Pi Innovo

Vehicle electronics innovators

Optimised Ground Vehicle Electronics Development

Pi Innovo
Vehicle electronics innovators

Optimised Ground Vehicle Electronics Development

Reference:	P0012xxxx	**Issue Date:**	**11/09/2010**
Status:	Confidential OR ©Copyright 2012		

Author:	David Price
Email:	David.price@pi-innovo.com
Telephone:	
Address:	Pi Innovo Ltd. Suite 1 The Old Grannary Westwick Oakington CB24 3AR UK

Prepared for

Client:

Recipient:

Distribution

Pi Innovo:

Client:

Revision History

Issue Date	Comments

Document control

$LastChangedBy:: agreenhill $

$LastChangedDate:: 2012-03-19 #$

$LastChangedRevision:: 2633+ $

$URL:: https://apple.ps.local/svn/Marketing/Templates/Pi Innovo Unapproved/Report US.dotx $

$Templateinfo:: $

Document control

$LastChangedBy:: agreenhill $

$LastChangedDate:: 2012-03-19 #$

$LastChangedRevision:: 2633+ $

$URL:: https://apple.ps.local/svn/Marketing/Templates/Pi Innovo Unapproved/Report US.dotx $

$Templateinfo:: $

Contents

1. Introduction and Scope **5**

2. Abstract **5**

3. Background **5**

3.1 Control System Development Tools 6

4. Open Architecture Control System Development **7**

4.1 Transferring from Prototype to Production 9

5. Development Process **9**

5.1 Electronics Hardware Development 9

5.2 Software Development 11

6. Testing and Usage **14**

6.1 Hardware Testing 14

6.2 Software Testing 14

6.3 System Testing 15

6.4 In-Vehicle Support 16

1. Introduction and Scope

This paper will discuss some of the approaches available when developing new electronic control systems. The automotive industry has a long track record of developing high integrity control systems and applying them quickly and efficiently to production vehicles. The OpenECU family of controllers has grown out of the automotive domain and has recently been applied to the first few military control systems in the areas of suspension control. This paper will highlight the perceived advantages of the OpenECU approach over other common approaches and will discuss the engineering development process which has led us to this point. We will consider both the software and the electronic hardware aspects of the controller.

2. Abstract

The modern military ground vehicle has an array of electronics including sensors, displays, control systems, and communications. Surveying currently fielded electronic packages results in multiple approaches to the same problem. This paper seeks to describe an open-architecture ruggedized electronics solution that covers the entire development cycle including; hardware, development toolchain, software, and field support. The motivation is to provide electronic standardisation, decreased form factors, and promote re-use of electronics.

This approach was used on two separate electronics development programs and resulted in a single common development toolchain for the developers and two hardware solutions. These two ECUs were able to provide wide ranging applications from, diesel engine control, semi-active suspension control, ABS/TCS/ESC control, and even active aerodynamics for a European supercar. The method used to provide this degree of flexibility while providing a lean development environment is discussed in this paper.

3. Background

New electronic controls and systems are usually developed from earlier versions of themselves or from a blank sheet of paper (in the case of novel technology systems). There is a tendency for electronic controllers to evolve in isolation from other vehicle control systems. The use of prototyping equipment is also traditionally tied in to one domain area. The net result is the use of different electronic controllers to solve what might at first glance appear to be different system problems. However, a deeper investigation reveals that the I/O and processing requirements may well have a lot in common. In particular, during the early-mid development phases of new control system, when more flexibility is required, the overlap of I/O requirements becomes more obvious. Once the product moves towards production and some optimisation is applied, the apparent differences can mask the underlying truth.

This paper will look at an alternative approach of using open, generic controllers which can be applied to a number of domains. The background for this approach has been developed in the commercial automotive domain and is now being applied to the military electronics domain. A number of examples discussed below are therefore taken from commercial electronic systems. Pi Shurlok has explored this space with the OpenECU concept. One example of which has allowed the same base electronics controller to be used for:

Variable suspension damping control

Semi-active suspension control

Electric vehicle supervisor ECU

Hydro-carbon injection for diesel aftertreatment

Selective catalytic reduction for diesel aftertreatment

Vehicle stability control

Common rail diesel engine control

3.1 Control System Development Tools

There have been two standard ways of developing new electronic control systems in the past. The first is to employ rapid prototyping equipment and the second is to use an existing controller and to work from there.

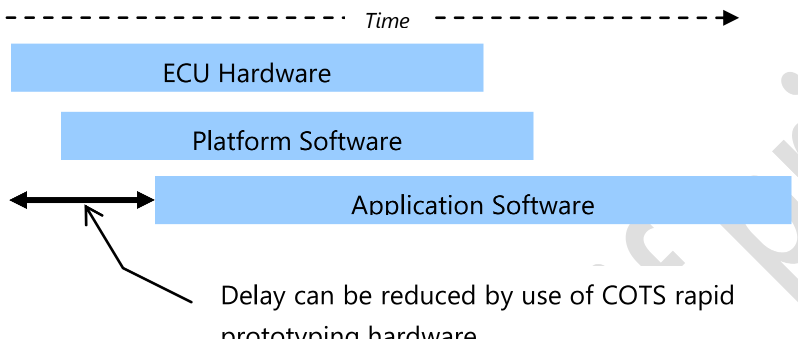
The advantages of the first approach are the flexibility of the electronic controls and the powerful processing resources that can be made available. The advantages of the second approach are the low initial cost and the robustness of the electronics when taken for field testing. One disadvantage that both approaches suffer is that they will both require a significant amount of re-engineering to become a final production solution.

Based on our own experiences, we believe the costs of developing new systems can be split down as follows:

Application Software	40%	
Platform Software	30%	Ideally, this should be re-usable
ECU Hardware	30%	

The split here is for an "average" control system developed from scratch. For complex control systems, the application software may take a larger percentage share of the effort to develop.

The timing (Gantt chart) will look approximately like this

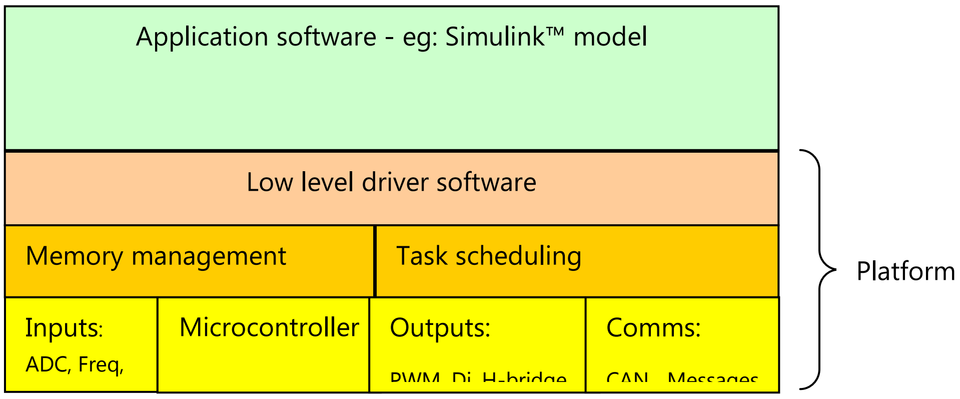


Although the use of COTS rapid prototyping hardware will speed up the application development, it will need additional time and effort to re-engineer the system into a formal product. The ideal solution is a system which is available immediately and has sufficient capability for prototyping, but also has a clear path forward to higher volumes.

The OpenECU approach takes the middle ground, with a good level of flexibility, processing power and robustness together with an increasingly well trodden path from prototyping to production. The development and use of these tools for new control applications is described in this paper.

4. Open Architecture Control System Development

One key aspect of all these development tools is the use of an open control system architecture. The control system development is allowed to take place in high level modelling language tools. The details of the electronic hardware and low level software are abstracted away so that the developer can get on with the job in hand; that of building the control system. A common way of viewing the software is shown below. The final software is a mix of auto-generated application code and well-proven low level driver code.



There are a number of ways of achieving this kind of arrangement. One option may be to use rapid prototyping tools which have a PC core with additional hardware to provide the physical connections into the actual control system. The PC-104 approach or Compact PCI form factors allow a large variety of control systems to be assembled from off-the-shelf components. Some companies include the software interfaces with their own flavour of hardware. Tool-chains such as those from National Instruments and dSPACE fit this category. Another option is to use some volume production control system hardware and purpose-build the additional interfaces required. In this case, the integration of low level software with high level development tools (such as Simulink™) requires some specialised software skills.

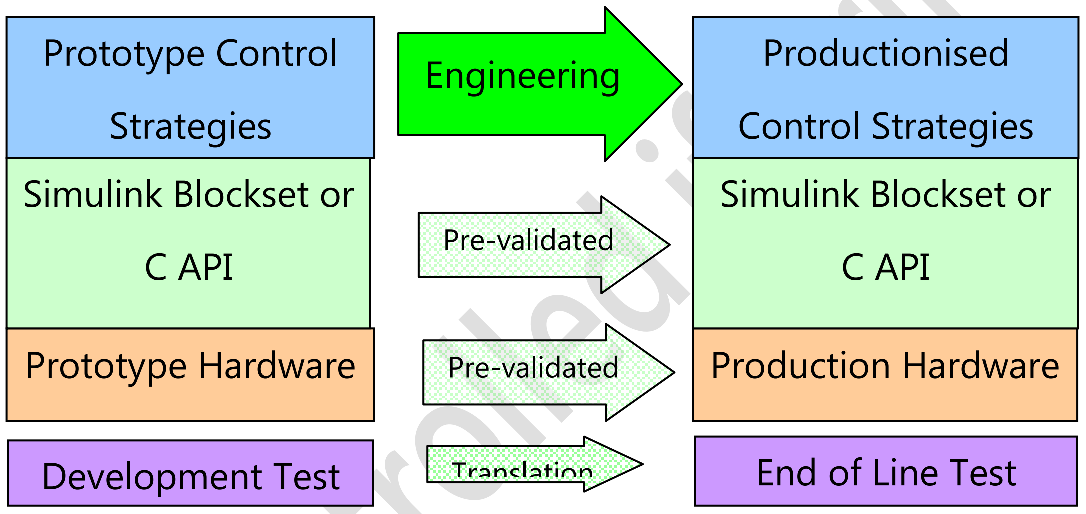
The OpenECU platform architecture is designed to blend these approaches and to provide an easy-to-use interface to both C-code and Simulink™ models that work well in prototyping sit

PC-104 stack housing and M461 OpenECU.

4.1 Transferring from Prototype to Production

In the automotive world, where production volumes are typically significantly higher than military applications, there is usually a clear division between the development phase and the cost optimised production phase. In order to bring a product to market quickly and efficiently, it is essential that right from the outset, the team understands the route that will be followed in order to turn the prototyping system into a production one. This route will almost inevitably involve some re-engineering of the prototype system. The challenge is to minimise the re-work that will be required, while still allowing sufficient flexibility and capability for the prototyping work to be performed without undue constraints.

This translation from the prototyping world to a production one can often be a stumbling block where numerous unseen difficulties arise. The OpenECU design philosophy makes this path as smooth as possible. This is done by pre-validating much of the platform and significant design re-use between OpenECU products in the family.



The remainder of this paper will explain how the development of the OpenECU platform has been performed in such a way as to provide this smooth transition from prototyping into production.

5. Development Process

5.1 Electronics Hardware Development

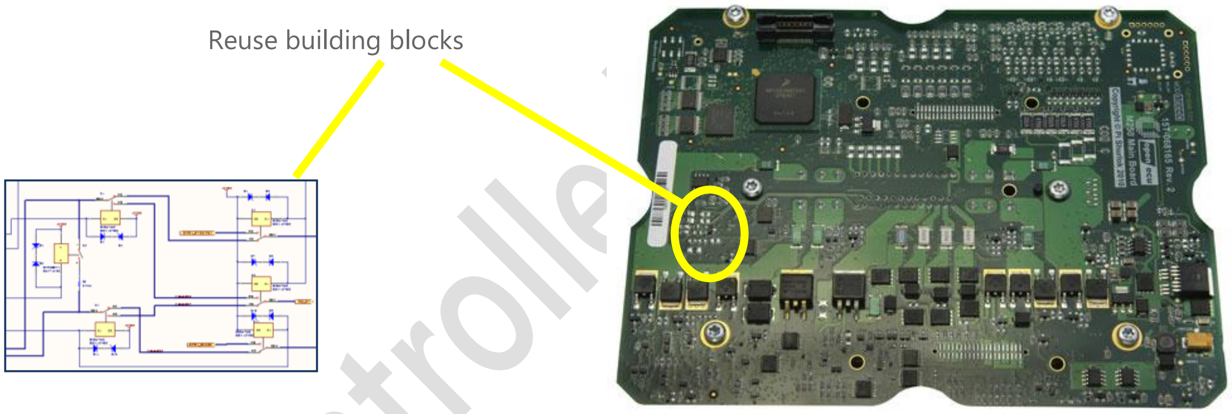
As discussed above, a key aim is to re-use electronics hardware as much as possible. At first glance, this might appear to be very difficult, given that everything from the size and shape to the range of I/O will vary through a family of controllers. The re-use is achieved by designing common blocks of electronic circuits. These are treated as "building blocks" for assembling a complete set of circuit schematic designs.

The re-use continues beyond the circuit schematic design and uses exactly the same components in each circuit block and also the same circuit layout on the PCB. This provides several further savings as well as having confidence that the circuit design will work:

PCB layout is already designed, tested for EMC & manufacturability

Reuse parts in manufacturing - no need to select or qualify additional parts to automotive or military standards.

The ability to work with smaller elemental "building blocks" means that the final product can be much more tightly integrated. For example by removing duplicated circuits and reducing power supplies. The integrated electronic design allows the mechanical design to fit into a much smaller form factor. This smaller form factor can be achieved without sacrificing the flexibility of other open systems (such as PC-104). The more compact design inevitably draws less current and therefore requires less heat dissipation than other systems. Flexibility is maintained by defining an interface and form factor standard for daughter cards that can be later added within the same enclosure. The single enclosure, without the rigidity requirements of a chassis to support many cards makes the weight reduction very significant when compared to alternatives.



As an example of circuit re-use, the circuits designed for driving injectors in engine control applications have been found to be equally suitable for controlling hydraulic valves in suspension applications. Similar speed of response and power requirements have meant that the same design can be re-used, building on the experience gained in earlier applications and speeding up deployment of the system.

It is difficult to quantify the size/weight reductions when comparing OpenECU to (say) PC-104, as two such different systems rarely offer exactly the same functionality. However, the reductions in size/weight are in the order of 50% or more.

The tools used during the hardware design also contribute towards this philosophy, as they include a library system for both circuit blocks and components. The component libraries include all the purchasing/cost information for the components as well as the supplier's data sheets and other procurement data.

Having the cost data built into the design tool allows the designers to very quickly work through the alternatives to ensure that they are designing the lowest cost solution.

The "digital core" is one of the major circuit blocks containing the microcontroller(s) and associated memory chips, crystals etc. Re-use of this core block together will much of the I/O assignment also promotes more software re-use. Freescale PowerPC microcontrollershave been selected as the standard choice for the OpenECU family. This family of microcontrollers is one of the most popular within the automotive environment, having peripheral options on board to suit a very wide variety of applications.

The use of a microcontroller (compared to a microprocessor) is commonplace in the automotive industry. The demands of size, weight and I/O peripherals are balanced against the raw processing power. However, the on-chip integration significantly speeds up peripheral access that out-weighs any benefits from an apparently "faster" separate microprocessor core.

OpenECU is designed for the, somewhat hostile, automotive under-bonnet environment. There is a wide operational temperature range of -40 to 125 degC; high levels of vibration and; the unit must be sealed from fluid ingress. The automotive environment transfers well to the military vehicle environments and so allows for some commonality in the housing and connector choices.

5.2 Software Development

As discussed above, the optimisation gained from using pre-validated circuit building blocks allows for a significantly smaller size and weight of controller to be developed very quickly when compared to assembling a system from off-the-shelf PC-104 components. However, the advantages of smaller and lighter hardware can be lost if the embedded software is dauntingly complex or has to be re-written each time there is a slight variation in the hardware. The OpenECU software has deliberately been designed to be as easy-to-use as possible.

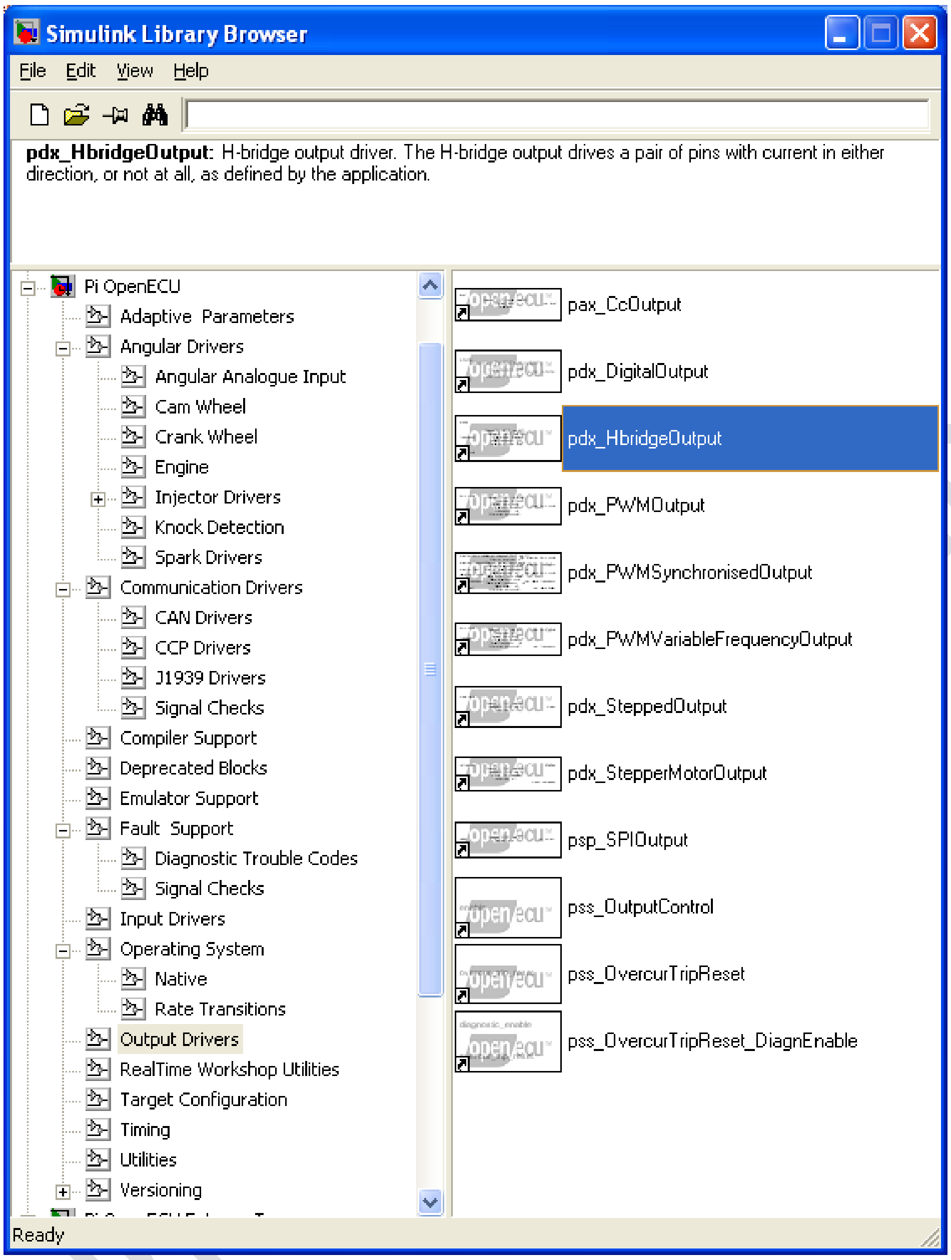
Note: the software product under discussion here is Developer Platform Sim32[1]. Developer Platform Sim32 contains a blockset to allow development within the Simulink model based development environment plus the embedded software libraries and build process required to automatically produce an embedded software image that can be downloaded to OpenECU hardware.

To make the whole system easy to use, significant functionality is "hidden" underneath the "drag & drop" Simulink blocks. The OpenECU software library is added to the standard Simulink library as shown below.

The method allows the developer to connect their control model to the outside world in a way that takes care of details such as:

- Correct initialisation of processor ports (especially multi-function I/O)
- Configuring alternative I/O options (e.g. H-bridges -> single sided outputs)
- Signal routing to peripheral accessory ICs
- Monitoring of status (for outputs)
- Resetting of outputs following overheat or s/c conditions
- Timing of data updates
- Diagnostic fault setting

[1] This is the version including the Simulink™ blockset. An alternative version operates with a C API.

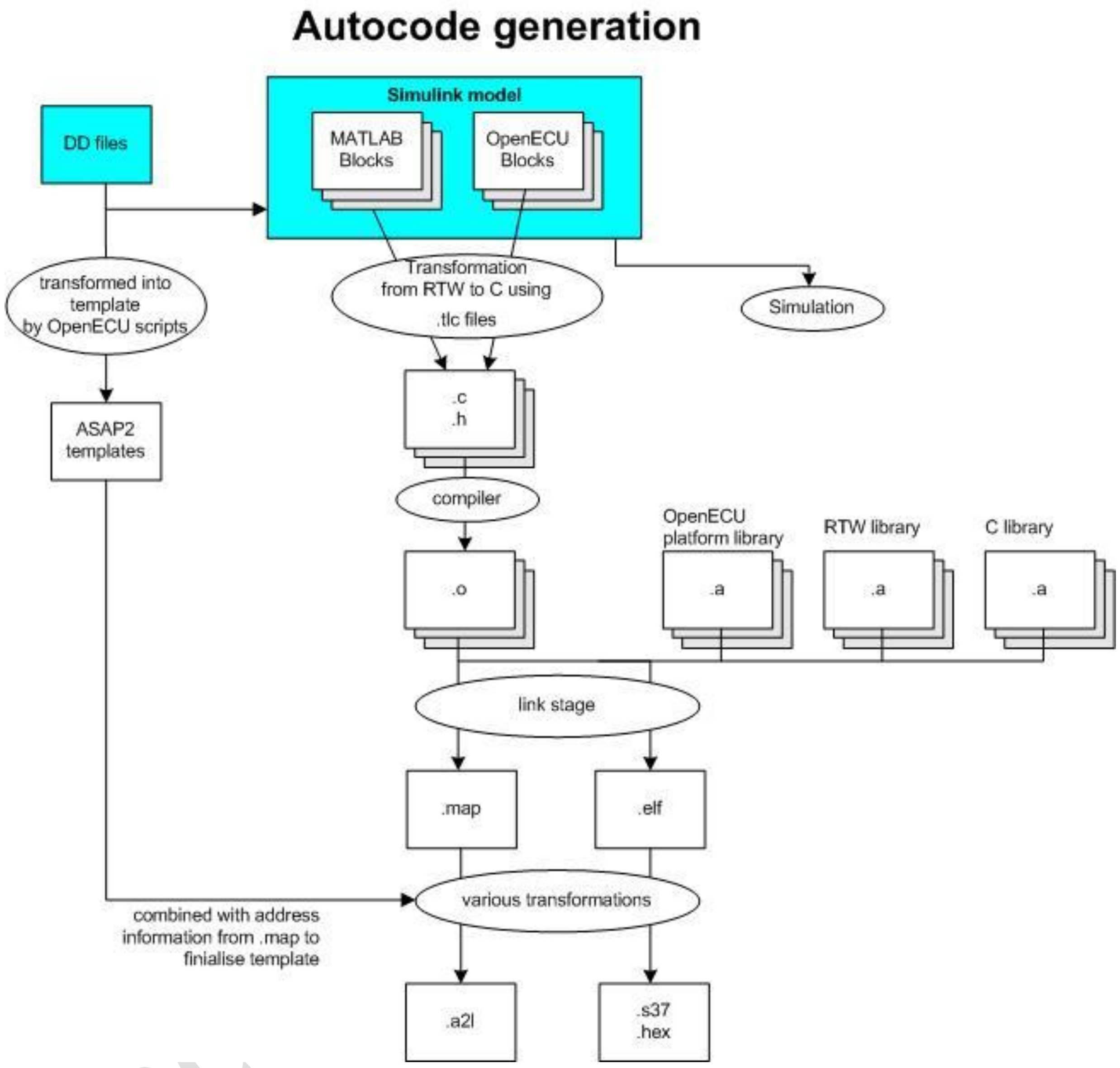

Simulink Library Browser

pdx_HbridgeOutput: H-bridge output driver. The H-bridge output drives a pair of pins with current in either direction, or not at all, as defined by the application.

The platform software that is built into the executable file has been developed & tested according to the safety standard, MISRA SIL 2. This demands a high degree of rigour in the development process to ensure that the platform software will be suitable for use in high volume safety-related applications up to the SIL2 Safety Integrity Level.

The software build process has also been carefully constructed to ensure ease of use from the Simulink (or 'C') environment. The software build tools pull together all the relevant files that are needed to support the model in question. They link the files together and invoke the specified compiler to produce an executable file which can then be downloaded to the target ECU. The build process also produces files required by calibration tools for in-vehicle use.

This flow chart shows the complex process which takes place in order to build the software. This happens automatically when the user presses "Ctrl-B". The user only has to work with the two coloured files at the top of the process. The rest are handled by the tool.



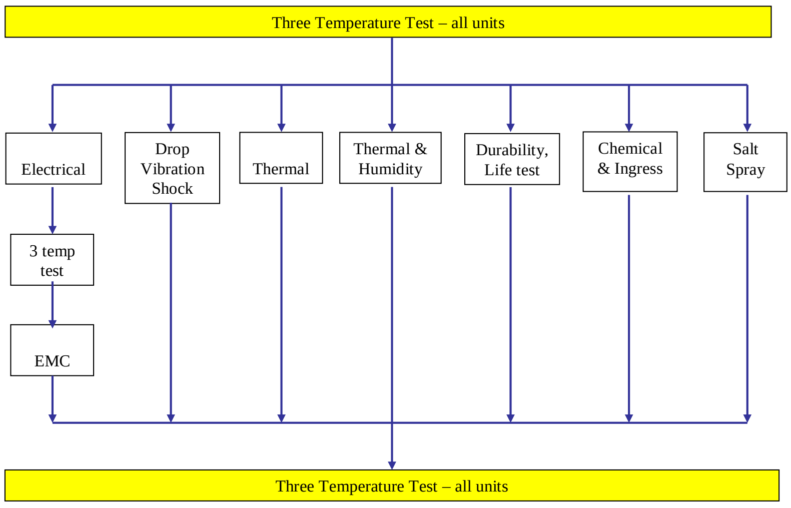
The use of autocode generation has become widely accepted in recent years for safety-related systems. Controlling the output of the code generator and ensuring testability of the resulting code are key safety factors. Style guides and automated checking tools for the models ensure that particular model constructs are avoided and that the code produced does not breach guidelines for a restricted subset of 'C'.

6. Testing and Usage

6.1 Hardware Testing

Hardware testing is carried out to various automotive and military environmental standards. This is conducted according to a "Design Validation" plan which defines an array of tests and functional verifications applied to a batch of units after each stage.

A typical DV plan might look like this:



A number of ECUs will be subjected to the tests on each "leg" and functionally verified before continuing to the next test.

Once designs have been through this level of rigour, any new circuit building blocks become available for re-use in other designs. This ensures that the library of building blocks grows to accommodate new features.

6.2 Software Testing

To test the whole software application, attention must be given to the interfaces between the application and the driver software. Normal practice is to take the results from simulation testing and re-use the results as "expected outcomes" for subsequent testing on the final target hardware. However, the simulations are inevitably split up to different sub-systems for ease of use. Therefore, when testing across the boundaries on the final system, there will be no data to be used as the "expected results". This makes the integration testing (which is always a key stage) even more important. The use of the same interfaces and boundaries when moving from prototyping to production can make this phase of testing run much more smoothly in practice.

For each platform software component, the process follows a traditional development life-cycle, with a set of requirements, a design, the code, integration testing against the design and formal testing against the requirements. There is traceability from the requirements through to the code and the test results. Each step is formally peer-reviewed and the whole system lives under a configuration management system. It should be emphasised that this really is configuration management and not just version control, as there are multiple hardware platforms and multiple Simulink versions to be supported.

6.3 System Testing

The most obvious form of system testing is to kit out a prototype vehicle and then drive it through a series of tests to check the system performance. However, this can be costly and time consuming and can also be very difficult to perform what may be potentially hazardous or destructive tests. Under these circumstances a Hardware-in-the-Loop (HIL) test rig can prove to be invaluable.

Typical HIL equipment - showing flexibility for additional cards to simulate a variety of I/O

The HIL test rig simulates the rest of the vehicle, or at least those parts which are necessary to convince the control system that it is operating a real vehicle. The HIL test rig allows the user to force the control system through all the conditions which will be encountered in the real world. With a HIL rig, it is easy to produce conditions which are difficult to create or may only rarely be encountered. An example of an extreme event would be very high speed driving in very cold conditions.

Another great advantage of a HIL test rig is the ability to automate testing. It is usual to start by automating a standard set of basic functional tests. This will then form the core of a set of regression tests to be performed at regular intervals (usually when new functions are added or customer release is made). The advantage here is not only that testing can be performed on unmanned rigs overnight or at weekends, but also that regression testing tends to be very repetitive and humans are usually poor at repetitive tasks. Automating this task helps to ensure that errors won't be missed if the test operator has been used to seeing them pass in previous versions.

6.4 In-Vehicle Support

The use of pc-based "calibration" tools is prevalent in the automotive world. There are communications standards to define the interface to these tools. Some companies develop their own bespoke calibration systems. Pi Shurlok have observed that many engineers don't like to change and prefer to remain with the same calibration tools. Therefore Pi Shurlok supports the common standard (CCP - CAN Calibration Protocol) and allows customer engineers to choose one of the commercially available calibration tools.