Programming in the MAXQ environment

The in-circuit debugging and program-loading features of the MAXQ2000 microcontroller combine with IAR's Embedded Workbench development environment to provide C or assembly-level application development and testing.

The MAXQ architecture was developed for application programmers. Each MAXQ microcontroller includes a hardware debug engine that is tightly integrated with the microcontroller core. The first chip in this architecture is the MAXQ2000, and this article provides examples and tips on the use of the IAR Embedded Workbench with the MAXQ2000 Evaluation Kit.

The in-circuit debugging and program-loading features of the MAXQ2000 microcontroller combine with IAR's Embedded Workbench development environment to provide C or assembly-level application development and testing. The hardware-based debug engine and bootloader of the MAXQ2000 run over a dedicated JTAG port to allow full debugging access with minimal impact on system resources.

In-circuit debug features

A hardware debug engine, which is tightly integrated with the microcontroller core, controls the MAXQ2000's debugging features. This debug engine can invoke service routines in the on-board utility ROM to support a wide array of debugging features.

- Read access to the integrated flash program memory.
- Read/write access to the on-board data SRAM.
- Read access to the 16 x 16 stack memory.
- Read/write access to all MAXQ2000 system and peripheral registers.
- Step-by-step (trace) program execution.
- Up to four address-based breakpoints to stop program execution at a particular location in code memory.
- Two data memory-matching breakpoints to stop program execution when a particular location in data memory is accessed.
- Two register-based breakpoints to stop program execution when write access to a particular system or peripheral register occurs (cannot be used simultaneously with the data memory matching breakpoint) and the data being written to the register matches a specified value.
- Password matching function (to unlock the remaining debug functions).

All communication with the debug engine takes place over the MAXQ2000's dedicated JTAG Test Access Port (TAP) interface, which is compatible with the JTAG IEEE Standard 1149.

All communication with the debug engine takes place over the MAXQ2000's dedicated JTAG Test Access Port (TAP) interface, which is compatible with the JTAG IEEE Standard 1149. This interface consists of four signals, multiplexed with MAXQ2000 port pins as follows: TMS (Test Mode Select)—multiplexed with P4.2; TCK (Test Clock)—multiplexed with P4.0; TDI (Test Data In)—multiplexed with P4.1; and TDO (Test Data Out)—multiplexed with P4.3.

While the JTAG TAP port is dedicated to in-system debug and in-system programming uses, the four port pins that carry the JTAG TAP port signals may be released for other purposes once application development is complete. The JTAG port is active by default following reset, but once running, the application software can deactivate the JTAG port, leaving the four associated port pins free for other uses.

The JTAG interface and the debug engine operate asynchronously with respect to the MAXQ2000 core. Communication over the JTAG port need not take place at the same clock rate that the MAXQ2000 is running, although the frequency of TCK is limited to a maximum of 1/8 the system clock rate for the MAXQ2000.

Breakpoint settings can be read and written through the debug engine while the MAXQ2000 is executing code. This mode is known as background mode, where the debug engine operates independently of the CPU core.

A hardware debug engine, which is tightly integrated with the microcontroller core, controls the MAXQ2000's debugging features.

To perform other operations such as memory and register read and write, the debug engine takes control of the MAXQ2000 core, and switches execution to one of the debug service routines located in the utility ROM. This mode is known as debug mode, in which the debug engine interrupts normal program execution. The user application is suspended temporarily in these cases and resumes execution once the debug function has been completed, in the same way that interrupt routines are handled.

Because the JTAG TAP port is not used for application software purposes, the port pins comprising the JTAG port can be reclaimed by the application software. All additional code required for debugging functions is located in the utility ROM, so the only system resources consumed by the debugging functions are a small amount of data SRAM and one level of the program stack (used to store the return address when a debugging routine is called). The highest 19 bytes of data SRAM (addresses 0x07ED to 0x07FF) are reserved for use by the debugging service routines. If in-circuit debugging will not be used for a particular application, these data SRAM locations are available for application use.

Integrated flash-memory programs over JTAG

The JTAG TAP port is also used for an additional bootloader function, which is available even if the debugging functions will not be used. By setting three configuration bits over the JTAG TAP interface and then releasing the MAXQ2000 from reset, control can be transferred to the built-in bootloader routines located in the utility ROM. The configuration bits that control access to the bootloader are as follows.

- SPE: System Program Enable Bit (ICDF.1). When this bit is set to 1, the MAXQ2000 executes the bootloader routine in the utility ROM following system reset.
- PSS[1:0]: Programming Source Select (ICDF.3-2). The settings of these bits determine whether the JTAG port (PSS[1:0] == 00b) or the serial 0 UART (PSS[1:0] == 01b) is used for bootloader communication.

Once these bits are set and the MAXQ2000 is released from reset, the utility ROM bootloader begins communicating with the host system over the selected port (JTAG or serial 0 UART). In either case, the protocol used is the same and provides the following functions.

- Reads the ID banner of the MAXQ2000 (identifies utility ROM version).
- Returns the size of internal program and data memory.
- Reads, writes, verifies, and CRCs the integrated flash program memory.
- Reads, writes, verifies, and CRCs the internal data SRAM.
- Password matches (to unlock memory read and write commands).

By setting three configuration bits over the JTAG TAP interface and then releasing the MAXQ2000 from reset, control can be transferred to the built-in bootloader routines located in the utility ROM.

While the bootloader can communicate over the serial 0 UART instead of the JTAG port, the JTAG interface must be used to place the bootloader into serial communications mode. However, the application software can also invoke the bootloader in serial communications mode by setting the SPE and PSS bits appropriately, then resetting the MAXQ2000 (by letting the watchdog timer expire or by external hardware means). The method for causing the bootloader to be invoked (such as a signal on a port pin) must be determined by the application software.

Password protection for debug and bootloader functions

A basic password-protection scheme restricts access to the debugging and bootloader functions on the MAXQ2000. This password must be provided by the host system before access is allowed to any functions that read or modify the contents of memory or system and peripheral registers.

A basic password-protection scheme restricts access to the debugging and bootloader functions on the MAXQ2000.

The password is 16 words or 32 bytes long. The value for the password is located in the internal flash memory at word locations 0x0010 to 0x001F. These values can be included in an application as a static array, or they can simply be the values of the instruction codes stored in those locations. Either way, the password is automatically written when the application is loaded. If no application has been loaded, the password will be a default value with all words equal to 0xFFFF.

Even if the password is not known, the MAXQ2000's internal flash memory can always be erased through the bootloader. This effectively clears the password value (to all 0xFFFF words) and allows other programming and debugging operations to proceed. The password protection simply ensures that existing code may not be read from the MAXQ2000 without first matching the 32-byte password value.

Using the serial-to-JTAG adapter module

Integrated development environments for the MAXQ2000 microcontroller (such as MAXIDE and IAR Embedded Workbench) include software libraries to support communication with the MAXQ2000 JTAG interface. However, as the PCs running this software do not typically have JTAG ports included, a hardware layer is needed to interface the two systems.

The serial-to-JTAG adapter module, included with the MAXQ2000 Evaluation Kit, provides a turnkey solution to this interface problem (**Figure 1**). Software running on the PC (such as IAR Embedded Workbench) communicates with the serial-to-JTAG adapter module over a standard COM serial port. The serial-to-JTAG adapter module then interfaces to the JTAG port of the MAXQ2000, passing commands to the bootloader or the debugging engine. The adapter module also handles level translation and supports MAXQ microcontrollers running over a range of different supply voltages, as well as removes the need for the PC to provide precise timing for the JTAG waveforms.

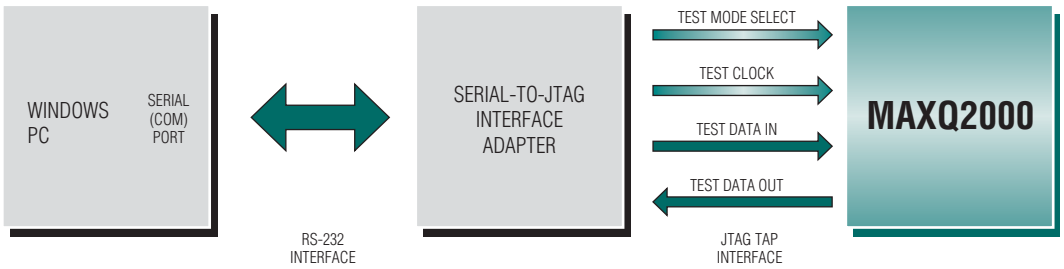

Figure 1. *The serial-to-JTAG adapter module allows software running on the PC to access the JTAG TAP interface of the MAXQ2000 microcontroller.*

Using the MAXQ2000 Evaluation Kit hardware

The MAXQ2000 Evaluation Kit provides a complete hardware development environment for the MAXQ2000 microcontroller, including the following features.

- On-board power supplies for the MAXQ2000 core and VDDIO supply rails.
- Adjustable power supply (1.8V to 3.6V), which can be used for the VDDIO or VLCD supply rails.
- Header pins for all MAXQ2000 signals and supply voltages.
- Separate LCD daughterboard connector.
- LCD daughterboard with 3V, 3.5-digit static LCD display.

- Full RS-232 level drivers for serial 0 UART including flow control lines.
- Pushbuttons for external interrupts and microcontroller system reset.
- MAX1407 multipurpose ADC/DAC IC, connected to the MAXQ2000 SPI bus interface.
- 1-Wire® interface, including iButton® clip and 1-Wire EEPROM IC.
- Bar graph LED display for levels at port pins P0.7 to P0.0.
- JTAG interface for application load and in-system debugging.

With the combination of the MAXQ2000 Evaluation Kit and the serial-to-JTAG adapter module, IAR Embedded Workbench has full access to the JTAG-based bootloader and in-circuit debugging features of the MAXQ2000.

Setting the MAXQ2000 Evaluation Kit board and the serial-to-TJAG interface modules for application development is straightforward. Simply connect the boards by the following steps.

1) Plug a 5V DC-regulated power supply (center post positive, ±5%) into the serial-to-JTAG board power jack J2.
2) Plug a 5V to 9V DC power supply into the MAXQ2000 Evaluation Kit board power jack J1.
3) Connect a straight-through DB9 serial cable from the serial-to-JTAG board J1 connector to one of the COM ports on the PC.
4) Connect the JTAG adapter cable from the 1 x 9 connector P2 on the serial-to-JTAG board to the 2 x 6 connector J4 on the MAXQ2000 Evaluation Kit board.
5) Turn both DC power supplies ON.
6) For standard operation, all DIP switches on the MAXQ2000 Evaluation Kit board should be in the OFF position.

Application development using IAR Embedded Workbench

The IAR Embedded Workbench development environment provides C-based or assembly-based application development for the MAXQ2000. Using the previous hardware configuration that includes the MAXQ2000 Evaluation Kit board and the serial-to-JTAG adapter module, IAR Embedded Workbench has full access to the JTAG-based bootloader and in-circuit debugging features of the MAXQ2000.

IAR Embedded Workbench provides the following features when developing applications for the MAXQ2000.

- Load compiled applications to the MAXQ2000 integrated program-flash memory.
- Step-by-step (trace) program execution at the C or assembly level.
- Display of code, data, hardware stack, and utility ROM memory.
- Call stack tracing.
- Breakpoint setting at the C or assembly level.
- View and edit of all MAXQ2000 system and peripheral registers.

Creating and compiling a project for the MAXQ2000

The IAR Embedded Workbench development environment provides C-based or assembly-based application development for the MAXQ2000.

Because IAR Embedded Workbench includes integrated support for the MAXQ microcontroller family, creating a new project for the MAXQ2000 microcontroller requires only a few specific settings.

After starting IAR, select **File**, then **New** from the menu. Select *Workspace* from the **New** dialog box and click **Ok**. Enter a new name for the project workspace (stored as a ".eww" file) and click **Save**.

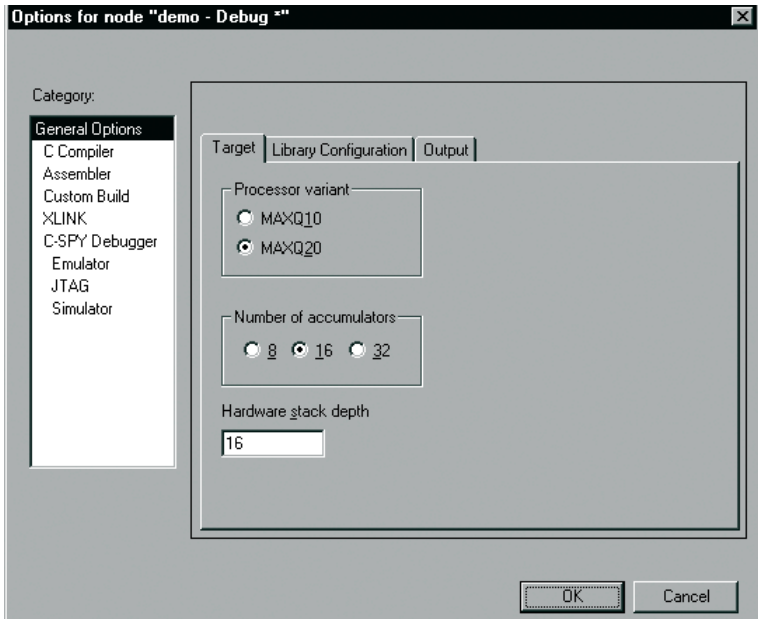

Figure 2. *The General Options section of the Options dialog allows the user to specify the processor core type (MAXQ10/20), the number of accumulators available, and the hardware stack depth. The settings shown are for the MAXQ2000.*

Once the workspace window opens, select **Project,** then **Create New Project** from the menu. The *MAXQ* tool chain is the default for the new project. Enter a file name for the new project (stored as a *.ewp file) and click **Create**.

Next, select **Project**, then **Settings** from the menu. A dialog box will appear with the settings for the newly created project, as shown in **Figure 2**.

In the *General Options* tab of the **Options** dialog box, the following settings should be selected for the MAXQ2000 microcontroller.

- **Processor Variant** should be set to **MAXQ20**, as the MAXQ2000 has a MAXQ20-type core.
- **Number of accumulators** should be set to **16** for the MAXQ2000.
- **Hardware stack depth** should be set to **16** for the MAXQ2000.

In the *C-SPY Debugger* tab of the **Options** dialog box, the following settings should be selected for the MAXQ2000 (**Figure 3**):

- Set the **Driver** setting to **JTAG** to connect to the serial-to-JTAG interface board over a PC COM port. The other two possible settings are **Simulator** (used to run with the MAXQ2000 software simulator) and **Emulator** (used to run with the MAXQ2000 In-Circuit Emulator system).

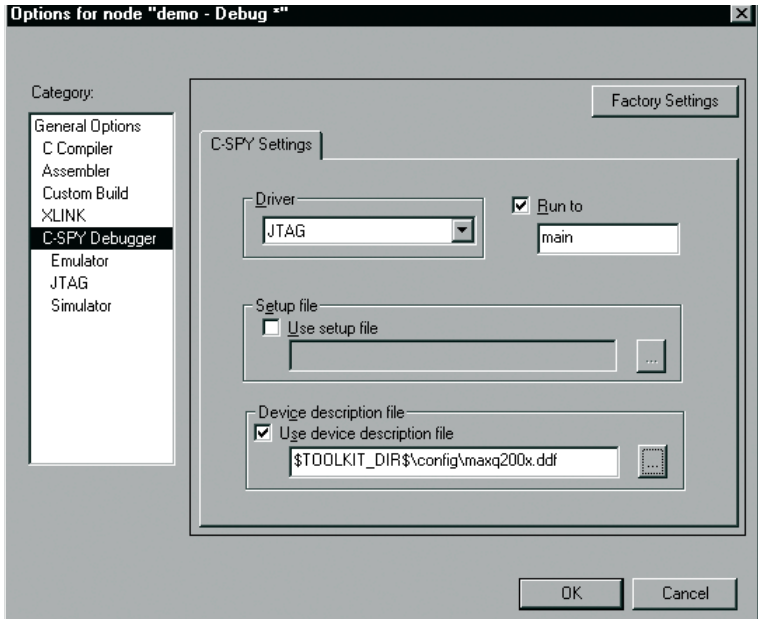

Figure 3. *The C-SPY Debugger section of the Options dialog allows the user to specify settings for debugging sessions. The settings shown are for debugging the MAXQ2000 using the serial-to-JTAG adapter module.*

- The **Use Device Description File** box should be checked. The device description file (*.ddf) should be the file provided for the MAXQ2000 microcontroller (maxq200x.ddf). This file defines the memory spaces and peripheral register set for a particular MAXQ microcontroller for use by the IAR environment.

Under the *JTAG* section of the **Options** dialog box, the **Command line options** field contains the COM port of the PC used to connect to the serial-to-JTAG board. **Figure 4** shows the option setting for connecting to COM port 1.

After setting the options for the project, select **Project**, then **Add Files** to add a C code file to the project. Once the project file(s) have been added, select **Project**, then **Make** to compile the project, followed by **Project**, then **Debug** to start a debugging session. This downloads the compiled project over the JTAG interface and places IAR into debug mode, as **Figure 5** shows.

Debugging operations in IAR

Once the debugging session has started, **Step Over** (F10), **Step Into** (F11), and **Step Out** (Shift+F11) can be used to trace through the C code of the project. To run code, select **Debug**, then **Go** from the menu, or hit F5.

Address breakpoints can be set or cleared by placing the cursor on a line of source code and clicking the **Toggle Breakpoint** button in the toolbar. Up to four address breakpoints can be set at once.

The **Memory** window can be used to display the **Code** (internal flash memory), **Data** (internal SRAM), **Hw stack** (internal 16-level stack), and utility ROM memories of the MAXQ2000. The memory display can be set to byte, word, or doubleword format, and displays in both hex (for all widths) and ASCII (for byte width) formats.

The **Register** window displays the system and peripheral registers for the MAXQ2000. These are displayed in logical groups.

- **CPU Registers**: Accumulator and accumulator control registers, data pointers and data pointer control registers, instruction pointer, loop counter, and program status flags.
- **Interrupt Control**: Interrupt vector, module mask, and identification registers.
- **Cycles**: Displays the number of instruction cycles that have executed.
- **Parallel Ports**: Input, output, and port direction registers for P0 to P4.
- **External Interrupt**: Enable, edge select, and flag registers for external interrupts.
- **Timers**: Registers for timer/counters 0 to 2.
- **Serial Port**: Control and buffer registers for the SPI and serial ports.
- **Multiplier**: Registers related to the hardware multiplier module.

Writeable registers can be edited by clicking on the register value and entering a new value. Display of the individual bits or bit fields within registers can be expanded or collapsed by clicking the plus/minus sign next to the register name.

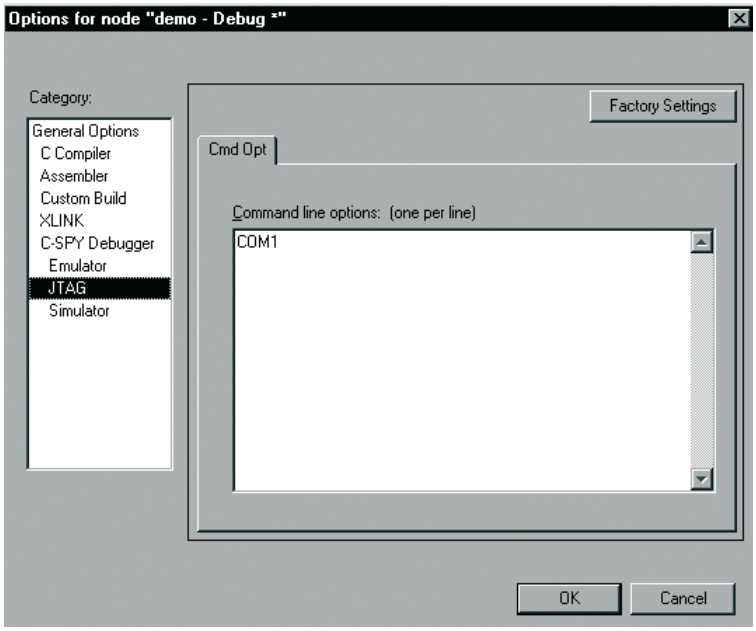

Figure 4. *The C-SPY Debugger (JTAG) section of the Options dialog allows the user to change settings specific to the serial-to-JTAG adapter module. The settings shown are for a serial-to-JTAG adapter connected to the PC port COM1.*

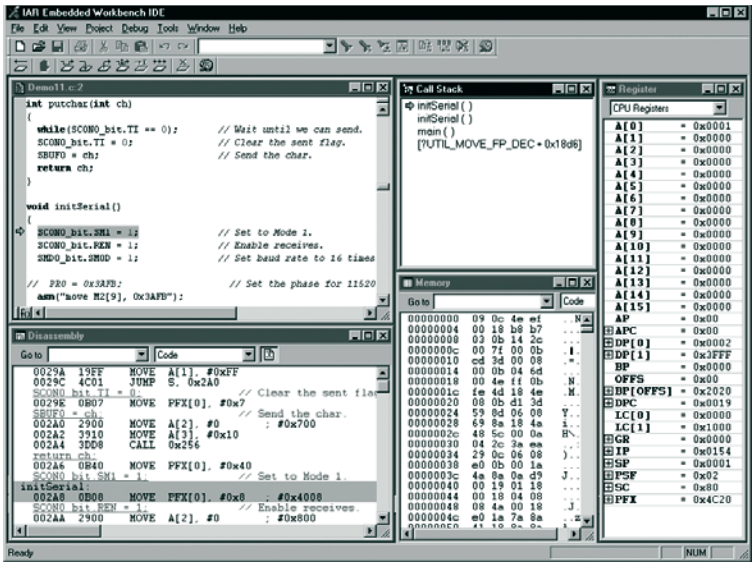

Figure 5. *Using the serial-to-JTAG adapter module, IAR Embedded Workbench can perform step-by-step execution on the MAXQ2000, as well as read and modify on-chip memory and register values.*

Conclusion

The high-level, C-project-based environment of IAR Embedded Workbench integrates with the MAXQ2000's low-level debugging interface to allow fine-tuned debugging at either the C or assembly code levels. The MAXQ2000's built-in debugging and in-circuit programming features, and their low-level impact on system resources, allow the same hardware design to be used for both the application development process and for the final release of the finished project.

1-Wire and iButton are registered trademarks of Dallas Semiconductor.