



6. Avalon インタフェース対応 UART コア

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。

NII51010-6.0.0

コアの概要

Avalon[®] インタフェース対応の Universal Asynchronous Receiver/Transmitter コアは、アルテラ FPGA 上のエンベデッド・システムと外部デバイスとの間で、キャラクタ・ストリームをシリアル通信する手段を実装しています。このコアは、RS-232 プロトコル・タイミングを実装し、可変ボー・レート、パリティ、ストップ・ビット、データ・ビット、およびオプションの RTS/CTS フロー制御信号を提供します。機能セットはコンフィギュレーション可能であり、設計者は特定のシステムに必要な機能のみ実装できます。

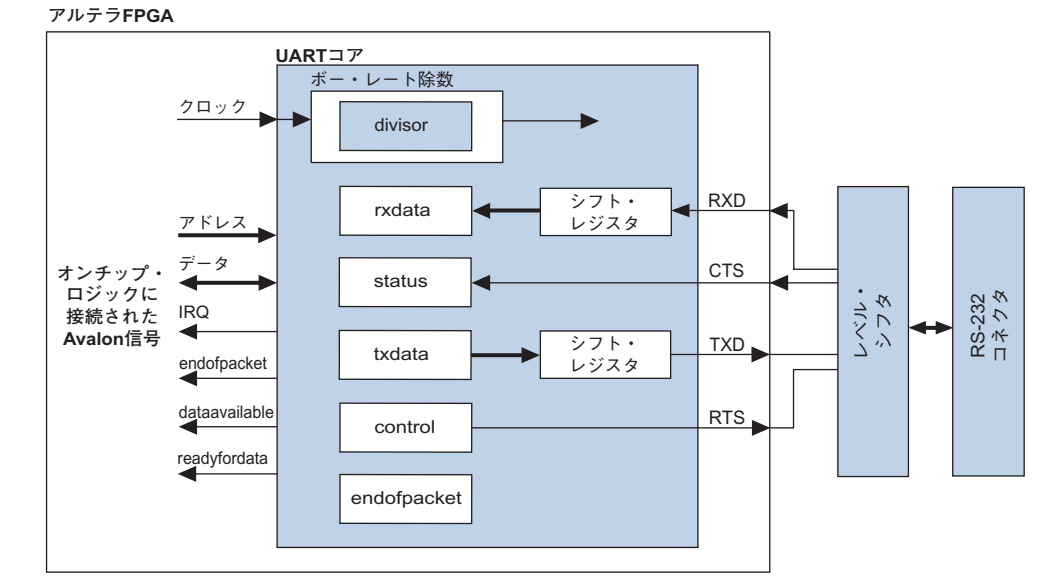
コアはレジスタにマップされたシンプルな Avalon スレーブ・インタフェースを備えており、このインタフェースによって、Avalon マスタ・ペリフェラル (Nios[®] II プロセッサなど) は、コントロール・レジスタおよびデータ・レジスタを読み書きするだけで、コアと通信できます。

UART コアは SOPC Builder に対応しており、SOPC Builder で生成されたどのシステムにも容易に統合できます。

機能の説明

図 6-1 に、UART コアのブロック図を示します。

図 6-1. 代表的なシステムにおける UART コアのブロック図



このコアには、ユーザが管理可能な以下の 2 つの部分があります。

- Avalon スレーブ・ポートを介してアクセスされるレジスタ・ファイル
- RS-232 信号、RXD、TXD、CTS、および RTS

Avalon スレーブ・インタフェースとレジスタ

UART コアは、内部レジスタ・ファイルへの Avalon スレーブ・インタフェースを提供します。UART コアへのユーザ・インタフェースは、control、status、rxdata、txdata、divisor、および endofpacket の 6 つの 16 ビット・レジスタで構成されています。Nios II プロセッサなどのマスタ・パリアフェラルは、レジスタにアクセスしてコアを管理し、さらにシリアル接続によってデータを転送します。

UART コアは、新しいデータを受信するかまたはコアが他のキャラクターを送信可能な状態になると割り込みを要求できるアクティブ High の割り込み要求 (IRQ) 出力を供給します。詳細については、6-23 ページの「割り込み動作」を参照してください。

Avalon スレーブ・ポートはフロー・コントロール機能付きの転送が可能です。UART コアは、Avalon フロー・コントロール機能付きの DMA (Direct Memory Access) ペリフェラルと連携して使用して、例えば UART コアとメモリの間で連続データ転送を自動化することができます。



詳しくは、「Avalon インタフェース対応タイマ・コア」の章を参照してください。Avalon インタフェースについて詳しくは、「Avalon Interface Specification」を参照してください。

RS-232 インタフェース

UART コアは、RS-232 非同期送信および受信ロジックを実装しています。UART コアは、TXD ポートと RXD ポートを介して、シリアル・データを送受信します。大部分のアルテラ FPGA ファミリの I/O バッファは、RS-232 の電圧レベルに適合していないため、RS-232 コネクタからの信号で直接ドライブすると損傷することがあります。RS-232 電圧信号規格に適合させるには、FPGA I/O ピンと外部 RS-232 コネクタとの間に、外部レベル・シフト・バッファ (MAX3237 など) が必要です。

UART コアはロジック 0 をマークに、ロジック 1 をスペースに使用します。FPGA 内部のインバータを使用し、任意の RS-232 信号の極性を必要に応じて反転させることができます。

トランスミッタ・ロジック

UART トランスミッタは、7 ビット、8 ビット、または 9 ビットの txdata ホールディング・レジスタと、対応する 7 ビット、8 ビット、または 9 ビットの送信シフト・レジスタで構成されています。Avalon マスタ・ペリフェラルは、Avalon スレーブ・ポートを介して txdata ホールディング・レジスタに書き込みます。送信シフト・レジスタは、シリアル送信シフト動作が実行されていないときに、txdata レジスタから自動的にロードされます。送信シフト・レジスタは、TXD 出力を直接供給します。データは、最下位ビット (LSB) から先に TXD にシフト・アウトされます。

これらの 2 つのレジスタはダブル・バッファリングを提供します。マスタ・ペリフェラルは、以前に書き込まれたキャラクタがシフト・アウトされる間に、txdata レジスタに新しい値を書き込むことができます。マスタ・ペリフェラルは、status レジスタのトランスミッタ・レディ (trdy) ビット、トランスミッタ・シフト・レジスタ・エンプティ (tmt) ビット、およびトランスミッタ・オーバラン・エラー (toe) ビットを読み出すことによって、トランスミッタのステータスをモニタできます。

トランスミッタ・ロジックは、RS-232 規格の要件に従って、シリアル TXD データ・ストリームに正しい数のスタート・ビット、ストップ・ビット、およびパリティ・ビットを自動的に挿入します。

レシーバ・ロジック

UART レシーバは、7 ビット、8 ビット、または 9 ビットのレシーバ・シフト・レジスタと、対応する 7 ビット、8 ビット、または 9 ビットの rxdata ホールディング・レジスタで構成されています。Avalon マスタ・ペリフェラルは、Avalon スレーブ・ポートを介して rxdata ホールディング・レジスタを読み出します。rxdata ホールディング・レジスタは、新しいキャラクタを完全に受信するたびに、レシーバ・シフト・レジスタから自動的にロードされます。

これらの 2 つのレジスタは、ダブル・バッファリングを提供します。rxdata レジスタは、後続のキャラクタがレシーバ・シフト・レジスタにシフト・インされている間、以前に受信したキャラクタを保持できません。

マスタ・ペリフェラルは、status レジスタのリード・レディ (rrdy) ビット、レシーバ・オーバラン・エラー (roe) ビット、ブレーク検出 (brk) ビット、パリティ・エラー (pe) ビット、およびフレーミング・エラー (fe) ビットを読み出すことによって、レシーバのステータスをモニタできます。レシーバ・ロジックは、RS-232 規格で要求されるシリアル RXD ストリーム内の正しい数のスタート・ビット、ストップ・ビット、およびパリティ・ビットを自動的に検出します。レシーバ・ロジックは、受信したデータで 4 つの例外条件 (フレーム・エラー、パリティ・エラー、受信オーバラン・エラー、およびブレーク) をチェックし、対応するステータス・レジスタ・ビット (fe, pe, roe, brk) を設定します。

ボー・レートの生成

UART コアの内部ボー・クロックは、Avalon クロック入力から供給されます。内部ボー・クロックはクロック分周器によって生成されます。除数値は、以下のいずれかで決まります。

- システム生成時に指定された定数
- divisor レジスタに格納された 16 ビット値

divisor レジスタは、オプションのハードウェア機能です。システム生成時にオフにすると除数値は固定され、ボー・レートは変更できません。

デバイスおよびツールのサポート

UART コアは、Stratix® や Cyclone™ デバイス・ファミリなど、すべてのアルテラ FPGA をターゲットにすることができます。

SOPC Builder でのコアのインスタンス化

UART をハードウェアでインスタンス化すると、各 UART ごとに少なくとも RXD 入力と TXD 出力の 2 つの I/O ポートが作成されます。オプションにより、ハードウェアにはフロー制御信号の CTS 入力と RTS 出力を含めることができます。

ハードウェア機能セットは、UART コアの SOPC Builder コンフィギュレーション・ウィザードで設定されます。以下のセクションでは、利用可能なオプションについて説明します。

コンフィギュレーション設定

このセクションでは、コンフィギュレーション設定について説明します。

ボー・レート・オプション

UART コアは RS-232 接続用の標準的なボー・レートを実装できます。ボー・レートは以下の 2 つの方法のいずれかでコンフィギュレーションできます。

- **Fixed rate** – ボー・レートはシステム生成時に固定され、Avalon スレーブ・ポートを介して変更することはできません。
- **Variable rate** – ボー・レートは、divisor レジスタに保持されるクロック除数に基づいて変更可能です。マスタ・ペリフェラルは、divisor レジスタに新しい値を書き込んでボー・レートを変更します。



ボー・レートは、Avalon インタフェースが供給するクロック周波数に基づいて計算されます。UART コア・ハードウェアを再生成しないで、ハードウェアでシステム・クロック周波数を変更すると、不正な信号が発生します。

Baud Rate (bps) 設定

Baud Rate 設定は、リセット後のデフォルト・ボー・レートを決定します。**Baud Rate** オプションは、標準的なプリセット値 (9600、57600、115200 bps など) を提供しますが、手動で任意のボー・レートを入力することもできます。

ボー・レート値は、希望のボー・レートを実装するのに適切なクロック除数値を計算するために使用されます。ボー・レートと除数値には、以下の関係があります。

$$\text{除数} = \text{int} ((\text{クロック周波数}) / (\text{ボー・レート}) + 0.5)$$

$$\text{ボー・レート} = (\text{クロック周波数}) / (\text{除数} + 1)$$

Baud Rate Can Be Changed By Software 設定

この設定をオンにすると、ハードウェアにはアドレス・オフセット 4 の位置に 16 ビットの divisor レジスタが搭載されます。divisor レジスタは書き込み可能なため、このレジスタに新しい値を書き込むとボー・レートを変更できます。

この設定をオフにすると、UART ハードウェアに divisor レジスタは搭載されません。UART ハードウェアは、一定の（変更不能な）ボー・除数を実装し、この値はシステム生成後には変更できません。この場合、アドレス・オフセット 4 に書き込んでも効果はなく、またアドレス・オフセット 4 から読み出すと結果は不定になります。

データ・ビット、ストップ・ビット、パリティ

UART コアのパリティ、データ・ビット、およびストップ・ビットはコンフィギュレーション可能です。これらの設定はシステム生成時に固定されるため、レジスタ・ファイルで変更することはできません。以下の設定が選択できます。

Data Bits 設定

表 6-1 を参照してください。

設定	許容値	説明
Data Bits	7, 8, 9	この設定は、txdata、rxdata、および endofpacket レジスタの幅を決定します。
Stop Bits	1, 2	この設定はコアが各キャラクタとともにストップ・ビットを 1 ビット送信するか、または 2 ビット送信するかを決定します。Stop Bits の設定に関係なく、コアは常に最初のストップ・ビット位置で受信トランザクションを終了し、以降のすべてのストップ・ビットを無視します。
Parity	None, Even, Odd	この設定は、UART がパリティ・チェック付きキャラクタを送信するかどうか、また UART が受信したキャラクタにパリティ・チェックが付加されていることを予期するかどうかを決定します。詳細については、次項を参照してください。

Parity 設定

Parity が **None** に設定されている場合、送信ロジックはパリティ・ビットのないデータを送信し、受信ロジックは受信するデータにパリティ・ビットが含まれていないものと想定します。パリティを **None** にすると、**status** レジスタの **pe** (パリティ・エラー) ビットは実装されず、読み出すと値は常に 0 になります。

Parity が **Odd** または **Even** に設定されていると、送信ロジックは必要なパリティ・ビットを計算し、送信 TXD ビットストリームにこのパリティ・ビットを挿入し、受信ロジックは受信 RXD ビットストリームでこのパリティ・ビットをチェックします。レシーバが不正なパリティを持つデータを検出した場合、**status** レジスタの **pe** が 1 に設定されます。パリティが **Even** のとき、パリティ・ビットはキャラクタに 1 のビットが偶数個ある場合には 1、そうでない場合は 0 になります。同様に、パリティが **Odd** のとき、パリティ・ビットはキャラクタに 1 のビットが奇数個ある場合には 1 になります。

フロー制御

以下のフロー制御オプションが選択できます。

Include CTS/RTS pins and control register bits

この設定をオンにすると、UART ハードウェアには以下が搭載されます。

- CTS_N (CTS の負論理) 入力ポート
- RTS_N (RTS の負論理) 出力ポート
- **status** レジスタの CTS ビット
- **status** レジスタの DCTS ビット
- **control** レジスタの RTS ビット
- **control** レジスタの IDCTS ビット

これらのハードウェア機能に基づいて、Avalon マスタ・ペリフェラルは CTS を検出し、RTS フロー制御信号を送信できます。CTS 入力ポートおよび RTS 出力ポートは、**status** レジスタおよび **control** レジスタのビットに直結されるため、コアのその他の部分に直接影響を与えることはありません。

Include CTS/RTS pins and control register bits 設定をオフにすると、上記のハードウェアはコアに搭載されません。コントロール / ステータス・ビットの CTS、DCTS、IDCTS、および RTS は実装されず、読み出すと値は常に 0 になります。

フロー・コントロール機能付きの Avalon 転送 (DMA)

UART コアの Avalon インタフェースは、オプションによってフロー・コントロール機能付きの Avalon 転送を実装します。これによって、Avalon マスタ・ペリフェラルは、UART コアが別のキャラクタを受け入れる準備が整ったときにのみデータを書き込み、コアに利用可能なデータが存在するときにのみデータを読み出すことが可能になります。また、UART コアにはオプションで EOP レジスタを搭載することもできます。

Include end-of-packet register

この設定をオンにすると、UART コアに以下が搭載されます。

- アドレス・オフセット 5 における 7 ビット、8 ビット、または 9 ビットの endofpacket レジスタ。データ幅は **Data Bits** 設定によって決定されます。
- status レジスタの eop ビット
- control レジスタの ieop ビット
- システム内の他のマスタ・ペリフェラルとの間でフロー・コントロール機能付きのデータ転送をサポートする、Avalon インタフェースの endofpacket 信号。

EOP (End-of-packet) 検出によって、UART コアはフロー・コントロール機能付きの Avalon マスタとのデータ・トランザクションを終了できます。EOP 検出を DMA コントローラと組み合わせて使用すると、例えば、受信 RXD ストリームで指定されたキャラクタが検出されるまで、受信したキャラクタを自動的にメモリに書き込む UART を実現することができます。終端 (EOP) キャラクタの値は、endofpacket レジスタで決定されます。

EOP レジスタがディセーブルされると、UART コアには上記のリソースは搭載されません。endofpacket レジスタに書き込んでも効果はなく、読み出した値は不定です。

シミュレーション設定

UART コアのロジックが生成されると、同時にシミュレーション・モデルも構築されます。シミュレーション・モデルを利用すれば、UART コアを使用するシステムのシミュレーションが簡略化および高速化されます。シミュレーション設定を変更しても、ハードウェアの UART コアの動作に影響はありません。設定は機能シミュレーションにのみ作用します。



以下の設定を使用して Nios II システムをシミュレートする方法の例は、「AN 351: Simulating Nios II Embedded Processor Designs」を参照してください。

RXD 入力キャラクタ・ストリームのシミュレーション

ユーザはシミュレートされたシステム・リセットで、RXD ポートへの入力がシミュレートされるキャラクタ・ストリームを入力できます。UART コアのコンフィギュレーション・ウィザードは任意のキャラクタ文字列を受け取り、後でこの文字列が UART シミュレーション・モデルに組み込まれます。リセット後、コアが新しいデータを受け入れることができるようになると、この文字列は RXD ポートにキャラクタ単位で入力されます。

インタラクティブ・ウィンドウの準備

システム生成時に、UART コア・ジェネレータはシミュレーション中に UART モデルとの連携を容易にする ModelSim マクロを作成できます。以下のオプションが用意されています。

Create ModelSim Alias to open streaming output window

TXD ポートからのすべての出力を表示するウィンドウを開くための ModelSim マクロが作成されます。

Create ModelSim Alias to open interactive stimulus window

RXD ポートに対するステイミュラスを受け入れるウィンドウを開くための ModelSim マクロが作成されます。このウィンドウはウィンドウ内で入力されたキャラクタを RXD ポートに送信します。

トランスミッタ・ボー・レートのシミュレーション

多くの場合、RS-232 の送信レートは、システム内の他のプロセスよりも低速であり、実際のボー・レートで機能モデルをシミュレートする利点はほとんどありません。例えば、115,200 bps では 1 キャラクタ転送するのに、標準で数千クロック・サイクルを要します。UART シミュレーション・モデルは、一定クロック除数 2 で動作することができます。これにより、シミュレートされる UART は、システム・クロック速度の 1/2、すなわち約 20 クロック・サイクルごとに 1 キャラクタを転送します。シミュレートされるトランスミッタのボー・レートには、以下のオプションのいずれかを選択できます。

- **accelerated (use divisor = 2)** — TXD は、シミュレーションで 2 クロック・サイクルごとに 1 ビットを送出します。

ハードウェア・シミュレーションの考慮点

- **actual (use true baud divisor)** – TXD は、divisor レジスタで決定される実際のボー・レートで送信します。

シミュレーション機能は、ModelSim シミュレータを使用するときに、Nios、Nios II または Excalibur™ プロセッサ・システムのシミュレーションを容易にするために作成されました。各プロセッサのドキュメントには、これらの機能の推奨される使用方法が記載されています。その他の用途にも使用できる可能性はありますが、その場合はユーザがカスタム・シミュレーション・プロセスを作成するために一層の作業が必要です。

シミュレーション・モデルは UART コアのトップ・レベル HDL ファイルで実装され、合成可能な HDL およびシミュレーション HDL は同じファイルで実装されます。シミュレーション機能は、HDL コードの特定のセクションを合成ツールに対してのみ開示する translate on および translate off 合成オプションを使用して実装されます。

アルテラの推奨シミュレーション手順を使用する場合は、シミュレーションオプションを編集しないでください。シミュレーションオプションを変更してカスタム・シミュレーション・フローを作成する場合は、SOPC Builder がシステム生成時に既存のファイルを上書きすることに留意する必要があります。変更内容が上書きされないよう注意してください。



Nios II プロセッサ・システムでの UART コアのシミュレーションの詳細については、「AN 351: Simulating Nios II Processor Designs」を参照してください。Nios エンベデッド・プロセッサ・システムでの UART コアのシミュレーションの詳細については、「AN 189: Simulating Nios II Embedded Processor Designs」を参照してください。

ソフトウェア・プログラミング・モデル

以下のセクションでは、ハードウェアにアクセスするためのレジスタ・マップやソフトウェア宣言など、UART コアのソフトウェア・プログラミング・モデルについて説明します。アルテラは、Nios II プロセッサ・ユーザ向けに printf() や getchar() などの ANSI C 標準ライブラリ関数を使用して、UART コアへのアクセスを可能にする HAL (Hardware Abstraction Layer) システム・ライブラリ・ドライバを提供しています。

HAL システム・ライブラリ・サポート

アルテラが提供するドライバは、Nios II システム用の HAL システム・ライブラリに統合される HAL キャラクタ・モード・デバイス・ドライバを実装します。HAL ユーザは、UART レジスタにせずに、使い慣れた HAL API および ANSI C 標準ライブラリを介して UART にアクセスすることができます。HAL ユーザによる UART のハードウェア依存部分の制御を可能にする、ioctl() 要求が定義されています。



ユーザ・プログラムで HAL デバイス・ドライバを使用して UART ハードウェアにアクセスする場合、デバイス・レジスタに直接アクセスすると、ドライバの正常な動作が妨害されます。

Nios II プロセッサ・ユーザ向けに、HAL システム・ライブラリ API は、UART コアの機能への完全なアクセスを提供します。Nios II プログラムは UART コアをキャラクタ・モード・デバイスとして扱い、ANSI C 標準ライブラリ関数を使用してデータを送受信します。

CTS/RTS コントロール信号が SOPC Builder でイネーブルされているときには、ドライバはそれらの信号をサポートします。6-12 ページの「[ドライバ・オプション: 高速および小型サイズの実装](#)」を参照してください。

以下のコードは、`printf()` を使用して `stdout` にメッセージを出力する最もシンプルな使用方法を示します。この例では、SOPC Builder システムに UART コアが搭載され、HAL システム・ライブラリは、このデバイスを `stdout` に使用するようにコンフィギュレーションされています。

例: UART コアへの `stdout` としてのキャラクタの出力

```
#include <stdio.h>
int main ()
{
    printf("Hello world.\n");
    return 0;
}
```

以下のコードは、C 標準ライブラリを使用して、UART デバイスからキャラクタを読み出す方法と、UART デバイスへメッセージを送信する方法を示します。この例では、SOPC Builder システムには、`uart1` という名前の UART コアが搭載されていますが、これは必ずしも `stdout` デバイスとしてコンフィギュレーションされるとは限りません。この場合、プログラムはこのデバイスを HAL ファイル・システムの他のノードと同じように扱います。

例：キャラクターの送信と受信

```
/* キャラクタ「t」と「v」を認識するシンプルなプログラム */
#include <stdio.h>
#include <string.h>
int main ()
{
    char* msg = "Detected the character 't'.\n";
    FILE* fp;
    char prompt = 0;

    fp = fopen ("/dev/uart1", "r+"); // 読み出しと書き込み用にファイルを開く。
    if (fp)
    {
        while (prompt != 'v')
        { // 「v」を受信するまでループする。
            prompt = getc(fp); // UARTからキャラクターを取得する。
            if (prompt == 't')
            { // キャラクターが「t」なら、メッセージを出力する。
                fwrite (msg, strlen (msg), 1, fp);
            }
        }

        fprintf(fp, "Closing the UART file.\n");
        fclose (fp);
    }

    return 0;
}
```

HAL システム・ライブラリの詳細については、「Nios II ソフトウェア開発ハンドブック」で解説しています。

ドライバ・オプション：高速および小型サイズの実装

多様なタイプのシステムの要件に対応するために、UART ドライバには高速バージョンと小型バージョンの2種類が用意されています。デフォルトでは、高速型動作が使用されます。高速ドライバと小型ドライバのどちらも、C 標準ライブラリ関数と HAL API を完全にサポートしています。

高速ドライバは割り込み駆動で実行されるため、デバイスがデータの送信または受信の準備が整っていないときにも、プロセッサは他のタスクを実行できます。UART のデータ・レートはプロセッサと比較して低速なため、一時的に他のタスクを実行する可能性があるシステムに対しては、高速ドライバによって大きな性能上の利点が得られます。

小型ドライバは、各キャラクタを送受信する前に UART ハードウェアを待機するポーリング形式を実装したものです。スモール・フットプリント・ドライバを有効にするには、以下の 2 つの方法があります。

- HAL システム・ライブラリ・プロジェクトのスモール・フットプリント設定をオンにします。このオプションは、システム内のすべてのデバイス用のデバイス・ドライバにも作用します。
- プリプロセッサ・オプション `-DALTEA_AVALON_UART_SMALL` を指定します。このオプションを使用すると、他のデバイスのドライバに影響を与えずに、小型の UART ドライバをポーリング形式で実装できます。



HAL プロパティおよびプリプロセッサ・オプションの設定方法に関する詳細については、Nios II IDE のヘルプ・システムを参照してください。

CTS/RTS フロー制御信号がハードウェアでイネーブルされた場合、高速ドライバは自動的にそれらの信号を使用します。小型ドライバはこれらの信号を常に無視します。

ioctl() 操作

UART ドライバは `ioctl()` 関数をサポートしているため、HAL ベースのプログラムはデバイス固有の操作を要求できます。表 6-2 に、UART ドライバがサポートする操作要求を定義します。

要求	意味
TIOCEXCL	排他的アクセス用にデバイスをロックします。このファイル記述子が閉じられるか、または TIOCNXCL <code>ioctl</code> 要求を使用してロックが解放されるまでは、このデバイスに対してさらに <code>open()</code> を呼び出しても失敗します。この要求を成功させるには、このデバイスに対する既存のファイル記述子が存在しないことが必要です。 <code>ioctl</code> の「arg」パラメータは無視されます。
TIOCNXCL	前の排他的アクセスのロックを解放します。詳細については、上記のコメントを参照してください。 <code>ioctl</code> の「arg」パラメータは無視されます。

表 6-3 に示すとおり、高速ドライバに対してのみオプションで追加の操作要求が実行できます。ユーザのプログラムでこれらの操作を有効にするには、プリプロセッサ・オプション `-DALTEA_AVALON_UART_USE_IOCTL` を設定する必要があります。

要求	意味
TIOCMGET	入力の <code>termios</code> 構造体 (1) の内容に情報を入力することによって、デバイスの現在の設定を返します。この構造体へのポインタは、 <code>ioctl</code> の「opt」パラメータの値として供給されます。
TIOCMSET	入力の <code>termios</code> 構造体 (1) に格納された値に従って、デバイスの設定を指定します。この構造体へのポインタは、 <code>ioctl</code> の「arg」パラメータの値として供給されます。

表 6-3 の注：

- (1) `termios` 構造体は、Newlib C 標準ライブラリによって定義されています。この定義はファイル `<Nios II キットのパス>/components/altera_hal/HAL/inc/sys/termios.h` にあります。



`ioctl()` 関数の詳細については、「Nios II ソフトウェア開発ハンドブック」を参照してください。

制限

UART コアの HAL ドライバは `endoofpacket` レジスタをサポートしていません。詳細については、6-15 ページの「レジスタ・マップ」を参照してください。

ソフトウェア・ファイル

UART コアには、以下のソフトウェア・ファイルが付属しています。これらのファイルは、ハードウェアへの低水準インターフェースを定義し、HAL ドライバを提供します。アプリケーション開発者がこれらのファイルを変更してはなりません。

- **altera_avalon_uart_regs.h** –このファイルは、コアのレジスタ・マップを定義し、低水準ハードウェアにアクセスするための記号定数を提供します。このファイル内の記号は、デバイス・ドライバ関数によってのみ使用されます。
- **altera_avalon_uart.h**、**altera_avalon_uart.c** –これらのファイルは、HAL システム・ライブラリ用の UART コア・デバイス・ドライバを実装します。

レガシー SDK ルーチン

UART コアは、第 1 世代の Nios プロセッサ用のレガシー SDK ルーチンでもサポートされています。これらのルーチンの詳細については、第 1 世代の Nios プロセッサに付属する UART のドキュメントを参照してください。レガシー SDK に基づいてプログラムを HAL システム・ライブラリ API にアップグレードする方法の詳細については、「AN 350: Upgrading Nios Processor Systems to the Nios II Processor」を参照してください。

レジスタ・マップ

HAL API または第 1 世代の Nios プロセッサ用のレガシー SDK を利用するプログラマが、レジスタを使用して UART コアに直接アクセスすることはありません。一般に、レジスタ・マップはコアのデバイス・ドライバを記述するプログラマにとってのみ有用です。



アルテラが提供する HAL デバイス・ドライバは、デバイス・レジスタに直接アクセスします。デバイス・ドライバを記述する場合に、HAL ドライバが同じデバイスに対してアクティブであれば、記述したドライバは衝突して、動作できません。

表 6-4 に、UART コアのレジスタ・マップを示します。デバイス・ドライバは、メモリ・マップド・レジスタを介してコアの制御と通信を行います。

オフセット	レジスタ名	R/W	説明 / レジスタ・ビット													
			15...13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	rxdata	RO	(1)					(2)	(2)	受信データ						
1	txdata	WO	(1)					(2)	(2)	送信データ						
2	status (3)	RW	(1)	eop	cts	dcts	(1)	e	rrdy	trdy	tmt	toe	roe	brk	fe	pe
3	control	RW	(1)	ieop	rts	idcts	trbk	ie	irrdy	itrdy	itmt	itoe	iroe	ibrk	ife	ipe
4	divisor (4)	RW	ボー・レートの除数													
5	endofpacket (4)	RW	(1)					(2)	(2)	EOP 値						

表 6-4 の注：

- (1) これらのビットは予約済みです。読み出すと不定値が返されます。ゼロを書き込みます。
- (2) これらのビットが存在するかどうかは、**Data Width** ハードウェア・オプションによって決まります。存在しない場合、これらのビットを読み出すと値はゼロになり、書き込んでも効果はありません。
- (3) **status** レジスタにゼロを書き込むと、**dcts**、**e**、**toe**、**roe**、**brk**、**fe**、および **pe** ビットがクリアされます。
- (4) このレジスタが存在するかどうかは、ハードウェア・コンフィギュレーション・オプションによって決まります。存在しない場合、読み出すと不定値が返され、書き込んでも効果はありません。

一部のレジスタとビットはオプションです。これらのレジスタとビットは、システム生成時にイネーブルされた場合にのみハードウェアに存在します。オプションのレジスタとビットを以下に示します。

rxdata レジスタ

rxdata レジスタは、RXD 入力を介して受信したデータを保持します。新しいキャラクタが RXD 入力を介して完全に受信されると、そのキャラクタは **rxdata** レジスタに転送され、**status** レジスタの **rrdy** ビットが 1 に設定されます。**rxdata** レジスタが読み出されると、**status** レジスタの **rrdy** ビットは 0 に設定されます。**rrdy** ビットが設定されている（つまり、前のキャラクタが取り出されていない）間に、あるキャラクタが **rxdata** レジスタに転送されると、レシーバ・オーバーラン・エラーが発生し、**status** レジスタの **roe** ビットは 1 に設定されます。新しいキャラクタは、前のキャラクタが読み出されたかどうかに関係なく、常に **rxdata** レジスタに転送されます。**rxdata** レジスタにデータを書き込んでも効果はありません。

txdata レジスタ

Avalon マスタ・ペリフェラルは、送信するキャラクタを txdata レジスタに書き込みます。トランスミッタが新しいキャラクタを送信可能な状態になる (status レジスタの TRDY ビットで示される) まで、キャラクタを txdata に書き込んではありません。txdata レジスタにキャラクタが書き込まれると、TRDY ビットは 0 に設定されます。このキャラクタが txdata レジスタからトランスミッタ・シフト・レジスタに転送されると、TRDY ビットは 1 に設定されます。TRDY が 0 のときにキャラクタが txdata レジスタに書き込まれると、結果は不定になります。txdata レジスタを読み出すと、不定値が返されます。

例えば、トランスミッタ・ロジックがアイドル状態で、Avalon マスタ・ペリフェラルが最初のキャラクタを txdata レジスタに書き込むと仮定します。TRDY ビットは 0 に設定され、そのキャラクタがトランスミッタ・シフト・レジスタに転送されると、1 に設定されます。次に、マスタは 2 番目のキャラクタを txdata レジスタに書き込むことができ、TRDY ビットは再び 0 に設定されます。ただし、このときシフト・レジスタは最初のキャラクタを TXD 出力にシフト・アウトしているため、まだビジー状態です。最初のキャラクタが完全にシフト・アウトされ、2 番目のキャラクタが自動的にトランスミッタ・シフト・レジスタに転送されるまで、TRDY ビットは 1 に設定されません。

status レジスタ

status レジスタは、UART コア内部の特定の状態を示す個別ビットで構成されています。各ステータス・ビットは、control レジスタの対応する割り込みイネーブル・ビットに関連付けられています。status レジスタはいつでも読み出すことができます。読み出しを行っても、どのビットの値も変化しません。status レジスタにゼロを書き込むと、DCTS、E、TOE、ROE、BRK、FE、および PE ビットがクリアされます。

status レジスタ・ビットを表 6-5 に示します。

表 6-5. status レジスタ・ビット (1 / 3)

ビット	ビット名	読み出し (R) / 書き込み (W) / クリア (C)	説明
0 (1)	PE	RC	<p>パリティ・エラー。パリティ・エラーは、受信したパリティ・ビットに予期しない（不正な）ロジック・レベルが含まれるときに発生します。コアが不正なパリティ・ビットを持つキャラクタを受信すると、PE ビットは 1 に設定されます。PE ビットは、status レジスタへの書き込みによって明示的にクリアされるまで、1 に設定されたままです。PE ビットが設定されているときに、rxdata レジスタから読み出すと不定値になります。</p> <p>Parity ハードウェア・オプションがオンにされていない場合、パリティ・チェックは実行されず、PE ビットを読み出すと値は常に 0 になります。6-6 ページの「データ・ビット、ストップ・ビット、パリティ」を参照してください。</p>
1	FE	RC	<p>フレーミング・エラー。フレーミング・エラーは、レシーバが正しいストップ・ビットの検出に失敗すると発生します。コアが不正なストップ・ビットを持つキャラクタを受信すると、FE ビットは 1 に設定されます。FE ビットは、status レジスタへの書き込みによって明示的にクリアされるまで、1 に設定されたままです。FE ビットが設定されているときに、rxdata レジスタから読み出すと不定値になります。</p>
2	BRK	RC	<p>ブレイク検出。レシーバ・ロジックは、RXD ピンが、全キャラクタ時間（データ・ビットにスタート・ビット、ストップ・ビット、パリティ・ビットを加えたもの）より長く継続的に Low（ロジック 0）に保持されるとブレイクを検出します。ブレイクが検出されると、BRK ビットは 1 に設定されます。BRK ビットは、status レジスタへの書き込みによって明示的にクリアされるまで、1 に設定されたままです。</p>
3	ROE	RC	<p>受信オーバーラン・エラー。前のキャラクタが読み出される前（つまり、RRDY ビットが 1 の間）に、新しく受信したキャラクタが rxdata ホールディング・レジスタに転送されると、受信オーバーラン・エラーが発生します。この場合、ROE ビットが 1 に設定され、rxdata の前の内容は、新しいキャラクタで上書きされます。ROE ビットは、status レジスタへの書き込みによって明示的にクリアされるまで、1 に設定されたままです。</p>
4	TOE	RC	<p>送信オーバーラン・エラー。前のキャラクタがシフト・レジスタに転送される前（つまり、TRDY ビットが 0 の間）に、新しいキャラクタが txdata ホールディング・レジスタに書き込まれると、送信オーバーラン・エラーが発生します。この場合、TOE ビットは 1 に設定されます。TOE ビットは、status レジスタへの書き込みによって明示的にクリアされるまで、1 に設定されたままです。</p>

表 6-5. status レジスタ・ビット (2 / 3)

ビット	ビット名	読み出し (R) / 書き込み (W) / クリア (C)	説明
5	TMT	R	送信エンプティ。TMT ビットはトランスミッタ・シフト・レジスタの現在の状態を示します。シフト・レジスタが、キャラクタを TXD ピンからシフト・アウトしている間、TMT は 0 に設定されます。シフト・レジスタがアイドル状態（つまり、キャラクタの送信中でない）のとき、TMT ビットは 1 です。Avalon マスタ・ペリフェラルは、TMT ビットをチェックして、送信が完了した（そして、シリアル・リンクの他端で受信された）かどうかを判断することができます。
6	TRDY	R	送信レディ。TRDY ビットは、txdata ホールディング・レジスタの現在の状態を示します。txdata レジスタが空で、新しいキャラクタを処理できる場合、trdy は 1 です。txdata レジスタが満杯の場合、TRDY は 0 です。Avalon マスタ・ペリフェラルは、TRDY が 1 になるのを待って、新しいデータを txdata に書き込む必要があります。
7	RRDY	R	受信キャラクタ・レディ。RRDY ビットは、rxdata ホールディング・レジスタの現在の状態を示します。rxdata レジスタが空のときは、まだ読み出し可能な状態ではなく、rrdy は 0 です。新しく受信した値が rxdata レジスタに転送されると、RRDY は 1 に設定されます。rxdata レジスタを読み出すと、RRDY ビットは 0 にクリアされます。Avalon マスタ・ペリフェラルは、RRDY が 1 になるのを待って、rxdata を読み出す必要があります。
8	E	RC	例外。E ビットは例外条件が発生したことを示します。E ビットは、TOE、ROE、BRK、FE、および PE ビットの論理 OR をしたものです。e ビットと、それに対応する control レジスタの割り込みイネーブル・ビット (IE) は、すべてのエラー条件に対する IRQ をイネーブル/ディセーブルする便利な方法を提供します。 E ビットは、status レジスタに書き込み操作を実行すると、0 に設定されます。
10 (1)	DCTS	RC	CTS (Clear To Send) 信号の変化。CTS_N 入力ポートでロジック・レベルの遷移が検出されると (Avalon クロックと同期してサンプリングされる)、DCTS ビットは 1 に設定されます。このビットは CTS_N の立ち下がり遷移と立ち上がり遷移で設定されます。DCTS ビットは、status レジスタへの書き込みによって明示的にクリアされるまで、1 に設定されたままです。 Flow Control ハードウェア・オプションがオンにされていない場合、DCTS ビットを読み出すと値は常に 0 になります。6-7 ページの「フロー制御」を参照してください。

表 6-5. status レジスタ・ビット (3 / 3)			
ビット	ビット名	読み出し (R) / 書き込み (W) / クリア (C)	説明
11 (1)	CTS	R	<p>CTS (Clear To Send) 信号。CTS ビットは、CTS_N 入力の瞬時的な状態 (Avalon クロックと同期してサンプリングされる) を反映します。CTS_N 入力は負論理なので、ロジック・レベル 0 が CTS_N 入力に適用されると、CTS ビットは 1 になります。</p> <p>CTS_N 入力は送信プロセスや受信プロセスには作用しません。CTS_N 入力の明示的な効果は、CTS ビットおよび DCTS ビットの状態、そして control レジスタの idcts ビットがイネーブルされているときに生成できる IRQ のみです。</p> <p>Flow Control ハードウェア・オプションがオンにされていない場合、CTS ビットを読み出すと値は常に 0 になります。6-7 ページの「フロー制御」を参照してください。</p>
12 (1)	EOP	R	<p>EOP の検出。EOP ビットは、以下のいずれかのイベントで 1 に設定されます。</p> <ul style="list-style-type: none"> ● EOP キャラクタは txdata に書き込まれたとき。 ● EOP キャラクタは rxdata から読み出されたとき。 <p>EOP キャラクタは、endofpacket レジスタの内容によって決定されます。EOP ビットは、status レジスタへの書き込みによって明示的にクリアされるまで、1 に設定されたままです。</p> <p>Include End-of-Packet Register ハードウェア・オプションがオンにされていない場合、EOP ビットを読み出すと値は常に 0 になります。6-8 ページの「フロー・コントロール機能付きの Avalon 転送 (DMA)」を参照してください。</p>

表 6-5 の注：

(1) このビットはオプションであり、ハードウェアに存在しない場合もあります。

control レジスタ

control レジスタは個別ビットで構成されており、各ビットが UART コアの動作を制御します。control レジスタの値はいつでも読み出すことができます。

control レジスタの各ビットは、status レジスタの対応するビットの IRQ をイネーブルします。status ビットとそれに対応する割り込みイネーブル・ビットが両方とも 1 のとき、コアは IRQ を生成します。例えば、pe ビットは status レジスタのビット 0、ipe ビットは control レジスタのビット 0 です。割り込み要求は、pe と ipe の両方が 1 のときに生成されます。

control レジスタのビットを表 6-6 に示します。

ビット	ビット名	読み出し (R) / 書き込み (W)	説明
0	IPE	RW	パリティ・エラーに対する割り込みのイネーブル。
1	IFE	RW	フレーミング・エラーに対する割り込みのイネーブル。
2	IBRK	RW	ブレーク検出に対する割り込みのイネーブル。
3	IROE	RW	レシーバ・オーバーラン・エラーに対する割り込みのイネーブル。
4	ITOE	RW	トランスミッタ・オーバーラン・エラーに対する割り込みのイネーブル。
5	ITMT	RW	トランスミッタ・シフト・レジスタ・エンプティに対する割り込みのイネーブル。
6	ITRDY	RW	送信レディに対する割り込みのイネーブル。
7	IRRDY	RW	読み出しレディに対する割り込みのイネーブル。
8	IE	RW	例外に対する割り込みのイネーブル。
9	TRBK	RW	送信ブレーク。TRBK ビットにより、Avalon マスタ・ペリフェラルは、TXD 出力にブレーク・キャラクタを送信することができます。TRBK ビットが 1 に設定されると、TXD 信号は強制的に 0 になります。TRBK ビットは、トランスミッタ・ロジックが TXD 出力でドライブするロジック・レベルよりも優先されます。TRBK ビットは処理中の送信を中断します。Avalon マスタ・ペリフェラルは、所定のブレーク期間が経過した後、TRBK ビットを再び 0 に設定する必要があります。
10	IDCTS	RW	CTS 信号の変化に対する割り込みのイネーブル。

表 6-6. control レジスタのビット (2 / 2)

ビット	ビット名	読み出し (R) / 書き込み (W)	説明
11 (1)	RTS	RW	(RTS) 信号の送信要求。RTS ビットは RTS_N 出力に直接供給されます。Avalon マスタ・ペリフェラルは、いつでも RTS ビットに書き込むことができます。RTS ビットの値は RTS_N 出力にのみ影響し、トランスミッタ・ロジックやレシーバ・ロジックには作用しません。RTS_N 出力は真論理なので、RTS ビットが 1 のとき、RTS_N 出力には Low ロジック・レベル (0) がドライブされます。 Flow Control ハードウェア・オプションがオンにされていない場合、RTS ビットを読み出すと値は常に 0 になり、書き込んでも効果はありません。6-7 ページの「 フロー制御 」を参照してください。
12	IEOP	RW	EOP 状態に対する割り込みのイネーブル。

表 6-6 の注：

(1) このビットはオプションであり、ハードウェアに存在しない場合もあります。

divisor レジスタ (オプション)

divisor レジスタの値は、ボー・レート・クロックの生成に使用されます。有効なボー・レートは、以下の公式から求められます。

$$\text{ボー・レート} = (\text{クロック周波数}) / (\text{除数} + 1)$$

divisor レジスタは、オプションのハードウェア機能です。**Baud Rate Can Be Changed By Software** ハードウェア・オプションがオンにされていない場合、divisor レジスタは存在しません。この場合、divisor に書き込んでも効果はなく、divisor を読み出すと不定値が返されます。詳しくは、6-5 ページの「[ボー・レート・オプション](#)」を参照してください。

endofpacket レジスタ (オプション)

endofpacket レジスタの値は、可変長 DMA トランザクションの EOP キャラクタを決定します。リセット後には、デフォルト値はゼロ、つまり ASCII のヌル・キャラクタ (\0) です。eop ビットの詳細については、6-18 ページの表 6-5 を参照してください。

endofpacket レジスタは、オプションのハードウェア機能です。**Include end-of-packet register** ハードウェア・オプションがオンにされていない場合、endofpacket レジスタは存在しません。この場合、endofpacket に書き込んでも効果はなく、読み出すと不定値が返されます。

割り込み動作

UART コアは1つの IRQ 信号を Avalon インタフェースに出力し、このインタフェースは Nios II プロセッサなど、システム内の任意のマスター・ペリフェラルに接続できます。マスター・ペリフェラルは status レジスタを読み出して、割り込みの原因を特定する必要があります。

すべての割り込み条件は、status レジスタの関連ビット、および control レジスタの割り込みイネーブル・ビットを持っています。いずれかの割り込み条件が発生すると、関連付けられた status ビットが1に設定され、明示的に認識応答が返されるまでこの設定のままです。対応する割り込みイネーブル・ビットが1の間に、status ビットのいずれかが設定されると、IRQ 出力がアサートされます。マスター・ペリフェラルは、status レジスタをクリアして IRQ に認識応答することができます。

リセット時に、すべての割り込みイネーブル・ビットは0に設定されるので、マスター・ペリフェラルが割り込みイネーブル・ビットの1つまたは複数を1に設定するまで、コアは IRQ をアサートできません。

可能なすべての割り込み条件とそれらに関連付けられた status および control (割り込みイネーブル) ビットを、6-18 ページの表 6-5 および 6-21 ページの表 6-6 に示します。各割り込み条件の詳細は、status ビットの説明に記載します。

