AN2462

AVR231 : AES(新暗号化規格)ブートローダ

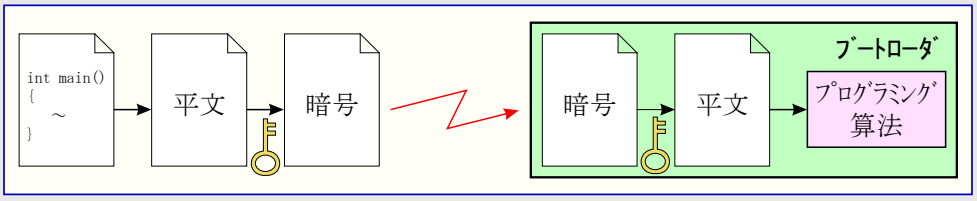
要点

- ブートローダ能力と最低1KバイトのSRAMを持つAVRマイクロ コントローラに適合
- AVRに基づく応用へファームウェアと重要なデータの安全な転送を許す
- 簡単な使用と構成設定可能な応用例を包含
  - 2進ファイルとデータの暗号化
  - 目的対象ブートローダ作成
  - 目的対象への暗号化ファイルのダウンロード
- 新暗号化規格(AES)実装
  - 128, 192, 256ビット鍵
- 2Kバイト内に合うAESブートローダ
- 64Kバイト応用、115200ボー、3.69MHz目的対象周波数での代表的な更新時間
  - AES128：27秒
  - AES192：30秒
  - AES256：33秒
- 応用はATmega328PB Xplained Miniで直ぐに使って評価することができます。
- ファームウェアは最小変更または変更なしで以下のデバイスで動作することを試験されています。
  - ATmega8/16/162/169/32/64/128/2S56
  - ATmega168PB
  - ATmega328PB

序説

この応用記述はブートローダ能力を持つAVR®マイクロ コントローラでファームウェアがどう安全に更新し得るかを記述します。その方法はファームウェアを暗号化するのに新暗号化規格(AES)を使います。

図1. 概要



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、Microchip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

要点 ··· 1
序説 ··· 1
1. 説明 ··· 3
2. 用語 ··· 4
3. 事前必要条件 ··· 4
4. 暗号概要 ··· 4
4.1. 暗号化 ··· 4
4.2. 解読 ··· 4
5. AES概要 ··· 4
5.1. AES実装 ··· 4
5.2. AES暗号化 ··· 6
5.3. AES解読 ··· 7
5.4. 鍵拡張 ··· 8
5.5. 暗号塊連鎖(CBC) ··· 8
6. ソフトウェア実装と使い方 ··· 9
6.1. 動機付け ··· 9
6.2. 使い方概要 ··· 9
6.3. 構成設定ファイル ··· 10
6.4. PC応用 – GenTemp ··· 10
6.5. PC応用 – Create ··· 11
6.6. PC応用 – Update ··· 14
7. ハードウェア構成設定 ··· 15
7.1. ATmega328PB Xplained Miniキット接続 ··· 15
7.2. プログラミングとデバッグ ··· 16
8. AVRブートローダ ··· 17
8.1. 鍵とヘッダ ファイル ··· 17
8.2. プロジェクト ファイル ··· 17
8.3. Atmel StudioとIARの設定 ··· 18
8.4. ブートローダのインストール ··· 23
8.5. 性能 ··· 23
9. 要約 ··· 24
10. Atmel | STARTからのソース コード取得 ··· 24
11. 参考 ··· 25
12. 改訂履歴 ··· 25
Microchipウェブ サイト ··· 26
お客様への変更通知サービス ··· 26
お客様支援 ··· 26
Microchipデバイス コード保護機能 ··· 26
法的通知 ··· 26
商標 ··· 27
DNVによって認証された品質管理システム ··· 27
世界的な販売とサービス ··· 28

1. 説明

マイクロ コントローラを含む電気設計は可搬型音楽再生機、ヘアドライヤ、ミシンなどでファームウェアを装備されることが常に必要です。多くの電気設計が急速に発達するため、既に出荷または販売されてしまった製品を更新することができる必要が増大しています。特に製品が既に最終顧客へ届いてしまっている場合、ハードウェアに変更を行うのは難しくなると言ってよいでしょうが、しかしAVRのようなフラッシュ マイクロ コントローラに基づく製品でファーム ウェアは容易に更新することができます。

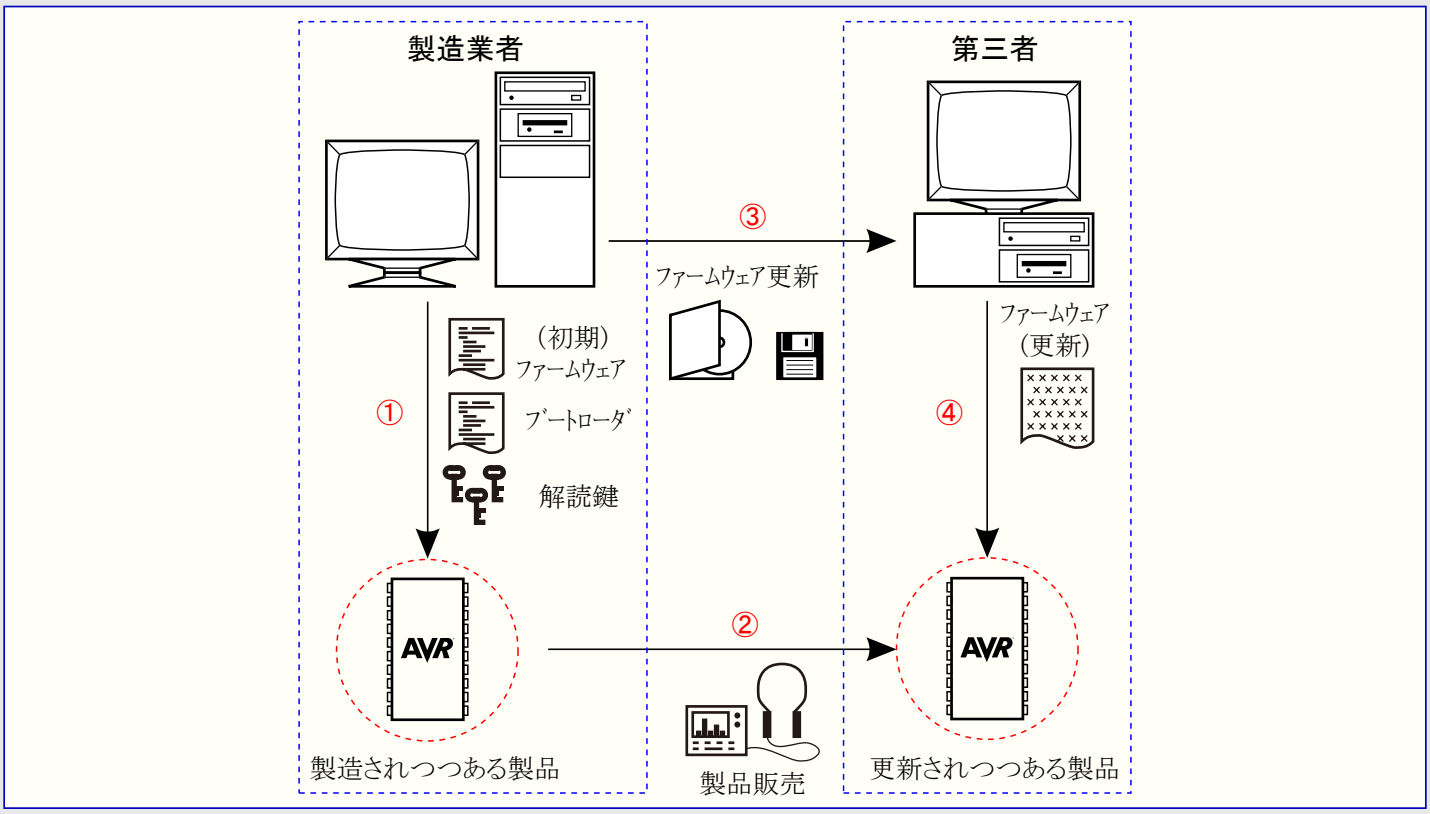
多くのAVRマイクロ コントローラは要求あり次第、ファームウェア更新を受け取ってフラッシュ メモリを書き換えることができるブートローダを作ることが可能なように構成設定されます。プログラム メモリ空間はブートローダ領域(BLS)と応用領域の2つの領域に分けられます。両領域はブートローダのコードがBLSで保護され得ると同時に未だ応用領域でコードを更新できるような読み書き保護用の専用施錠ビットを持ちます。従って、BLSの更新方法は外側のアクセスに備えて簡単に保護することができます。

ファームウェアに問題が残っており、それは代表的に、フラッシュ メモリに書かれて施錠ビットが設定される前に保護されないことです。これはファームウェアが現場で更新されることが必要な場合にそれが書き込み台または製造業者の建物を出る瞬間から不正アクセスに関して開いていることを意味します。

この応用記述はフラッシュ メモリとEEPROMへ転送されるべきデータが暗号法を用いることによって何時でも如何にして保護され得るかを示します。この考えは書き込み台を去る前にデータを暗号化して目的対象AVRにダウンロードされた後でだけそれを解読することです。この手順はファームウェアの不正な複製を防げませんが、暗号化した情報は正しい解読鍵なしでは実質的に無効です。解読鍵は書き込み環境の外側、目的対象AVRの内側の1つの位置にだけ格納されます。鍵は暗号化されたデータから再生成することができません。データへのアクセスを得る唯一の方法は正しい鍵を使うことによるだけです。

以下の図は製品がどのように最初に製造され、初期ファームウェアを設定され、販売されて、そして後で新版のファームウェアで更新される例を示します。

図1-1. 代表的な製品と更新手順の例



①. 製造中、マイクロ コントローラは最初にブートローダ、解読鍵、応用ファームウェアを装備されます。ブートローダは実際の応用とプログラミングを受け取ってそれをフラッシュ メモリ内にプログラミングする準備をし、同時に鍵はやって来るデータの解読に必要です。施錠ビットはAVR内側のファームウェアを保護するように設定されます。

②. 製品はその後に代理店へ出荷されるか、または最終的な顧客に販売されます。施錠ビット設定はAVR内側で保護されたファームウェアを維持し続けます。

③. ファームウェアの新規公開が完了され、既に配給されてしまった製品の更新の必要があります。ファームウェアは従って暗号化されて代理店に出荷されます。暗号化されたファームウェアは解読鍵なしで用を足さず、従ってソフトウェアの局所的な複製(例えば、代理店のハードドライブ)は安全性を危険に晒しません。

④. 代理店は在庫の全ての構成単位と顧客によって返された(例えば、修理中の)それらを更新します。暗号化されたファームウェアはAVRにダウンロードされ、マイクロ コントローラの内側で一度解読されます。施錠ビット設定はAVRの内側で更新されたファームウェアの保護を維持し続けます。

2. 用語

BLS	ﾌﾞｰﾄﾛｰﾀﾞ領域 (Bootloader Section)
EEPROM	電気的消去可能で書き込み可能な読み込み専用ﾒﾓﾘ (Electrically Erasable Programmable Read-Only Memory)
AES	新暗号規格 (Advanced Encryption Standard)
RAM	乱ｱｸｾｽ ﾒﾓﾘ (Random Access Memory)
CBC	暗号塊連鎖 (Clipher Block Chaining)
PC	個人用ｺﾝﾋﾟｭｰﾀ (Personal Computer)
CRC	巡回冗長符号検査 (Cyclic Redundancy Check)
USART	万能同期非同期送受信器 (Universal Synchronous Asynchronous Receiver Transmitter)
KB	ｷﾛ ﾊﾞｲﾄ (Kirobytes)
LCD	液晶表示器 (Liquid Crystal Display)
IDE	統合開発環境 (Integrated Development Envieonment)

3. 事前必要条件

この資料で検討される解決策は以下の基本的な熟知が必要です。

- Atmel® Studio 7またはそれ以降
- ATmega328PB Xplained Mini
- AESの概念
- ﾌﾞｰﾄﾛｰﾀﾞの疑念とAVRでのそれの実装

この応用記述は新暗号化規格(AES:Advanced Encryption Standard)の基本的な概要を網羅します。AESの概念のより良い理解を持ちたい使用者は関連文献で更なる読書を行うことが必要とされます。

4. 暗号概要

用語の暗号は情報が鍵を用いて施錠されて利用不能にされる時に用いられます。開錠情報は正しい鍵の使用でだけ達成することができます。

暗号鍵に基づく算法は対称と非対称の2つに分けられます。対称法は暗号化と解読に対して同じ鍵を用い、一方非対称法は異なる鍵を使います。AESは対称鍵法です。

4.1. 暗号化

暗号化はそれの内容が部外者から隠されるようにﾒｯｾｰｼﾞやﾃﾞｰﾀを符号化する方法です。その元の形式に於いて平文のﾒｯｾｰｼﾞやﾃﾞｰﾀはﾏｲｸﾛ ｺﾝﾄﾛｰﾗのﾌｧｰﾑｳｪｱのように作者や配給者が秘密維持を望む情報を含むかもしれません。例えば、ﾏｲｸﾛ ｺﾝﾄﾛｰﾗが現場で更新される時に、不正な複製の試みや逆行分析に対してﾌｧｰﾑｳｪｱを保護することが難しくなるかもしれません。ﾌｧｰﾑｳｪｱの暗号化はそれが解読されるまでそれを使えなくします。

4.2. 解読

解読は元のﾒｯｾｰｼﾞやﾃﾞｰﾀを取り戻す方法で、代表的に正しい鍵を知ることなく実行することができません。鍵はﾃﾞﾊﾞｲｽが暗号化したﾃﾞｰﾀを受け取ってそれを解読し、ﾌﾗｯｼｭ ﾒﾓﾘまたはEEPROMの選択した部分を書き換えることができるように、ﾏｲｸﾛ ｺﾝﾄﾛｰﾗのﾌﾞｰﾄﾛｰﾀﾞ内に格納することができます。解読鍵は暗号化したﾃﾞｰﾀから取り戻すことができず、それ故に施錠ﾋﾞｯﾄがﾌﾟﾛｸﾞﾗﾑ(0)されてしまっている場合、AVRﾏｲｸﾛ ｺﾝﾄﾛｰﾗから読むことができません。

5. AES概要

5.1. AES実装

本項はAES算法やその歴史の詳細な記述を意図していません。むしろその意図はこの算法の様々な部分に関してAVR特有実装を記述することです。組み込み応用でﾒﾓﾘは不足しがちな資源なので、焦点はｺｰﾄﾞ ﾒﾓﾘの節約にあります。ﾌﾞｰﾄﾛｰﾀﾞ応用は決して主ｺｰﾄﾞと同時に走行せず、従ってﾃﾞｰﾀ ﾒﾓﾘ必要条件がﾏｲｸﾛ ｺﾝﾄﾛｰﾗの容量を超えない限り、ﾃﾞｰﾀ ﾒﾓﾘ(SRAM)の節約は重要ではありません。

後続する副項に於いて、いくつかの基本算術演算とそれらのAVR特有実装が記述されます。数学から有限場理論へいくつかの参照があることに注意してください。この資料を読むのに有限場の知識は必要ありませんが、興味を持った読者はAES仕様を学ぶべきです。

注: 読者がAESの実装の充分な知識を持つ場合、継続性を失うことなく「6. ｿﾌﾄｳｪｱ実装と使い方」に飛ぶことができます。

5.1.1. ﾊﾞｲﾄ加算

AES算法に於いて、ﾊﾞｲﾄ加算はｷｬﾘｰ伝播なしの個別ﾋﾞｯﾄの加算として定義されます。これは一般的なXOR操作が意図されます。XOR操作はそれ自身の逆、従ってﾊﾞｲﾄ減算はAES算法での加算と同等です。XOR操作はAVRで実装するのに取るに足らない問題です。

5.1.2. ﾊﾞｲﾄ乗算

AES算法に於いて、ﾊﾞｲﾄ乗算は法$11B(2進数:1 0001 1011)での有限場乗算として定義されます。提案された実装は最初の因数を2で繰り返し乗算(法$11B)して、値1を持つ2つ目の因数の各ﾋﾞｯﾄに対する中間結果を積和します。

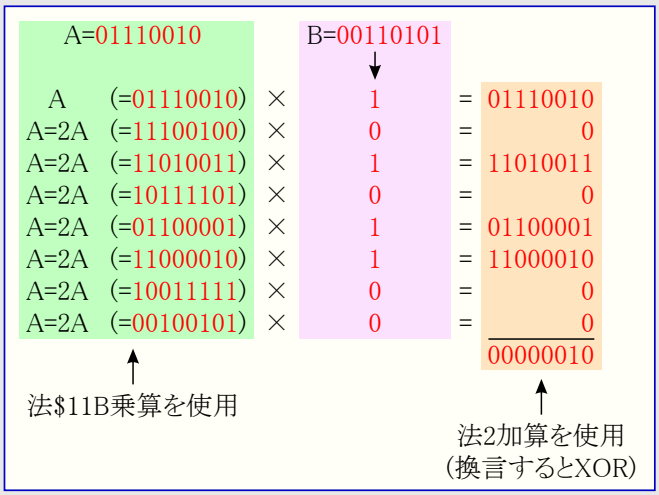
例:2つ目の因数が$1A(2進数: 0001 1010)の場合、その後に最初、3つ目、5つ目の中間結果が積和されるべきです。

別の例が右図で示されます。この方法は少しのﾒﾓﾘしか使わず、8ﾋﾞｯﾄ ﾏｲｸﾛ ｺﾝﾄﾛｰﾗに上手く適合します。

乗算算法は以下の擬似ｺｰﾄﾞによって記述することができます。

```
bitmask = 1
tempresult = 0
tempfactor = firstfactor
while bitmask < 0x100
    if bitmask AND secondfactor <> 0
        add tempfactor to tempresult using XOR
    end if
    shift bitmask left once
    multiply tempfactor by 2 modulo 0x11B
    end while
return tempresult
```

図5-1. ﾊﾞｲﾄ乗算



5.1.3. 乗法の逆元

有限場乗法の逆元、換言すると1/xを計算し得るために、この実装ではごまかしが使われます。共通の低で指数と対数を用いて、以下の恒等式を利用することができます。

1/xを計算するための指数と対数の使用

$$a^{-log_a x} = \frac{1}{x}$$

それが分解できない最も基本の根であるため、この場合は低の値は3が選ばれています。指数と対数を計算する時に有限場乗算を用いることにより、乗法の逆元は実装が容易です。毎回、指数と対数を計算する代わりに、2つの参照表が使われます。乗法の逆元は「5.1.4. Sﾎﾞｯｸｽ」で記述されるSﾎﾞｯｸｽを準備する時にだけ使われるので、2つの参照表に使われるﾒﾓﾘはSﾎﾞｯｸｽが準備されてしまっている時に他の目的に使うことができます。

参照表計算は以下の擬似ｺｰﾄﾞによって記述することができます。

```
tempexp = 0
tempnum = 1
do
    exponentiation_table[ tempexp ] = tempnum
    logarithm_table[ tempnum ] = tempexp
    increase tempexp
    multiply tempnum by 3 modulo 0x11B
loop while tempexp < 256
```

5.1.4. Sﾎﾞｯｸｽ

AES算法は置換表またはSﾎﾞｯｸｽの概念を使います。この算法の段階の1つは可逆変換をﾊﾞｲﾄに適用することです。Sﾎﾞｯｸｽは可能な全てのﾊﾞｲﾄ値に関して予め計算されたこの変換の結果です。この変換は次の2つの段階から成り、(1)「5.1.3. 乗法の逆元」で記述されるような乗法の逆元と、(2)以下の式に従う線形変換、ここでのa_iは結果のﾋﾞｯﾄ、b_iは段階1からの結果のﾋﾞｯﾄです。

より徹底的な行列の考察は、右手のﾍﾞｸﾄﾙと元のﾊﾞｲﾄの1,2,3,4回の左回転と、元のﾊﾞｲﾄの(XOR加算を用いた)総和として実装することができる操作を明らかにします。この方法は8ﾋﾞｯﾄ ﾏｲｸﾛ ｺﾝﾄﾛｰﾗに上手く適合します。

解読に使われる逆元Sﾎﾞｯｸｽは同様の構造を持ち、また、XOR加算と回転を用いて実装されます。対応する行列についてはAES仕様を、実装の詳細についてはｿｰｽ ｺｰﾄﾞを参照してください。

Sﾎﾞｯｸｽで使用される線形変換

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} = \begin{bmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

5.1.5. '状態'

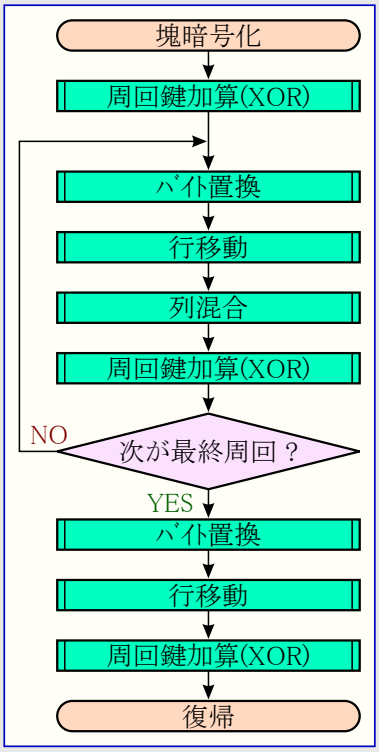
AES算法は塊暗号で、それはﾃﾞｰﾀが塊で管理されることを意味します。AES暗号についてはこの塊容量が16ﾊﾞｲﾄです。このAES塊は度々'状態'または'状態配列'と呼ばれる4×4の配列で構成されます。この状態の最も左側の列は上から下へ塊の最初の4ﾊﾞｲﾄを保持し、以下同様です。読者はAES仕様で4つの連続するﾊﾞｲﾄが語として参照されることにも注意すべきです。

5.2. AES暗号化

暗号化処理の段階を検討する前に、'暗号化周回'の導入が必要です。殆どの塊暗号は何回かの繰り返しで実行される少しの操作から成ります。各繰り返し反復は異なる暗号鍵を用います。各反復内の操作の最低1つでは鍵に依存します。繰り返し反復は暗号化周回として参照され、この周回に使われる鍵の系列が鍵計画と呼ばれます。周回数は鍵の大きさに依存します。

暗号化処理用の流れ図は右図で示されます。以下の副項は処理に於ける各種段階を説明します。各段階は利便性のためにｻﾌﾞﾙｰﾁﾝとして実装されます。ｺﾝﾊﾟｲﾗの最適化の使用はｺｰﾄﾞ ﾒﾓﾘを節約するために不要な関数呼び出しを取り去ります。

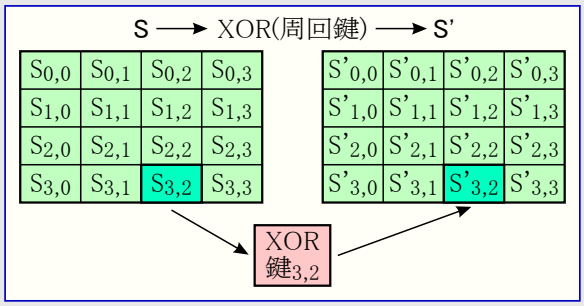
図5-2. 暗号化流れ図



5.2.1. 周回鍵加算

この段階は現在の周回鍵を現在の状態配列に加算するためにXOR加算を使います。周回鍵は状態と同じ大きさ、換言すると16ﾊﾞｲﾄまたは4語を持ちます。この操作は16回の繰り返しとして実装されます。

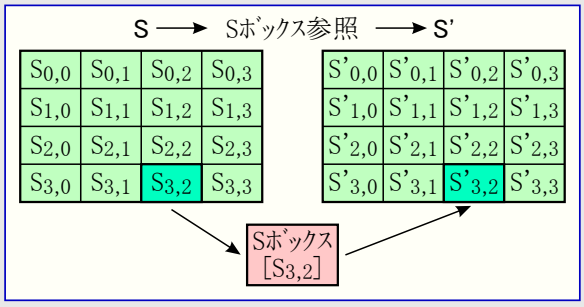
図5-3. 現状態への周回鍵加算



5.2.2. ﾊﾞｲﾄ置換

この段階は状態内でﾊﾞｲﾄを置換するために予め計算されたSﾎﾞｯｸｽ表を使います。この段階も前項のように16回の繰り返しとして実装されます。

図5-4. 現状態のﾊﾞｲﾄ置換

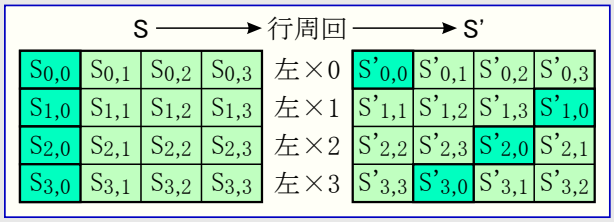


5.2.3. 行移動

この段階は現在の状態の行で操作します。最初の行は元のままで、一方後の3つは各々1、2、3回左に周回されます。1度の左周回には最も左側のﾊﾞｲﾄが最も右側の列に移動され、残りの3つのﾊﾞｲﾄは左に1列移動されます。この処理は右図で示されます。

素直な実装は行を左に1回周回するｻﾌﾞﾙｰﾁﾝを書き、そして各行で必要とされる回数、それを呼ぶことです。けれども、考査はどんな繰り返しやｻﾌﾞﾙｰﾁﾝもなしにﾊﾞｲﾄを直接的に混ぜる実装を示し、ｺｰﾄﾞ量での小さな不利だけなく、速度での重要な利益(3倍)に帰着します。従って直接実行が選ばれます。詳細についてはｿｰｽ ｺｰﾄﾞ内のShiftRows()関数を参照してください。

図5-5. 現状態の行を周回



5.2.4. 列混合

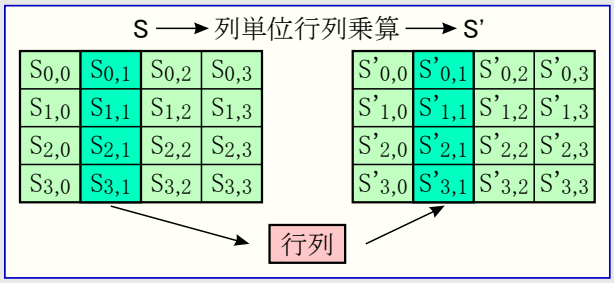
この段階は列単位で状態を操作します。各列はﾊﾞｲﾄのﾍﾞｸﾄﾙとして扱われ、変更した状態に対する列を得るために固定化した行列によって乗算されます。

この操作は混合された列のﾊﾞｲﾄ(a_i)と元の列のﾊﾞｲﾄ(b_i)である以下の式によって記述することができます。

1列混合時の行列乗算

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

図5-6. 現状態の列混合



この段階はどんな二次的関数呼び出しもなしに直接的に実装されます。行列式から混合された列の全てのﾊﾞｲﾄa_iは元のﾊﾞｲﾄb_iとそれらの2倍$2b_i$の組み組み合わせと見ることができます。詳細についてはｿｰｽ ｺｰﾄﾞ内のMixColumns()関数を参照してください。

注: 「5.1.1. ﾊﾞｲﾄ加算」と「5.1.2. ﾊﾞｲﾄ乗算」からのXOR加算と有限場乗算が使われます。

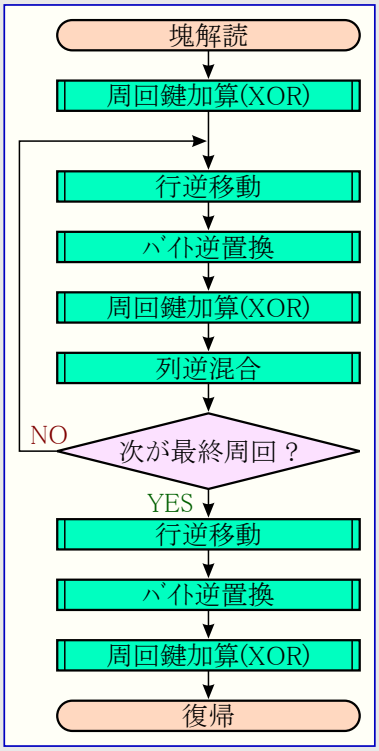
5.3. AES解読

この処理は段階の順番が変更されていることを除き、暗号化処理と非常に似ています。"周回鍵加算"を除く全ての段階は対応するそれらの逆元を持ちます。"行逆移動"は左の代わりに行を右に周回します。"ﾊﾞｲﾄ逆置換"は逆元Sﾎﾞｯｸｽを使います。

"列逆混合"も逆元変換を使います。対応する行列についてはAES仕様を、実装の詳細についてはｿｰｽ ｺｰﾄﾞを参照してください。解読処理用流れ図は右のように示されます。

注: 解読に使われる鍵計画は暗号化と同じですが、逆順です。

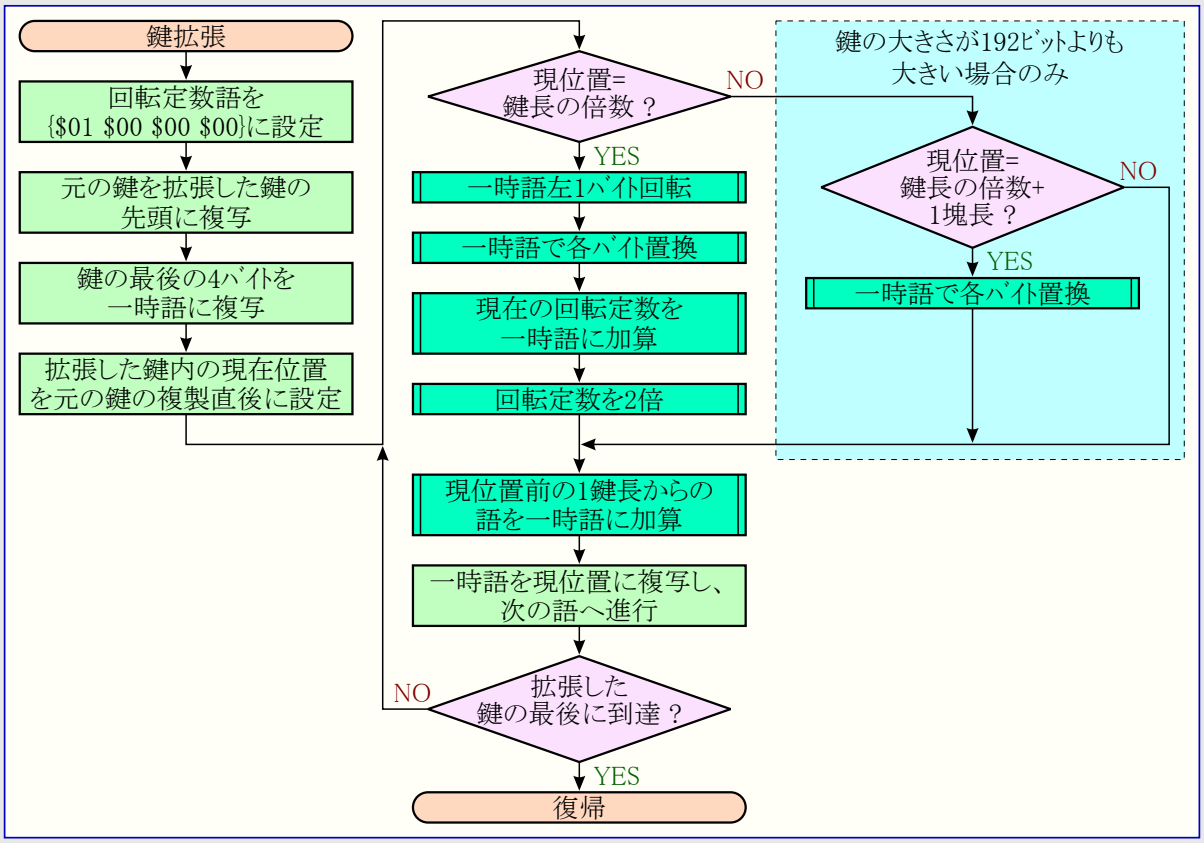
図5-7. 解読流れ図



5.4. 鍵拡張

鍵拡張は元の128,196,256ﾋﾞｯﾄ暗号鍵から鍵計画を生成する処理です。鍵拡張用の流れ図は次のように示されます。

図5-8. 鍵拡張流れ図



この算法はXOR加算、有限場乗算、置換、語周回のような既に記述した操作を用います。詳細についてはｿｰｽ ｺｰﾄﾞを参照してください。

注: 鍵拡張が暗号化と解読の両方に関して同一です。従って例え解読だけが使われる場合でも、暗号化に使われるSﾎﾞｯｸｽが必要とされます。AVR実装では、鍵拡張に先立って通常のSﾎﾞｯｸｽが計算され、それのﾒﾓﾘはその後に逆元Sﾎﾞｯｸｽを計算する時に再使用されます。

5.5. 暗号塊連鎖(CBC:Clipher Block Chaining)

AESは塊暗号で、この算法は固定量のﾃﾞｰﾀ塊で操作することを意味します。ﾃﾞｰﾀを16ﾊﾞｲﾄの塊に暗号化するのに暗号鍵が使われます。既知の入力塊と一定(例え未知でも)の暗号鍵に関して、その出力は常に同じです。これは暗号ｼｽﾃﾑの攻撃を欲する誰かにとって有用な情報を提供するかもしれません。

同じ平文の塊を違う暗号文の塊に暗号化させるのに一般的に使われる方法があります。そのような方法の1つは暗号塊連鎖(CBC)と呼ばれます。

CBCは先行塊が全ての後行塊に影響を及ぼすような暗号塊を接続する方法です。現在の平文の塊と直前の暗号文の塊で最初にXOR操作を実行することによって成し遂げられます。XORの結果はその後に平文の塊の代わりに暗号化されます。これは1つの暗号文ﾋﾞｯﾄが依存するところの平文ﾋﾞｯﾄ数を増します。

6. ソフトウェア実装と使い方

本章は系の安全性を改善するためのいくつかの重要な項目を最初に検討します。これらの項目は後のソフトウェア設計に於いて多くの決定に動機を与えます。

6.1. 動機付け

この応用記述は外側のアクセスから設計を保護する時に使うことができる技法を提供します。例え完全に保護することができる設計がないとは言え、それは安全性を破るために必要とされる労力が可能な限り高くなるように構成することができます。基本工学的な技能の人が複製することができる安全でない設計と、少数で非常に熟練した侵入者だけが破ることができる設計の間には重大な違いがあります。安全にされていない場合では、設計が容易に複製され、そして製造業者の知的所有権に違反して逆行分析さえもされ、そしてその設計に関する潜在的な市場をも危うくします。安全にされている場合では、設計を破るために必要とされる労力が大きいので、殆どの侵入者は単に彼ら自身の製品を開発することに集中します。

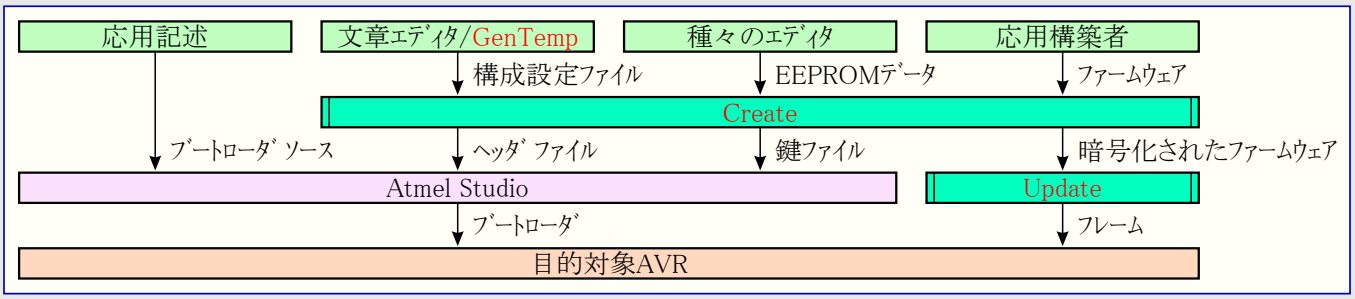
安全なシステム構築法には、可能な限り破るのを難しくするように設計すべきであると言う、一般的な1つの規則だけがあります。破ろうとする間に、保護を欺くのに使うことができるどの手法も試みられるでしょう。考慮されなければならない少しの例が以下で与えられます。

- ファームウェア更新中に電力が取り去られたなら、何が起きますか?。電力が戻って回復した時にマイクロ コントローラの状態はどうですか?。施錠ビットとリセット ベクタは何時も正しく設定されていますか?。
- 平文データのように見える何かを作ることができる何れかの仮定がありますか?。AESに関して総当り法によって破られるには探す様式があるに違いありません。攻撃者は単に全ての鍵の組み合わせを試み、理解できるファームウェアを探してその出力を監視する筈がありません。攻撃ソフトウェアはプログラム メモリの開始での割り込みベクタ、0または1で穴埋めされたメモリ領域、以下同様のような既知の様式を検索するように構成設定されなければなりません。
- 解読処理から得ることができる何れかの還元がありますか?。このようなどの還元も攻撃者を手助けし得ます。例えば、ブートローダ内側の解読算法が各塊処理に対して良/否形式の合図を与える場合、この信号は攻撃者への還元として使われ得ます。
- 符号化したフレームは別の順で送られるべきですか?。ブートローダに送られる最初のフレームが常に暗号化したファイルの最初の塊を含む場合、攻撃者はこれからいくつかの仮定を作ることができます。例えば、最初のフレームはアドレス0から始まるプログラム データの割付でそれは割り込みベクタ表を含むと仮定することができます。この情報は鍵検索を細かく区別する攻撃者を助けます。システムの安全を増すため、乱順でフレームを送ってください(どうせ、解読されたフレームはそれらの正しいアドレスに割付されます)。

6.2. 使い方概要

これと以下のサブルーチンは応用の構成設定と使い方を記述します。この処理は以下の図で図解されます。

図6-1. プロジェクトの流れの概要



主な段階は次のとおりです。

- 目的対象AVR用の応用を作成してください。必要ならば、独立したファイルでEEPROM設計(配置)を作成してください。
- プロジェクト依存情報で構成設定ファイルを作成してください。GenTempと呼ばれる応用がファイル フレーム作成に使えます。
- Createと呼ばれる応用を走らせてください。これはヘッダ ファイル、鍵ファイル、暗号化されたファイルを作成します。
- Atmel Studio 7またはそれ以降を使って、目的対象AVR用のブートローダを構成設定して構築してください。
- 目的対象AVRにブートローダをダウンロード(プログラミング)して施錠ビットとヒューズ ビットを設定してください。
- 今や暗号化したファームウェアは何時でもAVRにダウンロード(書くことが)できます。

この手続きは以下でもっと詳細に検討されます。

6.3. 構成設定ファイル

構成設定ファイルはプロジェクトを構成設定するのに使われるパラメータの一覧を含みます。パラメータは以下の表で記述されます。

表6-1. 構成設定ファイル任意選択の要約

パラメータ	説明	既定	必須
PAGE_SIZE	AVRフラッシュ ページの10進バイトでの大きさ。この値はデバイス依存です、データシートをご覧ください。	N/A	はい
KEY1	16進での暗号鍵の最初の(128ビット)部分。第8ビット後に挿入される奇数パリティ ビットを持つ16の乱数バイトであるべきで、合計18バイトを作成します。	なし: 暗号なし	いいえ、しかし強く推奨
KEY2	16進での暗号鍵の2つ目の(64ビット)部分。第8ビット後に挿入される奇数パリティ ビットを持つ8つの乱数バイトであるべきで、合計9バイトを作成します。省略されたなら、AES128が使用されます。	なし: AES128使用	いいえ、しかし推奨
KEY3	16進での暗号鍵の2つ目の(64ビット)部分。第8ビット後に挿入される奇数パリティ ビットを持つ8つの乱数バイトであるべきで、合計9バイトを作成します。省略されたなら、AES128またはAES192が使用されます。	なし: AES128または AES192を使用	いいえ、しかし推奨
INITIAL_VECTOR	暗号塊連鎖に使用。16進での16の乱数バイトであるべきです。	0	いいえ、しかし強く推奨
SIGNATURE	16進でのフレーム確認データ。これはどの4バイトにもできますが、乱数で選ばれた値が推奨されます。	00	いいえ
ENABLE_CRC	CRC検査許可。YesまたはNo。許可されたなら、全ての応用領域は上書きされ、応用はそれが開始を許される前にCRC検査を通らなければなりません。	なし	いいえ、しかし推奨
MEM_SIZE	目的対象AVR内の応用領域の大きさ(10進バイトでの)。	N/A	はい、CRC使用の場合

構成設定ファイルはどの有効なファイル名も与えることができます。この名前はプロジェクト ファイルを作成する応用へのパラメータとして後で与えられます。下はATmega328PB用の試供構成設定ファイルです。KEY1パラメータは挿入されたパリティ ビットを持つ128ビット鍵(16進の$0123456789ABCDEF0123456789ABCDEF)の例です。

```
PAGE_SIZE       = 128
MEM_SIZE        = 30720
CRC_ENABLE      = YES
KEY1            = EC38ECC4C11DBC16151A5E346A872AADAD36
INITIAL_VECTOR  = F24D994D5DD3E9F1EEE897616C425028
SIGNATURE       = 89DBF334
```

パラメータのいくつかは目的対象AVRの特定の知識なしに設定することができません。以下の表はブートローダ機能を持つ存在するいくつかのAVRマイクロ コントローラの特性を要約します。この表に存在しないデバイスについてはデバイスのデータシートを参照してください。

表6-2. AVR特性要約

特性	mega8	mega16	mega162	mega169	mega32	mega328PB	mega64	mega128
フラッシュ容量(バイト)	8K	16K			32K		64K	128K
フラッシュ ページ容量(バイト)	64	128					256	
フラッシュ ページ数	128				256			512
(最大)BLS容量(バイト)	2048				4096		8192	
BLSページ数	32	16			32			
MEM_SIZE(バイト)	6144	14336			28672		57344	122880
PAGE_SIZE(バイト)	64	128						256

6.4. PC応用 – GenTemp

この応用は構成設定ファイル用の雛形を生成します。この応用は乱数暗号鍵を生成し、ベクタを初期化し、(フラッシュ ページ容量のように)満たされるべき使用者用の他のパラメータをそのままにします。常にこの応用を用いる雛形生成で始めることが推奨されます。

この応用は次のように使われます。

```
gentemp FileName.ext
```

ここでFileName.extは作成されるべき構成設定ファイルの名前です。ファイルが生成された後で選んだ何れかの平文エディタを用いて編集することができます。

6.5. PC応用 – Create

この応用は構成設定ファイルから情報を読んで、ブートローダ用の鍵とヘッダ ファイルを生成します。これはファームウェアの暗号化にも使われます。代表的に、この応用は以下のように最低2回走行します。

1. ブートローダ用の鍵とヘッダ ファイルを生成するため
2. 新しいファームウェアが暗号化される時

注: プロジェクト ファイルの生成とファームウェアの暗号化の時に同じ暗号情報(構成設定ファイル)が使われることが非常に重要です。さもなければ、ブートローダは正しい暗号鍵の組を持たずにデータを解読できないかもしれません。暗号化されたファームウェアを解読するのに構成設定ファイル内の情報を用いることが可能なことにも注意すべきです。従って、構成設定ファイルは何時も安全を保たれなければならず、最初に使われた後で変更されるべきではありません。

6.5.1. 命令行引数

次表は利用可能な命令行引数を示します。

表6-3. 命令行引数の要約

引数	説明
-c <ファイル名.ext>	構成設定ファイルへのパス
-d	設定なら、各フラッシュ ページの内容は書く前に削除されます。そうでなく、特に書かれなければ直前のデータが保たれます。
-e <ファイル名.ext>	EEPROMファイル(EEPROM内容になるデータ)へのパス
-f <ファイル名.ext>	フラッシュ ファイル(応用領域内容になるコード)へのパス
-h <ファイル名.ext>	出力ヘッダ ファイル名。このファイルは後にブートローダでインクルードされなければなりません。
-k <ファイル名.ext>	出力鍵ファイル名。このファイルは後にブートローダでインクルードされなければなりません。
-l [BLB12][BLB11][BLB02][BLB01]	設定する施錠ビット。これらの施錠ビットは全データが転送された後で、制御が更新された応用に渡される前に設定されます。
-n	不感知。暗号化されたファイルに不感知記録の乱数を追加します。不感知記録がブートローダによって無視されるため、この設定は出力ファイルの予測性だけで、応用に影響を及ぼしません。
-o <ファイル名.ext>	出力ファイル名。これは配布され得る暗号化したファイルで、更新が必要な時に目的対象へ送られます。

6.5.2. 初回走行

最初の走行では代表的にブートローダ用の鍵とヘッダ ファイルが生成されるだけです。鍵とヘッダ ファイルの生成は命令行引数を用いて要求されます。

例: `Create -c Config.txt -h BootLdr.h -k AESKeys.inc`

鍵とヘッダ ファイルはブートローダ応用のプロジェクト フォルダに複写され、ブートローダ コード内にインクルードされなければなりません。

注: ブートローダ プロジェクト ファイル名は上で言及したファイル名、即ちBootLdr.hとAESKeys.incを使うように、予めコンパイルされています。これらのファイル名は変更されないことが推奨されます。

6.5.3. 後続走行

後続する走行ではファームウェアを符号化するのにこの応用が使われます。暗号化に先立って、ソース ファイルはコンパイル、アセンブルされて1つのコード セグメント ファイルと/または1つのEEPROMセグメント ファイルにリンクされなければなりません。ファイルはIntel®のHEX形式でなければなりません。

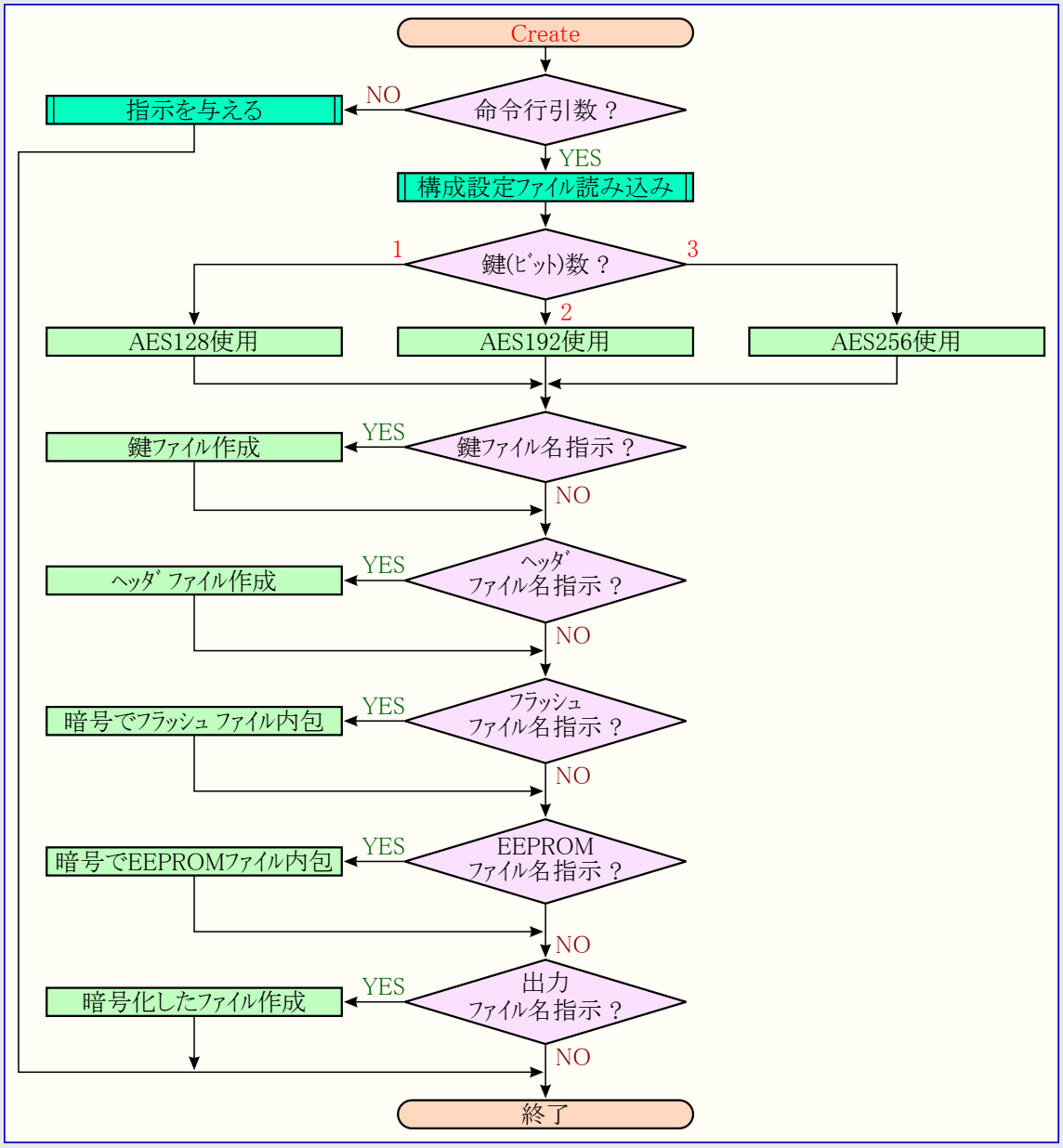
ファイル名はコマンド プロンプトで与えられ、構成設定ファイル内のデータに従って暗号化されたファイルが生成されます。

例: `Create -c Config.txt -e EEPROM.hex -f Flash.hex -oUpdate.enc -l BLB11 BLB12`

応用ソフトウェアとEEPROMのファイルは単一の暗号化したファイル内に結合されます。

6.5.4. ﾌﾟﾛｸﾞﾗﾑの流れ

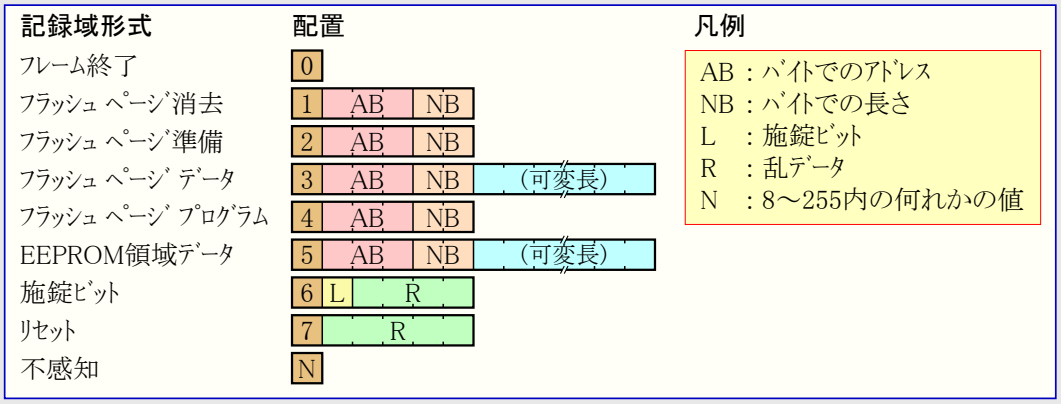
図6-2. CREATE応用の流れ図



6.5.5. 暗号化されたﾌｧｲﾙ

ﾌﾗｯｼｭとEEPROMのﾌｧｲﾙは暗号化されて1つの目的対象ﾌｧｲﾙに格納されます。けれども、暗号化の前にﾃﾞｰﾀは記録域内に編成されます。以下の図で図解されるように、7つの記録域形式があります。

図6-3. 暗号化ﾌｧｲﾙ用記録域

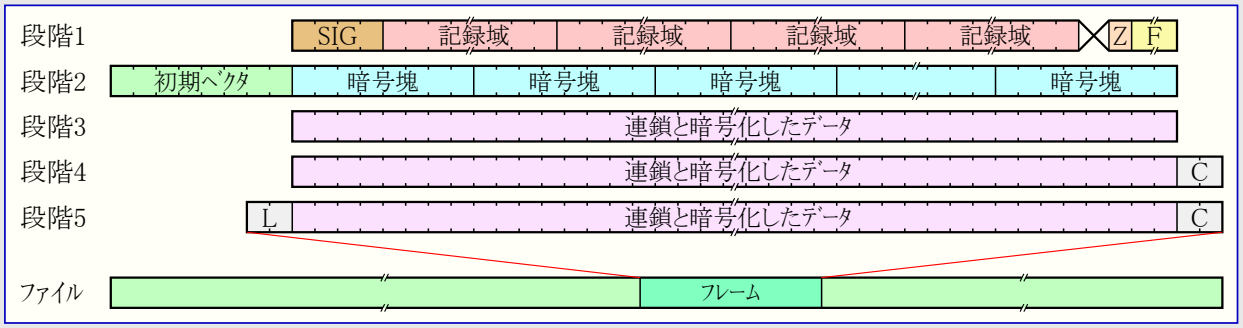


記録域形式はその記録域の先頭ﾊﾞｲﾄとして与えられます。応用ﾃﾞｰﾀは記録域形式1,2,3,4(即ち、消去、準備、ﾌﾗｯｼｭのﾍﾟｰｼﾞ緩衝部への読みと書き)に分類されます。EEPROM領域に関するﾃﾞｰﾀは記録域形式5内に構成されます。施錠ﾋﾞｯﾄは記録域形式6で送られます。記録域形式0と7は各々ﾌﾚｰﾑと転送の終了用です。

他の全ての記録域、即ち7を超える記録域識別子を持つそれらは不感知形式です。この任意選択が許可される(Createﾂｰﾙをご覧ください)と、不感知記録域の乱数はこのﾌｧｲﾙ内の散らされた不定位置に配置されます。

出力ﾌｧｲﾙは以下の図で図解されるように作成されます。

図6-4. 暗号化ﾌｧｲﾙ作成



段階は以下で記述されます(番号は上図参照)。

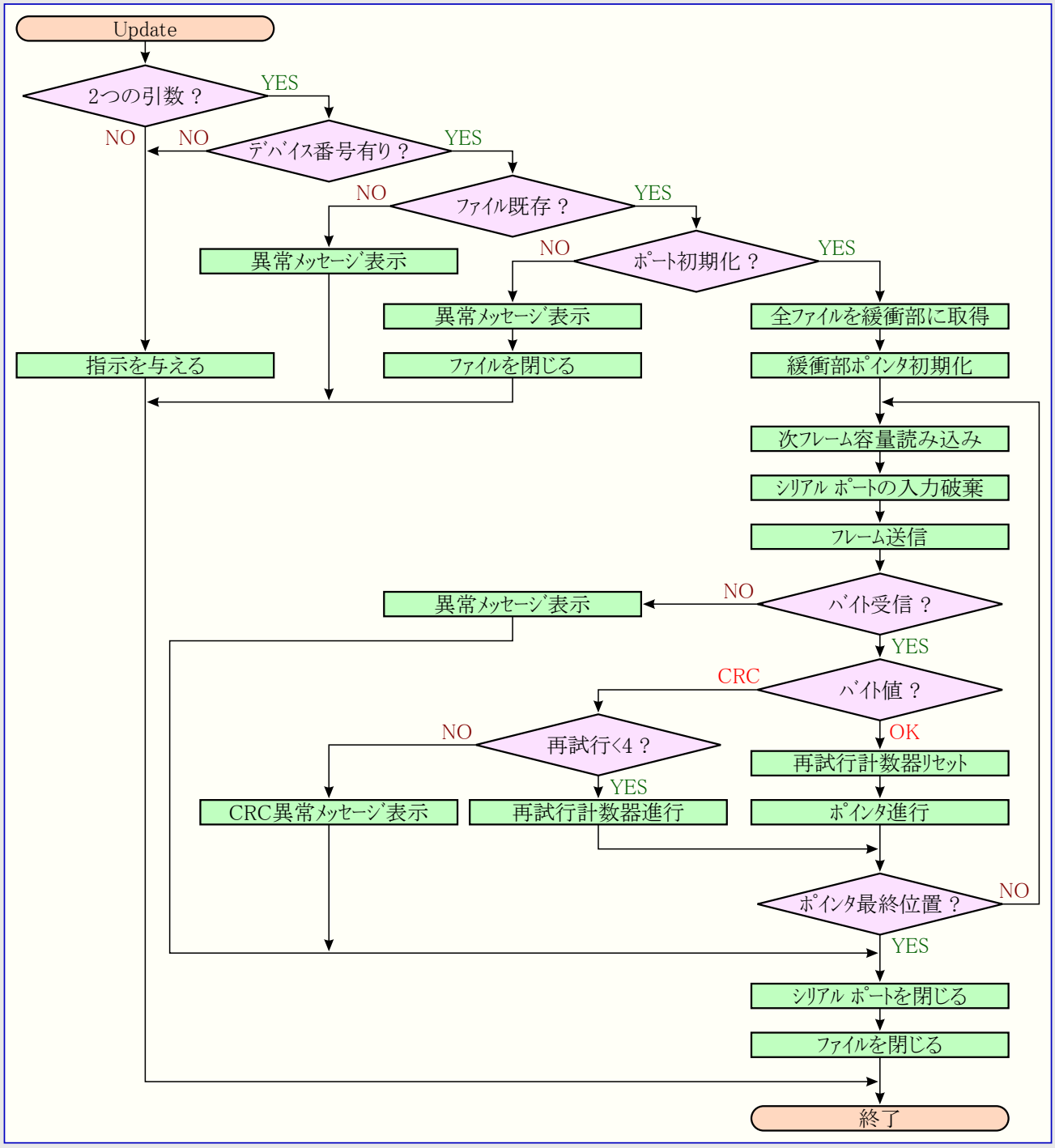
1. ﾃﾞｰﾀは記録域内に構成され、それはその後にﾌﾚｰﾑ識票(SIG)に後続して整列されます。ﾌﾚｰﾑの最後を記すために0(Z)が追加され、ﾌﾚｰﾑは16ﾊﾞｲﾄの倍数のﾌﾚｰﾑ容量を作成するために乱ﾃﾞｰﾀ(F)で穴埋めされます。
2. 初期ﾍﾞｸﾀがﾌﾚｰﾑに付着されます。先頭ﾌﾚｰﾑではそのﾍﾞｸﾀが構成設定ﾌｧｲﾙで与えられるものと等価です。後続ﾌﾚｰﾑでは初期ﾍﾞｸﾀが直前のﾌﾚｰﾑの最後の暗号塊と等価です。
3. 初期ﾍﾞｸﾀと暗号塊は連結されて暗号化されます。初期ﾍﾞｸﾀはその後にﾌﾚｰﾑから取り去られます。
4. CRC-16ﾁｪｯｸｻﾑ(C)が生成されてﾌﾚｰﾑに追加されます。
5. 長さ情報を除いたﾌﾚｰﾑの長さ(L)が計算されて、そのﾌﾚｰﾑの先頭に保存されます。

ﾌﾚｰﾑは出力ﾌｧｲﾙに書かれ、ﾃﾞｰﾀが処理されてしまうまでこの手順が繰り返されます。

6.6. PC応用 – Update

この応用は目的対象に暗号化ファイルを送るのに使われます。データはPC上のシリアル ポート経由で目的対象ハードウェア上のUSARTへ直接的に送ることができます。プログラムの流れは以下のように図解されます。

図6-5. Update応用の流れ図



Update応用はCreate応用で生成されたファイルを読みます。このファイルは暗号化したデータの1つまたはより多くの連結されたフレームから成ります。応用は一度に1フレームのデータを送信し、ブートローダからの応答を待つ間に一時停止します。次のフレームは応答が受信された後にだけ送信され、そうでなければ応用はフレームを再送するか、または通信を閉じるかのどちらかです。

Update応用はコマンド プロンプトから走行します。このコマンド プロンプト引数は下表で一覧にされます。

表6-4. Update応用の命令行引数

引数	説明
<filename.ext>	転送されるべき暗号化されたファイルへのパス
-COMn	シリアル ポート、nはシリアル ポート番号
-baudrate	ボーレート、ここでのボーレートは実際のボーレート数字

例:

```
update blinky.ext -COM1 -115200
```

更新システムが応用とEEPROMのファイルで指示されたフラッシュ メモリとEEPROMのそれらの部分を更新するだけであることに注意すべきです。応用領域のCRC検査が許可されている、またはcreateツールで消去任意選択が選択されている場合、全応用メモリはプログラミングする(書き込む)前に解除(消去)されます。

7. ハードウェア構成設定

目的対象ハードウェアは暗号化したファームウェアがブートローダに送られ得る前に正しく構成設定されなければなりません。この応用記述では評価用にATmega328PB Xplained Mini基板が使われると仮定されます。構成設定、プログラミングのための段階、デバッグの詳細は以降の項で説明されます。これらの項はATmega328PB Xplained Miniを作動するのに必要とされる基本的な情報だけを網羅します。より多くの情報については「ATmega328PB Xplained Mini使用者の手引き」を参照してください。

7.1. ATmega328PB Xplained Miniキット接続

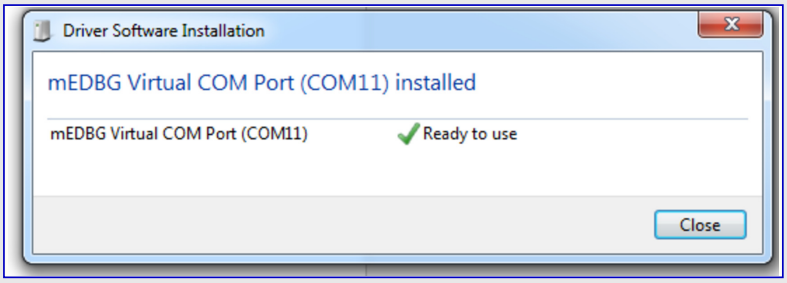
この項はAtmel Studio 7でATmega328PB Xplained Miniを接続する使用者を手助けします。

1. Atmel Studio 7版またはそれ以降をダウンロードしてインストールしてください。
2. Atmel Studioを起動してください。
3. ATmega328PB Xplained MiniをUSBポートに接続してください。それはAtmel Studioで見ることができます。

7.1.1. Xplained Miniキットの自動基板識別

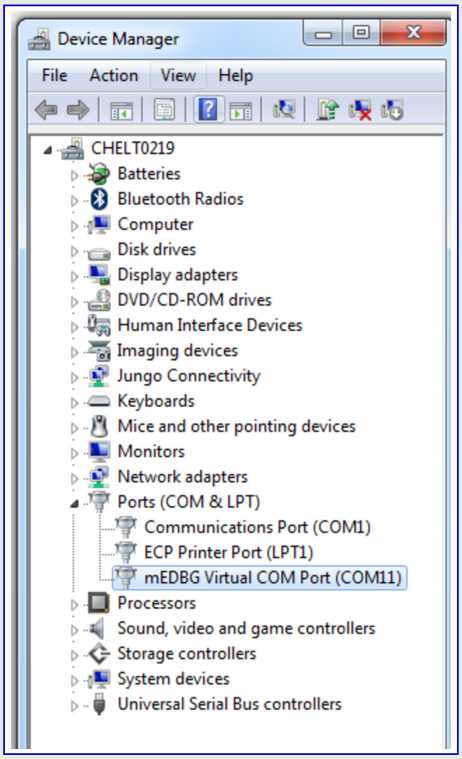
- 一旦ATmega328PB Xplained MiniがPCに接続されると、右で示されるようにWindows®のタスクバーがメッセージを立ち上げます。

図7-1. ATmega328PB Xplained Miniドライバ インストール



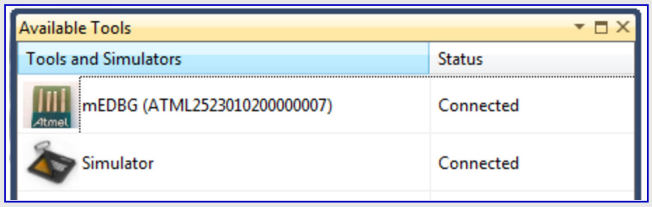
- ドライバ インストールがが適正なら、右で示されるようにデバイス マネージャーでEDBGが一覧にされます。

図7-2. 成功したmEDBGドライバ インストール



- Atmel Studioを開いてView(表示)⇒Available Tools(利用可能なツール)へ行ってください。EDBGはmEDBGとしてツールで一覧にされるべきで、ツール状態はConnected(接続済み)と表示されるべきです。これはツールがAtmel Studioと正しく通信していることを示します。

図7-3. 利用可能なツール下のmEDBG



7.1.2. mEDBG COMポートへのATmega328PB Xplained Mini UART接続

1. mEDBG USBをPCに接続してください。
2. COMポート番号を見つけるためにデバイス マネージャーを使ってください。
3. 既定COMポート設定は9600ボー、パリティなし、8ビット データ、1停止ビットです。COMポート設定はデバイス マネージャーを使うことによって応用に従って変更することができます。この応用は115200のボーレートで試験されています。

7.2. プログラミングとデバッグ

この項はmEDBGを用いることによってATmega328PB Xplained Miniキットのプログラミングとデバッグを手助けします。

7.2.1. mEDBG使用によるATmega328PB Xplained Miniプログラミング

1. mEDBG USBをPCに接続してください。
2. Atmel Studioへ行き、Tools(ツール)をクリック、Device Programming(デバイス書き込み)を選び、その後にATmega328PBを持つTool with Device(デバイス付きツール)として接続したmEDBG、Interface(インターフェース)としてISPを選び、Apply(適用)をクリックしてください。
3. Memories(メモリ)を選んでソースの.hexまたは.elfのファイルを示し、その後にProgram(書き込み)をクリックしてください。
4. ソースがヒューズ設定を含む場合、Production file(製造ファイル)へ行き、.elfファイルをアップロードしてヒューズを書いてください。

注: ISP書き込み失敗の場合、そのためにデバッグWIREが許可され得ます。デバッグWIRE動作を禁止する方法は「**7.2.2. mEDBG使用によるATmega328PB Xplained Miniデバッグ**」をご覧ください。

7.2.2. mEDBG使用によるATmega328PB Xplained Miniデバッグ

1. Atmel Studioを開始してください。
2. mEDBG USBをPCに接続してください。
3. あなたのプロジェクトを開いてください。
4. Project(プロジェクト)メニューでプロジェクト プロパティ ページを選んでください。Tools(ツール)タブを選び、DebuggerとしてmEDBG、InterfaceとしてdebugWireを選んでください。
5. Debug(デバッグ)メニューでStart Debugging and Break(デバッグ開始/中断)をクリックしてください。
6. ATmega328PBでDWENヒューズが許可されてない場合にAtmel Studioは異常メッセージを表示し、ISPインターフェースを用いてAtmel Studioにヒューズを設定させるためにYES(はい)をクリックしてください。
7. mainでの中断でデバッグ作業が開始されます。デバッグを開始することができます。
8. デバッグ動作を抜け出す時にDebug(デバッグ)メニューでDisable debugWIRE and Close(デバッグWIREを禁止して閉じる)を選んでください。

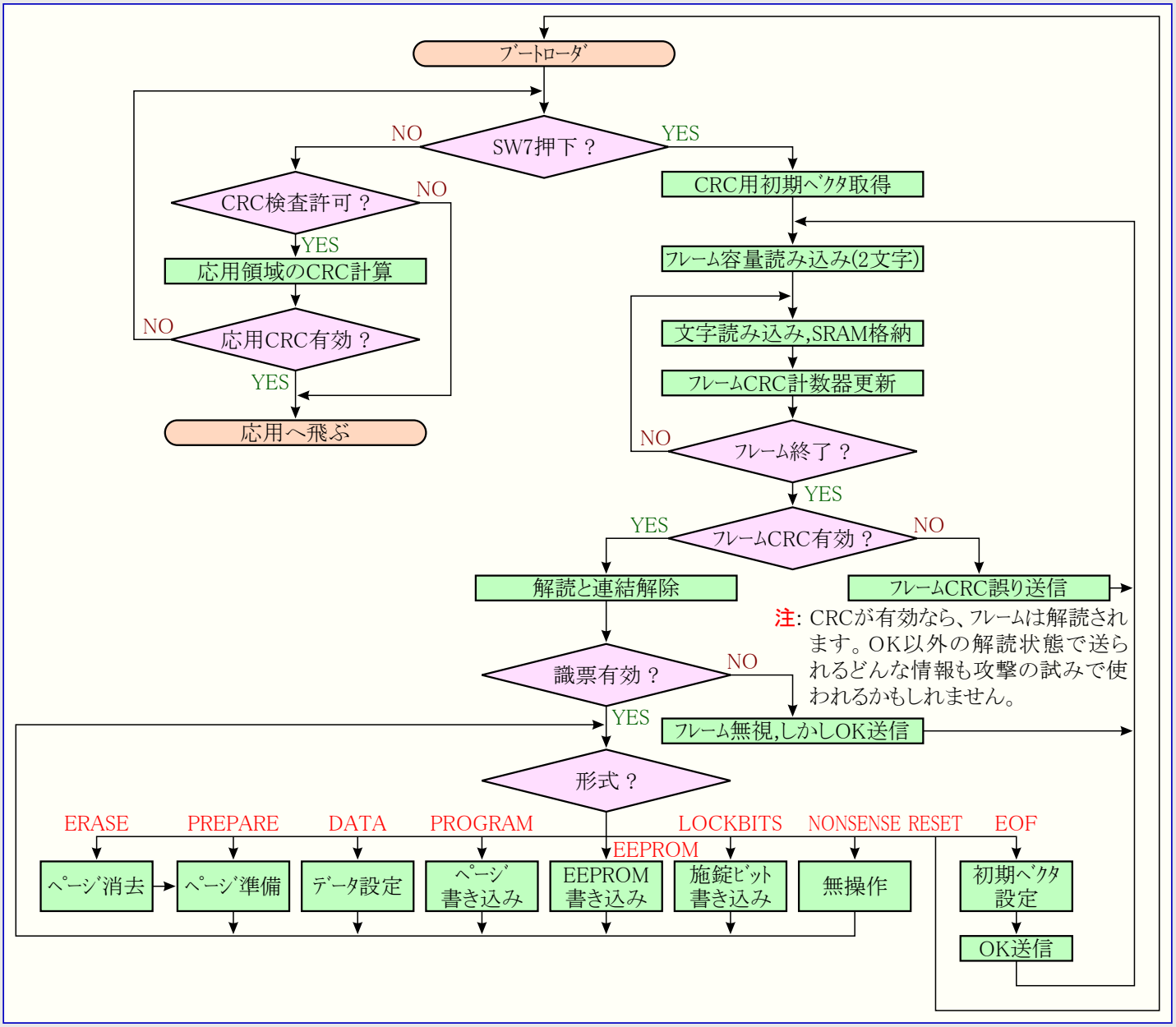
注: 1. デバッグ動作がDebug(デバッグ)メニューでDisable debugWIRE and Close(デバッグWIREを禁止して閉じる)を選ぶことによって抜け出されなかった場合、DWENヒューズが許可され、目的対象は未だデバッグ動作で、即ち、SPI(ISP)インターフェースを用いることによって目的対象をプログラミングすることが不可能です。

2. この応用記述で供給されるブートローダ コードは大きさに最適化され、故にデバッグを行うのが不可能かもしれません。最適化の削除はコード量を増してブートローダ領域に合わなくなるかもしれません。故にプログラミング動作でだけ使うことが推奨されます。

8. AVRﾌﾞｰﾄﾛｰﾀﾞ

ﾌﾞｰﾄﾛｰﾀﾞはﾃﾞﾊﾞｲｽが暗号化されたﾌｧｰﾑｳｪｱで更新され得る前に目的対象AVRに存在しなければなりません。ﾌﾞｰﾄﾛｰﾀﾞはPCと通信をしてﾌﾗｯｼｭ ﾒﾓﾘの応用領域とEEPROMをﾌﾟﾛｸﾞﾗﾐﾝｸﾞする能力があります。この応用記述と共に含まれるﾌﾞｰﾄﾛｰﾀﾞはAtmel Studio 7を使うことによって作成されています。ﾌﾟﾛｸﾞﾗﾑの流れは下図で図解されます。

図8-1. AVRﾌﾞｰﾄﾛｰﾀﾞのﾌﾟﾛｸﾞﾗﾑの流れ図



8.1. 鍵とﾍｯﾀﾞ ﾌｧｲﾙ

ﾌﾞｰﾄﾛｰﾀﾞをｺﾝﾊﾟｲﾙし得る前に、構成設定が必要ないくつかのﾊﾟﾗﾒｰﾀがあります。始めるにはPC応用のCreateによって生成された暗号鍵とﾍｯﾀﾞ ﾌｧｲﾙがﾌﾞｰﾄﾛｰﾀﾞ ﾌｫﾙﾀﾞに複写されなければなりません。このﾌｧｲﾙはﾌﾞｰﾄﾛｰﾀﾞのｿｰｽ ｺｰﾄﾞ内部の#include擬似命令でそれらが参照される時にｲﾝｸﾙｰﾄﾞされます。

8.2. ﾌﾟﾛｼﾞｪｸﾄ ﾌｧｲﾙ

この応用記述はATmega328PB用のﾃﾞﾊﾞｲｽ特定ﾌﾟﾛｼﾞｪｸﾄ ﾌｧｲﾙと共にやって来ます。

他のAVRﾃﾞﾊﾞｲｽについてはﾌﾟﾛｼﾞｪｸﾄ ﾀﾞｲﾙを用いて以下の項で与えられる手順に対してそれらを変更してください。

8.3. Atmel StudioとIARの設定

Atmel StudioとIAR™に対する共通設定が本項で一覧にされます。

1. 重要なﾋｭｰｽﾞ設定
   1.1. ﾌﾞｰﾄ ﾘｾｯﾄ ﾍﾞｸﾀが許可されなければなりません。
   1.2. Atmel STARTからﾀﾞｳﾝﾛｰﾄﾞしたATmega238PB例(ﾎﾞｰﾚｰﾄ38400)を正しく動かせるために下位ﾋｭｰｽﾞ ﾊﾞｲﾄを16MHz外部ｸﾛｯｸに設定してください。
2. SPMCSRﾚｼﾞｽﾀ ｱﾄﾞﾚｽはreg.hﾌｧｲﾙ内のSPMREG(下表参照)ﾏｸﾛを用いて定義しなければならず、このﾚｼﾞｽﾀ ｱﾄﾞﾚｽは間接ﾒﾓﾘ領域に配置された場合に変わり、このﾚｼﾞｽﾀのｱﾄﾞﾚｽはﾃﾞｰﾀｼｰﾄのﾚｼﾞｽﾀ要約項で見つけることができます。

図8-2. reg.hでのSPMREGｱﾄﾞﾚｽ設定

```
#ifndef REG_H_
#define REG_H_

// ﾋﾞｯﾄ値ﾏｸﾛ
#define EEMWE    2    // EEMCRでの位置2、値4
#define EEWE     1    // EEMCRでの位置1、値2

// SPMREGでのﾋﾞｯﾄとその他の位置
#define SPMEN    0    // SPMCRでの位置0
#define BLBSET   3
#define PGERS    1
#define PGWRT    2

#ifdef __RAMPZ__
#define _RAMPZ_  0x3B
#endif

#define SPMREG   0x37
#define SREG     0x3F
#define EEARL    0x21
#define EEARH    0x22
#define EECR     0x1F
#define EEDR     0x20

#endif /* INCFILE1_H_ */
```

3. reg.hﾌｧｲﾙで定義された他のﾚｼﾞｽﾀ ｱﾄﾞﾚｽはﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄに従って変更されなければなりません。
4. ｼﾝﾎﾞﾙの_RAMPZ_(下表参照)は64Kﾊﾞｲﾄよりも大きなﾌﾗｯｼｭ容量を持つﾃﾞﾊﾞｲｽに対してｺﾝﾊﾟｲﾗとｱｾﾝﾌﾞﾗで定義されなければなりません。
5. ｼﾝﾎﾞﾙの_MEMSPM_(下表参照)はSPMCSRﾚｼﾞｽﾀが直接ﾒﾓﾘ ｱｸｾｽ領域より上に配置される(即ち、SPMCSRﾚｼﾞｽﾀが$3Fより上に配置される)ﾃﾞﾊﾞｲｽに対してｱｾﾝﾌﾞﾗで定義されなければなりません。例えば、ATmega128については$68に配置されるSPMCSRを持ち、故にそれをｱｸｾｽするために定義が必要とされます。
6. ｼﾝﾎﾞﾙのBOOT_ADDRE=<ﾌﾞｰﾄ領域開始ｱﾄﾞﾚｽ>(下表参照)は64Kﾊﾞｲﾄよりも大きなﾌﾗｯｼｭ容量を持つﾃﾞﾊﾞｲｽに対してｺﾝﾊﾟｲﾗで定義されなければなりません。

表8-1. ｼﾝﾎﾞﾙ設定の要約

設定	m8	m16	m162	m168PA	m169	m32	m328PB	m64	m128	m256
RAMPZ	なし								有り	
MEMSPM	なし							有り		なし
BOOT_ADDR	なし								0x1E000	0x3E000
BOOT_START (IAR)	0x1800	0x3800				0x7000		0xE000	0x1E000	0x3E000
.text (Atmel Studio)										
SPMREG	0x37							0x68		0x37

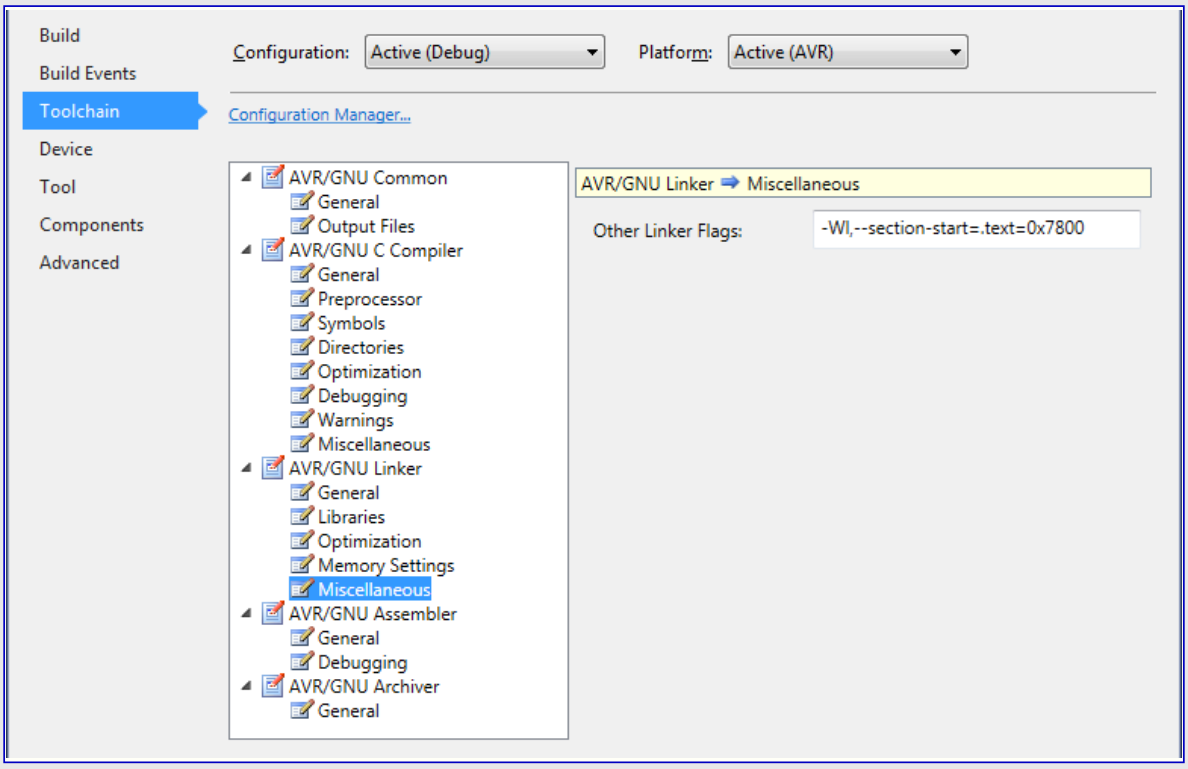
注: 上の表でのBOOT_ADD、BOOT_START、.textの値はﾌﾞｰﾄ領域が最大の時の値です。実際のﾌﾞｰﾄ容量に従ってこの値を変更することができます。

8.3.1. Atmel Studio設定

下で示されるｺﾝﾊﾟｲﾗ、ﾘﾝｶ、ｱｾﾝﾌﾞﾗの設定はAtmel Studioで特定ﾃﾞﾊﾞｲｽに対して構成設定する間に必要とされます。

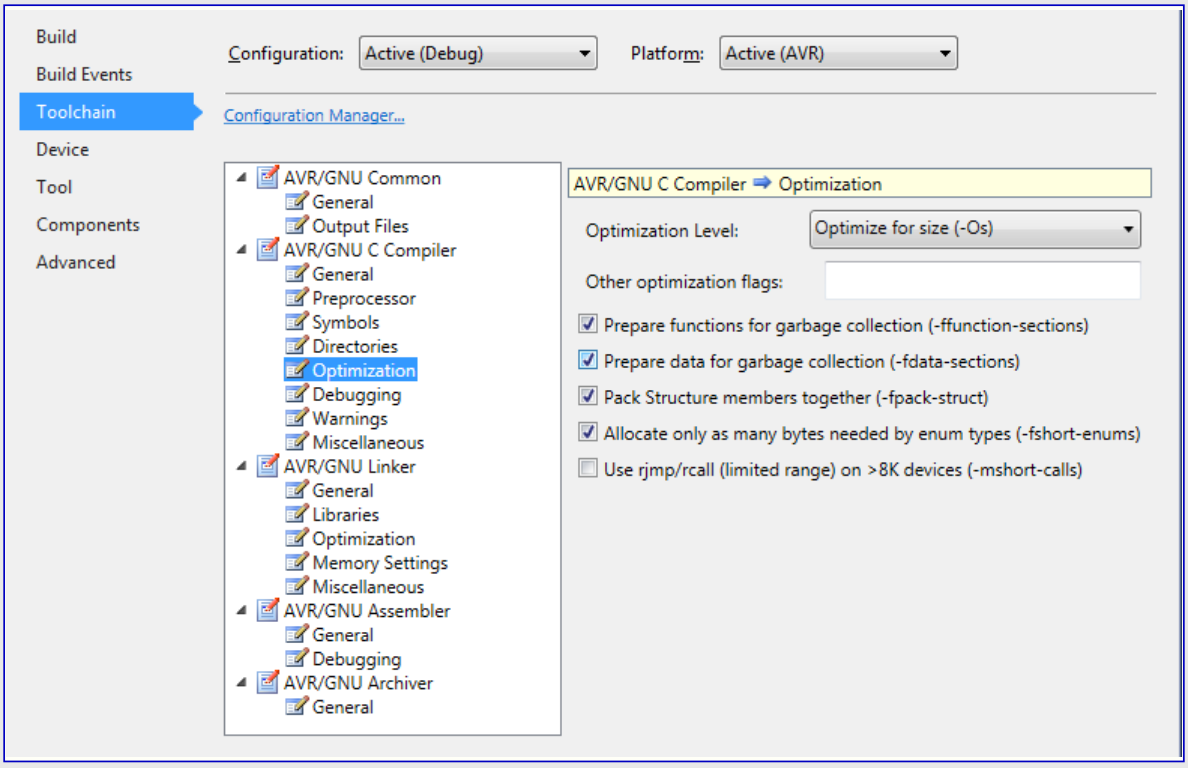
1. ﾌﾞｰﾄﾛｰﾀﾞ開始ｱﾄﾞﾚｽの.text(表8-1.参照)は(下図で示されるように)ｺｰﾄﾞをﾌﾞｰﾄﾛｰﾀﾞ開始ｱﾄﾞﾚｽにﾘﾝｸするためにﾘﾝｶ設定で定義されなければなりません。開始ｱﾄﾞﾚｽは(2倍した語ｱﾄﾞﾚｽ)ﾊﾞｲﾄ ｱﾄﾞﾚｽとして与えられるべきです。例えば、ATmega328PBのﾌﾞｰﾄﾛｰﾀﾞ語ｱﾄﾞﾚｽが0x3C00の場合、対応するﾊﾞｲﾄ ｱﾄﾞﾚｽは0x3C00×2=0x7800です。

図8-3. ﾌﾞｰﾄﾛｰﾀﾞ ｱﾄﾞﾚｽのｺｰﾄﾞ ﾘﾝｸ ｱﾄﾞﾚｽ設定



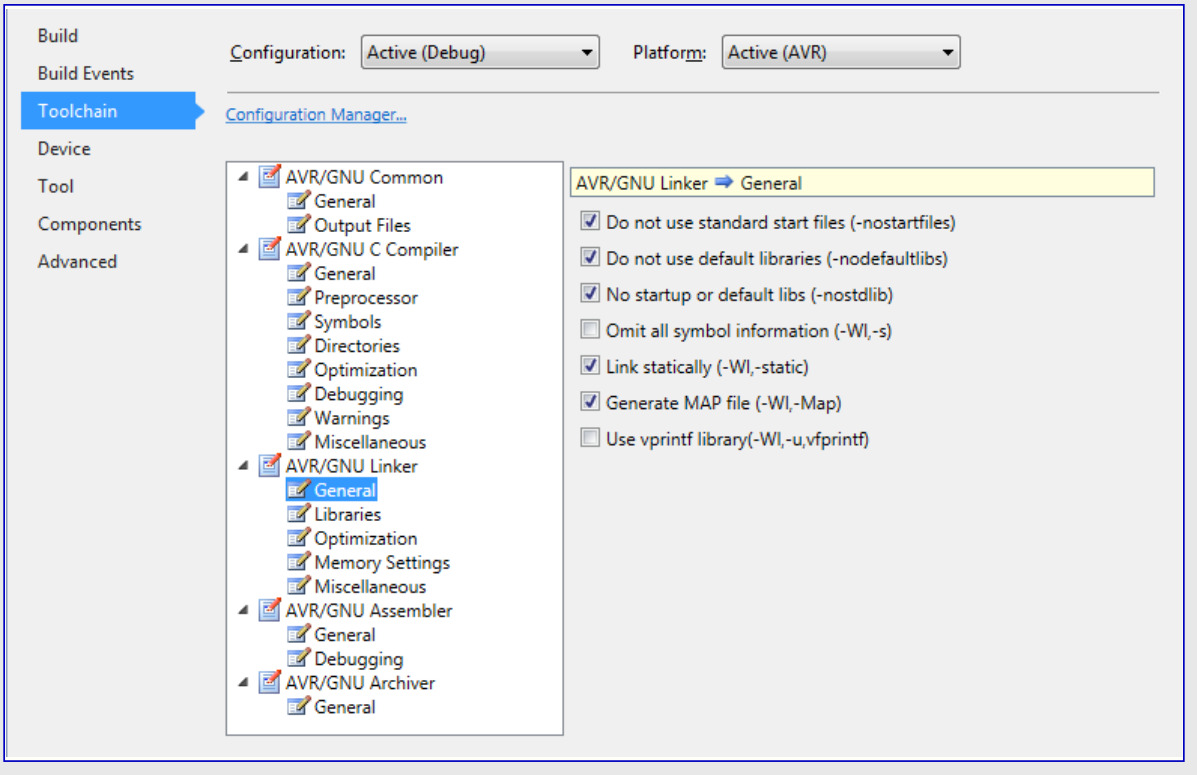
2. (下図で示されるように)最適化設定

図8-4. 最適化設定



3. (下図で示されるように)他のリンカ設定

図8-5. 他のリンカ設定

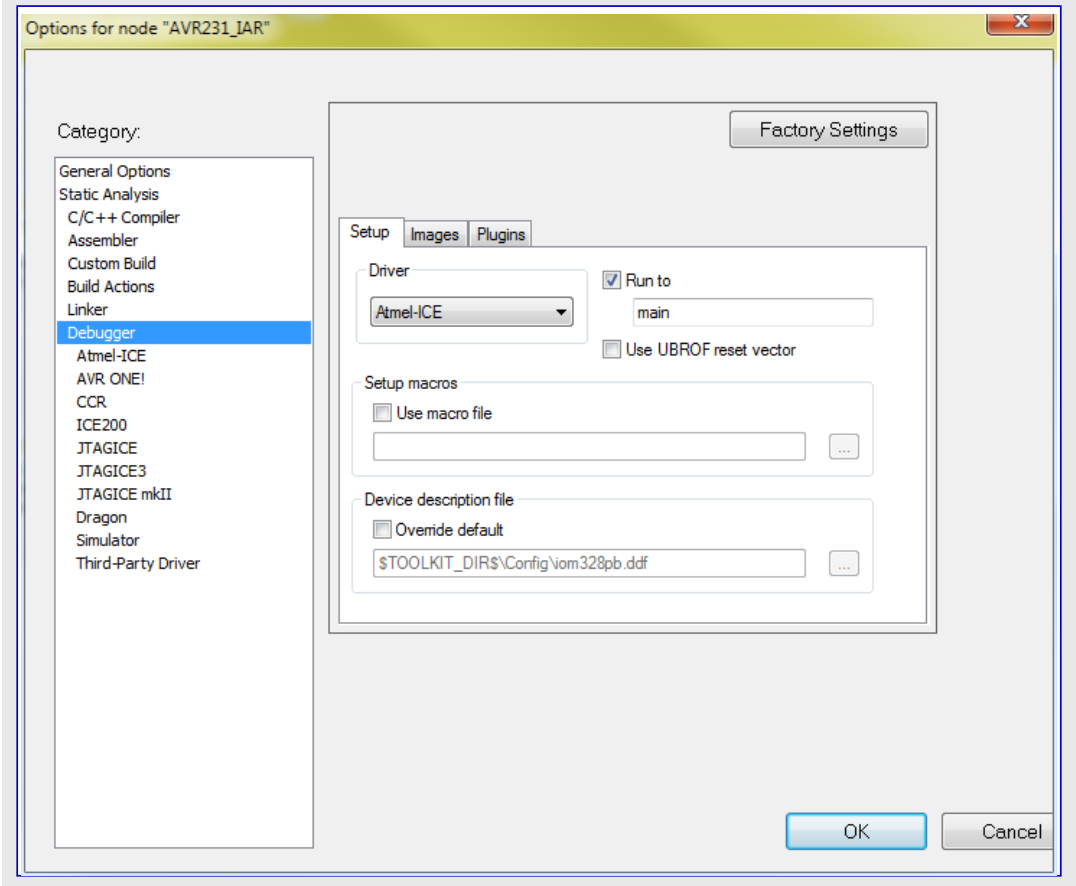


8.3.2. IAR設定

下で示されるｺﾝﾊﾟｲﾗ、ﾘﾝｶ、ｱｾﾝﾌﾞﾗの設定はIARで特定ﾃﾞﾊﾞｲｽに対して構成設定する間に必要とされます。

1. ATmega328PB Xplained Mini内のmEDBG使用時、(下図で示されるように)Debugger(ﾃﾞﾊﾞｯｶﾞ)のDriver(ﾄﾞﾗｲﾊﾞ)でAtmel-ICEが選ばれるべきです。

図8-6. ﾃﾞﾊﾞｯｶﾞ ﾄﾞﾗｲﾊﾞ



2. 既定ﾘﾝｶ ﾌｧｲﾙを上書きするために使用者定義された.xclﾌｧｲﾙが使われるべきです。ﾘﾝｶを正しく動かせるために下の手順に従ってください。

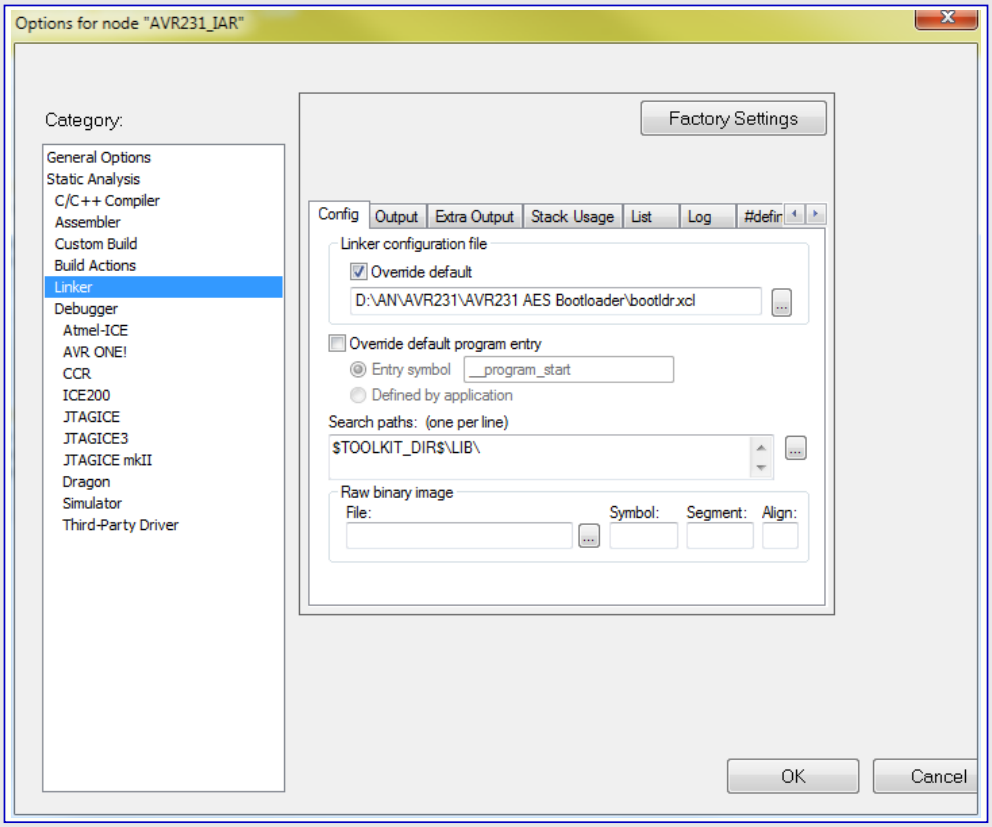
2.1. ﾃﾞﾊﾞｲｽに従って雛形ﾘﾝｶ ﾌｧｲﾙ(例えば、C:¥Program Files (x86)¥IAR Systems¥Embedded Workbench 7.4¥avr¥src¥template¥cfgm328pb.xcl)をあなたのﾌﾟﾛｼﾞｪｸﾄに複写してそれを好きな何かに改名してください。

2.2. このﾌｧｲﾙを開いて元の内容の下に下の寧陽を貼り付けてそれを保存してください。

```
-ca90
-w29

//==============================================================================
// Interrupt vectors

-Z(CODE)INTVEC=BOOT_START-(BOOT_START+_..X_INTVEC_SIZE-1)
//-H1895
-h(CODE)BOOT_START-(BOOT_START+_..X_INTVEC_SIZE-1)

//==============================================================================
// Code memory

-Z(CODE)NEAR_F,HUGE_F,SWITCH,INITTAB,DIFUNCT,CODE=BOOT_START-_..X_FLASH_END
-Z(FARCODE)FAR_F,FARCODE=BOOT_START-_..X_FLASH_END

//==============================================================================
// RAM
-Z(DATA)NEAR_I,NEAR_Z=_..X_SRAM_BASE-_..X_SRAM_END
-Z(DATA)RSTACK+_..X_RSTACK_SIZE=(_..X_SRAM_END-_..X_RSTACK_SIZE+1)-_..X_SRAM_END
-Z(DATA)CSTACK+_..X_CSTACK_SIZE=(_..X_SRAM_END-_..X_RSTACK_SIZE-_..X_CSTACK_SIZE
+1)-(_..X_SRAM_END-_..X_RSTACK_SIZE)
//-Z(DATA)RSTACK+40=(_..X_SRAM_END-40+1)-_..X_SRAM_END
//-Z(DATA)CSTACK+300=(_..X_SRAM_END-40-300+1)-(_..X_SRAM_END-40)
//-Z(DATA)TINY_I,TINY_Z,TINY_N=RAM_BASE-FF
//-Z(DATA)TINY_I,TINY_Z,TINY_N=RAM_BASE-100
```

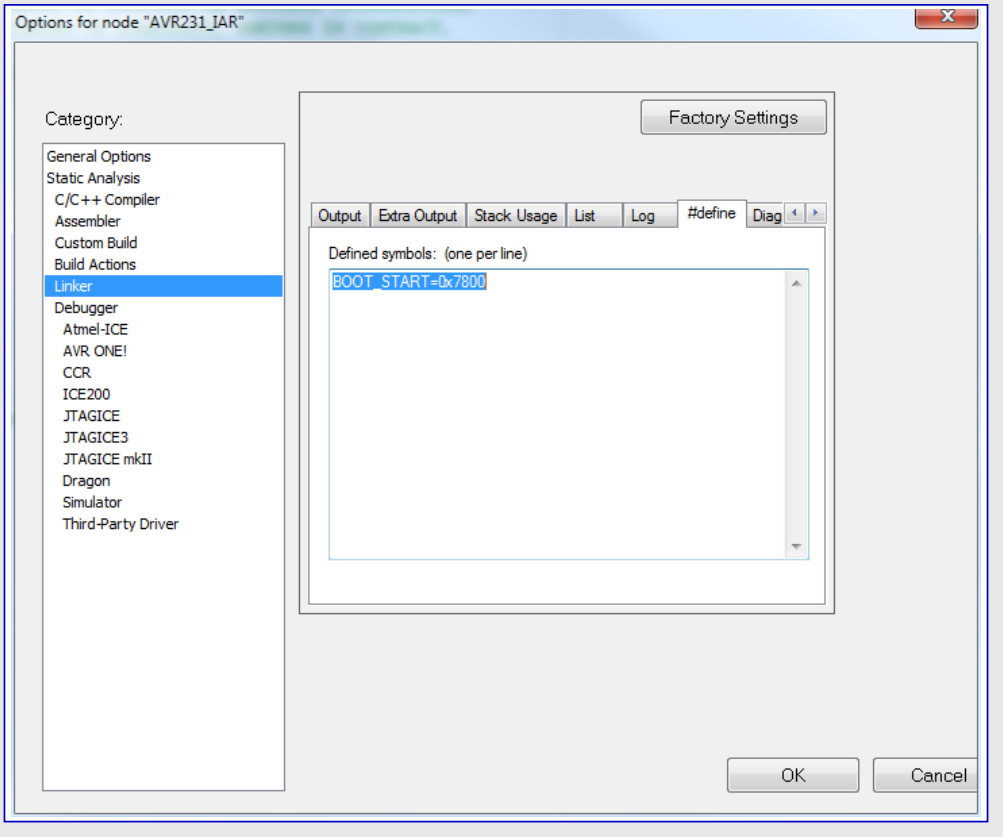
2.3. (下図で示されるように)既定ﾘﾝｶ ﾌｧｲﾙを上書きするのにこのﾌｧｲﾙを使ってください。

図8-7. 既定ﾘﾝｶ ﾌｧｲﾙ上書き



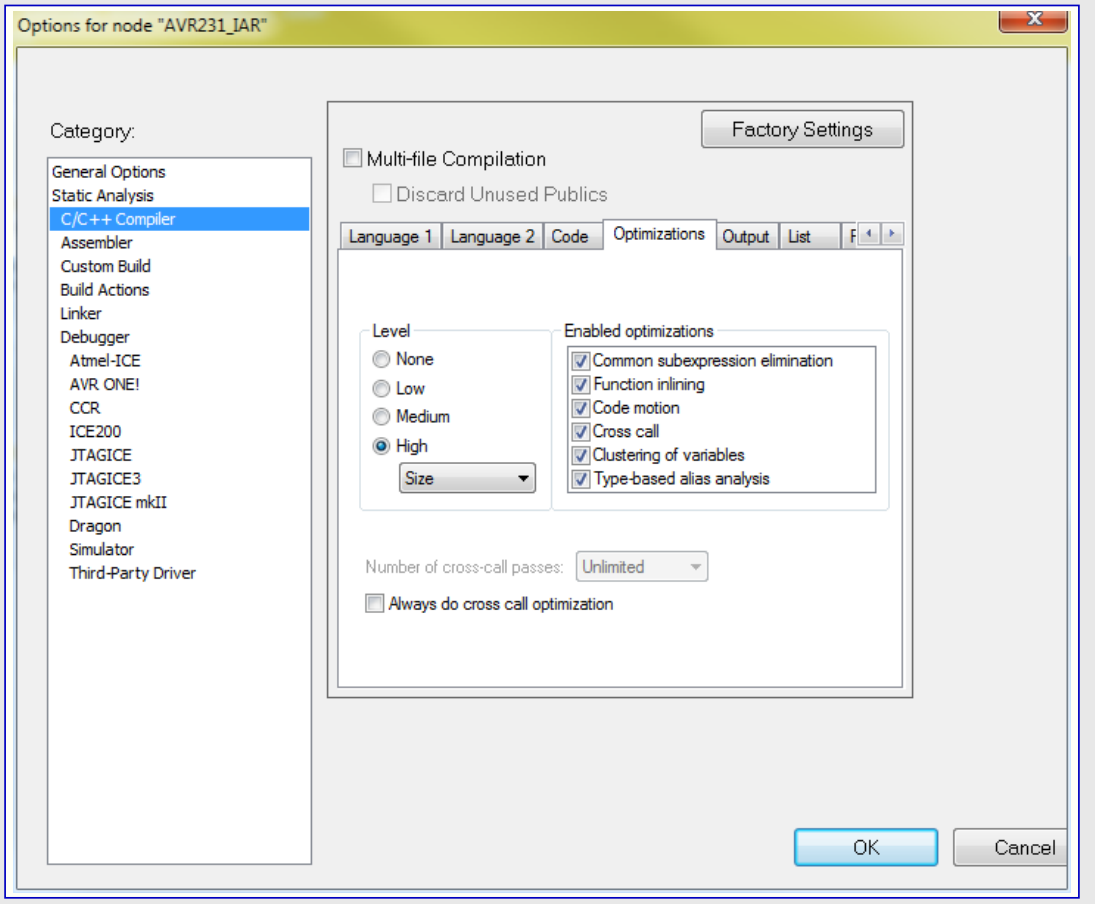
2.4. (下図で示されるように)デバイスに従ってBOOT_START(表8-1.参照)を定義してください。

図8-8. リンカ #define



3. (下図で示されるように)最適化設定

図8-9. 最適化設定



8.4. ブートローダのインストール

ブートローダをコンパイルし、その後にAtmel Studio 7またはそれ以降を用いてそれを目的対象に書き込んでください。ブートローダを走行する前に以下のヒューズビットが次のように構成設定されなければなりません。

- ブートローダ領域の容量。前の方で記述されるように、この領域容量がBOOT_SIZE設定に合うようにヒューズビットを設定してください。通常、BLSは語で与えられますが、BOOT_SIZEパラメータがバイトで与えられることに注意してください。
- ブート リセット ベクタ。ブート リセット ベクタは許可されなければなりません。
- 発振器任意選択。発振器ヒューズビットはデバイス依存です。それらは構成設定が必要かもしれません(UARTに影響を及ぼします)。

注: 正しい発振器任意選択設定に特別な注意を払ってください。例え小さな誤調整でも、おそらく通信失敗に終わり得ます。

推奨されるヒューズビット設定が下表で提供されます。デバイス依存のヒューズビットの詳細な説明についてはデバイスのデータシートをご覧ください。

表8-2. 推奨ヒューズビット

ヒューズビット名	ATmega8, 8515, 8535, 16, 162, 169, 32, 328PB, 64	ATmega128
BOOTSZ1,0	0,0	0,1
BOOTRST	0	

注: 0=プログラム、1=非プログラムの意味です。

応用メモリとブートローダの両方を保護するように施錠ビットを設定することが推奨されますが、それはヒューズビットが設定されてしまった後でだけです。施錠ビットはMicrochip IDE(Atmel Studio 7またはそれ以降)を用いて設定することができます。ファームウェアが暗号化される時に命令行の引数としてそれらが定義されていれば、ファームウェア更新中にBLS施錠ビットも設定されます。推奨施錠ビット設定は次のとおりです。

- メモリ施錠ビット: これらは認められていないメモリへのアクセスを防ぐように設定されるべきです。デバイスを消去することなく実装書き込み経由でアクセスすることができないことに注意してください。
- ブートローダ領域用保護形態: SPMとLPMの命令はBLSに対して読み書きを許されるべきではありません。これはブートローダを不正にする応用領域内のファームウェアを保護して解読鍵を安全に保ちます。
- 応用領域用保護形態: 応用領域をアクセスするSPMとLPMの命令に関して全く制限を設定すべきではなく、さもなければブートローダがそれをプログラミングできません。

注: デバイスが正しく施錠されない場合にメモリはISPインターフェース経由でアクセスすることができ、ファームウェアを暗号化する全体的な意味がないことを理解することが重要です。

存在するAVR MCUに対する推奨施錠ビット設定は下表で与えられます。施錠ビットの詳細な説明についてはデバイスのデータシートをご覧ください。

表8-3. 推奨施錠ビット

ヒューズビット名	ATmega8, 8515, 8535, 16, 162, 169, 32, 328PB, 64, 128
BLB12:BLB11	0 0
BLB02:BLB01	1 1
LB2:LB1	0 0

8.5. 性能

以降の項は実行時間とコード量に関してシステム性能を要約します。

8.5.1. 実行時間

受信、復号、データ プログラミング(書き込み)のために目的対象デバイスに必要とされる時間は以下の要素に依存します。

- ファイル容量。より多くのデータは、より長くかかります。
- ボーレート。より高い転送速度は転送時間をより短くします。
- 目的対象AVR速度。より高いクロック周波数は復号時間をより短くします。
- フラッシュ ページのプログラミング(書き込み)時間。これはデバイス不変で変更できません。
- 鍵の大きさ。AES128はAES256よりもより速い解読です。実際、AES192はAES256よりも遅いです。それは2のべき乗でない192で行うべき何かがあります。
- その他設定。例えば、応用領域のCRC検査は短い時間がかかります。

8.5.2. コード量

コンパイルに最高最適化設定を用いると、ブートローダは2Kバイトのフラッシュ メモリ内に上手く合います。

暗号鍵が全く与えられない場合、AESなしでブートローダが構築されることに注目すべきです。この応用記述はその後に標準的なブートローダ システムとして実行し、ブートローダ支援を持つどのAVRでも使うことができます。

9. 要約

この応用記述はﾌﾞｰﾄﾛｰﾀﾞ能力を持つAVRﾏｲｸﾛ ｺﾝﾄﾛｰﾗへ安全にﾃﾞｰﾀを転送する方法が提供されています。この資料は保護されたｼｽﾃﾑを構築する時に実行されるべき技法も強調されています。AVR設計の安全性を増すために、以下の問題が考慮されるべきです。

暗号化した形式でのﾀﾞｳﾝﾛｰﾄﾞを支援するﾌﾞｰﾄﾛｰﾀﾞの実装。(製造中に)ﾌﾞｰﾄﾛｰﾀﾞが最初にｲﾝｽﾄｰﾙされる時に将来のﾌｧｰﾑｳｪｱ更新に必要とされる解読鍵が搭載されなければなりません。ﾌｧｰﾑｳｪｱはその後に部外者から内容を保護し、暗号化形式で配布することができます。

応用とﾌﾞｰﾄﾛｰﾀﾞの領域を保護するためにAVRの施錠ﾋﾞｯﾄを使用。ﾃﾞﾊﾞｲｽからの読み込みを防ぐように施錠ﾋﾞｯﾄが設定されると、ﾒﾓﾘ内容は取り出すことができません。施錠ﾋﾞｯﾄが設定されない場合、ﾌｧｰﾑｳｪｱを暗号化する必要がありません。

配布前のﾌｧｰﾑｳｪｱ暗号化。暗号化されたﾌｧｰﾑｳｪｱは正しい解読鍵なしでどんな外部実体に対しても役に立ちません。

暗号鍵を安全に保護。暗号鍵は2つの場所にだけ格納されるべきで、それは施錠ﾋﾞｯﾄによって保護されているﾌﾞｰﾄﾛｰﾀﾞ内と、製造業者でのﾌｧｰﾑｳｪｱ開発台です。

連鎖暗号化ﾃﾞｰﾀ。ﾃﾞｰﾀが連結される時に暗号化された各塊は直前の塊に依存します。結果として、等しい平文は異なる暗号化出力を生成します。

ﾌｧｰﾑｳｪｱに於いて標準的で予測可能な様式を避ける。殆どのﾌﾟﾛｸﾞﾗﾑは侵入者の手助けのために働くだけの低いｱﾄﾞﾚｽへ飛ぶことで始まる割り込みﾍﾞｸﾀ表のような、共通する枠組みと何らかの予測可能な様式を持ちます。一定の数で未使用領域を穴埋めすることも避けてください。

方法を隠す。使われつつある算法やどの鍵長かに言及する必要は全くありません。侵入者がｼｽﾃﾑについて知らないことが良いことです。暗号化の方法を知ることは或る攻撃者を受け流すと主張されるかもしれませんが、方法について何も知らないことは労力を増して更に多くを防ぐかもしれません。

必要とされるなら、ﾌﾞｰﾄﾛｰﾀﾞは応用領域の消去にも使われるかもしれません。多くの攻撃の試みは通常の動作環境からﾃﾞﾊﾞｲｽを取り外して暴き台での通電を含みます。例えば、LCDが失われている、またはﾒﾓﾘ内にCRC誤りがあることの検出が(ﾌﾞｰﾄﾛｰﾀﾞ領域と解読鍵を含む)全ﾒﾓﾘの完全な消去を始めるかもしれません。

更新用の外部通信ﾁｬﾈﾙを使うことが不適または不可能な応用では、ﾌｧｰﾑｳｪｱをCryptoMemory®ﾃﾞﾊﾞｲｽに格納することができます。このﾒﾓﾘは更新が必要とされる時にﾃﾞﾊﾞｲｽのｽﾛｯﾄに簡単に挿入することができるﾘﾑｰﾊﾞﾌﾞﾙ ｽﾏｰﾄ ｶｰﾄﾞとして外装することができます。ﾏｲｸﾛ ｺﾝﾄﾛｰﾗは始動に於いてCryptoMemoryの存在に関して調べて必要とされる時にﾌｧｰﾑｳｪｱ更新を持ってきます。

安全なﾊｰﾄﾞｳｪｱを使用。ﾊｰﾄﾞｳｪｱが構造的欠陥を持つ場合、強力な暗号規約は役に立ちません。AVRﾏｲｸﾛ ｺﾝﾄﾛｰﾗでは安全性の問題が全く報告されていません。

この一覧はもっと長く作ることができますが、その目的は単に設計者を正しい方向に向けさせることです。敵の理解力や忍耐力を見くびらないでください。

10. Atmel | STARTからのｿｰｽ ｺｰﾄﾞ取得

ｺｰﾄﾞ例は画像使用者ｲﾝﾀｰﾌｪｰｽ(GUI)を通して応用ｺｰﾄﾞの構成設定を許すｳｪﾌﾞに基づくAtmel | STARTを通して利用可能です。ｺｰﾄﾞは下の直接ｺｰﾄﾞ例ﾘﾝｸまたはAtmel | START先頭頁のBROWSE EXAMPLES(例検索)釦経由Atmel StudioとIAR Embedded Workbench®の両方に対してﾀﾞｳﾝﾛｰﾄﾞすることができます。

Atmel | STARTｳｪﾌﾞ ﾍﾟｰｼﾞ : http://start.atmel.com/

ｺｰﾄﾞ例

- AVR231 AESﾌﾞｰﾄﾛｰﾀﾞ (AVR231 AES Bootloader):
  - http://start.atmel.com/#example/Atmel:AVR231_AES_Bootloader:0.0.1::Application:AVR231_AES_Bootloader:

例ﾌﾟﾛｼﾞｪｸﾄについての詳細と情報に関してはAtmel | STARTでUser guide(使用者の手引き)を押下してください。User guide釦はAtmel | STARTﾌﾟﾛｼﾞｪｸﾄ構成設定部内の一覧画面でﾌﾟﾛｼﾞｪｸﾄ名をｸﾘｯｸすることにより、例閲覧部で見つけることができます。

Atmel Studio

DOWNLOAD SELECTED EXAMPLE(選んだ例をﾀﾞｳﾝﾛｰﾄﾞ)をｸﾘｯｸすることにより、Atmel | STARTで例閲覧部からAtmel Studio用.atzipﾌｧｲﾙとしてｺｰﾄﾞをﾀﾞｳﾝﾛｰﾄﾞしてください。Atmel | START内からﾌｧｲﾙをﾀﾞｳﾝﾛｰﾄﾞするには、EXPORT PROJECT(ﾌﾟﾛｼﾞｪｸﾄをｴｸｽﾎﾟｰﾄ)に続いてDOWNLOAD PACK(一括ﾀﾞｳﾝﾛｰﾄﾞ)をｸﾘｯｸしてください。

ﾀﾞｳﾝﾛｰﾄﾞした.atzipﾌｧｲﾙをﾀﾞﾌﾞﾙ ｸﾘｯｸしてください。ﾌﾟﾛｼﾞｪｸﾄがAtmel Studio 7.0に導入されます。

IAR Embedded Workbench

IAR Embedded Workbenchでﾌﾟﾛｼﾞｪｸﾄをｲﾝﾎﾟｰﾄする方法の情報についてはAtmel | START使用者の手引きを開き、Using Atmel Start Output in External Tools(外部ﾂｰﾙでAtmel START出力を使用)とIAR Embedded Workbenchを選んでください。Atmel | START使用者の手引きへのﾘﾝｸは共に頁の右上隅に置かれたAtmel | START先頭頁からAbout(これについて)またはﾌﾟﾛｼﾞｪｸﾄ構成設定部内のHelp And Support(手助けと支援)をｸﾘｯｸすることによって見つけることができます。

11. 参考

- ATmega328PBﾃﾞｰﾀｼｰﾄ (http://www.microchip.com/wwwproducts/en/atmega328pb)
- ATmega328PB Xplained Miniｷｯﾄ (http://www.microchip.com/developmenttools/productdetails.aspx?partno=atmega328pb-xmini)
- Atmel Studio (http://www.atmel.com/tools/atmelstudio.aspx?tab=overview)
- Atmel START (http://start.atmel.com)
- AT10764 (http://ww1.microchip.com/downloads/en/appnotes/atmel-42508-software-library-for-aes-128-encryption-and-decryption_applicationnote_at10764.pdf)
- AVR230 DES(ﾃﾞｰﾀ暗号化規格)ﾌﾞｰﾄﾛｰﾀﾞ (http://ww1.microchip.com/downloads/en/appnotes/doc2541.pdf)
- Handbook of Applied Cryptography (http://cacr.uwaterloo.ca/hac)
- AES仕様 (http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf)

12. 改訂履歴

資料改訂	日付	注釈
2589D	2006年8月	いくつかの些細な修正
2589E	2012年3月	新雛形 「PC応用」項更新 ATmega32ﾌﾟﾛｼﾞｪｸﾄで修正された不正な割り込みﾍﾞｸﾀの大きさ ﾃﾞｰﾀに依存して削除された周回
A	2017年6月	Atmel資料番号2589をMicrochip形式に変換されて置き換えられました。資料は更新され応用ｺｰﾄﾞはATmega328PBに対して試験されています。
B	2018年4月	Atmel STARTｺｰﾄﾞ例へのﾘﾝｸと共に「Atmel｜STARTからのｿｰｽ ｺｰﾄﾞ取得」章を追加

Microchipｳｪﾌﾞ ｻｲﾄ

Microchipはhttp://www.microchip.com/で当社のｳｪﾌﾞ ｻｲﾄ経由でのｵﾝﾗｲﾝ支援を提供します。このｳｪﾌﾞ ｻｲﾄはお客様がﾌｧｲﾙや情報を容易に利用可能にする手段として使われます。お気に入りのｲﾝﾀｰﾈｯﾄ ﾌﾞﾗｳｻﾞを用いてｱｸｾｽすることができ、ｳｪﾌﾞ ｻｲﾄは以下の情報を含みます。

- **製品支援** – ﾃﾞｰﾀｼｰﾄと障害情報、応用記述と試供ﾌﾟﾛｸﾞﾗﾑ、設計資源、使用者の手引きとﾊｰﾄﾞｳｪｱ支援資料、最新ｿﾌﾄｳｪｱ配布と保管されたｿﾌﾄｳｪｱ
- **全般的な技術支援** – 良くある質問(FAQ)、技術支援要求、ｵﾝﾗｲﾝ検討ｸﾞﾙｰﾌﾟ、Microchip相談役ﾌﾟﾛｸﾞﾗﾑ員一覧
- **Microshipの事業** – 製品選択器と注文の手引き、最新Microchip報道発表、ｾﾐﾅｰとｲﾍﾞﾝﾄの一覧、Microchip営業所の一覧、代理店と代表する工場

お客様への変更通知ｻｰﾋﾞｽ

Microchipのお客様通知ｻｰﾋﾞｽはMicrochip製品を最新に保つのに役立ちます。加入者は指定した製品系統や興味のある開発ﾂｰﾙに関連する変更、更新、改訂、障害情報がある場合に必ず電子ﾒｰﾙ通知を受け取ります。

登録するにはhttp://www.microchip.com/でMicrochipのｳｪﾌﾞ ｻｲﾄをｱｸｾｽしてください。”Support”下で”Customer Change Notification”をｸﾘｯｸして登録指示に従ってください。

お客様支援

Microchip製品の使用者は以下のいくつかのﾁｬﾈﾙを通して支援を受け取ることができます。

- 代理店または販売会社
- 最寄りの営業所
- 現場応用技術者(FAE:Field Aplication Engineer)
- 技術支援

お客様は支援に関してこれらの代理店、販売会社、または現場応用技術者(FAE)に連絡を取るべきです。最寄りの営業所もお客様の手助けに利用できます。営業所と位置の一覧はこの資料の後ろに含まれます。

技術支援はhttp://www.microchip.com/supportでのｳｪﾌﾞ ｻｲﾄを通して利用できます。

Microchipﾃﾞﾊﾞｲｽ ｺｰﾄﾞ保護機能

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。

法的通知

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。

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。

© 2018年、Microchip Technology Incorporated、米国印刷、不許複製

DNVによって認証された品質管理ｼｽﾃﾑ

ISO/TS 16949

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。

日本語© *HERO* 2021.

本応用記述はMicrochipのAN2462応用記述(DS00002462B-2018年4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	**ｵｰｽﾄﾗﾘｱ - ｼﾄﾞﾆｰ** Tel: 61-2-9868-6733 **中国 - 北京** Tel: 86-10-8569-7000 **中国 - 成都** Tel: 86-28-8665-5511 **中国 - 重慶** Tel: 86-23-8980-9588 **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 **中国 - 香港特別行政区** Tel: 852-2943-5100 **中国 - 南京** Tel: 86-25-8473-2460 **中国 - 青島** Tel: 86-532-8502-7355 **中国 - 上海** Tel: 86-21-3326-8000 **中国 - 瀋陽** Tel: 86-24-2334-2829 **中国 - 深圳** Tel: 86-755-8864-2200 **中国 - 蘇州** Tel: 86-186-6233-1526 **中国 - 武漢** Tel: 86-27-5980-5300 **中国 - 西安** Tel: 86-29-8833-7252 **中国 - 厦門** Tel: 86-592-2388138 **中国 - 珠海** Tel: 86-756-3210040	**ｲﾝﾄﾞ - ﾊﾝｶﾞﾛｰﾙ** Tel: 91-80-3090-4444 **ｲﾝﾄﾞ - ﾆｭｰﾃﾞﾘｰ** Tel: 91-11-4160-8631 **ｲﾝﾄﾞ - ﾌﾟﾈｰ** Tel: 91-20-4121-0141 **日本 - 大阪** Tel: 81-6-6152-7160 **日本 - 東京** Tel: 81-3-6880-3770 **韓国 - 大邱** Tel: 82-53-744-4301 **韓国 - ｿｳﾙ** Tel: 82-2-554-7200 **ﾏﾚｰｼｱ - ｸｱﾗﾙﾝﾌﾟｰﾙ** Tel: 60-3-7651-7906 **ﾏﾚｰｼｱ - ﾍﾟﾅﾝ** Tel: 60-4-227-8870 **ﾌｨﾘﾋﾟﾝ - ﾏﾆﾗ** Tel: 63-2-634-9065 **ｼﾝｶﾞﾎﾟｰﾙ** Tel: 65-6334-8870 **台湾 - 新竹** Tel: 886-3-577-8366 **台湾 - 高雄** Tel: 886-7-213-7830 **台湾 - 台北** Tel: 886-2-2508-8600 **ﾀｲ - ﾊﾞﾝｺｸ** Tel: 66-2-694-1351 **ﾍﾞﾄﾅﾑ - ﾎｰﾁﾐﾝ** Tel: 84-28-5448-2100	**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 **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