

赤外線リモコンレシーバの製作

2003年2月15日(土)

九州プログラミング研究会

発表資料

今日お話すること

- 作った物の説明
- なぜ作ったか
- どういう部品を使っているか
- どういう構成になっているか
- プログラムは....
- 開発環境は...

何を作ったか

- 赤外線リモコンレシーバー
- 機能
 - 赤外線リモコンでAC100VをOn/Offする。
 - 学習機能があるので、既存のリモコンの適切なボタンでOn/Offできる。



なぜ作ったか

- 前回の研究会で乃村さんの発表に刺激を受けた。
 - 既存のリモコンで機器をコントロール
 - リモコンはいっぱいある。リモコンを使えると面白そうだ。
- TVのリモコンでステレオをon/offしたい
 - 私は家でTVを見ていないときは、ステレオでFMを聞いている。
 - TVを見ようとソファーに座ったときに、ステレオを消すことを思い出して立ち上がるのが面倒
 - TVのリモコンで、ステレオのOn/Offができればとても便利

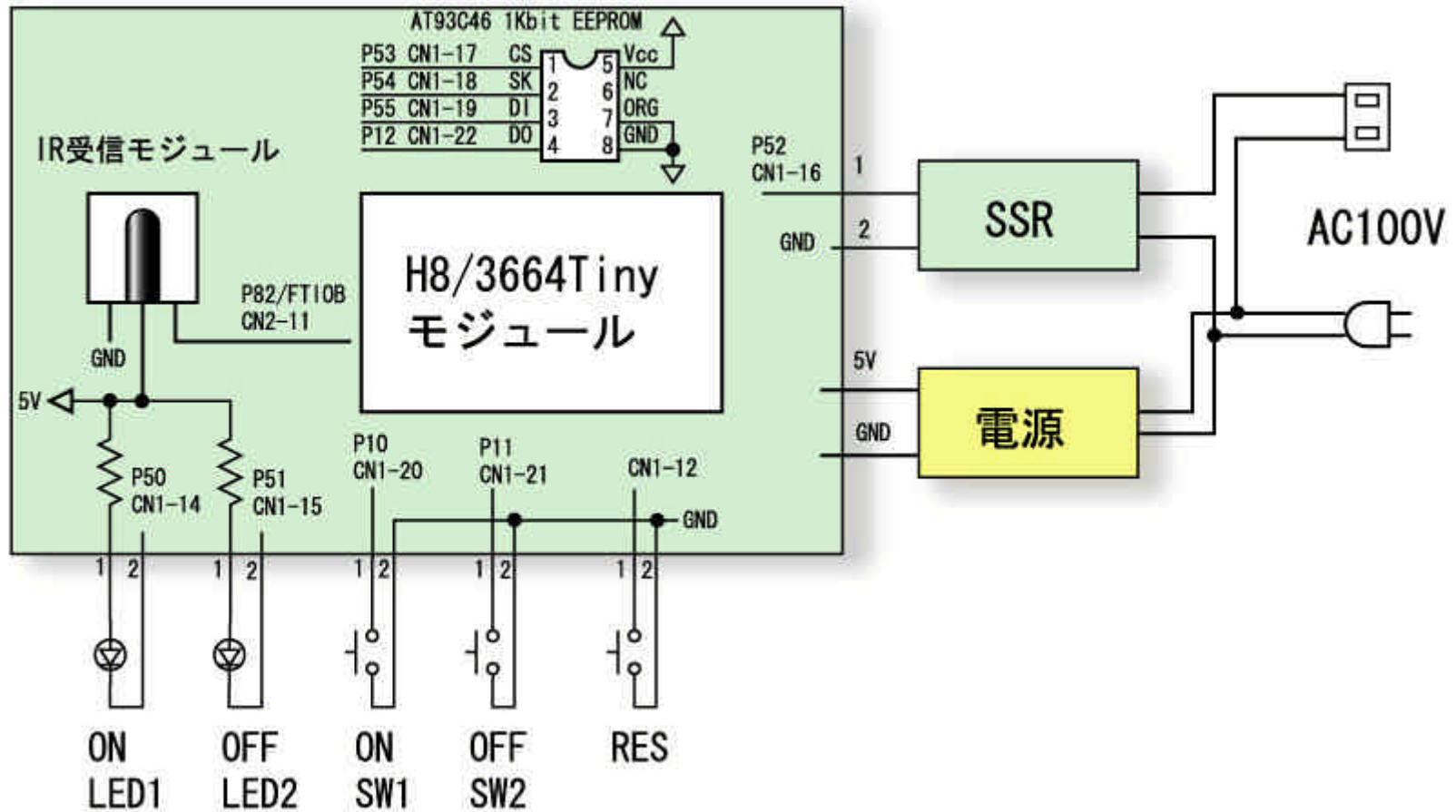
使用環境

- こういうところで使いたかった



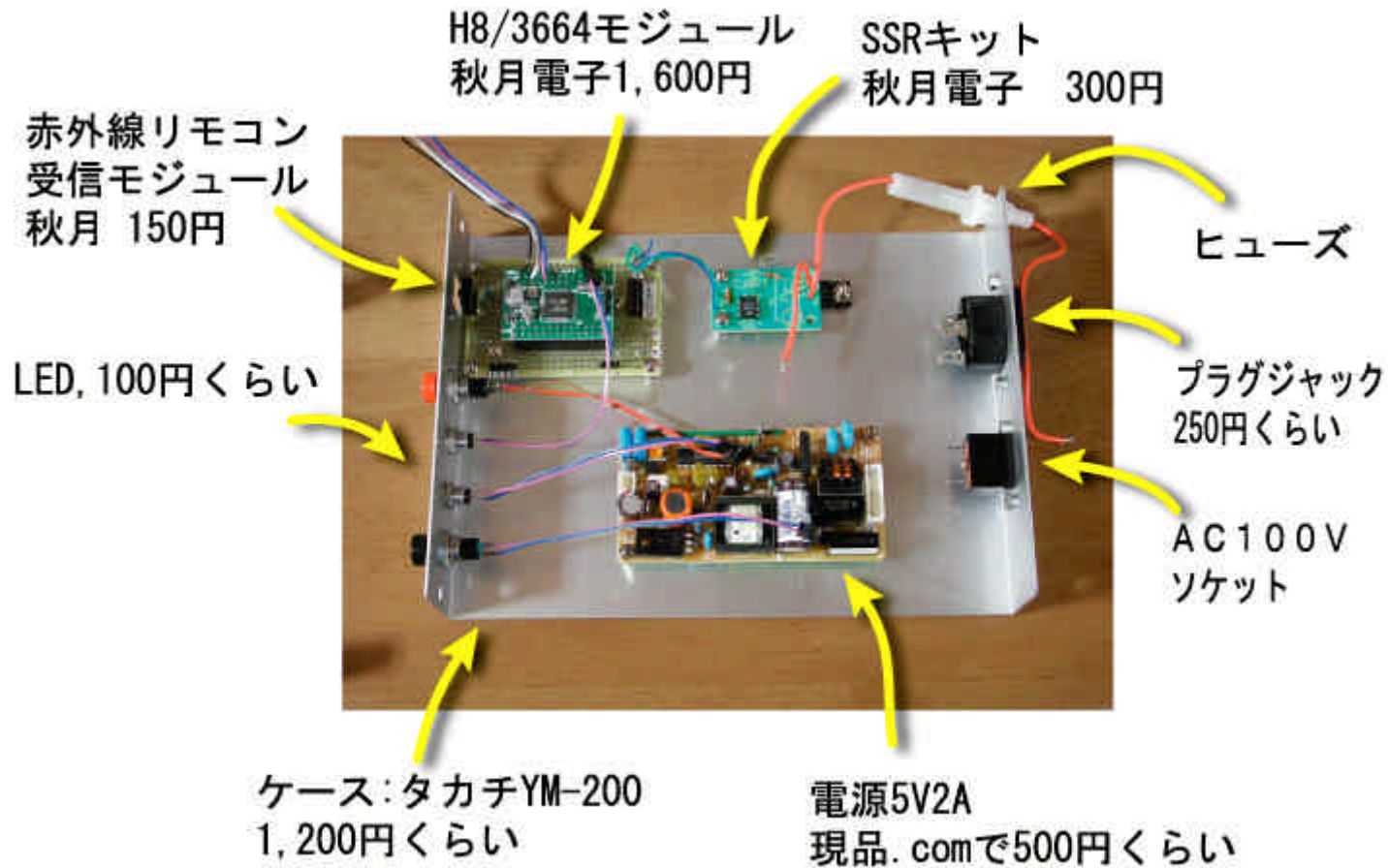
構成/回路

- 回路図



構成/部品

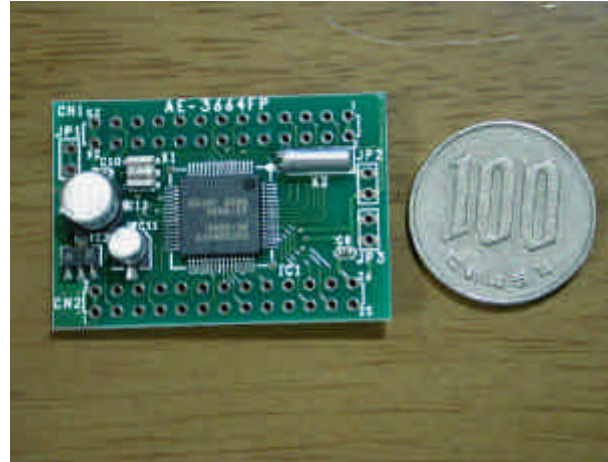
- 構成と部品の説明



H8/3664 Fとは

- 高機能安価な1チップマイコン
 - 16ビット高速H8/300H CPUを内蔵
 - 16ビット汎用レジスタを16本
 - gcc/binutilが対応している
 - ROM 32Kbyte, RS232C経由で書換可能
 - RAM 4Kbyte
 - 豊富なI/O - タイマー、RS232C F, I2C, 10bit 8入力A/D, 汎用入出力29本, 入力8本
 - 800円 ~ 650円/個ぐらい

H8/3664Fモジュール



- 秋月電子の商品

- K-158 AKI-H8/3664 QFP版超小型マイコンモジュール開発セット 4,800円
- K-159 マイコンモジュールのみ 1,600円
- K-158はK-159に開発ソフト(アセンブラ、モニタデバッガ、Cコンパイラ)の入ったCDROMを追加したもの。GNUを使用するのであれば不要

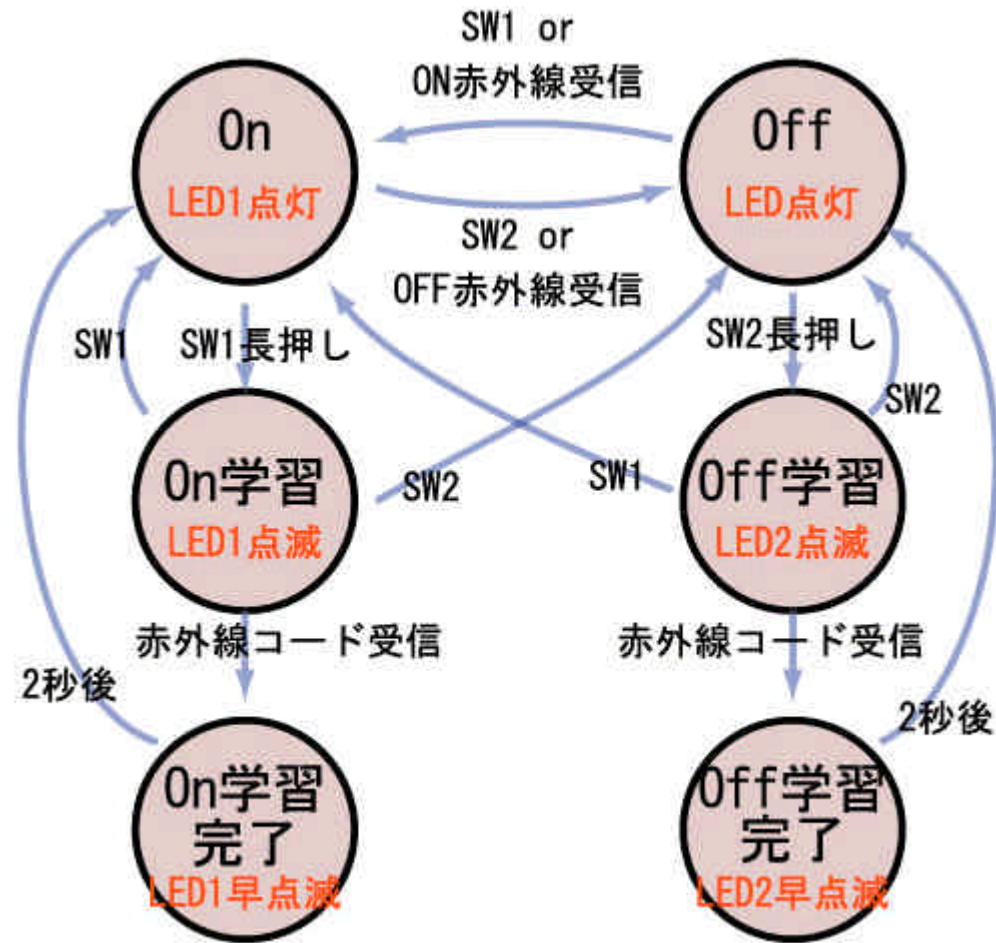
- H8/3664Fとクロック、RS232レベルコンバータ、電源レギュレータを基板に実装したもの。ピンヘッダ・コネクタも付属

赤外線リモコンのフォーマット

- ネットで検索したところ主要なものは以下の 2つ
 - 日本電気フォーマット
 - (参考URL http://stk500.hp.infoseek.co.jp/ir_tech.html)
 - 家電協フォーマット
 - (参考URL http://www5e.biglobe.ne.jp/~avm/Ir_Kaden.png)
- これら以外の独自フォーマットもあるらしいが、信号としては似たようなもの
- タイマーでパルス幅を記録、変化が無いとタイムアウト。
- タイムアウト後デコードする。
- とりあえず、手持ちのリモコンが使えるように対応した。

プログラムの動作

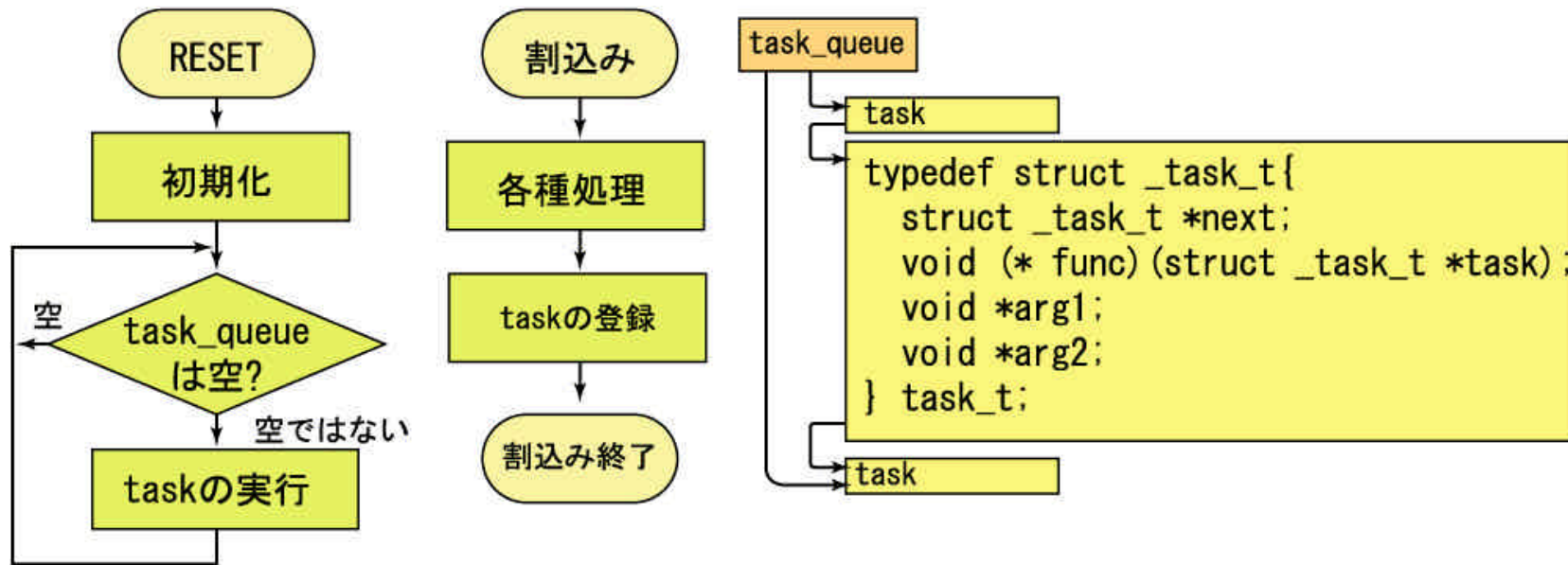
- 赤外線レシーバのプログラムは以下の状態遷移を行う



プログラム開発環境は？

- H8/3664Fのプログラム開発環境は日立のやつとか、Yellow Softの奴とかいろいろあるらしいが、私はCygwin上でbinutil+gccを使用。
 - binutil+gccを使う理由
 - GUIのプログラムは好きでない。makeを使用したい。
 - binutil/gccの使い方を憶えたほうが応用が効きそう。
 - エディタはemacs(Meadow)を使いたい
 - Cygwinを使う理由
 - 書込みソフトに日立のhterm (コマンドライン版)を使用
- 開発環境のインストール方法
 - ソースを入手し./configure & make
 - ネット上に詳しいメモがある
 - gccに h8300 normal-mode patchがまだ必要かも

プログラムの構成



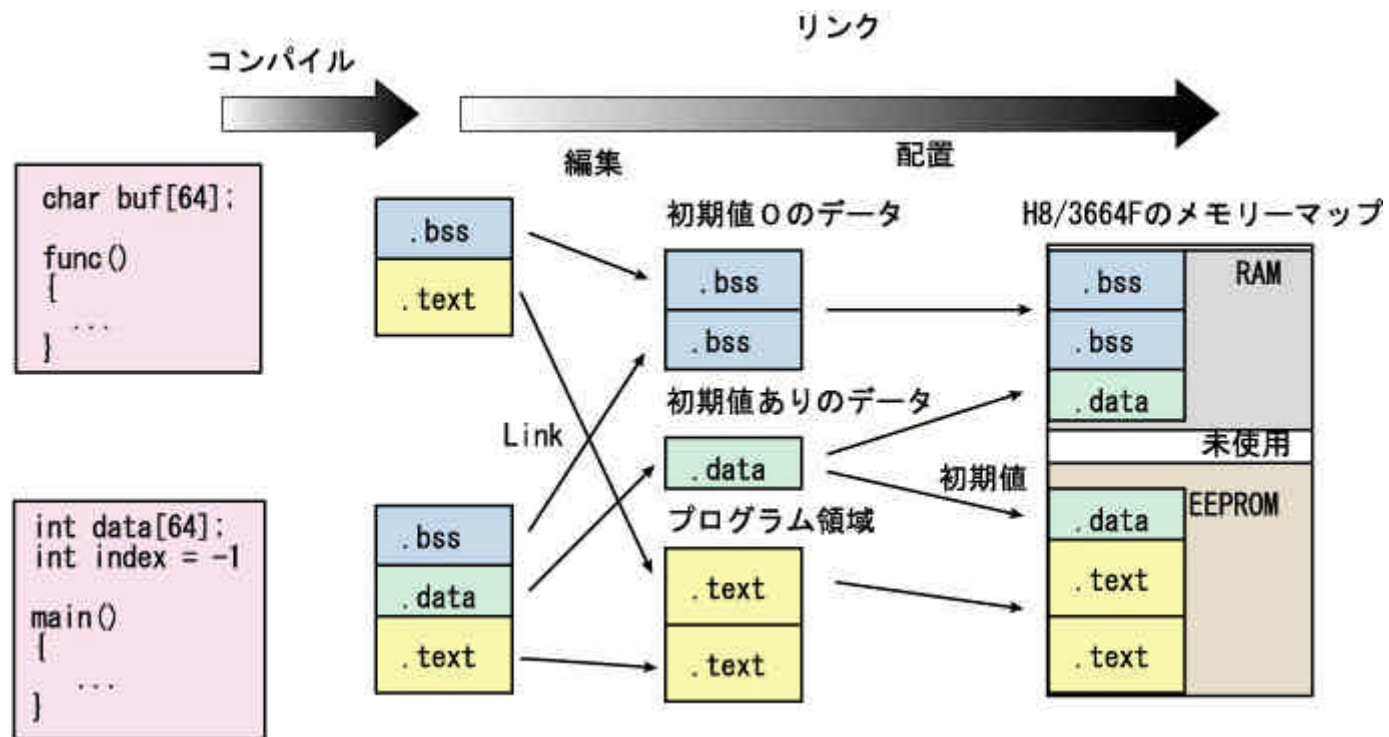
- 基本的に割り込みで処理。
- 時間がかかる処理はtaskを作成し後で実行
- 割り込み処理中は割り込み禁止
- シリアル受信は割り込みで処理される。
- シリアルへの出力は時間がかかるので taskで処理

組み込みプログラミングのために知っておかなければ いけないこと。

- 組み込み機器用のプログラムでは、通常のプログラム以外に以下のことに注意しなければいけない。
 - メモリーへのプログラムとデータの配置
 - どの番地にプログラムやデータを配置するか
 - リセット直後に動かすのか、モニター上で動かすのか
 - プログラムをROMに置くか、RAMに置くか
 - main()関数が走るためにやらなければならない各種初期化
 - ボード固有の初期化が必要な場合がある。
 - プログラムをROM上に置いた場合には
- これらのことは、環境 (ボード)ごとに異なる。
- H8/3664の環境はnewlibでサポートされていないので、自分で指定する必要がある。
- これらの指定ができれば、様々な環境でbinutil+gccを使用できるようになる。

プログラムとデータのメモリーへの配置

- リンカーの働き



リンカースクリプト

- リンカーの動作を指定するファイル
- `-T`オプションで指定

```
OUTPUT_FORMAT("coff-h8300")
OUTPUT_ARCH("h8300h")
ENTRY("_start")
MEMORY
{
    vectors : o = 0x0000 , l = 0x0034
    rom      : o = 0x0100 , l = 0x7f00
    ram      : o = 0xf780 , l = 0x0680
    stack    : o = 0xff7c , l = 0x0004
}
SECTIONS {
.vectors : {
    SHORT(ABSOLUTE(_start)) /* 0: reset, watchdog */
    ..中略..
    SHORT(DEFINED(_i2c_intr) ?ABSOLUTE(_i2c_intr) :ABSOLUTE(_start)) /* 24: I2C */
    SHORT(DEFINED(_ad_intr) ?ABSOLUTE(_ad_intr) :ABSOLUTE(_start)) /* 25: A/D */
    FILL(0xff)
} > vectors

.text : { *(.rodata) *(.text) _etext = .; } > rom
.data : AT (ADDR(.text) + SIZEOF(.text)) {
    __data = .;
    *(.strings)
    *(.data)
    *(.tiny)
    _edata = .;
} > ram

.bss : {
    _bss_start = .;
    *(.bss)
    *(COMMON)
    _end = .;
} > ram

.stack : {
    _stack = .;
    *(.stack)
} > stack
}
```

リセットベクターとmain()の間

- Cのプログラムはmain()から始まる。
- Cのプログラムが動くには準備が必要
 - スタックポインタの初期化
 - データへの初期値設定
- 通常c-runtimeというプログラムが準備を行いmain()を呼び出す。

```
; h8/300 and h8/300h start up file.
```

```
        .h8300h
        .section .text
        .global      _start
_start:
        andc.b       #0x3f, ccr
        mov.l        #_stack, sp
        mov.l        #_edata, er0
        mov.l        #_end, er1
        sub.w        r2, r2
        ;
.loop:  mov.w        r2, @er
        adds         #2, er0
        cmp.l        er1, er0
        blo          .loop

        mov.l        #__data, er0
        mov.l        #_edata, er1
        mov.l        #_etext, er2
```

```
loop2:
```

```
        mov.w        @er2, r3
        mov.w        r3, @er0
        adds         #2, er0
        adds         #2, er2
        cmp.l        er1, er0
        blo          .loop2
```

```
        mov.l        #10000, er0
        mov.l        #1, er1
```

```
.loop3:
```

```
        sub.l        er1, er0
        bgt          .loop3
```

```
        jsr          @_main
        bra          .
```

```
.section .stack
```

```
_stack: .long      1
```

割込みの使い方

- `#pragma interrupt` で割込み処理用の関数を作れます。
- `ldscript` で割込みベクトルに設定すれば使用可能

今後の課題

- 微小電源が欲しい
 - 今回の装置は常時通電する必要がある。
 - 5V 30mA、150mW程度の効率の良い小型の電源が欲しい。
 - スイッチングレギュレータ等の技術を憶えて実験してみたい。
- 次はEZ-USB(Cypress AN2131SCなど)の使い方も憶えたい。