Eclipse and Embedded Software Development

Aurangzeb
Mentor Graphics, Lahore, Pakistan
aurangzeb@mentor.com

Abstract

Embedded industry comprises of embedded system vendors, real time operating system manufacturers, embedded application developers and tool vendors for embedded software development. It is a mature industry and has a history. There have been major companies competing with each other in all aforementioned domains. Powerful and excellent proprietary tools for embedded software development have been created by the competing companies. But the emergence of an open source tools integration platform called Eclipse caused many embedded tool vendors to come to a common platform by adopting Eclipse because of its rich offerings and potential. Embedded software development is somewhat different from traditional software development and has some extra requirements. Many embedded tool vendors have joined the Eclipse Foundation and are working together to make Eclipse address their specific needs. In this paper we study the offerings of Eclipse and the successful initiatives it has taken to be an attraction as a universal platform for embedded software development tools.

1. Introduction

Eclipse is an open source IDE which has influenced embedded software development market like many other areas. It is enabling embedded tool vendors to integrate their proprietary tools into Eclipse taking full advantage of the rich IDE features that it offers, and is becoming a common environment for development of embedded software. In section 2 we describe how embedded industry needed a tool integration platform. In section 3 we describe Eclipse and how it became a successful tool integration platform. Section 4 and 5 deal with the IDE features and frameworks of Eclipse that attracted embedded tool vendors and made them embrace Eclipse. Section 6 describes how embedded software development is different from traditional software development and how embedded community of Eclipse ecosystem is working together to making Eclipse address their specific needs and be a de-facto standard for hosting embedded tools. Section 7 briefly describes our product as a successful example of use of Eclipse in embedded software development tools. Section 8 talks about future of Eclipse in embedded industry and section 9 concludes this paper.

2. Need for a tool integration platform in embedded industry

An embedded system is a special purpose computer system with specialized hardware and software designed to perform dedicated functions, often with real time computing constraints. Embedded systems have become integral part of our lives and are applied in a wide range of areas including consumer electronic devices (digital cameras, PDA’s, mobile phones, mp3 players, DVD players), network equipment (routers, switches), medical devices (electronic stethoscopes, blood-test machines, medical imaging), automotive applications (Electronic stability control, anti-lock braking system, engine controllers), industrial devices (process instrumentation, robots) and military applications (missiles, combat systems). Embedded systems require real time operating systems (RTOS) often with hard real time constraints to function properly in their domains.

Embedded industry is a vast industry comprising embedded system vendors, RTOS manufacturers, embedded application developers and tool vendors for embedded software development. Embedded software development has extra requirements as compared to traditional software development because embedded software has to run on a target different than the host development machine. Therefore target configuration, target management, cross-compilation and ability to debug remotely become necessary. Specific requirements of the areas where embedded systems are applied have resulted in emergence of different processor architectures and real time operating systems. Wide range of processors, development host systems, RTOSes and application specific requirements drove RTOS vendors to have their
3. Eclipse – a successful tool integration platform

Eclipse is an open source integrated development environment written in Java. IBM originally started this project in 1998 to build a common platform for IBM tools, bringing developers to Java based middleware, and to establish a partner ecosystem parallel to Microsoft ecosystem which at that time enjoyed complete dominance in the market. It was decided to make an extensible platform for tools integration and build a great Java IDE to attract developers and companies. In 2001 IBM made Eclipse an open source product and established Eclipse consortium with 9 members initially. Release of Eclipse 2.0 and 2.1 drove widespread adoption of Eclipse as a suitable Java IDE and tools integration platform and companies started adopting Eclipse for their tools integration and development. Eclipse enchanted developers and companies by its robust and extensible architecture, rich features, and potential for an ideal tools integration platform and more companies started joining Open Source Eclipse community and contributing in the development and improvement of Eclipse. In 2004 Eclipse Foundation, a not-for-profit corporation was formed and companies which were still reluctant to adopt and/or contribute in Eclipse considering it an IBM lead product started joining the Foundation as now it had become a vendor-neutral open ecosystem around royalty-free technology and a universal platform for tools integration, providing developers ultimate freedom in a multi-platform, multi-language and multi-vendor environment.

What makes Eclipse different from general open source products and what resulted in immense adoption of Eclipse among commercial companies is Eclipse Public License (EPL) [2]. This license makes Eclipse a royalty-free open source technology, gives redistribution rights to everyone, and allows companies to write proprietary plug-in extensions to Eclipse and sell the resulting application as a commercial product. This has resulted in competing vendors to make use of the common platform collaborating in the open extensible frameworks that bring standardization to tools solutions but at the same time maintaining their non-open intellectual property and ability to differentiate on features and implementations and compete on core areas.

Currently Eclipse targets a wide range of areas including web tools, business reporting, embedded systems, testing and profiling, modeling etc. Many companies belonging to these areas have joined the community and are joining hands to address specific needs of their domain. Consequently, Eclipse that was once a technology driven environment has now become a community driven environment.

4. Eclipse IDE offerings and frameworks

Eclipse provides an extensible framework for building integrated development environments. Among core constituents of this framework are project management and build systems, source code browsing and editing facilities, debug framework, and team support and version control system.

4.1. Project management

Eclipse platform provides a rich and flexible support for resource management. Tools work on projects that are resources in the workspace and contain files and folders. The framework supports basic resource management and provides resource change notifications like creation, deletion and changes in contents, to tools. Resources can be imported and exported from the workspace. Workspace provides a marker mechanism to annotate resources in the GUI. It also provides mechanism to associate custom data with resources which can be persist-able, and keeps a history of changes in the contents of files enabling users to revert changes.

Framework provides views for resource navigation which allow basic resource operations like copy, paste, delete, and support filtering, sorting and searching options. Notion of working sets is introduced to have a customized look of the workspace.

Projects have associated natures which tag them with tools and help builder associations with projects. Eclipse supports incremental builds such that only those resources are built that have been changed since the last build. Auto, full and clean builds are also supported. Users can perform the build and clean operations from the workbench, enable or disable builders, and also provide custom builders.
4.2. Source editors

Eclipse provides an extensible source editor with all general editing features like copy, paste, select all, redo, undo etc and allows clients to enrich it with language aware features such as support of syntax highlighting, showing code structure in the outline view, providing content sensitive help, and ability to view information about program elements or values of variables while debugging programs, in tool-tips. Contents of the source files can be compared and replaced from local history. Eclipse provides support for contributing new editors and enriching them with language-aware features.

4.3. Debug framework

Eclipse being a platform for building integrated tools supports integrated debuggers. It has a powerful debug framework which makes building integrated debuggers in Eclipse easier. Some excellent debuggers like Java debugger (JDT), IBM’s Websphere Application Server Debug, JSP™ Debug, Compile Language Debug, JavaScript™ Debug, DB2 Stored Procedure Debug, EGL Debug, etc have been built on robust debug framework of Eclipse. This framework is composed of following major frameworks and facilities.

- Debug model
  - Launch framework
  - Breakpoint management
  - Source Lookup framework
  - I/O console
  - Debug user interface

4.3.1 Debug model

Eclipse debug framework provides abstractions for entities common to execution environments like processes, threads, stack frames, variables, registers, breakpoints etc, and functions like stepping, suspending, terminating, resuming etc, and a user interface that works against the abstractions and functions leaving the implementation of abstractions and functions on the clients for their respective debugger. Debug model provides support for stepping, resuming, suspending, terminating etc. It defines debug events that describe the changes or happenings in executing process or target. User interface requires that debug model elements and process implementations generate debug events. Debug windows register themselves as listeners to events in the debug model and update accordingly. It also defines interfaces to provide labels and images for debug elements to be shown in the Debug UI.

4.3.2 Launch framework

Launch framework provides facilities for spawning an OS process, persisting information about how a process is launched, a user interface and framework to facilitate editing of launch parameters, an extensible set of launch modes (run, debug, profile, etc) and selection sensitive actions for launching.

4.3.3 Breakpoint management

Eclipse breakpoint framework provides facilities for adding and removing breakpoints, providing change notifications, persisting breakpoints across workbench invocations, temporarily skipping breakpoints, creating breakpoints from retarget-able actions, and exporting/importing breakpoints to/from a file. Platform provides an extension point for contributing kinds of breakpoints, and global retarget-able actions for creating common breakpoints by double clicking on the editor ruler.

4.3.4 Source lookup

Platform provides support to find source code for suspended binary location when execution is suspended and display and highlight that code in the source code editor. Text editors position themselves to stack frame line numbers, highlight statements or lines, distinguish between top and non-top stack frames and remove annotations upon thread resume/terminate. Debug platform provides support to override default instruction pointers displayed in editor rulers.

4.3.5 I/O console

Console framework manages multiple console windows and provides support for automatically attaching consoles with processes. A console is an input-output window which provides standard input, output and error streams to a process. It is like an operating system text shell but can be extended to support any kind of I/O and not just text. It supports regular expression matching and hyper-linking.

4.3.6 Debugger user interface

Platform provides a number of views that constitute user interface of the debugger. These views work on the abstractions and are fully functional. Clients can use them without any trouble. Platform has, however, kept them extensible and user can always provide hooks, control look and feel, and extend the functionality. Among these views are Debug, Breakpoints, Variables, Expressions, Memory, and Registers. Debugger vendors can also provide their
own custom debug views and perspectives, and contribute actions to UI components benefiting from Eclipse’s easily extendable architecture.

4.4. Modeling framework

Eclipse has a powerful modeling framework which adds value to it. It has support for providing application or data model in the form of XML schema, UML diagrams and Java interfaces. It supports code generation, regeneration and intelligent merge. Model description providers are supported to efficiently and easily display models in GUI and similarly editors for model data are provided.

4.5. Help

Eclipse has a powerful help system and tools can contribute their help pages and books following the procedures laid out by Eclipse.

4.6. Version management and team support

Eclipse furnishes team support by allowing projects to be placed under version and configuration management. Platform supports all industry standard source code versioning systems like and also provides extension points and API that allow new team repositories to be plugged into Eclipse.

These facilities and frameworks, the extensibility provided through plug-in mechanism and other features make Eclipse an ideal choice for tools integration having a distinctive edge on other tool integration platforms for software development [3].

Eclipse has built upon the provided frameworks to make an excellent IDE for Java development (JDT) which is most popular among the Java developers. Success and popularity of JDT which is actually the effective utilization of underlying extensible and powerful frameworks to build excellent tools has gained Eclipse a place in the industry and it is easily becoming a de-facto standard for tools integration [4, 5].

5. Eclipse into the embedded world

Eclipse was originally aimed at targeting Java desktop and enterprise applications. But its robust and extensible architecture and rich offerings as ideal tools integration platform did not let embedded industry ignore Eclipse and its potential as an excellent platform to build embedded software development environments. Therefore, they started embracing Eclipse and building their proprietary tools on Eclipse. Using Eclipse is helpful for both the vendors and the developers. Vendors do not have to worry about the IDE features; they get excellent features for free. So they can concentrate on their core areas to produce quality tools and developers now have a generic platform and look and feel, and can change processor architectures and RTOSEs without having to learn a new environment. It makes easier for them to move between embedded OS and target platforms [6]. Since every feature in Eclipse is contributed by plug-ins and there are myriad of open source and commercial plug-ins available, developers can always plug such plug-ins into their Eclipse and have extra features to offer themselves a rich development experience.

Embedded software is developed mainly in C and C++ which were not supported by Eclipse in the beginning. When embedded tool vendors embraced Eclipse and started integrating their tools in Eclipse, some started providing their own editor, project manager and debugger for C/C++ while others started contributing in the open source Eclipse project for building C/C++ development environment (CDT) like the famous JDT, lead by QNX Software Systems, an embedded tool vendor. CDT provides C/C++ project management, build systems and debugger based on GNU tool chains. CDT4.0 has an improved internal builder and indexer, support for pre-built index information and framework for project generation [7]. It provides rich editing experience of C/C++ source code by facilitating syntax highlighting, content assist, refactoring, code outline, code templates, formatting, searching etc.

CDT supports GNU tool chain, a set of free tools distributed under GNU Public License (GPL) and making a complete toolset for building and debugging embedded applications.

CDT with GNU tools makes a functional embedded cross-development system. What it lacks is that it does not support remote target connections to embedded processors. Using latest target management support in Eclipse or freely available Zylin plug-ins one can have remote target connection support and thus prepare an embedded development environment free of cost [8]. Powerful features of Eclipse and available tools and frameworks enable Eclipse to be effectively used for life cycle management of an embedded device solution [9].

6. Eclipse and the specific needs of embedded software development

Embedded software development has some extra requirements as compared to general purpose or enterprise software development. This is because embedded systems have different hardware model as they can have multiple processors in multiple cores and can have processors with DSP configurations. Moreover software is developed on host machine and cross-compiled and deployed on a
custom target which has different configuration than host development machine and is often constrained by processor type, speed, available memory and real-time responsiveness. Since such software runs on target so a connection to the remote target is required and debugger communication can be slow. Similarly more visibility into the target is required to help perform certain functions like programming flash, tracing instruction flow using on-chip trace buffers and performing low-level hardware diagnostics. Embedded tool vendors joined hands to address these specific needs of embedded development. A new project Device Software Development Project (DSDP) [10] was launched in June 2005. Goal of DSDP is to create an open, extensible, scalable, and standards-based development platform to address the needs of embedded software market by enabling developers and vendors to create differentiated, specialized, and interoperable solutions to help customers and users of Eclipse-based products develop device software faster, better, and at lower cost. [11]

Initially DSDP focused on general embedded support in Eclipse, support of device debugging and target management mainly, but did not stop at that. It started targeting other areas of embedded software development like Java and C/C++ application development for mobile devices and its ultimate goal includes addressing all phases of device software development i.e. hardware bring-up, platform software development and application software development, and making Eclipse a universal platform for embedded/device software development. Companies participating in this project are Accelerated Technologies (Mentor Graphics), AMI Semiconductor, Apogee Software, ARM, Freescale, Fujitsu, Digi, HP, IBM, Intel, MontaVista, Motorola, Nokia, palmSource, QNX, ShareME Technologies, SonyEricsson, Sybase, Symbian, Texas Instruments, Timesys, Wind River Systems, Wirelesxoft, etc. These are industry leaders in their respective domains which adopted Eclipse and are now contributing in DSDP which currently has six sub projects. Below we give an overview of these projects to emphasize how embedded community of Eclipse is addressing its specific needs and making Eclipse a de-facto standard for embedded software development.

6.1. Device Debugging (DD)

Device Debugging [12] is one of the two initial sub-projects proposed by embedded tool vendors and approved by Eclipse board in June 2005. Eclipse debug framework described in section 4.3 is a powerful framework which allowed excellent debuggers to be built on top of it but it targeted developing single applications on a fast workstation with simple hierarchical debug model. Whereas in case of embedded software development, embedded hardware models are complex, targets have limited hardware capabilities, applications are cross-compiled and deployed on targets, debug access is slower, deeper visibility into the target is required and developing and debugging multiple applications is desired. In general, device software debugging is much more complex and different from general software debugging. Mission of Device Debugging (DD) project is to build enhanced debug models, API’s and views that augment the Eclipse debug platform in order to address these added complexities of device software debugging.

DD project has successfully achieved its initial targets including a flexible and customizable debug element hierarchy, model driven updates, retarget-able debug actions, ability to debug without debug view, flexible view wiring, asynchronous interactions between UI and debug model, pluggable source lookup, ability to debug multiple sessions simultaneously and enhancement of Memory view with embedded-specific renderings. To support proprietary debug engines, Debugger Services Framework, a new debug model implementation that takes a more modular approach to integrate customized commercial debug engines in Eclipse, is introduced. It ensures thread safety and fast responsiveness for slow debugger operations. It is a fully asynchronous debug model with strict concurrency rules and provides a great framework to plug individual debugger components (breakpoints, expressions, registers, stack, symbols etc) into Eclipse hence making integration of commercial debug engines in Eclipse easier.

Proprietary debuggers have their own proprietary format of describing targets. DD project has decided to standardize IP-XACT as its language for describing hardware of debugger targets. IP-XACT is an open standard for describing electronic intellectual property, originally intended for use by EDA and ESL tools, created and owned by SPIRIT [13] consortium. DD project will be providing an IP-XACT editor, contributed by ARM, parsing engines, and a target description API based on IP-XACT for use in commercial DSF implementations. Debugger views are also being enhanced for multi-core and multi-process support. Similarly next DD major release is expected to have a reference GDB/mi implementation to serve as an exemplary tool to guide commercial debug engine adoption. Combined with CDT it will offer complete debug solution for embedded software development.

6.2. Target Management (TM)

Embedded systems can consist of multiple targets. Each target can contain multiple processors and each processor can consist of multiple cores. Embedded application
running on a core can have multiple processes and threads. Targets can be locally connected, shared or deployed in the field. There can be multiple possible connections like local serial or JTAG connections or remote socket connections, to connect to a target. Device software vendors need a highly modular and pluggable framework for connectivity, to provide which Target Management [14] project was approved in June 2005. It is one of the initial two sub-projects of Device Software Development Project (DSDP). Its mission is to create data models and frameworks to configure and manage embedded systems, their connections, and their services.

TM project built upon Remote Systems Explorer (RSE), a technology open sourced and contributed by IBM. It provides a data model for describing remote systems, a framework for dealing with remote system connections and subsystems and a feature-rich graphical user interface for visualizing and interacting with remote systems. It provides the abilities to add new remote systems that can be accessed by different methods, define an interface for additional resources on a remote system, define visualizations for additional properties of remote resources, and define actions on remote resources. Target Management project intends to provide a framework for pluggable target services both general such as debugger launcher, RAM download of arbitrary images, reset/reboot, flash programming utility and remote console, and OS-aware such as remote file system, remote process explorer and kernel module downloads.

Every embedded software development tool has its own method of communication with the target system which requires individual set up for each communication method and can be a problem for multiple cores. TM’s goal is to provide a single protocol for communication between all tools and targets, supporting auto-discovery, multiplexing and tunneling. So this project, like other DSDP projects, is truly working for open standards and integrations and many vendors have already started adopting RSE in their commercial tools such as Wind River Workbench 3.0, Atmel AVR32 Studio 1.0, MontaVista DevRocket 5.0, Tradescape Clearing Tool 1.0, EMAC Eclipse Distribution and ACCESS Linux Platform Development Suite 2.0.

6.3. Mobile Tools for the Java Platform (MTJ)

Mobile devices are among the most popular and widely used embedded devices. Like other embedded systems, mobile devices are becoming more powerful and advanced day by day. Java programming language is also becoming more popular in mobile devices. But using Java in mobile applications is significantly different as compared to other applications because there are a number of mobile profiles and configurations. Moreover mobile related Java Specification Requests (JSR’s) require development tools to assist in managing runtime libraries for development work and runtime binding. Similarly mobile Java applications have unique launching and debug requirement. Device emulators are required for debugging and testing on development host and developers need to be able to deploy, launch, debug, test, and analyze performance of applications on actual devices which can have different methods and levels of connectivity. To address all these special needs, Mobile Tools for the Java Platform (MTJ) [15] project was approved in January 2006. Goal of this project is to extend existing Eclipse frameworks to support mobile device Java application development, including a device and emulator framework, a deployment framework, generic and customizable build processes for mobile application development, mobile device debugging, application creation wizards, UI design tools, localization, obfuscation and performance optimization, and security.

6.4. Embedded Rich Client Platform (eRCP)

Eclipse was mainly designed to be a tool integration platform but in 3.0 it underwent major changes to adopt OSGI [16] technology and is now evolving towards a Rich Client Platform (RCP) thus supporting not only tools integration but also applications and services integration [17]. Success of Eclipse’s Rich Client Platform lead embedded community of Eclipse propose an extension of RCP for the embedded and mobile devices, called Embedded Rich Client Platform (eRCP) [18]. It will enrich Java platform for mobile devices by offering extensive and rich UI capabilities, higher level of device abstraction, integration with native look and feel, Eclipse programming model on devices, and OSGI service oriented features like dynamic install/uninstall and sharing of services for devices.

6.5. Native Application Builder (NAB)

Native Application Builder (NAB) [19] project focuses on building embedded GUI’s in C++. It is based on an open source project called WideStudio/MWT [20] which is a general tool and run-time library for building GUI applications on multiple host and embedded platforms and which allows user to write a GUI application once and build it for many platforms without changing code, a concept similar to Eclipse’s SWT and JFace but with a difference that WideStudio/MWT is written in C++ and designed for use in C++ applications. NAB extends Eclipse Visual Editor plug-in for GUI building and supports C/C++ code generation, and application building and debugging. With CDT, GCC/GDB and MWT library plug-ins it offers complete development environment.
6.6. Tools for Mobile Linux (TmL)

Tools for Mobile Linux (TmL) [21] project is a relatively new project of DSDP and is aimed at creating frameworks and tools for entire life cycle C/C++ application development (design, development, debug, deployment) targeted at mobile Linux platforms.

7. An example of use of Eclipse in Embedded Software Development Tools

In this section we briefly describe EDGE Developer Suite [22], a product we have been working on for some years in Embedded Systems division of Mentor Graphics, as an example of successful embedded software development environment based on Eclipse. We were among the first embedded tool vendors to appreciate great features and offerings of Eclipse and its potential as an excellent platform for embedded tools integration. We adopted Eclipse in 2002 and extended it to provide a general environment for all our embedded tools. We are also an Eclipse Foundation member and participate in the DSDP project.

EDGE Developer Suite contains an Eclipse based IDE providing our own Editor, Builder and Project Manager, and embedded tools like Debugger, Profiler, Simulator and Compiler which are integrated into it. This is illustrated in the following diagram.

![Diagram of Eclipse Based Development Tools](image)

Figure 1: EDGE Developer Suite

Before adopting Eclipse our tools had different environments and look and feel. We had to put extra resources and time to provide common IDE features for these tools and they were not tightly integrated. Similarly, supporting these tools on multiple host platforms required a great deal of extra work. Eclipse provided us with a common environment for all our tools. Tools were able to share common IDE features and Eclipse’s extensible plug-in architecture allowed seamless integration among them. Moreover, availability of Eclipse on multiple host platforms enabled us to support our tools on multiple platforms without any trouble and saved us much time and resources. Using Eclipse as a common platform for our tools has helped our customers too as they now have a single environment to be familiar with for all our embedded tools. They get an ideal platform based on Eclipse which they can easily extend by having other freely available or commercial plug-ins.

Our experience of extending Eclipse IDE to provide C/C++ development support has been great and much time saving because of Eclipse’s extensible nature and powerful frameworks. We provided an editor for C/C++ development supporting language aware features as syntax highlighting, source browsing, showing code structure in outline view, context sensitive tool-tips, auto-completion and general features like maintaining local history, content comparison, revision control etc.

![Source Code Example](image)

Figure 2: EDGE C/C++ Editor

Similarly powerful debug framework of Eclipse described in section 4.3 made building front end of our EDGE Debugger for embedded software very easy. Most of the common debug views were readily available for us and writing custom debug views was much easier because of the functionality and extensibility provided by the framework. We provided some custom debug views for our debugger: configurable register view, OS kernel view showing internal status of underlying real time operating system e.g. the status of system objects such as semaphores, pipes, events, etc., memory map view showing layout of the application in target memory and symbol browser view showing symbols of the loaded application, are some of them.
8. Future of Eclipse in Embedded Software Development

Eclipse has secured a future as a universal platform for embedded software development. Most of the embedded tool vendors have already based their tools on Eclipse and are participating and contributing in Eclipse. Eclipse Foundation has successfully launched DSDP project and initiated several projects proposed and contributed by the embedded tool vendors to address their immediate needs. But these projects are still in the development phase and need some time to be fully functional and public to prove their worth. There are still many areas where Eclipse can be extended to facilitate embedded industry like hardware bring up, operating system and middleware configuration, simulation and emulation tools, FPGA and DSP programming, Silicon vendor tool chain support and Electronic design automation. But with the successful track record that Eclipse has, adoption of Eclipse by many embedded tool vendors, and their active participation and contribution, Eclipse is well set to target all these areas and meet its ultimate goals in due time.

9. Conclusion

Eclipse offers rich features as an integrated development environment. Its extensible architecture, powerful frameworks, availability on multiple host platforms and a license which allows companies to build commercial tools using Eclipse, make it a suitable platform for tools integration. Embedded tool vendors are adopting Eclipse because it offers them a great IDE and they can better utilize their resources to work on their core embedded areas. Eclipse is good for embedded developers also as they do not have to learn new environments while working on different processors and operating systems. Open source Eclipse community is playing a leading role to make Eclipse a de-facto standard for tools integration in wide range of areas. Commercial companies are driving Eclipse enhancements to address their specific needs. Embedded software development is different from traditional software development and has extra requirements. Embedded Eclipse community is very active and is successfully leading projects to make Eclipse address their extra needs and be an ideal platform for embedded tools development.

References

[16] www.osgi.org
[18] www.eclipse.org/ecrp
[22] www.mentor.com/products/embedded_software/development_tools