

AVR1605 : XMEGA ブートローダ即時開始の手引き

8ビット AVR マイクロコントローラ

要点

- XMEGA®ブートローダ
- AVROSP適合
- 応用例
- 自己プログラミング用試供応用
- フラッシュメモリとEEPROMの両方の読み書き
- 施錠ビットの読み書き
- ヒューズビットの読み込み

概要

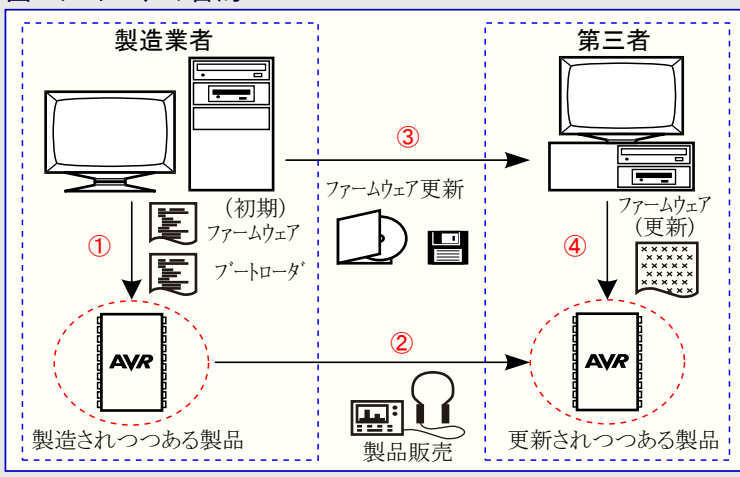
この応用記述はプログラムメモリ格納(SPM)命令を持つAVR®をどう自己プログラミングに構成設定し得るか、XMEGAシステムデバイスの1つ(例えばATxmega128A1)でブートローダ応用をどう使うかを記述します。試供応用は「AVR911 AVR公開ソース書き込み器」応用記述のAVR公開ソース書き込み器(AVROSP)が走行するPCとUART経由で通信します。これは外部書き込み器の必要なしにフラッシュ内への応用やEEPROMへのデータ格納とヒューズの読み書きを使用者に許します。例のソフトウェアは対象基板としてのSTK®600と共にATxmega128A1を使います。

携帯音楽再生器、ヘッドライヤー、シンであろうと、マイクロコントローラを含む電氣的な設計物は常にファームウェアと共に出荷されることを必要とします。多くの電気設計物が急速に進化するため、既に出荷または販売された製品を更新できるようにする必要が増えています。ハードウェアに対して変更をすることは、特に製品が既に最終顧客へ届いてしまっている場合に難しくなるでしょう。しかし、AVRシステムのようなフラッシュマイクロコントローラに基いた製品に於いて、ファームウェアは容易に更新することができます。

そのように構成設定された多くのAVRマイクロコントローラは求めに応じてファームウェア更新を受信してフラッシュメモリを書き換えられるブートローダを作成することが可能です。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。ブートローダプログラムはフラッシュメモリのブート領域内に配置されます。このプログラムはホストPCとの通信を処理してフラッシュとEEPROMの両方のプログラミングを容易にします。一旦プログラミングされると、フラッシュメモリのブートと応用の部分に異なる保護レベルを個別に適用することができます。従ってAVRは使用者に広範囲のメモリ保護を許す独特な柔軟性を提供します。

自己プログラミングについての一般的な情報に関しては「AVR109:自己プログラミング」応用記述を参照してください。

図1. ブートローダの目的



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。

1. 準備と実行

第1章はハードウェアを構成することによって準備と実行に対する基本段階を軽く済ませます。必要な構成と必要条件が関連情報と共に記述されます。

1.1. ハードウェア構成

1.1.項は実験用ハードウェアを構成するための段階的な手順を与えます。

この応用記述で提供されるブートローダ プログラムは使用者インターフェースとしてAVR公開ソース書き込み器(AVROSP)を使います。応用例は目的対象デバイスのフラッシュ メモリとEEPROMを読みまたは更新するための関数を実装します。STK600でATxmega128A1 AVRデバイスを使って、デバイスの施錠ビットの読みと更新、及びヒューズ ビットの読み込みも可能です。

STK600上でUSART信号をRS-232出力へ配線するためにPD2/PD3ピンをRXD/TXDピンに接続してください。

STK600にはCAN(オス型)とRS232(メス型)と記された2つのRS-232ポートがあります。STK600のRS232(メス型)ポートとPC間をシリアル ケーブルで接続してください。

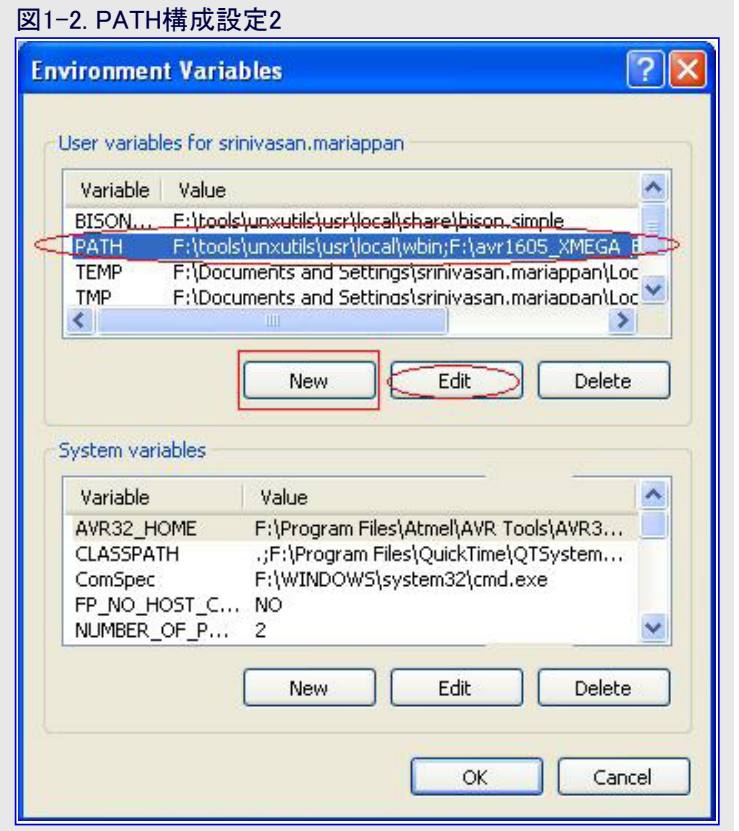
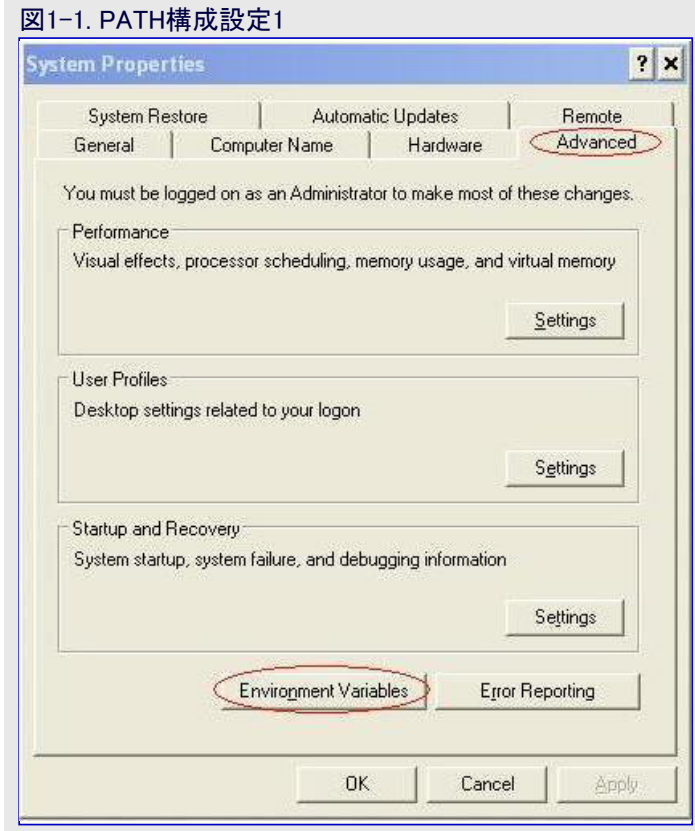
STK600上でブートローダ形態を許可するのにSW0マイクロ スwitchを使うためにPD4ピンをSW0ピンに接続してください。

正しい配線カードとソケット カードの組み合わせでデバイスを装着し、それをPCと接続するために、AVR Studio®のヘルプで利用可能なSTK600使用者の手引きを参照してください。

1.2. システム構成設定

1.2.項は実装開始前のPC構成についての情報を与えます。

変数名と値を作成することによってWindowsのパス(PATH)に(code¥AVROSP_Testで見つかる)xml_dev_filesフォルダ パスを含めてください。これはどのフォルダ位置からでも実行するためにAVROSPの既定位置をシステムが検索するための設定です。これが行われなかった場合、そのフォルダのAVROSP.exeを使えますが他のどの場所でもこれを呼び出すことができません。これは以下の図1-1、図1-2、図1-3で実演されるようにマイ コンピュータ アイコンの右クリックとプロパティ任意選択の選択によって設定することができます(変更を許可するのにコンピュータの再始動を必要とするかもしれません)。



1.3. デバッグ作業の開始

1.3.項はデバイス上でデバッグ作業を開始するための段階的な手順を与えます。

1.3.1. IARでのデバッグ作業の開始

IAR™(5.12C版またはそれ以降)を開始して“Open existing workspace”任意選択を選択してください。この応用記述と共にダウンロードした(以降で対象フォルダと呼ばれる)フォルダを閲覧してデバッグ用のbootloader.ewwを選択してください。コンパイルと構築によって異常や警告が全くないことを検証してください。

AVR Studioを開始し、bootldr.dbgを用いて新規プロジェクトを作成してください。デバイスとしてATxmega128A1、基盤としてJTAGICEmk IIを選択してください。

STK600とJTAGICEmk IIの両方が電源ONであることを確認してください。

AVR Studioでプログラミングダイアログを開き、JTAGまたはPDIインターフェースを用いてATxmega128A1に接続してください。ヒューズタブを選択し、BOOTRSTヒューズをブートローダリセットに設定してヒューズを書いてください。

STK600でSW0押下を保っている間に、AVR Studioでデバッグ作業を開始して応用を動かしてください(ツールの状態LEDを調べてください)。

プログラミング動作形態に於いてプログラムはUART経由でAVROSPからの指令を受け取ります。各指令は関連する作業を実行します。ブートローダプログラムによって認証されないどの指令もAVROSPへ送り返される“?”に帰着します。

エクスプローラを通して、多数のコマンド行例、フラッシュとEEPROM用のHEXファイル例、AVROSP.exe、ATxmega128A1用のデバイス記述XMLファイルが利用可能なAVROSP_Testフォルダを開いてください。

注: PCの既定RS232ポートはCOM1に設定され、STK600が他のポートに接続される場合、以降のバッチプログラムは変更が必要です。バッチファイルは次のとおりです。

- x128A1_chip_erase.bat
- x128A1_eeprom_dump.bat
- x128A1_eeprom_write_file.bat
- x128A1_flash_dump.bat
- x128A1_flash_write_file.bat

全てのバッチファイルに於いて先頭行のポート名COM1はCOMxに変更/置換されなければなりません。

以下で示されるバッチファイルの先頭行はノートブック应用到バッチファイルをドラッグ&ドロップすることによって見ることができます。

以下の指令はMSDOSコマンド行で与えられた時にWindowsでCOM1を供給します。

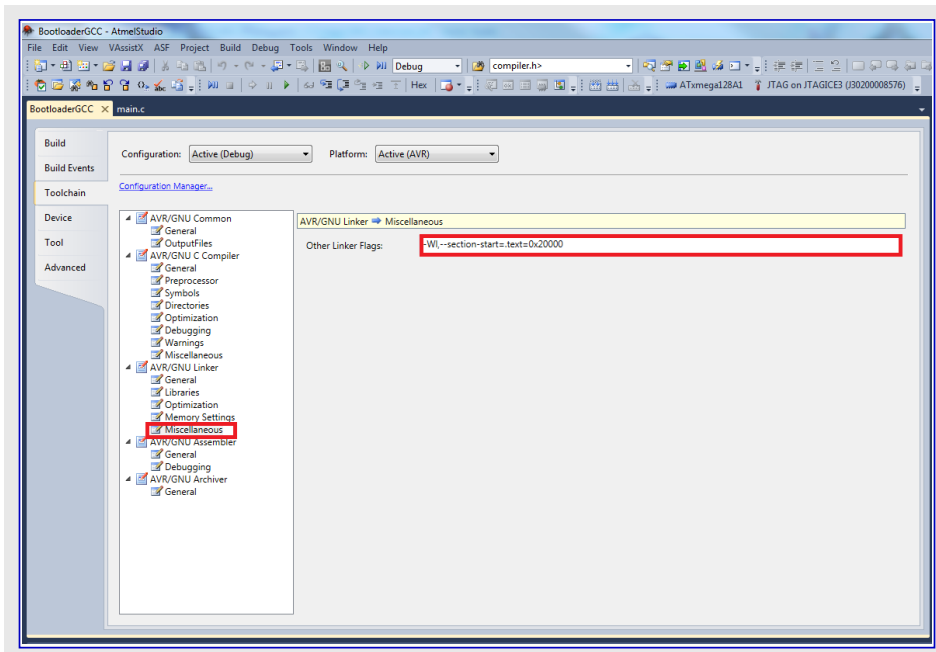
```
mode com1 Data=8 Parity=n Baud=9600 DTR=OFF RTS=OFF to=off
```

1.3.2. Atmel Studioでのデバッグ作業の開始

Atmel Studio(6.1またはそれ以降版)を開始して“Open project/solution”任意選択を選択してください。この応用記述と共にダウンロードした(以降で対象フォルダと呼ばれる、BootloaderGCC)フォルダを閲覧してデバッグ用のBootloaderGCC.cprojを選択してください。コンパイルと構築によって異常や警告が全くないことを検証してください。

注: 違うデバイスを使う場合、Atmel Studioでブートローダのバイトアドレスが定義されなければなりません。

Project→”プロジェクト名”Properties→Toolchain→AVR/GNU Linker→Miscellaneous→Other LinkerFlags:
“-Wl,--section-start=.text=0x20000” (ここでの20000はデータシートから得られたブートローダのバイトアドレスです。)



STK600とJTAGICEmk IIの両方が電源ONであることを確認してください。

AVR Studioでプログラミングダイアログを開き、JTAGまたはPDIインターフェースを用いてATxmega128A1に接続してください。ヒューズタイプを選択し、**BOOTRST**ヒューズをブートローダリセットに設定してヒューズを書いてください。

STK600で**SW0**押下を保っている間に、AVR Studioでデバッグ作業を開始して応用を動かしてください(ツールの状態LEDを調べてください)。

プログラミング動作形態に於いてプログラムはUART経由でAVROSPからの指令を受け取ります。各指令は関連する作業を実行します。ブートローダプログラムによって認証されないどの指令もAVROSPへ送り返される“?”に帰着します。

エクスプローラを通して、多数のコマンド行例、フラッシュとEEPROM用のHEXファイル例、AVROSP.exe、ATxmega128A1用のデバイス記述XMLファイルが利用可能な**AVROSP_Test**フォルダを開いてください。

注: PCの既定RS232ポートはCOM1に設定され、STK600が他のポートに接続される場合、以降のバッチプログラムは変更が必要です。バッチファイルは次のとおりです。

- x128A1_chip_erase.bat
- x128A1_eeprom_dump.bat
- x128A1_eeprom_write_file.bat
- x128A1_flash_dump.bat
- x128A1_flash_write_file.bat

全てのバッチファイルに於いて先頭行のポート名COM1はCOMxに変更/置換されなければなりません。

以下で示されるバッチファイルの先頭行はノートブック应用到にバッチファイルをドラッグ&ドロップすることによって見ることができます。

以下の指令はMSDOSコマンド行で与えられた時にWindowsでCOM1を供給します。

```
mode com1 Data=8 Parity=n Baud=9600 DTR=OFF RTS=OFF to=off
```

2. ブートローダとの通信

第2章はハードウェア構成を通してブートローダと通信する基本段階を簡単に片付けます。必要な構成と必要条件が関連情報と共に記述されます。

2.1. チップ消去

2.1項は“チップ消去”指令の実行方法を説明します。

デバイスがブートローダ領域で待機している段階であることを確実にしてください(STK600の外部**RESET**と**SW0**の両方を押下し、先に**RESET**を、その後に**SW0**を開放してください)。

現在の作業フォルダに於いて、**x128A1_chip_erase.bat**と呼ばれるバッチプログラムをダブルクリック(実行)してください。

バッチファイルはAVROSP使用時にフラッシュとEEPROMを消去するために次の指令を実行します。

```
AVROSP -dATxmega128A1 -e
```

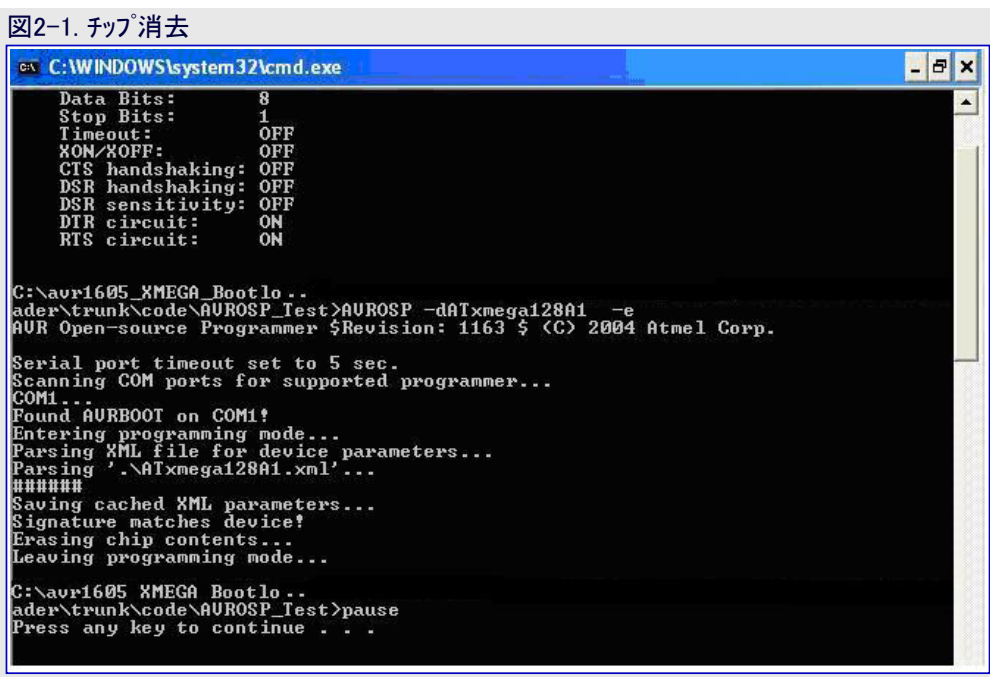


図2-1.チップ消去で示されるDOS窓が現れるでしょう。COM1に接続した後でAVROSPによってATxmega128A1に関してデバイス仕様ファイルが解析され、**AVROSP_Test**フォルダに出力ファイルが置かれます。更に後続操作によって識票が一致してチップ消去が実行されます。

- 1) 与えられたCOMポートが走査されます。
- 2) AVRBOOTと通信します。
- 3) プログラミング動作形態に移行します。
- 4) XMLファイルが解析され、現在の作業フォルダに新しいXMLファイルが生成されます。
- 5) 識票の一致が調べられます。
- 6) チップ消去が実行されます。
- 7) プログラミング動作形態を抜けます。

2.2. フラッシュへのファイル書き込み

2.2.項はファイルをフラッシュに書き込み/格納する方法を説明します。

デバイスがブートローダ領域で待機している段階であることを確実にしてください(STK600の外部RESETとSW0の両方を押下し、先にRESETを、その後SW0を開放してください)。

Windowsのエクスプローラでx128A1_flash_write_file.batを実行してください。このバッチファイルはフラッシュメモリの書き込みと照合のために次の指令を実行します。

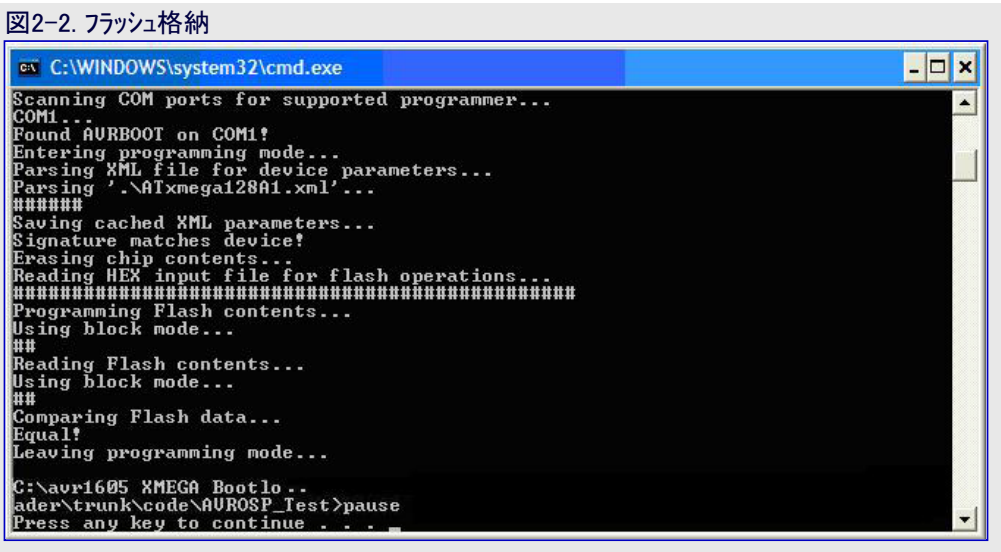
```
AVROSP -dATxmega128A1 -e -ifflash.hex -pf -vf
```

表2-1. 指令説明

指令	説明
-e	本指令はデバイスのフラッシュメモリを消去します。
-ifflash.hex	本指令はフラッシュ書き込み用の入力ファイルとして'flash.hex'ファイルを与えます。
-pf	本指令はフラッシュをプログラミング(書き込み)します。
-vf	本指令はプログラミング(書き込み)にフラッシュ内容を照合します。

照合指令による比較後、与えられた入力ファイルでフラッシュメモリプログラミング(書き込み)が何の異常もなく実行されたことの保証に、DOS窓での"Equal!"文字列が目されるべきです。

この"Equal!"文字列を見ることができる図2-2.フラッシュ格納を参照してください。



2.3. 0番地からのホスト返送応用実行

flash.hexはシリアル/RS232ポートを通して文字'E'を得た時に"Congratulations!"文字列を書くプログラムです。

このプログラムはリセット時にマイクロスイッチSW0が押されていないと、リセット後に応用領域へ飛びます。また、応用はBOOTRSTヒューズを応用ブートに設定してデバイスを再始動することによっても動かすことができます(他のデバッグ作業開始がチップ消去を実行するかもしれないことに注意してください)。

応用は直ぐにUSART0を通して文字'E'を待ちます。前節でフラッシュに応用が格納されているのが検証されているので、以下で記述されるように端末ソフトウェア(例えば、ハイパーターミナル)を通して文字'E'を書かなければなりません。

以下の経路によってWindowsに基く全てのシステムで利用可能なハイパーターミナルを開いてください(スタート⇒プログラム⇒アクセサリ⇒通信⇒ハイパーターミナル)。

この作業に対して何れかの名前(例えば、Test)を与え、COMポート設定を9600bps、8ビットデータ、パリティなし、1停止ビット、ハンドシェイクなしに構成設定してください。より多くの情報については図2-3.ハイパーターミナル名と図2-4.COMポート設定を参照してください。

図2-3. ハイパーターミナル名

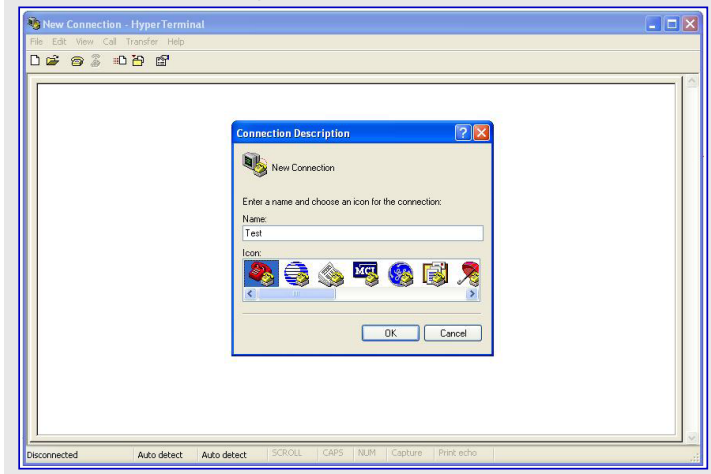
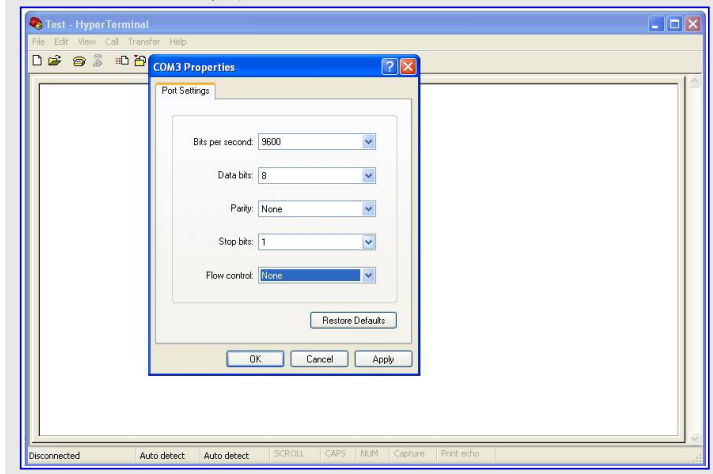


図2-4. COMポート設定



白紙空間に文字'E'を入力してください。

その返しに於いて、ブートローダを通してデバイスに格納された応用の正しい動作を保証するデバイスからの“Congratulations!”文字列を受信するでしょう。

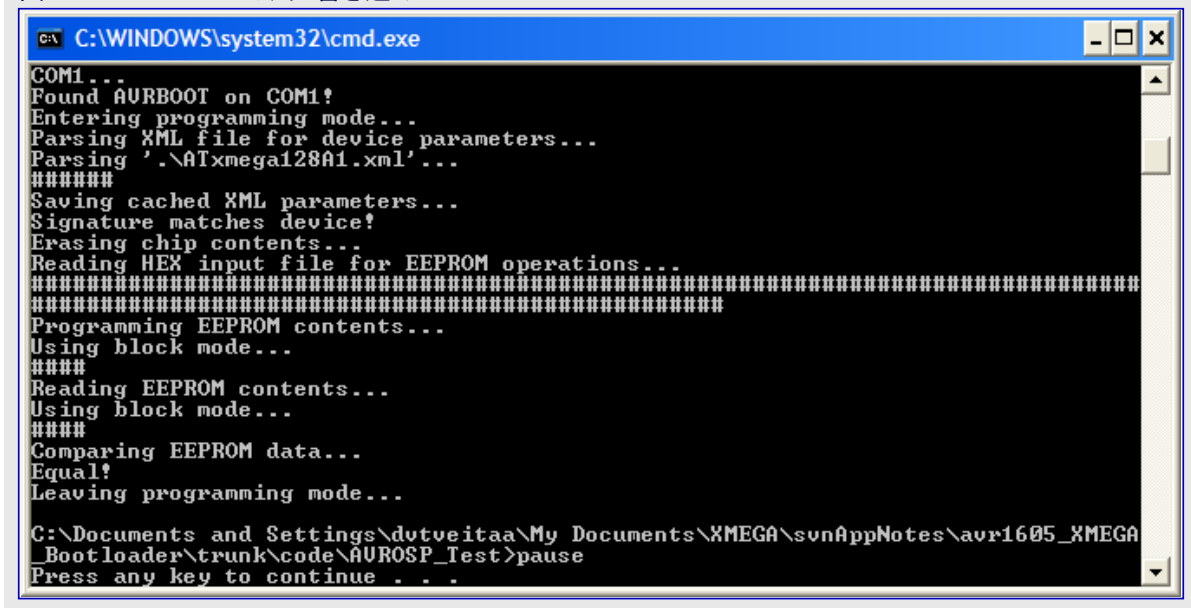
2.4. EEPROMへのファイル書き込み

デバイスがブートローダ領域で待機している段階であることを確実にしてください(STK600の外部RESETとSW0の両方を押下し、先にRESETを、その後SW0を開放してください)。

AVROSP_Testフォルダでx128A1_eeprom_write_file.batを実行してください。以下のAVROSP指令を使うことにより、“This is a test string to the XMEGA eeprom!”ASCII文字列を含むeeprom.hexファイルがEEPROMに書かれます。

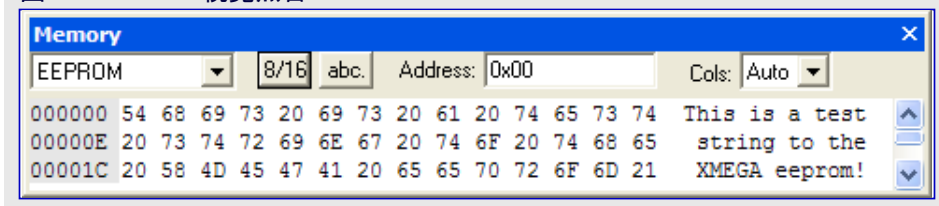
```
AVROSP -dATxmega128A1 -e -ieeprom.hex -pe -ve
```

図2-5. EEPROMへのファイル書き込み



AVR Studioでデバッグ作業を開始し、View(表示部)メニューからmemory view(メモリ表示部)を選択してください。eeprom.hexからの文字列が正しく書かれたことを検証するために、文字列が図2-6.EEPROM視覚照合のように0番地から配置されているのを検証してください。

図2-6. EEPROM視覚照合



2.5. メモリからのファイル読み出し

2.5.1. EEPROM

デバイスがブートローダ領域で待機している段階であることを確実にしてください(STK600の外部RESETとSW0の両方を押し、先にRESETを、その後にSW0を開放してください)。

EEPROMの内容をedump.hexファイルに読み出すために以下を与えるAVROSP指令を持つx128A1_eeeprom_dump.batを実行してください。

```
AVROSP -dATxmega128A1 -re -oeedump.hex
```

2.5.2. フラッシュ

デバイスがブートローダ領域で待機している段階であることを確実にしてください(STK600の外部RESETとSW0の両方を押し、先にRESETを、その後にSW0を開放してください)。

フラッシュの内容をfdump.hexファイルに読み出すために以下を与えるAVROSP指令を持つx128A1_flash_dump.batを実行してください。

```
AVROSP -dATxmega128A1 -rf -offdump.hex
```

3. Doxygen資料化

全てのソースファイルはDoxygenを使う自動資料生成用に準備されています。Doxygenは特別なキーワードを使ってソースコードを分析することによって、ソースコードから資料を作成するツールです。Doxygenについてのより多くの詳細に関しては<http://www.doxygen.org>を訪ねてください。予めコンパイルされたDoxygen資料は本応用記述に伴うソースコードと共に供給され、ソースコードフォルダのreadme.htmlファイルから利用可能です。

4. 改訂履歴

文書改訂	日付	注釈
8242A	2009年5月	初版文書公開
8242B	2014年7月	<ul style="list-style-type: none">1.3.1. IARでのデバッグ作業の開始項追加1.3.2. Atmel Studioでのデバッグ作業の開始項追加最新資料雛形に更新

Atmel®, Atmelロゴとそれらの組み合わせ、AVR®, AVR Studio®, Enabling Unlimited Possibilities®, STK®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のウェブサイトに表示する販売の条件と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AVR1605応用記述(Rev.8242B-07/2014)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識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。