WHITE PAPER

Four Steps to Creating Great 32-Bit Microcontroller Applications

A series of four step-by-step guides to using the ARM RealView Microcontroller Development Kit

Guide 1: Selecting the right microcontroller for your application

Rod Crawford June 2006

Abstract: Selecting the right ARM® processor-based microcontroller (MCU) for your application can be a daunting task. With more than 125 ARM processor-based microcontrollers available, finding one with the right peripheral set and performance criteria could take extensive research. This white paper describes how the parametric search engine in the Device Database®, which integrates with the RealView® Microcontroller Development Kit, can be used to quickly choose the right MCU for your application. In addition, we show how the Device Database can be used to configure the tools in the RealView Microcontroller Development Kit for the chosen MCU part.

Overview

In the modern world of embedded development, the demand to stay ahead of competitors, coupled with the ready availability of low-cost, high-performance 32-bit microcontrollers, is leading to widespread adoption of these parts in new products, taking over from the 8-bit and 16-bit microcontrollers of yesteryear. When choosing a microcontroller, embedded developers have four main criteria by which they make their choice: functionality, availability, cost and familiarity. In this paper, we discuss how an online Device Database can be used to aid developers in making the right choice for their application. We also demonstrate how the Device Database integrates with the RealView Microcontroller Development Kit to allow developers to automatically configure their toolchain for development with the chosen MCU.

The problem of choice

Today, when selecting an MCU that will deliver the required peripheral set and performance at the right price-point, developers are faced with a considerable breadth of choice. There are more than sixteen semiconductor companies shipping ARM processor-based MCUs, including some of the largest providers of MCUs in the world. Each semiconductor vendor offers not just one MCU but several families of MCUs that target specific market areas. At the time of writing, there are more than

125 ARM processor-based MCUs available and that number is growing week on week.

Some vendors offer some form of online search tool that allows developers to compare and contrast the MCUs in their product portfolio. However that doesn't help when wanting to compare the MCUs of one vendor with those of another vendor. What is really is needed is a tool that allows developers to compare and contrast MCUs within product families, across product families and across vendors. That's where the Device Database comes in.

The Device Database

The Device Database is a searchable database that contains information about the various MCUs supported by software development tools from Keil™ — an ARM Company, including those ARM processor-based MCUs supported by the RealView Microcontroller Development Kit. There are two editions of the Device Database, one online at www.keil.com/dd and one built into the RealView Microcontroller Development Kit. The parametric search function is available only on the Web-based edition.

The database includes the following information for supported MCUs:

Entry	Description
Part Description	A brief description of each device including part number, core and peripheral set.
Header Files	C and assembly language header files that describe the SFRs (Special Function Registers) available for this MCU.
Example Code	Code snippets and example programs that have been written for the selected MCU or MCU family.
Peripheral Simulation	List of the on-chip peripherals that are fully simulated by the µVision® Integrated Development Environment (IDE), which is integrated with the RealView Microcontroller Development Kit.
Data Sheets	Data sheets related to the MCU part that may be downloaded from the Keil web site.
Boards	The available evaluation boards for the selected MCU.
Emulators	The available emulators for the selected MCU.

Third-Party Software	Software that can run on the MCU, such as Real-Time Operating Systems and application libraries.
Consultants	A list of software and hardware consulting companies familiar this MCU.

.

In addition to the above, the database contains links to FindChips.com, which can list what distributors carry a particular MCU.

Database searches

The Device Database can be searched for MCUs in the following ways:

- *Architecture*, which shows all MCUs that support a particular processor architecture. E.g., searching by ARM Architecture will list all of the vendors that support the ARM architecture at the heart of an MCU.
- *Vendor*, which shows all MCUs produced by particular silicon vendor. E.g., searching for Philips as a vendor will list all MCU families and MCUs that Philips produces.
- *Parametric*, which allows developers to enter the requirements for the MCU they want to find.

Using the power of the parametric search

The parametric search capability of the Device Database is perhaps of the most interest to developers, since it enables them to search for specific peripheral sets and performance criteria across the whole range of MCUs without being constrained by the search capabilities of any single processor vendor. Using this kind of search, developers can quickly home in on a single MCU part or MCU family that will offer the best match of performance, peripheral set and scalability for future-proofing.

Parametric search example

To show some of the advanced capabilities of the parametric search of the Device Database, we will use the following example:

Example: Remote Temperature Logging Device

The device we are creating is a remote temperature sensing device with the ability to log information from a sensor over time and store it locally. The device should run on batteries when in the field and should have failsafe reboot should it crash. When the device is docked via USB, the device should be able to dumps its log via USB and synchronize its clock for further data collection. Based on these criteria, we can distill the requirements for the MCU to the following:

Peripheral	Parameter
On-chip RAM	>8K

On-chip Flash	>64K
A/D Channels	8+
Power-Down Mode	Yes
Idle Mode	Yes
Real-Time Clock	Yes
Watchdog Timer	Yes
USB	Yes

If we feed these parameters as a parametric search into the Device Database, we find that there are currently two ARM processor-based MCU families from different MCU vendors, each with four MCU parts that fit the bill. Using this kind of search across different MCU vendors has narrowed our choice from more than 125 MCUs to eight. From here, we can examine each individual part entry in the Device Database and make our final decision based on more subjective criteria, such as past experience, clarity of documentation, relevance of applications libraries and example code and, of course, price/volume.

RealView Microcontroller Development Kit configuration

In the above example, we have shown how the Device Database can be used as a stand-alone tool on the Web to make informed decisions regarding the choice of MCU for a particular application. However, the Device Database's usefulness extends beyond this, because it can also be used during the setup phase for a particular application project. The RealView Microcontroller Development Kit contains an edition of the Device Database which is invoked when a developer begins a new project. When the developer specifies the chosen MCU in the Device Database within the RealView Microcontroller Development Kit's µVision IDE, the database's knowledge of that MCU is used to automatically configure and tailor the development tools specifically to that MCU. This can significantly reduce tooling setup times. The automatic tools configuration includes the following:

Simulator setup

This configures the µVision simulator to model the appropriate MCU including default clock rate, instruction set, register set, internal ROM and RAM spaces and peripheral set.

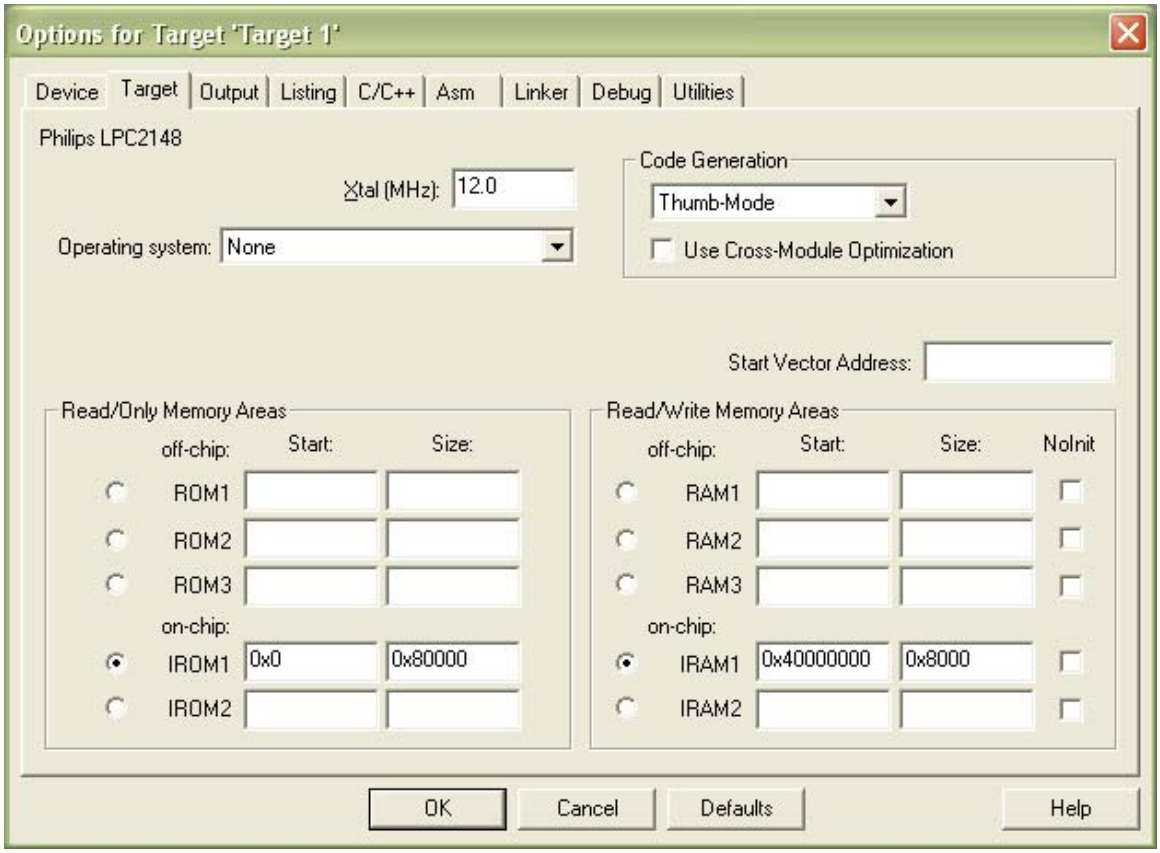

Figure 1: Setting options for the target MCU

Linker setup

This builds a linker readable memory map or Scatter File that defines where the linker should place the code to the right addresses and tells the MCU simulator the layout of the memory map (the simulator can trap writes to non-memory or read-only memory).

Compiler and assembler setup

This sets the compiler and assemblers include path to include the correct header files for this MCU.

Installation of startup code into the project

This prepends the source of the relevant example MCU startup code to the start of the project, ensuring that after reset the MCU is put into a known, well-defined state prior to execution of the application.

In addition to the above, the automatic configuration sets several useful defaults for first-time use of the tools, including compiling for debug, generation of a map file and call graph. Each of these options is shown in the Options for Target GUI in the µVision IDE as both a GUI changeable item and the set of command line arguments

for each tool the GUI generates. Figure 1 shows the target configuration for a Philips LPC2148 MCU with entries for its on-chip memory map and default clock rate automatically completed from the MCU's entry in the Device Database. The user can then add additional configuration options, such as additional on-board memory locations, to the configuration.

Conclusion

At the beginning of this guide, we stated that when choosing an MCU, embedded developers have four main criteria by which they make their choice: functionality, availability, cost and familiarity. The Device Database is unique in its ability to enable developers to make informed choices on these criteria by allowing them to:

- Search for specific functionality across the whole range of ARM processor-based MCUs.
- Quickly find pricing and availability of MCUs from distributors.
- Rapidly start working in the familiar µVision development environment that has been pre-configured to build applications for their specific chosen MCU.

Please read on. "Guide 2: Developing your first application for an ARM processor-based microcontroller" continues on the next page.

Guide 2: Developing your first application for an ARM processor-based microcontroller

Rod Crawford August 2006

Abstract: The power of 32-bit processor-based microcontrollers is enabling a significant shift in the embedded development world toward off-the-shelf software components and industry standard high-level languages. This guide describes how the RealView Microcontroller Development Kit from ARM and its associated RealView Real-Time Library deliver a high-performance configurable software platform that can be used as a foundation to jump-start the rapid creation of ARM processor-based microcontroller (MCU) applications.

Introduction

The high performance provided by 32-bit ARM processor-based microcontrollers is allowing the use of large-scale, off-the-shelf software components that 8-bit and16-bit microcontrollers could never adequately support. Embedded developers can now program their MCU applications using high-level languages such as ISO standard C and C++. The use of high-level languages more easily supports the use of off-the-shelf software components such as real-time operating systems, file systems and networking libraries. The process of software development becomes a matter of bolting components together, enabling embedded applications to be built quickly with little knowledge of the inner working of components or of the peripherals they service.

The ARM RealView Microcontroller Development Kit and RealView Real-Time Library offer a set of software components that address many of the common requirements of modern-day embedded applications. In addition, the Device Database contains many example programs that utilize these software components and provides high level template applications that can be adapted into a final application.

Target Configuration

Correctly setting up the target environment for an embedded application can take considerable time to get right. As noted in Guide 1, the RealView Microcontroller Development Kit can add specific startup code for a chosen microcontroller at the start of a project, enabling the developer's applications to begin execution when the MCU is in a known well-defined state. However, the memory requirements and peripheral behavioral requirements of each application that runs on a specific MCU can be significantly different, and it is impossible for the supplied startup code to cater for every possible setup that developers may require.

One approach is for developers to take a standard set of startup code and incrementally modify it until the required state of the MCU and its resources, such as stacks, is achieved. This approach, while undoubtedly leading to success, can be

time consuming. The RealView Microcontroller Development Kit offers an alternative approach to hand modifying the example startup code with a Configuration Wizard that provides a high-level view of the resources of the MCU that map onto the startup code. Figure 1 shows part of the configuration for two of the chip select registers on the external bus interface of an Atmel AT91M55800A microcontroller. The developer can easily enable and configure a specific chip select register by modifying the values for each entry.

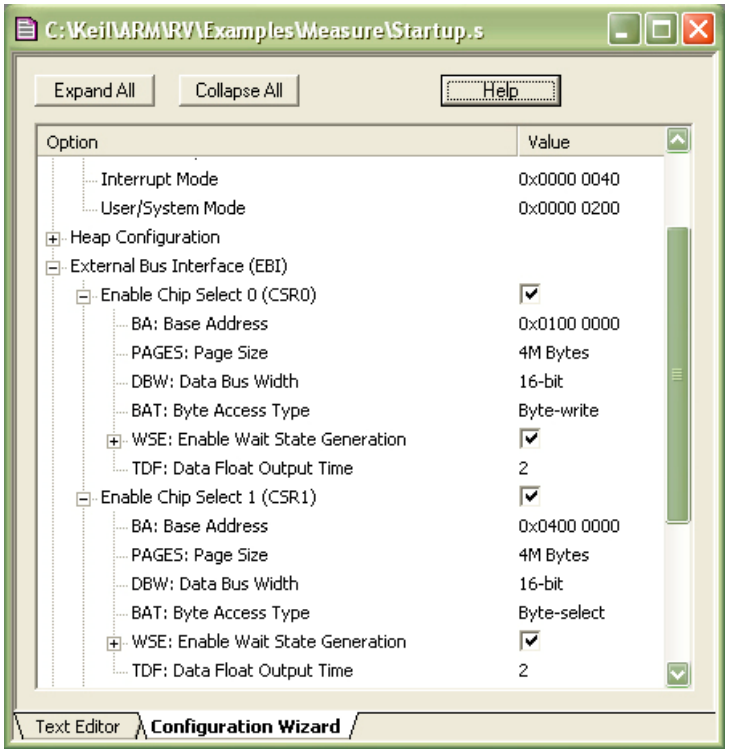

Figure 2

This approach significantly simplifies the creation of startup code since the startup code is automatically generated as the Configuration Wizard is completed. E.g. completion of the entries for Chip Select 0 (CSR0) automatically modifies the numeric value associated with an ARM assembler constant declaration of the form:

```
EBI_CSR0_Val    EQU     0x010024A9
```

This constant is used later in startup code for the initialization of the external bus interface.

Under the Hood of the Configuration Wizard

The Configuration Wizard uses a simple markup language to define what the user sees in its dialog box and how the user's choices generate numeric and textual items. Markup language can be embedded in the comments of C, C++ or assembler source code or any text file, for that matter. This allows the Configuration Wizard to

be used for any kind of configuration which is defined using a numeric string. Figure 2 shows an example of the Configuration Wizard being used to configure the heap size of a target system. Figure 3 shows the underlying markup language for this example. The tags <h> and </h> define the start and end of the heap entry with its associated label. The <o> tag defines a user editable entry within a numeric range. The actual entry to be edited in this is example is implicitly the first numeric field following the markup (0x00000000). The user can define a field offset by specifying a number after the o.

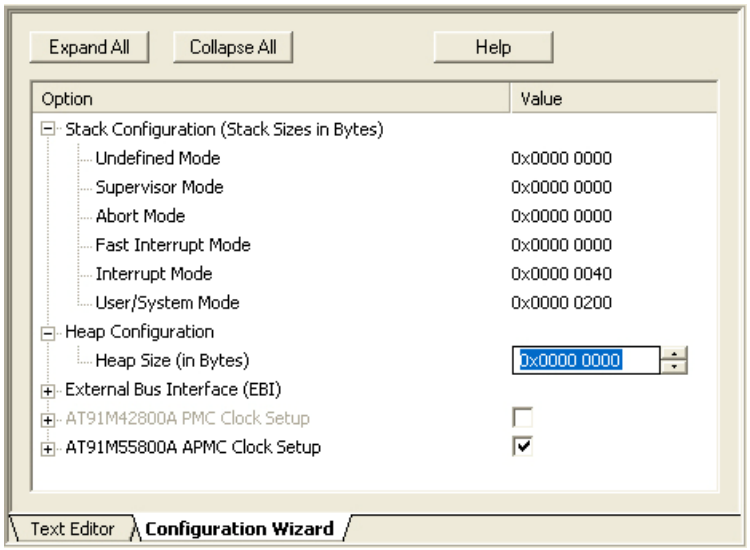

Figure 3

```
058 Stack_Top        EQU     Stack_Mem + Stack_Size
059
060
061 ;// <h> Heap Configuration
062 ;//   <o>  Heap Size (in Bytes) <0x0-0xFFFFFFFF>
063 ;// </h>
064
065 Heap_Size        EQU     0x00000000
066
067                  AREA    HEAP, NOINIT, READWRITE,
068 Heap_Mem         SPACE   Heap_Size
069
070
071 ;// <h> External Bus Interface (EBI)
072 EBI_BASE         EQU     0xFFE00000
073
074 ;//   <e.13> Enable Chip Select 0 (CSR0)
075 ;//     <o.20..31> BA: Base Address <0x0-0xFFF00(
```

Text Editor | Configuration Wizard

Figure 4

Software Applications Libraries

The high performance provided by ARM processor-based MCU's allows their use in market areas which are relatively more computationally intense than other markets addressed by 8- and 16-bit MCUs. Addressing these markets successfully requires

more complicated software. While a typical 8-bit application might contain a Real-Time Operating System (RTOS) and some control code, modern 32-bit MCU applications typically will contain complete communication stacks that are an order of magnitude more complex than the humble RTOS.

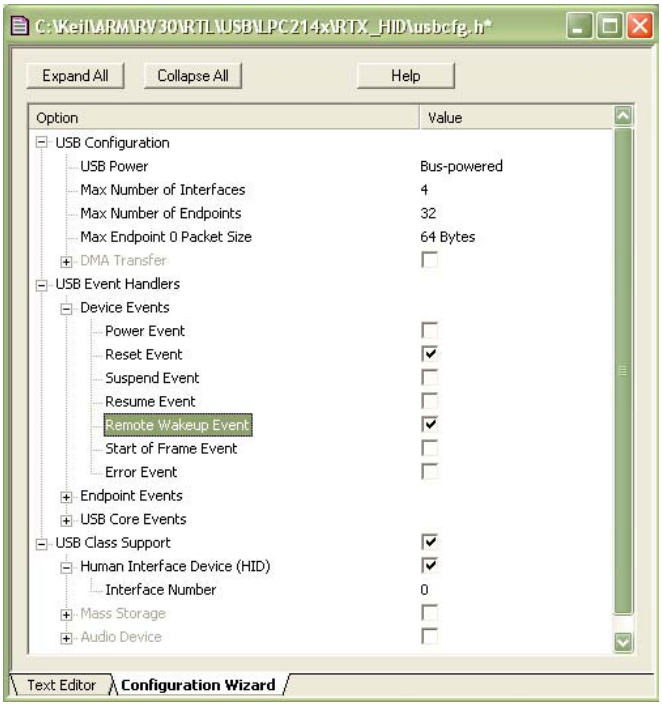

Figure 5

The RealView Real-Time Library provides a set of commonly required software components that can easily be deployed on ARM processor-based MCUs. The library contains such items as a TCP/IP networking stack, Flash file system, USB device driver and CAN device driver optimized for the specific peripherals of the MCUs they support. Developers can quickly configure the components using the Configuration Wizard in the RealView Microcontroller Development Kit. Figure 4 shows an example of the Configuration Wizard being used to configure a USB2 stack.

The RTX real-time kernel is included as a configurable object in the RealView Microcontroller Development Kit and is provided in source form in the RealView Real-Time Library. This is a fully featured real-time kernel offering many of the most common resource abstractions found in a modern RTOS, including threads, timers, queues, mailboxes, semaphores, mutexes, block pools and event flags. Developers can use the kernel to combine components from the Real-Time Library so that they execute as separate communicating tasks on a target system.

Code Templates and Application Examples

The RealView Microcontroller Development Kit contains several example applications for each supported MCU. Developers can use the examples as

templates for their applications, removing the need for in-depth knowledge of their chosen MCU just to begin development. The examples cover common initial development requirements such as setting up interrupts, flashing an LED or writing text to an output device, as well as complete analog data acquisition and real-time OS examples.

The RealView Real-Time Library also ships with several example applications that utilize components of the RealView Real-Time Library in typical real world scenarios. These can be used as basis for a complete embedded application. Examples include:

- Embedded Web server with CGI scripting
- SMTP email notification
- Telnet
- USB memory device
- USB human interface device
- USB audio device

Conclusion

Developing microcontroller applications using 32-bit ARM processor-based MCUs enables developers to leverage a broad range of off-the-shelf software components in a way that was previously not feasible on 8- and 16-bit MCUs. The sheer horsepower and memory addressing capabilities of an ARM processor-based MCU invariably removes the need for hand-coded assembler or proprietary software libraries. The RealView Microcontroller Development Kit and RealView Real-Time Library provide developers with a pool of common software components and examples that they can use as the fundamental building blocks for their application. The target system and software components can be easily configured for the desired application in the RealView Microcontroller Development Kit's µVision IDE by using the inbuilt Configuration Wizard. By using these configurable components, coupled with a real-time kernel, developers can get their application to market faster with a higher degree of confidence than ever before.