

マイコンカーラリー用
データ解析
実習マニュアル
kit 06 版

第 1.20 版より、EEP-ROM を初期化する関数「initI2CEeprom」の仕様が変わりました。

initI2CEeprom 宣言時に、EEP-ROM があるポートの指定や入出力設定を行います。詳しくは、マニュアル内の説明をご覧ください。

「C:\WorkSpace\common」フォルダなどにある i2c_eeprom.c の日付が 2006/09/21 以前のプログラムはバグがありました。お詫びして訂正致します。2006/09/22 以降の i2c_eeprom.c をご使用ください。

最新のプログラムは、マイコンカーラリーホームページ(<http://www.mcr.gr.jp/>)のダウンロードコーナーの、「ルネサス統合開発環境用マイコンカー関連プログラム」のダウンロード、インストールをお願い致します。

第 1.10 版より、ヘッダファイルや共通の C ソースファイルは「c:\workspace\common」フォルダに置き、そのファイルを参照するよう変更しました。

第 1.20 版
2007.03.20
ジャパンマイコンカーラリー実行委員会

注 意 事 項 (rev.1.1)

著作権

- ・本マニュアルに関する著作権はジャパンマイコンカーラリー実行委員会に帰属します。
- ・本マニュアルは著作権法および、国際著作権条約により保護されています。

禁止事項

ユーザーは以下の内容を行うことはできません。

- ・第三者に対して、本マニュアルを販売、販売を目的とした宣伝、使用、営業、複製などを行うこと
- ・第三者に対して、本マニュアルの使用権を譲渡または再承諾すること
- ・本マニュアルの一部または全部を改変、除去すること
- ・本マニュアルを無許可で翻訳すること
- ・本マニュアルの内容を使用しての、人命や人体に危害を及ぼす恐れのある用途での使用

転載、複製

本マニュアルの転載、複製については、文章によるジャパンマイコンカーラリー実行委員会のこと前の承諾が必要です。

責任の制限

本マニュアルに記載した情報は、正確を期すため、慎重に制作したものです。万一本マニュアルの記述誤りに起因する損害が生じた場合でも、ジャパンマイコンカーラリー実行委員会はその責任を負いません。

その他

本マニュアルに記載の情報は本マニュアル発行時点のものであり、ジャパンマイコンカーラリー実行委員会は、予告なしに、本マニュアルに記載した情報または仕様を変更することがあります。製作に当たりましては、こと前にマイコンカー公式ホームページ(<http://www.mcr.gr.jp/>)などを通じて公開される情報に常にご注意ください。

連絡先

ルネサステクノロジ マイコンカーラリー事務局
〒162-0824 東京都新宿区揚場町 2-1 軽子坂MNビル
TEL (03)-3266-8510
E-mail:official@mcr.gr.jp

目 次

1. EEP-ROM(24C256)を使ったデータ解析	1
1.1 概要	1
1.2 EEP-ROM とは？	2
1.3 EEP-ROM を使う意義	2
1.4 EEP-ROM への読み書きする仕組み	3
1.5 EEP-ROM の回路	5
1.6 EEP-ROM 基板の自作	6
1.7 市販の EEP-ROM 基板を使う	8
1.8 EEP-ROM 基板の回路図	9
1.9 「i2c_eeprom.c」で使える関数	10
1.10 「i2c_eeprom.c」を登録する方法	13
1.11 EEP-ROM の接続端子を変える場合の設定	14
2. サンプルプログラム	15
2.1 ルネサス統合開発環境	15
2.2 サンプルプログラムのインストール	15
2.3 ワークスペース「kit06rec」を開く	16
2.4 プロジェクト	17
3. プロジェクト「record_01」 内蔵 RAM にデータ保存	18
3.1 概要	18
3.2 接続	18
3.3 プロジェクトの構成	19
3.4 プログラム	19
3.5 データをパソコンへ送る	21
3.6 プログラムの解説	23
3.6.1 内蔵周辺機能の初期設定	23
3.6.2 データ取得関係の定義、変数	24
3.6.3 パソコンとの通信するための初期設定	25
3.6.4 パターン 0:1 秒待ち	25
3.6.5 パターン 1: データ保存	25
3.6.6 パターン 2: タイトル転送、準備	26
3.6.7 パターン 3: データの転送	26
3.6.8 パターン 4: 転送終了	27
3.6.9 割り込み処理	28
3.7 データの取り込み方	29
3.8 エクセルへの取り込み方	31
3.8 エクセルへの取り込み方	32
4. プロジェクト「record_02」 外付け EEP-ROM にデータ保存	45
4.1 概要	45
4.2 接続	45
4.3 プロジェクトの構成	46
4.4 プログラム	46
4.5 プログラムの解説	49
4.5.1 内蔵周辺機能の初期化	49

4.5.2 データ保存関連の変数	49
4.5.3 初期設定	50
4.5.4 メイン部の全体	50
4.5.5 パターン 0: データ保存の前準備	52
4.5.6 パターン 1: データ保存中の処理	52
4.5.7 パターン 2: タイトル転送、準備	53
4.5.8 パターン 3: データ転送	53
4.5.9 パターン 4: 転送終了	54
4.5.10 割り込みプログラム	54
4.5.11 int 型の値を保存する場合	55
5. プロジェクト「record_03」 外付け EEP-ROM にデータ保存(2 進数変換)	56
5.1 概要	56
5.2 接続	56
5.3 プロジェクトの構成	57
5.4 プログラム	57
5.5 プログラムの解説	58
5.5.1 変数の追加	58
5.5.2 2 進数変換を行う convertHexToBin 関数	59
5.5.3 printf 出力	59
5.6 データ例	60
6. プロジェクト「kit06rec_01」 走行データを内蔵 RAM にデータ保存	61
6.1 概要	61
6.2 マイコンカーの構成	61
6.3 プロジェクトの構成	62
6.4 プログラム	62
6.5 プログラムの概要	65
6.6 プログラムの解説	66
6.6.1 データ保存エリア	66
6.6.2 送信内容	66
6.7 プログラムの調整	67
6.8 走行からデータ転送までの流れ	68
6.9 エクセルへの取り込み方	71
7. プロジェクト「kit06rec_02」 外付け EEP-ROM にデータ保存	74
7.1 概要	74
7.2 マイコンカーの構成	74
7.3 プロジェクトの構成	75
7.4 プログラム	75
7.5 プログラムの概要	79
7.6 プログラムの解説	79
7.6.1 データの保存	79
7.6.2 送信内容	80
7.7 プログラムの調整	81
7.8 走行からデータ転送までの流れ	81
7.9 エクセルへの取り込み方	83
8. プロジェクト「kit06rec_03」 エンコーダプログラムの追加	86
8.1 概要	86

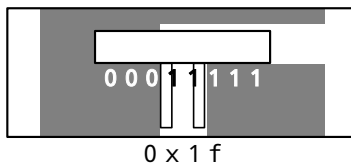
8.2	マイコンカーの構成	86
8.3	プロジェクトの構成	87
8.4	プログラムの解説	87
8.4.1	入出力設定	87
8.4.2	割り込みプログラム	88
8.4.3	送信内容	89
8.5	走行からデータ転送までの流れ	90
8.6	走行データのグラフ化	91
9.	データをエクセルで解析する	93
10.	大容量 EEP-ROM(24C1024)を使う	96
10.1	概要	96
10.2	回路図	96
10.3	プロジェクトへの登録方法	97
10.4	関数の変更点	98
11.	参考文献	99

1. EEP-ROM(24C256)を使ったデータ解析

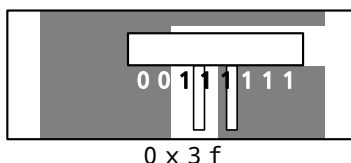
1.1 概要

マイコンカーを走らせると、脱輪することがあります。なぜ脱輪するのか... もちろん、回路の間違いやプログラムの文法的な間違いは、直さなければいけません。しかし、それらがきちんとできていても脱輪することがあります。これは、コースの検出状態や、スピード(エンコーダの値)など、想定とは違う状態になるからです。

例えば、kit06 では、右クランクと判断するセンサ状態は「0x1f」の状態です。



しかし、たまに右クランクをそのまま通過してしまい、脱輪することがありました。そのため、これから紹介する方法で脱輪したときのセンサの状態を 10ms ごとに記録、パソコンで解析してみました。すると、下図のように「0x1f」ではない状態で右クランクを検出していることが判明しました。



プログラムは、「0x3f は右クランクなので右大曲げしなさい」という内容が入っていません。そのため、そのまま進んでしまうのです。脱輪しますが、マイコンカーとしてはプログラムどおりに動いているだけです。プログラムに、「0x3f」になったならどうしないといけないか記述しておかなければいけません。

最近のマイコンカーは速度が速くなり、センサの状態を目で見て確認することは難しくなってきました。「カン」に頼っても、分からないものは分かりません。データを記録することにより、「カン」に頼らない論理的な解析ができ、プログラムに反映させることができます。

ただし、プログラムに反映させるためには、**自分が想定しているマイコンカー(センサ)の状態とプログラムを理解していなければいけません。**

- ・自分が想定しているセンサの値に対して、プログラムはこうなっている
- ・だから脱輪してしまう
- ・そのためには、このプログラムを直さなければいけない

というように、データ解析を有効活用するためには、制御プログラムの理解が不可欠です。データ解析はあくまで、プログラムをデバッグするための補助ツールなのです。

- 本マニュアルでは、
- ・データの記録方法
 - ・パソコンへの転送方法
 - ・解析方法

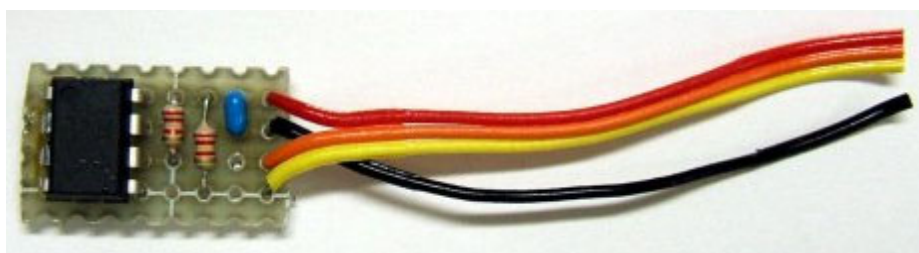
を紹介していきます。最後に、実例を紹介します。

1.2 EEP-ROM とは？

EEP-ROMとは、イーイーピーロムと読み、Electrically Erasable and Programmable Read Only Memory の略です。

電氣的に内容を書き換えることができるROMで、書き換えコマンドを送って内容を書き換えます。ROMなので、電源を切っても内容は消えません。

本書では、I2C(アイ・スクエア・シィ)バスインタフェース方式のEEP-ROMである「24C256」という型式のIC(ディップの8ピン)を使用します。



自作したEEP-ROM基板

1.3 EEP-ROM を使う意義

H8/3048F-ONE には内蔵 RAM が 4KB あります。そのうち 1.5KB 程度はプログラムで使用しますが、残りの 2.5KB 程度は空いています。このメモリをデータ記録に使用すれば、わざわざEEP-ROMを買って基板を作る必要はありません。なぜ、そこまでしてEEP-ROMを使う必要があるのでしょうか。下の表に長所、短所をまとめてみました。

記憶メモリ	マイコン内蔵 RAM	外部EEP-ROM(24C256)
記憶容量	2.5KB 程度	EEP-ROM 1個 32KB 24C256 を 4 つ接続すれば 4 倍(128KB)まで対応可能
長所	H8/3048F-ONE 内蔵のメモリを使用するため、 手軽に利用できる	8 ピンのディップ IC で基板作成が容易にできる 4 つまで増設可能(32KB × 4)で保存容量を増やすことができる 電源が消えてもデータが消えない！
短所	容量が少ない 電源を切ると消えてしまう	1 回データ書き込み後、最大 10ms 間は EEP-ROM へアクセスできない(1回に 1~64 バイトのデータを書き込み可能)

EEP-ROM(AT24C256)を使う意義は、

- ・記憶容量が 32KB もある
- ・電源が消えてもデータが消えない

というのが最大の理由です。短所は、EEP-ROMへデータ書き込み後、最大 10ms 間アクセスできません。そのため、最短でも 10ms ごとの書き込みしかできません。ただし、マイコンカーでのデータ記録には十分です。

参考までに記録時間を計算してみます。10ms ごとに 8 個のデータを保存することとします。

内蔵 RAM が保存できる容量 $2.5\text{KB} \div 1 \text{ 回の保存数 } 8 \text{ 個} \times \text{保存間隔 } 10\text{ms} = 3.2 \text{ 秒}$

EEP-ROM が保存できる容量 $32\text{KB} \div 1 \text{ 回の保存数 } 8 \text{ 個} \times \text{保存間隔 } 10\text{ms} = 41.0 \text{ 秒}$

内蔵 RAM の場合は、たったの 3 秒しか記録できません。EEP-ROM は 41 秒も保存できますので、地区大会レベルのコースなら 1 周分は記憶することができます。

このような理由から、EEP-ROM 基板を作り、データを保存します。

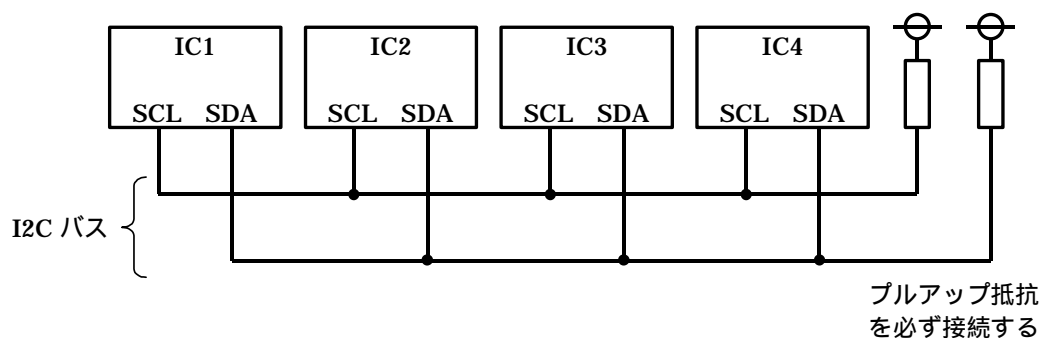
1.4 EEP-ROM への読み書きする仕組み

(1) I2C バスインタフェース方式

EEP-ROM は、AT24C256 を使用します。

AT24C256 は 2 本の線を CPU と接続します。2 本の線でシリアル通信を行って、データの書き込みや読み込みを行います。

使用するシリアル通信の方法は「I2C(アイ・スクエア・シィ)バスインタフェース方式」という通信方式です。この方式は、フィリップス社が考案した方式です。通信は、SDA(serial data)とSCL(serial clock)と呼ばれる 2 本の信号線を使用して行います。IC 間をこの信号線で数珠つなぎに接続していきます(下図)。詳しい内容は書籍やインターネットに掲載されていますので省略しますが、概要だけここでは紹介します。

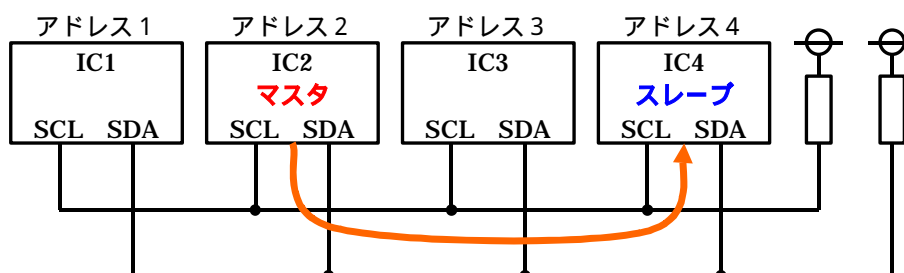


I2C とは「Inter Integrated Circuit」の略称です。「Integrated Circuit」は「集積回路(単に IC でも良いと思います)」です。「inter～」は「～の間」という意味になります。通信ということを考えると、「IC 間通信」というような意味合いになります。主に同一基板内などの近距離に配置された IC 間での高速通信(100Kbps/400Kbps/ 3.4Mbps)を行うための方式です。IIC と表記される場合もあります。「I2C」読み方は、「アイ・スクエア・シィ」と読みます。そのまま「アイ・ツー・シィ」と読みがちですが、この読み方は間違いです。

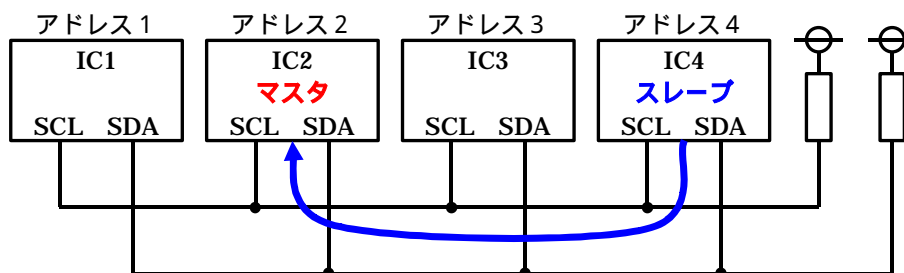
(2) マスタ・スレーブ方式

I2C は、マスタ・スレーブ方式という方法で通信を行います。これは、マスタ(主)がスレーブ(従)へ命令を送り、スレーブ(従)からマスタ(主)へ命令された内容を返し、データのやり取りを行います。

バスは 1 本ですが、IC は複数有ります。まず命令を出したい IC がマスタとなり、対象となる IC のアドレスを電文に含めて電文を送ります。今回はアドレス 4 とします(下図)。

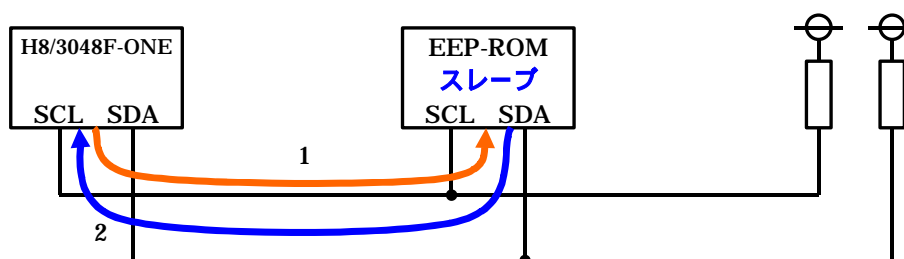


アドレス 4 の IC が通信できる状態なら、命令に対する電文を返信します(下図)。



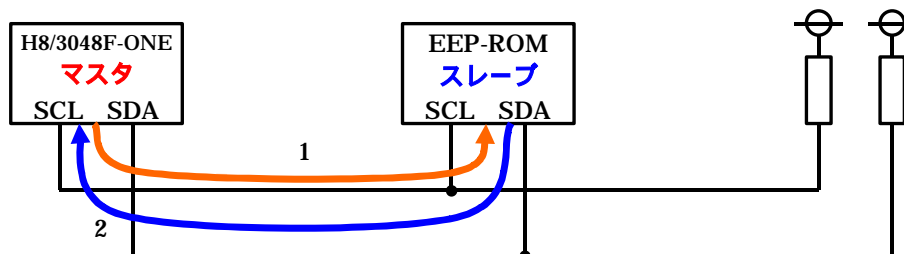
このように、マスタになったICがバスを占有します。スレーブは、マスタの質問に答える形でメッセージを返します。勝手に、メッセージを送ってはいけません。また、マスタでもスレーブでもないIC1やIC3は、メッセージが来ても無視していなければいけません。もしマスタになりたいなら、現在の通信が終わるのを待ってから、やり取りしたいICへ電文を送信します。

(3) AT24C256 へデータを書き込むとき



1. H8(マスタ)がEEP-ROM(スレーブ)へ書き込む番地とデータを送ります。
2. EEPROM が命令を受け取ると、H8へ命令を受け取った旨を返信し書き込み作業を行います。

(4) AT24C256 へデータを読み込むとき

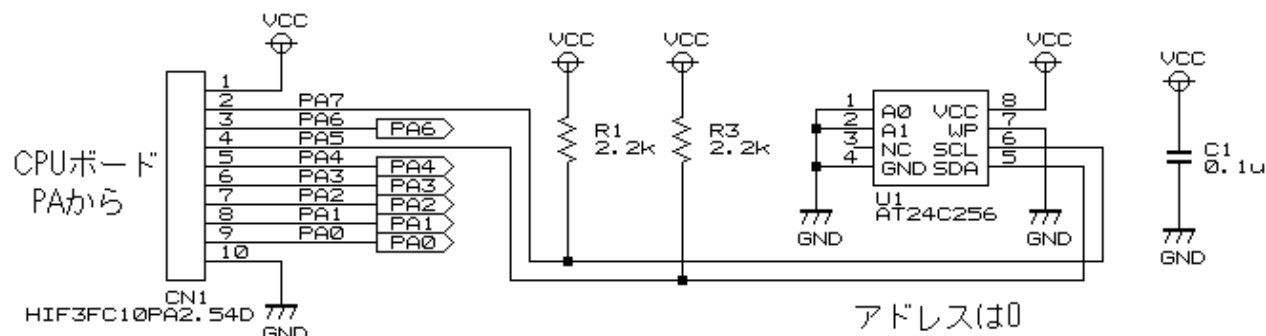


1. H8(マスタ)がEEP-ROM(スレーブ)へ読み込む番地を送ります。
2. EEPROM が命令を受け取ると、H8へ指示された番地のデータを返信します。

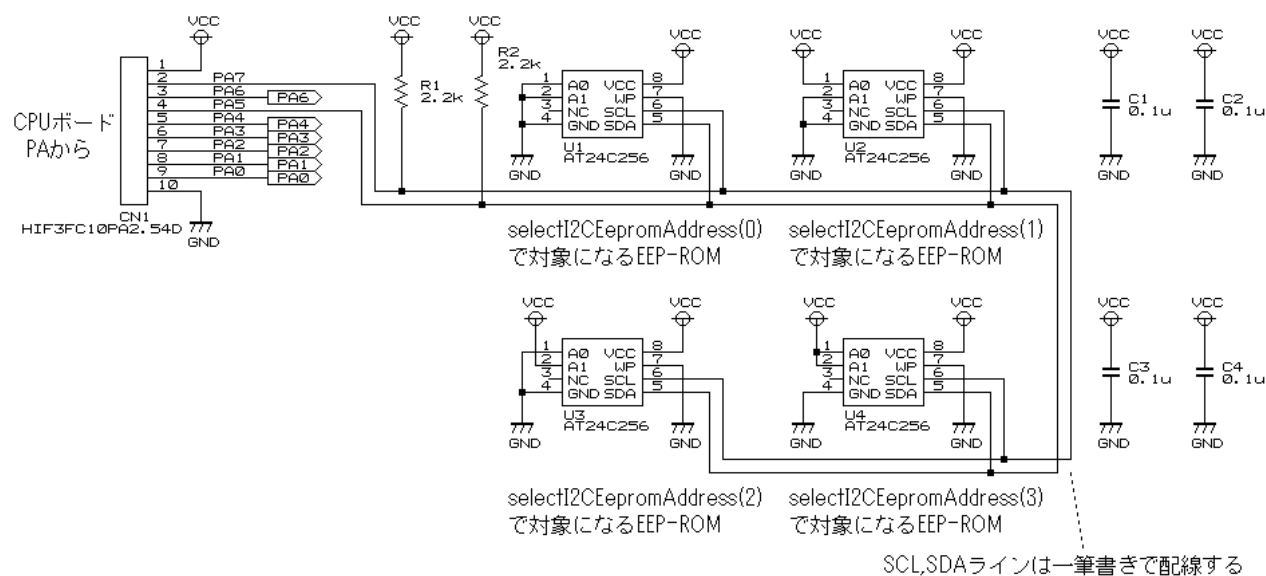
1.5 EEP-ROM の回路

EEP-ROM は、ポート A へ接続します。

番号	部品番号	部品名	型式・仕様	メーカ	数量	備考
1	U1	EEP-ROM	AT24C256	ATMEL	1	説明では、赤、 橙、黄、黒色の線 を 5cm 程使用しま す
2	R1,2	抵抗	2.2k		2	
3	C1	積層セラミックコンデンサ	0.1 μ F 程度		1	
4		ユニバーサル基板	少々		1	
5		線	少々		1	



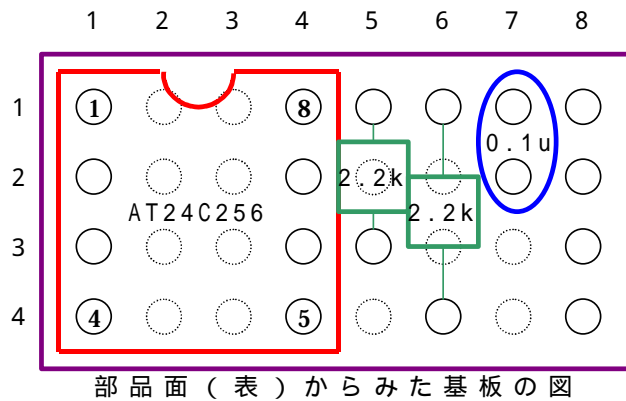
AT24C256 1個で 32KB のメモリ容量があります。足りない場合は下記のように A1,A0 の接続を変えて、増設することができます。下図に 4 個の AT24C256 を接続した回路例を示します。



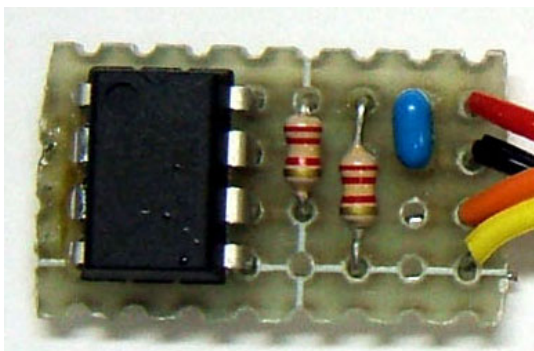
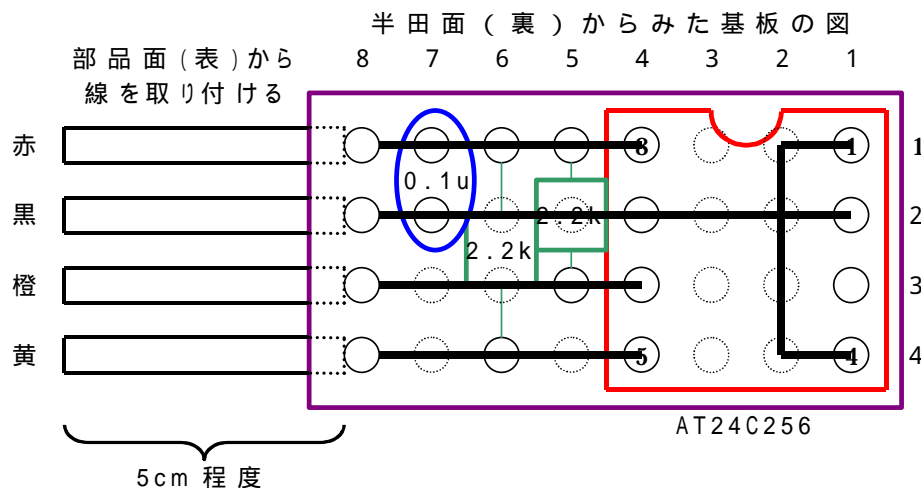
どの EEP-ROM に保存するかは、selectI2CEepromAddress 関数で選択します。

1.6 EEP-ROM 基板の自作

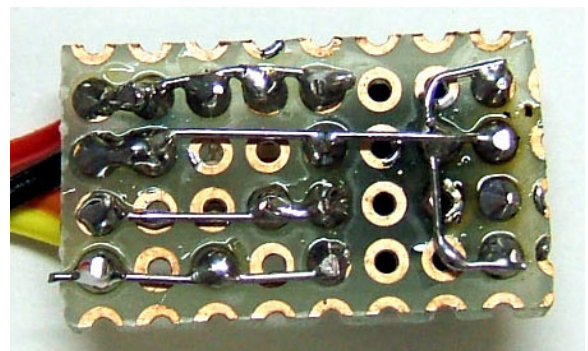
EEP-ROM 基板を製作してみましょう。まず、部品配置を考ます。ユニバーサル基板(穴あき基板)を縦 4×横 8 の大きさにカットします。下記のように 4 つの部品を取り付けます。



裏返(半田面)にして、下図のように配線します。線は最後に 5cm 程度、部品面から取り付けます。



上から見た基板

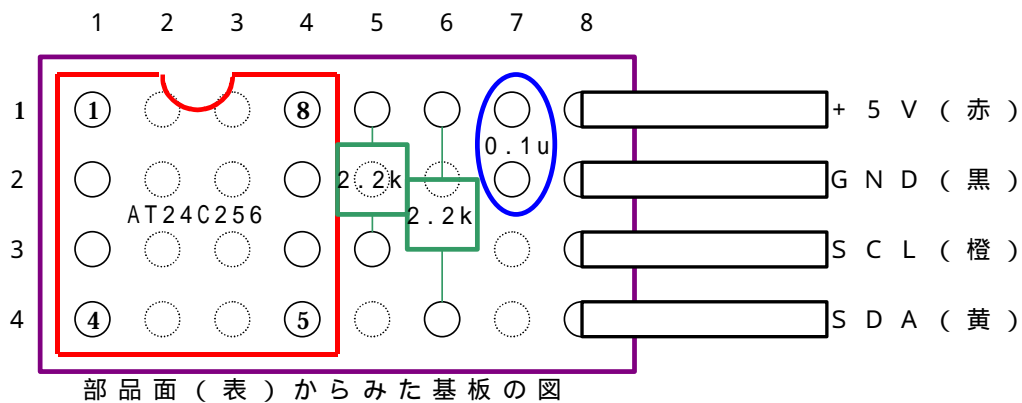


裏から見た基板



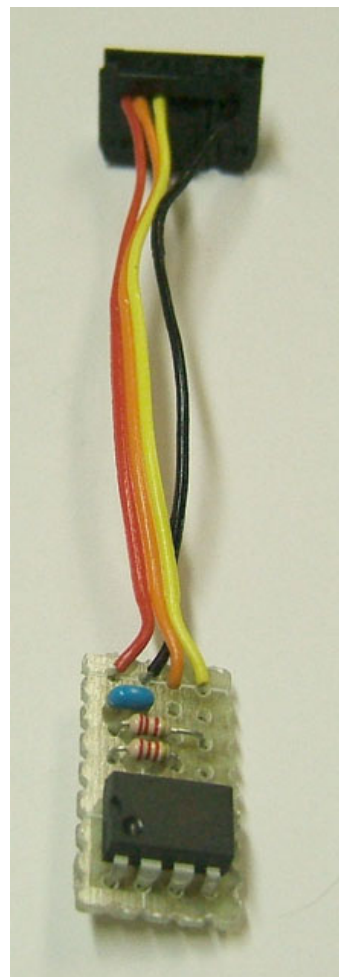
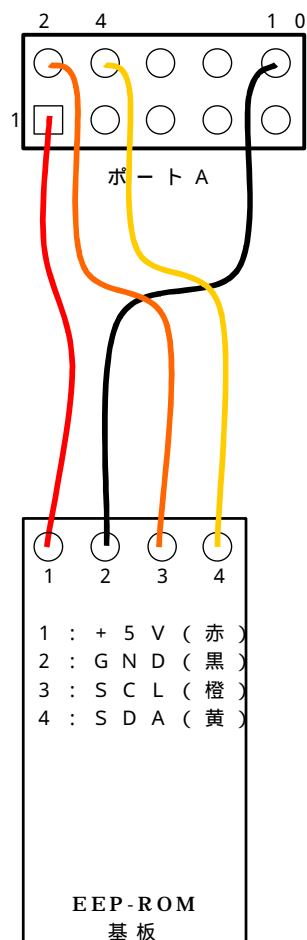
線を取り付けて完成

線の信号名は、下図のようになります。



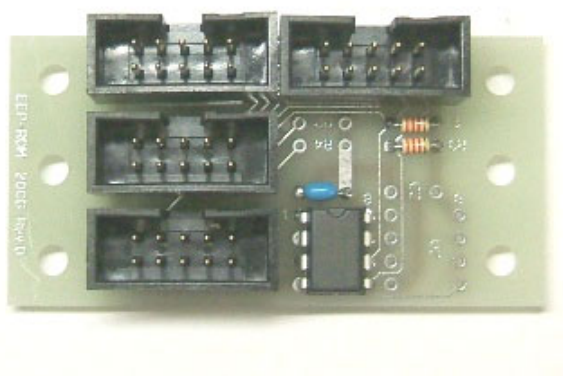
信号名	線の色	ポートAコネクタの ピン番号	接続先
+ 5 V	赤	1	+5V
GND	黒	10	GND
SCL	橙	2	PA7
SDA	黄	4	PA5

EEP-ROM 基板をポート A へ接続します。10 ピンのメスコネクタを下図のような結線で圧着します。CPU ボードに取り付けます。ただし、ポート A は、スタートバー検出センサ基板やロータリエンコーダで使用するポートです。うまく兼用できるようにケーブルを分配すると良いでしょう。

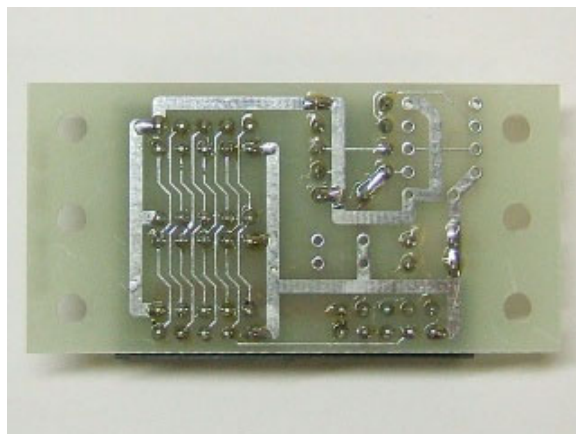


1.7 市販の EEP-ROM 基板を使う

EEP-ROM 基板キットが販売業者から販売されています。EEP-ROM に関わる部分は、自作 EEP-ROM 基板と同様です。



EEP-ROM 基板の部品面

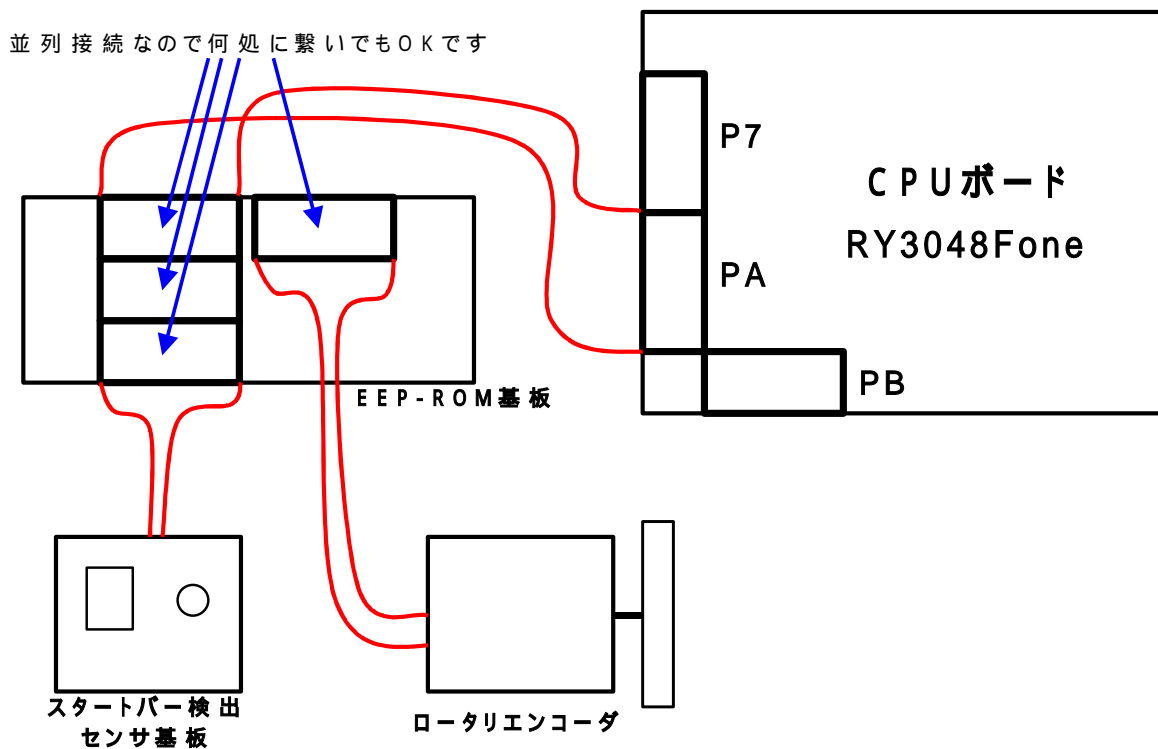


EEP-ROM 基板の半田面

EEP-ROM 回路部分の他に、10 ピンコネクタが並列に 4 個繋がっています。これは、

- ・EEP-ROM 基板 (PA7, PA5 と接続)
- ・スタートバー検出センサ基板 (PA3 と接続)
- ・ロータリエンコーダ (PA0 と接続)

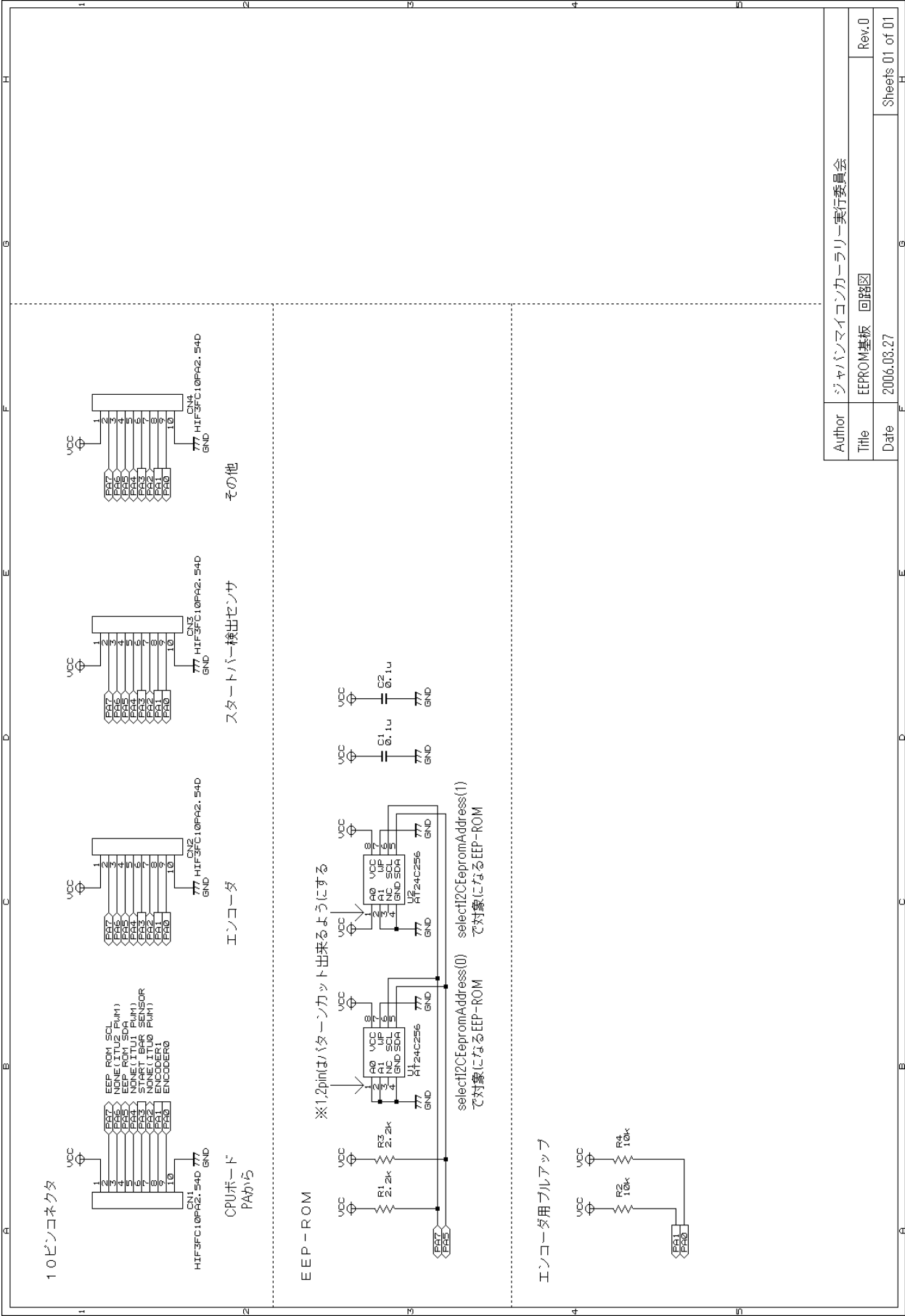
を 10 ピンコネクタで同時に接続するためです。下図に接続例を示します。



市販の EEP-ROM 基板の製作方法については、マイコンカーサイトの「EEP-ROM 基板製作マニュアル」を参照してください。

1.8 EEP-ROM 基板の回路図

EEP-ROM 基板の回路図は下図のようになっています。



1.9 「i2c_eeprom.c」で利用できる関数

EEP-ROM にデータを読み書きする、専用の関数が用意されています。ファイル名は、「i2c_eeprom.c」です。EEP-ROM を使用するとき、プロジェクトに「i2c_eeprom.c」を追加して使用します。

「i2c_eeprom.c」は、「C:\¥WorkSpace¥common」フォルダにあります。

このファイルを追加して実行できる関数は下記のとおりです。

関数名	内容
initI2CEeprom 2007.02 より使い方が変更になりました。	<pre>void initI2CEeprom(unsigned char* ddrport, unsigned char* drport, unsigned char ddrdata, unsigned char scl, unsigned char sda);</pre> EEPROM へ読み書きする準備をします。最初に必ず実行します。 引 数:EEP-ROM の繋がっている DDR ポートの指定(&を付ける) EEP-ROM の繋がっている DR ポートの指定(&を付ける) DDR ポートの入出力設定値 EEP-ROM の SCL 端子の繋がっているビット番号 EEP-ROM の SDA 端子の繋がっているビット番号 戻り値:なし 例) initI2CEeprom(&PADDR, &PADR, 0x5f, 7, 5); EEP-ROM はポート A に接続、1 つ目の引数は PADDR を指定します。 EEP-ROM はポート A に接続、2 つ目の引数は PADR を指定します。 ポート A の入出力設定値は 0x5f です。 SCL 端子は、bit7 に接続します。 SDA 端子は、bit5 に接続します。
selectI2CEepromAddress	<pre>void selectI2CEepromAddress(unsigned char address);</pre> I2C バスに接続されているどのアドレスの EEPROM を使用するか選択します。 initI2CEeprom 関数実行時は、アドレス 0 が選択されています。 引 数:EEP-ROM のアドレス 戻り値:なし 例)selectI2CEepromAddress(1); 1 番を使用 EEP-ROM A1="0" A0="1"に接続されている EEPROM を選択します。
readI2CEeprom	<pre>char readI2CEeprom(unsigned int address);</pre> EEPROM からデータを読み込みます。 引 数:unsigned int アドレス 0 ~ 32767(0x7fff) 戻り値:char データ 例) i = readI2CEeprom(0x0005); EEP-ROM の 0x0005 番地のデータを変数 i に代入します。
writel2CEeprom	<pre>void writel2CEeprom(unsigned int address, char write);</pre> EEPROM へデータを書き込みます。 書き込み後、最大 10ms は書き込み作業中のため、アクセスできません。 引 数:unsigned int アドレス 0 ~ 32767(0x7fff) ,char データ 戻り値:なし 例)writel2CEeprom(0x2000, -100); EEP-ROM の 0x2000 番地に-100 を書き込みます

<p>pageWrite I2CEeprom</p>	<pre>void pageWriteI2CEeprom(unsigned int address, int count, char* data);</pre> <p>EEP-ROM へ複数バイトのデータを書き込みます。この関数は書き込み作業が終わるまで終了しません(書き込み時間は次のページ参照)。また、書き込み後、最大 10ms は書き込み作業中のため、アクセスできません。データは 64 バイトまで指定できます。書き込む個数は、2 の n 乗個とします。</p> <p>引 数: unsigned int アドレス 0 ~ 32767 , int 個数 1 ~ 64 , char* データがあるアドレス</p> <p>戻り値: なし</p> <p>例) char d[4]; d[0]=5; d[1]=4; d[2]=1; d[3]=10; pageWriteI2CEeprom(0x1000, 4, d); EEP-ROM の 0x1000 番地から、配列 d にあるデータを 4 バイト分書き込みます。</p> <p>この関数は、書き込み時間が長いので、使用しません。 setPageWriteI2CEeprom 関数と I2CEepromProcess 関数を使用します。次を参照してください。</p>
<p>setPageWrite I2CEeprom</p>	<pre>void setPageWriteI2CEeprom (unsigned int address, int count, char* data);</pre> <p>EEP-ROM へ複数バイトのデータを書き込みます。書き込み準備を行うだけですぐに終了します。実際の書き込みは I2CEepromProcess 関数で行います。書き込む個数は、2 の n 乗個とします。2 バイト、4 バイト、8 バイト... です。</p> <p>引 数: unsigned int アドレス 0 ~ 32767 , int 個数 1 ~ 64 , char* データがあるアドレス</p> <p>戻り値: なし</p> <p>例) char d[4]; d[0]=5; d[1]=4; d[2]=1; d[3]=10; setPageWriteI2CEeprom(0x1000, 4, d); 書き込み準備のみ</p> <pre>while(1) { I2CEepromProcess(); 実際の書き込みはこの関数 }</pre>
<p>I2CEeprom Process</p>	<pre>void I2CEepromProcess(void);</pre> <p>この関数は、setPageWriteI2CEeprom 関数で書き込みの準備をしたデータを、実際にEEP-ROM に書き込みます。</p> <p>書き込み作業は、「少しずつ行いすぐに終わる」を何度も繰り返しておこなっています。こうして、書き込み処理にかかりっきりにならないようにしています。この関数は、書き込みデータ数+5 回以上実行してください。例えば、8 バイト書き込むときは最低でも 13 回、この関数を実行します。</p> <p>通常はループ内に入れておきます。書き込み作業がないときは何もみませんので、入れておくだけで OK です。</p> <p>引 数: なし</p> <p>戻り値: なし</p> <p>例) main { init(); while(1) { I2CEepromProcess(); 常に実行するようにする その他の処理 } }</p>
<p>clearI2CEeprom</p>	<pre>void clearI2CEeprom(void);</pre> <p>EEP-ROM のデータをオールクリアします。実行には数秒かかります。実測で 4 ~ 5 秒です。</p> <p>引 数: なし</p> <p>戻り値: なし</p> <p>例) clearI2CEeprom();</p>

checkI2CEeprom	<pre>int checkI2CEeprom(void);</pre> <p>書き込み後、書き込み作業が終わったかどうかチェックします。 引 数:なし 戻り値:1 読み書き可能 0 書き込み作業中(アクセス不可) 例) while(!checkEeprom()); 書き込みが終了したかチェックします。</p>
----------------	--

pageWritel2CEeprom 関数の書き込み時間について(このプログラムでは使用していません、説明用です)

pageWritel2CEeprom 関数を実行すると、EEP-ROM へ書き込みを行います。書き込むバイト数により実測で下記の時間がかかります。

pageWritel2CEeprom 実行時間 = $130 + 36.5 \times \text{書き込むバイト数}$ [μs]

例えば、今回 8 バイト書き込みますが、その場合は

pageWritel2CEeprom 実行時間 = $130 + 36.5 \times 8 = 422$ [μs]

かかります。この時間は、非常に大きいです。参考までに、kit05.c でそれぞれのパターンを実行する時間を実測してみました。

パターン	処理時間[μs]
0	5.5 ~ 6.5
1	EEP-ROM 初期化時間 約 5 秒
11	75 ~ 80 ただし、10ms に 1 回(EEP-ROM 書き込み時)500 程度
12,13	50
23	45

例えば、パターン 0 プログラムは 5.5 ~ 6.5[μs]ごとに繰り返されています。パターン 11 は 70 ~ 80[μs]ごとに繰り返されています。ただし、10[ms]に 1 回 500[μs]となっています。これは EEPROM にデータを書き込むときの時間です。いつもは、70 ~ 80[μs]ごとにセンサのチェック、モータの PWM 設定、サーボの PWM 設定を行っています。書き込むときだけ先ほどの計算のとおり、約 420[μs]間は書き込み処理だけに時間を取られてしまいます。この 420[μs]間はセンサのチェックやモータ PWM の設定ができない時間なのです。

サーボを自作して一定時間ごとにサーボ PWM を調整している場合、EEP-ROM にデータを書き込むときは間隔が変わってしまうのでうまくサーボ PWM を設定できないことがあります。

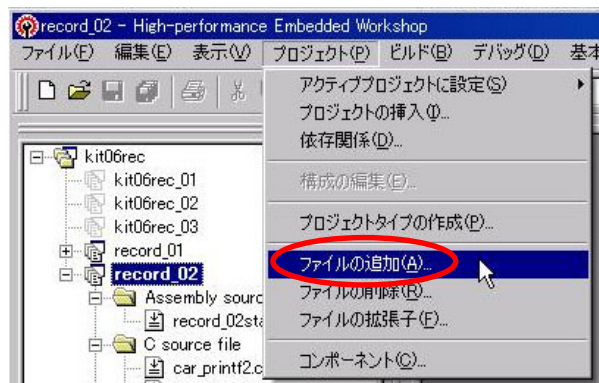
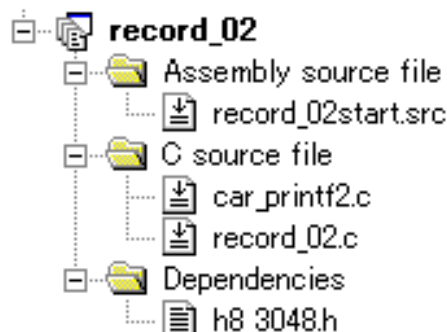
これを防ぐために今回のプログラムは、setPageWritel2CEeprom 関数と I2CEepromProcess 関数を併用します。

setPageWritel2CEeprom 関数は、書き込みを行う準備だけ行います。そのため、すぐに(数 μs で)処理を終えます。

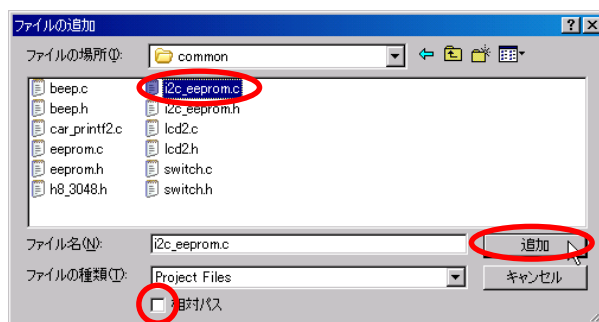
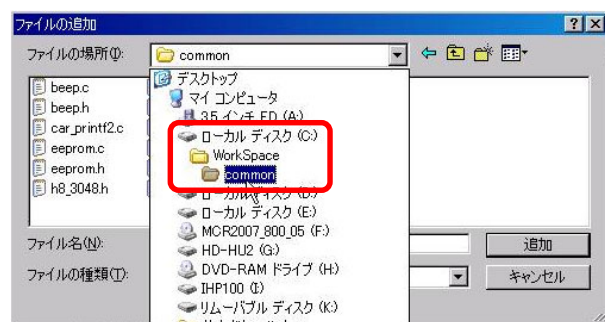
I2CEepromProcess 関数で実際に EEPROM へ書き込みを行います。書き込みは少しずつ行い、できるだけ時間をかけないようにします。その代わり、複数回、呼び出す必要があります。呼び出す回数は、書き込み数 + 5 回です。例えば、8 バイト書き込む場合、13 回この関数を呼び出す必要があります。実行回数をチェックするのは大変なので、I2CEepromProcess 関数はループ内に入れておき、常に実行するようにしておきます。実際の使い方は以後に説明します。書き込むデータがない場合は、何もしないので常に実行していて問題ありません。

1.10 「i2c_eeprom.c」を登録する方法

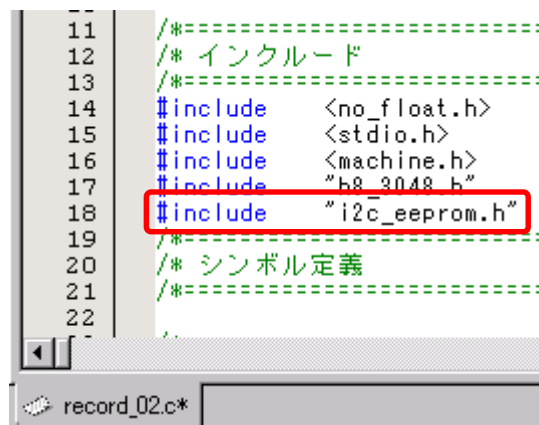
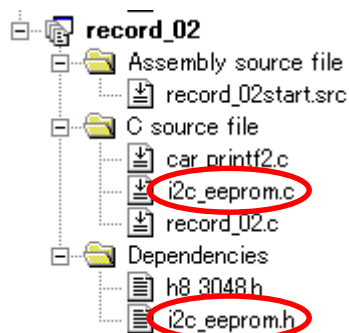
EEP-ROM を使うためには、プロジェクトに「i2c_eeprom.c」を登録しなければ行けません。例としてプロジェクト「record_02」に「i2c_eeprom.c」を追加する手順を説明します。



1. プロジェクト「record_02」には、まだ「i2c_eeprom.c」が登録されていないとします。
2. 「プロジェクト ファイルの追加」を選択します。



3. 「C:\¥Workspace¥common」フォルダを選択します。
4. 「相対パス」のチェックを外します。「i2c_eeprom.c」を選択、追加をクリックします。



5. 「i2c_eeprom.c」がファイルリストに登録されました。Dependencies 欄には、自動的に「i2c_eeprom.h」が登録されます。
6. 「record_02.c」で「i2c_eeprom.c」内の関数を使用するために、インクルード欄に 部分を追加します。

上記手順で「i2c_eeprom.c」を追加します。

EEP-ROMを使用するCソースファイルに「#include "i2c_eeprom.h"」の追加を忘れないようにしてください。

1.11 EEP-ROM の接続端子を変える場合の設定

本書で紹介している回路は、PA7 に EEP-ROM の SCL 端子、PA5 に EEP-ROM の SDA 端子を接続しています。これらの端子をすでに使って、違う端子を使用したい場合は、initI2CEeprom 関数の引数を変えます。引数は次のようになります。

```
initI2CEeprom( &1, &2, 3, 4, 5 );
```

1...&EEP-ROM の DDR レジスタ
2...&EEP-ROM の DR レジスタ
3...1で指定した DDR ポートの入出力設定
4...SCL 端子が接続されているビット
5...SDA 端子が接続されているビット

ポート7には接続できません。

例えば、EEP-ROM をポート6に接続するとします。ポート6に繋がっている内容と入出力設定は下記のようにします。

bit	7	6	5	4	3	2	1	0
接続名	-	EEP-ROM の SCL 端子	EEP-ROM の SDA 端子	未接続	ディップ スイッチ	ディップ スイッチ	ディップ スイッチ	ディップ スイッチ
入力 or 出力	出力	入力	入力	出力	入力	入力	入力	入力

DDR レジスタの設定は、入力"0"、出力"1"にするだけです。

bit	7	6	5	4	3	2	1	0
0 or 1	1	0	0	1	0	0	0	0

2 進数で"1001 0000"ですので、16 進数で 0x90 となります。これから、下記のように設定します。

```
initI2CEeprom( &P6DDR, &P6DR, 0x90, 6, 5 );
```

init 関数内での P6DDR の設定

initI2CEeprom 関数で設定していますので、init 関数内でポート6の入出力設定は必要ありません。今までのプログラムどおり、設定しておきたい場合は、initI2CEeprom 関数の引数と同じ入出力設定をしてください。下記にプログラム例を示します。

```
void init( void ) {
    ...
    P6DDR = 0x90;
    ...
}
```

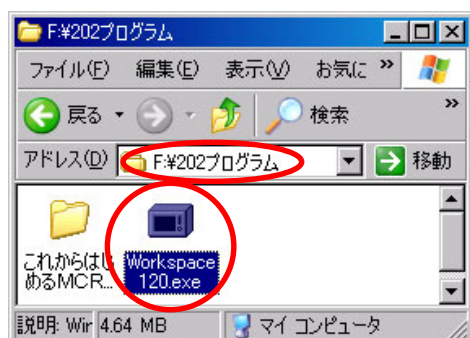
2. サンプルプログラム

2.1 ルネサス統合開発環境

サンプルプログラムは、ルネサス統合開発環境 (High-performance Embedded Workshop) を使用して開発するように作っています。ルネサス統合開発環境についてのインストール、開発方法は、「ルネサス統合開発環境操作マニュアル」を参照してください。

2.2 サンプルプログラムのインストール

サンプルプログラムをインストールします。



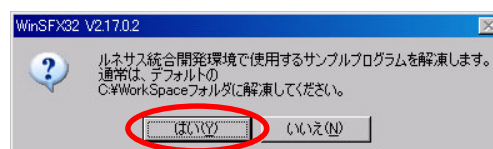
1. 講習会 CD の「CD ドライブ 202 プログラム」フォルダにある、Workspace120.exe を実行します。数字は、バージョンです。バージョンにより数字は異なります。
2. または、マイコンカーラリーサイト「<http://www.mcr.gr.jp/>」の技術情報 ダウンロード内のページへ行きます。

マイコンカーキットプログラム、開発環境の資料

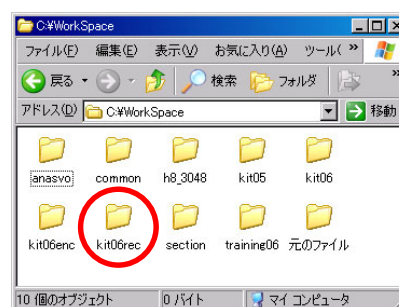
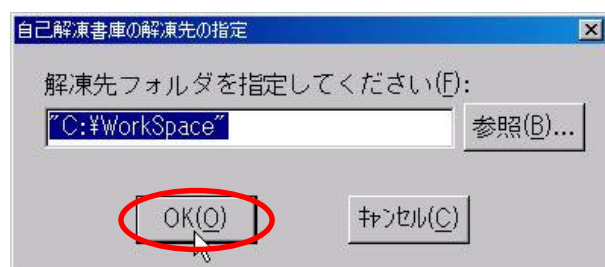
●ルネサス統合開発環境 操作マニュアル 第1.02版 2006.07.13 **NEW!**
ルネサス統合開発環境のダウンロード方法、インストール方法、操作方法、及び設定方法が載っています。ルネサス統合開発環境は、ルネサステクノロジサイトよりダウンロードして下さい(マニュアル内に方法が記載されています)。
→ [DOWNLOAD](#) (PDF 約19.3MB)

●ルネサス統合開発環境用他ソフト Ver1.00 2006.07.06 **NEW!**
ルネサス統合開発環境以外で使用するソフトをインストールします。
→ [DOWNLOAD](#) (EXE 約0.3MB)

●ルネサス統合開発環境用マイコンカー関連プログラム Ver1.00 2006.07.06 **NEW!**
ルネサス統合開発環境用のマイコンカー関係プログラムです。自己解凍方式で、実行すると自動でプログラムがインストールされます。
→ [DOWNLOAD](#) (EXE 約3.68MB)



3. 「ルネサス統合開発環境用マイコンカー関連プログラム」をダウンロードします。
4. CD またはダウンロードした「Workspace120.exe」を実行します。「はい」をクリックします。

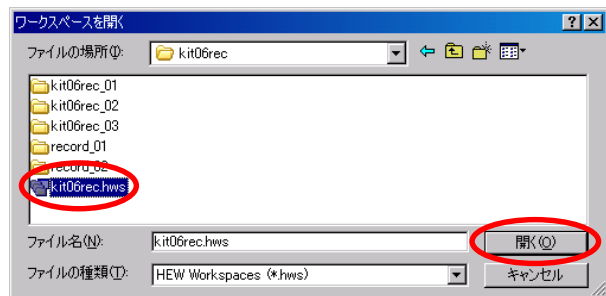
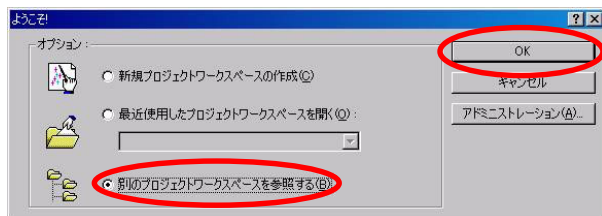


5. ファイルの解凍先を選択します。フォルダの変更はできません。OKをクリックします。
6. 解凍が終わったら、エクスプローラで「C ドライブ WorkSpace」フォルダを開いてみてください。複数のフォルダがあります。今回使用するのは、「kit06rec」です。

2.3 ワークスペース「kit06rec」を開く

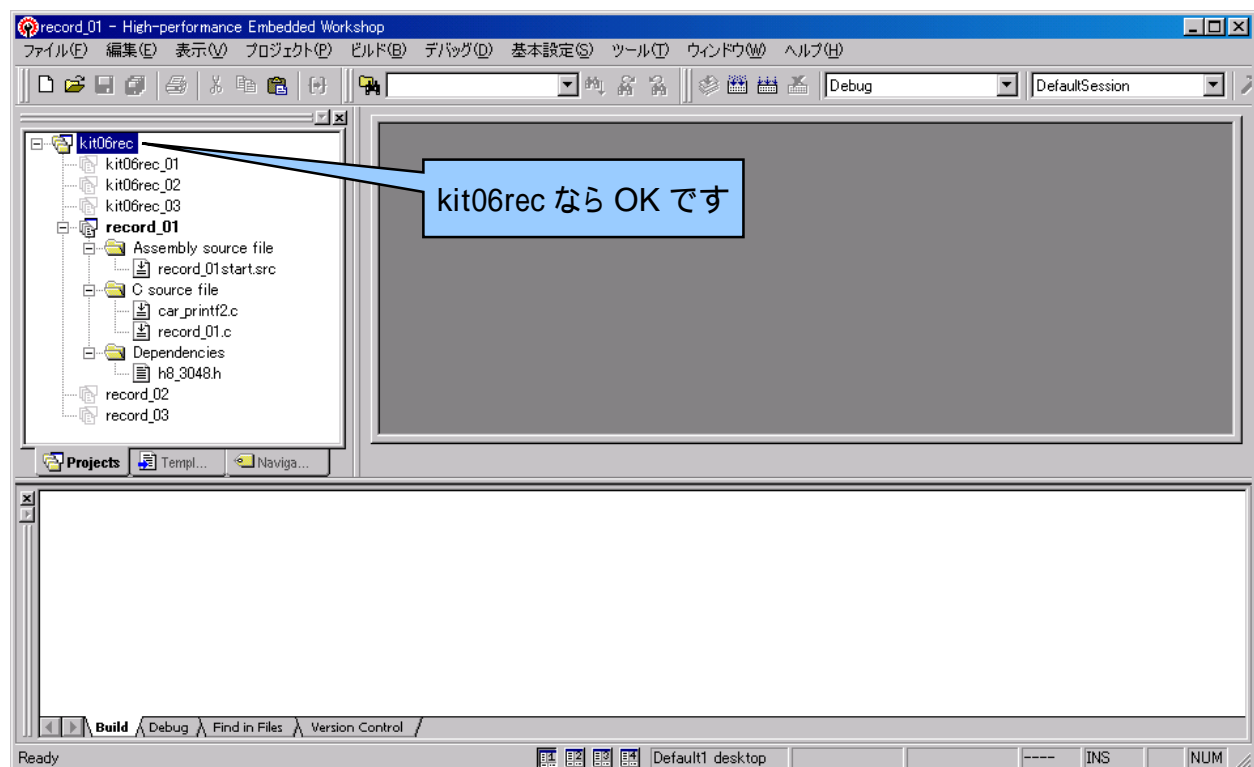


1. ルネサス統合開発環境を実行します。



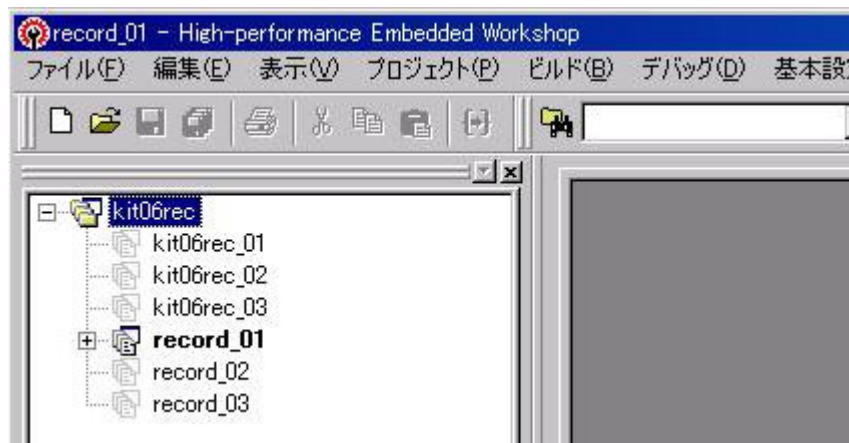
2. 「別のプロジェクトワークスペースを参照する」を選択、**OK**をクリックします。

3. Cドライブ Workspace kit06rec の「kit06rec.hws」を選択、**開く**をクリックします。



4. kit06rec というワークスペースが開かれます。

2.4 プロジェクト



ワークスペース「kit06rec」には、6 つのプロジェクトが登録されています。

プロジェクト名	内容
record_01	H8/3048F-ONE の内蔵 RAM を使用し、データを記録、転送するプログラムです。内蔵 RAM に保存する、動作理解用のサンプルプログラムです。
record_02	24C256 という外付けの EEP-ROM を使用し、データを記録、転送するプログラムです。EEP-ROM に保存する、動作理解用のサンプルプログラムです。
record_03	record_02.c を改造し、転送データをあらかじめ 2 進数に変換して出力します。
kit06rec_01	走行データの記録をします。記録には、H8/3048F-ONE の内蔵 RAM を使用します。約 2.5KB 保存することができます。
kit06rec_02	走行データの記録をします。記録には、24C256 という外付けの EEP-ROM を使用します。32KB 保存することができます。
kit06rec_03	kit06rec_02.c プログラムを改造して、ロータリエンコーダが使えるようにしたプロジェクトです。記録データは、今までの内容に走行スピードを追加しています。

3. プロジェクト「record_01」 内蔵 RAM にデータ保存

3.1 概要

このプログラムは、

- ・ポート7に接続されているディップスイッチの値
- ・CPU ボードのディップスイッチの値

を、10ms ごとに内蔵 RAM に保存します。保存時間は 10 秒です。保存時間は自由に変わりますが上限はメモリがいっぱいになるまでです(今回は約 12.5 秒、1 回に保存するデータ数で変わります)。保存後、RS232C を通してパソコンへ保存した情報を出力します。

ここでは、

- ・データを保存する方法
- ・CPU ボードからパソコンへデータを出力する方法
- ・パソコンで受信したデータを、エクセルに取り込む方法
- ・エクセルに取り込んだデータの解析方法

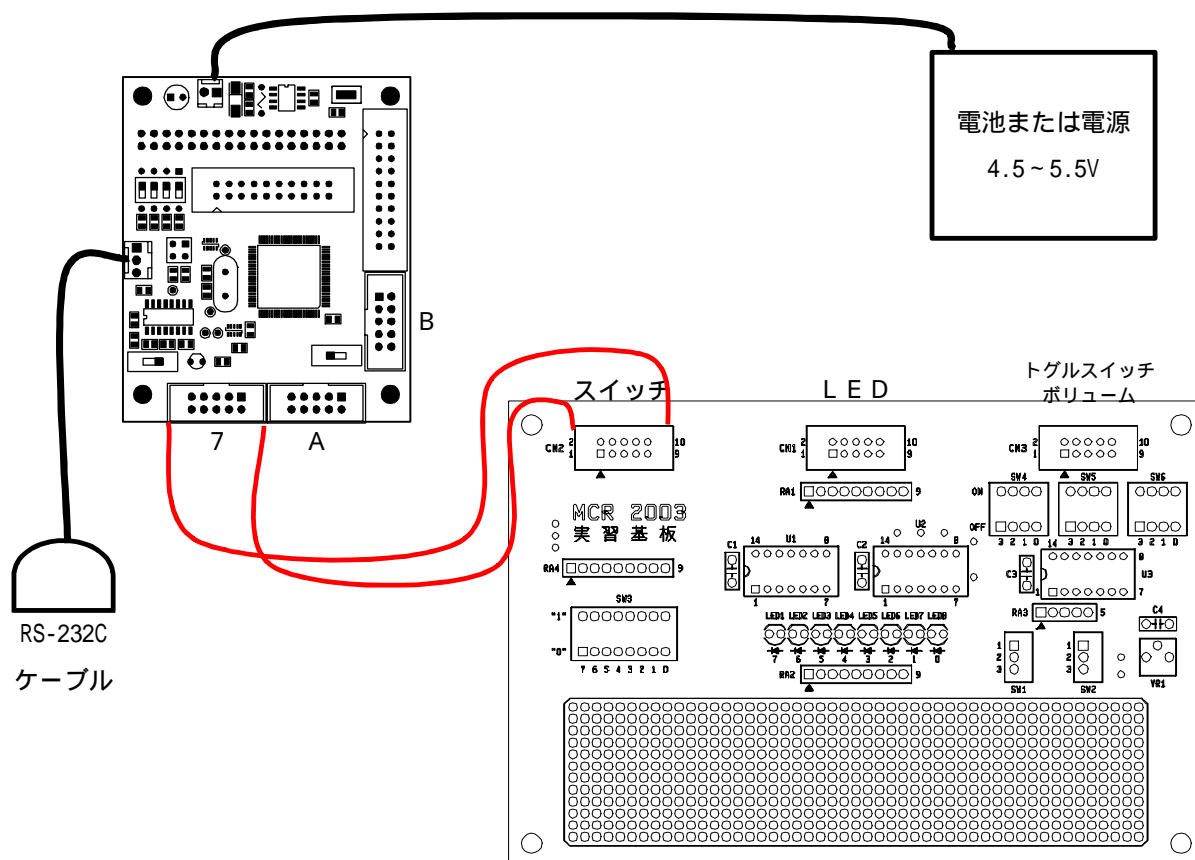
について、説明します。

内蔵 RAM については、「1.3 EEP-ROM を使う意義」を参照してください。

3.2 接続

- ・CPU ボードのポート7と、実習基板のスイッチ部をフラットケーブルで接続します。

ポート7のディップスイッチをセンサ基板に変えると、センサの反応を記録することができます。



3.3 プロジェクトの構成



・record_01start.src

・record_01.c

・car_printf2.c

の3ファイルあります。

h8_3048.h は record_01.c、car_printf2.c でインクルードされているファイルです。

3.4 プログラム

```

1 :  /******
2 :  /* 内蔵RAMデータ保存演習プログラム「record_01.c」
3 :  /* 2006.04 ジャパンマイコンカーラリー実行委員会
4 :  /******
5 :  /*
6 :  本プログラムはH8/3048F-ONEの内蔵RAM(2.5KB程度)にポート7のデータ、
7 :  CPUボード上のディップスイッチの値を10msごとに保存します。
8 :  その後、保存したデータをパソコンへ転送します。
9 :  */
10 :
11 :  /*=====
12 :  /* インクルード
13 :  /*=====
14 :  #include <no_float.h> /* stdioの簡略化 最初に置く*/
15 :  #include <stdio.h>
16 :  #include <machine.h>
17 :  #include "h8_3048.h"
18 :
19 :  /*=====
20 :  /* シンボル定義
21 :  /*=====
22 :  #define SAVE_SIZE 1000 /* データ保存数 */
23 :
24 :  /*=====
25 :  /* プロトタイプ宣言
26 :  /*=====
27 :  void init( void );
28 :  unsigned char dipsw_get( void );
29 :
30 :  /*=====
31 :  /* グローバル変数の宣言
32 :  /*=====
33 :  unsigned long cnt1; /* main内で使用 */
34 :  int pattern; /* パターン番号 */
35 :
36 :  /* データ保存関連 saveDataのサイズは最大で2.5KB程度としてください */
37 :  int iTimer10; /* 取得間隔計算用 */
38 :  unsigned char saveData[SAVE_SIZE][2]; /* 保存エリア */
39 :  int saveIndex; /* 保存インデックス */
40 :  int saveSendIndex; /* 送信インデックス */
41 :  int saveFlag; /* 保存フラグ */
42 :
43 :  /******
44 :  /* メインプログラム
45 :  /******
46 :  void main( void )
47 :  {
48 :      /* マイコン機能の初期化 */
49 :      init(); /* 初期化 */
50 :      init_sci1( 0x00, 79 ); /* SCI1初期化 */
51 :      set_ccr( 0x00 ); /* 全体割り込み許可 */
52 :
53 :      while( 1 ) {
54 :          switch( pattern ) {
55 :
56 :          case 0:
57 :              /* 1秒待ち */
58 :              if( cnt1 > 1000 ) {
59 :                  printf( "%n" );
60 :                  printf( "data recording...%n" );
61 :                  pattern = 1;
62 :                  saveIndex = 0;
63 :                  saveFlag = 1; /* データ保存開始 */
64 :                  cnt1 = 0;
65 :              }

```



```

66 :         break;
67 :
68 :     case 1:
69 :         /* データ保存中 保存自体は割り込みの中で行う */
70 :         if( saveFlag == 0 ) {
71 :             pattern = 2;
72 :         }
73 :         break;
74 :
75 :     case 2:
76 :         /* 転送 */
77 :         printf( "%n" );
78 :         printf( "record_01 Data Out%n" );
79 :         printf( "P7 data,dip sw data%n" );
80 :
81 :         saveSendIndex = 0;
82 :         pattern = 3;
83 :         break;
84 :
85 :     case 3:
86 :         /* データ転送 */
87 :         printf( "%02x,%d%n",
88 :             saveData[saveSendIndex][0],
89 :             saveData[saveSendIndex][1] );
90 :         saveSendIndex++;
91 :         if( saveIndex <= saveSendIndex ) {
92 :             pattern = 4;
93 :             cnt1 = 0;
94 :         }
95 :         break;
96 :
97 :     case 4:
98 :         /* 転送終了 */
99 :         break;
100 :
101 :     default:
102 :         /* どれも無い場合は待機状態に戻す */
103 :         pattern = 0;
104 :         break;
105 :     }
106 : }
107 :
108 :
109 : /*****
110 : /* H8/3048F-ONE 内蔵周辺機能 初期化 */
111 : *****/
112 : void init( void )
113 : {
114 :     /* I/Oポートの入出力設定 */
115 :     P1DDR = 0xff;
116 :     P2DDR = 0xff;
117 :     P3DDR = 0xff;
118 :     P4DDR = 0xff;
119 :     P5DDR = 0xff;
120 :     P6DDR = 0xf0; /* CPU基板上的DIP SW */
121 :     P8DDR = 0xff;
122 :     P9DDR = 0xf7; /* 通信ポート */
123 :     PADDR = 0xff;
124 :     PBDDR = 0xff;
125 :     /* ポート7は、入力専用なので入出力設定はありません */
126 :
127 :     /* ITU0 1msごとの割り込み */
128 :     ITU0_TCR = 0x20;
129 :     ITU0_GRA = 24575;
130 :     ITU0_IER = 0x01;
131 :
132 :     /* ITUのカウントスタート */
133 :     ITU_STR = 0x01;
134 : }
135 :
136 : /*****
137 : /* ITU0 割り込み処理 */
138 : *****/
139 : #pragma interrupt( interrupt_timer0 )
140 : void interrupt_timer0( void )
141 : {
142 :     ITU0_TSR &= 0xfe; /* フラグクリア */
143 :     cnt1++;
144 :
145 :     /* データ保存関連 */
146 :     iTimer10++;
147 :     if( iTimer10 >= 10 ) {
148 :         iTimer10 = 0;
149 :         if( saveFlag ) {
150 :             saveData[saveIndex][0] = P7DR;
151 :             saveData[saveIndex][1] = dipsw_get();
152 :             saveIndex++;
153 :             if( saveIndex >= SAVE_SIZE ) saveFlag = 0;
154 :         }
155 :     }
156 : }

```

```

157 :
158 : /*****
159 : /* ディップスイッチ値読み込み
160 : /* 戻り値 スイッチ値 0~15
161 : *****/
162 : unsigned char dipsw_get( void )
163 : {
164 :     unsigned char sw;
165 :
166 :     sw = P6DR;
167 :     sw &= 0x0f;
168 :
169 :     return sw;
170 : }
171 :
172 : /*****
173 : /* end of file
174 : *****/

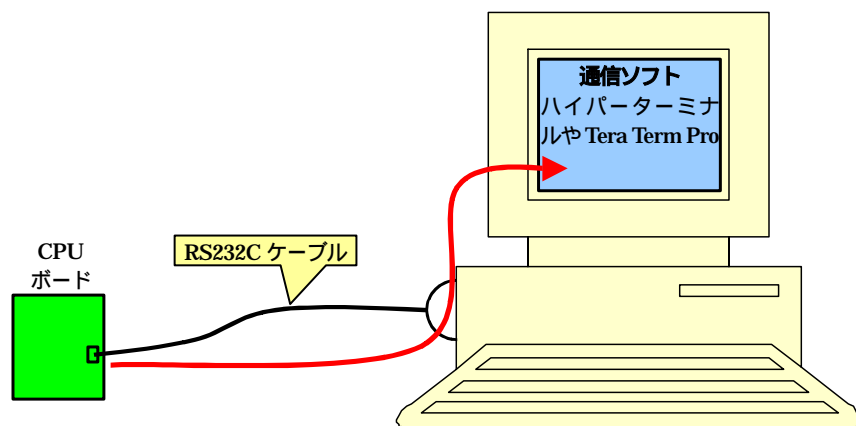
```

3.5 データをパソコンへ送る

詳しくは、H8/3048 実習マニュアルのプロジェクト「sio」部分を参照してください。ここでは簡単に説明します。

CPU ボードからパソコンへのデータ転送は、RS232C ケーブルを通して行います。

パソコンには、ハイパーターミナルや TeraTermPro などの通信ソフトを立ち上げておきます。通信ソフトは、CPU ボードから送られてきたデータを RS232C ケーブルを通して受信、通信ソフトの画面上に表示します。表示すると共に、受信データをファイルに保存することもできます。エクセルなどでそのファイルを取り込むことにより、さまざまな解析をすることができます。



CPU ボード RS232C ケーブル 通信ソフトの画面やファイルへ保存

今回、

```
printf("Hello World!\n");
```

とすると、H8 マイコンは RS232C に printf 文のメッセージを出力します。通信ソフトは、RS232C からデータを受信します。そして通信ソフトの画面上に

```
Hello World!
```

と表示します。

H8 の C 言語プログラムとパソコンの C 言語プログラムが printf 文を実行したとき、出力先が違おうことを押さえておいてください。

printf 文を実行する機器	出力先	表示
パソコン	ディスプレイ端子	ディスプレイ
H8 マイコン	RS232C 端子	通信ソフトの画面

H8 マイコンの場合、printf 文の出力先を RS232C にするのが、「car_printf2.c」です。本プロジェクトには 3 つの C ソースファイルや src ソースファイルがあります。それぞれのファイル内容は、下記のようになっています。

ファイル名	内容
record_01start.src	ベクタアドレスの設定、スタートアップルーチンが入っているファイルです。
record_01.c	C 言語のメインプログラムファイルです。
car_printf2.c	<ul style="list-style-type: none"> ・通信するための設定 ・printf 関数の出力先、scanf 関数の入力元を RS232C ポートにするための設定 ・セクション D,R,B を初期化する設定 を行っています。 プロジェクト「record_01」に限らず、これらを行うプロジェクトには、car_printf2.c ファイルを追加します。

3.6 プログラムの解説

3.6.1 内蔵周辺機能の初期設定

```

109 :  /***** /
110 :  /* H8/3048F-ONE 内蔵周辺機能 初期化 */
111 :  *****/
112 :  void init( void )
113 :  {
114 :      /* I/O ポートの入出力設定 */
115 :      P1DDR = 0xff;
116 :      P2DDR = 0xff;
117 :      P3DDR = 0xff;
118 :      P4DDR = 0xff;
119 :      P5DDR = 0xff;
120 :      P6DDR = 0xf0;          /* CPU 基板上の DIP SW */
121 :      P8DDR = 0xff;
122 :      P9DDR = 0xf7;          /* 通信ポート */
123 :      PADDR = 0xff;
124 :      PBDDR = 0xff;
125 :      /* ポート 7 は、入力専用なので入出力設定はありません */
126 :
127 :      /* ITU0 1ms ごとの割り込み */
128 :      ITU0_TCR = 0x20;
129 :      ITU0_GRA = 24575;
130 :      ITU0_IER = 0x01;
131 :
132 :      /* ITU のカウントスタート */
133 :      ITU_STR = 0x01;
134 :  }

```

ポート 6 の bit3 ~ 0 は、ディップスイッチ入力です。ポート 7 は、実習基板のディップスイッチ、またはセンサ基板入力です。それぞれの端子を入出力設定します。
他に、ITU0 を 1ms ごとの割り込み設定にします。

3.6.2 データ取得関係の定義、変数

```

19 : /*=====*/
20 : /* シンボル定義 */
21 : /*=====*/
22 : #define      SAVE_SIZE    1000      /* データ保存数 */
23 :
24 : /*=====*/
25 : /* プロトタイプ宣言 */
26 : /*=====*/
27 : void init( void );
28 : unsigned char dipsw_get( void );
29 :
30 : /*=====*/
31 : /* グローバル変数の宣言 */
32 : /*=====*/
33 : unsigned long cnt1; /* main 内で使用 */
34 : int pattern; /* パターン番号 */
35 :
36 : /* データ保存関連 saveData のサイズは最大で 2.5kbytes 程度としてください */
37 : int iTimer10; /* 取得間隔計算用 */
38 : unsigned char saveData[SAVE_SIZE][2]; /* 保存エリア */
39 : int saveIndex; /* 保存インデックス */
40 : int saveSendIndex; /* 送信インデックス */
41 : int saveFlag; /* 保存フラグ */

```

太字部分が、今回追加した変数です。

変数名	意味	内容
iTimer10	保存間隔計算用	データを保存する間隔を調整するために、この変数で計算します。割り込みの間隔は 1ms、保存する間隔は 10ms です。そのため、この変数を割り込みごとに+1 します。10 になったら、10ms たったと判断して、データ保存処理を行います。
saveData	データ保存用	saveData は 2 次元配列で、SAVE_SIZE 個分の値を保存します。saveData は unsigned char 型なので、メモリは 1 データ 1 バイト使用します。「saveData[SAVE_SIZE][2]」なので、1 回の保存で 2 バイト使用します。H8/3048F-ONE の RAM 領域は 4KB ありますが、すべて使えるわけではありません。関数の戻り先アドレスの保存、レジスタの値保存など、マイコン側でも使用します。そのため、最大で約 2.5KB の確保が限界です。今回は 1 回で 2 バイト分のメモリを使用するので、保存回数は 1250 回が限界です。
saveIndex	保存インデックス	配列の何番目に保存するかを指定する変数です。
saveSendIndex	保存送信インデックス	配列の何番目のデータを送信するかを指定する変数です。
saveFlag	保存フラグ	1 なら割り込みプログラム内で 10ms ごとにデータを保存します。0 なら保存しません。

3.6.3 パソコンとの通信するための初期設定

```

48 :      /* マイコン機能の初期化 */
49 :      init();                      /* 初期化                */
50 :      init_sci1( 0x00, 79 );        /* SCI1初期化            */
51 :      set_ccr( 0x00 );              /* 全体割り込み許可      */

```

パソコンと通信するために、H8/3048F-ONE の内蔵周辺機能である SCI1 の初期化を行います。car_printf2.c 内に、簡単に SCI1 関係のレジスタを初期化することができる init_sci1 関数がありますので、この関数を使って初期化します。この設定により、9600bps で通信することができます。詳細については、「H8/3048 実習マニュアル」のプロジェクト「sio」の解説を参照してください。

3.6.4 パターン 0:1 秒待ち

```

56 :      case 0:
57 :          /* 1 秒待ち */
58 :          if( cnt1 > 1000 ) {
59 :              printf( "%n" );
60 :              printf( "data recording...%n" );
61 :              pattern = 1;
62 :              saveIndex = 0;
63 :              saveFlag = 1;          /* データ保存開始        */
64 :              cnt1 = 0;
65 :          }
66 :          break;

```

CPU ボードに電源を入れてから、1 秒待ちます。1 秒たつと
 59～60 行...データを記録する旨、メッセージを出力します。
 61 行...次に実行するときはパターン 1 になるよう、変数の設定をします。
 62 行...保存する配列の番号用の変数を初期化します。
 63 行...データの保存を開始します。
 64 行...時間計測用の cnt1 変数をクリアします。

次から、パターン 1 に移ります。

3.6.5 パターン 1:データ保存

```

68 :      case 1:
69 :          /* データ保存中 保存自体は割り込みの中で行う */
70 :          if( saveFlag == 0 ) {
71 :              pattern = 2;
72 :          }
73 :          break;

```

1 なら保存中
0 なら保存終了

データの保存を行います。保存は 10ms ごとに割り込み内で行われています。そのため、メインプログラムでは保存処理はしていません。保存する配列が一杯になると、割り込み内で saveFlag を 0 にします。メインプログラムでは、saveFlag 変数が 0 になったかを常にチェック、なったなら保存が終了したと判断して、パターン 2 へ移ります。

3.6.6 パターン 2:タイトル転送、準備

```

75 :      case 2:
76 :          /* タイトル転送、準備 */
77 :          printf( "¥n" );
78 :          printf( "record_01 Data Out¥n" );
79 :          printf( "P7 data,dip sw data¥n" );
80 :
81 :          saveSendIndex = 0;
82 :          pattern = 3;
83 :          break;

```

データを送信する前にタイトルなどを送信します。次に、saveSendIndex 変数を 0 にして、送信するデータの番号を初期化します。パターン 3 へ移ります。

3.6.7 パターン 3:データの転送

```

85 :      case 3:
86 :          /* データ転送 */
87 :          printf( "%02x,%d¥n",
88 :              saveData[saveSendIndex][0],
89 :              saveData[saveSendIndex][1] );
90 :          saveSendIndex++;
91 :          if( saveIndex <= saveSendIndex ) {
92 :              pattern = 4;
93 :              cnt1 = 0;
94 :          }
95 :          break;

```

ポート 7 のデータ読み込み
ディップスイッチのデータ読み込み

データを送信します。

87 ~ 89 行...配列からデータを読み込みながら、printf 文でデータを送信します。

printf 文のカッコ内の意味は、

「%02x」... 2 桁に満たない場合は 0 で埋めて 16 進数表記

「,」.....そのまま表示されます。

「%d」..... 10 進数表記

「¥n」..... 改行

となります。

90 行...saveSendIndex 変数を + 1 して、saveData 配列のデータを次に出力するデータにしておきます。

91 行...送信しようとしている配列が保存数より大きくなったならパターン 4 に移り、送信を終えます。

下記に、出力例を示します。

data recording...	ff,12
	ff,12
record_01 Data Out	ff,8
P7 data,dip sw data	ff,0
aa,15	ff,0
aa,15	ff,4
ab,15	ff,4
ab,15	ff,4
af,15	ff,6
af,15	ff,6
bf,15	ff,6
bf,15	ff,6
ff,15	ff,7
ff,15	ff,7
ff,14	ff,7
ff,14	ff,7
ff,14	ff,7

3.6.8 パターン 4:転送終了

97 :	case 4:
98 :	/* 転送終了 */
99 :	break;

終わりです。何もしません。

保存するデータを変えた
い場合は、ここを変更する

153 行で配列の上限に達していないかチェック、上限になったら saveFlag を 0 にして、保存を止めます。

	B=0	B=1
saveData[0][B]	10ms 後の P7DR	10ms 後のスイッチ値
saveData[1][B]	20ms 後の P7DR	20ms 後のスイッチ値
saveData[2][B]	30ms 後の P7DR	30ms 後のスイッチ値
saveData[3][B]	40ms 後の P7DR	40ms 後のスイッチ値
saveData[4][B]	50ms 後の P7DR	50ms 後のスイッチ値
...		
saveData[998][B]	9990ms 後の P7DR	9990ms 後のスイッチ値
saveData[999][B]	10000ms 後の P7DR	10000ms 後のスイッチ値

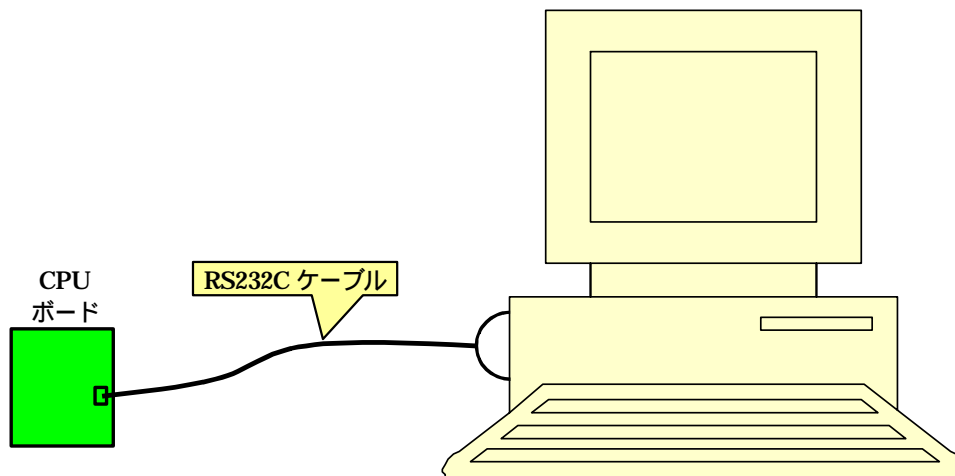
1000 行

2 列

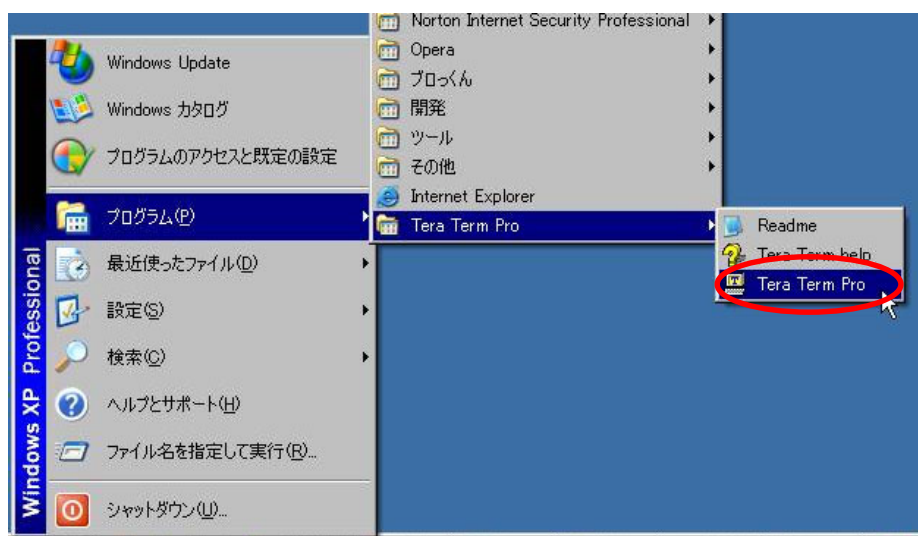
- 28 -

3.7 データの取り込み方

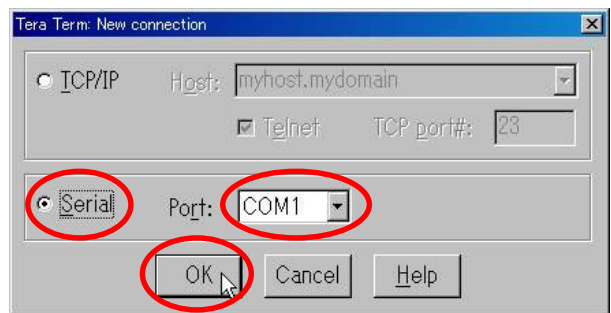
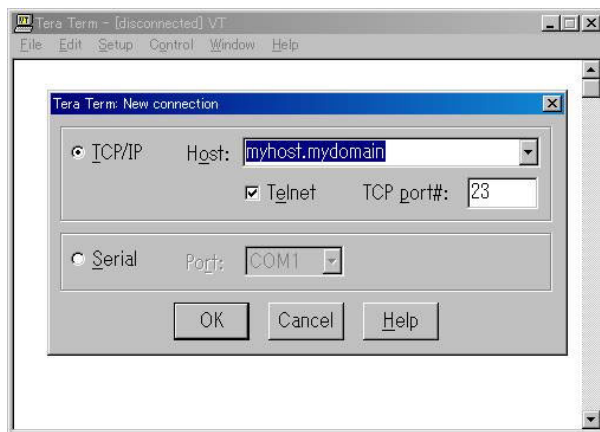
パソコンと通信ソフトを使ってデータを取り込む方法を説明します。



1. プロジェクト「record_01」をビルドして、「record_01.mot」ファイルを CPU ボードに書き込んでください。CPU ボードとパソコン間の RS232C ケーブルは繋いだままにしておきます。

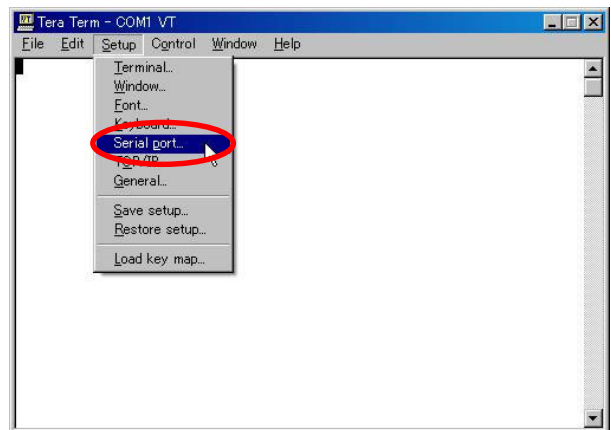
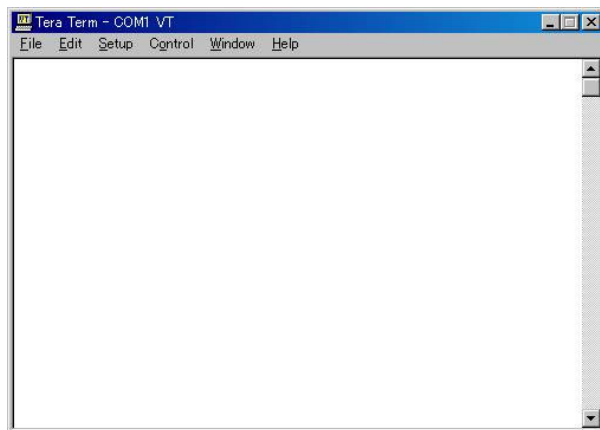


2. 「スタート すべてのプログラム(またはプログラム) Tera Term Pro Tera Term Pro」で Tera Term Pro を立ち上げます。Tera Term Pro をまだインストールしていない場合は、H8/3048 実習マニュアルのプロジェクト「sio」にある Tera Term Pro のインストール欄を参照してインストールしてください。



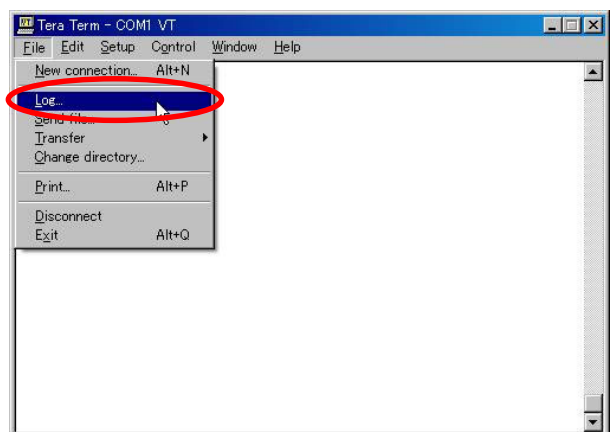
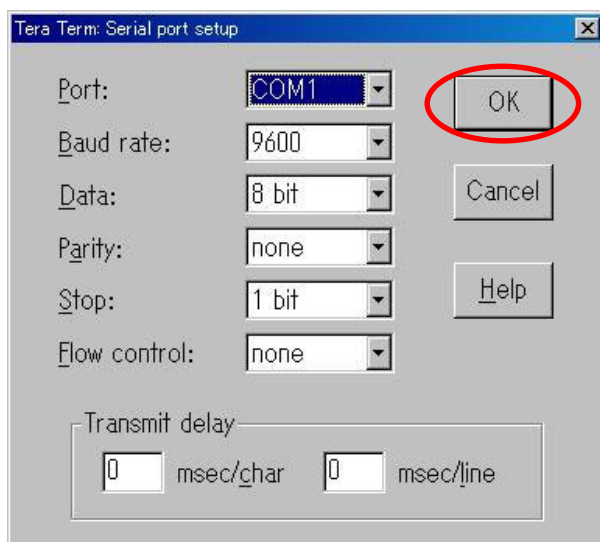
3. 接続先を確認する画面が表示されます。

4. 「Serial」を選んで、各自のパソコンに合わせてポート番号を選びます。選択後、**OK**をクリックして次へ進みます。



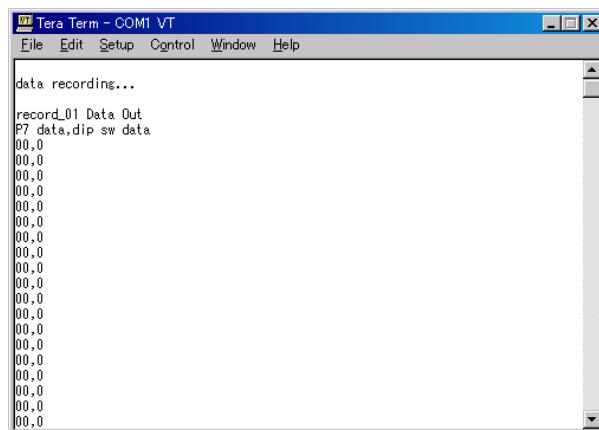
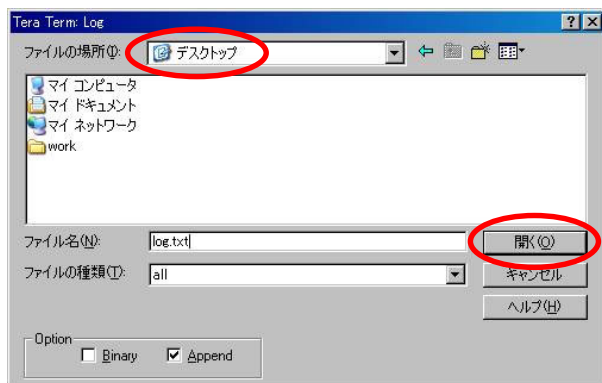
5. 立ち上がりました。詳細設定をします。

6. 「Setup Serial port」を選択します。



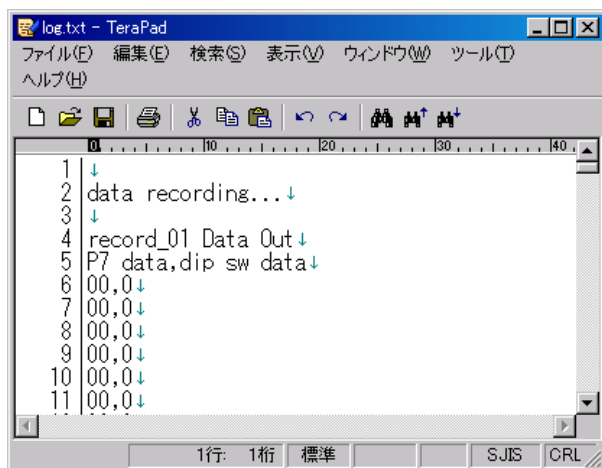
7. 通信設定を確認します。画面のように設定して、**OK**をクリックします (Port は変えません)。

8. 受信データをファイルに保存する設定をします。「File Log」を選択します。



9. 保存ファイル名を入力します。ここでは「log.txt」と入力します。保存するフォルダも分かりやすい位置に変更しておきましょう。今回は、「デスクトップ」にしています。ファイル名を設定できたら、**開く**をクリックします。

10. CPU ボードの電源を入れると、「data recording...」と表示され、10ms ごとにポート7とディップスイッチの値が保存されます。配列は 1000 個分確保しているので 10 秒保存します。10 秒たつと、保存したデータが TeraTermPro に送られてきます。転送が終了したら TeraTermPro は終了します。



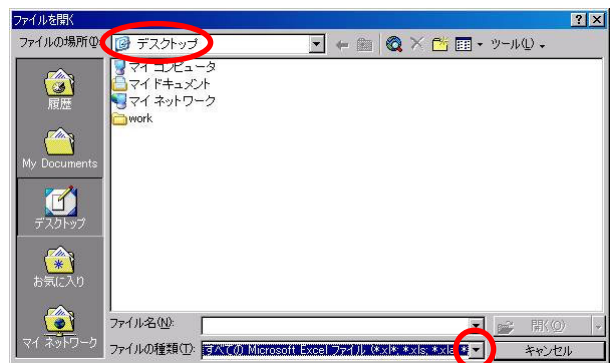
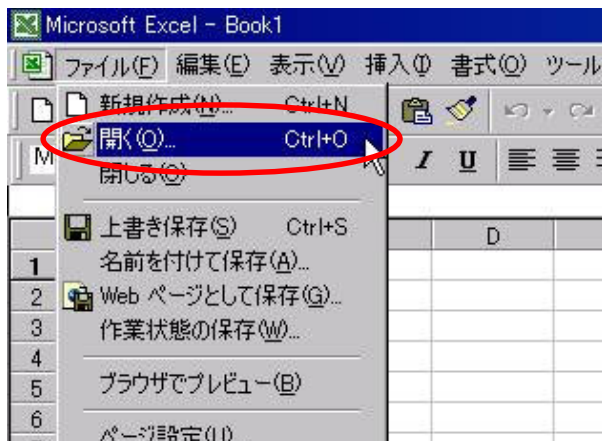
11. log.txt をエディタで開いてみました。保存されています。次は、このデータをエクセルに取り込んでみましょう。

注意

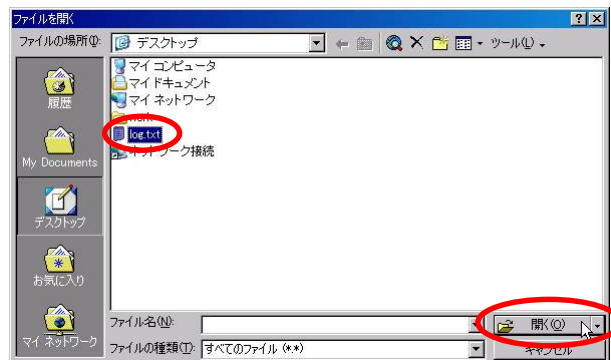
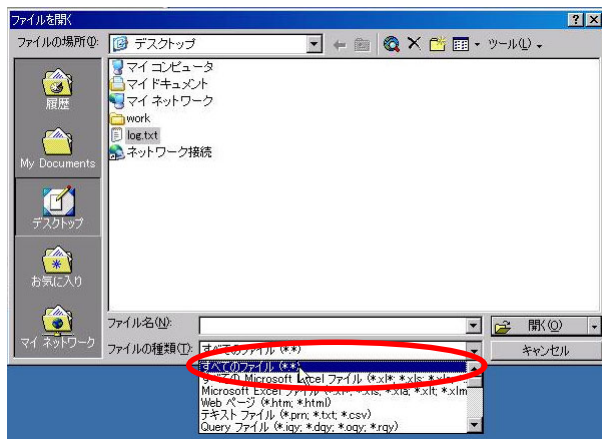
TeraTermPro は、受信前に「File Log」で保存するファイル名を決めます。その後、受信したデータをファイルに保存していきます。

受信したデータは画面に表示されますが、表示されるだけで残しません。データを受信してから、「File Log」を実行しても受信データは保存されませんので気をつけてください。

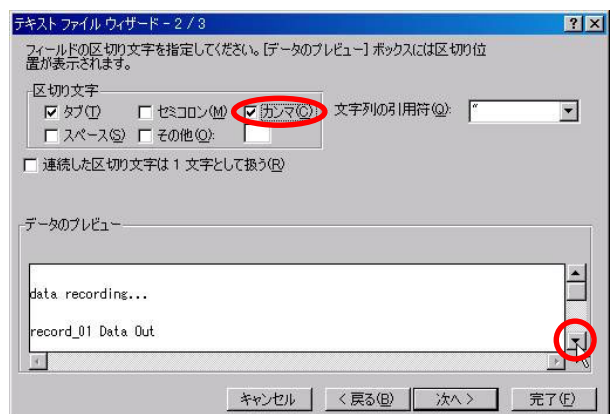
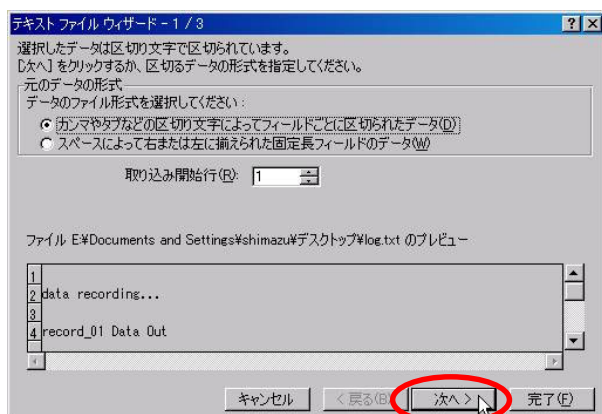
3.8 エクセルへの取り込み方



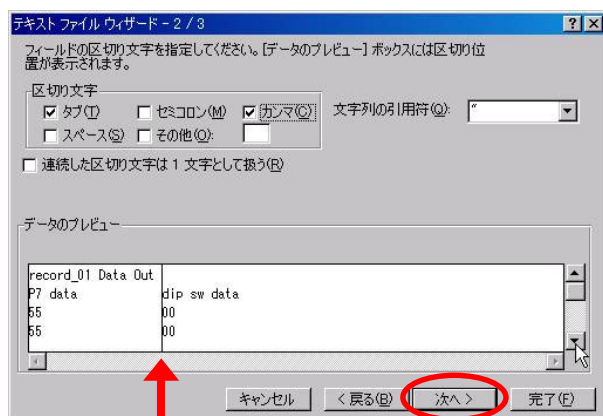
1. エクセルを立ち上げます。「ファイル 開く」を選択します。
2. 「ファイルの場所」は、先ほど保存したフォルダを選びます。「ファイルの種類」の□をクリックします。



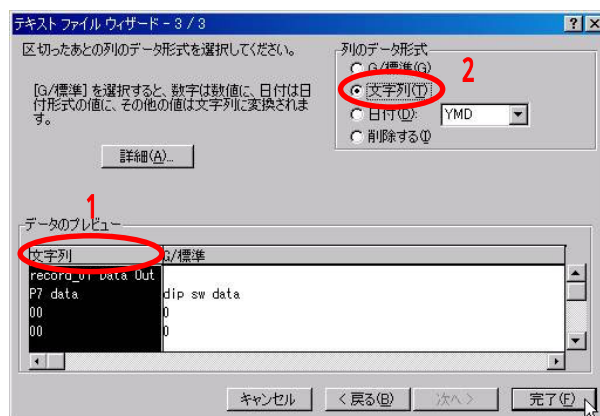
3. 「すべてのファイル (*.*)」を選択します。
4. 「log.txt」が表示されました。「log.txt」を選択、開くをクリックします。



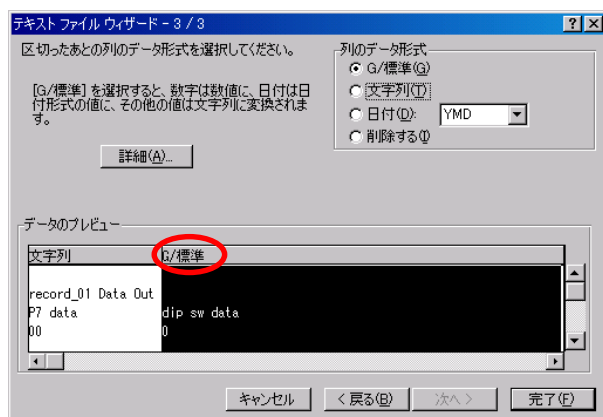
5. 開こうとしているデータは、テキストデータです。エクセルデータ(セル)に変換する作業を行います。次へをクリックします。
6. データはカンマで区切られています。区切り文字の「カンマ」のチェックを付けます。□でデータのプレビュー欄を下げます。



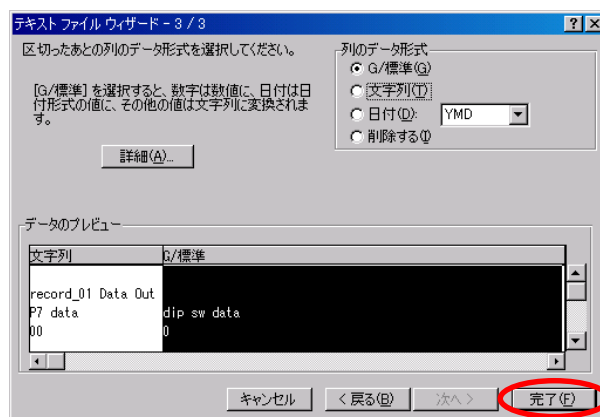
7. 矢印 () 部分に縦線が入りました。これが列の区切りです。縦線が入ってなければ「区切り文字」のチェック指定がおかしいので、確かめます。次へをクリックします。



8. 次に、列のデータ形式を指定します。まず、1列目 (1の) 部分をクリックします。次に2部分の「文字列」を選択します。ポート7データの列は、16進数です。そのため、文字列にします。



9. 次に、2列目の 部分をクリックします。2列目はディップスイッチ値を10進数で記録した内容なので、「標準」にします。最初から標準になっているので特に変更する必要はありません。



10. 完了をクリックして完了します。

	A	B	C	D	E
1					
2	data recording...				
3					
4	record_01 Data Out				
5	P7 data	dip sw data			
6	00	0			
7	00	0			
8	00	0			
9	00	0			
10	00	0			
11	00	0			
12	00	0			
13	00	0			
14	00	0			
15	00	0			

11. セルに変換されました。A列は文字列です。例えば、A6は「00」という文字です。B列は数値です。例えば、B6は「0」という数値です。関数で変換するときは、全く別な扱いになりますので気をつけてください。

	A	B	C	D	E
1					
2	data recording...				
3					
4	record_01 Data Out				
5	P7 data	dip sw data			
6	00	0			
7	00	0			
8	00	0			
9	00	0			
10	00	0			
11	00	0			

12. ポート7のデータは例えばセンサの値とします。センサ値が「00~ff」の16進数で保存されています。16進数といっても、保存されている扱いは文字列です。これを2進数に変換してみましょう。0を、1を に変換してみます。打ち込んでみました。これをすべて自分で打ち込むのは非常に大変です。

エクセルには、関数という便利な機能があります。関数を使えば、元のデータを様々な形に変換することができます。今回のように 16 進数を 2 進数に変換するなど、お手の物です。この関数を使ってポート7の 16 進数データを 2 進数に変換してみます。エクセルの関数については、書籍やホームページで多数紹介されていますので、詳しく知りたい場合はそちらを参照してください。

	A	B	C	D	E	F	G	H	I	J
1										
2	data recording...									
3										
4	record_01 Data Out									
5	P7 data	dip sw data								
6	00	0							0	
7	00	0								
8	00	0								
9	00	0								
10	00	0								
11	00	0								
12	00	0								
13	00	0								
14	00	0								
15	00	0								
16	00	0								
17	00	0								
18	00	0								
19	00	0								
20	00	0								
21	00	0								
22	00	0								

I 列に 0~f までの 16 進数を入力します。I6 に「'0」と半角で入力します。この「'」がポイントです。ただの「0」と入力すると数値になります。「'0」と先頭にアポストロフィを付けると、文字の「0」となります。数字ではありません。「'0」と入力して、エンタを押すと、「'0」とだけ表示されます。しかし、左詰めで表示されます。これが文字の「0」ですよという意味です。ちなみに B6 は右詰で「0」が表示されています。これは数字の「0」ですよ、という意味です。

	A	B	C	D	E	F	G	H	I	J
1										
2	data recording...									
3										
4	record_01 Data Out									
5	P7 data	dip sw data								
6	00	0							0	
7	00	0							1	
8	00	0							2	
9	00	0							3	
10	00	0							4	
11	00	0							5	
12	00	0							6	
13	00	0							7	
14	00	0							8	
15	00	0							9	
16	00	0							a	
17	00	0							b	
18	00	0							c	
19	00	0							d	
20	00	0							e	
21	00	0							f	
22	00	0								

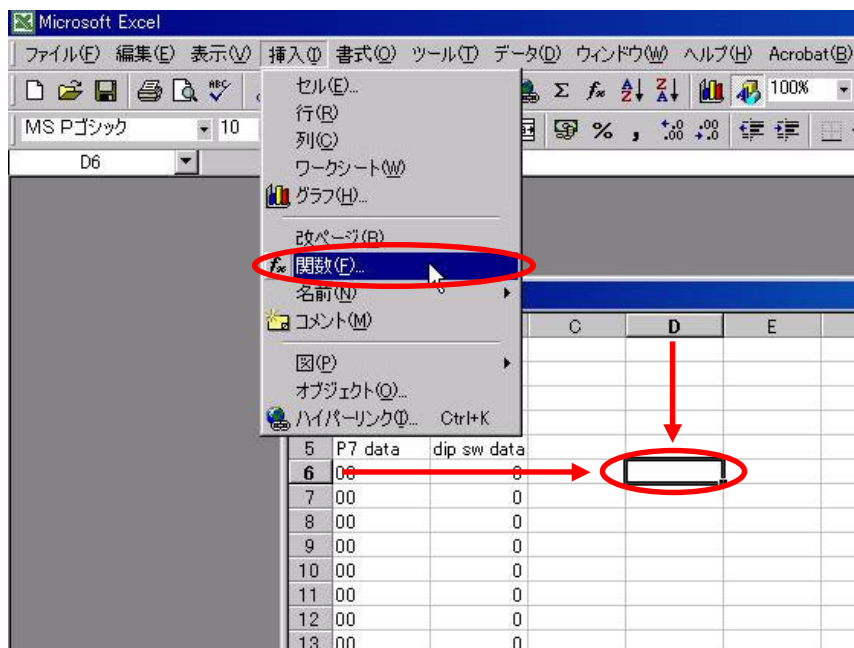
その下に「'1」と入力、エンタを押します。その下に「'2」...というように入力していきます。16 進数なので、「'9」の次は「'a」です。最後は「'f」です。

アルファベットは小文字で入力します。

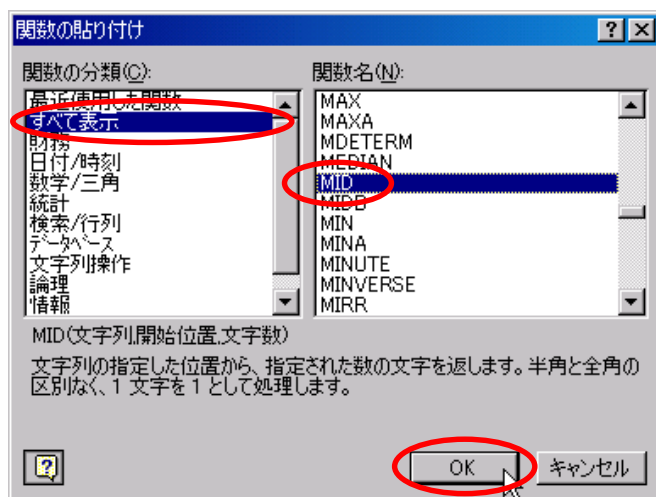
	A	B	C	D	E	F	G	H	I	J
1										
2	data recording...									
3										
4	record_01 Data Out									
5	P7 data	dip sw data								
6	00	0							0	●●●●
7	00	0							1	●●●○
8	00	0							2	●●○●
9	00	0							3	●●○○
10	00	0							4	●○●●
11	00	0							5	●○●○
12	00	0							6	●○○●
13	00	0							7	●○○○
14	00	0							8	○●●●
15	00	0							9	○●●○
16	00	0							a	○●○●
17	00	0							b	○●○○
18	00	0							c	○●●○
19	00	0							d	○●○○
20	00	0							e	○●●○
21	00	0							f	○●○○
22	00	0								

今打ち込んだ 16 進数の右側のセルに 2 進数を入力します。0 = 、1 = として入力します。0 なら「☐」と入力します。最後の f は「☐」となります。

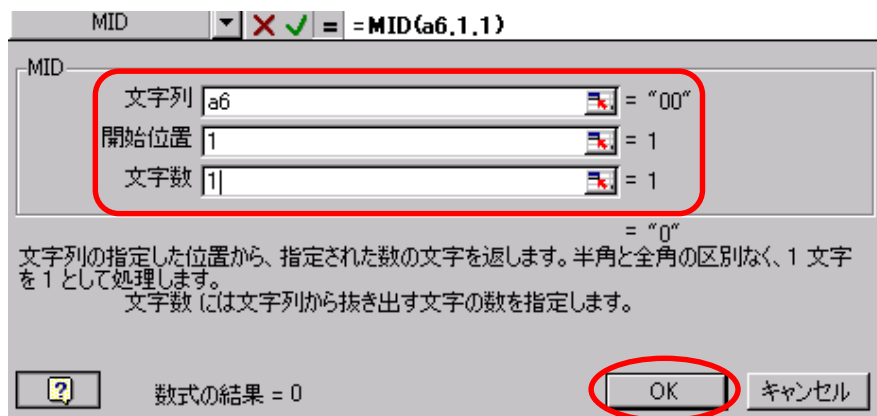
この I6 から J21 までが、16 進数 2 進数変換表になります。



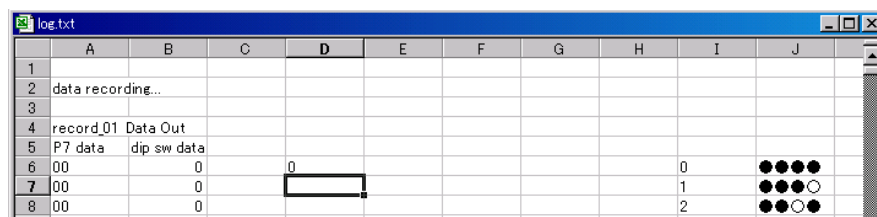
D6をクリックします。「挿入関数」を選択します。



関数の分類を「すべて表示」、関数名は「MID」を選択、OKをクリックします。



「文字列: a6」
「開始位置: 1」
「文字数: 1」
を入力、OKをクリックします。
これは、
セルA6にある文字の
左1文字目から
1文字
取り出さない
という意味です。



D6に「0」と表示されました。
セルA6の文字の
左1文字目から
1文字
取り出されています。

	A	B	C	D	E	F	G	H	I	J
1										
2	data recording...									
3										
4	record_01 Data Out									
5	P7 data	dip sw data								
6	00	0		0					0	
7	00	0							1	

次に E6 を選択します。
「挿入 関数」を選択します。
関数の分類を「すべて表示」、関数名は「MID」を選択、OK をクリックします。

MID

文字列: a6 = "00"

開始位置: 2 = 2

文字数: 1 = 1

数式の結果 = 0

OK キャンセル

「文字列: a6」
「開始位置: 2」
「文字数: 1」
を入力、OK をクリックします。

これは、
セル A6 の文字の
左 2 文字目から
1 文字
取り出さない
という意味です。

	A	B	C	D	E
1					
2	data recording...				
3					
4	record_01 Data Out				
5	P7 data	dip sw data			
6	00	0		0	0
7	00	0			

E6 に「0」と表示されました。
セル A6 の文字の
左 2 文字目から
1 文字
取り出されています。

関数の貼り付け

関数の分類(C): すべて表示

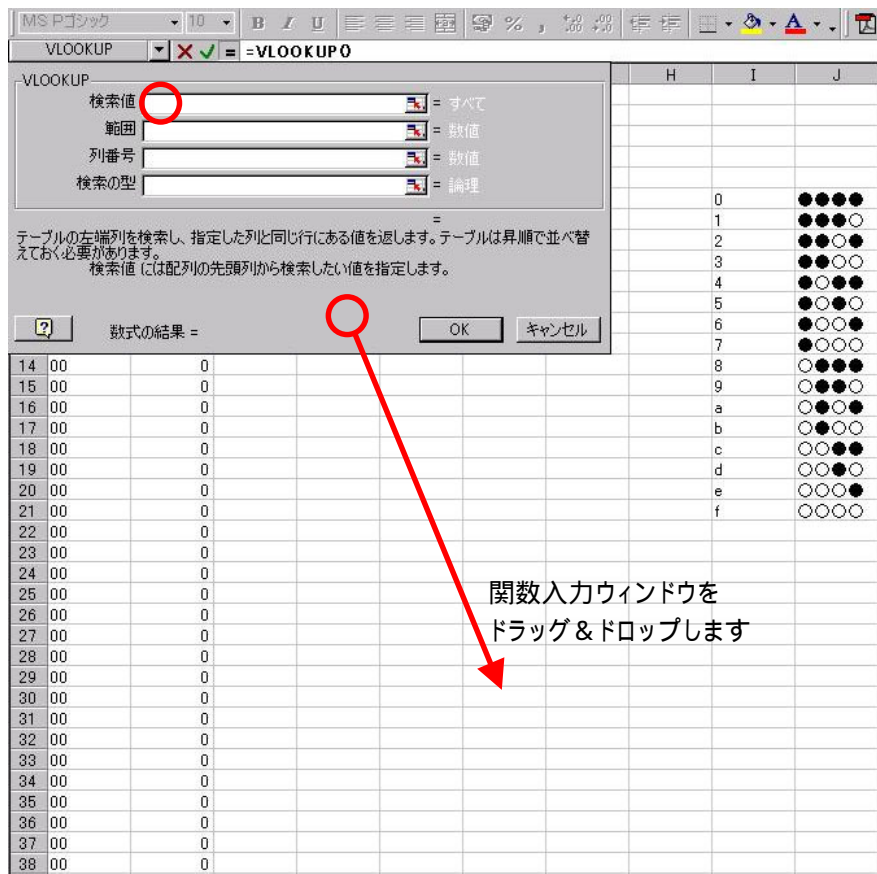
関数名(N): VLOOKUP

VLOOKUP(検索値,範囲,列番号,検索の型)

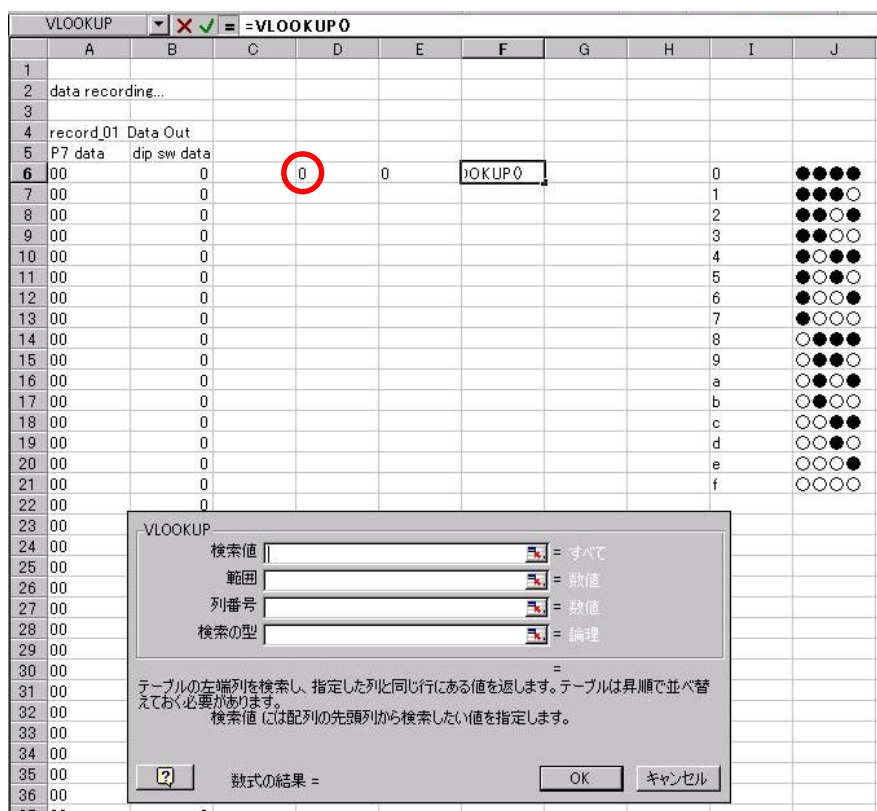
テーブルの左端列を検索し、指定した列と同じ行にある値を返します。テーブルは昇順で並べ替えておく必要があります。

OK キャンセル

次にセル F6 を選択します。
「挿入 関数」を選択します。
関数の分類を「すべて表示」、関数名は「VLOOKUP」を選択、OK をクリックします。



まず、検索値の空欄をクリックします。
関数入力ウィンドウをドラッグ
&ドロップで、下に移動します。



変換元である D6 をクリックします。

VLOOKUP

検索値 = "0"

範囲 = 数値

列番号 = 数値

検索の型 = 論理

=

テーブルの左端列を検索し、指定した列と同じ行にある値を返します。テーブルは昇順で並べ替えておく必要があります。
検索値 には配列の先頭列から検索したい値を指定します。

数式の結果 =

OK キャンセル

検索値の欄には
D6
と入力されます。

次に「範囲」の欄をクリックし
ます。 部分です。

log.txt

	A	B	C	D	E	F	G	H	I	J
1										
2	data recording...									
3										
4	record_01 Data Out									
5	P7 data	dip sw data								
6	00	0		0	0	P06:J21			0	●●●●●●●●
7	00	0							1	●●●●●●●●
8	00	0							2	●●●●●●●●
9	00	0							3	●●●●●●●●
10	00	0							4	●●●●●●●●
11	00	0							5	●●●●●●●●
12	00	0							6	●●●●●●●●
13	00	0							7	●●●●●●●●
14	00	0							8	●●●●●●●●
15	00	0							9	●●●●●●●●
16	00	0							a	●●●●●●●●
17	00	0							b	●●●●●●●●
18	00	0							c	●●●●●●●●
19	00	0							d	●●●●●●●●
20	00	0							e	●●●●●●●●
21	00	0								●●●●●●●●
22	00	0								●●●●●●●●

クリックしたまま
下げる

ここで離す

変換表全体を選択します。I
6の「0」をクリックします。クリ
ックしたまま、J21の「
」までマウスを持っていき、
離します。
(変換表の左上から右下)

VLOOKUP

検索値 = "0"

範囲 = {"0","●●●●●●●●","1"}

列番号 = 数値

検索の型 = 論理

=

テーブルの左端列を検索し、指定した列と同じ行にある値を返します。テーブルは昇順で並べ替えておく必要があります。
範囲 には検索したいデータが含まれる情報テーブルの範囲を指定します。

数式の結果 =

OK キャンセル

範囲の欄には
I6:J21
と入力されます。

VLOOKUP

検索値 = "0"

範囲 = {"0","●●●●●●●●","1"}

列番号 = 数値

検索の型 = 論理

=

テーブルの左端列を検索し、指定した列と同じ行にある値を返します。テーブルは昇順で並べ替えておく必要があります。
範囲 には検索したいデータが含まれる情報テーブルの範囲を指定します。

数式の結果 =

OK キャンセル

範囲欄を下記のように「\$」
を4箇所追加します。

\$I\$6:\$J\$21

「\$」は絶対参照という意
味です。検索エンジンで
「エクセル 絶対参照」
で調べるといろいろ出てきま
す。

VLOOKUP

検索値 = "0"

範囲 = {"0","●","●","●","●","1"}


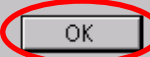
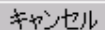
列番号 = 2

検索の型 = TRUE

= {"●","●","●","●","●"}

テーブルの左端列を検索し、指定した列と同じ行にある値を返します。テーブルは昇順で並べ替えておく必要があります。

検索の型 には <検索値> と完全に一致する値だけを検索するか、その近似値を含めて検索するかを、論理値で指定します。

 数式の結果 = ●●●●●  

「列番号:2」
「検索の型:1」
と入力します。

OKをクリックします。

log.txt

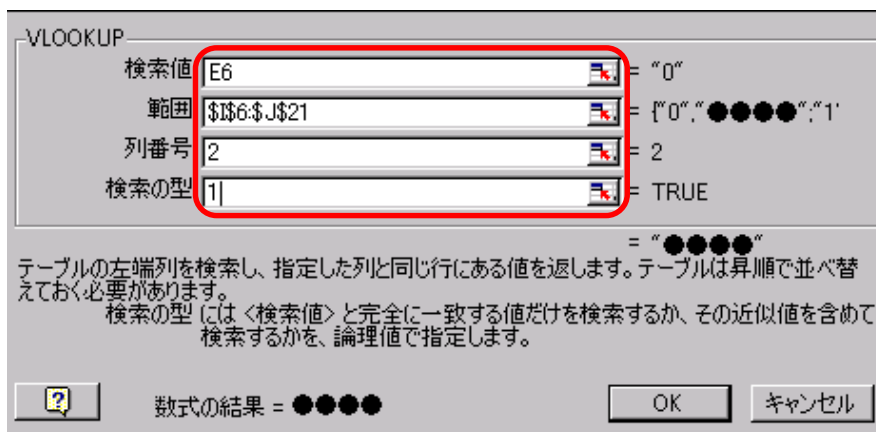
	A	B	C	D	E	F	G	H	I	J
1										
2	data recording...									
3										
4	record_01	Data Out								
5	P7 data	dip sw data								
6	00	0	0	0	0	●●●●●●			0	●●●●●●
7	00	0	0	0	0	●●●●●●			1	●●●●●●
8	00	0	0	0	0	●●●●●●			2	●●●●●●
9	00	0	0	0	0	●●●●●●			3	●●●●●●
10	00	0	0	0	0	●●●●●●			4	●●●●●●
11	00	0	0	0	0	●●●●●●			5	●●●●●●
12	00	0	0	0	0	●●●●●●			6	●●●●●●
13	00	0	0	0	0	●●●●●●			7	●●●●●●
14	00	0	0	0	0	●●●●●●			8	●●●●●●
15	00	0	0	0	0	●●●●●●			9	●●●●●●
16	00	0	0	0	0	●●●●●●			a	●●●●●●
17	00	0	0	0	0	●●●●●●			b	●●●●●●
18	00	0	0	0	0	●●●●●●			c	●●●●●●
19	00	0	0	0	0	●●●●●●			d	●●●●●●
20	00	0	0	0	0	●●●●●●			e	●●●●●●
21	00	0	0	0	0	●●●●●●			f	●●●●●●
22	00	0	0	0	0	●●●●●●				●●●●●●

セル F6 には、セル D6 に入力されている 16 進数を 2 進数に変換した値が表示されます。

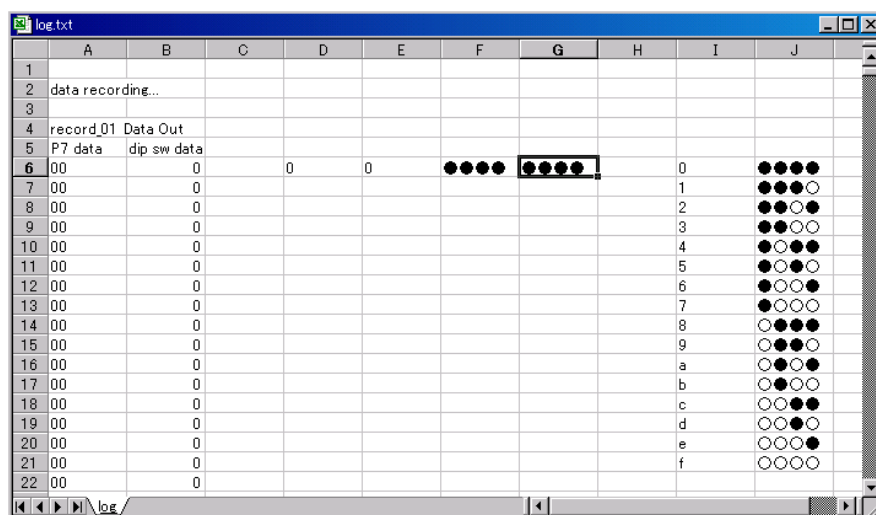
今回、D6 は「0」なので「
」です。

	A	B	C	D	E	F	G	H	I	J
1										
2	data recording...									
3										
4	record_01	Data Out								
5	P7 data	dip sw data								
6	00	0	0	0		●●●●			0	●●●●
7	00	0							1	●●●○
8	00	0							2	●●●●
9	00	0							3	●●●○
10	00	0							4	○●●○
11	00	0							5	●●●○
12	00	0							6	●●●●
13	00	0							7	●●○○
14	00	0							8	○●●●
15	00	0							9	○●●○
16	00	0							a	○●●●
17	00	0							b	○●●○
18	00	0							c	○○●●
19	00	0							d	○○●○
20	00	0							e	○○○●
21	00	0							f	○○○○
22	00	0								

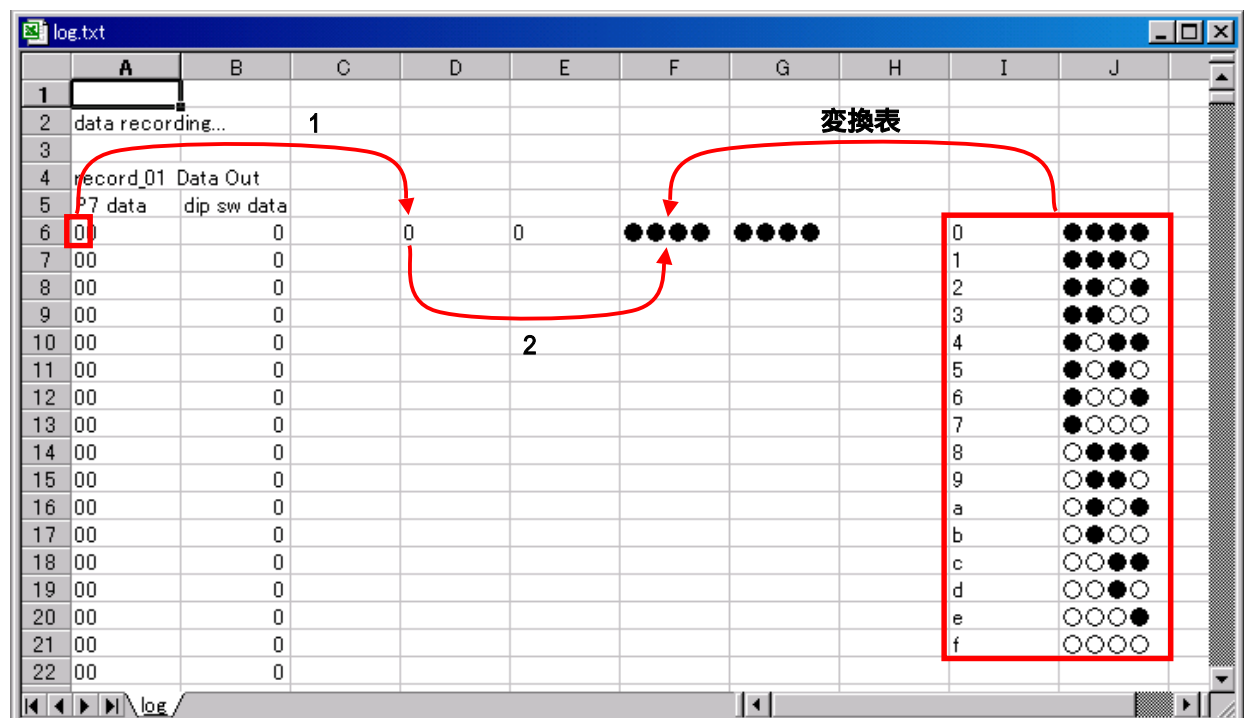
次にセル G6 を選択します。
「挿入 関数」を選択します。
関数の分類を「**すべて表示**」、関数名は「**VLOOKUP**」を選択、**OK**をクリックします。



「検索値: E6 (変換元)」
 「範囲: \$I\$6:\$J\$21」
 (変換表の左上から右下)
 「列番号: 2」
 「検索の型: 1」
 を入力します。OK をクリック
 します。



セル G 6 には、セル E 6 の値を
 16 進数に変換した値が表示
 されます。
 今回、E 6 は「0」なので「
 」です。



まとめると、

- 1 ...D6 は、A6 の 16 進数 2 桁のデータから、10 の位のみを取り出します。
- 2 ...F6 は、D6 と変換表を見比べます。変換表内の 16 に「0」があります。すぐ右の「
 」を取り出して表示
 します。

	A	B	C	D	E	F	G	H	I	J
1										
2	data recording...		1							
3										
4	record_01	Data Out								
5	P7 data	dip sw data								
6	0	0	0	0	0000	0000				
7	00	0								
8	00	0								
9	00	0								
10	00	0								
11	00	0								
12	00	0								
13	00	0								
14	00	0								
15	00	0								
16	00	0								
17	00	0								
18	00	0								
19	00	0								
20	00	0								
21	00	0								
22	00	0								

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●
1	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●
2	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●
3	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●
4	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●
5	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●
6	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●
7	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●
8	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●
9	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●
a	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●
b	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●
c	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●
d	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●
e	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●
f	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●	●●●●

- 1 ...E6 は、A6 の16進数2桁のデータから、1の位のみを取り出します。
- 2 ...G6 は、E6 と変換表を見比べます。変換表内の16に「0」があります。すぐ右の「」を取り出して表示します。

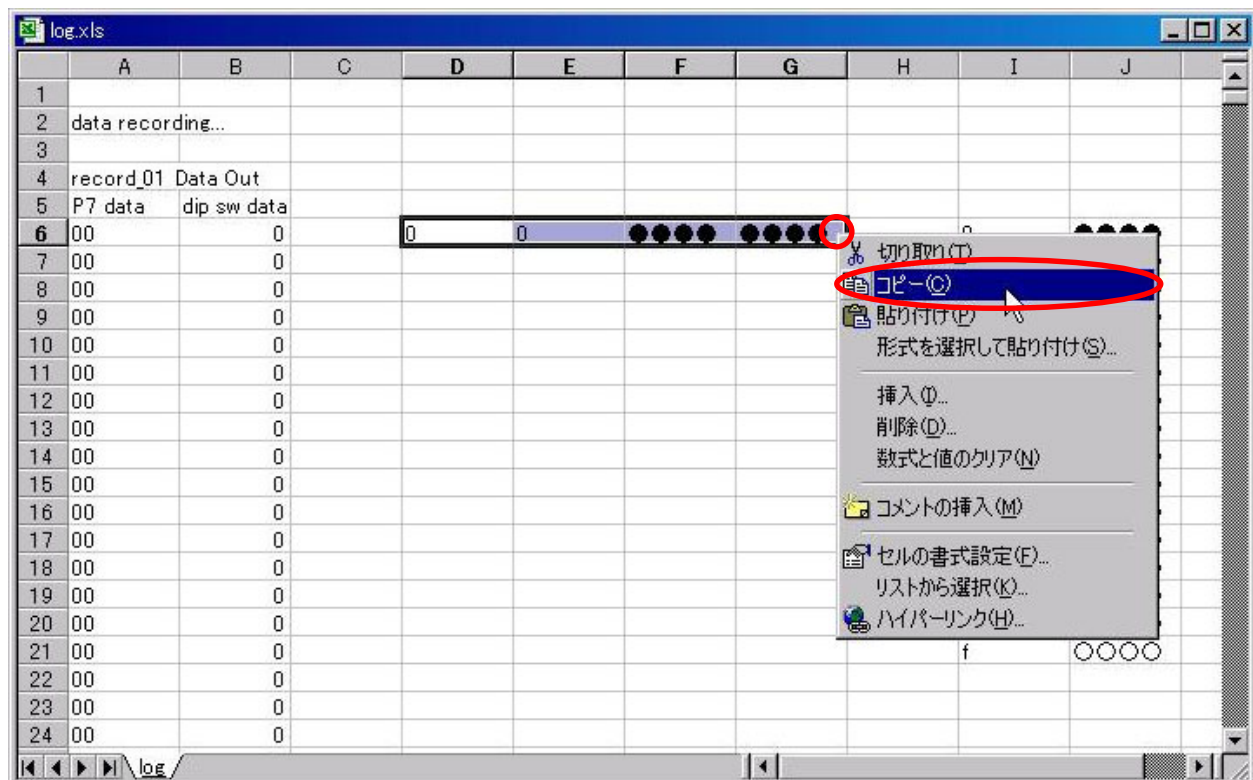
	A	B	C	D	E	F	G
1							
2	data recording...						
3							
4	record_01	Data Out					
5	P7 data	dip sw data					
6	00	0	0	0	0	0000 0000	
7	00	0					
8	00	0					
9	00	0					
10	00	0					
11	00	0					
12	00	0					

結果、あたかも、セルA6にあるの16進数2桁「00」が、セルF6とG6に2進数「0000 0000」で変換されているようになります。

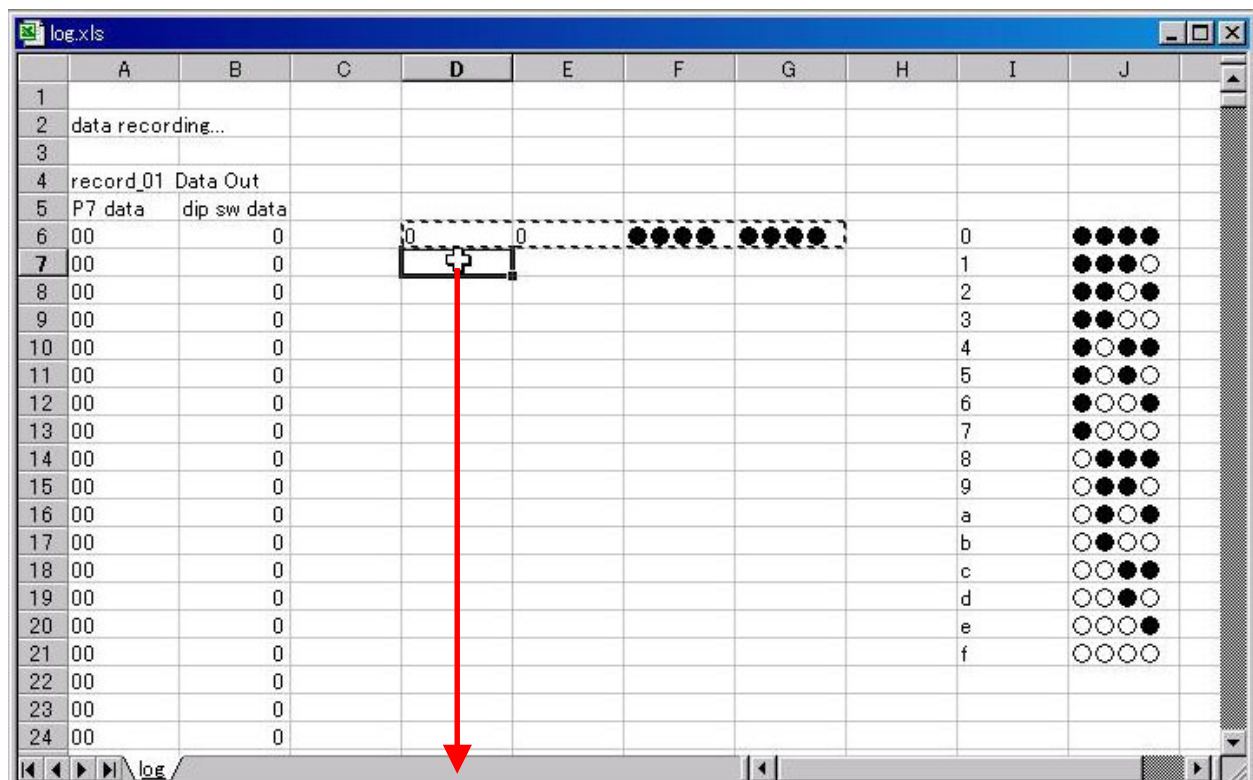
本当は、セル A6 から直接 2 進数に変換すれば良いのですが、このような関数がありません。そのため、段階的に変換して、最後に表示したい形式にします。

変換表も「0 0」～「f f」まで 256 通り記入すれば良いのですが、入力が大変です。そのため、16 進数を 1 桁ずつに分けて「0」～「f」までの 16 通りを入力、変換しています。

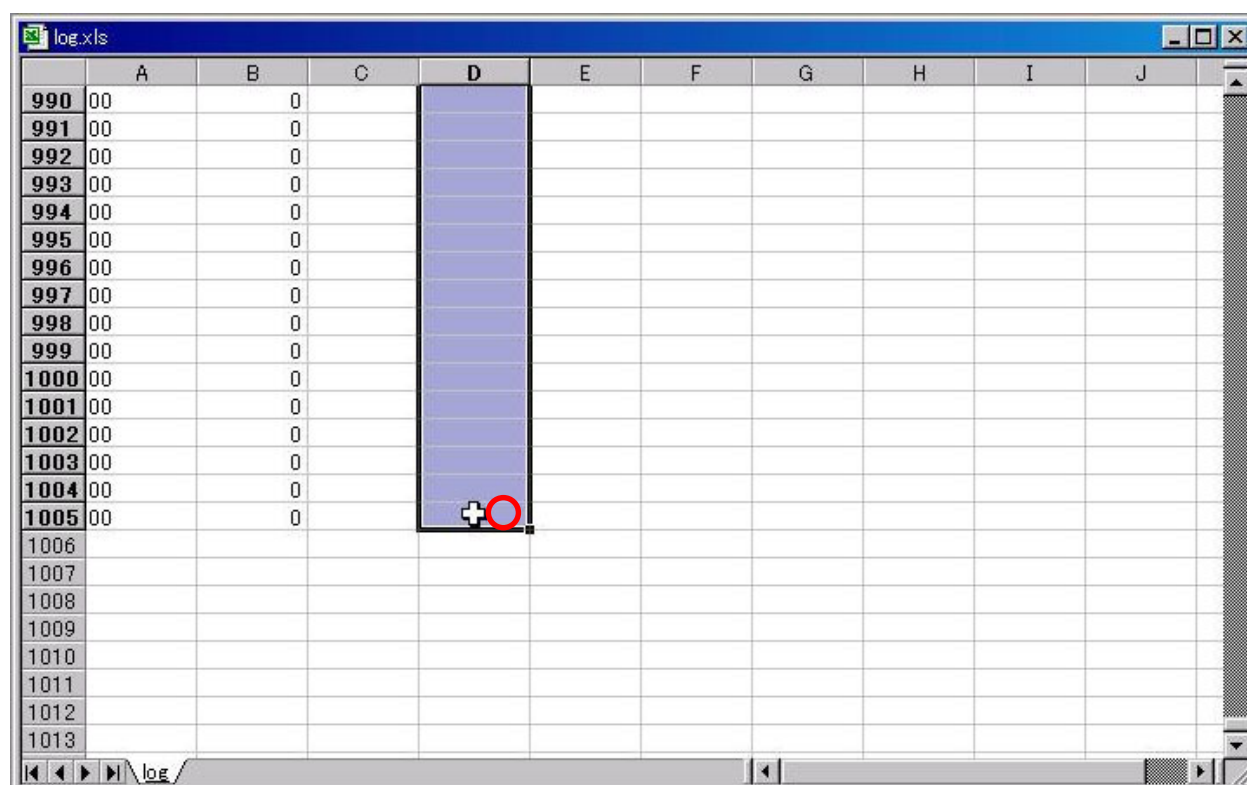
今回、6 行目のデータを変換しました。6 行目より下には 999 個分のデータが続きます。同様に 999 回繰り返します・・・ そんな時間はありません！コピー作業を使えば、1 回の作業で実現できます。



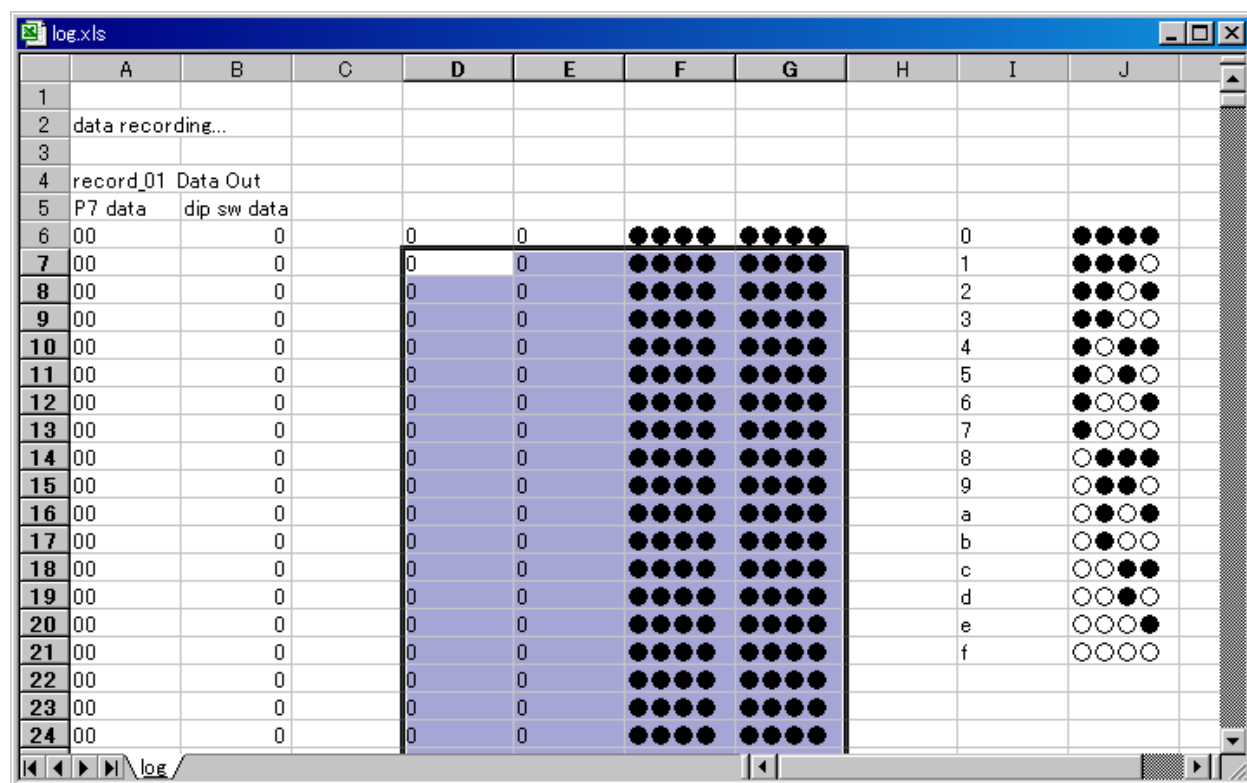
D6～G6のセルを選択して、 当たりで右クリック、「コピー」を選択します。



セル D7 をクリック、そのまま下に降ろしていきます。画面をはみ出してをクリックしたままにしておきます。自動的に下へスクロールします。



最後の行までマウスを左クリックしたまま下に持ってきます。下へ行きすぎたら、上にカーソルを上げると、戻ります(マウスの左ボタンは押したままです)。D1005 まで来たら、左クリックを止めます。 部分で右クリック、「貼り付け」を選択します。



全行に貼り付けられました。

	A	B	C	D	E	F	G	H	I	J
571	80	0		8	0	○●●●	●●●●			
572	80	0		8	0	○●●●	●●●●			
573	80	0		8	0	○●●●	●●●●			
574	80	0		8	0	○●●●	●●●●			
575	80	0		8	0	○●●●	●●●●			
576	80	0		8	0	○●●●	●●●●			
577	80	0		8	0	○●●●	●●●●			
578	80	0		8	0	○●●●	●●●●			
579	80	0		8	0	○●●●	●●●●			
580	80	0		8	0	○●●●	●●●●			
581	80	0		8	0	○●●●	●●●●			
582	80	0		8	0	○●●●	●●●●			
583	c0	0		c	0	○●●●	●●●●			
584	c0	0		c	0	○●●●	●●●●			
585	c0	0		c	0	○●●●	●●●●			
586	c0	0		c	0	○●●●	●●●●			
587	c0	0		c	0	○●●●	●●●●			
588	c0	0		c	0	○●●●	●●●●			
589	c0	0		c	0	○●●●	●●●●			
590	c0	0		c	0	○●●●	●●●●			
591	c0	0		c	0	○●●●	●●●●			
592	c0	0		c	0	○●●●	●●●●			
593	c0	0		c	0	○●●●	●●●●			
594	c0	0		c	0	○●●●	●●●●			

例えば、582 行は 16 進数「80」、2 進数「
」、583 行は、16 進数「C0」、2 進数「

」と変換されています。きちんと変換されていることが分かります。

☐ キーを押し続けます。 と が、あたかも実際に反応して動いているように表示されます。パラパラアニメのようです。マイコンカーのセンサの値をこの方法で変換、解析してみるとセンサの反応状態が分かります。もしかしたら、自分が予想していないセンサ状態があるかもしれません。いろいろ解析してみると良いでしょう。

4. プロジェクト「record_02」 外付け EEP-ROM にデータ保存

4.1 概要

このプログラムは、

- ・ポート7に接続されているディップスイッチの値
- ・CPU ボードのディップスイッチの値

を、10ms ごとに**外付け EEP-ROM** に保存します。保存時間は 10 秒です。保存時間は自由に変わりますが上限はメモリがいっぱいになるまでです(今回は約 164 秒、1 回に保存するデータ数で変わります)。保存後、RS232C を通してパソコンへ保存した情報を出力します。

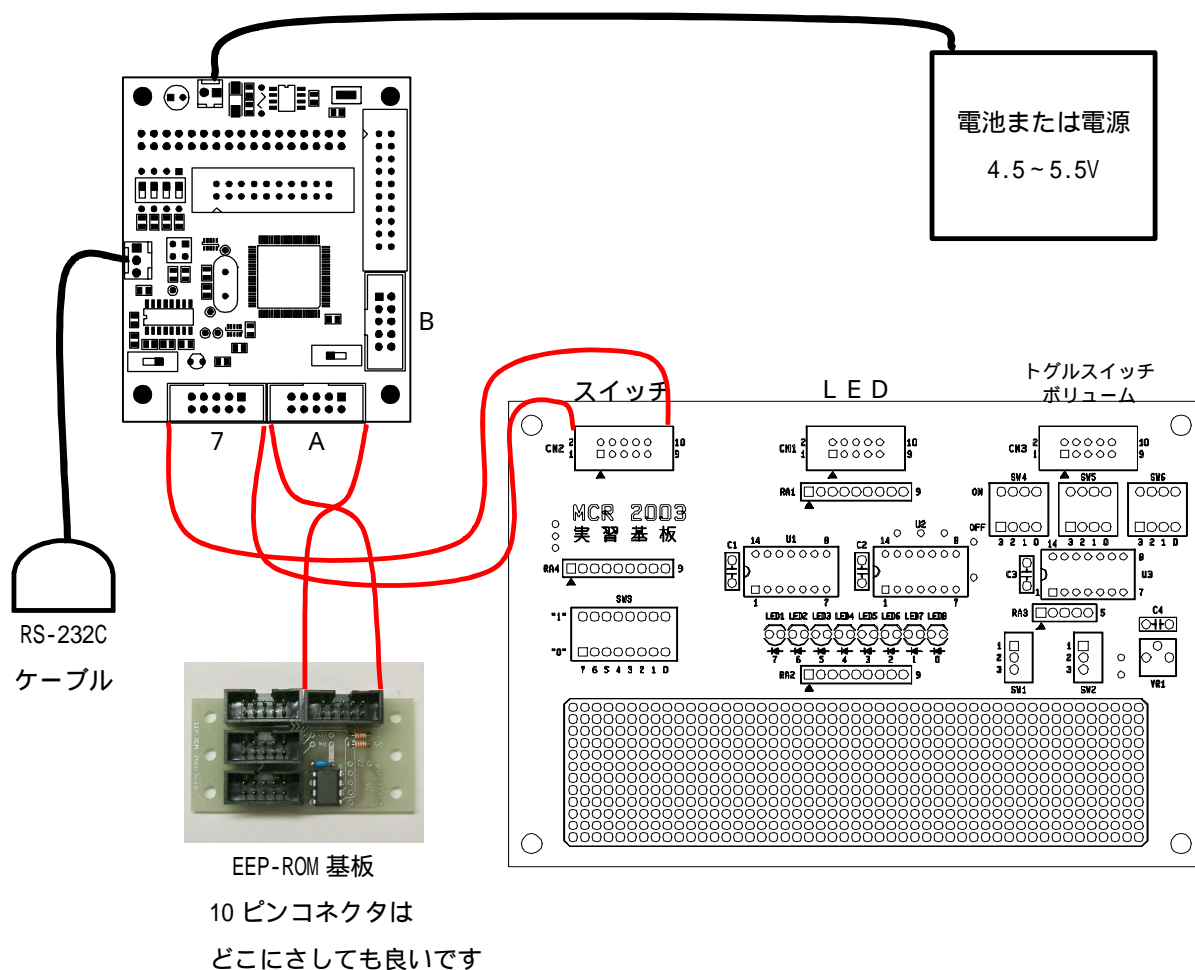
ここでは、

- ・前章と比べて、EEP-ROM を使った場合の変更点について、説明します。

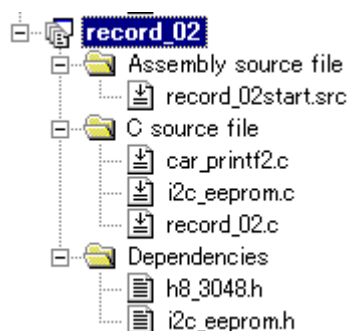
4.2 接続

- ・CPU ボードのポート7と、実習基板のスイッチ部をフラットケーブルで接続します。
- ・CPU ボードのポートAと、EEP-ROM 基板をフラットケーブルで接続します。

ポート7のディップスイッチをセンサ基板に変えると、センサの反応を記録することができます。



4.3 プロジェクトの構成



· record_02start.src

· record_02.c

· car_printf2.c

· i2c_eeprom.c

の 4 ファイルあります。

h8_3048.h は record_02.c、car_printf2.c、i2c_eeprom.c でインクルードされているファイルです。

i2c_eeprom.h は record_02.c、i2c_eeprom.c でインクルードされているファイルです。

4.4 プログラム

```

1 :  /*****
2 :  /* 外付けEEP-ROMデータ保存演習プログラム「record_02.c」
3 :  /* 2006.04 ジャパンマイコンカーラリー実行委員会 */
4 :  /*****
5 :  /*
6 :  本プログラムは外付けEEP-ROM(24C256 32KB)にポート 7 のデータ、
7 :  CPUボード上のディップスイッチの値を10msごとに保存します。
8 :  その後、保存したデータをパソコンへ転送します。
9 :  */
10 :
11 :  /*=====*/
12 :  /* インクルード */
13 :  /*=====*/
14 :  #include <no_float.h> /* stdioの簡略化 最初に置く */
15 :  #include <stdio.h>
16 :  #include <machine.h>
17 :  #include "h8_3048.h"
18 :  #include "i2c_eeprom.h" /* EEP-ROM追加(データ記録) */
19 :
20 :  /*=====*/
21 :  /* シンボル定義 */
22 :  /*=====*/
23 :
24 :  /*=====*/
25 :  /* プロトタイプ宣言 */
26 :  /*=====*/
27 :  void init( void );
28 :  unsigned char dipsw_get( void );
29 :
30 :  /*=====*/
31 :  /* グローバル変数の宣言 */
32 :  /*=====*/
33 :  unsigned long cnt1; /* main内で使用 */
34 :  int pattern; /* パターン番号 */
35 :
36 :  /* データ保存関連 */
37 :  int iTimer10; /* 取得間隔計算用 */
38 :  int saveIndex; /* 保存インデックス */
39 :  int saveSendIndex; /* 送信インデックス */
40 :  int saveFlag; /* 保存フラグ */
41 :  char saveData[8]; /* 一時保存エリア */
42 :
43 :  /*****
44 :  /* メインプログラム */
45 :  /*****
46 :  void main( void )
47 :  {
48 :      /* マイコン機能の初期化 */
49 :      init(); /* 初期化 */
50 :      initI2CEeprom( &PADDR, &PADR, 0x5f, 7, 5); /* EEP-ROM初期設定 */
51 :      init_sci1( 0x00, 79 ); /* SCI1初期化 */
52 :      set_ccr( 0x00 ); /* 全体割り込み許可 */
53 :
54 :      while( 1 ) {
55 :
56 :          I2CEepromProcess(); /* I2C EEP-ROM保存処理 */
57 :
58 :          switch( pattern ) {
59 :          case 0:
60 :              /* EEP-ROMクリア */
61 :              clearI2CEeprom(); /* 数秒かかる */

```

データ解析実習マニュアル kit06 版

```

62 :     printf( "%n" );
63 :     printf( "data recording...%n" );
64 :     pattern = 1;
65 :     saveIndex = 0;
66 :     saveFlag = 1;          /* データ保存開始          */
67 :     cnt1 = 0;
68 :     break;
69 :
70 : case 1:
71 :     /* データ保存中 保存自体は割り込みの中で行う */
72 :     if( cnt1 >= 10000 ) {
73 :         saveFlag = 0;      /* 保存終了          */
74 :         pattern = 2;      /* データ転送処理へ  */
75 :     }
76 :     break;
77 :
78 : case 2:
79 :     /* タイトル転送、準備 */
80 :     while( !checkI2CEeprom() ); /* 最後のデータ書き込むまで待つ */
81 :
82 :     printf( "%n" );
83 :     printf( "record_02 Data Out%n" );
84 :     printf( "P7 data,dip sw data%n" );
85 :
86 :     saveSendIndex = 0;
87 :     pattern = 3;
88 :     break;
89 :
90 : case 3:
91 :     /* データ転送 */
92 :     printf( "%02x,%02x%n",
93 :         (unsigned char)readI2CEeprom( saveSendIndex+0 ),
94 :         (unsigned char)readI2CEeprom( saveSendIndex+1 ) );
95 :     saveSendIndex += 2;
96 :     if( saveIndex <= saveSendIndex ) {
97 :         pattern = 4;
98 :         cnt1 = 0;
99 :     }
100 :    break;
101 :
102 : case 4:
103 :     /* 転送終了 */
104 :     break;
105 :
106 : default:
107 :     /* どれでもない場合は待機状態に戻す */
108 :     pattern = 0;
109 :     break;
110 : }
111 : }
112 : }
113 :
114 : /*****
115 : /* H8/3048F-ONE 内蔵周辺機能 初期化          */
116 : /*****
117 : void init( void )
118 : {
119 :     /* I/Oポートの入出力設定 */
120 :     P1DDR = 0xff;
121 :     P2DDR = 0xff;
122 :     P3DDR = 0xff;
123 :     P4DDR = 0xff;
124 :     P5DDR = 0xff;
125 :     P6DDR = 0xf0;          /* CPU基板上のDIP SW          */
126 :     P8DDR = 0xff;
127 :     P9DDR = 0xf7;          /* 通信ポート                  */
128 :     PADDR = 0x5f;          /* EEP-ROM基板                */
129 :     PBDDR = 0xff;
130 :     /* ポート7は、入力専用なので入出力設定はありません */
131 :
132 :     /* ITU0 1msごとの割り込み */
133 :     ITU0_TCR = 0x20;
134 :     ITU0_GRA = 24575;
135 :     ITU0_IER = 0x01;
136 :
137 :     /* ITUのカウンタスタート */
138 :     ITU_STR = 0x01;
139 : }
140 :
141 : /*****
142 : /* ITU0 割り込み処理          */
143 : /*****
144 : #pragma interrupt( interrupt_timer0 )
145 : void interrupt_timer0( void )
146 : {
147 :     ITU0_TSR &= 0xfe;      /* フラグクリア          */
148 :     cnt1++;
149 :
150 :     /* データ保存関連 */
151 :     iTimer10++;
152 :     if( iTimer10 >= 10 ) {

```

```

153 :         iTimer10 = 0;
154 :         if( saveFlag ) {
155 :             saveData[0] = P7DR;
156 :             saveData[1] = dipsw_get();
157 :             setPageWrite12CEeprom( saveIndex, 2, saveData );
158 :             saveIndex += 2;
159 :             if( saveIndex >= 0x8000 ) saveFlag = 0;
160 :         }
161 :     }
162 : }
163 :
164 : /*****
165 :  /* ディップスイッチ値読み込み */
166 :  /* 戻り値 スイッチ値 0~15 */
167 :  *****/
168 : unsigned char dipsw_get( void )
169 : {
170 :     unsigned char sw;
171 :
172 :     sw = P6DR;                /* ディップスイッチ読み込み */
173 :     sw &= 0x0f;
174 :
175 :     return sw;
176 : }
177 :
178 : /*****
179 :  /* end of file */
180 :  *****/

```

4.5. プログラムの解説

4.5.1 内蔵周辺機能の初期化

```

114 :  /*****
115 :  /* H8/3048F-ONE 内蔵周辺機能 初期化
116 :  *****/
117 :  void init( void )
118 :  {
119 :      /* I/Oポートの入出力設定 */
120 :      P1DDR = 0xff;
121 :      P2DDR = 0xff;
122 :      P3DDR = 0xff;
123 :      P4DDR = 0xff;
124 :      P5DDR = 0xff;
125 :      P6DDR = 0xf0;          /* CPU基板上のDIP SW
126 :      P8DDR = 0xff;
127 :      P9DDR = 0xf7;          /* 通信ポート
128 :      PADDR = 0x5f;          /* EEP-ROM基板
129 :      PBDDR = 0xff;
130 :      /* ポート7は、入力専用なので入出力設定はありません
131 :
132 :      /* ITU0 1msごとの割り込み */
133 :      ITU0_TCR = 0x20;
134 :      ITU0_GRA = 24575;
135 :      ITU0_IER = 0x01;
136 :
137 :      /* ITUのカウントスタート */
138 :      ITU_STR = 0x01;
139 :  }
```

ポートAは、EEP-ROMを接続します。EEP-ROMの接続端子は入力設定にします。詳しくは、14ページ「1.11 EEP-ROMの接続端子を変える場合の設定」を参照してください。今回は、0x5fとなります。

ITU0は1msごとの割り込み設定を行います。

4.5.2 データ保存関連の変数

```

36 :  /* データ保存関連 */
37 :  int      iTimer10;          /* 取得間隔計算用
38 :  int      saveIndex;         /* 保存インデックス
39 :  int      saveSendIndex;     /* 送信インデックス
40 :  int      saveFlag;          /* 保存フラグ
41 :  char     saveData[8];       /* 一時保存エリア
```

37～40行の変数は、プロジェクト「record_01」と同じです。

41行のsaveData配列は、EEP-ROMに保存するデータを一時的に格納する変数です。

saveData[0]= ポート7のデータ	saveData[1]= ディップスイッチ値	saveData[2]= 未使用
saveData[3]= 未使用	saveData[4]= 未使用	saveData[5]= 未使用
saveData[6]= 未使用	saveData[7]= 未使用	

のそれぞれのデータを格納します。

4.5.3 初期設定

```

46 : void main( void )
47 : {
48 :     /* マイコン機能の初期化 */
49 :     init();                               /* 初期化 */
50 :     initI2CEeprom( &PADDR, &PADR, 0x5f, 7, 5); /* EEP-ROM初期設定 */
51 :     init_sci1( 0x00, 79 );                /* SCI1初期化 */
52 :     set_ccr( 0x00 );                      /* 全体割り込み許可 */

```

EEP-ROM 関連の初期設定を行います。この関数は、EEP-ROM の内容をクリアするのではなく、EEP-ROM を使用するための準備をする関数です。

EEP-ROM はポート A に接続、ポート A の入出力設定は 0x5f、EEP-ROM の SCL 端子は bit7、EEP-ROM の SDA 端子は bit5 という設定です。

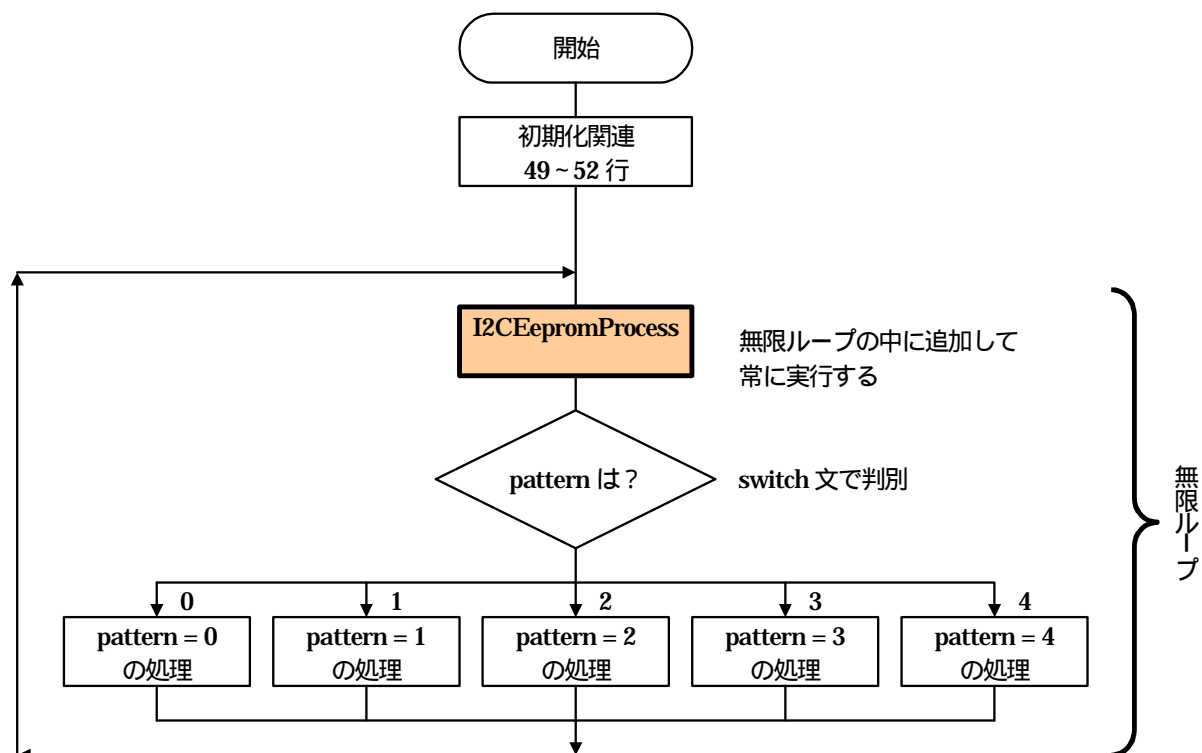
4.5.4 メイン部の全体

```

54 : while( 1 ) {
55 :
56 :     I2CEepromProcess();                  /* I2C EEP-ROM保存処理 */
57 :
58 :     switch( pattern ) {
59 :     case 0:
60 :         EEP-ROMの内容クリア、保存開始、パターン 1 へ
61 :         break;
62 :
63 :     case 1:
64 :         10秒たったなら保存終了、パターン 2 へ
65 :         割り込みプログラムで10msごとにデータを取得、EEP-ROMへ保存する準備を行う
66 :         break;
67 :
68 :     case 2:
69 :         タイトルの送信
70 :         break;
71 :
72 :     case 3:
73 :         データ送信
74 :         break;
75 :
76 :     case 4:
77 :         /* 転送終了 */
78 :         break;
79 :
80 :     default:
81 :         /* どれでもない場合は待機状態に戻す */
82 :         pattern = 0;
83 :         break;
84 :     }
85 : }

```

56 行で、I2CEepromProcess 関数をループの中に置いて、常に行われるようにしています。イメージとしては下記ようになります。



EEP-ROM への書き込み作業は、かなり時間がかかります。そのため、それだけにかかり続けると他の処理ができなくなり問題となることがあります。マイコンカーの場合は、センサのチェック漏れやモータ制御の遅れが
おこり、脱輪してしまうこともあるかもしれません。

そこで、setPageWriteI2CEeprom 関数と I2CEepromProcess 関数を使用します。この関数は

- ・setPageWriteI2CEeprom 関数で書き込む準備をします (すぐに終わります)。10ms ごとに割り込み内で行います。後述します。
- ・I2CEepromProcess 関数で少しずつ書き込んで、1 回の処理を短時間で終わるようにしています。

I2CEepromProcess 関数は、ループ内に入れて、常に行われるようにします。

4.5.5 パターン 0: データ保存の前準備

```

59 :      case 0:
60 :          /* EEP-ROMクリア */
61 :          clearI2CEeprom();          /* 数秒かかる          */
62 :          printf( "%n" );
63 :          printf( "data recording...%n" );
64 :          pattern = 1;
65 :          saveIndex = 0;
66 :          saveFlag = 1;          /* データ保存開始          */
67 :          cnt1 = 0;
68 :          break;

```

61 行...EEP-ROM のクリアを行います。数秒かかります。

62～63 行...データを記録する旨、メッセージを出力します。

64 行...次に実行するときはパターン 1 になるよう、変数の設定をします。

65 行...保存する配列の番号用の変数を初期化します。

66 行...データの保存を開始します。10ms ごとに割り込み内でポート 7 の値と、ディップスイッチの値を保存します。

67 行...時間計測用の cnt1 変数をクリアします。

4.5.6 パターン 1: データ保存中の処理

```

70 :      case 1:
71 :          /* データ保存中 保存自体は割り込みの中で行う */
72 :          if( cnt1 >= 10000 ) {
73 :              saveFlag = 0;          /* 保存終了          */
74 :              pattern = 2;          /* データ転送処理へ          */
75 :          }
76 :          break;

```

データの保存自体は、割り込み内で行っています。

72 行...10000ms たったかチェックして、たったなら保存を終了します。

74 行...パターンを 2 に移します。

EEP-ROM は 32KB (32768 バイト) あります。10ms ごとに 2 バイト分のデータを保存するので、
 $32768 \times 0.01 \text{ 秒} / 1 \text{ 回の保存数 } 2 \text{ バイト} = 163.84 \text{ 秒} = 2 \text{ 分 } 43 \text{ 秒 } 84$
 の時間だけ、保存することができます。保存時間を変えたい場合は、71 行の 10000 を変えてください。ただし、上限は 163.84 秒 (163840ms) です。

4.5.7 パターン 2:タイトル転送、準備

```

78 :      case 2:
79 :          /* タイトル転送、準備 */
80 :          while( !checkI2CEeprom() );      /* 最後のデータ書き込むまで待つ*/
81 :
82 :          printf( "¥n" );
83 :          printf( "record_02 Data Out¥n" );
84 :          printf( "P7 data,dip sw data¥n" );
85 :
86 :          saveSendIndex = 0;
87 :          pattern = 3;
88 :          break;

```

80 行...最後の書き込みが終わるまで待ちます。

82～84 行...タイトルを送信します。

86 行...送信位置をクリアしています。

87 行...パターンを 3 に移します。

4.5.8 パターン 3:データ転送

```

90 :      case 3:
91 :          /* データ転送 */
92 :          printf( "%02x,%d¥n",
93 :              (unsigned char)readI2CEeprom( saveSendIndex+0 ),
94 :              (unsigned char)readI2CEeprom( saveSendIndex+1 ) );
95 :          saveSendIndex += 2;
96 :          if( saveIndex <= saveSendIndex ) {
97 :              pattern = 4;
98 :              cnt1 = 0;
99 :          }
100 :          break;

```

データを転送します。

送信フォーマットは、92 行の「%02x,%d¥n」です。まず、

「%02x」... 2 桁に満たない場合は 0 で埋めて 16 進数表記

「,」..... そのまま表示されます。

「%d」..... 10 進数表記

「¥n」..... 改行

となります。

EEP-ROM からデータを読み込む関数は、

```
readI2CEeprom( EEP-ROMから読み込みたいアドレス );
```

です。saveSendIndex が読み込むアドレスになります。

```

printf( "%02x,%d¥n",
    (unsigned char)readI2CEeprom( saveSendIndex+0 ),
    (unsigned char)readI2CEeprom( saveSendIndex+1 ) );

```

ポート7のデータ読み込み
ディップスイッチのデータ読み込み

データを読み込みながら、printf 文でデータを送信します。因みに、readI2CEeprom 関数の戻り値は、char 型 (-128 ~ 127) です。出力値は 16 進数なので、unsigned char 型に型変換しています。

```
saveSendIndex += 2;
```

でアドレスを+2 して、次に読み込むアドレスにします。

96 行で、保存数より送信しようとしているアドレスの方が大きくなったら、送信終了と見なしてパターン 4 へ移ります。

4.5.9 パターン 4: 転送終了

```
102 :      case 4:
103 :          /* 転送終了 */
104 :          break;
```

終わりです。何もしません。

4.5.10 割り込みプログラム

```
141 :  /* **** */
142 :  /* ITU0 割り込み処理 */
143 :  /* **** */
144 :  #pragma interrupt( interrupt_timer0 )
145 :  void interrupt_timer0( void )
146 :  {
147 :      ITU0_TSR &= 0xfe;          /* フラグクリア */
148 :      cnt1++;
149 :
150 :      /* データ保存関連 */
151 :      iTimer10++;
152 :      if( iTimer10 >= 10 ) {
153 :          iTimer10 = 0;
154 :          if( saveFlag ) {
155 :              saveData[0] = P7DR;
156 :              saveData[1] = dipsw_get();
157 :              setPageWriteI2CEeprom( saveIndex, 2, saveData );
158 :              saveIndex += 2;
159 :              if( saveIndex >= 0x8000 ) saveFlag = 0;
160 :          }
161 :      }
162 :  }
```

割り込みプログラムは、1ms ごとに実行されます。データの保存は、10ms ごとなので 151 行の iTimer 変数で 10 回数えます。10 なら 10ms たったと判断して、EEP-ROM にデータを書き込みます。手順は、

- ・保存したいデータを saveData 配列にセットします (155 ~ 156 行)。
- ・setPageWriteI2CEeprom 関数を実行します (157 行)。

setPageWritel2CEeprom 関数は、EEP-ROM に書き込む**準備をするだけ**です。この関数の引数は、

```
setPageWritel2CEeprom( saveIndex, 2, saveData );
```

1 2 3

- 1...EEP-ROM の何番地に保存するか指定します。
- 2...保存するデータ数を指定します。
- 3...保存データの格納されている配列を指定します。

実際の書き込みは、I2CEepromProcess 関数で行っています。

158 行で、保存するアドレスを+2 して、次に保存するアドレスをセットしています。

159 行で、アドレスの上限である 0x8000 を超えていないかチェック、超えていたらデータ保存を終了します。

4.5.11 int 型の値を保存する場合

EEP-ROM に保存できるデータは、char 型(8bit 幅、-128 ~ 127、または 0 ~ 255)です。そのため、int 型(16bit 幅)を保存する場合は、下記のように上位 8bit と下位 8bit に分けて保存します。

```
i = int 型のデータ
saveData[0] = i >> 8;          /* 上位 8bit 保存 */
saveData[1] = i & 0xff;        /* 下位 8bit 保存 */
```

呼び出すときは、

```
i = (int)((unsigned char)readI2CEeprom( アドレス )*0x100 +
          (unsigned char)readI2CEeprom( アドレス+1 ) );
```

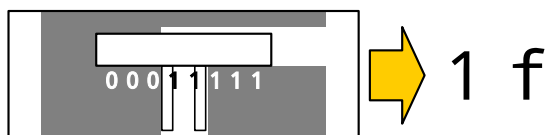
とします。

5. プロジェクト「record_03」 外付け EEP-ROM にデータ保存(2 進数変換)

5.1 概要

このプログラムは、プロジェクト「record_02」を一部改造した内容です。

今までのプログラムは、保存したデータを出力するとき、16 進数や 10 進数でした。例えば、ポート 7 に接続されているセンサ基板の情報を記録し、16 進数でパソコンへ出力したとき、下記ようになります。



printf 文が出力する数値の形式は、8 進数、10 進数、16 進数しかありません。そのため、センサ情報は"0"か"1"かの 2 進数情報ですが、いちばん分かりやすい 16 進数で出力していました。といっても、元々 2 進数なので、やはり 2 進数で出力した方が分かりやすいです。プロジェクト「record_01」では、エクセルを使った 16 進数 2 進数変換方法を紹介しました。

23	18	0	34	34	1	8	●●●○	○●●●
23	18	0	34	34	1	8	●●●○	○●●●
23	18	0	34	34	1	8	●●●○	○●●●
23	18	0	34	34	1	8	●●●○	○●●●
41	1f	38	42	8	1	f	●●●○	○●●○
41	1f	38	42	8	1	f	●●●○	○●●○
41	1f	38	42	8	1	f	●●●○	○●●○
41	1f	38	42	8	1	f	●●●○	○●●○
41	3f	38	42	8	3	f	●●●○	○●●○
41	1f	38	42	8	1	f	●●●○	○●●○
41	07	38	42	8	0	7	●●●○	○●●○
41	00	38	42	8	0	0	●●●○	○●●○

エクセルの関数を使って変換

この方法なら、センサの状態が分かりますが、転送するたびにエクセルの関数を使って変換するのはちょっと大変です。

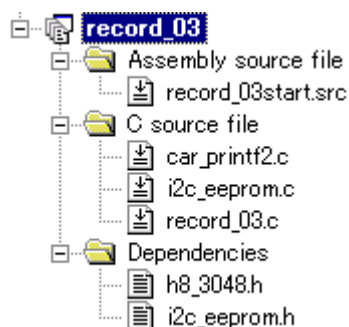
そこで、マイコン側のプログラムで、16 進数から 2 進数に変換する関数を作り、2 進数で直接出力します。そうすれば、いちいち変換する必要が無く、解析が楽になります。

本章では、変換プログラムと出力方法を紹介します。

5.2 接続

「4.2 接続」と同じです。

5.3 プロジェクトの構成



・record_03start.src

・record_03.c

・car_printf2.c

・i2c_eeprom.c

の 4 ファイルあります。

h8_3048.h は record_03.c、car_printf2.c、i2c_eeprom.c でインクルードされているファイルです。

i2c_eeprom.h は record_03.c、i2c_eeprom.c でインクルードされているファイルです。

5.4 プログラム

```

1 :  /*=====*/
2 :  /* 外付けEEP-ROMデータ保存演習プログラム「record_03.c」 */
3 :  /*                      2006.04 ジャパンマイコンカーラリー実行委員会 */
4 :  /*=====*/
5 :  /*
6 :  本プログラムは外付けEEP-ROM(24C256 32KB)にポート 7 のデータ、
7 :  CPUボード上のディップスイッチの値を10msごとに保存します。
8 :  その後、保存したデータをパソコンへ転送します。
9 :  */
10 :
11 :  /*=====*/
12 :  /* インクルード */
13 :  /*=====*/
14 :  #include <no_float.h> /* stdioの簡略化 最初に置く */
15 :  #include <stdio.h>
16 :  #include <machine.h>
17 :  #include "h8_3048.h"
18 :  #include "i2c_eeprom.h" /* EEPROM追加(データ記録) */
19 :
20 :  /*=====*/
21 :  /* シンボル定義 */
22 :  /*=====*/
23 :
24 :  /*=====*/
25 :  /* プロトタイプ宣言 */
26 :  /*=====*/
27 :  void init( void );
28 :  unsigned char dipsw_get( void );
29 :  void convertHexToBin( unsigned char hex, char *s );
30 :
31 :  /*=====*/
32 :  /* グローバル変数の宣言 */
33 :  /*=====*/
34 :  unsigned long cnt1; /* main内で使用 */
35 :  int pattern; /* パターン番号 */
36 :
37 :  /* データ保存関連 */
38 :  int iTimer10; /* 取得間隔計算用 */
39 :  int saveIndex; /* 保存インデックス */
40 :  int saveSendIndex; /* 送信インデックス */
41 :  int saveFlag; /* 保存フラグ */
42 :  char saveData[8]; /* 一時保存エリア */
43 :
44 :  /*=====*/
45 :  /* メインプログラム */
46 :  /*=====*/
47 :  void main( void )
48 :  {
49 :      char s[8];
50 :      /* マイコン機能の初期化 */
51 :      init(); /* 初期化 */
52 :      initI2CEeprom( &PADDR, &PADR, 0x5f, 7, 5 ); /* EEPROM初期設定 */
53 :      init_sci1( 0x00, 79 ); /* SCI1初期化 */
54 :      set_ccr( 0x00 ); /* 全体割り込み許可 */
55 :
56 :      while( 1 ) {
57 :
58 :          I2CEepromProcess(); /* I2C EEPROM保存処理 */
59 :
60 :          switch( pattern ) {

```

convertHexToBin 関数を新たに作りました。プロトタイプ宣言を忘れずに行います。

2 進数変換後のデータを保存する配列です。

中略

```

93 :     case 3:
94 :         /* データ転送 */
95 :         convertHexToBin( readI2CEeprom( saveSendIndex+0 ), s );
96 :         printf( "%8s,%02x\n", s,
97 :             (unsigned char)readI2CEeprom( saveSendIndex+1 ) );
98 :         saveSendIndex += 2;
99 :         if( saveIndex <= saveSendIndex ) {
100 :             pattern = 4;
101 :             cnt1 = 0;
102 :         }
103 :         break;

```

2 進数に変換して printf 文で出力します。

中略

```

181 : /*****
182 :  /* 1 6 進数 2 進数変換
183 :  /* 引数 1 6 進数データ、変換後のデータ格納アドレス
184 :  /* 戻り値 なし
185 :  *****/
186 : void convertHexToBin( unsigned char hex, char *s )
187 : {
188 :     int i;
189 :
190 :     s += 7;
191 :     for( i=0; i<8; i++ ) {
192 :         if( hex & 0x01 ) {
193 :             *s-- = '1';
194 :         } else {
195 :             *s-- = '0';
196 :         }
197 :         hex >>= 1;
198 :     }
199 : }
200 :
201 : /*****
202 :  /* end of file
203 :  *****/

```

unsigned char 型の変数(0 ~ 255 または 0x00 ~ 0xff)を 2 進数に変換する関数です。

5.5 プログラムの解説

5.5.1 変数の追加

```

44 : /*****
45 :  /* メインプログラム
46 :  *****/
47 : void main( void )
48 : {
49 :     char s[8];

```

char 型の s という配列を追加します。8 バイト分です。convertHexToBin 関数で使います。配列 s には、変換後の 2 進数データが文字列で格納されます。詳しくは、convertHexToBin 関数の説明を参照してください。

5.5.2 2進数変換を行う convertHexToBin 関数

```

181 :  /*****
182 :  /* 16進数 2進数変換
183 :  /* 引数 16進数データ、変換後のデータ格納アドレス
184 :  /* 戻り値 なし
185 :  *****/
186 :  void convertHexToBin( unsigned char hex, char *s )
187 :  {
188 :      int i;
189 :
190 :      for( i=0; i<8; i++ ) {
191 :          if( hex & 0x80 ) {
192 :              *s++ = '1';
193 :          } else {
194 :              *s++ = '0';
195 :          }
196 :          hex <<= 1;
197 :      }
198 :  }

```

convertHexToBin という関数を作りました。使い方は下記のようになります。

```
convertHexToBin( 0x55, s );
```

① ②

①...2進数に変換する元データです。

②...2進数に変換したデータを保存する配列を指定します。ここで指定する配列は、char型で8バイト以上の大きさが必要です。

上記を実行した場合、配列 s には下記のように変換されたデータが、'1'または'0'の文字として保存されます。

s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]
'0'	'1'	'0'	'1'	'0'	'1'	'0'	'1'

printf 文で配列 s に格納されている文字列 8 文字を出力すれば、変換された 2 進数データが出力されることになります。

5.5.3 printf 出力

```

95 :      convertHexToBin( readI2CEeprom( saveSendIndex+0 ), s );
96 :      printf( "%8s,%02x\n", s,
97 :              (unsigned char)readI2CEeprom( saveSendIndex+1 ) );

```

95 行...EEP-ROM の (saveSendIndex+0) 番地に保存されたデータを 2 進数に変換します。

96 行...2進データは、文字列 8 文字なので、printf 文では、「%8s」で配列 s に格納している文字列を出力します。

5.6 データ例

下記に、出力例を示します。

data recording...	11111110,12
	11111100,12
record_03 Data Out	01111100,8
P7 data,dip sw data	01111100,0
01010101,15	01111100,0
01010101,15	01111100,4
01010111,15	01111100,4
01010111,15	00111100,4
01011111,15	00111100,6
01011111,15	00111000,6
11011111,15	00011000,6
11011111,15	00011000,6
11111111,15	00011000,7
11111111,15	00010000,7
11111111,14	00010000,7
11111111,14	00000000,7
11111110,14	00000000,7

6. プロジェクト「kit06rec_01」 走行データを内蔵 RAM にデータ保存

6.1 概要

マイコンカーの走行中のデータを、H8 マイコンの内蔵 RAM に保存します。保存する内容は、

- ・パターン番号
- ・センサの値

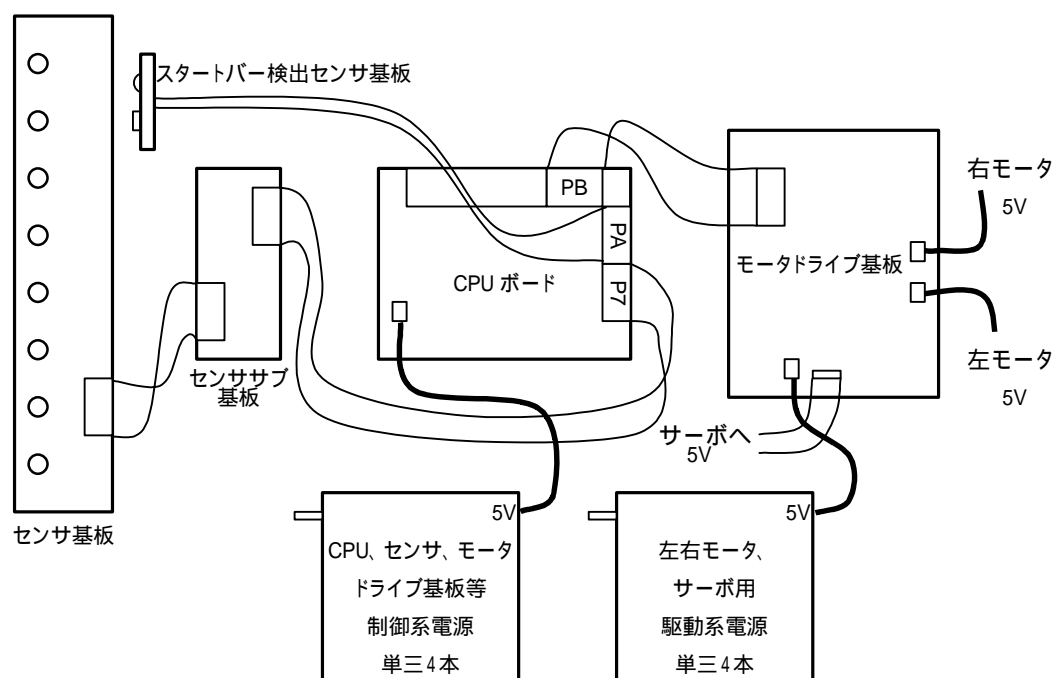
の2つです。これらのデータが走行開始から 10ms ごとに、1000 回分保存されます。10 秒間です。10 秒たっても走行はそのまま行いますが、データ保存は行いません。

保存した走行データをパソコンに送り、マイコンカーがどう走ったかパソコン上で解析します。この情報を基に、プログラムのデバッグに役立てます。

6.2 マイコンカーの構成

マイコンカーキット Vol.3 に、スタートバー検出センサ基板を追加した構成です。LM350 追加セットで電池 8 本を直列に繋いでいる構成でも OK です。

- ・CPU ボードのポート 7 と、マイコンカーキット Vol.3 のセンサ基板を接続します。
- ・CPU ボードのポート B と、マイコンカーキット Vol.3 のモータドライブ基板 Vol.3 を接続します。
- ・CPU ボードのポート A と、スタートバー検出センサ基板を接続します。



6.3 プロジェクトの構成

kit06rec_01



・kit06rec_01start.src

・kit06rec_01.c

・car_printf2.c

の 3 ファイルあります。

h8_3048.h は kit06rec_01.c、car_printf2.c でインクルードされているファイルです。

6.4 プログラム

```

1 :  /*
2 :  /* 走行データ記録マイコンカートレース基本プログラム「kit06rec_01.c」
3 :  /* 2006.04 ジャパンマイコンカーラリー実行委員会
4 :  /*
5 :  /*
6 :  本プログラムはkit06.cをベースに走行データを記録するプログラムです。
7 :
8 :  kit06rec_01.cは、H8/3048F-ONEの内蔵RAM(2.5KB程度)にマイコンカーの走行
9 :  データを保存します。その後、保存したデータをパソコンへ転送します。
10 :  パソコンでデータを解析すれば、なぜ脱輪したかなど問題を「見える化」
11 :  することが出来、プログラムのデバッグに役立てることが出来ます。
12 :  */
13 :
14 :  /*=====*/
15 :  /* インクルード */
16 :  /*=====*/
17 :  #include <no_float.h> /* stdioの簡略化 最初に置く*/
18 :  #include <stdio.h>
19 :  #include <machine.h>
20 :  #include "h8_3048.h"
21 :
22 :  /*=====*/
23 :  /* シンボル定義 */
24 :  /*=====*/
25 :
26 :  /* 定数設定 */
27 :  #define TIMER_CYCLE 3071 /* タイマのサイクル 1ms */
28 :  /* /8で使用する場合、 */
29 :  /* /8 = 325.5[ns] */
30 :  /* TIMER_CYCLE = */
31 :  /* 1[ms] / 325.5[ns] */
32 :  /* = 3072 */
33 :  #define PWM_CYCLE 49151 /* PWMのサイクル 16ms */
34 :  /* PWM_CYCLE = */
35 :  /* 16[ms] / 325.5[ns] */
36 :  /* = 49152 */
37 :  #define SERVO_CENTER 5000 /* サーボのセンタ値 */
38 :  #define HANDLE_STEP 26 /* 1° 分の値 */
39 :
40 :  /* マスク値設定 x : マスクあり(無効) ; マスク無し(有効) */
41 :  #define MASK2_2 0x66 /* x x x x */
42 :  #define MASK2_0 0x60 /* x x x x x */
43 :  #define MASK0_2 0x06 /* x x x x x */
44 :  #define MASK3_3 0xe7 /* x x x x x */
45 :  #define MASK0_3 0x07 /* x x x x x */
46 :  #define MASK3_0 0xe0 /* x x x x x */
47 :  #define MASK4_0 0xf0 /* x x x x x */
48 :  #define MASK0_4 0x0f /* x x x x x */
49 :  #define MASK4_4 0xff /* x x x x x */
50 :
51 :  #define SAVE_SIZE 1000 /* データ保存数 */
52 :
53 :  /*=====*/
54 :  /* プロトタイプ宣言 */
55 :  /*=====*/
56 :  void init( void );
57 :  void timer( unsigned long timer_set );
58 :  int check_crossline( void );
59 :  int check_rightline( void );
60 :  int check_leftline( void );
61 :  unsigned char sensor_inp( unsigned char mask );
62 :  unsigned char dipsw_get( void );
63 :  unsigned char pushsw_get( void );

```

printf 文を使用するので、stdio.h をインクルードします。

データを保存する数です。
保存時間は、
保存間隔 10ms × 1000 = 10 秒
となります。

データ解析実習マニュアル kit06 版

```

64 : unsigned char startbar_get( void );
65 : void led_out( unsigned char led );
66 : void speed( int accele_l, int accele_r );
67 : void handle( int angle );
68 : char unsigned bit_change( char unsigned in );
69 : void convertHexToBin( unsigned char hex, char *s );
70 :
71 : /*=====*/
72 : /* グローバル変数の宣言 */
73 : /*=====*/
74 : unsigned long cnt0; /* timer関数用 */
75 : unsigned long cnt1; /* main内で使用 */
76 : int pattern; /* パターン番号 */
77 :
78 : /* データ保存関連 saveDataのサイズは最大で2.5KB程度としてください */
79 : int iTimer10; /* 取得間隔計算用 */
80 : unsigned char saveData[SAVE_SIZE][2]; /* 保存エリア */
81 : int saveIndex; /* 保存インデックス */
82 : int saveSendIndex; /* 送信インデックス */
83 : int saveFlag; /* 保存フラグ */
84 :
85 : /*=====*/
86 : /* メインプログラム */
87 : /*=====*/
88 : void main( void )
89 : {
90 :     int i;
91 :     char s[8];
92 :
93 :     /* マイコン機能の初期化 */
94 :     init(); /* 初期化 */
95 :     init_sci1( 0x00, 79 ); /* SC11初期化 */
96 :     set_ccr( 0x00 ); /* 全体割り込み許可 */
97 :
98 :     /* マイコンカーの状態初期化 */
99 :     handle( 0 );
100 :    speed( 0, 0 );
101 :
102 :    while( 1 ) {
103 :
104 :        P4DR = ~pattern; /* デバッグ用にパターン出力 */
105 :
106 :        switch( pattern ) {

```

データ保存関係の変数です。
record_01 と同じです。

printf 文で通信を使用しますので、init_sci1 で通信を初期化します。

デバッグ用としてポート 4 にパターンを出力します。LED をポート 4 に繋がます。

中略

```

148 :    case 1:
149 :        /* スタートバーが開いたかチェック */
150 :        if( !startbar_get() ) {
151 :            /* スタート!! */
152 :            led_out( 0x0 );
153 :            pattern = 11;
154 :            saveIndex = 0;
155 :            saveFlag = 1; /* データ保存開始 */
156 :            cnt1 = 0;
157 :            break;
158 :        }
159 :        if( cnt1 < 50 ) { /* LED点滅処理 */
160 :            led_out( 0x1 );
161 :        } else if( cnt1 < 100 ) {
162 :            led_out( 0x2 );
163 :        } else {
164 :            cnt1 = 0;
165 :        }
166 :        break;
167 :
168 :    case 11:
169 :        /* 通常トレース */
170 :        if( pushsw_get() ) {
171 :            pattern = 71; /* データ転送処理へ */
172 :            break;
173 :        }
174 :
175 :        if( check_crossline() ) { /* クロスラインチェック */
176 :            pattern = 21;
177 :            break;
178 :        }
179 :        if( check_rightline() ) { /* 右ハーフラインチェック */
180 :            pattern = 51;
181 :            break;
182 :        }
183 :        if( check_leftline() ) { /* 左ハーフラインチェック */
184 :            pattern = 61;
185 :            break;
186 :        }
187 :        switch( sensor_inp(MASK3_3) ) {
188 :            case 0x00:
189 :                /* センタ まっすぐ */
190 :                handle( 0 );
191 :                speed( 100, 100 );
192 :                break;

```

saveIndex は保存する配列の位置を指定します。最初は 0 です。
saveFlag=1 で保存を開始します。

パターン 11 のときに、スイッチを押すと、パターン 71 へ移ります。
パターン 71 は、データを転送する部分です。

中略

```

507 : case 71:
508 :     /* 停止 */
509 :     handle( 0 );
510 :     speed( 0, 0 );
511 :     saveFlag = 0;
512 :     saveSendIndex = 0;
513 :     pattern = 72;
514 :     cnt1 = 0;
515 :     break;
516 :
517 : case 72:
518 :     /* 1s待ち */
519 :     if( cnt1 > 1000 ) {
520 :         pattern = 73;
521 :         cnt1 = 0;
522 :     }
523 :     break;
524 :
525 : case 73:
526 :     /* スイッチが離されたかチェック */
527 :     if( lpushsw_get() ) {
528 :         pattern = 74;
529 :         cnt1 = 0;
530 :     }
531 :     break;
532 :
533 : case 74:
534 :     /* スイッチが押されたかチェック */
535 :     led_out( (cnt1/500) % 2 + 1 );
536 :     if( pushsw_get() ) {
537 :         pattern = 75;
538 :         cnt1 = 0;
539 :     }
540 :     break;
541 :
542 : case 75:
543 :     /* タイトル転送、準備 */
544 :     printf( "\n" );
545 :     printf( "kit06rec_01 Data Out\n" );
546 :     printf( "Pattern, Sensor\n" );
547 :     pattern = 76;
548 :     break;
549 :
550 : case 76:
551 :     /* データ転送 */
552 :     led_out( (saveSendIndex/32) % 2 + 1 ); /* LED点滅処理 */
553 :
554 :     convertHexToBin( saveData[saveSendIndex][1], s );
555 :     printf( "%d,%8s\n", saveData[saveSendIndex][0], s );
556 :     saveData[saveSendIndex][0], /* パターン */
557 :     s /* センサ値(2進数) */
558 :
559 :     /* 終わりのチェック */
560 :     saveSendIndex++;
561 :     if( saveIndex <= saveSendIndex ) {
562 :         pattern = 77;
563 :         cnt1 = 0;
564 :     }
565 :     break;
566 :
567 : case 77:
568 :     /* 転送終了 */
569 :     led_out( 0x3 );
570 :     break;
571 :
572 : default:
573 :     /* どれでもない場合は待機状態に戻す */
574 :     pattern = 0;
575 :     break;
576 : }
577 : }
578 :
579 : /*****
580 : /* H8/3048F-ONE 内蔵周辺機能 初期化 */
581 : /*****
582 : void init( void )
583 : {
584 :     /* I/Oポートの入出力設定 */
585 :     P1DDR = 0xff;
586 :     P2DDR = 0xff;
587 :     P3DDR = 0xff;
588 :     P4DDR = 0xff;
589 :     P5DDR = 0xff;
590 :     P6DDR = 0xf0; /* CPU基板上的DIP SW */
591 :     P8DDR = 0xff;
592 :     P9DDR = 0xf7; /* 通信ポート */
593 :     PADDR = 0xf7; /* スタートバー検出センサ */
594 : }

```

パターン 71 以降は、保存したデータを転送する部分です。
まず、マイコンカーを停止させます。
saveFlag=0 でデータ保存を終了します。saveSendIndex は転送する配列の位置を指定する変数です。0 から順番に転送するので、最初にクリアします。パターン 72 へ移ります。

単純に1秒待ちます。1秒後、パターン 73 へ移ります。

スイッチが離されていれば、パターン 74 へ移ります。

スイッチが押されれば、パターン 75 へ移ります。
押されるまでの間、LED を点滅させ、転送待機状態だということを外部に知らせます。

タイトルを転送します。すぐに、パターン 76 へ移ります。

センサ値を2進数に変換します。

データがある限り、転送し続けます。
保存数を超えてデータを転送しようとするとパターン 77 へ移り、転送を終了します。

何もしません。電源が切れるのを待ちます。

```

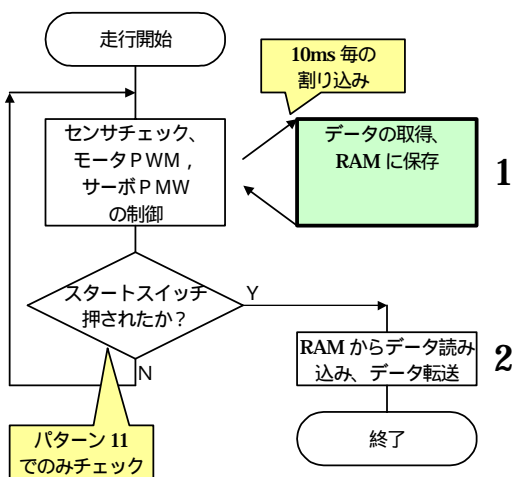
595 :   PBDR = 0xc0;
596 :   PBDDR = 0xfe; /* モータドライブ基板 Vol.3 */
597 :   /* センサ基板のP7は、入力専用なので入出力設定はありません */
598 :
599 :   /* ITU0 1msごとの割り込み */
600 :   ITU0_TCR = 0x23;
601 :   ITU0_GRA = TIMER_CYCLE;
602 :   ITU0_IER = 0x01;
603 :
604 :   /* ITU3,4 リセット同期PWMモード 左右モータ、サーボ用 */
605 :   ITU3_TCR = 0x23;
606 :   ITU_FCR = 0x3e;
607 :   ITU3_GRA = PWM_CYCLE; /* 周期の設定 */
608 :   ITU3_GRB = ITU3_BRB = 0; /* 左モータのPWM設定 */
609 :   ITU4_GRA = ITU4_BRA = 0; /* 右モータのPWM設定 */
610 :   ITU4_GRB = ITU4_BRB = SERVO_CENTER; /* サーボのPWM設定 */
611 :   ITU_TOER = 0x38;
612 :
613 :   /* ITUのカウンタスタート */
614 :   ITU_STR = 0x09;
615 : }
616 :
617 : /* ITU0 割り込み処理 */
618 : /* ITU0 割り込み処理 */
619 : /* ITU0 割り込み処理 */
620 : #pragma interrupt( interrupt_timer0 )
621 : void interrupt_timer0( void )
622 : {
623 :     ITU0_TSR &= 0xfe; /* フラグクリア */
624 :     cnt0++;
625 :     cnt1++;
626 :
627 :     /* データ保存関連 */
628 :     iTimer10++;
629 :     if( iTimer10 >= 10 ) {
630 :         iTimer10 = 0;
631 :         if( saveFlag ) {
632 :             saveData[saveIndex][0] = pattern; /* パターン */
633 :             saveData[saveIndex][1] = sensor_inp(0xff); /* センサ */
634 :             saveIndex++;
635 :             if( saveIndex >= SAVE_SIZE ) saveFlag = 0;
636 :         }
637 :     }
638 : }

```

10ms ごとに配列に保存します。1 回にパターンとセンサ値の2バイトを保存します。配列が一杯になると、saveFlag を 0 にして保存を強制的に終了します。

以下、略

6.5 プログラムの概要



マイコンカーは、

- ・センサ(コースセンサ、エンコーダなど)を見ながら
- ・駆動モータの制御
- ・サーボ(ステアリングモータ)の制御

を行っています。これらの制御に影響が無いようにデータ保存を行わなければいけません。

データの取得、保存は割り込み内で行います。できるだけ時間をかけないようにします(1部分)。

走行終了後、マイコンカーを取り上げてスタートスイッチを押します。パソコンと RS232C ケーブルを接続してデータを転送し、取得したデータがパソコンに取り込まれます(2部分)。

ポイントは、

- ・走行中、スイッチを押したときに止めることができるのはパターン 11 のみです。パターン 11 以外のときにマイコンカーを取り上げた場合、手でセンサの反応を上手く切り替え、パターン 11 にしてからスイッチを押します。
- ・電源を切ると保存したデータが消えてしまうので、電源を入れたままで転送を行います。
- ・保存するデータは、パターン番号とセンサ値です。自由に変更できます。

6.6 プログラムの解説

6.6.1 データ保存エリア

saveData 配列が、データを保存するエリアです。2 次元配列になっています。SAVE_SIZE は 1000 とします。

	A=0	A=1	
saveData[0][A]	10ms 後のパターン	10ms 後のセンサ値	} 1000 行
saveData[1][A]	20ms 後のパターン	20ms 後のセンサ値	
saveData[2][A]	30ms 後のパターン	30ms 後のセンサ値	
saveData[3][A]	40ms 後のパターン	40ms 後のセンサ値	
saveData[4][A]	50ms 後のパターン	50ms 後のセンサ値	
...			
saveData[998][A]	9990ms 後のパターン	9990ms 後のセンサ値	
saveData[999][A]	10000ms 後のパターン	10000ms 後のセンサ値	
		2 列	

2 列 1000 行、1 マスが 1 バイトなので、saveData 配列は合計 2000 バイトのサイズとなります。

H8 の RAM 4KB の内、2000 バイトをデータ保存エリアとして使用します。

保存方法やデータ転送方法などは、「record_01」と同じです。

6.6.2 送信内容

```

550 :      case 76:
551 :          /* データ転送 */
552 :          led_out( (saveSendIndex/32) % 2 + 1 ); /* LED 点滅処理 */
553 :
554 :          convertHexToBin( saveData[saveSendIndex][1], s );
555 :          printf( "%d,%8s¥n",
556 :                  saveData[saveSendIndex][0], /* パターン */
557 :                  s ); /* センサ値(2 進数) */
558 :
559 :          /* 終わりのチェック */
560 :          saveSendIndex++;
561 :          if( saveIndex <= saveSendIndex ) {
562 :              pattern = 77;
563 :              cnt1 = 0;
564 :          }
565 :          break;

```

送信内容は、554 ~ 557 行の内容です。

printf("%d,%8s¥n",

↑ ↑

```

      saveData[saveSendIndex][0], /* パターン */
      s ); /* センサ値(2 進数) */

```

パターンは 10 進数、センサ値は 2 進数で出力します。10 進数は「%d」、2 進数は printf 文では直接出力できないので、convertHexToBin 関数でいったん 16 進数を 2 進数に変換します。その文字列を出力するので、「%8s」となります。

6.7 プログラムの調整

「kit06rec_01.c」の下記の内容を、自分のマイコンカーに合わせて調整します。

37 :	#define	SERVO_CENTER	5000	/* サーボのセンタ値	*/
中略					
309 :		handle(-38);		左クランクを曲がる時のハンドル角度	
中略					
318 :		handle(38);		右クランクを曲がる時のハンドル角度	

走らせた後は、電源を切らずに持ち上げ、スイッチを1回押します。マイコンカーはデータ停止して転送モードになり、LED がゆっくり点滅します。このとき、マイコンカーの状態はパターン 11 を実行している状態であればいいません。これは、下記のようにスイッチが押されたらデータ転送処理へ移るプログラム部分が、パターン 11 にしか無いからです。

168 :	case 11:				
169 :		/* 通常トレース */			
170 :		if(pushsw_get()) {			
171 :		pattern = 71;	/* データ転送処理へ	*/	
172 :		break;			
173 :		}			

もし、パターン 12 や 13、その他でもスイッチを押したときに停止してデータ転送処理へ移るようにしたい場合は、それぞれのパターンの先頭部分に、上記4行を追加します。例として、パターン 12 に追加した例を下記に示します。

249 :	case 12:				
250 :		/* 右へ大曲げの終わりのチェック */			
*** :		if(pushsw_get()) {			
*** :		pattern = 71;	/* データ転送処理へ	*/	追加
*** :		break;			
*** :		}			
251 :		if(check_crossline()) {	/* 大曲げ中もクロスラインチェック */		
252 :		pattern = 21;			
253 :		break;			
254 :		}			

他にも、調整する部分は調整してください。

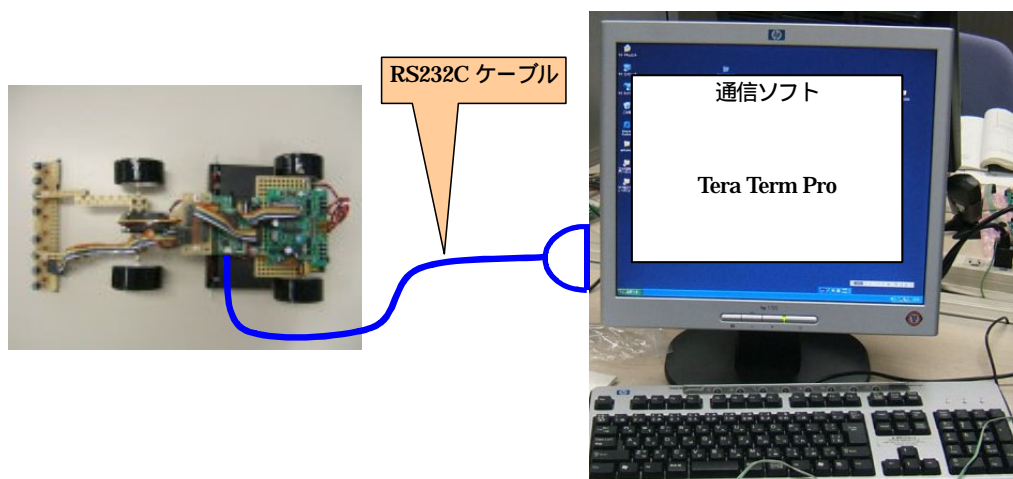
調整ができれば、プロジェクト「kit06rec_01」をビルドして、「kit06rec_01.mot」ファイルを CPU ボードに書き込みます。

6.8 走行からデータ転送までの流れ

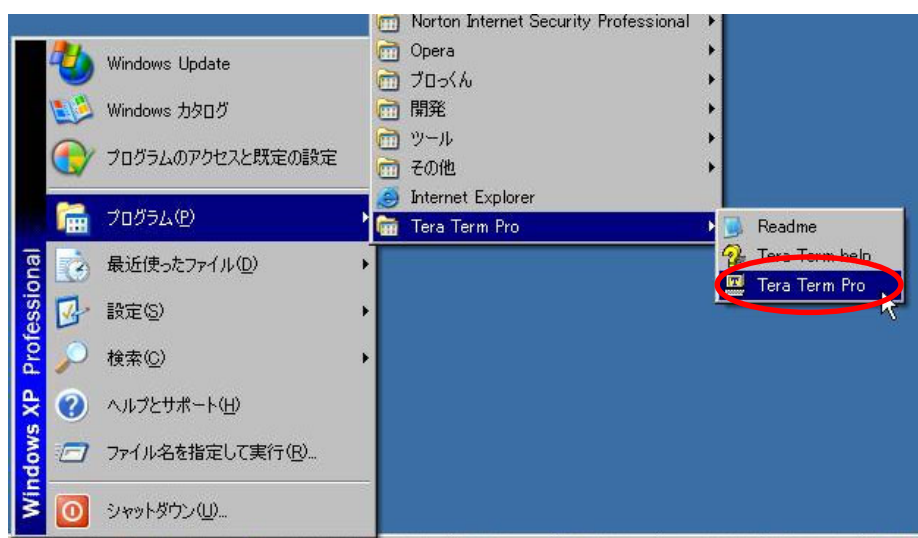
走行データを取りたい部分のコースを普通に走らせます。走行データが保存できるのは、スタートしてから 10 秒です。

走らせた後、**電源を切らずに持ち上げ、モータドライブ基板のスイッチを1回押します。マイコンカーは停止して転送モードになり、LED がゆっくり点滅します。**

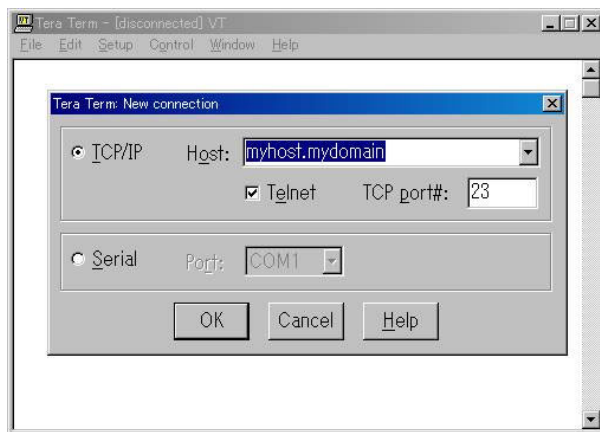
もし、スイッチを押しても転送モードにならない場合、マイコンカーが動作しているパターンに、スイッチを押したことを検出する部分が無いためです。「6.7 プログラムの調整」を参照してプログラムを追加するか、パターン 11 の状態でスイッチを押してください。



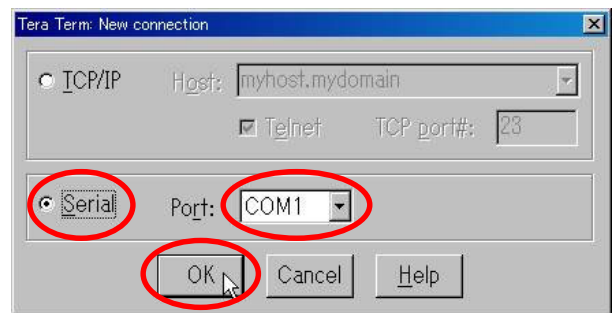
1. マイコンカーが止まり LED がゆっくり点滅している状態で、RS232C ケーブルをマイコンカーに接続します。



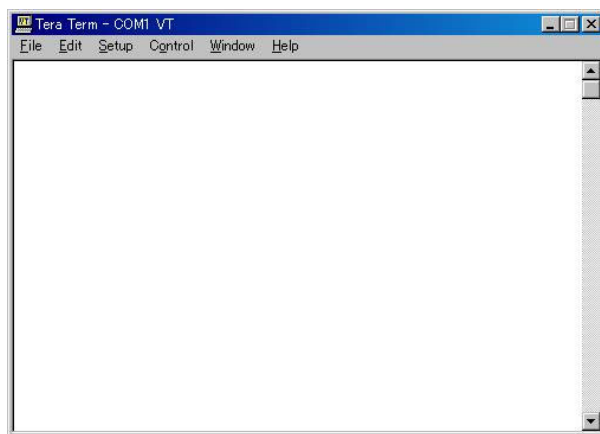
2. 「スタート すべてのプログラム (またはプログラム) Tera Term Pro Tera Term Pro」で Tera Term Pro を立ち上げます。



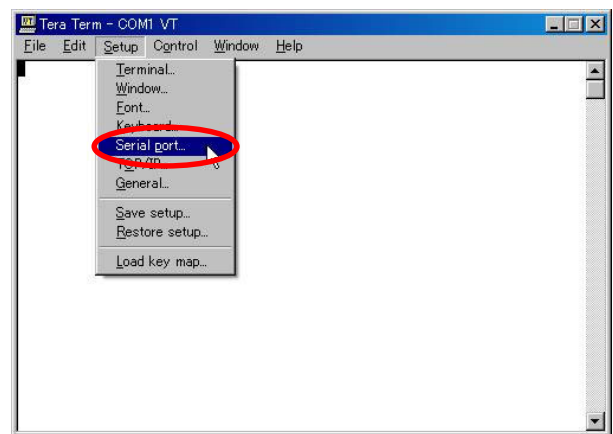
3. 最初に接続先を確認する画面が出てきます。



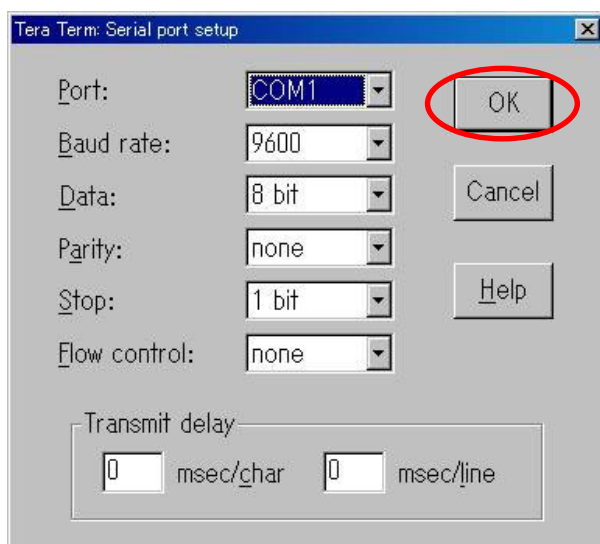
4. 「Serial」を選んで、各自のパソコンに合わせてポート番号を選びます。選択後、**OK**をクリック、次へ進みます。



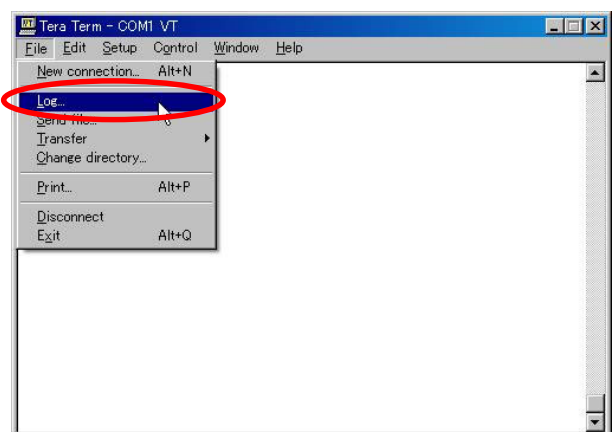
5. 立ち上がりました。詳細設定をします。



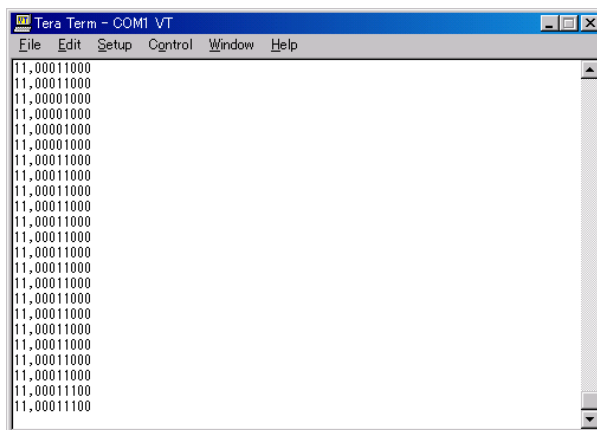
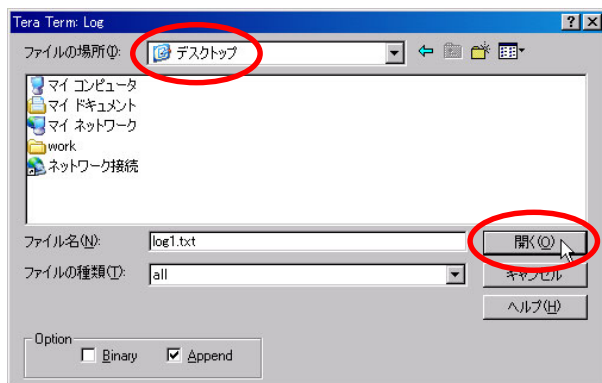
6. 「Setup Serial port」を選択します。



7. 通信設定を確認します。画面のように設定して、**OK**をクリックします。

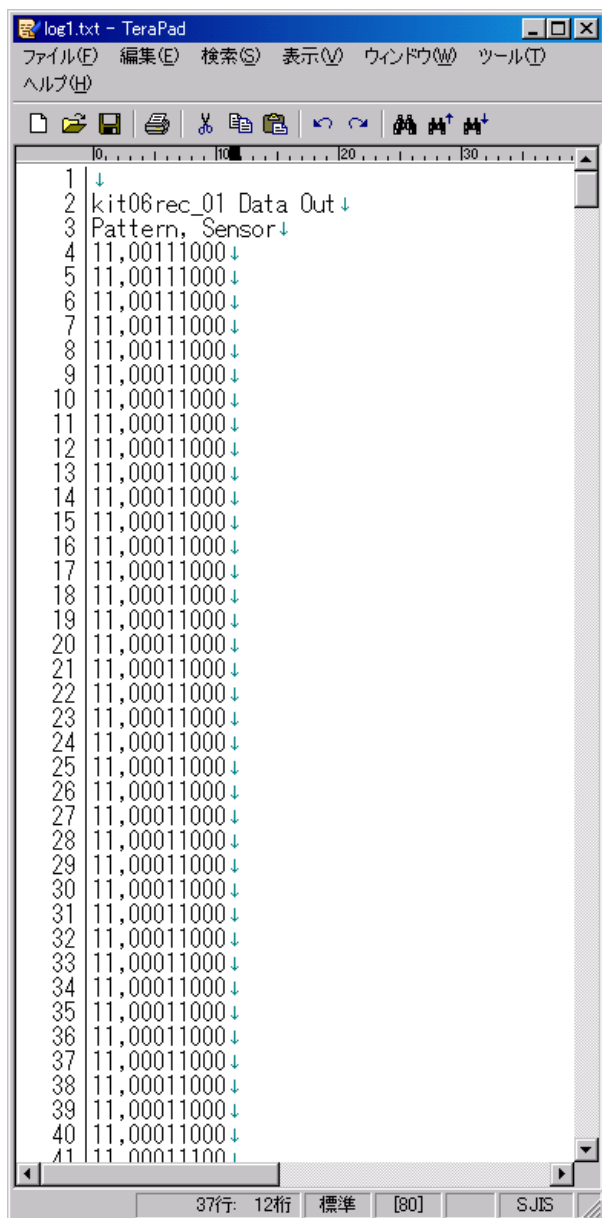


8. データをファイルに保存します。「File Log」を選択します。



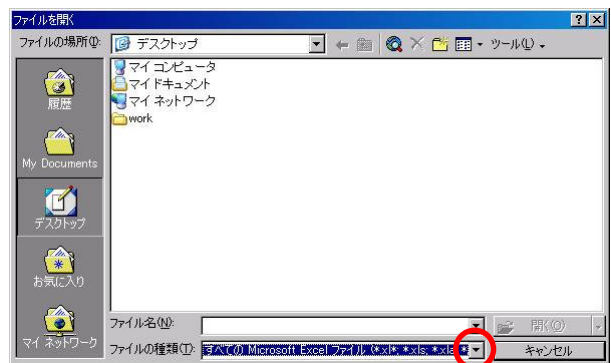
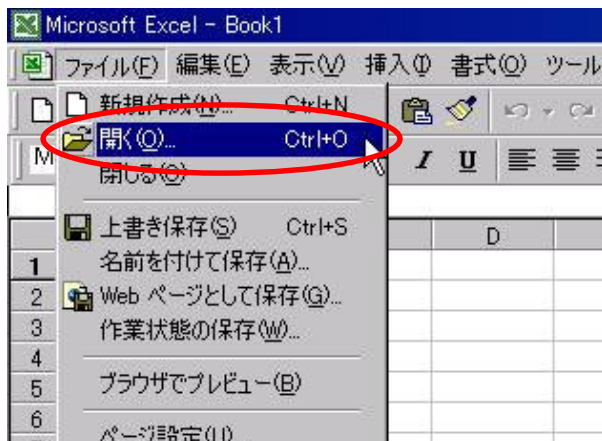
9. 「ファイルの場所」に保存するフォルダを選択します。「ファイル名」には、保存するファイルの名前を入力します。ここでは「log1.txt」と入力します。**開く**をクリックします。

10. モータドライブ基板のスイッチを押すと、データが転送されます。転送が終わったら、TeraTermPoを終了して、マイコンカーの電源を切ってください。

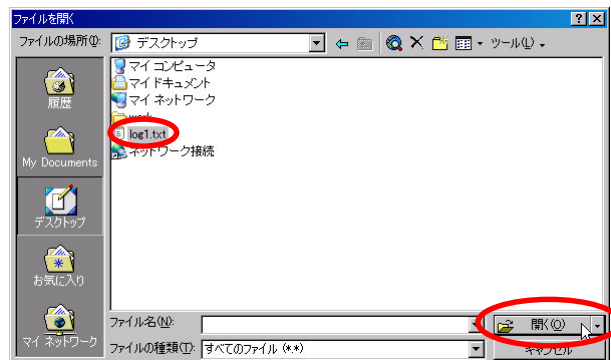
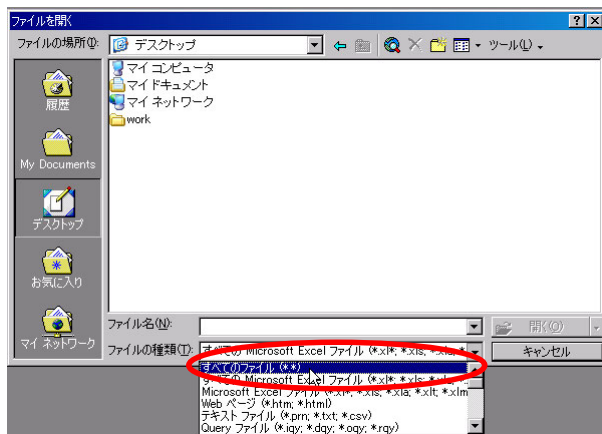


11. log1.txt をエディタで開いてみました。保存されています。このデータをエクセルに取り込んでみましょう。

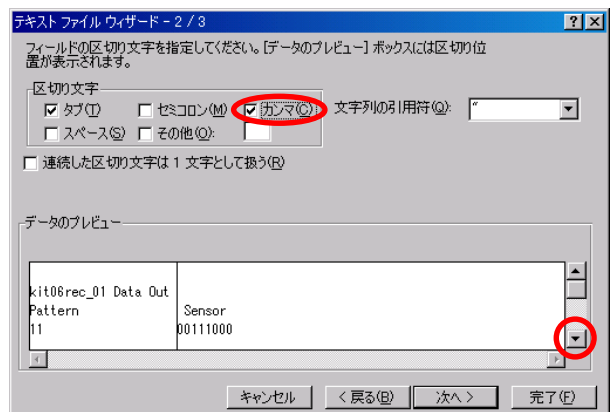
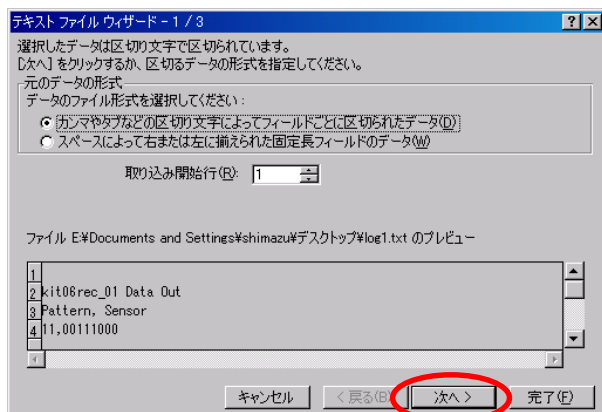
6.9 エクセルへの取り込み方



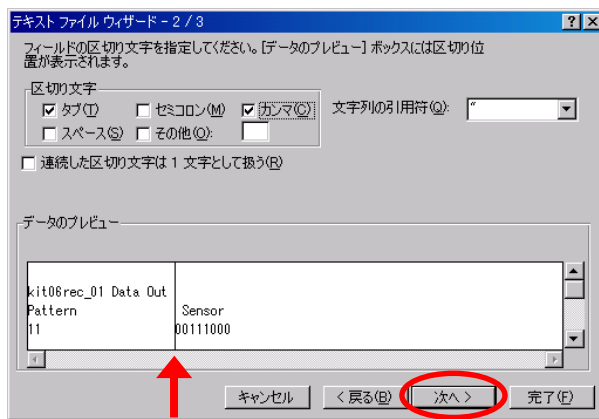
1. エクセルを立ち上げます。「ファイル 開く」を選択します。
2. 「ファイルの種類」の□をクリックします。



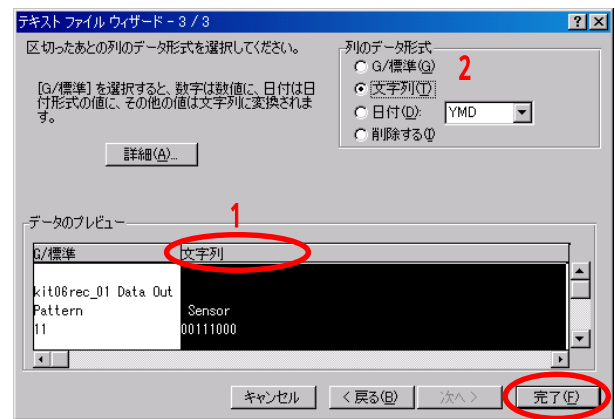
3. 「すべてのファイル (*.*)」を選択します。
4. 「log1.txt」が表示されました。「log1.txt」を選択、「開く」をクリックします。



5. 開こうとしているデータは、テキストデータです。エクセルデータ(セル)に変換する作業を行います。「次へ」をクリックします。
6. データはカンマで区切られています。区切り文字の「カンマ」のチェックを付けます。□でデータのプレビュー欄を下げます。



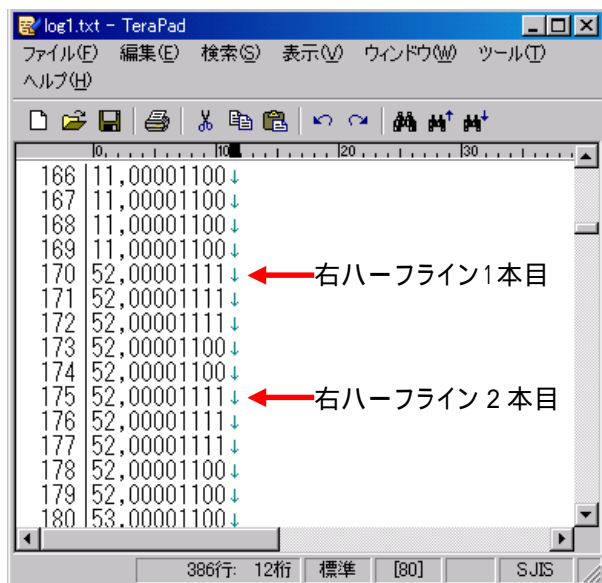
7. 矢印()部分に縦線が入りました。これが列の区切りです。縦線が入ってなければ「区切り文字」のチェック指定がおかしいので確かめます。次へをクリックします。



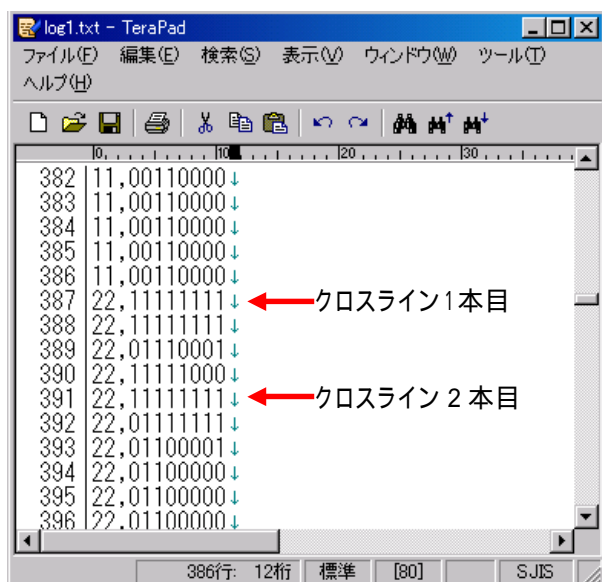
8. 次に、列のデータ形式を指定します。1列目は、パターンです。パターンは数値なので、標準のままでOKです。2列目は、センサ値です。16進数なので文字列扱いにします。1部分をクリックします。次に2部分の「文字列」を選択します。完了で完了です。

	A	B	C	D	E
1					
2	kit06rec_01 Data Out	Sensor			
3	Pattern				
4	11	00111000			
5	11	00111000			
6	11	00111000			
7	11	00111000			
8	11	00111000			
9	11	00011000			
10	11	00011000			
11	11	00011000			
12	11	00011000			
13	11	00011000			
14	11	00011000			
15	11	00011000			
16	11	00011000			
17	11	00011000			
18	11	00011000			
19	11	00011000			
20	11	00011000			
21	11	00011000			
22	11	00011000			
23	11	00011000			
24	11	00011000			
25	11	00011000			
26	11	00011000			
27	11	00011000			
28	11	00011000			
29	11	00011000			
30	11	00011000			
31	11	00011000			
32	11	00011000			
33	11	00011000			
34	11	00011000			

9. セルに変換されました。A列はパターンを示す数値です。例えば、A4は「11(じゅういち)」という10進数の数字です。B列はセンサの状態を2進数で示す文字列です。



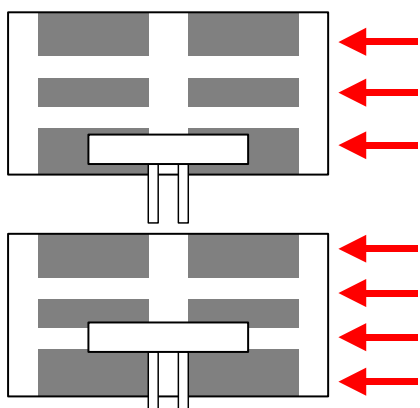
10. 右ハーフラインを検出したところです。
 convertHexToBin 関数で2進数に変換されているので、log1.txt を直接見てセンサの状態が分かります。
 16進数でも想像できますが、やはり2進表記の方が分かりやすいです。



11. クロスラインを検出したところです。

取得タイミングについて

取得タイミングとマイコンカーのスピードによってはデータの取れ方が異なります。 クロスラインを通過するときを例として見てみます。矢印が、ちょうど 10ms ごとにデータを取得している瞬間です。



左図のように、たまたま 10ms ごとの取得が、クロスラインを検出しない状態でした。ただし、パターンは 21,22,23 と変わってきますので、パターン番号からクロスラインを越えたと判断できます。

下左図は、クロスラインの白線を検出していますが、白線と白線間の黒を検出していません。こちらもパターン番号から、クロスラインを越えたと判断できます。

7. プロジェクト「kit06rec_02」 外付け EEP-ROM にデータ保存

7.1 概要

マイコンカーの走行データを、外付け EEP-ROM に保存します。保存する内容は、

- ・パターン番号
- ・センサの値
- ・ハンドル角度
- ・左モータPWM値
- ・右モータPWM値

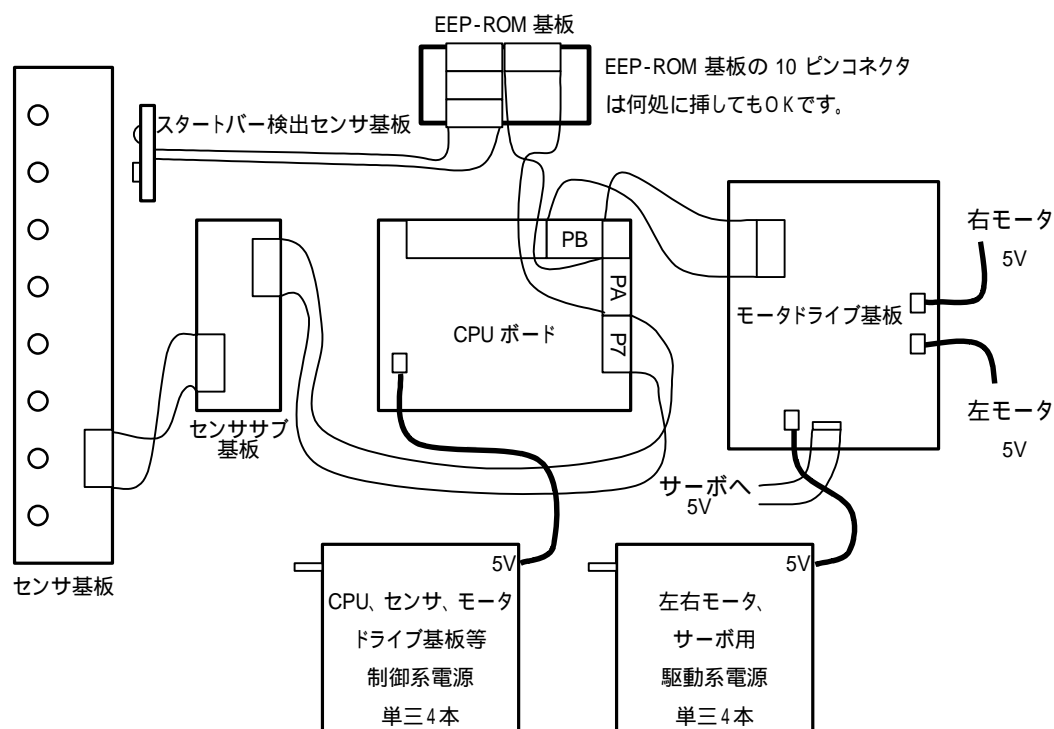
の5つです。これらのデータが走行開始から 10ms ごとに、4096 回分保存されます。40.96 秒間です。40.96 秒後、走行はそのまま行いますがデータ保存は自動で終了します。

保存した走行データをパソコンに送り、マイコンカーがどう走ったかパソコン上で解析します。この情報を基に、プログラムのデバッグに役立てます。

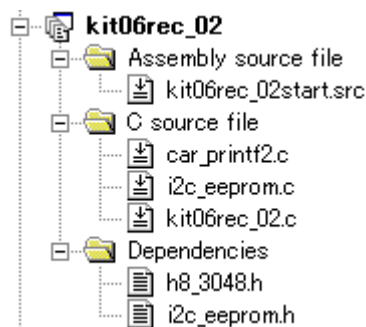
7.2 マイコンカーの構成

マイコンカーキット Vol.3 に、スタートバー検出センサ基板、EEP-ROM 基板を追加した構成です。LM350 追加セットで電池 8 本を直列に繋いでいる構成でも OK です。

- ・CPU ボードのポート 7 と、マイコンカーキット Vol.3 のセンサ基板を接続します。
- ・CPU ボードのポート B と、マイコンカーキット Vol.3 のモータドライブ基板 Vol.3 を接続します。
- ・CPU ボードのポート A と、EEP-ROM 基板を接続します。
- ・EEP-ROM 基板と、スタートバー検出センサ基板を接続します。



7.3 プロジェクトの構成



・kit06rec_02start.src

・kit06rec_02.c

・car_printf2.c

・i2c_eeprom.c

の 4 ファイルあります。

h8_3048.h は kit06rec_02.c、car_printf2.c、i2c_eeprom.c でインクルードされているファイルです。

i2c_eeprom.h は kit06rec_02.c、i2c_eeprom.c でインクルードされているファイルです。

7.4 プログラム

```

1 : /*****
2 : /* 走行データ記録マイコンカートレース基本プログラム「kit06rec_02.c」 */
3 : /* 2006.04 ジャパンマイコンカーラリー実行委員会 */
4 : /*****
5 : /*
6 : 本プログラムはkit06.cをベースに走行データを記録するプログラムです。
7 :
8 : kit06rec_02.cは、外付けEEP-ROM(24C256 32KB)にマイコンカーの走行
9 : データを保存します。その後、保存したデータをパソコンへ転送します。
10 : パソコンでデータを解析すれば、なぜ脱輪したかなど問題を「見える化」
11 : することが出来、プログラムのデバッグに役立てることが出来ます。
12 : */
13 :
14 : /*=====*/
15 : /* インクルード */
16 : /*=====*/
17 : #include <no_float.h> /* stdioの簡略化 最初に置く */
18 : #include <stdio.h>
19 : #include <machine.h>
20 : #include "h8_3048.h"
21 : #include "i2c_eeprom.h" /* EEPROM追加(データ記録) */
22 : /*=====*/
23 : /* シンボル定義 */
24 : /*=====*/
25 :
26 : /* 定数設定 */
27 : #define TIMER_CYCLE 3071 /* タイマのサイクル 1ms */
28 : /* /8で使用する場合、 */
29 : /* /8 = 325.5[ns] */
30 : /* TIMER_CYCLE = */
31 : /* 1[ms] / 325.5[ns] */
32 : /* = 3072 */
33 : #define PWM_CYCLE 49151 /* PWMのサイクル 16ms */
34 : /* PWM_CYCLE = */
35 : /* 16[ms] / 325.5[ns] */
36 : /* = 49152 */
37 : #define SERVO_CENTER 5000 /* サーボのセンタ値 */
38 : #define HANDLE_STEP 26 /* 1° 分の値 */
39 :
40 : /* マスク値設定 x : マスクあり(無効) ; マスク無し(有効) */
41 : #define MASK2_2 0x66 /* x x x x */
42 : #define MASK2_0 0x60 /* x x x x x */
43 : #define MASK0_2 0x06 /* x x x x x */
44 : #define MASK3_3 0xe7 /* x x x x x */
45 : #define MASK0_3 0x07 /* x x x x x */
46 : #define MASK3_0 0xe0 /* x x x x x */
47 : #define MASK4_0 0xf0 /* x x x x x */
48 : #define MASK0_4 0x0f /* x x x x x */
49 : #define MASK4_4 0xff /* x x x x x */
50 :
51 : /*=====*/
52 : /* プロトタイプ宣言 */
53 : /*=====*/
54 : void init( void );
55 : void timer( unsigned long timer_set );
56 : int check_crossline( void );
57 : int check_rightline( void );
58 : int check_leftline( void );
59 : unsigned char sensor_inp( unsigned char mask );
60 : unsigned char dipsw_get( void );
61 : unsigned char pushsw_get( void );
62 : unsigned char startbar_get( void );
63 : void led_out( unsigned char led );
64 : void speed( int accele_l, int accele_r );

```

printf 文を使用するので、stdio.h をインクルードします。

i2c_eeprom.c 内の関数を使うために、i2c_eeprom.h をインクルードします。


```

65 : void handle( int angle );
66 : char unsigned bit_change( char unsigned in );
67 : void convertHexToBin( unsigned char hex, char *s );
68 :
69 : /*=====*/
70 : /* グローバル変数の宣言 */
71 : /*=====*/
72 : unsigned long cnt0; /* timer関数用 */
73 : unsigned long cnt1; /* main内で使用 */
74 : int pattern; /* パターン番号 */
75 :
76 : /* データ保存関連 */
77 : int iTimer10; /* 取得間隔計算用 */
78 : int saveIndex; /* 保存インデックス */
79 : int saveSendIndex; /* 送信インデックス */
80 : int saveFlag; /* 保存フラグ */
81 : char saveData[8]; /* 一時保存エリア */
82 : /*
83 : 保存内容
84 : 0:pattern 1:Sensor 2:handle 3:motor_l
85 : 4:motor_r 5: 6: 7:
86 : */
87 :
88 : /*=====*/
89 : /* メインプログラム */
90 : /*=====*/
91 : void main( void )
92 : {
93 :     int i;
94 :     char s[8];
95 :
96 :     /* マイコン機能の初期化 */
97 :     init(); /* 初期化 */
98 :     initI2CEeprom( &PADDR, &PADR, 0x57, 7, 5); /* EEPROM初期設定 */
99 :     init_sci1( 0x00, 79 ); /* SC11初期化 */
100 :     set_ccr( 0x00 ); /* 全体割り込み許可 */
101 :
102 :     /* マイコンカーの状態初期化 */
103 :     handle( 0 );
104 :     speed( 0, 0 );
105 :
106 :     /* スタート時、スイッチが押されていればデータ転送モード */
107 :     if( pushsw_get() ) {
108 :         pattern = 71;
109 :         cnt1 = 0;
110 :     }
111 :
112 :     while( 1 ) {
113 :
114 :         P4DR = ~pattern; /* デバッグ用にパターン出力 */
115 :         I2CEepromProcess(); /* I2C EEPROM保存処理 */
116 :
117 :         switch( pattern ) {
118 :
119 :             中略
120 :
121 :             case 0:
122 :                 /* スイッチ入力待ち */
123 :                 if( pushsw_get() ) {
124 :                     clearI2CEeprom(); /* 数秒かかる */
125 :                     pattern = 1;
126 :                     cnt1 = 0;
127 :                     break;
128 :                 }
129 :                 if( cnt1 < 100 ) { /* LED点滅処理 */
130 :                     led_out( 0x1 );
131 :                 } else if( cnt1 < 200 ) {
132 :                     led_out( 0x2 );
133 :                 } else {
134 :                     cnt1 = 0;
135 :                 }
136 :                 break;
137 :
138 :             case 1:
139 :                 /* スタートバーが開いたかチェック */
140 :                 if( !startbar_get() ) {
141 :                     /* スタート!! */
142 :                     led_out( 0x0 );
143 :                     pattern = 11;
144 :                     saveIndex = 0;
145 :                     saveFlag = 1; /* データ保存開始 */
146 :                     cnt1 = 0;
147 :                     break;
148 :                 }
149 :                 if( cnt1 < 50 ) { /* LED点滅処理 */
150 :                     led_out( 0x1 );
151 :                 } else if( cnt1 < 100 ) {
152 :                     led_out( 0x2 );
153 :                 } else {
154 :                     cnt1 = 0;
155 :                 }
156 :                 break;
157 :
158 :             case 2:
159 :                 /* ... */
160 :                 break;
161 :
162 :             case 3:
163 :                 /* ... */
164 :                 break;
165 :
166 :             case 4:
167 :                 /* ... */
168 :                 break;
169 :
170 :             case 5:
171 :                 /* ... */
172 :                 break;
173 :
174 :             case 6:
175 :                 /* ... */
176 :                 break;
177 :
178 :             case 7:
179 :                 /* ... */
180 :                 break;
181 :
182 :             case 8:
183 :                 /* ... */
184 :                 break;
185 :
186 :             case 9:
187 :                 /* ... */
188 :                 break;
189 :
190 :             case 10:
191 :                 /* ... */
192 :                 break;
193 :
194 :             case 11:
195 :                 /* ... */
196 :                 break;
197 :
198 :             case 12:
199 :                 /* ... */
200 :                 break;
201 :
202 :             case 13:
203 :                 /* ... */
204 :                 break;
205 :
206 :             case 14:
207 :                 /* ... */
208 :                 break;
209 :
210 :             case 15:
211 :                 /* ... */
212 :                 break;
213 :
214 :             case 16:
215 :                 /* ... */
216 :                 break;
217 :
218 :             case 17:
219 :                 /* ... */
220 :                 break;
221 :
222 :             case 18:
223 :                 /* ... */
224 :                 break;
225 :
226 :             case 19:
227 :                 /* ... */
228 :                 break;
229 :
230 :             case 20:
231 :                 /* ... */
232 :                 break;
233 :
234 :             case 21:
235 :                 /* ... */
236 :                 break;
237 :
238 :             case 22:
239 :                 /* ... */
240 :                 break;
241 :
242 :             case 23:
243 :                 /* ... */
244 :                 break;
245 :
246 :             case 24:
247 :                 /* ... */
248 :                 break;
249 :
250 :             case 25:
251 :                 /* ... */
252 :                 break;
253 :
254 :             case 26:
255 :                 /* ... */
256 :                 break;
257 :
258 :             case 27:
259 :                 /* ... */
260 :                 break;
261 :
262 :             case 28:
263 :                 /* ... */
264 :                 break;
265 :
266 :             case 29:
267 :                 /* ... */
268 :                 break;
269 :
270 :             case 30:
271 :                 /* ... */
272 :                 break;
273 :
274 :             case 31:
275 :                 /* ... */
276 :                 break;
277 :
278 :             case 32:
279 :                 /* ... */
280 :                 break;
281 :
282 :             case 33:
283 :                 /* ... */
284 :                 break;
285 :
286 :             case 34:
287 :                 /* ... */
288 :                 break;
289 :
290 :             case 35:
291 :                 /* ... */
292 :                 break;
293 :
294 :             case 36:
295 :                 /* ... */
296 :                 break;
297 :
298 :             case 37:
299 :                 /* ... */
300 :                 break;
301 :
302 :             case 38:
303 :                 /* ... */
304 :                 break;
305 :
306 :             case 39:
307 :                 /* ... */
308 :                 break;
309 :
310 :             case 40:
311 :                 /* ... */
312 :                 break;
313 :
314 :             case 41:
315 :                 /* ... */
316 :                 break;
317 :
318 :             case 42:
319 :                 /* ... */
320 :                 break;
321 :
322 :             case 43:
323 :                 /* ... */
324 :                 break;
325 :
326 :             case 44:
327 :                 /* ... */
328 :                 break;
329 :
330 :             case 45:
331 :                 /* ... */
332 :                 break;
333 :
334 :             case 46:
335 :                 /* ... */
336 :                 break;
337 :
338 :             case 47:
339 :                 /* ... */
340 :                 break;
341 :
342 :             case 48:
343 :                 /* ... */
344 :                 break;
345 :
346 :             case 49:
347 :                 /* ... */
348 :                 break;
349 :
350 :             case 50:
351 :                 /* ... */
352 :                 break;
353 :
354 :             case 51:
355 :                 /* ... */
356 :                 break;
357 :
358 :             case 52:
359 :                 /* ... */
360 :                 break;
361 :
362 :             case 53:
363 :                 /* ... */
364 :                 break;
365 :
366 :             case 54:
367 :                 /* ... */
368 :                 break;
369 :
370 :             case 55:
371 :                 /* ... */
372 :                 break;
373 :
374 :             case 56:
375 :                 /* ... */
376 :                 break;
377 :
378 :             case 57:
379 :                 /* ... */
380 :                 break;
381 :
382 :             case 58:
383 :                 /* ... */
384 :                 break;
385 :
386 :             case 59:
387 :                 /* ... */
388 :                 break;
389 :
390 :             case 60:
391 :                 /* ... */
392 :                 break;
393 :
394 :             case 61:
395 :                 /* ... */
396 :                 break;
397 :
398 :             case 62:
399 :                 /* ... */
400 :                 break;
401 :
402 :             case 63:
403 :                 /* ... */
404 :                 break;
405 :
406 :             case 64:
407 :                 /* ... */
408 :                 break;
409 :
410 :             case 65:
411 :                 /* ... */
412 :                 break;
413 :
414 :             case 66:
415 :                 /* ... */
416 :                 break;
417 :
418 :             case 67:
419 :                 /* ... */
420 :                 break;
421 :
422 :             case 68:
423 :                 /* ... */
424 :                 break;
425 :
426 :             case 69:
427 :                 /* ... */
428 :                 break;
429 :
430 :             case 70:
431 :                 /* ... */
432 :                 break;
433 :
434 :             case 71:
435 :                 /* ... */
436 :                 break;
437 :
438 :             case 72:
439 :                 /* ... */
440 :                 break;
441 :
442 :             case 73:
443 :                 /* ... */
444 :                 break;
445 :
446 :             case 74:
447 :                 /* ... */
448 :                 break;
449 :
450 :             case 75:
451 :                 /* ... */
452 :                 break;
453 :
454 :             case 76:
455 :                 /* ... */
456 :                 break;
457 :
458 :             case 77:
459 :                 /* ... */
460 :                 break;
461 :
462 :             case 78:
463 :                 /* ... */
464 :                 break;
465 :
466 :             case 79:
467 :                 /* ... */
468 :                 break;
469 :
470 :             case 80:
471 :                 /* ... */
472 :                 break;
473 :
474 :             case 81:
475 :                 /* ... */
476 :                 break;
477 :
478 :             case 82:
479 :                 /* ... */
480 :                 break;
481 :
482 :             case 83:
483 :                 /* ... */
484 :                 break;
485 :
486 :             case 84:
487 :                 /* ... */
488 :                 break;
489 :
490 :             case 85:
491 :                 /* ... */
492 :                 break;
493 :
494 :             case 86:
495 :                 /* ... */
496 :                 break;
497 :
498 :             case 87:
499 :                 /* ... */
500 :                 break;
501 :
502 :             case 88:
503 :                 /* ... */
504 :                 break;
505 :
506 :             case 89:
507 :                 /* ... */
508 :                 break;
509 :
510 :             case 90:
511 :                 /* ... */
512 :                 break;
513 :
514 :             case 91:
515 :                 /* ... */
516 :                 break;
517 :
518 :             case 92:
519 :                 /* ... */
520 :                 break;
521 :
522 :             case 93:
523 :                 /* ... */
524 :                 break;
525 :
526 :             case 94:
527 :                 /* ... */
528 :                 break;
529 :
530 :             case 95:
531 :                 /* ... */
532 :                 break;
533 :
534 :             case 96:
535 :                 /* ... */
536 :                 break;
537 :
538 :             case 97:
539 :                 /* ... */
540 :                 break;
541 :
542 :             case 98:
543 :                 /* ... */
544 :                 break;
545 :
546 :             case 99:
547 :                 /* ... */
548 :                 break;
549 :
550 :             case 100:
551 :                 /* ... */
552 :                 break;
553 :
554 :             case 101:
555 :                 /* ... */
556 :                 break;
557 :
558 :             case 102:
559 :                 /* ... */
560 :                 break;
561 :
562 :             case 103:
563 :                 /* ... */
564 :                 break;
565 :
566 :             case 104:
567 :                 /* ... */
568 :                 break;
569 :
570 :             case 105:
571 :                 /* ... */
572 :                 break;
573 :
574 :             case 106:
575 :                 /* ... */
576 :                 break;
577 :
578 :             case 107:
579 :                 /* ... */
580 :                 break;
581 :
582 :             case 108:
583 :                 /* ... */
584 :                 break;
585 :
586 :             case 109:
587 :                 /* ... */
588 :                 break;
589 :
590 :             case 110:
591 :                 /* ... */
592 :                 break;
593 :
594 :             case 111:
595 :                 /* ... */
596 :                 break;
597 :
598 :             case 112:
599 :                 /* ... */
600 :                 break;
601 :
602 :             case 113:
603 :                 /* ... */
604 :                 break;
605 :
606 :             case 114:
607 :                 /* ... */
608 :                 break;
609 :
610 :             case 115:
611 :                 /* ... */
612 :                 break;
613 :
614 :             case 116:
615 :                 /* ... */
616 :                 break;
617 :
618 :             case 117:
619 :                 /* ... */
620 :                 break;
621 :
622 :             case 118:
623 :                 /* ... */
624 :                 break;
625 :
626 :             case 119:
627 :                 /* ... */
628 :                 break;
629 :
630 :             case 120:
631 :                 /* ... */
632 :                 break;
633 :
634 :             case 121:
635 :                 /* ... */
636 :                 break;
637 :
638 :             case 122:
639 :                 /* ... */
640 :                 break;
641 :
642 :             case 123:
643 :                 /* ... */
644 :                 break;
645 :
646 :             case 124:
647 :                 /* ... */
648 :                 break;
649 :
650 :             case 125:
651 :                 /* ... */
652 :                 break;
653 :
654 :             case 126:
655 :                 /* ... */
656 :                 break;
657 :
658 :             case 127:
659 :                 /* ... */
660 :                 break;
661 :
662 :             case 128:
663 :                 /* ... */
664 :                 break;
665 :
666 :             case 129:
667 :                 /* ... */
668 :                 break;
669 :
670 :             case 130:
671 :                 /* ... */
672 :                 break;
673 :
674 :             case 131:
675 :                 /* ... */
676 :                 break;
677 :
678 :             case 132:
679 :                 /* ... */
680 :                 break;
681 :
682 :             case 133:
683 :                 /* ... */
684 :                 break;
685 :
686 :             case 134:
687 :                 /* ... */
688 :                 break;
689 :
690 :             case 135:
691 :                 /* ... */
692 :                 break;
693 :
694 :             case 136:
695 :                 /* ... */
696 :                 break;
697 :
698 :             case 137:
699 :                 /* ... */
700 :                 break;
701 :
702 :             case 138:
703 :                 /* ... */
704 :                 break;
705 :
706 :             case 139:
707 :                 /* ... */
708 :                 break;
709 :
710 :             case 140:
711 :                 /* ... */
712 :                 break;
713 :
714 :             case 141:
715 :                 /* ... */
716 :                 break;
717 :
718 :             case 142:
719 :                 /* ... */
720 :                 break;
721 :
722 :             case 143:
723 :                 /* ... */
724 :                 break;
725 :
726 :             case 144:
727 :                 /* ... */
728 :                 break;
729 :
730 :             case 145:
731 :                 /* ... */
732 :                 break;
733 :
734 :             case 146:
735 :                 /* ... */
736 :                 break;
737 :
738 :             case 147:
739 :                 /* ... */
740 :                 break;
741 :
742 :             case 148:
743 :                 /* ... */
744 :                 break;
745 :
746 :             case 149:
747 :                 /* ... */
748 :                 break;
749 :
750 :             case 150:
751 :                 /* ... */
752 :                 break;
753 :
754 :             case 151:
755 :                 /* ... */
756 :                 break;
757 :
758 :             case 152:
759 :                 /* ... */
760 :                 break;
761 :
762 :             case 153:
763 :                 /* ... */
764 :                 break;
765 :
766 :             case 154:
767 :                 /* ... */
768 :                 break;
769 :
770 :             case 155:
771 :                 /* ... */
772 :                 break;
773 :
774 :             case 156:
775 :                 /* ... */
776 :                 break;
777 :
778 :             case 157:
779 :                 /* ... */
780 :                 break;
781 :
782 :             case 158:
783 :                 /* ... */
784 :                 break;
785 :
786 :             case 159:
787 :                 /* ... */
788 :                 break;
789 :
790 :             case 160:
791 :                 /* ... */
792 :                 break;
793 :
794 :             case 161:
795 :                 /* ... */
796 :                 break;
797 :
798 :             case 162:
799 :                 /* ... */
800 :                 break;
801 :
799 :             case 163:
800 :                 /* ... */
801 :                 break;
802 :
803 :             case 164:
804 :                 /* ... */
805 :                 break;
806 :
807 :             case 165:
808 :                 /* ... */
809 :                 break;
810 :
811 :             case 166:
812 :                 /* ... */
813 :                 break;
814 :
815 :             case 167:
816 :                 /* ... */
817 :                 break;
818 :
819 :             case 168:
820 :                 /* ... */
821 :                 break;
822 :
823 :             case 169:
824 :                 /* ... */
825 :                 break;
826 :
827 :             case 170:
828 :                 /* ... */
829 :                 break;
830 :
831 :             case 171:
832 :                 /* ... */
833 :                 break;
834 :
835 :             case 172:
836 :                 /* ... */
837 :                 break;
838 :
839 :             case 173:
840 :                 /* ... */
841 :                 break;
842 :
843 :             case 174:
844 :                 /* ... */
845 :                 break;
846 :
847 :             case 175:
848 :                 /* ... */
849 :                 break;
850 :
851 :             case 176:
852 :                 /* ... */
853 :                 break;
854 :
855 :             case 177:
856 :                 /* ... */
857 :                 break;
858 :
859 :             case 178:
860 :                 /* ... */
861 :                 break;
862 :
863 :             case 179:
864 :                 /* ... */
865 :                 break;
866 :
867 :             case 180:
868 :                 /* ... */
869 :                 break;
870 :
871 :             case 181:
872 :                 /* ... */
873 :                 break;
874 :
875 :             case 182:
876 :                 /* ... */
877 :                 break;
878 :
879 :             case 183:
880 :                 /* ... */
881 :                 break;
882 :
883 :             case 184:
884 :                 /* ... */
885 :                 break;
886 :
887 :             case 185:
888 :                 /* ... */
889 :                 break;
890 :
891 :             case 186:
892 :                 /* ... */
893 :                 break;
894 :
895 :             case 187:
896 :                 /* ... */
897 :                 break;
898 :
899 :             case 188:
900 :                 /* ... */
901 :                 break;
902 :
903 :             case 189:
904 :                 /* ... */
905 :                 break;
906 :
907 :             case 190:
908 :                 /* ... */
909 :                 break;
910 :
911 :             case 191:
912 :                 /* ... */
913 :                 break;
914 :
915 :             case 192:
916 :                 /* ... */
917 :                 break;
918 :
919 :             case 193:
920 :                 /* ... */
921 :                 break;
922 :
923 :             case 194:
924 :                 /* ... */
925 :                 break;
926 :
927 :             case 195:
928 :                 /* ... */
929 :                 break;
930 :
931 :             case 196:
932 :                 /* ... */
933 :                 break;
934 :
935 :             case 197:
936 :                 /* ... */
937 :                 break;
938 :
939 :             case 198:
940 :                 /* ... */
941 :                 break;
942 :
943 :             case 199:
944 :                 /* ... */
945 :                 break;
946 :
947 :             case 200:
948 :                 /* ... */
949 :                 break;
950 :
951 :             case 201:
952 :                 /* ... */
953 :                 break;
954 :
955 :             case 202:
956 :                 /* ... */
957 :                 break;
958 :
959 :             case 203:
960 :                 /* ... */
961 :                 break;
962 :
963 :             case 204:
964 :                 /* ... */
965 :                 break;
966 :
967 :             case 205:
968 :                 /* ... */
969 :                 break;
970 :
971 :             case 206:
972 :                 /* ... */
973 :                 break;
974 :
975 :             case 207:
976 :                 /* ... */
977 :                 break;
978 :
979 :             case 208:
980 :                 /* ... */
981 :                 break;
982 :
983 :             case 209:
984 :                 /* ... */
985 :                 break;
986 :
987 :             case 210:
988 :                 /* ... */
989 :                 break;
990 :
991 :             case 211:
992 :                 /* ... */
993 :                 break;
994 :
995 :             case 212:
996 :                 /* ... */
997 :                 break;
998 :
999 :             case 213:
1000 :                 /* ... */
1001 :                 break;
1002 :
1003 :             case 214:
1004 :                 /* ... */
1005 :                 break;
1006 :
1007 :             case 215:
1008 :                 /* ... */
1009 :                 break;
1010 :
1011 :             case 216:
1012 :                 /* ... */
1013 :                 break;
1014 :
1015 :             case 217:
1016 :                 /* ... */
1017 :                 break;
1018 :
1019 :             case 218:
1020 :                 /* ... */
1021 :                 break;
1022 :
1023 :             case 219:
1024 :                 /* ... */
1025 :                 break;
1026 :
1027 :             case 220:
1028 :                 /* ... */
1029 :                 break;
1030 :
1031 :             case 221:
1032 :                 /* ... */
1033 :                 break;
1034 :
1035 :             case 222:
1036 :                 /* ... */
1037 :                 break;
1038 :
1039 :             case 223:
1040 :                 /* ... */
1041 :                 break;
1042 :
1043 :             case 224:
1044 :                 /* ... */
1045 :                 break;
1046 :
1047 :             case 225:
1048 :                 /* ... */
1049 :                 break;
1050 :
1051 :             case 226:
1052 :                 /* ... */
1053 :                 break;
1054 :
1055 :             case 227:
1056 :                 /* ... */
1057 :                 break;
1058 :
1059 :             case 228:
1060 :                 /* ... */
1061 :                 break;
1062 :
1063 :             case 229:
1064 :                 /* ... */
1065 :                 break;
1066 :
1067 :             case 230:
1068 :                 /* ... */
1069 :                 break;
1070 :
1071 :             case 231:
1072 :                 /* ... */
1073 :                 break;
1074 :
1075 :             case 232:
1076 :                 /* ... */
1077 :                 break;
1078 :
1079 :             case 233:
1080 :                 /* ... */
1081 :                 break;
1082 :
1083 :             case 234:
1084 :                 /* ... */
1085 :                 break;
1086 :
1087 :             case 235:
1088 :                 /* ... */
1089 :                 break;
1090 :
1091 :             case 236:
1092 :                 /* ... */
1093 :                 break;
1094 :
1095 :             case 237:
1096 :                 /* ... */
1097 :                 break;
1098 :
1099 :             case 238:
1100 :                 /* ... */
1101 :                 break;
1102 :
1103 :             case 239:
1104 :                 /* ... */
1105 :                 break;
1106 :
1107 :             case 240:
1108 :                 /* ... */
1109 :                 break;
1109 :
1110 :             case 241:
1111 :                 /* ... */
1112 :                 break;
1113 :
1114 :             case 242:
1115 :                 /* ... */
1116 :                 break;
1115 :
1116 :             case 243:
1117 :                 /* ... */
1118 :                 break;
1117 :
1118 :             case 244:
1119 :                 /* ... */
1120 :                 break;
1119 :
1120 :             case 245:
1121 :                 /* ... */
1122 :                 break;
1121 :
1122 :             case 246:
1123 :                 /* ... */
1124 :                 break;
1123 :
1124 :             case 247:
1125 :                 /* ... */
1126 :                 break;
1125 :
1126 :             case 248:
1127 :                 /* ... */
1128 :                 break;
1127 :
1128 :             case 249:
1129 :                 /* ... */
1130 :                 break;
1129 :
1130 :             case 250:
1131 :                 /* ... */
1132 :                 break;
1131 :
1132 :             case 251:
1133 :                 /* ... */
1134 :                 break;
1133 :
1134 :             case 252:
1135 :                 /* ... */
1136 :                 break;
1135 :
1136 :             case 253:
1137 :                 /* ... */
1138 :                 break;
1137 :
1138 :             case 254:
1139 :                 /* ... */
1140 :                 break;
1139 :
1140 :             case 255:
1141 :                 /* ... */
1142 :                 break;
1141 :
1142 :             case 256:
1143 :                 /* ... */
1144 :                 break;
1143 :
1144 :             case 257:
1145 :                 /* ... */
1146 :                 break;
1145 :
1146 :             case 258:
1147 :                 /* ... */
1148 :                 break;
1147 :
1148 :             case 259:
1149 :                 /* ... */
1150 :                 break;
1149 :
1150 :             case 260:
1151 :                 /* ... */
1152 :                 break;
1151 :
1152 :             case 261:
1153 :                 /* ... */
1154 :                 break;
1153 :
1154 :             case 262:
1155 :                 /* ... */
1156 :                 break;
1155 :
1156 :             case 263:
1157 :                 /* ... */
1158 :                 break;
1157 :
1158 :             case 264:
1159 :                 /* ... */
1160 :                 break;
1159 :
1160 :             case 265:
1161 :                 /* ... */
1162 :                 break;
1161 :
1162 :             case 266:
1163 :                 /* ... */
1164 :                 break;
1163 :
1164 :             case 267:
1165 :                 /* ... */
1166 :                 break;
1165 :
1166 :             case 268:
1167 :                 /* ... */
1168 :                 break;
1167 :
1168 :             case 269:
1169 :                 /* ... */
1170 :                 break;
1169 :
1170 :             case 270:
1171 :                 /* ... */
1172 :                 break;
1171 :
1172 :             case 271:
1173 :                 /* ... */
1174 :                 break;
1173 :
1174 :             case 272:
1175 :                 /* ... */
1176 :                 break;
1175 :
1176 :             case 273:
1177 :                 /* ... */
1178 :                 break;
1177 :
1178 :             case 274:
1179 :                 /* ... */
1180 :                 break;
1179 :
1180 :             case 275:
1181 :                 /* ... */
1182 :                 break;
1181 :
1182 :             case 276:
1183 :                 /* ... */
1184 :                 break;
1183 :
1184 :             case 277:
1185 :                 /* ... */
1186 :                 break;
1185 :
1186 :             case 278:
1187 :                 /* ... */
1188 :                 break;
1187 :
1188 :             case 279:
1189 :                 /* ... */
1190 :                 break;
1189 :
1190 :             case 280:
1191 :                 /* ... */
1192 :                 break;
1191 :
1192 :             case 281:
1193 :                 /* ... */
1194 :                 break;
1193 :
1194 :             case 282:
1195 :                 /* ... */
1196 :                 break;
1195 :
1196 :             case 283:
1197 :                 /* ... */
1198 :                 break;
1197 :
1198 :             case 284:
1199 :                 /* ... */
1200 :                 break;
1199 :
1200 :             case 285:
1201 :                 /* ... */
1202 :                 break;
1201 :
1202 :             case 286:
1203 :                 /* ... */
1204 :                 break;
1203 :
1204 :             case 287:
1205 :                 /* ... */
1206 :                 break;
1205 :
1206 :             case 288:
1207 :                 /* ... */
1208 :                 break;
1207 :
1208 :             case 289:
1209 :                 /* ... */
1210 :                 break;
1209 :
1210 :             case 290:
1211 :                 /* ... */
1212 :                 break;
1211 :
1212 :             case 291:
1213 :                 /* ... */
1214 :                 break;
1213 :
1214 :             case 292:
1215 :                 /* ... */
1216 :                 break;
1215 :
1216 :             case 293:
1217 :                 /* ... */
1218 :                 break;
1217 :
1218 :             case 294:
1219 :                 /* ... */
1220 :                 break;
1219 :
1220 :             case 295:
1221 :                 /* ... */
1222 :                 break;
1221 :
1222 :             case 296:
1223 :                 /* ... */
1224 :                 break;
1223 :
1224 :             case 297:
1225 :                 /* ... */
1226 :                 break;
1225 :
1226 :             case 298:
1227 :                 /* ... */
1228 :                 break;
1227 :
1228 :             case 299:
1229 :                 /* ... */
1230 :                 break;
1229 :
1230 :             case 300:
1231 :                 /* ... */
1232 :                 break;
1231 :
1232 :             case 301:
1233 :                 /* ... */
1234 :                 break;
1233 :
1234 :             case 302:
1235 :                 /* ... */
1236 :                 break;
1235 :
1236 :             case 303:
1237 :                 /* ... */
1238 :                 break;
1237 :
1238 :             case 304:
1239 :                 /* ... */
1240 :                 break;
1239 :
1240 :             case 305:
1241 :                 /* ... */
1242 :                 break;
1241 :
1242 :             case 306:
1243 :                 /* ... */
1244 :                 break;
1243 :
1244 :             case 307:
1245 :                 /* ... */
1246 :                 break;
1245 :
1246 :             case 308:
1247 :                 /* ... */
1248 :                 break;
1247 :
1248 :             case 309:
1249 :                 /* ... */
1250 :                 break;
1249 :
1250 :             case 310:
1251 :                 /* ... */
1252 :                 break;
1251 :
1252 :             case 311:
1253 :                 /* ... */
1254 :                 break;
1253 :
1254 :             case 312:
1255 :                 /* ... */
1256 :                 break;
1255 :
1256 :             case 313:
1257 :                 /* ... */
1258 :                 break;
1257 :
1258 :             case 314:
1259 :                 /* ... */
1260 :                 break;
1259 :
1260 :             case 315:
1261 :                 /* ... */
1262 :                 break;
1261 :
1262 :             case 316:
1263 :                 /* ... */
1264 :                 break;
1263 :
1264 :             case 317:
1265 :                 /* ... */
1266 :                 break;
1265 :
1266 :             case 318:
1267 :                 /* ... */
1268 :                 break;
1267 :
1268 :             case 319:
1269 :                 /* ... */
1270 :                 break;
1269 :
1270 :             case 320:
1271 :                 /* ... */
1272 :                 break;
1271 :
1272 :             case 321:
1273 :                 /* ... */
1274 :                 break;
1273 :
```

```

178 :         break;
179 :
180 :     case 11:
181 :         /* 通常トレース */
182 :         if( pushsw_get() ) {
183 :             pattern = 71;          /* データ転送処理へ */
184 :             break;
185 :         }
186 :
187 :         if( check_crossline() ) { /* クロスラインチェック */
188 :             pattern = 21;
189 :             break;
190 :         }
191 :         if( check_rightline() ) { /* 右ハーフラインチェック */
192 :             pattern = 51;
193 :             break;
194 :         }
195 :         if( check_leftline() ) { /* 左ハーフラインチェック */
196 :             pattern = 61;
197 :             break;
198 :         }
199 :         switch( sensor_inp(MASK3_3) ) {
200 :             case 0x00:
201 :                 /* センタ まっすぐ */
202 :                 handle( 0 );
203 :                 speed( 100, 100 );
204 :                 break;

```

パターン 11 のときにスイッチを押すと、パターン 71 へ移ります。パターン 71 は、データを転送する部分です。

中略

```

519 :     case 71:
520 :         /* 停止 */
521 :         handle( 0 );
522 :         speed( 0, 0 );
523 :         saveFlag = 0;
524 :         saveSendIndex = 0;
525 :         pattern = 72;
526 :         cnt1 = 0;
527 :         break;
528 :
529 :     case 72:
530 :         /* 1s待ち */
531 :         if( cnt1 > 1000 ) {
532 :             pattern = 73;
533 :             cnt1 = 0;
534 :         }
535 :         break;
536 :
537 :     case 73:
538 :         /* スイッチが離されたかチェック */
539 :         if( !pushsw_get() ) {
540 :             pattern = 74;
541 :             cnt1 = 0;
542 :         }
543 :         break;
544 :
545 :     case 74:
546 :         /* スイッチが押されたかチェック */
547 :         led_out( (cnt1/500) % 2 + 1 );
548 :         if( pushsw_get() ) {
549 :             pattern = 75;
550 :             cnt1 = 0;
551 :         }
552 :         break;
553 :
554 :     case 75:
555 :         /* タイトル転送、準備 */
556 :         printf( "\n" );
557 :         printf( "kit06rec_02 Data Out\n" );
558 :         printf( "Pattern, Sensor, ハンドル, 左モータ, 右モータ\n" );
559 :         pattern = 76;
560 :         break;
561 :
562 :     case 76:
563 :         /* データ転送 */
564 :         led_out( (saveSendIndex/32) % 2 + 1 ); /* LED点滅処理 */
565 :
566 :         /* 終わりのチェック */
567 :         if( (readI2CEeprom( saveSendIndex )==0) ||
568 :             (saveSendIndex >= 0x8000) ) {
569 :             pattern = 77;
570 :             cnt1 = 0;
571 :             break;
572 :         }
573 :
574 :         /* データの転送 */
575 :         convertHexToBin( readI2CEeprom(saveSendIndex+1), s );
576 :         printf( "%d,%8s,%d,%d,%d\n",
577 :             (char)readI2CEeprom( saveSendIndex+0 ), /* パターン */
578 :             s, /* センサ */
579 :             (char)readI2CEeprom( saveSendIndex+2 ), /* ハンドル */

```

パターン 71 以降は、保存したデータを転送する部分です。

まず、マイコンカーを停止させます。

saveFlag=0 でデータ保存を終了します。saveSendIndex は転送するアドレスを指定する変数です。0 から順番に転送するので、最初にクリアします。パターン 72 へ移ります。

単純に1秒待ちます。1 秒後、パターン 73 へ移ります。

スイッチが離されていれば、パターン 74 へ移ります。

スイッチが押されれば、パターン 75 へ移ります。

このとき、LED を点滅させ、転送待機状態と言うことを外部に知らせます。

タイトルを転送します。すぐに、パターン 76 へ移ります。

データがある限り、転送し続けます。転送するデータは、5 種類です。保存数を超過してデータを転送しようとする次のパターンへ移り、転送を終了します。

センサ値を 2 進数に変換します。

```

580 :         (char)readI2CEeprom( saveSendIndex+3 ), /* 左モータ */
581 :         (char)readI2CEeprom( saveSendIndex+4 ) /* 右モータ */
582 :     );
583 :
584 :     saveSendIndex += 8; /* 次の準備 */
585 :     break;
586 :
587 : case 77:
588 :     /* 転送終了 */
589 :     led_out( 0x3 );
590 :     break;
591 :
592 : default:
593 :     /* どれでもない場合は待機状態に戻す */
594 :     pattern = 0;
595 :     break;
596 : }
597 : }
598 : }
599 :
600 : /*****
601 : /* H8/3048F-ONE 内蔵周辺機能 初期化 */
602 : /*****/
603 : void init( void )
604 : {
605 :     /* I/Oポートの入出力設定 */
606 :     P1DDR = 0xff;
607 :     P2DDR = 0xff;
608 :     P3DDR = 0xff;
609 :     P4DDR = 0xff;
610 :     P5DDR = 0xff;
611 :     P6DDR = 0xf0; /* CPU基板上のDIP SW */
612 :     P8DDR = 0xff;
613 :     P9DDR = 0xf7; /* 通信ポート */
614 :     PADDR = 0x57; /* スタートバー検出センサ */
615 :     PBDR = 0xc0;
616 :     PBDDR = 0xfe; /* モータドライブ基板Vol.3 */
617 :     /* センサ基板のP7は、入力専用なので入出力設定はありません */
618 :
619 :     /* ITU0 1msごとの割り込み */
620 :     ITU0_TCR = 0x23;
621 :     ITU0_GRA = TIMER_CYCLE;
622 :     ITU0_IER = 0x01;
623 :
624 :     /* ITU3,4 リセット同期PWMモード 左右モータ、サーボ用 */
625 :     ITU3_TCR = 0x23;
626 :     ITU_FCR = 0x3e;
627 :     ITU3_GRA = PWM_CYCLE; /* 周期の設定 */
628 :     ITU3_GRB = ITU3_BRB = 0; /* 左モータのPWM設定 */
629 :     ITU4_GRA = ITU4_BRA = 0; /* 右モータのPWM設定 */
630 :     ITU4_GRB = ITU4_BRB = SERVO_CENTER; /* サーボのPWM設定 */
631 :     ITU_TOER = 0x38;
632 :
633 :     /* ITUのカウンタスタート */
634 :     ITU_STR = 0x09;
635 : }
636 :
637 : /*****
638 : /* ITU0 割り込み処理 */
639 : /*****/
640 : #pragma interrupt( interrupt_timer0 )
641 : void interrupt_timer0( void )
642 : {
643 :     ITU0_TSR &= 0xfe; /* フラグクリア */
644 :     cnt0++;
645 :     cnt1++;
646 :
647 :     /* データ保存関連 */
648 :     iTimer10++;
649 :     if( iTimer10 >= 10 ) {
650 :         iTimer10 = 0;
651 :         if( saveFlag ) {
652 :             saveData[0] = pattern; /* パターン */
653 :             saveData[1] = sensor_inp(0xff); /* センサ */
654 :             /* 2はハンドル関数内で保存 */
655 :             /* 3はモータ関数内で左モータPWM値保存 */
656 :             /* 4はモータ関数内で右モータPWM値保存 */
657 :             saveData[5] = 0; /* 予備 */
658 :             saveData[6] = 0; /* 予備 */
659 :             saveData[7] = 0; /* 予備 */
660 :             setPageWriteI2CEeprom( saveIndex, 8, saveData );
661 :             saveIndex += 8;
662 :             if( saveIndex >= 0x8000 ) saveFlag = 0;
663 :         }
664 :     }
665 : }

```

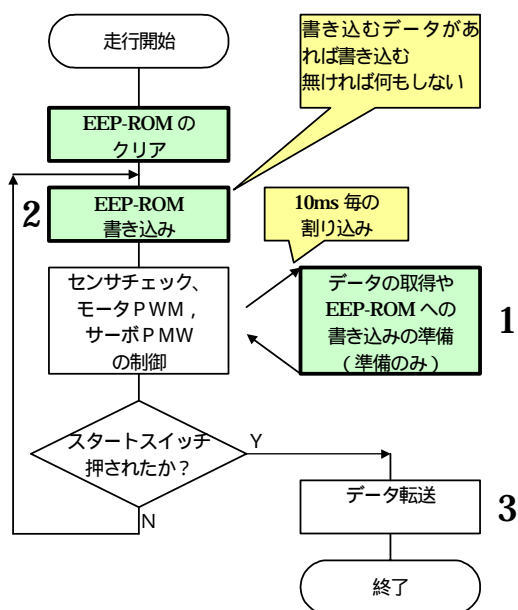
何もしません。電源が切られるのを待ちます。

ポートA には、
bit7:EEP-ROM SCL 端子
bit5:EEP-ROM SDA 端子
bit3:スタートバー検出センサ
が接続されています。上記3端子
はすべて入力です。
入出力設定は
01010111 0x57
となります。

10ms ごとに 8 バイト分のデータを
EEP-ROM に書き込みます。ただ
し、割り込み内では、書き込みの準
備をしているだけです。
EEP-ROM が一杯 になると、
saveFlag を 0 にして保存を強制的
に終了します。

以下、略

7.5 プログラムの概要



マイコンカーは、

- ・センサ(コースセンサ、エンコーダ)を見ながら
- ・駆動モータの制御
- ・サーボ(ステアリングモータ)の制御

を行っています。これらの走行に影響が無いようにデータ記録を行わなければいけません。

データの取得やEEP-ROMへの書き込みの準備は、10msごとに割り込みの中で行います。できるだけ早く命令を終わらせて走行に影響のないようにします(1部分)。

実際の書き込みも、走行に影響のないように少しずつ行います(2部分)。

このように、走行が最優先、その合間をぬってデータの書き込みを行います。

走行終了後、マイコンカーを取り上げてスタートスイッチを押します。パソコンとRS232Cケーブルを接続してデータを転送し、取得したデータがパソコンに取り込まれます(3部分)。

ポイントは、

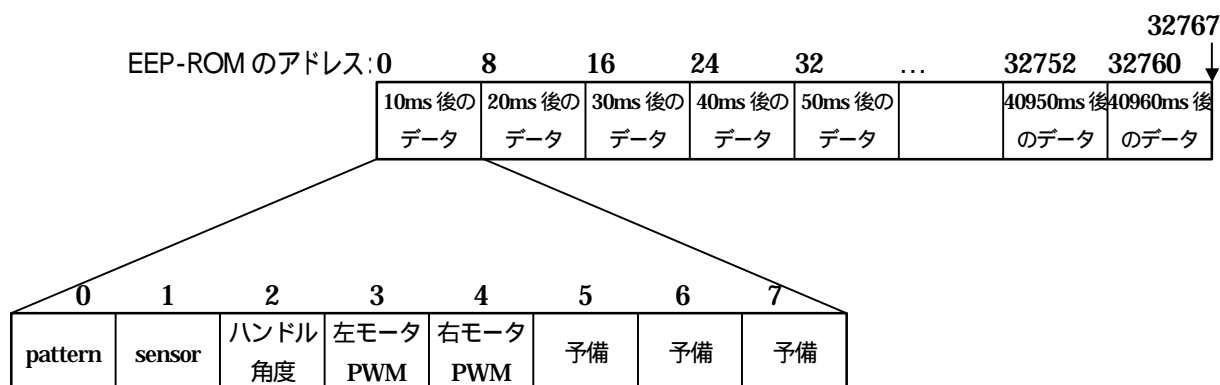
- ・EEP-ROMへの書き込みは遅いので、準備のみを行います(1部分)。
- ・無限ループ内で、あまり実行時間をかけないようにしながら少しずつ書き込みます(2部分)。
- ・EEP-ROMなので電源を切っても、保存データは消えません。
- ・保存するデータは、パターン番号、センサの値、ハンドル角度、左モータPWM値、右モータPWM値の5つです。

7.6 プログラムの解説

7.6.1 データの保存

EEP-ROMは32KBあります。データは5バイトですが、保存数は2のn乗個ごとなので8個保存します。保存間隔は10msです。保存できる時間は、

保存できる時間 = EEP-ROMの容量 32768 × 保存間隔 10ms ÷ 1回の保存数 8バイト = 40.960秒となります。



7.6.2 送信内容

```

562 :     case 76:
563 :         /* データ転送 */
564 :         led_out( (saveSendIndex/32) % 2 + 1 ); /* LED 点滅処理 */
565 :
566 :         /* 終わりのチェック */
567 :         if( (readI2CEeprom( saveSendIndex )==0) ||
568 :             (saveSendIndex >= 0x8000) ) {
569 :             pattern = 77;
570 :             cnt1 = 0;
571 :             break;
572 :         }
573 :
574 :         /* データの転送 */
575 :         convertHexToBin( readI2CEeprom(saveSendIndex+1), s );
576 :         printf( "%d,%8s,%d,%d,%d\n",
577 :             (char)readI2CEeprom( saveSendIndex+0 ), /* パターン */
578 :             s, /* センサ */
579 :             (char)readI2CEeprom( saveSendIndex+2 ), /* ハンドル */
580 :             (char)readI2CEeprom( saveSendIndex+3 ), /* 左モータ */
581 :             (char)readI2CEeprom( saveSendIndex+4 ) /* 右モータ */
582 :         );
583 :
584 :         saveSendIndex += 8; /* 次の準備 */
585 :         break;

```

送信内容は、576～582 行の内容です。

```

printf( "%d,%8s,%d,%d,%d\n",
    ↑   ↑   ↑   ↑   ↑
    (char)readI2CEeprom( saveSendIndex+0 ), /* パターン */
    s, /* センサ */
    (char)readI2CEeprom( saveSendIndex+2 ), /* ハンドル */
    (char)readI2CEeprom( saveSendIndex+3 ), /* 左モータ */
    (char)readI2CEeprom( saveSendIndex+4 ) /* 右モータ */

```

パターン、ハンドル、左モータ、右モータは 10 進数、センサは 8 文字の文字列で出力します。センサ値は、printf 文を実行する前に、convertHexToBin 関数で 2 進数に変換しています。その変換した文字列を出力するだけです。

7.7 プログラムの調整

まず初めに、「kit06rec_02.c」の下記内容を自分のマイコンカーに合わせて調整します。他にも、調整する部分は調整してください。

37 :	#define	SERVO_CENTER	5000	/* サーボのセンタ値	*/
中略					
321 :		handle(-38);		左クランクを曲がる時のハンドル角度	
中略					
330 :		handle(38);		右クランクを曲がる時のハンドル角度	

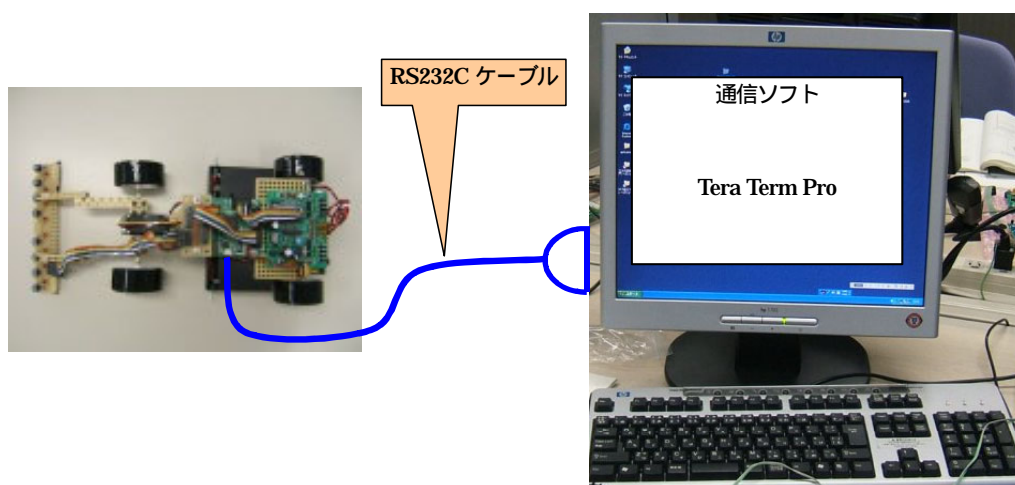
調整ができれば、プロジェクト「kit06rec_02」をビルドして、「kit06rec_02.mot」ファイルを CPU ボードに書き込みます。

7.8 走行からデータ転送までの流れ

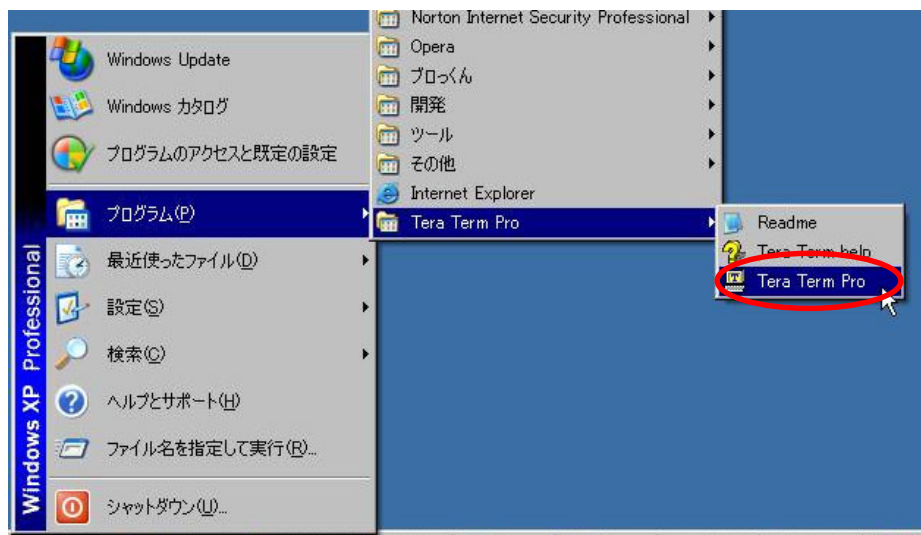
走行データを取りたい部分のコースを普通に走らせます。走行データが保存できるのは、スタートしてから約 40 秒です。

走らせた後、**電源を切ります**。EEP-ROM なので、**電源を切ってもデータは消えません**。

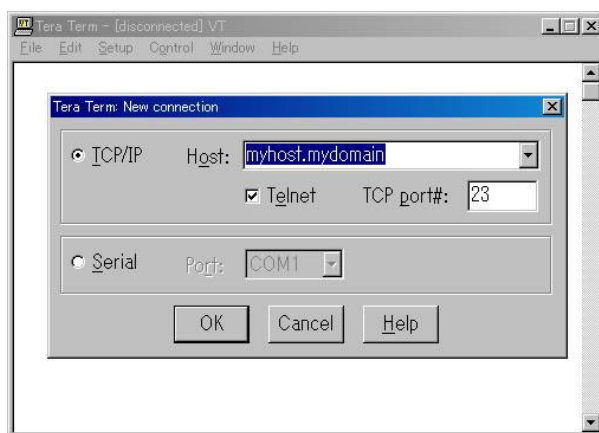
RS232C ケーブルをマイコンカーに接続します。



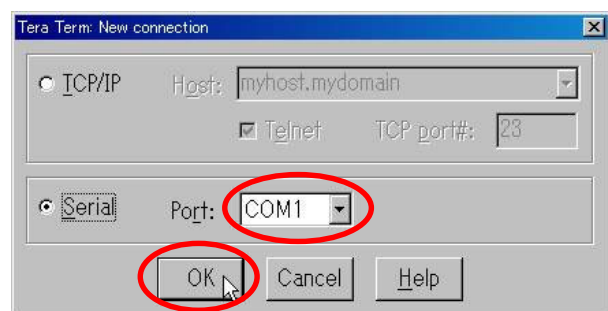
1. マイコンカーの電源は、まだ入れません。Tera Term Pro を立ち上げます。



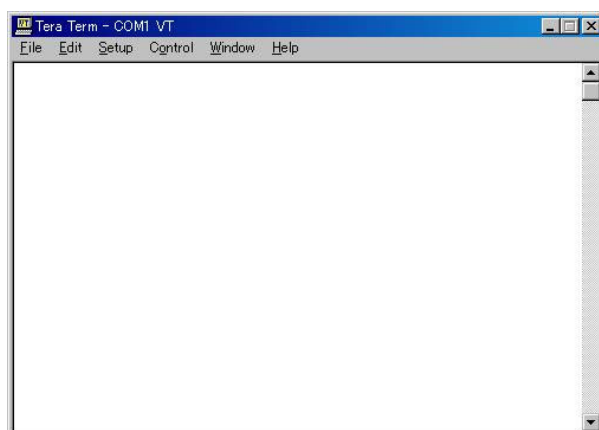
2. 「スタート すべてのプログラム(またはプログラム) Tera Term Pro Tera Term Pro」
で Tera Term Pro を立ち上げます。



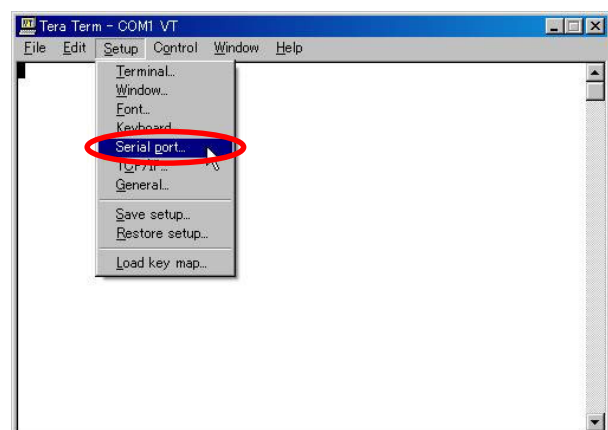
3. 最初に接続先を確認する画面が表示されます。



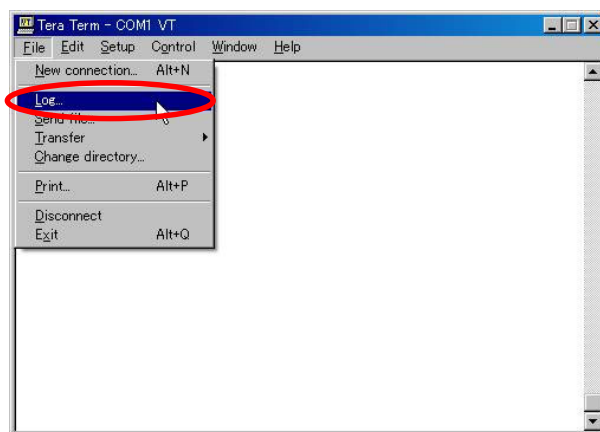
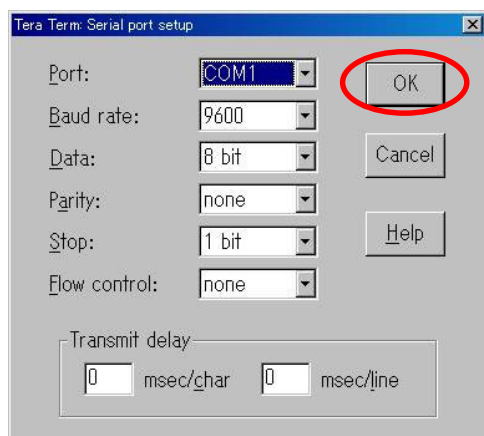
4. 「Serial」を選んで、各自のパソコンに合わせてポート番号を選びます。選択後、**OK**をクリック、次へ進みます。



5. 立ち上がりました。詳細設定をします。

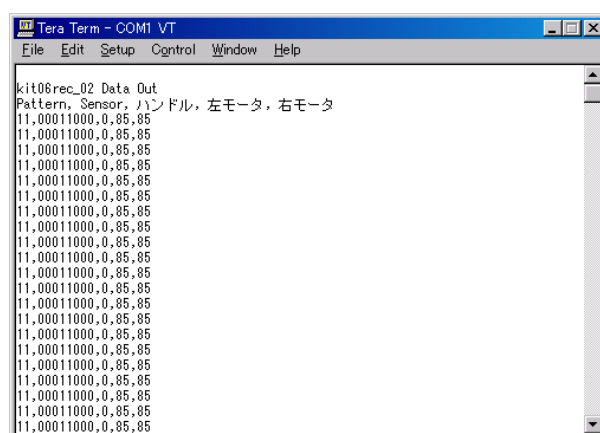
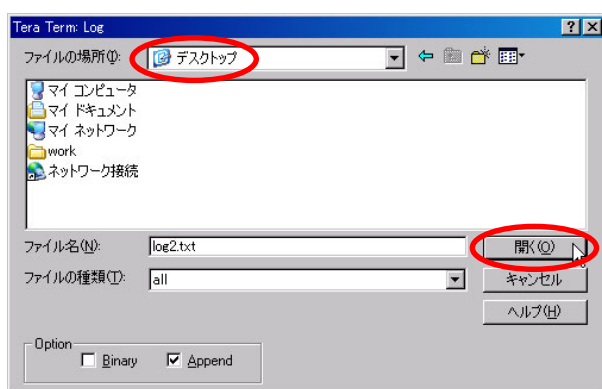


6. 「Setup Serial port」を選択します。



7. 通信設定を確認します。画面のように設定して、OK ボタンを押します。「Port」は各自のパソコンに合わせます。設定が出来たら **OK** をクリックします。

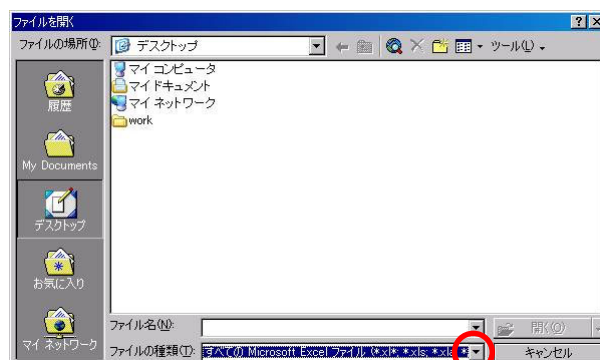
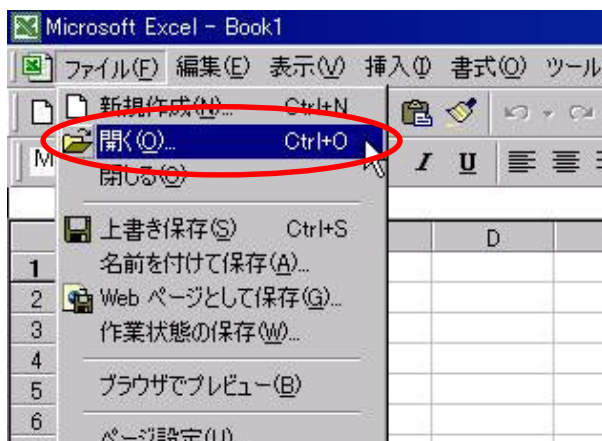
8. データをファイルに保存します。「File Log」を選択します。



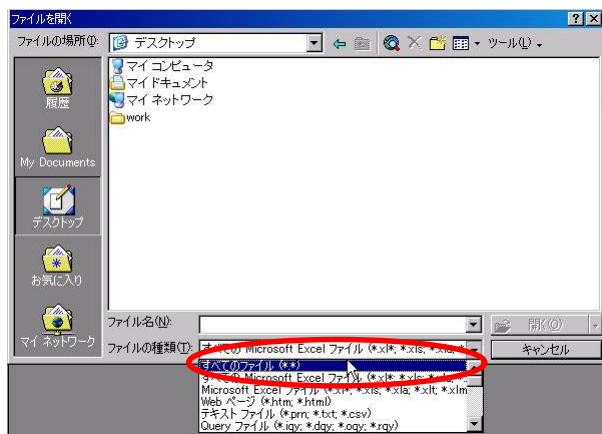
9. 「ファイルの場所」に保存するフォルダを選択します。「ファイル名」には、保存するファイルの名前を入力します。ここでは「log1.txt」と入力します。**開く** をクリックします。これで、パソコン側の準備は完了です。いよいよ、マイコンカーの操作をします。

10. マイコンカーのモータドライブ基板のスイッチを押しながらマイコンカーの電源を入れ、すぐにスイッチを離します。LED の点滅が遅くなっているはずですが、これが、データ転送モードの状態です。再度、スイッチを押すとデータが転送されます。転送中は、LED の点滅が高速になります。LED が 2 個とも点灯したら転送終了です。マイコンカーの電源を切って、TeraTermPro を終了してください。

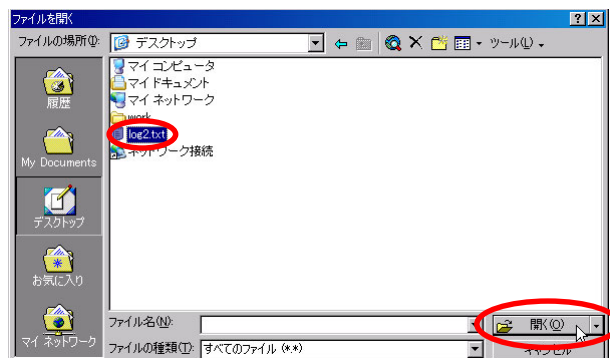
7.9 エクセルへの取り込み方



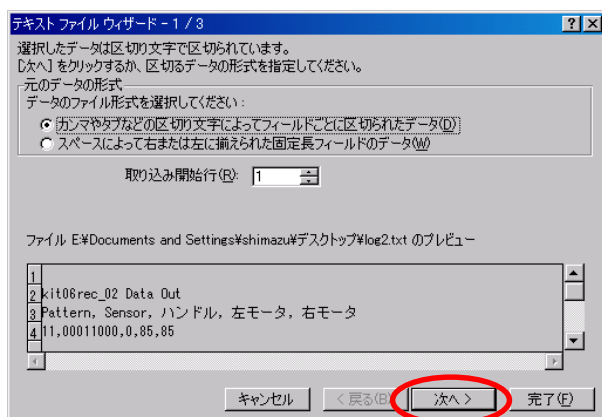
1. エクセルを立ち上げます。「ファイル 開く」を選択します。 2. 「ファイルの種類」の ☐ をクリックします。



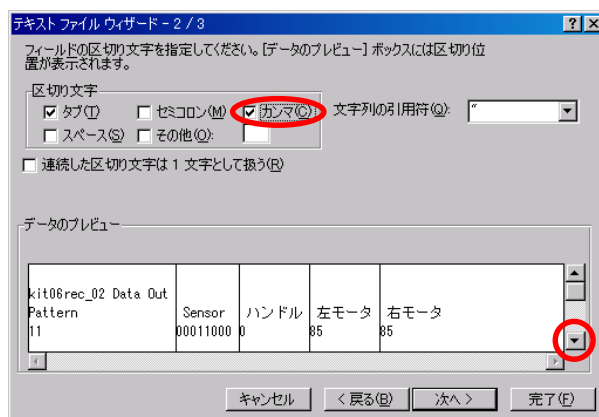
3. 「すべてのファイル (*.*)」を選択します。



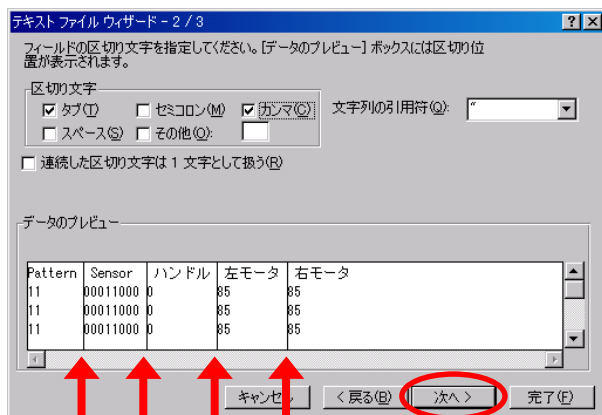
4. 「log2.txt」が表示されました。「log2.txt」を選択、**開く**をクリックします。



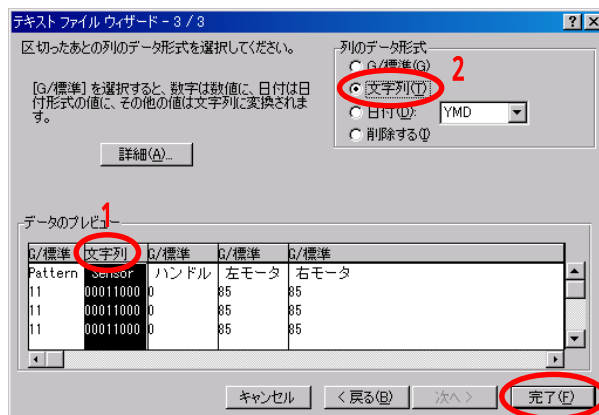
5. 開こうとしているデータは、テキストデータです。エクセルデータ(セル)に変換する作業を行います。**次へ**をクリックします。



6. データはカンマで区切られています。区切り文字の「カンマ」のチェックを付けます。**次へ**でデータのプレビュー欄を下げます。



7. 矢印()部分に縦線が入りました。これが列の区切りです。縦線が入ってなければ「区切り文字」のチェック指定がおかしいので確かめます。**次へ**をクリックします。



8. 次に、列のデータ形式を指定します。

- ・1 列目 パターン 数値
- ・2 列目 センサ値 2 進数 **文字列**
- ・3 列目 ハンドル 数値
- ・4 列目 左モータ 数値
- ・5 列目 右ハンドル 数値

2 列目のみ文字列に変換します。1部分をクリックします。2部分の「文字列」を選択します。**完了**をクリックして完了します。

	A	B	C	D	E	F
1						
2	kit06rec_02 Data Out					
3	Pattern	Sensor	ハンドル	左モータ	右モータ	
4	11	00011000	0	85	85	
5	11	00011000	0	85	85	
6	11	00011000	0	85	85	
7	11	00011000	0	85	85	
8	11	00011000	0	85	85	
9	11	00011000	0	85	85	
10	11	00011000	0	85	85	
11	11	00011000	0	85	85	
12	11	00011000	0	85	85	

9. セルに変換されました。解析してみます。

	パターン	センサ値	ハンドル	左モータ	右モータ	
178	11	00011000	0	85	85	
179	11	00011000	0	85	85	
180	11	00011000	0	85	85	
181	11	00011000	0	85	85	
182	22	11111111	0	0	0	← クロスライン 1 本目
183	22	11111111	0	0	0	
184	22	00011000	0	0	0	
185	22	11111000	0	0	0	← クロスライン 2 本目
186	22	11111111	0	0	0	
187	22	11111111	0	0	0	
188	22	00011000	0	0	0	
189	22	00011000	0	0	0	
190	22	00011000	0	0	0	

233	23	00011000	0	34	34	
234	23	00011000	0	34	34	
235	23	00011000	0	34	34	
236	31	11111000	-38	8	42	← 左クラック発見!!
237	31	11111000	-38	8	42	
238	31	11111000	-38	8	42	
239	31	11111000	-38	8	42	
240	31	11111000	-38	8	42	
241	31	11111000	-38	8	42	
242	31	11111000	-38	8	42	
243	31	11000000	-38	8	42	
244	31	00000000	-38	8	42	
245	31	00000000	-38	8	42	
246	31	00000000	-38	8	42	

339	32	10000011	-38	8	42	
340	32	10000001	-38	8	42	
341	32	10000001	-38	8	42	
342	32	10000001	-38	8	42	
343	32	11000001	-38	8	42	
344	32	11000001	-38	8	42	
345	32	11000000	-38	8	42	
346	32	11000000	-38	8	42	
347	32	11000000	-38	8	42	
348	32	11100000	-38	8	42	
349	32	11100000	-38	8	42	
350	11	01100000	-10	58	68	← 中心線へ復帰
351	11	01100000	-10	58	68	
352	11	01110000	-10	58	68	

8. プロジェクト「kit06rec_03」 エンコーダプログラムの追加

8.1 概要

本プログラムは、

・kit06rec_02.c...外付け EEP-ROM(24C256 32kB)にマイコンカーの走行

・kit06enc_03.c...エンコーダによる速度制御

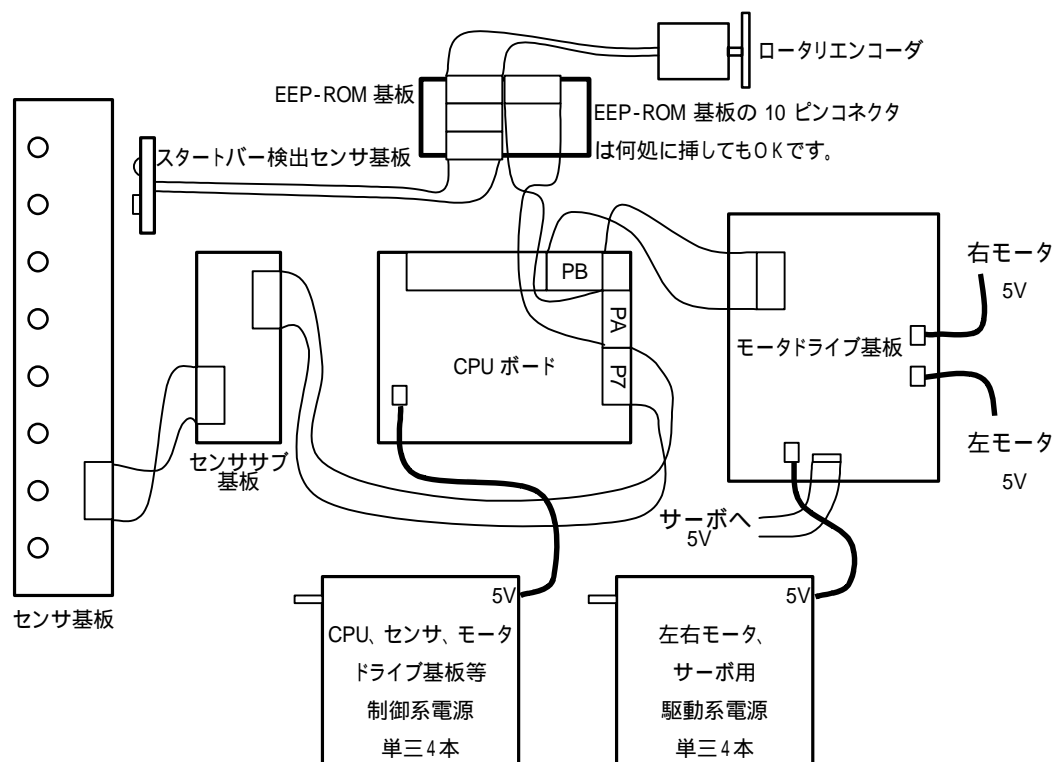
を合わせたプログラムです。マイコンカーの走行データを、外付け EEP-ROM に保存します。エンコーダ値も保存するので、速度が分かります。

kit06enc_03.c は、ワークスペース「kit06enc」のプロジェクト「kit06enc_03」のファイルです。詳しくは、エンコーダ実習マニュアルを参照してください。

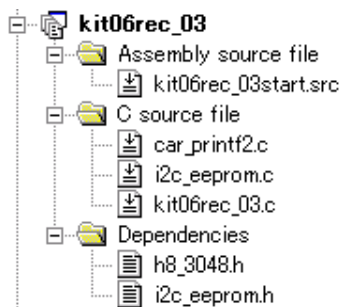
8.2 マイコンカーの構成

マイコンカーキット Vol.3 に、スタートバー検出センサ基板、EEP-ROM 基板を追加した構成です。LM350 追加セットで電池 8 本を直列に繋いでいる構成でも OK です。

- ・CPU ボードのポート 7 と、マイコンカーキット Vol.3 のセンサ基板を接続します。
- ・CPU ボードのポート B と、マイコンカーキット Vol.3 のモータドライブ基板 Vol.3 を接続します。
- ・CPU ボードのポート A と、EEP-ROM 基板を接続します。
- ・EEP-ROM 基板と、スタートバー検出センサ基板を接続します。
- ・EEP-ROM 基板と、ロータリエンコーダを接続します。



8.3 プロジェクトの構成



・kit06rec_03start.src

・kit06rec_03.c

・car_printf2.c

・i2c_eeprom.c

の 4 ファイルあります。

h8_3048.h は kit06rec_03.c、car_printf2.c、i2c_eeprom.c でインクルードされているファイルです。

i2c_eeprom.h は kit06rec_03.c、i2c_eeprom.c でインクルードされているファイルです。

8.4 プログラムの解説

8.4.1 入出力設定

ポート A の bit0 にロータリエンコーダを追加します。そのため、ポート A の bit0 を入力端子にします。ポート A の接続は下記ようになります。

bit	7	6	5	4	3	2	1	0
入力 or 出力	SCL 入力	出力	SDA 入力	出力	スタートパ ー検出セン サ入力	出力	出力	エンコー ダ入力

入力は"0"、出力は"1"とします。

bit	7	6	5	4	3	2	1	0
0 or 1	0	1	0	1	0	1	1	0

下記がプログラムの変更部分です。

```

633 : void init( void )
634 : {
635 :     /* I/O ポートの入出力設定 */
636 :     P1DDR = 0xff;
637 :     P2DDR = 0xff;
638 :     P3DDR = 0xff;
639 :     P4DDR = 0xff;
640 :     P5DDR = 0xff;
641 :     P6DDR = 0xf0;          /* CPU 基板上の DIP SW */
642 :     P8DDR = 0xff;
643 :     P9DDR = 0xf7;          /* 通信ポート */
644 :     PADDR = 0x56;        /* スタートパ-検出センサ */
645 :     PBDR  = 0xc0;
646 :     PBDDR = 0xfe;          /* モータドライブ基板 Vol.3 */

```

2 進数で"0101 0110"なので、16 進数で 0x56 となります。「kit06rec_03.c」は下記ようになります。

```

99 : void main( void )
100 : {
101 :     int    i;
102 :     char   s[8];
103 :
104 :     /* マイコン機能の初期化 */
105 :     init();                               /* 初期化 */
106 :     initI2CEeprom( &PADDR, &PADR, 0x56, 7, 5); /* EEPROM 初期設定 */
107 :     init_sci1( 0x00, 79 );                /* SCI1 初期化 */
108 :     set_ccr( 0x00 );                      /* 全体割り込み許可 */

```

EEP-ROM の初期化関数の入出力設定も 0x56 とします。

8.4.2 割り込みプログラム

```

673 : #pragma interrupt( interrupt_timer0 )
674 : void interrupt_timer0( void )
675 : {
676 :     unsigned int i;
677 :
678 :     ITU0_TSR &= 0xfe;                    /* フラグクリア */
679 :     cnt0++;
680 :     cnt1++;
681 :
682 :     iTimer10++;
683 :     if( iTimer10 >= 10 ) {
684 :         iTimer10 = 0;
685 :
686 :         /* エンコーダ関連 */
687 :         i = ITU2_CNT;
688 :         iEncoder = i - uEncoderBuff;
689 :         lEncoderTotal += iEncoder;
690 :         if( iEncoder > iEncoderMax )
691 :             iEncoderMax = iEncoder;
692 :         uEncoderBuff = i;
693 :
694 :         /* データ保存関連 */
695 :         if( saveFlag ) {
696 :             saveData[0] = pattern;        /* パターン */
697 :             saveData[1] = sensor_inp(0xff); /* センサ */
698 :             /* 2 はハンドル関数内で保存 */
699 :             /* 3 はモータ関数内で左モータ PWM 値保存 */
700 :             /* 4 はモータ関数内で右モータ PWM 値保存 */
701 :             saveData[5] = iEncoder;        /* エンコーダ */
702 :             saveData[6] = 0;               /* 予備 */
703 :             saveData[7] = 0;               /* 予備 */
704 :             setPageWriteI2CEeprom( saveIndex, 8, saveData );
705 :             saveIndex += 8;
706 :             if( saveIndex >= 0x8000 ) saveFlag = 0;
707 :         }
708 :     }
709 : }

```

エンコーダ処理を追加しています。エンコーダ処理も 10ms ごとの処理なので、iTimer 変数を兼用して、10ms ごととの処理部分に入れています。

701 行で、エンコーダ値を保存しています。

8.4.3 送信内容

```

591 :      case 76:
592 :          /* データ転送 */
593 :          led_out( (saveSendIndex/32) % 2 + 1 ); /* LED 点滅処理 */
594 :
595 :          /* 終わりのチェック */
596 :          if( (readI2CEeprom( saveSendIndex )==0) ||
597 :              (saveSendIndex >= 0x8000) ) {
598 :              pattern = 77;
599 :              cnt1 = 0;
600 :              break;
601 :          }
602 :
603 :          /* データの転送 */
604 :          convertHexToBin( readI2CEeprom(saveSendIndex+1), s );
605 :          printf( "%d,%8s,%d,%d,%d,%d\n",
606 :                 (char)readI2CEeprom( saveSendIndex+0 ), /* パターン */
607 :                 s, /* センサ */
608 :                 (char)readI2CEeprom( saveSendIndex+2 ), /* ハンドル */
609 :                 (char)readI2CEeprom( saveSendIndex+3 ), /* 左モータ */
610 :                 (char)readI2CEeprom( saveSendIndex+4 ), /* 右モータ */
611 :                 (char)readI2CEeprom( saveSendIndex+5 ) /* エンコーダ */
612 :          );
613 :
614 :          saveSendIndex += 8; /* 次の準備 */
615 :          break;

```

送信内容は、605 ~ 612 行の内容です。

```

printf( "%d,%8s,%d,%d,%d,%d\n",

```

```

    (char)readI2CEeprom( saveSendIndex+0 ), /* パターン */
    s, /* センサ */
    (char)readI2CEeprom( saveSendIndex+2 ), /* ハンドル */
    (char)readI2CEeprom( saveSendIndex+3 ), /* 左モータ */
    (char)readI2CEeprom( saveSendIndex+4 ), /* 右モータ */
    (char)readI2CEeprom( saveSendIndex+5 ) /* エンコーダ */

```

パターン、ハンドル、左モータ、右モータは 10 進数、センサは 8 文字の文字列で出力します。センサ値は、printf 文を実行する前に、convertHexToBin 関数で 2 進数に変換しています。その変換した文字列を出力するだけです。

8.5 走行からデータ転送までの流れ

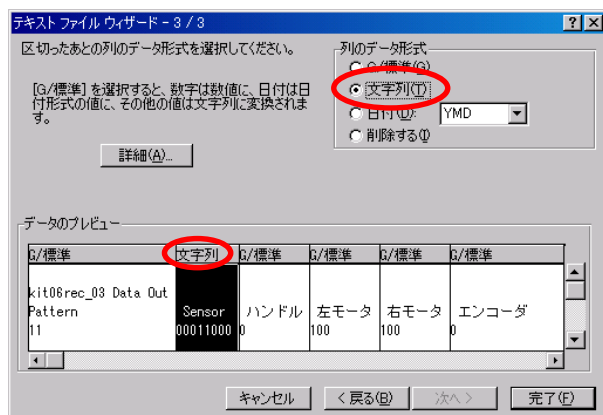
まず初めに、「kit06rec_03.c」の下記内容を自分のマイコンカーに合わせて調整します。他にも、調整する部分は調整してください。

37 :	#define	SERVO_CENTER	5000	/* サーボのセンタ値	*/
中略					
284 :	if(iEncoder >= 10) {			パターン 12: 右大曲げ処理時の速度	
285 :	speed2(0 ,0);				
286 :	} else {				
287 :	speed2(60 ,41);				
288 :	}				
中略					
308 :	if(iEncoder >= 10) {			パターン 13: 左大曲げ処理時の速度	
309 :	speed2(0 ,0);				
310 :	} else {				
311 :	speed2(41 ,60);				
312 :	}				
中略					
341 :	handle(-38);			左クランクを曲がるときのハンドル角度	
中略					
350 :	handle(38);			右クランクを曲がるときのハンドル角度	
中略					
356 :	if(iEncoder >= 10) {			パターン 23: クロスライン検出後の速度	
357 :	speed2(0 ,0);				
358 :	} else {				
359 :	speed2(70 ,70);				
360 :	}				
中略					
444 :	if(iEncoder >= 10) {			パターン 53: 右ハーフライン検出後の速度	
445 :	speed2(0 ,0);				
446 :	} else {				
447 :	speed2(70 ,70);				
448 :	}				
中略					
509 :	if(iEncoder >= 10) {			パターン 63: 左ハーフライン検出後の速度	
510 :	speed2(0 ,0);				
511 :	} else {				
512 :	speed2(70 ,70);				
513 :	}				

調整ができれば、プロジェクト「kit06rec_03」をビルドして、「kit06rec_03.mot」ファイルを CPU ボードに書き込みます。

8.6 走行データのグラフ化

今回、スピードデータが取れたので、線グラフ化してみます。



	A	B	C	D	E	F
1						
2	kit06rec_03	Data Out				
3	Pattern	Sensor	ハンドル	左モータ	右モータ	エンコーダ
4	11	00011000	0	100	100	0
5	11	00011000	0	100	100	0
6	11	00011000	0	100	100	0
7	11	00011000	0	100	100	0
8	11	00011000	0	100	100	1
9	11	00011000	0	100	100	1
10	11	00011000	0	100	100	1
11	11	00011000	0	100	100	1
12	11	00011000	0	100	100	2
13	11	00011000	0	100	100	1
14	11	00011000	0	100	100	3
15	11	00011000	0	100	100	2
16	11	00011000	0	100	100	2
17	11	00011000	0	100	100	3
18	11	00011000	0	100	100	3

1. エクセルでデータを取り込みます。センサの値のみ、「文字列」に変更します。他は、標準で OK です。

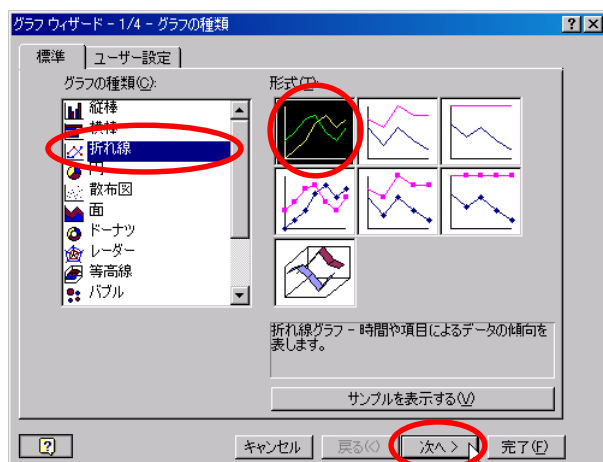
2. エクセルに取り込みました。

	A	B	C	D	E	F
1						
2	kit06rec_03	Data Out				
3	Pattern	Sensor	ハンドル	左モータ	右モータ	エンコーダ
4	11	00011000	0	100	100	0
5	11	00011000	0	100	100	0
6	11	00011000	0	100	100	0
7	11	00011000	0	100	100	0
8	11	00011000	0	100	100	1
9	11	00011000	0	100	100	1
10	11	00011000	0	100	100	1
11	11	00011000	0	100	100	1
12	11	00011000	0	100	100	2
13	11	00011000	0	100	100	1
14	11	00011000	0	100	100	3
15	11	00011000	0	100	100	2
16	11	00011000	0	100	100	2
17	11	00011000	0	100	100	3
18	11	00011000	0	100	100	3

	D	E	F	G	H	I
	ル	左モータ	右モータ	エンコーダ		
	0	100	100	0		
	0	100	100	0		
	0	100	100	0		
	0	100	100	0		
	0	100	100	1		
	0	100	100	1		

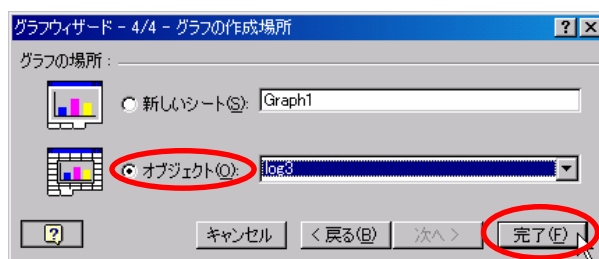
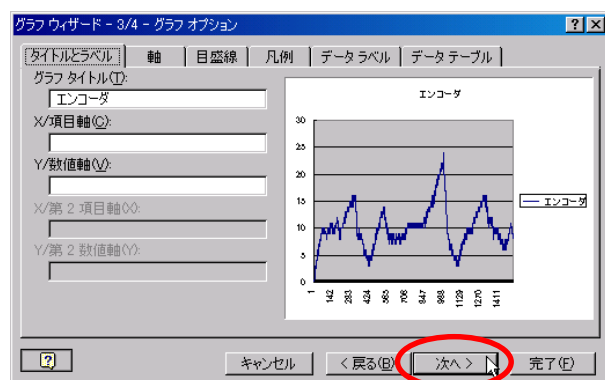
3. エンコーダ値のセルを選択します。

4. グラフ ウィザードを選択します。

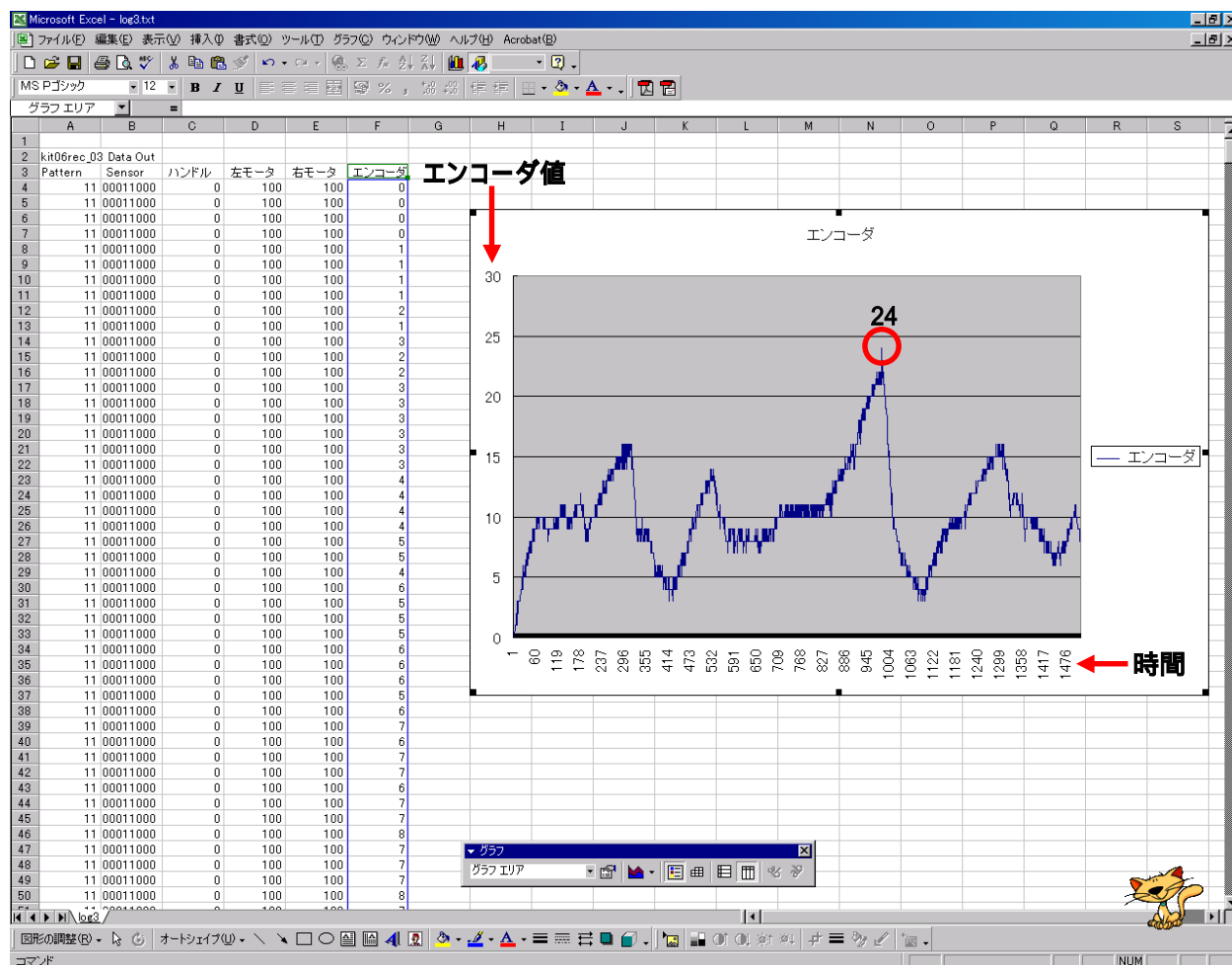


5. 折れ線グラフを選択します。次へをクリックします。

6. 次へをクリックします。



7. 各項目は、各自設定してください。特に設定しなくとも問題ありません。**次へ**で進みます。
8. 何処にグラフを追加するか選択します。とりあえず、「オブジェクト」を選択して、**完了**で完了です。



エンコーダ値のグラフが追加されました。

x軸が時間です。10ms ごとにデータを取っているの、1 当たり 10ms です。画面では、一番右が 1476 と表示されています。これは、スタートしてから 14760ms 後という意味です。

y 軸がスピードです。エンコーダ値が直接表示されています。

今回のエンコーダ値とスピードの関係は、「9.65 パルスで、1m/s」です。最速は 24 なので、

$9.65 : 1 = 24 : \text{最速のスピード}$

最速のスピード = 2.4m/s

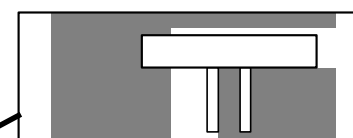
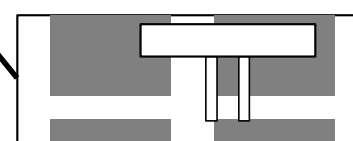
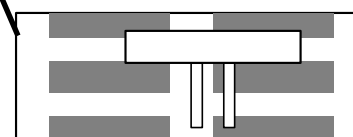
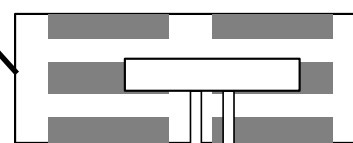
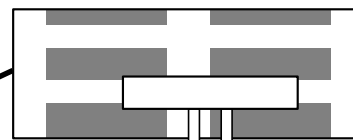
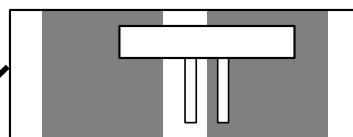
となります。その後、一気にスピードが落ちていますが、この部分は下り坂の後のクロスラインです。

例えば、カーブで脱輪したとします。このときのエンコーダ値を解析することにより、「そのスピード以上でカーブに進入したならブレーキをかけなさい」とプログラムすれば、脱輪を防ぐことができます。

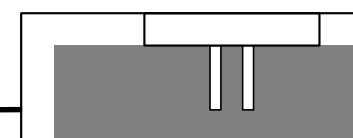
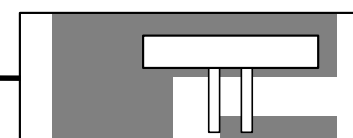
9. データをエクセルで解析する

これは、実際にあったデータです。なぜか、直角部分をまっすぐ行ってしまい、脱輪してしまう現象が多発していました。そこで、データ取得して、解析してみました。

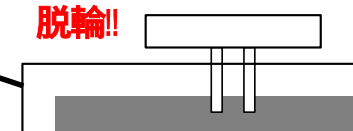
パターン	センサ 16 進数	センサ 左を2進数で	センサ 右を2進数で
11	30		
11	30		
11	30		
11	30		
11	30		
22	ff		
22	30		
22	ff		
22	30		
23	30		
23	30		
23	30		
23	30		
23	30		
23	30		
23	70		
23	70		
23	70		
23	30		
23	30		
23	30		
23	30		
23	70		
23	70		
23	70		
23	70		
23	7f		
23	7f		
23	00		
23	00		
23	00		
23	00		
23	00		
23	00		
23	00		
23	e0		
23	ff		
23	ff		
23	ff		
23	80		
23	00		
23	00		



右クランクである 0x1f ではないのでそのまま進む!!



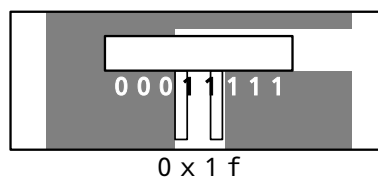
脱輪!!



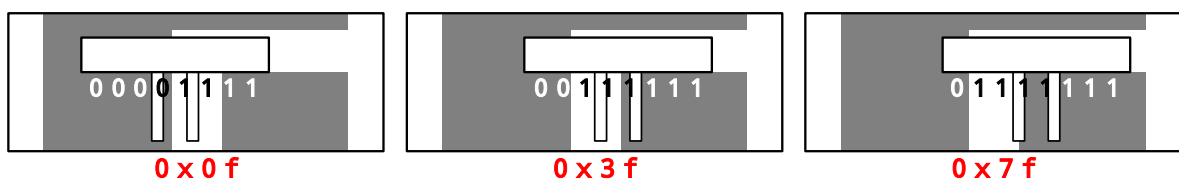
プログラムを見えます。

```
case 23:
/* クロスライン後のトレース、クランク検出 */
if( sensor_inp(MASK4_4)==0xf8 ) {
/* 左クランクと判断 左クランククリア処理へ */
led_out( 0x1 );
handle( -38 );
speed( 10 ,50 );
pattern = 31;
cnt1 = 0;
break;
}
if( sensor_inp(MASK4_4)==0x1f ) {
/* 右クランクと判断 右クランククリア処理へ */
led_out( 0x2 );
handle( 38 );
speed( 50 ,10 );
pattern = 41;
cnt1 = 0;
break;
}
switch( sensor_inp(MASK3_3) ) {
case 0x00:
/* センタ まっすぐ */
handle( 0 );
speed( 40 ,40 );
break;
case 0x04:
case 0x06:
case 0x07:
case 0x03:
/* 左寄り 右曲げ */
handle( 8 );
speed( 40 ,36 );
break;
case 0x20:
case 0x60:
case 0xe0:
case 0xc0:
/* 右寄り 左曲げ */
handle( -8 );
speed( 36 ,40 );
break;
}
break;
```

センサ8つの状態が 0x1f でなければ右クランクとは見なしません(下図)。



しかし、データ解析を何度も行うことにより、下図のような状態があることが分かりました。



そこで、右クランクと判断するセンサの状態を 0x1f の他、0x0f、0x3f、0x7f も追加します。

```
void main( void )
{
    int    i;
    unsigned char b;
    ===== 中略 =====

    case 23:
        /* クロスライン後のトレース、クランク検出 */
        b = sensor_inp(MASK4_4);
        if( b==0xf8 ) {
            /* 左クランクと判断 左クランククリア処理へ */
            led_out( 0x1 );
            handle( -38 );
            speed( 10 ,50 );
            pattern = 31;
            cnt1 = 0;
            break;
        }
        if( b==0x1f || b==0x0f || b==0x3f || b==0x7f ) {
            /* 右クランクと判断 右クランククリア処理へ */
            led_out( 0x2 );
            handle( 38 );
            speed( 50 ,10 );
            pattern = 41;
            cnt1 = 0;
            break;
        }
        switch( sensor_inp(MASK3_3) ) {
            case 0x00:
                /* センタ まっすぐ */
                handle( 0 );
                speed( 40 ,40 );
                break;
            case 0x04:
            case 0x06:
            case 0x07:
            case 0x03:
                /* 左寄り 右曲げ */
                handle( 8 );
                speed( 40 ,36 );
                break;
            case 0x20:
            case 0x60:
            case 0xe0:
            case 0xc0:
                /* 右寄り 左曲げ */
                handle( -8 );
                speed( 36 ,40 );
                break;
        }
        break;
}
```

この追加を行うことで、右クランクをクリアすることができました。

今回は、たまたま右クランクでセンサをチェックする状態が不足していましたが、左クランクもあり得ます。左クランクであり得るセンサの状態を自分で考えて、上記プログラムに追加してみてください。

10. 大容量 EEP-ROM(24C1024)を使う

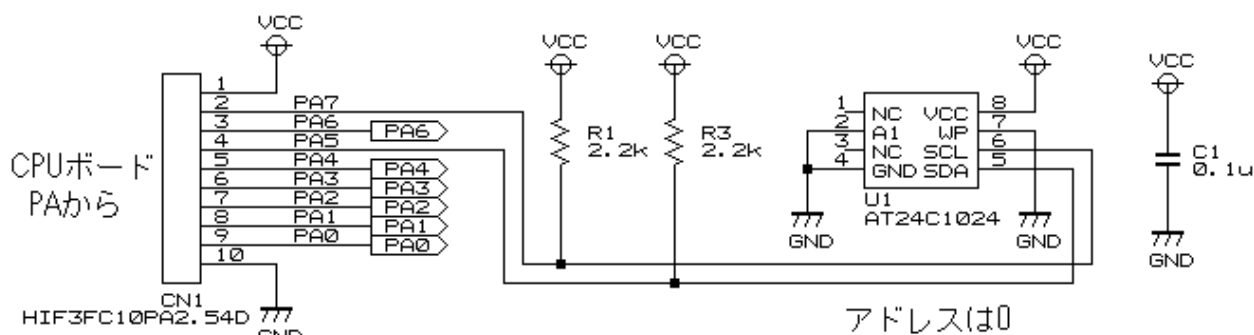
10.1 概要

ここまで、24C256 を使って説明してきました。メモリ容量の大きい 24C1024 という IC もあります。下記に違いをまとめます。

型式	24C256	24C1024
容量	32KB アドレスは 0 ~ 32767(0x7fff)	128KB アドレスは 0 ~ 131071 (0x1ffff)
同時 接続数	4 個まで 32KB × 4 = 128KB まで増設可能	2 個まで 128KB × 2 = 256KB まで増設可能
プロジェクトに追加 するファイル	i2c_eeprom.c	i2c_eeprom_1024.c
IC の ピンの違い		24C256 に対して、1 ピンが未接続 他は変更無し
特徴	値段が手頃	メモリ容量が 24C256 に比べ 4 倍ある

特徴のように、24C1024 は 24C256 に比べメモリ容量が 4 倍あるので、たくさんの情報を記録したいときに有効です。ただし、24C256 より値段が高いです。

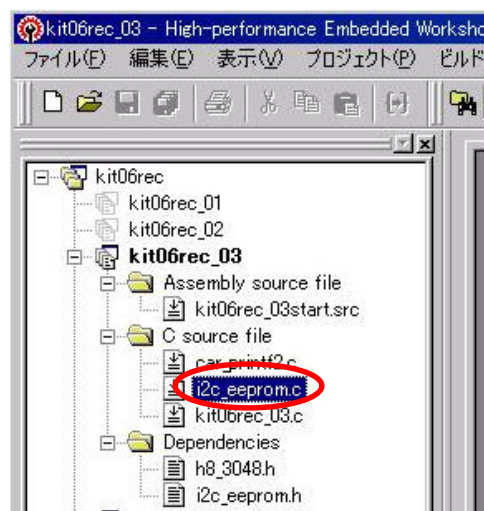
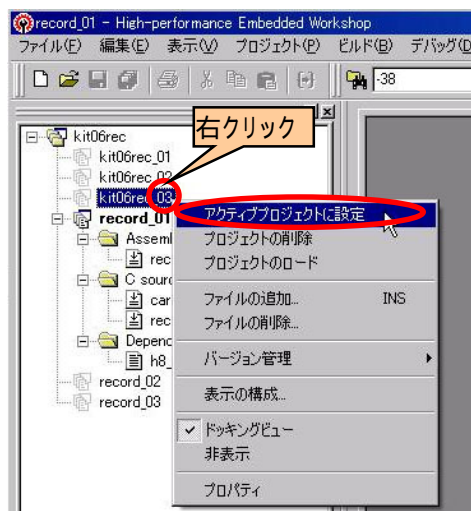
10.2 回路図



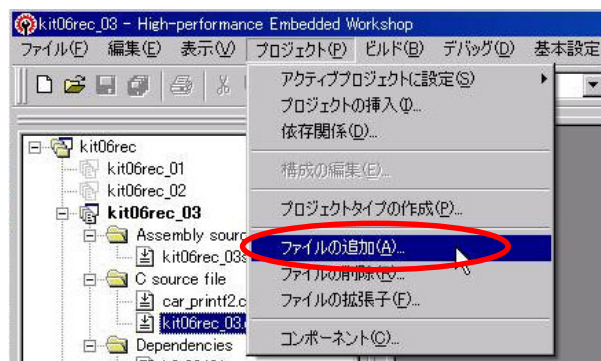
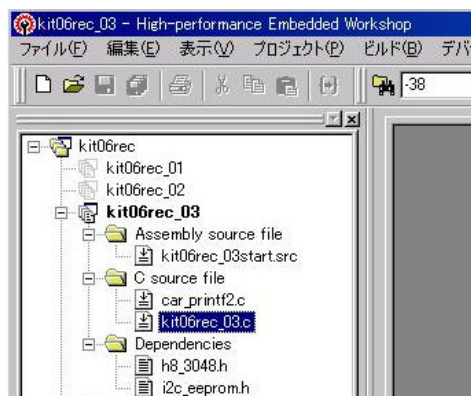
1 ピンが未接続になるだけで、他のピンの接続は変わりません。ソケットにしておけば、24C256 と 24C1024 は交換可能です。

10.3 プロジェクトへの登録方法

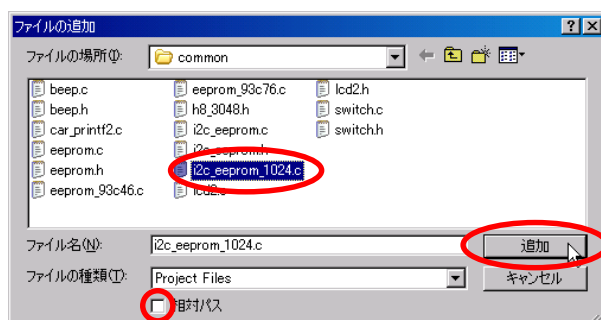
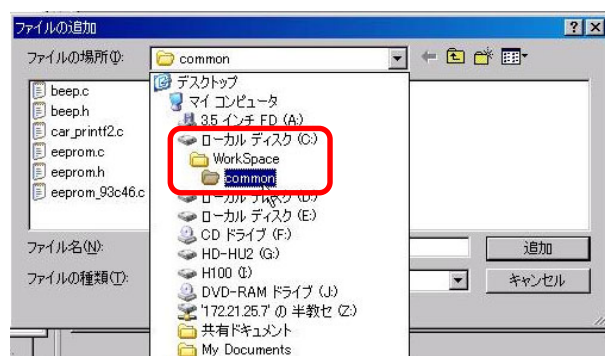
プロジェクト「kit06rec_03」を例に説明します。



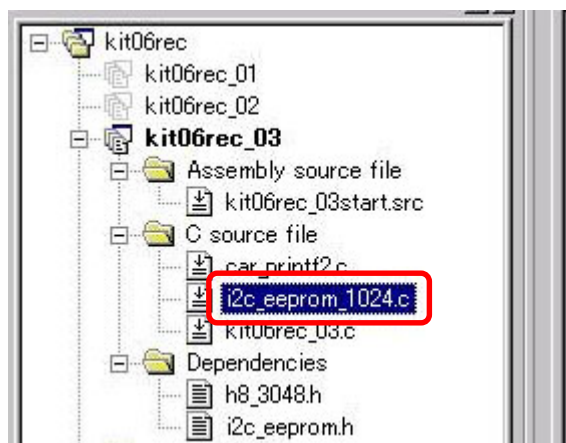
1. ワークスペース「kit06rec」を開きます。プロジェクト「kit06rec_03」をアクティブプロジェクトにします。
2. プロジェクト「kit06rec_03」の「i2c_eeprom.c」をクリックして、**DEL**キーを押します。



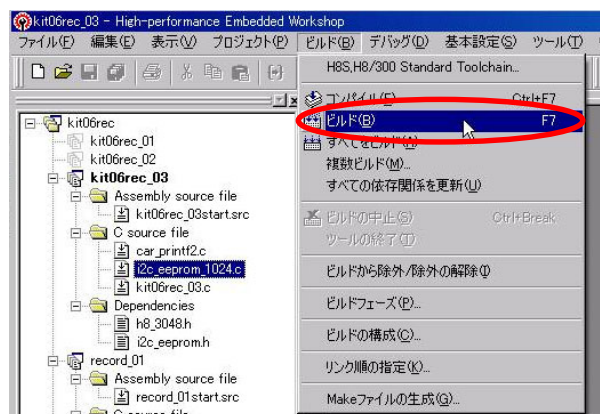
3. 「i2c_eeprom.c」が削除されました。
4. 「プロジェクト ファイルの追加」を選択します。



5. 「Cドライブ Workspace common」フォルダを選びます。
6. 「i2c_eeprom_1024.c」を選択します。「相対パス」のチェックは外します。**追加**をクリックします。



7. 「i2c_eeprom_1024.c」が追加されました。



8. 「ビルド ビルド」でビルドすれば、24C1024 対応のプログラムの完成です。

24C256 を接続しているときは「i2c_eeprom.c」、24C1024 を接続しているときは「i2c_eeprom_1024.c」を使用してください。互換はありません。

10.4 関数の変更点

initI2CEeprom	変更ありません。
selectI2C EepromAddress	どの EEP-ROM を使用するか選択します。 24C1024 の 2 ピンに接続されている電圧により変わります。 2 ピンが 0V の 24C1024 を対象にしたい場合 selectI2CEepromAddress(0); とします。最初はこの状態です。 2 ピンが 5V の 24C1024 を対象にしたい場合 selectI2CEepromAddress(1); とします。
readI2CEeprom	アドレスが 0 ~ 0x1ffff になります。他は変わりません。 例) EEP-ROM の 0x1f000 番地のデータを読み込む場合 i = readI2CEeprom(0x1f000);
writel2CEeprom	アドレスが 0 ~ 0x1ffff になります。他は変わりません。 例) EEP-ROM の 0x1f000 番地に、5 を書き込む場合 writel2CEeprom(0x1f000, 5);
pageWrite I2CEeprom	アドレスが 0 ~ 0x1ffff になります。他は変わりません。
setPageWrite I2CEeprom	アドレスが 0 ~ 0x1ffff になります。他は変わりません。
I2CEeprom Process	変更ありません。
clearI2CEeprom	変更ありません。

11. 参考文献

1. (株)ルネサス テクノロジ
H8/3048 シリーズ、H8/3048F-ZTAT™ (H8/3048F、H8/3048F-ONE)ハードウェアマニュアル 第7版
2. (株)ルネサス テクノロジ 半導体トレーニングセンター C言語入門コーステキスト 第1版
3. (株)オーム社 H8 マイコン完全マニュアル 藤澤幸穂著 第1版
4. 電波新聞社 マイコン入門講座 大須賀威彦著 第1版
5. 電波新聞社 C言語でH8マイコンを使いこなす 鹿取祐二著 第1版
6. ソフトバンク(株) 新C言語入門シニア編 林晴比古著 初版
7. 共立出版(株) プログラマのための ANSI C 全書 L.Ammersaal 著
吉田敬一・竹内淑子・吉田恵美子訳 初版
8. I2C バス仕様書バージョン 2.1
フィリップス社のホームページよりダウンロード可能
<http://jp.semiconductors.philips.com/markets/mms/protocols/i2c/>
http://www.semiconductors.philips.com/acrobat/literature/9398/39340011_jp.pdf
9. AT24C256 のデータシート
ATMEL 社のホームページよりダウンロード可能
<http://www.atmel.com/>
http://www.atmel.com/dyn/resources/prod_documents/doc0670.pdf

マイコンカーラーについての詳しい情報は、マイコンカーラー公式ホームページをご覧ください。

<http://www.mcr.gr.jp/>

H8 マイコンについての詳しい情報は、(株)ルネサス テクノロジのホームページをご覧ください。

<http://japan.renesas.com/>

の「マイコン」 「H8 ファミリ」 「H8/3048B グループ」でご覧頂けます

リンクは、2007 年 3 月現在の情報です。