Table of Contents

1 RTSC Creation Review Docuware

1.1 Introduction

The RTSC project is focused on developing Eclipse tools for the development and configuration of C/C++ applications from components for highly constrained devices such as Digital Signal Processors (DSPs) and micro-controllers. RTSC supports a C-based programming model for developing, delivering, and deploying embedded real-time software components targeted for diverse highly resource-constrained hardware platforms. To meet the size and performance constraints of DSPs and 16-bit micro-controllers, RTSC focuses on development-time and configuration-time tooling to generate highly-optimized C/C++ applications. Unlike typical Java runtimes, there is little to no infrastructure that needs to be pre-deployed onto a device for RTSC to work. In addition to a component's C/C++ runtime code, each component includes code - written in JavaScript - that runs both in the component's development environment during application assembly and in rich client platforms to monitor the execution of the C/C++ code within a deployed application.

The RTSC project will start from the current XDCtools product code base. XDCtools is a product from TI that supports the creation, development, integration, and deployment of RTSC packages. This code base is primarily written in Java or JavaScript and leverages some existing Eclipse components (SWT). While XDCtools can be easily used within the Eclipse development platform, there a numerous opportunities to simplify the RTSC development process by extending the Eclipse platform to become "RTSC aware". This includes, for example, integration with CDT to create RTSC project wizards, enable multiple target support, and simplify the debug experience.

The RTSC project scope will be limited to the creation of command-line tools, "infrastructure" components, and plug-ins required to integrate with existing Eclipse projects. For example, rather than creating a new UI for developing and working with RTSC components, the RTSC project will provide appropriate CDT plug-ins. Similarly, to enable higher-level UML design tools, additions to the core XDCtools infrastructure may be required, but the RTSC project will not provide graphical editing tools where existing tools already exist.

However, in cases where a graphical tool is critical to the ease use of RTSC and no similar tool exists, the RTSC project will provide basic graphical tools. For example, we expect to provide a graphical configuration tool for the assembly of multiple RTSC components into an executable. Community supplied replacements will be encouraged.

Community response to the proposal has been positive but entirely hearsay; although the RTSC team has had numerous positive discussions with a variety of interested parties, no public comments have been posted to the RTSC newsgroup (yet).

1.2 Extensible Frameworks and Exemplary Tools

The starting point for the RTSC project is the XDCtools product from Texas Instruments. This product is specifically designed to be independent of any specific C compiler toolchain, host development system, or Software Configuration Management (SCM) system. In particular, these tools

- support a variety of non-TI compilers (e.g., GNU and Microsoft)
- are available on Windows, Linux, and Solaris host development platforms
- are used by development groups using Clearcase, CVS, and proprietary SCM systems.

To enable this independence and to support a diverse set of embedded applications, the XDCtools product is specifically designed to be extended by the development community along several dimensions. The extensions enable the addition of

1. targets to enable use of alternative C compiler tool chains
2. platforms to enable creation of executables for custom hardware platforms
3. platform or domain specific embedded target content services
4. new RTSC component development tools

In each case, the extensions require the implementation of one or more well defined interfaces. Since some of these interfaces must be implemented on the embedded target and others are required on the developer's workstation, these interfaces are specified using the RTSC Interface Definition Language (IDL) which can simultaneously specify both "meta content" (code that runs on rich client platforms) and "target content" (code that runs on the embedded platform). For example, to support *any* new compiler tool chain, it is sufficient to create a RTSC package containing a module that implements the `xdc.bld.ITarget` interface. On the other hand, to make the `xdc.runtime` target content package thread-safe, it's sufficient to create a RTSC package containing a module that implements the `xdc.runtime.IGateProvider` interface (using the thread synchronization primitives provided by the RTOS that created the threads).

With few exceptions, the XDCtools code base itself consists entirely of RTSC packages created using the XDCtools. This ensures that the XDCtools are regularly used by the developers of the tools and makes the XDCtools product itself an exemplar of the RTSC component model. For example, the RTSC interface `xdc.bld.ITarget` defines the boundary between all C compiler tool chains and the XDCtools product. Using this interface, XDCtools contains built-in support for gcc, TI, and Microsoft C compiler tool chains, and this built-in support serves as exemplars for the addition of other tool chains.

1.3 Community Support

The XDCtools product is a foundational element of at least two major embedded software content products from TI:

- DSP/BIOS - a real-Time Operating System (RTOS) for Digital Signal Processors (DSPs), and
- Codec Engine - a multi-media framework that enables portable reuse of high performance DSP algorithms on a variety of platforms (including Arm-based Linux as well as DSP-based DSP/BIOS).

Together these products are actively being used by thousands of developers outside of TI and have served as catalysts to grow a community of developers around the XDCtools. After all, most companies build products out of components not component tools.

1.3.1 Developer Community

Together the DSP/BIOS and Codec Engine products are actively being used by thousands of developers outside of TI. As a consequence, TI has a significant interest in ensuring that the development and evolution of these tools is adequately staffed. Moreover, TI has an interest in encouraging the development of new tools as well as integration into the Eclipse platform. TI's latest release of its embedded development tools builds atop the Eclipse platform.

To ensure robust future releases for existing users, key members of TI's XDCtools development team will form the initial set of committers. The current XDCtools development team currently consists of 5 full-time XDCtools developers, two of which are technical leads. The initial committers and contributors are listed below.

At this time, there is no diversity in the Developer Community. We are optimistic that as more and more content becomes available, we'll be able to attract tools developers from outside TI who will be able to facilitate the use of this content and eventually the creation of new content.

1.3.2 Adopter Community

As mentioned above, there are several distinct areas were XDCtools is designed to be extended. At the current time, the only externally provided extensions are in the area of support for new targets and this has been driven by customers who want RTSC embedded content products (such as Codec Engine) to support these targets. As more RTSC content becomes available, we hope to see adopters who extend RTSC with new tools and application-specific embedded content.

For the most part, extensions of XDCtools to support new C compiler tool chains (such as uclibc) have been added by the XDCtools team. However, because the Codec Engine product provides services for both DSPs and General Purpose Processors (e.g., Arm), there is interest by the community to enable non-TI and even non-GCC based tool chain support. For example, with the help of the XDCtools development team, Greenhills has created a package that allows developers to use the Greenhills C Compiler with the existing XDCtools product. As more RTSC components such as the Codec Engine become available, we expect to see a growing number of adopters who add support for alternative toolchains and platforms.

While the adopter community is relatively small, we expect that making the XDCtools available under EPL will remove a significant adoption barrier posed by a proprietary TI toolchain. For example, the RTSC team has started preliminary discussions with Freescale about the possibility of using XDCtools for the development of Freescale-specific target content. As with Greenhills, the only technical obstacle is the creation of appropriate targets (which is possible today).

Beyond relicensing XDCtools under EPL, increasing the diversity of the developer community will greatly help build the adopter community; companies are reluctant to adopt a technology that can be easily wrenched or killed by a single competitor.

1.3.3 User Community

Since the XDCtools product is a foundational element of both the DSP/BIOS and Codec Engine products, the RTSC project will start with a very large and diverse user community. DSP/BIOS is a popular Real-Time Operating System for DSPs and DSPs are found in a wide variety of applications. DSP/BIOS design wins include cell phones and wireless infrastructure equipment, speaker phones, video surveillance equipment, and even heart defibrillators.

The User Community can be roughly partitioned into two groups:

1. consumers who simply use RTSC components within their existing environment, and
2. producers who create new RTSC components.

While the popularity of DSP/BIOS has resulted in a large *consumer* user base, DSP/BIOS users are not required to create RTSC components. On the other hand, the Code Engine requires all DSP algorithms usable within its framework to be delivered as RTSC components. As a result, the increasing popularity of the Codec Engine is helping drive the number of component *producers*.

1.4 Collaborations

Although the XDCtools has several tools that leverage SWT, the development team has had little or no discussions with other Eclipse projects. However, a major focus of the team for the next six months will be on improving the "development and debug experience" when using RTSC packages. Since Eclipse will be the platform for debugging all embedded content from TI, the RTSC project team has recently started in-depth architectural design reviews of how to best integrate with CDT.

As a result, we expect to become active participants in the CDT community. For example,

- all RTSC target-side symbols follow a naming convention that includes the package name. From the IDE we would like to allow the user to avoid having to specify package names for each symbol when: viewing memory, setting break points, etc.
- RTSC packages often contain support for multiple targets (e.g., TI C6x, native Windows, and native Linux), we would like to enable the package developer to specify the set of targets *within a single RTSC project* that creates a deployable package.
- a key capability of the RTSC model is that it enables developers to create a configuration of multiple components that "optimally" matches an application's needs. This requires a separate "configuration step" prior to linking an application that uses the components, and we want to make it very easy to add this configuration step (or its output) to existing CDT projects.

A second focus for the team is on Real-Time Analysis (RTA): the monitoring of the real-time execution of a deployed embedded system. Although the XDCtools provides configuration support to ensure the runtime footprint is small and efficient, XDCtools does not define standard interfaces for connecting and communicating with an embedded target from a rich client platform. Here there is opportunity to leverage and participate in the work of the Device Kit portion of the SODA project (which has similar needs for monitoring) as well as the monitoring and data collection components of COSMOS.

1.5 Mentors

Doug Gaff and Martin Oberhuber

1.6 Roadmap and Maturity Plan

XDCtools is both mature and actively evolving. The XDCtools team currently operates on a weekly integration cycle where a new engineering release is available almost every week. Major releases occur roughly every 6 months with the feature list and focus determined by community feedback. Since DSP/BIOS and Codec Engine users and developers currently make up the bulk of the community, these products often define the near-term roadmap.

The current plan for XDCtools includes a major release, 3.10, in the July 2008 time frame with the following enhancements:

1. Eclipse CDT project and debug integration
2. addition of a graphical configuration tool
3. Real-Time Analysis (RTA) and RTSC Object Viewer (ROV) infrastructure to enable monitoring of deployed embedded targets
4. completely revamped user documentation

With each release, compatibility with previous releases is carefully managed. Except in rarely used parts of the product, interface changes are required to be compatible with existing implementations. Existing components must be "consumable" using a new release of XDCtools. In addition, components produced with a new release of XDCtools must be consumable by previous releases. These compatibility requirements are currently tested by using an appropriate mix of versions of DSP/BIOS, Codec Engine, and XDCtools.

In parallel with this development, the RTSC team must vet the XDCtools code base through the Eclipse IP process and transition from TI internal SCM, bug tracking, and project management processes to the Eclipse development tools and processes. As a first milestone in this transition, our goal is to move the XDCtools sources into CVS/Subversion by the end of this year.

We expect that a 1.0 release of the RTSC product will include the functionality planned for XDCtools 3.10 together with bug fixes and incremental improvements to the newly added capabilities. Our goal is to have this release available as part of the June 2009.

1.7 Committers

The initial set of committers is listed below in order of their experience with the XDCtools code base.

- Dave Russo, PhD, Distinguished Member of Technical Staff, TI
  - ♦ Over 25 years embedded C experience, especially embedded Digital Signal Processors.
  - ♦ Co-Creator of RTSC and technical lead. 8 years experience with XDCtools code base. Original author of XDCtools product, and continues to review changes and lead its technical evolution.
- Bob Frankel, TI Fellow, TI
  - ♦ Over 25 years embedded C experience, taught numerous University of California Santa Barbara courses in compilers and databases.
  - ♦ Co-Creator of RTSC and technical lead. 8 years experience with XDCtools code base. Original author of the RTSC IDL and lead the development of the RTSC target runtime model, and continues to lead technical direction.
- Sasha Slijepcevic, PhD, TI
  - ♦ Recent University of California Los Angeles graduate (thesis in sensor arrays).
  - ♦ 5 years experience with the XDCtools code base. Currently leads development and evolution of the RTSC target and platform model (key extension points of the XDCtools), the XDCtools runtime package (containing all the of embedded runtime code), and the IDL generator.
- Jon Rowlands, Senior Member of Technical Staff, TI
  - ♦ 20 years DSP algorithm experience especially audio and video compression algorithms.
  - ♦ 3 years experience with the XDCtools code base. Currently the XDCtools Engineering Manager, original author of standalone RTSC configuration tool (configuro) and documentation tool (cdoc), and technical lead of graphical configuration tool and RTSC documentation tooling.
- Joe Cusano, software engineer, TI
  - ♦ 20 years GUI development experience especially Integrated Development Environments for embedded DSP programmers and Engineering Analysis tools.
  - ♦ 3 years experience with the XDCtools code base. Original author of the XDCtools repository management tools (repoman) and current lead developer of the documentation tooling.

1.7.1 Contributors

- Amit Mookerjee, software developer, TI
  - ♦ 5 years experience high-performance DSP codec development.
  - ♦ 6 months experience with XDCtools code base. Enhanced and unified error reporting within XDCtools, created several prototype Eclipse plugins that add "RTSC package awareness" to the Eclipse platform, and added meta-domain trace capabilities.

