

MAXQ環境における プログラミング

MAXQ2000マイクロコントローラが持つインサーキットデバッグ機能とプログラムロード機能をIARのEmbedded Workbench開発環境と組みあわせると、C言語やアセンブラによるアプリケーションの開発や試験を行うことができます。

MAXQアーキテクチャはアプリケーションのプログラマを対象として開発されました。MAXQマイクロコントローラはいずれも、マイクロコントローラコアと緊密に統合されたハードウェアデバッグエンジンを持ちます。このようなアーキテクチャを持つチップとして最初に登場したのがMAXQ2000ですが、このアーティクルでは、IAR Embedded WorkbenchとMAXQ2000評価キットという組み合わせの例と活用ヒントを紹介いたします。

MAXQ2000マイクロコントローラが持つインサーキットデバッグ機能とプログラムロード機能をIARのEmbedded Workbench開発環境と組みあわせると、C言語やアセンブラによるアプリケーションの開発や試験を行うことができます。MAXQ2000では、ハードウェアベースのデバッグエンジンとブートローダが専用JTAGポートから利用できるため、システムリソースへの影響を最小限に抑えながらあらゆるデバッグを行うことができます。

インサーキットデバッグの特長

MAXQ2000のデバッグ機能は、マイクロコントローラコアと緊密に統合されたハードウェアデバッグエンジンによってコントロールされます。このデバッグエンジンは、オンボードのユーティリティROMに格納されたサービスルーチン呼び出し、以下に示すさまざまなデバッグ機能を実現します。

- プログラム用内蔵フラッシュメモリへの読出しアクセス。
- オンボードのデータSRAMに対する読出し/書込みアクセス。
- 16x16スタックメモリに対する読出しアクセス。
- MAXQ2000のシステムレジスタと周辺機器レジスタのすべてに対する読出し/書込みアクセス。
- ステップバイステップのプログラム実行(トレース)。
- コード内の指定位置でプログラムの実行を停止するアドレスベースのブレークポイントを最大4カ所設定可能。
- データメモリの指定位置に対してアクセスがあったらプログラムの実行を停止するデータメモリマッチングブレークポイントを2カ所設定可能。
- 指定したシステムレジスタまたは周辺回路レジスタに対して書込みアクセスがあり、かつ、レジスタに書き込まれるデータが指定値であったらプログラムの実行を停止するレジスタベースのブレークポイントを2カ所設定可能(データメモリマッチングブレークポイントと同時に設定することは不可能)。
- パスワードマッチング機能(他のデバッグ機能をアンロックするため)。

デバッグエンジンとの通信は、すべて、MAXQ2000が持つ専用JTAGテストアクセスポート(TAP)インタフェースを経由して行います。なお、このインタフェースは、JTAG IEEE 1149互換です。インタフェースは4つの信号で構成されており、MAXQ2000ポート端子と次のように多重化されています。P4.2とTMS(Test Mode Select)、P4.0とTCK(Test Clock)、P4.1とTDI(Test Data In)、P4.3とTDO(Test Data Out)です。

デバッグエンジンとの通信は、すべて、MAXQ2000が持つ専用JTAG TAPインタフェース(JTAG IEEE 1149互換)を経由して行われます。

JTAG TAPポートはインシステムデバッグ及びインシステムプログラミングに専用に使われるポートですが、JTAG TAPポート信号を伝える4本のポート端子は、アプリケーションの開発を終えた後は、他の用途に利用することが可能です。JTAGポートはリセット後に自動的にアクティブとなりますが、起動されたアプリケーション側でJTAGポートを停止すれば、4本のポート端子を他の用途に使えるようになります。

JTAGインタフェースとデバッグエンジンの動作は、MAXQ2000コアと非同期です。JTAGポート経由の通信がMAXQ2000の動作クロックレートと等しい必要はありません。なお、TCKの周波数は、MAXQ2000のシステムクロックの1/8以下となっています。

MAXQ2000がコード実行中にも、デバッグエンジンからブレークポイントの設定を読んだり書いたりすることが可能です。このモードはいわゆるバックグラウンドモードであり、デバッグエンジンがCPUコアとは無関係に動作します。

メモリやレジスタの読取りや書込みなどの動作をする場合、デバッグエンジンがMAXQ2000コアを制御し、ユーティリティROMにあるデバッグサービスルーチンの1つに実行を分岐させます。これはデバッグモードと呼ばれ、通常のプログラミング実行にデバッグエンジンが割り込む形です。この場合、ユーザアプリケーションは一時中断され、デバッグ機能の実行が終了したら、割込ルーチンの処理と同じ方法でユーザアプリケーションの実行が再開されます。

JTAG TAPポートは、アプリケーションソフトウェア用に使用されないため、JTAGポートを構成するポート端子をアプリケーションから利用することも可能です。デバッグが必要となるコードは、すべてユーティリティROM上に存在するため、デバッグで使われるシステムリソースは若干のデータSRAMと1レベルのプログラムスタック(デバッグルーチンが呼び出されたとき、戻りアドレスが保存される)だけです。データSRAMの最上位19バイト(0x07EDから0x07FFまでのアドレス)が、デバッグサービスルーチン用に予約されています。インサーキットデバッグ機能を使用しないアプリケーションでは、この部分のデータSRAMもアプリケーションで利用することができます。

JTAG経由の内蔵フラッシュメモリプログラム

JTAG TAPポートは、ブートローダにも使用されます(デバッグ機能を使用しない場合も、ブートローダは使用されます)。JTAG TAPインタフェースのコンフィギュレーションビットの3カ所を設定し、MAXQ2000をリセットから復帰させると、ユーティリティROMの内蔵ブートローダルーチンに処理が移ります。ブートローダへのアクセスを制御するコンフィギュレーションビットは、次のとおりです。

- SPE: System Program Enableビット(ICDF.1)。このビットを1にすると、システムリセット後、ユーティリティROMのブートローダルーチンを実行します。
- PSS[1:0]: Programming Source Select(ICDF.3-2)。この2ビットの設定によって、JTAGポート(PSS[1:0] == 00b)とシリアル0 UART(PSS[1:0] == 01b)のどちらがブートローダとの通信に使用されるかが決まります。

これらのビットを設定し、MAXQ2000をリセットから復帰させると、ユーティリティROMのブートローダが選択されたポート(JTAGまたはシリアル0 UART)経由でホストシステムと通信を始めます。どちらのポートを経由する場合も、以下の機能を持つ同じプロトコルを使用します。

- MAXQ2000のIDバナーの読出し(ユーティリティROMバージョンの特定)。
- プログラムとデータ用内蔵メモリ容量の返送。
- プログラム用内蔵フラッシュメモリの読出し、書込み、検証、CRCの実行。
- 内蔵データSRAMの読出し、書込み、検証、CRCの実行。
- パスワードの照合(メモリ読出し/書込みコマンドをアンロック)。

ブートローダの通信はJTAGポートではなくシリアル0 UART経由で行うことも可能ですが、ブートローダをシリアル通信モードにするためにはJTAGインタフェースを使う必要があります。または、SPEビットとPSSビットの設定によって、MAXQ2000のリセット後、シリアル通信モードでブートローダを起動することも可能です(MAXQ2000をリセットするためには、ウォッチドッグタイマの満了を待つか、外部からハードウェア的な方法によります)。ブートローダの起動方法は(ポート端子に信号出力するなど)、アプリケーション側で決定する必要があります。

MAXQ2000のデバッグ機能は、マイクロコントローラコアと緊密に統合されたハードウェアデバッグエンジンによってコントロールされます。

JTAG TAPインタフェースのコンフィギュレーションビットを3カ所設定し、MAXQ2000をリセットから復帰させると、ユーティリティROMの内蔵ブートローダルーチンに処理が移ります。

デバッグ機能とブートローダ機能のパスワードによる保護

MAXQ2000のデバッグ機能やブートローダ機能へのアクセスは、簡単なパスワード保護による制限が施されています。ホストシステムは、メモリやシステムレジスタ、周辺回路レジスタの内容を読み出したり書き換えたりする前に、パスワードによってアクセス許可を取得する必要があります。

パスワード長は16ワードか32バイトです。パスワード値は内蔵フラッシュメモリの0x0010~0x001Fのワード位置にあります。この値は、アプリケーションに静的配列として含めることも可能ですし、その位置に記録されたコマンドコードの値とすることも可能です。いずれの場合も、アプリケーションがロードされる時、パスワードが自動的に書き込まれます。アプリケーションがロードされていない状態では、デフォルト値として、すべてのワード内容が0xFFFFというパスワードが使用されます。

パスワードが分からなくても、ブートローダからMAXQ2000の内蔵フラッシュメモリを消去することは可能です。このとき、パスワード値はクリアになり(すべてが0xFFFFのワードとなり)、プログラミング操作やデバッグ操作を行うことができますようになります。パスワード保護は、正しい32バイトのパスワードと一致しなければ、MAXQ2000から既存コードを読み出せないようにするためのものです。

Serial-to-JTAGアダプタモジュールの用法

MAXQ2000マイクロコントローラ統合開発環境(MAXIDEやIAR Embedded Workbenchなど)では、MAXQ2000 JTAGインタフェースとの通信をサポートするライブラリが用意されています。ただし、このソフトウェアを実行するPCには、ふつう、JTAGポートが存在しないため、これら2つのシステム間のインタフェースにはハードウェア層が必要となります。

このようなインタフェースに関する問題のターンキーソリューションとなるのが、MAXQ2000評価キットのserial-to-JTAGアダプタモジュールです(図1)。PC上で実行されるソフトウェア(IAR Embedded Workbenchなど)は、標準COMシリアルポート経由でserial-to-JTAGアダプタモジュールと通信を行います。これを受けてserial-to-JTAGアダプタモジュールは、MAXQ2000のJTAGポートとのインタフェースをとり、ブートローダやデバッグエンジンにコマンドを転送します。アダプタモジュールはレベル変換も行うことができるため、MAXQマイクロコントローラがどのような電源電圧で動作していても問題はありません。また、PCがJTAG波形に必要とする正確なタイミングを作る必要もなくなります。

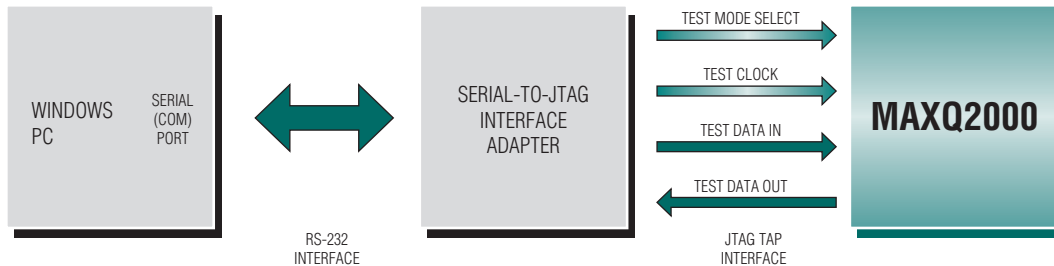


図1. Serial-to-JTAGアダプタモジュールはPCのソフトウェアを動作させることによって、MAXQ2000マイクロコントローラのJTAG TAPインタフェースへのアクセスを可能にします。

MAXQ2000評価キットハードウェアの用法

MAXQ2000評価キットは、以下のような特長を持つMAXQ2000マイクロコントローラの完全なハードウェア開発環境を提供します。

- MAXQ2000コアとVDDIO電源レール用の電源を搭載。
- VDDIO電源レール及びVLCD電源レールに使用可能な可変電源(1.8Vから3.6V)。
- MAXQ2000の各信号と各電源電圧のすべてに対応したヘッダピンを搭載。
- 専用のLCDドーターボードコネクタ。
- 3V、3.5桁のスタティックLCDディスプレイを搭載したLCDドーターボードを添付。

- フローコントロールラインを含んでシリアル0 UART用のRS-232レベルのドライバを完備。
- 外部割込及びマイクロコントローラシステムリセット用のプッシュボタン。
- MAXQ2000 SPIバスインタフェースに接続されたMAX1407(多目的ADC/DAC IC)を搭載。
- iButton[®]クリップと1-Wire EEPROM ICを持つ1-Wire[®]インタフェース。
- P0.7からP0.0までのポートピンのレベルを棒グラフで表示するLEDディスプレイ。
- アプリケーションのロード及びインシステムデバッグを行うJTAGインタフェースを装備。

MAXQ2000評価キットと serial-to-JTAGアダプタモジュールを使うと、IAR Embedded Workbench から、MAXQ2000が持つ JTAGベースのブートローダやインサーキットデバッグ機能のすべてにアクセスすることができます。

アプリケーション開発を行うためにMAXQ2000評価キットボードとserial-to-JTAGインタフェースモジュールをセットアップするのは簡単です。単純に以下のステップでボードの接続を行ってください。

- 1) 5VのDC安定化電源(センタポストをプラスとして、±5%)をserial-to-JTAGボードの電源ジャック(J2)に接続。
- 2) 5V~9VのDC電源をMAXQ2000評価キットボードの電源ジャック(J1)に接続。
- 3) ストレートタイプのDB9シリアルケーブルで、serial-to-JTAGボードのJ1コネクタとPCのCOMポートを接続。
- 4) JTAGアダプタケーブルで、serial-to-JTAGボードの1 x 9コネクタのP2とMAXQ2000評価キットボードの2 x 6コネクタのJ4を接続。
- 5) 2つのDC電源のスイッチをオン。
- 6) 通常動作では、MAXQ2000評価キットボードのDIPスイッチは、すべてOFFとしなければなりません。

IAR Embedded Workbenchによるアプリケーション開発

IAR Embedded Workbench開発環境では、C言語やアセンブラを使ってMAXQ2000用アプリケーションの開発を行うことができます。MAXQ2000評価キットボードとserial-to-JTAGアダプタモジュールを用いる前述のハードウェア構成を使うと、IAR Embedded Workbenchから、MAXQ2000が持つJTAGベースのブートローダやインサーキットデバッグ機能のすべてにアクセスすることができます。

IAR Embedded WorkbenchはMAXQ2000アプリケーションの開発のために、以下の特長を備えています。

- コンパイルしたアプリケーションをMAXQ2000内蔵プログラムフラッシュメモリにロード。
- C言語やアセンブリ言語レベルで、1ステップごとの実行(トレース)。
- コード、データ、ハードウェアスタック、及びユーティリティROMメモリの表示。
- スタックトレースの呼出し。
- C言語またはアセンブリ言語レベルでのブレークポイントの設定。
- MAXQ2000のシステムレジスタと周辺機器レジスタ内容の閲覧と編集。

MAXQ2000用プロジェクトの作成とコンパイル

IAR Embedded WorkbenchはMAXQマイクロコントローラファミリに対する統合サポートが用意されているため、若干の設定を行うだけで、簡単に、MAXQ2000マイクロコントローラの新規プロジェクトを作成することができます。

IARを起動したら、メニューから「File」、続いて「New」を選択してください。「New」ダイアログボックスでWorkspaceを選択し、「Ok」をクリックしてください。プロジェクトワークスペースの名前を入力し(“.eww”という拡張子を持つファイルとなります)、「Save」をクリックしてください。

IAR Embedded Workbench開発環境では、C言語やアセンブラを使ってMAXQ2000用アプリケーションの開発が行うことができます。

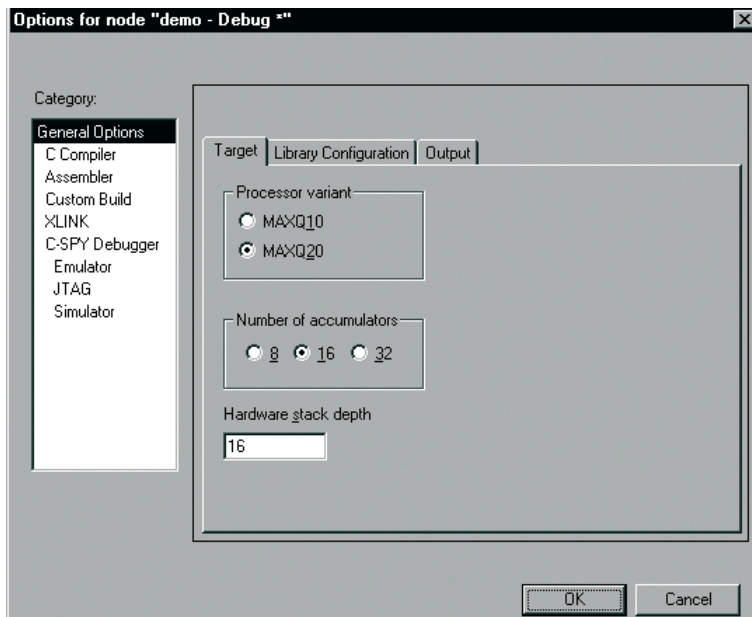


図2. OptionsダイアログボックスのGeneral Optionsタブでは、プロセッサコアタイプ (MAXQ10/20)、使用可能なアキュムレータ数、ハードウェアスタック深さを設定することができます。ここに示す設定は、MAXQ2000用です。

「Options」ダイアログボックスのC-SPY Debuggerタブでは、MAXQ2000の場合は、以下の設定を行う必要があります(図3)。

- 「Driver」で「JTAG」を選び、PC COMポート経由でserial-to-JTAGインタフェースに接続できるようにしてください。選択肢は、この他に、「Simulator」(MAXQ2000ソフトウェアシミュレータを使用する場合)と「Emulator」(MAXQ2000インサーキットエミュレータを使用する場合)があります。

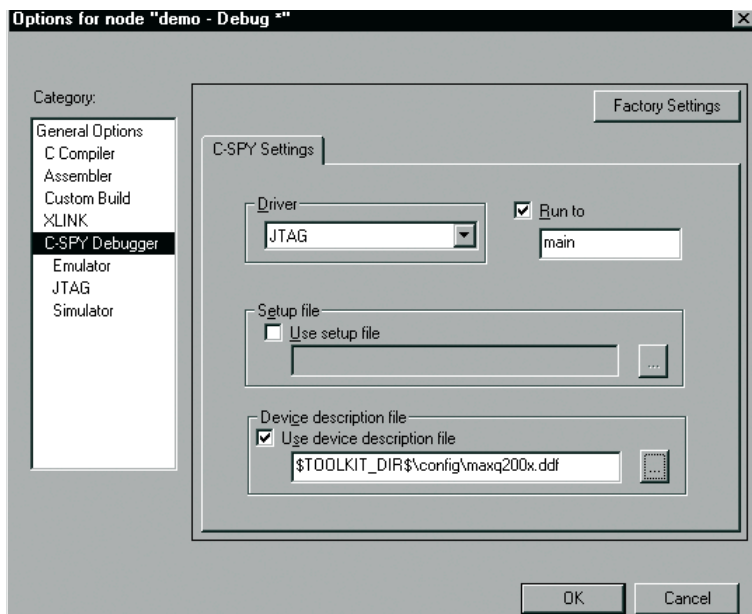


図3. OptionsダイアログボックスのC-SPY Debuggerタブではデバッグセッションに関する設定が可能です。ここに示す設定は、serial-to-JTAGアダプタモジュール経由でMAXQ2000のデバッグを行う場合のもです。

IARによるデバッグ

デバッグセッションでは、「Step Over」(F10)、「Step Into」(F11)、及び「Step Out」(Shift+F11)を使って、プロジェクトのCプログラムをトレースすることができます。コードを実行するためには、メニューから「Debug」、続いて「Go」を選ぶか、またはF5キーを押してください。

ワークスペースウィンドウが開いたら、「Project」メニューから「Create New Project」を選んでください。新規プロジェクトのデフォルトは、MAXQツールチェーンとなっています。新規プロジェクトの名前を入力し(“*.ewp”という拡張子を持つファイルとなります)、「Create」をクリックしてください。

次に、「Project」メニューから「Settings」を選んでください。すると、図2のように、新規プロジェクトの設定を行うダイアログボックスが開かれます。

MAXQ2000マイクロコントローラの場合、「Options」ダイアログボックスの「General Options」タブで以下の設定を行う必要があります。

- 「Processor Variant」では「MAXQ20」を選択 (MAXQ2000にはMAXQ20タイプのコアを搭載)。
- 「Number of accumulators」を16とする。
- 「Hardware stack depth」を16とする。

- 「Use Device Description File」ボックスをチェックする必要があります。デバイス記述ファイル(*.ddf)には、MAXQ2000マイクロコントローラ用のファイル(maxq200x.ddf)を指定しなければなりません。このファイルは、各MAXQマイクロコントローラのメモリ空間や周辺機器レジスタセットをIAR環境から使用するための情報を記述したものです。

「Options」ダイアログボックスのJTAGセクションには、serial-to-JTAGボードとの接続に使用するCOMポートについて記述する「Command line options」フィールドがあります。図4に、COMポート1を使用する場合の設定を示します。

プロジェクトの設定が終わったら、「Project」メニューから「Add Files」を選び、プロジェクトにCコードファイルを追加してください。プロジェクトファイルの追加が終わったら、「Project」メニューから「Make」を選び、プロジェクトのコンパイルを行った後、「Project」メニューから「Debug」を選んでデバッグセッションに入ってください。これにより、自動的に、コンパイルされたプロジェクトがJTAGインタフェース経由でダウンロードされ、図5に示すように、IARがデバッグモードに入ります。

アドレスブレークポイントのセットまたはクリアは、ソースコードのライン上にカーソルを置き、ツールバーの「Toggle Breakpoint」ボタンをクリックすることによって行うことができます。同時に設定することができるブレークポイントは、4カ所までです。

「Memory」ウィンドウには、「Code」(内蔵フラッシュメモリ)、「Data」(内蔵SRAM)、「Hw stack」(内蔵16レベルスタック)、及びユーティリティROMメモリを表示させることができます。表示フォーマットはバイト、ワード、ダブルワードが選べ、表示は16進(全フォーマット)とASCII(バイト幅のみ)の両方が可能です。

「Register」ウィンドウには、MAXQ2000のシステムレジスタと周辺機器レジスタが、以下の論理グループに分けて表示されます。

- 「CPU Registers」: アキュムレータレジスタとアキュムレータコントロールレジスタ、データポインタレジスタとデータポインタコントロールレジスタ、命令ポインタ、ループカウンタ、及びプログラムステータスフラグ。
- 「Interrupt Control」: 割込ベクトル、モジュールマスク、及び識別レジスタ。
- 「Cycles」: 実行された命令サイクル数を表示。
- 「Parallel Ports」: P0、P1、P2、P3、及びP4の入力レジスタ、出力レジスタ、及びポート方向レジスタ。
- 「External Interrupt」: 外部割込のイネーブル、エッジセレクト、及びフラグレジスタ。
- 「Timers」: タイマ/カウンタ0~2のレジスタ。
- 「Serial Port」: SPIポートとシリアルポートのコントロールレジスタとバッファレジスタ。
- 「Multiplier」: ハードウェア乗算モジュール関連のレジスタ。

書き込み可能なレジスタは、レジスタ値をクリックし、新しい値を入力することにより、書き換えられます。レジスタ名の横にある+/-をクリックすると、レジスタのビットフィールドを展開したり戻すことができます。

まとめ

IAR Embedded Workbenchの持つ高レベルのCプロジェクトによる環境をMAXQ2000が持つ低レベルのデバッグインタフェースと共に使用することによって、C言語レベルやアセンブリ言語レベルで細かいデバッグを行うことが可能となります。MAXQ2000に内蔵されたデバッグ及びインサーキットプログラミング機能と、システムリソースの消費が少ないことが、アプリケーション開発プロセスと完成したプロジェクトの最終リリースとで同じハードウェア設計を利用することを可能とします。

1-Wire及びButtonは、Dallas Semiconductor Corp.の登録商標です。

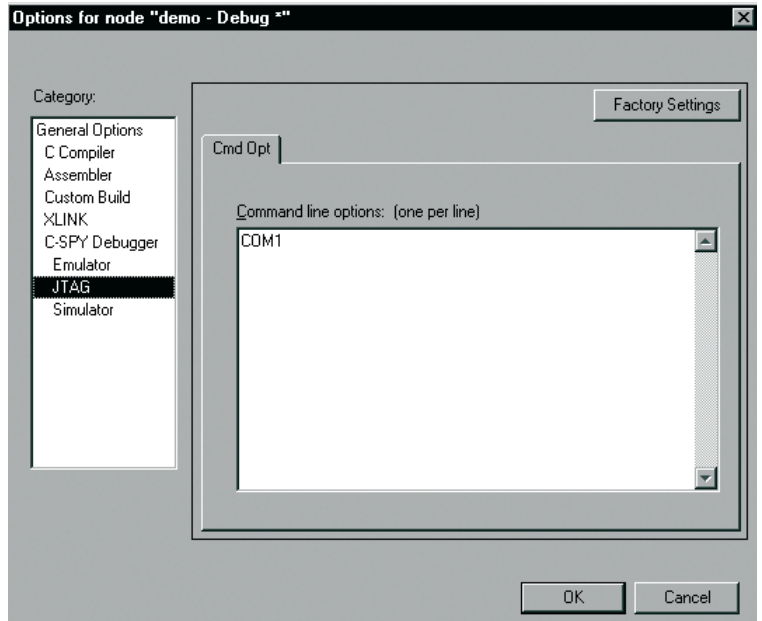


図4. OptionsダイアログにあるC-SPY DebuggerのJTAGセクションでは、serial-to-JTAGアダプタモジュールの設定を変更することができます。ここに示す設定は、serial-to-JTAGアダプタをPCのCOM1ポートに接続するときのものです。

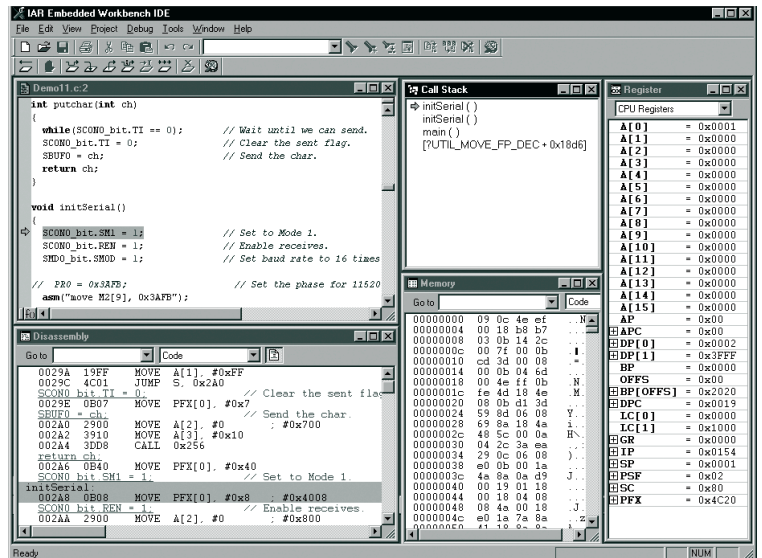


図5. Serial-to-JTAGアダプタモジュールを使用すると、IAR Embedded WorkbenchからMAXQ2000のステップごとの実行をしたり、オンチップメモリやレジスタの値を読み出したり、書き換えることができます。