nRF51 シリーズ　リファレンスマニュアル　（翻訳）

Version 3.0

（略）

# 目次

1. 改版履歴
2. 本ドキュメントについて
3. システム概要
4. CPU
5. メモリー
6. 不揮発メモリー・コントローラー (NVMC)
7. ファクトリー・インフォメーション　設定レジスター (FUCR)
8. ユーザー・インフォメーション設定レジスター (UICR)
9. メモリー保護ユニット (MPU)
10. 周辺装置インターフェイス
11. デバッガー・インターフェイス (DIF)
12. パワー・マネジメント (POWER)
13. クロック・マネジメント (CLOCK)
14. 汎用IO (GPIO)
15. GPIO タスク・イベント (GPIOTE)
16. プログラマブル周辺装置相互接続 (PPI)
17. 2.4 GHz 無線 (RADIP)
18. タイマー／カウンター (TIMER)
19. リアルタイムカウンター (RTC)
20. ウォッチドッグタイマー (TDT)
21. 乱数発生器 (RNG)
22. 温度センサー (TEMP)
23. AES EBCモード暗号化 (ECB)
24. AES CCMモード暗号化 (CCM)
25. 促進的アドレス解決 (AAR)
26. シリアル・ペリフェラル・インターフェイス (SPI) マスター
27. シリアル・ペリフェラル・インターフェイス (SPI) スレーブ
28. I2C互換 ２線インターフェイス (TWI)
29. 汎用非同期送受信機 (UART)
30. クアッド・レーター・デコーダー (QDEC)
31. A/Dコンバーター (ADC)
32. 低出力コンパレーター (LPCOMP)
33. ソフトウェア割り込み (SWI)

# 改版履歴

# 本ドキュメントについて

# システム概要

## 要約

（略）

## ブロック図

　本ブロック図はシステム全体を図示しています。白い矢印は他のシグナルと物理ピンによって共有されるシグナルを示します。

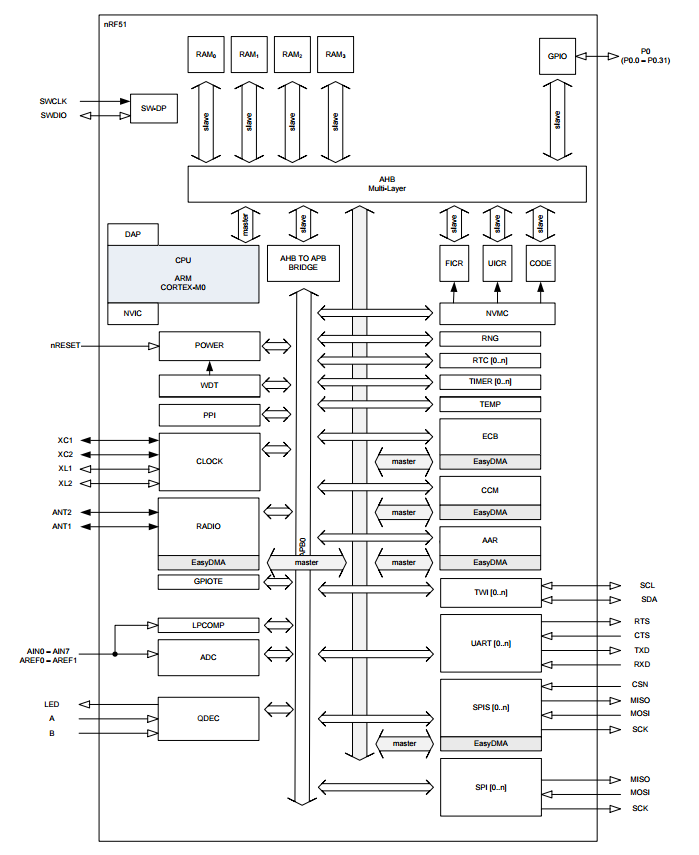


図１：ブロック図

## システムブロック

　本章では、nrF51シリーズを構成する主なブロックについて解説します。

### ARM Cortex-M0

（略）

### 2.4 GHz 無線

（略）

### パワー・マネジメント

　nRF51 シリーズの電源管理システムは、シンプルなON・OFFモードにより装置全体を管理する、非常に柔軟かつ独立したシステムです。システムOFFモードでは、全ての電源が落とされますが、RAM領域は保つことができます。リセットか全てのGPIOからの起動によりデバイスの状態をシステムONにすることができます。システムONモードでは、全ての機能ブロックにアクセス可能であり、それぞれの機能ブロックをIDLEモードにしておき、要求のあった場合にのみRUNモードに入ります。

### PPIシステム

　プログラマブル・ペリフェラル・インターコネクト（PPI）により、異なるペリフェラルがタスクとイベントを使用し、CPUを使用することなく、自動的に相互作用することが可能になります。PPIはあるペリフェラルで発生したイベントの結果を別のペリフェラルのタスクの自動トリガーとする仕組みを提供します。タスクはPPIチャンネルを経由し、イベントと連携されます。

### デバッガーサポート

（略）

# CPU

　低電力32bit CPU、ARM Cortex-M0 が全ての nRF51 シリーズのデバイスに組み込まれています。 ARM Cortex-M0 には 16bit の命令、32bit に拡張された(Thumb-2 technology) があり、高密度のコードかつ小メモリーフットプリントのデバイスです。1サイクルでの32bit乗算器、3ステージパイプライン、ネスト型ベクター割り込みコントローラー (NVIC) が、ARM Cortex-M0 CPUを高効率かつシンプルなプログラム実行を可能にしています。

　nRF51のデーターアライメントはリトルエンディアンです。

　ARM Cortex-M プロセッサーシリーズに実装されている、ARM Cortex マイクロコントローラー・ソフトウェア・インターフェース・スタンダード (CMSIS)ハードウェア抽象レイヤー は、M0 CPUにおいても利用可能です。コードはARM Cortex-M3ベースのデバイスにおいて前方互換性を持ちます。

　ARM Cotex-M0 CPU に関しての、より詳しい情報は、ARM Cotex M0を参照してください。

# メモリー

# 不揮発メモリー・コントローラー (NVMC)

## 機能説明

　不揮発メモリー・コントローラー (NVMC) は不揮発メモリー (NVM) への書き込みと消去に使用されます。

　書き込み操作が事項される前に、CONFIG.WENにより、NVMは書き込みが有効にされる必要があります。同様に消去が実行される前に、CONFIG.EENにより、NVMは消去を有効にされる必要があります。ユーザーは書き込みと消去が同時に有効にされることが無いことを確認しなくてはなりません。さもなくば、予期せぬ動作が発生する可能性があります。

### NVM への書き込み

　書き込みが有効な場合、NVMはCODEもしくはUICRに対し、アラインされたアドレスでワード単位の書き込みが行われます。NVMC はNVMが消去された領域、つまり1にセットされているビットにのみ書き込みが可能です。

　NVMにワードを書き込むのに必要な時間は製品仕様書のtWRITEに言及があります。CPUはNVMCがNVMに書き込んでいる間、停止します。

　ワード単位にアラインされた書き込みのみが許可されています。バイトや半ワードにアラインされた書き込みは、ハードフォルトを引き起こします。

### ユーザー・インフォメーション・設定レジスターへの書き込み

　UICR レジスターは通常の不揮発メモリーです。UICRが書き込まれたのち、新しいUICR設定は、リセット後にのみ有効になります。

### 全消去

　消去が有効な場合、すべての CODE と UICR は ERASEALL レジスターを使用した一度の操作で削除されることができます。ERASEALL はファクトリー・インフォメーション・設定レジスター (FICR) を消去することはありません。

### code region 1 内のページ消去

　消去が有効な場合、 NVM は ERASEPAGE レジスターもしくは ERASEPCR1 レジスターを使用することにより、ページごとに消去されることができます。消去後には NVM ページのすべてのページ内のビットは 1 にセットされます。ページを消去するために必要な時間は製品仕様書のtPAGEERASEに記述されています。NVMC が消去操作を行っている間 CPU は停止します。より詳細な情報は UICR の章を参照してください。

### code region 0 内のページ消去

　ERASEPCR0 は code region 0 のページを消去するために使用されます。 ERASEPCR0 レジスターは core region 0 を実行中のプログラムによりのみアクセスすることができます。

　code region 0 を実行しているプログラムが不揮発性記憶を有効にするため、このプログラムは全てのコードページの消去及び再書き込みが可能です。これはcode region 0 内におけるこの目的のため指定します。 ERASEPCR0 はこの目的のために使用されます。ERASEPCR 0 レジスターはその使用にMPUによって強制された、code region 0 から実行されるコードのみ書き換えが可能と言う、制限を持っています。code region 0 を実行しているプログラムはERASEPCR1 を使用することにより、code region 1 内のページを消去することができます。

　ページを消去するために必要な時間は製品仕様書のtPAGEERASEに記述されています。

## レジスター概要

（略）

## レジスター詳細

（略）

# ファクトリー・インフォメーション設定レジスター (FUCR)

## 機能説明

ファクトリー・インフォメーション設定レジスターは、製造元により事前にプログラムされ、ユーザーからは削除不可能です。このレジスターはチップ特有の情報や設定を含んでいます。

## パラメーターの上書き

（略）

## レジスター概要

（略）

## レジスター詳細

（略）

# ユーザー・インフォメーション設定レジスター (UICR)

## 機能説明

　ユーザー・インフォメーション設定レジスターはユーザー特有の設定のための不揮発性メモリーのレジスターです。

ｄ：　コード読み取りは全てのコードエリアに対し保護されていますが、一部のコードエリアはUICRとして、設定され有効化されます。UICRはERASEALLによってのみ、削除可能です。

　コードエリアは code region 0 (CR0) と code region 1 (CR1) の２つの領域に分割することができます。code region 0 は0x00000000 から開始され、CLENR0 レジスターとして示されるコード領域まで拡張されます。CLENR0を超えた領域core region 1として定義されます。CLENR0が定義されていない場合、つまり値が0xFFFFFFFFFの場合、全コード領域はcode region 1(CR1)として、定義されます。

　コードは code region 1 より実行され、code region 0に書き込まれることはありません。加えて、core region 0 が読み出し保護の場合、code region 0 は core region 1 により実行されているコードや、SWDインターフェイスからは読み込まれることはありません。詳細は RBPCONF の PR0 を確認してください。

　UICRを通して設定される、主読み出し保護機能は、 code region 0 と code region 1 の全コードを保護します。

　NVMC の PASSASGE コマンドは、code ragion 1 にのみ作用します。コード領域とUICRへの消去とプログラムの実施方法はNVMCの章を参照してください。

## レジスター概要

（略）

## レジスター詳細

（略）

# メモリー保護ユニット (MPU)

# 周辺装置インターフェイス

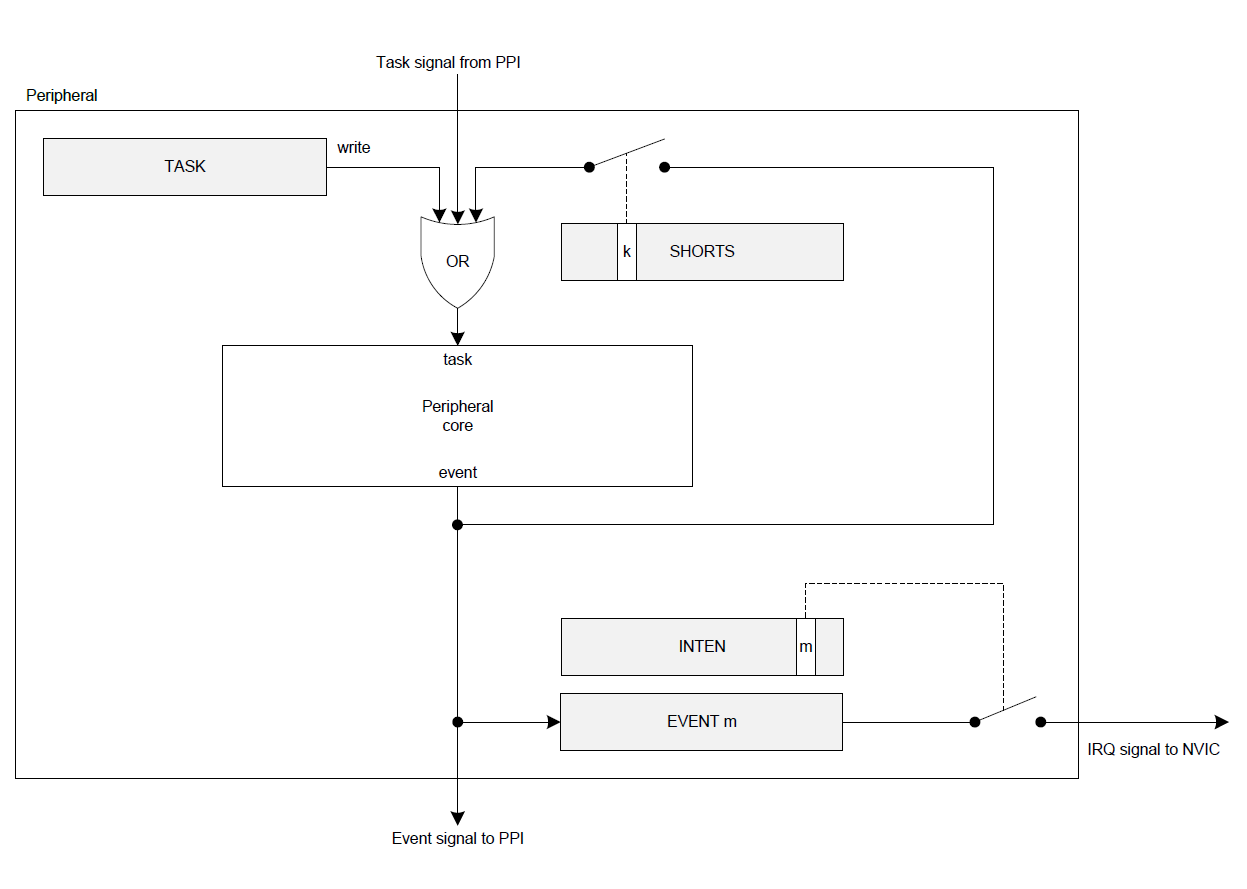


図６：タスク、イベント、ショートカット、割り込み

## 機能説明

全てのペリフェラルは標準ARM Cortex アドバンス・ペリフェラル・バス（APB）もしくはAMBAハイパフォーマンス・バス（AHB）のレジスターを通じアクセス、タスク、イベント、割り込み、を登録することができます。

### ペリフェラルID

全てのペリフェラルには0x1000バイトの固定されたブロックが割り当てられており、それは、1024×32ビットのレジスターと等価です。このパターンはAPBバス、AHBバスに配置されているすべてのペリフェラルに対して適用されます。どのペリフェラルが使用可能で、アドレス・マップのどこにそれらが配置されているか等の、より詳細な情報は、ページ17のインスタンス化を参照してください。

　APBバス上のペリフェラルについてはIDとベースアドレスの間に直接的な関連があります。ベースアドレス0x40000000のペリフェラルであれば、ID:0に割り当てられ、0x40001000がベースアドレスのペリフェラルはIDが1に割り当てられます。ベースアドレスが0x4001F000のペリフェラルはID:31 が割り当てられます。

ペリフェラルは以下の、１つ、もしくは複数の制約により、同じIDを共有することがあります。

* ペリフェラルは複数のレジスターもしくは共通リソースを共有してはいけません。ただし、各ペリフェラルの使用可能なレジスターの最大数は固定IDをもつペリフェラルと比較し、少ないです。
* ペリフェラルはレジスターや他の共通のリソースを共有します。
* 同時に使用することのできるペリフェラルは一つのみです。
* これらのペリフェラルは一連の中で選択的であり、これらのうちの一つのみが与えられたチップ内でインスタンス化されます。
* 一つのペリフェラルから他のペリフェラルにスイッチする際、特定の手順を踏む必要があります。（初めのペリフェラルを無効化し、２つ目のペリフェラルを有効化します）

### ビットの設定／クリア

複数の単一ビット、ビットフィールドを持つレジスターは、『セット』と『クリア』パターンが実装されているでしょう。これらのパターンはファームウェアがレジスターの各々のビットを、メインレジスター上で読み込み-編集-書き込みすることなしに、セットやクリアすることを可能にしています。

このパターンは命令内において、メインレジスターに続きSET、CLRレジスターの順で、レジスター・マップ内の3つの連続したアドレスとして実装されています。

SETレジスターはメインレジスター内の各々のビットをセットするために使用され、対してCLRレジスターはメインレジスター内の各々のビットをクリアするために使用されます。SETもしくはCLRレジスターのビットに ’1’ が書き込まれると、メインレジスターの該当するビットがセットもしくはクリアされます。SETもしくはCLRレジスターのビットに ’0’ が書き込まれても、何も起こりません。SETもしくはCLRレジスターの値を読み込んだ場合は、メインレジスターの値が戻されます。

注：どのような場合でも、メインレジスターは不可視であり、それゆえに直接アクセスすることは出来ません。

### タスク

タスクは、例えば特定の動作を開始する等、ペリフェラルがトリガーをアクションするために使用されます。ペリフェラルはペリフェラル・タスク・レジスター・グループ内に、それぞれ別個のレジスターを持った複数のタスクを実装することができます。

　タスクは、ファームウェアがタスク・レジスターに対して ‘1’ を書き込む、もしくは自他のペリフェラルが対応したタスク信号をトグルした場合にトリガーされます。ページ●●の、図６：タスク、イベント、ショートカット、割り込みを参照してください。

### イベント

イベントは、ペリフェラルやCPUに対して、例えばペリフェラルの状態が変化した等、イベントが発生したことを知らせるために使用されます。ペリフェラルは複数のイベントを発生させることができ、それぞれのイベントはペリフェラル内のイベント・レジスター・グループに独立したレジスターを持ちます。

　イベントはペリフェラル自身がイベント・シグナル信号をトグルした場合発生させられ、その反映のため、イベント・レジスターは即座に更新されます。ページ●●の、図６：タスク、イベント、ショートカット、割り込みを参照してください。イベント・レジスターはファームウェアが ‘0’ をイベント・レジスターに書き込んだ場合にのみクリアされます。

　イベントは、イベント・レジスターが ‘1’ に設定されている場合においても、ペリフェラルにより発生させられます。

### ショートカット

　ショートカットは同じペリフェラル内において、イベントとタスクを直接接続する目的で使用されます。ショートカットが有効な場合、関連付けられたイベントが発生した場合、関連付けられたタスクが自動的にトリガーされます。

　ショートカットの使用は、PPIを使用し、外部のペリフェラルと接続を実施することと同じです。しかし、ショートカットを使用した伝播遅延は、一般的にPPIを使用した伝搬遅延より短くなります。

　ショートカットは定義済みであり、それはファームウェアによって接続を設定できないことを意味しています。それぞれのペリフェラルには、ショートカット1つに付き1ビット、最大32個のショートカットを各々、ショートカット・レジスターを通じ、個々に有効／無効の切り替えを行うことができます。

### 割り込み

割り込みとは、イベントによって発生させられる例外や、CPUのプログラム・フローの割り込みにより、生成されるイベントを指します。APBバス上のすべてのペリフェラルが割り込みに対応しています。一つのペリフェラルは一つの割り込みを占めており、割り込み番号は、ペリフェラルIDを追っています。例えば、ID=4のペリフェラルは、ネスト・ベクター割り込みコントローラー内の割り込み番号4に接続されています。

　INTENとINTENSETとINTENCLRレジスターを使用することにより、ペリフェラルの割り込みが発生するための、それぞれのイベントを設定することができます。同時に割り込みを発生する、複数のイベントを有効化することができます。接続された割り込みソースを解決するため、ファームウェアはペリフェラル・レジスター・マップ内の、イベントグループ内で見つけられるイベント・レジスターをクエリすることができます。

いくつかのペリフェラルは、INTENSETとINTENCLRのみ実装されており、これらのペリフェラルにおいては、INTENレジスターが利用できません。詳細はそれぞれの章を確認してください。しかしながら、すべての場合において、INTENSETもしくはINTENCLRレジスターの値を読み戻すと、INTENと同じ情報を戻します。

ペリフェラル内のそれぞれのイベント実装はINTEN、INTENSET、INTENCLRレジスターの特定のビット位置に関連付けられています。正確なビット位置はイベントのアドレスによって分けられています。アドレス0x100のイベントはINTENレジスターの0ビット目に関連付けられており、アドレス0x104のイベントは、1ビット目に関連付けられています。その他も同様です。アドレス0x17CのイベントはINTENレジスターの31ビット目として同定されます。このパターンが有効な範囲はペリフェラルのイベントの最大値の32までです。

タスクとイベントとショートカット及び割り込みの関係は、ページ●●の、図６：タスク、イベント、ショートカット、割り込みを参照してください。

# デバッガー・インターフェイス (DIF)

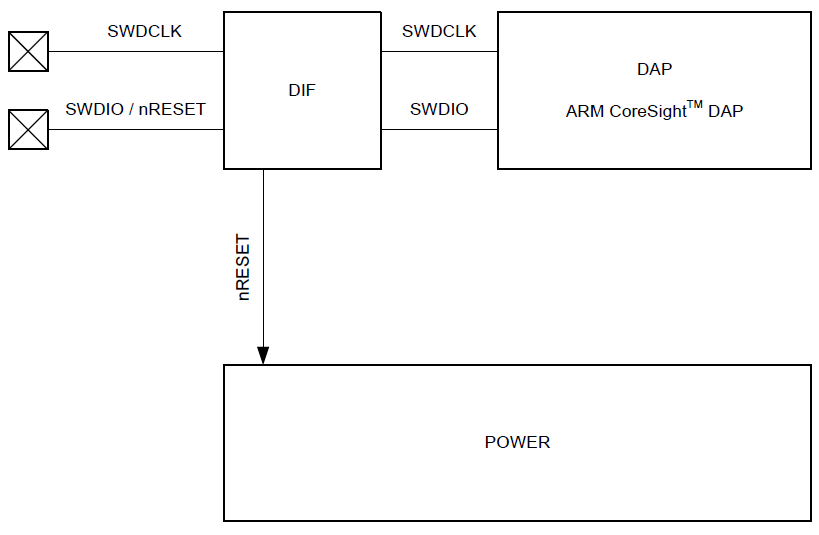


図７：デバッガー・インターフェイス

## 機能説明

nRF51デバイスはARMのシリアル・ワイヤーデバッグ（SWD）インターフェイスをサポートしています。このインターフェイスは二つのSWDCLKとSWDIOのワイヤーを持ちます。SWDIOとnRESETは同じ物理ピンを共用しています。デバッガー・インターフェイス（DIF）モジュールはSWDのやりとりと、リセット機能の両方の扱いを管理しています。SWDCLKピンは内部でプルダウン・レジスター持っており、SWDIO/nRESETピンは内部でプルアップ・レジスターを持っています。

### ノーマル・モード

DIFモジュールはパワーオン・リセット後には通常モードになります。このモードでは、SWDIO/nRESETピンは通常のアクティブローのリセットピンとして働きます。

ノーマル・モードでありつづけることを保証すするために、SWDCLKラインはローであり続ける（常に’0’であること）必要があります。さもなくば、DIFは未知の状態に陥る可能性があり、望まぬ動作及び電力消費に導かれる可能性があります。

### デバッグ・インターフェイス・モード

デバッグ・インターフェイス・モードはSWDIOが1の状態でSWCLKに１クロックサイクルが与えられることにより、起動されます。DAPのパワードメイン起動の遅れのため、SWDIOが1の状態で、SWDCLKに最低125 kHz／最低150クロックサイクルの周期が与えられると、DAPは最低50クロックサイクルをキャプチャすることが保障されています。

もしデバイスがシステム・オフ・モードの場合、デバッグ・インターフェイス・モードへの遷移は、起動を促します。（システム・オフ・モードに関してはページ●●の、[パワーマネジメント（POWER）](#_パワーマネジメント_(POWER))を参照してください）

デバッグ・インターフェイス・モードでは、SWDIO/nRESETピンはSWDIOとして使用されます。ピンリセット機能はデバイスがデバッグ・インターフェイス・モードの間は」無効になります。

デバッグ・インターフェイス・モードでは、デバイスのデバッグを容易にするため、システム・オフはエミュレートされます。パワー・ナンバーは通常のシステム・オフより、エミュレートされたシステム・オフの方が高くなります。詳細は●●ページの、エミュレーティテッド・システム・オフ・モードを参照してください。

### ノーマル・モードの再開

どのような場合にも、SWDインターフェイスに『ハードリセット』を行う事により、ノーマル・モードの再開が可能です。

* + - 1. デバッグ・インターフェイス・モードに遷移してください。
      2. POWERペリフェラルのRESETレジスターを使用し、リセットを行ってください。
      3. 少なくとも100 μ秒間 SWDCLK と SWDIO/nRESET ラインをローにしてください。

ハードリセットは、パワーオン・リセット、もしくは、ブラウンアウト・リセットによっても可能です。

# パワー・マネジメント (POWER)

# クロック・マネジメント (CLOCK)

# 汎用IO (GPIO)

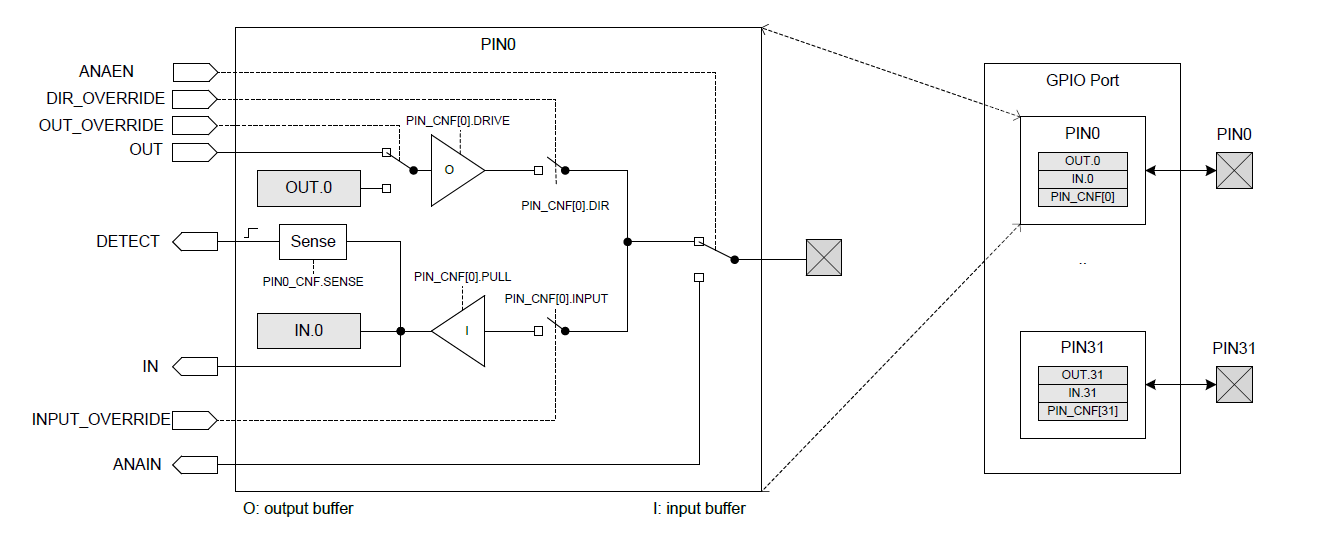


図１５：GPIOポート、GPIOピン詳細

ページ●●の図１５：GPIOポート、GPIOピン詳細は、32の独立したピンを含んでいるGPIOポートを描いています。ピン0は参照としてより詳細に描かれています。図の左側のすべてのシグナルはシステムの他のペリフェラルに使用されます。つまり、これらは直接CPUからは利用可能ではありません。

## 機能説明

　GPIOポートペリフェラルは、PIN0からPIN31までの32までのピンを実装しています。各々ピンはPIN\_CNF[n]レジスター（n=0～31）に別個に設定されています。以下のパラメーターがこのレジスターを通じ設定されます。

* 入出力方向
* ドライブ強度
* プルアップ・レジスター／プルダウン・レジスターの有効化
* ピン検出
* 入力バッファー切断
* アナログ入力（選択したピン）

PIN\_CNFレジスターは保持レジスターです。保持レジスターに関する詳細は[POWER](#_パワー・マネジメント_(POWER))章を参照してください。

ピン検出機能を通じ、ピンの入力がHIGHレベル、もしくはLOWレベルを検出するかを各々のピンに対して設定可能です。いずれかの設定されたピンにおいて、正しいレベルが検出された場合、検出機能はDITECTシグナルをHIGHにします。各々のピンは独立したDITECTシグナルを持っており、全GPIOポートのすべてのピンのDITECTシグナルのデフォルト動作は、他のペリフェラルにより使いやすくするため、システムを通じ、共通DITECTシグナルに関連付けられています。ページ●●の図１５：GPIOポート、GPIOピン詳細を参照してください。この機能はONモードでもOFFモードでも機能します。

DITECTシグナルがどのように使用されるかの詳細に関しては、以下のペリフェラルを参照してください。

* POWER：DITECTシグナルをシステムOFFからの遷移に使用します。
* GPIOTE：DITECTシグナルをPORTイベントの生成に使用します。

GPIOピンの入力バッファーは、ピンが入力として使用されない場合、パワーセーブの有効化からGPIOを切り離します。ページ●●の図１５：GPIOポート、GPIOピン詳細を参照してください。入力はINレジスターと、検出機能がピンにアクセスし、正しい入力値を得るために接続されていなくてはなりません。

システム内の他のペリフェラルはGPIOピンに自分自身を接続し、出力値と設定の上書き、もしくは、アナログかデジタル入力を読み取ることが可能です。ページ●●の図１５：GPIOポート、GPIOピン詳細を参照してください。

選択されたPINはアナログ信号入力機能を提供します。ページ●●の図１５：GPIOポート、GPIOピン詳細のANAINを参照してください。アナログ信号入力が有効なピンはデバイスに応じて変化します。詳細に関しては、所有デバイスの製品仕様書を確認してください。

## レジスター概要

（略）

## レジスター詳細

（略）

# GPIO タスク・イベント (GPIOTE)

## 機能説明

GPIOタスク・イベント（GPIOTE）モジュールはタスクとイベントを使用した、GPIOピンへのアクセス機能を提供します。ピン1つ毎に、GPIOTEチャンネルが提供されています。

　タスクはGPIOTEチャンネルごとに、ピンに対し以下の書き込み操作を実行する目的で使用されます。

* セット
* クリア
* トグル

　イベントはGPIOTEチャンネルごとに、以下の入力状態のいずれかを発生させることができます。

* 立ち上がりエッジ
* 立ち下りエッジ
* 状態変化

### ピン・イベント、ピン・タスク

GPIOTEモジュールは、OUT[n]タスクとIN[n]イベントを持ち、個々のGPIOピンに対して作用するように設定可能な、複数のタスクとイベンを持ちます。タスクは個々のピンに対して書き込むために使用され、イベントは個々のピンへの入力が変化することにより発生させられます。

タスクとイベントはCONFIG[n]レジスターを使用することにより設定されます。すべてのOUT[n]タスクとIN[n]イベントのペアは、それに対して関連付けられている1つのCONFIG[n]レジスターを持ちます。

ピンに対して、OUT[n]タスクもしくはIN[n]イベントが設定された場合、ピンンはGPIOTEモジュールによってのみ書き込み可能になります。通常のGPIOピンによるピンへの書き込み試行は何も効果を及ぼさなくなります。

OUT[n] タスクもしくは IN[n] イベントがピンnを制御するよう設定されている限り、ピンの出力値はGPIOTEモジュールによってのみ更新されます。GPIOによって指定されたピンの出力値は、ピンがGPIOTEにより制御されている限り、無視されます。GPIOTEがピンから切断された場合、関連付けられていたピンは、GPIOモジュールによって指定された出力と設定された値を受け取るでしょう。CONFIG[n] レジスターのMODEフィールドを参照してください。

ピンをタスクとして実行するように、GPIOTEチャンネルが設定されている場合、このピンの初期値はCONFIG[n]のOUTINITフィールドにより設定されます。

### ポートイベント

PORTとはGPIO DIRECTシグナルを使用している複数の入力ピンにより発生させることのできるイベントです。このイベントはDITECTシグナルの立ち上がりにより発生させられます。DITECTシグナルの詳細については、[GPIO](#_汎用IO_(GPIO))チャプターを参照してください

この機能は、ペリフェラルがIDLEの時にも常に有効です、そのため、クロックが停止中、もしくは、他の電源集中基盤はこの機能を有効にしておくことを要求しています。本機能はすなわち、CPUが全てのペリフェラルとCPUのアイドルから、システムONの中WFIもしくはWFEのようなスリープから起動するために使用され

すなわち、本機能はシステムONモードにおいて、WFIもしくはWFEのようなスリープ中、全ペリフェラルとCPUがアイドル中から、CPUが起動するために使用されます。それはシステムONモードの消費電力を最小にします。

### タスク・ピン・イベント・ピン設定

各々のGPIOチャンネルはCONFIG.PINSELフィールドにより、一つの物理GPIOピンに関連付けられています。もし、イベント・モードがCONFIG.MODEによって選択されている場合、CONFIG.PSELにより指定されたピンは、GPIOの設定を上書きし、入力として設定されます。同様にCONFIG.MODEにより、タスク・モードが設定されている場合、CONFIG.PSELにより指定されたピンはGPIOの設定を上書きし、出力として設定されます。もし、CONFIG.MODEにより無効が設定されている場合、CONFIG.PSELにより指定されたピンはGPIOのPIN\_CNFレジスターの設定を使用します。

１つのGPIOTEチャンネルは一つの物理ピンに関連付けられます。さもなくば、予期せぬ挙動を引き起こします。

## レジスター概要

（略）

## レジスター詳細

（略）

# プログラマブル・ペリフェラル相互接続 (PPI)

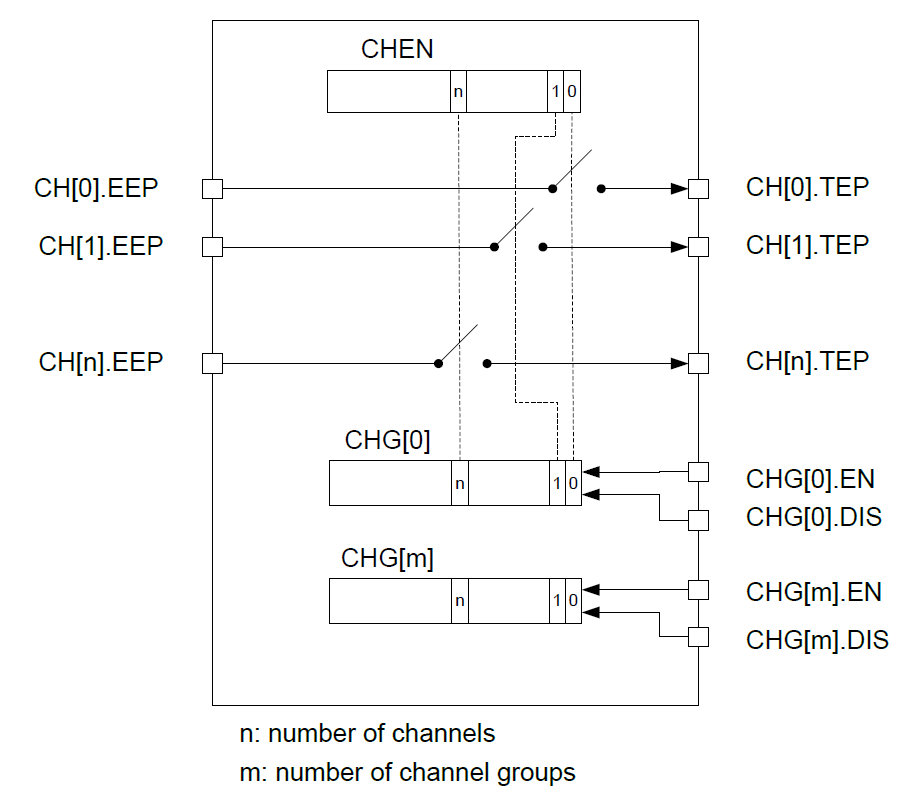


図１６：PPIブロック図

## 機能説明

プログラマブル・ペリフェラル・相互接続（PPI）はCPUを使用することなしに、異なるペリフェラル同士が、タスクとイベントを使用し自動的に相互作用することを可能にします。

PPIはあるペリフェラルのイベント発生の結果として、他のペリフェラルのタスクが自動的にトリガーする機能を提供します。タスクとイベントはPPIチャンネルを通じて接続されます。PPIチャンネルは、イベント・エンドポイント（EEP）とタスク・エンドポイント（TEP）の二つのエンドポイント・レジスターにより構成されます。ペリフェラルのタスクは、タスクと関連付けられたタスク・レジスターのアドレスを使用し、タスク・エンドポイントと接続されます。同様にペリフェラルのイベントは、イベントと関連付けられたイベント・レジスターのアドレスを使用し、イベント・エンドポイントと接続されます。

PPIチャンネルの有効化／無効化方法は２つあります。

* CHEN, CHENSET, CHENCLRレジスターを使用し、各々のPPIチャンネルを有効化／無効化します。
* PPIチャンネルグループのグループENABLE／DISABLEタスクを通して、PPIチャンネルを有効化／無効化します。これらのタスク開始がトリガーされる前に、どのPPIチャンネルがどのグループに属しているか、PPIチャンネルグループを設定する必要があります。

PPIタスク（例えば、CHG0EN）は、他のタスクと同様にPPIを通してトリガーされます。これはPPIチャンネルをTEPのようにとらえることができるという事です。複数のチャンネルをしようすれば、一つのイベントから複数のタスクをトリガーすることができ、また同様の方法で、一つのタスクを複数のイベントによりトリガーすることもできます。

プレプログラムド・チャンネル

ページ●●のテーブル90：プレプログラムド・チャンネルに表示されているよう、PPIチャンネルのいくつかは、事前にプログラムされています。これらのチャンネルはCPUにより設定することは不可能ですが、汎用PPIチャンネルのように、グループに追加、有効化／無効化することができます。

## レジスター概要

（略）

## レジスター詳細

（略）

# 2.4 GHz 無線 (RADIO)

# タイマー／カウンター (TIMER)

# リアルタイムカウンター (RTC)

# ウォッチドッグタイマー (WDT)

# 乱数発生器 (RNG)

# 温度センサー (TEMP)

# AES EBCモード暗号化 (ECB)

# AES CCMモード暗号化 (CCM)

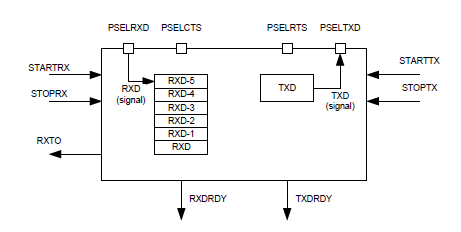
# 促進的アドレス解決 (AAR)

# シリアル・ペリフェラル・インターフェイス (SPI) マスター

# シリアル・ペリフェラル・インターフェイス (SPI) スレーブ

# I2C互換 ２線インターフェイス (TWI)

# 汎用非同期送受信機 (UART)



**Figure 67: UART configuration**

## 機能説明

UARTは以下の機能をサポートしています。

• 全２重動作

•自動フロー制御

• Parity checking and generation for the 9th data bit　9番目のデータービットへのパリティチェック及び生成

本UARTの構造は本ページFigure 67となります。 本UARTはTXDおよびRXDを直接利用し、データ送信およびデータ受信します。 本UARTは1つのストップビットを用います。

## ピン　コンフィグレーション

　UARTに関連するピンは、それぞれ信号、RXD、CTS(Clear To Send, アクティブLOW)、RTS（Request To Send, アクティブLOW）、TXDが、PSELRXD、PSELCTS、PSELRTS及びPSELTXDのレジスタ設定によって、物理ピンにマッピングされます。

If a value of 0xFFFFFFFF is specified in any of these registers, the associated UART signal will not be connected to any physical pin. The PSELRXD, PSELCTS, PSELRTS, and PSELTXD registers and their configurations are only used as long as the UART is enabled, and retained only for the duration the device is in ON mode. PSELRXD, PSELRTS, PSELTRTS and PSELTXD must only be configured when the UART is disabled.

To secure correct signal levels on the pins by the UART when the system is in OFF mode, the pins must be configured in the GPIO peripheral as described in Table 273: GPIO configuration on page 151.

Only one peripheral can be assigned to drive a particular GPIO pin at a time. Failing to do so may result in unpredictable behavior.

**Table 273: GPIO configuration**

|  |  |  |
| --- | --- | --- |
| UART pin | Direction | Output value |
| **RXC** | **Input** | **Not applicable** |
| **CTS** | **Input** | **Not applicable** |
| **RTS** | **Output** | **1** |
| **TXD** | **Output** | **1** |

## Shared resources

The UART shares registers and other resources with other peripherals that have the same ID as the UART. Therefore, you must disable all peripherals that have the same ID as the UART before the UART can be 29 Universal Asynchronous Receiver/Transmitter (UART) Page 152

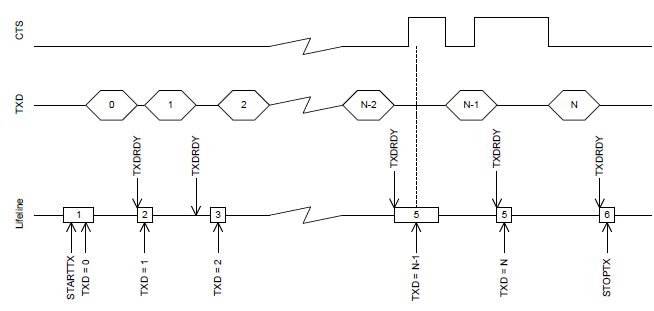
configured and used. Disabling a peripheral that has the same ID as the UART will not reset any of the registers that are shared with the UART. It is therefore important to configure all relevant UART registers explicitly to ensure that it operates correctly.

See the Instantiation table in Instantiation on page 17 for details on peripherals and their IDs.

## 送信

単体のUART 送信シーケンスはSTARTTXタスクを要因して開始されます。 複数バイトデータはTXDレジスタへの書き込みによって転送されます。1バイト送信成功時、 UARTは次の1バイトを TXD レジスタに書き込み可能状態となった後、TXDRDYイベントを出力します。 UART 送信シーケンスはSTOPTXタスクを要因として速やかに停止されます。

フロー制御が有効な場合、CTSが非アクティブになったとき、および再開されてCTSが再びアクティブになったとき、送信は自動的にサスペンドされます。（Figure 68） CTSが非アクティブにされたとき、送信中の1バイトデータは、転送がサスペンドに入る前に完全に送信されます。



**Figure 68: UART送信**

## 受信

A UART 受信シーケンスはSTARTRX タスクを要因として開始されます。 The UART 受信 chain implements a FIFO capable of storing six incoming RXD bytes before data is overwritten. Bytes are extracted from this FIFO by reading the RXD register. When a byte is extracted from the FIFO a new byte pending in the FIFO will be moved to the RXD register. The UART will generate an RXDRDY event every time a new byte is moved to the RXD register.

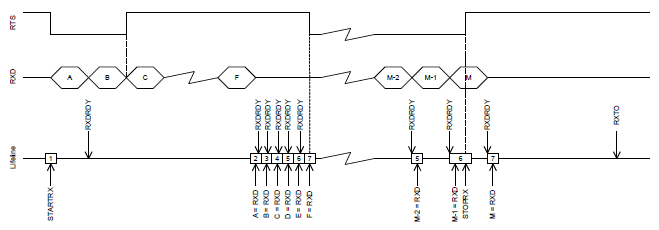
When flow control is enabled, the UART will deactivate the RTS signal when there is only space for four more bytes in the receiver FIFO. The counterpart transmitter is therefore able to send up to four bytes afterthe RTS signal is deactivated before data is being overwritten. To prevent overwriting data in the FIFO, the counterpart UART transmitter must therefore make sure to stop transmitting data within four bytes after the RTS line is deactivated.

The RTS signal will first be activated again when the FIFO has been emptied, that is, when all bytes in the FIFO have been read by the CPU, see Figure 69: UART reception on page 153.

The RTS signal will also be deactivated when the receiver is stopped through the STOPRX task as illustrated in Figure 69: UART reception on page 153. The UART will be able to receive up to four bytes if they are sent in succession immediately after the RTS signal has been deactivated. This is possible because the UART is, even after the STOPRX task is triggered, able to receive bytes for an extended period equal to the 29 Universal Asynchronous Receiver/Transmitter (UART) Page 153 time it takes to send four bytes on the configured baud rate. The UART will generate a receiver timeout event (RXTO) when this period has elapsed.

To prevent loss of incoming data the RXD register must only be read one time following every RXDRDY event.

To secure that the CPU can detect all incoming RXDRDY events through the RXDRDY event register, the RXDRDY event register must be cleared before the RXD register is read. The reason for this is that the UART is allowed to write a new byte to the RXD register, and therefore can also generate a new event, immediately after the RXD register is read (emptied) by the CPU.



**Figure 69: UART reception**

As indicated in occurrence 2 in Figure 69: UART reception on page 153, the RXDRDY event associated with byte B is generated first after byte A has been extracted from RXD.

## Suspending the UART

The UART can be suspended by triggering the SUSPEND task. SUSPEND will affect both the UART receiver and the UART transmitter, i.e. the transmitter will stop transmitting and the receiver will stop receiving. UART transmission and reception can be resumed, after being suspended, by triggering STARTTX and STARTRX respectively.

Following a SUSPEND task, an ongoing TXD byte transmission will be completed before the UART is suspended.

When the SUSPEND task is triggered, the UART receiver will behave in the same way as it does when the STOPRX task is triggered.

## Error conditions

有効なストップビットがフレームで検出されていない場合、エラーイベントは、フレーミングエラーの形で、生成されます。 RXDラインは、データフレームの長さよりも長いため、アクティブローに保持されている場合は別のエラーイベントは、ブレーク条件の形で、生成されます。ブレーク条件が発生する前に、フレーミングエラーが常に発生します。

## Using the UART without flow control

If flow control is not enabled the interface will behave as if the CTS and RTS lines are kept active all the time.

## Parity configuration

When parity is enabled, the parity will be generated automatically from the even parity of TXD and RXD for transmission and reception respectively.

## Register Overview

**Table 274: Instances**

**Table 275: Register Overview**

|  |  |  |
| --- | --- | --- |
| レジスタ | Offset | 説明 |
| **Tasks** |  |  |
| **STARTRX** | **0x000** | **UART受信機の開始** |
| **STOPRX** | **0x004** | **UART受信機の停止** |
| **STARTTX** | **0x008** | **UART送信機の開始** |
| **STOPTX** | **0x00C** | **UART送信機の停止** |
| **SUSPEND** | **0x01C** | **UARTの一時中断** |
| **Events** |  |  |
| **CTS** | **0x100** | **CTS is activated (set low). Clear To Send.** |
| **NCTS** | **0x104** | **CTS is deactivated (set high). Not Clear To Send.** |
| **RXDRDY** | **0x108** | **Data received in RXD** |
| **TXDRDY** | **0x11C** | **Data sent from TXD** |
| **ERROR** | **0x11C** | **Error detected** |
| **RXTO** | **0x144** | **Receiver timeout** |
| **Registers** |  |  |
| **INTEN** | **0x300** | **割り込みの利用可／不可** |
| **INTENSET** | **0x304** | **割り込みの有効化** |
| **INTENCLR** | **0x308** | **割り込みの無効化** |
| **ERRORSRC** | **0x480** | **エラー原因** |
| **ENABLE** | **0x500** | **UARTの有効化** |
| **PSELRTS** | **0x508** | **Pin select for RTS** |
| **PSELTXD** | **0x50C** | **Pin select for TXD** |
| **PSELCTS** | **0x510** | **Pin select for CTS** |
| **PSELRXD** | **0x514** | **Pin select for RXD** |
| **RXD** | **0x518** | **RXD register** |
| **TXD** | **0x51C** | **TXD register** |
| **BAUDRATE** | **0x524** | **ボーレート** |
| **CONFIG** | **0x56C** | **パリティ、ハードウェアフロー制御の設定** |

**29.11 Register Details**

**Table 276: INTEN**

**Table 277: INTENSET**

**Table 278: INTENCLR**

**Table 279: ERRORSRC**

**Table 280: ENABLE**

**Table 281: PSELRTS**

**Table 282: PSELTXD**

**Table 283: PSELCTS**

**Table 284: PSELRXD**

**Table 285: RXD**

**Table 286: TXD**

**Table 287: BAUDRATE**

**Table 288: CONFIG**

# クアッド・レーター・デコーダー (QDEC)

# A/Dコンバーター (ADC)

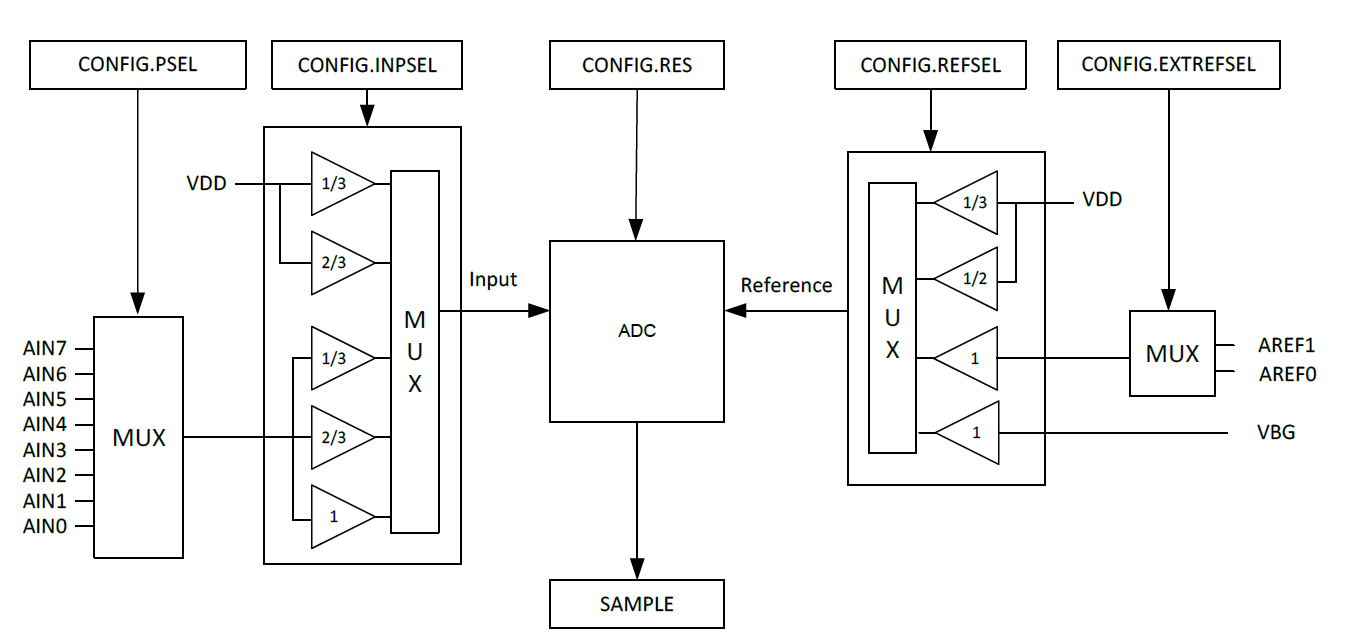


図71: A/D コンバーター

## A/Dコンバーター (ADC)

### 入力電圧レンジの設定

　ADCの設定を行い、入力電圧レンジとADC電圧レンジを一致させることは非常に重要です。

　もし、入力電圧レンジが ADC 電圧レンジより低い場合、解像度が十分に活用されなくなります。

　もし入力電圧レンジがADC電圧レンジより高い場合、ADC最大値を超える値はすべて、最大値となってしまいます。これはサチレーションポイント（飽和点）と呼ばれます。

　入力電圧レンジとサチレーションポイントはADCリファレンス電圧と選択されたプリスケールに因ります。もし、1.2 V VBG内部リファレンス電圧を使用する場合、ADCレンジは0～1.2 V（サチレーションポイント1.2 V）となります。これは、AINシグナルにおいて1/1プリスケールを使用している場合、適切な変換結果を得るためには、入力レンジが0～1.2 Vである必要があることを意味しています。1.2 Vを超える入力はADCの最大値に変換されます。

　しかし、もし例えば1/3プリスケールを使用した場合、AINの入力は1/3に縮小します。これによりAINの入力電圧範囲は3.6 V入力が3.6 / 3 =1.2 V に縮小されるため、0～3.6 Vになります。表１に参照電圧とプリスケール設定と対応するサチレーションポイントのADC AIN入力を示します。

表311：サチレーションポイント例

|  |  |  |
| --- | --- | --- |
| リファレンス | プリスケール | AIN最大電圧 |
| 1.2 V VBG | 1/1 | 1.2 V |
| 1.2 V VBG | 2/3 | 1.8 V |
| 1.2 V VBG | 1/3 | 3.6 V |
| 1.0 V AREF | 1/1 | 1.0 V |
| 1.0 V AREF | 2/3 | 1.5 V |
| 1.0 V AREF | 1/3 | 3.0 V |
| VDD 3.0V VDD 1/2 | 1/1 | 1.5 V |

**分圧器**

　下記2つのルールがAINピンへの最大入力許可電圧を規定するルールです。

1. ADCにおいてAINピンは、プリスケール後に2.4 Vを超える電圧を印加してはいけません。

入力電圧　×　プリスケール = 最大 2.4 V

1. GPIO ピンにはVDD + 0.3 V より高い電圧を印加してはいけません。nRF51X22 製品仕様の最大定格を参照してください。

　例えば、2/3プリスケールを使用している場合、AIN ピンには、2.4 V / (2/3) = 3.6 V まで印加することができます。そして、ルール２に違反しないためには、VDDは3.3 V 以上である必要があります。

　表２に供給電圧と、プリスケール設定に応じたADCのAINピンに印加可能な最大電圧の例を示します。

表312　AIN最大電圧例

|  |  |  |  |
| --- | --- | --- | --- |
| 供給電圧 | プリスケール | AIN 最大電圧 | 制限ルール |
| 3.6 V | 1/1 | 2.4 V | ルール１ |
| 3.6 V | 2/3 | 3.6 V | ルール１ |
| 3.6 V | 1/3 | 3.9 V | ルール１ |
| 3.3 V | 1/1 | 2.4 V | ルール１ |
| 3.3 V | 2/3 | 3.6 V | ルール１＆ルール２ |
| 3.3 V | 1/3 | 3.3 V | ルール２ |
| 1.8 V | 1/1 | 2.1 V | ルール２ |
| 1.8 V | 2/3 | 2.1 V | ルール２ |
| 1.8 V | 1/3 | 2.1 V | ルール２ |

　もし、AINに印加可能な最大電圧以上を測定したい場合は、分圧器を使用する必要があります。ページ６、セクション2.4の 『分圧器を使用した電圧降下』を参照してください。

### 分圧器を使用した電圧降下

　センサー、バッテリーの出力電圧がADCの電圧範囲を超えている場合、ADCの入力ピンに電圧を印加する前に、電圧を下げる必要があります。分圧器を使用してこれを行うことができます。Li-Ionバッテリーから電圧を下げる例を図３に示します。

　ADCの内部インピーダンスのため、大きな抵抗値を持つ分圧器はADCの出力に対して、誤差を生じます。分圧器のインピーダンスが1k#以下の場合、誤差は無視できるほど小さくなります。分圧器のインピーダンスを大きくするほど、誤差も大きくなります。

　しかし、分圧器を流れる電流を小さくするため、分圧器に高い抵抗値を使用したいという望みもあります。高い抵抗値においても誤差を減らすためにはAINとGNDの間にコンデンサーを追加することによって実現できます。大きな値のコンデンサーはADCの出力誤差を減らしますが、代わりにサンプリング周期も減らせることとなります。

　UAINの電圧を下げるため、RAINが120k～400k#の分圧器が接続されている状態でサンプリングする場合。またAIN、GND間にキャパシタが接続されており、AIN電圧をサンプリングの間、十分な時間前回のレベルに保ちR2の高い抵抗値の影響を最小限にすることができます。キャパシタ容量は次回のサンプル時までに満たされるほど小さくなくてはなりません。すなわち、キャパシタが接続されている場合、そのサイズは精確さおサンプリング周期のトレードオフになる。サンプリングを行っていない間は、RAINは高い値であり、それを接続されていないとみなすことができます。

　入力電圧レンジがADC電圧レンジを超えており、高い正確性と高いサンプリング周期を求められる場合、電圧バッファーが必要となります。

　他の可能性としてはFETトランスミッターを電源と分圧器間に接続し、サンプリング前に、一時的にオープンにし、分圧器に電流を流します。分圧器は（1k#以下の）低い抵抗値にすることが可能で、キャパシタを必要としません。分圧器はサンプリング時に比較的大きな電流を消費し、サンプリングをしていないときは全く電流を消費しません。

### 入力インピーダンス

nRF51822製品仕様に記載されているADC誤差仕様を実現するために、電源に接続の出力インピーダンスは1k#以下でなくてはなりません。出力インピーダンスが1k#以下である利点は、ADC入力において異なるプリスケールの設定においてもADCの正確性に特に影響がないことです。

　もしハイインピーダンスの電源が選択された場合、追加ゲインとオフセット誤差が加わり、これはまた、プリスケール設定の差により変わります。

　図４はADCのサンプリングを行っている、nRF51 ADCの入力モデルで、表５は異なるプリスケールの設定によるRAINの値を示します。内部VBG参照電圧が 1.2 V であれば、ADC内部電源は VBG/2 = 0.6 V になります。

　ADCがサンプリングを行っていないとき、AINピンはとてもハイインピーダンスであり、接続されていないとみなすことができます。表５は異なるプリスケール設定による内部インピーダンスの統計値です。99.7%のデバイス（±3σ）が6.3%以内であると期待されます。たとえば、1/1プリスケールの場合121.5～137.9k#です。

表313：RAINの入力インピーダンス統計値

|  |  |  |
| --- | --- | --- |
| プリスケール | 平均インピーダンス | 標準偏差 |
| 1/1 | 129.7 kΩ | 2.74 kΩ |
| 2/3 | 194.6 kΩ | 4.1 kΩ |
| 1/3 | 389.2 kΩ | 8.2 kΩ |

### 設定

　入力の選択、参照の選択、分解能、プリスケール、等の全てのパラメーターは CONFIG レジスターを使用し、設定します。

注：ADC変換の実施中(ADC BUSY中)に、ADCの設定を行ってはなりません。

### 使用方法

　ADC変換はSTARTタスクを使用することにより開始します。STARTタスクはCPUからタスク・レジスターに直接書き込む方法、PPIを使用しタスクの実施を開始する方法、どちらでも可能です。

　サンプリングの実施中、ADCはBUSY状態になります。ADCがBUSY状態、もしくはREADY状態なのかは、BUSYレジスターをモニターすることにより判別可能です。

　ADC変換が完了した場合、ENDイベントが生成され、変換結果はRESULTレジスターから読み取ることができます。

　省電力のためADCのアナログ回路は、ADC変換の完了後にパワーダウンします。

### 1ショット／連続動作

　ADCそれ自身は、１ショット操作のみサポートします。それは、各々の変換はSTARTタスクを使用し、明示的に開始させる必要があることを意味しています。

　しかし、連続的なADC操作は、例えばPPIをタイマーを通し、連続的にSTARTタスクをトリガーすることにより、実施されます。

### ピン設定

　ユーザーはAIN0からAIN7までのアナログ入力ピンをのうち一つを選択するため、PSELレジスターを使用することができます。デバイスに応じ、どのアナログピンが使用可能かの情報に関する詳細はデバイス製品仕様を参照してください。選択されたアナログピンはENABLEレジスターにより有効化された際にADCに取得されます。アナログピンの選択方法はGPIOの章を参照してください。

### 共有リソース

ADCは、ADCと同じIDを持つレジスターとペリフェラルのリソースを共有しています。そのため、ユーザーはADCが設定及び使用可能になる前に、ADCと同じIDを持つすべてのペリフェラルを無効化しなくてはなりません。ADCはLPCMPと同じアナログピンを使用しています。そのためADCを有効化する前にLPCOMPを無効化してください。ADCが正しく動作することを保障するために、明示的に関連する全てのADCレジスターを設定することが重要です。

　ペリフェラル及びそのIDの詳細に関してはページ17のInstantation内にある、インスタシエインションテーブルを参照してください。

## レジスター概要

表314：インスタンス

|  |  |  |  |
| --- | --- | --- | --- |
| ベースアドレス | ペリフェラル | インスタンス | 説明 |
| 0x40007000 | ADC | ADC | アナログ・デジタルコンバーター |

表315：レジスター概要

|  |  |  |
| --- | --- | --- |
| レジスター | オフセット | 概要 |
| ***Tasks*** |  |  |
| START | 0x000 | ADC変換開始 |
| STOP | 0x004 | ADC変換停止 |
| ***Events*** |  |  |
| END | 0x100 | ADC変換完了 |
| ***Registers*** |  |  |
| INTEN | 0x300 | 割り込みの有効化／無効化 |
| INTENSET | 0x304 | 割り込み有効化 |
| INTENCLR | 0x308 | 割り込み無効化 |
| BUSY | 0x400 | ADCビジー（変換中） |
| ENABLE | 0x500 | ADC有効化。ADCが有効化された際に、ADCはCONFIGレジスターにより指定されたアナログ入力ピンへのアクセスを得ます。 |
| CONFIG | 0x504 | ADC設定 |
| RESULT | 0x508 | 前回のADC変換結果 |

## レジスター詳細

（略）

# 低出力コンパレーター (LPCOMP)

# ソフトウェア割り込み (SWI)

## 機能説明

いくつかの割り込みはソフトウェア割り込み用として予約されています。

## レジスター概要

表335：インスタンス

|  |  |  |  |
| --- | --- | --- | --- |
| ベースアドレス | ペリフェラル | インスタンス | 説明 |
| 0x40014000 | SWI | SWI0 | ソフトウェア割り込み |
| 0x40015000 | SWI | SWI1 | ソフトウェア割り込み |
| 0x40016000 | SWI | SWI2 | ソフトウェア割り込み |
| 0x40017000 | SWI | SWI3 | ソフトウェア割り込み |
| 0x40018000 | SWI | SWI4 | ソフトウェア割り込み |
| 0x40019000 | SWI | SWI5 | ソフトウェア割り込み |

**訳：プログレス・テクノロジーズ**

コロニー第一事業部 tsumikiグループ　黒田<[e-kuroda@progresstech.jp](mailto:e-kuroda@progresstech.jp)>

追訳、追記、検証、修正歓迎