イン・システム・プログラマビリティ・ガイドライン

この資料は英語版を翻訳したもので、内容に相違が生じる場合には原文を優先します。こちらの日本語版は参考用としてご利用ください。設計の際には、最新の英語版で内容をご確認ください。

AN-100-4.0 アプリケーション・ノート

このアプリケーション・ノートでは、イン・システム・プログラマビリティ（ISP）を使用して適切にデザインするためのガイドラインを説明します。アルテラの ISP 対応のデバイスは、IEEE Std. 1149.1 JTAG（Joint Test Action Group）インタフェースを介してイン・システムでプログラムおよび再プログラムできます。このインタフェースによって、デバイスをプログラムし、PCB の機能を 1 つの製造ステップでテストできるため、テスト時間や組み立てコストを低減できます。

「Time-to-Market」に対する要求の高まりに伴い、デザイン・エンジニアは開発と製造を問題なく進めるために、高度なシステム・レベルの製品を必要とします。ISP を備えたプログラマブル・ロジック・デバイス（PLD）は、開発時間の短縮、イン・フィールド・アップグレードと製造フローの簡略化、在庫コストの低減、およびプリント基板（PCB）テスト機能の改善に役立ちます。

このアプリケーション・ノートでは、以下について説明します。

全体的な ISP ガイドライン

この項では、ISP 対応のデバイスを適切にデザインするためのガイドラインを示します。どのデザイン実装でもこれらのガイドラインに従うことが必要です。

動作条件

アルテラの各デバイスには、適正動作に必要なパラメータ定格（動作条件）がいくつかあります。ユーザー・モードではこれらの条件を超えることがあり、その場合も正しく動作しますが、イン・システム・プログラミング中はこれらの条件を超えないようにしなければなりません。イン・システム・プログラミング中に動作条件のいずれかに違反すると、プログラミング障害が発生したり、デバイスが不正にプログラムされることがあります。ISP が機能するには、すべての I/O バンクの V_{CCIO} とデバイスの V_{CCINT} を完全にパワーアップする必要があります。

101 Innovation Drive
San Jose, CA 95134
www.altera.com

2010 年 12 月
Altera Corporation

Subscribe

ISP 電圧

MAX® II および MAX V デバイスの場合、イン・システム・プログラミング中は、デバイスのフラッシュ・セルが確実に正しくプログラムされるように、V_{CCINT} および V_{CCIO} レベルを V_{CCINT} および V_{CCIO} ピン上で維持する必要があります。V_{CCINT} レベルおよび V_{CCIO} レベルは、該当する「*Device Family Datasheet*」の「*Operating Conditions*」の表に示されます。V_{CCINT} および V_{CCIO} の使用は、コマーシャルおよびインダストリアル温度範囲の製品の両方に適用されます。

MAX 3000、MAX 7000、および MAX 9000 デバイスは、V_{CCISP} と呼ばれる特定の ISP 電圧を持っています。デバイスの EEPROM セルを正しくプログラムするには、ISP 時に VCCINT ピンまたは V_{CCISP} レベルを維持する必要があります（例えば、$V_{CCINT} = V_{CCISP}$）。V_{CCISP} の仕様は、コマーシャルおよびインダストリアル温度範囲の製品の両方に適用されます。

ISP 時の消費電力がユーザー・モード時の消費電力を超えることがあるため、両方のモードで ISP 設定を調整して正しい電圧レベルを維持する必要がある場合があります。ISP 時の消費電力がユーザー・モード時の消費電力を超えることがあるため、両方のモードで ISP 設定を調整して正しい電圧レベルを維持する必要がある場合があります。アルテラでは、オシロスコープを使用してデバイスの VCCINT ピン上の V_{CCISP} レベルをテストすることが推奨されています。最初に、オシロスコープのトリガ・レベルを該当するデバイス・ファミリ・データシートの推奨動作条件表に記載された最小 V_{CC} レベルに設定することで、V_{CCISP} レベルをテストします。デバイスのピンでプローブされる、VCCINT とグランド間の電圧を測定します。そして、オシロスコープのトリガ・レベルを該当するデバイス・ファミリ・データシートの推奨動作条件表に記載された最大 V_{CC} レベルに設定し、このテストを繰り返します。オシロスコープがいずれかの電圧レベルでトリガされる場合、プログラミング設定を調整する必要があります。

入力電圧

各デバイスのファミリ・データシートでは、絶対最大定格表および推奨動作条件表にデバイスの入力電圧仕様を記載しています。絶対最大定格表の入力電圧は、デバイスが恒久的な損傷を受けずに許容できる最大電圧を表します。

推奨動作条件表は、デバイスが正常に動作するための電圧範囲を規定しています。イン・システム・プログラミング中に遷移するすべてのピンがグランドまたは V_{CC} オーバーシュートを発生しないことを確認してください。通常、オーバーシュート問題は自走クロックまたはイン・システム・プログラミング中にトグル可能なデータ・バスで発生します。1.0 V を超えるオーバーシュートを持つピンに直列終端が必要です。

アルテラの CPLD の推奨動作条件および絶対最大定格について詳しくは、以下の資料を参照してください。

- 「*MAX V デバイス・ハンドブック*」のセクション 1 での「*DC and Switching Characteristics for MAX V Devices*」
- 「*MAX V デバイス・ハンドブック*」の「*DC and Switching Characteristics*」
- *MAX 3000A Programmable Logic Device Family Datasheet*
- *MAX 7000 Programmable Logic Device Family Datasheet*
- *MAX 7000A Programmable Logic Device Family Datasheet*
- *MAX 7000B Programmable Logic Device Family Datasheet*

- *MAX 9000 Programmable Logic Device Family Datasheet*

デバイスの終端処理について詳しくは、「AN 75: High-Speed Board Designs」を参照してください。

イン・システム・プログラミング中の UFM 操作

MAX II または MAX V デバイスの UFM に書き込みまたは消去可能なデザインでは、UFM のすべての消去または書き込み操作が必ず ISP セッション（スタンドアロン検証、検査、セキュリティ・ビットの設定、および UFM の内容の読み出しを含む）の開始前に完了する必要があります。デバイスが回復不可能な状態になる可能性があるため、UFM の消去または書き込み操作の実行中には、絶対に ISP セッションを開始しないでください。ただし、この制限は UFM の読み出し操作には適用されません。

MAX II または MAX V デバイスに対する ISP 動作の試行前に、UFM の消去または書き込み操作を確実に完了できない場合、リアルタイム ISP 機能をイネーブルにする必要があります。この機能を適切に使用すると、UFM/ISP 操作の競合を防止するのに役立ちます。リアルタイム ISP 機能をイネーブルにすると、Quartus II ソフトウェアまたは Jam（**.jam**）/Jam Byte-Code（**.jbc**）ファイルは、500 ms 待機してから動作を開始します。この待機時間は、1 つの UFM セクタを消去するのに要する時間と同じです（つまり、リアルタイム ISP プログラミング・アルゴリズムは、前に開始された UFM 消去シーケンスが完了するまで待機します）。

ただし、リアルタイム ISP 機能を使用している場合、この間は他の UFM 操作は許可されません（アドレス・シフト、データ・シフト、および読み出し、書き込み、または消去操作は不可）。これは、ALTUFM_NONE メガファンクションの `RTP_BUSY` 信号をモニタすることによって制御できます。リアルタイム ISP を実行する場合、UFM ブロックの `RTP_BUSY` 出力信号は High に変化します。この信号をモニタして、リアルタイム ISP が完了するまで、ロジック・アレイからすべての UFM 操作を停止させることが確実にできます。このユーザ生成コントロール・ロジックは、自動生成ロジックを持たない ALTUFM_NONE メガファンクションに対してのみ必要です。ALTUFM メガファンクションのその他のパラメータ・エディタ（ALTUFM_PARALLEL、ALTUFM_SPI、ALTUFM_I2C）は、`RTP_BUSY` 信号を自動的にモニタし、リアルタイム ISP 動作が進行中の場合に UFM に対する操作を中止するコントロール・ロジックを備えています。

イン・システム・プログラミングの割り込み

部分的にプログラムされたデバイスが予期しない動作を行うことを防止するために、アルテラは、プログラミング・プロセスへの割り込みを推奨していません。また、部分的にプログラムされたデバイスによって、信号の衝突が発生する可能性があります。この衝突は、デバイスの破壊およびボード上のほかのデバイスへの影響を引き起こす可能性があります。

ただし、MAX II および MAX V デバイスには正常なプログラム・シーケンスの最後でのみ設定される `ISP_DONE` ビットがあります。I/O ピンはこのビットが設定されている場合にのみドライブ・アウトします。これによって、部分的にプログラムされたデバイスがドライブ・アウトしたり、予期しない動作を行うことを防止します。

MultiVolt デバイスおよびパワーアップ・シーケンス

イン・システム・プログラミングまたはバウンダリ・スキャン・テスト中に JTAG 回路が正しく動作するには、JTAG チェイン内のすべてのデバイスが同じ状態にあることが必要です。したがって、複数の電源電圧を持つシステムでは、チェイン内のすべてのデバイスが完全にパワーアップされるまで、JTAG ピンをテスト・ロジック・リセット状態に維持する必要があります。複数の電源を使用するシステムでは、すべての電圧レベルを同時に供給できないため、この手順は特に重要です。

MAX デバイスは、MultiVolt™ 機能を備えているので、V_{CCINT} や各 I/O バンクに対する V_{CCIO} など、複数の電源を使用できます。V_{CCINT} は JTAG 回路に電源を供給します。V_{CCIO} は、入力ピンと TDO などの出力ピンの出力ドライバに電源を供給します。したがって、2 つの電源電圧を使用する場合、両方の電源がオンになるまで、JTAG 回路をテスト・ロジック・リセット状態に維持する必要があります。JTAG ピンをテスト・ロジック・リセット状態に維持しない場合、イン・システム・プログラミング・エラーが発生することがあります。

V_{CCIO} より前の V_{CCINT} のパワーアップ

V_{CCIO} より前に V_{CCINT} がパワーアップされると、JTAG 回路はアクティブになりますが、信号をドライブアウトできなくなります。このように、TCK が変化すると、ステート・マシンは不定の JTAG ステートに遷移します。TMS および TCK ピンが V_{CCIO} に接続され、V_{CCIO} がパワーアップされていない場合、JTAG 信号はフローティング状態のままです。フローティング状態の値はデバイスを意図しない JTAG ステートに遷移させ、V_{CCIO} が最終的にパワーアップされたときに誤動作することがあります。したがって、7 ページの「IEEE Std. 1149.1 回路のディセーブル」に示すようにすべての JTAG 信号をディセーブルする必要があります。

V_{CCINT} より前の V_{CCIO} のパワーアップ

V_{CCINT} より前に V_{CCIO} がパワーアップされると、JTAG 回路はアクティブにならず、TDO ピンがトライ・ステートになります。JTAG 回路がアクティブでなくても、JTAG チェインの次のデバイスが V_{CCIO} と同じトレースでパワーアップされた場合、その JTAG 回路はテスト・ロジック・リセット状態に留まる必要があります。TMS 信号と TCK 信号はすべて共通のため、チェイン内のすべてのデバイスでディセーブルにする必要があります。したがって、TCK を Low にプルダウン、TMS を High にプルアップすることによって、JTAG ピンをディセーブルする必要があります。

イン・システム・プログラミング中にトライ・ステートになる I/O ピン

デフォルトでは、すべてのデバイス I/O ピンはイン・システム・プログラミング中にトライ・ステートになります。さらに、MAX CPLD は、ISP 中にウィーク・プルアップ抵抗を提供します。このウィーク・プルアップ抵抗の目的は、トライ・ステートの I/O ピン上で外部プルアップ抵抗を不要にすることです。

信号をドライブするために使用される、イン・システム・プログラミング中に特定の値（例えば、出力イネーブルまたはチップ・イネーブル信号）を必要とするピンに対しては、十分なプルアップ抵抗またはプルダウン抵抗を追加する必要があります。プルアップ抵抗またはプルダウン抵抗が追加されていない場合、ISP 時にデバイスに高い電流（ボード上の衝突による）、認識されないデバイスまたは検証エラーによる ISP 障害、あるいは ISP 障害後のパワーアップが発生することがあります。

MAX II および MAX V デバイスの場合、各ピンが ISP 時に特定のステートにクランプされるために、イン・システム・プログラミング・クランプ機能またはリアルタイム ISP 機能を使用できます。

MAX II デバイスおよび MAX V デバイスのイン・システム・プログラミングについて詳しくは、下記の資料の「In-System Programming Clamp and Real-Time ISP」の項を参照してください。

- 『MAX V デバイス・ハンドブック』のセクション II の「JTAG and In-System Programmability in MAX V Devices」の章
- 『MAX II デバイス・ハンドブック』のセクション I の「JTAG and In-System Programmability」の章

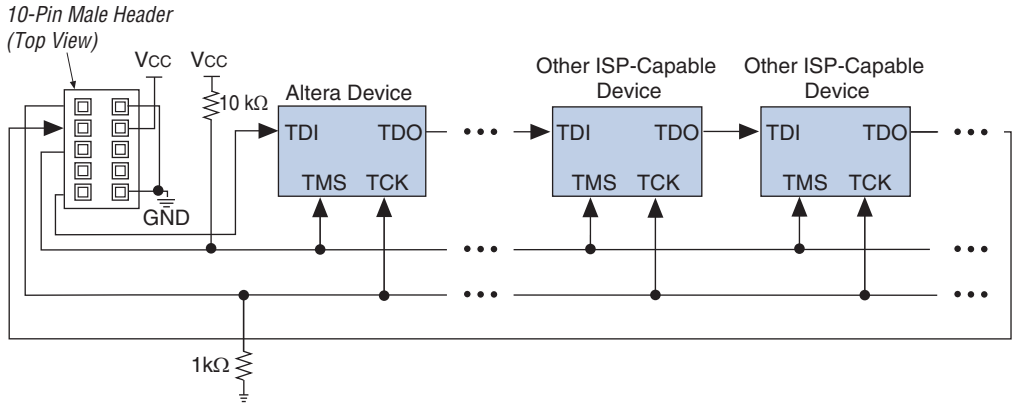
イン・システム・プログラミング中の JTAG ピンのプルアップおよびプルダウン

イン・システム・プログラミング・モードで動作するアルテラ・デバイスには、TDI、TDO、TMS、および TCK の 4 本のピンが必要です。

4 本の JTAG ピンのうちの 3 本は、内部ウィーク・プルアップまたはプルダウン抵抗を備えています。TDI ピンと TMS ピンは内部ウィーク・プルアップ抵抗を備えており、TCK ピンは内部ウィーク・プルダウン抵抗を備えています。ただし、JTAG チェインでのデバイス・プログラミングの場合、内部プルアップ抵抗または内部プルダウン抵抗を持たないデバイスが存在する場合があります。アルテラは、10k 抵抗を通して TMS 信号を外部で High にプルアップし、1k 抵抗を通して TCK 信号を外部で Low にプルダウすることを推奨しています。TDI 信号の外部プルアップ抵抗は、任意で構いません。

図 1 に、アルテラ・デバイスでの JTAG チェインの TMS と TCK に対する外部プルアップおよびプルダウン抵抗を示します。TDO ピンには、内部プルアップ抵抗やプルダウン抵抗がなく、外部プルアップ抵抗やプルダウン抵抗も必要としません。

図 1. アルテラ・デバイスでの JTAG チェインの TMS と TCK に対する外部プルアップ抵抗および外部プルダウン抵抗



TCK 信号から入力がある場合でも、TAP（test access port）コントローラを TEST_LOGIC または RESET 状態に維持するために TMS 信号は High にプルアップされます。TCK に High のパルスが生成されないようにするために、パワーアップ中に TCK ピンを Low にプルダウンする必要があります。プルアップ抵抗への電源供給が増加すると、TCK 信号が High のパルスを生成し、それによって TAP コントローラが予期しない状態に変化する可能性があるため、TCK 信号を High にプルアップすることは推奨されません。

各ピンの説明および機能について詳しくは、下記の資料を参照してください。

- 『*MAX V デバイス・ハンドブック*』のセクション II の「*JTAG Boundary-Scan Testing for MAX V Devices*」の章
- 『*MAX II デバイス・ハンドブック*』のセクション IV の「*IEEE 1149.1 (JTAG) Boundary-Scan Testing for MAX II Devices*」の章

IEEE Std. 1149.1 信号

この項では、IEEE Std. 1149.1（JTAG）インタフェースを使用したプログラミングのガイドラインを示します。

TCK 信号

大部分のイン・システム・プログラミング障害は、TCK 信号のノイズによって発生します。立ち上がりエッジまたは立ち下がりエッジ遷移時のノイズによって、IEEE Std. 1149.1 TAP コントローラに不正にクロックが供給される可能性があります。不正なクロックが発生すると、ステート・マシンが不定状態に遷移し、イン・システム・プログラミング障害の原因になることがあります。

TCK 信号はチェイン内のすべての IEEE Std. 1149.1 デバイスをパラレルにドライブしなければならないため、信号のファンアウトが大きくなることがあります。ファンアウトの大きいその他のユーザー・モード・クロックと同様に、クロック・ツリーを活用して、シグナル・インテグリティを維持する必要があります。クロック・インテグリティの問題によって発生する代表的なエラーには、無効な ID メッセージ、ブランク・チェック・エラー、検証エラーなどがあります。

アルテラは、内部ウィーク・プルダウン抵抗または外部 1 kΩ 抵抗を使用して、TCK 信号を Low にプルダウンすることを推奨しています。

高速 TCK エッジがボード・インダクタンスと結合すると、オーバーシュート問題が発生することがあります。この組み合わせが生じる場合、トレース上のインダクタンスを低減するか、または低速なスルー・レートの TTL（Transistor-to-Transistor Logic）ドライバ・チップを選択することによって、スイッチング・レートを低くする必要があります。デバイスの入力仕様に違反する可能性があるため、エッジ・レートを低速化する目的で抵抗とコンデンサ（RC）の回路網を使用してはいけません。ドライバ・チップを使用すると、エッジ・レートが過度に低下することを回避できます。アルテラは、パワーアップ後にグリッチを発生しないドライバ・チップの使用を推奨しています。

ダウンロード・ケーブルを使用したプログラミング

アルテラのデバイスは、MasterBlaster™、ByteBlasterMV™、ByteBlaster™ II、ByteBlaster、BitBlaster™、EthernetBlaster、または USB Blaster ダウンロード・ケーブルを使用してプログラムできます。Quartus II プログラマを搭載した PC または UNIX ワークステーションを使用すると、ダウンロード・ケーブルを介して Programmer Object File（**.pof**）、**.jam** ファイル、または **.jbc** ファイルをアルテラ・デバイスにダウンロードできます。

ダウンロード・ケーブルを使用して、JTAG チェインに 3 個以上のデバイスが含まれている場合、アルテラはチェインにバッファを追加することを推奨しています。ノイズを最小にするには、遷移が低速なバッファを選択する必要がありますが、その場合も遷移レートが必ず JTAG チェインの TCK 性能要件に適合するようにする必要があります。

ダウンロード・ケーブルを延長する必要がある場合は、標準の PC パラレル・ポートまたは USB ポート・ケーブルをダウンロード・ケーブルに取り付けることができます。ダウンロード・ケーブルの 10 ピン・ヘッダ部分は延長しないでください。ケーブルのこの部分を延長すると、ノイズやイン・システム・プログラミングの問題が発生することがあります。

ダウンロード・ケーブルによって、プログラミング時間が異なります。MasterBlaster、ByteBlasterMV、ByteBlaster II、ByteBlaster、BitBlaster、EthernetBlaster、または USB Blaster ダウンロード・ケーブルについて詳しくは、下記の資料のいずれかを参照してください。

- *MasterBlaster Serial/USB Communications Cable User Guide*
- *ByteBlasterMV Download Cable User Guide*
- *ByteBlaster II Download Cable User Guide*
- *BitBlaster Serial Download Cable Datasheet*
- *USB-Blaster Download Cable User Guide*
- *EthernetBlaster Communications Cable User Guide*

IEEE Std. 1149.1 回路のディセーブル

ISP またはバウンダリ・スキャン・テスト（BST）回路を使用しないデザインでは、IEEE Std. 1149.1 回路をディセーブルすることを推奨します。

表 1 に、必要ないときに IEEE Std. 1149.1 回路をディセーブルする方法を説明しています。

表 1. MAX V デバイスの IEEE Std. 1149.1 回路のディセーブル

<table>
<tr><th>デバイス</th><th>恒久的にディセーブル</th><th>ISP および BST に対してイネーブル、ユーザー・モードに対してディセーブル</th></tr>
<tr><td>MAX II
MAX V</td><td colspan="2">TMS 信号を High にプルアップし、TCK 信号を Low にプルダウンします。あるいは、TCK 信号を High にプルアップする前に、TMS 信号を High にプルアップします。</td></tr>
<tr><td>MAX 7000S
MAX 7000B
MAX 7000A
MAX 7000AE
MAX 3000A</td><td>Quartus® II ソフトウェアで、Enable JTAG BST Support オプションをオフにします。</td><td>TMS 信号を High にプルアップし、TCK 信号を Low にプルダウンします。あるいは、TCK 信号を High にプルアップする前に、TMS 信号を High にプルアップします。</td></tr>
<tr><td>MAX 9000
MAX 9000A</td><td colspan="2">TMS 信号を High にプルアップし、TCK 信号を Low にプルダウンします。あるいは、TCK 信号を High にプルアップする前に、TMS 信号を High にプルアップします。</td></tr>
</table>

IEEE 1149.1 回路のディセーブルについて詳しくは、下記の資料を参照してください。

- 『MAX V デバイス・ハンドブック』のセクション II の 『JTAG Boundary-Scan Testing for MAX V Devices』の章
- 『MAX II デバイス・ハンドブック』のセクション IV の 『IEEE 1149.1 (JTAG) Boundary-Scan Testing for MAX II Devices』の章
- AN 039: IEEE 11.49.1 JTAG Boundary-Scan Testing in Altera Devices

JTAG が 恒久的にディセーブされる場合（MAX 7000S、MAX 7000B、MAX 7000A、MAX 7000AE および MAX 3000A デバイス）

MAX 7000S、MAX 7000B、MAX 7000A、MAX 7000AE、および MAX 3000A デバイスの JTAG ピンは、JTAG ポートまたは I/O ピンのいずれかとして使用できます。Quartus II ソフトウェアでデザインをコンパイルする前に、**Enable JTAG BST Support** オプションのオン / オフによってピンの使い方を指定する必要があります。このオプションをオンにすると、ピンは ISP および BST 用の JTAG ポートとして動作します。このオプションをオフにすると、ピンが I/O ピンとして動作し、ISP または BST が実行できなくなります。

JTAG が恒久的にディセーブルされる場合（MAX V、MAX II、MAX 9000 および MAX 9000A デバイス）

デフォルトでは、MAX V、MAX II、MAX 9000、および MAX 9000A デバイスの JTAG 回路には JTAG 専用ピンと専用回路があるため、常にイネーブルされています。JTAG 回路は、ISP およびバウンダリ・スキャン・テストの間はイネーブルにし、それ以外は常にディセーブルにしなければなりません。したがって、ISP および BST 回路を使用しない場合、JTAG ピンを通してこれらの回路をディセーブルすることができます。JTAG をディセーブルには、JTAG 仕様は TMS 信号を High にプルアップすると指示しますが、TCK 信号については説明していません。アルテラは、TMS 信号を High にプルアップし、TCK 信号を Low にプルダウンすることを推奨しています。TCK 信号を Low にプルダウンすると、パワーアップ・シーケンス中に TCK 信号に立ち上がりエッジが発生することはありません。TCK 信号を Low にプルダウンすると、パワーアップ・シーケンス中に TCK 信号に立ち上がりエッジが発生することはありません。

TCK 信号を High にプルアップしても構いませんが、TMS 信号を High にプルアップした後でのみ実行してください。先に TMS を High にプルアップすると、TCK 信号上の立ち上がりエッジが発生しても、JTAG ステート・マシンがテスト・ロジック・リセット状態から遷移することはありません。

ISP または BST の JTAG イネーブルおよびユーザー・モードの JTAG ディセーブル

JTAG を ISP または BST に使用するアルテラの ISP 対応のデバイス JTAG 回路は、ISP およびバウンダリ・スキャン・テストの間はイネーブルにし、それ以外は常にディセーブルにしなければなりません。JTAG の動作は、JTAG ピンを通して制御されます。MAX V、MAX II、MAX 9000、および MAX 9000A デバイス上の JTAG 回路を恒久的にディセーブルするには、TMS 信号を High にプルアップして TCK 信号を Low にプルダウンし、あるいは TCK 信号を High にプルアップする前に TMS 信号を High にプルアップします。

異なる電圧レベルにおける動作

JTAG チェイン内のデバイスが異なる電圧で動作する場合、デバイスの出力電圧仕様は、後続デバイスの入力電圧仕様を満たす必要があります。デバイスがこの基準に適合しない場合は、レベル・シフタなどの付加的な回路を追加して、電圧レベルを調整する必要があります。例えば、5.0 V デバイスが 2.5 V デバイスをドライブする場合、2.5 V デバイスの入力電圧仕様を満たすように、5.0 V デバイスの出力電圧を調整する必要があります。

JTAG チェインのすべてのデバイスは互いに結線されているため、デバイスのチェインを適切にプログラムするには、最初のデバイスの TDO 出力が後続デバイスの TDI 入力電圧仕様を満たすようにする必要があります。

アルテラのすべての ISP 対応のデバイスには、MultiVolt I/O 機能が搭載されているため、これらのデバイスは異なる電圧を使用するシステムにインタフェースすることができます。すべての MultiVolt デバイスは、3.3 V、2.5 V、1.8 V、または 1.5 V の I/O 動作に設定できます。

MultiVolt デバイスの JTAG ピンは、これらの電圧レベルをサポートしています。

- 「*MAX V デバイス・ハンドブック*」のセクション I の「*MAX V Device Architecture*」
- 「*MAX II デバイス・ハンドブック*」のセクション I の「*MAX II Device Architecture*」
- *MAX 3000A Programmable Logic Device Family Datasheet*
- *MAX 7000 Programmable Logic Device Family Datasheet*
- *MAX 7000A Programmable Logic Device Family Datasheet*
- *MAX 7000B Programmable Logic Device Family Datasheet*
- *MAX 9000 Programmable Logic Device Family Datasheet*

例えば、V_{CCIO} が 3.3 V の場合、JTAG 入力ピンは 1.8 V または 1.5 V 信号を入力できません。

シーケンシャル・プログラミングと同時プログラミングの比較

この項では、シーケンシャル・プログラミングおよび同時プログラミングを使用して、複数のデバイスをプログラムする方法について説明します。シーケンシャル・プログラミングおよび同時プログラミングに対する JTAG チェインの設定は類似しており、プログラミング・アルゴリズムのみ異なります。

シーケンシャル・プログラミング

シーケンシャル・プログラミングとは、チェイン内の複数のデバイスを一度に 1 デバイスずつプログラミングするプロセスです。チェイン内の最初のデバイスのプログラミングが完了すると、次のデバイスがプログラムされます。このシーケンスは、JTAG チェイン内の指定されたデバイスがすべてプログラムされるまで継続されます。デバイスが正常にプログラムされた後に、バイパス・モードになり、チェイン内の後続デバイスにデータを渡すことができます。チェイン内のデバイスは、すべてのデバイスがプログラムされるまでユーザー・モードに入りません。

同時プログラミング

同時プログラミングは、同じファミリ（MAX V ファミリなど）のデバイスをパラレルにプログラムするために使用します。プログラミング時間は、チェイン内で最大のデバイスをプログラムするのに必要な時間より多少長いだけで、シーケンシャル・プログラミングと比べてはるかに短時間です（プログラミング時間は、すべてのデバイスを個別にプログラムする時間の合計に等しくなる）。データ・シフトのクロック・レートを増やすと、さらに多くの時間を節約できます。

Quartus II ソフトウェアで作成された Serial Vector Format ファイル（**.svf**）、**.jam** ファイル、または **.jbc** ファイルを使用してデバイスの同時プログラミングを実行するには、次のステップに従います。

1. Tools メニューの **Programmer** をクリックします。
2. **Add File** をクリックし、デバイスごとにプログラミング・ファイルを選択します。
3. File メニューの **Create/Update** をポイントし、**Create JAM, SVF, or ISC File** をクリックします。
4. File format リストでファイルを指定します。
5. **OK** をクリックします。

シーケンシャル・プログラミングと同時プログラミング

Programmer Object File (**.pof**) および MasterBlaster、ByteBlasterMV、ByteBlaster、または BitBlaster ダウンロード・ケーブルを使用してプログラミングをする場合、シーケンシャル・プログラミングが自動的に選択されます。**.jam** ファイルまたは Serial Vector Format（**.svf**）ファイルを使用する場合、デバイスは下記の順序でプログラムまたはコンフィギュレーションされます。

1. FLEX 10K デバイス（連続的に）
2. APEX™ 20K デバイス（連続的に）
3. MAX 7000S および MAX 7000A デバイス（同時に）
4. MAX 7000AE および MAX 3000A デバイス（同時に）
5. EPC2 デバイス（連続的に）
6. MAX 9000 デバイス（同時に）

デバイスごとに個別にファイルを作成する場合、**.jam** ファイルまたは **.svf** を使用してシーケンシャル・プログラミングを実行できます。この手法では、MAX+PLUS II Programmer で **Configure** ボタンをクリックするまで、FLEX および APEX デバイスはコンフィギュレーションを開始しません。

異なるモードのデバイス

チェイン内にほかのデバイスがプログラミングされて、一部のデバイスが動作中の場合、エラーが発生する可能性があります。このため、MAX 7000S、MAX 7000A、MAX 7000AE、MAX 7000B、および MAX 3000A デバイスは特別の ISP 命令を使用します。この命令により、チェイン内のすべてのデバイスが ISP を完了するまで、デバイスは通常動作に入りません。このモードでは、動作を開始する前に、これらのデバイスはすべてのバウンダリ・スキャン・データを同期転送しながら、同じファ

ミリ内のほかのデバイスがプログラミングを完了するまで待ちます。現在、APEX 20K、FLEX 10K、MAX 9000、および MAX 9000A デバイスはこのモードをサポートしません。これらのデバイスは、すべてのデバイス・ファミリのプログラムまたはコンフィギュレーションが完了するまでトライ・ステート・モードに保持されます。

ISP トラブルシューティング・ガイドライン

この項では、ISP に関連する問題を解決するためのヒントをいくつか示します。

無効 ID およびデバイスが認識されないメッセージ

イン・システム・プログラミング時の最初のステップは、デバイスのシリコン ID をチェックすることです。シリコン ID が一致しない場合、Invalid ID または Unrecognized Device エラーが発生します。

このエラーの一般的な原因を以下に示します。

- 「ダウンロード・ケーブルが正しく接続されていない」
- 「TDO が接続されていない」
- 「JTAG チェインが不完全」
- 「TCK 信号のノイズ」
- 「Jam Player の移植が不適切」

ダウンロード・ケーブルが正しく接続されていない

ダウンロード・ケーブルがパラレルまたは USB ポートに正しく接続されていない場合、またはボードから電源が供給されていない場合は、エラーが発生します

TDO が接続されていない

チェイン内の 1 つのデバイスの TDO ポートが接続されていない場合は、エラーが発生します。イン・システム・プログラミング中に、JTAG チェイン内の各デバイスに JTAG ピンを通してデータをシフト・インおよびシフト・アウトする必要があります。したがって、各デバイスの TDO ポートは後続デバイスの TDI ポートに接続し、最後のデバイスの TDO ポートはダウンロード・ケーブルの TDO ポートに接続しなければなりません。

JTAG チェインが不完全

JTAG チェインが不完全な場合は、エラーが発生します。エラーが不完全な JTAG チェインのために発生しているかどうか確認するには、オシロスコープを使用して、チェイン内の各デバイスから出力されるベクトルをモニタします。イン・システム・プログラミング中に、各デバイスの TDO ポートがトグルしない場合、JTAG チェインは不完全です。

TCK 信号のノイズ

TCK 信号のノイズは、イン･システム・プログラミング・エラーにおいて最も一般的な原因です。立ち上がりエッジまたは立ち下がりエッジでの遷移にノイズがあると、IEEE Std. 1149.1 TAP コントローラに不適切なクロック供給が行われ、ステート・マシンが失われて、イン・システム・プログラミングが失敗します。ノイズのある TCK 信号について詳しくは、6 ページの「TCK 信号」を参照してください。

Jam Player の移植が不適切

Jam Player がプラットフォームに正しく移植されていない場合は、エラーが発生します。エラーが Jam Player によって発生しているかどうかを確認するには、**.jam** ファイルを使用して IDCODE 命令をターゲット・デバイスに発行します。**.jam** ファイルを使用して IDCODE 命令をロードし、次に IDCODE の値をシフト・アウトすることができます。このテストでは、JTAG チェインが正しく設定されているかどうか、JTAG チェインに対して正しく読み出しと書き込みが実行できるかどうかを確認します。

アルテラ・ウェブサイトの「IDCODE Reader Jam File」ページから **idcode.zip** ファイルをダウンロードして、**idcode.jam** ファイルを入手できます。

トラブルシューティングのヒント

この項では、ISP の問題を解決するためのいくつかのヒントを提供します。

JTAG チェイン接続の検証

イン・システム・プログラミングが正常に動作するには、JTAG チェイン内のデバイス数が Quartus II ソフトウェアまたは MAX+Plus II ソフトウェアでレポートされる数と一致しなければなりません。Quartus II ソフトウェアで JTAG チェインが適切に接続されていることを検証するには、次のステップに従います。

1. Quartus II ソフトウェアで、Programmer を開きます。
2. Programmer で **Auto Detect** をクリックします。Quartus II ソフトウェアは、JTAG チェイン上で検出されたデバイス数をレポートします。これに失敗する場合は、JTAG チェインが分断されていないことを確認します。

MAX+Plus II ソフトウェアで JTAG チェインが適切に接続されていることを検証するには、次のステップに従います。

1. MAX+Plus II Programmer で、**Multi-Device JTAG Chain Setup** を選択します。
2. **Multi-Device JTAG Chain Setup** ダイアログ・ボックスで、**Detect JTAG Chain Info** ボタンをクリックします。MAX+Plus II ソフトウェアは、JTAG チェイン上で検出されたデバイス数をレポートします。

イン・システム・プログラミング中の V_{cc} レベルのチェック

オシロスコープを使用して、JTAG チェイン上の V_{CCINT} 信号をモニタし、トリガを該当するデバイス・ファミリ・データシートの推奨動作条件表に記載された最小 V_{cc} レベルに設定します。イン・システム・プログラミング中にトリガが発生する場合、デバイスは既存の電源から供給されている電流以上の電流量を必要としていると考えられます。既存の電源を容量の大きい電源に交換しても構いません。

パワーアップに関する問題

パワーアップ中に過剰な電圧または電流が I/O ピンに供給されると、JTAG チェイン内のデバイスの 1 つがラッチ・アップを起こすおそれがあります。デバイスを手で触れて、高温になっていないか確認します。高温のデバイスは、ラッチ・アップが発生して損傷している可能性があります。この場合は、すべての電圧源をチェックして、過剰な電圧または電流がデバイスに供給されていないか確認します。次に、影響を受けたデバイスを交換し、再度プログラムしてみます。

JTAG ピン上のランダム信号

通常の動作中、各デバイスの TAP コントローラは、テスト・ロジック・リセット状態になければなりません。デバイスを強制的にこの状態に戻すには、TMS 信号を High にプルアップし、TCK 信号を 6 回パルス生成します。このとき、デバイスが正常にパワーアップされた場合は、より高いプルダウン抵抗を TCK 信号に追加する必要があります。

ソフトウェアの問題

イン・システム・プログラミング中の障害は、Quartus II ソフトウェアまたは MAX+Plus II ソフトウェアに関係していることもあります。ソフトウェア関連の問題は、アルテラ・ウェブサイトのサポート・センタの Find Answers セクションに記載されています。データベースで、イン・システム・プログラミングの障害となるソフトウェア問題に関連する情報を検索してください。

エンベデッド・プロセッサを使用した ISP

この項では、Jam STAPL とエンベデッド・プロセッサを使用して、ISP 対応デバイスをプログラムするためのガイドラインを示します。

プロセッサおよびメモリ要件

Jam Byte-Code Player は 8 ビット以上のプロセッサをサポートし、ASCII Jam Player は 16 ビット以上のプロセッサをサポートしています。Jam Player は、予測可能な方式でメモリを使用します。この方式では、アップデートが **.jam** ファイルに限定されるため、イン・フィールド・アップグレードが簡単になります。Jam Player のメモリには、ROM とダイナミック・メモリ（RAM）の両方が使用されます。ROM は Jam Player バイナリおよび **.jam** ファイルの格納に使用され、ダイナミック・メモリは Jam Player が呼び出されたときに使用されます。

Jam Player が必要とする RAM および ROM の最大容量を推定する方法については、*「AN 425: Using the Command-Line Jam STAPL Solution for Device Programming」*を参照してください。

Jam Player の移植

アルテラ Jam Player（Byte-Code バージョンおよび ASCII バージョン）は、PC のパラレル・ポートで動作します。Jam Player は、**jamstub.c** または **jbistub.c** ファイル（それぞれ ASCII Jam Player または Jam Byte-Code Player に対応）を変更するだけでプロセッサに移植できます。その他のファイルはすべて同じです。Jam Player が不適切に移植された場合、`Unrecognized Device` エラーが発生します。このエラーの原因として最も一般的なものを以下に示します。

- Jam Player を移植した後、TDO の値が極性を反転して読み出されることがあります。この問題は、Jam Player のデフォルト I/O コードが PC のパラレル・ポートの使用を想定しているために発生する場合があります。
- TMS および TDI 信号は TCK の立ち上がりエッジでクロック制御されますが、出力は TCK の立ち下がりエッジまで変化しません。この状況では、TCK クロック半サイクルの遅延が生じます。立ち上がりエッジ上で TDO の遷移が予期される場合、データは 1 クロック分オフセットされているように見えます。

- アルテラは、レジスタを使用して出力の遷移を同期させることを推奨しています。さらに、プロセッサのデータ・ポートには、レジスタを使用して出力信号を同期させるものもあります。例えば、PC のパラレル・ポートの読み出しと書き込みは、レジスタを読み書きすることによって実現します。JTAG チェインに対して読み出しと書き込みを行うときには、これらのレジスタの使用を考慮する必要があります。これらのレジスタを正しく考慮しないと、値が予想値よりも進んだり遅れることになります。

.jam ファイルをテストすることで、Jam Player が正しく移植されたかどうかを確認できます。例 1 ～ 17 ページの例 4 に、可能な移植問題のデバッグに役立つサンプル **.jam** ファイルの一部を示します。

例 1. 移植の問題をデバッグするためのサンプル .jam ファイル

```
NOTE JAM_VERSION "1.1 ";
NOTE DESIGN "IDCODE.jam version 1.4 4/28/98";
'############################################################################
'#This Jam File compares the IDCODE read from a JTAG chain with the
'#expected IDCODE. There are 5 parameters that can be set when executing
'#this code.
'#
'#COMP_IDCODE_[device #]=1, for example -dCOMP_IDCODE_9400=1
'#compares the IDCODE with an EPM9400 IDCODE.
'#PRE_IR=[IR_LENGTH] is the length of the instruction registers you want
'#to bypass after the target device. The default is 0, so if your
'#JTAG length is 1, you don't need to enter a value.
'#POST_IR=[IR_LENGTH] is the length of the instruction registers you
'#want to bypass before the target device. The default is 0, so if
'#your JTAG length is 1, you don't need to enter a value.
'#PRE_DR=[DR_LENGTH] is the length of the data registers you want
'#to bypass after the target device. The default is 0, so if your
'#JTAG length is 1, you don't need to enter a value.
'#POST_DR=[DR_LENGTH] is the length of the data registers you want
'#to bypass before the target device. The default is 0, so if your
'#JTAG length is 1, you don't need to enter a value.
'#Example: This example reads the IDCODE out of the second device in the
'#chain below:
'#
'#TDI -> EPM7128S -> EPM7064S -> EPM7256S -> EPM7256S -> TDO
'#
'#In this example, the IDCODE is compared to the EPM7064S IDCODE. If the JTAG
'#chain is set up properly, the IDCODEs should match.
'# C:\> jam -dCOMP_IDCODE_7064S=1 -dPRE_IR=20 -dPOST_IR=10 -dPRE_DR=2
'#-dPOST_DR=1 -p378 IDCODE.jam
'#
'#
'# Example: This example reads the IDCODE of a single device JTAG chain
'# and compares it to an EPM9480 IDCODE:
'#
'# C:\> jam -dCOMP_IDCODE_9480=1 -p378 IDCODE.jam
'############################################################################
```

例 2. 移植の問題をデバッグするためのサンプル .jam ファイル

```
'########################## Initialization ##########################

BOOLEAN read_data[32];
BOOLEAN I_IDCODE[10] = BIN 1001101000;
BOOLEAN I_ONES[10] = BIN 1111111111;
BOOLEAN ONES_DATA[32]= HEX FFFFFFFF;
BOOLEAN ID_9320[32]= BIN 10111011000000000100110010010000;
BOOLEAN ID_9400[32] = BIN 10111011000000000000000010010000;
BOOLEAN ID_9480[32]= BIN 10111011000000000001001010010000;
BOOLEAN ID_9560[32] = BIN 10111011000000000110101010010000;
BOOLEAN ID_7032S[32] = BIN 10111011000001001100000011100000;
BOOLEAN ID_7064S[32] = BIN 10111011000000100110000011100000;
BOOLEAN ID_7128S[32] = BIN 10111011000000010100100011100000;
BOOLEAN ID_7128A[32] = BIN 10111011000000010100100011100000;
BOOLEAN ID_7160S[32] = BIN 10111011000000000110100011100000;
BOOLEAN ID_7192S[32] = BIN 10111011000001001001100011100000;
BOOLEAN ID_7256S[32] = BIN 10111011000001101010010011100000;
BOOLEAN ID_7256A[32] = BIN 10111011000001101010010011100000;


BOOLEAN COMP_9320_IDCODE = 0;
BOOLEAN COMP_9400_IDCODE  = 0;
BOOLEAN COMP_9480_IDCODE  = 0;
BOOLEAN COMP_9560_IDCODE = 0;
BOOLEAN COMP_7032S_IDCODE = 0;
BOOLEAN COMP_7064S_IDCODE = 0;
BOOLEAN COMP_7096S_IDCODE = 0;
BOOLEAN COMP_7128S_IDCODE = 0;
BOOLEAN COMP_7128A_IDCODE = 0;
BOOLEAN COMP_7160S_IDCODE = 0;
BOOLEAN COMP_7192S_IDCODE = 0;
BOOLEAN COMP_7256S_IDCODE = 0;
BOOLEAN COMP_7256A_IDCODE = 0;
BOOLEAN COMP_7032AE_IDCODE = 0;
BOOLEAN COMP_7064AE_IDCODE = 0;
BOOLEAN COMP_7128AE_IDCODE = 0;
BOOLEAN COMP_7256AE_IDCODE = 0;
BOOLEAN COMP_7512AE_IDCODE = 0;
INTEGER PRE_IR= 0;
INTEGER PRE_DR = 0;
INTEGER POST_IR = 0;
INTEGER POST_DR= 0;
BOOLEAN SET_ID_EXPECTED[32];
BOOLEAN COMPARE_FLAG1 = 0;
BOOLEAN COMPARE_FLAG2 = 0;
BOOLEAN COMPARE_FLAG  = 0;

' This information is what is expected to be shifted out of the instruction
' register

BOOLEAN expected_data[10] = BIN 0101010101;
BOOLEAN ir_data[10];
```

例 3. 移植の問題をデバッグするためのサンプル .jam ファイル

```
' These values default to 0, so if you have a single device JTAG chain, you do
' not have to set these values.

PREIR PRE_IR;
POSTIR POST_IR;
PREDR PRE_DR;
POSTDR POST_DR;

INTEGER i;

' ######################### Determine Action ########################

LET COMPARE_FLAG1= COMP_9320_IDCODE || COMP_9400_IDCODE || COMP_9480_IDCODE ||
COMP_9560_IDCODE || COMP_7032S_IDCODE || COMP_7064S_IDCODE ||
COMP_7096S_IDCODE || COMP_7032AE_IDCODE || COMP_7064AE_IDCODE ||
COMP_7128AE_IDCODE;

LET COMPARE_FLAG2 = COMP_7128S_IDCODE || COMP_7128A_IDCODE || COMP_7160S_IDCODE
|| COMP_7192S_IDCODE || COMP_7256S_IDCODE || COMP_7256A_IDCODE ||
COMP_7256AE_IDCODE || COMP_7512AE_IDCODE;

LET COMPARE_FLAG = COMPARE_FLAG1 || COMPARE_FLAG2;

IF COMPARE_FLAG != 1 THEN GOTO NO_OP;

FOR i=0 to 31;
IF COMP_9320_IDCODE == 1 THEN LET SET_ID_EXPECTED[i] = ID_9320[i];
IF COMP_9400_IDCODE == 1 THEN LET SET_ID_EXPECTED[i] = ID_9400[i];
IF COMP_9480_IDCODE == 1 THEN LET SET_ID_EXPECTED[i] = ID_9480[i];
IF COMP_9560_IDCODE== 1 THEN LET SET_ID_EXPECTED[i] = ID_9560[i];
IF COMP_7032S_IDCODE== 1 THEN LET SET_ID_EXPECTED[i] = ID_7032S[i];
IF COMP_7064S_IDCODE== 1 THEN LET SET_ID_EXPECTED[i] = ID_7064S[i];
FOR i=0 to 31;
IF COMP_9320_IDCODE == 1 THEN LET SET_ID_EXPECTED[i] = ID_9320[i];
IF COMP_9400_IDCODE == 1 THEN LET SET_ID_EXPECTED[i] = ID_9400[i];
IF COMP_9480_IDCODE == 1 THEN LET SET_ID_EXPECTED[i] = ID_9480[i];
IF COMP_9560_IDCODE== 1 THEN LET SET_ID_EXPECTED[i] = ID_9560[i];
IF COMP_7032S_IDCODE== 1 THEN LET SET_ID_EXPECTED[i] = ID_7032S[i];
IF COMP_7064S_IDCODE== 1 THEN LET SET_ID_EXPECTED[i] = ID_7064S[i];
IF COMP_7128S_IDCODE== 1 THEN LET SET_ID_EXPECTED[i] = ID_7128S[i];
IF COMP_7128A_IDCODE== 1 THEN LET SET_ID_EXPECTED[i] = ID_7128A[i];
IF COMP_7160S_IDCODE== 1 THEN LET SET_ID_EXPECTED[i] = ID_7160S[i];
IF COMP_7192S_IDCODE== 1 THEN LET SET_ID_EXPECTED[i] = ID_7192S[i];
IF COMP_7256S_IDCODE== 1 THEN LET SET_ID_EXPECTED[i] = ID_7256S[i];
IF COMP_7256A_IDCODE== 1 THEN LET SET_ID_EXPECTED[i] = ID_7256A[i];
IF COMP_7032AE_IDCODE== 1 THEN LET SET_ID_EXPECTED[i] = ID_7032AE[i];
IF COMP_7064AE_IDCODE== 1 THEN LET SET_ID_EXPECTED[i] = ID_7064AE[i];
IF COMP_7128AE_IDCODE== 1 THEN LET SET_ID_EXPECTED[i] = ID_7128AE[i];
IF COMP_7256AE_IDCODE== 1 THEN LET SET_ID_EXPECTED[i] = ID_7256AE[i];
IF COMP_7512AE_IDCODE== 1 THEN LET SET_ID_EXPECTED[i] = ID_7512AE[i];


NEXT I;
```

例 4. 移植の問題をデバッグするためのサンプル .jam ファイル

```
' ########################## Actual Loading ########################

IRSTOP IRPAUSE;
STATE RESET;
IRSCAN 10, I_IDCODE[0..9], CAPTURE ir_data[0..9];
STATE IDLE;

DRSCAN 32, ONES_DATA[0..31], CAPTURE read_data[0..31];

' ########################## Printing ########################

PRINT "EXPECTED IRSCAN : 1010101010";
PRINT "ACTUAL IRSCAN: ",ir_data[0], ir_data[1], ir_data[2], ir_data[3],
   ir_data[4], ir_data[5], ir_data[6], ir_data[7], ir_data[8], ir_data[9];

PRINT "";PRINT "EXPECTED IDCODE : ", SET_ID_EXPECTED[0], SET_ID_EXPECTED[1],
   SET_ID_EXPECTED[2], SET_ID_EXPECTED[3], SET_ID_EXPECTED[4],
   SET_ID_EXPECTED[5], SET_ID_EXPECTED[6], SET_ID_EXPECTED[7],
   SET_ID_EXPECTED[8], SET_ID_EXPECTED[9], SET_ID_EXPECTED[10],
   SET_ID_EXPECTED[11], SET_ID_EXPECTED[12], SET_ID_EXPECTED[13],
   SET_ID_EXPECTED[14], SET_ID_EXPECTED[15], SET_ID_EXPECTED[16],
   SET_ID_EXPECTED[17], SET_ID_EXPECTED[18], SET_ID_EXPECTED[19],
   SET_ID_EXPECTED[20], SET_ID_EXPECTED[21], SET_ID_EXPECTED[22],
   SET_ID_EXPECTED[23], SET_ID_EXPECTED[24], SET_ID_EXPECTED[25],
   SET_ID_EXPECTED[26], SET_ID_EXPECTED[27], SET_ID_EXPECTED[28],
   SET_ID_EXPECTED[29], SET_ID_EXPECTED[30], SET_ID_EXPECTED[31];

PRINT "ACTUAL IDCODE : ", READ_DATA[0], READ_DATA[1], READ_DATA[2],
   READ_DATA[3], READ_DATA[4], READ_DATA[5], READ_DATA[6], READ_DATA[7],
   READ_DATA[8], READ_DATA[9], READ_DATA[10], READ_DATA[11], READ_DATA[12],
   READ_DATA[13], READ_DATA[14], READ_DATA[15], READ_DATA[16], READ_DATA[17],
   READ_DATA[18], READ_DATA[19], READ_DATA[20], READ_DATA[21], READ_DATA[22],
   READ_DATA[23], READ_DATA[24], READ_DATA[25], READ_DATA[26], READ_DATA[27],
   READ_DATA[28], READ_DATA[29], READ_DATA[30], READ_DATA[31];

GOTO END;


' ########################## If no parameters are set ########################

NO_OP: PRINT "jam [-d<var=val>] [-p<port>] [-s<port>] IDCODE.jam";
PRINT "-d : initialize variable to specified value";
PRINT "-p : parallel port number or address <for ByteBlaster>";
PRINT "-s : serial port name <for BitBlaster>";
PRINT " ";
PRINT "Example: To compare IDCODE of the 4th device in a chain of 5 Altera "; PRINT
"devices with EPM7192S IDCODE";
PRINT " ";
PRINT "jam -dCOMP_7192S_IDCODE=1 -dPRE_IR=10 -dPOST_IR=30 -dPRE_DR=1";
PRINT "dPOST_DR=3 -p378 IDCODE.jam";
PRINT " ";

END:

EXIT 0;
```

イン・サーキット・テスタによる ISP

このセクションでは、イン・サーキット・テスタによる ISP 対応のデバイスのプログラミングに関連する問題について説明します。

Agilent's 3070 イン・サーキット・テスタによる MAX II および MAX V の ISP 作業について詳しくは、*「AN 425: Using the Command-Line Jam STAPL Solution for Device Programming」*を参照してください。

F デバイス vs 非 F デバイスの使用

MAX デバイスは、固定アルゴリズム（F）または分岐アルゴリズム（非 F）のいずれかを使用します。ほとんどのイン・サーキット・テスタのファイル・フォーマット（**.svf**、**.pcf**、DTS、および ASC など）は固定であり、1 つの分岐しない固定アルゴリズムのみをサポートできます。MAX+PLUS II ソフトウェアは F デバイスに対して SVF ファイルを生成します。SVF ファイル内のアルゴリズムが一定であるため、これらのファイルを使用して将来の F デバイスをプログラムすることができます。

アルテラは、イン・サーキット・テスタで非 F デバイスをプログラムすることを推奨していません。非 F デバイスはデバイスから読み出される 3 つの変数（プログラム・パルス時間、消去パルス時間、およびシリコン ID）に基づく分岐を必要とします。これらの 3 つの変数は、アルテラのすべての非 F デバイスにプログラムされています。F デバイスのみを使用することで、これらの変数の変更による問題を避けることができます。

ファイルあたりの最大ベクタ

通常、イン・サーキット・テスタ用のファイル・フォーマットは ISP のために非常に大きなベクタ・ファイルを必要とします。ファイルがテスタ内の使用可能なメモリを超える場合、このファイルをより小さいファイルに分割する必要があります。例えば、アルテラの **svf2pcf** ユーティリティは自動的に 1 つの **.svf** ファイルをいくつかの小さいファイルに分割します。さらに、このユーティリティにより、ユーザーはファイルあたりの最大ベクタ数を指定するか、あるいはデフォルト値を使用するかを選択できます。ファイルあたりのベクタ数が多すぎる場合、エラー・メッセージが表示されます。このエラーが発生する場合、ファイルあたりのベクタ数を低減してください。

プルアップ抵抗およびプルダウン抵抗

テスタでは、信号ラインの配線パターンによって、プルダウン抵抗またはプルアップ抵抗が必要となる場合があります。詳細については、イン・サーキット・テスタのメーカーにお問い合わせください。

改訂履歴

表 2 に、本資料の改訂履歴を示します。

表 2. 改訂履歴

日付	バージョン	変更内容
2010 年 12 月	4.0	■ 新しいテンプレートに変換 ■ MAX II and MAX V デバイスの情報を追加して更新