

Model-Based Design Case Study: Low Cost Audio Head Unit

2011-01-0052

Published
04/12/2011Jason Marx and Weiqian Sun
Ford Motor Company

Copyright © 2011 SAE International

doi:[10.4271/2011-01-0052](https://doi.org/10.4271/2011-01-0052)

ABSTRACT

The use of model-based software development in automotive applications has increased in recent years. Current vehicles contain millions of lines of code, and millions of dollars are spent each year fixing software issues. Most new features are software controlled and many times include distributed functionality, resulting in increased vehicle software content and accelerated complexity. To handle rapid change, OEMs and suppliers must work together to accelerate software development and testing. As development processes adapt to meet this challenge, model-based design can provide a solution.

Model-based design is a broad development approach that is applied to a variety of applications in various industries. This paper reviews a project using the MATLAB/Simulink/Stateflow environment to complete a functional model of a low cost radio. We approached the project independent of any specific software architecture, using only written specifications and current working units as a guide to functionality. The final result is a model that captures the functional behavior of the radio. While the model could be used for auto code-generation, this project focused on utilizing the radio model to prove-out the functional and HMI specifications.

A discussion of our design approach, including architecture, testing, related HMI components, and lessons learned, are included in the paper.

INTRODUCTION

There has been rapid growth of new vehicle electrical and multimedia features in recent years. Automakers are now keeping pace with consumer electronics, pushing the

envelope on software complexity while driving a shorter time-to-market for new features [1]. To manage this challenge, there is a strategic push to increase the use of model-based design. Many functional areas of Ford Motor Company have actively utilized model-based design for years, and here we report on a case where this approach is applied to electrical commodity development successfully.

Our approach is to create a functional model of a low cost radio, similar to the recent work in [2] and [3]. We demonstrate the usefulness of applying this approach early in the process to accelerate development while simultaneously improving the quality of the initial feature design and specification. We worked jointly with the core radio and HMI teams to develop the model concurrently with their specifications. The model is considered to be an executable version of the requirements where testing validation ensures a mature design.

It should be noted that a comprehensive model-based design approach typically includes the development of a full application model, auto code-generation from the model, and an automated integration of the generated code with the basic software (operating system, drivers, etc) of the target. A good review of strategies for managing the adoption a model-based design can be found in [4]. Since this case study is being completed in a short time, it is purposely bounded to proving-out requirements, understanding modeling tool capabilities, developing best practices, and evaluating next steps for future model-based design projects. While the model will be authored in a style to support auto code-generation, it is not a project goal as it is target independent.

The low cost radio under development is meant to augment the current mixture of available radios. It was purposely chosen for this pilot project as it is essentially a self contained

unit, and so is the model development. Although this low cost radio has limited features and is not connected to the CAN, it does contain a complete feature set that includes AM, FM, AUX sources, a full menu and bezel diagnostics strategy, ability to set market configuration, and RDS functionality for related markets.

In this paper, we describe the development process and effectiveness of applying a model-based design approach. The methodology section addresses the process followed to transform the specifications into a model. The architecture and testing sections detail the model design and our approach to validation. We subsequently include sections that discuss our creation of a radio bezel GUI, additional user-interface design considerations, and conclude with our lessons learned.

METHODOLOGY

The process of creating the radio model follows a systems engineering approach. We first perform a requirement analysis, where we review specification documents and decide the final requirement content to be included in the model. In this project, all functional requirements that drive the application software are included while detailed hardware requirements (packaging, thermal and EMC conditions, etc) are not considered. We make reasonable assumptions for signals that we use to communicate with the radio bezel, display, audio, and tuner circuits. The included requirements are then grouped by feature and ranked by increasing complexity. A development plan is created from the prioritized requirements list. Our plan divides model development into three sequential phases: basic operations, advanced features, and global features and diagnostics.

The next step is to begin the actual model development by first defining the model architecture at the highest-level. Here, we define the top-level input and output signals to the radio model based on our analysis of the requirements. Input signals include items such as radio bezel buttons and hardwired circuits. Outputs include signals that would drive the display, sound circuits, and tuner circuits. We consider this to be our “system-level”. All core logic included within the system-level is based entirely on the functional requirements.

After the system-level model architecture is defined, we move to implement the lower levels of the model. These lower levels are where each requirement is realized into the model. Stateflow chart blocks, basic Simulink blocks, and Embedded MATLAB blocks are used to model individual requirements. We consider this to be the “unit-level” of the model. Typically, requirements are first modeled at this unit-level, and then as several related requirements are developed, they are grouped together to form a subsystem. As related subsystems are created, they are grouped again into a higher-level subsystem, eventually forming several model layers.

The model layer just below the system-level includes super-subsystems dedicated to broad features such as Tuner_Logic, Menu_Strategy, and Diagnostics_Strategy. Testing of the model is carried out throughout the process at both the unit-level and system-level.

ARCHITECTURE

The top-level of the radio model is shown in [Figure 1](#). This view contains the system-level core radio model in the middle with supplemental input and output subsystems that serve as harnesses to help drive the system-level radio model. Subsystems on the left create the input signals that feed the radio while subsystems on the right manage the radio output signals. The input subsystems contain an array of switches that are controlled by buttons on a GUI that simulates the radio bezel. Debounce logic is not included in the system-level radio model. The output subsystems manage the radio output signals by applying logic to drive the radio bezel display and provide realistic tuner feedback to the radio. The radio bezel, tuner subsystem harness, and other ancillary design elements are discussed in detail later in this paper.

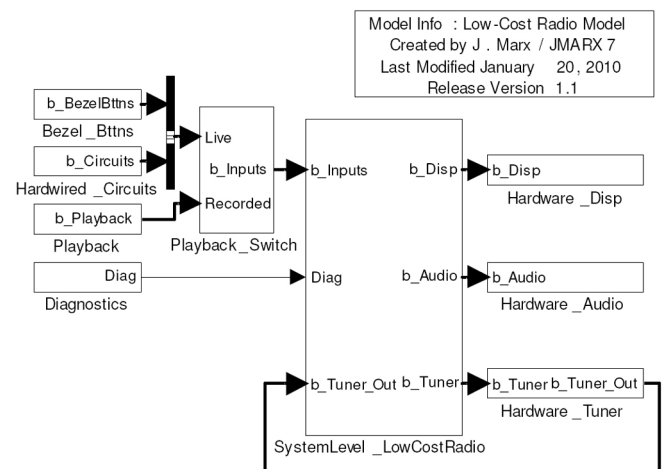


Figure 1. Top-level view of radio model.

The contents of the system-level radio are shown in [Figure 2](#). This view shows the super-subsystems dedicated to broad features. Inside each subsystem are additional layers of grouped features, which become more granular at every lower level until the final unit-level is reached. The boxes located near the top of the figure are Data Store Memory blocks that are used to hold system-level data such as the clock time and audio settings (treble, bass, balance, fade, etc).

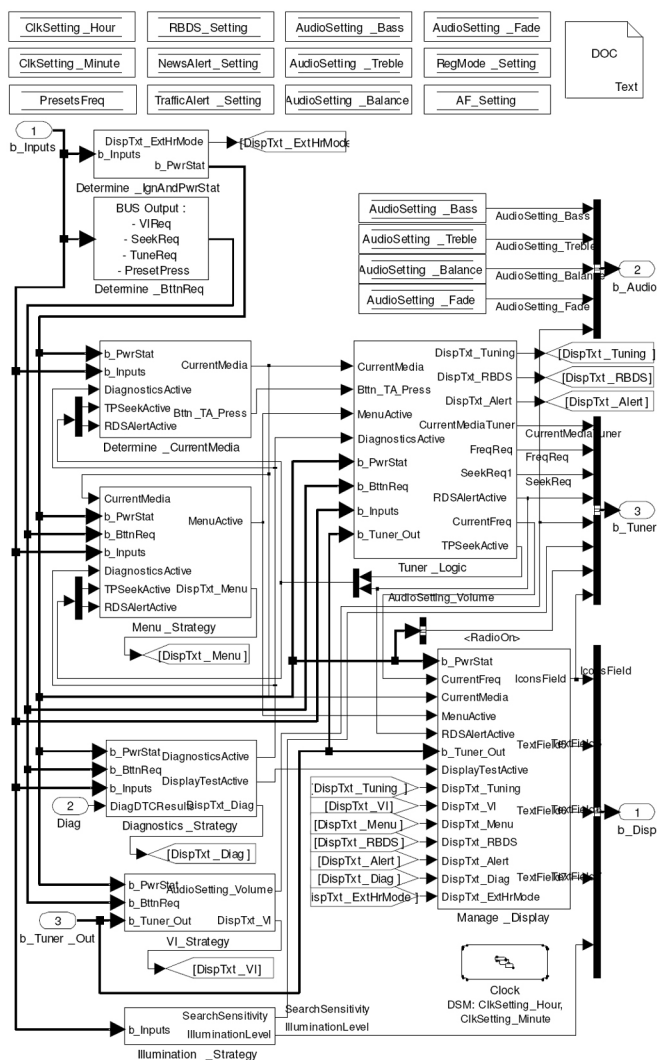


Figure 2. System-level view of radio model.

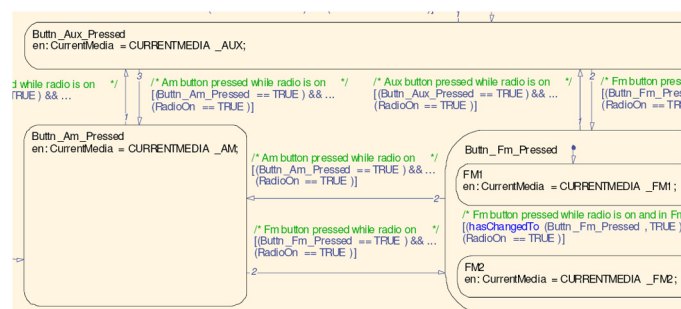


Figure 3. Example Stateflow Chart.

In addition to the Simulink model, a MATLAB script is used to populate the model workspace with parameter and configuration data. Since no data dictionary was included with the radio specification, this data was derived manually from the requirements specification.

TRACEABILITY

To provide traceability between the requirements and model, each model subsystem includes a documentation block that describes the function of the subsystem and lists the satisfied requirements. The lower-levels of the model are also amply commented to document design intent and enhance the user's comprehension. While we would like to provide explicit traceability of each requirement to the lowest levels of the model, the radio specification only contains high-level requirements that are more logically traced to the subsystem level. An example of a typical requirement could be to tune the radio to the AM band based on a button press. This requirement involves first analyzing a button press input signal and deciding whether the radio is in a valid state to change the radio source to AM. If the change is valid, the volume is temporarily muted while the source band is changed, the last AM frequency is recalled and tuned, and the display is updated. This requirement is best mapped and traced to a subsystem level.

TESTING

The model is continuously tested throughout the development process both at the unit-level and system-level. Testing at the unit-level occurs both informally and formally as requirements are modeled. For formal unit-level testing, the target block is isolated and connected to a test harness. Block input and expected output signals make up the test cases and are defined in an MS Excel spreadsheet. A MATLAB script is then used to translate test cases to a Simulink Signal Builder block that is connected to the block under test. Upon execution of the test, the actual versus expected results are reviewed for accuracy using a Simulink Scope block. Additionally, the built-in Simulink Verification and Validation tool is utilized to provide a report on decision, condition, and MC/DC coverage for the block under test.