

マイコンカーラリーキット Ver.4 対応

**kit07データ解析
実習マニュアル
microSD 編
H8/3048F-ONE 版**

**microSD 基板の使用は、Basic Class 出場選手は
認められていません(2010.04 現在)**

**マイコンで書き込んだ microSD は、Windows で読むことが来ません。
(Windows などパソコンで書き込んだデータは消えますのでご注意ください)**

第 1.05 版

2010.06.07

ジャパンマイコンカーラリー実行委員会

注意事項 (rev.3.0J)

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、文書によるジャパンマイコンカーラリー実行委員会の事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、慎重に制作したのですが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、最新の内容を確認いただきますようお願いいたします。

連絡先

(株)ルネサスソリューションズ ルネサス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

※記載されている会社名・製品名は、各社の商標または登録商標です。

目次

1. microSDを使ったデータ解析	1
1.1 概要.....	1
1.2 microSDカードを使う.....	2
1.3 microSDを使う意義.....	4
1.4 microSD基板.....	5
1.5 microSD基板のブロック図.....	5
1.6 microSD基板の回路.....	6
1.7 microSD基板をマイコンカーに取り付ける.....	7
2. microSD制御ライブラリ.....	8
2.1 「microsd_lib.c」で使用できる関数.....	8
2.2 「microsd_lib.c」を登録する方法.....	13
2.3 microSD基板の接続端子を変える場合の設定.....	15
3. サンプルプログラム.....	17
3.1 ルネサス統合開発環境.....	17
3.2 サンプルプログラムのインストール.....	17
3.2.1 CDからサンプルプログラムを取得する.....	17
3.2.2 ホームページからサンプルプログラムを取得する.....	17
3.2.3 インストール.....	18
3.3 ワークスペース「kit07msd」を開く.....	19
3.4 プロジェクト.....	20
4. プロジェクト「sd_01」 関数の実行時間確認.....	21
4.1 概要.....	21
4.2 接続.....	21
4.3 プロジェクトの構成.....	22
4.4 プログラム.....	22
4.5 プログラムの解説.....	24
4.5.1 変数.....	24
4.5.2 内蔵周辺機能の初期設定.....	25
4.5.3 microSDの初期化.....	26
4.5.4 microSDのイレース(0クリア).....	26
4.5.5 microSDへデータ書き込み.....	27
4.5.6 microSDからデータ読み込み.....	27
4.6 実行時間の測定方法.....	28
4.7 演習.....	30
4.8 関数の使用場面.....	31
5. プロジェクト「sd_02」 microSDにデータ記録.....	32
5.1 概要.....	32
5.2 接続.....	32
5.3 プロジェクトの構成.....	33
5.4 プログラム.....	33
5.5 setMicroSDdata関数とmicroSDProcess関数.....	37
5.5.1 概要.....	37

5.5.2	プログラムの流れ	38
5.5.3	各関数の処理内容	38
5.6	プログラムの解説	39
5.6.1	変数	39
5.6.2	main関数の開始部分	40
5.6.3	パターン 0:スタート	41
5.6.4	パターン 1: microSDクリア、書き込みアドレスセット	41
5.6.5	パターン 2: データ記録中	42
5.6.6	パターン 3:最後のデータが書き込まれるまで待つ	42
5.6.7	パターン 4:終了処理が終わるまで待つ	43
5.6.8	パターン 5:タイトル転送、準備	43
5.6.9	パターン 6: microSDよりデータ読み込み	43
5.6.10	パターン 7: パソコンへデータ転送	44
5.6.11	パターン 99: 終了	45
5.6.12	割り込み処理	45
5.7	データの取り込み方	47
5.8	int型、long型を記録するには	51
5.9	演習	51
5.10	演習の回答例	52
6.	プロジェクト「kit07sd_01」 走行データをmicroSDに記録	54
6.1	概要	54
6.2	マイコンカーの構成	54
6.3	プロジェクトの構成	55
6.4	プログラム	55
6.5	プログラムの解説	62
6.5.1	変数	62
6.5.2	main関数の開始部分	63
6.5.3	パターン 0:スイッチ入力待ち	64
6.5.4	パターン 1:スタートバーが開いたかチェック	65
6.5.5	パターン 71: 走行データ転送準備	66
6.5.6	パターン 72:最後のデータ書き込むまで待つ	66
6.5.7	パターン 73、74: スイッチが離されたかチェック	67
6.5.8	パターン 75: スイッチが押されたかチェック	67
6.5.9	パターン 76:タイトル送信	68
6.5.10	パターン 77: microSDよりデータ読み込み	68
6.5.11	パターン 78: データ転送	69
6.5.12	パターン 99: 転送終了	70
6.5.13	割り込み処理	71
6.5.14	記録データをバッファに保存	72
6.6	プログラムの調整	73
6.6.1	自分のマイコンカーに合わせて調整	73
6.6.2	記録間隔の変更	73
6.7	走行からデータ転送までの流れ	74
6.8	エクセルへの取り込み方	77
7.	プロジェクト「kit07sd_02」 エンコーダプログラムの追加	79
7.1	概要	79
7.2	マイコンカーの構成	79
7.3	プロジェクトの構成	80

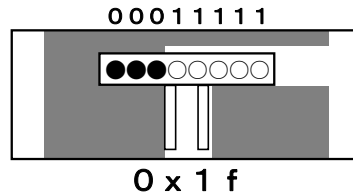
7.4 プログラムの解説.....	81
7.4.1 入出力設定.....	81
7.4.2 割り込みプログラム.....	82
7.4.3 送信内容.....	84
7.5 ロータリエンコーダに関わる計算.....	85
7.6 プログラムの調整.....	85
7.7 走行データのグラフ化.....	87
8. データをエクセルで解析する.....	90
9. 参考文献.....	93

1. microSDを使ったデータ解析

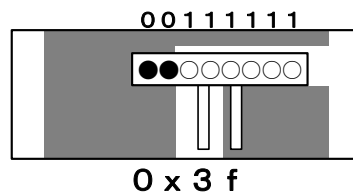
1.1 概要

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 kit07 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。



プログラムには、「0x3f は右クランクなので右に曲がりなさい」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、マイコンカーはプログラムどおりに動いているだけです。脱輪しないためには「0x3f」になったなら、どうしないといけないかプログラムを追加し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。

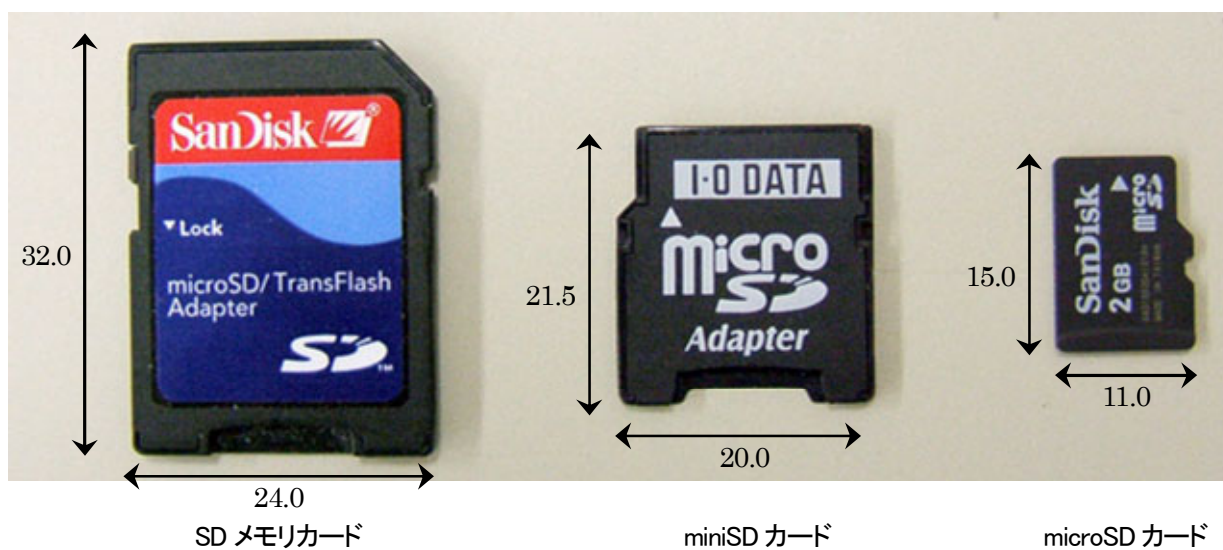
本マニュアルでは、
・データの記録方法
・パソコンへの転送方法
・解析方法
を紹介していきます。最後に、実例を紹介します。

1.2 microSDカードを使う

本書では、データ解析を行うためにマイコンカーの状態を記録するデバイスとして、microSD (マイクロエスディ) カード (以下、microSD) を使用します。microSD は、携帯電話などの記憶メディアとしてごく一般的なデバイスで、縦 15mm×横 11mm×厚さ 1mm、重さ 1g 未満と非常に小さいにも関わらず大容量です。

■SD メモリカードの種類

SD メモリカード (Secure Digital memory card) には、大きさにより SD メモリカード、miniSD カード、microSD カードの 3 種類あります。microSD は 3 種類の中でいちばん小さいカードです。



各 SD メモリカードの仕様を下表にまとめます。

	SD メモリカード	miniSD カード	microSD カード
幅	24.0mm	20.0mm	11.0mm
長さ	32.0mm	21.5mm	15.0mm
厚さ	2.1mm	1.4mm	1.0mm
体積	1,596mm ³	589mm ³	165mm ³
重量	約 2g	約 1g	約 0.4g
動作電圧	2.7~3.6V	2.7~3.6V	2.7~3.6V
誤消去防止スイッチ	あり	なし	なし
端子ガード突起	あり	なし	なし
端子数	9ピン	11ピン	8ピン
容量	最大 2GB	最大 2GB	最大 2GB

■SD メモリカードの規格

SD メモリカードの規格を下表に示します。

	通常	SDHC	SDXC
制定年度	1999 年	2006 年 1 月	2009 年 1 月
正式名称		SD High Capacity	SD eXtended Capacity
ファイル管理システム	FAT12、FAT16	FAT32	exFAT
容量	～2GB	2GB～32GB	32GB～2TB
今回対応しているか	○	×	×

今回、SDHC、SDXC には対応しておりません。よって、2GB までの容量の microSD となります。ただし、最近では、2GB でも SDHC の規格の microSD もあり、使うなら 1GB の容量が安心です。

■SD メモリカードの通信モード

SD メモリカードには、SD バスモードと SPI モードという 2 種類の通信モードがあります。

	SD (Secure Digital) バスモード	SPI (Serial Peripheral Interface) モード
信号線	CMD、DAT0、DAT1、DAT2、DAT3、CLK の 6 本	CS、CLK、DIN、DOUT の 4 本
通信速度	高速	低速
マイコンでの制御のしやすさ	難しい	比較的簡単
ライセンス	あり	なし

SD バスモードはライセンスがあり、ライセンスを購入しないと使用できません。本システムでは、ライセンスの問題と制御のしやすさで SPI モードを使用します。

1. microSD を使ったデータ解析

1.3 microSDを使う意義

今回、microSD を記憶媒体として使います。RY3048Fone ボードに搭載している H8/3048F-ONE マイコンにも内蔵 RAM があります。また、2008 年度までは、EEP-ROM(24C256)を使用して状態を記録していました。なぜわざわざ microSD を使うのでしょうか。表に長所、短所をまとめてみました。

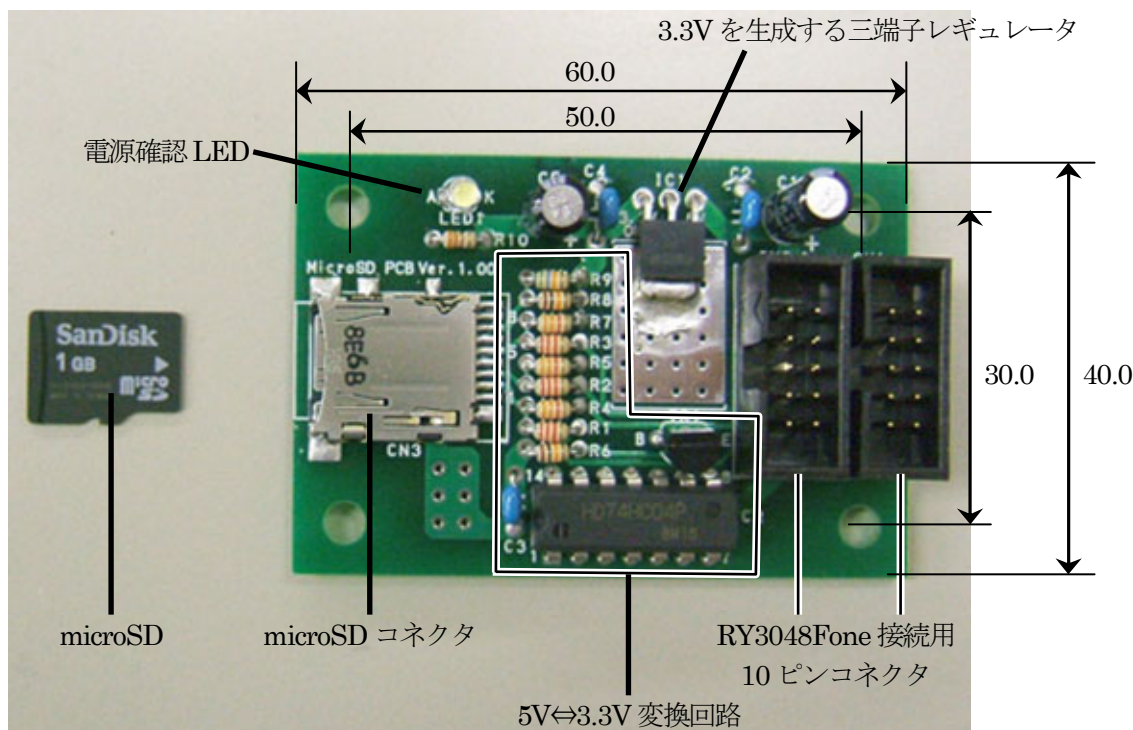
記憶メモリ	マイコン内蔵 RAM	外付け EEP-ROM (24C256)	microSD (SDHC ではないカード)
記憶容量	3KB 程度 3×1024 =3,072bytes	EEP-ROM 1個 32KB 32×1024 =32,768bytes ※マイコン内蔵 RAM の約 10 倍	最大 2GB $2 \times 1024 \times 1024 \times 1024$ =2,147,483,648bytes ※24C256 の 65536 倍
時間当たりの書き込み数	約 $1 \mu s$ で 1byte 書き込み可能	マイコンカーで使用する場合、10ms で 16bytes 書き込み可能	マイコンカーで使用する場合、10ms で 64bytes 書き込み可能
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	・基板が小型で、簡単に製作可能 ・電源が消えてもデータが消えない!	・ 大容量 ・ 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)	・ 基板の値段が EEP-ROM 基板より高い ・ 作業用として RAM を 1024 バイト使用する
記録時間	10ms ごとに 8bytes のデータを書き込むとすると、 $3072 \div 1$ 回の記録数 8bytes \times 記録間隔 10ms =3840[ms] =3.84 秒	10ms ごとに 8bytes のデータを書き込むとすると、 $32768 \div 1$ 回の記録数 8bytes \times 記録間隔 10ms =40960[ms] =40.96 秒 10ms ごとに 16bytes のデータを書き込むとすると、 $32768 \div 1$ 回の記録数 16bytes \times 記録間隔 10ms =20.48 秒	10ms ごとに 8bytes のデータを書き込むとすると、 $2147483648 \div 1$ 回の記録数 8bytes \times 記録間隔 10ms =2684354560[ms] =2684354.56 秒 \approx 745 時間 10ms ごとに 64bytes のデータを書き込むとすると、 $2147483648 \div 1$ 回の記録数 64bytes \times 記録間隔 10ms \approx 93 時間 ※2GB で計算

※マイコンで microSD への書き込みを行うと、FAT を壊します。Windows などを書き込んだデータは消されてしまいますので、内容を消しても良い microSD を使ってください。なお、マイコンで書き込んだ microSD を再度 Windows などを使用する場合、フォーマットすれば通常どおり使用することができます。

1.4 microSD基板

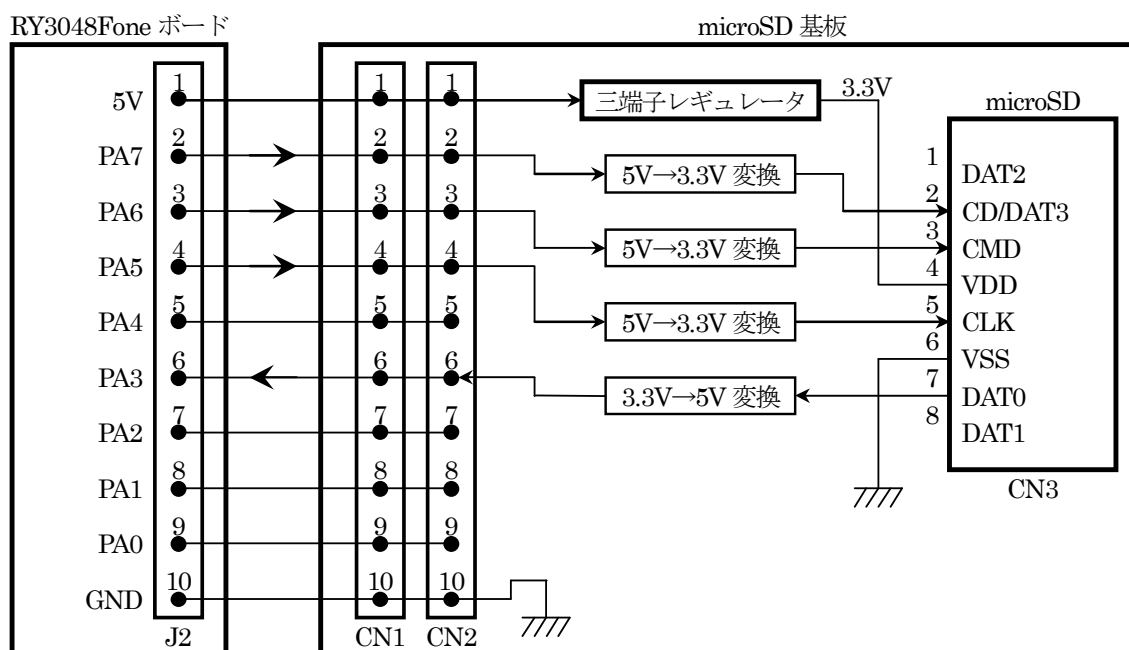
microSD 基板は、RY3048Fone ボードと microSD を接続する基板で、電圧変換も兼ねています。

microSD を差し込むためのコネクタと、RY3048Fone ボードと接続するための 10 ピンコネクタが付いています。10 ピンコネクタは 2 個付いていますが並列接続されており、どちらに繋いでも同じです。



1.5 microSD基板のブロック図

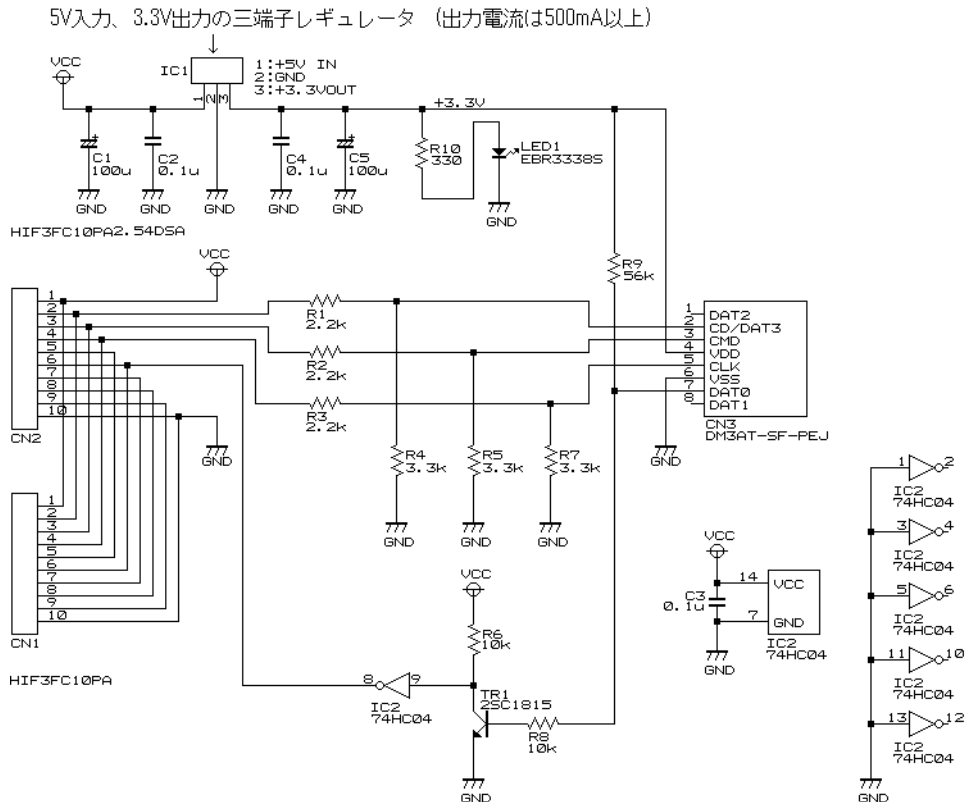
microSD 基板の主な役割は、RY3048Fone ボードの 5V 信号を microSD の 3.3V 信号に変換することです。microSD は、4bit の信号線で制御します。今回は、RY3048Fone ボードのポート A のあるコネクタと接続して、PA7,PA6,PA5,PA3 を使用します。ただし、使用するポート(端子)はプログラムの設定で変更することができます。



1. microSD を使ったデータ解析

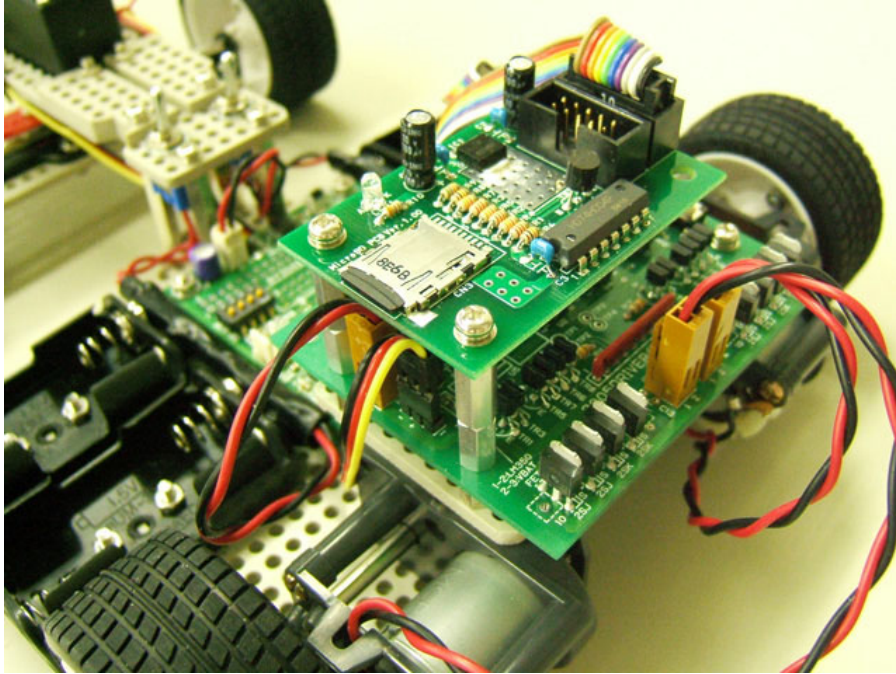
1.6 microSD基板の回路

番号	部品番号	部品名	型式・仕様	メーカ	数量	備考
1	IC1	三端子レギュレータ	5V 入力 3.3V 出力 出力電流 500mA 以上	各社	1	秋月電子 I-00538
2	IC2	ロジック IC	HD74HC04	ルネサス エレクトロ ニクス	1	
3	CN1,CN2	10P ストレートタイ プオス	HIF3FC10PA2.54DSA	ヒロセ電機	2	
4	CN3	microSD コネクタ	DM3AT-SF-PEJ	ヒロセ電機	1	秋月電子 C-02395
5	C2,C3,C4	積層セラミックコン デンサ	0.1 μ F	各社	3	
6	C1,C5	電解コンデンサ	10V/100 μ F	各社	2	
7	R1,R2,R3	抵抗	2.2K Ω 1/4W	各社	3	
8	R4,R5,R7	抵抗	3.3K Ω	各社	3	
9	R6,R8	抵抗	10K Ω	各社	2	
10	R9	抵抗	56K Ω	各社	1	
11	R10	抵抗	330 Ω	各社	1	
12	TR1	トランジスタ	2SC1815	東芝	1	
13	LED1	発光ダイオード	EBR3338S	スタンレー電気	1	
14		microSD	入手できる容量 ※ただし 2GB 以内	SanDisk 他	1	SDHC、 SDXC は 不可

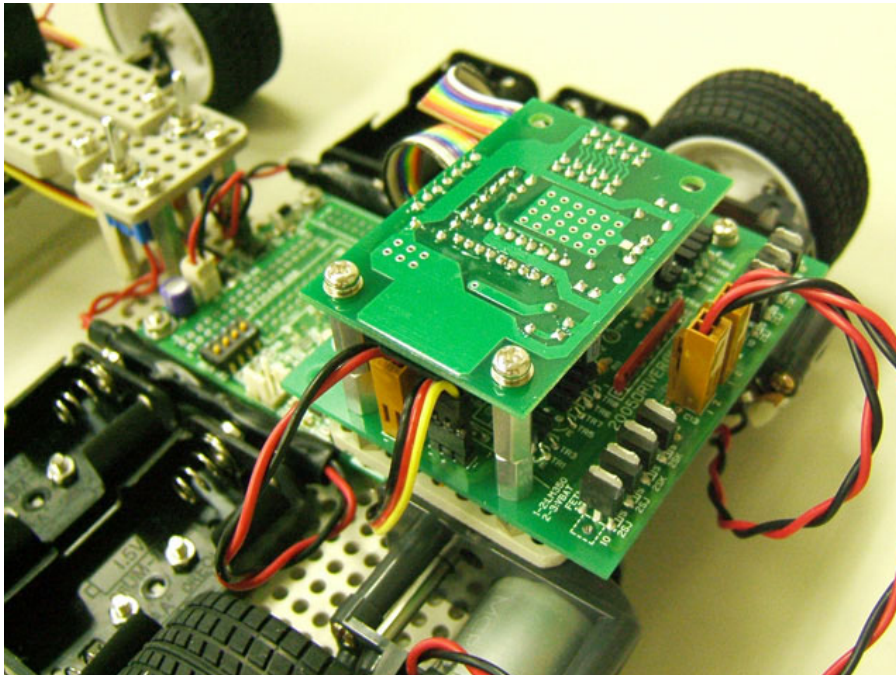


1.7 microSD基板をマイコンカーに取り付ける

microSD 基板の四隅にはねじ穴がありますので、マイコンカーの付けやすい位置に固定します。下写真は、モータドライブ基板のネジ位置にスタットを立てて、microSD 基板を取り付けたところです。



半田面を上向きに取り付ければ、脱輪したときなど部品が壊れづらくなります。ただし、電源確認用の LED が見えなくなります。



2. microSD制御ライブラリ

2.1 「microsd_lib.c」で使用できる関数

「microsd_lib.c」は、microSD にデータを読み書きする専用の関数が用意されているファイルです。microSD 基板を使用するときは、プロジェクトに「microsd_lib.c」を追加して使用します。

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

このファイルを追加すると、次の関数を実行することができます。

■initMicroSD関数

書式	int initMicroSD(void)
内容	microSD にデータを読み書きする準備をします。最初に必ず実行します。
引数	なし
戻り値	0:正常終了(準備完了) 1:CMD0 の返信コマンド受信エラー 2:CMD1 の返信コマンド受信エラー 3:CMD16 の返信コマンド受信エラー 4:未接続エラー、またはその他のエラー 0 以外はエラーです。エラーの多くは、microSD がソケットに入っていないか、microSD 基板との接続が正しくないかです。
使用例	<pre>ret = initMicroSD() ; if(ret != 0x00) { printf("microSD Initialize Error!!\n"); /* 初期化できず */ }</pre>

■readMicroSD関数

書式	int readMicroSD(unsigned long address, char *read)
内容	microSD から 512 バイトのデータを読み込みます。
引数	<ul style="list-style-type: none"> • unsigned long microSD から読み込むアドレス • char* 読み込んだデータを格納する配列 <p>アドレスは、必ず 512(0x200)の倍数で指定してください。 読み込むデータ数は、必ず 512 バイトとなります。読み込んだデータを格納する配列は 512 バイト以上確保しておいてください。</p>
戻り値	0:正常終了(読み込み完了) 11:CMD17 の返信コマンド受信エラー 12:データ受信待ちタイムアウト(時間切れ) 0 以外はエラーです。エラーの多くは、microSD がソケットに入っていないか、microSD 基板との接続が正しくないかです。

使用例	<pre>char msdBuff[512]; /* 一時記録バッファ */ ret = readMicroSD(0x0400 , msdBuff); // 0x400 番地から 512 バイト読み込む // 読み込んだデータは msdBuff に格納 if(ret != 0x00) { /* 読み込みエラー */ printf("microSD Read Error!!\n"); }</pre>
-----	---

■writeMicroSD関数

書式	int writeMicroSD(unsigned long address, char *write)
内容	microSD に 512 バイトのデータを書き込みます。
引数	<ul style="list-style-type: none"> • unsigned long microSD に書き込むアドレス • char* 書き込むデータを格納する配列 <p>アドレスは、必ず 512(0x200)の倍数で指定してください。 書き込むデータ数は、必ず 512 バイトとなります。書き込むデータを格納している配列は、必ず 512 バイト以上確保しておいてください。</p>
戻り値	<p>0:正常終了(書き込み完了) 21: CMD24 の返信コマンド受信エラー 22:書き込みエラー 23:その他のエラー</p> <p>0 以外はエラーです。エラーの多くは、microSD がソケットに入っていないか、microSD 基板との接続が正しくないかです。</p>
使用例	<pre>char msdBuff[512]; /* 一時記録バッファ */ ret = writeMicroSD(0x2000 , msdBuff); // 0x2000 番地に 512 バイト読み込む // 書き込むデータは msdBuff に格納 if(ret != 0x00) { /* 書き込みエラー */ printf("microSD Write Error!!\n"); }</pre>

■getMicroSD_CSD関数

書式	int getMicroSD_CSD(unsigned char *p)
内容	microSD から CSD (Card Specific Data:カード固有データ)を読み込みます。
引数	<ul style="list-style-type: none"> • char* CSD データを格納する配列(16 バイト以上) <p>正常に実行されると、指定した配列に 16 バイトのデータが格納されます。配列は 16 バイト以上の大きさにしてください。</p>
戻り値	<p>0:正常終了(CSD 読み込み完了) 31: CMD9 の返信コマンド受信エラー</p> <p>0 以外はエラーです。エラーの多くは、microSD がソケットに入っていないか、microSD 基板との接続が正しくないかです。</p>

2. microSD 制御ライブラリ

使用例	<pre>char msdCsdBuff[16]; /* 一時記録バッファ */ ret = getMicroSD_CSD(msdCsdBuff); // msdCsdBuff 配列に CSD データ格納 if(ret != 0x00) { /* CSD 読み込みエラー */ printf("microSD CSD Data Read Error!!\n"); }</pre>
-----	---

■eraseMicroSD関数

書式	int eraseMicroSD(unsigned long st_address, unsigned long ed_address)
内容	microSD のデータを消去します(0 書き込み)。
引数	<ul style="list-style-type: none"> • unsigned long 消去開始アドレス(512 の倍数) • unsigned long 消去終了アドレス(512 の倍数-1) <p>消去開始アドレスは 512 の倍数、消去終了アドレスは 512 の倍数-1 になるように設定します。ただし、「消去開始アドレス<消去終了アドレス」になるようにしてください。</p>
戻り値	<p>0:正常終了(イレース完了) 41:CMD32 の返信コマンド受信エラー 42:CMD33 の返信コマンド受信エラー 43:CMD38 の返信コマンド受信エラー 44:イレース後のテスト書き込みエラー</p> <p>0 以外はエラーです。エラーの多くは、microSD がソケットに入っていないか、microSD 基板との接続が正しくないかです。</p>
使用例	<pre>ret = eraseMicroSD(0x0200, 0x0fff); // 0x0200~0x0fff 番地をイレース if(ret != 0x00) { /* イレースエラー */ printf("microSD Erase Error!!\n"); }</pre>

■microSDProcessStart関数

書式	int microSDProcessStart(unsigned long address)
内容	setMicroSDdata 関数、microSDProcess を使う前に、この関数を宣言します。microSD に書き込むアドレスを指定します。
引数	<ul style="list-style-type: none"> • unsigned long microSD に書き込むアドレス <p>setMicroSDdata 関数で書き込む microSD の開始アドレスを指定します。開始アドレスは、必ず 512(0x200)の倍数で指定してください。</p>
戻り値	<p>0:正常終了(セット完了) 0 以外:異常終了</p> <p>0 以外はエラーです。0 以外は、既に microSDProcessStart 関数を実行しているか、コマンド送信エラーです。</p>
使用例	<pre>microSDProcessStart(0x1000); // 0x1000 番地から書き込みを行います</pre>

■microSDProcessEnd関数

書式	int microSDProcessEnd(void)
内容	setMicroSDdata 関数、microSDProcess 関数を実行し終わった後、この関数を宣言します。
引数	なし
戻り値	0:正常終了(セット完了) 0 以外:異常終了 0 以外はエラーです。0 以外はコマンド送信エラーです。
使用例	microSDProcessEnd();

■setMicroSDdata関数

書式	int setMicroSDdata(char *p)
内容	microSD にデータを書き込む準備をします。 書き込み処理自体は、次で説明する microSDProcess 関数で行います。
引数	<ul style="list-style-type: none"> char* 書き込むデータを格納する配列 書き込むデータ数は、必ず 512 バイトとなります。書き込むデータを格納している配列は、必ず 512 バイト以上確保しておいてください。
戻り値	0:正常終了(セット完了) 0 以外:前回の setMicroSDdata でセットした書き込みをまだ実行中で、今回のセットは無効 0 以外はエラーです。0 以外は、前回の setMicroSDdata 関数でセットした書き込みをまだ実行中で、今回のセットは無効になります。この場合、戻り値が 0 になるまで繰り返し実行します。ただし、繰り返しチェックすると通常のプログラム(マイコンカーの場合は、トレース)が実行できなくなる場合は、無視して次に進みます。
使用例	次の microSDProcess 関数で説明します。

■microSDProcess関数

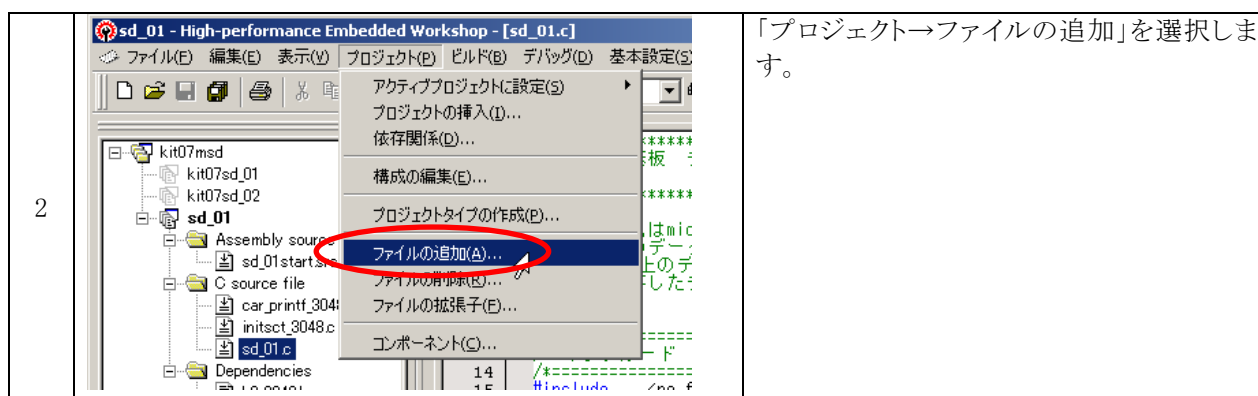
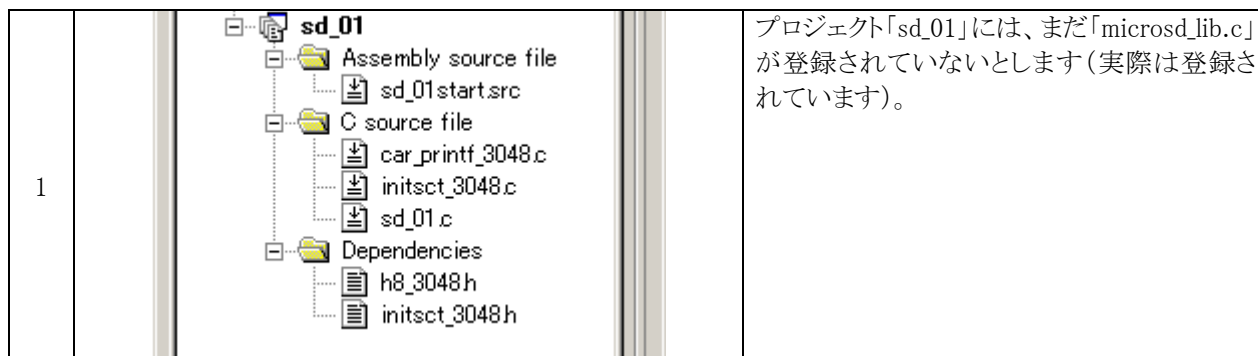
書式	void microSDProcess(void)
内容	setMicroSDdata 関数でセットしたアドレスに、セットしたデータ 512 バイトを、実際に書き込む作業を行う関数です。この関数は、割り込み内で呼び出します。1ms ごとに実行します。
引数	なし
戻り値	なし
使用例	<pre> void interrupt(void) { /* 割り込みで、1ms ごとに実行される関数 */ microSDProcess(); 1ms ごとに実行します iTimer10++; if(iTimer10 >= 10) { /* この中は 10ms ごとに実行されます */ iTimer10 = 0; buff[address+ 0] = P7DR; buff[address+ 1] = センサ; buff[address+ 2] = エンコーダ; ... buff[address+63] = 記録したい値; address += 64; if(address >= 512) { /* データが 512 バイト分たったら microSD に書き込み */ setMicroSDdata(buff); buff 配列のデータを 512 バイト書き込み msdAddress += 512; } } } </pre>

■checkMicroSDProcess関数

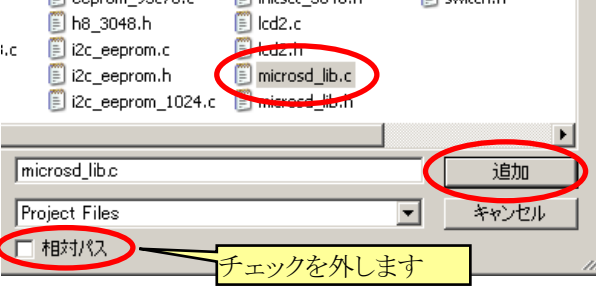
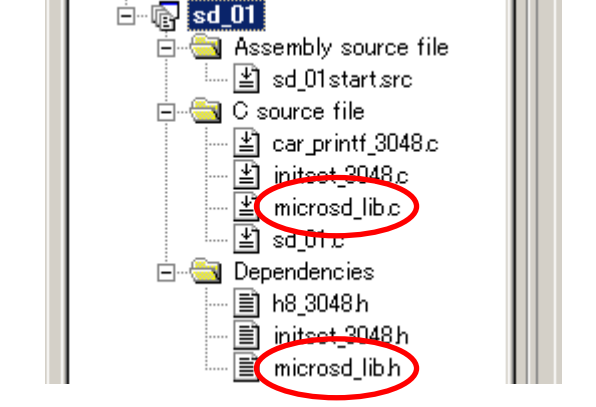
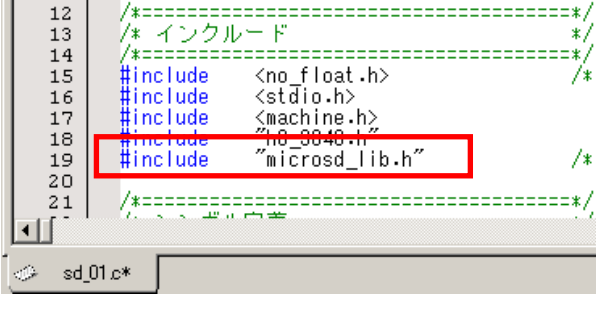
書式	int checkMicroSDProcess(void)
内容	microSDProcess 関数で現在実行してる状態を確認します。
引数	なし
戻り値	0:何もしていない 11:次の書き込み待機中 0と11以外:書き込み作業中 11 なら、setMicroSDdata 関数で書き込み内容をセットできます。それ以外なら、前回セットした内容を書き込み中なので setMicroSDdata 関数を実行してもエラーとなります。
使用例	<p>●例 1</p> <pre>while(checkMicroSDProcess() != 11); /* 書き込みが終わるまで待つ */</pre> <p>●例 2</p> <pre>if(checkMicroSDProcess() == 11) { /* 次の書き込み待機中なら実行 */ }</pre>

2.2 「microsd_lib.c」を登録する方法

microSD 基板を使うためには、プロジェクトに「microsd_lib.c」を登録します。例としてプロジェクト「sd_01」に「microsd_lib.c」を追加する手順を説明します。



2. microSD 制御ライブラリ

<p>3</p>		<p>「C:\¥Workspace¥common」フォルダを選択します。</p>
<p>4</p>		<p>「相対パス」のチェックを外して、「microsd_lib.c」を選択、「追加」をクリックします。</p>
<p>5</p>		<p>「microsd_lib.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「microsd_lib.h」が登録されます。</p>
<p>6</p>		<p>「sd_01.c」内のプログラムには「microsd_lib.c」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。</p>

2.3 microSD基板の接続端子を変える場合の設定

本書で紹介している microSD の端子と H8/3048F-ONE のポートとの関係は、次のような関係です。

microSD の端子名称	信号の方向	H8/3048F-ONE のポート
CS	←	PA7 (出力)
DIN(CMD)	←	PA6 (出力)
CLK	←	PA5 (出力)
DOUT(DAT0)	→	PA3 (入力)

ポートを変更したい場合、「microsd_lib.c」ファイルの一部を変更します。

18	: #define MSD_CS_PORT	PADR	/* CS 端子のポート	*/
19	: #define MSD_CS_BIT	0x80	/* CS 端子のビット	*/
20	: #define MSD_DI_PORT	PADR	/* DI 端子のポート	*/
21	: #define MSD_DI_BIT	0x40	/* DI 端子のビット	*/
22	: #define MSD_CK_PORT	PADR	/* CK 端子のポート	*/
23	: #define MSD_CK_BIT	0x20	/* CK 端子のビット	*/
24	: #define MSD_DO_PORT	PADR	/* D0 端子のポート	*/
25	: #define MSD_DO_BIT	0x08	/* D0 端子のビット	*/

「PORT」と書かれた行で、端子のポートをどれにするか指定します。データレジスタ名を直接記述します。

「BIT」と書かれた行で、端子のビットをどれにするか設定します。接続するビットと、その時の設定値の関係を次に示します。

接続するビット	設定値
7	0x80
6	0x40
5	0x20
4	0x10
3	0x08
2	0x04
1	0x02
0	0x01

2. microSD 制御ライブラリ

例として、次のように端子を変更するとします。ポート7は出力端子として使うことができません。

microSD の端子名称	信号の方向	H8/3048F-ONE の ポート
CS	←	P84 (出力)
DIN(CMD)	←	P83 (出力)
CLK	←	P82 (出力)
DOUT(DAT0)	→	P37 (入力)

「microsd_lib.c」の内容は、次のとおりになります。

```

18 : #define MSD_CS_PORT    P8DR          /* CS 端子のポート */
19 : #define MSD_CS_BIT    0x10         /* CS 端子のビット */
20 : #define MSD_DI_PORT    P8DR          /* DI 端子のポート */
21 : #define MSD_DI_BIT    0x08         /* DI 端子のビット */
22 : #define MSD_CK_PORT    P8DR          /* CK 端子のポート */
23 : #define MSD_CK_BIT    0x04         /* CK 端子のビット */
24 : #define MSD_DO_PORT    P3DR          /* D0 端子のポート */
25 : #define MSD_DO_BIT    0x80         /* D0 端子のビット

```

ポートの入出力設定も変更します。「sd_01.c」の init 関数部分です。P8DDRは、元々出力設定でしたので変更ありません。P3DDRは、bit7を入力にします。

```

197 : /******
198 : /* H8/3048F-ONE 内蔵周辺機能 初期化 */
199 : /******
200 : void init( void )
201 : {
202 :     /* I/Oポートの入出力設定 */
203 :     P1DDR = 0xff;
204 :     P2DDR = 0xff;
205 :     P3DDR = 0x7f;          /* 7:DOUT */
206 :     P4DDR = 0xff;
207 :     P5DDR = 0xff;
208 :     P6DDR = 0xf0;        /* CPU基板上的DIP SW */
209 :     P8DDR = 0xff;        /* 4:CS 3:DIN 2:CLK */
210 :     P9DDR = 0xf7;        /* 通信ポート */
211 :     PADDR = 0xf7;        /* microSD基板 */
212 :     PBDDR = 0xff;
213 :     /* ※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 */

```

3. サンプルプログラム

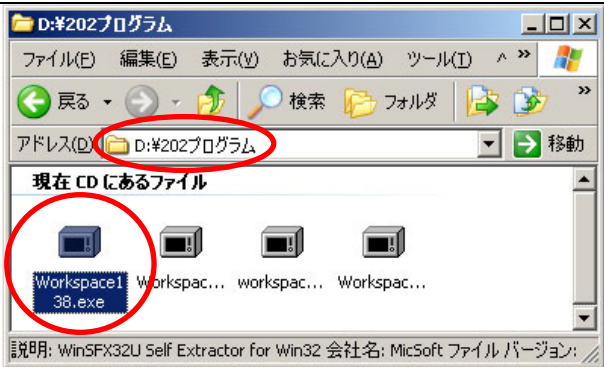
3.1 ルネサス統合開発環境

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 導入編」を参照してください。


3.2 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。

3.2.1 CDからサンプルプログラムを取得する

	<p>2009年以降の講習会 CD がある場合、「CDドライブ→202 プログラム」フォルダにある、「Workspace138.exe」を実行します。数字の138は、バージョンにより異なります。</p>
--	---

3.2.2 ホームページからサンプルプログラムを取得する

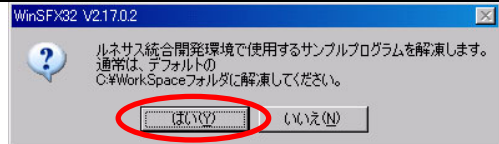
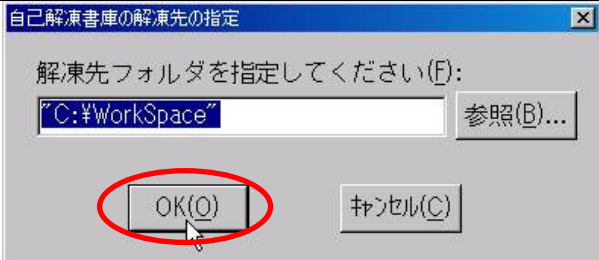
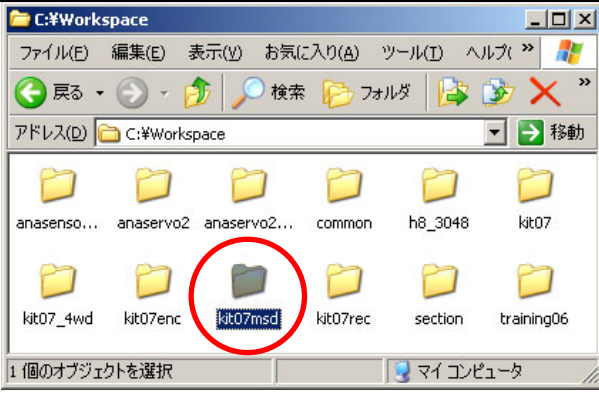
	<p>マイコンカーラーサイト「http://www.mcr.gr.jp/」の技術情報→ダウンロード内のページへ行きます。</p>
---	--

<p>2</p> <p>免責事項</p> <p>「マニュアル」、「ソフトウェア」は万全な体制で制作されており、通常の使用環境においては正常に動作するように作成されていますが、万が一「マニュアル」、「ソフトウェア」による損失・損害が発生した時には、『ジャパンマイコンカーラー実行委員会』はいかなる場合も責任を負いません。個人の免責が取れる範囲内であらかじめ了承した上でご使用くださるようお願いいたします。</p> <p>サンプルプログラム、書き込みソフトのダウンロード(H8編) 2010.04.01更新</p> <p>サンプルプログラム、書き込みソフトのダウンロード(R8C編) 2010.05.17更新</p> <p>マイコンに関する資料(H8編) 2009.05.25更新</p>	<p>「サンプルプログラム、書き込みソフトのダウンロード(H8 編)」をクリックします。</p>
--	--


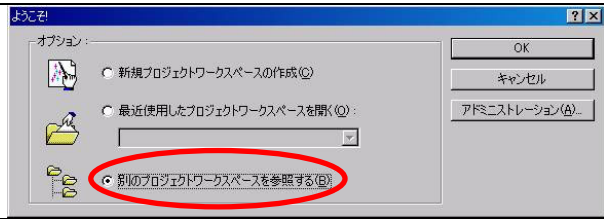
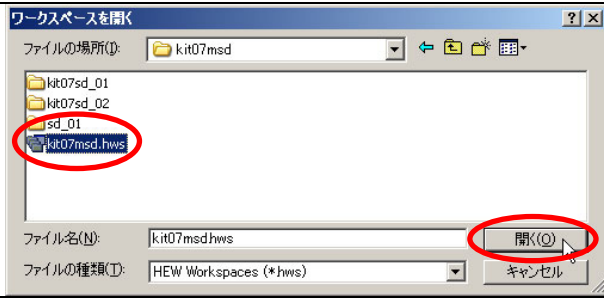
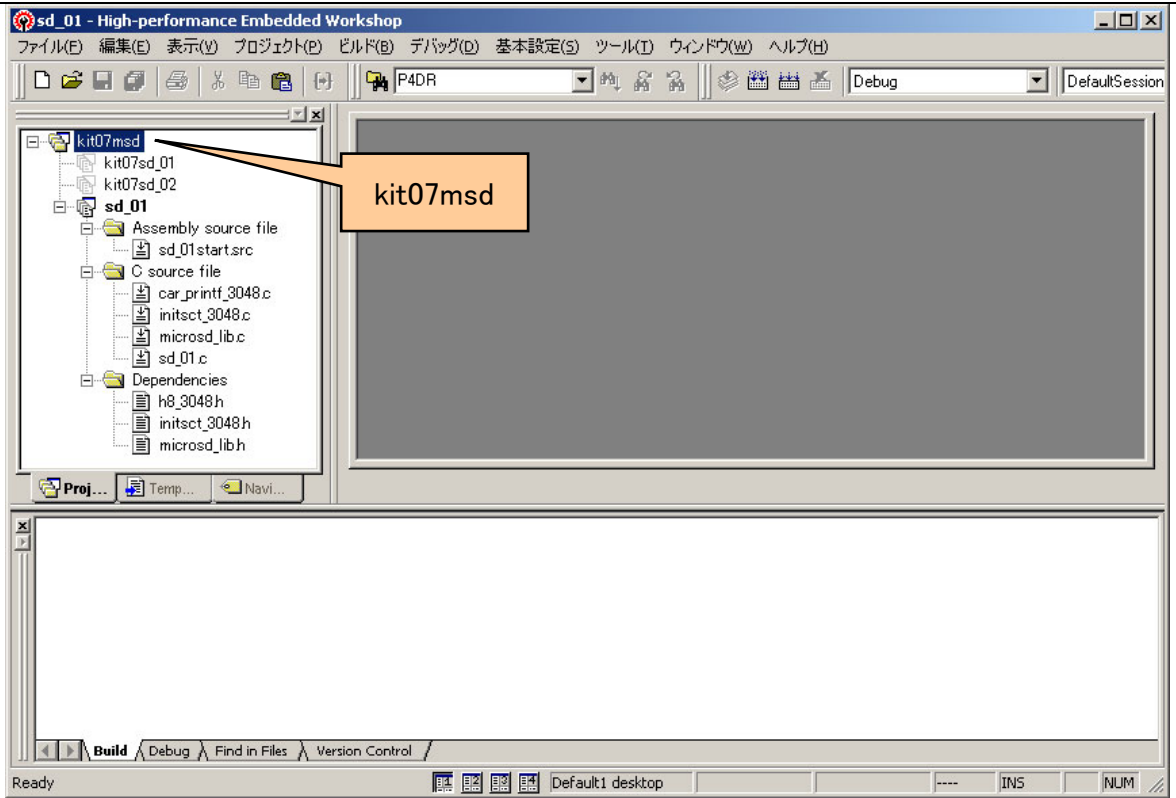
3. サンプルプログラム

3	<p>●ルネサス統合開発環境用その他ソフト Ver1.22 2007.04.24 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。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0.4MB)</p> <p>●ルネサス統合開発環境 H8/3048関連プログラム Ver1.26 2007.09.14 ルネサス統合開発環境で使用するH8/3048関係のサンプルプログラムです。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。 ※ Ver1.10より、ヘッダファイルなどの共通のファイルは、「c:\workspace\common」フォルダに入れています。 →DOWNLOAD (EXE 約4.47MB)</p> <p>●ルネサス統合開発環境 H8/3687関連プログラム Ver1.04 2007.09.02 ルネサス統合開発環境で使用するH8/3687関係のサンプルプログラムです。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。 ※ Ver1.03では、ワークスペースの複製を作ったときにビルドエラーが出る ことがありました。Ver1.04以降ではそのエラーを解消しています。 →DOWNLOAD (EXE 約2.65MB)</p>	<p>「ルネサス統合開発環境 H8/3048 関連プログラム」をダウンロードします。</p>
---	---	--

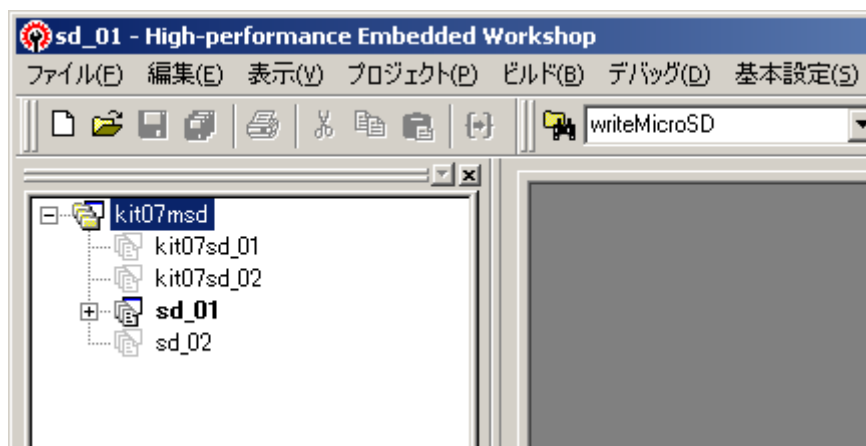
3.2.3 インストール

1		<p>CD またはダウンロードした「Workspace138.exe」を実行します。「はい」をクリックします。</p>
2		<p>ファイルの解凍先を選択します。「OK」をクリックします。このフォルダは変更できません。</p>
3		<p>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、「kit07msd」です。</p>

3.3 ワークスペース「kit07msd」を開く

1		<p>ルネサス統合開発環境を実行します。</p>
2		<p>「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。</p>
3		<p>Cドライブ → Workspace → kit07msd の「kit07msd.hws」を選択、開くをクリックします。</p>
4	 <p>ワークスペース「kit07msd」が開かれます。</p>	

3.4 プロジェクト



ワークスペース「kit07msd」には、4 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
sd_01	microSD を制御する関数の実行時間を確認するサンプルプログラムです。
sd_02	microSD 基板にデータを記録、転送するプログラムです。本プログラムでは、連続して書き込む方法を説明します。
kit07sd_01	kit07.c を元に、microSD 基板にデータを記録するプログラムを追加しました。走行後、パソコンに走行データを転送することができます。
kit07sd_02	プロジェクト「kit07sd_01」を元に、ロータリエンコーダを追加しました。記録するデータにもロータリエンコーダの値を追加しています。

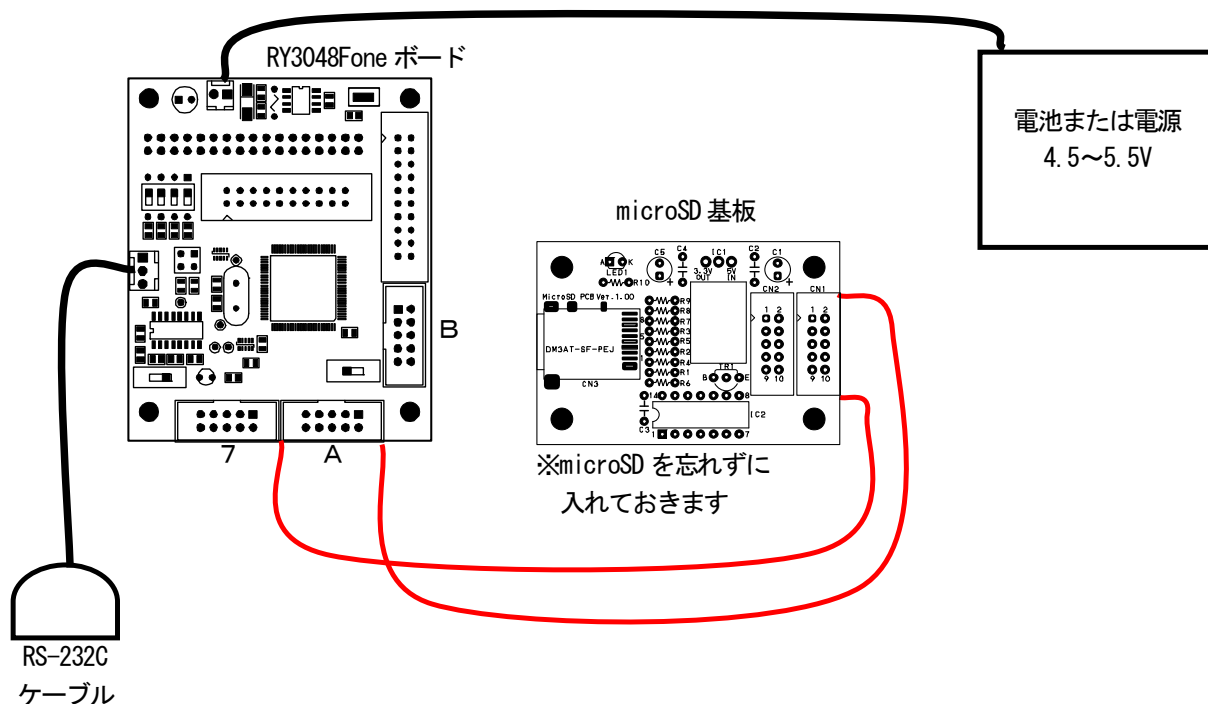
4. プロジェクト「sd_01」 関数の実行時間確認

4.1 概要

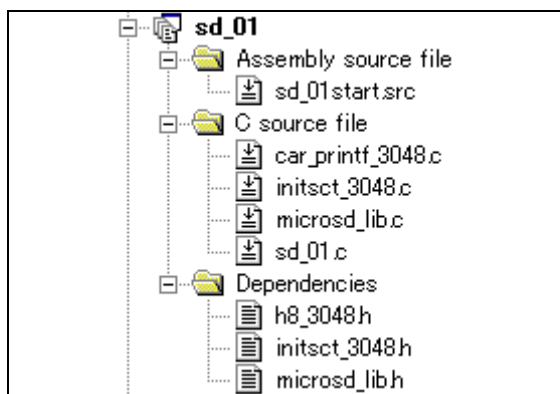
このプログラムは、microSD 基板を制御するための関数が正常に実行できるかチェックするとともに、実行時間を測定する確認用のプログラムです。

4.2 接続

・マイコンボードのポート A と、microSD 基板をフラットケーブルで接続します。



4.3 プロジェクトの構成



	ファイル名	内容
1	sd_01start.src	アセンブリ言語で記述されたアセンブリソースファイルです。このファイルの構造は下記のようになっています。 ベクタアドレス + スタートアップルーチン
2	car_printf_3048.c	•通信するための設定 •printf 関数の出力先、scanf 関数の入力元を通信にするための設定を行っています。
3	initsct_3048.c	初期値のないグローバル変数(セクション B 領域)、初期値のあるグローバル変数(セクション R 領域)の初期化用です。
4	microsd_lib.c	microSD 基板を制御するための関数が記載されています。
5	sd_01.c	実際に制御するプログラムが書かれています。H8/3048F-ONE の内蔵周辺機能の初期化も行います。
6	h8_3048.h	H8/3048F-ONE の内蔵周辺機能の I/O レジスタを定義したファイルです。
7	initsct_3048.h	initsct_3048.c のヘッダファイルです。
8	microsd_lib.h	microsd_lib.c のヘッダファイルです。

4.4 プログラム

```

1 : /******
2 : /* microSD基板 演習プログラム「sd_01.c」 */
3 : /* 2009.06 ジャパンマイコンカーラリー実行委員会 */
4 : /******
5 : /*
6 : microSD基板を制御するための関数が正常に実行できるかチェックするとともに、
7 : 実行時間を測定する確認用のプログラムです。
8 : */
9 :
10 : /*=====*/
11 : /* インクルード */
12 : /*=====*/
13 : #include <no_float.h> /* stdioの簡略化 最初に置く */
14 : #include <stdio.h>
15 : #include <machine.h>
16 : #include "h8_3048.h"
17 : #include "microsd_lib.h" /* microSD制御用 */
18 :
19 : /*=====*/
20 : /* シンボル定義 */
21 : /*=====*/

```

microSD を使用するので「microsd_lib.h」をインクルードします。

```

22 :
23 : /*=====*/
24 : /* プロトタイプ宣言 */
25 : /*=====*/
26 : void init( void );
27 :
28 : /*=====*/
29 : /* グローバル変数の宣言 */
30 : /*=====*/
31 : unsigned long cnt1; /* 時間計測用 */
32 :
33 : /* microSD関連変数 */
34 : char msdBuff[ 512 ]; /* 一時記録バッファ */
35 :
36 : /*=====*/
37 : /* メインプログラム */
38 : /*=====*/
39 : void main( void )
40 : {
41 :     int i, ret;
42 :     unsigned long l;
43 :
44 :     /* マイコン機能の初期化 */
45 :     init(); /* 初期化 */
46 :     init_scil( 0x00, 79 ); /* SC11初期化 */
47 :     set_ccr( 0x00 ); /* 全体割り込み許可 */
48 :
49 :     printf( "¥nmicroSD Test Program Ver.1.00¥n¥n" );
50 :
51 :     /* microSD初期化 */
52 :     cnt1 = 0;
53 :     ret = initMicroSD();
54 :     l = cnt1;
55 :     if( ret != 0x00 ) {
56 :         printf( "microSD Initialize Error!!¥n" ); /* 初期化エラー */
57 :         while( 1 ); /* 終了 */
58 :     } else {
59 :         printf( "microSD Initialize Time = %ldms¥n", l );
60 :     }
61 :
62 :     /* microSDイレース */
63 :     cnt1 = 0;
64 :     ret = eraseMicroSD( 0x0000, 0x5dc00-1 );
65 :     l = cnt1;
66 :     if( ret != 0x00 ) {
67 :         printf( "microSD Erase Error!!¥n" ); /* イレースエラー */
68 :         while( 1 ); /* 終了 */
69 :     } else {
70 :         printf( "microSD Erase Time = %ldms¥n", l );
71 :     }
72 :
73 :     /* バッファにダミーデータ書き込み */
74 :     for( i=0; i<512; i++ ) {
75 :         msdBuff[i] = i % 0x100;
76 :     }
77 :
78 :     /* microSD書き込み */
79 :     cnt1 = 0;
80 :     ret = writeMicroSD( 0x0000, msdBuff );
81 :     l = cnt1;
82 :     if( ret != 0x00 ) {
83 :         printf( "microSD Write Error!!¥n" ); /* 書き込みエラー */
84 :         while( 1 ); /* 終了 */
85 :     } else {
86 :         printf( "microSD Write Time = %ldms¥n", l );
87 :     }
88 :
89 :     /* バッファクリア */
90 :     for( i=0; i<512; i++ ) {
91 :         msdBuff[i] = 0x00;
92 :     }
93 :
94 :     /* microSD読み込み */
95 :     cnt1 = 0;
96 :     ret = readMicroSD( 0x0000, msdBuff );
97 :     l = cnt1;
98 :     if( ret != 0x00 ) {
99 :         printf( "microSD Read Error!!¥n" ); /* 読み込みエラー */
100 :         while( 1 ); /* 終了 */
101 :     } else {
102 :         printf( "microSD Read Time = %ldms¥n", l );
103 :     }
104 :
105 :     printf( "Program End...¥n¥n" );
106 :
107 :     while( 1 );
108 : }
109 :
110 : /*=====*/
111 : /* H8/3048F-ONE 内蔵周辺機能 初期化 */
112 : /*=====*/

```

microSD 関係の変数です。

microSD を初期化します。

「0x0000」～「0x5dc00-1」番地をイレース(0 でクリア)します。

msdBuff 配列の 512 バイト分のデータを、microSD の 0x0000 番地から書き込みます。

microSD の 0x0000 番地から 512 バイト分のデータを読み込み、msdBuff 配列に格納します。

4. プロジェクト「sd_01」 関数の実行時間確認

```

113 : void init( void )
114 : {
115 :     /* I/Oポートの入出力設定 */
116 :     P1DDR = 0xff;
117 :     P2DDR = 0xff;
118 :     P3DDR = 0xff;
119 :     P4DDR = 0xff;
120 :     P5DDR = 0xff;
121 :     P6DDR = 0xf0; /* CPU基板上的DIP SW */
122 :     P8DDR = 0xff; /* 通信ポート */
123 :     P9DDR = 0xf7; /* microSD基板 */
124 :     PADDR = 0xf7; /* microSD基板 */
125 :     PBDDR = 0xff;
126 :     /* ※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 */
127 :
128 :     /* ITU0 1msごとの割り込み */
129 :     ITU0_TCR = 0x23;
130 :     ITU0_GRA = 3071;
131 :     ITU0_IER = 0x01;
132 :
133 :     /* ITUのカウントスタート */
134 :     ITU_STR = 0x01;
135 : }
136 :
137 : /*****
138 : /* ITU0 割り込み処理 */
139 : /*****
140 : #pragma interrupt( interrupt_timer0 )
141 : void interrupt_timer0( void )
142 : {
143 :     ITU0_TSR &= 0xfe; /* フラグクリア */
144 :     cnt1++;
145 : }
146 :
147 : /*****
148 : /* end of file */
149 : /*****/
    
```

microSD 基板と接続しているポート A の入出力設定を変更します。

4.5 プログラムの解説

4.5.1 変数

31 :	unsigned long cnt1;	/* 時間計測用	*/
32 :			
33 :	/* microSD 関連変数 */		
34 :	char msdBuff[512];	/* 一時記録バッファ	*/

それぞれの変数は、次のような意味です。

変数名／配列名	意味	内容
cnt1	カウンタ	1ms ごとに+1 する変数です。変数の値をチェックすることにより経過時間が分かります。
msdBuff[512]	microSD 用バッファ	microSD に書き込むデータを記録したり、読み込んだデータを記録する配列です。microSD からのデータの読み書きは 512 バイト単位ですので、512 バイト以上確保してください。ただ、マイコンの場合はメモリ容量に限りがあるので、512 バイト確保すれば問題ありません。

4.5.2 内蔵周辺機能の初期設定

```

113 : void init( void )
114 : {
115 :     /* I/O ポートの入出力設定 */
116 :     P1DDR = 0xff;
117 :     P2DDR = 0xff;
118 :     P3DDR = 0xff;
119 :     P4DDR = 0xff;
120 :     P5DDR = 0xff;
121 :     P6DDR = 0xf0;          /* CPU 基板上の DIP SW          */
122 :     P8DDR = 0xff;
123 :     P9DDR = 0xf7;          /* 通信ポート                  */
124 :     PADDR = 0xf7;        /* microSD 基板                */
125 :     PBDDR = 0xff;
126 :     /* ※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 */
127 :
128 :     /* ITU0 1ms ごとの割り込み */
129 :     ITU0_TCR = 0x23;
130 :     ITU0_GRA = 3071;
131 :     ITU0_IER = 0x01;
132 :
133 :     /* ITU のカウントスタート */
134 :     ITU_STR = 0x01;
135 : }

```

microSD 基板は、マイコンボードのポート A に接続しています。ポート A の接続を下表に示します。microSD の CS、DIN、CLK が接続されている端子は出力、DOUT が接続されている端子は入力、未接続端子は出力に設定します。

マイコンボード J2 のピン番号	信号名	方向	microSD 基板の 接続端子	PADDR の 設定
1	+5V	電源		
2	PA7	→	CS	1
3	PA6	→	DIN	1
4	PA5	→	CLK	1
5	PA4		未接続	1
6	PA3	←	DOUT	0
7	PA2		未接続	1
8	PA1		未接続	1
9	PA0		未接続	1
10	GND	電源		

ポート A の入出力方向を決める PADDR レジスタの設定は、2 進数で「1111 0111」、16 進数で「0xf7」となります。124 行目で PADDR に 0xf7 を設定しています。

4.5.3 microSDの初期化

```

51 :      /* microSD 初期化 */
52 :      cnt1 = 0;
53 :      ret = initMicroSD();
54 :      l = cnt1;
55 :      if( ret != 0x00 ) {
56 :          printf( "microSD Initialize Error!!\n" ); /* 初期化エラー */
57 :          while( 1 ); /* 終了 */
58 :      } else {
59 :          printf( "microSD Initialize Time = %ldms\n", l );
60 :      }

```

initMicroSD 関数は、microSD を初期化する関数です。今回は 53 行目で実行しています。ret 変数に関数を実行した結果が格納されます。0 なら、正常に初期化ができたということです。0 以外なら初期化できていません。microSD がコネクタに挿入されているか、マイコンボードと microSD 基板が正しく接続されているかなど、確認してください。

時間の算出方法は次のようになります。

```

52 :      cnt1 = 0; cnt1 変数を 0 にクリア
53 :      初期化
54 :      l = cnt1; cnt1 変数の値を l 変数に保存 → l には初期化にかかった時間が格納!!
中略
59 :      printf( "microSD Initialize Time = %ldms\n", l ); 結果表示

```

初期化する前に、52 行で cnt1 変数をクリアしています。関数実行後、54 行で cnt1 変数の値を l (エル) 変数に保存しています(cnt1 の値は 1ms ごとに増えていくので)。よって、変数 l には、初期化にかかった時間がミリ秒単位で格納されています。59 行でその値を表示し、初期化にかかった時間を確認することができます。

4.5.4 microSDのイレース(0 クリア)

```

62 :      /* microSD イレース */
63 :      cnt1 = 0;
64 :      ret = eraseMicroSD( 0x00000, 0x5dc00-1 );
                        ↑      ↑
                        開始アドレス 終了アドレス
65 :      l = cnt1;
66 :      if( ret != 0x00 ) {
67 :          printf( "microSD Erase Error!!\n" ); /* イレースエラー */
68 :          while( 1 ); /* 終了 */
69 :      } else {
70 :          printf( "microSD Erase Time = %ldms\n", l );
71 :      }

```

eraseMicroSD 関数は、microSD をイレースする関数です。今回は 64 行目で実行しています。ret 変数に関数を実行した結果が格納されます。0 なら、正常にイレースができたということです。0 以外ならイレースできていません。

eraseMicroSD 関数を実行する前に cnt1 変数をクリアして、実行後に cnt1 の値を l 変数に保存します。l 変数の値が、イレース時間になります。

eraseMicroSD 関数の引数は、イレース開始アドレスとイレース終了アドレスを代入します。引数は次のように設定してください。

- イレース開始アドレス … 512(0x200)の倍数
- イレース終了アドレス … 512(0x200)の倍数-1
- ただし、イレース開始アドレス < イレース終了アドレス

4.5.5 microSDヘデータ書き込み

```

78 :    /* microSD 書き込み */
79 :    cnt1 = 0;
80 :    ret = writeMicroSD( 0x0000 , msdBuff );
                        ↑           ↑
                        書き込むアドレス 書き込む 512 バイトのデータを格納している配列名
81 :    l = cnt1;
82 :    if( ret != 0x00 ) {
83 :        printf( "microSD Write Error!!\n" );    /* 書き込みエラー */
84 :        while( 1 ); /* 終了 */
85 :    } else {
86 :        printf( "microSD Write Time = %ldms\n", l );
87 :    }
    
```

writeMicroSD 関数は、microSD ヘデータを書き込む関数です。今回は 80 行目で実行しています。**書き込むデータ数は 512 バイト固定です**。ret 変数に関数を実行した結果が格納されます。0 なら、正常に書き込みができたということです。0 以外なら書き込まれていません。

writeMicroSD 関数を実行する前に cnt1 変数をクリアして、実行後に cnt1 の値を l 変数に保存します。l 変数の値が、書き込み時間になります。

writeMicroSD 関数の引数は、書き込み開始アドレスと書き込むデータを格納している配列名を代入します。引数は次のように設定してください。

- 書き込み開始アドレス …………… 512(0x200)の倍数
- データが格納されている配列名 … char 型で 512 バイト以上の配列

4.5.6 microSDからデータ読み込み

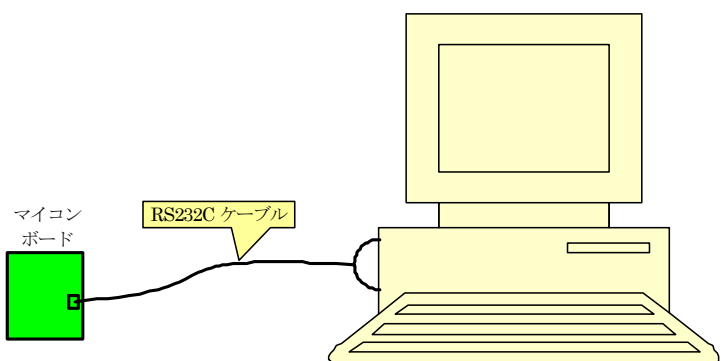
```

94 :    /* microSD 読み込み */
95 :    cnt1 = 0;
96 :    ret = readMicroSD( 0x0000 , msdBuff );
                        ↑           ↑
                        読み込むアドレス 読み込んだ 512 バイトのデータを格納する配列名
97 :    l = cnt1;
98 :    if( ret != 0x00 ) {
99 :        printf( "microSD Read Error!!\n" );    /* 読み込みエラー */
100 :        while( 1 ); /* 終了 */
101 :    } else {
102 :        printf( "microSD Read Time = %ldms\n", l );
103 :    }
    
```

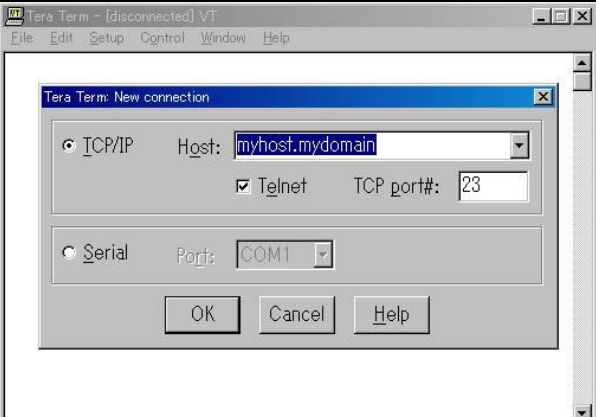
readMicroSD 関数は、microSD からデータを読み込む関数です。今回は 96 行目で実行しています。**読み込むデータ数は 512 バイト固定です**。ret 変数に関数を実行した結果が格納されます。0 なら、正常に読み込みができたということです。0 以外なら読み込まれていません。

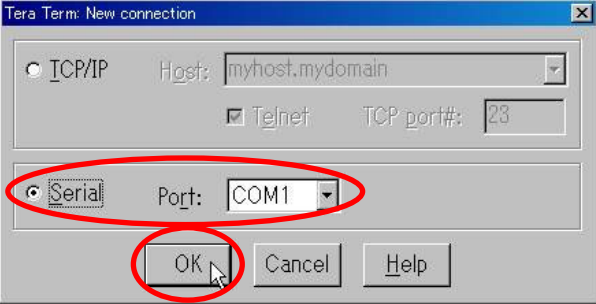
readMicroSD 関数を実行する前に cnt1 変数をクリアして、実行後に cnt1 の値を l 変数に保存します。l 変数の値が、読み込み時間になります。

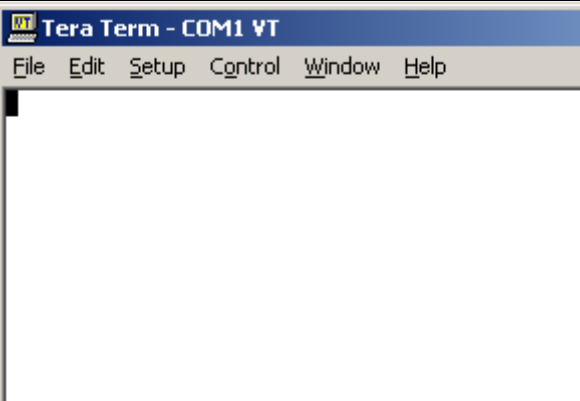
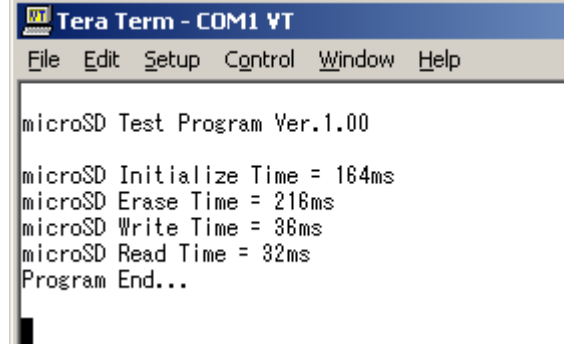
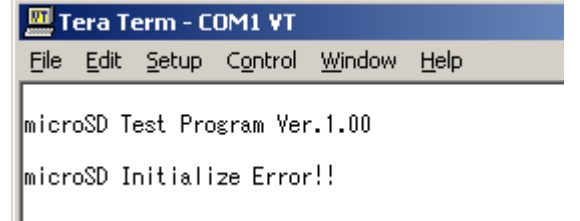
4.6 実行時間の測定方法

1		<p>プロジェクト「sd_01」をビルドして、「sd_01.mot」ファイルをマイコンボードに書き込んでください。書き込みができれば、書き込みスイッチを FWE の逆側にして、電源を切っておきます。マイコン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。</p>
---	---	---

2		<p>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、H8/3048F-ONE 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。</p>
---	---	---

3		<p>接続先を確認する画面が表示されます。</p>
---	---	---------------------------

4		<p>「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。</p>
---	---	--

5		立ち上がりました。
6	 <pre> microSD Test Program Ver.1.00 microSD Initialize Time = 164ms microSD Erase Time = 216ms microSD Write Time = 36ms microSD Read Time = 32ms Program End... </pre>	<p>マイコンボードの電源を入れます。 マイコンボードからデータが送られてきます。 Tera Term Pro に表示されれば成功です。</p> <p>それぞれの関数の実行時間が、ミリ秒単位で表示されます。</p>
7	 <pre> microSD Test Program Ver.1.00 microSD Initialize Error!! </pre>	microSD が挿入されていないか、microSD 基板と正しく結線されていないか「Initialize Error!!」と表示されます。マイコンボードの電源を切り確認後、再度電源を入れてください。

4.7 演習

TeraTermPro の画面に表示される内容は、次のとおりです。

microSD Initialize Time = ??ms	← initMicroSD 関数の実行時間
microSD Erase Time = ??ms	← eraseMicroSD 関数の実行時間
microSD Write Time = ??ms	← writeMicroSD 関数の実行時間
microSD Read Time = ??ms	← readMicroSD の実行時間

※ ??には、実際にかかった時間が入ります。

下表にしたがって、「sd_01.c」内の microSD を制御する関数の引数を変更して(変更しないときもあります)、実行してみましょう。このとき、同じプログラムを2回実行します(ただ、リセットボタンを押して再実行だけです)。表の1回目の部分と2回目の部分に、実際に表示された時間を記録します。1回目と2回目で変化があるか確かめてみましょう。

演習で使 用した microSD のメーカ		microSD の 容量	
--------------------------------	--	-----------------	--

行	変更内容	実行時間 1回目	実行時間 2回目
修正 なし	53 initMicroSD();		
	64 eraseMicroSD(0x0000 , 0x5dc00-1);		
	80 writeMicroSD(0x0000 , msdBuff);		
	96 readMicroSD(0x0000 , msdBuff);		
修正 1 回 目	53 initMicroSD(); // この行は修正無し		
	64 eraseMicroSD(0x0000 , 0x1000-1);		
	80 writeMicroSD(0x10000 , msdBuff);		
	96 readMicroSD(0x10000 , msdBuff);		
修正 2 回 目	53 initMicroSD(); // この行は修正無し		
	64 eraseMicroSD(0x0000 , 0x8000000-1);		
	80 writeMicroSD(0x7000000 , msdBuff);		
	96 readMicroSD(0x7000000 , msdBuff);		

4.8 関数の使用場面

microSD を制御する 4 つの関数の実行時間を調べました。

これらの 4 つの関数をマイコンカーで使用する場合、いつ使用するか(実行するか)考えてみます。また、このときの実行時間も考えて、マイコンカー制御でデータ記録用として使えるか検討してみます。


関数名	いつ使用するか (いつ実行するか)	許容できる関数の実行時間	実際の 実行時間	使用 可能か?
initMicroSD();	マイコンカーの電源を入れたとき(走行直前)に実行する。	走行前なので、イニシャライズが終わるまで待てるが、長すぎると人間がいらいらする。実用的に、1 秒程度なら待てる。	最大 1 秒以内	可能
eraseMicroSD();	走行直前(マイコンカーの電源を入れたとき)に実行する。	走行前なので、イレースが終わるまで待てるが、長すぎると人間がいらいらする。実用的に、1 秒程度なら待てる。	最大 1 秒以内	可能
writeMicroSD();	走行中に実行する。	走行中なので、1 つの関数の実行時間は極力短くしなければいけない。 (参考: startbar_get 関数は $2 \mu s$ 、sensor_inp 関数は $3 \mu s$ 、speed 関数は約 $30 \mu s$ 程度の実行時間です) マイコンカーの制御に影響を与えないようにするには、1 回の実行が 500μ 以内でないと実用に耐えない。	30~100ms	使用不可
readMicroSD();	走行後(データをパソコンに転送するとき)に実行する。	走行後なので、読み込みが終わるまで待てるが、長すぎると人間がいらいらする。実用的に、512 バイトの読み込みが 0.1 秒程度なら待てる。	30~100ms	可能

よって、writeMicroSD 関数はマイコンカー走行中は使えません。例えばマイコンカーが秒速 5m/s で走行して 30ms 間 writeMicroSD 関数の処理に時間がかかったとすると、マイコンカーは 150mm も進んでしまいます。150mm の間、センサの状態を見られず、モータの制御もできないと言うことです。これでは、正確な制御ができません。

実は、writeMicroSD 関数の他にも、microSD ヘデータを書き込む関数があります。次の章では、writeMicroSD 関数を使わずに書き込む方法を紹介します。

※ループの実行時間

マイコンカーのようにリアルタイムで制御する場合、ループの繰り返しは 1ms 間に 10 回程度、遅くとも 1ms 間に 1 回は実行しなければ、正確な制御はできません。また、割り込みは、割り込み周期より短い時間で終わらなければいけません。

<pre>void main(void) { init(); while(1) { プログラム プログラム } }</pre> 	<pre>#pragma interrupt(interrupt_timer0) void interrupt_timer0(void) { プログラム プログラム }</pre> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p>1ms 間隔の割り込みなので、必ず 1ms 以内に終わらさなければいけない。できれば、$500 \mu s$ 以下で終わらせる。</p> </div>
---	---

5. プロジェクト「sd_02」 microSD にデータ記録

5.1 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
- ・マイコンボードのディップスイッチの値

を、10ms ごとに内蔵 RAM に一時的に記録します。512 バイトになったら、microSD に記録します。microSD 記録処理中も 10ms ごとの記録は続けます。

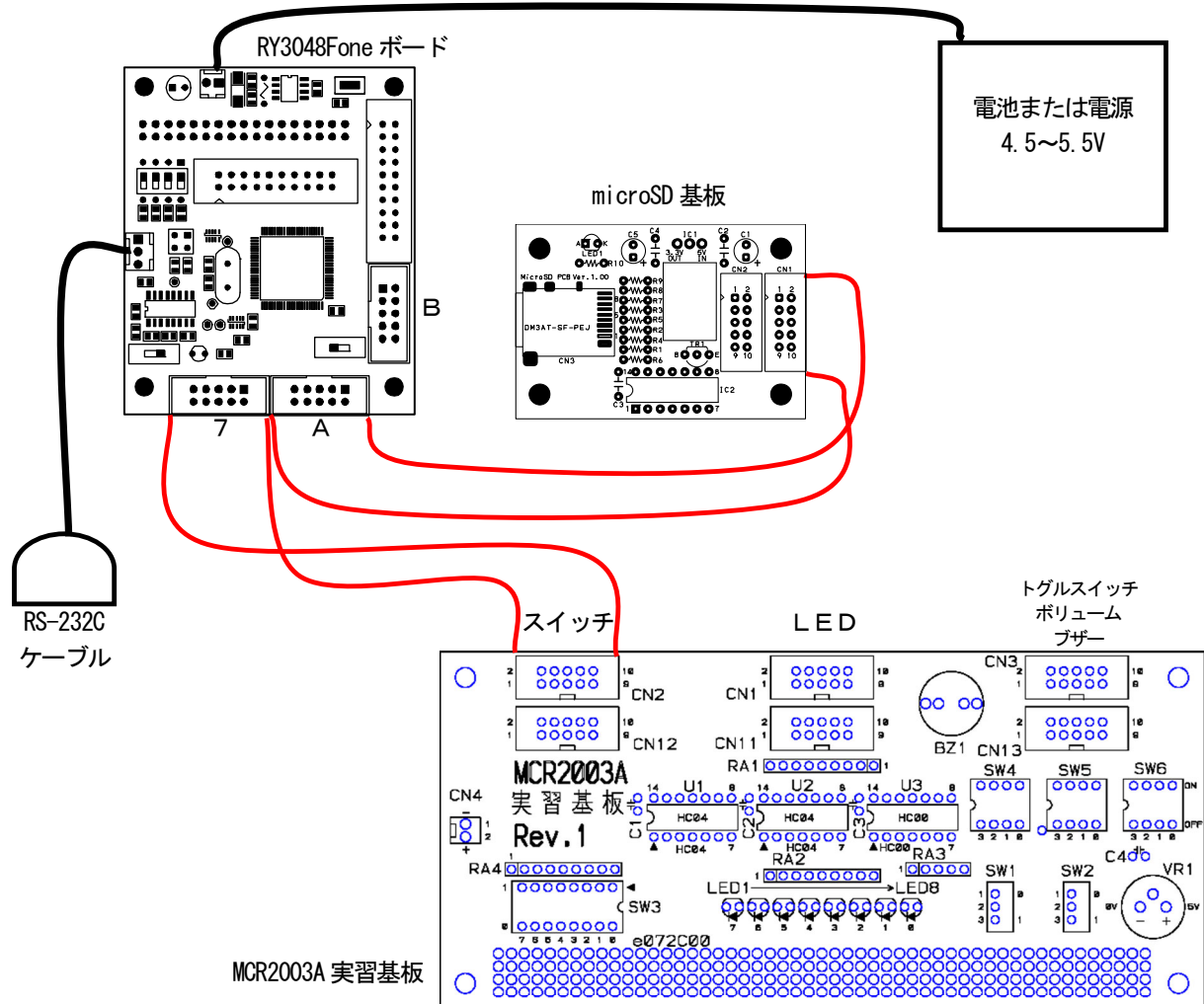
記録終了後、RS232C を通してパソコンへ記録した情報を出力します。

ここでは、writeMicroSD 関数を使用しない書き込み方法を説明します。

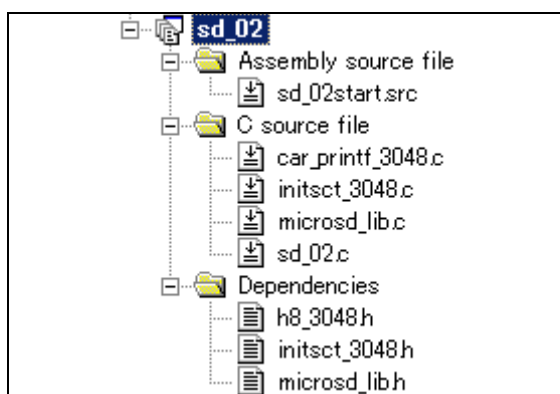
5.2 接続

- ・マイコンボードのポート 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。
- ・マイコンボードのポート A と、microSD 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。



5.3 プロジェクトの構成



	ファイル名	内容
1	sd_02start.src	アセンブリ言語で記述されたアセンブリソースファイルです。このファイルの構造は下記のようになっています。 ベクタアドレス + スタートアップルーチン
2	car_printf_3048.c	<ul style="list-style-type: none"> 通信するための設定 printf 関数の出力先、scanf 関数の入力元を通信にするための設定を行っています。
3	initsct_3048.c	初期値のないグローバル変数(セクション B 領域)、初期値のあるグローバル変数(セクション R 領域)の初期化用です。
4	microsd_lib.c	microSD 基板を制御するための関数が記載されています。
5	sd_02.c	実際に制御するプログラムが書かれています。H8/3048F-ONE の内蔵周辺機能の初期化も行います。
6	h8_3048.h	H8/3048F-ONE の内蔵周辺機能の I/O レジスタを定義したファイルです。
7	initsct_3048.h	initsct_3048.c のヘッダファイルです。
8	microsd_lib.h	microsd_lib.c のヘッダファイルです。

5.4 プログラム

```

1 : /******  

2 : /* microSD基板 演習プログラム「sd_02.c」 */  

3 : /* 2009.06 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microSDに、次のデータを10[ms]ごとに記録します。  

7 : ・ポート7のデータ  

8 : ・CPUボード上のディップスイッチの値  

9 : その後、記録したデータを読み出して、パソコンへ転送します。  

10 : */  

11 :  

12 : /*=====*/  

13 : /* インクルード */  

14 : /*=====*/  

15 : #include <no_float.h> /* stdioの簡略化 最初に置く */  

16 : #include <stdio.h>  

17 : #include <machine.h>  

18 : #include "h8_3048.h"  

19 : #include "microsd_lib.h" /* microSD制御 */  

20 :  

21 : /*=====*/
    
```

microSD を使用するので「microsd_lib.h」をインクルードします。

5. プロジェクト「sd_02」 microSD にデータ記録

```

22 : /* シンボル定義 */
23 : /*=====*/
24 :
25 : /*=====*/
26 : /* プロトタイプ宣言 */
27 : /*=====*/
28 : void init( void );
29 : unsigned char dipsw_get( void );
30 : void convertHexToBin( unsigned char hex, char *s );
31 :
32 : /*=====*/
33 : /* グローバル変数の宣言 */
34 : /*=====*/
35 : unsigned long cnt1; /* 時間計測用 */
36 : int pattern; /* パターン番号 */
37 : int countDown; /* 表示作業用 */
38 :
39 : /* microSD関連変数 */
40 : char msdBuff[ 512 ]; /* 一時記録バッファ */
41 : int msdBuffAddress; /* 一時記録バッファ書込アドレス */
42 : int msdFlag; /* 1:データ記録 0:記録しない */
43 : int msdTimer; /* 取得間隔計算用 */
44 : unsigned long msdStartAddress; /* 記録開始アドレス */
45 : unsigned long msdEndAddress; /* 記録終了アドレス */
46 : unsigned long msdWorkAddress; /* 作業用アドレス */
47 :
48 : /*=====*/
49 : /* メインプログラム */
50 : /*=====*/
51 : void main( void )
52 : {
53 :     int i, j, ret;
54 :     char c;
55 :     char s[10]; /* 16進数→2進数変換用バッファ */
56 :
57 :     /* マイコン機能の初期化 */
58 :     init(); /* 初期化 */
59 :     init_scil( 0x00, 79 ); /* SC11初期化 */
60 :     set_ccr( 0x00 ); /* 全体割り込み許可 */
61 :
62 :     // microSD 書き込み開始アドレス
63 :     // 0x200(512)の倍数に設定する
64 :     msdStartAddress = 0x00000000;
65 :
66 :     // microSD 書き込み終了アドレス
67 :     // 書き込みしたい時間[ms] : x = 10[ms] : 64バイト
68 :     // 500msなら、x = 500 * 64 / 10 = 32000 = 0x7d00
69 :     // 結果は0x200の倍数になるように繰り上げる。よって、7e00にする。
70 :     msdEndAddress = 0x00007e00;
71 :     msdEndAddress += msdStartAddress; /* スタート分足す */
72 :
73 :     /* microSD初期化 */
74 :     ret = initMicroSD(); /* microSD を初期化します。 */
75 :     if( ret != 0x00 ) {
76 :         printf( "%nmicroSD Initialize Error!!%n" ); /* 初期化できず */
77 :         pattern = 99;
78 :     }
79 :
80 :     printf( "Ready " );
81 :
82 :     while( 1 ) {
83 :
84 :         switch( pattern ) {
85 :         case 0:
86 :             /* カウントダウン表示 */
87 :             if( cnt1 / 1000 != countDown ) {
88 :                 countDown = cnt1 / 1000;
89 :                 printf( "%d ", 4 - countDown );
90 :                 if( cnt1 / 1000 == 4 ) { /* 4秒たったら開始 */
91 :                     pattern = 1;
92 :                 }
93 :             }
94 :             break;
95 :
96 :         case 1:
97 :             /* microSDクリア */
98 :             ret = eraseMicroSD( msdStartAddress, msdEndAddress-1 );
99 :             if( ret != 0x00 ) {
100 :                 printf( "%nmicroSD Erase Error!!%n" ); /* エラー */
101 :                 pattern = 99;
102 :                 break;
103 :             }
104 :             /* microSDProcess開始処理 */
105 :             ret = microSDProcessStart( msdStartAddress );
106 :             if( ret != 0x00 ) {
107 :                 printf( "%nmicroSD microSDProcess Error!!%n" ); /* エラー */
108 :                 pattern = 99;
109 :                 break;
110 :             }
111 :             printf( "%n" );
112 :             printf( "Data recording " );

```

microSD 関係の変数です。

microSD に書き込む、開始アドレスを設定します。
「512の倍数」で設定します。

microSD に書き込む、終了アドレスを設定します。「512の倍数」で設定します。

microSD を初期化します。

開始アドレスから終了アドレスをイレース(0でクリア)します。

microSDProcessStart 関数で書き込むアドレスを設定します。


```

113 :     msdBuffAddress = 0;
114 :     msdWorkAddress = msdStartAddress;
115 :     msdFlag = 1;           /* データ記録開始 */
116 :     pattern = 2;
117 :     cnt1 = 0;
118 :     break;
119 :
120 : case 2:
121 :     /* データ記録中 記録は割り込みの中で行う */
122 :     /* 書き込み終了アドレスになると、割り込み内でmsdFlagが0になる */
123 :     if( msdFlag == 0 ) {
124 :         pattern = 3;
125 :         break;
126 :     }
127 :
128 :     /* 時間表示 */
129 :     if( cnt1 / 1000 != countDown ) {
130 :         countDown = cnt1 / 1000;
131 :         printf( "%d ", countDown );
132 :     }
133 :     break;
134 :
135 : case 3:
136 :     /* 最後のデータが書き込まれるまで待つ*/
137 :     if( checkMicroSDProcess() == 11 ) {
138 :         microSDProcessEnd(); /* microSDProcess終了処理 */
139 :         pattern = 4;
140 :     }
141 :     break;
142 :
143 : case 4:
144 :     /* 終了処理が終わるまで待つ*/
145 :     if( checkMicroSDProcess() == 0 ) {
146 :         pattern = 5;
147 :     }
148 :     break;
149 :
150 : case 5:
151 :     /* タイトル転送、準備 */
152 :     printf( "\n\n" );
153 :     printf( "sd_01 Data Out\n" );
154 :     printf( "P7 Data, DIP SW Data\n" );
155 :
156 :     msdWorkAddress = msdStartAddress; /* 読み込み開始アドレス */
157 :     pattern = 6;
158 :     break;
159 :
160 : case 6:
161 :     /* microSDよりデータ読み込み */
162 :     if( msdWorkAddress >= msdEndAddress ) {
163 :         /* 書き込み終了アドレスになったら、終わり */
164 :         pattern = 99;
165 :         break;
166 :     }
167 :     ret = readMicroSD( msdWorkAddress , msdBuff );
168 :     if( ret != 0x00 ) {
169 :         /* 読み込みエラー */
170 :         printf( "\nmicroSD Read Error!!\n" );
171 :         pattern = 99;
172 :         break;
173 :     } else {
174 :         /* エラーなし */
175 :         msdWorkAddress += 512; /* microSDのアドレスを+512する */
176 :         msdBuffAddress = 0; /* 配列からの読み込み位置を0に */
177 :         pattern = 7;
178 :     }
179 :     break;
180 :
181 : case 7:
182 :     /* データ転送 */
183 :     convertHexToBin( msdBuff[msdBuffAddress+0], s );
184 :     printf( "\n%s", "%02x\n",
185 :         s,
186 :         (unsigned char)msdBuff[msdBuffAddress+1]
187 :     );
188 :     msdBuffAddress += 64;
189 :
190 :     if( msdBuffAddress >= 512 ) {
191 :         pattern = 6;
192 :     }
193 :     break;
194 :
195 : case 99:
196 :     /* 終了 */
197 :     break;
198 :
199 : default:
200 :     /* どれでもない場合は待機状態に戻す */
201 :     pattern = 0;
202 :     break;
203 : }

```

msdFlag を 1 にすると、データ記録を開始します。記録処理は割り込み内で行います。

終了アドレスまで書き込むと、割り込み内で msdFlag を 0 にします。0 かチェックして、0 になったらパターン 3 へ移ります。

書き込み処理が終わるまで待ちます。
書き込みが終わったら、microSDProcessEnd 関数で終了処理を行います。

終了処理が終わるまで待ちます。

転送アドレス > 書き込み終了アドレスになったら、転送終了です。

microSD から、記録したデータを読み込みます。

microSD からデータを読み込んだら、パターン 7 へ進みます。

msdBuff 配列からデータを読み込んで、printf 文でパソコンへ出力します。

msdBuff からすべてのデータを読み込むと、パターン 6 へ戻ります。

5. プロジェクト「sd_02」 microSD にデータ記録

```

204 :     }
205 : }
206 :
207 : /*****
208 : /* H8/3048F-ONE 内蔵周辺機能 初期化 */
209 : /*****/
210 : void init( void )
211 : {
212 :     /* I/Oポートの入出力設定 */
213 :     P1DDR = 0xff;
214 :     P2DDR = 0xff;
215 :     P3DDR = 0xff;
216 :     P4DDR = 0xff;
217 :     P5DDR = 0xff;
218 :     P6DDR = 0xf0; /* CPU基板上的DIP SW */
219 :     P8DDR = 0xff;
220 :     P9DDR = 0xf7; /* 通信ポート */
221 :     PADDR = 0xf7; /* microSD基板 */
222 :     PBDDR = 0xff;
223 :     /* ※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 */
224 :
225 :     /* ITUO 1ms ほどの割り込み */
226 :     ITUO_TCR = 0x23;
227 :     ITUO_GRA = 3071;
228 :     ITUO_IER = 0x01;
229 :
230 :     /* ITUのカウントスタート */
231 :     ITU_STR = 0x01;
232 : }
233 :
234 : /*****
235 : /* ITUO 割り込み処理 */
236 : /*****/
237 : #pragma interrupt( interrupt_timer0 )
238 : void interrupt_timer0( void )
239 : {
240 :     unsigned int i;
241 :     char *p;
242 :
243 :     ITUO_TSR &= 0xfe; /* フラグクリア */
244 :     cnt1++;
245 :
246 :     microSDProcess(); /* microSD 間欠書き込み処理 */
247 :
248 :     /* microSD記録処理 */
249 :     if( msdFlag == 1 ) {
250 :         /* 記録間隔のチェック */
251 :         msdTimer++;
252 :         if( msdTimer >= 10 ) {
253 :             msdTimer = 0;
254 :             p = msdBuff + msdBuffAddress;
255 :
256 :             /* RAMに記録 ここから */
257 :             *p++ = P7DR;
258 :             *p++ = dipsw_get();
259 :             /* RAMに記録 ここまで */
260 :
261 :             msdBuffAddress += 64; /* RAMの記録アドレスを次へ */
262 :
263 :             if( msdBuffAddress >= 512 ) {
264 :                 /* 512個になったら、microSDに記録する */
265 :                 msdBuffAddress = 0;
266 :                 setMicroSDdata( msdBuff );
267 :                 msdWorkAddress += 512;
268 :                 if( msdWorkAddress >= msdEndAddress ) {
269 :                     /* 記録処理終了 */
270 :                     msdFlag = 0;
271 :                 }
272 :             }
273 :         }
274 :     }
275 : }
276 :
277 : /*****
278 : /* デイップスイッチ値読み込み */
279 : /* 戻り値 スイッチ値 0~15 */
280 : /*****/
281 : unsigned char dipsw_get( void )
282 : {
283 :     unsigned char sw;
284 :
285 :     sw = ~P6DR; /* デイップスイッチ読み込み */
286 :     sw &= 0x0f;
287 :
288 :     return sw;
289 : }
290 :
291 : /*****
292 : /* 1 6 進数→2 進数変換 Ver2.00 */
293 : /* 引数 1 6 進数データ、変換後のデータ格納アドレス */
294 : /* 戻り値 なし */

```

microSD 基板と接続しているポート A の入出力設定を変更します。

microSDProcess 関数は 1ms ごとに実行します。

10ms ごとに msdBuff 配列(RAM)にデータを記録していきます。
512 バイト分たまったら、setMicroSDdata 関数で、microSD に書き込む準備をします。
実際の書き込み作業は microSDProcess 関数で行います。

microSD に書き込むアドレスが、終了アドレスになったら、自動で記録処理を終了します。

```
295 : /*****/
296 : void convertHexToBin( unsigned char hex, char *s )
297 : {
298 :     int    i;
299 :
300 :     for( i=0; i<8; i++ ) {
301 :         if( hex & 0x80 ) {
302 :             *s++ = '1';           /* "1"のときの変換データ */
303 :         } else {
304 :             *s++ = '0';           /* "0"のときの変換データ */
305 :         }
306 :         hex <<= 1;
307 :     }
308 :     *s = '\0';
309 : }
310 :
311 : /*****/
312 : /* end of file */
313 : /*****/
```

5.5 setMicroSDdata関数とmicroSDProcess関数

5.5.1 概要

マイコンカーは、常に次の作業を行っています。

- 各種センサ(コースセンサ、ロータリエンコーダ、上り坂検出スイッチなど)の読み込み
- 駆動モータの制御
- サーボ(ステアリングモータ)の制御

データ記録はデバッグの一種なので、**マイコンカー制御にできるだけ影響が無いように記録作業を行わなければいけません**。データ記録を行うためにマイコンカー制御がおろそかになっては意味がありません。

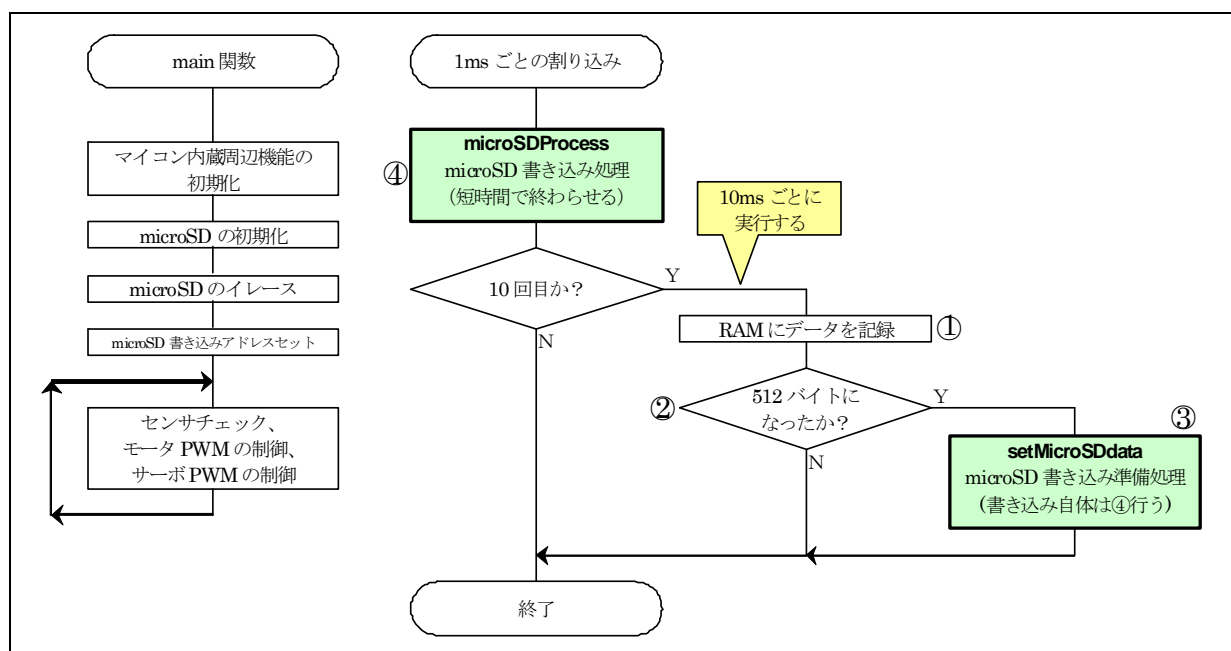
データの記録は割り込み内で行い、できるだけ時間をかけないように処理します。今回は、setMicroSDdata 関数と microSDProcess 関数をペアで使います。

5.5.2 プログラムの流れ

今回のプログラムは、RAM にデータを記録する作業を 10ms ごとに行います。割り込みは 1ms ごとなので、割り込みが 10 回目かどうか確認して、10 回目であれば 10ms 経ったと判断して、RAM に現在の状態を記録します (①部分)。

RAM に記録したデータが 512 バイトかどうかチェック (②部分)、なったなら **setMicroSDdata 関数** で RAM のデータを microSD に書き込む準備をします (③部分)。あくまで準備だけで書き込み作業は次で説明する **microSDProcess 関数** で行います。

setMicroSDdata 関数 で書き込み準備をしたら、**microSDProcess 関数** で実際の書き込み処理を行います (④部分)。**microSDProcess 関数** は、「microSD への書き込み処理を短時間だけ行ってすぐに終了」を何度も繰り返すことにより **writeMicroSD 関数** と同じことを行います。



5.5.3 各関数の処理内容

それぞれの関数は、下表のような処理を行います。

関数	内容
writeMicroSD 関数	microSD に 512 バイトのデータを一気に書き込みます。約 30~50ms かかり、その間は何も処理ができません。
setMicroSD 関数	microSD に書き込む準備だけを行います。実行時間は、400~500 μs です。
microSDProcess 関数	setMicroSD 関数でセットされたデータを、実際に microSD に書き込みます。書き込み中の実行時間は、約 500 μs です。約 70~80 回実行すると 512 バイト書き込むことができます。microSDProcess 関数は、1ms ごとに実行されるため、 80ms(80回実行)で 512 バイト書き込めることとなります。 ちなみに書き込むデータがないときは、約 6.5 μs で処理を終わらせます。

※時間はすべて実測です。

簡単にまとめると、次のような関係になります。

$$\text{writeMicroSD 関数(512 バイト書き込み)} = \text{setMicroSDdata 関数} \times 1 \text{ 回} + \text{microSDProcess 関数} \times \text{約 } 70 \sim 80 \text{ 回}$$

5.6 プログラムの解説

5.6.1 変数

```

32 : /*=====*/
33 : /* グローバル変数の宣言 */
34 : /*=====*/
35 : unsigned long cnt1; /* 時間計測用 */
36 : int pattern; /* パターン番号 */
37 : int countDown; /* 表示作業用 */
38 :
39 : /* microSD 関連変数 */
40 : char msdBuff[ 512 ]; /* 一時記録バッファ */
41 : int msdBuffAddress; /* 一時記録バッファ書込アドレス */
42 : int msdFlag; /* 1:データ記録 0:記録しない */
43 : int msdTimer; /* 取得間隔計算用 */
44 : unsigned long msdStartAddress; /* 記録開始アドレス */
45 : unsigned long msdEndAddress; /* 記録終了アドレス */
46 : unsigned long msdWorkAddress; /* 作業用アドレス */

```

それぞれの変数は、次のような意味です。

変数名／配列名	意味	内容
cnt1	カウンタ	1ms ごとに+1 する変数です。変数の値をチェックすることにより経過時間が分かります。
pattern	パターン	この変数によりどのプログラムを実行するか決めます。
countDown	カウントダウン	記録を開始するまでの時間、または記録中に printf 文でカウントダウンして後何秒か知らせています。その時に使用します。
msdBuff[512]	microSD 用バッファ	microSD に書き込むデータを記録したり、読み込んだデータを記録する配列です。
msdBuffAddress	msdBuff 配列の処理アドレス	msdBuff 配列に値を書き込んだり読み込むときの位置を指定します。
msdFlag	microSD 用フラグ	1 ならデータを記録します。0 なら記録しません。
msdTimer ※	microSD 用タイマ	割り込みは 1ms 間隔ですが、記録の間隔が割り込み間隔と違うとき、この変数を補助的に使用します。 例えば、割り込みごとに+1 して、10 なら実行するようにすれば 10ms ごとに処理することになります。
msdStartAddress ※	記録開始アドレス	microSD ヘデータを書き込むときの「開始アドレス」を指定します。512 の倍数で指定します。
msdEndAddress ※	記録終了アドレス	microSD ヘデータを書き込むときの「終了アドレス+1」を指定します。512 の倍数で指定します。
msdWorkAddress	作業用アドレス	microSD ヘデータを書き込んだり読み込んだりするときにアドレスを指定します。

※がプログラムによって代入する値やチェックする値を変える変数です。

5.6.2 main関数の開始部分

```

51 : void main( void )
52 : {
53 :     int    i, j, ret;
54 :     char   c;
55 :     char   s[10];           /* 16進数→2進数変換用バッファ */
56 :
57 :     /* マイコン機能の初期化 */
58 :     init();                 /* 初期化 */
59 :     init_scil( 0x00, 79 );  /* SCI1 初期化 */
60 :     set_ccr( 0x00 );       /* 全体割り込み許可 */
61 :
62 :     // microSD 書き込み開始アドレス
63 :     // 0x200(512)の倍数に設定する
64 :     msdStartAddress = 0x00000000;
65 :
66 :     // microSD 書き込み終了アドレス
67 :     // 書き込みしたい時間[ms] : x = 10[ms] : 64バイト
68 :     // 5000ms なら、x = 5000 * 64 / 10 = 32000 = 0x7d00
69 :     // 結果は 0x200 の倍数になるように繰り上げする。よって、7e00 にする。
70 :     msdEndAddress = 0x00007e00;
71 :     msdEndAddress += msdStartAddress; /* スタート分足す */
72 :
73 :     /* microSD 初期化 */
74 :     ret = initMicroSD();
75 :     if( ret != 0x00 ) {
76 :         printf( "%nmicroSD Initialize Error!!%n" ); /* 初期化できず */
77 :         pattern = 99;
78 :     }
79 :
80 :     printf( "Ready " );

```

64行で microSD に書き込む開始アドレスを設定します。

70行で microSD に書き込む容量を指定します。71行でスタート分を加えて終了アドレスにしています。今回、データの記録の条件を次のようにしました。

- データ記録の間隔 … 10ms ごと
- データ記録数 …………… 64 バイト
- データ記録時間 ……… 5 秒

microSD に確保しなければいけない容量は、次のようになります。

$$\text{容量} = \text{記録したい時間[ms]} \div \text{記録する間隔[ms]} \times 1 \text{ 回に記録するバイト数}$$

よって、容量は次のとおりです。

$$\begin{aligned} \text{容量} &= \text{記録したい時間[ms]} \div \text{記録する間隔[ms]} \times 1 \text{ 回に記録するバイト数} \\ &= 5,000 \div 10 \times 64 \\ &= 32,000 \end{aligned}$$

値は、512 の倍数にしなければいけません。512 の倍数かどうか確かめます。

$$32,000 \div 512 = 62 \text{ 余り } 256$$

余りがあるので、512 の倍数ではありません。答えを 1 つ足して、512 でかけた値を容量とします。よって、

$$63 \times 512 = 32,256$$

プログラムでは 32,256 を 16 進数に変換した 0x7e00 を設定しています。10 進数で記述しても問題はありませ
ん。

5.6.3 パターン 0:スタート

```

80 :    printf( "Ready " );
81 :
82 :    while( 1 ) {
83 :
84 :        switch( pattern ) {
85 :        case 0:
86 :            /* カウントダウン表示 */
87 :            if( cnt1 / 1000 != countDown ) {
88 :                countDown = cnt1 / 1000;
89 :                printf( "%d ", 4 - countDown );
90 :                if( cnt1 / 1000 == 4 ) { /* 4 秒たったら開始          */
91 :                    pattern = 1;
92 :                }
93 :            }
94 :            break;

```

リセット後、データ記録をすぐに始めてしまうと最初の方のディップスイッチの変更などができないので、プログラ
ムスタート後 3 秒待ちます。そのとき、何もしないとプログラムが動作していないと思われるため、3→2→1→0、
というようにカウントダウン処理を行います。3 秒たったら、パターン 1 へ移ります。

5.6.4 パターン 1: microSD クリア、書き込みアドレスセット

```

96 :    case 1:
97 :        /* microSD クリア */
98 :        ret = eraseMicroSD( msdStartAddress, msdEndAddress-1 );
99 :        if( ret != 0x00 ) {
100 :            printf( "\nmicroSD Erase Error!!\n" ); /* エラー          */
101 :            pattern = 99;
102 :            break;
103 :        }
104 :        /* microSDProcess 開始処理 */
105 :        ret = microSDProcessStart( msdStartAddress );
106 :        if( ret != 0x00 ) {
107 :            printf( "\nmicroSD microSDProcess Error!!\n" ); /* エラー  */
108 :            pattern = 99;
109 :            break;
110 :        }
111 :        printf( "\n" );
112 :        printf( "Data recording " );

```


5. プロジェクト「sd_02」 microSD にデータ記録

```

113 :      msdBuffAddress = 0;
114 :      msdWorkAddress = msdStartAddress;
115 :      msdFlag = 1;          /* データ記録開始          */
116 :      pattern = 2;
117 :      cnt1 = 0;
118 :      break;

```

98 行で、microSD の開始アドレスから終了アドレスまでイレースします。

105 行で、microSD へ書き込む開始アドレスをセットします。

113 行で、msdBuff 配列 (RAM) を参照する変数を 0 にクリアしています。

114 行で、microSD の作業アドレスを開始アドレスに設定します。実際の書き込みアドレスは、105 行でセットしていますが、書き込み終了の計算用としてセットしています。

115 行でフラグを 1 にします。この行以降の 1ms ごとの割り込みから、記録が開始されます。

5.6.5 パターン 2: データ記録中

```

120 :      case 2:
121 :          /* データ記録中 記録は割り込みの中で行う */
122 :          /* 書き込み終了アドレスになると、割り込み内で msdFlag が 0 になる */
123 :          if( msdFlag == 0 ) {
124 :              pattern = 3;
125 :              break;
126 :          }
127 :
128 :          /* 時間表示 */
129 :          if( cnt1 / 1000 != countDown ) {
130 :              countDown = cnt1 / 1000;
131 :              printf( "%d ", countDown );
132 :          }
133 :          break;

```

123 行で、フラグが 0 かどうかチェックします。書き込み終了アドレスになると、割り込み内で msdFlag が 0 になるので、if 文で 0 かどうかチェックします。0 になったなら、書き込み終了と判断してパターン 3 へ移ります。

129~132 行は、記録中何も表示していないと記録しているのか分からないため、1 秒ごとにカウント表示させています。

5.6.6 パターン 3: 最後のデータが書き込まれるまで待つ

```

135 :      case 3:
136 :          /* 最後のデータが書き込まれるまで待つ*/
137 :          if( checkMicroSDProcess() == 11 ) {
138 :              microSDProcessEnd();    /* microSDProcess 終了処理    */
139 :              pattern = 4;
140 :          }
141 :          break;

```

msdFlag が 0 になっても、setMicroSDdata 関数がまだ書き込み作業をしているかもしれません。137 行で checkMicroSDProcess 関数を実行して書き込み作業が終わったかどうかチェックします。11 なら書き込み終了です。138 行で、microSDProcessEnd 関数を実行して、書き込み作業を終了します。その後、パターン 4 へ移ります。

5.6.7 パターン 4:終了処理が終わるまで待つ

```
143 :     case 4:
144 :         /* 終了処理が終わるまで待つ*/
145 :         if( checkMicroSDProcess() == 0 ) {
146 :             pattern = 5;
147 :         }
148 :         break;
```

145 行で、microSDProcessEnd 関数を実行し終わるまで待ちます。checkMicroSDProcess 関数を実行して戻り値が 0 なら、完了です。

5.6.8 パターン 5:タイトル転送、準備

```
150 :     case 5:
151 :         /* タイトル転送、準備 */
152 :         printf( "¥n¥n" );
153 :         printf( "sd_01 Data Out¥n" );
154 :         printf( "P7 Data,DIP SW Data¥n" );
155 :
156 :         msdWorkAddress = msdStartAddress; /* 読み込み開始アドレス */
157 :         pattern = 6;
158 :         break;
```

転送する準備を行います。

156 行で microSD から読み込むアドレスをセットしています。セット後、パターン 6 に移ります。

5.6.9 パターン 6:microSDよりデータ読み込み

```
160 :     case 6:
161 :         /* microSD よりデータ読み込み */
162 :         if( msdWorkAddress >= msdEndAddress ) {
163 :             /* 書き込み終了アドレスになったら、終わり */
164 :             pattern = 99;
165 :             break;
166 :         }
167 :         ret = readMicroSD( msdWorkAddress , msdBuff );
168 :         if( ret != 0x00 ) {
169 :             /* 読み込みエラー */
170 :             printf( "¥nmicroSD Read Error!!¥n" );
171 :             pattern = 99;
172 :             break;
173 :         } else {
174 :             /* エラーなし */
175 :             msdWorkAddress += 512; /* microSD のアドレスを+512 する */
176 :             msdBuffAddress = 0; /* 配列からの読み込み位置を 0 に */
177 :             pattern = 7;
178 :         }
179 :         break;
```

5. プロジェクト「sd_02」 microSD にデータ記録

162 行で、現在読み込んでいるアドレス(msdWorkAddress)が書き込み終了アドレス(msdEndAddress)より大きいかチェックします。大きいなら読み込むデータはないので、終了します。

167 行で、microSD から 512 バイト読み込みます。読み込みエラーなら終了します。正しく読み込めたならパターン 7 へ移ります。

5.6.10 パターン 7: パソコンへデータ転送

```

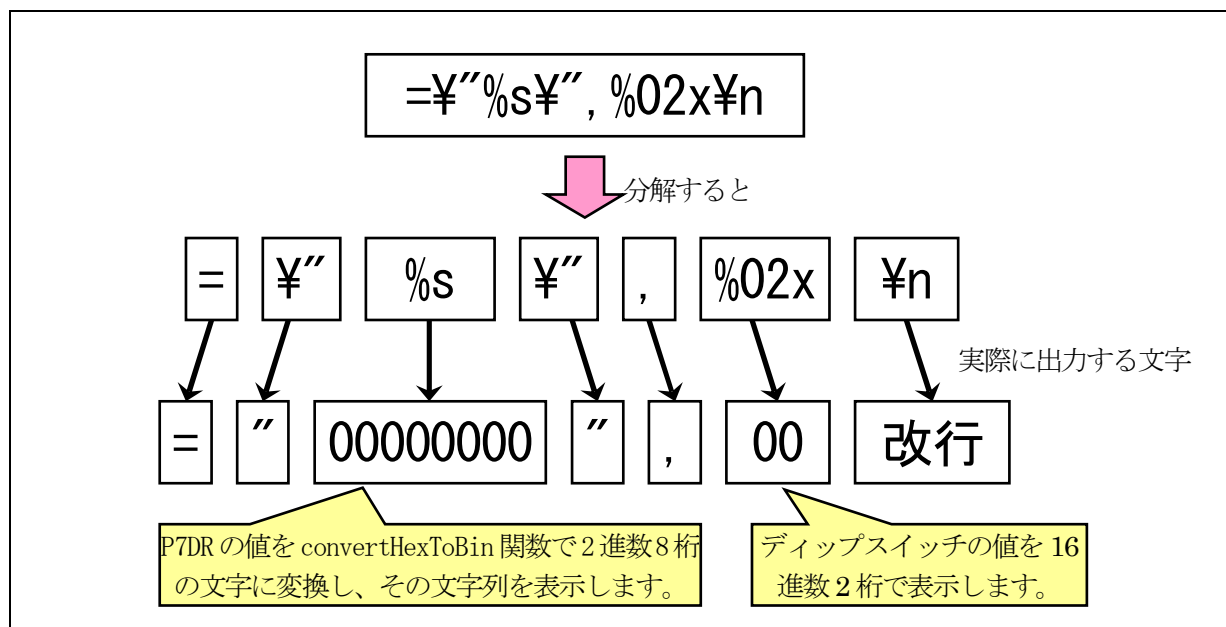
181 :     case 7:
182 :         /* データ転送 */
183 :         convertHexToBin( msdBuff[msdBuffAddress+0], s );
184 :         printf( "=¥"%s¥", %02x¥n",
185 :             s,
186 :             (unsigned char)msdBuff[msdBuffAddress+1]
187 :         );
188 :         msdBuffAddress += 64;
189 :
190 :         if( msdBuffAddress >= 512 ) {
191 :             pattern = 6;
192 :         }
193 :         break;

```

記録したデータが msbBuff 配列に格納されています。それらを読み込んで printf 文でパソコンへ出力します。188 行で、msdBuffAddress 変数に 1 回に記録するデータ数分を足して次に読み込む位置をセットします。**1 回の記録数を換えたときは、この数値も変更します。**

190 行で、msdBuffAddress 変数が 512 以上なら、msbBuff 配列からデータをすべて呼び出したのでパターン 6 に戻って、microSD から次のデータを読み込みます。

パソコンへの転送書式は次のようになります。



5.6.11 パターン 99:終了

```
180 :     case 99:
181 :         /* 終了 */
182 :         break;
```

何もありません。データ転送を終えた状態です。

5.6.12 割り込み処理

```
237 : #pragma interrupt( interrupt_timer0 )
238 : void interrupt_timer0( void )
239 : {
240 :     unsigned int i;
241 :     char *p;
242 :
243 :     ITU0_TSR &= 0xfe;          /* フラグクリア          */
244 :     cnt1++;
245 :
246 :     microSDProcess();      /* microSD 間欠書き込み処理 */
247 :
248 :     /* microSD 記録処理 */
249 :     if( msdFlag == 1 ) {
250 :         /* 記録間隔のチェック */
251 :         msdTimer++;
252 :         if( msdTimer >= 10 ) {
253 :             msdTimer = 0;
254 :             p = msdBuff + msdBuffAddress;
255 :
256 :             /* RAM に記録 ここから */
257 :             *p++ = P7DR;
258 :             *p++ = dipsw_get();
259 :             /* RAM に記録 ここまで */
260 :
261 :             msdBuffAddress += 64; /* RAM の記録アドレスを次へ */
262 :
263 :             if( msdBuffAddress >= 512 ) {
264 :                 /* 512 個になったら、microSD に記録する */
265 :                 msdBuffAddress = 0;
266 :                 setMicroSDdata( msdBuff );
267 :                 msdWorkAddress += 512;
268 :                 if( msdWorkAddress >= msdEndAddress ) {
269 :                     /* 記録処理終了 */
270 :                     msdFlag = 0;
271 :                 }
272 :             }
273 :         }
274 :     }
275 : }
```

記録する間隔

記録内容

1回に記録する数

246 行で、microSDProcess 関数を 1ms ごとに実行します。

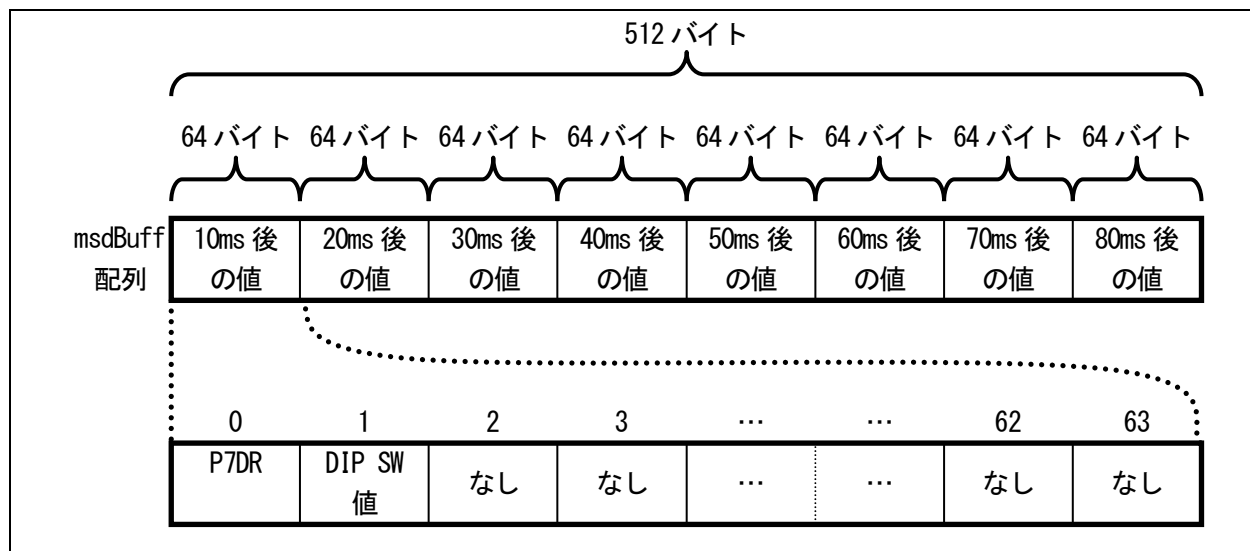
249 行で、フラグが 1 かどうかチェックします。1 なら microSD への記録処理を実行します。

252 行で記録間隔のチェックをしています。今回は、msdTimer が 10 以上になったなら、すなわち 10ms たった

5. プロジェクト「sd_02」 microSD にデータ記録

なら記録処理を行います。

257 行、258 行が msdBuff 配列に記録している内容です。今回はポート 7 の状態、ディップスイッチの状態を記録しています。1 回に 64 バイト分のデータを記録できますが、今回はこの 2 つのみ記録しています。記録イメージは次のようです。



ポイントは、80ms 間隔で 512 バイト記録できるということです。この値を基本として時間を細かく区切り、記録するバイト数を減らせば、細かい間隔で記録することができます。代表的な記録時間、記録数を下表に示します。

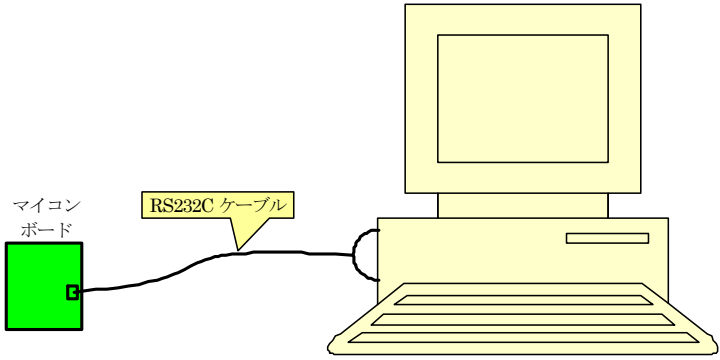
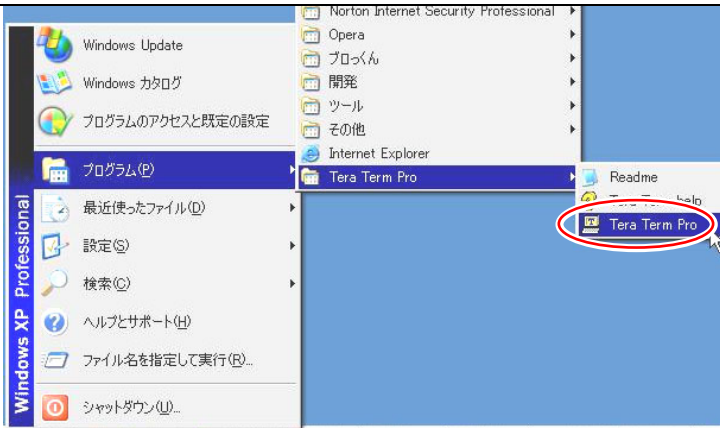
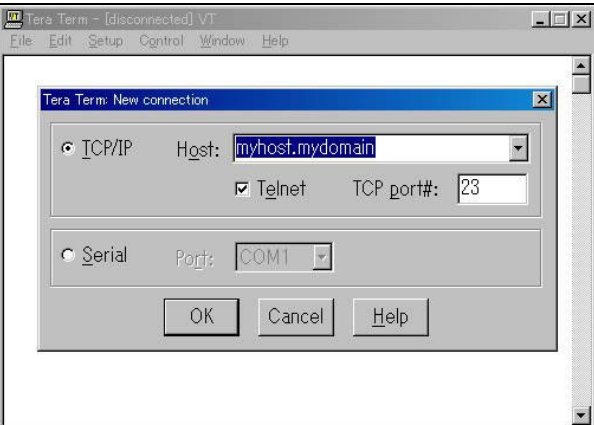
記録間隔	記録数	備考
80ms	512 バイト	
40ms	256 バイト	
20ms	128 バイト	
10ms	64 バイト	今回の記録間隔、記録容量です。
5ms	32 バイト	
2.5ms		割り込みは、1ms 間隔なので 2.5ms ごとに記録はできません。ただし、2ms 後、3ms 後を交互にすれば、擬似的に 5ms 間で 2 回実行することになります。
2ms	12 バイト	12×40 回=480 バイトで 32 バイトはあまりになります。
1ms	6 バイト	6×80 回=480 バイトで 32 バイトはあまりになります。

266 行で、記録データが 512 バイトになったら setMicroSDdata 関数で microSD へ書き込み準備処理を行います。

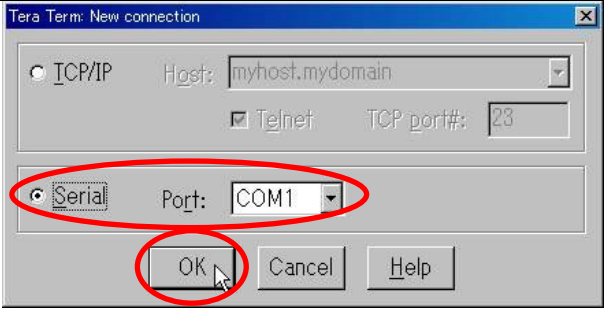
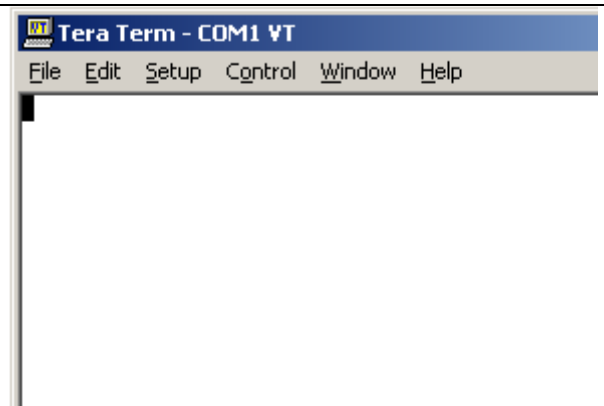
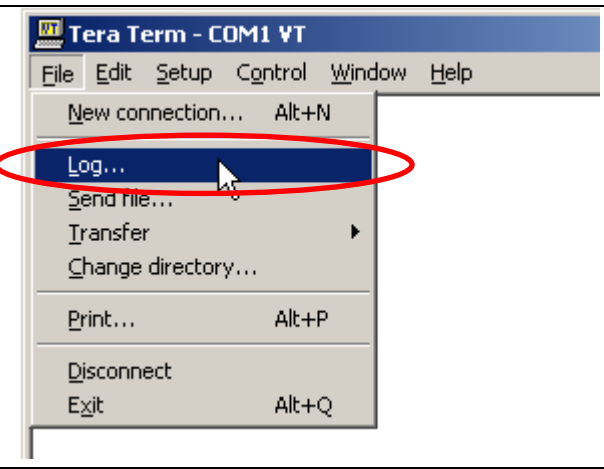
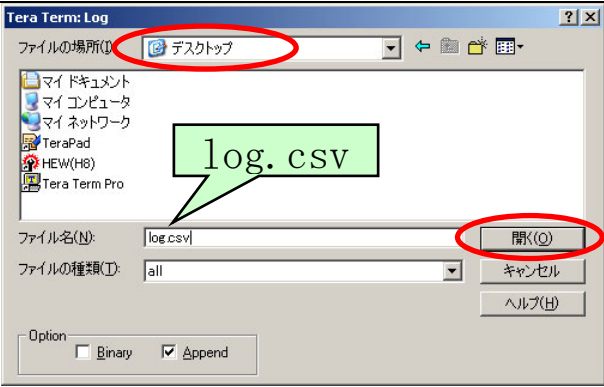
268 行で、記録終了アドレスになったかチェックして、なったならば msdFlag を 0 にして、記録処理を終了します。

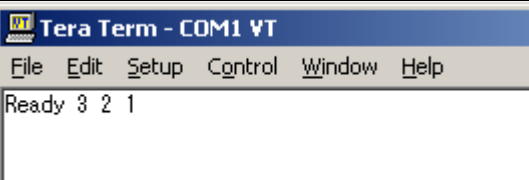
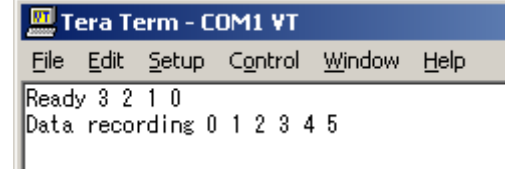
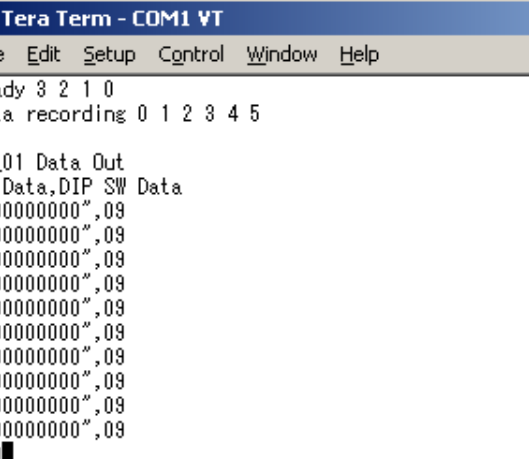
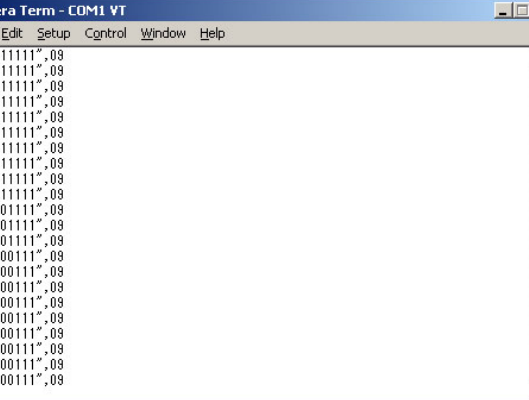
5.7 データの取り込み方

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。

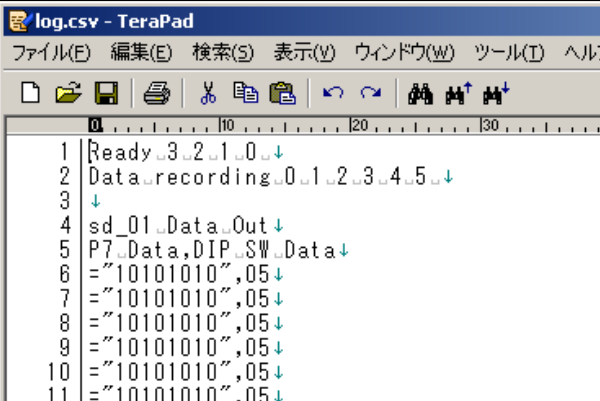
1		<p>プロジェクト「sd_02」をビルドして、「sd_02.mot」ファイルをマイコンボードに書き込んでください。書き込みができれば、書き込みスイッチを FWE の逆側にして、電源を切っておきます。マイコンボードとパソコン間の RS232C ケーブルは繋いだままにしておきます。</p>
2		<p>Tera Term Pro を立ち上げます。Tera Term Pro をまだインストールしていない場合は、H8/3048F-ONE 実習マニュアルのプロジェクト「sio」にある Tera Term Pro のインストール欄を参照してインストールしてください。</p>
3		<p>接続先を確認する画面が表示されます。</p>

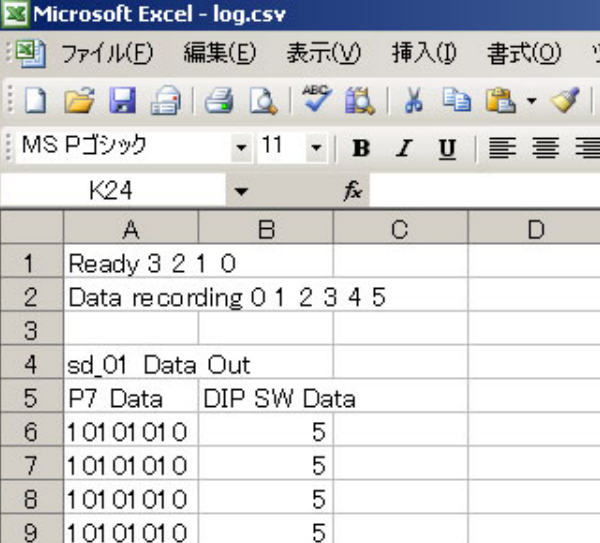
5. プロジェクト「sd_02」 microSD にデータ記録

<p>4</p>		<p>「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。</p>
<p>5</p>		<p>立ち上がりました。</p>
<p>6</p>		<p>受信データをファイルに保存します。「File→Log」を選択します。</p>
<p>7</p>		<p>保存ファイル名を入力します。ここでは「log.csv」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。 ※拡張子は必ず「csv」にします。</p>

8		<p>マイコンボードの電源を入れると、「Ready」と表示され、カウントダウンが開始されます。「0」が表示されてから5秒間、記録されます。</p>
9		<p>Data recording 0 から Data recording 0 1 2 3 4 5 と表示されている間の5秒間、ポート7に繋がっている実習基板のディップスイッチの値と、マイコンボード上のディップスイッチの値が10ms ごとに記録されます。この間に、実習基板のディップスイッチとマイコンボード上のディップスイッチの状態を適当に変えてみてください。</p>
10		<p>5秒経つと、自動的に記録したデータが送られてきます。データが止まるまで待ちます。</p>
11		<p>データが止まったら転送終了です。マイコンボードの電源を切って、Tera Term Pro を終了してください。</p>

5. プロジェクト「sd_02」 microSD にデータ記録

12		<p>Tera Term Pro の log 操作で保存したファイルをエディタで開いてみました。マニュアルどおりのファイル名で保存した場合、「log.csv」を開きます。</p>
----	---	---

13		<p>パソコンにエクセルなどの表計算ソフトがインストールされている場合、「log.csv」をダブルクリックするとソフトが立ち上がります。ソフトをインストールしているにも関わらず、立ち上がらない場合はソフトを立ち上げてから読み込んでください。</p>
----	--	--

※注意

TeraTermPro は、**受信前**に「Flie→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りません。データを受信してから、「Flie→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。

5.8 int型、long型を記録するには

microSD 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、次のように上位 8bit と下位 8bit に分けて保存します。

```
i = int 型のデータ
*p++ = i >> 8;          /* 上位 8bit 保存 */
*p++ = i & 0xff;       /* 下位 8bit 保存 */
```

呼び出すときは、次のようにします。□部分はそれぞれのデータ格納先を呼び出してください。

```
i = (int)((unsigned char)msdBuff[msdBuffAddress+2]*0x100 +
         (unsigned char)msdBuff[msdBuffAddress+3]);
```

microSD に long 型(32bit 幅)を保存する場合は、次のように 4 分割して保存します。保存する変数は、lEncoderTotal 変数を例にします。

```
l = lEncoderTotal;          /* 走行距離          */
*p++ = l >> 24;
*p++ = (l & 0x00ff0000) >> 16;
*p++ = (l & 0x0000ff00) >> 8;
*p++ = l & 0x000000ff;
```

呼び出すときは、次のようにします。□部分はそれぞれのデータ格納先を呼び出してください。

```
l = (long)(unsigned char)msdBuff[msdBuffAddress+2]*0x1000000;
l += (long)(unsigned char)msdBuff[msdBuffAddress+3]*0x10000;
l += (long)(unsigned char)msdBuff[msdBuffAddress+4]*0x100;
l += (long)(unsigned char)msdBuff[msdBuffAddress+5];
```

ちなみに printf 文で出力するとき、この変数は long 型なので変換指定文字は「%ld」(エルとディ)を使用します。

```
printf( "%ld¥n", l );
```

5.9 演習

- (1) データ記録間隔は 10ms のままで、記録時間を 10 秒に変更しましょう。
- (2) データ記録間隔を 5ms にしてみましょう。記録時間は 10 秒で変更しません。
- (3) 通信速度を 9600bps から 38400bps に変更して、通信ソフトで受信してみましょう。
通信速度の設定については、「H8/3048F-ONE 実習マニュアル(ルネサス統合開発環境版)」のプロジェクト「sio」を参照してください。

5.10 演習の回答例

(1) 記録時間の変更

40 ページの式に当てはまると、次のようになります。

$$\begin{aligned} \text{容量} &= \text{記録したい時間[ms]} \div \text{記録する間隔[ms]} \times 1 \text{ 回に記録するバイト数} \\ &= 10,000 \div 10 \times 64 \\ &= 64,000 \end{aligned}$$

512 の倍数か調べます。

$$64000 \div 512 = 125 \text{ 余り } 0$$

よって、今回はこのまま使用します。余りが出た場合は、あまりが出ない値に切り上げます。

70 : msdEndAddress = **0x0000fa00**; // 10 進数表記で 64000 でも構いません

(2) 記録間隔の変更

記録間隔を変更するときの変更する行と数値は下表のようです。今回の問題の回答例は、下表の 5ms の行 (ゴシック体部分) です。

記録間隔	188 行の変更	190 行の変更	252 行の変更	261 行の変更	263 行の変更
80ms	512	512	80	512	512
40ms	256	512	40	256	512
20ms	128	512	20	128	512
10ms(変更前)	64	512	10	64	512
5ms (今回の回答)	32	512	5	32	512
2ms	12	480	2	12	480
1ms	6	480	1	6	480

(2) 通信速度の変更

今回は、38,400bps に設定します。

init_scil(SMR, BRR);

SMR と BRR の計算方法は、下記の手順です。

$$BRR = \phi \div (A \times \text{設定したいボーレート}) - 1$$

※A=32,128,512,2048 の 4 種類のどれか

※φ = クリスタルの値、RY3048Fone ボードは 24.576MHz

A の値は 4 種類あるので、4 とおりに計算します。φ = 24.576MHz、設定したいボーレートは 9600 です。代入すると、

A=32 のとき	→	$BRR = 24.576 \times 10^6 \div (32 \times 38400) - 1$	= 19.00	…	1
A=128 のとき	→	$BRR = 24.576 \times 10^6 \div (128 \times 38400) - 1$	= 4.00	…	2
A=512 のとき	→	$BRR = 24.576 \times 10^6 \div (512 \times 38400) - 1$	= 0.25	…	3
A=2048 のとき	→	$BRR = 24.576 \times 10^6 \div (2048 \times 38400) - 1$	= -0.6875	…	4

init_scil 関数に値を設定しようとしても 4 種類もあり、どれを設定すればよいか困ってしまいます。選定基準は、

- (1) BRR の値が 0~255 の範囲である設定値を選びます。
- (2) 整数の (小数点のない) 設定値を選びます。
- (3) 1 と 2 に当てはまる設定値が 2 つ以上ある場合は、A の値が小さい設定を選びます。
- (4) 1 と 2 に当てはまる設定値が無い場合、BRR を四捨五入した値が「0~255」の範囲内であることを条件に、下記の計算を行います。

$$\text{実際の通信速度} = \phi \div (A \times (\text{BRR を四捨五入した値} + 1))$$

今回は、

- (1) の条件 → 1, 2, 3 が当てはまる
- (2) の条件 → 1, 2 が当てはまる
- (3) の条件 → 1 と 2 では、1 の方が A の値が小さい

よって、**A=32、BRR=19** を使用します。

A の値を直接 init_scil 関数に設定するわけではありません。下表のような SMR 値となります。

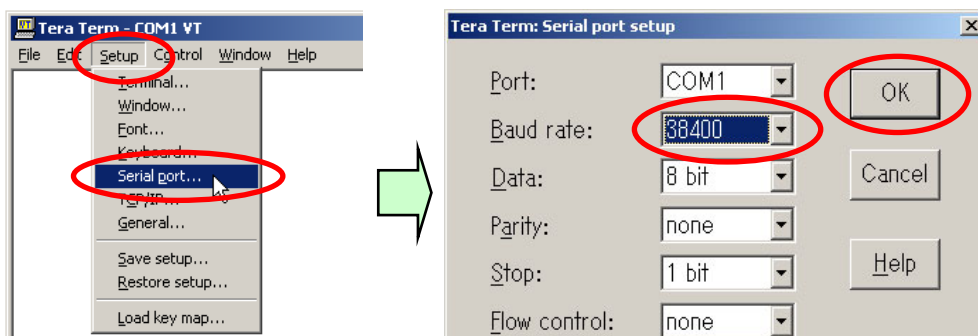
A の値	SMR の設定値
32	0x00
128	0x01
512	0x02
2048	0x03

A=32 なので、SMR は 0x00 となります。

```

59  init_scil( 0x00, 19 );          /* SCI1 初期化          */
      ↑      ↑
      SMR  BRR
    
```

また、TeraTermPro 側も、通信速度を変更する必要があります。



6. プロジェクト「kit07sd_01」 走行データをmicroSDに記録

6.1 概要

マイコンカーの走行中のデータを、microSD に記録します。記録する内容は次のとおりです。

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

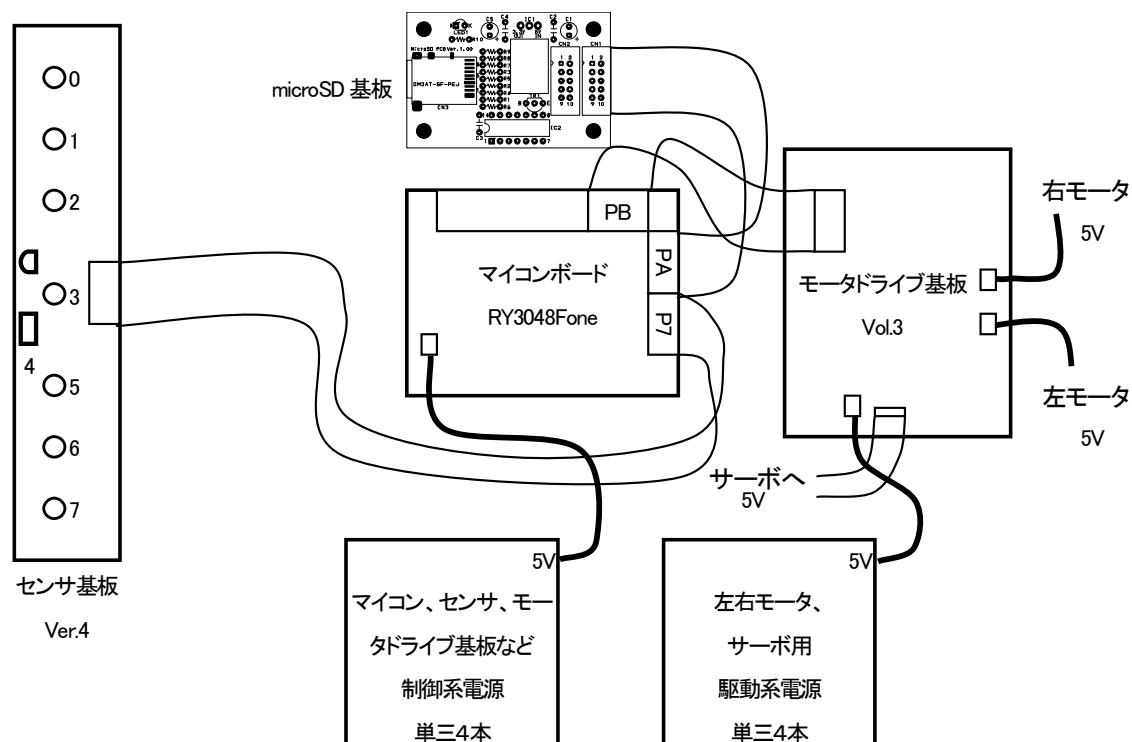
これらのデータを、走行開始から 10ms ごとに 60 秒間記録します。60 秒間経った場合は、データ記録は止めますが走行はそのままおこないます。

走行後、microSD に記録した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。

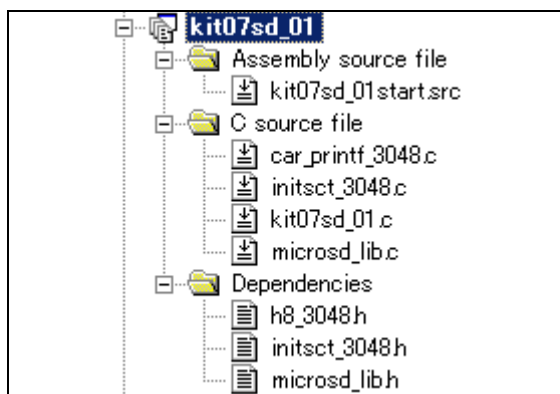
6.2 マイコンカーの構成

マイコンカーキット Ver.4 の構成です。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。

- ・マイコンボードのポート 7 と、マイコンカーキット Ver.4 のセンサ基板 Ver.4 を接続します。
- ・マイコンボードのポート B と、マイコンカーキット Ver.4 のモータドライブ基板 Vol.3 を接続します。
- ・**マイコンボードのポート A と、microSD 基板を接続します。**



6.3 プロジェクトの構成



	ファイル名	内容
1	kit07sd_01start.src	アセンブリ言語で記述されたアセンブリソースファイルです。このファイルの構造は下記のようになっています。 <div style="border: 1px solid black; padding: 2px; display: inline-block;">ベクタアドレス</div> + <div style="border: 1px solid black; padding: 2px; display: inline-block;">スタートアップルーチン</div>
2	car_printf_3048.c	<ul style="list-style-type: none"> 通信するための設定 printf 関数の出力先、scanf 関数の入力元を通信にするための設定を行っています。
3	initsct_3048.c	初期値のないグローバル変数(セクション B 領域)、初期値のあるグローバル変数(セクション R 領域)の初期化用です。
4	microsd_lib.c	microSD 基板を制御するための関数が記載されています。
5	kit07sd_01.c	実際に制御するプログラムが書かれています。H8/3048F-ONE の内蔵周辺機能の初期化も行います。
6	h8_3048.h	H8/3048F-ONE の内蔵周辺機能の I/O レジスタを定義したファイルです。
7	initsct_3048.h	initsct_3048.c のヘッダファイルです。
8	microsd_lib.h	microsd_lib.c のヘッダファイルです。

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「kit07sd_01.c」 */
3 : /*                               2009.06 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kit07.cをベースに次の機能を追加したプログラムです。
7 : ・microSD基板を追加し、走行データの記録、パソコンへの転送します。
8 :
9 :
10 : */
11 :
12 : /*=====*/
13 : /* インクルード */
14 : /*=====*/
15 : #include <no_float.h> /* stdioの簡略化 最初に置く */
16 : #include <stdio.h>
17 : #include <machine.h>
18 : #include "h8_3048.h"
19 : #include "microsd_lib.h" /* microSD制御用 */
20 :
21 : /*=====*/
    
```

microSD を使用するので「microsd_lib.h」をインクルードします。

6. プロジェクト「kit07sd_01」 走行データを microSD に記録

```

22 : /* シンボル定義 */
23 : /*=====*/
24 :
25 : /* 定数設定 */
26 : #define TIMER_CYCLE 3071 /* タイマのサイクル 1ms */
27 : /* φ/8で使用する場合、 */
28 : /* φ/8 = 325.5[ns] */
29 : /* ∴TIMER_CYCLE = */
30 : /* 1[ms] / 325.5[ns] */
31 : /* = 3072 */
32 : #define PWM_CYCLE 49151 /* PWMのサイクル 16ms */
33 : /* ∴PWM_CYCLE = */
34 : /* 16[ms] / 325.5[ns] */
35 : /* = 49152 */
36 : #define SERVO_CENTER 5000 /* サーボのセンタ値 */
37 : #define HANDLE_STEP 26 /* 1°分の値 */
38 :
39 : /* マスク値設定 ×:マスクあり(無効) ○:マスク無し(有効) */
40 : #define MASK2_2 0x66 /* ×○○×○○× */
41 : #define MASK2_0 0x60 /* ×○○××××× */
42 : #define MASK0_2 0x06 /* ××××○○× */
43 : #define MASK3_3 0xe7 /* ○○○××○○○ */
44 : #define MASK0_3 0x07 /* ×××××○○○ */
45 : #define MASK3_0 0xe0 /* ○○○××××× */
46 : #define MASK4_0 0xf0 /* ○○○○×××× */
47 : #define MASK0_4 0x0f /* ××××○○○○ */
48 : #define MASK4_4 0xff /* ○○○○○○○○ */
49 :
50 : /*=====*/
51 : /* プロトタイプ宣言 */
52 : /*=====*/
53 : void init( void );
54 : void timer( unsigned long timer_set );
55 : int check_crossline( void );
56 : int check_rightline( void );
57 : int check_leftline( void );
58 : unsigned char sensor_inp( unsigned char mask );
59 : unsigned char dipsw_get( void );
60 : unsigned char pushsw_get( void );
61 : unsigned char startbar_get( void );
62 : void led_out( unsigned char led );
63 : void speed( int accele_l, int accele_r );
64 : void handle( int angle );
65 : char unsigned bit_change( char unsigned in );
66 : void convertHexToBin( unsigned char hex, char *s );
67 :
68 : /*=====*/
69 : /* グローバル変数の宣言 */
70 : /*=====*/
71 : unsigned long cnt0; /* timer関数用 */
72 : unsigned long cnt1; /* 時間計測用 */
73 : int pattern; /* パターン番号 */
74 :
75 : /* microSD関連変数 */
76 : char msdBuff[ 512 ]; /* 一時記録バッファ */
77 : int msdBuffAddress; /* 一時記録バッファ書込アドレス */
78 : int msdFlag; /* 1:データ記録 0:記録しない */
79 : int msdTimer; /* 取得間隔計算用 */
80 : unsigned long msdStartAddress; /* 記録開始アドレス */
81 : unsigned long msdEndAddress; /* 記録終了アドレス */
82 : unsigned long msdWorkAddress; /* 作業用アドレス */
83 : int msdError; /* エラー番号記録 */
84 :
85 : int handleBuff; /* 現在のハンドル角度記録 */
86 : int leftMotorBuff; /* 現在の左モータPWM値記録 */
87 : int rightMotorBuff; /* 現在の右モータPWM値記録 */
88 :
89 : /*=====*/
90 : /* メインプログラム */
91 : /*=====*/
92 : void main( void )
93 : {
94 :     int i, ret;
95 :     char s[10];
96 :
97 :     /* マイコン機能の初期化 */
98 :     init(); /* 初期化 */
99 :     init_scil( 0x00, 79 ); /* SCI1初期化 */
100 :     set_ccr( 0x00 ); /* 全体割り込み許可 */
101 :
102 :     /* 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 */
103 :     if( pushsw_get() ) {
104 :         pattern = 71;
105 :     }
106 :
107 :     ret = initMicroSD(); /* microSD初期化 */
108 :     if( ret != 0x00 ) {
109 :         /* 初期化できず */
110 :         msdError = 1;
111 :         /* 初期化できなければ3秒間、LEDの点灯方法を変える */
112 :         cnt1 = 0;

```

microSD 関係の変数です。

microSD を初期化します。

```

113 :     while( cnt1 < 3000 ) {
114 :         if( cnt1 % 200 < 100 ) {
115 :             led_out( 0x3 );
116 :         } else {
117 :             led_out( 0x0 );
118 :         }
119 :     }
120 : }
121 :
122 : // microSD 書き込み開始アドレス
123 : // 0x200 (512) の倍数に設定する
124 : msdStartAddress = 0x00000000;
125 :
126 : // microSD 書き込み終了アドレス
127 : // 書き込みしたい時間[ms] : x = 10[ms] : 64バイト
128 : // 60000msなら、x = 60000 * 64 / 10 = 384000 = 0x5dc00
129 : // 結果は0x200の倍数になるように繰り上げる。
130 : msdEndAddress = 0x0005dc00;
131 : msdEndAddress += msdStartAddress; /* スタート分不足 */
132 :
133 : /* マイコンカーの状態初期化 */
134 : handle( 0 );
135 : speed( 0, 0 );
136 :
137 : while( 1 ) {
138 :
139 :     switch( pattern ) {
140 :
141 :     /******
142 :     パターンについて
143 :     0 : スイッチ入力待ち
144 :     1 : スタートバーが開いたかチェック
145 :     11 : 通常トレース
146 :     12 : 右へ大曲げの終わりのチェック
147 :     13 : 左へ大曲げの終わりのチェック
148 :     21 : 1本目のクロスライン検出時の処理
149 :     22 : 2本目を読み飛ばす
150 :     23 : クロスライン後のトレース、クランク検出
151 :     31 : 左クランククリア処理 安定するまで少し待つ
152 :     32 : 左クランククリア処理 曲げ終わりのチェック
153 :     41 : 右クランククリア処理 安定するまで少し待つ
154 :     42 : 右クランククリア処理 曲げ終わりのチェック
155 :     51 : 1本目の右ハーフライン検出時の処理
156 :     52 : 2本目を読み飛ばす
157 :     53 : 右ハーフライン後のトレース
158 :     54 : 右レーンチェンジ終了のチェック
159 :     61 : 1本目の左ハーフライン検出時の処理
160 :     62 : 2本目を読み飛ばす
161 :     63 : 左ハーフライン後のトレース
162 :     64 : 左レーンチェンジ終了のチェック
163 :     *****/
164 :
165 :     case 0:
166 :         /* スイッチ入力待ち */
167 :         if( pushsw_get() ) {
168 :             ret = eraseMicroSD( msdStartAddress, msdEndAddress-1 );
169 :             if( ret != 0x00 ) {
170 :                 /* イレースできず */
171 :                 msdError = 2;
172 :                 break;
173 :             }
174 :             /* microSDProcess開始処理 */
175 :             ret = microSDProcessStart( msdStartAddress );
176 :             if( ret != 0x00 ) {
177 :                 /* 開始処理できず */
178 :                 msdError = 3;
179 :                 break;
180 :             }
181 :             pattern = 1;
182 :             cnt1 = 0;
183 :             break;
184 :         }
185 :         if( cnt1 < 100 ) { /* LED点滅処理 */
186 :             led_out( 0x1 );
187 :         } else if( cnt1 < 200 ) {
188 :             led_out( 0x2 );
189 :         } else {
190 :             cnt1 = 0;
191 :         }
192 :         break;
193 :
194 :     case 1:
195 :         /* スタートバーが開いたかチェック */
196 :         if( !startbar_get() ) {
197 :             /* スタート!!! */
198 :             led_out( 0x0 );
199 :             pattern = 11;
200 :             msdBuffAddress = 0;
201 :             msdWorkAddress = msdStartAddress;
202 :             msdFlag = 1; /* データ記録開始 */
203 :             cnt1 = 0;
    
```

初期化できなければ、LED を全点灯、全消灯を 3 秒間繰り返します。

microSD に書き込む、開始アドレスを設定します。「512 の倍数」で設定します。

microSD に書き込む、終了アドレスを設定します。「512 の倍数」で設定します。

開始アドレスから終了アドレスをイレース(0 でクリア)します。

microSDProcessStart 関数で書き込むアドレスを設定します。

msdFlag を 1 にすると、データ記録を開始します。記録処理は割り込み内で行います。

6. プロジェクト「kit07sd_01」 走行データを microSD に記録

```

204 :         break;
205 :     }
206 :     if( cnt1 < 50 ) {                /* LED点滅処理          */
207 :         led_out( 0x1 );
208 :     } else if( cnt1 < 100 ) {
209 :         led_out( 0x2 );
210 :     } else {
211 :         cnt1 = 0;
212 :     }
213 :     break;
214 :

```

中略

```

549 :     case 71:
550 :         /* 走行データ転送準備 */
551 :         handle( 0 );
552 :         speed( 0, 0 );
553 :         msdFlag = 0;
554 :         if( msdError != 0 ) {
555 :             /* microSDに不具合があったなら終了 */
556 :             printf( "microSD Initialize Error!!\n" );
557 :             pattern = 99;
558 :         } else {
559 :             pattern = 72;
560 :             cnt1 = 0;
561 :         }
562 :         break;
563 :
564 :     case 72:
565 :         /* 最後のデータ書き込みまで待つ*/
566 :         if( checkMicroSDProcess() == 0 ) {
567 :             pattern = 73;          /* データ転送処理へ          */
568 :             break;
569 :         }
570 :         if( checkMicroSDProcess() == 11 ) {
571 :             microSDProcessEnd(); /* microSDProcess終了処理    */
572 :             while( checkMicroSDProcess() ) {
573 :                 pattern = 73;    /* データ転送処理へ          */
574 :             }
575 :             break;
576 :
577 :     case 73:
578 :         /* スイッチが離されたかチェック */
579 :         if( !pushsw_get() ) {
580 :             pattern = 74;
581 :             cnt1 = 0;
582 :         }
583 :         break;
584 :
585 :     case 74:
586 :         /* 0.2s待ち */
587 :         if( cnt1 > 200 ) {
588 :             pattern = 75;
589 :             cnt1 = 0;
590 :             break;
591 :         }
592 :         if( pushsw_get() ) {
593 :             pattern = 73;
594 :         }
595 :         break;
596 :
597 :     case 75:
598 :         /* スイッチが押されたかチェック */
599 :         led_out( (cnt1/500) % 2 + 1 );
600 :         if( pushsw_get() ) {
601 :             pattern = 76;
602 :             cnt1 = 0;
603 :         }
604 :         break;
605 :
606 :     case 76:
607 :         /* タイトル転送、準備 */
608 :         printf( "\n" );
609 :         printf( "Your Car Name Data Out\n" );
610 :         printf( "Pattern, Sensor, ハンドル, 左モータ, 右モータ\n" );
611 :
612 :         msdWorkAddress = msdStartAddress; /* 読み込み開始アドレス */
613 :         pattern = 77;
614 :         break;
615 :
616 :     case 77:
617 :         /* microSDよりデータ読み込み */
618 :         if( msdWorkAddress >= msdEndAddress ) {
619 :             /* 書き込み終了アドレスになったら、終わり */
620 :             pattern = 99;
621 :             break;
622 :         }
623 :         ret = readMicroSD( msdWorkAddress, msdBuff );
624 :         if( ret != 0x00 ) {
625 :             /* 読み込みエラー */

```

microSD への処理を何もしていないなら次へ進みます。

microSDProcess 関数の実行が終わるまで待ちます。

microSDProcessEnd 関数で終了処理を行います。

終了処理が終わるまで待ちます。

転送アドレス > 書き込み終了アドレスになったら、転送終了です。

microSD から、記録したデータを読み込みます。


```

626 :     printf( "¥nmicroSD Read Error!!¥n" );
627 :     pattern = 99;
628 :     break;
629 : } else {
630 :     /* エラーなし */
631 :     msdWorkAddress += 512;
632 :     msdBuffAddress = 0;
633 :     pattern = 78;
634 : }
635 : break;
636 :
637 : case 78:
638 :     /* データ転送 */
639 :     led_out( (cnt1/100) % 2 + 1 ); /* LED点滅処理 */
640 :
641 :     if( msdBuff[msdBuffAddress+0] == 0 ) {
642 :         /* パターンが0なら終了 */
643 :         pattern = 99;
644 :         break;
645 :     }
646 :
647 :     convertHexToBin( msdBuff[msdBuffAddress+1], s );
648 :     printf( "%d,=%¥s¥", %d, %d, %d¥n",
649 :         (char)msdBuff[msdBuffAddress+0], /* パターン */
650 :         s, /* センサ */
651 :         (char)msdBuff[msdBuffAddress+2], /* ハンドル */
652 :         (char)msdBuff[msdBuffAddress+3], /* 左モータ */
653 :         (char)msdBuff[msdBuffAddress+4] /* 右モータ */
654 :     );
655 :
656 :     msdBuffAddress += 64;
657 :
658 :     if( msdBuffAddress >= 512 ) {
659 :         pattern = 77;
660 :     }
661 :     break;
662 :
663 : case 99:
664 :     /* 転送終了 */
665 :     led_out( 0x3 );
666 :     break;
667 :
668 : default:
669 :     /* どれでもない場合は待機状態に戻す */
670 :     pattern = 0;
671 :     break;
672 : }
673 : }
674 : }
675 :
676 : /*****
677 : /* H8/3048F-ONE 内蔵周辺機能 初期化 */
678 : /*****
679 : void init( void )
680 : {
681 :     /* I/Oポートの入出力設定 */
682 :     P1DDR = 0xff;
683 :     P2DDR = 0xff;
684 :     P3DDR = 0xff;
685 :     P4DDR = 0xff;
686 :     P5DDR = 0xff;
687 :     P6DDR = 0xf0; /* CPU基板上のDIP SW */
688 :     P8DDR = 0xff;
689 :     P9DDR = 0xf7; /* 通信ポート */
690 :     PADDR = 0xf7; /* microSD基板 */
691 :     PBDR = 0xc0;
692 :     PBDDR = 0xfe; /* モータドライブ基板Vol.3 */
693 :     /* ※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 */
694 :
695 :     /* ITU0 1msごとの割り込み */
696 :     ITU0_TCR = 0x23;
697 :     ITU0_GRA = TIMER_CYCLE;
698 :     ITU0_IER = 0x01;
699 :
700 :     /* ITU3, 4 リセット同期PWMモード 左右モータ、サーボ用 */
701 :     ITU3_TCR = 0x23;
702 :     ITU_FCR = 0x3e;
703 :     ITU3_GRA = PWM_CYCLE; /* 周期の設定 */
704 :     ITU3_GRB = ITU3_BRB = 0; /* 左モータのPWM設定 */
705 :     ITU4_GRA = ITU4_BRA = 0; /* 右モータのPWM設定 */
706 :     ITU4_GRB = ITU4_BRB = SERVO_CENTER; /* サーボのPWM設定 */
707 :     ITU_TOER = 0x38;
708 :
709 :     /* ITUのカウンタスタート */
710 :     ITU_STR = 0x09;
711 : }
712 :

```

microSD からデータを読み込んだら、パターン 78 へ進みます。

msdBuff 配列からデータを読み込んで、printf 文でパソコンへ出力します。

msdBuff からすべてのデータを読み込むと、パターン 77 へ戻ります。

microSD 基板と接続しているポート A の入出力設定を変更します。

6. プロジェクト「kit07sd_01」 走行データを microSD に記録

```

713 : /*****/
714 : /* ITU0 割り込み処理 */
715 : /*****/
716 : #pragma interrupt( interrupt_timer0 )
717 : void interrupt_timer0( void )
718 : {
719 :     char *p;
720 :
721 :     ITU0_TSR &= 0xfe;          /* フラグクリア */
722 :     cnt0++;
723 :     cnt1++;
724 :
725 :     microSDProcess();          /* microSD 間欠書き込み処理 */
726 :
727 :     /* microSD記録処理 */
728 :     if( msdFlag == 1 && msdError == 0 ) {
729 :         /* 記録間隔のチェック */
730 :         msdTimer++;
731 :         if( msdTimer >= 10 ) {
732 :             msdTimer = 0;
733 :             p = msdBuff + msdBuffAddress;
734 :
735 :             /* バッファに記録 ここから */
736 :             *p++ = pattern;      /* パターン */
737 :             *p++ = sensor_inp(0xff); /* センサ */
738 :             *p++ = handleBuff;   /* ハンドル */
739 :             *p++ = leftMotorBuff; /* 左モータPWM値 */
740 :             *p++ = rightMotorBuff; /* 右モータPWM値 */
741 :             /* バッファに記録 ここまで */
742 :
743 :             msdBuffAddress += 64; /* 記録アドレスを次へ */
744 :
745 :             if( msdBuffAddress >= 512 ) {
746 :                 /* 512個になったら、microSDに記録する */
747 :                 msdBuffAddress = 0;
748 :                 setMicroSDdata( msdBuff );
749 :                 msdWorkAddress += 512;
750 :                 if( msdWorkAddress >= msdEndAddress ) {
751 :                     /* 記録処理終了 */
752 :                     msdFlag = 0;
753 :                 }
754 :             }
755 :         }
756 :     }
757 : }
    
```

microSDProcess 関数は 1ms ごとに実行します。

10ms ごとに msdBuff 配列(RAM)にデータを記録していきます。512 バイト分たまったら、setMicroSDdata 関数で、microSD に書き込む準備をします。実際の書き込み作業は microSDProcess 関数で行います。

microSD に書き込むアドレスが、終了アドレスになったら、記録処理を終了します。

中略

```

897 : /*****/
898 : /* 速度制御 */
899 : /* 引数 左モータ:-100~100 , 右モータ:-100~100 */
900 : /* 0で停止、100で正転100%、-100で逆転100% */
901 : /*****/
902 : void speed( int accele_l, int accele_r )
903 : {
904 :     unsigned char sw_data;
905 :     unsigned long speed_max;
906 :
907 :     sw_data = dipsw_get() + 5; /* ディップスイッチ読み込み */
908 :     speed_max = (unsigned long)(PWM_CYCLE-1) * sw_data / 20;
909 :
910 :     leftMotorBuff = accele_l * sw_data / 20; /* バッファに保存 */
911 :     rightMotorBuff = accele_r * sw_data / 20; /* バッファに保存 */
912 :
913 :     /* 左モータ */
914 :     if( accele_l >= 0 ) {
915 :         PBDR &= 0xfb;
916 :         ITU3_BRB = speed_max * accele_l / 100;
917 :     } else {
918 :         PBDR |= 0x04;
919 :         accele_l = -accele_l;
920 :         ITU3_BRB = speed_max * accele_l / 100;
921 :     }
922 :
923 :     /* 右モータ */
924 :     if( accele_r >= 0 ) {
925 :         PBDR &= 0xf7;
926 :         ITU4_BRA = speed_max * accele_r / 100;
927 :     } else {
928 :         PBDR |= 0x08;
929 :         accele_r = -accele_r;
930 :         ITU4_BRA = speed_max * accele_r / 100;
931 :     }
932 : }
933 :
934 : /*****/
935 : /* サーボハンドル操作 */
936 : /* 引数 サーボ操作角度 : -90~90 */
937 : /* -90で左へ90度、0でまっすぐ、90で右へ90度回転 */
938 : /*****/
939 : void handle( int angle )
    
```

左 PWM 値、右 PWM 値を変数に保存します。データ記録処理では、この値を使います。

```

940 : {
941 :     handleBuff = angle;          /* バッファに保存          */
942 :     ITU4_BRB = SERVO_CENTER - angle * HANDLE_STEP;
943 : }
944 :
945 : /*****/
946 : /* ビット入れ替え          */
947 : /* 引数 入れ替えする値          */
948 : /* 戻り値 入れ替え後の値          */
949 : /*****/
950 : char unsigned bit_change( char unsigned in )
951 : {
952 :     unsigned char ret;
953 :     int i;
954 :
955 :     for( i = 0; i < 8; i++ ) {
956 :         ret >>= 1;          /* 戻り値の右シフト          */
957 :         ret |= in & 0x80;    /* ret bit7 = in bit7          */
958 :         in <<= 1;          /* 引数の左シフト          */
959 :     }
960 :     return ret;
961 : }
962 :
963 : /*****/
964 : /* 16進数→2進数変換 Ver2.00          */
965 : /* 引数 16進数データ、変換後のデータ格納アドレス          */
966 : /* 戻り値 なし          */
967 : /*****/
968 : void convertHexToBin( unsigned char hex, char *s )
969 : {
970 :     int i;
971 :
972 :     for( i=0; i<8; i++ ) {
973 :         if( hex & 0x80 ) {
974 :             *s++ = '1';          /* "1"のときの変換データ          */
975 :         } else {
976 :             *s++ = '0';          /* "0"のときの変換データ          */
977 :         }
978 :         hex <<= 1;
979 :     }
980 :     *s = '\0';
981 : }
982 :
983 : /*****/
984 : /* end of file          */
985 : /*****/
    
```

サーボ操舵角度を変数に保存
 します。データ記録処理では、こ
 の値を使います。

6.5 プログラムの解説

6.5.1 変数

```

68 : /*=====*/
69 : /* グローバル変数の宣言 */
70 : /*=====*/
71 : unsigned long   cnt0;           /* timer関数用 */
72 : unsigned long   cnt1;           /* 時間計測用 */
73 : int             pattern;        /* パターン番号 */
74 :
75 : /* microSD関連変数 */
76 : char            msdBuff[ 512 ]; /* 一時記録バッファ */
77 : int             msdBuffAddress; /* 一時記録バッファ書込アドレス */
78 : int             msdFlag;        /* 1:データ記録 0:記録しない */
79 : int             msdTimer;       /* 取得間隔計算用 */
80 : unsigned long   msdStartAddress; /* 記録開始アドレス */
81 : unsigned long   msdEndAddress;   /* 記録終了アドレス */
82 : unsigned long   msdWorkAddress;  /* 作業用アドレス */
83 : int           msdError;      /* エラー番号記録 */
84 :
85 : int           handleBuff;    /* 現在のハンドル角度記録 */
86 : int           leftMotorBuff; /* 現在の左モータPWM値記録 */
87 : int           rightMotorBuff; /* 現在の右モータPWM値記録 */

```

それぞれの変数は、次のような意味です(プロジェクト「sd_02」で使用した変数以外のみ記述します)。

変数名／配列名	意味	内容
msdError	エラー番号	microSD 処理エラーが合った場合、この変数にエラー番号を代入します。0 はエラーなし、0 以外はエラーがあることを示します。
handleBuff	ハンドル値保存	ハンドルの値を保存します。データ記録時にこの変数の値をハンドル角度の値とします。
leftMotorBuff	左モータ値保存	左モータの値を保存します。データ記録時にこの変数の値を左モータの値とします。
rightMotorBuff	右モータ値保存	右モータの値を保存します。データ記録時にこの変数の値を右モータの値とします。

6.5.2 main関数の開始部分

```

92 : void main( void )
93 : {
94 :     int    i, ret;
95 :     char   s[10];
96 :
97 :     /* マイコン機能の初期化 */
98 :     init();                          /* 初期化                */
99 :     init_scil( 0x00, 79 );            /* SCI1初期化            */
100 :    set_ccr( 0x00 );                  /* 全体割り込み許可     */
101 :
102 :    /* 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 */
103 :    if( pushsw_get() ) {
104 :        pattern = 71;
105 :    }
106 :
107 :    ret = initMicroSD();                /* microSD初期化        */
108 :    if( ret != 0x00 ) {
109 :        /* 初期化できず          */
110 :        msdError = 1;
111 :        /* 初期化できなければ3秒間、LEDの点灯方法を変える */
112 :        cnt1 = 0;
113 :        while( cnt1 < 3000 ) {
114 :            if( cnt1 % 200 < 100 ) {
115 :                led_out( 0x3 );
116 :            } else {
117 :                led_out( 0x0 );
118 :            }
119 :        }
120 :    }
121 :
122 :    // microSD 書き込み開始アドレス
123 :    // 0x200(512)の倍数に設定する
124 :    msdStartAddress = 0x00000000;
125 :
126 :    // microSD 書き込み終了アドレス
127 :    // 書き込みしたい時間[ms] : x = 10[ms] : 64バイト
128 :    // 60000msなら、x = 60000 * 64 / 10 = 384000 = 0x5dc00
129 :    // 結果は0x200の倍数になるように繰り上げする。
130 :    msdEndAddress = 0x0005dc00;
131 :    msdEndAddress += msdStartAddress; /* スタート分足す      */
    
```

103 行でスタートスイッチが押されているかチェックします。起動時にスタートスイッチが押されていたら、データ転送モード(パターン 71)にします。

107 行で microSD を初期化します。初期化できなければ、112～119 行で 3 秒間 LED を全点灯、全消灯を繰り返して、エラーであることを知らせます。

124 行で microSD に書き込む開始アドレスを設定します。

130 行で microSD に書き込む容量を指定します。131 行でスタート分を加えて終了アドレスにしています。

今回、データ記録の条件を次のようにしました。

- ・データ記録の間隔 … 10ms ごと
- ・データ記録数 …………… 64 バイト
- ・データ記録時間 …… 60 秒(60000ms)

microSD に確保しなければいけない容量は、次のようになります。

$$\text{容量} = \text{記録したい時間[ms]} \div \text{記録する間隔[ms]} \times 1 \text{ 回に記録するバイト数}$$

よって、容量は次のとおりです。

$$\begin{aligned} \text{容量} &= \text{記録したい時間[ms]} \div \text{記録する間隔[ms]} \times 1 \text{ 回に記録するバイト数} \\ &= 60,000 \div 10 \times 64 \\ &= 384,000 \end{aligned}$$

値は、512 の倍数にしなければいけません。512 の倍数かどうか確かめます。

$$384,000 \div 512 = 750 \text{ 余り } 0$$

割り切れていますので、この値で OK です。プログラムでは、16 進数に変換した 0x5dc00 を設定します。

6.5.3 パターン 0: スイッチ入力待ち

```

165 :     case 0:
166 :         /* スイッチ入力待ち */
167 :         if( pushsw_get() ) {
168 :             ret = eraseMicroSD( msdStartAddress, msdEndAddress-1 );
169 :             if( ret != 0x00 ) {
170 :                 /* イレースできず */
171 :                 msdError = 2;
172 :                 break;
173 :             }
174 :             /* microSDProcess開始処理 */
175 :             ret = microSDProcessStart( msdStartAddress );
176 :             if( ret != 0x00 ) {
177 :                 /* 開始処理できず */
178 :                 msdError = 3;
179 :                 break;
180 :             }
181 :             pattern = 1;
182 :             cnt1 = 0;
183 :             break;
184 :         }
185 :         if( cnt1 < 100 ) { /* LED点滅処理 */
186 :             led_out( 0x1 );
187 :         } else if( cnt1 < 200 ) {
188 :             led_out( 0x2 );
189 :         } else {
190 :             cnt1 = 0;
191 :         }
192 :         break;

```

パターン 0 は、スタートスイッチ入力待ちで、スイッチを押されたら 168 行目以降を実行します。

168 行で、microSD の記録開始アドレスから終了アドレスまでをイレースします。イレースエラーなら、171 行で msdError 変数に 2 を代入してエラー情報を保存します。

175 行で、microSD の書き込み開始アドレスを設定します。

その後、パターン 1 へ移ります。イレースエラーであっても走行は可能ですので、パターン 1 へ移ります。

6.5.4 パターン 1:スタートバーが開いたかチェック

```

194 :     case 1:
195 :         /* スタートバーが開いたかチェック */
196 :         if( !startbar_get() ) {
197 :             /* スタート!! */
198 :             led_out( 0x0 );
199 :             pattern = 11;
200 :             msdBuffAddress = 0;
201 :             msdWorkAddress = msdStartAddress;
202 :             msdFlag = 1;                /* データ記録開始                */
203 :             cnt1 = 0;
204 :             break;
205 :         }
206 :         if( cnt1 < 50 ) {                /* LED点滅処理                */
207 :             led_out( 0x1 );
208 :         } else if( cnt1 < 100 ) {
209 :             led_out( 0x2 );
210 :         } else {
211 :             cnt1 = 0;
212 :         }
213 :         break;

```

パターン 1 は、スタートバーが開いたかどうかチェックします。スタートバーが開いたなら(スタートバー検出センサの反応が無くなったら)、197 行目以降を実行します。

200 行で、msdBuff 配列 (RAM) を参照する変数を 0 にクリアしています。

201 行で、microSD の作業アドレスを開始アドレスに設定します。今回、msdStartAddress には 0 が入っているので 0 番地から書き込みを開始します。

202 行でフラグを 1 にします。この行以降の 1ms ごとの割り込みから、記録が開始されます。

6.5.5 パターン 71: 走行データ転送準備

```
549 :     case 71:
550 :         /* 走行データ転送準備 */
551 :         handle( 0 );
552 :         speed( 0, 0 );
553 :         msdFlag = 0;
554 :         if( msdError != 0 ) {
555 :             /* microSDに不具合があったなら終了 */
556 :             printf( "microSD Initialize Error!!\n" );
557 :             pattern = 99;
558 :         } else {
559 :             pattern = 72;
560 :             cnt1 = 0;
561 :         }
562 :         break;
```

パターン 71 は、走行データの転送準備を行います。

554 行で、microSD へのアクセスエラーがなかったかチェックします。エラーがあれば読み込みができませんので printf 文でエラーの旨を出力し、パターン 99 へ移り何もしません。

エラーが特になければ、パターン 72 へ移ります。

6.5.6 パターン 72: 最後のデータ書き込むまで待つ

```
564 :     case 72:
565 :         /* 最後のデータ書き込むまで待つ*/
566 :         if( checkMicroSDProcess() == 0 ) {
567 :             pattern = 73;          /* データ転送処理へ          */
568 :             break;
569 :         }
570 :         if( checkMicroSDProcess() == 11 ) {
571 :             microSDProcessEnd(); /* microSDProcess終了処理 */
572 :             while( checkMicroSDProcess() );
573 :             pattern = 73;          /* データ転送処理へ          */
574 :         }
575 :         break;
```

パターン 72 は、microSD への書き込み処理が行われているかチェックしています。処理が終わっていれば、パターン 73 へ移ります。

今回のプログラムは、電源を入れたときにスタートスイッチが押されていれば転送モードになりますので、基本的には書き込み処理が行われていることはありません。ただし、プログラムを改造して走行終了後すぐにデータ転送するときのことを考えて、パターン 72 を入れています。

6.5.7 パターン 73、74:スイッチが離されたかチェック

```
577 :     case 73:
578 :         /* スイッチが離されたかチェック */
579 :         if( !pushsw_get() ) {
580 :             pattern = 74;
581 :             cnt1 = 0;
582 :         }
583 :         break;
584 :
585 :     case 74:
586 :         /* 0.2s待ち */
587 :         if( cnt1 > 200 ) {
588 :             pattern = 75;
589 :             cnt1 = 0;
590 :             break;
591 :         }
592 :         if( pushsw_get() ) {
593 :             pattern = 73;
594 :         }
595 :         break;
```

パターン 73、74 は、スイッチが離されたかチェックします。

まず、パターン 73 でスイッチが離されたかチェックして、離されたならパターン 74 へ移ります。

次に、パターン 74 では、再度スイッチが押されていないか 0.2 秒間チェックして、押されていないければパターン 75 へ移ります。押されたならパターン 73 へ戻って再度チェックします。

6.5.8 パターン 75:スイッチが押されたかチェック

```
597 :     case 75:
598 :         /* スイッチが押されたかチェック */
599 :         led_out( (cnt1/500) % 2 + 1 );
600 :         if( pushsw_get() ) {
601 :             pattern = 76;
602 :             cnt1 = 0;
603 :         }
604 :         break;
```

パターン 75 は、スイッチが押されたかチェックします。押されたなら、パターン 76 へ移ります。

6.5.9 パターン 76:タイトル送信

```

606 :     case 76:
607 :         /* タイトル転送、準備 */
608 :         printf( "\n" );
609 :         printf( "Your Car Name  Data Out\n" );
610 :         printf( "Pattern, Sensor, ハンドル, 左モータ, 右モータ\n" );
611 :
612 :         msdWorkAddress = msdStartAddress; /* 読み込み開始アドレス */
613 :         pattern = 77;
614 :         break;

```

パターン 76 は、パソコンへデータ転送前の文字を送ります。送信後、パターン 77 へ移ります。

6.5.10 パターン 77: microSD よりデータ読み込み

```

616 :     case 77:
617 :         /* microSDよりデータ読み込み */
618 :         if( msdWorkAddress >= msdEndAddress ) {
619 :             /* 書き込み終了アドレスになったら、終わり */
620 :             pattern = 99;
621 :             break;
622 :         }
623 :         ret = readMicroSD( msdWorkAddress , msdBuff );
624 :         if( ret != 0x00 ) {
625 :             /* 読み込みエラー */
626 :             printf( "\nmicroSD Read Error!!\n" );
627 :             pattern = 99;
628 :             break;
629 :         } else {
630 :             /* エラーなし */
631 :             msdWorkAddress += 512; 次にmicroSDから読み込むアドレスをセット
632 :             msdBuffAddress = 0; 今回読み込んだデータを参照する変数をクリア
633 :             pattern = 78;
634 :         }
635 :         break;

```

パターン 77 は、microSD からデータを読み込みます。

618 行は、読み込むアドレス(msdWorkAddress)が書き込み終了アドレス(msdEndAddress)以上になったら、読み込み完了と判断して終了します。

623 行は、microSD からデータを読み込みます。正常にデータを読み込めたら、パターン 78 へ移ります。移る前に、631 行で次に読み込みアドレスを+512しておきます。また 632 行で今回読み込んだデータを参照する変数をクリアしておきます。

6.5.11 パターン 78: データ転送

```
637 :     case 78:
638 :         /* データ転送 */
639 :         led_out( (cnt1/100) % 2 + 1 ); /* LED点滅処理 */
640 :
641 :         if( msdBuff[msdBuffAddress+0] == 0 ) {
642 :             /* パターンが0なら終了 */
643 :             pattern = 99;
644 :             break;
645 :         }
646 :
647 :         convertHexToBin( msdBuff[msdBuffAddress+1], s );
648 :         printf( "%d,=%¥"%s¥", %d, %d, %d¥n",
649 :             (char)msdBuff[msdBuffAddress+0], /* パターン */
650 :             s, /* センサ */
651 :             (char)msdBuff[msdBuffAddress+2], /* ハンドル */
652 :             (char)msdBuff[msdBuffAddress+3], /* 左モータ */
653 :             (char)msdBuff[msdBuffAddress+4] /* 右モータ */
654 :         );
655 :
656 :         msdBuffAddress += 64;
657 :
658 :         if( msdBuffAddress >= 512 ) {
659 :             pattern = 77;
660 :         }
661 :         break;
```

パターン 78 は、パソコンへデータを転送する部分です。

641 行は、パターン番号をチェック、0 ならデータはもうないと判断して転送を終了します。

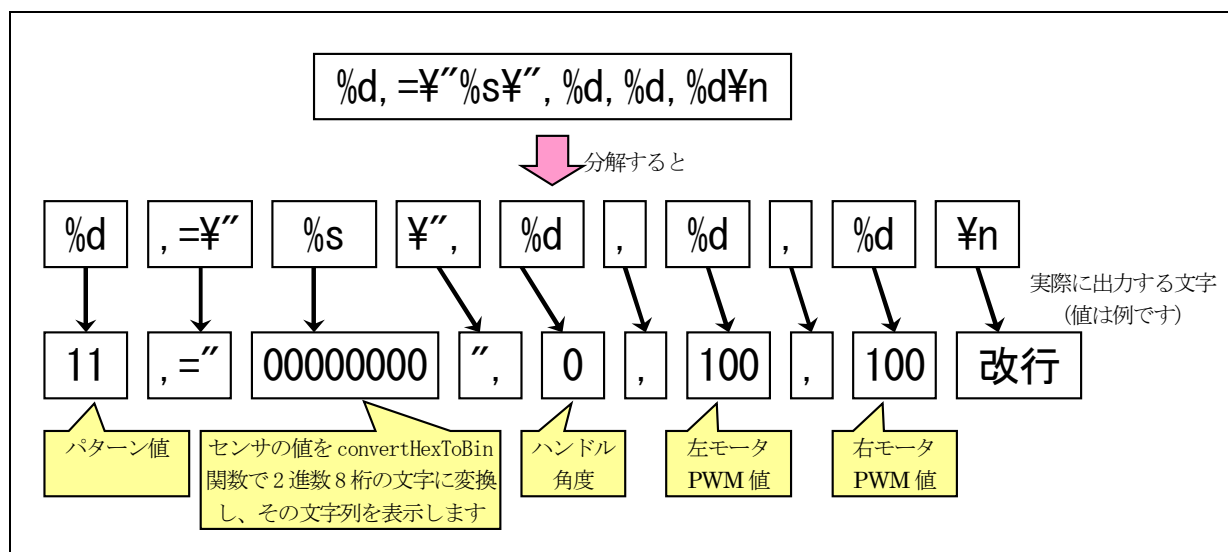
647～654 行でパソコンへデータを転送しています。

656 行で次に転送する msdBuff 配列の位置をセットします。

658 行で、msdBuff 配列の内容をすべて転送したならパターン 77 へ戻って、次のデータを microSD から読み込みます。

6. プロジェクト「kit07sd_01」 走行データを microSD に記録

パソコンへの転送書式は次のようになります。センサの値は、printf 文を実行する前に convertHexToBin 関数で2進数("0"と"1"の文字列)に変換しています。printf 文は、変換した文字列が格納されている s 配列の内容を出力するだけです。



実際の転送データ例を示します。

```

11,="00011000",0,85,85
11,="00011000",0,85,85
11,="00011000",0,85,85
22,="11111111",0,0,0
22,="11111111",0,0,0
22,="00011000",0,0,0
22,="11111000",0,0,0
22,="11111111",0,0,0
22,="11111111",0,0,0
22,="00011000",0,0,0
22,="00011000",0,0,0
22,="00011000",0,0,0
22,="00011000",0,0,0
22,="00011000",0,0,0
23,="00011000",0,34,34
23,="00011000",0,34,34
23,="00011000",0,34,34
    
```

6.5.12 パターン 99: 転送終了

```

663 :     case 99:
664 :         /* 転送終了 */
665 :         led_out( 0x3 );
666 :         break;
    
```

パターン 99 は、処理が終わると実行する部分です。LED を 2 個光らせ、何もしません。

6.5.13 割り込み処理

```

716 : #pragma interrupt( interrupt_timer0 )
717 : void interrupt_timer0( void )
718 : {
719 :     char *p;
720 :
721 :     ITU0_TSR &= 0xfe;          /* フラグクリア          */
722 :     cnt0++;
723 :     cnt1++;
724 :
725 :     microSDProcess();        /* microSD 間欠書き込み処理 */
726 :
727 :     /* microSD記録処理 */
728 :     if( msdFlag == 1 && msdError == 0 ) {
729 :         /* 記録間隔のチェック */
730 :         msdTimer++;
731 :         if( msdTimer >= 10 ) {
732 :             msdTimer = 0;
733 :             p = msdBuff + msdBuffAddress;
734 :
735 :             /* バッファに記録 ここから */
736 :             *p++ = pattern;          /* パターン          */
737 :             *p++ = sensor_inp(0xff); /* センサ          */
738 :             *p++ = handleBuff;      /* ハンドル          */
739 :             *p++ = leftMotorBuff;   /* 左モータPWM値    */
740 :             *p++ = rightMotorBuff;  /* 右モータPWM値    */
741 :             /* バッファに記録 ここまで */
742 :
743 :             msdBuffAddress += 64;    /* 記録アドレスを次へ */
744 :
745 :             if( msdBuffAddress >= 512 ) {
746 :                 /* 512個になったら、microSDに記録する */
747 :                 msdBuffAddress = 0;
748 :                 setMicroSDdata( msdBuff );
749 :                 msdWorkAddress += 512;
750 :                 if( msdWorkAddress >= msdEndAddress ) {
751 :                     /* 記録処理終了 */
752 :                     msdFlag = 0;
753 :                 }
754 :             }
755 :         }
756 :     }
757 : }
    
```

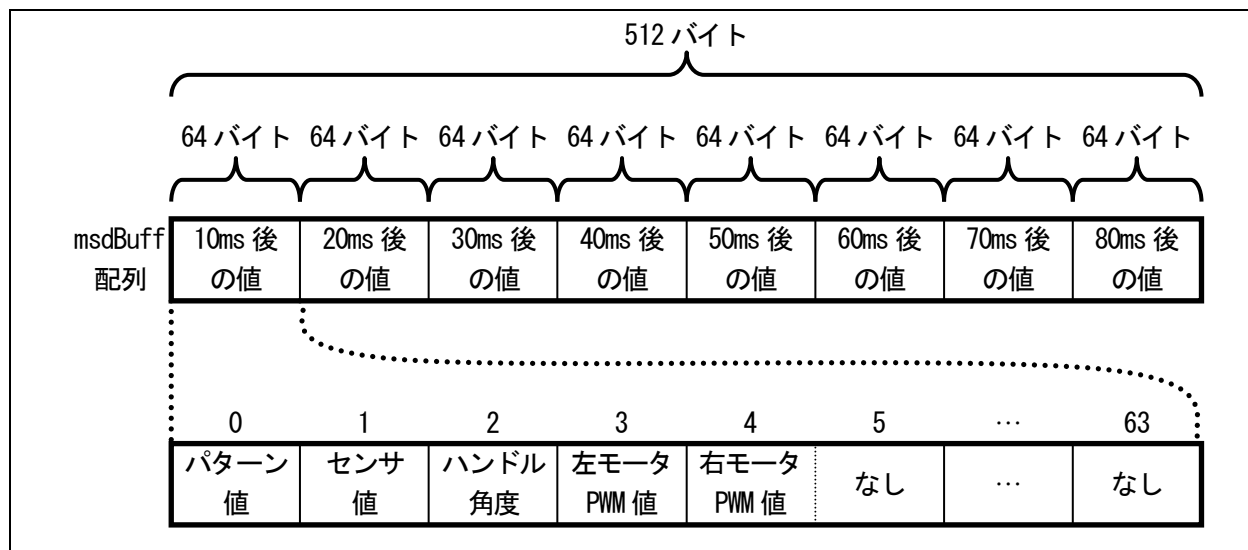
記録する間隔

記録内容

1回に記録する数

731行で記録間隔のチェックをしています。今回は、msdTimerが10以上になったなら、すなわち10msたったなら記録処理を行います。

736～740 行が msdBuff 配列に記録している内容です。記録イメージは次のようです。



748 行で、記録データが 512 バイトになったら setMicroSDdata 関数で microSD へ書き込み準備処理を行います。

750 行で、記録終了アドレスになったかチェックして、なったならば msdFlag を 0 にして、記録処理を終了します。

6.5.14 記録データをバッファに保存

```

902 : void speed( int accele_l, int accele_r )
903 : {
904 :     unsigned char    sw_data;
905 :     unsigned long    speed_max;
906 :
907 :     sw_data = dipsw_get() + 5;          /* ディップスイッチ読み込み */
908 :     speed_max = (unsigned long)(PWM_CYCLE-1) * sw_data / 20;
909 :
910 :     leftMotorBuff = accele_l * sw_data / 20; /* バッファに保存 */
911 :     rightMotorBuff = accele_r * sw_data / 20; /* バッファに保存 */

```

中略

```

939 : void handle( int angle )
940 : {
941 :     handleBuff = angle;                /* バッファに保存 */
942 :     ITU4_BRB = SERVO_CENTER - angle * HANDLE_STEP;
943 : }

```

以下、略

speed 関数で設定した PWM 値を、leftMotorBuff 変数、rightMotorBuff 変数に保存します。データ記録処理では、この値を現在の PWM 値として記録します。handleBuff 変数も同様です。

6.6 プログラムの調整

6.6.1 自分のマイコンカーに合わせて調整

「kit07sd_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。他にも、調整する部分は調整してください。

行	現在のプログラム	変更内容
36	#define SERVO_CENTER 5000	自分のマイコンカーのサーボセンタ値に変更します。
351	handle(-38);	自分のマイコンカーの左最大切れ角の値に変更します。
360	handle(38);	自分のマイコンカーの右最大切れ角の値に変更します。

調整ができれば、プロジェクト「kit07sd_01」をビルドして、「kit07sd_01.mot」ファイルをマイコンボードに書き込みます。

6.6.2 記録間隔の変更

記録間隔を変更するときの変更する行と数値は下表のようです。

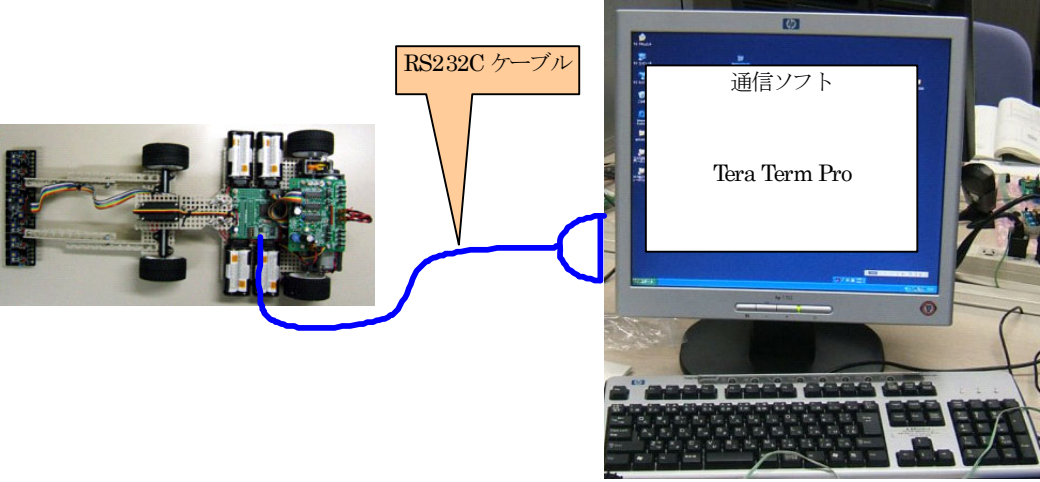
記録間隔	656 行の変更	658 行の変更	731 行の変更	743 行の変更	745 行の変更
80ms	512	512	80	512	512
40ms	256	512	40	256	512
20ms	128	512	20	128	512
10ms(変更前)	64	512	10	64	512
5ms	32	512	5	32	512
2ms	12	480	2	12	480
1ms	6	480	1	6	480

6.7 走行からデータ転送までの流れ

マイコンカーは普通に走らせます。走行データが記録できるのは、スタートしてから 60 秒です。

走らせた後、電源を切ります。microSD はフラッシュ ROM なので、電源を切ってもデータは消えません。

1



RS232C ケーブル

通信ソフト
Tera Term Pro

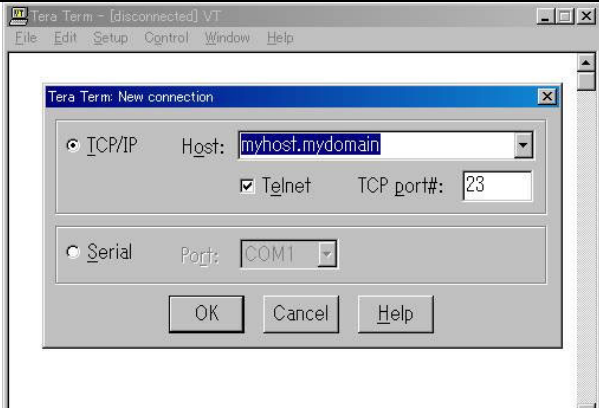
RS232C ケーブルをマイコンカーに接続します。

2

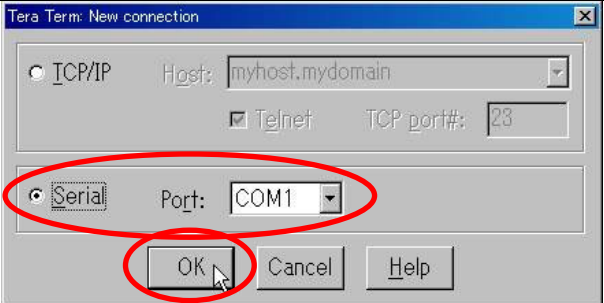


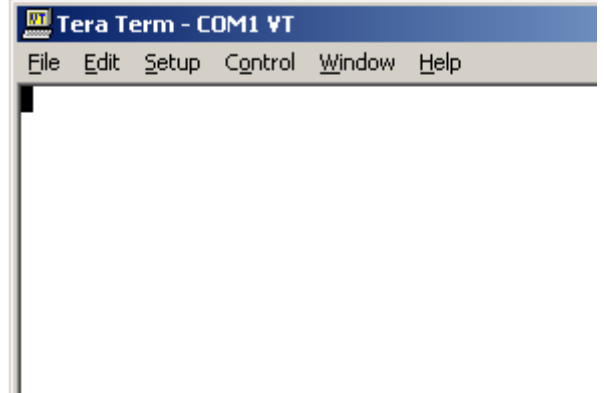
マイコンカーの電源は、まだ入れません。
Tera Term Pro を立ち上げます。

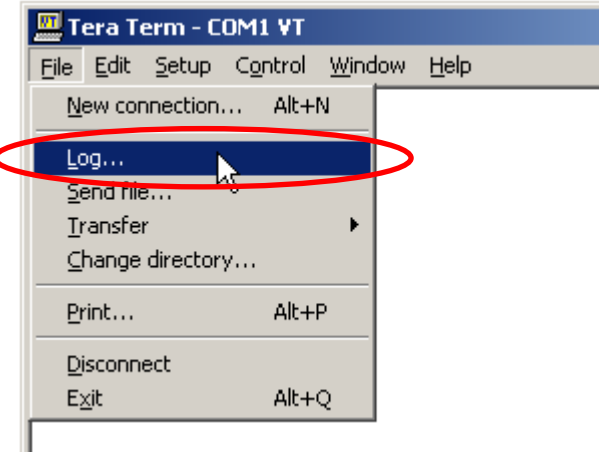
3

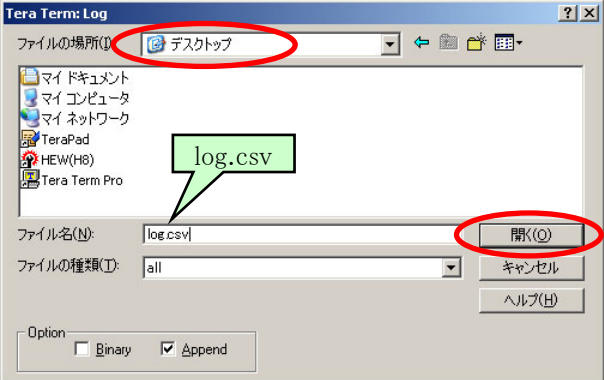


最初に接続先を確認する画面が表示されます。

4		<p>「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。</p>
---	---	---

5		<p>立ち上がりました。</p>
---	---	------------------

6		<p>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。</p>
---	--	--

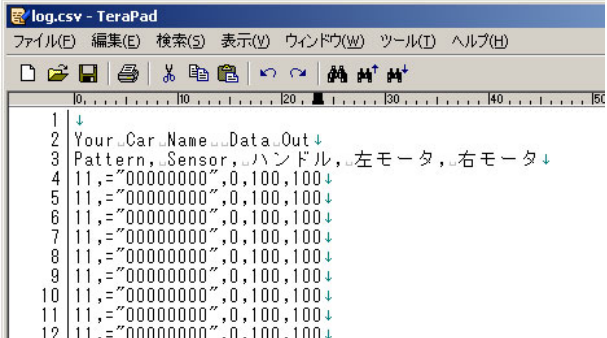
7		<p>保存ファイル名を入力します。ここでは「log.csv」と入力します。保存するフォルダも分かりやすい位置に変更しておきましょう。今回は、「デスクトップ」にしています。ファイル名を設定できたら、開くをクリックします。これでパソコン側の準備は完了です。いよいよマイコン側の操作をします。 ※拡張子は必ず「csv」にします。</p>
---	---	--

6. プロジェクト「kit07sd_01」 走行データを microSD に記録

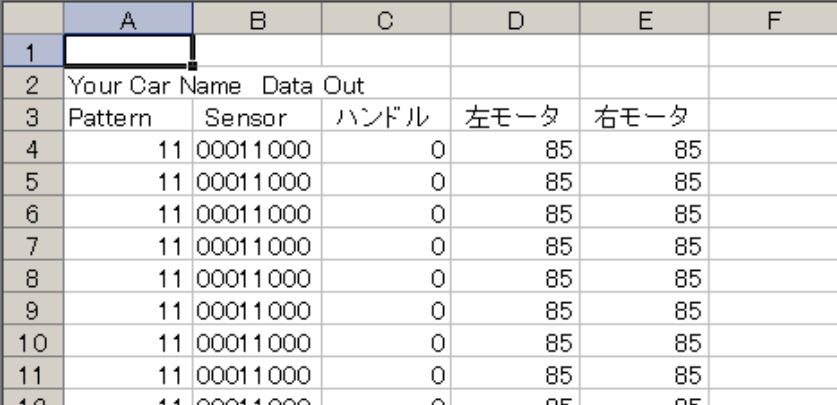
8		<p>マイコンカーは、モータドライブ基板のスタートスイッチを押しながら、マイコンボードの電源を ON にします。</p>
---	--	--

9		<p>モータドライブ基板の LED 2個が、いつもよりゆっくりと交互に点滅します。これがデータ転送モードです。 もし LED が 3 秒程度、両方 ON、両方 OFF を繰り返し、最後に両方点いた場合は、microSD 基板と接続できていないので確認してください。</p>
---	--	--

10		<p>マイコンカーのスタートスイッチを再度押しします。 Tera Term Pro の画面に文字が一気に表示されると思います。 画面が止まって、マイコンカーの LED が 2 つとも点いたら転送終了です。マイコンカーの電源を切ってください。 Tera Term Pro は終了します。</p>
----	--	--

11		エディタで「log.csv」を開きました。マイコンカーから転送された走行データが、パソコンに保存されました。
----	---	--

6.8 エクセルへの取り込み方

1		エクセルなどの表計算ソフトがインストールされている場合、csv ファイルをダブルクリックするとソフトが立ち上がります。ソフトをインストールしているにも関わらず、立ち上がらない場合はソフトを立ち上げてから読み込んでください。
---	---	---

	パターン	センサ	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	

1 行あたり、10ms の時間になります。1 本目のクロスラインを検出してから 2 本目のクロスラインを検出し終わるまで 6 行あります。よって、60ms かかって 2 本のクロスラインを通過したことになります。

6. プロジェクト「kit07sd_01」 走行データを microSD に記録

	パターン	センサ	ハンドル	左モータ	右モータ	
3	233	23 00011000	0	34	34	
	234	23 00011000	0	34	34	
	235	23 00011000	0	34	34	
	236	31 11111000	-38	8	42	← 左クランク発見!!
	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	

	パターン	センサ	ハンドル	左モータ	右モータ	
4	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	

7. プロジェクト「kit07sd_02」 エンコーダプログラムの追加

7.1 概要

本プログラムは、次のプログラムを合わせた内容です。

- ・kit07sd_01.c…microSD 基板をマイコンカーに追加して走行
- ・kit07enc_03.c…エンコーダによる速度制御

※kit07enc_03.c は、ワークスペース「kit07enc」のプロジェクト「kit07enc_03」のファイルです。詳しくは、ロータリエンコーダ実習マニュアルを参照してください。

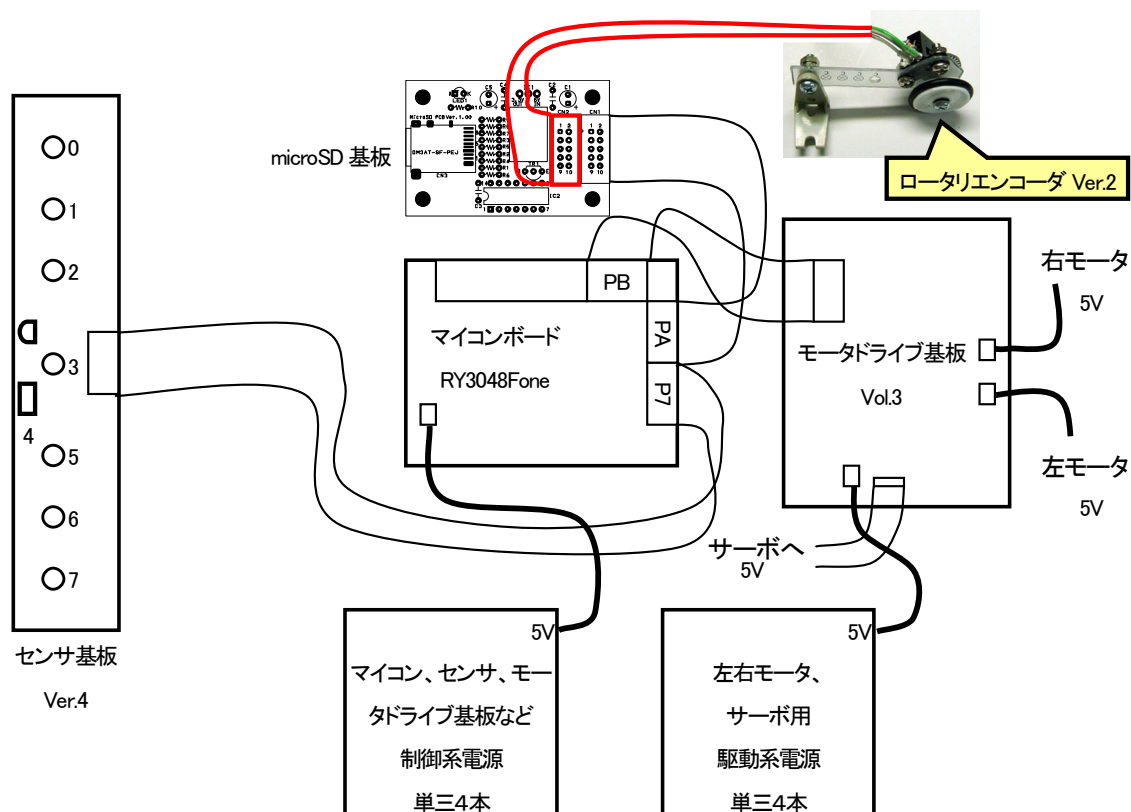
マイコンカーの走行データを、microSD に記録しますが、今回はエンコーダ値も記録します。そのため、走行スピードが分かります。

7.2 マイコンカーの構成

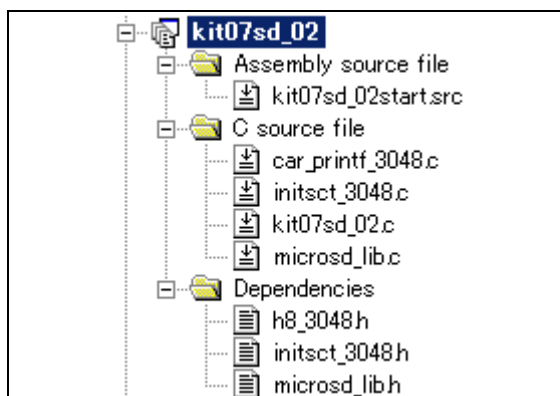
マイコンカーキット Ver.4 の構成です。LM350 追加セットで電池 8 本を直列に繋いでいる構成でも OK です。

- ・マイコンボードのポート 7 と、マイコンカーキット Ver.4 のセンサ基板 Ver.4 を接続します。
- ・マイコンボードのポート B と、マイコンカーキット Ver.4 のモータドライブ基板 Vol.3 を接続します。
- ・**マイコンボードのポート A と、microSD 基板を接続します。**
- ・**microSD 基板と、ロータリエンコーダを接続します。**

※microSD 基板の 2 つの 10 ピンコネクタは、並列に接続されています。そのため、ロータリエンコーダはポート A に接続されていることとなります。



7.3 プロジェクトの構成



	ファイル名	内容
1	kit07sd_02start.src	アセンブリ言語で記述されたアセンブリソースファイルです。このファイルの構造は下記のようになっています。 <div style="border: 1px solid black; display: inline-block; padding: 2px;">ベクタアドレス</div> + <div style="border: 1px solid black; display: inline-block; padding: 2px;">スタートアップルーチン</div>
2	car_printf_3048.c	<ul style="list-style-type: none"> •通信するための設定 •printf 関数の出力先、scanf 関数の入力元を通信にするための設定を行っています。
3	initsct_3048.c	初期値のないグローバル変数(セクション B 領域)、初期値のあるグローバル変数(セクション R 領域)の初期化用です。
4	microsd_lib.c	microSD 基板を制御するための関数が記載されています。
5	kit07sd_02.c	実際に制御するプログラムが書かれています。H8/3048F-ONE の内蔵周辺機能の初期化も行います。
6	h8_3048.h	H8/3048F-ONE の内蔵周辺機能の I/O レジスタを定義したファイルです。
7	initsct_3048.h	initsct_3048.c のヘッダファイルです。
8	microsd_lib.h	microsd_lib.c のヘッダファイルです。

7.4 プログラムの解説

7.4.1 入出力設定

```

711 : void init( void )
712 : {
713 :     /* I/Oポートの入出力設定 */
714 :     P1DDR = 0xff;
715 :     P2DDR = 0xff;
716 :     P3DDR = 0xff;
717 :     P4DDR = 0xff;
718 :     P5DDR = 0xff;
719 :     P6DDR = 0xf0;          /* CPU基板上的DIP SW          */
720 :     P8DDR = 0xff;
721 :     P9DDR = 0xf7;          /* 通信ポート                  */
722 :     PADDR = 0xf6;        /* microSD基板、エンコーダ(bit0)*/
723 :     PBDR  = 0xc0;
724 :     PBDDR = 0xfe;          /* モータドライブ基板Vol.3     */
725 :     /* ※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
以下、略

```

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、PA0 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。

マイコンボード J2 のピン番号	信号名	方向	接続機器	PADDR の 設定
1	+5V	電源		
2	PA7	→	microSD CS	1
3	PA6	→	microSD DIN	1
4	PA5	→	microSD CLK	1
5	PA4		未接続	1
6	PA3	←	microSD DOUT	0
7	PA2		未接続	1
8	PA1		未接続	1
9	PA0		ロータリエンコーダ	0
10	GND	電源		

ポート A の入出力方向を決める PADDR レジスタの設定は、2 進数で「1111 0110」、16 進数で「0xf6」となります。722 行目で PADDR に 0xf6 を設定しています。

7.4.2 割り込みプログラム

```

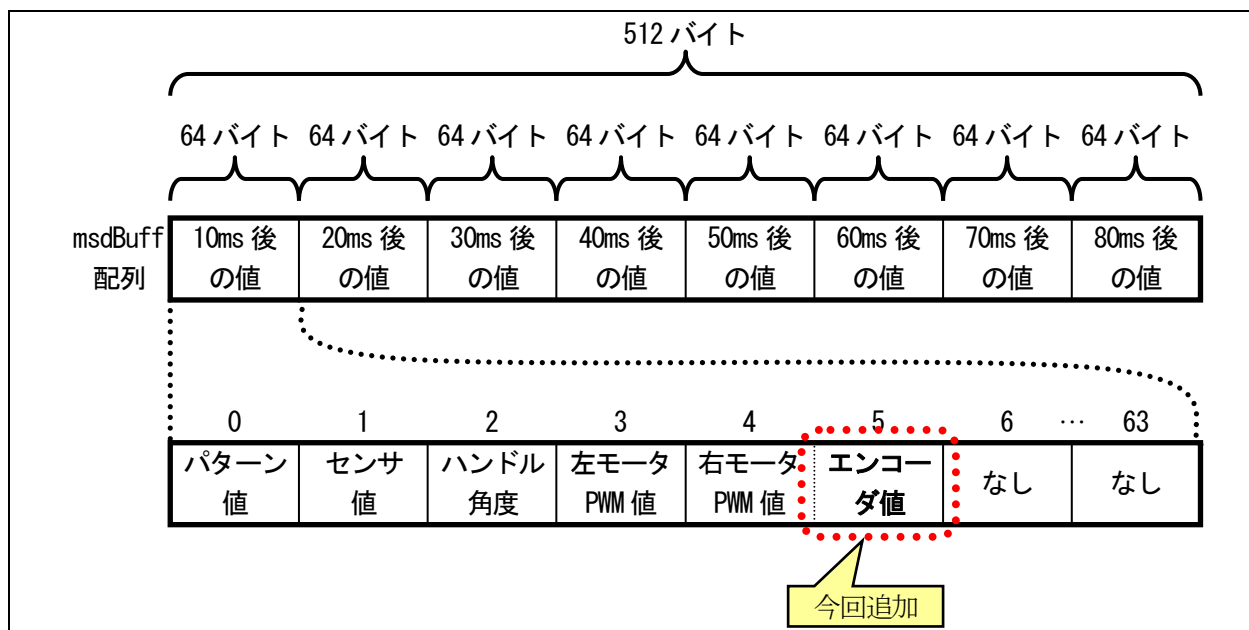
751 : #pragma interrupt( interrupt_timer0 )
752 : void interrupt_timer0( void )
753 : {
754 :     unsigned int i;
755 :     char *p;
756 :
757 :     ITU0_TSR &= 0xfe;          /* フラグクリア          */
758 :     cnt0++;
759 :     cnt1++;
760 :
761 :     microSDProcess();        /* microSD 間欠書き込み処理 */
762 :
763 :     iTimer10++;
764 :     if( iTimer10 >= 10 ) {
765 :         /* 10ms ごとの処理 */
766 :         iTimer10 = 0;
767 :         i = ITU2_CNT;
768 :         iEncoder      = i - uEncoderBuff;
769 :         lEncoderTotal += iEncoder;
770 :         if( iEncoder > iEncoderMax )
771 :             iEncoderMax = iEncoder;
772 :         uEncoderBuff = i;
773 :     }
774 :
775 :     /* microSD 記録処理 */
776 :     if( msdFlag == 1 && msdError == 0 ) {
777 :         /* 記録間隔のチェック */
778 :         msdTimer++;
779 :         if( msdTimer >= 10 ) {
780 :             msdTimer = 0;
781 :             p = msdBuff + msdBuffAddress;
782 :
783 :             /* バッファに記録 ここから */
784 :             *p++ = pattern;          /* パターン          */
785 :             *p++ = sensor_inp(0xff); /* センサ            */
786 :             *p++ = handleBuff;       /* ハンドル          */
787 :             *p++ = leftMotorBuff;    /* 左モータ PWM 値   */
788 :             *p++ = rightMotorBuff;   /* 右モータ PWM 値   */
789 :             *p++ = iEncoder;         /* エンコーダ        */
790 :             /* バッファに記録 ここまで */
791 :
792 :             /* バッファに記録 ここまで */
793 :
794 :     }

```

ロータリエンコーダ処理です。割り込みは 1ms ごとですが、ロータリエンコーダ処理は 10ms ごとなので、iTimer10 をカウント、10 回目なら、ロータリエンコーダ処理を実行します。

763～773 行で、エンコーダ処理を追加しています。

784～789 行が msdBuff 配列に記録している内容です。エンコーダ値を追加しました。記録イメージは次のよう
 です。



7.4.3 送信内容

```

668 :     case 78:
669 :         /* データ転送 */
670 :         led_out( (cnt1/100) % 2 + 1 ); /* LED 点滅処理 */
671 :
672 :         if( msdBuff[msdBuffAddress+0] == 0 ) {
673 :             /* パターンが 0 なら終了 */
674 :             pattern = 99;
675 :             break;
676 :         }
677 :
678 :         convertHexToBin( msdBuff[msdBuffAddress+1], s );
679 :         printf( "%d,=%#s%", %d, %d, %d, %d\n",
680 :             (char)msdBuff[msdBuffAddress+0], /* パターン */
681 :             s, /* センサ */
682 :             (char)msdBuff[msdBuffAddress+2], /* ハンドル */
683 :             (char)msdBuff[msdBuffAddress+3], /* 左モータ */
684 :             (char)msdBuff[msdBuffAddress+4], /* 右モータ */
685 :             (char)msdBuff[msdBuffAddress+5] /* エンコーダ */
686 :         );
687 :
688 :         msdBuffAddress += 64;
689 :
690 :         if( msdBuffAddress >= 512 ) {
691 :             pattern = 77;
692 :         }
693 :         break;

```

679～686 行がパソコンへデータを転送する内容です。「kit07md_01.c」と比べ、エンコーダ値の出力を追加しました。データ形式は、

パターン, センサ, ハンドル, 左モータ, 右モータ, エンコーダ

です。

出力例を下記に示します。

```

11,="00011000",0,100,100,16
11,="00011000",0,100,100,16
11,="00011000",0,100,100,15
22,="11111111",0,0,0,16
22,="11111111",0,0,0,16
22,="00011000",0,0,0,15
22,="11111110",0,0,0,16
22,="11111111",0,0,0,14
22,="00011100",0,0,0,15
22,="00011000",0,0,0,14
23,="00011000",0,0,0,14
23,="00011000",0,0,0,13
23,="00011000",0,0,0,13

```

7.5 ロータリエンコーダに関わる計算

プログラムを変更するに当たって、ロータリエンコーダに関わる値も変更する必要があります。プログラムの変更前に、下記内容について計算しておきましょう。

ロータリエンコーダのタイヤの半径	mm (A)
1回転のパルス数(ロータリエンコーダ Ver.2 は 72) ※標準は立ち上がり、立ち下がりでカウントする設定です。	パルス (B)
円周 = $2\pi \times (A)$	mm (C)
1000mm 進んだときのパルス数は、 $1000 : x = (C) : (B) \therefore x = 1000 \times (B) \div (C)$	パルス (D)
100mm 進んだときのパルス数は、 $(E) = (D) \times 0.1$	パルス (E) ※四捨五入した整数
1m/s で進んだとき、10ms 間のパルス数は、 $(F) = (D) \times 0.01$	パルス (F) ※四捨五入した整数
2m/s で進んだとき、10ms 間のパルス数は、 $(G) = (D) \times 0.02$	パルス (G) ※四捨五入した整数

7.6 プログラムの調整

「kit07sd_02.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。他にも、調整する部分は調整してください。


行	現在のプログラム	変更内容
36	#define SERVO_CENTER 5000	自分のマイコンカーのサーボセンタ値に変更します。
317	if(iEncoder >= 11) {	右へ大曲げ時、このスピードより速ければ、ブレーキをかけます。秒速 1m/s に設定するなら、(F)の値に変更します。
341	if(iEncoder >= 11) {	左へ大曲げ時、このスピードより速ければ、ブレーキをかけます。秒速 1m/s に設定するなら、(F)の値に変更します。
363	if(lEncoderTotal - lEncoderLine >= 109) {	クロスラインを見つけた瞬間から、センサを読まない距離です。10cm の値に設定するなら、(E)の値に変更します。
374	handle(-38);	自分のマイコンカーの左最大切れ角の値に変更します。
383	handle(38);	自分のマイコンカーの右最大切れ角の値に変更します。
389	if(iEncoder >= 11) {	クロスラインを見つけた後、クランクを検出するまでのスピードです。秒速 1m/s に設定するなら、(F)の値に変更します。

462	<code>if(lEncoderTotal-lEncoderLine >= 109) {</code>	右ハーフラインを見つけた瞬間から、センサを読まない距離です。10cm に設定するなら、 (E) の値に変更します。
477	<code>if(iEncoder >= 11) {</code>	右ハーフラインを見つけた後、レンジチェンジ開始までのスピードです。秒速2m/s に設定するなら、 (G) の値に変更します。
527	<code>if(lEncoderTotal-lEncoderLine >= 109) {</code>	左ハーフラインを見つけた瞬間から、センサを読まない距離です。10cm に設定するなら、 (E) の値に変更します。
542	<code>if(iEncoder >= 11) {</code>	左ハーフラインを見つけた後、レンジチェンジ開始までのスピードです。秒速2m/s に設定するなら、 (G) の値に変更します。
733	<code>ITU2_TCR = 0x14;</code>	エンコーダのパルスカウントを立ち上がりのみにする場合、「0x04」に設定します。ちなみに「0x14」は、立ち上がりと立ち下りの両方でカウントする設定です。

修正ができれば、プロジェクト「kit07sd_02」をビルドして、「kit07sd_02.mot」ファイルをマイコンボードに書き込みます。

7.7 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。アプリケーションは、エクセルを使います。

1		<p>エクセルでデータを取り込みます。csv ファイルをダブルクリックするとエクセルで立ち上がります。</p>
---	---	---

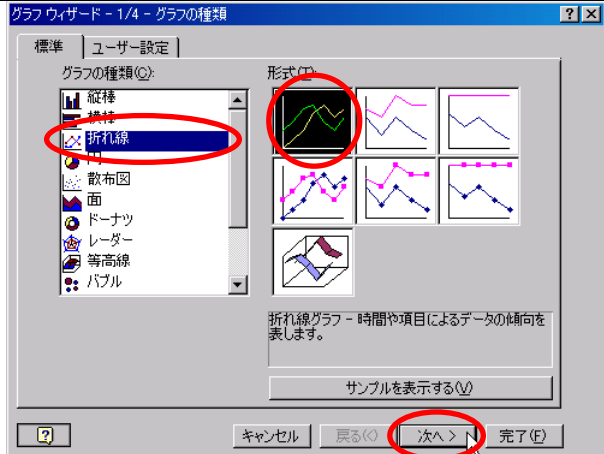
2	<table border="1"> <thead> <tr> <th></th> <th>A</th> <th>B</th> <th>C</th> <th>D</th> <th>E</th> <th>F</th> </tr> </thead> <tbody> <tr><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>2</td><td>Your Car Name</td><td>Data Out</td><td></td><td></td><td></td><td></td></tr> <tr><td>3</td><td>Pattern</td><td>Sensor</td><td>ハンドル</td><td>左モータ</td><td>右モータ</td><td>エンコーダ</td></tr> <tr><td>4</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>0</td></tr> <tr><td>5</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>0</td></tr> <tr><td>6</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>0</td></tr> <tr><td>7</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>0</td></tr> <tr><td>8</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>1</td></tr> <tr><td>9</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>1</td></tr> <tr><td>10</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>1</td></tr> <tr><td>11</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>1</td></tr> <tr><td>12</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>2</td></tr> <tr><td>13</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>1</td></tr> <tr><td>14</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>3</td></tr> <tr><td>15</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>2</td></tr> <tr><td>16</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>2</td></tr> <tr><td>17</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>3</td></tr> <tr><td>18</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>3</td></tr> </tbody> </table>		A	B	C	D	E	F	1							2	Your Car Name	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	<p>エクセルに取り込みました。</p>
	A	B	C	D	E	F																																																																																																																																	
1																																																																																																																																							
2	Your Car Name	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																																																																																																																																	

3	<table border="1"> <thead> <tr> <th></th> <th>A</th> <th>B</th> <th>C</th> <th>D</th> <th>E</th> <th>F</th> </tr> </thead> <tbody> <tr><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>2</td><td>Your Car Name</td><td>Data Out</td><td></td><td></td><td></td><td></td></tr> <tr><td>3</td><td>Pattern</td><td>Sensor</td><td>ハンドル</td><td>左モータ</td><td>右モータ</td><td>エンコーダ</td></tr> <tr><td>4</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>0</td></tr> <tr><td>5</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>0</td></tr> <tr><td>6</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>0</td></tr> <tr><td>7</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>0</td></tr> <tr><td>8</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>1</td></tr> <tr><td>9</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>1</td></tr> <tr><td>10</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>1</td></tr> <tr><td>11</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>1</td></tr> <tr><td>12</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>2</td></tr> <tr><td>13</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>1</td></tr> <tr><td>14</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>3</td></tr> <tr><td>15</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>2</td></tr> <tr><td>16</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>2</td></tr> <tr><td>17</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>3</td></tr> <tr><td>18</td><td>11</td><td>00011000</td><td>0</td><td>100</td><td>100</td><td>3</td></tr> </tbody> </table>		A	B	C	D	E	F	1							2	Your Car Name	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	<p>エンコーダ値のセルを選択します。</p>
	A	B	C	D	E	F																																																																																																																																	
1																																																																																																																																							
2	Your Car Name	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																																																																																																																																	

4	 <table border="1"> <thead> <tr> <th></th> <th>D</th> <th>E</th> <th>F</th> <th>G</th> <th>H</th> <th>I</th> </tr> </thead> <tbody> <tr><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>3</td><td>左モータ</td><td>右モータ</td><td>エンコーダ</td><td></td><td></td><td></td></tr> <tr><td>4</td><td>0</td><td>100</td><td>100</td><td></td><td></td><td></td></tr> <tr><td>5</td><td>0</td><td>100</td><td>100</td><td></td><td></td><td></td></tr> <tr><td>6</td><td>0</td><td>100</td><td>100</td><td></td><td></td><td></td></tr> <tr><td>7</td><td>0</td><td>100</td><td>100</td><td></td><td></td><td></td></tr> <tr><td>8</td><td>0</td><td>100</td><td>100</td><td></td><td></td><td></td></tr> <tr><td>9</td><td>0</td><td>100</td><td>100</td><td></td><td></td><td></td></tr> <tr><td>10</td><td>0</td><td>100</td><td>100</td><td></td><td></td><td></td></tr> <tr><td>11</td><td>0</td><td>100</td><td>100</td><td></td><td></td><td></td></tr> <tr><td>12</td><td>0</td><td>100</td><td>100</td><td></td><td></td><td></td></tr> <tr><td>13</td><td>0</td><td>100</td><td>100</td><td></td><td></td><td></td></tr> <tr><td>14</td><td>0</td><td>100</td><td>100</td><td></td><td></td><td></td></tr> <tr><td>15</td><td>0</td><td>100</td><td>100</td><td></td><td></td><td></td></tr> <tr><td>16</td><td>0</td><td>100</td><td>100</td><td></td><td></td><td></td></tr> <tr><td>17</td><td>0</td><td>100</td><td>100</td><td></td><td></td><td></td></tr> <tr><td>18</td><td>0</td><td>100</td><td>100</td><td></td><td></td><td></td></tr> </tbody> </table>		D	E	F	G	H	I	1							2							3	左モータ	右モータ	エンコーダ				4	0	100	100				5	0	100	100				6	0	100	100				7	0	100	100				8	0	100	100				9	0	100	100				10	0	100	100				11	0	100	100				12	0	100	100				13	0	100	100				14	0	100	100				15	0	100	100				16	0	100	100				17	0	100	100				18	0	100	100				<p>グラフ ウィザードを選択します。</p>
	D	E	F	G	H	I																																																																																																																																	
1																																																																																																																																							
2																																																																																																																																							
3	左モータ	右モータ	エンコーダ																																																																																																																																				
4	0	100	100																																																																																																																																				
5	0	100	100																																																																																																																																				
6	0	100	100																																																																																																																																				
7	0	100	100																																																																																																																																				
8	0	100	100																																																																																																																																				
9	0	100	100																																																																																																																																				
10	0	100	100																																																																																																																																				
11	0	100	100																																																																																																																																				
12	0	100	100																																																																																																																																				
13	0	100	100																																																																																																																																				
14	0	100	100																																																																																																																																				
15	0	100	100																																																																																																																																				
16	0	100	100																																																																																																																																				
17	0	100	100																																																																																																																																				
18	0	100	100																																																																																																																																				

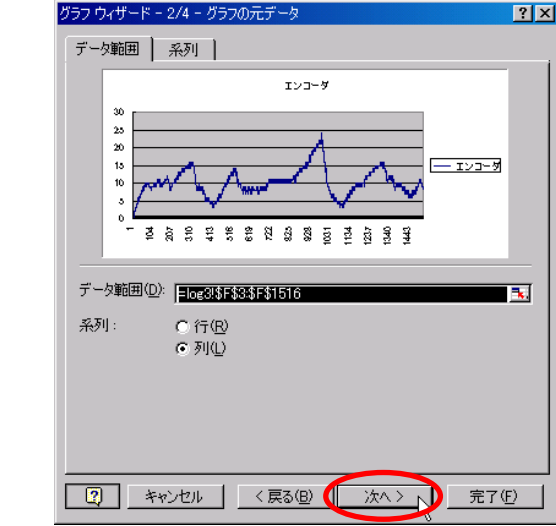
7. プロジェクト「kit07sd_02」 エンコーダプログラムの追加

5



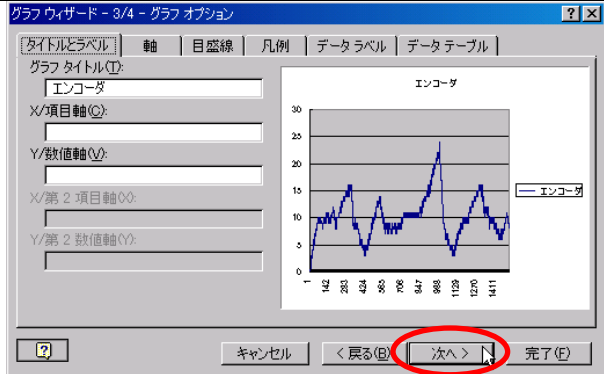
折れ線グラフを選択します。**次へ**をクリックします。

6




次へをクリックします。

7

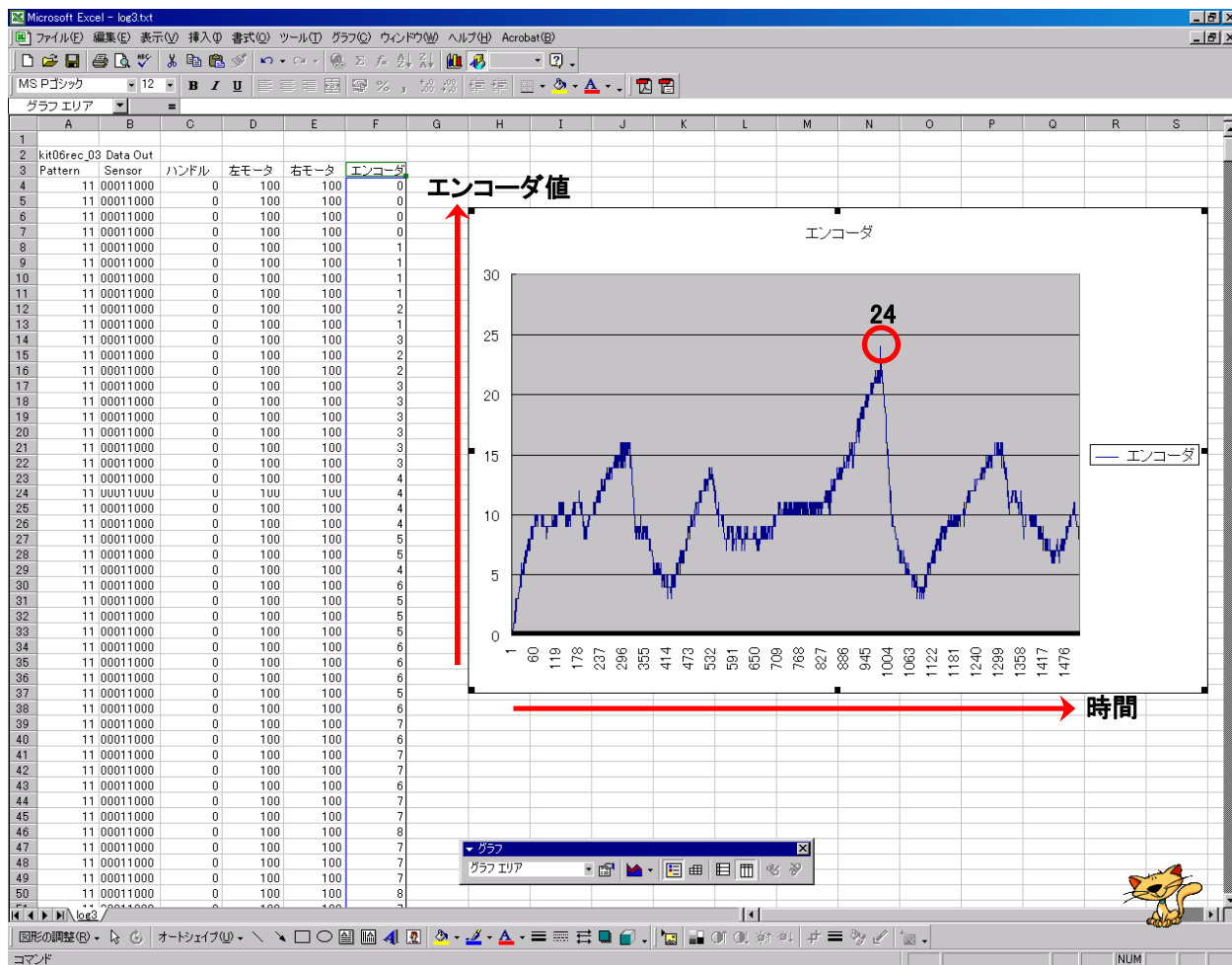


各項目は、各自設定してください。特に設定しなくても問題ありません。**次へ**をクリックします。

8



何処にグラフを追加するか選択します。「オブジェクト」を選択して、**完了**をクリックして完了です。



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ので、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、「10.92 パルスで、1m/s」です。最速は 24 なので、

$$10.92 : 1 = 24 : \text{最速のスピード}$$

$$\text{最速のスピード} = 2.2\text{m/s}$$

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。

8. 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 2進数	
11	00110000	
11	00110000	
11	00110000	
11	00110000	
11	00110000	
22	11111111	
22	00110000	
22	11111111	
22	00110000	
23	00110000	
23	00110000	
23	00110000	
23	00110000	
23	00110000	
23	01110000	
23	01110000	
23	01110000	
23	00110000	
23	00110000	
23	00110000	
23	01110000	
23	01110000	
23	01110000	
23	01110000	
23	01111111	
23	01111111	
23	00000000	
23	00000000	
23	00000000	
23	00000000	
23	00000000	
23	00000000	
23	00000000	
23	11100000	
23	11111111	
23	11111111	
23	10000000	
23	00000000	
23	00000000	

右クランクと判断するセンサ状態である 0x1f ではないので、そのまま進む!!

脱輪!!

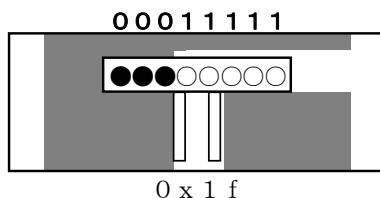
プログラムを見てみます。

```

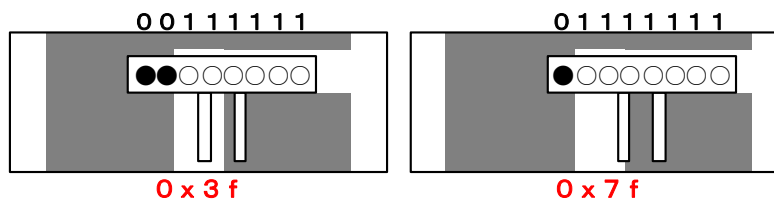
case 23:
  /* クロスライン後のトレース、クランク検出 */
  if( sensor_inp(MASK4_4)==0xf8 ) {
    /* 左クランクと判断→左クランククリア処理へ */
    led_out( 0x1 );
    handle( -38 );
    speed( 10 , 50 );
    pattern = 31;
    cnt1 = 0;
    break;
  }
  if( sensor_inp(MASK4_4)==0x1f ) {
    /* 右クランクと判断→右クランククリア処理へ */
    led_out( 0x2 );
    handle( 38 );
    speed( 50 , 10 );
    pattern = 41;
    cnt1 = 0;
    break;
  }
  if( iEncoder >= 11 ) { /* クロスライン後のスピード制御 */
    speed2( 0 , 0 );
  } else {
    speed2( 70 , 70 );
  }
  switch( sensor_inp(MASK3_3) ) {
    case 0x00:
      /* センタ→まっすぐ */
      handle( 0 );
      break;
    case 0x04:
    case 0x06:
    case 0x07:
    case 0x03:
      /* 左寄り→右曲げ */
      handle( 8 );
      break;
    case 0x20:
    case 0x60:
    case 0xe0:
    case 0xc0:
      /* 右寄り→左曲げ */
      handle( -8 );
      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)。



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。



8. データをエクセルで解析する

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

```

void main( void )
{
    int    i;
    unsigned char b;                                ローカル変数の追加

    ===== 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);                        センサ値をいったん b に保存
        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==0x3f || b==0x7f ) {          右クランクと判断する状態を追加
            /* 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;
        }
        if( iEncoder >= 11 ) {      /* クロスライン後のスピード制御 */
            speed2( 0 , 0 );
        } else {
            speed2( 70 , 70 );
        }
        switch( sensor_inp(MASK3_3) ) {
            case 0x00:
                /* センタ→まっすぐ */
                handle( 0 );
                break;
            case 0x04:
            case 0x06:
            case 0x07:
            case 0x03:
                /* 左寄り→右曲げ */
                handle( 8 );
                break;
            case 0x20:
            case 0x60:
            case 0xe0:
            case 0xc0:
                /* 右寄り→左曲げ */
                handle( -8 );
                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。

9. 参考文献

- ・ルネサス エレクトロニクス(株)
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
- ・ルネサス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
- ・(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
- ・電波新聞社 マイコン入門講座 大須賀威彦著 第1版
- ・ソフトバンク(株) C言語でH8マイコンを使いこなす 鹿取祐二著 第1版
- ・ソフトバンク(株) 新C言語入門シニア編 林晴比古著 初版
- ・共立出版(株) プログラマのための ANSI C 全書 L.Ammeraal 著
吉田敬一・竹内淑子・吉田恵美子訳 初版

マイコンカーラリーについての詳しい情報は、マイコンカーラリ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、ルネサス エレクトロニクス(株)のホームページをご覧ください。

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

の「マイコン」→「H8」、H8 ファミリページの「H8/300H シリーズ」でご覧頂けます

※リンクは、2010年5月現在の情報です。

