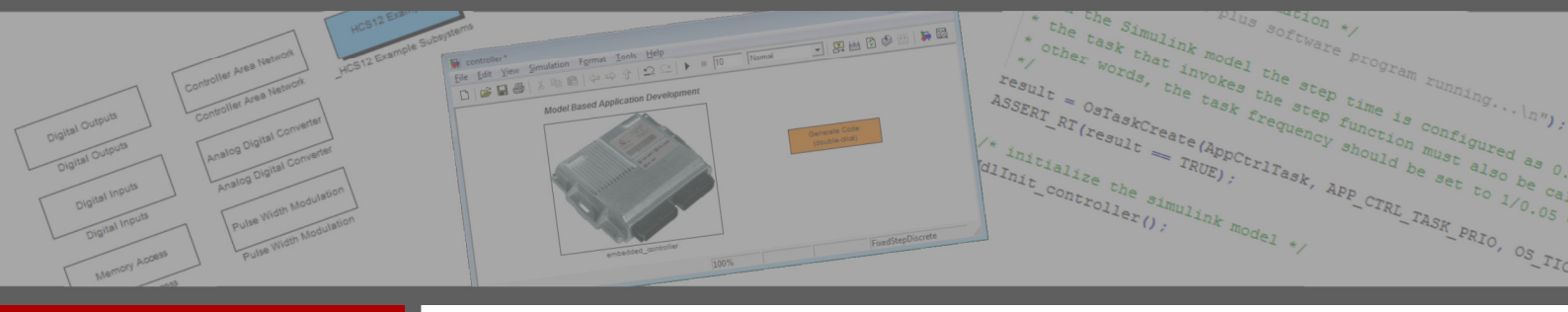
FEASER



ErtHA Blockset

for

Freescale HCS12

using the

EVBplus Dragon12+

Frank Voorburg

Feaser
http://www.feaser.com

Table of Contents

Chapter 1

Introduction to ErtHA

The ErtHA blockset enables fast model based application development in combination with Mathworks Matlab/Simulink. The ErtHA blockset described in this document is targeted towards a Freescale HCS12 microcontroller based embedded system.

The ErtHA blockset consists of three related parts: (1) a Simulink Blockset to access those Freescale HCS12 microcontroller inputs and outputs commonly required by an embedded software program, (2) a pre-configured embedded software project and (3) an interface between the first two parts that makes the functionality of a Simulink model's generated code readily available to the embedded software project.

Developing software as a Simulink model as opposed to manually coding in the C programming language has several benefits. Instead of first preparing a design document and then implementing the design as C code, the design is directly constructed as a Simulink model. Due to its visual nature the design is easier to understand. Furthermore, during the design phase the full Matlab/Simulink toolset is available, aiding the design validation and consequently shortening the design phase altogether. Thanks to Matlab's Real-Time Workshop Embedded Coder toolset, quality and efficient C code can automatically be generated out of the Simulink model, and thus drastically shortens the implementation phase.

Although it is a one-time task, setting up the Simulink model for code generation and integrating the generated code into an existing software project can be a daunting and time consuming task. Thanks to the ErtHA blockset these tasks are simplified. After reading this chapter, you can generate code out of a Simulink model and have it running on your Dragon12+ board within a few minutes.

Prerequisites

To develop your software program with the aid of the ErtHA blockset and run on on your Freescale HCS12 microcontroller system, the following tools should be installed on the developer’s PC:

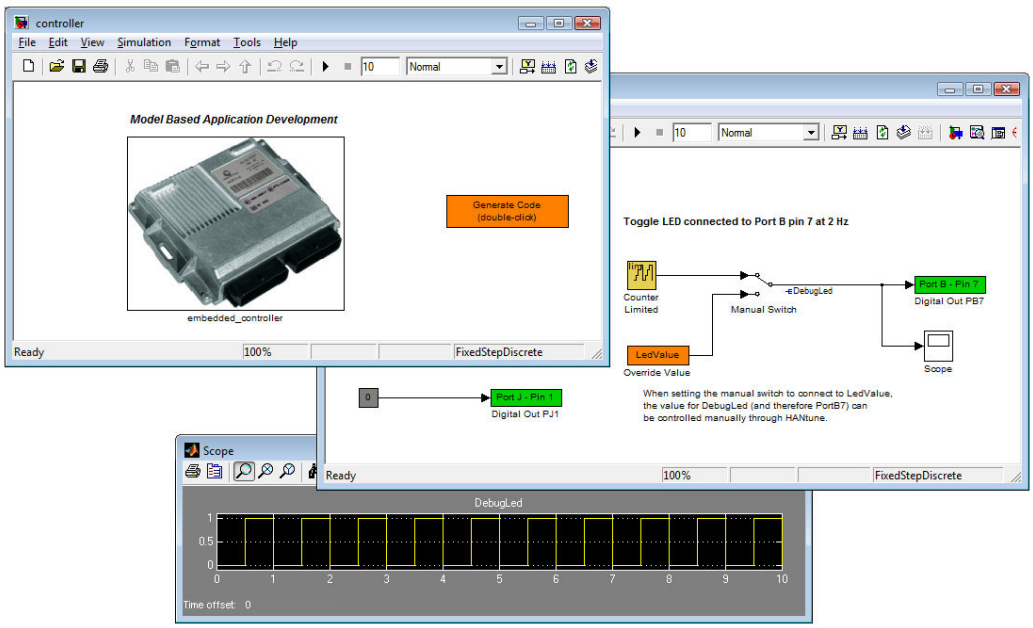
- Eclipse editor and GNU GCC HCS12 compiler toolset. An installer with these two components packed together is available for free from: http://www.feaser.com/zip/eclipse_m6811_setup.zip.
- Mathworks Matlab/Simulink version 2008a or higher (32-bit version), including the Real-Time Workshop Embedded Coder option. More information is found on the Mathworks website. For evaluation purposes, contact Mathworks to request a time limited demo.

Getting Started Tutorial

The ErtHA blockset contains a preconfigured template project targeted towards a Dragon12+ board from EVBplus (http://www.evbplus.com). This section illustrates the steps involved in generating code out of a Matlab/Simulink model, building the code into a Motorola S-record and programming the S-record into the flash EEPROM of the HCS12 microcontroller on the Dragon12+ board.

Opening the Simulink Model

To open the preconfigurated Simulink model and consequently Matlab/Simulink, double-click the file **.\Target\Demo\HCS12_Dragon12_GCC\Workspace\Dragon12\src\app\model\controller.mdl** from Windows Explorer. This method automatically sets the correct Matlab workpath.

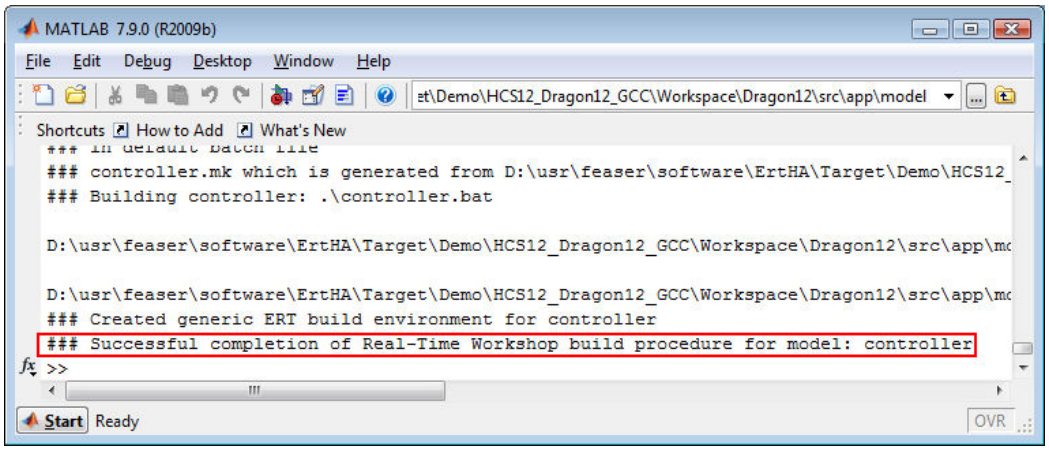


The template Simulink model contains a few blocks to initialize some I/O on the Dragon12+ board and enables the control of digital output pin 7 on Port B. Combined with a limited counter block, a blinking LED on the Dragon12+ board can be created.

Code Generation

To start the code generation with the Real-Time Workshop Embedded Coder, simply double-click the orange *Generate Code* button on the main screen of the Simulink model.

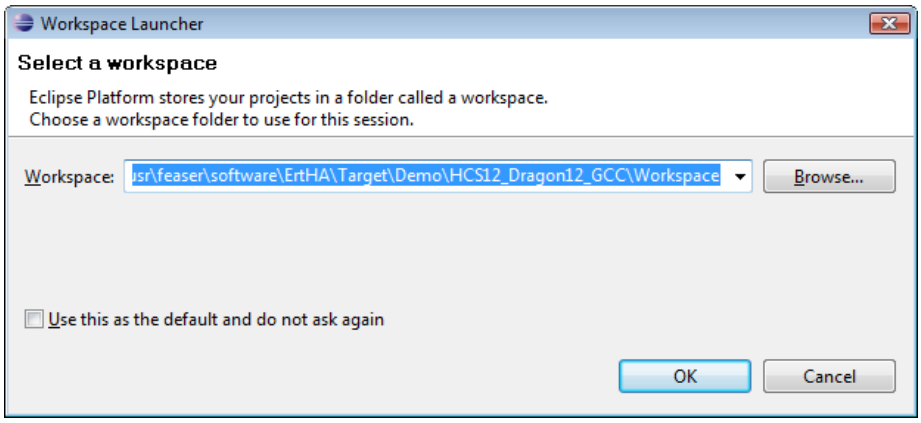
When the line **### Successful completion of Real-Time Workshop build procedure for model: controller** appears in the Matlab command window, the code generation is completed.



Software Building

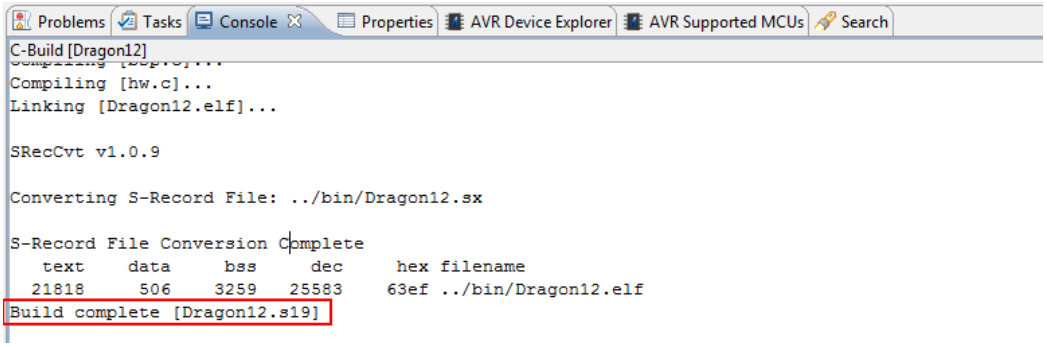
At this point the code out of the Simulink model is generated into C code. This C code, together with the manually coded C code6 now needs to be compiled and linked together to create a final executable in Motorola S-record format.

Open the Eclipse editor and select the workspace located at **.\Target\Demo\HCS12_Dragon12_GCC\Workspace\.**



When opening the workspace for the first time, you might get the recommendation to first refresh the workspace. This is achieved by select **File->Refresh** from the program menu. This only has to be done once.

To build the software program, select **Project->Build All** from the program menu and observe the output in the console window to make sure no error occurred:



The final executable in Motorola S-record format is now created and available at:
.\Target\Demo\HCS12_Dragon12_GCC\Workspace\Dragon12\bin\Dragon12.s19

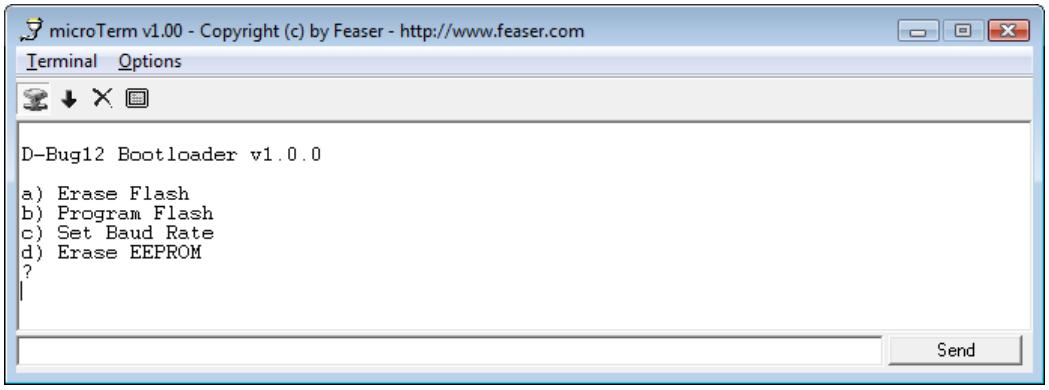
Programming the flash EEPROM

Once the Motorola S-record is created, it can be programmed into the flash EEPROM of the microcontroller and the software program can be started. The exact procedure for this depends on the development environment. This example demonstrates the Dragon12+ board in combination with the resident DBUG12 bootloader program.

Make sure both jumpers of SW7 at in the BOOT configuration, the board is connected to the PC's serial port through the SUBD9 connector (P1).

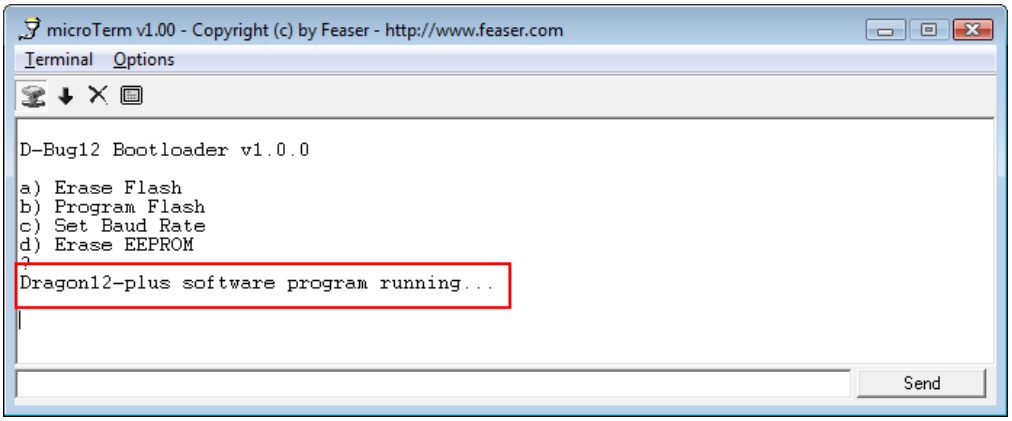
Next, open the **microTerm.exe** program located at **.\Target\Demo\HCS12_Dragon12_GCC\Workspace\Dragon12\cmd**. Select **Options->Settings** from the program menu to configure the correct COM-port. The other settings are already correct (baudrate = 9600). Click the **OK**-button to store the communication settings. Now connect to the COM-port by selecting **Terminal->Connect** from the program menu.

If you reset the HCS12 microcontroller through the reset-button on the Dragon12+ board, you should now see the following in the terminal screen:



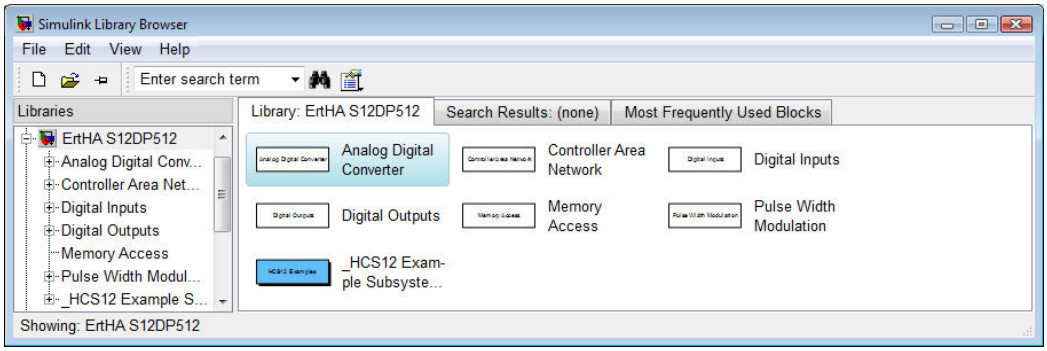
To program the Motorola S-record into flash EEPROM with them microTerm program, enter the following commands:

1. First erase the flash EEPROM by typing '**a**' in the lower edit-box, followed by a click on the **Send**-button.
2. Enter programming mode by typing '**b**' in the lower edit-box, followed by a click on the **Send**-button. Next, select **Options->Download File** from the program menu and choose the **Dragon12.s19** file.
3. You should now see ***** appear in the terminal screen, which indicates the progress of the flash EEPROM programming. When programming is done, the initial menu appears again.
4. To start the program, make sure both jumpers of SW7 are in the EVB configuration and reset the microcontroller. You should now see a blinking LED on the Dragon12+ board and an info message in the terminal screen:

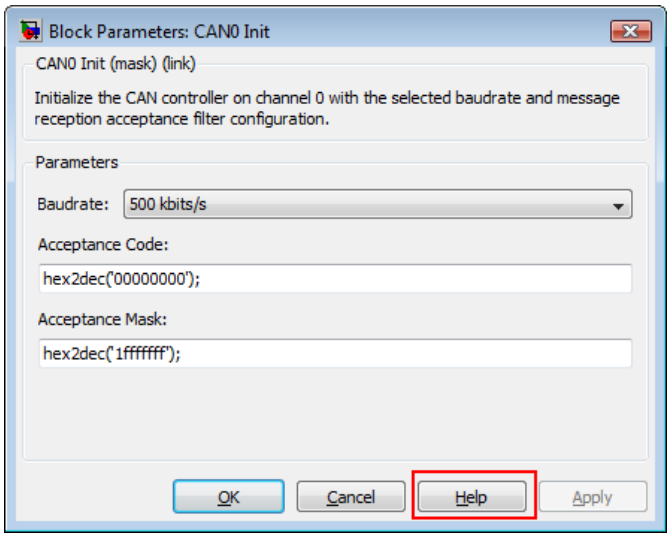


Blockset Overview

The ErtHA blocks, like all other Simulink blocks, are accessible through the Simulink library browser. Select **View->Library Browser** from the Simulink program menu to open the library browser. The ErtHA blocks are located in a section called ErtHA S12DP512:

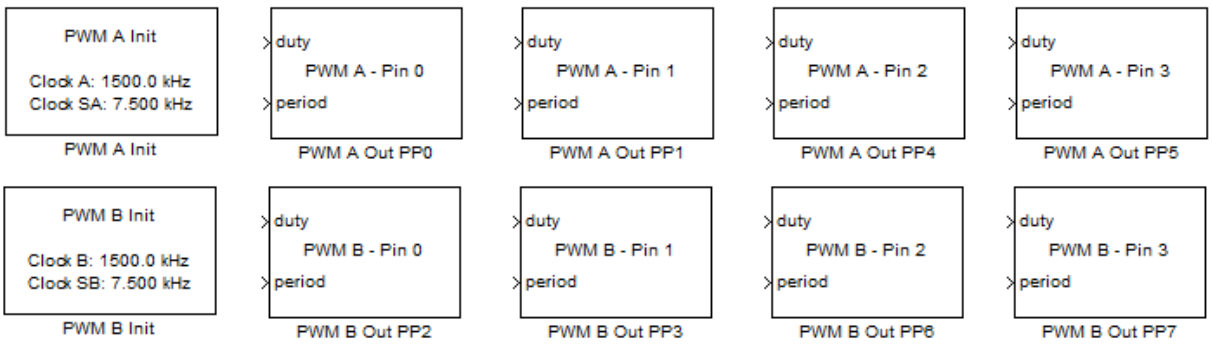


For reference purposes, this section contains an overview of all the supported ErtHA blocks. For more information on a particular block, click its **Help**-button:



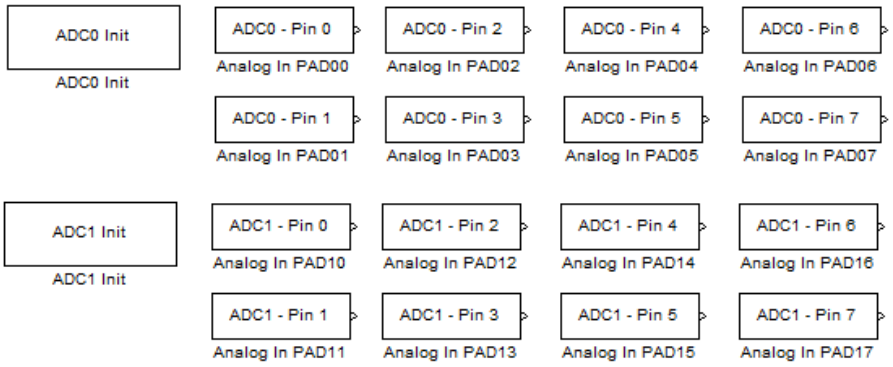
Digital Inputs

Pulse Width Modulation

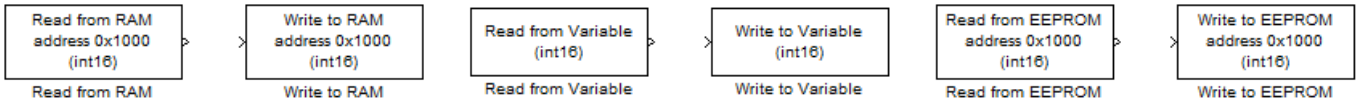


Digital Outputs

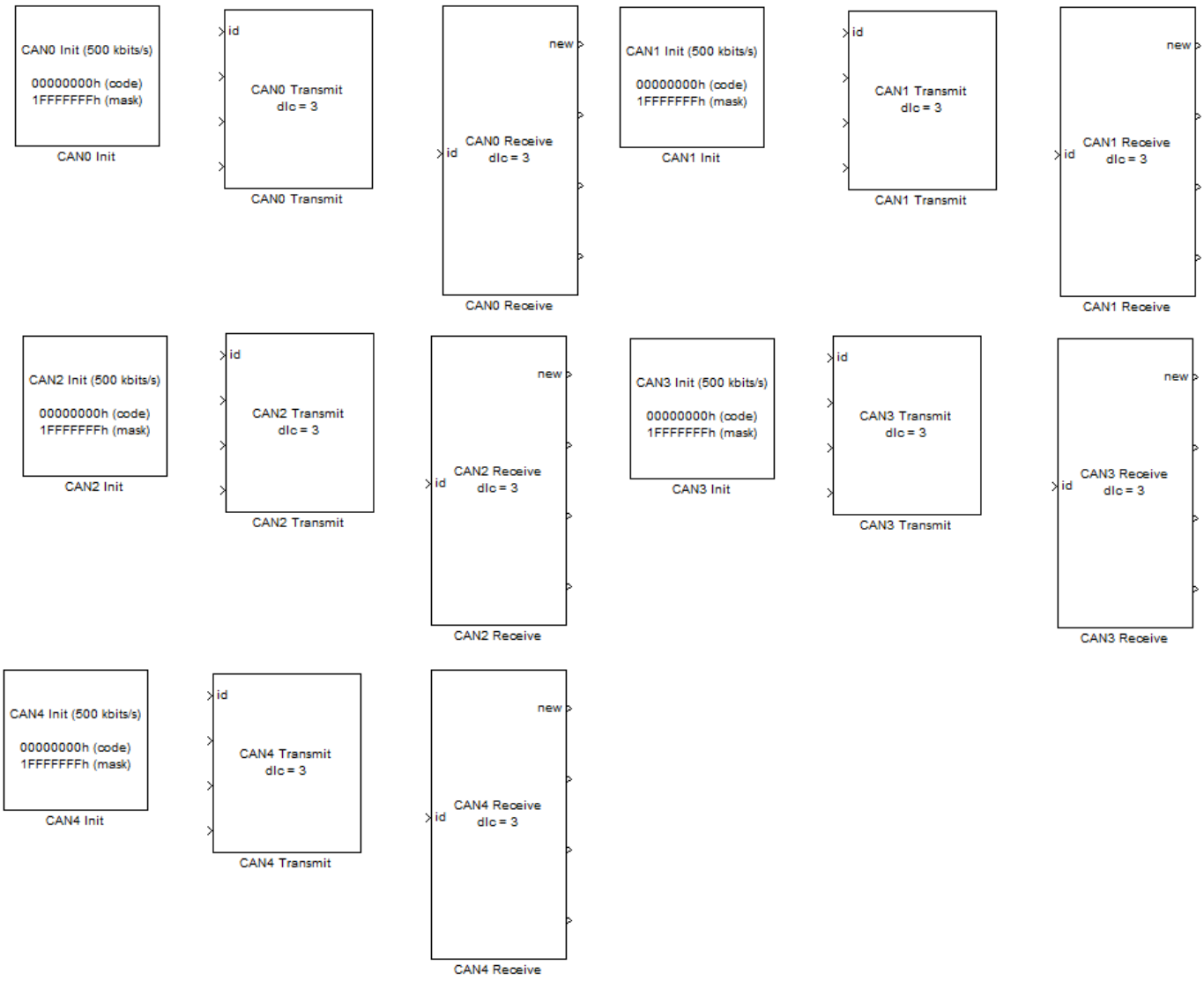
Analog Digital Converter



Memory Access



Controller Area Network



HANtune – Realtime Monitoring

With special permission from HAN Automotive, we were allowed to include a beta version of the HANtune program into the ErtHA software package.

HANtune is a Java based program that runs on a Windows PC and communicates with the microcontroller. The communication is through the CAN bus, based on the XCP protocol. HANtune currently supports the Peak CAN USB interface. For more information about HANtune, please refer to the file **.\Host\README.TXT**.

This chapter contains instructions on how to get HANtune up-and-running. It is assumed that the software program described earlier in this chapter is running on the Dragon12+ board.

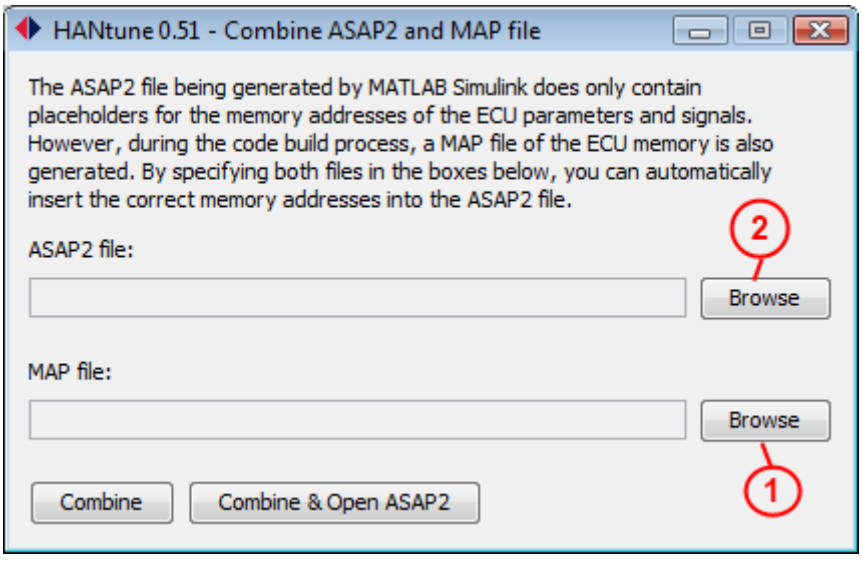
Hardware Setup

Start by connecting the one end of the Peak CAN USB interface to your PC and the other end to a CAN bus. As always, the CAN bus should have 120 Ohm termination resistors on both sides. Next, connect the CAN-HI and CAN-LO pins of the Dragon12+ board on the CAN0 connector to CAN bus.

Important: make sure the jumper J39, labeled PM0, is closed. This jumper is located on the right side of the LCD screen on the Dragon12+ board.

Project Setup

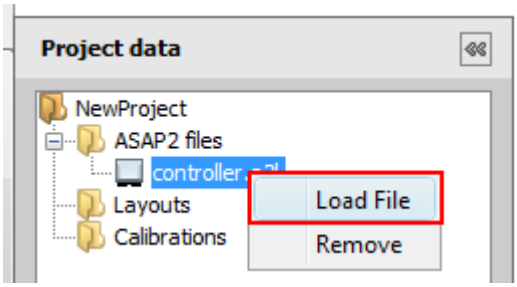
Start HANtune by double-clicking **.\Host\HANtune.exe**. Now select from the program menu **File->Combine ASAP2+MAP**.



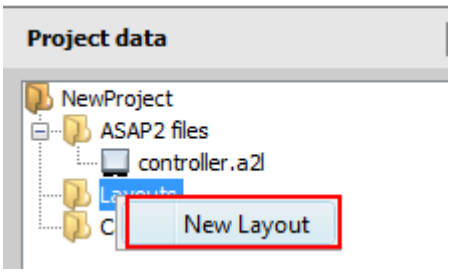
First set the MAP file to:
.\Target\Demo\HCS12_Dragon12_GCC\Workspace\Dragon12\bin\Dragon12.map

Next set the ASAP2 file to:
.\Target\Demo\HCS12_Dragon12_GCC\Workspace\Dragon12\src\app\model\controller_ert_rtw\controller.a2l

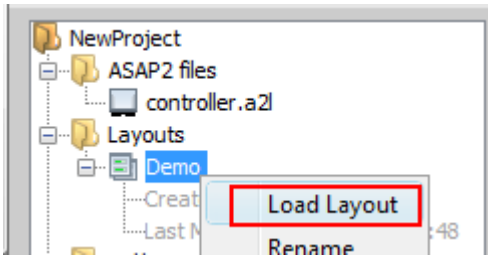
Once the files are set, click the **Combine & Open ASAP2** button, to add this ASAP2 file to the project. One more step is needed to actually load the ASAP2 data. This is achieved by right-clicking the **controller.a2l** file and selecting **Load File** from the popup menu:



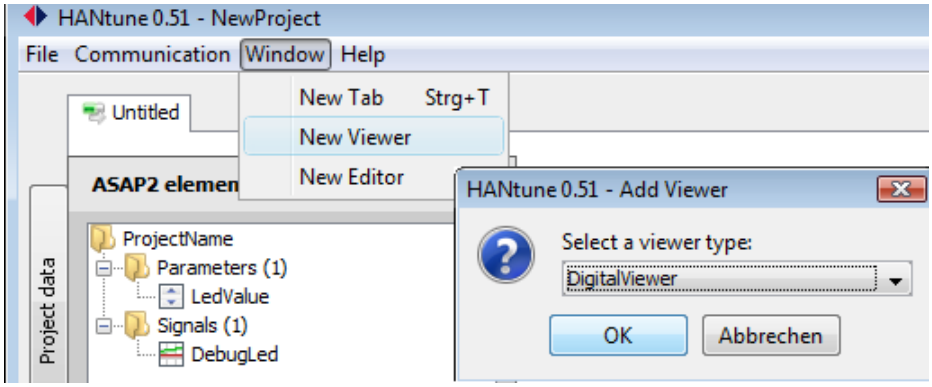
In order to visualize signals on the screen, a layout is required. Right-click on **Layouts** and select **New Layout** from the popup menu and enter a name for the layout, e.g. **Demo**:



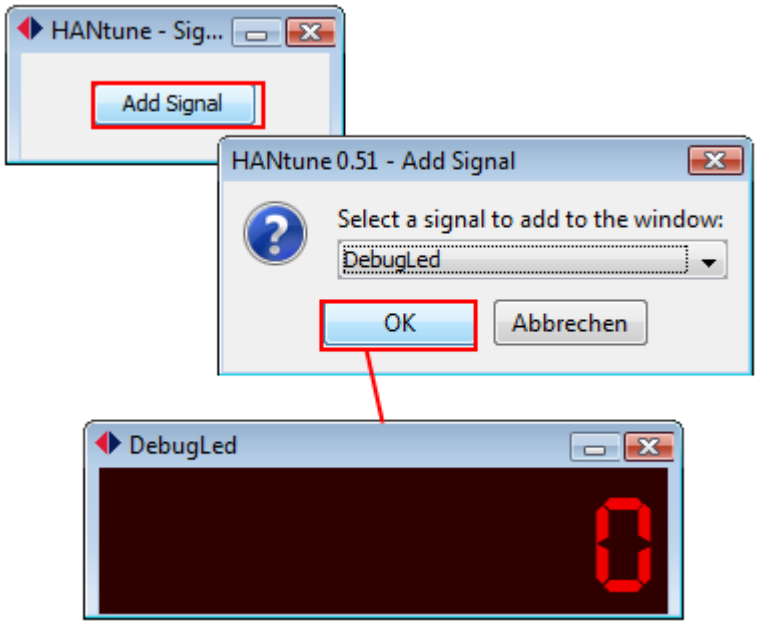
After creation of the new layout, it can be loaded. Right-click on **Demo** and select **Load Layout** from the popup menu:



The program currently running on the Dragon12+ boards has one signal that can be visualized on the layout in HANtune. The name of this signal is **DebugLed**. Signals are visualized on the layout through so called viewers. Add a viewer by selecting **Window->New Viewer** from the program menu. You can choose different viewers. For this tutorial, go for the **DigitalViewer**:



After placing a DigitalViewer on the layout, the DebugLed signal can be added to it:

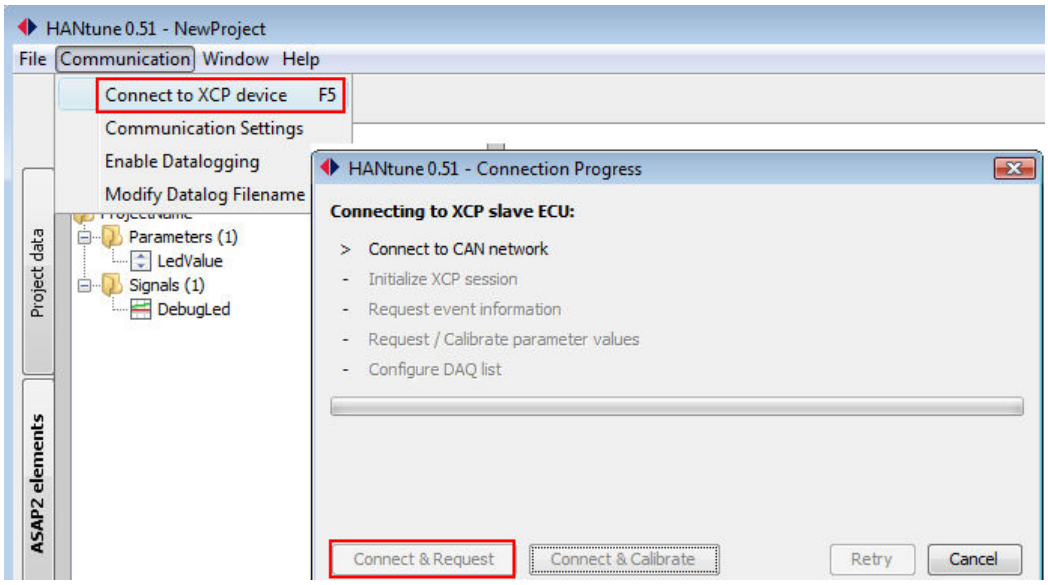


The project setup is now complete. It's a good idea to save it at this point, so these steps won't have to be repeated. Select **File->Save As** from the program menu and follow the instructions.

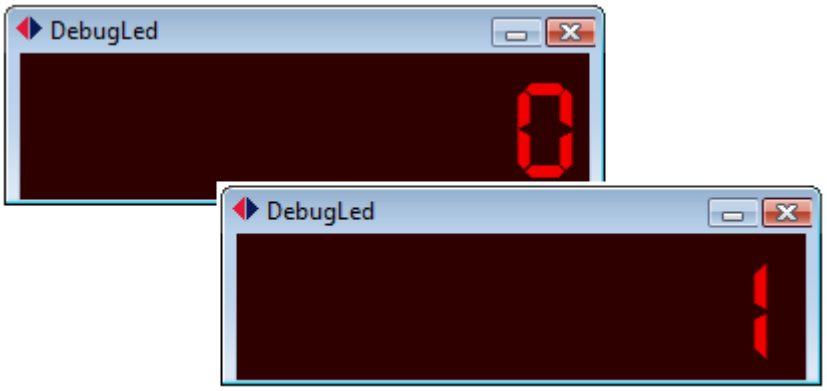
Note: For an improved workflow the next time you load the project, it is a good idea to save the project into the same directory as where the controller.a2l file is located:
.\Target\Demo\HCS12_Dragon12_GCC\Workspace\Dragon12\src\app\model\controller_ert_rtw

Establishing a Connection

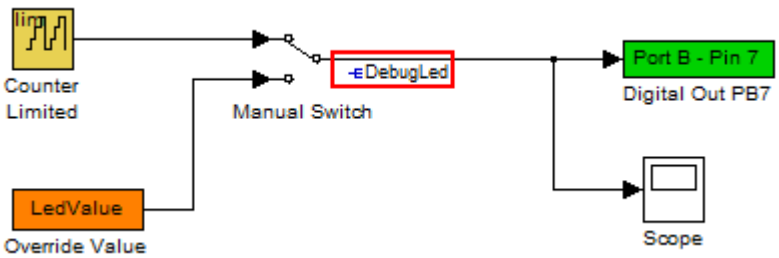
To connect HANtune to the microcontroller via the CAN bus and to start a realtime monitoring session, select **Communication->Connect to XCP Device** from the program menu:



The realtime monitoring is now active and in this simple example, the value of the DebugLed is visualized:



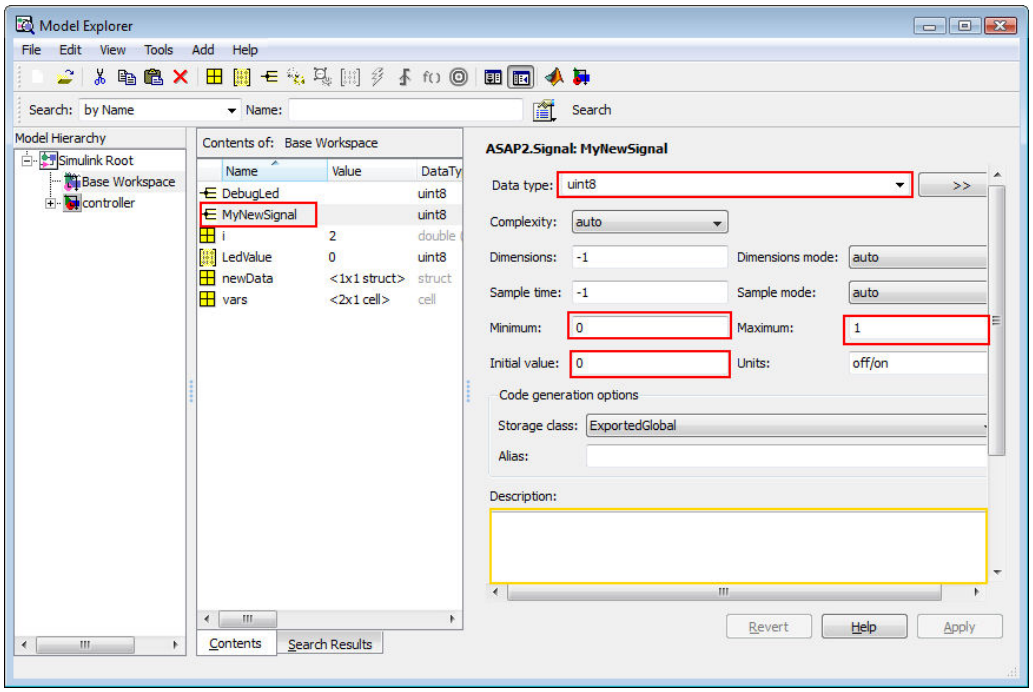
To summarize: with HANtune you can visualize all signal line values from your Simulink model. This enables you to perform runtime software program verification on an application level. Combining the features of the ErtHA blockset and HANtune gives you a good toolset for model based rapid application development for embedded targets.



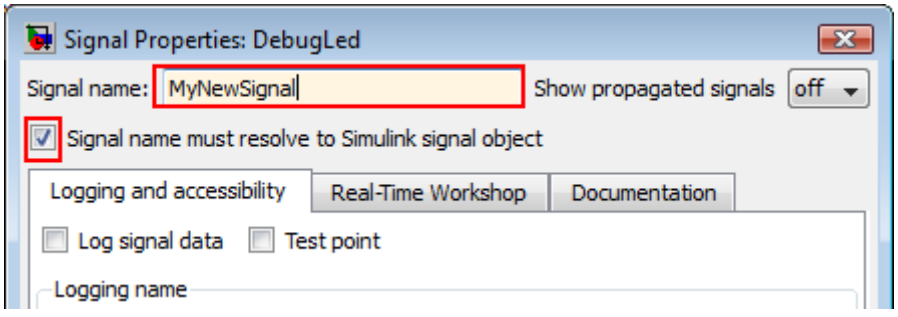
Adding Signals

At this point only the value of the **DebugLed** signal from the Simulink model is available in the ASAP2 file. There are a few steps involved in adding more signals from the Simulink model to the ASAP2 file, which are described here:

1. Select **View->Model Explorer** from Simulink's menu and look into the **Base Workspace**.
2. Add a new signal definition. The easy way of doing this is by copy-pasting the existing **DebugLed** signal and changing its name. Next, set the datatype and minimum, maximum and initial values. Optionally you can add a description of the signal for you own reference:



3. To link the newly defined signal to a signal line in the Simulink model, right-click the signal line and select **Signal Properties** from the popup menu. Enter the **Signal name** and check the box **Signal name must resolve to Simulink signal object.**



Before you can visualize the new signal, don't forget that you need to generate code out of the Simulink model again, build the software and program it in the internal flash EEPROM. Also, in HANtune, the ASAP2 and MAP files need to be combined again, etc.

Note that the signal definitions are stored in Matlab's Base Workspace. This means that they are not stored in the ***.mdl file of the Simulink model. To store the signal definitions, select **File->Save Workspace As** from the Matlab Command Window and select the file **matlab.mat** that is located in the same directory as the **controller.mdl** file.

The demo project is configured in such a way that, when starting Matlab by double-clicking the **controller.mdl** file, the data in **matlab.mat** is automatically loaded into Matlab's Base Workspace. Refer to the file **startup.m** for details on how this is achieved.

ErtHA Blockset for Freescale HCS12

The ErtHA blockset enables fast model based application development in combination with Mathworks Matlab/Simulink. The ErtHA blockset was specifically created for embedded software developers who want to reap the benefits of model based development, yet are not that familiar with or do not have the time available to setup a development environment that enables quick and easy code generation from a Simulink model.

This document accompanies an example reference project and outlines the steps to get started with the ErtHA blockset in combination with a Dragon12+ boards from EVBplus.