

1-1. USBドライバ

基板を PC に USB 接続すると、PC からは仮想 COM ポートとして認識されシリアル通信できるようになります。設定方法は公開されているマニュアル(PDF)を参照してください。

基板上の ATtiny45 は、USB と RS-232C の変換をソフトウェアで行います。Low Speed の USB で CDC プロトコルを動作させる技術 AVR-CDC を用いています。Windows では INF ファイルを指定するだけで標準ドライバが組み込まれますが、Vista では若干のバッチが必要です。ドライバを指定するときに注意してください。

1-2. ターミナルソフトウェア

通信用のターミナルソフトウェアはさまざまですが、基本的にはどれでも使えます。

ハイパーターミナル

Windows XP までは標準装備されていました。機能・操作とも簡単ですが、仮想 COM ポートに対してはデータ送出時の転送効率が低く、1 バイトごとに 1 パケットが生成されます。設定方法は基板付属のマニュアルを参照してください。

TeraTerm

高性能なフリーのソフトウェアです。プロ向けの機能をほぼすべて備えていますが設定がやや難しいようです。低速ボーレートでクリップボードからのペースト送出がタイムアウトするので、ForCy 基板へのダウンロードには文字送信間隔を設定するか、ファイル転送機能を用いてください。

HypoTerm

リカーゾン社の低機能ターミナルソフトです。

<http://www.recursion.jp/prose/hypoterm/>

起動するとターミナルウィンドウが開きます。基板接続後に、File -> New でダイアログを開き、仮想 COM ポート番号とともに、4800bps, 8databits, None Parity, 1 stopbit に設定。

いずれのターミナルソフトウェアでも、必ずセッションを閉じてから基板を取り外してください。接続中に基板を取り外すと、仮想 COM ポートに接続された無効なファイルハンドルが残留するため再接続できなくなります。

1-3. 統合型開発環境

AVR Studio は、本基板に使われるマイクロコントローラの製造元 Atmel 社が提供する統合型開発環境です。エディタ、アセンブラ、シミュレータやマイコンチップへの書き込みインターフェースを備えています。

Atmel 社のサイトから AVR Studio 4 をダウンロードし、インストールします。サイト内で“AVR Studio”を検索すれば見つかります。ダウンロードには登録が必要ですが、AVR Wiki の最新情報では登録なしの入り口も紹介されています。

<http://avrwiki.jpn.ph/wiki.cgi>

インストーラではすべてデフォルトのまま進んでよいですが、インストールディレクトリに日本語の混じったパス(“マイドキュメント”)を使わないよう気をつけてください。サービスパックがあれば、追加インストールしておきます。

1-4. C コンパイラ

WinAVR はオープンソースの GCC プロジェクトを AVR マイコンに対応した AVR-GCC の Windows 版です。以下のサイトから WinAVR をダウンロードし、インストールします。

<http://sourceforge.net/projects/winavr/>

こちらもデフォルトのまま、日本語の混じらないパスにインストールします。このコンパイラは AVR-GCC プロジェクトを作成すれば、AVR Studio から自動的に呼び出されます。

1-5. ブートローダ

ForCy 基板は出荷時の状態では、リセット後に ForCy 言語のモニターが起動します。C 言語での開発では、このモニターの代わりにユーザープログラムが起動するようにします。cboot はリセット後のシステムの割り込みベクタ領域を 0 番地基点とし、そのリセットベクタからユーザープログラムを起動するよう設定変更します。このプログラムを最初に 1 度だけ実行しておいてください。

基板を USB 接続

ターミナルソフトを起動 ForCy モニター起動

SW1, 2 を押しながリセット = 表示

cboot.hex のテキストをコピー & ペースト oooooo/

これまで ForCy 言語処理系が格納されていたメモリ領域が C 言語プログラムで使用できません(約 7 K B)。残りの 1 K B には、通信機能と HEX ファイルの読み込み機能が格納されています。

2 . プロジェクトの作成と実行

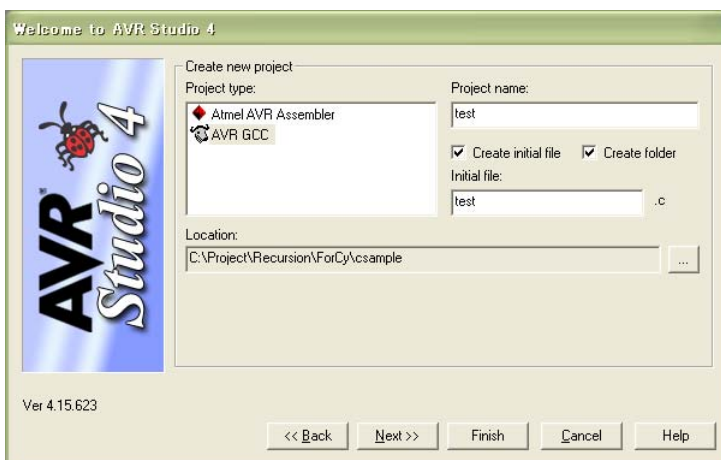
2-1. プロジェクト設定とダウンロード

AVR Studio 4 を起動し、New Project を選びます。

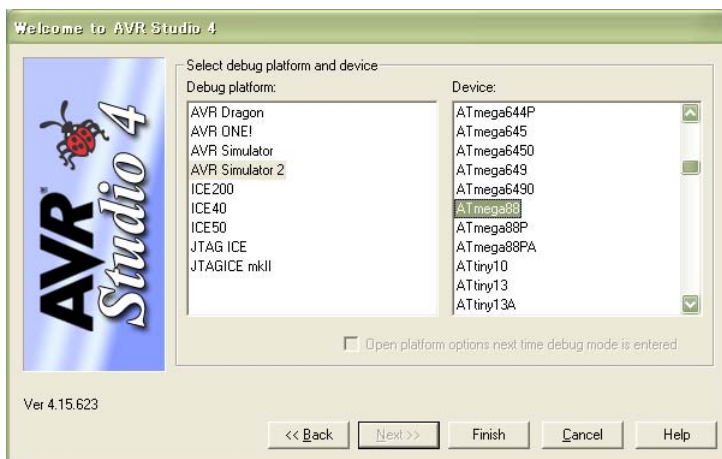


任意のディレクトリ下にプロジェクトを作成します。

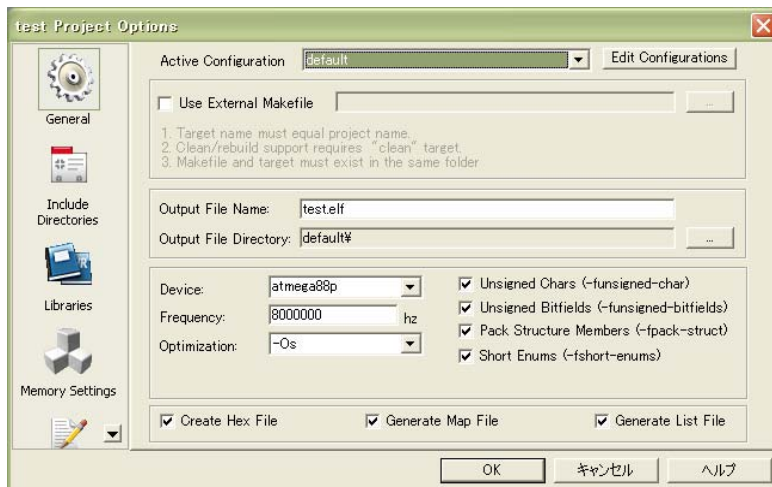
このときパスに日本語(“デスクトップ”など)を含まない場所を指定してください(日本語パス名では make 処理でファイルを見つけれないようです)。Next>> へ。



デバッグ環境はとりあえず AVR Simulator 2 とし、デバイスには基板上のマイコンの型番 ATmega88（または 88p）を選びます。Finish で完了。

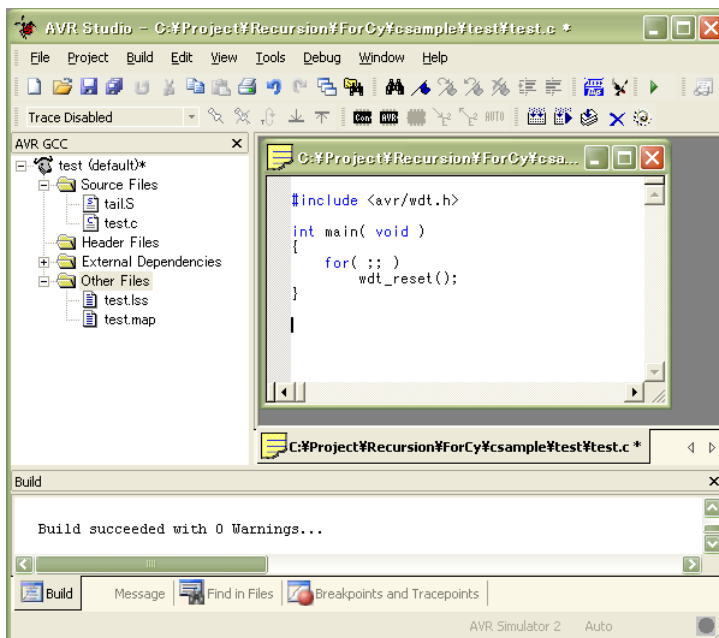


Project->Configuration Options でマイコンの型番 ATmega88(または 88p)を確認し、クロック（8000000Hz）を設定します。



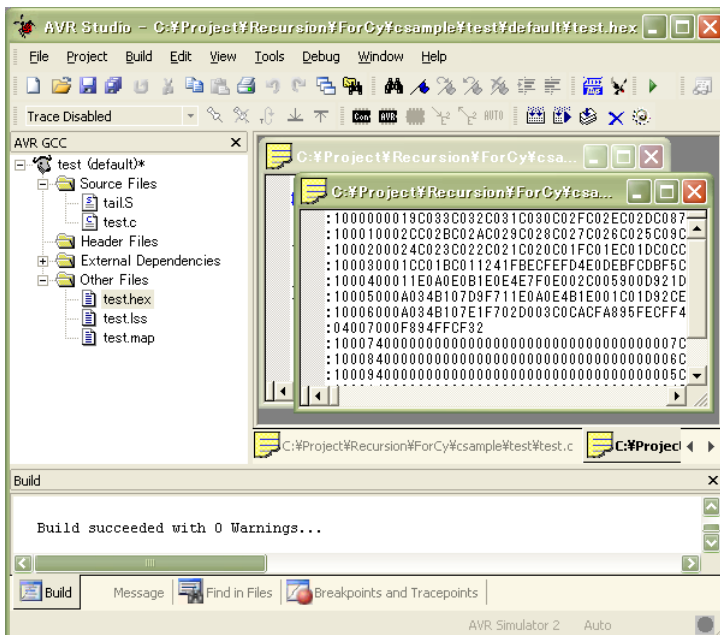
ファイルマネージャを起動し、プロジェクトで生成されたフォルダにブートローダ用パッチ tail.S をコピーします。つぎにこれを AVR Studio の Source Files に追加 (ドラッグ&ドロップ) します。

このパッチは、Ver 1.0 基板でプログラムサイズが 193-256 バイトのとき、ブートローダがフラッシュ ROM へ正しく書き込めない不具合を補うものです (Ver 2.0 以降は不要)。小さいプログラムを開発しているときにに入れてください。



*.c に簡単なプログラムを書き、メニューの Build -> Build でコンパイルします。Other Files に *.lss と *.map が生成されます。

プロジェクトのフォルダ下に default のフォルダが作られます。ファイルマネージャを開き、この中の *.hex を AVR Studio の Other Files に追加 (ドラッグ&ドロップ) して開きます。



この *.hex の内容をすべて選択 (Ctrl-A) し、クリップボードにコピー (Ctrl-C) します。この状態でターミナルソフトに移り、基板をブート起動状態 (SW1,2 を押したままリセット) してからクリップボード内容をペースト (Ctrl-V) します。プログラムが転送され、起動します。

プログラムを編集

コンパイル

HEX ファイルをリロード & コピー

基板をブート起動しターミナルへペースト

を繰り返してプログラムを完成させます。