AN2474

Atmelソフトウェア枠組み(ASF)プロジェクトのASFv3.3からASFv4への移植

序説

この応用記述はAtmel Start ASFv3ﾌﾟﾛｼﾞｪｸﾄからASFv4ﾌﾟﾛｼﾞｪｸﾄへの移植の検討導入を提供します。

ASFv4のｺｰﾄﾞ密度に対する最適化と追加された柔軟性のため、2つの枠組みは相互排他です。そしてASF v3.3ﾌﾟﾛｼﾞｪｸﾄがASF v4.0によって構築することができないことを意味します。これはﾌﾟﾛｼﾞｪｸﾄ移植に”根本的”手法が必要です。

ASFv3

ASF 3基本構造は主な目標としてｺｰﾄﾞ量、性能、低電力の最適化で開発されました。周辺機能ﾄﾞﾗｲﾊﾞとﾐﾄﾞﾙｳｪｱはASFが許されたﾌﾟﾛｼﾞｪｸﾄ使用時の設計段階減少を提供するために開発されました。

ASFの意図はお客様の設計時間を減らすためにAtmelの熟練者によって開発された実績のあるﾄﾞﾗｲﾊﾞとｺｰﾄﾞ単位部の豊富な一式を提供することです。これはﾊｰﾄﾞｳｪｱと高価値ﾐﾄﾞﾙｳｪｱに対する抽象化を提供して、ﾏｲｸﾛ ｺﾝﾄﾛｰﾗの使い方を簡単化します。

ASFはｿｰｽ ｺｰﾄﾞ単位部とこれらの使い方を実演する応用から成ります。

- **ﾄﾞﾗｲﾊﾞ**は周辺機能やﾃﾞﾊﾞｲｽ特有機能をｱｸｾｽするための低位ﾚｼﾞｽﾀ ｲﾝﾀｰﾌｪｰｽ関数を提供するdriver.cとdriver.hから成ります。ｻｰﾋﾞｽと部品はこのﾄﾞﾗｲﾊﾞをｲﾝﾀｰﾌｪｰｽします。
- **ｻｰﾋﾞｽ**はUSBｸﾗｽ、FATﾌｧｲﾙｼｽﾃﾑ、基本構造最適化したDSPﾗｲﾌﾞﾗﾘ、図画ﾗｲﾌﾞﾗﾘなどのようなもっと応用指向ｿﾌﾄｳｪｱを提供する単位部型です。
- **部品**はﾒﾓﾘ(他えば、Atmel DataFlash®、SDRAM、SRAM、NANDﾌﾗｯｼｭ)、表示器、感知器、無線などのような外部ﾊｰﾄﾞｳｪｱ部品をｱｸｾｽするためのｿﾌﾄｳｪｱ ﾄﾞﾗｲﾊﾞを提供する単位部型です。
- **基板**はAtmelの開発ｷｯﾄの各入出力ﾋﾟﾝに対する全てのﾃﾞｼﾞﾀﾙとｱﾅﾛｸﾞの周辺機能の割り当てを含みます。

ASFv4

Atmel Startではﾄﾞﾗｲﾊﾞとｿﾌﾄｳｪｱ階層がAtmelｿﾌﾄｳｪｱ枠組み(Atmel Software Framework)の次のｿﾌﾄｳｪｱ開発の一部(ASFv4)として提供されます。この版は0から構築され、旧ASF版の使用者と貢献者によって報告された問題を解決するためとAtmel Startｳｪﾌﾞ使用者ｲﾝﾀｰﾌｪｰｽでのより良い統合のためにこの枠組み全体の完全な再設計と実装です。

それにもかかわらず、ASFを経験した使用者に対してASFv4の親近感を保ち、新規使用者に対して未だ開始が容易なことが目標でした。ASFv4でのいくつかの変更はこの版に対する必要条件に合致するために必要で、最も重要な変更はAtmel Start使用者の手引きの「ASFv4対ASFv3評価基準」で一覧にされます。

ASFv4はAtmel Startに密接に統合され、そしてそれはASFv4ｺｰﾄﾞが以前よりもよりもっと使用者の仕様に誂えることができることを意味します。例えば、許可/禁止ｺｰﾄﾞ部に対してC前処理条件式を使う代わりに、ﾌﾟﾛｼﾞｪｸﾄ ｿｰｽから禁止ｺｰﾄﾞ部を完全に取り去ることができ、ｺｰﾄﾞを読むことをより綺麗でより容易にします。Atmel Startへの統合はｿﾌﾄｳｪｱ構成設定がよりもっと使用者に友好的な環境で行われ、ﾃﾞﾊﾞｲｽで設定された構成設定情報だけが生の周辺機能ﾚｼﾞｽﾀ内容で、それはﾌｧｰﾑｳｪｱ ｲﾒｰｼﾞをよりもっと簡潔にします。

我々が取り掛かった1つの重要な問題はASFに基づくｺｰﾄﾞのﾒﾓﾘ量と性能です。ASFv3に基づくｺｰﾄﾞを走らせるためのﾌﾗｯｼｭ ﾒﾓﾘ必要条件は多くの使用者によって多すぎると思われていました。これはｺｰﾄﾞ生成を使って周辺機能が初期化される方法を変更することによって処理されました。報告されている性能の問題は代表的に大きな割り込み遅延/遅いｺｰﾄﾞ実行で、より小さくて複雑でない割り込み処理部を作ることによって解決されました。

移植

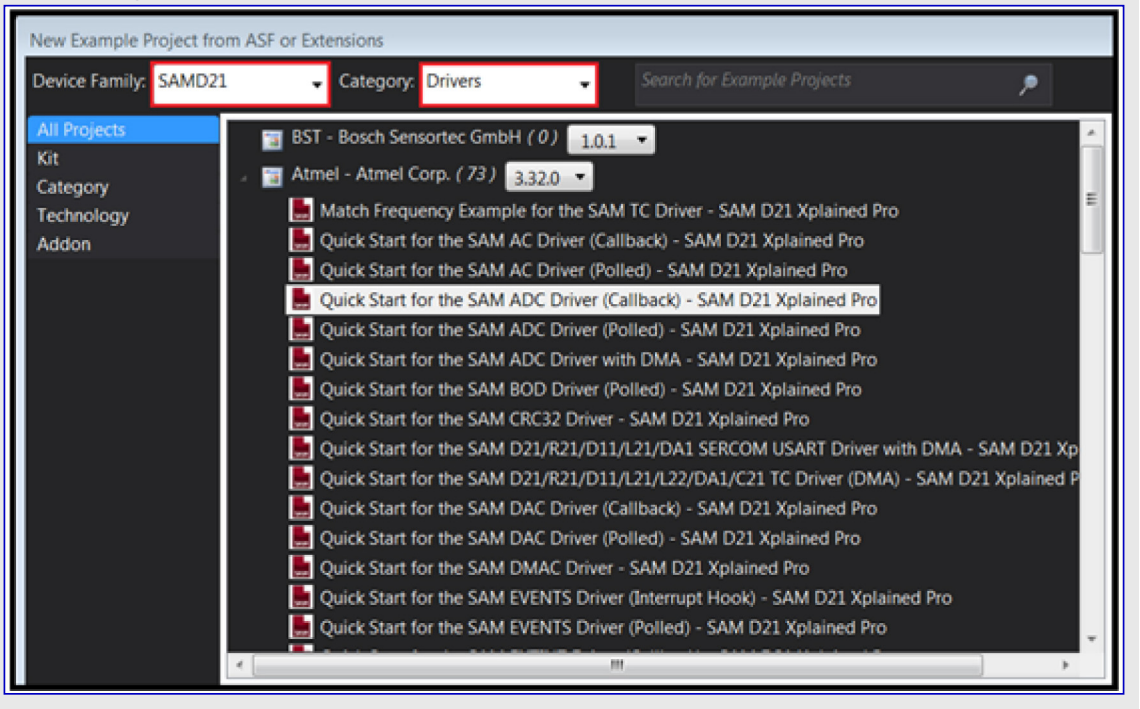
あいにく、ASFv3ﾌﾟﾛｼﾞｪｸﾄをASFv4ﾌﾟﾛｼﾞｪｸﾄへ直接移植する方法がありません。新しい基本設計とAPIで、移植の方法はｵﾝﾗｲﾝのAtmel Startﾂｰﾙ(start.atmel.com)で0から(元の)ASFv3ﾌﾟﾛｼﾞｪｸﾄを再構築するだけです。同時にこれは現在手応えのある量を示すかもしれず、ASFv3ﾌﾟﾛｼﾞｪｸﾄを用いるﾃﾞﾊﾞｲｽ間移植よりもASFv4ﾌﾟﾛｼﾞｪｸﾄを用いるﾃﾞﾊﾞｲｽ間の将来の移植はよりもっと容易でしょう。

例ﾌﾟﾛｼﾞｪｸﾄ開始

移植処理を通して段階実行するには、例ﾌﾟﾛｼﾞｪｸﾄがASF v3.3例ﾌﾟﾛｼﾞｪｸﾄ一覧から使われます。使われる例ﾌﾟﾛｼﾞｪｸﾄはSAMD21 Xplained Pro評価基板で使われるように選ばれます。

1. Atmel StudioでFile(ﾌｧｲﾙ)⇒New(新規)⇒Example Project(例ﾌﾟﾛｼﾞｪｸﾄ)を選ぶことでASFv3ﾌﾟﾛｼﾞｪｸﾄを読み込んでください。
2. 図1で示されるように、Device Family(ﾃﾞﾊﾞｲｽ系統)ﾒﾆｭｰでSAMD21を選んでください。

図1. ASFv3例ﾌﾟﾛｼﾞｪｸﾄのｲﾝﾎﾟｰﾄ

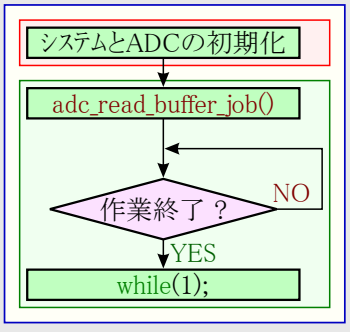


3. 全てのAtmelﾌﾟﾛｼﾞｪｸﾄを表示するためにAll Projects(全ﾌﾟﾛｼﾞｪｸﾄ)をｸﾘｯｸし、その後にQuick Start for the SAM ADC Driver (Callb ack) – SAM D21 Xplained Pro(SAM ADCﾄﾞﾗｲﾊﾞ(呼び戻し)用即時開始 – SAMD21 Xplained Pro)を選んでください。
4. 既定のAtmel Sudio作業空間でﾌﾟﾛｼﾞｪｸﾄを作成するためにOKをｸﾘｯｸしてください。
5. 例ﾌﾟﾛｼﾞｪｸﾄをｺﾝﾊﾟｲﾙして走らせてください。
6. 応用説明

   例のADCﾌﾟﾛｸﾞﾗﾑ流れ図が図2で示されます。例応用は試料の一式を収集するためにADCを開始し、その後に一度ﾃﾞｰﾀ集合が完了するのを永遠に待ちます。

   応用は非常に単純ですが、移植に対して2つの段階に分断される必要があります。赤で外廓を描かれた段階1は後でAtmel Startで使うためにASFv3でｼｽﾃﾑとADC初期化から必要とされる情報を抽出します。緑で外廓を描かれた段階2は応用で必要とされるAPIの機能的な確認で、Atmel StartﾌﾟﾛｼﾞｪｸﾄがAtmel Studio内にｲﾝﾎﾟｰﾄされた後で完了されます。

図2. 例ADCﾌﾟﾛｸﾞﾗﾑ流れ図



7. 応用構成部品

   応用の機能はｼｽﾃﾑ(ｸﾛｯｸなど)だけなくADC周辺機能の理解も必要です。図2の赤部分はどのﾌﾟﾛｼﾞｪｸﾄに対しても必要とされるｼｽﾃﾑ ﾄﾞﾗｲﾊﾞを含みます。いくつかの構成設定はｼｽﾃﾑが周辺機能に対して必要なｸﾛｯｸと構成設定を提供することができることを保証するために必要とされます。ADCもこの部分に含まれ、応用に対して必要とされる構成設定を転送するための評価が必要です。

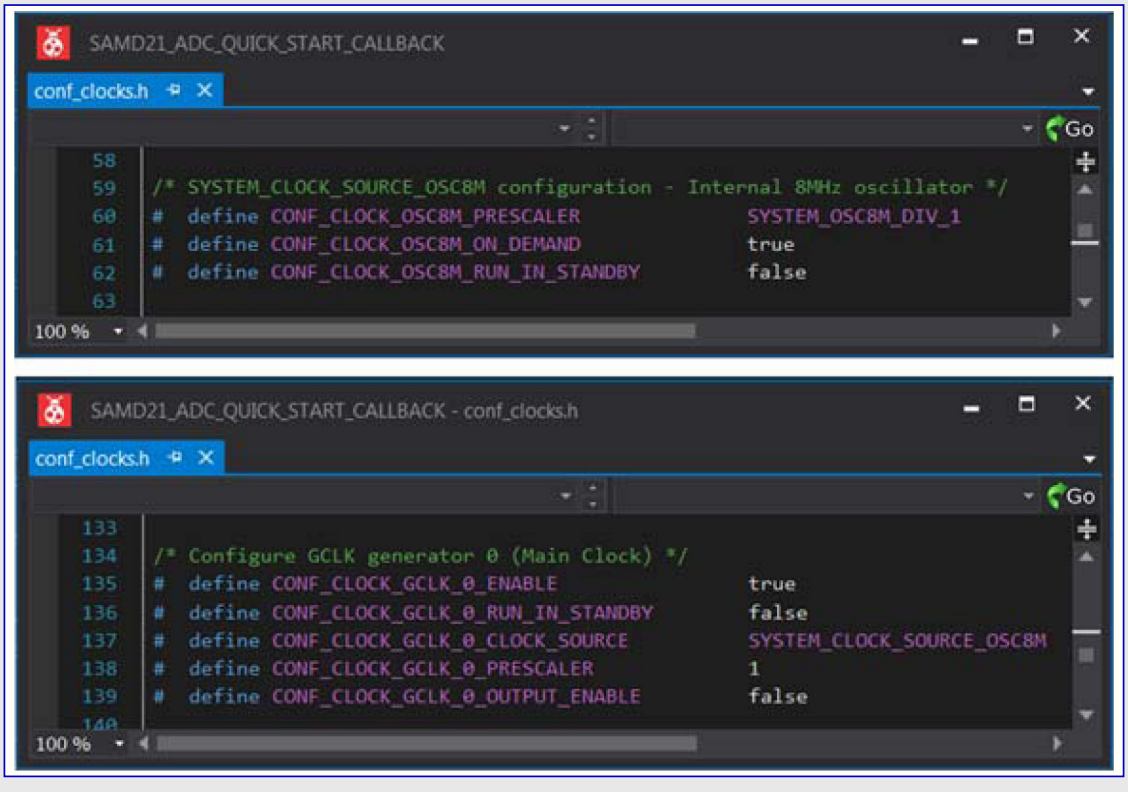
ｼｽﾃﾑとADCのASFv3構成設定抽出

1. ｸﾛｯｸ構成設定記録

   ASFv3用ｼｽﾃﾑ ｸﾛｯｸ設定は例ﾌﾟﾛｼﾞｪｸﾄのconf_clocks.hﾌｧｲﾙに置くことができます。ここから、主ｸﾛｯｸ(GCLK_0)が内部8MHz発振器によって供給されるのを見ることができます。

   図3.はGCLK_0が前置分周されれず、8MHzでSAMD21ｺｱを走らせることを示します。これらの設定はADCﾌﾟﾛｼﾞｪｸﾄのｸﾛｯｸ樹形を構成設定するためにAtmel Studioで使われます。このﾌﾟﾛｼﾞｪｸﾄのADC周辺機能に対して、ｸﾛｯｸ元としてGCLK_0が使われます。これらの値は走行時に一般ｸﾛｯｸ発生器(GCLK)とｼｽﾃﾑ制御(SYSCTRL)下でI/Oｳｨﾝﾄﾞｳでも見つけることができます。

図3. システム クロック構成設定



2. 周辺機能構成設定記録

ASFv3の周辺機能単位部は応用に対して周辺機能を初期化するのに構成設定構造体を使います。ASFv3は初期化のために主辺機能に書かれる既定値を含みます。これは周辺機能構成設定構造体が走行時に移入されることに注意することが重要です。ASFv4は違う手法を取り、前処理部ﾏｸﾛで全ての応用構成設定を処理します。

この例応用ではconfigure_adc()関数内でADCを初期化するのにadc_config構造体が使われます。ﾌﾟﾛｼﾞｪｸﾄ実行中に監視(Watch)ｳｨﾝﾄﾞｳを用い、初期化ﾊﾟﾗﾒｰﾀは**図4**で捕獲されます0。構成設定値の命名規則を維持するためにASFv4開発で努力が払われました。そんな訳で、**図4**の中央列内の値はAtmel StartでADCを構成設定するのに使われます。

図4. ASFv3 ADC構成設定ﾊﾟﾗﾒｰﾀ

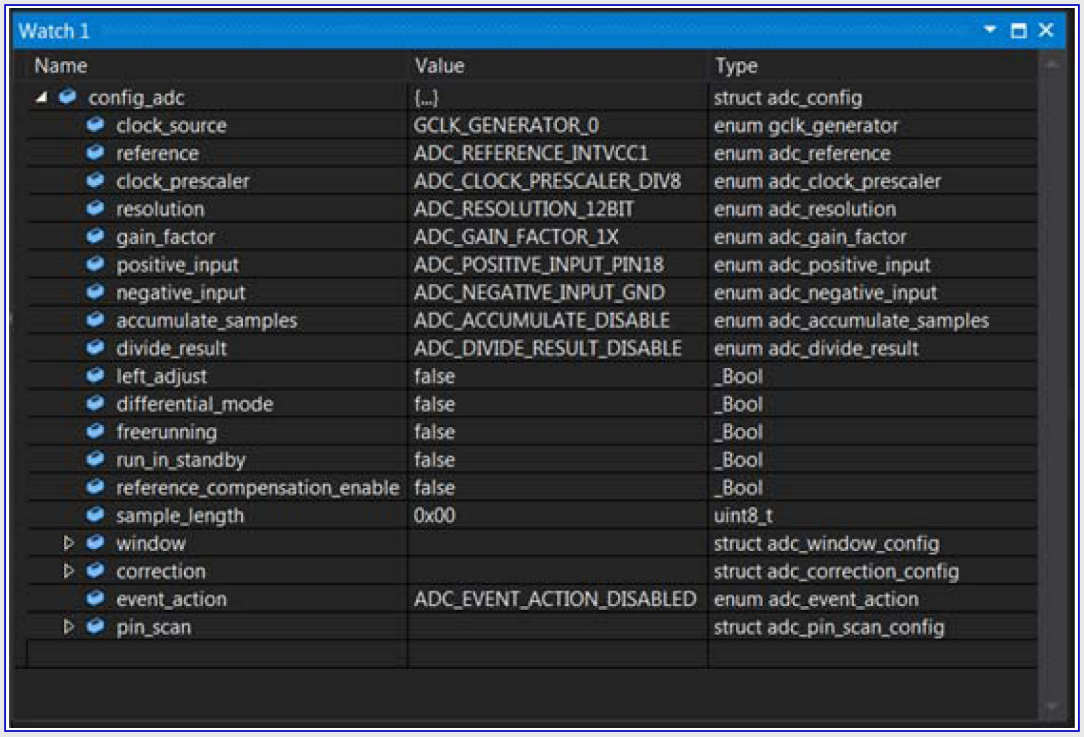
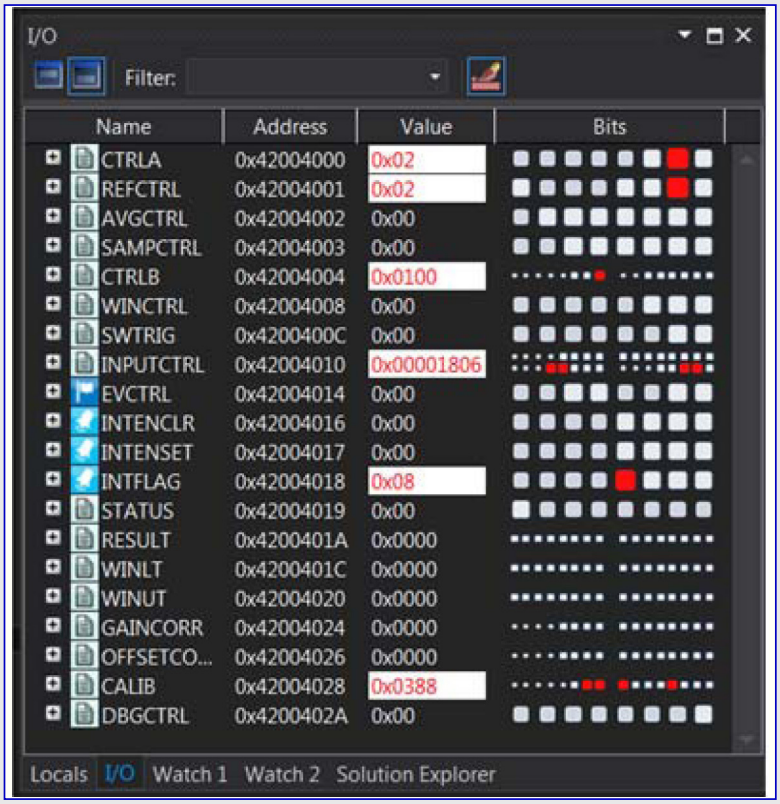


図5のADCレジスタはASFv3で初期化後のレジスタ値を表示し、Atmel StartでADC初期化を確認するのに使われます。

3. ピン構成設定記録

   選んだ例プロジェクトは応用に対して1つのADCチャネルを使うだけです。この場合、ピン割り当ては見つけるのが容易で、前に参照されたconfig_adc構造体またはconfigure_adc()関数に配置されます。Xplained Proのような Atmelハードウェアを使うASFv3例プロジェクトはハードウェアがsystem_board_init()関数で初期化される場所でこの情報を持つかもしれません。

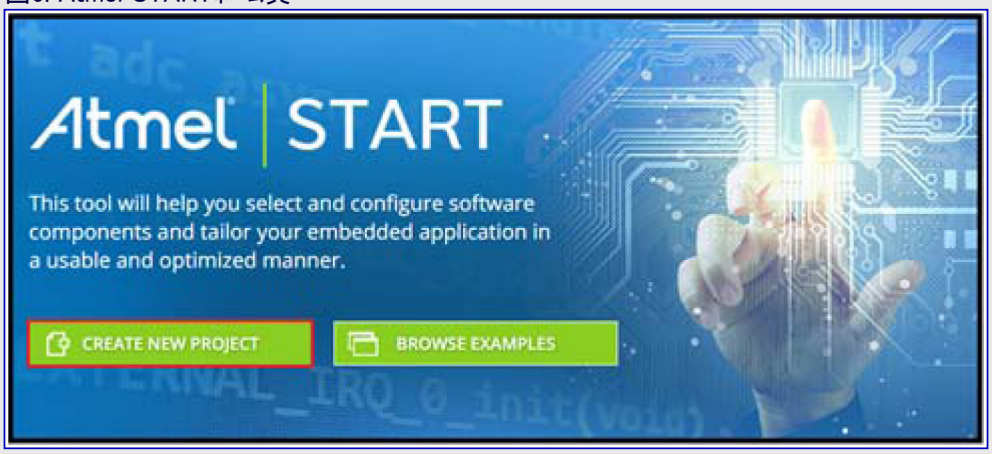
図5. ASFv3 ADCレジスタ値



Atmel START – システム初期化

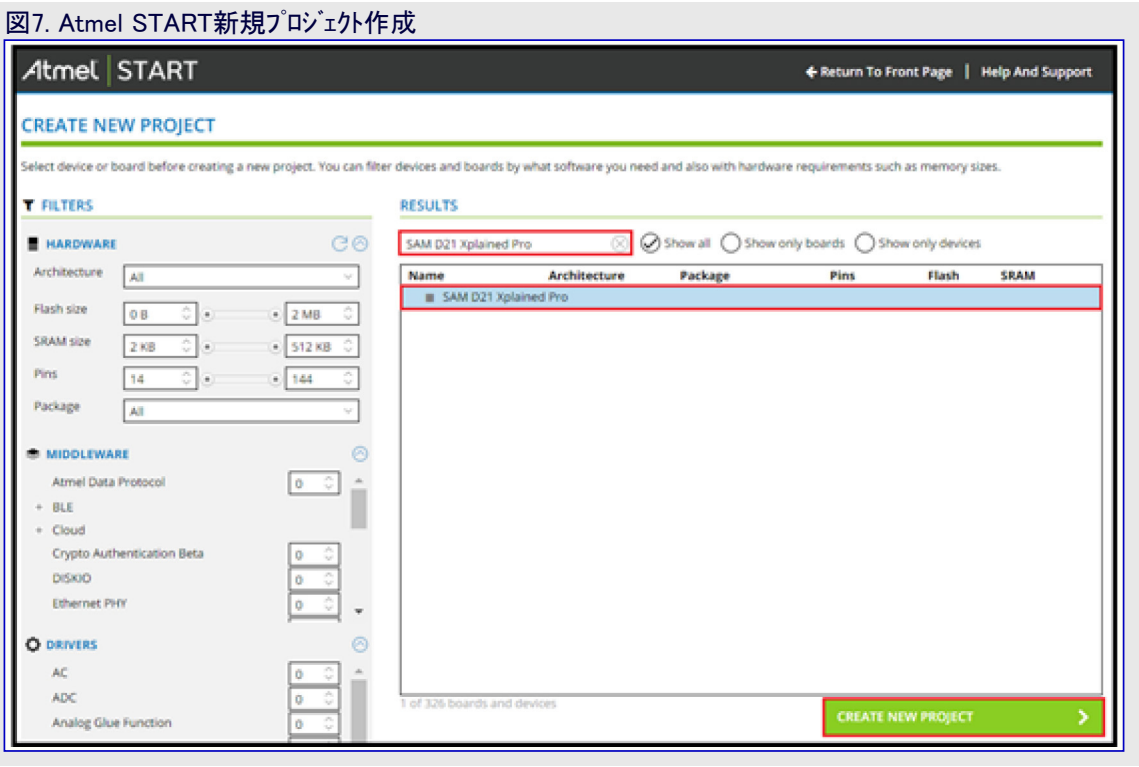
1. 図6で示されるように、start.atmel.comに対して閲覧してその後にCreate New Project(新規プロジェクト作成)タブをクリックすることによって新しいAtmel Startプロジェクトを作成してください。

図6. Atmel STARTホーム頁



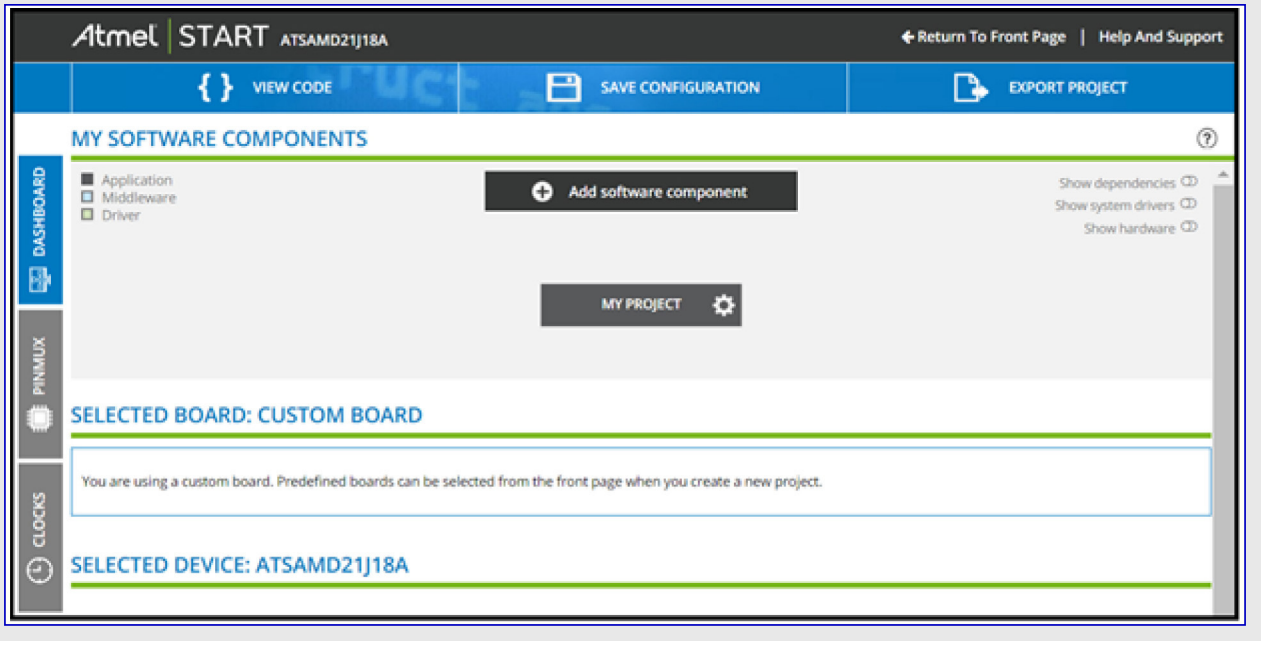
2. Create New Projects(新規ﾌﾟﾛｼﾞｪｸﾄ作成)ｳｨﾝﾄﾞｳで、Results(結果)項目下でSAM D21 Xplained Proを探してSAM D21 Xplained Proを選び、その後にCreate New Project(新規ﾌﾟﾛｼﾞｪｸﾄ作成)をｸﾘｯｸしてください。図7.をご覧ください。

図7. Atmel START新規ﾌﾟﾛｼﾞｪｸﾄ作成



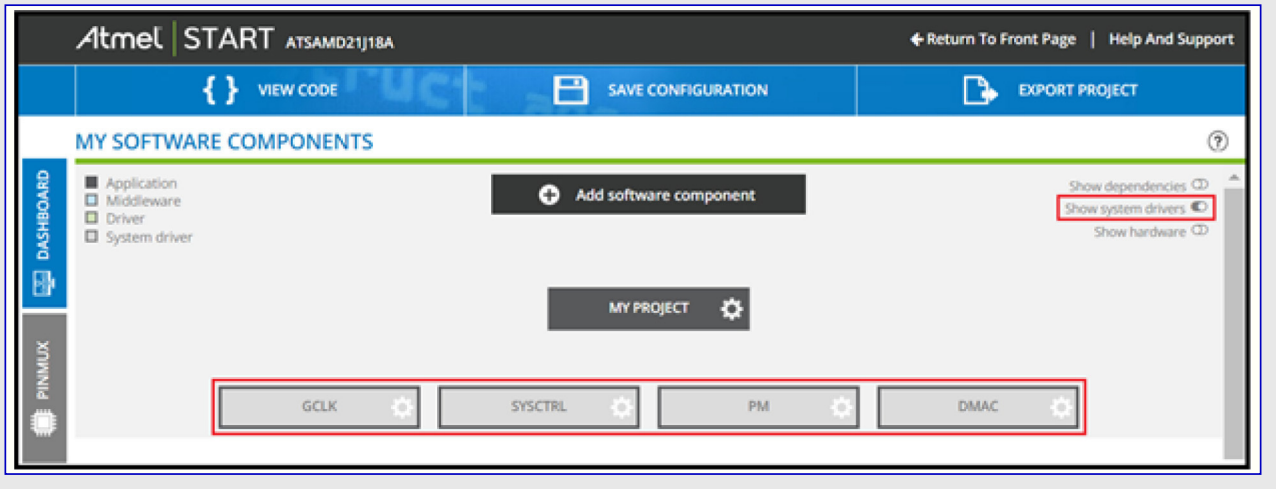
3. 図8.で示されるように新規ﾌﾟﾛｼﾞｪｸﾄのDASHBOARD(指標表示板)が表示されます。

図8. 新規ﾌﾟﾛｼﾞｪｸﾄ指標表示版



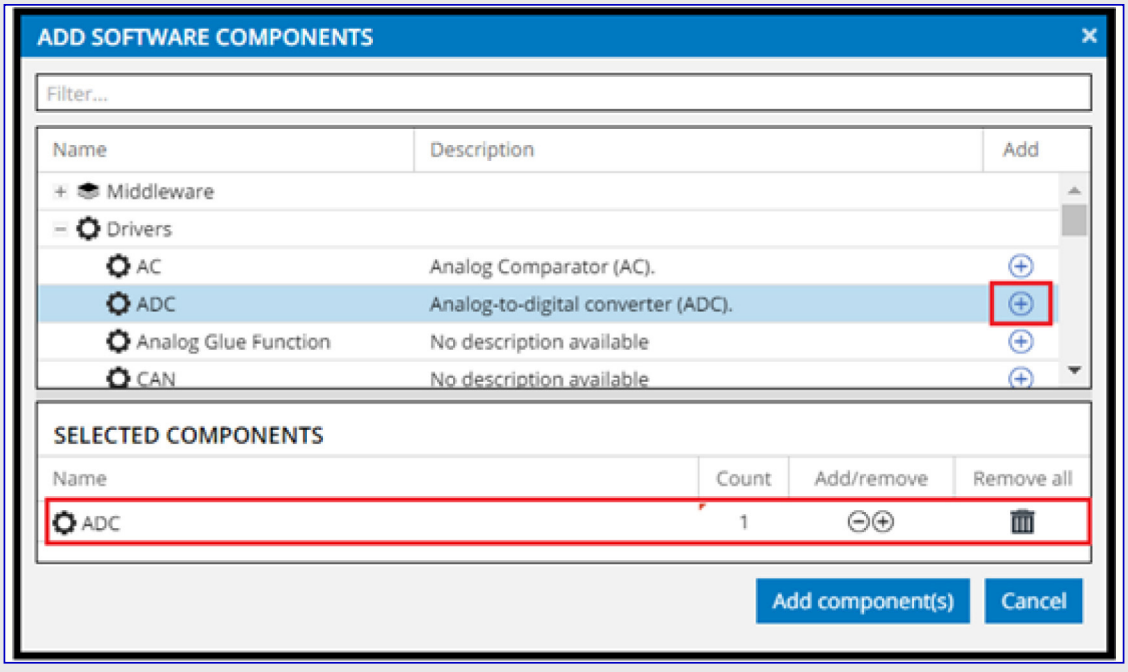
ASFv3でのように全てのASFv4ﾌﾟﾛｼﾞｪｸﾄでいくつかのｼｽﾃﾑ ﾄﾞﾗｲﾊﾞがｲﾝｸﾙｰﾄﾞされることに注意してください。これらのﾄﾞﾗｲﾊﾞは開始構成設定で初めは非表示です。自動的にｲﾝｸﾙｰﾄﾞされたｼｽﾃﾑ ﾄﾞﾗｲﾊﾞ、ｸﾛｯｸを見るにはShow system drivers(ｼｽﾃﾑ ﾄﾞﾗｲﾊﾞ表示)指示部をｸﾘｯｸしてください。図9.で示されるように、4つの構成部品(SYSCTRL、DMAC、PM、GCLK)はDASHBOARD(指標表示板)画面でｸﾛｯｸ、ﾊﾞｽ、NVM、DMAの設定の構成設定を許します。

図9. ｲﾝｸﾙｰﾄﾞされたｼｽﾃﾑ ﾄﾞﾗｲﾊﾞ



4. DASHBOARD(指標表示板)ﾀﾌﾞで、ﾌﾟﾛｼﾞｪｸﾄに周辺機能単位部を追加するためにAdd software component(ｿﾌﾄｳｪｱ部品追加)をｸﾘｯｸしてください。Add Software Components(ｿﾌﾄｳｪｱ部品追加)ﾀﾞｲｱﾛｸﾞは図10.で示されるようにSAMD21に対して利用可なﾄﾞﾗｲﾊﾞを表示します。
5. 全てのﾄﾞﾗｲﾊﾞを表示するためにName(名前)列の”+”をｸﾘｯｸし、その後にAdd(追加)列の”+”ｼﾝﾎﾞﾙをｸﾘｯｸすることによってADCを選んでください。Selected Components(選んだ部品)項目下にADC単位部が表示されます。
6. 選んだ部品をﾌﾟﾛｼﾞｪｸﾄに追加するためにAdd Componet(s)(部品追加)をｸﾘｯｸしてください。

図10. Atmel STARTｿﾌﾄｳｪｱ部品追加



7. 周辺機能単位部を構成設定してください。一旦ADC単位部が追加されると、図11.で示されるようにそれが(画面中央の)MY PROJECT(私のﾌﾟﾛｼﾞｪｸﾄ)ﾀﾌﾞ下に表示されます。

図11. 周辺機能単位部構成設定

8. ADC_0ﾀﾌﾞをｸﾘｯｸし、その後に構成設定任意選択を移入するためにASFv3例ﾌﾟﾛｼﾞｪｸﾄでadc_config構造体から収集した情報を使い、その後に図12.で示されるようにADC入力に対してPA06(AIN/6)を許可してください。

図12. ADC周辺機能構成設定(1/2)

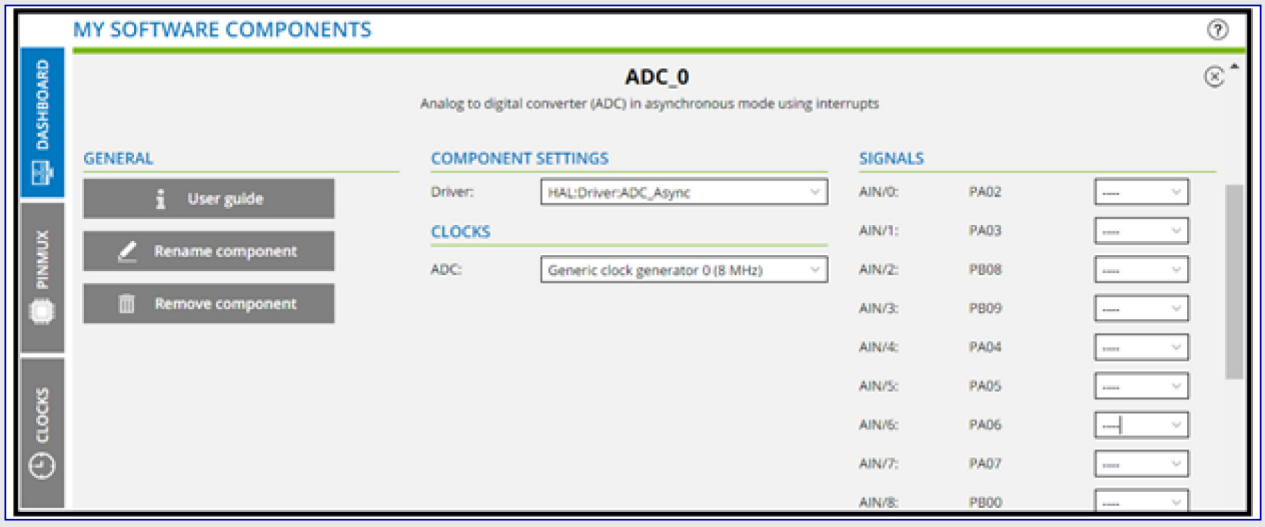
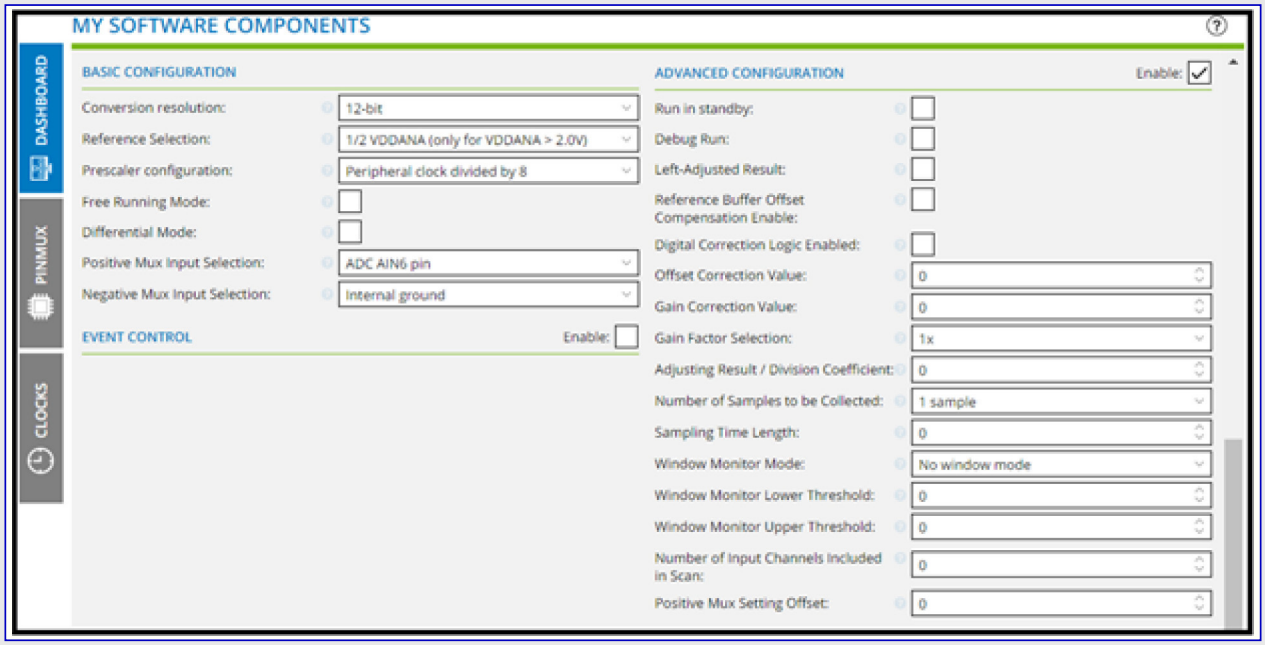


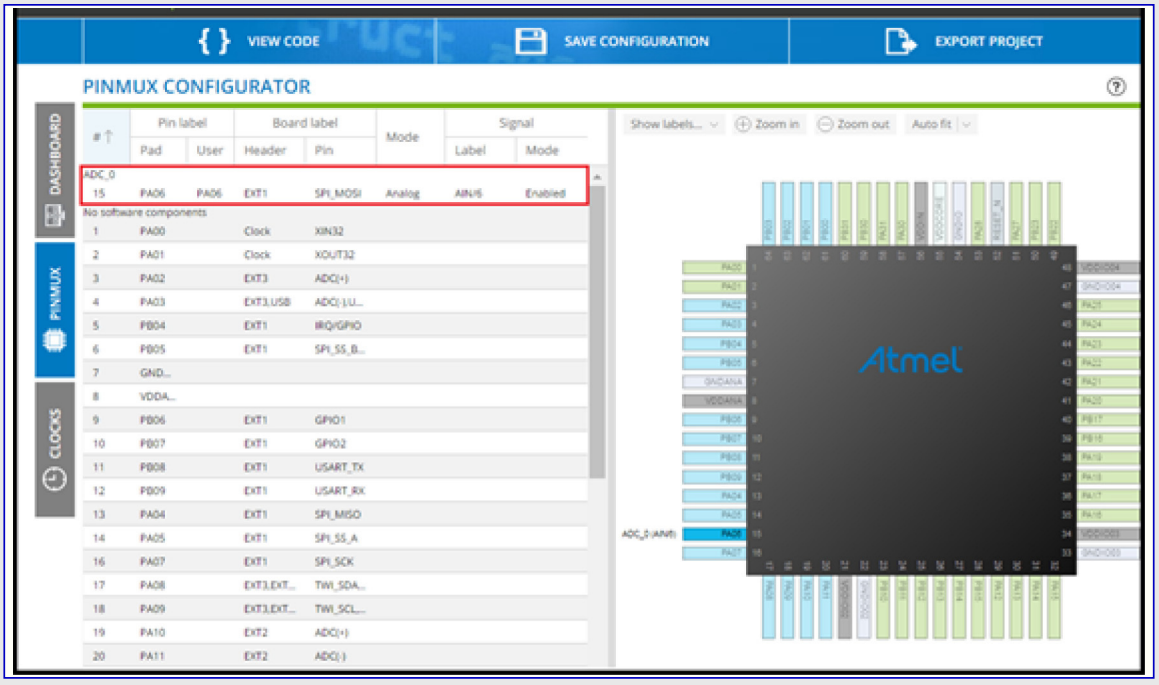
図13. ADC周辺機能構成設定(2/2)



9. ﾋﾟﾝ配置構成設定

ASFv3ﾌﾟﾛｼﾞｪｸﾄから情報を収集する時に、ADCはこのﾌﾟﾛｼﾞｪｸﾄで1つのﾋﾟﾝだけを使い、それはDASHBOARD(指標表示板)上のADC構成設定で構成設定されます。けれども、汎用入出力(GPIO)やﾋﾟﾝの名前付けはPINMUX構成設定部で行われます。図14.をご覧ください。PINMUX構成設定部で変更することを望むﾋﾟﾝを選ぶことにより、ﾊﾟﾗﾒｰﾀを設定するために画面の下近くに使用者ｳｨﾝﾄﾞｳが開きます。

図14. Atmel START – PINMUX

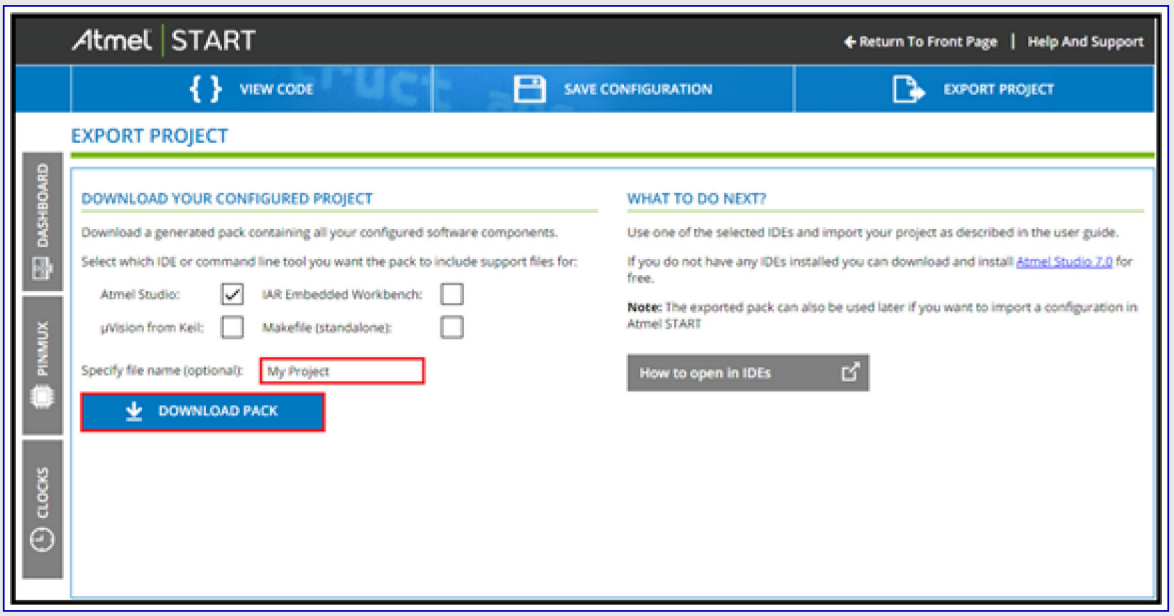


10. ｸﾛｯｸ樹形構成設定

ｸﾛｯｸ樹形はStartﾂｰﾙで、ｼｽﾃﾑ ﾄﾞﾗｲﾊﾞ単位部を通したDASHBOARD(指標表示板)と、ｸﾛｯｸ構成設定部の2つの場所で構成設定することができます。ｸﾛｯｸ構成設定部はｸﾛｯｸ樹形の図画的な表現を提供し、DASHBOARD(指標表示板)と同じ構成設定能力を提供します。この例についてはｸﾛｯｸ構成設定部が使われます。

11. 図15.で示されるように、ﾌﾟﾛｼﾞｪｸﾄをｴｸｽﾎﾟｰﾄするため、Atmel Start画面の右上部分のExport Project(ﾌﾟﾛｼﾞｪｸﾄをｴｸｽﾎﾟｰﾄ)ﾀﾌﾞをｸﾘｯｸすることによってﾌﾟﾛｼﾞｪｸﾄをｴｸｽﾎﾟｰﾄしてください。これがこのｴｸｽﾎﾟｰﾄしたﾌﾟﾛｼﾞｪｸﾄの使用を意図される唯一のIDEのため、Atmel Studioだけが選ばれるべきです。Atmel Startからの出力はatzipﾌｧｲﾙです。

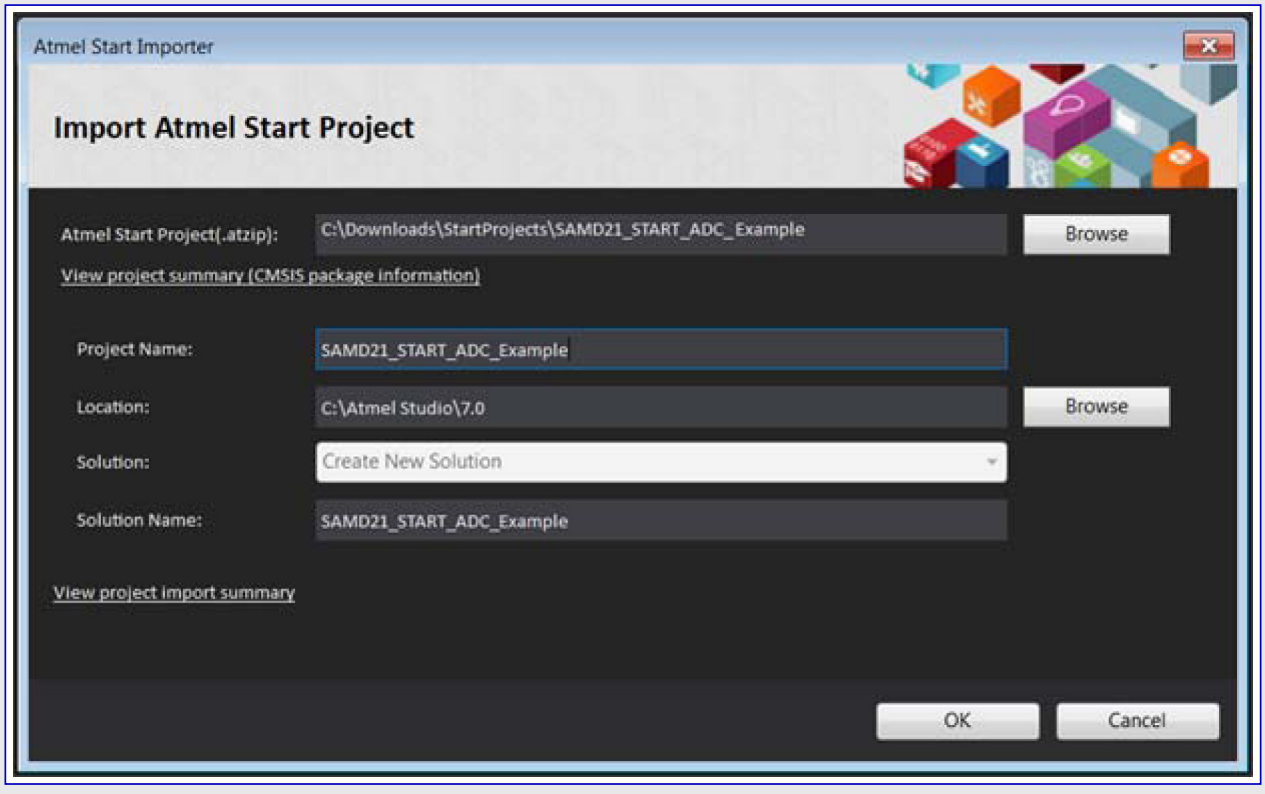
図15. Atmel START – ﾌﾟﾛｼﾞｪｸﾄのｴｸｽﾎﾟｰﾄ



ｼｽﾃﾑ構成設定の確認

1. Atmel StudioでFile(ﾌｧｲﾙ)⇒Import(ｲﾝﾎﾟｰﾄ)⇒Atmel Start Project(Atmel STARTﾌﾟﾛｼﾞｪｸﾄ)を選ぶことによってｴｸｽﾎﾟｰﾄしたatzipﾌｧｲﾙをｲﾝﾎﾟｰﾄしてください。

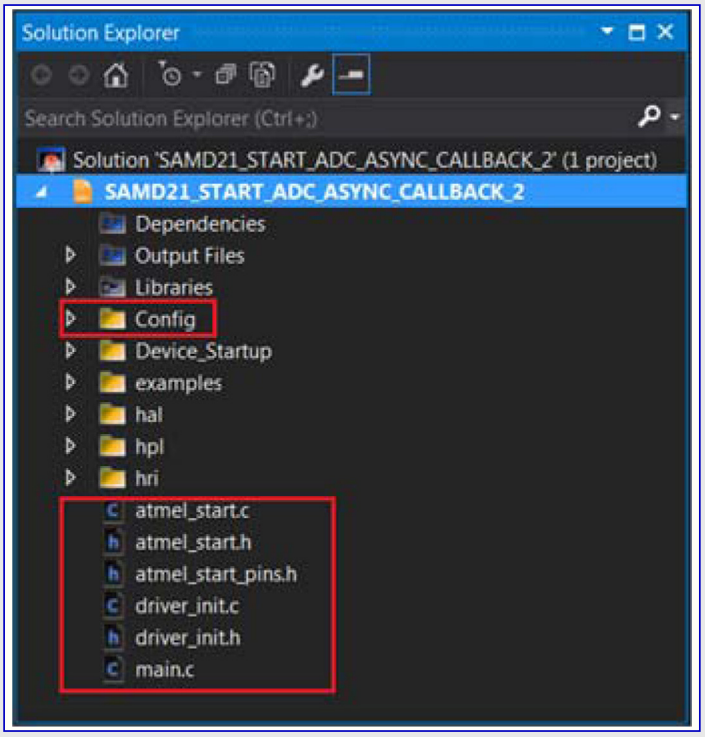
図16. Atmel START – ﾌﾟﾛｼﾞｪｸﾄのｲﾝﾎﾟｰﾄ



Atmel Startでのﾊﾟﾗﾒｰﾀｰ一式は都合の良いことにｼｽﾃﾑ初期化に対してはconfigﾌｫﾙﾀﾞ、ﾋﾟﾝ割り当てに対してはatmel_start_pins.h、周辺機能初期化に対してはdriver_init.c/driver_init.hに置かれます。図17をご覧ください。

- Config – 前処理部ﾃﾞﾊﾞｲｽ構成設定に使われるAtmel Startからのﾚｼﾞｽﾀ設定を含みます。
- atmel_start.c – MCU、ﾄﾞﾗｲﾊﾞ、ﾐﾄﾞﾙｳｪｱを初期化します。
- atmel_start_pins.h – Atmel Start構成設定からMCUﾋﾟﾝ割り当てと命名を保持します。
- driver_init.c – 周辺機能用の初期化関数を含みます。
- main.c – ｼｽﾃﾑを初期化するためにatmel_start_init()を呼びます。

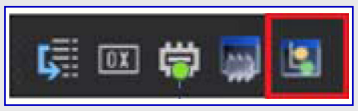
図17. ASFv4 ﾌｧｲﾙ構造



2. ﾌﾟﾛｼﾞｪｸﾄの構築と走行

新しいﾌﾟﾛｼﾞｪｸﾄがどの応用ｺｰﾄﾞも含まないため、Atmel Startはsystem_init()で定義された周辺機能に対する初期化ｺｰﾄﾞを出力します。入出力(I/O)ｳｨﾝﾄﾞｳでﾚｼﾞｽﾀ設定を見るためにﾌﾟﾛｼﾞｪｸﾄを走らせて初期化手順後に停止してください。

入出力(I/O)ｳｨﾝﾄﾞｳは入出力(I/O)ｱｲｺﾝをｸﾘｯｸすることによってAtmel Sudioで開くことで利用可能です。



3. 入出力(I/O)ｳｨﾝﾄﾞｳを開いてA/D変換器を選び、ASFv3から収集したそれらの結果を比較することによってASFv3とASFv4間のﾚｼﾞｽﾀ設定を比較してください(図18.をご覧ください)。

図18. ASFv3とASFv4のADCﾚｼﾞｽﾀ比較

前の比較が行われる時に2つの違いが直ぐに明らかになります。それはADC割り込みが許可される場所とADCの校正設定です。

ASFv3では割り込みがADC作業関数内で許可される一方で、ASFv4は呼び戻し関数が割り当てられた時に割り込みを許可します。

加えて、ASFv4のADCﾄﾞﾗｲﾊﾞはADCに対して工場校正設定をｲﾝﾎﾟｰﾄしません。これは使用者によって行われなければなりません。この校正はもっと正確なADC測定値に対する偏りと直線性の設定を提供します。追加情報についてはSAM D21系統ﾃﾞｰﾀｼｰﾄ(DS40001882)の「**9.3.2. NVMｿﾌﾄｳｪｱ校正領域割り当て**」項を参照してください。ASFv4ﾌﾟﾛｼﾞｪｸﾄに構成をｲﾝﾎﾟｰﾄするには「**追補A：応用ｺｰﾄﾞ例**」を参照してください。

応用の変換

1. 応用説明

   ADCは初期化後に開始され、128個の12ﾋﾞｯﾄ試料を収集してそのﾃﾞｰﾀを緩衝部に格納します。試料が収集されると、ADC呼び戻しに移行して採取の完了を示すﾌﾗｸﾞが設定されます。

2. API比較

   ADC単位部の使用はAtmel Start構成設定で提供された以外にどんな追加の構成設定も必要としません。ASFv3は提供された既定の組と非常に似ていますが、初期化関数を作成して既定ﾃｨﾚｸﾄﾘを変更するのに開発者を必要とします。

   例ﾌﾟﾛｸﾞﾗﾑは定義した試料数で緩衝部を満たすのにadc_read_buffer_jobを使います。この機能はASFv4での直接一致がありません。ASFv4でadc_read_buffer_jobに最も近いのはadc_async_buffer_start_conversionです。けれども、これら2つの関数はかなり違う結果を生じます。ASFv3ではADC_adc_interrupt_handlerが作業を完了させるように構成設定され、割り当てた緩衝部に要求した試料数でﾃﾞｰﾀを入れます。ASFv4のADC処理部はもっと一般的な手法を取るように設計され、呼び戻しで追加の採取を扱うために開発者を必要とします。ASFv3の機能に合わせるため、ASFv4のADC呼び戻しは配列内にADC結果を置いて次のADC作業を開始します。

   完全な構成設定ｺｰﾄﾞについては「**追補A：応用ｺｰﾄﾞ例**」を参照してください。

3. 呼び戻し構成設定

   一般的な応用に関して、主ﾌﾟﾛｼﾞｪｸﾄ樹形内の例ﾌｫﾙﾀﾞはﾌﾟﾛｼﾞｪｸﾄで許可された周辺機能を構成設定するｺｰﾄﾞを含みます。この場合、例ﾌｫﾙﾀﾞは応用に対して呼び戻しを許可するのに必要とされる全てのものを含みます。

   ADC呼び戻しはASFv3とASFv4で同様に構成設定されます。違いは割り込みが合図された時のADC割り込み処理関数の方法です。ASFv3とASFv4間の違いを見るには_adc_interrup_handlerを検索してください。ASFv4は低い付随負荷を提供し、呼び戻しで特別な機能を扱うことを開発者に許します。

   完全な構成設定ｺｰﾄﾞについては「**追補A：応用ｺｰﾄﾞ例**」を参照してください。

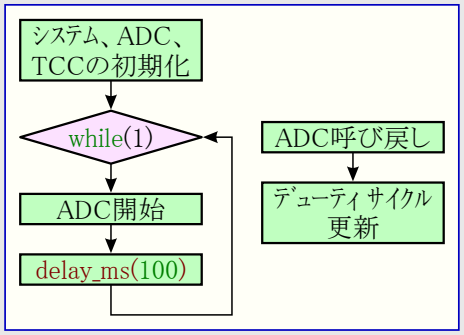
ﾌﾟﾛｼﾞｪｸﾄの拡張

次のようにASFv3ﾌﾟﾛｼﾞｪｸﾄに計時器制御(Timer Control)単位部を追加してください。

1. 応用説明

   更新した応用はLEDに接続されたPWMﾁｬﾈﾙのﾃﾞｭｰﾃｨ ｻｲｸﾙを更新するのにADCの呼び戻しを使います。ﾃﾞｭｰﾃｨ ｻｲｸﾙはADCから読んだ値を用いて計算されます。加えて、ADC値の塊を収集する単一作業を持つよりもむしろADCは100ms毎にPWMﾁｬﾈﾙのﾃﾞｭｰﾃｨ ｻｲｸﾙを更新するための継続的な繰り返しで開始されます。

図19. 拡張したﾌﾟﾛｼﾞｪｸﾄの流れ図



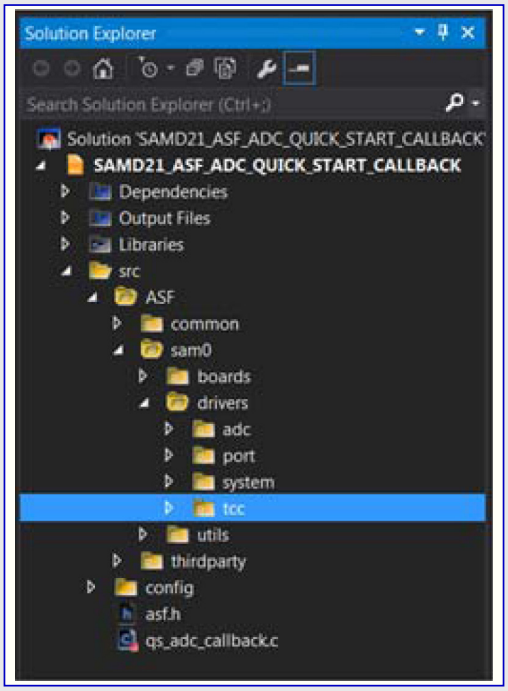
2. ﾌﾟﾛｼﾞｪｸﾄの新しい機能を達成するにはASFｳｨｻﾞｰﾄﾞから遅延ｻｰﾋﾞｽとTCC0ﾄﾞﾗｲﾊﾞ単位部が追加されなければなりません。
3. ASFv3ﾌﾟﾛｼﾞｪｸﾄを開いて計時器制御(Timer Control)単位部を追加してください。

   Atmel Studioに於いてProject下でASF Wizardを選んでください。一旦ｳｨﾝﾄﾞｳが開いたなら、変更されるべき例ﾌﾟﾛｼﾞｪｸﾄがﾌﾟﾛｼﾞｪｸﾄ引き落としﾒﾆｭｰに入れられているのを確認してください。

   Available Modules(利用可能な単位部)ﾒﾆｭｰで選択した呼び戻し(callback)任意選択と共にTC - Timer Counter (driver)(TC - 計時器/計数器(ﾄﾞﾗｲﾊﾞ))とDelay services(遅延ｻｰﾋﾞｽ)単位部を選んでください。ｳｨﾝﾄﾞｳの下部に配置されたApply(適用)をｸﾘｯｸすることによって追加した単位部をﾌﾟﾛｼﾞｪｸﾄに適用してください。TCﾄﾞﾗｲﾊﾞと遅延ｻｰﾋﾞｽの両方がSelect Modules(単位部選択)ﾒﾆｭｰで表示されます。

   ASFｳｨｻﾞｰﾄﾞによって一旦ﾌﾟﾛｼﾞｪｸﾄにTCCが追加されると、**図20.**で示されるようにﾌﾟﾛｼﾞｪｸﾄでtccﾄﾞﾗｲﾊﾞ ﾌｫﾙﾀﾞを見つけることができ、応用で使うために適切なﾍｯﾀﾞ ﾌｧｲﾙがasf.hに追加されます。

図20. 追加したTCC0ﾄﾞﾗｲﾊﾞ



ASFv3計時器構成の構成設定

1. 計時器構成の構成設定

ﾌﾟﾛｼﾞｪｸﾄがXplained Proで構築されているため、PWM出力として使うために基板上のLEDにいくつかの定義が割り当てられています。この定義はﾌﾟﾛｼﾞｪｸﾄ内のsamd21_xplained_pro.hﾌｧｲﾙに含められます。**図21.**をご覧ください。

図21. Xplained Pro定義

```
116  #define LED0_GPIO                LED0_PIN
117  #define LED0                     LED0_PIN
118
119  #define LED_0_PWM4CTRL_MODULE    TCC0
120  #define LED_0_PWM4CTRL_CHANNEL   0
121  #define LED_0_PWM4CTRL_OUTPUT    0
122  #define LED_0_PWM4CTRL_PIN       PIN_PB30E_TCC0_WO0
123  #define LED_0_PWM4CTRL_MUX       MUX_PB30E_TCC0_WO0
124  #define LED_0_PWM4CTRL_PINMUX    PINMUX_PB30E_TCC0_WO0
```

samd21_xplained_pro.hで定義されたﾏｸﾛを用いて**図22.**で示されるようにconfigure_t cc関数を作成してください。tcc_instanceと呼ばれるt cc_module構造体の実体を作成してください。configure_tcc関数の最初の行の1つが既定TCC単位部の構成設定用の値一式を得るための関数呼び出しであることに注目してください。ASFv3は各周辺機能用の既定設定一式を含みます。これらの既定はこのﾌﾟﾛｼﾞｪｸﾄ用の基準として使われ、必要とされる場所で変更されます。

図22. configure_tcc関数

```
struct tcc_module tcc_instance;

static void configure_tcc(void)
{
    struct tcc_config config_tcc;

    tcc_get_config_defaults(&config_tcc, LED_0_PWM4CTRL_MODULE);

    config_tcc.counter.clock_prescaler = TCC_CLOCK_PRESCALER_DIV8;
    config_tcc.counter.period = 0x7C;
    config_tcc.compare.wave_generation = TCC_WAVE_GENERATION_SINGLE_SLOPE_PWM;
    config_tcc.compare.match[LED_0_PWM4CTRL_CHANNEL] = 0x3E;

    config_tcc.pins.enable_wave_out_pin[LED_0_PWM4CTRL_OUTPUT] = true;
    config_tcc.pins.wave_out_pin[LED_0_PWM4CTRL_OUTPUT]        = LED_0_PWM4CTRL_PIN;
    config_tcc.pins.wave_out_pin_mux[LED_0_PWM4CTRL_OUTPUT]    = LED_0_PWM4CTRL_MUX;
    //config_tcc.double_buffering_enabled  = false;

    tcc_init(&tcc_instance, LED_0_PWM4CTRL_MODULE, &config_tcc);

    tcc_enable(&tcc_instance);
}
```

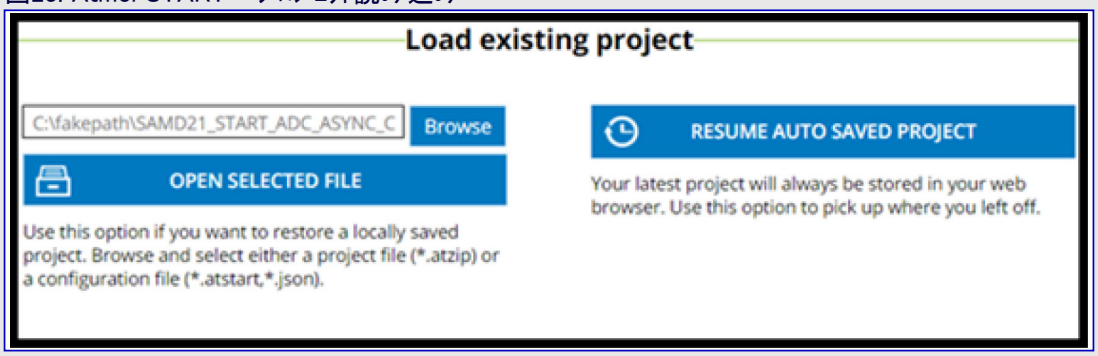
2. TCC0レジスタ設定入手

   一旦TCC単位部が構成設定されてしまうと、Atmel Studioでコンパイルして走らせてください。ADCに対して前に入出力(I/O)ウィンドウを用いて行ったようにレジスタ設定を取り込んでください。これらの値はAtmel Startからの出力と比較するのに使われます。

Atmel START – 計時器の追加

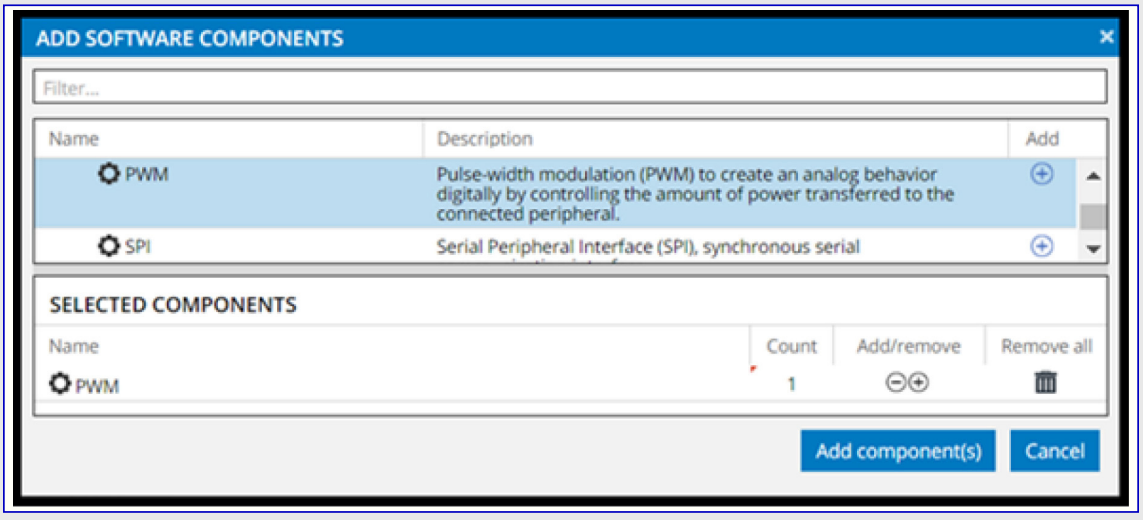
1. Atmel Startホーム頁を閲覧してLoad existing project(既存プロジェクト読み込み)任意選択を選ぶことによってAtmel Startプロジェクトをインポートしてください。その後、ダウンロードしたatzipファイルを検索してOPEN SELECTED FILE(選んだファイルを開く)をクリックしてください。

図23. Atmel START – プロジェクト読み込み



2. ADD SOFTWARE COMPONENTS(ソフトウェア部品追加)をクリックし、その後にウィンドウでPWM部品を選ぶことによってプロジェクトに周辺機能単位部を追加してください。共にTCC単位部を使うとは言え、PWM部品はTCC部品と異なることに注意してください。

図24. Atmel START – ソフトウェア部品追加



3. 図25.と図26.で示されるように、新しく追加したPWM_0単位部を選んで構成設定するためにASFv3プロジェクトから収集した値を用いることによって周辺機能単位部を構成設定してください。

図25. Atmel START – PWM構成設定(1/2)

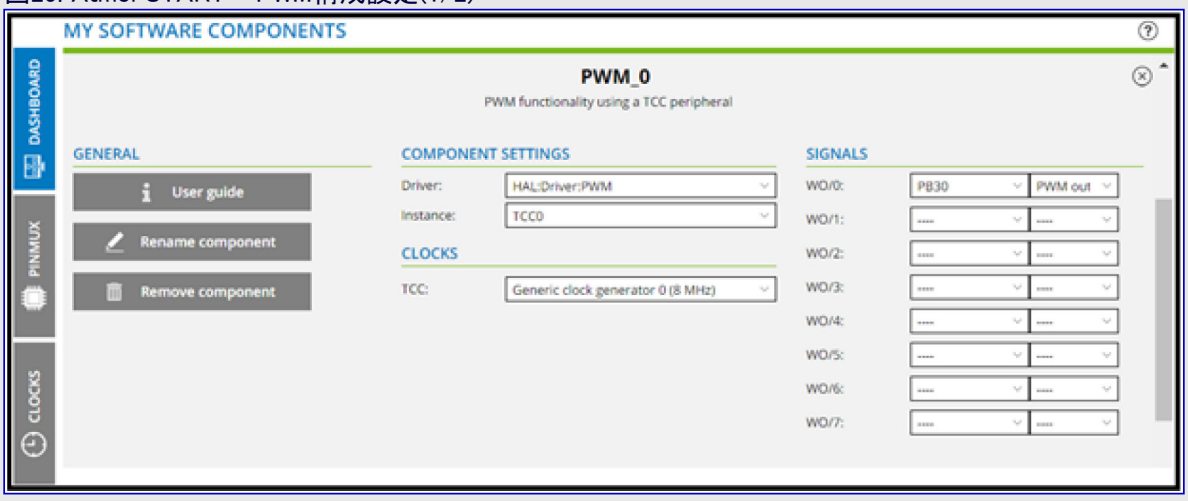
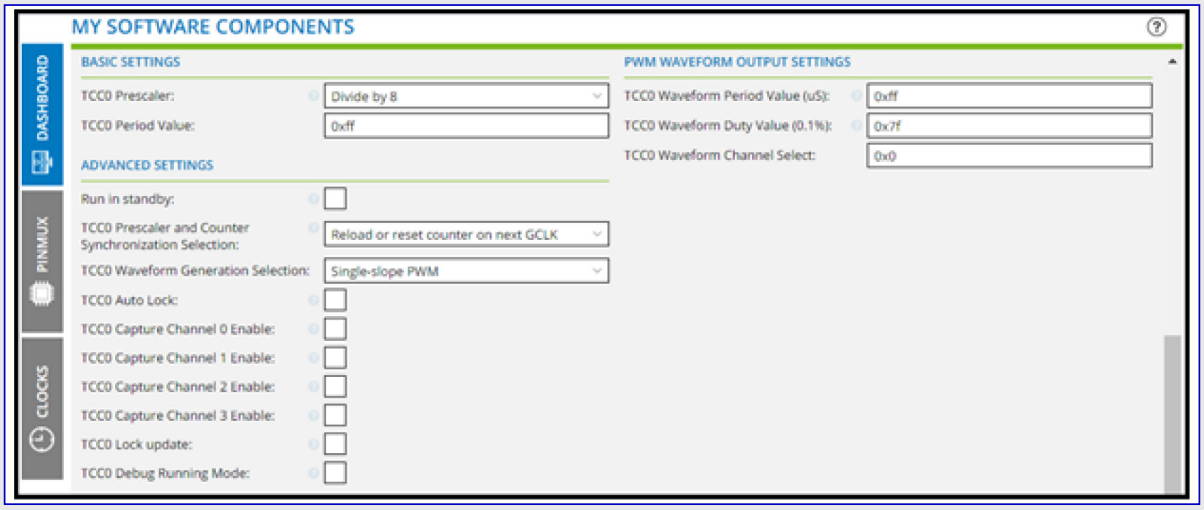


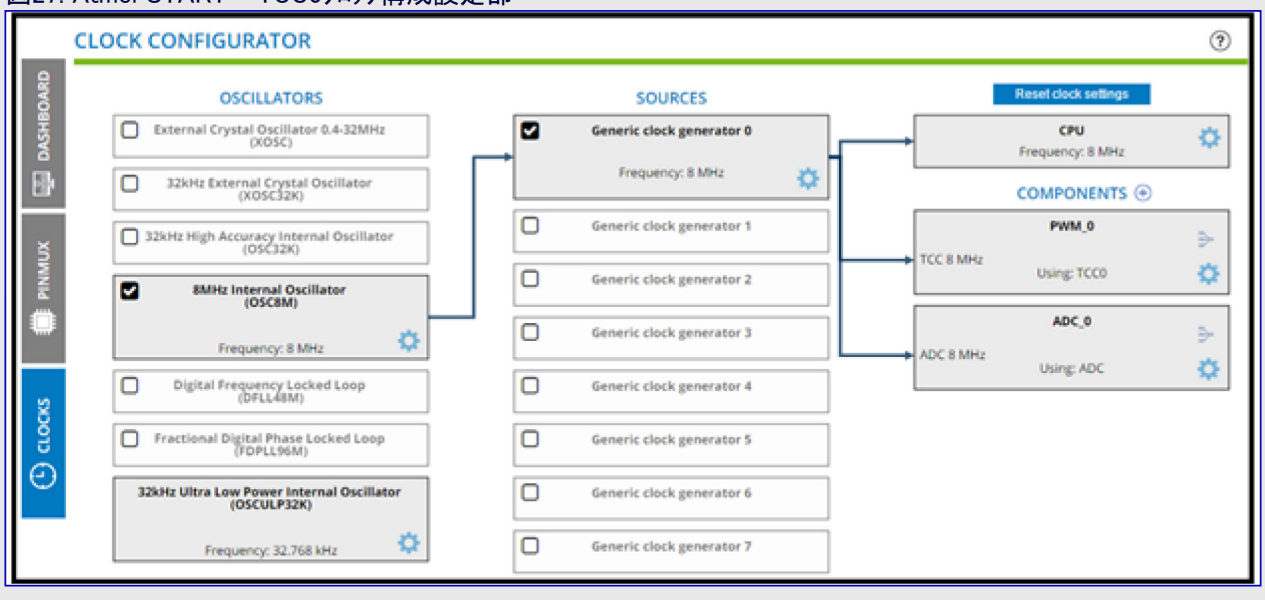
図26. Atmel START – PWM構成設定(2/2)



4. クロック樹形の構成設定

このプロジェクトの初版は一般クロック生成器0(GCLK_0)へ行く8MHzを持っていました。ADC_0とWPM_0の両方がGCLK_0によってクロック信号を供給されることを確実にしてください。PWM_0部品がGCLK_0によって供給されない場合、COMPONENTS(部品)部の設定アイコンをクリックしてTCCクロック元をGLK_0に設定してください。

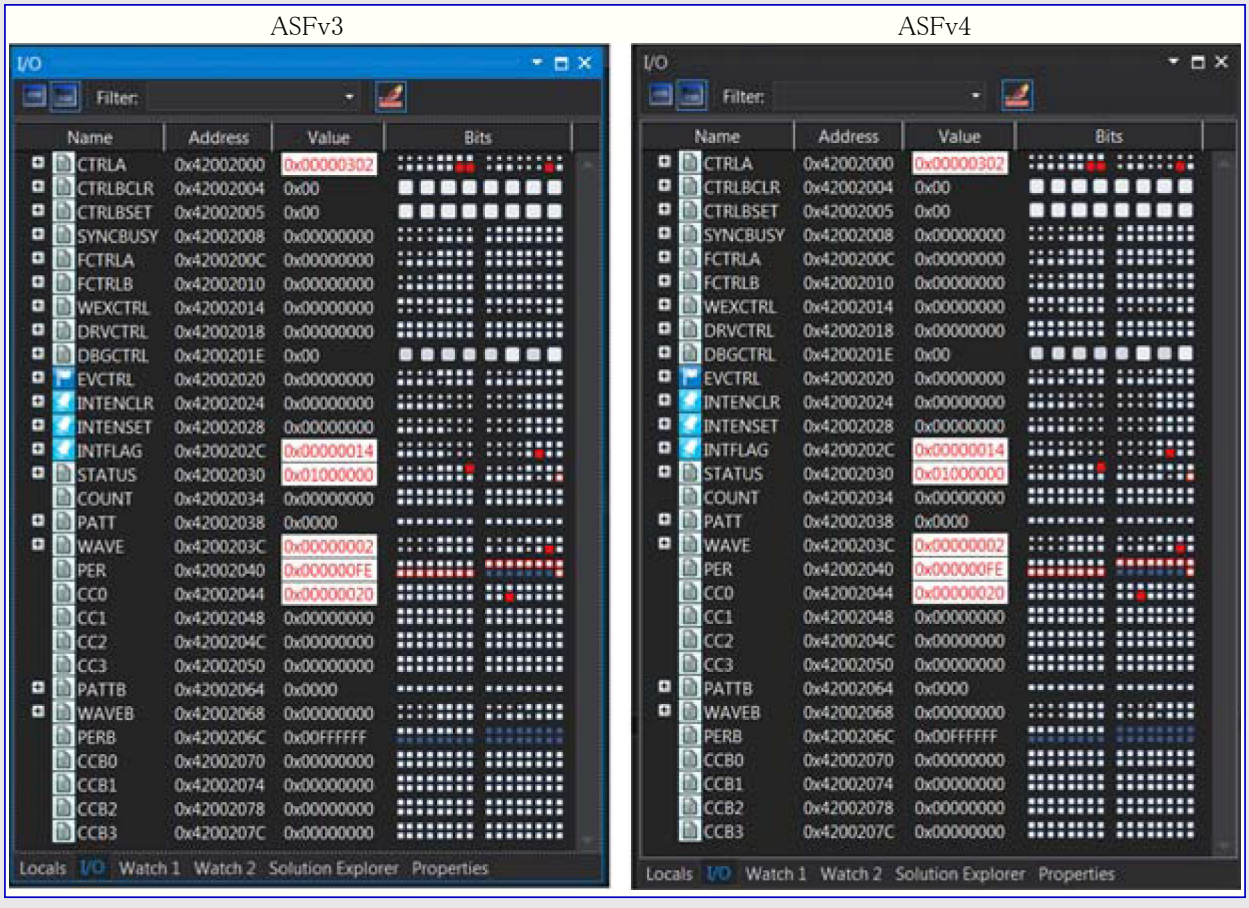
図27. Atmel START – TCC0クロック構成設定部



システム構成設定の確認

1. 以前に記述されたようにプロジェクトをインポートして構築することによってASFv3とASFv4間のレジスタ設定を比較してください。
2. TCC0周辺機能が初期化されて許可されてしまうまでプロジェクトを走らせてください。図28.はASFv3とASFv4間の初期化の違いを表示します。図28.での比較から、TCC0単位部の初期化はASFv4で構成設定されます。

図28. Atmel START – TCC0レジスタ比較



応用の更新

1. 応用説明

   PWMデューティ サイクル計算用の入力を提供するにはADC呼び戻しでの算法が実装されることだけが必要です。ADCは単一12ビット試料収集に構成設定され、その後に呼び戻しを起動します。ADC呼び戻しでは、ADCの結果(RESULT)レジスタが直接読まれて8ビットの結果を提供するために位置移動されます。この8ビット値はPWMのデューティ サイクルに書かれます。PWMは8ビット周期を持つように構成設定され、故にデューティ サイクルは適切に尺度調整されます。

2. API比較

   TCC0単位部の使用はAtmel Start構成設定から提供されたもの以外にどんな追加の構成設定も必要としません。ASFv3は既定の一式で提供されたものと非常に似ていますが、初期化関数を作成して既定を直接変更するために関発者を必要とします。

   ASFv4は枠組み全体を通してより多くの使用事例で駆動される模式を持ちます。これはTCC部品がASFv3で追加され、PWM部品がAt mel startで追加された時から見ることができます。ASFv4でのPWM部品用APIはpwm_set_parameterでこれをもっと良く表示します。ASFv3は同じ機能を達成するのにtcc_set_compare_valueを使います。これらの関数で渡されるパラメータは劇的に違いませんが、ASFv4で達成される事例駆動手法を用いて関数を掘り下げてもっと薄くします。

3. 呼び戻し構成設定

   このプロジェクトのTCC0単位部は呼び戻しを使いませんが、ADC呼び戻し関数でデューティ サイクルが変更されます。

結び

ASFv4はコード量と応用駆動使用事例手法でのドライバ効率性での改良を含む新しい枠組みです。ASFv4の基本構造がASFv3のものと異なる一方で、デバイスに関する命名規則と基本的な側面は残ります。

ASFv3からASFv4への変換に取り組むための予め準備された解決策はありません。移植は0から完全に生じなければなりません。この資料が2つの機能的な違いだけを網羅した一方で、機能的な移植を達成するために基本的な原則が提供されます。

追補A：応用ｺｰﾄﾞ例

例1：ASFv3ﾌﾟﾛｼﾞｪｸﾄ拡張 – ADCとPWM

```
#define ADC_SAMPLES 1
uint16_t adc_result_buffer[ADC_SAMPLES];

struct adc_module adc_instance;
struct tcc_module tcc_instance;

volatile bool adc_read_done = false;void adc_complete_callback(struct adc_module *const module)
{
    uint32_t duty;
    duty = (adc_result_buffer[0] >> 4) & 0xFFFF;
    tcc_set_compare_value(&tcc_instance, (enum tcc_match_capture_channel)(CONF_PWM_CHANNEL), duty);
}

void configure_adc(void)
{
    struct adc_config config_adc;

    adc_get_config_defaults(&config_adc);

    config_adc.clock_prescaler = ADC_CLOCK_PRESCALER_DIV8;
    config_adc.reference       = ADC_REFERENCE_INTVCC1;
    config_adc.positive_input  = ADC_POSITIVE_INPUT_PIN18;
    config_adc.resolution      = ADC_RESOLUTION_12BIT;

    adc_init(&adc_instance, ADC, &config_adc);
    adc_enable(&adc_instance);
}

static void configure_tcc(void)
{
    struct tcc_config config_tcc;

    tcc_get_config_defaults(&config_tcc, LED_0_PWM4CTRL_MODULE);

    config_tcc.counter.clock_prescaler = TCC_CLOCK_PRESCALER_DIV8;
    config_tcc.counter.period = 0xFE;
    config_tcc.compare.wave_generation = TCC_WAVE_GENERATION_SINGLE_SLOPE_PWM;
    config_tcc.compare.match[LED_0_PWM4CTRL_CHANNEL] = 0x7F;

    config_tcc.pins.enable_wave_out_pin[LED_0_PWM4CTRL_OUTPUT] = true;
    config_tcc.pins.wave_out_pin[LED_0_PWM4CTRL_OUTPUT]        = LED_0_PWM4CTRL_PIN;
    config_tcc.pins.wave_out_pin_mux[LED_0_PWM4CTRL_OUTPUT]    = LED_0_PWM4CTRL_MUX;

    tcc_init(&tcc_instance, LED_0_PWM4CTRL_MODULE, &config_tcc);
    tcc_enable(&tcc_instance);
}

void configure_adc_callbacks(void)
{
    adc_register_callback(&adc_instance, adc_complete_callback, ADC_CALLBACK_READ_BUFFER);
    adc_enable_callback(&adc_instance, ADC_CALLBACK_READ_BUFFER);
}

int main(void)
{
    system_init();
    delay_init();

    configure_adc();
    configure_adc_callbacks();
```

```
    configure_tcc();

    system_interrupt_enable_global();

    while (1) {
        adc_read_buffer_job(&adc_instance, adc_result_buffer, ADC_SAMPLES);
        delay_cycles_ms(100);
    }
}
```

例2 : ASFv3移植 – ADCのみ

```
void adc_complete_callback(const struct adc_async_descriptor *const descr, const uint8_t channel)
{
    static uint8_t i = 0;

    if (i < ADC_SAMPLES)
    {
        adc_result_buffer[i++] = ADC->RESULT.reg;
        adc_async_start_conversion(&ADC_0);
    }else
    {
        adc_read_done = true;
    }
}

int main(void)
{
    system_init();

    adc_async_register_callback(&ADC_0, 0, ADC_ASYNC_CONVERT_CB, adc_complete_callback);
    adc_async_enable_channel(&ADC_0, 0);

    adc_async_start_conversion(&ADC_0);

    while (adc_read_done == false)
    {
        /* 非同期ADC読み込み完了待機 */
    }

    while(1)
    {
        asm("nop");
    }
}
```

例3 : ﾌﾟﾛｼﾞｪｸﾄ拡張 – ADCとPWM

```
#define PWM_PERIOD 254
static uint32_t pwm_duty;
static uint16_t adc_value;

static void adc_cb(const struct adc_async_descriptor *const descr, const uint8_t channel)
{
    adc_value = ADC->RESULT.reg;

    pwm_duty = (adc_value >> 4) & 0xFF;

    pwm_set_parameters(&PWM_0, PWM_PERIOD, pwm_duty);
}

int main(void)
{
    atmel_start_init();
```

```
    adc_async_register_callback(&ADC_0, 0, ADC_ASYNC_CONVERT_CB, adc_cb);

    ADC->CALIB.reg = ADC_CALIB_BIAS_CAL((*(uint32_t *)ADC_FUSES_BIASCAL_ADDR >>
        ADC_FUSES_BIASCAL_Pos)) |
    ADC_CALIB_LINEARITY_CAL((*(uint64_t *)ADC_FUSES_LINEARITY_0_ADDR >>
        ADC_FUSES_LINEARITY_0_Pos));

    adc_async_enable_channel(&ADC_0, 0);

    while(1)
    {
        adc_async_start_conversion(&ADC_0);delay_ms(100);
    }
}
```

Microchipﾃﾞﾊﾞｲｽでの以下のｺｰﾄﾞ保護機能の詳細に注意してください。

- Microchip製品はそれら特定のMicrochipﾃﾞｰﾀｼｰﾄに含まれる仕様に合致します。
- Microchipは意図した方法と通常条件下で使われ使われる時に、その製品系統が今日の市場でその種類の最も安全な系統の1つであると考えます。
- ｺｰﾄﾞ保護機能を破るのに使われる不正でおそらく違法な方法があります。当社の知る限りこれらの方法の全てはMicrochipのﾃﾞｰﾀｼｰﾄに含まれた動作仕様外の方法でMicrochip製品を使うことが必要です。おそらく、それを行う人は知的財産の窃盗に関与しています。
- Microchipはそれらのｺｰﾄﾞの完全性について心配されているお客様と共に働きたいと思います。
- Microchipや他のどの半導体製造業者もそれらのｺｰﾄﾞの安全を保証することはできません。ｺｰﾄﾞ保護は当社が製品を”破ることができない”として保証すると言うことを意味しません。

ｺｰﾄﾞ保護は常に進化しています。Microchipは当社製品のｺｰﾄﾞ保護機能を継続的に改善することを約束します。Microchipのｺｰﾄﾞ保護機能を破る試みはﾃﾞｼﾞﾀﾙ ﾐﾚﾆｱﾑ著作権法に違反するかもしれません。そのような行為があなたのｿﾌﾄｳｪｱや他の著作物に不正なｱｸｾｽを許す場合、その法律下の救済のために訴権を持つかもしれません。

ﾃﾞﾊﾞｲｽ応用などに関してこの刊行物に含まれる情報は皆さまの便宜のためにだけ提供され、更新によって取り換えられるかもしれません。皆さまの応用が皆さまの仕様に合致するのを保証するのは皆さまの責任です。**Microchipはその条件、品質、性能、商品性、目的適合性を含め、明示的にも黙示的にもその情報に関連して書面または表記された書面または黙示の如何なる表明や保証もしません。**Microchipはこの情報とそれの使用から生じる全責任を否認します。生命維持や安全応用でのMicrochipﾃﾞﾊﾞｲｽの使用は完全に購入者の危険性で、購入者はそのような使用に起因する全ての損害、請求、訴訟、費用からMicrochipを擁護し、補償し、免責にすることに同意します。他に言及されない限り、Microchipのどの知的財産権下でも暗黙的または違う方法で許認可は譲渡されません。

Microchipはその世界的な本社、ｱﾘｿﾞﾅ州のﾁｬﾝﾄﾞﾗｰとﾃﾝﾍﾟ、ｵﾚｺﾞﾝ州ｸﾞﾗｼｬﾑの設計とｳｪﾊｰ製造設備とｶﾘﾌｫﾙﾆｱとｲﾝﾄﾞの設計ｾﾝﾀｰに対してISO/TS-16949:2009認証を取得しました。当社の品質ｼｽﾃﾑの処理と手続きはPIC® MCUとdsPIC® DSC、KEELOQ®符号飛び回りﾃﾞﾊﾞｲｽ、直列EEPROM、ﾏｲｸﾛ周辺機能、不揮発性ﾒﾓﾘ、ｱﾅﾛｸﾞ製品用です。加えて、開発ｼｽﾃﾑの設計と製造のためのMicrochipの品質ｼｽﾃﾑはISO 9001:2000認証取得です。

DNVによって認証された
品質管理ｼｽﾃﾑ
= ISO/TS 16949 =

商標

Microchipの名前とﾛｺﾞ、Mcicrochipﾛｺﾞ、AnyRate、AVR、AVRﾛｺﾞ、AVR Freaks、BeaaconThings、BitCloud、CryptoMemory、CryptoRF、dsPIC、FlashFlex、flexPWR、Heldo、JukeBlox、KeeLoq、KeeLoqﾛｺﾞ、Kleer、LANCheck、LINK MD、maXStylus、maXTouch、MediaLB、megaAVR、MOST、MOSTﾛｺﾞ、MPLAB、OptoLyzer、PIC、picoPower、PICSTART、PIC32ﾛｺﾞ、Prochip Designer、QTouch、RightTouch、SAM-BA、SpyNIC、SST、SSTﾛｺﾞ、SuperFlash、tinyAVR、UNI/O、XMEGAは米国と他の国に於けるMicrochip Technology Incorporatedの登録商標です。

ClockWorks、The Embedded Control Solutions Company、EtherSynch、Hyper Speed Control、HyperLight Load、IntelliMOS、mTouch、Precision Edge、Quiet-Wireは米国に於けるMicrochip Technology Incorporatedの登録商標です。

Adjacent Key Suppression、AKS、Analog-for-the-Digital Age、Any Capacitor、AnyIn、AnyOut、BodyCom、chipKIT、chipKITﾛｺﾞ、CodeGuard、CryptoAuthentication、CryptoCompanion、CryptoController、dsPICDEM、dsPICDEM.net、Dynamic Average Matching、DAM、ECAN、EtherGREEN、In-Circuit Serial Programming、ICSP、Inter-Chip Connectivity、JitterBlocker、KleerNet、KleerNetﾛｺﾞ、Mindi、MiWi、motorBench、MPASM、MPF、MPLAB Certifiedﾛｺﾞ、MPLAB、MPLINK、MultiTRAK、NetDetach、Omniscient Code Generation、PICDEM、PICDEM.net、PICkit、PICtail、PureSilicon、QMatrix、RightTouchﾛｺﾞ、REAL ICE、Ripple Blocker、SAM-ICE、Serial Quad I/O、SMART-I.S.、SQI、SuperSwitcher、SuperSwitcher II、Total Endurance、TSHARC、USBCheck、VariSense、View Sense、WiperLock、Wireless DNA、ZENAは米国と他の国に於けるMicrochip Technology Incorporatedの商標です。

SQTPは米国に於けるMicrochip Technology Incorporatedの役務標章です。

Silicon Storage Technologyは他の国に於けるMicrochip Technology Inc.の登録商標です。

GestICは他の国に於けるMicrochip Technology Inc.の子会社であるMicrochip Technology Germany II GmbH & Co. KGの登録商標です。

ここで言及した以外の全ての商標はそれら各々の会社の所有物です。

日本語© *HERO* 2021.

本応用記述はMicrochipのAN2474応用記述(DS00002474A-2017年10月)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。

世界的な販売とサービス

米国	亜細亜/太平洋	亜細亜/太平洋	欧州
本社 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 技術支援: http://www.microchip.com/support ｳｪﾌﾞｱﾄﾞﾚｽ: www.microchip.com **ｱﾄﾗﾝﾀ** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **ｵｰｽﾁﾝ** TX Tel: 512-257-3370 **ﾎﾞｽﾄﾝ** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **ｼｶｺﾞ** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **ﾀﾞﾗｽ** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **ﾃﾞﾄﾛｲﾄ** Novi, MI Tel: 248-848-4000 **ﾋｭｰｽﾄﾝ** TX Tel: 281-894-5983 **ｲﾝﾃﾞｱﾅﾎﾟﾘｽ** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **ﾛｻﾝｾﾞﾙｽ** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **ﾛｰﾘｰ** NC Tel: 919-844-7510 **ﾆｭｰﾖｰｸ** NY Tel: 631-435-6000 **ｻﾝﾎｾ** CA Tel: 408-735-9110 Tel: 408-436-4270 **ｶﾅﾀﾞ - ﾄﾛﾝﾄ** Tel: 905-695-1980 Fax: 905-695-2078	**亜細亜太平洋支社** Suites 3707-14, 37th Floor Tower 6, The Gateway Harbour City, Kowloon **香港** Tel: 852-2943-5100 Fax: 852-2401-3431 **ｵｰｽﾄﾗﾘｱ - ｼﾄﾞﾆｰ** Tel: 61-2-9868-6733 Fax: 61-2-9868-6755 **中国 - 北京** Tel: 86-10-8569-7000 Fax: 86-10-8528-2104 **中国 - 成都** Tel: 86-28-8665-5511 Fax: 86-28-8665-7889 **中国 - 重慶** Tel: 86-23-8980-9588 Fax: 86-23-8980-9500 **中国 - 東莞** Tel: 86-769-8702-9880 **中国 - 広州** Tel: 86-20-8755-8029 **中国 - 杭州** Tel: 86-571-8792-8115 Fax: 86-571-8792-8116 **中国 - 香港特別行政区** Tel: 852-2943-5100 Fax: 852-2401-3431 **中国 - 南京** Tel: 86-25-8473-2460 Fax: 86-25-8473-2470 **中国 - 青島** Tel: 86-532-8502-7355 Fax: 86-532-8502-7205 **中国 - 上海** Tel: 86-21-3326-8000 Fax: 86-21-3326-8021 **中国 - 瀋陽** Tel: 86-24-2334-2829 Fax: 86-24-2334-2393 **中国 - 深圳** Tel: 86-755-8864-2200 Fax: 86-755-8203-1760 **中国 - 武漢** Tel: 86-27-5980-5300 Fax: 86-27-5980-5118 **中国 - 西安** Tel: 86-29-8833-7252 Fax: 86-29-8833-7256	**中国 - 廈門** Tel: 86-592-2388138 Fax: 86-592-2388130 **中国 - 珠海** Tel: 86-756-3210040 Fax: 86-756-3210049 **ｲﾝﾄﾞ - ﾊﾝｶﾞﾛｰﾙ** Tel: 91-80-3090-4444 Fax: 91-80-3090-4123 **ｲﾝﾄﾞ - ﾆｭｰﾃﾞﾘｰ** Tel: 91-11-4160-8631 Fax: 91-11-4160-8632 **ｲﾝﾄﾞ - ﾌﾟﾈｰ** Tel: 91-20-3019-1500 **日本 - 大阪** Tel: 81-6-6152-7160 Fax: 81-6-6152-9310 **日本 - 東京** Tel: 81-3-6880-3770 Fax: 81-3-6880-3771 **韓国 - 大邱** Tel: 82-53-744-4301 Fax: 82-53-744-4302 **韓国 - ｿｳﾙ** Tel: 82-2-554-7200 Fax: 82-2-558-5932 or 82-2-558-5934 **ﾏﾚｰｼｱ - ｸｱﾗﾙﾝﾌﾟｰﾙ** Tel: 60-3-6201-9857 Fax: 60-3-6201-9859 **ﾏﾚｰｼｱ - ﾍﾟﾅﾝ** Tel: 60-4-227-8870 Fax: 60-4-227-4068 **ﾌｨﾘﾋﾟﾝ - ﾏﾆﾗ** Tel: 63-2-634-9065 Fax: 63-2-634-9069 **ｼﾝｶﾞﾎﾟｰﾙ** Tel: 65-6334-8870 Fax: 65-6334-8850 **台湾 - 新竹** Tel: 886-3-5778-366 Fax: 886-3-5770-955 **台湾 - 高雄** Tel: 886-7-213-7830 **台湾 - 台北** Tel: 886-2-2508-8600 Fax: 886-2-2508-0102 **ﾀｲ - ﾊﾞﾝｺｸ** Tel: 66-2-694-1351 Fax: 66-2-694-1350	**ｵｰｽﾄﾘｱ - ｳﾞｪﾙｽ** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **ﾃﾞﾝﾏｰｸ - ｺﾍﾟﾝﾊｰｹﾞﾝ** Tel: 45-4450-2828 Fax: 45-4485-2829 **ﾌｨﾝﾗﾝﾄﾞ - ｴｽﾎﾟｰ** Tel: 358-9-4520-820 **ﾌﾗﾝｽ - ﾊﾟﾘ** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **ﾌﾗﾝｽ - ｻﾝ ｸﾙｰ** Tel: 33-1-30-60-70-00 **ﾄﾞｲﾂ - ｶﾞﾙﾋﾝｸﾞ** Tel: 49-8931-9700 **ﾄﾞｲﾂ - ﾊｰﾝ** Tel: 49-2129-3766400 **ﾄﾞｲﾂ - ﾊｲﾙﾌﾞﾛﾝ** Tel: 49-7131-67-3636 **ﾄﾞｲﾂ - ｶｰﾙｽﾙｰｴ** Tel: 49-721-625370 **ﾄﾞｲﾂ - ﾐｭﾝﾍﾝ** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **ﾄﾞｲﾂ - ﾛｰｾﾞﾝﾊｲﾑ** Tel: 49-8031-354-560 **ｲｽﾗｴﾙ - ﾗｰﾅﾅ** Tel: 972-9-744-7705 **ｲﾀﾘｱ - ﾐﾗﾉ** Tel: 39-0331-742611 Fax: 39-0331-466781 **ｲﾀﾘｱ - ﾊﾟﾄﾞﾊﾞ** Tel: 39-049-7625286 **ｵﾗﾝﾀﾞ - ﾃﾞﾙｰﾈﾝ** Tel: 31-416-690399 Fax: 31-416-690340 **ﾉﾙｳｪｰ - ﾄﾛﾝﾊｲﾑ** Tel: 47-7289-7561 **ﾎﾟｰﾗﾝﾄﾞ - ﾜﾙｼｬﾜ** Tel: 48-22-3325737 **ﾙｰﾏﾆｱ - ﾌﾞｶﾚｽﾄ** Tel: 40-21-407-87-50 **ｽﾍﾟｲﾝ - ﾏﾄﾞﾘｰﾄﾞ** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **ｽｳｪｰﾃﾞﾝ - ｲｪｰﾃﾎﾞﾘ** Tel: 46-31-704-60-40 **ｽｳｪｰﾃﾞﾝ - ｽﾄｯｸﾎﾙﾑ** Tel: 46-8-5090-4654 **ｲｷﾞﾘｽ - ｳｫｰｷﾝｶﾞﾑ** Tel: 44-118-921-5800 Fax: 44-118-921-5820