AVR1913 : Xplain表示ファームウェアの開始に際して

要点

- 抵抗膜タッチパネル付きQVGA(320×240)LCD
- Xplain表示ソフトウェア ライブラリ
  - 図画ライブラリ：低位図画、ウィンドウ、ウィジェット
  - クリック可能なアイコンを持つデスクトップ応用
  - 写真観賞応用
- 小規模単純ファイル システム(TSFS:Tiny Simple File System)画像生成
- USB大容量記憶部装置へのファイル システム イメージ直接書き込み

1. 序説

Xplain表示基板はXplain系統の基板に対する表示拡張基板です。この基板の特徴は上面に抵抗膜接触フィルムを持つQVGA(320×240)LCDです。この基板が自身のマイクロ コントローラ付きでやって来ないので、これはAVR®マイクロ コントローラ付きのXplain系統の別の基板によって制御されるように設計されています。この応用記述ではXplain表示基板へ装着されるXplain系統の基板を持っていると仮定されます。

これはXplain表示ファームウェアの箱の中から使用者が開始できる各種応用に対するアイコンを持つデスクトップになるでしょう。応用の1つは簡単な画像枠で、それはXplain基板上のどれかの適切なフラッシュ メモリを走査してLCD画面上に支援する画像を表示します。使用者は画像内で誘導するのに抵抗膜接触を用い、自動スライドショーを開始/停止します。

この応用記述はXplain表示を目的対象としたソフトウェア枠組み応用をコンパイルしてアップロードする方法を通して進めます。それは小規模単純ファイル システム(TSFS)イメージの生成とそれのXplain フラッシュ メモリへのアップロードを通して案内します。

この応用記述にはXplain表示ソフトウェア枠組みを使用する方法の例として、よりもっと簡単な応用を通して進める追補もあります。

図1-1. Xplain表示デスクトップ応用

2. 開発必要条件

ソフトウェア枠組みは開発者がXplaine表示ソフトウェア枠組み構築システムの使用を始める前に予めインストールされたいくつかの重要なソフトウェアが必要です。本章は根本的に必要なものを網羅しあます。

2.1. AVRマイクロ コントローラ用の一連の道具

この開発で使用されるソース コードはGNU GCCとIAR™の一連の道具を用いて作られています。

支援される一連の道具の一覧は以下です。

- www.iar.seを通して入手可能なAVR用IAR Embedded Workbench®
- winavr.sf.netを通して入手可能な、avr-gccを提供するWinAVR

8ビット AVR® マイクロ コントローラ

応用記述

暫定

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。

Rev. 8294A-03/10, 8294AJ1-03/14

2.2. 必要なソフトウェア ツール

ソフトウェア枠組み構築システムは構築するマシン上にインストールされたいくつかの根本的なソフトウェア ツールを持つことを仮定します。これらのツールは最近のオペレーティング システムでのインストールが容易です。Micrsoft®のWindows®システムではこれらのソフトウェア応用がWinAVR一式でインストールされ、一方Linux®システムでは通常、それらは配布一括システムを用いて入手可能です。

ソフトウェア枠組みに関してインストールされなければならない有用なソフトウェアは以下です。

- 構築の本質的要素として度々参照される、make, sed, grep, sort, tac, bcなど
- sh – 命令解釈部(シェル)

開発者がXplain基板上のフラッシュ メモリに新しい内容のアップロードを望むなら、システムにPythonもインストールされなければなりません。Pythonは http://www.python.org/download からダウンロードしてインストールすることができます。

ソフトウェア枠組みはC:¥tmpフォルダを持つようなMicrosoft Windows使用者も必要とし、また、Microsoft Windowsのインストールが別のドライブに配置されている場合、そのドライブは最上位のtmpフォルダを持つ必要があります。

2.2.1. プログラミング ツール

目的対象デバイスへファームウェアをアップロードすることができるために、開発者は或るプログラミング ツールが必要です。8ビットAVRデバイスの使用者については次から選ぶことができます。

- avrdude – WinAVRの一部で殆どのLinux配布で利用可能です。
- Microsoft Windowsマシン上のAVR Studio®

また、AVRデバイスをインターフェースする能力があるハードウェア書き込み器も必要とされます。書き込み器は勿論、上で一覧にされたツールによって支援されなければなりません。ATMELによって提供される書き込み器は次の通りです。

- AVR Dragon
- AVRISP (AVRISPmk II によって置き換え)
- AVRISPmk II
- AVR ONE!
- JTAGICE (JTAGICEmk II によって置き換え)
- JTAGICEmk II

2.3. 開発環境

最近のx86コンピュータは8ビットAVRマイクロ コントローラ用のファームウェアを開発して展開するのに大変便利です。使用者はそれらのコンピュータが連結するツールと、使用するならば画像的な前処理部に対する最小必要条件に合うことを保証しなければなりません。

2.4. Xplain表示ソフトウェア枠組み

Xplain表示用のソフトウェア枠組みはこの応用記述と共に入手可能な圧縮書庫ファイルで供給されます。

2.5. Xplain基板制御器

複数のデバイスを持つXplain基板に関しては基板制御器に対して大容量記憶機能が許可されることが必要とされます。これは使用者が基板上のフラッシュ メモリにイメージ データの追加を望む時に必要とされます。Xplain基板上の基板制御器を網羅する独立した応用記述があります。

3. Xplain表示ソフトウェア枠組みフォルダ構造

ソフトウェア枠組みは最上位から多数のフォルダに分かれます。これはお互いに異なる系を指示するように行われ、一旦何処に何があるかを知れば、異なる副系で動くことをより容易にします。

3.1.1. appsフォルダ

ソフトウェア枠組みの各種部分を使用する全ての応用はapps¥フォルダに格納されます。各々の応用はそれ自身の副フォルダに属します。各応用がどう形態設定されて実装されるかについてのより多くの詳細は、13頁の「7. 追補A:UART出戻り応用への導入」で得られます。

3.1.2. archフォルダ

異種構造支援は共通構造の機能をarch¥フォルダに分割することによって実装されます。このフォルダは支援する構造の各々に対する副フォルダを含みます。Xplain表示ソフトウェア枠組み8ビットAVR構造が存在します。

構造部分はその構造に於いて全てのチップに対する共通部分を扱います。これはmain関数呼び出し前の始動アセンブリ コードの実装、コンパイラの抽象化、それと休止、文字列関数、割り込み処理、ソフトウェア割り込みのような基本的な構造関数を含みます。

3.1.3. boardフォルダ

board¥フォルダには多くの応用を動かすように形態設定できる、各基板に対して1つの副フォルダがあります。1つの応用が多数の基板で動き得るため、応用コードを含む各基板フォルダは各基板について知ることが必要です。

通常、各基板はLED、釦、SPIデバイスへのチップ選択、外部メモリなどのように基板へ付随する各種ハードウェアの構成を含みます。

3.1.4. buildフォルダ

使用者が応用を作る時に出力ファイルはbuild¥フォルダに格納されます。このフォルダは使用者がコードをコンパイルする応用と基板の量に依存して多くの副フォルダを含みます。オブジェクト ファイル、リスト ファイル、elfファイル、hexファイルと資料の全てがここになります。

GNU GCCでの表示実演(display-demo)応用コンパイル時の代表的な内容例の出力ファイルはbuild¥display-demo¥xplain¥GNUになります。

3.1.5. chipフォルダ

chip¥フォルダはソフトウェア枠組みによって支援される個別AVRデバイスの各々に対して1つの副フォルダを含み、それらのフォルダは支援される各チップに対するデバイスの特有の動きを含みます。

3.1.6. cpuフォルダ

チップと構造レベル間にはcpu¥フォルダがあり、ここでCPU系列に対して共通する定義とコードが構成されます。AVR XMEGA系列に対するxmega¥のように、各系列に対して1つの副フォルダがあります。

3.1.7. docフォルダ

doc¥フォルダに於いて使用者はドライバ、様々なソフトウェア階層、規約、そしてソフトウェア ユーティリティを通して利用可能なインターフェースに関する付加的な資料と共にソース コード ファイルを見つけるでしょう。通常、各ファイルの内容はDoxygen形式の資料で、それらの応用に対して使用者が資料を構築する時に既定で追加されます。

doc¥フォルダから直接資料を読むのは、それがDoxygen形式なので推奨されません。応用に対するDoxygen資料構築については「4.1. 利用可能なMake目的対象 – ヘルプ メニュー」項をご覧ください。

3.1.8. driversフォルダ

全てのハードウェア ドライバはdrivers¥フォルダに配置されます。ハードウェア ドライバはチップ上のハードウェアと、フラッシュ メモリのような外部ハードウェアの両方を含みます。ハードウェア ドライバは全ての構造に対する共通インターフェースを持ちますが、チップ レベルへ落とす様々な実装を持つかもしれません。

3.1.9. includeフォルダ

include¥フォルダはソフトウェア枠組みに於ける様々な応用プログラミング インターフェース(API)を定義する全てのヘッダ ファイルを含みます。これは使用者が応用を作る時にドライバ、ソフトウェア階層、コード ユーティリティへのインターフェースを含むべき場所です。

3.1.10. makeフォルダ

Xplain表示ソフトウェア枠組みはmake¥フォルダに配置されたそれ自身の構築システムを持ちます。定例的な使用者はここに於かれた各種ファイルのどれも変えることを必要とすべきではありません。

簡単に纏めると、全てのフォルダに於いてコードを含む最上位レベルからの1つのファイル、subdir.mkがあります。このファイルは使用者が選んだ形態設定に依存して何を構築するかを構築システムに告げます。

3.1.11. utilフォルダ

文字列ユーティリティ、待ち行列管理、一覧表などのような共通ユーティリティの全てはutil¥フォルダ内です。ユーティリティはどのドライバまたはハードウェアにも依存しませんが、それらは度々様々なドライバによって使用されます。

4. Xplain表示ソフトウェア枠組み構築システム

ソフトウェア枠組みは多数のMakefileから成る自身の構築システムを含みます。構築システムの形態設定と使用は定義されたシンボルを持つMakefileと形態設定ファイルを通して応用フォルダで行われます。

4.1. 利用可能なmake目的対象 – ヘルプ メニュー

構築システムは統合されたヘルプ メニューを持ち、これは異なるmake目的対象を一覧にして、それらの使用法に対して短い序説を与えます。ヘルプ メニューは応用構築時にmakeの動きを変えるのに使用される、最も頻繁に使用された変数も一覧にします。

例: Xplain表示ｿﾌﾄｳｪｱ枠組み構築ｼｽﾃﾑ ﾍﾙﾌﾟ目的対象

```
~ > make help

Usage: make [CONFIG_VARIABLE=VALUE]... [TARGET]...

Available targets:
  all             Default target if no target is specified.
                  This will build the default configuration
                  for this application.
  docs            Generate documentation for the selected
                  configuration.
  tar-archive     Pack all application files into an archive.
  archive-check   Pack all application files into an archive,
                  unpack them again and build the application
                  from the archive files.
  clean           Delete ELF file and all intermediate build
                  products.
  program         Build the ELF file and program it onto the
                  flash on the target.
  run             Start executing code on the target.
  reset           Reset the target.
  help            This help text.

Build system configuration variables:
  CONFIG          Configuration to build. If none is given the
                  default configuration is used. Each
                  configuration needs a corresponding config
                  file in the application directory. E.g. to
                  build the config-atevk1104.mk configuration,
                  run 'make CONFIG=atevk1104'.
  V               Set to 1 for verbose output.
  TOOLCHAIN       Toolchain to use for the build, GNU or IAR.
                  The default toolchain is specified in the
                  application Makefile.
  IAR_PATH        Path to IAR EWB installation.

Further information is available in doc/build_system_doc.c
```

```
~ > make help

使用法: make [CONFIG_VARIABLE=VALUE]... [TARGET]...

利用可能な目的対象:
  all             目的対象無指定の場合の既定目的対象。
                  これはこの応用に対する既定形態設定を
                  構築します。
  docs            選択した形態設定に対する資料を作成。

  tar-archive     全応用ﾌｧｲﾙを書庫に詰め込み。
  archive-check   全応用ﾌｧｲﾙを書庫に詰め込み、それらを
                  再び解いて、書庫ﾌｧｲﾙから応用を構築。

  clean           ELFﾌｧｲﾙと全ての中間構築生成物を削除。

  program         ELFﾌｧｲﾙを構築して目的対象上のﾌﾗｯｼｭ
                  に書き込み。
  run             目的対象上のｺｰﾄﾞの実行開始。
  reset           目的対象をﾘｾｯﾄ。
  help            このﾍﾙﾌﾟ文。

構築ｼｽﾃﾑ形態設定変数:
  CONFIG          構築のための形態設定。何も与えられない
                  場合、既定形態設定が使用されます。各形
                  態設定は応用ﾌｫﾙﾀﾞに対応する形態設定
                  ﾌｧｲﾙが必要です。例えば、config-atevk
                  1104.mk形態設定を構築するには、'make
                  CONFIG=atevk1104'を走らせてください。
  V               冗長出力に対して1を設定します。
  TOOLCHAIN       構築、GNU,IARに対して用いる連結ﾂｰﾙ。
                  既定連結ﾂｰﾙは応用のMakefileで指定され
                  ます。
  IAR_PATH        IAR EWBｲﾝｽﾄｰﾙへのﾊﾟｽ。

更なる情報はdoc¥build_system_doc.cで利用可能です。
```

4.2. 応用の構築

応用を構築するには、単にapps¥ﾌｫﾙﾀﾞ下の適切なﾌｫﾙﾀﾞに移行してmakeを入力してください。それが未だ存在していないなら、最上位のbuild¥ﾌｫﾙﾀﾞ下に新しいﾌｫﾙﾀﾞが作成されます。このﾌｫﾙﾀﾞには構築処理の結果が格納されます。この手順を1度実行した後、その後の構築は前のように応用ﾌｫﾙﾀﾞからmakeを走らせるか、または構築ﾌｫﾙﾀﾞからmakeを走らせるかのどちらでも行うことができます。

各応用は使用するための既定連結ﾂｰﾙだけでなく、他に指定されなければ、応用が構築されるための既定形態設定ﾌｧｲﾙを持ちます。異なる形態設定を持つ応用を構築するには、例えば以下のようにmake走行時に望む形態設定の名前に対して単にCONFIG環境変数を設定してください。

```
make CONFIG=xplain
```

既定と異なる連結ﾂｰﾙを使用するには、例えば以下のようにTOOLCHAIN変数をGNUまたはIARのどちらかに設定してください。

```
make TOOLCHAIN=GNU
```

構築の結果は最上位build¥応用名¥形態設定名¥連結ﾂｰﾙ名ﾌｫﾙﾀﾞに置かれます。例えば、Xplain基板とIAR連結ﾂｰﾙを用いて表示実演(display-demo)応用構築時、その結果はbuild¥display-demo¥xplain¥IARﾌｫﾙﾀﾞになります。

既定での構築処理は殆ど無口で、構築されつつあるﾌｧｲﾙの一覧化だけで、それらを構築するのに全部の命令行が使用される訳ではありません。全部の命令行を見るには、V環境変数を1に設定することによって冗長形態を許可してください。

```
make V=1
```

4.3. 応用のﾌﾟﾛｸﾞﾗﾐﾝｸﾞと走行

目的対象ﾁｯﾌﾟに応用をﾌﾟﾛｸﾞﾗﾐﾝｸﾞする(書き込む)には、apps¥ﾌｫﾙﾀﾞ下の適切なﾌｫﾙﾀﾞに移行して、make programを入力してください。これは必要ならば構築して、ﾌｧｰﾑｳｪｱを目的対象にｱｯﾌﾟﾛｰﾄﾞします。各種形態設定の選択は応用構築と同様に、CONFIG環境変数を用いて行われます。

目的対象上の応用の走行はmake run目的対象を動かすことによって行われます。

例: Xplain基板上のATxmega128A1ﾁｯﾌﾟへの表示実演(dispaly-demo)応用のﾌﾟﾛｸﾞﾗﾐﾝｸﾞ(書き込み)

```
cd apps¥display-demo
make CONFIG=xplain program
```

使用者が応用を目的対象にﾌﾟﾛｸﾞﾗﾐﾝｸﾞする(書く)時に、ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ ﾂｰﾙ、書き込み器ﾊｰﾄﾞｳｪｱ、書き込み器ﾎﾟｰﾄを選ぶためのいくつかの環境変数が設定されなければなりません。

例: ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ ﾂｰﾙとしてavrdudeを用いてXplain基板上のATxmega128A1ﾁｯﾌﾟへの表示実演応用のﾌﾟﾛｸﾞﾗﾐﾝｸﾞ(書き込み)

```
make CONFIG=xplain AVR_PROGTOOL=avrdude CONFIG_PROGRAMMER=jtagicemkii CONFIG_PROG_PORT=usb program
```

- CONFIG – ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ(書き込み)のための形態設定の名前、ここではXplain基板です。
- AVR_PROGTOOL – 8ﾋﾞｯﾄAVRﾃﾞﾊﾞｲｽに使用するためのﾌﾟﾛｸﾞﾗﾐﾝｸﾞ ﾂｰﾙの名前、ここではavrdudeです。32ﾋﾞｯﾄAVRﾃﾞﾊﾞｲｽに関して、これはAVR32_PROGTOOLと名付けられ、代表的にavr32programに設定されます。
- CONFIG_PROGRAMMER – 使用する書き込み器の名前、ここではJTAGICEmk Ⅱが使用されます。構築ｼｽﾃﾑは次のﾂｰﾙ、avrdragon、avrone、avrisp、avrispmkii、jtagice、jtagicemkiiを支援します。
- CONFIG_PROG_PORT – 書き込み器が接続されるPCの通信ﾎﾟｰﾄ、ここではUSBﾎﾟｰﾄです。書き込み器のﾎﾟｰﾄは代表的に次の¥dev¥ttySn、¥dev¥ttyUSBn、comN、またはusbの1つです。

命令行を簡略化するために使用者は最上位config.mkに、上で詳述した環境変数を追加することができます。「4.4.1. 最上位形態設定ﾌｧｲﾙ」項をご覧ください。

4.4. 形態設定

各応用は構築ｼｽﾃﾑに対して1つ以上の形態設定ﾌｧｲﾙを提供しなければなりません。各支援基板に対してconfig-CONFIG.mkと名付けられます。CONFIGは一般的に基板名、時々特定の機能と組み合わせた基板名を表します。

例: Xplain用表示実演(display-demo)

```
config-xplain.mk
```

全ての応用は既定基板に対して形態設定ﾌｧｲﾙを提供しなければならず、他のどの基板用の形態設定ﾌｧｲﾙも任意選択です。

形態設定ﾌｧｲﾙはmakefile変数定義の全種類を含むかもしれませんが、それの根本的な目的は形態設定変数を指定することです。これらの変数は構築ｼｽﾃﾑとｿｰｽ ﾌｧｲﾙの両方で利用可能で、故にそれらは以下のような少しの特別な規則に従うことが必要です。

- 変数名は'CONFIG_'で始まらなければなりません。
- 変数名は間に空白なしで'='文字と値が後続しなければなりません。
- 文字列値は二重引用符によって囲まれなければなりません。
- ﾌﾞｰﾙ代数値は'y'または'n'のどちらかでなければなりません。
- 整数値は何れかの有効なC書式を用いて表されなければなりません。

構築処理中、autoconf.hﾌｧｲﾙを生成するために供給元としてこのﾌｧｲﾙが使用され、そしてこれは全てのｿｰｽ ﾌｧｲﾙに於いて暗黙的にｲﾝｸﾙｰﾄﾞされます。変数は以下のように変換されます。

- 'y'に設定されたﾌﾞｰﾙ変数は1として定義されるﾌﾟﾘﾌﾟﾛｾｯｻ ｼﾝﾎﾞﾙになります。
- 'n'に設定されたﾌﾞｰﾙ変数は定義されず、故に#ifdefはﾌﾞｰﾙ代数形態設定変数の値を検査するのに使用することができます。
- 文字列と整数の変数は文字列変数に対して引用符を含む、それら各々の直解値として定義されるﾌﾟﾘﾌﾟﾛｾｯｻ ｼﾝﾎﾞﾙになります。

config.mkと名付けられた支援される基板、ﾁｯﾌﾟ、CPU、構造に対する同様の形態設定ﾌｧｲﾙもあり、各々、board-,chip-,cpu-,arch-特定副ﾌｫﾙﾀﾞに置かれます。これらは応用形態設定ﾌｧｲﾙによって無効にすることができる妥当な既定を提供します。

4.4.1. 最上位形態設定ﾌｧｲﾙ

より簡単な入力の構築とﾌﾟﾛｸﾞﾗﾐﾝｸﾞ命令を作るため、使用者は最上位のconfig.mkﾌｧｲﾙを追加し、応用間で共用される静的な定義を追加することができます。最上位のconfig.mk.exampleﾌｧｲﾙは資料化された利用可能な全ての定義を持ち、雛形として使用することができます。

4.5. 最適化

ｺﾝﾊﾟｲﾗとﾘﾝｶの最適化は形態設定ｼﾝﾎﾞﾙを通して制御されます。既定では全てのﾌｧｲﾙが最大量最適化でｺﾝﾊﾟｲﾙされますが、これは基礎応用毎に独自設定することができます。

量の代わりに速度で最適化するには、CONFIG_OPTIMIZE_SIZEを'n'に設定してください。異なる最適化ﾚﾍﾞﾙを選択するには、CONFIG_OPTIMIZE_LEVELを望むﾚﾍﾞﾙ(0～3の数値)に設定してください。下表は様々な可能性を要約します。

表4-1. CONFIG_OPTIMIZE_SIZEとCONFIG_OPTIMIZE_LEVELの選択からのGCCとIARの任意選択

CONFIG_OPTIMIZE_SIZE	CONFIG_OPTIMIZE_LEVEL	GCC任意選択	IAR任意選択
y	0	-Os	-z2
y	1	-Os	-z3
y	2	-Os	-z6
y	3	-Os	-z9
n	0	-O0	-s2
n	1	-O1	-s3
n	2	-O2	-s6
n	3	-O3	-s9

加えて、GNU連結ﾂｰﾙ使用時に既定でﾘﾝｸ時最適化の2つの形式が許可されます。それらは次の通りです。

- CONFIG_RELAX：ﾘﾝｸ時軽減。より短い代替によって或る命令の置き換えを試みます。
- CONFIG_GC_SECTIONS：ﾘﾝｸ時塵掃除。これは参照されない関数とﾃﾞｰﾀ項目の全てを消去します。

これらの最適化のどれかを禁止するには、対応する形態設定ｼﾝﾎﾞﾙを'n'に設定してください。

4.6. 副ﾌｫﾙﾀﾞのMakefile

各副ﾌｫﾙﾀﾞはｵﾌﾞｼﾞｪｸﾄに構築すべきｿｰｽ ﾌｧｲﾙや時々構築に於いてｲﾝｸﾙｰﾄﾞするための追加の副ﾌｫﾙﾀﾞを指定するMakefileの断片を含みます。このMakefileの断片は一般的にsubdir.mkと呼ばれますが、この規則に対して次のようないくつかの例外があります。

- 応用副ﾌｫﾙﾀﾞのMakefileは$(app).mkと呼ばれます。
- chip-,cpu-,arch-,board-特定Makefileは各々、chip.mk、cpu.mk、arch.mk、board.mkと呼ばれます。

通常、副ﾌｫﾙﾀﾞMakefileは非常に小さく、単一変数に構築するためにどのｵﾌﾞｼﾞｪｸﾄかを指定するobj-yを追加するだけですが、更に副ﾌｫﾙﾀﾞのMakefileによって変えられるかもしれない次のような他の変数があります。

- cppflags-toolchain-y：全てのﾌｧｲﾙに対するCﾌﾟﾘﾌﾟﾛｾｯｻ ﾌﾗｸﾞ
- cflags-toolchain-y：全てのﾌｧｲﾙに対するCｺﾝﾊﾟｲﾗ ﾌﾗｸﾞ
- ldflags-toolchain-y：最終ﾘﾝｸに対するﾘﾝｶ ﾌﾗｸﾞ
- ldlibs-toolchain-y：最終ﾘﾝｸでｲﾝｸﾙｰﾄﾞされるﾗｲﾌﾞﾗﾘ

これら全ての変数が何故接尾子-yを持つかの理由は条件ｺﾝﾊﾟｲﾙと条件ﾌﾗｸﾞの支援をより容易にするためです。例えば、CONFIG_USBが'y'に設定されている時にだけ、或るｵﾌﾞｼﾞｪｸﾄが構築されるべきなら、これは以下のようにして達成することができます。

```
obj-$(CONFIG_USB) += drivers¥usb¥usb.o
```

4.7. 応用によって提供されるMakefile

4.7.1. 最上位応用Makefile

apps¥副ﾌｫﾙﾀﾞ下で見つかる各応用はそれの特定の応用を構築するのに使用することができるそれ自身の最上位Makefileを含ます。通常、それは非常に簡単で、構築ｼｽﾃﾑの核に作業の大多数を付与します。

例: apps¥display-demo¥Makefile

```
src                     := ..¥..
app                     := display-demo
DEFAULT_CONFIG          := xplain
DEFAULT_TOOLCHAIN       := GNU

include $(src)¥make¥app.mk
```

最上位応用Makefileによって以下の変数が定義されなければなりません。

- src：最上位ｿｰｽ ﾌｫﾙﾀﾞへの相対ﾊﾟｽ。
- app：応用の名前。$(src)¥apps¥$(app)はMakefileが属するﾌｫﾙﾀﾞに対して解決されなければなりません。
- DEFAULT_CONFIG：使用者がどんな特定の形態設定も指定しない時に対する構築のための形態設定の名前。
- DEFAULT_TOOLCHAIN：使用者がどんな特定の連結ﾂｰﾙも指定しない時に使用されるGNUまたはIARのどちらかの連結ﾂｰﾙ。

これらの変数定義後、残りを処理するためにMakefileは単純に$(src)¥make¥app.mkをｲﾝｸﾙｰﾄﾞすべきです。

4.7.2. 応用副ﾌｫﾙﾀﾞMakefile

応用は何れかの応用特有のｵﾌﾞｼﾞｪｸﾄ ﾌｧｲﾙとﾌﾗｸﾞを指定する$(app).mkと名付けられたﾌｧｲﾙをｲﾝｸﾙｰﾄﾞしなければなりません。このﾌｧｲﾙは6頁の「4.6. 副ﾌｫﾙﾀﾞのMakefile」項で記述される構文規則に従います。

5. Xplain表示応用

Xplain表示実演応用のためのｿｰｽ ｺｰﾄﾞはこの応用記述と連携する圧縮された書庫ﾌｧｲﾙにあります。

5.1. ﾀﾞｳﾝﾛｰﾄﾞ、ｺﾝﾊﾟｲﾙ、ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ

以下の段階的案内はXplain表示基板用の基のﾌｧｰﾑｳｪｱを再構築します。

1. www.atmel.com/products/AVR⇒Application Notes⇒AVR1913 Display Xplained Firmware - Getting Started containing the Display Xplained source codeから圧縮された書庫ﾌｧｲﾙをﾀﾞｳﾝﾛｰﾄﾞして圧縮を解除してください。
2. お気に入りのｺﾏﾝﾄﾞ行ｺﾝｿｰﾙを開き、圧縮された書庫ﾌｧｲﾙから抽出したﾌｫﾙﾀﾞ内に変更してください。
3. apps¥display-demoﾌｫﾙﾀﾞに移行してください。
4. 使用したい形態設定の種類を構築ｼｽﾃﾑに告げることによって応用をｺﾝﾊﾟｲﾙしてください。これはCONFIGｼﾝﾎﾞﾙを設定することによって行われます。追加のﾍﾙﾌﾟに関しては、ｺﾏﾝﾄﾞ行上でmake helpを入力してください。Xplain基板に対するｺﾝﾊﾟｲﾙ法については下の例をご覧ください。

```
make CONFIG=xplain
```

5. 応用2進ｺｰﾄﾞ出力は最上位build¥display-demo¥xplain¥GNUﾌｫﾙﾀﾞ内で、display-demo.elfとdispaly-demo.hexが多分最も興味あるものです。使用者がGNU連結ﾂｰﾙを使用しない場合、ﾊﾟｽのGCC部分は適切に変更されなければなりません。また、別の形態設定が使用された場合、基板名も変更されなければなりません。
6. ELFまたはHEXﾌｧｲﾙをXplain基板にﾌﾟﾛｸﾞﾗﾐﾝｸﾞする(書く)ためにお気に入りのﾌﾟﾛｸﾞﾗﾐﾝｸﾞ ｿﾌﾄｳｪｱとﾂｰﾙを使用してください。

5.1.1. avrdudeを用いた応用のﾌﾟﾛｸﾞﾗﾐﾝｸﾞ

この例に於いて使用者はXplain基板に接続されたJTAGICEmk Ⅱを持つと仮定されます。その後に以下の命令を用いてATxmega128A1ﾃﾞﾊﾞｲｽにdisplay-demo.hexﾌｧｲﾙをｱｯﾌﾟﾛｰﾄﾞすることが可能です。

```
make CONFIG=xplain PROGTOOL=avrdude CONFIG_PROGRAMMER=jtagicemkii CONFIG_PROG_PORT=usb program
```

他のﾌﾟﾛｸﾞﾗﾐﾝｸﾞ ﾂｰﾙに関して、使用者は正しいﾂｰﾙと通信ﾎﾟｰﾄを選択する適切な引数を渡さなければなりません。

5.1.2. AVR Studio 4を用いた応用のﾌﾟﾛｸﾞﾗﾐﾝｸﾞ

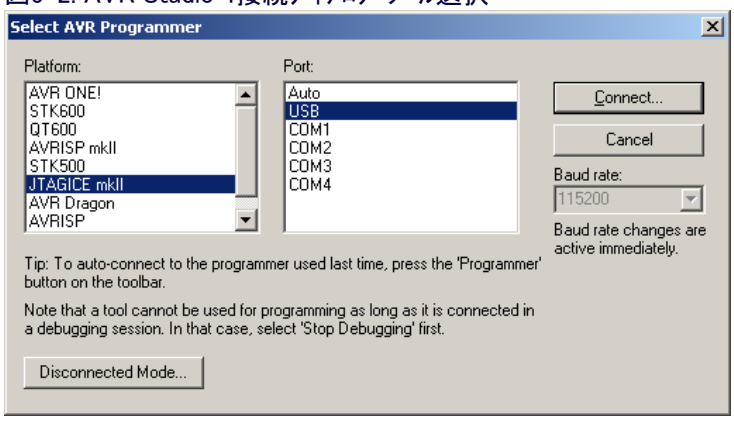
AVR Studio® 4を用いてdisplay-demo.hexﾌｧｲﾙをﾌﾟﾛｸﾞﾗﾐﾝｸﾞする(書く)ために、使用者は以下の段階を通して行うことが必要です。

最初に接続ﾀﾞｲｱﾛｸﾞが開かれることが必要で、これはﾂｰﾙﾊﾞｰ上に於いて釦をｸﾘｯｸするか、またはToolsﾒﾆｭｰ⇒program AVR⇒Connect…からそれを選ぶことによって行われます。

図5-1. ﾂｰﾙﾊﾞｰ上のAVR Studio接続ﾀﾞｲｱﾛｸﾞ釦

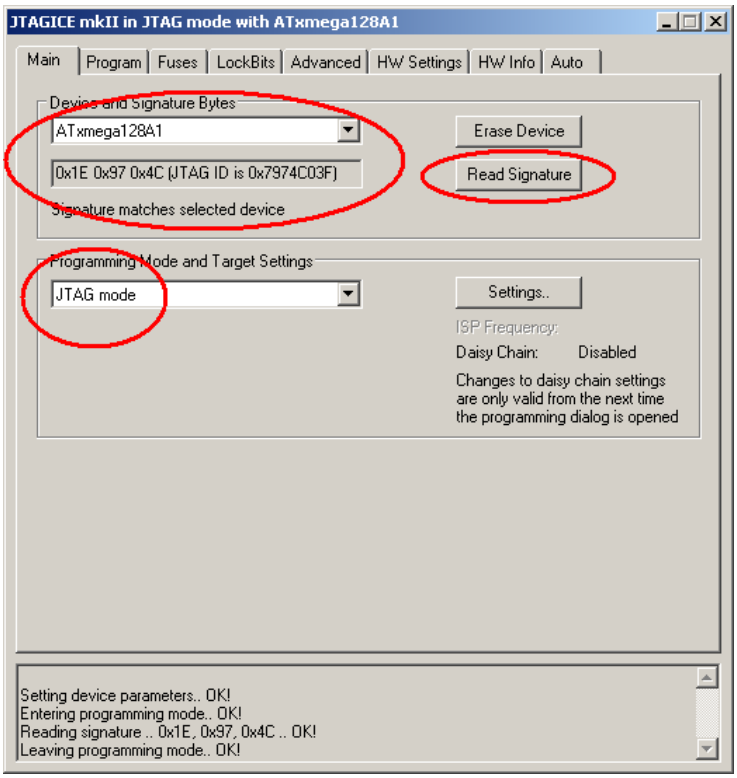
直前で言及した接続ﾀﾞｲｱﾛｸﾞ釦をｸﾘｯｸした時に、右図で見られるような接続ﾀﾞｲｱﾛｸﾞがﾎﾟｯﾌﾟｱｯﾌﾟします。この接続ﾀﾞｲｱﾛｸﾞに於いて、使用者は適切なﾂｰﾙと、そのﾂｰﾙが接続されるﾎﾟｰﾄも選択しなければなりません。この例はUSBﾎﾟｰﾄに接続するJTAGICEmk Ⅱを使用します。

図5-2. AVR Studio 4接続ﾀﾞｲｱﾛｸﾞ ﾂｰﾙ選択



デバイスとインターフェースを始める前に、使用者はMainタブのドロップダウン メニューで目的対象デバイスを選択しなければなりません。デバイス選択後、次のドロップダウン ボックスでプログラミング動作形態が選択されなければなりません。そして最後に、デバイス識票を読むことによって接続が終わって動いていることを確認してください。

図5-3. AVR Studio 4ツールとデバイスの構成設定



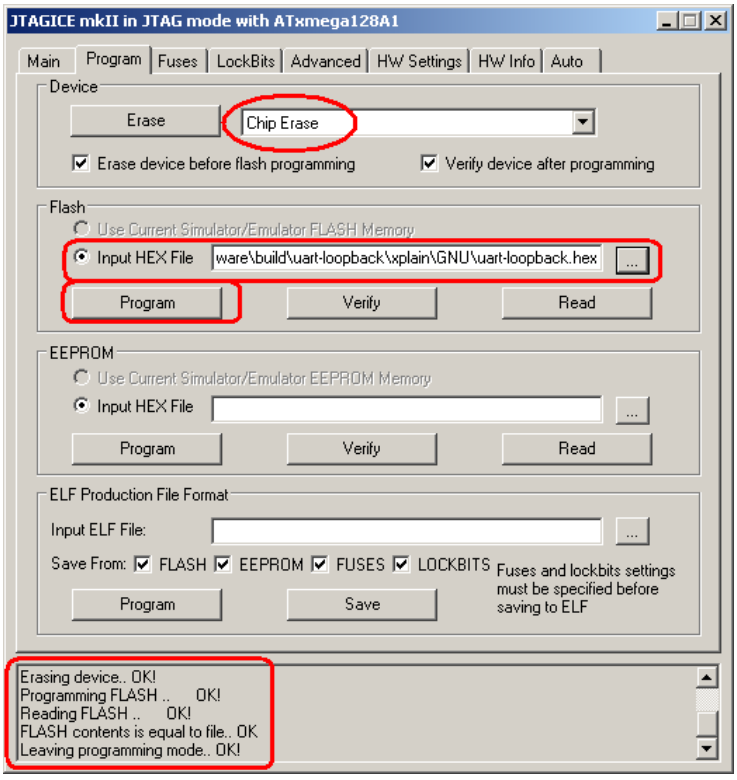
プログラミングは接続ダイアログ ウィンドウのProgramタブで行われます。どの施錠ヒューズをも解除するために、先にチップ消去を行うことに気付くかもしれません。これはタブのDevice区分で行われます。

プログラミングに関して使用者はHEXファイルが置かれている場所への指示が必要です。ファイル閲覧部を開くためにInput HEX File領域の後ろの小さな箱をクリックし、uart-loopback.hexファイルの場所を示してください。

最後にデバイスのプログラミングを開始するためにProgram釦をクリックしてください。タブの下部の状態領域で成功か失敗を監視することができます。

デバイスのプログラミング時に何れかの問題を経験した場合、AVR Studio 4のヘルプを調べることが高く推奨されます。これはメニューバーのHelp⇒AVR Tools Users Guideにあります。

図5-4. AVR Studio 4ツールのHEXファイル プログラミング タブ



5.2. Xplain基板フラッシュ メモリへの画像アップロード

本項についてはXplain基板がUSBに接続された時にUSB大容量記憶インターフェースを持つと仮定されます。、単一チップXplain基板または主チップがUSB機能を制御する、USB基板制御器付きXplain基板(12頁の「6.1. Xplain基板制御器」項をご覧ください)については、display-demoファームウェアがUSB大容量記憶機能を許可します。

この応用記述と連携する書庫ファイルで、display-demo応用で使用される画像ファイルを見つけるでしょう。これらのファイルはビットマップで利用可能で、Xplain表示LCD用に仕立てられたピクセルに対して生の16ビットのイメージ ファイルに変換されます。圧縮ファイルで配置された書庫はTSFSやイメージとして略称される生の小規模単純ファイル システムも予め作成され、これはXplain基板のフラッシュ メモリに直接プログラミングすることができます。小規模単純ファイル システムを生成する方法についての更なる深い知識は5.2.1.項です。

5.2.1. 小規模単純ファイル システム イメージ ファイルの生成法

圧縮ファイル書庫はTSFSイメージ ファイルを生成するためのツールを含み、このツールはmkfs.tsfsと呼ばれ、tsfs-toolsフォルダに配置されています。mkfs.tsfsを使用する方法について指示は--helpパラメータ付きで実行する時に表示されます。

```
~ > mkfs.tsfs --help
Usage: mkfs.tsfs [OPTIONS] [FILES]
Description
     mkfs.tsfs will combine all the files listed in FILES into a TSFS image
     file with a TSFS structure.  By default mkfs.tsfs will write to
     'raw.out', unless a file is specified by the -o option.

Options
     -o FILE
            Write TSFS image to file FILE.  Default is 'raw.out'.
     -h, --help
            Display help text.
```

```
~ > mkfs.tsfs --help
使用法: mkfs.tsfs [任意選択] [FILES]
説明
     mkfs.tsfsはFILESで一覧にされた全ファイルをTSFS構造でTSFSファイル イメージに結合します。
     ファイルが-o任意選択によって指定されていなければ、既定のmkfs.tsfsは'raw.out'に書きます。

任意選択
     -o FILE
            TSFSイメージをFILEファイルに書きます。既定は'raw.out'です。
     -h, --help
            ヘルプ文を表示します。
```

デスクトップ応用でXplain基板上のDataFlash用のファイル システム イメージを生成する例について、mkfs.tsfs用の構文法は以下のようなものに見えるでしょう。

```
~ > mkfs.tsfs -o dataflash.img icon* picture*
Generating Tiny Simple File System image...
Completed: image file `dataflash.img` done, wrote 614976 bytes.
```

これはUSBインターフェースを通してXplain基板上のDataFlashへアップロードする準備が整ったdataflash.imgファイルを生成します。

5.2.2. 基板フラッシュ メモリへのファイル システム イメージのアップロード法

mkfs.tsfsツールで作成されたイメージはdisplay-demo応用に対して画像、アイコンなどを使用することができるようにXplain基板のフラッシュ メモリにアップロードされなければなりません。このファイル システム イメージは生のイメージなので、USB大容量記憶塊装置に直接書かれなければなりません。display-demo応用はFATと同様のファイル システムを理解しません。

塊装置への直接書き込みがMicrosoft WindowsやLinuxのオペレーティング システム間でかなり異なるため、下のXplain基板上のフラッシュ メモリをプログラミングする(書く)方法には各オペレーティング システムに1つで、2つの異なる説明があります。

5.2.2.1. Microsoft Windows使用者

Microsoft Windowsオペレーティング システムは塊装置へ直接書き込む容易なアクセスを持ちません。これを容易にするためにrawwriteと呼ばれるツールがtsfs-toolsフォルダにあります。このツールは塊装置に2進ファイルを直接書く能力があります。

rawwriteの使用法についての指示は--helpパラメータ付きで実行される時に表示されます。

```
~ > rawwrite --help
Usage: rawwrite [OPTIONS] [FILE]
Description
     rawwrite will write the FILE file to a block device. When the user
     executes rawwrite the application will scan the system for possible
     block devices and present them in a list. The user must then select the
     appropriate block device in the list.

     rawwrite will request a final confirmation before continuing with the
     direct write to the block device.

     WARNING! rawwrite writes directly to the block device, potentially
     destroying data already present on the device. If the user wrongly
     selects the block device used by the operating system installation the
     user will destroy vital data and possibly make the system unusable.

Options
     --expert
            Enable expert mode. This mode will list ALL the block devices on
            the system and let the user freely select the appropriate target
            and proceed with the write.

     -h, --help
            Display help text.
```

```
~ > rawwrite --help
使用法: rawwrite [OPTIONS] [FILE]
説明
     rawwriteは塊装置へFILEファイルを書きます。使用者がrawwriteを実行すると、応用は可能な塊装置に関してシステムを走査し、それを一覧に表示します。そして使用者は一覧の適切な塊装置を選択しなければなりません。

     rawwriteは塊装置への直接書き込みを継続する前に最後の確認を求めます。

     警告! rawwriteは塊装置に直接書き、装置上に既に存在するデータを潜在的に破壊します。使用者がオペレーティング システムのインストールによって使用される塊装置を誤って選択した場合、使用者は重要なデータを破壊し、システムを使用不能にする可能性があります。

任意選択
     --expert
            熟練者動作形態許可。この動作形態システム上の塊装置を全て一覧にし、使用者に適切な目的対象を自由に選択させて書き込みを進めます。

     -h, --help
            ヘルプ文を表示します。
```

ﾃﾞｽｸﾄｯﾌﾟ応用でXplain基板上のDataFlashにdataflash.imgﾌｧｲﾙを書く例について、rawwrite用の構文法は以下のようなものに見えるでしょう。

```
~ > rawwrite dataflash.img
rawwrite is scanning system for block device targets:
1) ATMEL Corp. Xplain-BC
Please select block device target [number]: 1
Are you sure you want to write 'dataflash.img' to block device ATMEL Corp.
Xplain-BC? [yes/no] yes
Writing 'dataflash.img' to block device Atmel Corp. Xplain-BC
Write operation completed successfully
```

```
~ > rawwrite dataflash.img
rawwrite塊装置目的対象についてｼｽﾃﾑを走査中です:
1) ATMEL Corp. Xplain-BC
目的対象の塊装置を選択してください。 [番号]: 1
塊装置ATMEL Corp. Xplain-BCに'dataflash.img'を書くことを望んだのは確かですか? [yes/no] yes
塊装置ATMEL Corp. Xplain-BCに'dataflash.img'を書いています。
書き込み操作は成功裏に完了しました。
```

これはUSB大容量記憶ｲﾝﾀｰﾌｪｰｽを通してXplain基板上のDataFlashにdataflash.imgﾌｧｲﾙを書きます。

rawwriteﾂｰﾙは使用者が書き込み操作を進めるための入力を与えるまで破壊的などんな操作も行いません。この問いはﾌﾟﾛｸﾞﾗﾑ走行中に与えられます。

rawwriteﾂｰﾙは塊装置に直接書きます。装置一覧から正しい塊装置の選択が不確かな場合、ｺﾝﾋﾟｭｰﾀ上のﾃﾞｰﾀを破壊するでしょう。

5.2.2.2. Linux使用者

Linux使用者については塊装置へ直接ﾃﾞｰﾀを書く能力があるddと呼ばれるﾂｰﾙがあります。dd応用は一般的に配布ｺｱ ﾕｰﾃｨﾘﾃｨ1式の一部です。

先ず最初に使用者はXplin基板を表す装置節点(ﾉｰﾄﾞ)を識別しなければなりません。装置挿入後、Linuxｼｽﾃﾑ装置を探して適切なｼｽﾃﾑ節点を作ります。これはdmesgを入力することによって観測することができます。

```
usb 1-8.4: new full speed USB device using ehci_hcd and address 60
usb 1-8.4: configuration #1 chosen from 1 choice
scsi4 : SCSI emulation for USB Mass Storage devices
(...)
 sdc: unknown partition table
sd 4:0:0:0: [sdc] Assuming drive cache: write through
sd 4:0:0:0: [sdc] Attached SCSI removable disk
```

直前の例に於いてXplain基板は装置節点sdc、より詳述にﾙｰﾄ ﾌｧｲﾙ ｼｽﾃﾑ内の/dev/sdc装置節点として利用可能です。

今や使用者はｺﾏﾝﾄﾞ ｼｪﾙからdd命令を使用することによってTSFSｲﾒｰｼﾞ ﾌｧｲﾙを塊装置に書くことができます。

```
dd if=dataflash.img of=/dev/sdc bs=512 oflag=sync
```

この命令はｽｰﾊﾟｰ使用者ｱｸｾｽを必要とし、配布版に依存して使用者はﾙｰﾄ使用者のためのSUか、またはｽｰﾊﾟｰ使用者としてddを実行するためにsudoで上の命令を接尾にすることが必要になるかもしれないことに注意してください。

dd応用は塊装置に直接書きます。正しい塊装置を入力しない場合、、ｺﾝﾋﾟｭｰﾀ上のﾃﾞｰﾀを破壊するでしょう。

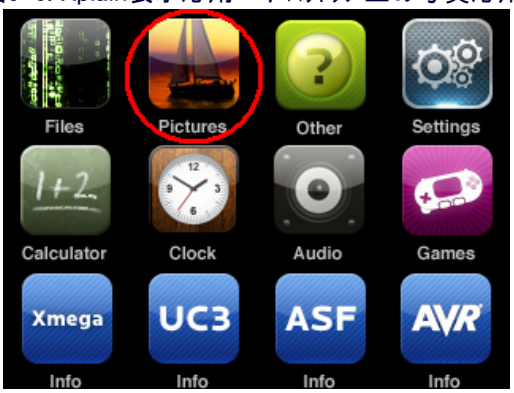
5.3. Xplain表示応用の使用

Xplain表示用の応用は本節で先に言及された段階が完了された時に始動してｱｲｺﾝが満たされたﾃﾞｽｸﾄｯﾌﾟを表示します(右図をご覧ください)。LCDの抵抗膜接触を使用することにより、使用者はｱｲｺﾝを押して各種応用を開始することができます。

図5-5. Xplain表示応用 – ﾃﾞｽｸﾄｯﾌﾟ ｱｲｺﾝ

写真応用を開始するにはｱｲｺﾝで記された画像を押してください(右図をご覧ください)。これは画像枠と殆ど同じ応用を開始します。

図5-6. Xplain表示応用 – ﾃﾞｽｸﾄｯﾌﾟ上の写真応用

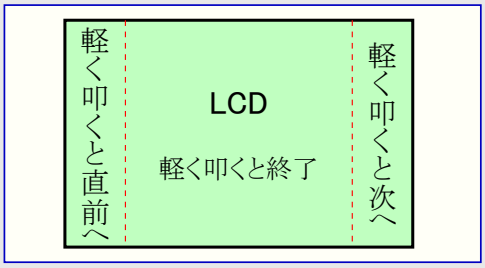


写真表示応用は外部ﾌﾗｯｼｭ ﾒﾓﾘから写真を読んで、それらをLCDﾊﾟﾈﾙに表示します。抵抗膜接触入力は画面の狭い方の端近くを軽く叩くことによって前後に飛ばすのに使用することができます。応用を抜けるには、LCDの中央を軽く叩いてください。

display-demo応用に関する注意、それはﾌｧｲﾙがﾌﾗｯｼｭ ﾒﾓﾘに複写または削除された時に直ちに通知されません。これはｲﾒｰｼﾞでﾌﾗｯｼｭ ﾒﾓﾘが満たされた後で使用者はｺﾝﾋﾟｭｰﾀのｵﾍﾟﾚｰﾃｨﾝｸﾞ ｼｽﾃﾑから大容量記憶装置を安全に排出してその後にXplain基板をﾘｾｯﾄしなければなりません。

ｲﾒｰｼﾞはLCDに合わせてﾋﾟｸｾﾙ当たり16ﾋﾞｯﾄで320×240の分解能と仮定されます。他の全ての画像も表示されますが、LCD使用に合致しないそれらは歪んで見えるでしょう。

図5-7. Xplain表示応用 – 画像枠入力領域



5.4. 資料

この応用記述用の書庫ﾌｧｲﾙは、実装、ﾄﾞﾗｲﾊﾞ、規約、ﾗｲﾌﾞﾗﾘ、ｺｰﾄﾞ ﾕｰﾃｨﾘﾃｨ、display-demo応用の徹底的詳細の技術資料を含みます。それらはbuild¥display-demo¥xplain¥GNU¥docsﾌｫﾙﾀﾞに置かれています。Doxygen形式の資料を見るには何れかのｳｪﾌﾞ ﾌﾞﾗｳｻﾞでindex.htmlﾌｧｲﾙを開いてください。

Doxygenはｿｰｽ ｺｰﾄﾞの分析と既知のｷｰﾜｰﾄﾞを用いることによってｿｰｽ ｺｰﾄﾞから資料を生成するためのﾂｰﾙです。より多くの詳細についてはwww.stack.nl/~dimitri/doxygenをご覧ください。

6. 提唱される読み物

6.1. Xplain基板制御器

複数のﾏｲｸﾛ ｺﾝﾄﾛｰﾗを持つXplain基板の使用者については、「Xplain基板制御器ﾌｧｰﾑｳｪｱの開始に際して」応用記述に目を通すことが推奨されます。これはwww.atmel.com/Xplainの基板ﾍﾟｰｼﾞから入手可能です。

この応用記述はXplain基板に対する構築、ｱｯﾌﾟﾛｰﾄﾞ、既定基板制御器応用の使用の方法を記述します。

6.2. Xplain基板 – 始める前に

Xplain基板の一般的な使い方に目を通すことが推奨されます。これはhttp://www.atmel.com/dyn/products/app_notes_mcu.asp?family_id=607から入手可能なXMEGAﾃﾞﾊﾞｲｽを目的対象とした多数の応用記述で網羅されます。

7. 追補A : UART出戻り応用への導入

uart-loopback応用はXplain表示ｿﾌﾄｳｪｱ枠組みのかなり簡単且つ基本的な使い方です。その機能はｿﾌﾄｳｪｱ出戻り形態でUARTを構成設定することで、換言すると、RX線で受信された何かがCPUによって読まれてTX線に送出されます。

ｿﾌﾄｳｪｱ枠組みにuart-loopbackのような新しい応用を追加する時に、使用者はｿｰｽ ｺｰﾄﾞと、ﾄﾞﾗｲﾊﾞ、ﾕｰﾃｨﾘﾃｨ、それとｲﾝｸﾙｰﾄﾞされた他のｿｰｽ ｺｰﾄﾞからｲﾝｸﾙｰﾄﾞするためのものを構築する方法に関連するいくつかの追加ﾌｧｲﾙとｿｰｽ ｺｰﾄﾞを追加する必要があります。本章はｿﾌﾄｳｪｱ枠組みに於ける応用の基本的な考え方を使用者に与えるために、UART出戻り応用を追加するのに必要な段階を簡単に片付けます。

この追補用のｿｰｽ ｺｰﾄﾞはこの応用記述と共に配布される圧縮書庫ﾌｧｲﾙに含まれています。

7.1. 応用ﾌｫﾙﾀﾞ

新しい応用はapps¥ﾌｫﾙﾀﾞ内の副ﾌｫﾙﾀﾞに追加されることが必要で、この例についてはapps¥uart-loopback¥ﾌｫﾙﾀﾞが選ばれました。それは短く簡潔で応用に関する全ての考えを与えます。

使用者は応用ﾌｫﾙﾀﾞ内に応用を詳述する全てのﾌｧｲﾙを置かなければなりません。これはｿｰｽ ﾌｧｲﾙ、ﾍｯﾀﾞ ﾌｧｲﾙ、形態設定ﾌｧｲﾙ、構築ｼｽﾃﾑ ﾌｧｲﾙを含みます。

通常、応用ﾌｫﾙﾀﾞは最低以下のﾌｧｲﾙとﾌｫﾙﾀﾞから成ります。

- apps¥uart-loopback¥Makefile
- apps¥uart-loopback¥config-xplain.mk
- apps¥uart-loopback¥include¥app¥version.h
- apps¥uart-loopback¥main.c
- apps¥uart-loopback¥uart-loopback.mk

各種ﾌｧｲﾙについてのより多くの詳細は以降の項です。

7.2. 応用ｿｰｽ ｺｰﾄﾞ

ｿｰｽ ｺｰﾄﾞ ﾌｧｲﾙは応用ﾌｫﾙﾀﾞに配置されるべきです。この例ではmain.cﾌｧｲﾙだけがあり、それは機能の全てを含みます。どのﾍｯﾀﾞﾌｧｲﾙも応用ﾌｫﾙﾀﾞ内のinlude¥app¥副ﾌｫﾙﾀﾞに置かれるべきです。

UART出戻り応用についてはapps¥uart-loopback¥include¥app¥ﾌｫﾙﾀﾞ内に簡単なversion.hﾍｯﾀﾞ ﾌｧｲﾙがあります。

7.2.1. 応用構築ｼｽﾃﾑ ﾌｧｲﾙ

応用ﾌｫﾙﾀﾞ内の各種ｿｰｽ ｺｰﾄﾞ ﾌｧｲﾙの物理的な追加に加えて、応用はどのﾌｧｲﾙが後置されるべきで、どのﾌｧｲﾙが応用に対して重要なﾍｯﾀﾞ ﾌｧｲﾙかを構築ｼｽﾃﾑに告げることも必要です。

何がｺﾝﾊﾟｲﾙを必要とするかについて構築ｼｽﾃﾑに通知するのはMakefileを通して行われます。これは応用ﾌｫﾙﾀﾞ内に置かれ、応用と同じ名前でなければなりません。UART出戻り応用でこのﾌｧｲﾙはapps¥uart-loopback¥uart-loopback.mkに配置されます。

例: apps¥uart-loopback¥uart-loopback.mkﾌｧｲﾙ

```
incdir-y += $(src)¥apps¥$(app)¥include
src-y += apps¥$(app)¥main.c

app-hdr-y += version.h

hdr-y += $(addprefix apps¥$(app)¥include¥app¥, $(app-hdr-y))
```

これはsrc-y変数でｿｰｽ ｺｰﾄﾞ ﾌｧｲﾙの一覧にmain.cﾌｧｲﾙがｲﾝｸﾙｰﾄﾞされるべきであることを構築ｼｽﾃﾑに指示します。

応用Makefileは構築ｼｽﾃﾑ応用特定ｲﾝｸﾙｰﾄﾞ ﾌｫﾙﾀﾞも定義し、これはincdir-y変数でｲﾝｸﾙｰﾄﾞ ﾌｫﾙﾀﾞ一覧にﾌｫﾙﾀﾞを追加することによって行われます。最後にversion.hﾍｯﾀﾞ ﾌｧｲﾙがapp-hdr-y局所変数に追加され、これはhdr-y変数での全体ﾍｯﾀﾞ一覧に完全な前置ﾊﾟｽでその次に追加されます。

構築ｼｽﾃﾑは今や応用に関して必要なｿｰｽとﾍｯﾀﾞのﾌｧｲﾙ全てについて知ります。使用者がmakeを動かす時に構築ｼｽﾃﾑはsrc-yで一覧にされたﾌｧｲﾙをｺﾝﾊﾟｲﾙしてﾘﾝｸし、ｺﾝﾊﾟｲﾗ ﾌﾗｸﾞとしてincdir-yとhdr-yを渡します。

7.3. 応用Makefile

Makefileは構築システムに関して鍵となるファイルで、構築システムに関して基本Makefileの1つの命令と少しの定義が必要なだけです。より多くの詳細については6頁の「4.7.1. 最上位応用Makefile」項をご覧ください。

例: apps¥uart-loopback¥Makefileファイル

```
src                         := ..¥..
app                         := uart-loopback
DEFAULT_CONFIG              := xplain
DEFAULT_TOOLCHAIN           := GNU

include $(src)¥make¥app.mk
```

7.4. 応用形態設定ファイル

応用形態設定ファイルは応用によって必要とされる様々な定義とインクルードに対しての責任があります。これはどの基板が応用で動き、どのドライバが許可されるかの定義と様々なドライバ特有定義、その他をインクルードします。形態設定ファイルに関するより多くの詳細については5頁の「4.4. 形態設定」項をご覧ください。

例: apps¥uart-loopback¥config-xplain.mkファイル

```
include $(src)¥board¥xplain¥config.mk

CONFIG_CPU_HZ=2000000UL

CONFIG_SERIAL_UART=y
CONFIG_UART_BAUD_RATE=9600
CONFIG_UART_ID=0
```

始めに基板形態設定ファイルが、そして直列UARTドライバ(CONFIG_SERIAL_UART)が許可されるの先立って、次にHzでのCPUクロック(CONFIG_CPU_HZ)がインクルードされます。

加えてUART出戻りの応用で使用するためにボーレート(CONFIG_BAUD_RATE)とどのUART ID(CONFIG_UART_ID)かの2つの定義を設定します。UART IDの0はチップ上のUART部署の最初のものに対応し、ATxmega128A1についてそれはUSARTC0になります。

7.4.1. 複数形態設定

makeを呼ぶ時にCONFIG環境変数を設定することによって構築時に異なる形態設定ファイルを読み込むことができます。複数形態設定ファイルに関するより多くの情報については5頁の「4.4. 形態設定」項をご覧ください。

例: Xplain基板制御器(AT90USB1287)用uart-loopback応用構築

```
make CONFIG=xplain-bc
```

例: Xplain基板制御器(ATxmega128A1)用uart-loopback応用構築

```
make CONFIG=xplain
```

7.5. 応用構築と目的対象での走行

応用構築に関し、使用者は応用フォルダへ移行してソフトウェア枠組みのために必要とされるmakeソフトウェアを使用しなければなりません。UART出戻り応用に関し、XplainK版上のATxmega128A1用にコンパイルするために使用者はコマンド行で以下の命令を走行しなければならないでしょう。

```
cd apps/uart-loopback
make CONFIG=xplain
```

これは以下のような最上位構築フォルダでのELFファイルに帰着するでしょう。

```
build/uart-loopback/xplain/GNU/uart-loopback.elf
```

使用者が目的対象デバイスにELFファイルをプログラミングする(書く)時にいくつかの環境変数が設定されなければなりません。より多くの詳細については5頁の「4.3. 応用のプログラミングと走行」項をご覧ください。

例: プログラミング ツールとしてavrdudeを使用

```
make CONFIG=xplain AVR_PROGTOOL=avrdude CONFIG_PROGRAMMER=jtagicemkii CONFIG_PROG_PORT=usb program
```

上の例はuart-loopback応用をXplain基板上のATxmega128A1チップにプログラミングし(書き)ます。応用を動かすため、使用者は基板で電源をOFF/ONしなければなりません。

本社

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
TEL 1(408) 441-0311
FAX 1(408) 487-2600

国外営業拠点

Atmel Asia
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
TEL (852) 2245-6100
FAX (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
TEL (33) 1-30-60-70-00
FAX (33) 1-30-60-71-11

Atmel Japan
104-0033 東京都中央区
新川1-24-8
東熱新川ﾋﾞﾙ 9F
ｱﾄﾒﾙ ｼﾞｬﾊﾟﾝ株式会社
TEL (81) 03-3523-3551
FAX (81) 03-3523-7581

製品窓口

ｳｪﾌﾞｻｲﾄ
www.atmel.com

技術支援
avr@atmel.com

販売窓口
www.atmel.com/contacts

文献請求
www.atmel.com/literature

本応用記述はATMELのAVR1913応用記述(doc8294.pdf Rev.8294A-03/10)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。