Model-based techniques for space microcontroller applications

Steve Duncan, Thales Alenia Space UK Ltd

Mixed-signal microcontrollers have become dominant in terrestrial electronics applications because of their low cost, high integration and rapid development cycle. The recent advent of rad-hard and rad-tolerant devices with onboard analogue interfacing has started a similar revolution in the space domain, with potential reductions in component count, board space, power consumption, harness mass and, above all, cost.

Typical microcontroller application programs are simple, deterministic and repetitive, which makes them simple to analyse and relatively straightforward to develop. However, they are often deployed as part of a larger distributed system, in which case a substantial proportion of the behavioural complexity arises from the interaction between the nodes rather than the nodes themselves. This is especially true for bus-based command and control protocols, particularly where the communications medium is not perfectly reliable and there is a possibility that messages between nodes may be lost or corrupted.

The management of emergent complexity is an important part of distributed system design. Experience has shown that despite high test coverage, it is still possible for systems to contain latent faults that cause an unrecoverable state when confronted with rare but unfortunate events, for example the permutation of the address field in a message to an incorrect but legal value. A very desirable goal would be the ability to prove formally that the system is free from such defects, i.e. it will always recover itself to a known state following any sort of upset.

Conventionally, the dynamic interactions between nodes in a system are designed using Message Sequence Charts. These diagrams are good for capturing stationary sequences (the "happy path") but tend to become unmanageable when branching due to nondeterministic decisions or message errors is included.

SDL, as used in ESA's TASTE toolchain, is a formal language for specifying and modelling the behaviour of systems. It was developed by an ITU working group in the 1970s and was widely used in the design of call control schemes for circuit-switched landline and mobile telephony. SDL represents a distributed system as a collection of interacting finite state machines (FSMs). It has a graphical from that lends itself readily to visual design capture and analysis, and a textual form that can be compiled into an executable. Both representations are equivalent and each may be readily transformed into the other.

SDL encourages the development of predictable systems by allowing key simplifying constraints on the programming model to be enforced:

- Execution occurs only during state transitions
- The execution path through any state transition is acyclic.
- An FSM may only affect another through tightly constrained points of interaction.

In this presentation, we will describe the results of our research into the model-based specification, design and validation of microcontroller-based subsystems using the TASTE toolset. We cover the following topics:

Behavioural Design

The decomposition of the system into SDL Finite State Machines is described.

Data Modelling

The modelling of data structures in ASN.1 is described, and the application of the data model in the generation of secondary representations (e.g. object dictionary, Electronic Datasheets, spacecraft database) is discussed. Particular attention is paid to the separation of system and protocol software from application, leading to a configurable model that can be easily reused for new applications and targets.

Model Validation

The construction of an executable simulation model is described, including the modelling of abstracted system components related to the interacting elements (e.g. data buses, application software). Validation of the system within this model is discussed. Specifically, the use of scripting and other automated techniques to explore and/or enumerate the state space of the system.

Code Generation

The automatic generation of code for heterogeneous systems of 32-bit and 16-bit processors is described, together with the necessary procedures for ensuring correct encoding from ASN.1 representations to legacy packet structures. We then show how it is possible to generate memory-efficient data encodings that are compatible with the small program memory spaces generally available on microcontrollers.

Subsystem Integration

The incremental integration of a system is demonstrated, whereby the TASTE model is replaced, one component at a time, with real subsystems, in order to validate the implementation against the model. The possibility for the TASTE tool to become the basis of the EGSE is discussed.

Case Studies

Finally, we present the results of three breadboard projects developed using the MBSE approach with TASTE:

- A CANbus Backplane with heterogeneous nodes based on the Thales Alenia Space DPC microcontroller and the Cobham Gaisler GR712,
- A distributed thermal control System based on the Cobham Gaisler GR716,
- A microcontroller-based Rate Gyro using the Microchip SAMV71.