応用記述

AT11487 : XMEGA Xplainedｷｯﾄ用低消費電力技法

Atmel AVR XMEGA

序説

この応用記述はXMEGA® Xplainedｷｯﾄに対して可能な最低電力値を達成するための技法を説明します。この応用記述では参照基準ｷｯﾄとしてXMEGA-A3BU Xplainedｷｯﾄが使われ、目的対象ﾃﾞﾊﾞｲｽを書くための書き込み器としてAtmel-ICEが使われます。加えて、この応用記述は様々な休止動作形態へ移行するためのｺｰﾄﾞ例を提供し、各種休止動作形態に於けるXMEGA-A3BU Xplained基盤によって消費される電流値を提供します。

この応用記述は周辺機能(参照記述としてA/D変換器(ADC)が使われます。)が動いている間にﾃﾞﾊﾞｲｽの消費電流をどう低減するかも説明します。

本書は一般の方々の便宜のため有志により作成されたもので、Atmel社とは無関係であることを御承知ください。しおりの[はじめに]での内容にご注意ください。

目次

1. 消費電力に影響を及ぼす要素 ・・・

1. 消費電力に影響を及ぼす要素

動作電圧、動作周波数などのような多くの要素が消費電力に影響を及ぼしますが、いくつかは本質的にその他よりももっと影響を及ぼします。以下で一覧にされるものは勧告と考察を伴う重要な要素です。

1.1. 動作電圧

消費電力はﾃﾞﾊﾞｲｽの電源電圧の2乗に比例し、故に可能な限り低く保たれるべきです。

供給電圧の低減は最大ｼｽﾃﾑ ｸﾛｯｸ周波数の限度をより低くし得て、故に与えられたｺｰﾄﾞ量を実行するのに活動(ACTIVE)動作で必要とされる時間を増します。

可能な限り低い電圧を使うことによって消費電力を最小にしてください。

1.2. 活動動作

活動(ACTIVE)動作では、換言すると休止動作が使われない時に、消費電力はｼｽﾃﾑ ｸﾛｯｸ周波数に比例します。これは休止動作が使われない場合、消費電力を最小にするのに可能な最低ｼｽﾃﾑ ｸﾛｯｸ周波数で走行すべきことを意味します。

休止動作が使われない場合、可能な限り低い周波数を保つことによって消費電力を最小にしてください。

1.3. ｸﾛｯｸ前置分周

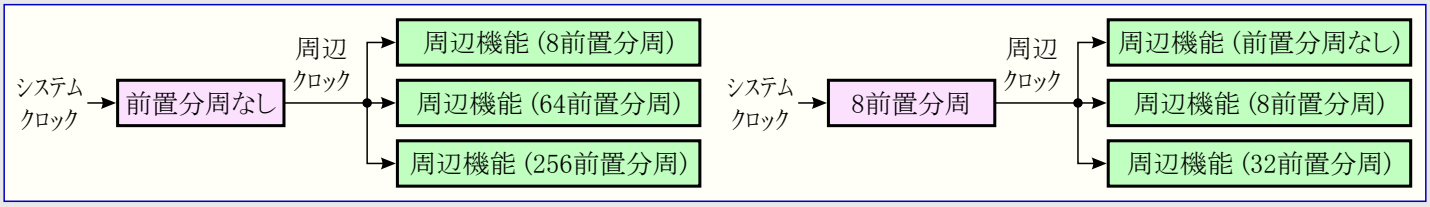
例え活動(ACTIVE)動作で費やされる時間を最小とするために可能な限りCPUを速く走行することが推奨されるとは言え、ｸﾛｯｸ速度を減らす方がより良い状況があります。これらの状況は一般的に一定の時間量がかかる、例えば直列通信のｺｰﾄﾞに対して活動(ACTIVE)動作またはｱｲﾄﾞﾙ(IDLE)動作での待機を意味します。これらの場合、活性な周辺機能に対して必要とされるのよりも高いCPUと周辺機能のｸﾛｯｸ周波数の生成を避けるべきです。これはｸﾛｯｸ信号での不具合発生なしで変更することができるｸﾛｯｸ前置分周器を使うことによって達成することができます。

前置分周が周辺機能の内側で行われる場合、消費電力はｸﾛｯｸ分配連結で利用可能な最初の前置分周器に於いて最も大きな共通係数で前置分周することによって浪費を防ぐことができます。この原理は図1-1.で図解されます。

注: 前置分周はCPUｸﾛｯｸにも影響を及ぼすため、活動(ACTIVE)動作この共通前置分周を実行するのは計算がより長くかかるので常に好ましいとは限りません。

特に活動(ACTIVE)動作とｱｲﾄﾞﾙ(IDLE)動作で待機する時に、効果的に前置分周器を使うことによって消費電力を最小にしてください。

図1-1. 最大係数による前置分周を持たない/持つ周辺機能部



1.4. ｸﾛｯｸ元選択

実際に必要とされるよりもより高いｼｽﾃﾑ ｸﾛｯｸ速度を生成することを避けるべきです。理想的な場合では前置分周が不要です。これは適切なｸﾛｯｸ元を選択することによって達成することができます。

例として、16MHzへの前置分周を持つ32MHz RC発振器よりもむしろ参照基準として2MHz RC発振器を持つPLLの使用によって16MHzのｼｽﾃﾑ ｸﾛｯｸを生成することが望まれます。16MHzの外部ｸﾛｯｸ元も良い選択になり得、特にそれらがｼｽﾃﾑで既に利用可能な場合、故に追加費用なしになります。

ﾃﾞﾊﾞｲｽに対する起動遅延はｼｽﾃﾑ ｸﾛｯｸにどのｸﾛｯｸ元が使われるかに依存します。この遅延を減らす1つの方法はﾃﾞﾊﾞｲｽが休止になって高速応答ｸﾛｯｸ元で起動するようにｸﾛｯｸ元を切り替えることです。

ｸﾛｯｸ速度を減らすために前置分周だけに頼るよりもむしろｸﾛｯｸ元を切り替えることによって消費電力を最小にしてください。

1.5. 起動遅延

ﾃﾞﾊﾞｲｽが(2つのｽﾀﾝﾊﾞｲ(STANDBY)動作の例外を含み)ｱｲﾄﾞﾙ(IDLE)動作よりも深い休止動作から起動するとき、CPUが動作を開始する前にｼｽﾃﾑ ｸﾛｯｸ元は安定でなければなりません。これは選択したｸﾛｯｸ元に依存した短い遅延を誘引します。内部RC発振器または外部ｸﾛｯｸが使われる場合、始動遅延は6周期です。これはRC発振器始動時間に加えてです。ｸﾘｽﾀﾙ用発振器が使われる場合、始動時間は設定可能です。周波数安定性が必要とされる場合、各々、ｾﾗﾐｯｸ振動子に対して1000周期、水晶発振子に対して16000周期の始動遅延が推奨されます。これは振動子/発振子と負荷容量に依存する発振器始動時間に加えてです。

加えて、起動後に割り込み処理ﾙｰﾁﾝ(ISR:Interrupt Service Routine)が実行を開始する前に13周期の最小遅延があります。これはﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀがﾌﾟｯｼｭ(ｽﾀｯｸに保存)されてISRへ飛ぶためです。

始動時間中の消費電力はｱｲﾄﾞﾙ(IDLE)動作での消費電力に近く、故に"非効率的"電力を意味します。従って、可能ならば、可能な限り滅多に起動しない、そしてﾃﾞﾊﾞｲｽ起動毎にむしろ"もっと行う"ことが推奨されます。

起動遅延を最小にして電力を浪費しないように、RC発振器か外部ｸﾛｯｸ元を選び、可能な限り滅多に起動しないようにしてください。

1.6. 電力削減ﾚｼﾞｽﾀ

大抵の周辺機能と内部単位部は活動(ACTIVE)動作とｱｲﾄﾞﾙ(IDLE)動作でそれらの電力引き込みを避けるために個別に停止することができます。これはそれらを周辺機能ｸﾛｯｸ領域から切断させる電力削減ﾚｼﾞｽﾀ(PRR)内のそれら各々のﾋﾞｯﾄを設定(1)することによって行われます。効果的な電力削減のため、PRRﾋﾞｯﾄを設定(1)する前にそれら各々の制御ﾚｼﾞｽﾀ経由で単位部や周辺機能を禁止することが必要とされます。いくつかの単位部はそれらのPRRﾋﾞｯﾄ解除(0)後に再初期化されなければなりません。より多くの情報に関してはﾃﾞｰﾀｼｰﾄ手引書で個別のPRRﾋﾞｯﾄについての項を参照してください。

ﾊﾟﾜｰｾｰﾌﾞ(POWER-SAVE)とﾊﾟﾜｰﾀﾞｳﾝ(POWER-DOWN)では周辺機能ｸﾛｯｸ領域が禁止されるため、PRRに拘わらず単位部が停止されます。

消費電力を最小とするために、使われない周辺機能と単位部を禁止するのにPRRを使ってください。

1.7. RTCｸﾛｯｸ元

ｱｲﾄﾞﾙ(IDLE)、ﾊﾟﾜｰｾｰﾌﾞ(POWER-SAVE)、拡張ｽﾀﾝﾊﾞｲ(EXTENDED STANBY)を使う理由の1つはRTCは、これのｸﾛｯｸはこれらの休止形態で活性です。RTCは一般的に間隔時間でﾃﾞﾊﾞｲｽを起こすのに使われます。

大抵のAtmel® AVR® XMEGA®系統に関して、RTCをｸﾛｯｸ駆動するのに外部32kHzｸﾘｽﾀﾙ用発振器、内部32kHz RC発振器、内部32kHz超低電力(ULP)発振器の3つの異なる発振器を使うことができます。全ての場合に於いて、前置分周された1kHzｸﾛｯｸ信号が利用可能で、消費電力低減のために使われるべきです。外部32kHzｸﾘｽﾀﾙ用発振器については特別な低電力動作(X32KLP M)も利用可能です。

X32KLPM許可で外部32kHzｸﾘｽﾀﾙ用発振器を使うことが推奨されます。これはULPよりも低い消費電力と、更に内部RC発振器よりも大きな精度を与えます。この発振器はこのような低周波数が受け入れ可能な場合にｼｽﾃﾑ ｸﾛｯｸ元としても使われ得ます。

表1-1. ｸﾛｯｸ元とRTCに対する精度と消費電流の例

発振器	精度 (注3)	消費電流 (注1)
32kHz超低電力発振器 (ULP)	±30%	1μA
内部32kHz RC発振器 (RC32k)	±1%	30μA
外部32kHzｸﾘｽﾀﾙ用発振器 (TOSC) (注2)	±0.001% (10ppm)	0.6μA

注1: 動作電圧3V

注2: 例えばｸﾘｽﾀﾙの品質に依存

注3: 正確な値と条件についてはﾃﾞｰﾀｼｰﾄを参照してください。ここで述べられた値は指針としての意味だけです。

電池代替単位部と32ﾋﾞｯﾄRTCを持つAtmel® AVR® XMEGA®系統についてはｸﾛｯｸ元として32kHzｸﾘｽﾀﾙ用発振器だけが使われ得ます。これらのﾃﾞﾊﾞｲｽでは休止に拘わらずRTCは走行のままにされます。

低電力動作での外部ｸﾘｽﾀﾙ用発振器の1kHzでRTCをｸﾛｯｸ駆動することによって消費電力を最小にしてください。

1.8. ﾃﾞｼﾞﾀﾙ入出力ﾋﾟﾝの状態

全てのﾃﾞｼﾞﾀﾙ入出力ﾋﾟﾝは既定によって入力として構成設定され、それらはﾊｰﾄﾞｳｪｱの衝突を避けるために浮いています。全ての未使用ﾋﾟﾝは取り付けられる外部部品なしで未接続のままにすることができます。これらのﾋﾟﾝがﾃﾞｼﾞﾀﾙ入力緩衝部を持つため、突発的な内部切り替えと漏れを避けるために入出力ﾋﾟﾝのﾚﾍﾞﾙを上手く定義されるのを保証することが重要です。浮いた入出力によって引き起こされる漏れは相対的に小さく、主に休止に於いて注目すべきですが、ﾋﾟﾝの状態をHighかLowのどちらかであることを保証することによって最小にすることができます。従って使用者は消費電力を最小とするため、これらのﾋﾟﾝで内部のﾌﾟﾙｱｯﾌﾟまたはﾌﾟﾙﾀﾞｳﾝを許可すべきです。

ﾋﾟﾝがｱﾅﾛｸﾞ源に接続される場合、例えそれが入力として構成設定されていても、そのﾋﾟﾝのﾃﾞｼﾞﾀﾙ入力緩衝部は禁止されるべきです。これは個別ﾎﾟｰﾄに対するﾋﾟﾝn構成設定(PINnCTRL)ﾚｼﾞｽﾀの使用によって行われます。この構成設定をどう行うかについてはﾃﾞﾊﾞｲｽ手引書を参照してください。

消費電力を最小にするため、未使用ﾋﾟﾝのﾌﾟﾙｱｯﾌﾟかﾌﾟﾙﾀﾞｳﾝを許可し、ｱﾅﾛｸﾞ源が接続されたﾋﾟﾝのﾃﾞｼﾞﾀﾙ入力緩衝部を禁止してください。

1.9. 仮想ﾎﾟｰﾄ ﾚｼﾞｽﾀ

活動(ACTIVE)動作で費やす時間を最小とするため、仮想ﾎﾟｰﾄ ﾚｼﾞｽﾀを使うことができます。これは4つまでの入出力ﾎﾟｰﾄに対して、DIR,IN,OUT,INTGLAGSﾚｼﾞｽﾀに対するﾋﾞｯﾄ操作とIN,OUTのようなI/Oﾒﾓﾘ特定命令での単一周期ｱｸｾｽを許します。

消費電力を最小にするため、入出力ﾎﾟｰﾄ ｱｸｾｽに対して仮想ﾎﾟｰﾄ ﾚｼﾞｽﾀを使ってださい。

1.10. 汎用入出力ﾚｼﾞｽﾀ

活動(ACTIVE)動作で費やす時間を最小とするためのべつな方法は、変数の記憶に汎用入出力(GPIO)ﾚｼﾞｽﾀを使うことです。これはI/Oﾒﾓﾘ特定命令での単一周期ｱｸｾｽの可能性のためでもあります。

注: GPIOﾚｼﾞｽﾀは揮発性(volatile)として定義され、故に或る場合に於いてこれらのﾚｼﾞｽﾀに格納された変数を操作する時に一時変数が使われるべきです。さもなければ、動作利得を失うでしょう。

消費電力を最小にするため、変数記憶に汎用入出力(GPIO)ﾚｼﾞｽﾀを使ってださい。

1.11. ｳｫｯﾁﾄﾞｯｸﾞ

ｳｫｯﾁﾄﾞｯｸﾞは基本的に独立したｸﾛｯｸ元を持つ計時器です。許可された場合、休止に於ける電力消費の一因になります。ｳｫｯﾁﾄﾞｯｸﾞは1kHzに前置分周された内部32kHz超低電力(ULP)発振器によってだけｸﾛｯｸ駆動され得ます。

消費電力を最小とするために、ｳｫｯﾁﾄﾞｯｸﾞを禁止してください。

1.12. 低電圧検出器

低電圧検出器(BOD:Brown Out Detector)の目的はﾃﾞﾊﾞｲｽが低すぎる電圧で動かないことを保証することです。ﾃﾞﾊﾞｲｽが常に仕様内で動くことを保証するために内部BODを使うことが強く推奨されます。

けれども、休止中にﾃﾞﾊﾞｲｽは"動いていない"、いやむしろｺｰﾄﾞを実行しません。この理由のため、BODは活動(ACTIVE)/ｱｲﾄﾞﾙ(IDLE)と休止動作に対して独立して構成設定することができます。これは活動(ACTIVE)とｱｲﾄﾞﾙ(IDLE)の動作でだけ許可されるBODを許します。BODの全ての構成設定はﾃﾞﾊﾞｲｽ ﾋｭｰｽﾞで行われます。

更に消費電流を減らすため、BODは採取動作で動くことができます。採取速度は前置分周されたULP発振器からｸﾛｯｸ駆動されるので概ね1kHzです。BODはこの動作に於いて採取間の電圧沈下を検知することができず、故に電池給電のように緩やかに変化する動作電圧の応用でだけ使われるべきです。

消費電力を減らすために、BODを禁止するか、より良いのは休止中にBODを禁止してください。動作電圧で緩やかな変化しかなさそうならば採取動作を使ってください。

1.13. JTAGｲﾝﾀｰﾌｪｰｽとﾁｯﾌﾟ上ﾃﾞﾊﾞｯｸﾞ

JTAGｲﾝﾀｰﾌｪｰｽはﾌﾟﾛｸﾞﾗﾐﾝｸﾞとﾃﾞﾊﾞｯｸﾞに使われますが、最終製品の動作中に何の機能を持ちません。ﾁｯﾌﾟ上ﾃﾞﾊﾞｯｸﾞ(OCD:On-chip Debugging)機能が許可されると、それは休止中にｸﾛｯｸ駆動されて活性です。従ってOCDとJTAGｲﾝﾀｰﾌｪｰｽは必要とされない場合に禁止されるべきです。

OCDはﾋｭｰｽﾞで禁止することができ、一方JTAGｲﾝﾀｰﾌｪｰｽはﾋｭｰｽﾞとｿﾌﾄｳｪｱの両方で禁止することができます。ｿﾌﾄｳｪｱでのJTAG禁止は、JTAGｲﾝﾀｰﾌｪｰｽがﾘｾｯﾄで再許可されるので、ﾃﾞﾊﾞｲｽが再ﾌﾟﾛｸﾞﾗﾐﾝｸﾞできるのを保証します。

代わりに、ﾌﾟﾛｸﾞﾗﾐﾝｸﾞとﾃﾞﾊﾞｯｸﾞにPDIｲﾝﾀｰﾌｪｽを使うことができます。この場合、JTAGｲﾝﾀｰﾌｪｰｽは全てに必要とされず、ﾋｭｰｽﾞによって禁止することができます。PDIｲﾝﾀｰﾌｪｰｽは全ての休止動作でも働きます。

消費電力を最小とするために、OCDとJTAGｲﾝﾀｰﾌｪｰｽを禁止してください。

1.14. ﾌﾗｯｼｭ ﾒﾓﾘとEEPROMの電力削減動作

Atmel® AVR® XMEGA®不揮発性ﾒﾓﾘ(NVM:Non-Volatile Memory)制御部に於いて、EEPROMとﾌﾗｯｼｭ ﾒﾓﾘに対して電力削減動作を許可することが可能です。これらの動作形態に於いて、ﾌﾗｯｼｭ ﾒﾓﾘの現在の未使用領域(換言すると、応用領域またはﾌﾞｰﾄ領域)とEEPROMは活動(ACTIVE)動作で、それらが丁度何れかの休止動作でのように電力低下されます。これらの電力削減動作は休止での消費電力に影響を及ぼしません。

CPUが電力削減動作中の不揮発性ﾒﾓﾘをｱｸｾｽしようとすると、ﾒﾓﾘが再活性される間にCPUはｱｲﾄﾞﾙ(IDLE)休止からの起動に対応する時間間隔の間停止されます。

注: ﾌﾗｯｼｭ ﾒﾓﾘの電力削減動作と休止に関する障害があります。影響を及ぼされるﾃﾞﾊﾞｲｽについて、対策は休止に以降する前にﾌﾗｯｼｭ電力削減動作を禁止し、そして起動で再びそれを許可することです。休止での消費電力はこれによって影響を及ぼされません。

活動(ACTIVE)動作での消費電力を削減するために、ﾌﾗｯｼｭ ﾒﾓﾘとEEPROMに対する電力削減動作を許可してください。

1.15. EEPROM書き込み

EEPROMに1ﾊﾞｲﾄよりも多くが書かれるべき場合、ﾊﾞｲﾄ単位書き込みを行うよりもむしろEEPROMﾍﾟｰｼﾞ緩衝部の使用を行うべきです。これはEEPROMへの完全なﾍﾟｰｼﾞ書き込みにかかるのが丁度1ﾊﾞｲﾄ書き込みと同じ長さがかかるためです。例えば、2ﾊﾞｲﾄが書かれるべき場合、必要な同じ長さの2倍かかります。EEPROM書き込み中に消費電力も増加するため、これは"2倍の不利"を与えます。

消費電力を最小とするため、EEPROMへのﾊﾞｲﾄ単位よりもむしろﾍﾟｰｼﾞ単位書き込みを使ってください。

1.16. 休止動作

殆どの応用では消費電力を最小にすることが好まれますが、主に系/製品の高速処理と素早い応答を保証するために、ｼｽﾃﾑ ｸﾛｯｸ周波数を下げるのは好まれません。このような応用ではそれらが処理することがない時にﾃﾞﾊﾞｲｽを低電力状態に置くために、Atmel® AVR® XMEGA®の"休止動作"を使うことができます。そしてその主な原則は可能な限り速く動き、可能な限り多く休止することです。例えば活動(ACTIVE)動作で許可されている不揮発性ﾒﾓﾘのため、可能な限り速く動くことは静的消費電力の影響を低減します(換言すると、ｸﾛｯｸ周波数と無関係)。

休止動作での消費電力と周辺機能の動作はどの休止動作が使われるかに依存します。表1-2.はXMEGAﾃﾞﾊﾞｲｽで利用可能な各種休止動作の特性を示します。応用はその間でどの動作形態が最適かに依存して、動作中に休止動作を切り替えることができます。

表1-2. 利用可能なXMEGA用休止動作形態

休止動作形態	活動クロック範囲			発振器		起動元			
	CPUクロック	周辺機能クロック	RTCクロック	システムクロック元	RTCクロック元	非同期ポート割り込み	TWIアドレス一致割り込み	RTC割り込み	その他割り込み
アイドル		○	○	○	○	○	○	○	○
パワーダウン						○	○		
パワーセーブ			○		○	○	○	○	
スタンバイ				○		○	○		
拡張スタンバイ			○	○	○	○	○	○	

2. XMEGA Xplainedキットに対する消費電力低減

第2章はデバイス(ATxmega256A3BU)を休止動作内へ移行させることによってXMEGA-A3BU Xplainedキットの消費電力を減らす手順を記述します。本章は様々な休止動作(アイドル、パワーダウン、パワーセーブ、スタンバイ、拡張スタンバイ)へ移行するコード例と一定の動作条件下の各々の休止動作で得られた消費電力値も提供します。

2.1. 動作条件と測定構成設定

動作条件

XMEGA-A3BU Xplainedキット上のATxmega256A3BUは3.3Vの供給電圧と内部2MHz RC発振器からの2MHzの周波数で走行します。この条件がデータシートの条件と異なるため、データシートで言及された値とこの応用記述で測定されたそれらの間で消費電力の数値に於いて違いが有り得ます。故にこれらの数値(やがて来る項で赤文字(訳注:原書では太文字)で協調される電流値)は各種休止動作への移行で消費電流が減ることを示すための参照基準だけです。

XMEGA-A3BU Xplainedキットが外部クロック元によってクロック駆動されるべき設備をもたないため、既定の内部2MHz RC発振器を使います。

2.2. ファームウェアでの開始に際して

この応用記述は低電力技法用のコード例を含むファームウェアzip一括と連携されます。本項はコード例の使用法を説明します。zip一括は3つのプロジェクト(Sleep_modes, Active_mode_with_ADC, Idle_mode_with_ADC)を持つ1つのAtmel Studio解決策から成ります。(2.3.～2.8.項の)様々な休止動作で消費電流を測定するには下で示されるように始動プロジェクト(Start up project)として"Sleep_modes"プロジェクトを選んでください。そして解決策をコンパイルしてAtmel-ICEを使ってXMEGA-A3BU Xplainedキット内にhexファイルを焼いて電力を測定してください。

消費電力を測定する間、以下の手順に従うべきです。

- hexファイル書き込み前に、プログラミング ウィンドウでBODACTとBODPDのヒューズが禁止されるのを確実にしてください。
- Atmel-ICEを使ったhexファイル書き込み後、XMEGA-A3BU Xplainedキットに接続されたAtmel-ICEヘッダを切断してください。
- 次にXMEGA-A3BU Xplainedキットの電源をOFF/ONし、そして消費電力値を測定するために図2-1.で示されるようにマルチメータを接続してください。マルチメータは電力測定中だけ接続されなければなりません。プログラミング(書き込み)中は切断されるべきです。

図2-1. 測定構成設定

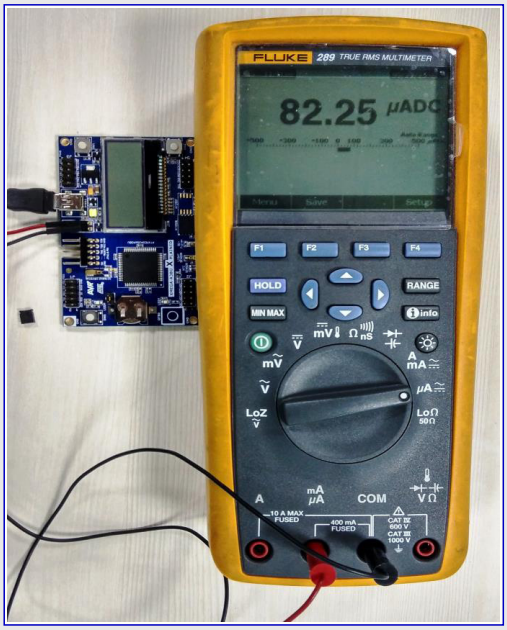
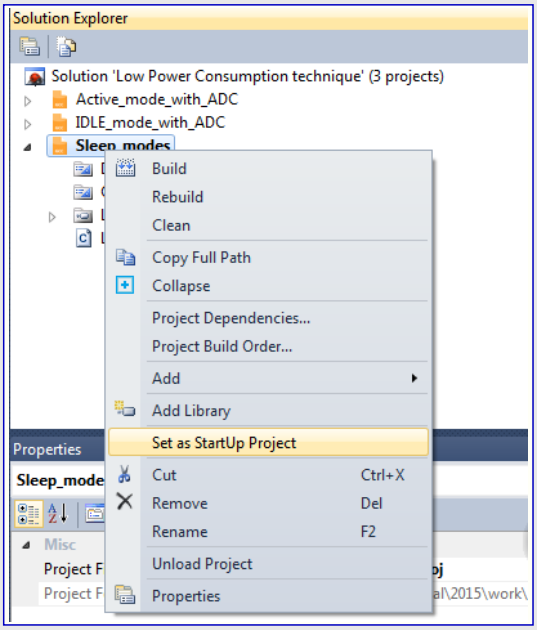


図2-2. 始動プロジェクトとして"Sleep_modes"選択



2.3. 活動動作での消費電流

活動動作では、換言すると、XMEGA-Z3BU XplainedｷｯﾄがUSB経由で給電されて何の休止動作も許可されていない時に、ｷｯﾄの消費電流は高くなります(概ね3mA)。この電流は「1. 消費電力に影響を及ぼす要素」章で言及される技法を適用することによって減らすことができます。消費電力は使われない周辺機能と単位部を禁止するためにPRR(電力削減ﾚｼﾞｽﾀ)を使うことによって削減されます。この例では何の周辺機能も使われず、故に全ての周辺機能に対してPRRﾋﾞｯﾄを設定(1)します。

以下の関数は電力削減ﾚｼﾞｽﾀへの書き込みによって全ての未使用周辺機能へのｸﾛｯｸを禁止します。

```
void disable_peripherals(void)
{
    PR.PRGEN = PR_USB_bm | PR_AES_bm | PR_EBI_bm | PR_RTC_bm | PR_EVSYS_bm | PR_DMA_bm; // USB,
AES,EBI,RTC,事象ｼｽﾃﾑ,DMA制御器へのｸﾛｯｸを停止
    PR.PRPA = PR_DAC_bm | PR_ADC_bm | PR_AC_bm; // DAC,ADC,ｱﾅﾛｸﾞ比較器へのｸﾛｯｸを停止
    PR.PRPB = PR_DAC_bm | PR_ADC_bm | PR_AC_bm;
    PR.PRPC = PR_TWI_bm | PR_USART1_bm | PR_USART0_bm | PR_SPI_bm | PR_HIRES_bm | PR_TC1_bm |
PR_TC0_bm; // 2線ｲﾝﾀｰﾌｪｰｽ,USART1,USART0,SPI,ﾀｲﾏ/ｶｳﾝﾀ0/1へのｸﾛｯｸを停止
    PR.PRPD = PR_TWI_bm | PR_USART1_bm | PR_USART0_bm | PR_SPI_bm | PR_HIRES_bm | PR_TC1_bm |
PR_TC0_bm;
    PR.PRPE = PR_TWI_bm | PR_USART1_bm | PR_USART0_bm | PR_SPI_bm | PR_HIRES_bm | PR_TC1_bm |
PR_TC0_bm;
    PR.PRPF = PR_TWI_bm | PR_USART1_bm | PR_USART0_bm | PR_SPI_bm | PR_HIRES_bm | PR_TC1_bm |
PR_TC0_bm;
}
```

上のｺｰﾄﾞ実行で全ての周辺機能へのｸﾛｯｸが停止され、故にXMEGA-A3BU Xplainedｷｯﾄの消費電流が減らされます(概ね2.5mA)。

更なる消費電力最小化のため、JTAGｲﾝﾀｰﾌｪｰｽを禁止してください。JTAGｲﾝﾀｰﾌｪｰｽはﾌﾟﾛｸﾞﾗﾐﾝｸﾞとﾃﾞﾊﾞｯｸﾞに使われますが、最終製品の動作中に何の機能も持ちません。OCDとJTAGｲﾝﾀｰﾌｪｰｽはﾋｭｰｽﾞとｿﾌﾄｳｪｱの両方で禁止することができます。

ｿﾌﾄｳｪｱでのJTAG禁止は、JTAGｲﾝﾀｰﾌｪｰｽがﾘｾｯﾄで再許可されるので、ﾃﾞﾊﾞｲｽが再ﾌﾟﾛｸﾞﾗﾐﾝｸﾞできるのを保証します。

代わりに、ﾌﾟﾛｸﾞﾗﾐﾝｸﾞとﾃﾞﾊﾞｯｸﾞにPDIｲﾝﾀｰﾌｪｽを使うことができます。この場合、JTAGｲﾝﾀｰﾌｪｰｽは全てに必要とされず、ﾋｭｰｽﾞによって禁止することができます。PDIｲﾝﾀｰﾌｪｰｽは全ての休止動作でも働きます。

以下のｺｰﾄﾞはｿﾌﾄｳｪｱからJTAGｲﾝﾀｰﾌｪｰｽを禁止するのに使われます。

```
void disable_JTAG(void) (注1)
{
    CCP          = CCP_IOREG_gc;
    MCU.MCUCR    = MCU_JTAGD_bm;
}
```

JTAGｲﾝﾀｰﾌｪｰｽの禁止で消費電力はかなり低減されます(概ね1.8mA)。

注: ｿﾌﾄｳｪｱからJTAGDﾋﾞｯﾄが設定(1)されると、通常動作でAtmel Studioによって更なるﾌﾟﾛｸﾞﾗﾐﾝｸﾞが防がれます。故に、ﾃﾞﾊﾞｲｽ ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ(Device programming)ｳｨﾝﾄﾞｳまたはﾌﾟﾛｸﾞﾗﾐﾝｸﾞ/ﾃﾞﾊﾞｯｸﾞ(Programming/Debugging)用のﾌﾟﾛｼﾞｪｸﾄ設定(Project setting)内のﾂｰﾙ(Tool)ﾀﾌﾞで"Use external reset"任意選択を許可してください。

注1: 最適化ﾚﾍﾞﾙが-O0よりも大きいことを確実にしてください。

2.4. ｱｲﾄﾞﾙ動作での消費電流

ｱｲﾄﾞﾙ動作では、全ての周辺機能が活性でAtmel AVR CPUｺｱと不揮発性ﾒﾓﾘ(ﾌﾗｯｼｭ ﾒﾓﾘとEEPROM)だけが停止されます。DMA1制御器と事象ｼｽﾃﾑはこの動作形態で未だ活性で、例えばADC結果ﾚｼﾞｽﾀからUSARTﾃﾞｰﾀ ﾚｼﾞｽﾀへﾃﾞｰﾀを転送するためのADCH変換完了起動DMAのように、DMAﾃﾞｰﾀ転送と事象起動を許し、一方でCPUは動きません。ﾃﾞﾊﾞｲｽは全ての割り込みで起動することができます。ﾃﾞﾊﾞｲｽは以下のｺｰﾄﾞを実行することによってｱｲﾄﾞﾙ(IDLE)休止動作へ移行します。

```
#include <avr/io.h>
#include <avr/sleep.h>

int main(void)
{
    set_sleep_mode(CURRENT_SLEEP_MODE);
    sleep_enable();
    sleep_cpu();
    while(1)
    {

    }
}
```

注: 追加の電力削減が必要とされる場合、先に説明されたdisable_peripherals()とdisable_JTAG()の関数が応用内に含められ、休止動作へ移行する前に呼ばれなければなりません。

例のｺｰﾄﾞに於いて、ﾃﾞﾊﾞｲｽをｱｲﾄﾞﾙ(IDLE)休止動作に置くには、下で示されるように活動(ACTIVE)動作選択行が注釈にされ、ｱｲﾄﾞﾙ(IDLE)動作選択行は注釈を外されなければなりません。

```
/* 使用者が望む休止動作を選んでください。既定では休止動作は選ばれません。*/

//#define CURRENT_SLEEP_MODE SLEEP_MODE_ACTIVE
#define CURRENT_SLEEP_MODE  SLEEP_MODE_IDLE
//#define CURRENT_SLEEP_MODE SLEEP_MODE_STANDBY
//#define CURRENT_SLEEP_MODE SLEEP_MODE_EXT_STANDBY
//#define CURRENT_SLEEP_MODE SLEEP_MODE_PWR_DOWN
//#define CURRENT_SLEEP_MODE SLEEP_MODE_PWR_SAVE
```

ｱｲﾄﾞﾙ(IDLE)動作ではAVR CPUｺｱと不揮発性ﾒﾓﾘが停止されるため、消費電流は活動動作で得られるよりもより低くなります(概ね1.4mA)。先で言及したように周辺機能とJTAGｲﾝﾀｰﾌｪｰｽを禁止することによってこの電力は更に低減することができます。

main()関数の内側でｱｲﾄﾞﾙ(IDLE)休止動作内へ移行する前にdisable_peripherals()関数を呼ぶことによって未使用周辺機能が禁止されると、消費電流は概ね1mAへの低下になります。

この消費電流はｱｲﾄﾞﾙ(IDLE)休止動作内へ移行する前にdisable_JTAG()関数を呼ぶことにより、ｿﾌﾄｳｪｱからJTAGｲﾝﾀｰﾌｪｰｽを禁止することによって更に概ね0.8mAに減らされます。

2.5. ﾊﾟﾜｰﾀﾞｳﾝ動作での消費電流

ﾊﾟﾜｰﾀﾞｳﾝ(POWER DOWN)は最も深い休止動作形態です。この動作ではﾃﾞﾊﾞｲｽの周辺機能の殆どが停止されます。周辺機能ｸﾛｯｸとRTCｸﾛｯｸの両方が禁止されるため、ﾃﾞﾊﾞｲｽはこの動作から自身で起動することができません。従ってこの動作形態は例えば非同期ﾋﾟﾝ割り込みやTWIｱﾄﾞﾚｽ一致割り込みでﾃﾞﾊﾞｲｽを起こすような外部信号を頼ります。けれども、電池代替単位部を持つXMEGAは、これらのﾃﾞﾊﾞｲｽのRTCが休止動作と無関係に動くために、32ﾋﾞｯﾄRTC単位部を使ってﾊﾟﾜｰﾀﾞｳﾝ休止動作からﾃﾞﾊﾞｲｽを起こすことができます。

ﾃﾞﾊﾞｲｽは以下のｺｰﾄﾞを実行することによってﾊﾟﾜｰﾀﾞｳﾝ(POWER DOWN)動作内へ移行します。

```
#include <avr/io.h>
#include <avr/sleep.h>

int main(void)
{
    set_sleep_mode(CURRENT_SLEEP_MODE);
    sleep_enable();
    sleep_cpu();
    while(1)
    {

    }
}
```

注: 追加の電力削減が必要とされる場合、先に説明されたdisable_peripherals()とdisable_JTAG()の関数が応用内に含められ、休止動作へ移行する前に呼ばれなければなりません。

この応用記述に於いて、ﾃﾞﾊﾞｲｽをﾊﾟﾜｰﾀﾞｳﾝ(POWER DOWN)動作に置くには、ﾊﾟﾜｰﾀﾞｳﾝ(POWER DOWN)動作選択行が注釈を外されなければなりません。

```
/* 使用者が望む休止動作を選んでください。既定では休止動作は選ばれません。*/

//#define CURRENT_SLEEP_MODE SLEEP_MODE_ACTIVE
//#define CURRENT_SLEEP_MODE SLEEP_MODE_IDLE
//#define CURRENT_SLEEP_MODE SLEEP_MODE_STANDBY
//#define CURRENT_SLEEP_MODE SLEEP_MODE_EXT_STANDBY
  #define CURRENT_SLEEP_MODE SLEEP_MODE_PWR_DOWN
//#define CURRENT_SLEEP_MODE SLEEP_MODE_PWR_SAVE
```

ﾊﾟﾜｰﾀﾞｳﾝ(POWER DOWN)動作ではXMEGA-A3BU Xplainedｷｯﾄの消費電流は概ね0.8mAです。これはJTAGｲﾝﾀｰﾌｪｰｽを禁止して未使用の周辺機能を禁止することによって概ね0.8μAへ著しく低減することができます。

2.6. パワーセーブ動作での消費電流

パワーセーブ(POWER SAVE)動作は1つの例外付きでパワーダウンと同じです。実時間計数器(RTC)が許可される場合、それは休止中に走行を維持し、デバイスはRTCの溢れまたは比較一致の割り込みで起きることができます。この休止動作でシステム クロック元が停止されるため、システム クロックは動作前に安定でなければならないので、起動はアイドル(IDLE)動作よりももう少しかかります。

デバイスは以下のコードを実行することによってパワーセーブ(POWER SAVE)動作内へ移行します。

```
#include <avr/io.h>
#include <avr/sleep.h>

int main(void)
{
    set_sleep_mode(CURRENT_SLEEP_MODE);
    sleep_enable();
    sleep_cpu()
    while(1)
    {

    }
}
```

注: 追加の電力削減が必要とされる場合、先に説明されたdisable_peripherals()とdisable_JTAG()の関数が応用内に含められ、休止動作へ移行する前に呼ばれなければなりません。

この応用記述に於いて、デバイスをパワーセーブ(POWER SAVE)動作に置くには、パワーセーブ(POWER SAVE)動作選択行が注釈を外されなければなりません。

```
/* 使用者が望む休止動作を選んでください。既定では休止動作は選ばれません。*/

//#define CURRENT_SLEEP_MODE SLEEP_MODE_ACTIVE
//#define CURRENT_SLEEP_MODE SLEEP_MODE_IDLE
//#define CURRENT_SLEEP_MODE SLEEP_MODE_STANDBY
//#define CURRENT_SLEEP_MODE SLEEP_MODE_EXT_STANDBY
//#define CURRENT_SLEEP_MODE SLEEP_MODE_PWR_DOWN
  #define CURRENT_SLEEP_MODE SLEEP_MODE_PWR_SAVE
```

パワーセーブ(POWER SAVE)動作ではXMEGA-A3BU Xplainedキットの消費電流は約0.8mAです。これはJTAGインターフェースを禁止して未使用の周辺機能を禁止することによって概ね0.8µAへ著しく低減することができます。

注: 電池代替単位部を持つXMEGAについてはパワーダウン(POWER DOWN)とパワーセーブ(POWER SAVE)の休止動作間に違いはありません。

2.7. スタンバイ動作での消費電流

スタンバイ(STANDBY)動作はCPU、周辺機能、RTCのクロックが停止される一方で許可されたシステム クロック元が動き続けることを除いてパワーダウン(POWER DOWN)と同じです。これは起き上がり時間を減らします。従ってこれは短い起動時間が必要とされる時に有用です。

デバイスは以下のコードを実行することによってスタンバイ(STANDBY)動作内へ移行します。

```
#include <avr/io.h>
#include <avr/sleep.h>

int main(void)
{
    set_sleep_mode(CURRENT_SLEEP_MODE);
    sleep_enable();
    sleep_cpu();
    while(1)
    {

    }
}
```

この応用記述に於いて、ﾃﾞﾊﾞｲｽをｽﾀﾝﾊﾞｲ(STANDBY)動作に置くには、ｽﾀﾝﾊﾞｲ(STANDBY)動作選択行が注釈を外されなければなりません。

```
/* 使用者が望む休止動作を選んでください。既定では休止動作は選ばれません。*/

//#define CURRENT_SLEEP_MODE SLEEP_MODE_ACTIVE
//#define CURRENT_SLEEP_MODE SLEEP_MODE_IDLE
  #define CURRENT_SLEEP_MODE SLEEP_MODE_STANDBY
//#define CURRENT_SLEEP_MODE SLEEP_MODE_EXT_STANDBY
//#define CURRENT_SLEEP_MODE SLEEP_MODE_PWR_DOWN
//#define CURRENT_SLEEP_MODE SLEEP_MODE_PWR_SAVE
```

ｽﾀﾝﾊﾞｲ(STANDBY)動作ではXMEGA-A3BU Xplainedｷｯﾄの消費電流は概ね0.9mAです。これはJTAGｲﾝﾀｰﾌｪｰｽを禁止して未使用の周辺機能を禁止することによって約82µAへ著しく低減することができます。

2.8. 拡張ｽﾀﾝﾊﾞｲ動作での消費電流

拡張ｽﾀﾝﾊﾞｲ(EXTENDED STANDBY)動作はCPUと周辺機能のｸﾛｯｸが停止される一方で許可されたｼｽﾃﾑ ｸﾛｯｸ元が動き続けることを除いてﾊﾟﾜｰｾｰﾌﾞ(POWER SAVE)と同じです。これは起き上がり時間を減らします。

ﾃﾞﾊﾞｲｽは以下のｺｰﾄﾞを実行することによって拡張ｽﾀﾝﾊﾞｲ(EXTENDED STANDBY)動作内へ移行します。

```
#include <avr/io.h>
#include <avr/sleep.h>

int main(void)
{
    set_sleep_mode(CURRENT_SLEEP_MODE);
    sleep_enable();
    sleep_cpu();
    while(1)
    {

    }
}
```

この応用記述に於いて、ﾃﾞﾊﾞｲｽを拡張ｽﾀﾝﾊﾞｲ(EXTENDED STANDBY)動作に置くには、拡張ｽﾀﾝﾊﾞｲ(EXTENDED STANDBY)動作選択行が注釈を外されなければなりません。

```
/* 使用者が望む休止動作を選んでください。既定では休止動作は選ばれません。*/

//#define CURRENT_SLEEP_MODE SLEEP_MODE_ACTIVE
//#define CURRENT_SLEEP_MODE SLEEP_MODE_IDLE
//#define CURRENT_SLEEP_MODE SLEEP_MODE_STANDBY
  #define CURRENT_SLEEP_MODE SLEEP_MODE_EXT_STANDBY
//#define CURRENT_SLEEP_MODE SLEEP_MODE_PWR_DOWN
//#define CURRENT_SLEEP_MODE SLEEP_MODE_PWR_SAVE
```

拡張ｽﾀﾝﾊﾞｲ(EXTENDED STANDBY)ではXMEGA-A3BU Xplainedｷｯﾄの消費電流は概ね0.9mAです。これはJTAGｲﾝﾀｰﾌｪｰｽを禁止して未使用の周辺機能を禁止することによって概ね82µAへ著しく低減することができます。

表2-1. 様々な休止動作形態に於けるXMEGA-A3BU Xplainedｷｯﾄの電力消費

休止動作形態	電力消費	JTAG禁止での電力消費 (注)
ｱｲﾄﾞﾙ(IDLE)	約2.1mA	約800µA
ﾊﾟﾜｰﾀﾞｳﾝ(POWER DOWN)	約800µA	約0.8µA
ﾊﾟﾜｰｾｰﾌﾞ(POWER SAVE)	約800µA	約0.8µA
ｽﾀﾝﾊﾞｲ(STANDBY)	約900µA	約82µA
拡張ｽﾀﾝﾄﾞﾊﾞｲ(EXTENDED STANDBY)	約900µA	約82µA

注: この応用記述での測定条件がﾃﾞｰﾀｼｰﾄの条件と異なるため、ﾃﾞｰﾀｼｰﾄで言及された値とこの応用記述で言及されたそれらの間で消費電力の数値に違いが有り得ます。

VCC=3.3V、ｸﾛｯｸ元=2MHz内部RC発振器。

3. A/D変換器電力測定

第3章は周辺機能(この応用記述では例としてA/D変換器(ADC)が選ばれます)が走行すると同時にﾃﾞﾊﾞｲｽの消費電力をどう減らせるかを説明します。動作条件と測定構成設定は2.1.項下で言及したものと同じです。ADCは以下の基本構成設定で許可されます。

ADCは(右揃えされた)12ﾋﾞｯﾄ分解能を持つ符号なしｼﾝｸﾞﾙ ｴﾝﾄﾞ動作と高電流制限で構成設定されます。内部2MHzｸﾛｯｸ元が主ｸﾛｯｸ前置分周器で2分周され、更にADCの前置分周器がｸﾛｯｸを4分周します。基準電圧は内部VCC/1.6として選択されます。ﾁｬﾈﾙ0変換完了割り込みが高優先度で許可されます。

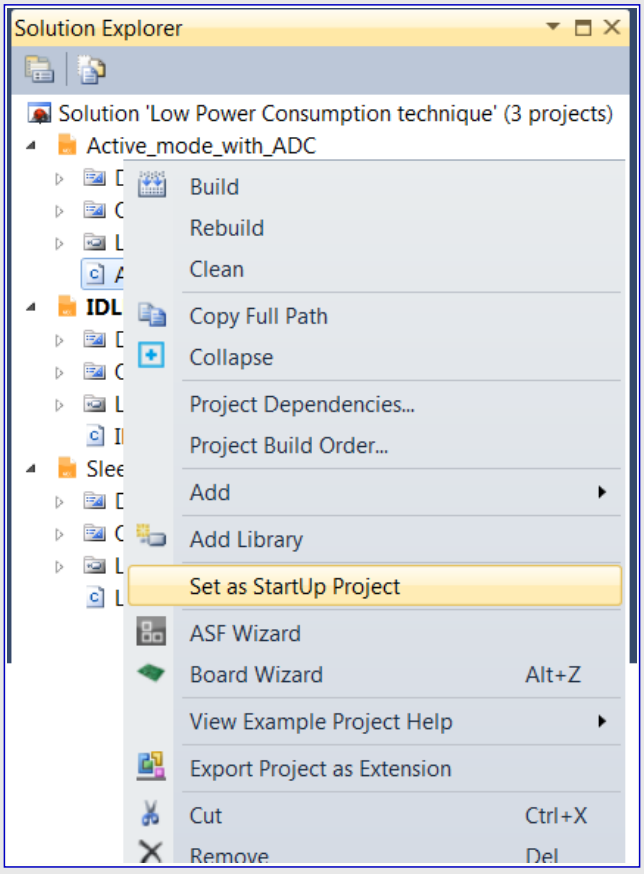
```
void ADC_initialize(void)
{
    CCP = 0xD8;
    CLK.PSCTRL = CLK_PSADIV_2_gc;                      // 前置分周器はｸﾛｯｸを2分周
    ADCA.CTRLB = ADC_CURRLIMIT_HIGH_gc;                // 12ﾋﾞｯﾄADC分解能(右揃え)と高電流制限
    ADCA.REFCTRL = ADC_REFSEL_INTVCC_gc;               // VCC/1.6に設定したADC基準電圧
    ADCA.PRESCALER = ADC_PRESCALER_DIV4_gc;            // ADCｸﾛｯｸ元はFosc/4
    ADCA.CH0.CTRL = ADC_CH_INPUTMODE_SINGLEENDED_gc;   // ｼﾝｸﾞﾙ ｴﾝﾄﾞとしてADCﾁｬﾈﾙ選択
    ADCA.CH0.MUXCTRL = ADC_CH_MUXPOS_PIN0_gc;          // CH0をPB1に配線
    // 変換完了割り込み許可とそれを高優先割り込みとして定義
    ADCA.CH0.INTCTRL = ADC_CH_INTMODE_COMPLETE_gc | ADC_CH_INTLVL_HI_gc;
    ADCA.CTRLA = ADC_ENABLE_bm;                        // ADC許可
}
```

3.1. ADC走行中の活動動作での消費電流

3.1.項は活動動作下でのADC走行中の最小電流を達成するためのｺｰﾄﾞ例を提供します。2.3.項下で言及したように、JTAGｲﾝﾀｰﾌｪｰｽと未使用の周辺機能の禁止が消費電力を減らします。故に、消費電力を減らすように(ADCを除く)未使用周辺機能とJTAGｲﾝﾀｰﾌｪｰｽを禁止してください。

この応用記述ではADC走行用のｺｰﾄﾞ例がActive_mode_with_ADCﾌﾟﾛｼﾞｪｸﾄで利用でき、故にこのﾌﾟﾛｼﾞｪｸﾄが始動ﾌﾟﾛｼﾞｪｸﾄ(Start up project)として設定されなければならず、その後にその解決策がｺﾝﾊﾟｲﾙされるべきです。

図3-1. 始動ﾌﾟﾛｼﾞｪｸﾄとして"Active_mode_with_adc"選択



```
#include <avr/io.h>
#include <avr/sleep.h>
#include <avr/interrupt.h>

uint16_t adc_results =0;

void disable_peripherals(void)
{
    // USB、AES、EBI、RTC、事象ｼｽﾃﾑ、DMA制御器へのｸﾛｯｸ停止
    PR.PRGEN = PR_USB_bm | PR_AES_bm | PR_EBI_bm | PR_RTC_bm | PR_EVSYS_bm | PR_DMA_bm;
    PR.PRPA = PR_DAC_bm | PR_AC_bm;                    // DACとｱﾅﾛｸﾞ比較器へのｸﾛｯｸ停止
    PR.PRPB = PR_DAC_bm | PR_AC_bm;
    // 2線ｲﾝﾀｰﾌｪｰｽ、USART1、USART0、SPI、ﾀｲﾏ/ｶｳﾝﾀ0/1へのｸﾛｯｸ停止
    PR.PRPC = PR_TWI_bm | PR_USART1_bm | PR_USART0_bm | PR_SPI_bm | PR_HIRES_bm | PR_TC1_bm | PR_TC0_bm;
    PR.PRPD = PR_TWI_bm | PR_USART1_bm | PR_USART0_bm | PR_SPI_bm | PR_HIRES_bm | PR_TC1_bm | PR_TC0_bm;
    PR.PRPE = PR_TWI_bm | PR_USART1_bm | PR_USART0_bm | PR_SPI_bm | PR_HIRES_bm | PR_TC1_bm | PR_TC0_bm;
    PR.PRPF = PR_TWI_bm | PR_USART1_bm | PR_USART0_bm | PR_SPI_bm | PR_HIRES_bm | PR_TC1_bm | PR_TC0_bm;
}

void disable_JTAG (void)
{
    CCP = CCP_IOREG_gc;
    MCU.MCUCR = MCU_JTAGD_bm;                          // JTAG禁止
}

void ADC_initialize(void)
{
    CCP = 0xD8;
    CLK.PSCTRL = CLK_PSADIV_2_gc;                      // 前置分周器はｸﾛｯｸ元を2分周
    ADCA.CTRLB = ADC_CURRLIMIT_HIGH_gc;                // 12ﾋﾞｯﾄADC分解能(右揃え)と高電流制限
    ADCA.REFCTRL = ADC_REFSEL_INTVCC_gc;               // ADC基準電圧をVCC/1.6設定
    ADCA.PRESCALER = ADC_PRESCALER_DIV4_gc;            // ADCｸﾛｯｸ元はFosc/4
    ADCA.CH0.CTRL = ADC_CH_INPUTMODE_SINGLEENDED_gc;   // ｼﾝｸﾞﾙ ｴﾝﾄﾞとしてADCﾁｬﾈﾙ選択
    ADCA.CH0.MUXCTRL = ADC_CH_MUXPOS_PIN0_gc;          // CH0をPB1に配線
    // 変換完了割り込み許可とそれを高優先割り込みとして定義
    ADCA.CH0.INTCTRL = ADC_CH_INTMODE_COMPLETE_gc | ADC_CH_INTLVL_HI_gc;
    ADCA.CTRLA = ADC_ENABLE_bm;  // ADC許可
}

int main(void)
{
    disable_peripherals();
    disable_JTAG();
    ADC_initialize();                                  // ADCﾁｬﾈﾙ0に対して変換開始
    ADCA.CH0.CTRL = ADC_CH_START_bm;
    PMIC.CTRL = PMIC_HILVLEN_bm; sei();
    while(1)
    {

    }
}

ISR(ADCA_CH0_vect)
{
    adc_results = ADCA_CH0RES;                         // ADC結果
    ADCA.CH0.INTFLAGS = ADC_CH_CHIF_bm;
    ADCA.CH0.CTRL = ADC_CH_START_bm;                   // ADCﾁｬﾈﾙ0に対して変換開始
}
```

上のﾌﾟﾛｼﾞｪｸﾄ実行での消費電力は概ね3.5mAです。ADC変換結果は全域定義されたadc_results変数に格納されます。1.16.項下で言及したように、休止動作を許可することによって電力は更に減らすことができます。

3.2. ADC走行中のｱｲﾄﾞﾙ動作での消費電流

許可されたADCを持つｱｲﾄﾞﾙ(IDLE)動作下のXMEGA-A3BU Xplainedｷｯﾄによる消費電力は次のように与えられます(ADCはｱｲﾄﾞﾙ(IDLE)動作でだけ使うことができるため、ｱｲﾄﾞﾙ(IDLE)動作だけが検討されます)。3.2.項はｱｲﾄﾞﾙ休止動作下でのADC走行中の最小電流を達成するためのｺｰﾄﾞ例を提供します。このﾌﾟﾛｼﾞｪｸﾄでは休止動作が許可され、電力削減技法も適用されます。

次に添付した解決策と共に利用可能な'IDLE_mode_with_ADC'ﾌﾟﾛｼﾞｪｸﾄが始動ﾌﾟﾛｼﾞｪｸﾄ(Start up project)として設定され、その後にその解決策がｺﾝﾊﾟｲﾙされるべきです。

```
#include <avr/io.h>
#include <avr/sleep.h>
#include <avr/interrupt.h>

uint16_t adc_results =0;

void disable_peripherals(void)
{
    // USB、AES、EBI、RTC、事象ｼｽﾃﾑ、DMA制御器へのｸﾛｯｸ停止
    PR.PRGEN = PR_USB_bm | PR_AES_bm | PR_EBI_bm | PR_RTC_bm | PR_EVSYS_bm | PR_DMA_bm;
    PR.PRPA = PR_DAC_bm | PR_AC_bm;                    // DACとｱﾅﾛｸﾞ比較器へのｸﾛｯｸ停止
    PR.PRPB = PR_DAC_bm | PR_AC_bm;
    // 2線ｲﾝﾀｰﾌｪｰｽ、USART1、USART0、SPI、ﾀｲﾏ/ｶｳﾝﾀ0/1へのｸﾛｯｸ停止
    PR.PRPC = PR_TWI_bm | PR_USART1_bm | PR_USART0_bm | PR_SPI_bm | PR_HIRES_bm | PR_TC1_bm | PR_TC0_bm;
    PR.PRPD = PR_TWI_bm | PR_USART1_bm | PR_USART0_bm | PR_SPI_bm | PR_HIRES_bm | PR_TC1_bm | PR_TC0_bm;
    PR.PRPE = PR_TWI_bm | PR_USART1_bm | PR_USART0_bm | PR_SPI_bm | PR_HIRES_bm | PR_TC1_bm | PR_TC0_bm;
    PR.PRPF = PR_TWI_bm | PR_USART1_bm | PR_USART0_bm | PR_SPI_bm | PR_HIRES_bm | PR_TC1_bm | PR_TC0_bm;
}

void disable_JTAG (void)
{
    CCP = CCP_IOREG_gc;
    MCU.MCUCR = MCU_JTAGD_bm;                          // JTAG禁止
}

void ADC_initialize(void)
{
    CCP = 0xD8;
    CLK.PSCTRL = CLK_PSADIV_2_gc;                      // 前置分周器はｸﾛｯｸ元を2分周
    ADCA.CTRLB = ADC_CURRLIMIT_HIGH_gc;                // 12ﾋﾞｯﾄADC分解能(右揃え)と高電流制限
    ADCA.REFCTRL = ADC_REFSEL_INTVCC_gc;               // ADC基準電圧をVCC/1.6設定
    ADCA.PRESCALER = ADC_PRESCALER_DIV4_gc;            // ADCｸﾛｯｸ元はFosc/4
    ADCA.CH0.CTRL = ADC_CH_INPUTMODE_SINGLEENDED_gc;   // ｼﾝｸﾞﾙ ｴﾝﾄﾞとしてADCﾁｬﾈﾙ選択
    ADCA.CH0.MUXCTRL = ADC_CH_MUXPOS_PIN0_gc;          // CH0をPB1に配線
    // 変換完了割り込み許可とそれを高優先割り込みとして定義
    ADCA.CH0.INTCTRL = ADC_CH_INTMODE_COMPLETE_gc | ADC_CH_INTLVL_HI_gc;
    ADCA.CTRLA = ADC_ENABLE_bm;                        // ADC許可
}

int main(void)
{
    disable_peripherals();
    disable_JTAG();
    ADC_initialize();
    set_sleep_mode(SLEEP_MODE_IDLE);                   // ｱｲﾄﾞﾙ動作選択
    sleep_enable();
    ADCA.CH0.CTRL = ADC_CH_START_bm;                   // ADCﾁｬﾈﾙ0に対して変換開始
    PMIC.CTRL = PMIC_HILVLEN_bm;
    sei();
    while(1)
    {
        sleep_cpu();                                   // 休止許可
    }
}

    ISR(ADCA_CH0_vect)
{
    adc_results = ADCA_CH0RES;                         // ADC結果
    ADCA.CH0.INTFLAGS = ADC_CH_CHIF_bm;
    ADCA.CH0.CTRL = ADC_CH_START_bm;                   // ADCﾁｬﾈﾙ0に対して変換開始
}
```

先のｺｰﾄﾞの事項に於けるXMEGA-A3BUｷｯﾄの電力消費は概ね約2.8mAです。活動動作下で(ADC_initialize()断片ｺｰﾄﾞで説明されるような基本条件で)ADCが走行する時の消費電流は概ね約3.9mAで、これは様々な電力消費技術を適用することによって2.8mAへ相当減らされます。

4. 参考文献

[1] Atmel AVR XMEGA AU手引書

[2] ATxmegaA3BU完全版ﾃﾞｰﾀｼｰﾄ

[3] Atmel AVR1935: XMEGA A3BU Xplained開始の手引き

[4] Atmel AVR1923: XMEGA A3BU Xplainedﾊｰﾄﾞｳｪｱ使用者の手引き

[5] Atmel AVR1934: XMEGA A3BU Xplainedｿﾌﾄｳｪｱ使用者の手引き

[6] AVR1010: XMEGAﾃﾞﾊﾞｲｽの電力消費最小化

5. 改訂履歴

文書改訂	日付	注釈
42456A	2015年5月	初版文書公開

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA TEL:(+1)(408) 441-0311 FAX: (+1)(408) 436-4200 | www.atmel.com

© 2015 Atmel Corporation. / 改訂:Atmel-42456A-Low-Power-Consumption-Techniques-for-XMEGA-XPLAINED-Kits_ApplicationNote_052015

Atmel®、Atmelﾛｺﾞとそれらの組み合わせ、AVR®、Enabling Unlimited Possibilities®、XMEGA®とその他は米国と他の国に於けるAtmel Corporationの登録商標または商標です。他の用語と製品名は一般的に他の商標です。

お断り: 本資料内の情報はAtmel製品と関連して提供されています。本資料またはAtmel製品の販売と関連して承諾される何れの知的所有権も禁反言あるいはその逆によって明示的または暗示的に承諾されるものではありません。**Atmelのｳｪﾌﾞｻｲﾄに位置する販売の条件とAtmelの定義での詳しい説明を除いて、商品性、特定目的に関する適合性、または適法性の暗黙保証に制限せず、Atmelはそれらを含むその製品に関連する暗示的、明示的または法令による如何なる保証も否認し、何ら責任がないと認識します。たとえAtmelがそのような損害賠償の可能性を進言されたとしても、本資料を使用できない、または使用以外で発生する(情報の損失、事業中断、または利益と損失に関する制限なしの損害賠償を含み)直接、間接、必然、偶然、特別、または付随して起こる如何なる損害賠償に対しても決してAtmelに責任がないでしょう。**Atmelは本資料の内容の正確さまたは完全性に関して断言または保証を行わず、予告なしでいつでも製品内容と仕様の変更を行う権利を保留します。Atmelはここに含まれた情報を更新することに対してどんな公約も行いません。特に別の方法で提供されなければ、Atmel製品は車載応用に対して適当ではなく、使用されるべきではありません。Atmel製品は延命または生命維持を意図した応用での部品としての使用に対して意図、認定、または保証されません。

安全重視、軍用、車載応用のお断り: Atmel製品はAtmelが提供する特別に書かれた承諾を除き、そのような製品の機能不全が著しく人に危害を加えたり死に至らしめることがかなり予期されるどんな応用(”安全重視応用”)に対しても設計されず、またそれらとの接続にも使用されません。安全重視応用は限定なしで、生命維持装置とｼｽﾃﾑ、核施設と武器ｼｽﾃﾑの操作用の装置やｼｽﾃﾑを含みます。Atmelによって軍用等級として特に明確に示される以外、Atmel製品は軍用や航空宇宙の応用や環境のために設計も意図もされていません。Atmelによって車載等級として特に明確に示される以外、Atmel製品は車載応用での使用のために設計も意図もされていません。

© *HERO* 2021.

本応用記述はAtmelのAT11487応用記述(Rev.42456A-05/2015)の翻訳日本語版です。日本語では不自然となる重複する形容表現は省略されている場合があります。日本語では難解となる表現は大幅に意訳されている部分もあります。必要に応じて一部加筆されています。頁割の変更により、原本より頁数が少なくなっています。

必要と思われる部分には( )内に英語表記や略称などを残す形で表記しています。

青字の部分はﾘﾝｸとなっています。一般的に赤字の0,1は論理0,1を表します。その他の赤字は重要な部分を表します。