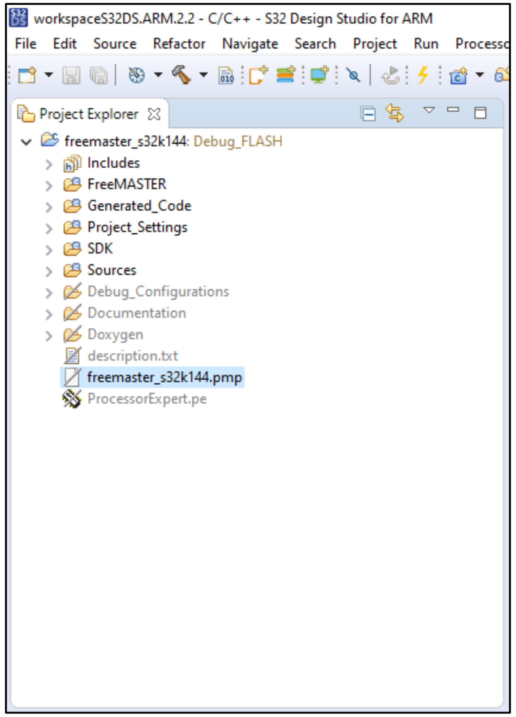
The purpose of this demo application is to show you how to use the FreeMASTER serial communication using S32K144 on OpenSDA with S32 SDK.
The example documentation can be found in the S32 SDK documentation at Examples and Demos section.
(<SDK_PATH>/doc/Start_Here.html)



CREATE FREEMASTER PROJECT FROM EXAMPLE STEP2

Open: freemaster_s32k144.pmp

FreeMASTER is launched automatically based on file extension

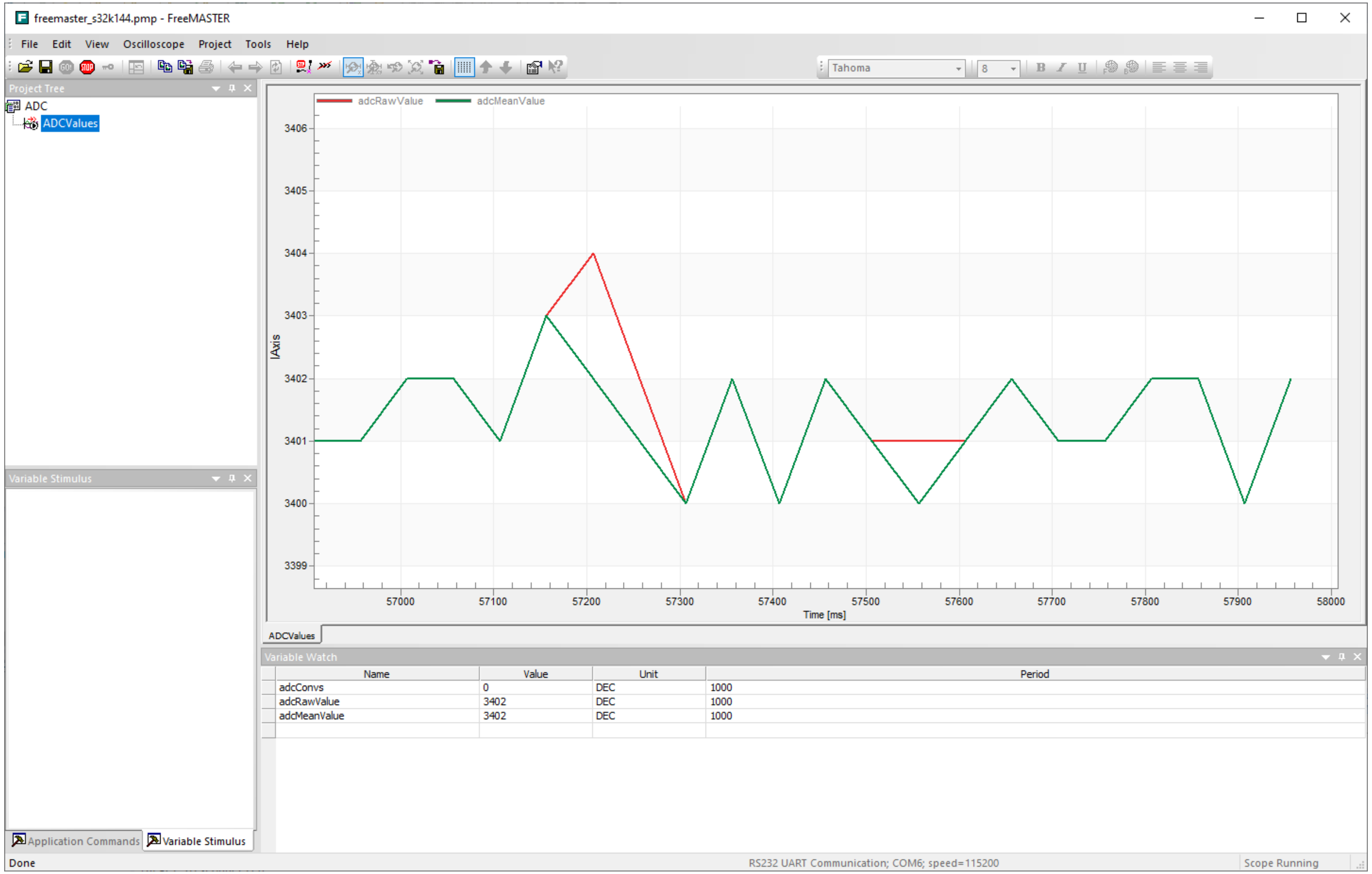


workspaceS32DS.ARM.2.2 - C/C++ - S32 Design Studio for ARM

File Edit Source Refactor Navigate Search Project Run Processo

Project Explorer

- freemaster_s32k144: Debug_FLASH
  - Includes
  - FreeMASTER
  - Generated_Code
  - Project_Settings
  - SDK
  - Sources
  - Debug_Configurations
  - Documentation
  - Doxygen
  - description.txt
  - freemaster_s32k144.pmp
  - ProcessorExpert.pe



freemaster_s32k144.pmp - FreeMASTER

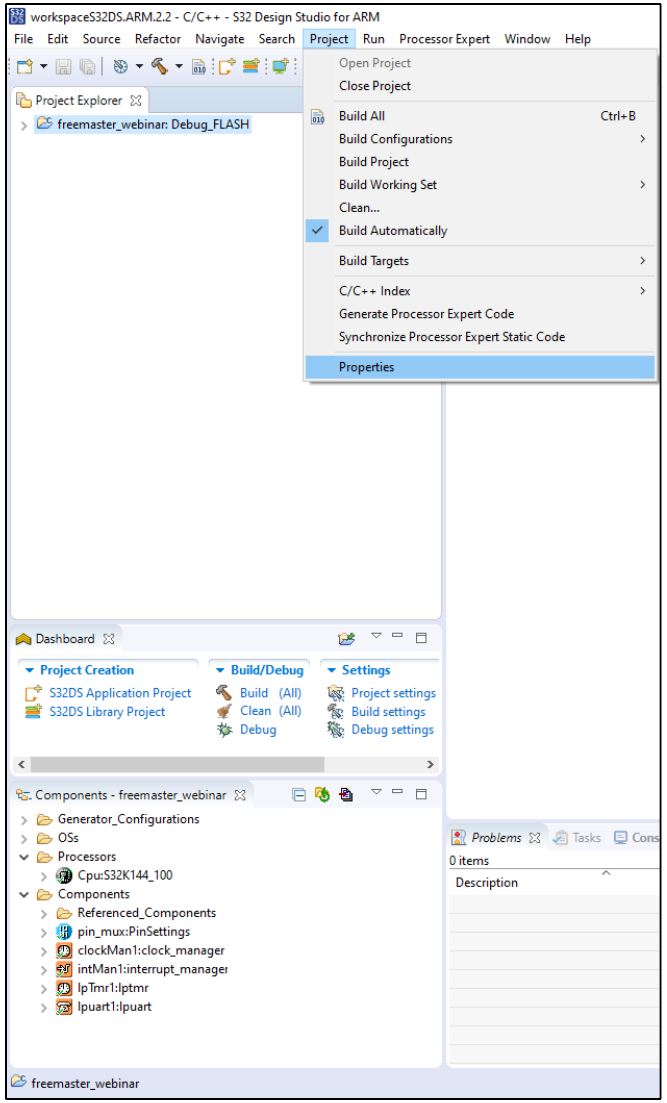
Variable Watch

Name	Value	Unit	Period
adcConvs	0	DEC	1000
adcRawValue	3402	DEC	1000
adcMeanValue	3402	DEC	1000

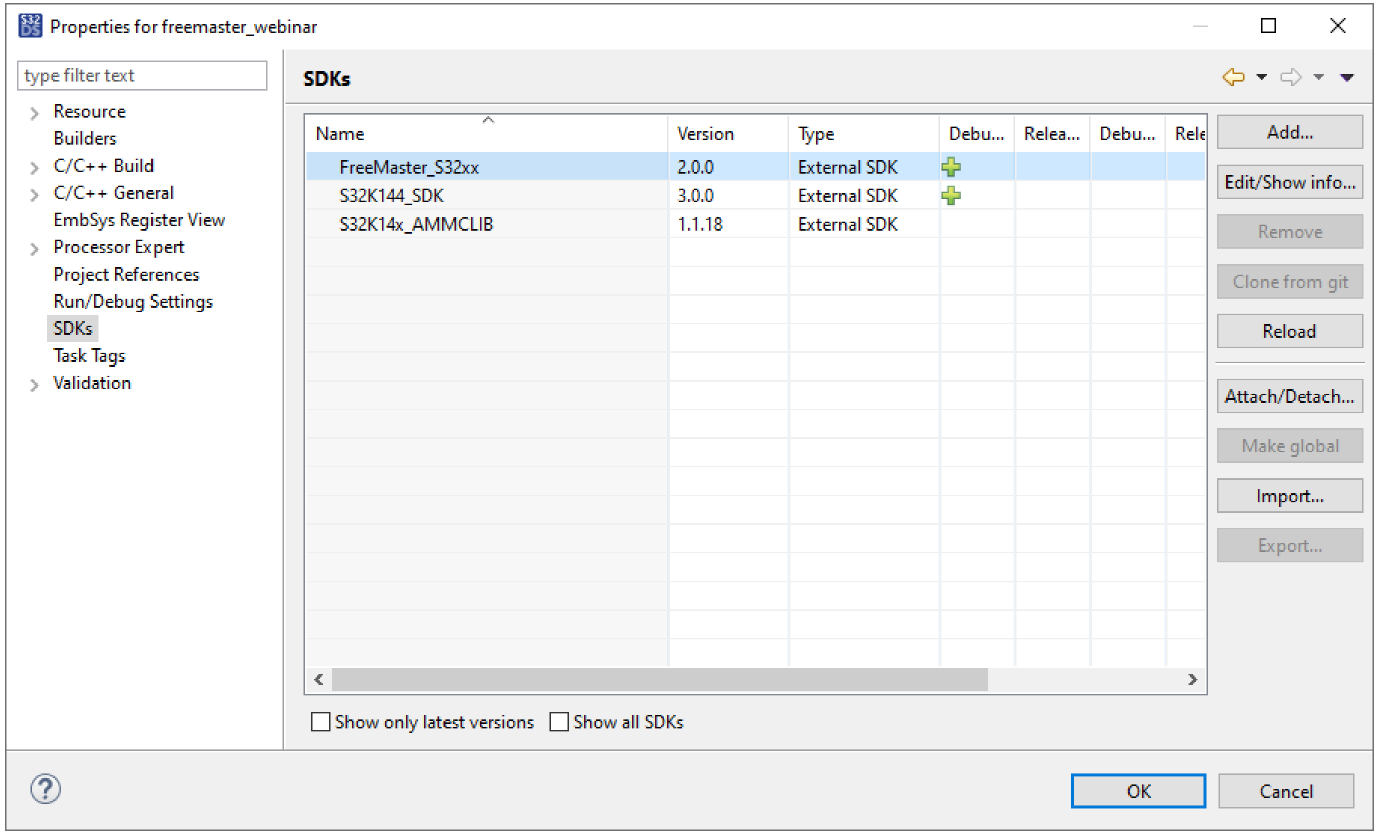
Done   RS232 UART Communication; COM6; speed=115200   Scope Running

ADDING FREEMASTER SDK TO EXISTING PROJECT STEP1

Project→Properties



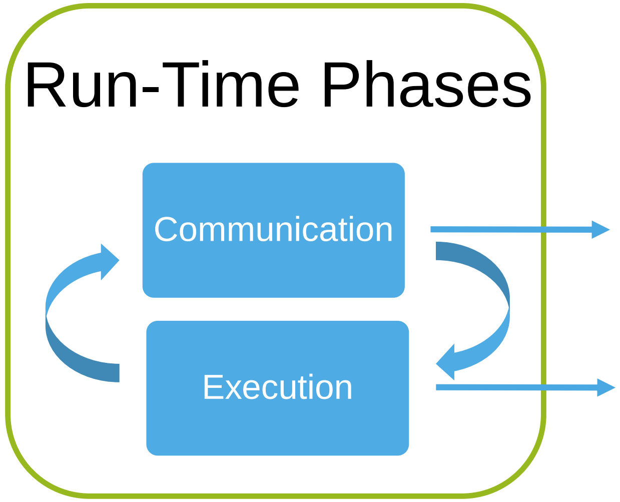
Select: FreeMASTER_S32xx→Press: Attach/Detach



Name	Version	Type	Debu...	Relea...	Debu...	Rele
FreeMaster_S32xx	2.0.0	External SDK	✚			
S32K144_SDK	3.0.0	External SDK	✚			
S32K14x_AMMCLIB	1.1.18	External SDK				

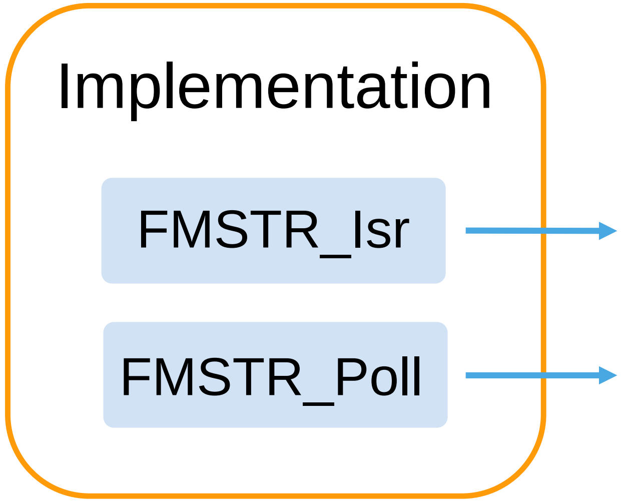
☐ Show only latest versions ☐ Show all SDKs

FREEMASTER DRIVER IMPLEMENTATION & RUN-TIME



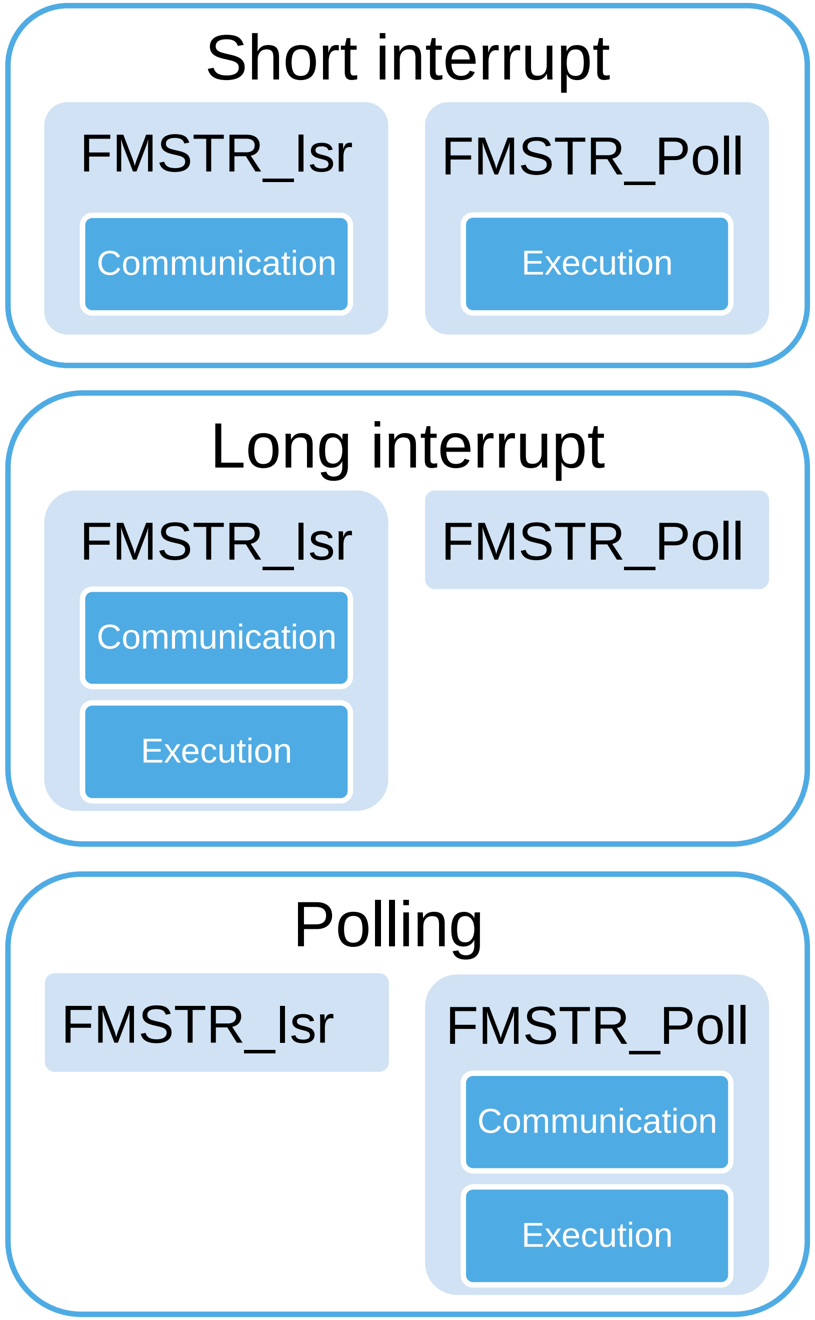
Communication → Sends and receives commands from host PC

Execution → Decodes and executes received commands



FMSTR_Isr → Is assigned and called by a system interrupt

FMSTR_Poll → Is called from user code



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

MBDT

Tips for Enhancing Embedded Applications with FreeMASTER UI from MATLAB/Simulink

Daniel Popa
MBDT Product Manager & Architect

APRIL 30,2020

SECURE CONNECTIONS
FOR A SMARTER WORLD

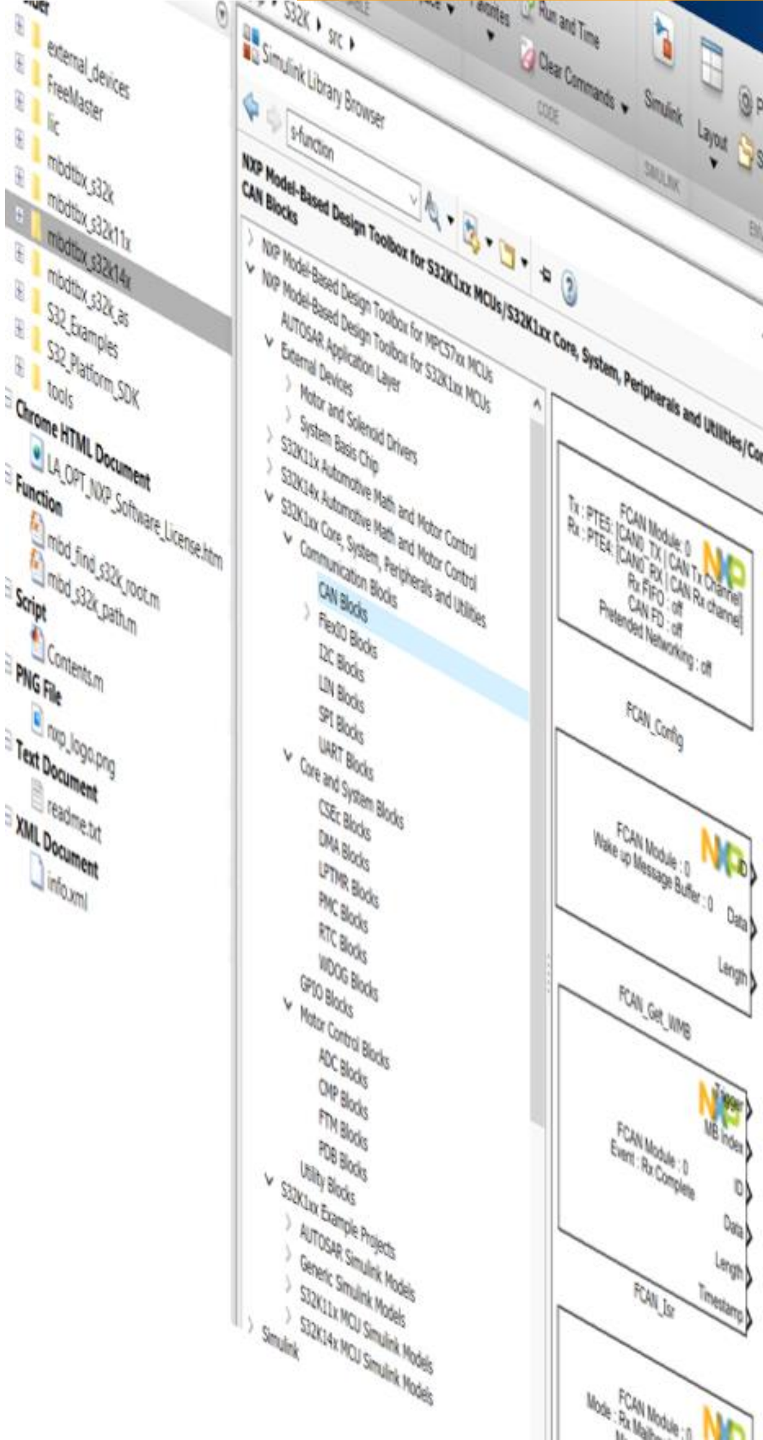
EXTERNAL

File Edit View Display Diagram Analysis Help
mbd_s32k14_examples
S32K On-chip Modules
PORT GPIO TRGMUX
Core & Systems
ADC CMP
Analog
PDB FTM LPIT
Timers
Modelling, Verification and Validation
SIL (software-in-the-loop)
Simulation on Host with ANSI C Code
PIL (processor-in-the-loop)
Simulation on Host with ANSI C Code running on Target
UTILITI
Logging and Performan
Applications
AMMCLIB
Automotive Math and Motor Control Library
MC
Motor Control
Ready
62%
mc_gmclib0_s32k14
FixedStepD
51%
FixedStepDiscre

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

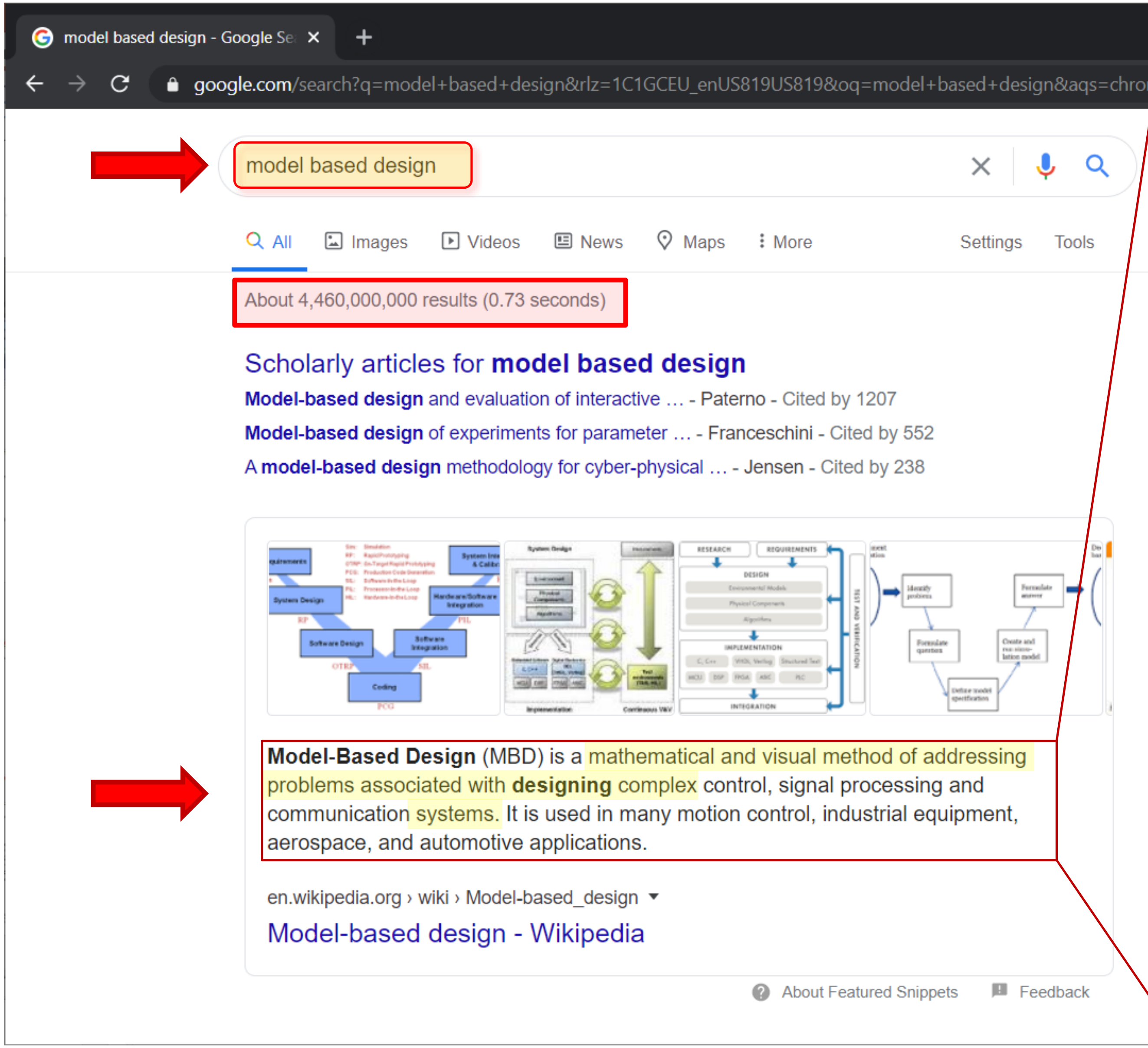
Model-Based Design “at a glance”



SECURE CONNECTIONS
FOR A SMARTER WORLD

EXTERNAL

Model-Based Design – Concept



model based design - Google Se
google.com/search?q=model+based+design&rlz=1C1GCEU_enUS819US819&oq=model+based+design&aqs=chrome.0.69i59j69i61j69i60j69...
Incognito

model based design

All
Images
Videos
News
Maps
More
Settings
Tools

About 4,460,000,000 results (0.73 seconds)

Scholarly articles for **model based design**

Model-based design and evaluation of interactive … - Paterno - Cited by 1207

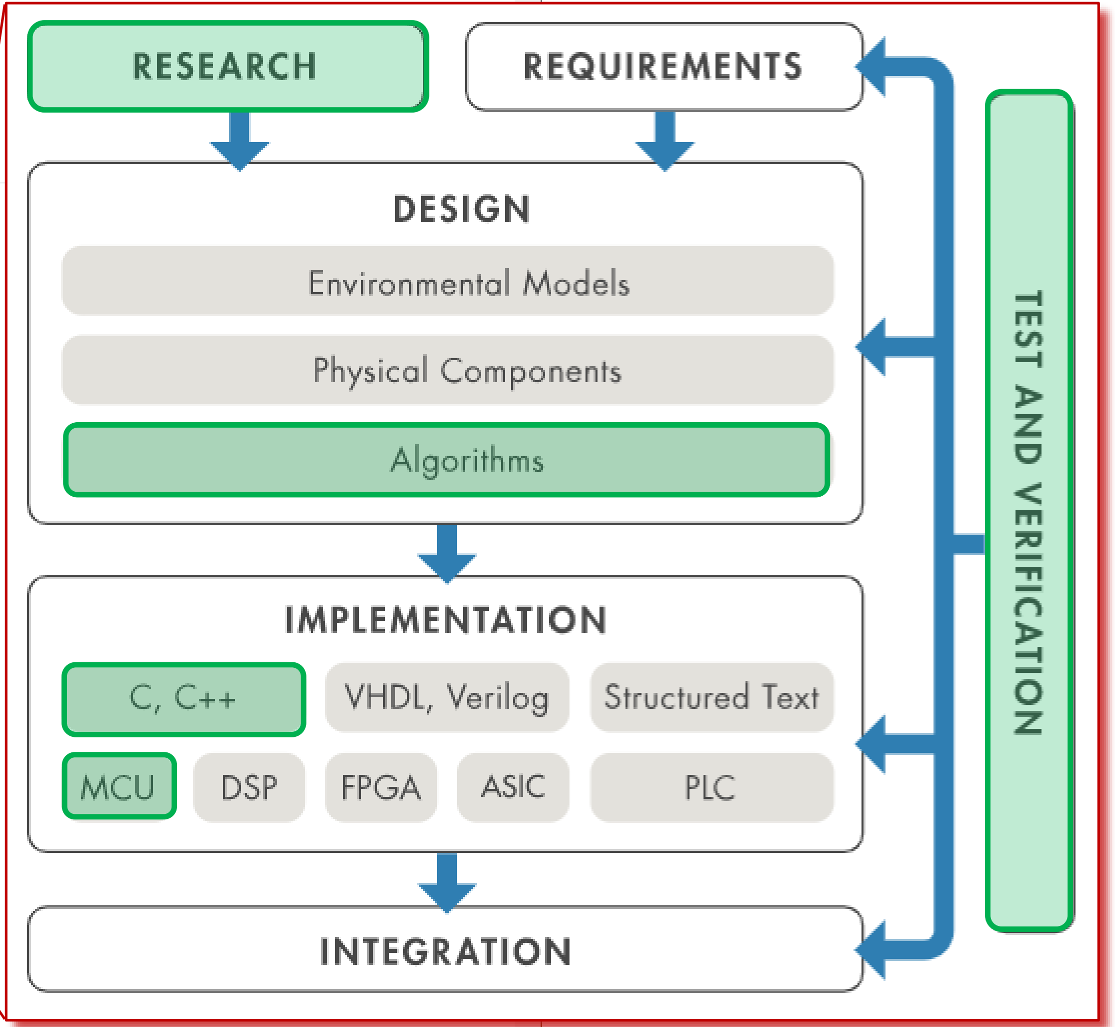
Model-based design of experiments for parameter … - Franceschini - Cited by 552

A **model-based design** methodology for cyber-physical … - Jensen - Cited by 238

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.

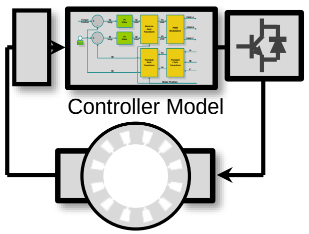
en.wikipedia.org › wiki › Model-based_design

Model-based design - Wikipedia

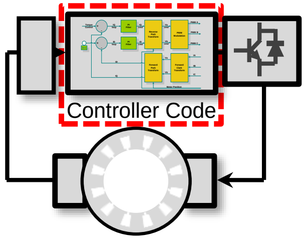


How Is That Possible ?

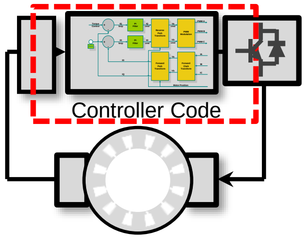
Idea incubation → To SIL → **Automatic Code Generation** → To PIL → **Code Validation** → To MCU → **Final Product**

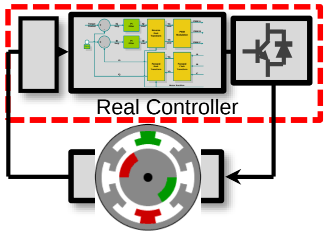


PC Environment



PC Environment





PC Environment + MCU

MCU with Embedded Control Module (ECM)

FreeMASTER

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

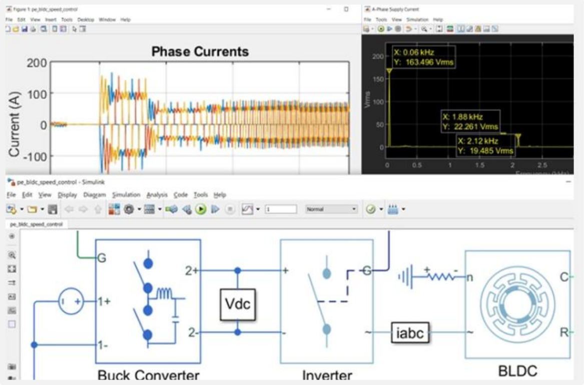
- Controller code generation
- Determine execution time on MCU
- Verify algorithm on MCU
- See memory/stack usage on MCU

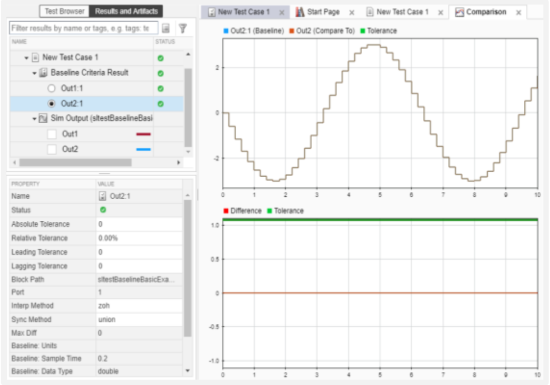
Step 4 – Target MCU Implementation

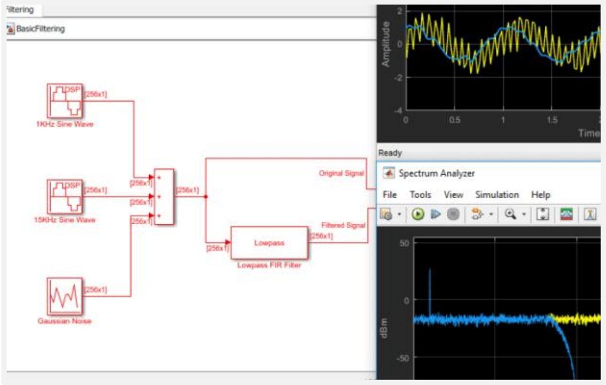
MCU Final Application

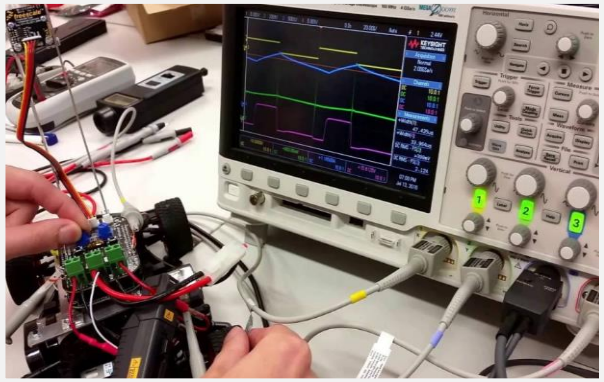
Validation/verification phase

- Controller code generation
- Test system in target environment using tools for data logging and parameter tuning









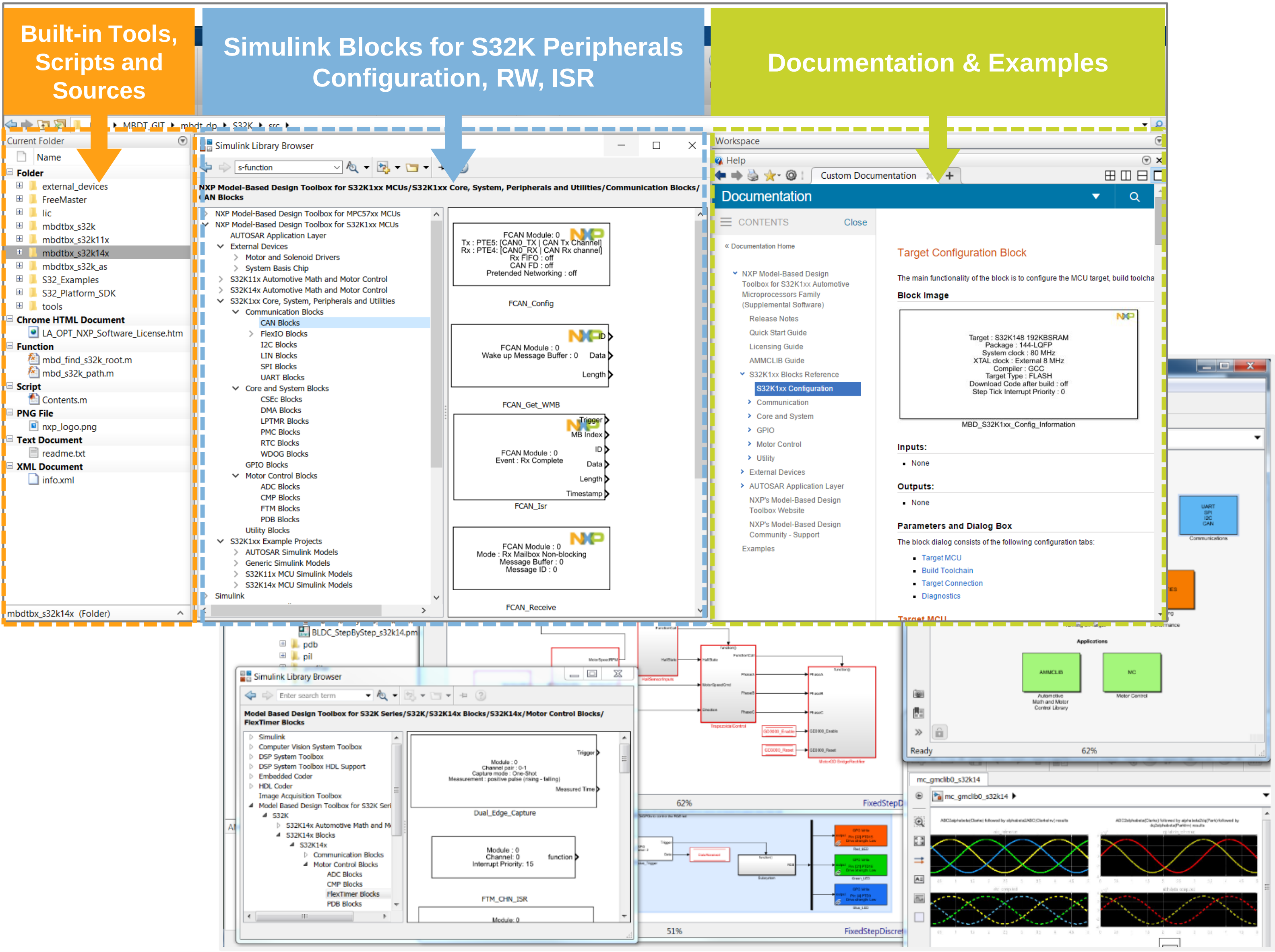
NXP Model-Based Design Solutions

- 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



Built-in Tools, Scripts and Sources

Simulink Blocks for S32K Peripherals Configuration, RW, ISR

Documentation & Examples

FCAN Module : 0
Tx : PTE5: [CAN0_TX | CAN Tx Channel]
Rx : PTE4: [CAN0_RX | CAN Rx channel]
Rx FIFO : off
CAN FD : off
Pretended Networking : off

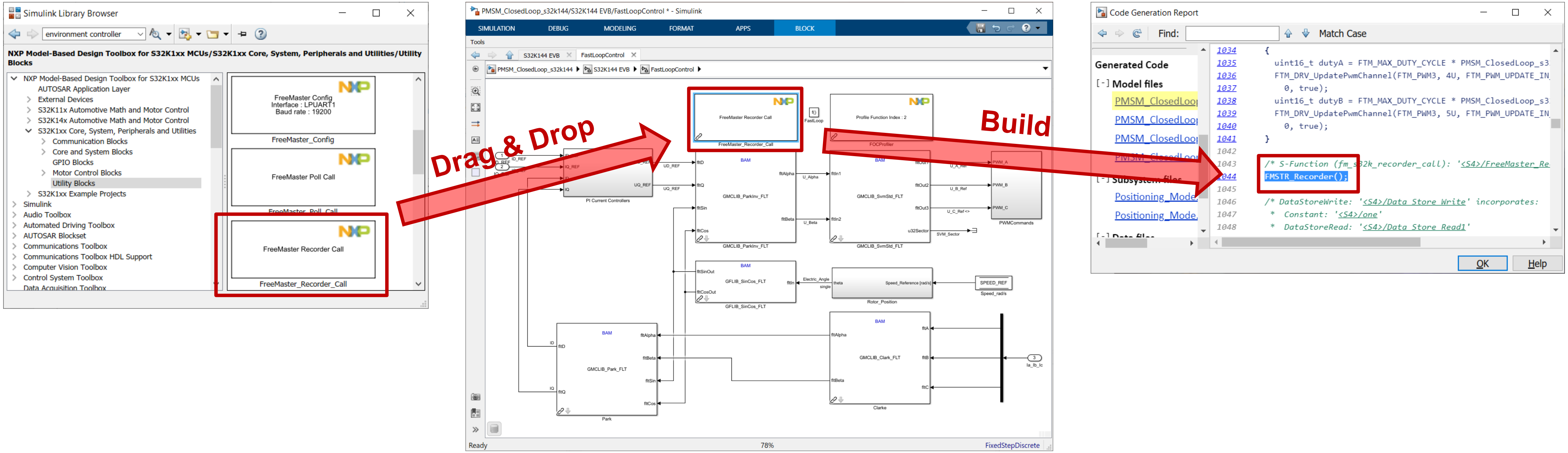
Target : S32K148 192KBSRAM
Package : 144-LQFP
System clock : 80 MHz
XTAL clock : External 8 MHz
Compiler : GCC
Target Type : FLASH
Download Code after build : off
Step Tick Interrupt Priority : 0

The main functionality of the block is to configure the MCU target, build toolcha

The block dialog consists of the following configuration tabs:

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

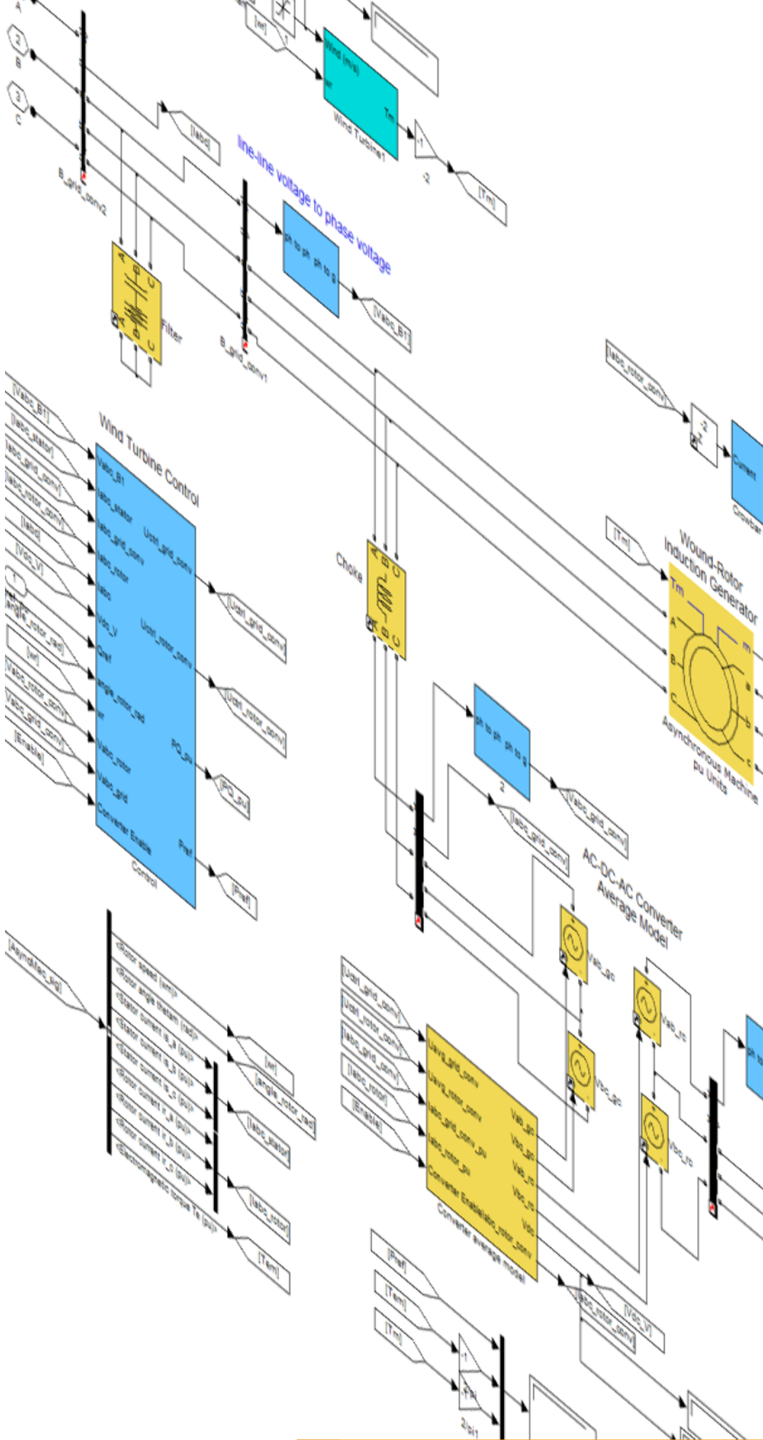


```
    {
      uint16_t dutyA = FTM_MAX_DUTY_CYCLE * PMSM_ClosedLoop_s3
      FTM_DRV_UpdatePwmChannel(FTM_PWM3, 4U, FTM_PWM_UPDATE_IN
        0, true);
      uint16_t dutyB = FTM_MAX_DUTY_CYCLE * PMSM_ClosedLoop_s3
      FTM_DRV_UpdatePwmChannel(FTM_PWM3, 5U, FTM_PWM_UPDATE_IN
        0, true);
    }

    /* S-Function (fm_s32k_recorder_call): '<S4>/FreeMaster_Re
    FMSTR_Recorder();

    /* DataStoreWrite: '<S4>/Data Store Write' incorporates:
     *  Constant: '<S4>/one'
     *  DataStoreRead: '<S4>/Data Store Read1'
```

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



Wind Turbine1
line-line voltage to phase voltage
Filter
Wind Turbine Control
Choke
Wound-Rotor Induction Generator
Asynchronous Machine pu Units
AC-DC-AC Converter Average Model
Converter average model

SECURE CONNECTIONS
FOR A SMARTER WORLD

EXTERNAL

Example#1: Modelling & Verification with Simulink & FreeMASTER

Simulink:

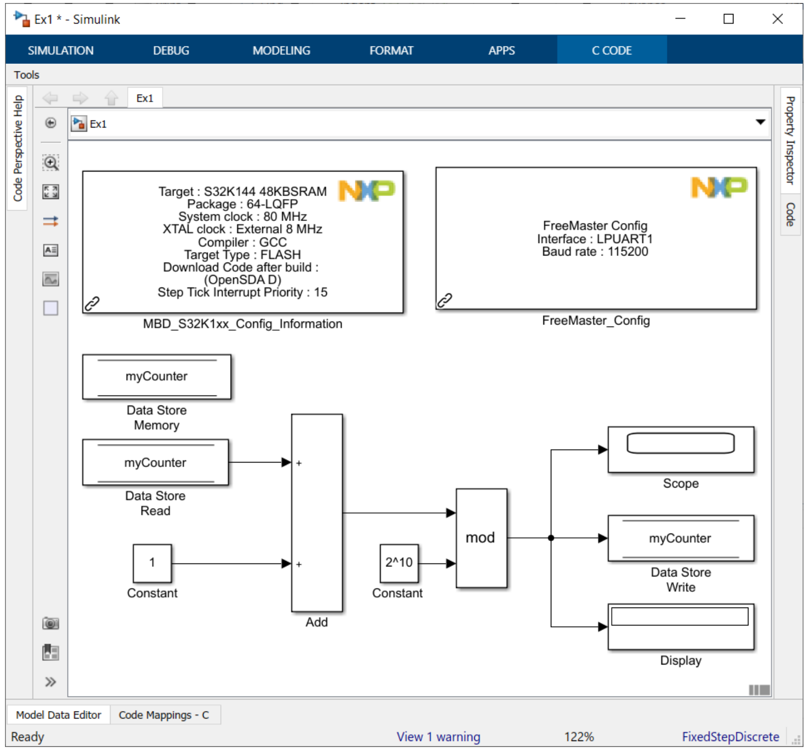
- ✓ Modelling & Simulations
- ✓ FreeMASTER configuration
- ✓ Automatic Code Generation
- ✓ Automatic Target Deployment

S32K144 EVB:

- ✓ Embedded Application Execution

FreeMASTER 3.0.2

- ✓ Real-Time data inspection
- ✓ Variable Watch
- ✓ Oscilloscope monitoring

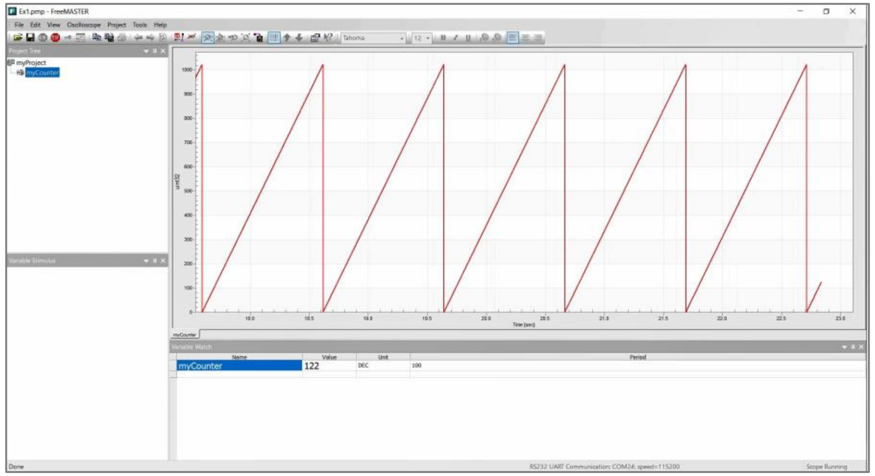


Target : S32K144 48KBSRAM
Package : 64-LQFP
System clock : 80 MHz
XTAL clock : External 8 MHz
Compiler : GCC
Target Type : FLASH
Download Code after build : (OpenSDA D)
Step Tick Interrupt Priority : 15

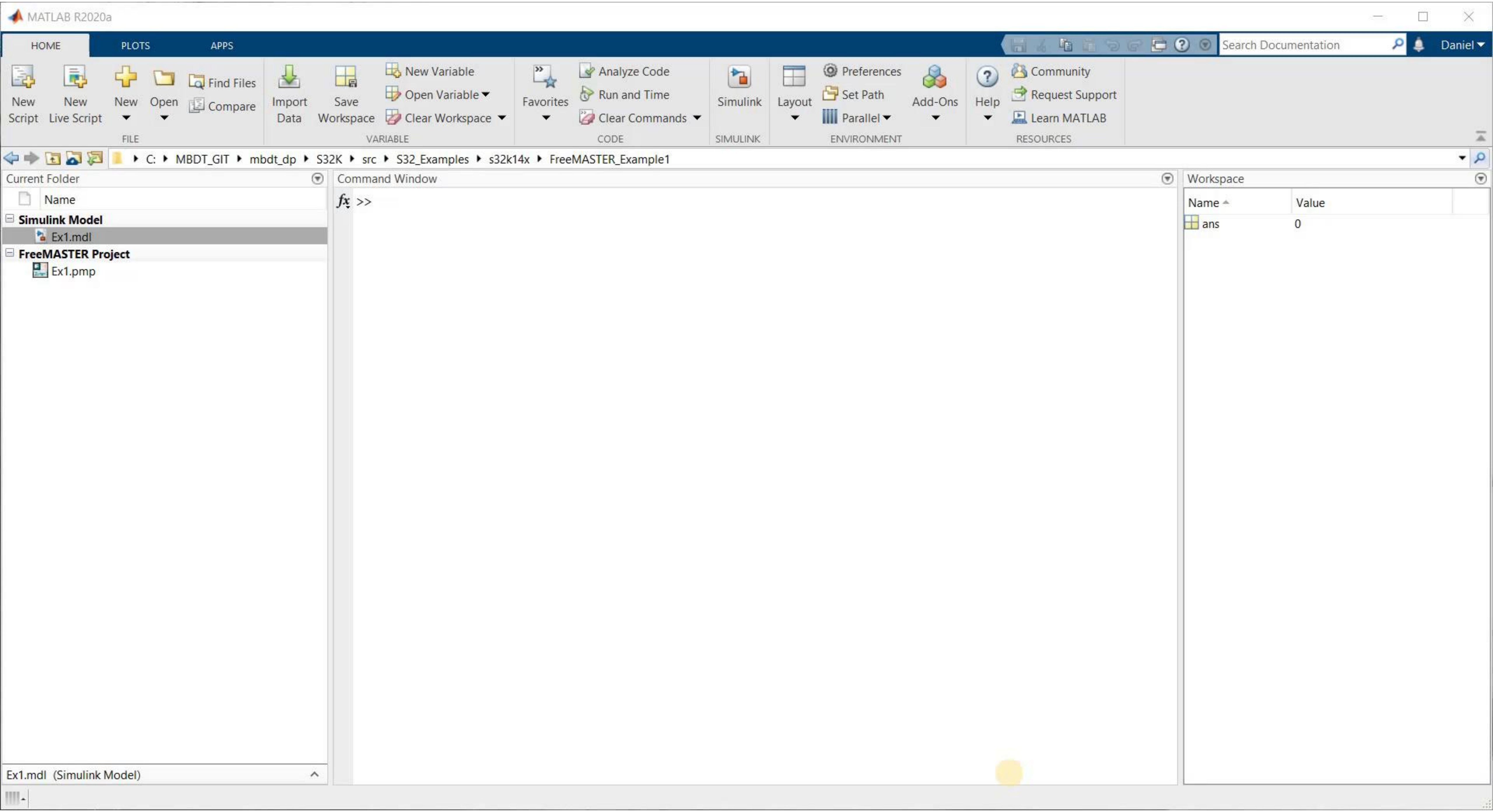
MBD_S32K1xx_Config_Information

FreeMaster Config
Interface : LPUART1
Baud rate : 115200

FreeMaster_Config



Name	Value	Unit	Period
myCounter	122	DEC	100



C: ▸ MBDT_GIT ▸ mbdt_dp ▸ S32K ▸ src ▸ S32_Examples ▸ s32k14x ▸ FreeMASTER_Example1

Current Folder:

- Name
- Simulink Model
  - Ex1.mdl
- FreeMASTER Project
  - Ex1.pmp

Ex1.mdl (Simulink Model)

Command Window:

```
fx >>
```

Workspace:

Name	Value
ans	0

Example#2: MATLAB Data Logger with FreeMASTER

S32K144 EVB:

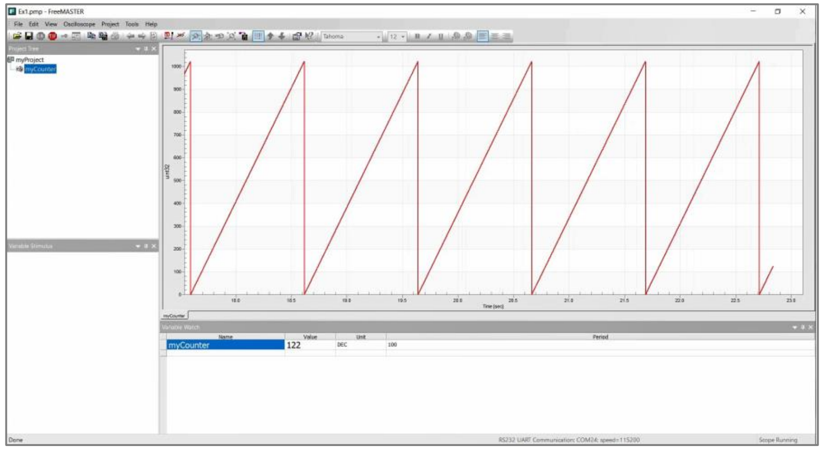
- ✓ Embedded Application Execution

FreeMASTER 3.0.2

- ✓ Real-Time data inspection
- ✓ Variable Watch
- ✓ Oscilloscope monitoring

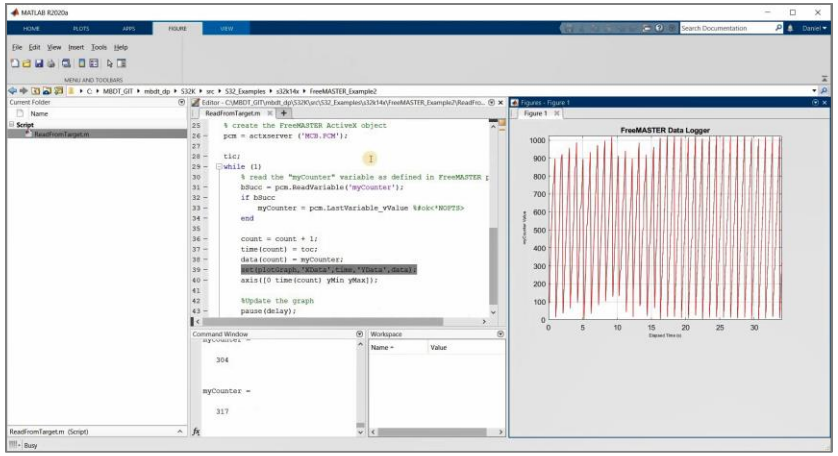
MATLAB:

- ✓ Read/Write Data via ACTXSERVER
- ✓ Plot Data
- ✓ Manipulate Data



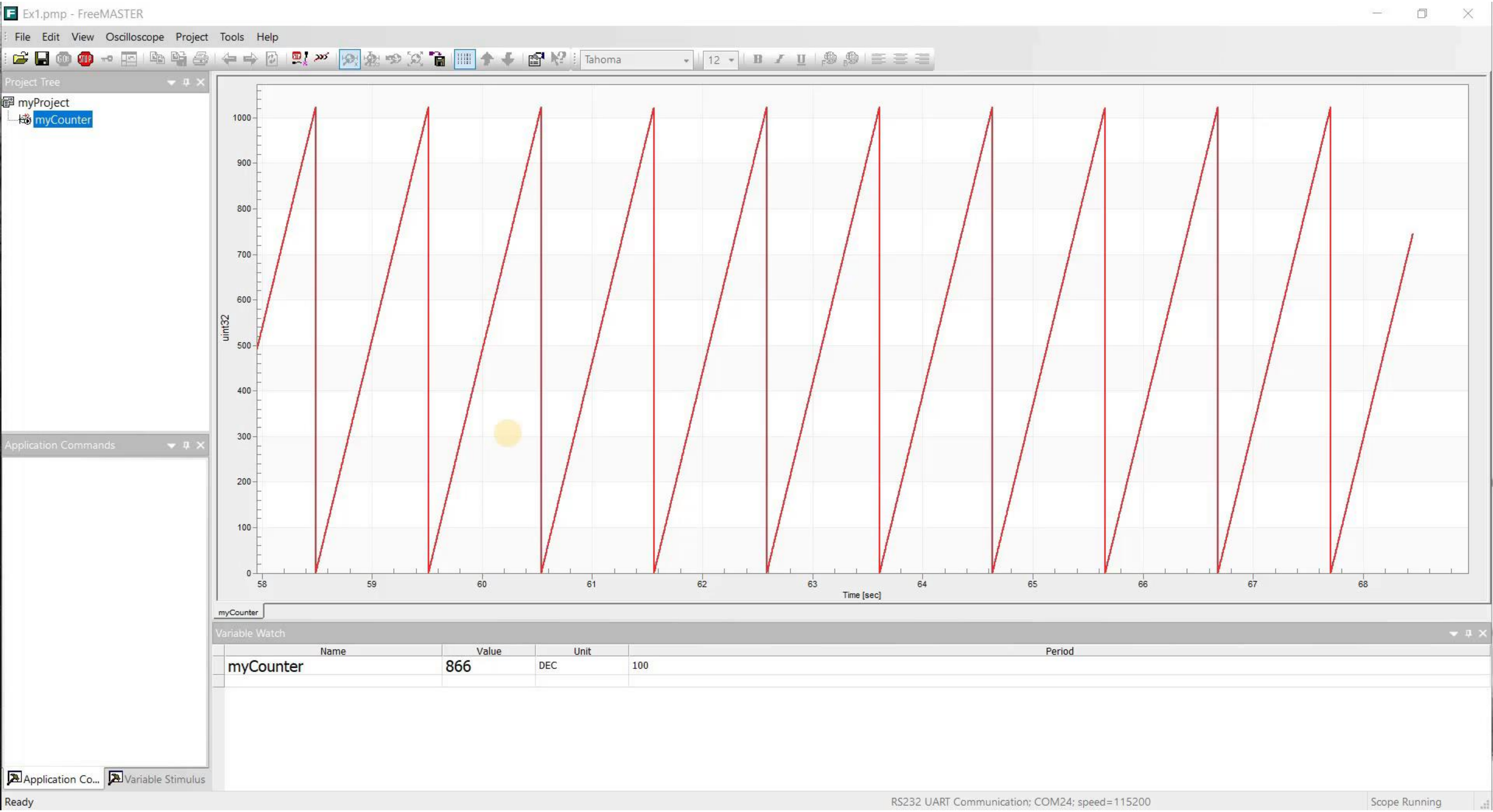
Name	Value	Unit	Period
myCounter	122	DEC	100

Done RS232 UART Communication; COM24; speed=115200 Scope Running



```
% create the FreeMASTER ActiveX object
pcm = actxserver ('MCB.PCM');

tic;
while (1)
    % read the "myCounter" variable as defined in FreeMASTER p
    bSucc = pcm.ReadVariable('myCounter');
    if bSucc
        myCounter = pcm.LastVariable_vValue %#ok<*NOPTS>
    end

    count = count + 1;
    time(count) = toc;
    data(count) = myCounter;
    set(plotGraph,'XData',time,'YData',data);
    axis([0 time(count) yMin yMax]);

    %Update the graph
    pause(delay);
```

```
  304


myCounter =

  317
```

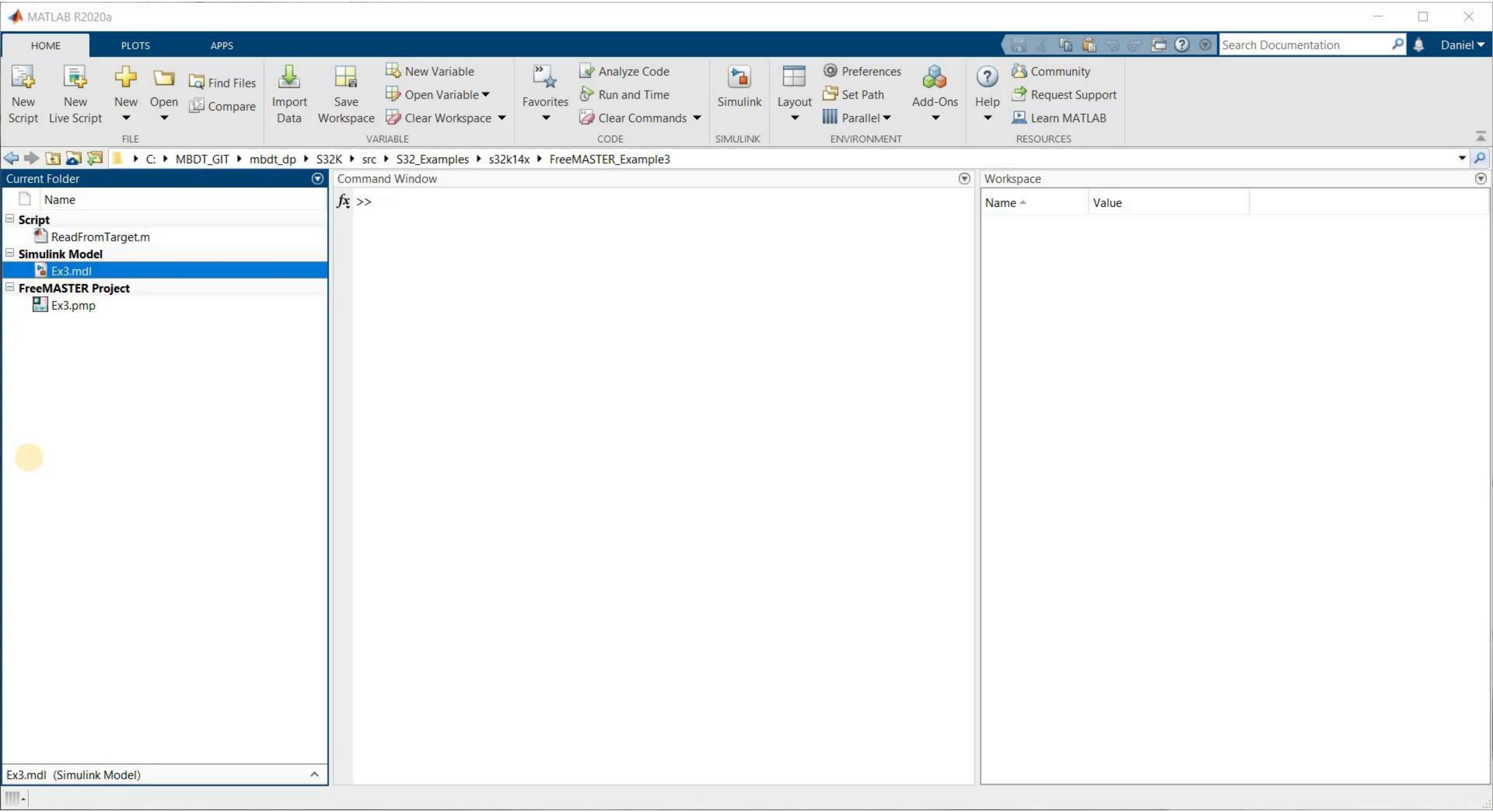


Name	Value	Unit	Period
myCounter	866	DEC	100

RS232 UART Communication; COM24; speed=115200

Bonus Example: Optimize Data Throughput

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



C: ▸ MBDT_GIT ▸ mbdt_dp ▸ S32K ▸ src ▸ S32_Examples ▸ s32k14x ▸ FreeMASTER_Example3

Script
- ReadFromTarget.m

Simulink Model
- Ex3.mdl

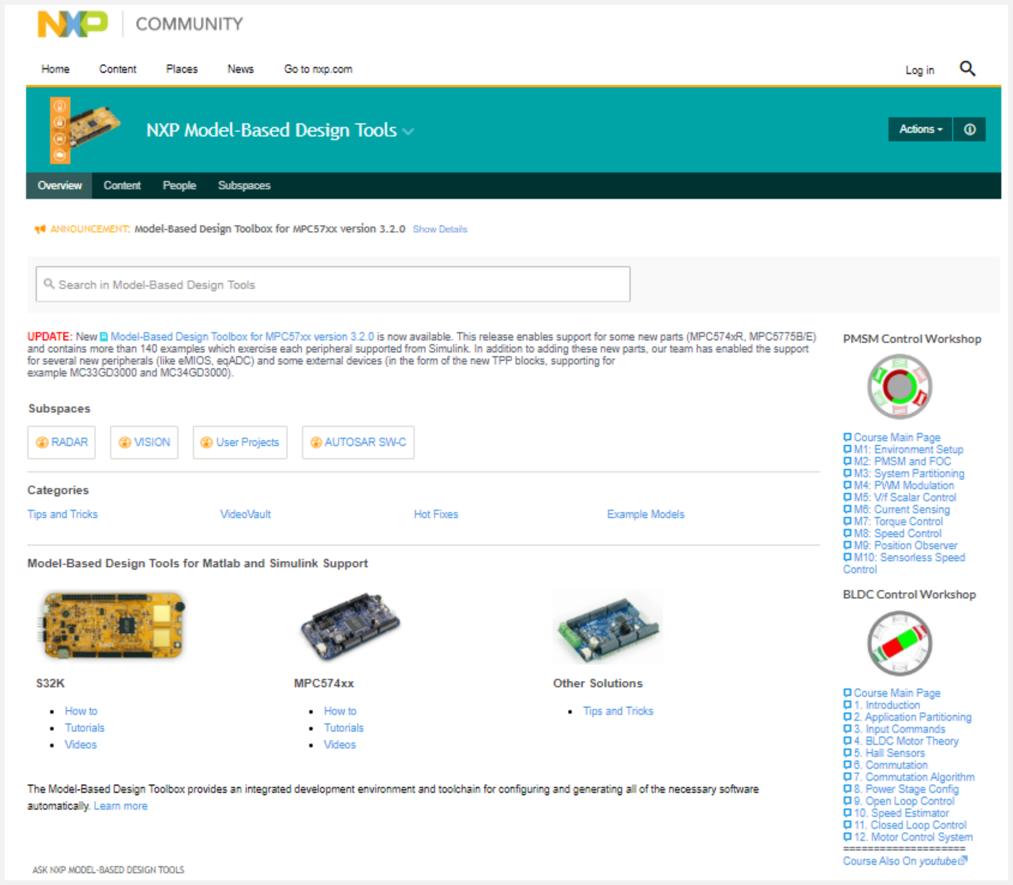
FreeMASTER Project
- Ex3.pmp

Ex3.mdl (Simulink Model)

```
fx >>
```

GETTING HELP

MBDT Online Community Examples & Help



COMMUNITY

Home Content Places News Go to nxp.com Log in

NXP Model-Based Design Tools

Actions

Overview Content People Subspaces

ANNOUNCEMENT: Model-Based Design Toolbox for MPC57xx version 3.2.0 Show Details

Search in Model-Based Design Tools

UPDATE: New Model-Based Design Toolbox for MPC57xx version 3.2.0 is now available. This release enables support for some new parts (MPC574xR, MPC5775B/E) and contains more than 140 examples which exercise each peripheral supported from Simulink. In addition to adding these new parts, our team has enabled the support for several new peripherals (like eMIOS, eqADC) and some external devices (in the form of the new TPP blocks, supporting for example MC33GD3000 and MC34GD3000).

Subspaces

RADAR VISION User Projects AUTOSAR SW-C

Categories

Tips and Tricks VideoVault Hot Fixes Example Models

Model-Based Design Tools for Matlab and Simulink Support

S32K

- How to
- Tutorials
- Videos

MPC574xx

- How to
- Tutorials
- Videos

Other Solutions

- Tips and Tricks

The Model-Based Design Toolbox provides an integrated development environment and toolchain for configuring and generating all of the necessary software automatically. Learn more

ASK NXP MODEL-BASED DESIGN TOOLS

PMSM Control Workshop

- Course Main Page
- M1: Environment Setup
- M2: PMSM and FOC
- M3: System Partitioning
- M4: PWM Modulation
- M5: V/f Scalar Control
- M6: Current Sensing
- M7: Torque Control
- M8: Speed Control
- M9: Position Observer
- M10: Sensorless Speed Control

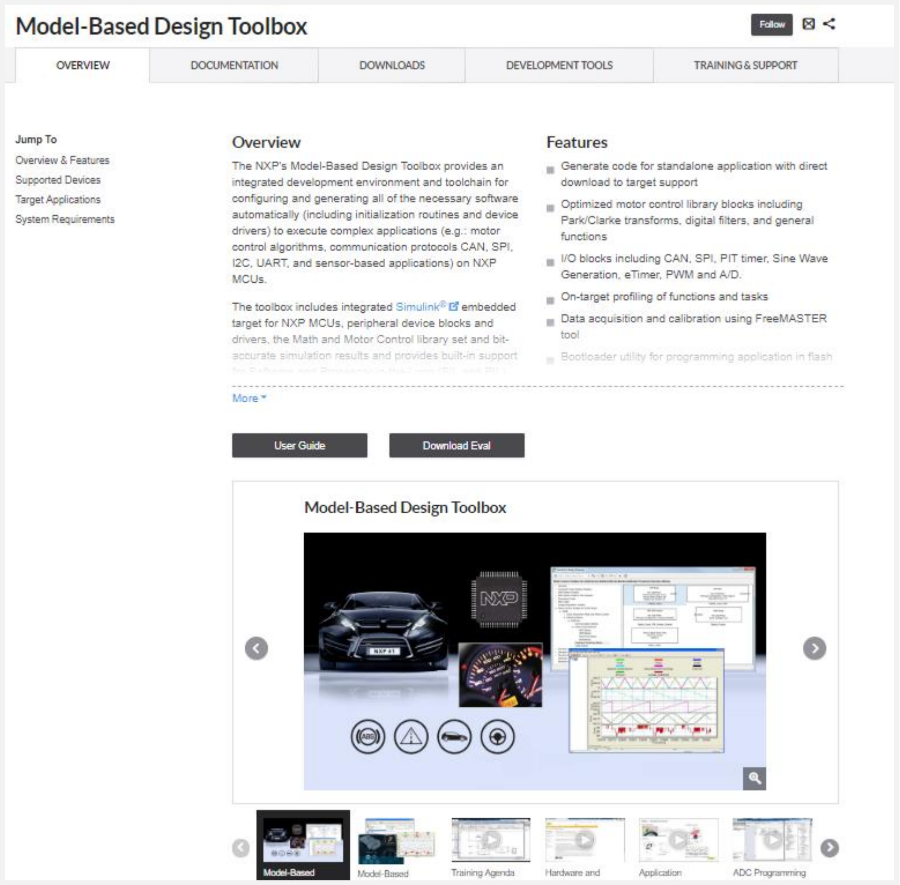
BLDC Control Workshop

- Course Main Page
- 1. Introduction
- 2. Application Partitioning
- 3. Input Commands
- 4. BLDC Motor Theory
- 5. Hall Sensors
- 6. Commutation
- 7. Commutation Algorithm
- 8. Power Stage Config
- 9. Open Loop Control
- 10. Speed Estimator
- 11. Closed Loop Control
- 12. Motor Control System

====================

Course Also On *youtube*

MBDT home page www.nxp.com/mbdt



Model-Based Design Toolbox

Follow

OVERVIEW DOCUMENTATION DOWNLOADS DEVELOPMENT TOOLS TRAINING & SUPPORT

Jump To

Overview & Features
Supported Devices
Target Applications
System Requirements

Overview

The NXP's Model-Based Design Toolbox provides an integrated development environment and toolchain for configuring and generating all of the necessary software automatically (including initialization routines and device drivers) to execute complex applications (e.g.: motor control algorithms, communication protocols CAN, SPI, I2C, UART, and sensor-based applications) on NXP MCUs.

The toolbox includes integrated Simulink® embedded target for NXP MCUs, peripheral device blocks and drivers, the Math and Motor Control library set and bit-accurate simulation results and provides built-in support

Features

- Generate code for standalone application with direct download to target support
- Optimized motor control library blocks including Park/Clarke transforms, digital filters, and general functions
- I/O blocks including CAN, SPI, PIT timer, Sine Wave Generation, eTimer, PWM and A/D.
- On-target profiling of functions and tasks
- Data acquisition and calibration using FreeMASTER tool
- Bootloader utility for programming application in flash

More

User Guide Download Eval

Model-Based Design Toolbox

Model-Based Model-Based Training Agenda Hardware and Application ADC Programming

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 SMARTER WORLD