

マイコンカーラリー用  
**液晶・microSD 基板 (Ver.2)**  
**kit12\_38a プログラム**  
**解説マニュアル**  
**データ解析(microSD)編**  
**(R8C/38A 版)**

msdPrintf 文を使用する場合は、プロジェクトに「printf\_lib.c」ファイルを追加してください。「printf\_lib.c」が無い場合は、コンパイルエラーになります。

2013年度から、RY\_R8C38 ボードに搭載されているマイコンが R8C/38A から R8C/38C に変更されました。R8C/38A マイコンと R8C/38C マイコンは、機能的にほぼ互換で、マイコンカーで使う範囲においてはプログラムの変更はほとんどありません。よって、本マニュアルではマイコンの名称を『R8C/38A』で統一します。

本マニュアルで説明しているセット内容	液晶・microSD 基板、及び液晶・microSD 基板 Ver.2 ※本マニュアルの「液晶・microSD 基板」は、「液晶・microSD 基板、及び液晶・microSD 基板 Ver.2」と読み替えてください(別途、明記されている部分は除く)。 ※どちらの基板も同じプログラムで動作します
対象マイコンボード	RY_R8C38 ボード(R8C/38A マイコン)

第 1.02 版

2015.04.20

ジャパンマイコンカーラリー実行委員会  
株式会社日立ドキュメントソリューションズ

# 注意事項 (rev.6.0J)

## 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。
- ・すべての商標および登録商標は、それぞれの所有者に帰属します。

## 連絡先

株式会社 日立ドキュメントソリューションズ

〒135-0016 東京都江東区東陽六丁目 3 番 2 号 イースト 21 タワー

E-mail:himdx.m-carrally.dd@hitachi.com

# 目次

1. 概要	1
2. microSD カード	2
2.1 microSD カードについて	2
2.1.1 SD メモ리카ードの種類	2
2.1.2 SD メモ리카ードの規格	3
2.1.3 SD メモ리카ードの通信モード	3
2.2 microSD を使う	4
3. サンプルプログラム	5
3.1 プログラムの開発環境	5
3.2 サンプルプログラムのインストール	5
3.2.1 プログラムのダウンロード	5
3.2.2 インストール	6
3.3 ワークスペース「kit12msd_38a」を開く	7
3.4 プロジェクト	8
4. microSD 制御ライブラリ	9
4.1 「microsd_lib.c」で使用できる関数	9
4.2 プロジェクトに microsd_lib.c を追加する	21
4.3 コンパイラオプション	22
5. printf、scanf 制御ライブラリ	23
6. プロジェクト「msd01_38a」 microSD 関数の実行時間確認	25
6.1 概要	25
6.2 接続	25
6.3 プロジェクトの構成	26
6.4 プログラム	27
6.5 プログラムの解説	29
6.5.1 ヘッダファイルの取り込み	29
6.5.2 変数	29
6.5.3 ポートの入出力設定	29
6.5.4 microSD の初期化	30
6.5.5 microSD のイレーズ(0 クリア)	31
6.5.6 microSD ヘッダ書き込み	31
6.5.7 microSD からデータ読み込み	32
6.6 実行時間の測定方法	33
6.7 演習	35
6.8 関数の使用場面	36
7. プロジェクト「msd02_38a」 microSD にデータ記録	37
7.1 概要	37
7.2 接続	37
7.3 プロジェクトの構成	38
7.4 プログラム	38

7.5 setMicroSDdata 関数と microSDProcess 関数.....	42
7.5.1 概要.....	42
7.5.2 プログラムの流れ.....	43
7.5.3 各関数の処理内容.....	43
7.6 プログラムの解説.....	44
7.6.1 プロトタイプ宣言.....	44
7.6.2 変数.....	44
7.6.3 main 関数(初期化).....	45
7.6.4 パターン 0:スタート.....	47
7.6.5 パターン 1:microSD クリア、書き込みアドレスセット.....	47
7.6.6 パターン 2:データ記録中.....	48
7.6.7 パターン 3:最後のデータが書き込まれるまで待つ.....	48
7.6.8 パターン 4:終了処理が終わるまで待つ.....	48
7.6.9 パターン 5:タイトル転送、準備.....	49
7.6.10 パターン 6:microSD よりデータ読み込み.....	49
7.6.11 パターン 7:パソコンへデータ転送.....	50
7.6.12 パターン 99:終了.....	51
7.6.13 割り込み処理.....	51
7.7 データの取り込み方.....	53
7.8 int 型、long 型を記録するには.....	56
7.9 演習.....	56
7.10 演習の回答例.....	57
<b>8. プロジェクト「kit12msd01_38a」 走行データを microSD に記録.....</b>	<b>59</b>
8.1 概要.....	59
8.2 マイコンカーの構成.....	59
8.3 プロジェクトの構成.....	60
8.4 プログラム.....	60
8.5 プログラムの解説.....	67
8.5.1 変数.....	67
8.5.2 main 関数(初期化).....	68
8.5.3 パターン 0:スイッチ入力待ち.....	69
8.5.4 パターン 1:スタートバーが開いたかチェック.....	70
8.5.5 パターン 71:走行データ転送準備.....	70
8.5.6 パターン 72:最後のデータ書き込むまで待つ.....	71
8.5.7 パターン 73、74:プッシュスイッチが離されたかチェック.....	71
8.5.8 パターン 75:スイッチが押されたかチェック.....	72
8.5.9 パターン 76:タイトル送信.....	72
8.5.10 パターン 77:microSD よりデータ読み込み.....	72
8.5.11 パターン 78:データ転送.....	73
8.5.12 パターン 99:転送終了.....	74
8.5.13 割り込み処理.....	75
8.5.14 記録データをバッファに保存.....	76
8.6 プログラムの調整.....	77
8.6.1 自分のマイコンカーに合わせて調整.....	77
8.6.2 記録間隔の変更.....	77
8.7 走行からデータ転送までの流れ.....	78
8.7.1 走行データの取り込み.....	78
8.7.2 Tera Term の設定: 文字化けに対する設定.....	81
8.8 エクセルへの取り込み方.....	82

---

8.9 データをエクセルで解析する.....	83
<b>9. プロジェクト「msd_fat11_38a」 microSD にデータ記録(FAT32 版).....</b>	<b>86</b>
9.1 概要.....	86
9.2 接続.....	86
9.3 プロジェクトの構成.....	87
9.4 プログラム.....	87
9.5 FAT32 形式で microSD へデータを書き込む.....	92
9.6 プログラムの解説.....	93
9.6.1 変数.....	93
9.6.2 main 関数 (microSD の初期化).....	94
9.6.3 main 関数 (FAT32 でマウント).....	94
9.6.4 パターン 0: タイトル表示.....	97
9.6.5 パターン 1: タイトル表示.....	97
9.6.6 パターン 2: データ記録開始.....	97
9.6.7 パターン 3: データ記録中.....	98
9.6.8 パターン 4、5、99: 終了処理.....	98
9.6.9 割り込み処理.....	99
9.6.10 記録する内容.....	100
9.6.11 記録できる文字数と記録間隔について.....	101
<b>10. プロジェクト「kit12msd_fat11_38a」 走行データを microSD に記録(FAT32 対応版).....</b>	<b>103</b>
10.1 概要.....	103
10.2 接続.....	103
10.3 プロジェクトの構成.....	103
10.4 プログラム.....	104
10.5 プログラムの解説.....	108
10.5.1 変数.....	108
10.5.2 main 関数 (microSD の初期化).....	109
10.5.3 パターン 0: スイッチ入力待ち.....	110
10.5.4 パターン 1: スタートバーが開いたかチェック.....	112
10.5.5 パターン 101~103: microSD 終了処理.....	113
10.5.6 割り込み処理.....	113
10.5.7 記録する内容.....	114
<b>11. 参考文献.....</b>	<b>115</b>



## 1. 概要

※本マニュアルで使用している基板は「液晶・microSD 基板」と「および液晶・microSD 基板 Ver.2」です。どちらの基板も回路は互換なので、プログラムの変更はありません。

※本マニュアルの「液晶・microSD 基板」は、「液晶・microSD 基板、及び液晶・microSD 基板 Ver.2」と読み替えてください(別途、明記されている部分は除く)。

マイコンカーが自分の思い通りに走らない場合、マイコンカーの動きやセンサの状態を確認し、車体やプログラムを改良します。しかし、最近のマイコンカーは速度が速くなり、センサの状態を目で見て確認することが難しくなってきました。プログラムのこの辺りを変えれば走るかな？いやこっちはかな？などと「カン」に頼っても、分からないものは分かりません。そこで走行データを microSD に記録し、「カン」に頼らない論理的な解析ができるように、RY\_R8C38 ボードに搭載する「液晶・microSD 基板」を開発しました。

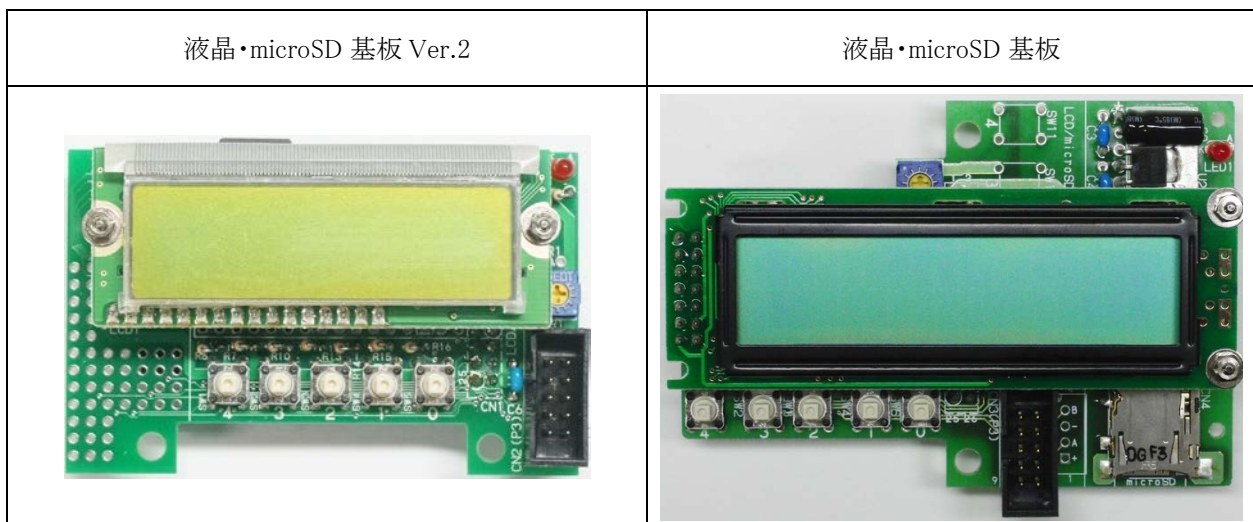
ただし、走行データを解析し、プログラムに反映させるためには、次のように**自分が想定しているマイコンカー(センサ)の状態とプログラムを理解していなければいけません。**

- ・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、ここのプログラムを直さなければいけない

このように、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です。

本マニュアルでは液晶・microSD 基板の microSD の仕様や使い方、マイコンカーの走行データを記録、取得し、解析する方法について説明していきます。液晶、プッシュスイッチの仕様や使い方については、「液晶・microSD 基板 kit12\_38a プログラム解説マニュアル液晶編(R8C/38A 版)」を参照してください。

2 種類の基板の違いは、液晶・microSD 基板 Ver.2 の液晶の方が小型、軽量になり、それに伴い基板外形も変更されています。回路的な違いはほとんど無く、どちらの基板も同じプログラムで動作します。



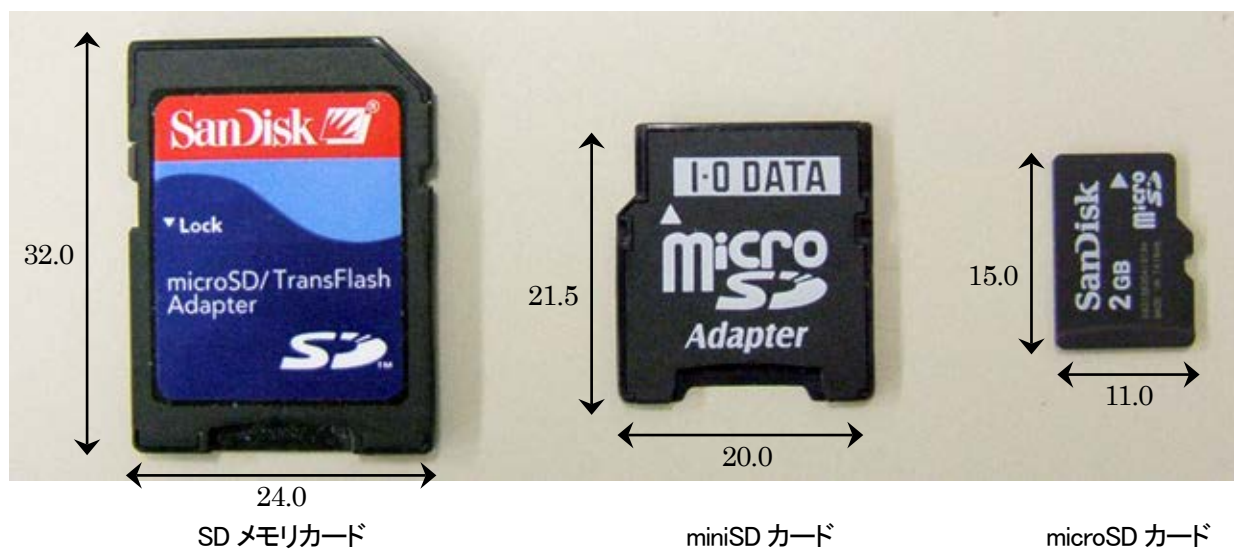
## 2. microSD カード

本書では、マイコンカーの走行状態を記録するためのデバイスとして、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。

### 2.1 microSD カードについて

#### 2.1.1 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示します。

	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 <sup>3</sup>	589mm <sup>3</sup>	165mm <sup>3</sup>
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



### 2.1.2 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。

	SD	SDHC	SDXC
制定年度	1999 年	2006 年 1 月	2009 年 1 月
正式名称	Secure Digital	SD High Capacity	SD eXtended Capacity
ファイル管理システム	FAT12、FAT16、FAT32	FAT32	exFAT
容量	～2GB	2GB～32GB	32GB～2TB
今回のプログラムでの対応	○	○	×

今回のプログラムでは、SDXC には対応していません。使用できるのは 32GB までの microSD となります。ただし、マイコンカーで使う場合は 4GB 以下の microSD で十分です。

### 2.1.3 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。

## 2.2 microSD を使う

マイコンカーでデータ記録を行う場合、内蔵 RAM と microSD を使用したときの特徴を下記に示します。

記憶メモリ	R8C/38A マイコン 内蔵 RAM	※	microSD (SD カード、SDHC カード)
記憶容量	9KB 程度 9×1024 =9,216bytes  ※RY_R8C38 ボード(R8C/38A マイコン)は、内蔵 RAM:10KB が内蔵されています。	<<<	2GB の microSD なら 2×1024×1024×1024 =2,147,483,648bytes
時間当たりの書き込み数	約 1μs で 1bytes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
外付け部品	不要(マイコン内蔵の RAM を使用)	>	microSD と、microSD 接続回路が必要
電源断での記録	RAM なので消えてしまう	<	フラッシュメモリなので電源が消えてもデータが消えない
プログラム容量	配列を確保するだけなので、ほとんどプログラム容量を使わない	>	microSD を FAT32 形式で書き込むプログラムで、ROM 約 10KB、RAM 約 1KB 使用
記録時間	10ms ごとに 64bytes のデータを書き込む場合は、 9216÷1 回の記録数 64bytes×記録間隔 10ms =1440[ms] =1.44 秒	<<<	10ms ごとに 64bytes のデータを書き込む場合は、 2147483648÷1 回の記録数 64bytes× 記録間隔 10ms ≒93 時間  ※容量 2GB で FAT 領域は除いた場合の計算
連続走行	1 回の走行分しか記録できない	<	数十回分の走行を記録可能 (FAT32 形式の場合、記録数が多くなると初めの領域確保に 10 秒程度の時間がかかる)

※A>Bで「Aの方が扱いやすい、性能がよい」 A<Bで「Bの方が扱いやすい、性能がよい」という意味です。

※msd01\_38a プロジェクト、msd02\_38a プロジェクト、kit12msd01\_38a プロジェクトの各プログラムで microSD への書き込みを行うと、FAT を壊します。Windows など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。

※msd\_fat11\_38a プロジェクト、kit12msd\_fat11\_38a プロジェクトの各プログラムで microSD へ書き込みを行っても FAT32 は壊しませんが、万が一 FAT32 を壊してしまうことを考えて、内容を消しても良い microSD を使ってください。

※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。

### 3. サンプルプログラム

#### 3.1 プログラムの開発環境

プログラムの開発は、ルネサス統合開発環境(High-performance Embedded Workshop)を使います。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 (R8C/38A 版)」を参照してください。


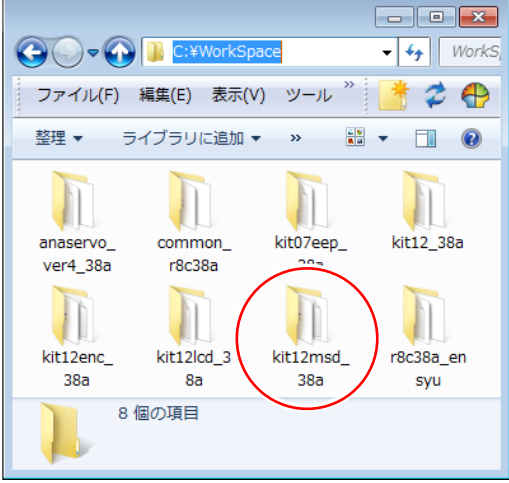
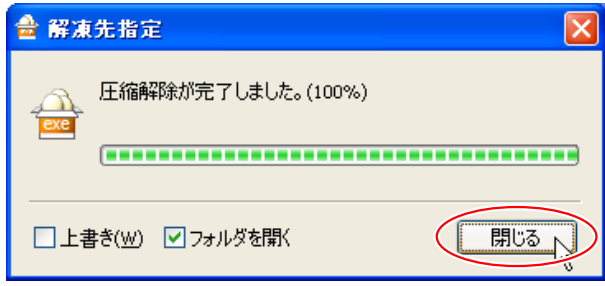
#### 3.2 サンプルプログラムのインストール

##### 3.2.1 プログラムのダウンロード


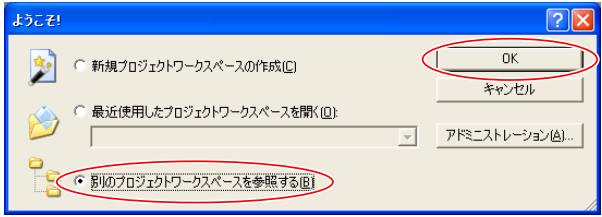
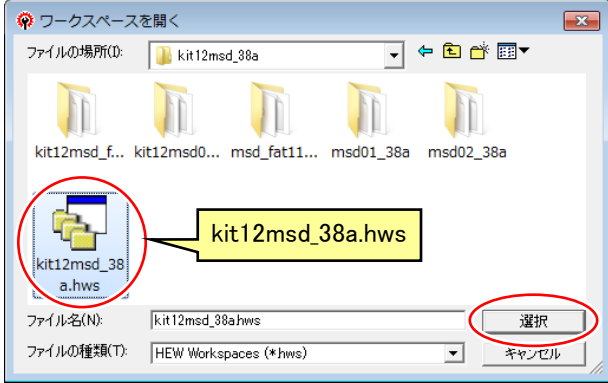
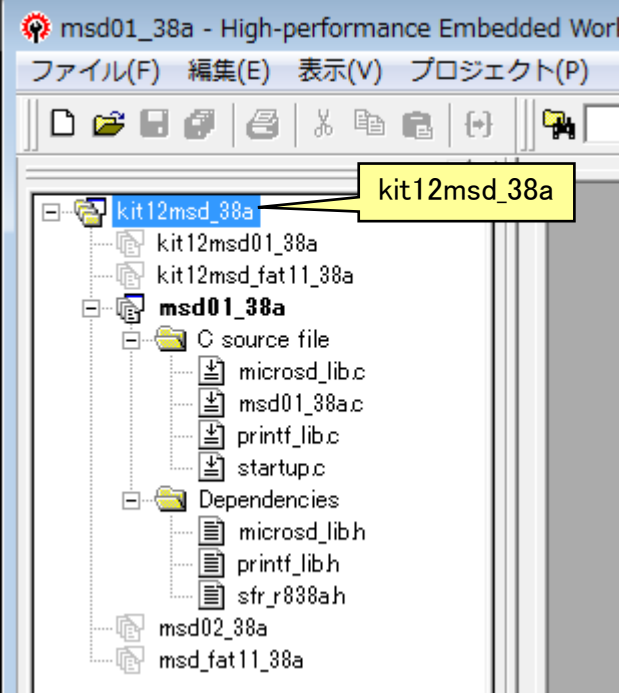
1		マイコンカーラリーサイト 「 <a href="http://www.mcr.gr.jp/index2.html">http://www.mcr.gr.jp/index2.html</a> 」 の「技術情報→ダウンロード」をアクセスします。												
2	<p><b>免責事項</b></p> <p>「マニュアル」、「ソフトウェア」は万全な体制で制作されており、通常の使用環境においては正常に動作するように作成されていますが、万が一「マニュアル」、「ソフトウェア」による損失・損害が発生した時には、『ジャパンマイコンカーラリー実行委員会』はいかなる場合も責任を負いません。個人の免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> <table border="1" data-bbox="240 1301 858 1496"> <thead> <tr> <th>対象マイコン</th> <th>内容</th> <th>更新日</th> </tr> </thead> <tbody> <tr> <td>R8C/38A</td> <td>R8C/38Aマイコン(RY_R8C38ボード)に関する資料</td> <td>2013.06.03 <b>NEW!!</b></td> </tr> <tr> <td>H8/3048F-ONE</td> <td>H8/3048F-ONEマイコン(RY3048Foneボード)用のサンプルプログラム、書き込みソフト</td> <td>2010.10.07</td> </tr> <tr> <td>H8/3048F</td> <td>H8/3048F-ONEマイコン(RY3048Foneボード)に</td> <td>~~~~~</td> </tr> </tbody> </table>	対象マイコン	内容	更新日	R8C/38A	R8C/38Aマイコン(RY_R8C38ボード)に関する資料	2013.06.03 <b>NEW!!</b>	H8/3048F-ONE	H8/3048F-ONEマイコン(RY3048Foneボード)用のサンプルプログラム、書き込みソフト	2010.10.07	H8/3048F	H8/3048F-ONEマイコン(RY3048Foneボード)に	~~~~~	「R8C/38A マイコン(RY_R8C38 ボード)」に関する資料」をクリックします。
対象マイコン	内容	更新日												
R8C/38A	R8C/38Aマイコン(RY_R8C38ボード)に関する資料	2013.06.03 <b>NEW!!</b>												
H8/3048F-ONE	H8/3048F-ONEマイコン(RY3048Foneボード)用のサンプルプログラム、書き込みソフト	2010.10.07												
H8/3048F	H8/3048F-ONEマイコン(RY3048Foneボード)に	~~~~~												
3	<table border="1" data-bbox="240 1518 858 1861"> <tbody> <tr> <td>液晶・microSD基板 「液晶・microSD基板 液晶・スイッチセット」 液晶とスイッチでパソコンを使わずにパラメータを設定、保存することができます。</td> <td>液晶・microSD基板 製作マニュアル 第1.01版 2012.07.11</td> <td>液晶・microSD基板 kit12_38aプログラム解説マニュアル 液晶編 第1.00版 2013.05.28</td> <td><a href="#">kit12lcd_38a.exe</a></td> </tr> <tr> <td>液晶・microSD基板 「拡張用microSD部品セット」 マイコンカーの走行状態をmicroSDに記録して、解析することのできる基板です。「液晶・microSD基板 液晶・スイッチセット」に追加するセットです。本セットのみでは、組み立てはできません。</td> <td>液晶・microSD基板 製作マニュアル 第1.01版 2012.07.11</td> <td>液晶・microSD基板 kit12_38aプログラム解説マニュアル データ解析(microSD)編 第1.00版 2013.05.28</td> <td><a href="#">kit12msd_38a.exe</a></td> </tr> </tbody> </table> <p style="text-align: center;">kit12msd_38a.exe</p>	液晶・microSD基板 「液晶・microSD基板 液晶・スイッチセット」 液晶とスイッチでパソコンを使わずにパラメータを設定、保存することができます。	液晶・microSD基板 製作マニュアル 第1.01版 2012.07.11	液晶・microSD基板 kit12_38aプログラム解説マニュアル 液晶編 第1.00版 2013.05.28	<a href="#">kit12lcd_38a.exe</a>	液晶・microSD基板 「拡張用microSD部品セット」 マイコンカーの走行状態をmicroSDに記録して、解析することのできる基板です。「液晶・microSD基板 液晶・スイッチセット」に追加するセットです。本セットのみでは、組み立てはできません。	液晶・microSD基板 製作マニュアル 第1.01版 2012.07.11	液晶・microSD基板 kit12_38aプログラム解説マニュアル データ解析(microSD)編 第1.00版 2013.05.28	<a href="#">kit12msd_38a.exe</a>	「kit12msd_38a.exe」をクリック、ダウンロードします。				
液晶・microSD基板 「液晶・microSD基板 液晶・スイッチセット」 液晶とスイッチでパソコンを使わずにパラメータを設定、保存することができます。	液晶・microSD基板 製作マニュアル 第1.01版 2012.07.11	液晶・microSD基板 kit12_38aプログラム解説マニュアル 液晶編 第1.00版 2013.05.28	<a href="#">kit12lcd_38a.exe</a>											
液晶・microSD基板 「拡張用microSD部品セット」 マイコンカーの走行状態をmicroSDに記録して、解析することのできる基板です。「液晶・microSD基板 液晶・スイッチセット」に追加するセットです。本セットのみでは、組み立てはできません。	液晶・microSD基板 製作マニュアル 第1.01版 2012.07.11	液晶・microSD基板 kit12_38aプログラム解説マニュアル データ解析(microSD)編 第1.00版 2013.05.28	<a href="#">kit12msd_38a.exe</a>											

3. サンプルプログラム

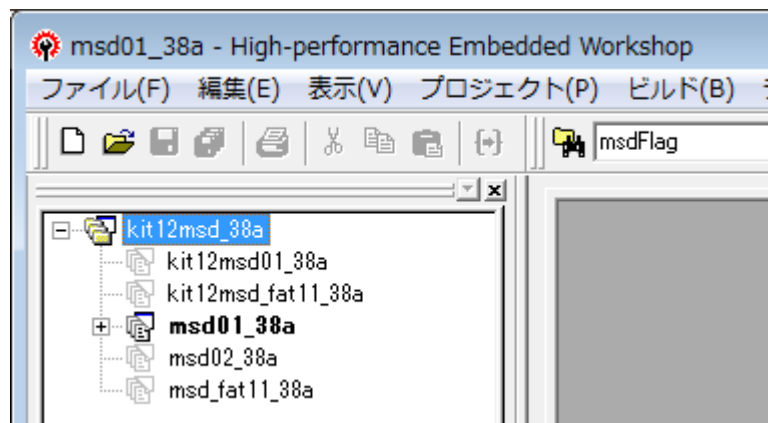
3.2.2 インストール

<p>1</p>		<p>「kit12msd_38a.exe」を実行します。                  圧縮解除をクリックします。                  ※解凍先フォルダは変更しないでください。</p>
<p>2</p>		<p>解凍が終わったら、                  「C ドライブ→Workspace」フォルダが開かれます。複数のフォルダがあります。今回使用するのは、「kit12msd_38a」です。</p>
<p>3</p>		<p>閉じるをクリックします。</p>

### 3.3 ワークスペース「kit12msd\_38a」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」を選択し、<b>OK</b>をクリックします。</p>
3		<p>Cドライブ→Workspace→kit12msd_38a の「kit12msd_38a.hws」を選択し、<b>選択</b>をクリックします。</p>
4		<p>ワークスペース「kit12msd_38a」が開かれます。</p>

### 3.4 プロジェクト



ワークスペース「kit12msd\_38a」には、5 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	FAT32 ※
msd01_38a	液晶・microSD 基板の microSD 制御関数の実行時間を確認するサンプルプログラムです。	未対応
msd02_38a	液晶・microSD 基板の microSD にデータを記録、パソコンへ転送するプログラムです。本プログラムでは、連続してデータを書き込む方法を説明します。	未対応
kit12msd01_38a	液晶・microSD 基板の microSD にマイコンカーの走行データを記録し、パソコンへデータを転送するプログラムです。走行プログラムは、「kit12_38a」を使用しています。	未対応
msd_fat11_38a	液晶・microSD 基板の microSD にデータを記録します。 記録は、FAT32 でフォーマットされた microSD にファイルとして書き込みます。	<b>対応</b>
kit12msd_fat11_38a	液晶・microSD 基板の microSD にマイコンカーの走行データを記録します。 記録は、FAT32 でフォーマットされた microSD にファイルとして書き込みます。 走行プログラムは、「kit12_38a」を使用しています。	<b>対応</b>

**※FAT32 未対応のプログラムを一度でも実行すると、FAT32 を壊します。FAT32 対応プログラムを実行するときは必ず FAT32 でフォーマットしてから使用してください。**

## 4. microSD 制御ライブラリ

### 4.1 「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。液晶・microSD 基板の microSD を使用する場合は、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\_r8c38a」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。

※Ver.は、「microsd\_lib.c」内の 4 行目に書かれています。

#### ■initMicroSD 関数

書式	int initMicroSD( void )
内容	microSD を初期化します。最初に必ず実行し、データを読み書きする準備をします。 <b>※Ver.2.00 より SDHC(2~32GB の microSD)に対応しました。</b>
引数	なし
戻り値	0:正常終了(準備完了) 1:ダミーデータ送信時、不正データ入力 2:CMD0 の返信コマンド受信エラー 3:CMD8 の返信コマンド受信エラー 4:CMD1 の返信コマンド受信エラー 5:CMD16 の返信コマンド受信エラー 6:CMD58 の返信コマンド受信エラー 7:ACMD41 の返信コマンド受信エラー 8:ACMD41 後の CMD58 の返信コマンド受信エラー 9: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、液晶・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 ) {     /* 初期化エラー */     printf( "microSD Initialize Error!!%n" ); }</pre>

4. microSD 制御ライブラリ

■readMicroSD 関数

書式	int readMicroSD( unsigned long address, signed 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"> <li>• unsigned long      microSD から読み込むアドレス</li> <li>• signed char*      読み込んだデータを格納する配列</li> </ul> <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	<p>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)</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、液晶・microSD 基板との接続が正しくないかです。</p>
使用例	<pre>signed char    msdBuff[ 512 ];                                /* 一時保存バッファ */  ret = readMicroSD( 0x0000 , msdBuff ); 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, signed 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"> <li>• unsigned long      microSD に書き込むアドレス</li> <li>• signed char*      書き込むデータを格納する配列</li> </ul> <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、液晶・microSD 基板との接続が正しくないかです。</p>
使用例	<pre>signed char    msdBuff[ 512 ];                                /* 一時保存バッファ */  ret = writeMicroSD( 0x0000 , msdBuff ); 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( signed 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。 ※CSD から、カード特性データ、カード固有情報などが分かります。詳しくはインターネットなどで検索してください。
引数	<ul style="list-style-type: none"> <li>●signed char* CSD データを格納する配列(16 バイト以上)</li> </ul> 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。
戻り値	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  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、液晶・microSD 基板との接続が正しくないかです。
使用例	<pre>signed char    msdBuff[ 512 ];          /* 一時保存バッファ          */  ret = getMicroSD_CSD( msdBuff );      /* msdBuff 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"を書き込みます)。
引数	<ul style="list-style-type: none"> <li>●unsigned long 消去開始アドレス(512 の倍数)</li> <li>●unsigned long 消去終了アドレス(512 の倍数-1)</li> </ul> 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。
戻り値	0:正常終了(イレーズ完了) 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:イレーズ後のテスト書き込みエラー  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、液晶・microSD 基板との接続が正しくないかです。
使用例	<pre>ret = eraseMicroSD( 0x0200, 0x0fff );  if( ret != 0x00 ) {     /* イレーズエラー */     printf( "microSD Erase Error!!\n" ); }</pre>

## 4. microSD 制御ライブラリ

## ■setMicroSDdata 関数

書式	int setMicroSDdata( signed 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"> <li>signed char * 書き込むデータを格納する配列</li> </ul> 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、ライントレース)が実行できなくなるので、この場合は無視して次に進むようにします。 <b>※microSD に書き込むアドレスは、microSDProcessStart 関数で指定したアドレスです。setMicroSDdata 関数が正常に終了すると次に書き込むアドレスは、512 バイト先のアドレスになります。</b>
使用例	microSDProcess 関数で説明します。

## ■microSDProcessStart 関数

書式	int microSDProcessStart( unsigned long address )
内容	setMicroSDdata 関数、microSDProcess 関数を実行する前に、この関数を実行します。microSD に書き込むアドレスを指定します。
引数	<ul style="list-style-type: none"> <li>unsigned long microSD に書き込むアドレス</li> </ul> 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。
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、既に microSDProcessStart 関数を実行しているか、コマンド送信エラーです。
使用例	ret = microSDProcessStart( 0x1000 ); /* 0x1000 番地から書き込みを行います */

## ■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、この関数を実行します。
引数	なし
戻り値	0: 正常終了(セット完了) 0 以外: 書き込み処理中  0 以外は書き込み処理中です。0 になるまで繰り返し実行してください。
使用例	// 書き込み処理が終わるまで繰り返す while( microSDProcessEnd() != 0 );

■microSDProcess 関数

書式	void microSDProcess( void )
内容	setMicroSDdata 関数でセットした 512 バイトのデータを、実際に書き込み作業を行う関数です。この関数は、割り込み処理などで、1ms ごとに実行してください。
引数	なし
戻り値	なし
使用例	<pre> #pragma interrupt intTRB(vect=24) void intTRB( void ) <b>タイマRB割り込み(1msごとの割り込み)</b> {     signed char *p;      cnt1++;      /* microSD間欠書き込み処理(1msごとに実行) */     <b>microSDProcess();</b> <b>1msごとに実行する</b>      /* microSD記録処理 */     if( msdFlag == 1 ) {         /* 記録間隔のチェック */         msdTimer++;         if( msdTimer &gt;= 10 ) { <b>10msごとに実行する</b>             msdTimer = 0;             p = msdBuff + msdBuffAddress;              /* RAMに記録 ここから */             *p++ = p0;             *p++ = dipsw_get();             /* RAMに記録 ここまで */              msdBuffAddress += 64; /* RAMの記録アドレスを次へ */              if( msdBuffAddress &gt;= 512 ) {                 /* 512個になったら、microSDに記録する */                 msdBuffAddress = 0;                 <b>setMicroSDdata( msdBuff );</b> <b>80msごとに実行</b>                 msdWorkAddress += 512;                 if( msdWorkAddress &gt;= msdEndAddress ) {                     /* 記録処理終了 */                     msdFlag = 0;                 }             }         }     } } </pre>
その他	msdPrintf関数を使用するときは、setMicroSDdata関数は使用しません。詳しくは、msdPrintf関数を参照してください。

4. microSD 制御ライブラリ

■checkMicroSDProcess 関数

書式	int checkMicroSDProcess( void )
内容	microSDProcess 関数で実行している状態を確認します。
引数	なし
戻り値	<p>0 : 処理なし                      11 : 次の書き込み待機中                      0 と 11 以外: 書き込み処理中</p> <p>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。</p>
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 ) {     /* 書き込み待機中なら実行 */ }</pre>

■setMicroSDLedPort 関数

書式	void setMicroSDLedPort( char *p, char *pd, int bit )
内容	microSD の動作をモニタする LED のポートを設定します。 microSD にデータを読み書きしているとき、この関数で設定したポートの LED を点滅させます。
引数	<ul style="list-style-type: none"> <li>● char * モニタ LED のあるポート</li> <li>● char * モニタ LED のあるポートの入出力設定ポート</li> <li>● int モニタ LED のあるポートのビット</li> </ul>
戻り値	なし
使用例	<pre>void main( void ) {     int i, ret;      /* マイコン機能の初期化 */     init(); /* 初期化 */     setMicroSDLedPort( &amp;p6, &amp;pd6, 0 ); /* microSD モニタLED設定 */     asm( " fset I " ); /* 全体の割り込み許可 */     以下、略 }  #pragma interrupt intTRB(vect=24) void intTRB( void ) {     /* microSD間欠書き込み処理(1msごとに実行) */     microSDProcess(); }</pre> <p>※モニタLEDを制御している関数は、microSDProcess関数です。 setMicroSDLedPort関数でポートの設定をした場合は、microSDProcess関数を割り込み処理などで1msごとに実行してください。</p>

■mountMicroSD\_FAT32 関数 (Ver.3.00～)

書式	int mountMicroSD_FAT32( void )
内容	microSD を FAT32 で書き込むための準備します。 <b>FAT32 以外には対応していません。必ず FAT32 でフォーマットした microSD を使ってください。</b> <b>initMicroSD 関数が成功した後に実行してください。</b>
引数	なし
戻り値	0:FAT32 でマウント完了(成功) 1:マウントできず(エラー)
使用例	<pre> /* microSD 初期化 */ ret = initMicroSD();  /* microSD を FAT32 でマウント */ ret = <b>mountMicroSD_FAT32()</b>; if( ret != 0x00 ) {     printf( "microSD は FAT32 のフォーマットではありません。¥n" );     printf( "FAT32 でフォーマットしてください。¥n" ); } else {     printf( "microSD は FAT32 フォーマットです。¥n" ); } </pre>

■readMicroSDNumber 関数 (Ver.3.00～)

書式	int readMicroSDNumber( void )
内容	microSD から番号を取得します。 書き込むファイル名を連番にするため、BPB(BIOS Parameter Block)領域に、番号を埋め込んでいます。この番号を読み込みます。パソコンからは見えない領域に書き込んでいるので、パソコンからは変更できません。
引数	なし
戻り値	-1: エラー 0 以上: 値
使用例	i = readMicroSDNumber(); /* microSD の空き領域から番号読み込み*/

■writeMicroSDNumber 関数 (Ver.3.00～)

書式	int writeMicroSDNumber( int number )
内容	microSD に番号を書き込みます。 書き込むファイル名を連番にするため、BPB(BIOS Parameter Block)領域に、番号を埋め込んでいます。この番号を書き込みます。
引数	● int 書き込む番号
戻り値	-1:エラー 0:書き込み完了
使用例	writeMicroSDNumber( i ); /* microSD の空き領域へ番号書き込み */

## 4. microSD 制御ライブラリ

## ■writeFile 関数(Ver.3.00～)

書式	<code>int writeFile( const char *s, unsigned long fileSize )</code>
内容	FAT32 形式でファイルを開き、ファイルサイズで指定した領域を確保します。 <b>microSD の中に、たくさんのファイルや容量の大きいファイルがあるとき、領域の確保に時間がかかります。できるだけファイル数は少なく、保存されているファイルの容量は小さい microSD を使用してください。</b>
引数	<ul style="list-style-type: none"> <li>• char *           ファイル名(8+ピリオド+3 形式)</li> <li>• unsigned long   ファイルサイズ(512 の倍数)</li> </ul>
戻り値	0:成功 0 以外:失敗
使用例	<code>ret = writeFile( "abcd.csv", 64000 );</code>
その他	writeFile 関数で開いたファイルへのデータ書き込みは、setMicroSDdata 関数を使います。

## ■setDateStamp 関数(Ver.3.00～)

書式	<code>void setDateStamp( int y, int m, int d )</code>
内容	writeFile 関数でファイルを作るとき、日付を設定します。
引数	<ul style="list-style-type: none"> <li>• int 年</li> <li>• int 月</li> <li>• int 日</li> </ul>
戻り値	なし
使用例	<code>setDateStamp( 2012, 4, 19 ); // 2012 年 4 月 19 日</code>

## ■setTimeStamp 関数(Ver.3.00～)

書式	<code>void setTimeStamp( int h, int m, int s )</code>
内容	writeFile 関数でファイルを作るとき、時刻を設定します。
引数	<ul style="list-style-type: none"> <li>• int 時</li> <li>• int 分</li> <li>• int 秒(偶数のみ)</li> </ul>
戻り値	なし
使用例	<code>setTimeStamp( 20, 30, 40 ); // 20 時 30 分 40 秒</code>

■getCompileYear 関数(Ver.3.00～)

書式	int getCompileYear( const char *p )
内容	コンパイル時の年を取得します。 writeFile 関数でファイルを作るとき、マイコンには日付データが無いので、コンパイルしたときの日付をファイルの日付にすると便利です。
引数	const char * __DATE__配列の位置
戻り値	コンパイル時の年
使用例	<pre>const char *C_DATE = __DATE__;          /* コンパイルした日付          */       setDateStamp( <b>getCompileYear( C_DATE )</b>,                     getCompileMonth( C_DATE ), getCompileDay( C_DATE ) );</pre>

■getCompileMonth 関数(Ver.3.00～)

書式	int getCompileMonth( const char *p )
内容	コンパイル時の月を取得します。 writeFile 関数でファイルを作るとき、マイコンには日付データが無いので、コンパイルしたときの日付をファイルの日付にすると便利です。
引数	const char * __DATE__配列の位置
戻り値	コンパイル時の月
使用例	<pre>const char *C_DATE = __DATE__;          /* コンパイルした日付          */       setDateStamp( getCompileYear( C_DATE ),                     <b>getCompileMonth( C_DATE )</b>, getCompileDay( C_DATE ) );</pre>

■getCompileDay 関数(Ver.3.00～)

書式	int getCompileDay( const char *p )
内容	コンパイル時の日を取得します。 writeFile 関数でファイルを作るとき、マイコンには日付データが無いので、コンパイルしたときの日付をファイルの日付にすると便利です。
引数	const char * __DATE__配列の位置
戻り値	コンパイル時の日
使用例	<pre>const char *C_DATE = __DATE__;          /* コンパイルした日付          */       setDateStamp( getCompileYear( C_DATE ),                     getCompileMonth( C_DATE ), <b>getCompileDay( C_DATE )</b> );</pre>

## 4. microSD 制御ライブラリ

## ■getCompileHour 関数 (Ver.3.00～)

書式	<code>int getCompileHour( const char *p )</code>
内容	コンパイル時の時(hour)を取得します。 writeFile 関数でファイルを作るとき、マイコンには日付データが無いので、コンパイルしたときの時刻をファイルの時刻にすると便利です。
引数	<code>const char * __TIME__</code> 配列の位置
戻り値	コンパイル時の時(hour)
使用例	<pre>const char *C_TIME = __TIME__;          /* コンパイルした時間          */  setTimeStamp( <b>getCompileHour( C_TIME )</b>,               getCompilerMinute( C_TIME), getCompilerSecond( C_TIME ) );</pre>

## ■getCompilerMinute 関数 (Ver.3.00～)

書式	<code>int getCompilerMinute( const char *p )</code>
内容	コンパイル時の分を取得します。 writeFile 関数でファイルを作るとき、マイコンには日付データが無いので、コンパイルしたときの時刻をファイルの時刻にすると便利です。
引数	<code>const char * __TIME__</code> 配列の位置
戻り値	コンパイル時の分
使用例	<pre>const char *C_TIME = __TIME__;          /* コンパイルした時間          */  setTimeStamp( getCompileHour( C_TIME ),               <b>getCompilerMinute( C_TIME)</b>, getCompilerSecond( C_TIME ) );</pre>

## ■getCompilerSecond 関数 (Ver.3.00～)

書式	<code>int getCompilerSecond( const char *p )</code>
内容	コンパイル時の秒を取得します。 writeFile 関数でファイルを作るとき、マイコンには日付データが無いので、コンパイルしたときの時刻をファイルの時刻にすると便利です。
引数	<code>const char * __TIME__</code> 配列の位置
戻り値	コンパイル時の秒
使用例	<pre>const char *C_TIME = __TIME__;          /* コンパイルした時間          */  setTimeStamp( getCompileHour( C_TIME ),               getCompilerMinute( C_TIME), <b>getCompilerSecond( C_TIME )</b> );</pre>



■convertDecimalToStr 関数(Ver.3.00～)

書式	void convertDecimalToStr( int value, int keta, signed char *p )
内容	int 型のデータを、10 進数文字列に変換します。
引数	<ul style="list-style-type: none"> <li>• int 変換する値</li> <li>• int 変換する桁数(マイナスも含んだ数)</li> <li>• char* 変換した値を格納する配列</li> </ul>
戻り値	なし
使用例	<pre>convertDecimalToStr( 1234, 8, p ); // p ポインタが示す位置に、'00001234' を                                      書き込み p += 8;                               // ポインタを進ませる</pre>

■convertHexToStr 関数(Ver.3.00～)

書式	void convertHexToStr( unsigned int value, int keta, signed char *p )
内容	unsigned int 型のデータを、16 進数文字列に変換します。
引数	<ul style="list-style-type: none"> <li>• unsigned int 変換する値</li> <li>• int 変換する桁数(マイナスも含んだ数)</li> <li>• char* 変換した値を格納する配列</li> </ul>
戻り値	なし
使用例	<pre>convertHexToStr( 0x1a1b, 6, p ); // p ポインタが示す位置に、'001a1b' を                                      書き込み p += 6;                               // ポインタを進ませる</pre>

■convertBinaryToStr 関数(Ver.3.00～)

書式	void convertBinaryToStr( unsigned char value, int keta, signed char *p )
内容	unsigned char 型のデータを、2 進数文字列に変換します。
引数	<ul style="list-style-type: none"> <li>• unsigned char 変換する値</li> <li>• int 変換する桁数(マイナスも含んだ数)</li> <li>• char* 変換した値を格納する配列</li> </ul>
戻り値	なし
使用例	<pre>convertBinaryToStr( 0xa5, 8, p ); // p ポインタが示す位置に、'10100101' を                                      書き込み p += 8;                               // ポインタを進ませる</pre>

4. microSD 制御ライブラリ

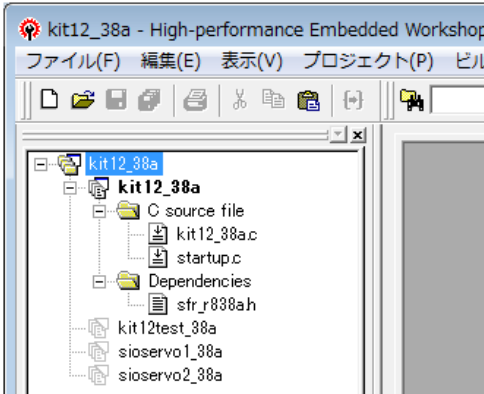
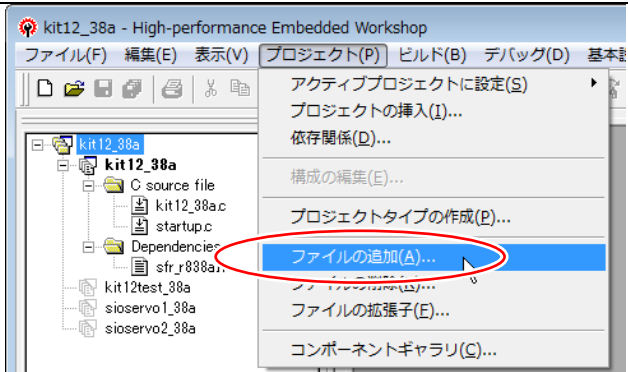
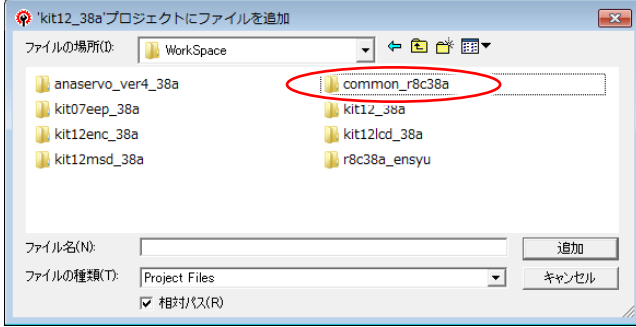
■msdPrintf 関数(Ver.3.10～)

書式	<code>int msdPrintf( char *fmt, arg1, arg2, ... )</code>
内容	printf 関数とほぼ同じ電文で、microSD へ文字列を書き込むことができます。 <b>msdPrintf 関数を使用するときは、「printf.lib.c」も追加してください。</b>
引数	<ul style="list-style-type: none"> <li>• *fmt            フォーマット変換を指定する文字列                         <ul style="list-style-type: none"> <li>• %[1~6]d … int 型の値を整数に変換します。数値は桁数です。 負の数の場合は“-”を含めて指定した桁数に変換します。</li> <li>• %[1~8]b … char 型の値を 2 進数に変換します。数値は桁数です。</li> <li>• %[1~4]x … int 型の値を 16 進数に変換します。数値は桁数です。</li> <li>• %c            … 文字に変換します。</li> </ul> </li> <li>• arg1～            定数、または表示データの格納された変数や式</li> </ul>
戻り値	0:成功 1:書き込み中で書き込みできず 2:書き込み中止(ファイルクローズ)
使用例 1	<pre>#pragma interrupt intTRB(vect=24) void intTRB( void ) <b>タイマRB割り込み(1msごとの割り込み)</b> {     static int line_no;           /* 行番号 */     int ret;      cnt1++;      /* microSD間欠書き込み処理(1msごとに実行) */     <b>microSDProcess();</b>           <b>1msごとに実行する</b>      /* microSD記録処理 */     if( msdFlag == 1 ) {         /* 記録間隔のチェック */         msdTimer++;         if( msdTimer &gt;= 10 ) { <b>10msごとに実行する</b>             msdTimer = 0;              ret = msdPrintf( "%4d,=%8b%", %4x¥r¥n",                 line_no,           // 行番号                 p0,               // ポート0                 dipsw_get()       // ディップスイッチ             );             if( ret == 2 ) msdFlag = 0; // ファイルクローズなら終了              if( ++line_no &gt;= 10000 ) line_no = 0;         }     } }</pre>
使用例 2	<p>msdPrintf文が終了するまで待つ場合は、次のようにしてください。 ただし、最大で10ms間、この行で処理が止まります。</p> <pre>msdPrintf ( "%6d, %4x¥r¥n" , 123, 0x4567 ); while( checkMsdPrintf() ); // msdPrintf処理完了待ち msdPrintf ( "%6d, %4x¥r¥n" , 4567, 0xabcd ); while( checkMsdPrintf() ); // msdPrintf処理完了待ち</pre>

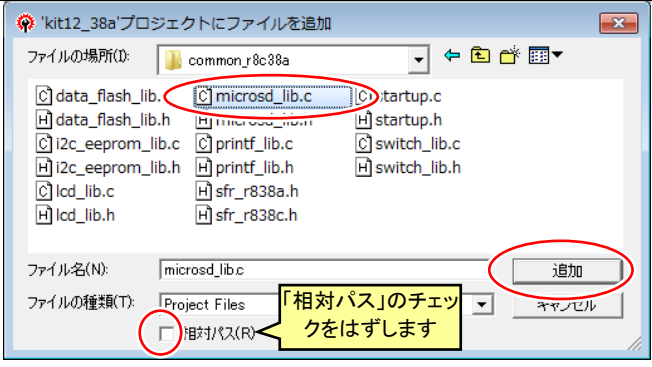
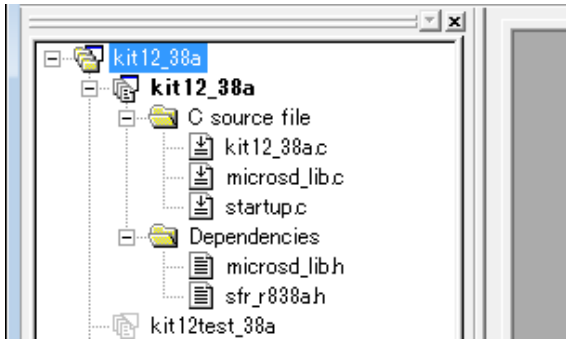
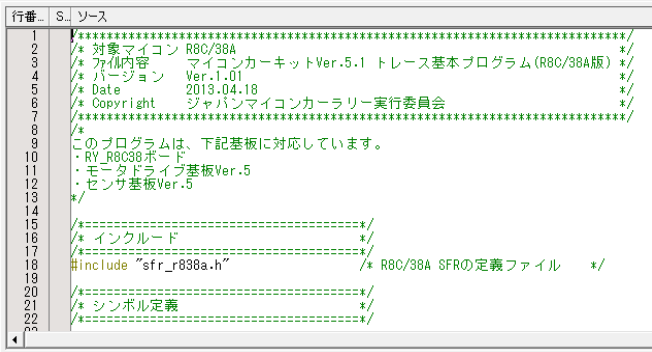
<p>使用例3</p>	<pre>msdPrintf( "%6d" , 123 ); // 出力:000123 msdPrintf( "%6d" , -123 ); // 出力:-00123 msdPrintf( "%8b" , 0x59 ); // 出力:01011001 msdPrintf( "%4x" , 23456 ); // 出力: 5ba0 msdPrintf( "%c" , 'a' ); // 出力: a msdPrintf( "MCR!%n" ); // 出力: MCR! (改行)</pre>
<p>注意点</p>	<ul style="list-style-type: none"> <li>microSDに展開される文字数は、1行(CR(¥r), LF(¥n)を含めて)64文字までです。</li> <li>msdPrintf関数は、checkMsdPrintf関数の戻り値が0の状態で行ってください。最大で10ms以内に終わりますので、割り込み処理などで10ms毎に書き込む場合は、書き込みが終わったかのチェックは不要です。</li> <li>引数は、20個までです。</li> <li>10ms以下でデータを記録したい場合は、変数に値を保存しておき、msdPrintf関数を実行するときに、まとめて出力してください。                  例) msdPrintf( "%3d%3d¥r¥n%3d%3d¥r¥n" , s1, m1, s2, m2 );                  s1とm1:5ms前の値、s2とm2:今回の値</li> </ul>

## 4.2 プロジェクトに microsd\_lib.c を追加する

ワークスペース「kit12\_38a」のプロジェクト「kit12\_38a」に、microsd\_lib.c を追加する方法を下記に示します。

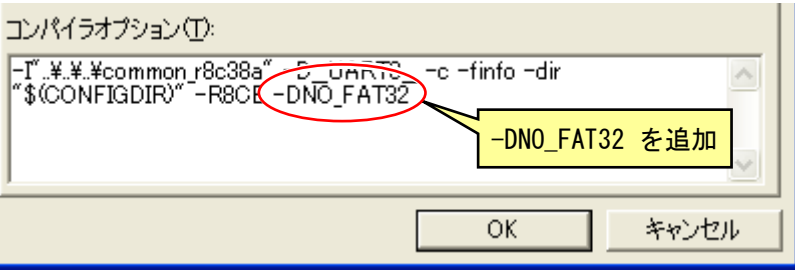
<p>1</p>		<p>ルネサス統合開発環境でワークスペース「kit12_38a」を開きます。                  プロジェクト「kit12_38a」を有効なプロジェクトにします(太字であれば OK です)。</p>
<p>2</p>		<p>「プロジェクト→ファイルの追加」を選択します。差し上げます</p>
<p>3</p>		<p>「C:\¥Workspace」フォルダにある、「common_r8c38a」フォルダを開きます。</p>

4. microSD 制御ライブラリ

4		<p>最初に相対パス欄のチェックを<b>はずします</b>。 「microsd_lib.c」を選択し、「追加」をクリックします。</p> <p>※「microsd_lib.h」は追加しません。</p>
5		<p>リストに、「microsd_lib.c」が追加されました。 「microsd_lib.h」は Dependencies 欄に自動で追加されます。</p>
6		<p>後は、「kit12_38a.c」ファイル内に、microSD 制御に関するプログラムを追加してください。</p>

4.3 コンパイラオプション

「microsd\_lib.c」には、1 つのコンパイラオプションがあります。

コンパイラオプション	説明
-DNO_FAT32	<p>microSD をメモリとしてのみ使用し、FAT32 を使わない場合は、FAT32 部分のプログラムを OFF にすることができます。OFF にする場合は、下記のコンパイラオプションを追加してください。</p> 

## 5. printf、scanf 制御ライブラリ

「printf\_lib.c」は、printf 関数、scanf 関数を使い、パソコンなどで文字のやり取りをできるようにするファイルです。

printf 関数や scanf 関数を使用する場合は、プロジェクトに「printf\_lib.c」を追加して使用します。

「printf\_lib.c」は、「C:\¥WorkSpace¥common\_r8c38a」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。

### ■init\_uart0\_printf 関数

書式	void init_uart0_printf( int sp )
内容	printf 関数、scanf 関数の初期化と通信機能(UART0)の設定をします。通信速度は選ぶことができます。
引数	<ul style="list-style-type: none"> <li>• int sp    SPEED_4800    …通信速度 4800bps</li> <li>          SPEED_9600    …通信速度 9600bps</li> <li>          SPEED_19200 …通信速度 19200bps</li> <li>          SPEED_38400 …通信速度 38400bps</li> </ul> <p>「SPEED_xxxxx」は、printf_lib.h ファイルで定義されています。引数には上記の 4 種類から選択します。</p>
戻り値	なし
使用例	<code>init_uart0_printf( SPEED_9600 );    /* UART0とprintf関連の初期化    */</code>

※UART0(シリアルコミュニケーション 0)や通信については、「マイコン実習マニュアル(R8C/38A 板)」を参照してください。

■printf 関数

書式	<code>int printf( const char *format, ... )</code>
内容	引数で設定した文字(書式文字列)をパソコンなどへ出力します。
引数	<ul style="list-style-type: none"> <li>● <code>const char *format</code>    書式文字列</li> <li>●                            <code>...</code>        可変個引数</li> </ul>
戻り値	出力成功:出力したバイト数    出力失敗:負の数
使用例	<pre>int a = 10;  printf( "Hello world\n" ); printf( "a = %d\n", a );</pre> <p>実行すると下記のように、表示されます。</p> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 10px auto;"> <pre>Hello world a = 10</pre> </div>

■scanf 関数

書式	<code>int scanf( const char *format, ... )</code>
内容	パソコンなどから送られてきた文字を、引数で設定した内容(書式文字列)に応じて取得します。
引数	<ul style="list-style-type: none"> <li>● <code>const char *format</code>    書式文字列</li> <li>●                            <code>...</code>        可変個引数</li> </ul>
戻り値	入力成功:入力された値    入力失敗:EOF(-1)
使用例	<pre>int i;  printf( "数字を入力してください: %n" ); scanf( "%d", &amp;i ); printf( "入力した値: %d\n", i );</pre> <p>実行すると下記のように、表示されます。</p> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 10px auto;"> <pre>数字を入力してください: 10 入力した値: 10</pre> </div> <p style="margin-left: 200px;">←キーボードから入力した値</p> <p style="margin-left: 200px;">←入力された値を表示</p>

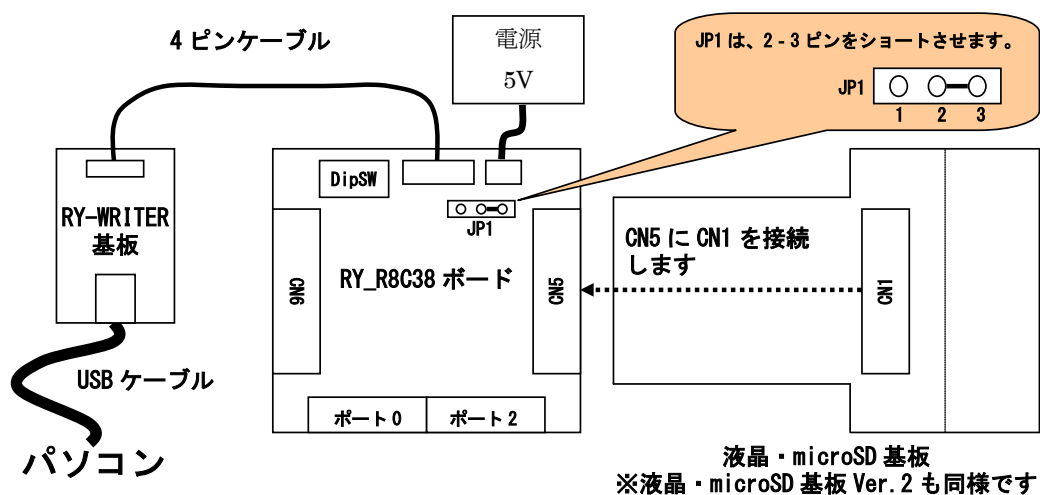
## 6. プロジェクト「msd01\_38a」 microSD 関数の実行時間確認

### 6.1 概要

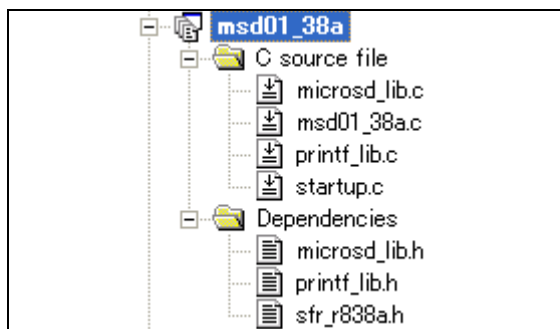
このプログラムは、液晶・microSD 基板の microSD を制御するための関数が正常に実行できるかチェックするとともに、実行時間を測定する確認用のプログラムです。

### 6.2 接続

- RY\_R8C38 ボードの CN5(ポート 3、ポート 5、ポート 6)と液晶・microSD 基板の CN1 のコネクタを重ね合わせまして接続します。
- RY\_R8C38 ボードとパソコン間を RY-WRITER 基板、USB ケーブル、4 ピンケーブルで接続します。



### 6.3 プロジェクトの構成



	ファイル名	内容
1	microsd_lib.c	microSD 制御ライブラリです。microSD を使用する場合は、このファイルを追加します。 ファイルの位置→C:\¥WorkSpace¥common_r8c38a¥microsd_lib.c
2	msd01_38a.c	実際に制御するプログラムが書かれています。R8C/38A 内蔵周辺機能(SFR)の初期化も行います。 ファイルの位置→C:\¥WorkSpace¥kit12msd_38a¥msd01_38a¥msd01_38a.c
3	printf_lib.c	通信をするための設定、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。 ファイルの位置→C:\¥WorkSpace¥common_r8c38a¥printf_lib.c
4	startup.c	固定割り込みベクタアドレスの設定、スタートアッププログラム、RAM の初期化(初期値のないグローバル変数、初期値のあるグローバル変数の設定)などを行います。 ファイルの位置→C:\¥WorkSpace¥kit12msd_38a¥msd01_38a¥startup.c
5	microsd_lib.h	microSD 制御ライブラリのヘッダファイルです。 ファイルの位置→C:\¥WorkSpace¥common_r8c38a¥microsd_lib.h
6	printf_lib.h	printf、scanf 制御ライブラリのヘッダファイルです。 ファイルの位置→C:\¥WorkSpace¥common_r8c38a¥printf_lib.h
7	sfr_r838a.h	R8C/38A マイコンの内蔵周辺機能を制御するためのレジスタ(Special Function Register)を定義したファイルです。 ファイルの位置→C:\¥WorkSpace¥common_r8c38a¥sfr_r838a.h



## 6.4 プログラム

プログラムのゴシック体部分が、microSD 制御で追加した部分です。

```

1 : /******
2 : /* 対象マイコン R8C/38A */
3 : /* ファイル内容 microSD基板の実験 */
4 : /* バージョン Ver. 1.00 */
5 : /* Date 2011.04.01 */
6 : /* Copyright ジャパンマイコンカーラー実行委員会 */
7 : /******
8 :
9 : /*
10 : 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、
11 : 実行時間を測定する確認用のプログラムです。
12 : */
13 :
14 : /*=====*/
15 : /* インクルード */
16 : /*=====*/
17 : #include <stdio.h>
18 : #include "sfr_r838a.h" /* R8C/38A SFRの定義ファイル */
19 : #include "printf_lib.h" /* printf使用ライブラリ */
20 : #include "microsd_lib.h" /* microSD制御ライブラリ */
21 :
22 : /*=====*/
23 : /* シンボル定義 */
24 : /*=====*/
25 :
26 : /*=====*/
27 : /* プロトタイプ宣言 */
28 : /*=====*/
29 : void init( void );
30 :
31 : /*=====*/
32 : /* グローバル変数の宣言 */
33 : /*=====*/
34 : unsigned long cnt1; /* 時間計測用 */
35 :
36 : /* microSD関連変数 */
37 : signed char msdBuff[ 512 ]; /* 一時保存バッファ */
38 :
39 : /******
40 : /* メインプログラム */
41 : /******
42 : void main( void )
43 : {
44 :     int i, ret;
45 :     unsigned long l;
46 :
47 :     init(); /* SFRの初期化 */
48 :     init_uart0_printf( SPEED_9600 ); /* UART0とprintf関連の初期化 */
49 :     asm(" fset I "); /* 全体の割り込み許可 */
50 :
51 :     printf( "YnmicroSD Test Program (RY_R8C38) Ver.1.00YnYn" );
52 :
53 :     /* microSD初期化 */
54 :     cnt1 = 0;
55 :     ret = initMicroSD();
56 :     l = cnt1;
57 :     if( ret != 0x00 ) {
58 :         printf( "microSD Initialize Error!!Yn" ); /* 初期化エラー */
59 :         while( 1 ); /* 終了 */
60 :     } else {
61 :         printf( "microSD Initialize Time = %ldmsYn", l );
62 :     }
63 :
64 :     /* microSDイレーズ */
65 :     cnt1 = 0;
66 :     ret = eraseMicroSD( 0x00000, 0x5dc00-1 );
67 :     l = cnt1;
68 :     if( ret != 0x00 ) {
69 :         printf( "microSD Erase Error!!Yn" ); /* イレーズエラー */
70 :         while( 1 ); /* 終了 */
71 :     } else {
72 :         printf( "microSD Erase Time = %ldmsYn", l );
73 :     }
74 :
75 :     /* バッファにダミーデータ書き込み */
76 :     for( i=0; i<512; i++ ) {
77 :         msdBuff[i] = i % 0x100;
78 :     }
79 :
80 :     /* microSD書き込み */
81 :     cnt1 = 0;
82 :     ret = writeMicroSD( 0x0000 , msdBuff );

```

## 6. プロジェクト「msd01\_38a」 microSD 関数の実行時間確認

```

83 :     l = cnt1;
84 :     if( ret != 0x00 ) {
85 :         printf( "microSD Write Error!!\n" );    /* 書き込みエラー */
86 :         while( 1 ); /* 終了 */
87 :     } else {
88 :         printf( "microSD Write Time = %ldms\n", l );
89 :     }
90 :
91 :     /* バッファクリア */
92 :     for( i=0; i<512; i++ ) {
93 :         msdBuff[i] = 0x00;
94 :     }
95 :
96 :     /* microSD読み込み */
97 :     cnt1 = 0;
98 :     ret = readMicroSD( 0x0000 , msdBuff );
99 :     l = cnt1;
100 :    if( ret != 0x00 ) {
101 :        printf( "microSD Read Error!!\n" );    /* 読み込みエラー */
102 :        while( 1 ); /* 終了 */
103 :    } else {
104 :        printf( "microSD Read Time = %ldms\n", l );
105 :    }
106 :
107 :    printf( "Program End...\n\n" );
108 :
109 :    while( 1 );
110 : }
111 :
112 : /*****
113 : /* R8C/38A スペシャルファンクションレジスタ(SFR)の初期化 */
114 : /*****
115 : void init( void )
116 : {
117 :     int i;
118 :
119 :     /* クロックをXINクロック(20MHz)に変更 */
120 :     prc0 = 1;          /* プロテクト解除 */
121 :     cm13 = 1;         /* P4_6, P4_7をXIN-XOUT端子にする */
122 :     cm05 = 0;         /* XINクロック発振 */
123 :     for(i=0; i<50; i++ ); /* 安定するまで少し待つ(約10ms) */
124 :     ocd2 = 0;         /* システムクロックをXINにする */
125 :     prc0 = 0;         /* プロテクトON */
126 :
127 :     /* ポートの入出力設定 */
128 :     prc2 = 1;          /* PD0のプロテクト解除 */
129 :     pd0 = 0x00;        /*
130 :     pd1 = 0xd0;        /* 5:RXD0 4:TXD0 3-0:DIP SW */
131 :     pd2 = 0xff;        /* 7-0:LED */
132 :     pd3 = 0xff;        /*
133 :     p4 = 0x20;         /* P4_5のLED:初期は点灯 */
134 :     pd4 = 0xb8;        /* 7:XOUT 6:XIN 5:LED 2:VREF */
135 :     pd5 = 0x7f;        /* 7-0:LCD/microSD基板 */
136 :     pd6 = 0xef;        /* 4-0:LCD/microSD基板 */
137 :     pd7 = 0xff;        /*
138 :     pd8 = 0xff;        /*
139 :     pd9 = 0x3f;        /*
140 :     pur0 = 0x04;       /* P1_3~P1_0のプルアップON */
141 :
142 :     /* タイマRBの設定 */
143 :     /* 割り込み周期 = 1 / 20[MHz] * (TRBPRE+1) * (TRBPR+1)
144 :     = 1 / (20*10-6) * 200 * 100
145 :     = 0.001[s] = 1[ms]
146 :
147 :     trbmr = 0x00;      /* 動作モード、分周比設定 */
148 :     trbpre = 200-1;    /* プリスケアラレジスタ */
149 :     trbpr = 100-1;     /* プライマリレジスタ */
150 :     trbic = 0x07;      /* 割り込み優先レベル設定 */
151 :     trbcr = 0x01;      /* カウント開始 */
152 : }
153 :
154 : /*****
155 : /* タイマRB 割り込み処理 */
156 : /*****
157 : #pragma interrupt intTRB(vect=24)
158 : void intTRB( void )
159 : {
160 :     cnt1++;
161 : }
162 :
163 : /*****
164 : /* end of file */
165 : /*****

```

## 6.5 プログラムの解説

### 6.5.1 ヘッダファイルの取り込み

```

14 : /*=====*/
15 : /* インクルード */
16 : /*=====*/
17 : #include <stdio.h>
18 : #include "sfr_r838a.h" /* R8C/38A SFRの定義ファイル */
19 : #include "printf_lib.h" /* printf使用ライブラリ */
20 : #include "microsd_lib.h" /* microSD制御ライブラリ */
    
```

17 行目	「stdio.h」は、標準ライブラリと呼ばれるファイルの一つで、ルネサス統合開発環境(コンパイラ側)で用意されているヘッダファイルです。今回は、「printf」関数を使用するためにインクルードしています。
19 行目	「printf_lib.h」は、「printf_lib.c」を使用するときに、インクルードしなければいけないヘッダファイルです。
20 行目	「microsd_lib.h」は、「microsd_lib.c」を使用するときに、インクルードしなければいけないヘッダファイルです。

### 6.5.2 変数

```

31 : /*=====*/
32 : /* グローバル変数の宣言 */
33 : /*=====*/
34 : unsigned long cnt1; /* 時間計測用 */
35 :
36 : /* microSD関連変数 */
37 : signed char msdBuff[ 512 ]; /* 一時保存バッファ */
    
```

msdBuff 配列変数は、microSD に書き込むデータや読み込んだデータを格納する配列変数です。microSD からデータの読み込み、書き込みは512バイト単位で行います。そのため、512バイト以上確保してください。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。

### 6.5.3 ポートの入出力設定

```

127 : /* ポートの入出力設定 */
128 : prc2 = 1; /* PD0のプロテクト解除 */
129 : pd0 = 0x00; /* */
130 : pd1 = 0xd0; /* 5:RXD0 4:TXD0 3-0:DIP SW */
131 : pd2 = 0xff; /* 7-0:LED */
132 : pd3 = 0xff; /* */
133 : p4 = 0x20; /* P4_5のLED:初期は点灯 */
134 : pd4 = 0xb8; /* 7:XOUT 6:XIN 5:LED 2:VREF */
135 : pd5 = 0x7f; /* 7-0:LCD/microSD基板 */
136 : pd6 = 0xef; /* 4-0:LCD/microSD基板 */
137 : pd7 = 0xff; /* */
138 : pd8 = 0xff; /* */
139 : pd9 = 0x3f; /* */
140 : pur0 = 0x04; /* P1_3~P1_0のプルアップON */
    
```

※135 行目の PD5 については、「液晶・microSD 基板 kit12\_38a プログラム解説マニュアル液晶編(R8C/38A 版)」を参照してください。

液晶・microSD 基板の microSD 部は、RY\_R8C38 ボードのポート 6 を使用しています。ポート 6 のポート表を下  
 表に示します。

番号	ポート	信号名	マイコンから見た 入出力方向
1	+5V	-	-
2	P6_7	-	出力
3	P6_6	-	出力
4	P6_5	-	出力
5	P6_4	DAT0 (RXD1)	入力
6	P6_3	CMD (TXD1)	出力
7	P6_2	CLK (CLK1)	出力
8	P6_1	CD	出力
9	P6_0	モニタ LED	出力
10	GND	-	-

microSD 部を使用する場合の PD6 の入出力設定を下表に示します。

ビット	7	6	5	4	3	2	1	0
ポート 6 の 入出力設定	出力	出力	出力	入力	出力	出力	出力	出力

入力は“0”、出力は“1”を設定します。初期設定値を 16 進数に直すと、1110 1111 → 0xef となります。

#### 6.5.4 microSD の初期化

```

53 :      /* microSD 初期化 */
54 :      cnt1 = 0;
55 :      ret = initMicroSD();
56 :      l = cnt1;
57 :      if( ret != 0x00 ) {
58 :          printf( "microSD Initialize Error!!\n" ); /* 初期化エラー */
59 :          while( 1 ); /* 終了 */
60 :      } else {
61 :          printf( "microSD Initialize Time = %ldms\n", l );
62 :      }
    
```

initMicroSD 関数は、microSD を初期化する関数です。今回は 55 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、RY\_R8C38 ボードと液晶・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。

関数の実行時間の算出方法は次のように行います。

```

54 :      cnt1 = 0; cnt1変数を0にクリア
55 :      初期化
56 :      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
61 :          printf( "microSD Initialize Time = %ldms\n", l ); 結果表示
    
```

54 行目	cnt1 変数をクリアします。
56 行目	55 行目の関数実行後、cnt1 変数の値を l(エル)変数に保存します。(cnt1 変数は、タイマ RB 割り 込みの中で 1ms ごとにカウントアップします)。l変数には、初期化にかかった時間がミリ秒単位で格 納されます。
61 行目	l変数の値を表示し、初期化にかかった時間を確認することができます。

### 6.5.5 microSD のイレーズ(0 クリア)

```
64 :      /* microSDイレーズ */
65 :      cnt1 = 0;
66 :      ret = eraseMicroSD( 0x0000, 0x5dc00-1 );
                        ↑           ↑
                        開始アドレス 終了アドレス
67 :      l = cnt1;
68 :      if( ret != 0x00 ) {
69 :          printf( "microSD Erase Error!!\n" );    /* イレーズエラー      */
70 :          while( 1 ); /* 終了 */
71 :      } else {
72 :          printf( "microSD Erase Time = %ldms\n", l );
73 :      }
```

eraseMicroSD 関数は、microSD をイレーズする関数です。今回は 66 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、正常にイレーズができたということです。0 以外ならイレーズできていません。

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、イレーズ時間になります。

eraseMicroSD 関数の引数は、イレーズ開始アドレスとイレーズ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。

- イレーズ開始アドレス … 512(0x200)の倍数
- イレーズ終了アドレス … 512(0x200)の倍数-1
- ただし、イレーズ開始アドレス < イレーズ終了アドレス

### 6.5.6 microSD へデータ書き込み

```
80 :      /* microSD書き込み */
81 :      cnt1 = 0;
82 :      ret = writeMicroSD( 0x0000, msdBuff );
                        ↑           ↑
                        書き込み開始アドレス 書き込む512バイトのデータを格納している配列名
83 :      l = cnt1;
84 :      if( ret != 0x00 ) {
85 :          printf( "microSD Write Error!!\n" );    /* 書き込みエラー      */
86 :          while( 1 ); /* 終了 */
87 :      } else {
88 :          printf( "microSD Write Time = %ldms\n", l );
89 :      }
```

writeMicroSD 関数は、microSD へデータを書き込む関数です。今回は 82 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 …… signed char 型で 512 バイト以上の配列

### 6.5.7 microSD からデータ読み込み

```
96 :      /* microSD読み込み */
97 :      cnt1 = 0;
98 :      ret = readMicroSD( 0x0000 , msdBuff );
                        ↑           ↑
                読み込むアドレス 読み込む512バイトのデータを格納する配列名
99 :      l = cnt1;
100 :      if( ret != 0x00 ) {
101 :          printf( "microSD Read Error!!\n" );      /* 読み込みエラー      */
102 :          while( 1 ); /* 終了 */
103 :      } else {
104 :          printf( "microSD Read Time = %ldms\n", l );
105 :      }
```

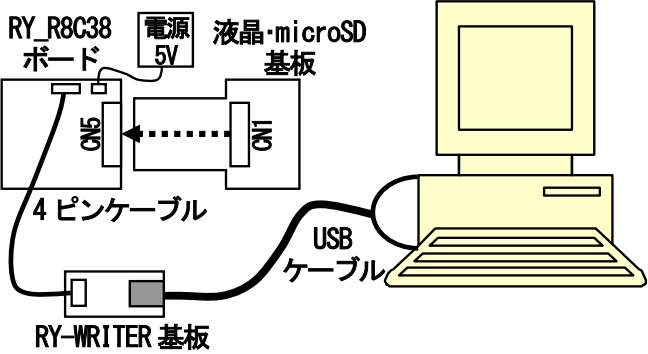



readMicroSD 関数は、microSD からデータを読み込む関数です。今回は 98 行目で実行しています。**読み込むデータ数は 512 バイト固定です**。ret 変数に関数を実行した結果が格納されます。0 なら、正常に読み込みができたということです。0 以外なら読み込まれていません。

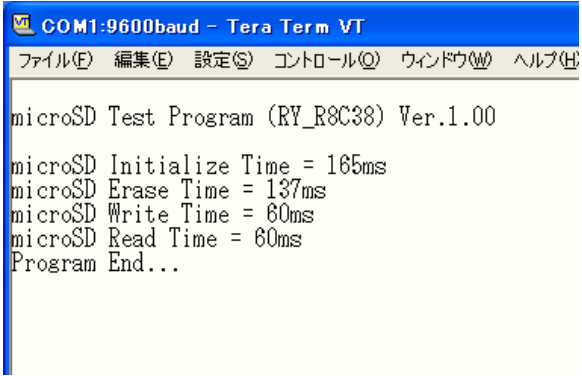

readMicroSD 関数を実行する前に cnt1 変数をクリアして、実行後に cnt1 の値を l 変数に保存します。l 変数の値が、読み込み時間になります。

readMicroSD 関数の引数は、読み込み開始アドレスと読み込むデータを格納する配列を代入します。引数は次のように設定してください。

- 読み込み開始アドレス ..... 512(0x200)の倍数
- データを格納する配列 ..... signed char 型で 512 バイト以上の配列

## 6.6 実行時間の測定方法

1		<p>プロジェクト「msd01_38a」をビルドして、「msd01_38a.mot」ファイルを RY_R8C38 ボードに書き込んでください。書き込みができたなら、RY_R8C38 ボードの電源を切って、書き込みスイッチを FWE の逆側にしておきます。</p> <p>RY_R8C38 ボードとパソコン間の USB ケーブル、RY-WRITER 基板、4 ピンケーブルは繋いだままにしておきます。</p>
2		<p>Tera Term を立ち上げます。          ※Tera Term をインストールしていない場合は、「マイコン実習マニュアル R8C/38A 版」のプロジェクト「uart0」にある「Tera Term のインストール」を参照してインストールしてください。</p>
3		<p>①: 「シリアルポート」を選択します。ポート番号は、R8C Writer で選択している番号と同じ番号にします。RY-WRITER 基板の場合は、「Prolific USB-to-Serial Com Port」と表示されている番号です。</p> <p>②: <b>OK</b> をクリックします。</p>
4		<p>立ち上がりました。</p>

5	 <pre>COM1:9600baud - Tera Term VT ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)  microSD Test Program (RY_R8C38) Ver.1.00 microSD Initialize Time = 165ms microSD Erase Time = 137ms microSD Write Time = 60ms microSD Read Time = 60ms Program End...</pre>	<p>RY_R8C38 ボードの電源を入れます。 RY_R8C38 ボードからデータが送られてきます。 Tera Term 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>
6	 <pre>COM1:9600baud - Tera Term VT ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)  microSD Test Program (RY_R8C38) Ver.1.00 microSD Initialize Error!!</pre>	<p>「microSD Initialize Error!!」と表示された場合は、microSD が挿入されていないか、液晶・microSD 基板と RY_R8C38 ボードが正しく接続されているか確認してください。</p>



## 6.7 演習

TeraTerm の画面に表示される内容は、次のとおりです。

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

※ ??には、実際にかかった時間が入ります。

下表にしたがって、「msd01\_38a.c」内のmicroSDを制御する関数の引数を変更して、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(ただし、2回目はリセットボタンを押して再実行するだけです)。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。

演習で使った microSDのメーカー		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	55	initMicroSD();		
	66	eraseMicroSD( 0x0000 , 0x5dc00-1 );		
	82	writeMicroSD( 0x0000 , msdBuff );		
	98	readMicroSD( 0x0000 , msdBuff );		
修正1回目	55	initMicroSD(); // この行は修正無し		
	66	eraseMicroSD( 0x0000 , <b>0x1000-1</b> );		
	82	writeMicroSD( <b>0x10000</b> , msdBuff );		
	98	readMicroSD( <b>0x10000</b> , msdBuff );		
修正2回目	55	initMicroSD(); // この行は修正無し		
	66	eraseMicroSD( 0x0000 , <b>0x800000-1</b> );		
	82	writeMicroSD( <b>0x7000000</b> , msdBuff );		
	98	readMicroSD( <b>0x7000000</b> , msdBuff );		

## 6.8 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。

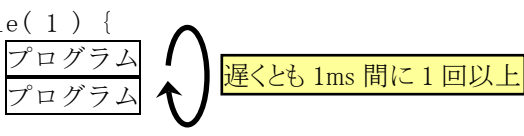
関数名	いつ使用するか (いつ実行するか)	許容できる関数の実行時間	実際の 実行時間	マイコンカーの 制御で 使用可能か?
initMicroSD	マイコンカーの電源を入れたとき(走行直前)に実行する。	走行前なので、イニシャライズが終わるまで待てるが、長すぎると人間がいらいらする。実用的に、1 秒程度なら待てる。	最大 1 秒以内	可能
eraseMicroSD	走行直前(マイコンカーの電源を入れたとき)に実行する。	走行前なので、イレーズが終わるまで待てるが、長すぎると人間がいらいらする。実用的に、1 秒程度なら待てる。	最大 1 秒以内	可能
writeMicroSD	<b>走行中に実行する。</b>	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 $4\mu s$ 、sensor_inp 関数は $4.6\mu s$ 、motor 関数は約 $45\mu s$ 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 $500\mu$ 以内でないと実用に耐えない。	30~100ms	<b>使用不可</b>
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

結果、**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 で走行しているとき writeMicroSD 関数の処理に 30ms 間かかったとすると、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 へ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、正確な制御ができません。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 intTRB(vect=24) void intTRB( 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 以内に終わらなければいけない。できれば、<math>500\mu s</math> 以下で終わらせる。</p> </div>
---	---

## 7. プロジェクト「msd02\_38a」 microSD にデータ記録

### 7.1 概要

このプログラムは、

- ポート 0 に接続されているディップスイッチの値
- RY\_R8C38 ボードのディップスイッチの値

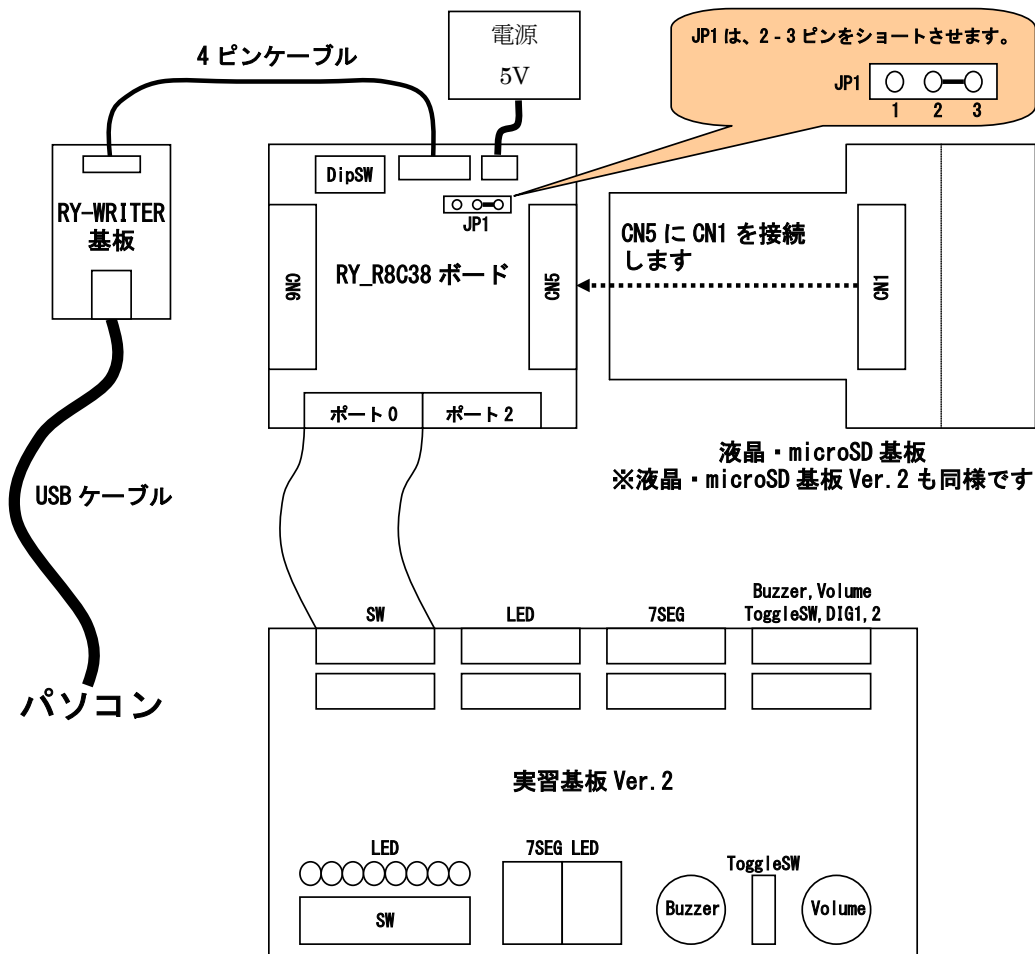
を、10ms ごとに内蔵 RAM に一時的に記録します。512 バイトになったら、microSD に書き込みます。microSD に書き込み処理を行っていても 10ms ごとの記録は続けます。

記録終了後、RY-WRITER 基板を通して記録した情報をパソコンへ出力します。

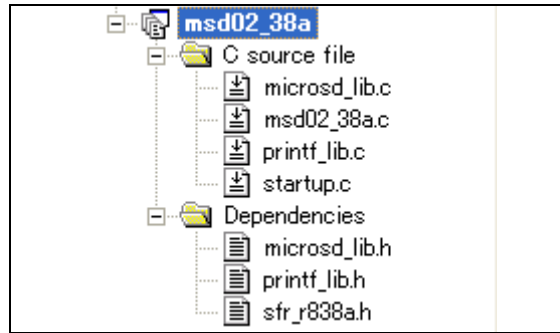
ここでは、writeMicroSD 関数を使用しない書き込み方法を説明します。

### 7.2 接続

- RY\_R8C38 ボードの CN5(ポート 3、ポート 5、ポート 6)と、液晶・microSD 基板の CN1 のコネクタを重ね合わせて接続します。
- RY\_R8C38 ボードのポート 0 と、実習基板 Ver.2 のスイッチ部分をフラットケーブルで接続します。  
**※ポート 0 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。**
- RY\_R8C38 ボードとパソコン間を RY-WRITER 基板、USB ケーブル、4 ピンケーブルで接続します。



### 7.3 プロジェクトの構成



	ファイル名	内容
1	microsd_lib.c	microSD 制御ライブラリです。microSD を使用する場合は、このファイルを追加します。 ファイルの位置→C:\¥WorkSpace¥common_r8c38a¥microsd_lib.c
2	msd02_38a.c	実際に制御するプログラムが書かれています。R8C/38A 内蔵周辺機能(SFR)の初期化も行います。 ファイルの位置→C:\¥WorkSpace¥kit12msd_38a¥msd02_38a¥msd02_38a.c
3	printf_lib.c	通信をするための設定、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。 ファイルの位置→C:\¥WorkSpace¥common_r8c38a¥printf_lib.c
4	startup.c	固定割り込みベクタアドレスの設定、スタートアッププログラム、RAM の初期化(初期値のないグローバル変数、初期値のあるグローバル変数の設定)などを行います。 ファイルの位置→C:\¥WorkSpace¥kit12msd_38a¥msd02_38a¥startup.c
5	microsd_lib.h	microSD 制御ライブラリのヘッダファイルです。 ファイルの位置→C:\¥WorkSpace¥common_r8c38a¥microsd_lib.h
6	printf_lib.h	printf、scanf 制御ライブラリのヘッダファイルです。 ファイルの位置→C:\¥WorkSpace¥common_r8c38a¥printf_lib.h
7	sfr_r838a.h	R8C/38A マイコンの内蔵周辺機能を制御するためのレジスタ(Special Function Register)を定義したファイルです。 ファイルの位置→C:\¥WorkSpace¥common_r8c38a¥sfr_r838a.h

### 7.4 プログラム

プログラムのゴシック体部分が、間欠処理をできるように改良した部分です。

```

1 : /*****
2 : /* 対象マイコン R8C/38A */
3 : /* ファイル内容 microSD基板の実験 */
4 : /* バージョン Ver. 1.00 */
5 : /* Date 2011.04.01 */
6 : /* Copyright ジャパンマイコンカーラー実行委員会 */
7 : /*****
8 :
9 : /*
10 : 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。
11 : ・ポート0のデータ
12 : ・マイコンボード上のディップスイッチの値
13 : 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。
14 : */
15 :
```

```

16 : /*=====*/
17 : /* インクルード */
18 : /*=====*/
19 : #include <stdio.h>
20 : #include "sfr_r838a.h" /* R8C/38A SFRの定義ファイル */
21 : #include "printf_lib.h" /* printf使用ライブラリ */
22 : #include "microsd_lib.h" /* microSD制御ライブラリ */
23 :
24 : /*=====*/
25 : /* シンボル定義 */
26 : /*=====*/
27 :
28 : /*=====*/
29 : /* プロトタイプ宣言 */
30 : /*=====*/
31 : void init( void );
32 : unsigned char dipsw_get( void );
33 : unsigned long convertBCD_CharToLong( unsigned char hex );
34 :
35 : /*=====*/
36 : /* グローバル変数の宣言 */
37 : /*=====*/
38 : unsigned long cnt1; /* 時間計測用 */
39 : int pattern; /* パターン番号 */
40 : int countDown; /* 表示作業用 */
41 :
42 : /* microSD関連変数 */
43 : signed char msdBuff[ 512 ]; /* 一時保存バッファ */
44 : int msdBuffAddress; /* 一時記録バッファ書込アドレス */
45 : int msdFlag; /* 1:データ記録 0:記録しない */
46 : int msdTimer; /* 取得間隔計算用 */
47 : unsigned long msdStartAddress; /* 記録開始アドレス */
48 : unsigned long msdEndAddress; /* 記録終了アドレス */
49 : unsigned long msdWorkAddress; /* 作業用アドレス */
50 :
51 : /*=====*/
52 : /* メインプログラム */
53 : /*=====*/
54 : void main( void )
55 : {
56 :     int i, ret;
57 :
58 :     init(); /* SFRの初期化 */
59 :     init_uart0_printf( SPEED_9600 ); /* UART0とprintf関連の初期化 */
60 :     setMicroSDLedPort( &p6, &pd6, 0 ); /* microSD モニタLED設定 */
61 :     asm( " fset I " ); /* 全体の割り込み許可 */
62 :
63 :     // microSD 書き込み開始アドレス
64 :     // 512の倍数に設定する
65 :     msdStartAddress = 0;
66 :
67 :     // microSD 書き込み終了アドレス
68 :     // 書き込みしたい時間[ms] : x = 10[ms] : 64バイト(保存バイト数)
69 :     // 500msなら、x = 500 * 64 / 10 = 3200
70 :     // 結果は512の倍数になるように繰り上げる。よって、32256にする。
71 :     msdEndAddress = 32256;
72 :     msdEndAddress += msdStartAddress; /* スタート分足す */
73 :
74 :     /* microSD初期化 */
75 :     ret = initMicroSD();
76 :     if( ret != 0x00 ) {
77 :         printf( "\nmicroSD Initialize Error!!\n" ); /* 初期化できず */
78 :         pattern = 99;
79 :     } else {
80 :         printf( "\nmicroSD Initialize OK!!\n" ); /* 初期化完了 */
81 :         printf( "Ready " );
82 :     }
83 :
84 :     while( 1 ) {
85 :
86 :         switch( pattern ) {
87 :         case 0:
88 :             /* カウントダウン表示 */
89 :             if( cnt1 / 1000 != countDown ) {
90 :                 countDown = cnt1 / 1000;
91 :                 printf( "%d ", 4 - countDown );
92 :                 if( cnt1 / 1000 == 4 ) { /* 4秒たったら開始 */
93 :                     pattern = 1;
94 :                 }
95 :             }
96 :             break;
97 :
98 :         case 1:
99 :             /* microSDクリア */
100 :            ret = eraseMicroSD( msdStartAddress, msdEndAddress-1 );
101 :            if( ret != 0x00 ) {
102 :                printf( "\nmicroSD Erase Error!!\n" ); /* エラー */
103 :                pattern = 99;
104 :                break;
105 :            }

```

## 7. プロジェクト「msd02\_38a」 microSD にデータ記録

```

106 :      /* microSDProcess開始処理 */
107 :      ret = microSDProcessStart( msdStartAddress );
108 :      if( ret != 0x00 ) {
109 :          printf( "¥nmicroSD microSDProcess Error!!¥n" ); /* エラー */
110 :          pattern = 99;
111 :          break;
112 :      }
113 :      printf( "¥n" );
114 :      printf( "Data recording " );
115 :      msdBuffAddress = 0;
116 :      msdWorkAddress = msdStartAddress;
117 :      msdFlag = 1;          /* データ記録開始          */
118 :      pattern = 2;
119 :      cnt1 = 0;
120 :      break;
121 :
122 : case 2:
123 :     /* データ記録中 記録は割り込みの中で行う */
124 :     /* 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 */
125 :     if( msdFlag == 0 ) {
126 :         pattern = 3;
127 :         break;
128 :     }
129 :
130 :     /* 時間表示 */
131 :     if( cnt1 / 1000 != countDown ) {
132 :         countDown = cnt1 / 1000;
133 :         printf( "%d ", countDown );
134 :     }
135 :     break;
136 :
137 : case 3:
138 :     /* 最後のデータが書き込まれるまで待つ*/
139 :     if( checkMicroSDProcess() == 11 ) {
140 :         microSDProcessEnd(); /* microSDProcess終了処理 */
141 :         pattern = 4;
142 :     }
143 :     break;
144 :
145 : case 4:
146 :     /* 終了処理が終わるまで待つ*/
147 :     if( checkMicroSDProcess() == 0 ) {
148 :         pattern = 5;
149 :     }
150 :     break;
151 :
152 : case 5:
153 :     /* タイトル転送、準備 */
154 :     printf( "¥n¥n" );
155 :     printf( "msd_02 Data Out¥n" );
156 :     printf( "Time,P0 Data,DIP SW Data¥n" );
157 :
158 :     msdWorkAddress = msdStartAddress; /* 読み込み開始アドレス */
159 :     i = 0;
160 :     pattern = 6;
161 :     break;
162 :
163 : case 6:
164 :     /* microSDよりデータ読み込み */
165 :     if( msdWorkAddress >= msdEndAddress ) {
166 :         /* 書き込み終了アドレスになったら、終わり */
167 :         printf( "End.¥n" );
168 :         pattern = 99;
169 :         break;
170 :     }
171 :     ret = readMicroSD( msdWorkAddress , msdBuff );
172 :     if( ret != 0x00 ) {
173 :         /* 読み込みエラー */
174 :         printf( "¥nmicroSD Read Error!!¥n" );
175 :         pattern = 99;
176 :         break;
177 :     } else {
178 :         /* エラーなし */
179 :         msdWorkAddress += 512; /* microSDのアドレスを+512する */
180 :         msdBuffAddress = 0; /* 配列からの読み込み位置を0に */
181 :         pattern = 7;
182 :     }
183 :     break;
184 :
185 : case 7:
186 :     /* データ転送 */
187 :     printf( "%4d,¥%08ld¥", 0x%02x¥n",
188 :         i,
189 :         convertBCD_CharToLong( msdBuff[msdBuffAddress+0] ),
190 :         msdBuff[msdBuffAddress+1]
191 :     );
192 :

```

```

193 :         i += 10;
194 :         msdBuffAddress += 64;
195 :
196 :         if( msdBuffAddress >= 512 ) {
197 :             pattern = 6;
198 :         }
199 :         break;
200 :
201 :     case 99:
202 :         /* 終了 */
203 :         break;
204 :
205 :     default:
206 :         /* どれも無い場合は待機状態に戻す */
207 :         pattern = 0;
208 :         break;
209 :     }
210 : }
211 : }
212 :
213 : /*****
214 : /* R8C/38A スペシャルファンクションレジスタ(SFR)の初期化
215 : /*****
216 : void init( void )
217 : {
218 :     int i;
219 :
220 :     /* クロックをXINクロック(20MHz)に変更 */
221 :     prc0 = 1; /* プロテクト解除 */
222 :     cm13 = 1; /* P4_6, P4_7をXIN-XOUT端子にする */
223 :     cm05 = 0; /* XINクロック発振 */
224 :     for(i=0; i<50; i++ ); /* 安定するまで少し待つ(約10ms) */
225 :     ocd2 = 0; /* システムクロックをXINにする */
226 :     prc0 = 0; /* プロテクトON */
227 :
228 :     /* ポートの入出力設定 */
229 :     prc2 = 1; /* PD0のプロテクト解除 */
230 :     pd0 = 0x00; /*
231 :     pd1 = 0xd0; /* 5:RXD0 4:TXD0 3-0:DIP SW
232 :     pd2 = 0xff; /* 7-0:LED
233 :     pd3 = 0xff; /*
234 :     p4 = 0x20; /* P4_5のLED:初期は点灯
235 :     pd4 = 0xb8; /* 7:XOUT 6:XIN 5:LED 2:VREF
236 :     pd5 = 0x7f; /* 7-0:LCD/microSD基板
237 :     pd6 = 0xef; /* 4-0:LCD/microSD基板
238 :     pd7 = 0xff; /*
239 :     pd8 = 0xff; /*
240 :     pd9 = 0x3f; /*
241 :     pur0 = 0x04; /* P1_3~P1_0のプルアップON
242 :
243 :     /* タイマRBの設定 */
244 :     /* 割り込み周期 = 1 / 20[MHz] * (TRBPRE+1) * (TRBPR+1)
245 :     = 1 / (20*10^-6) * 200 * 100
246 :     = 0.001[s] = 1[ms]
247 :
248 :     trbmr = 0x00; /* 動作モード、分周比設定
249 :     trbpre = 200-1; /* プリスケアラレジスタ
250 :     trbpr = 100-1; /* プライマリレジスタ
251 :     trbic = 0x07; /* 割り込み優先レベル設定
252 :     trbcr = 0x01; /* カウント開始
253 : }
254 :
255 : /*****
256 : /* タイマRB 割り込み処理
257 : /*****
258 : #pragma interrupt intTRB(vect=24)
259 : void intTRB( void )
260 : {
261 :     signed char *p;
262 :
263 :     cnt1++;
264 :
265 :     /* microSD間欠書き込み処理(1msごとに実行) */
266 :     microSDProcess();
267 :
268 :     /* microSD記録処理 */
269 :     if( msdFlag == 1 ) {
270 :         /* 記録間隔のチェック */
271 :         msdTimer++;
272 :         if( msdTimer >= 10 ) {
273 :             msdTimer = 0;
274 :             p = msdBuff + msdBuffAddress;
275 :
276 :             /* RAMに記録 ここから */
277 :             *p++ = p0;
278 :             *p++ = dipsw_get();
279 :             /* RAMに記録 ここまで */
280 :
281 :             msdBuffAddress += 64; /* RAMの記録アドレスを次へ
282 :
    
```

## 7. プロジェクト「msd02\_38a」 microSD にデータ記録

```

283 :         if ( msdBuffAddress >= 512 ) {
284 :             /* 512個になったら、microSDに記録する */
285 :             msdBuffAddress = 0;
286 :             setMicroSDdata( msdBuff );
287 :             msdWorkAddress += 512;
288 :             if ( msdWorkAddress >= msdEndAddress ) {
289 :                 /* 記録処理終了 */
290 :                 msdFlag = 0;
291 :             }
292 :         }
293 :     }
294 : }
295 :
296 :
297 : /*****
298 : /* ディップスイッチ値読み込み
299 : /* 戻り値 スイッチ値 0~15
300 : /*****
301 : unsigned char dipsw_get( void )
302 : {
303 :     unsigned char sw;
304 :
305 :     sw = pl & 0x0f;          /* P1_3~P1_0読み込み */
306 :
307 :     return sw;
308 : }
309 :
310 : /*****
311 : /* char型データの値をlong型変数に2進数で変換
312 : /* 引数 unsigned char 変換元の8bitデータ
313 : /* 戻り値 unsigned long 変換先の変数(0~11111111) ※0か1しかありません
314 : /*****
315 : unsigned long convertBCD_CharToLong( unsigned char hex )
316 : {
317 :     int i;
318 :     unsigned long l = 0;
319 :
320 :     for( i=0; i<8; i++ ) {
321 :         l *= 10;
322 :         if( hex & 0x80 ) l += 1;
323 :         hex <<= 1;
324 :     }
325 :
326 :     return l;
327 : }
328 :
329 : /*****
330 : /* end of file
331 : /*****

```

## 7.5 setMicroSDdata 関数と microSDProcess 関数

## 7.5.1 概要

マイコンカーは、常に次の作業を行っています。

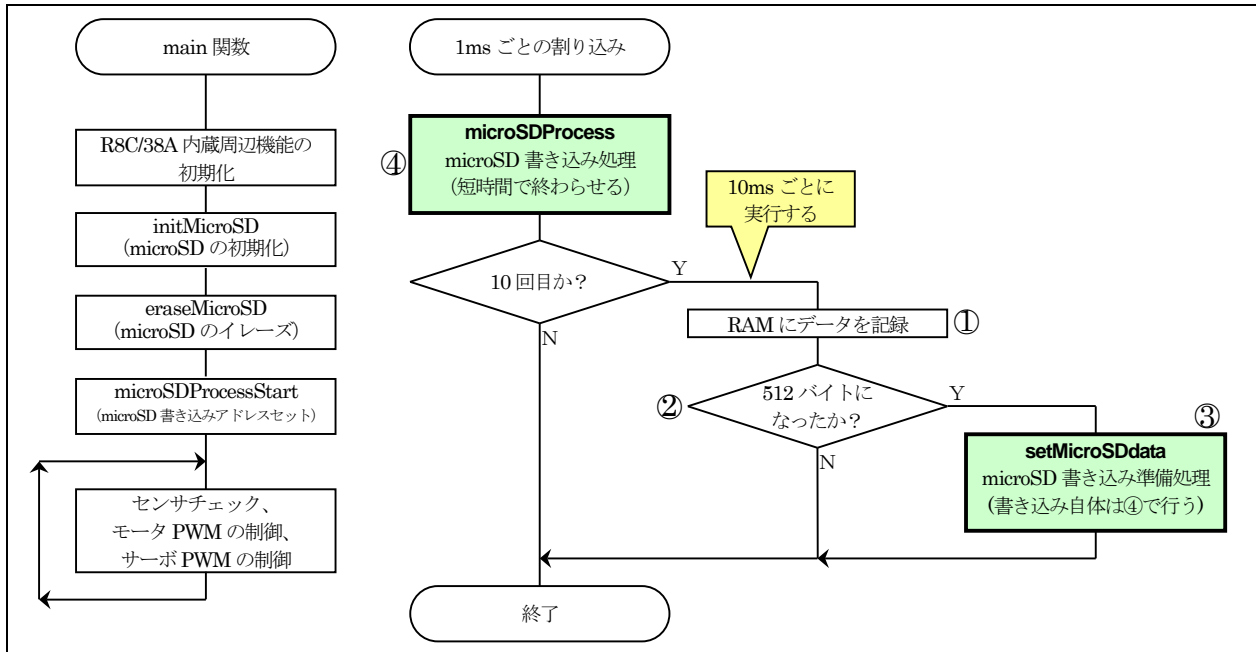
- 各種センサ(コースセンサ、ロータリエンコーダ、上り坂検出スイッチなど)の読み込み
- 駆動モータの制御
- サーボ(ステアリングモータ)の制御

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限り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。



### 7.5.2 プログラムの流れ



①	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 たったと判断して、RAM に現在の状態を記録します。
②	RAM に記録したデータが 512 バイトかどうかチェックします。
③	512 バイトになったなら <b>setMicroSDdata 関数</b> で RAM のデータを microSD に書き込む準備をします。あくまで準備だけで書き込み作業は次で説明する <b>microSDProcess 関数</b> で行います。
④	<b>setMicroSDdata 関数</b> で書き込み準備をしたら、 <b>microSDProcess 関数</b> で実際の書き込み処理を行います。 <b>microSDProcess 関数</b> 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 <b>writeMicroSD 関数</b> と同じことを行います。

### 7.5.3 各関数の処理内容

各関数の処理内容を下表に示します。

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。
setMicroSDdata 関数	microSD に書き込む準備だけを行います。実行時間は、約 100 $\mu$ s です。
microSDProcess 関数	setMicroSDdata 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 100 $\mu$ s です。約 50~60 回実行すると 512 バイト書き込むことができます。ちなみに書き込むデータがないときは、約 5 $\mu$ s で処理を終わらせます。

※時間はすべて実測です。

## 7.6 プログラムの解説

### 7.6.1 プロトタイプ宣言

```

28 : /*=====*/
29 : /* プロトタイプ宣言 */
30 : /*=====*/
31 : void init( void );
32 : unsigned char dipsw_get( void );
33 : unsigned long convertBCD_CharToLong( unsigned char hex );
    
```

33 行目で、convertBCD\_CharToLong 関数を宣言しています。この関数の内容を下記に示します。

#### ■convertBCD\_CharToLong 関数

書式	unsigned long convertBCD_CharToLong( unsigned char hex )
内容	符号なし char 型データを 2 進数 ("0"or"1") に変換します。変換後の型は、unsigned long 型です。 例えば、printf 関数を使用して P0 から読み込んだセンサ情報を 2 進数で表示したいとき、printf 関数で 2 進数を表示することができません。そのため、センサの "1", "0" 情報を文字列、又は 10 進数に変換して表示する必要があります。この関数は、符号なし char 型データを 0~11111111 (2 進数) に変換する関数です。
引数	unsigned char 符号なし char 型の 8 ビットデータ
戻り値	2 進数(0~11111111)の値を unsigned long 型で返します。 ※0 か 1 しかありません。
使用例	/* ポート 0 の値を表示します */ printf("%08ld", convertBCD_CharToLong( p0 ));  P0 に 0x5a が格納されていた場合は、「01011010」が出力されます。

### 7.6.2 変数

```

35 : /*=====*/
36 : /* グローバル変数の宣言 */
37 : /*=====*/
38 : unsigned long cnt1; /* 時間計測用 */
39 : int pattern; /* パターン番号 */
40 : int countDown; /* 表示作業用 */
41 :
42 : /* microSD関連変数 */
43 : signed char msdBuff[ 512 ]; /* 一時保存バッファ */
44 : int msdBuffAddress; /* 一時記録バッファ書込アドレス */
45 : int msdFlag; /* 1:データ記録 0:記録しない */
46 : int msdTimer; /* 取得間隔計算用 */
47 : unsigned long msdStartAddress; /* 記録開始アドレス */
48 : unsigned long msdEndAddress; /* 記録終了アドレス */
49 : 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	内容
countDown	記録を開始するまでのカウントダウン、または記録中にカウントアップして何秒たったか printf 文で時間を知らせるためのタイマ用変数です。
msdBuff[ 512 ]	microSD に書き込むデータや読み込んだデータを格納する配列変数です。
msdBuffAddress	msdBuff 配列変数にデータを書き込んだり、読み込んだりする位置を指定します。
msdFlag	1 ならデータを記録します。0 なら記録しません。
<b>msdTimer</b> ※	タイマ RB の割り込み関数内で 1ms ごとにカウントアップするタイマ用変数です。microSD の記録間隔は 10ms 間隔ですので、msdTimer 変数が 10 になったなら、記録処理を実行するようにします。
<b>msdStartAddress</b> ※	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。
<b>msdEndAddress</b> ※	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	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。

※の変数の値は、制御する内容によって変更します。

### 7.6.3 main 関数(初期化)

```

51 : /*****
52 : /* メインプログラム */
53 : /*****
54 : void main( void )
55 : {
56 :     int    i, ret;
57 :
58 :     init();                /* SFRの初期化 */
59 :     init_uart0_printf( SPEED_9600 ); /* UART0とprintf関連の初期化 */
60 :     setMicroSDLedPort( &p6, &pd6, 0 ); /* microSD モニタLED設定 */
61 :     asm(" fset I ");      /* 全体の割り込み許可 */
62 :
63 :     // microSD 書き込み開始アドレス
64 :     // 512の倍数に設定する
65 :     msdStartAddress = 0;
66 :
67 :     // microSD 書き込み終了アドレス
68 :     // 書き込みしたい時間[ms] : x = 10[ms] : 64バイト(保存バイト数)
69 :     // 5000msなら、x = 5000 * 64 / 10 = 32000
70 :     // 結果は512の倍数になるように繰り上げる。よって、32256にする。
    
```

```

71 :      msdEndAddress = 32256;
72 :      msdEndAddress += msdStartAddress; /* スタート分足す */
73 :
74 :      /* microSD初期化 */
75 :      ret = initMicroSD();
76 :      if( ret != 0x00 ) {
77 :          printf( "¥nmicroSD Initialize Error!!¥n" ); /* 初期化できず */
78 :          pattern = 99;
79 :      } else {
80 :          printf( "¥nmicroSD Initialize OK!!¥n" ); /* 初期化完了 */
81 :          printf( "Ready " );
82 :      }
    
```

59 行目	printf 関数、scanf 関数の初期化と UART0(通信)を設定します。今回、通信速度は 9600bps にすることとします。引数は「SPEED_9600」を設定します。
60 行目	液晶・microSD 基板のモニタ LED のポートを設定します。
65 行目	microSD の書き込み開始アドレスを設定します。
71 行目	<p>microSD に書き込む容量を指定します。今回の記録条件は下記のようにしました。</p> <ul style="list-style-type: none"> <li>・データ記録の間隔 … 10ms ごと</li> <li>・データ記録数 …………… 64 バイト</li> <li>・データ記録時間 ……… 5 秒</li> </ul> <p>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。</p> $\text{容量} = \text{記録したい時間[ms]} \div \text{記録する間隔[ms]} \times 1 \text{ 回に記録するバイト数}$ <p>よって、容量は次のとおりです。</p> $= 5,000 \div 10 \times 64$ $= 32,000$ <p>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。</p> $32,000 \div 512 = 62 \text{ 余り } 256$ <p>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、</p> $63 \times 512 = 32,256$ <p>となります。</p> <p>「 <b>msdEndAddress = 32256</b> 」と設定します。</p>
72 行目	書き込む容量に、書き込み開始アドレスを加えて終了アドレスを設定します。

#### 7.6.4 パターン 0:スタート

```

84 :   while( 1 ) {
85 :
86 :     switch( pattern ) {
87 :     case 0:
88 :       /* カウントダウン表示 */
89 :       if( cnt1 / 1000 != countDown ) {
90 :         countDown = cnt1 / 1000;
91 :         printf( "%d ", 4 - countDown );
92 :         if( cnt1 / 1000 == 4 ) { /* 4 秒たったら開始          */
93 :           pattern = 1;
94 :         }
95 :       }
96 :       break;
    
```

リセット後、データの記録をすぐに始めてしまうと、記録開始直後はディップスイッチなどの値が変更できないので、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。

#### 7.6.5 パターン 1: microSD クリア、書き込みアドレスセット

```

98 :   case 1:
99 :     /* microSD クリア */
100 :     ret = eraseMicroSD( msdStartAddress, msdEndAddress-1 );
101 :     if( ret != 0x00 ) {
102 :       printf( "%nmicroSD Erase Error!!%n" ); /* エラー          */
103 :       pattern = 99;
104 :       break;
105 :     }
106 :     /* microSDProcess 開始処理 */
107 :     ret = microSDProcessStart( msdStartAddress );
108 :     if( ret != 0x00 ) {
109 :       printf( "%nmicroSD microSDProcess Error!!%n" ); /* エラー */
110 :       pattern = 99;
111 :       break;
112 :     }
113 :     printf( "%n" );
114 :     printf( "Data recording " );
115 :     msdBuffAddress = 0;
116 :     msdWorkAddress = msdStartAddress;
117 :     msdFlag = 1; /* データ記録開始          */
118 :     pattern = 2;
119 :     cnt1 = 0;
120 :     break;
    
```

100 行目	microSD の開始アドレスから終了アドレスまでをイレーズします。今回は、0~32255 番地をイレーズします。
107 行目	microSD へ書き込み開始アドレスをセットします。今回は、0 番地から開始します。
115 行目	msdBuff 配列(RAM)を参照する変数を 0 にクリアしています。
116 行目	microSD の作業アドレスを開始アドレスに設定します。実際の書き込み開始アドレスは、107 行目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。
117 行目	msdFlag 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。

### 7.6.6 パターン 2: データ記録中

```

122 :     case 2:
123 :         /* データ記録中 記録は割り込みの中で行う */
124 :         /* 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 */
125 :         if( msdFlag == 0 ) {
126 :             pattern = 3;
127 :             break;
128 :         }
129 :
130 :         /* 時間表示 */
131 :         if( cnt1 / 1000 != countDown ) {
132 :             countDown = cnt1 / 1000;
133 :             printf( "%d ", countDown );
134 :         }
135 :         break;
    
```

125 行目	msdFlag 変数が 0 かどうかチェックします。書き込み終了アドレスまで書き込みが終わると、タイマ RB 割り込み関数内で msdFlag 変数が 0 になります。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。
131 行目 ~ 134 行目	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。

### 7.6.7 パターン 3: 最後のデータが書き込まれるまで待つ

```

137 :     case 3:
138 :         /* 最後のデータが書き込まれるまで待つ*/
139 :         if( checkMicroSDProcess() == 11 ) {
140 :             microSDProcessEnd();          /* microSDProcess終了処理      */
141 :             pattern = 4;
142 :         }
143 :         break;
    
```

139 行目	msdFlag 変数が 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。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。戻り値が 11 なら書き込み終了と判断します。
140 行目	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。

### 7.6.8 パターン 4: 終了処理が終わるまで待つ

```

145 :     case 4:
146 :         /* 終了処理が終わるまで待つ*/
147 :         if( checkMicroSDProcess() == 0 ) {
148 :             pattern = 5;
149 :         }
150 :         break;
    
```

147 行目	140 行目で実行した microSDProcessEnd 関数の処理が終わるまで待ちます。checkMicroSDProcess 関数を実行して microSDProcessEnd 関数の処理が終わったかチェックします。戻り値が 0 なら処理が終わったと判断し、パターン 5 へ移ります。
--------	---

### 7.6.9 パターン 5:タイトル転送、準備

```

152 :     case 5:
153 :         /* タイトル転送、準備 */
154 :         printf( "¥n¥n" );
155 :         printf( "msd_02 Data Out¥n" );
156 :         printf( "Time,P0 Data,DIP SW Data¥n" );
157 :
158 :         msdWorkAddress = msdStartAddress; /* 読み込み開始アドレス */
159 :         i = 0;
160 :         pattern = 6;
161 :         break;
    
```

158 行目	転送する準備を行います。microSD から読み込むアドレスをセットします。セット後、パターン 6 に移ります。
--------	--

### 7.6.10 パターン 6:microSD よりデータ読み込み

```

163 :     case 6:
164 :         /* microSDよりデータ読み込み */
165 :         if( msdWorkAddress >= msdEndAddress ) {
166 :             /* 書き込み終了アドレスになったら、終わり */
167 :             printf( "End.¥n" );
168 :             pattern = 99;
169 :             break;
170 :         }
171 :         ret = readMicroSD( msdWorkAddress , msdBuff );
172 :         if( ret != 0x00 ) {
173 :             /* 読み込みエラー */
174 :             printf( "¥nmicroSD Read Error!!¥n" );
175 :             pattern = 99;
176 :             break;
177 :         } else {
178 :             /* エラーなし */
179 :             msdWorkAddress += 512; /* microSDのアドレスを+512する */
180 :             msdBuffAddress = 0; /* 配列からの読み込み位置を0に */
181 :             pattern = 7;
182 :         }
183 :         break;
    
```

165 行目	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。大きくなったら読み込むデータがないと判断し、終了します。パターン 99 へ移ります。
171 行目	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。

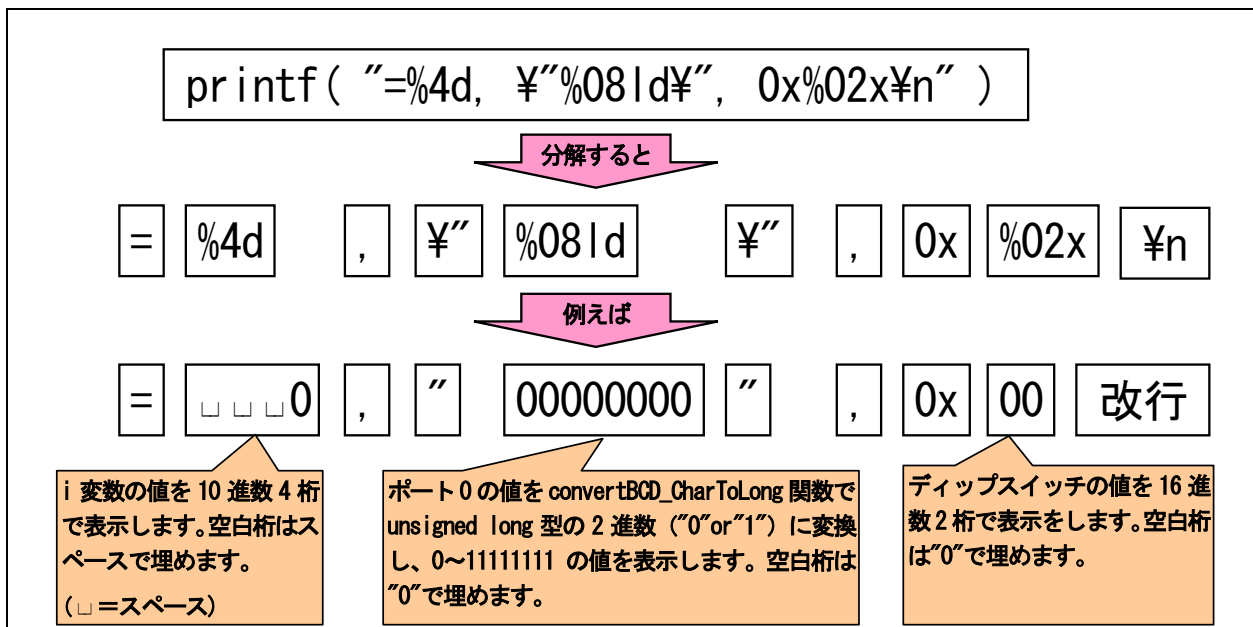
7.6.11 パターン 7: パソコンへデータ転送

```

185 :     case 7:
186 :         /* データ転送 */
187 :         printf( "%4d, ¥"%08ld¥", 0x%02x¥n",
188 :             i,
189 :             convertBCD_CharToLong( msdBuff[msdBuffAddress+0] ),
190 :             msdBuff[msdBuffAddress+1]
191 :         );
192 :
193 :         i += 10;
194 :         msdBuffAddress += 64;      1回の記録数を変えたいときは、ここも変える
195 :
196 :         if( msdBuffAddress >= 512 ) {
197 :             pattern = 6;
198 :         }
199 :         break;
    
```

187 行目 ～ 191 行目	記録したデータが msdBuff 配列変数に格納されています。msdBuff 配列変数からデータを読み込み、printf 文を使用してパソコンへ出力します。
194 行目	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(アドレス)をセットします。
196 行目	msdBuffAddress 変数が 512 以上なら、msdBuff 配列変数からデータをすべて読み込み、パソコンへ出力したので、パターン 6 に戻って、microSD から次の 512 バイト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。





### 7.6.12 パターン 99: 終了

```
201 :     case 99:
202 :         /* 終了 */
203 :         break;
```

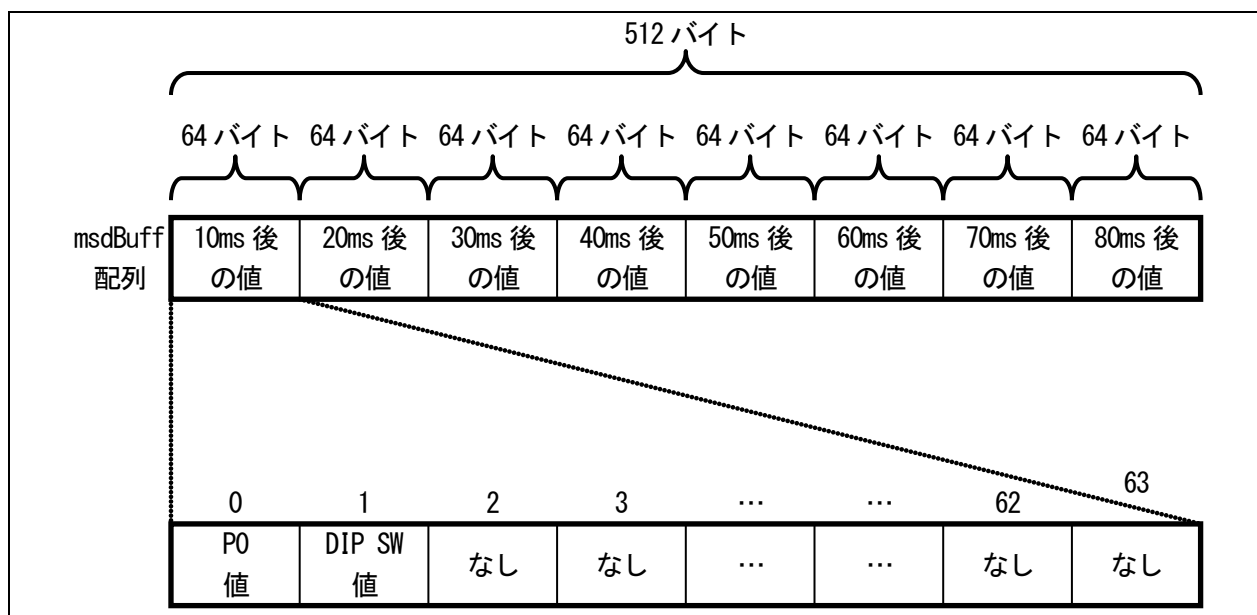
何もありません。データ転送を終えた状態です。

### 7.6.13 割り込み処理

```
255 :  /*****
256 :  /* タイマRB 割り込み処理
257 :  *****/
258 :  #pragma interrupt intTRB(vect=24)
259 :  void intTRB( void )
260 :  {
261 :      signed char *p;
262 :
263 :      cnt1++;
264 :
265 :      /* microSD間欠書き込み処理(1msごとに実行) */
266 :      microSDProcess();
267 :
268 :      /* microSD記録処理 */
269 :      if( msdFlag == 1 ) {
270 :          /* 記録間隔のチェック */
271 :          msdTimer++;
272 :          if( msdTimer >= 10 ) {
273 :              msdTimer = 0;
274 :              p = msdBuff + msdBuffAddress;
275 :
276 :              /* RAMに記録 ここから */
277 :              *p++ = p0;
278 :              *p++ = dipsw_get();
279 :              /* RAMに記録 ここまで */
280 :
281 :              msdBuffAddress += 64; /* RAMの記録アドレスを次へ */
282 :
283 :              if( msdBuffAddress >= 512 ) {
284 :                  /* 512個になったら、microSDに記録する */
285 :                  msdBuffAddress = 0;
286 :                  setMicroSDdata( msdBuff );
287 :                  msdWorkAddress += 512;
288 :                  if( msdWorkAddress >= msdEndAddress ) {
289 :                      /* 記録処理終了 */
290 :                      msdFlag = 0;
291 :                  }
292 :              }
293 :          }
294 :      }
295 :  }
```

266 行目	microSDProcess 関数を 1ms ごとに実行します。
269 行目	msdFlag 変数が 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。
272 行目	記録間隔のチェックをしています。今回は、msdTimer 変数が 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。
277 行目 ～ 278 行目	msdBuff 配列変数に記録している内容です。今回はポート 0 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 つのみを記録しています。
281 行目	msdBuff 配列変数の番地を次に記録する番地を設定します。今回は、64 バイトごとに記録をしていますので、msdBuffAddress 変数に 64 を足します。
283 行目 ～ 287 行目	msdBuff 配列変数の記録データが 512 バイトになったら setMicroSDdata 関数で microSD へ書き込む準備を行います。 287 行目で msdWorkAddress 変数に 512 を足して、microSD に記録した番地を保存しておきます。
288 行目 ～ 291 行目	msdWorkAddress 変数が書き込み終了アドレスより大きくなったかチェックします。大きくなったなら、書き込み終了アドレスまでデータを記録したと判断し、msdFlag 変数を 0 にして、記録処理を終了します。

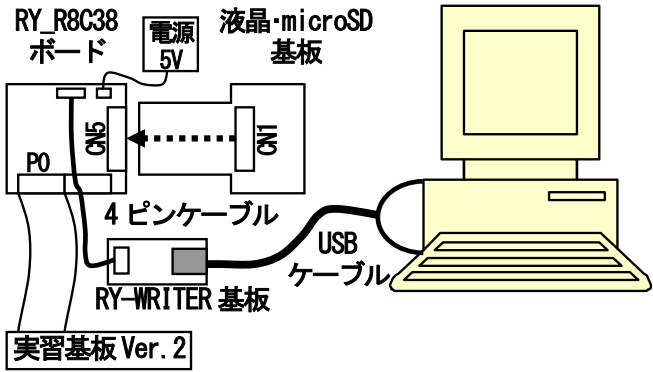
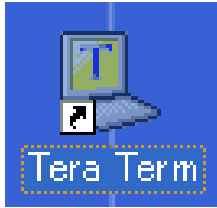


記録イメージを下図に示します。

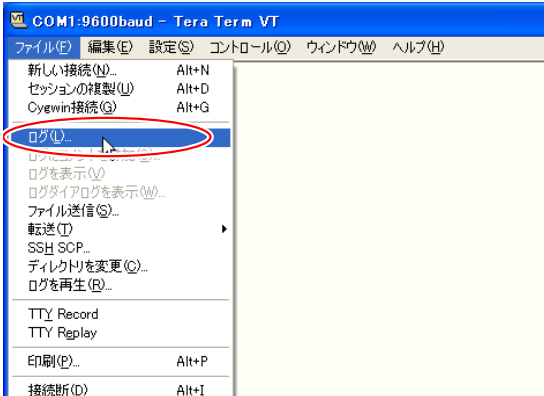
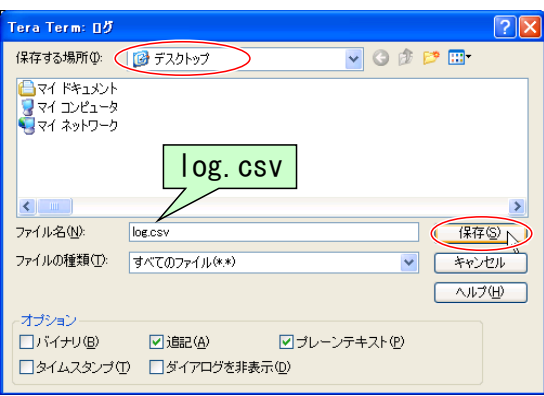
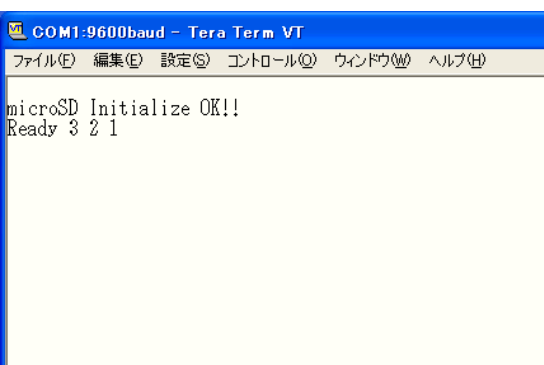
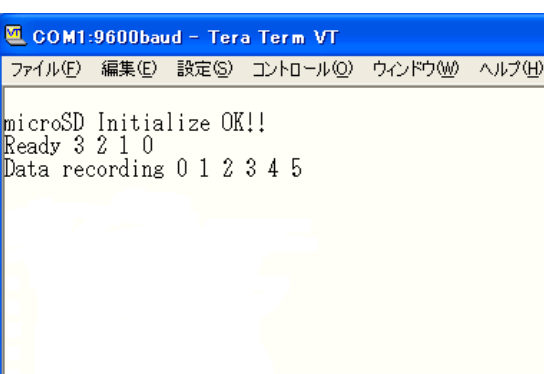


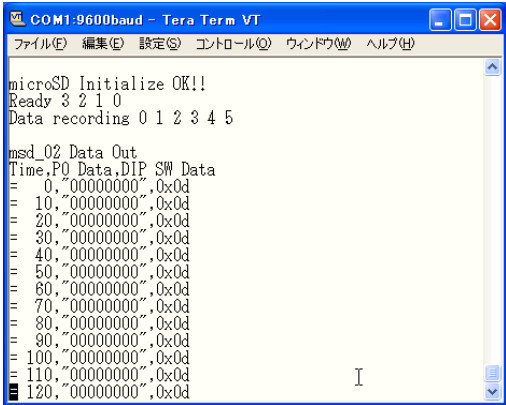
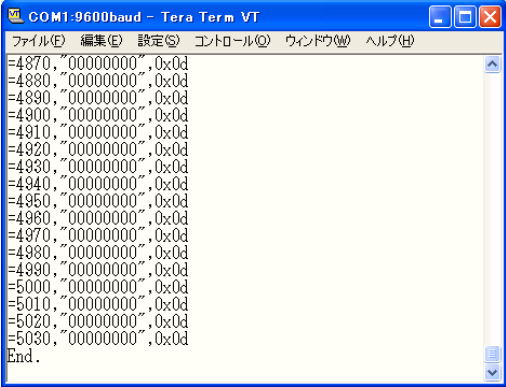
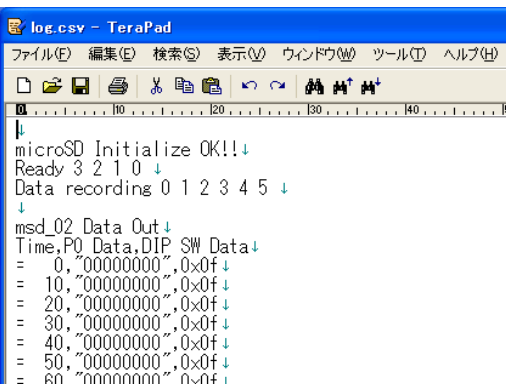
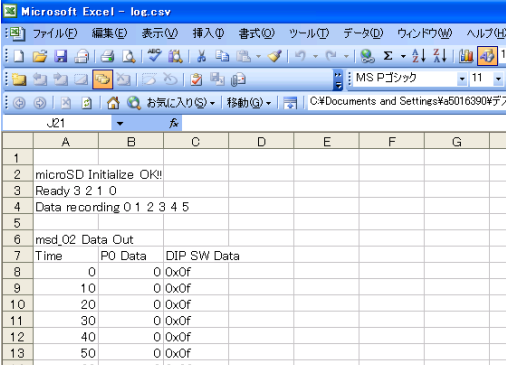
**ポイントは、80ms 間隔で 512 バイト記録できるということです。**この値を基本として時間を細かく区切り、記録するバイト数を減らせば、細かい間隔で記録することができます。代表的な記録時間、記録数を下表に示します。

記録間隔	記録数	備考
80ms	512 バイト	
40ms	256 バイト	
20ms	128 バイト	
<b>10ms</b>	<b>64 バイト</b>	今回の記録間隔、記録数です。
5ms	32 バイト	
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。

## 7.7 データの取り込み方

1		<p>プロジェクト「msd02_38a」をビルドして、「msd02_38a.mot」ファイルを RY_R8C38 ボードに書き込んでください。書き込みができたなら、RY_R8C38 ボードの電源を切って、書き込みスイッチを FWE の逆側にしておきます。</p> <p>RY_R8C38 ボードとパソコン間の USB ケーブル、RY_WRITER 基板、4 ピンケーブルは繋いだままにしておきます。</p>
2		<p>Tera Term を立ち上げます。</p> <p>※Tera Term をインストールしていない場合は、「マイコン実習マニュアル R8C/38A 版」のプロジェクト「uart0」にある「Tera Term のインストール」を参照してインストールしてください。</p>
3		<p>①: 「シリアルポート」を選択します。ポート番号は、R8C Writer で選択している番号と同じ番号にします。RY-WRITER 基板の場合は、「Prolific USB-to-Serial Com Port」と表示されている番号です。</p> <p>②: <b>OK</b> をクリックします。</p>
4		<p>立ち上がりました。</p>

5		<p>受信データをファイルに保存します。「ファイル→ログ」を選択します。</p>
6		<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、「保存」をクリックします。  <b>※拡張子は必ず「csv」にします。</b></p>
7		<p>RY_R8C38 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>
8		<p><b>Data recording 0</b>          から  <b>Data recording 0 1 2 3 4 5</b>          と表示されている間の 5 秒間、ポート 0 に繋がっている実習基板のディップスイッチの値と、RY_R8C38 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 RY_R8C38 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>

9		5 秒経つと、自動的に記録したデータが表示されます。データが止まるまで待ちます。
10		データの表示が止まったら転送終了です。RY_R8C38 ボードの電源を切って、Tera Term を終了してください。
11		Tera Term のログ操作で保存したファイルをエディタで開いてみました。「log.csv」を開きます。
12		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。

**※注意**

**TeraTerm は、受信前に「ファイル→ログ」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、「ファイル→ログ」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。

## 7.8 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。□部分はそれぞれデータの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、cnt1 変数を例にします。

```
l = cnt1;
*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。□部分はそれぞれデータの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 );
```

## 7.9 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、「5. printf、scanf 制御ライブラリ」を参照してください。

## 7.10 演習の回答例

### (1) 記録時間の変更

「7.6.3 main 関数(初期化)」の式に当てはめると、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。

71 :       msdEndAddress = <b>64000</b> ;
---

### (2) 記録間隔の変更

記録間隔を変更するとき、変更する行と数値を下表のようにします。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	193 行の 変更	194 行の 変更	196 行の 変更	272 行の 変更	281 行の 変更	283 行の 変更
80ms	80	512	512	80	512	512
40ms	40	256	512	40	256	512
20ms	20	128	512	20	128	512
10ms(変更前)	10	64	512	10	64	512
<b>5ms (今回の回答)</b>	<b>5</b>	<b>32</b>	<b>512</b>	<b>5</b>	<b>32</b>	<b>512</b>
2ms	2	12	480	2	12	480
1ms	1	6	480	1	6	480

(3) 通信速度の変更

通信速度は、「init\_uart0\_printf」関数を使用して設定します。設定できる通信速度を下表に示します。

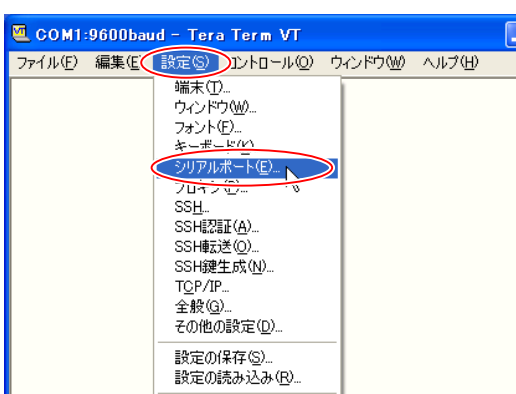
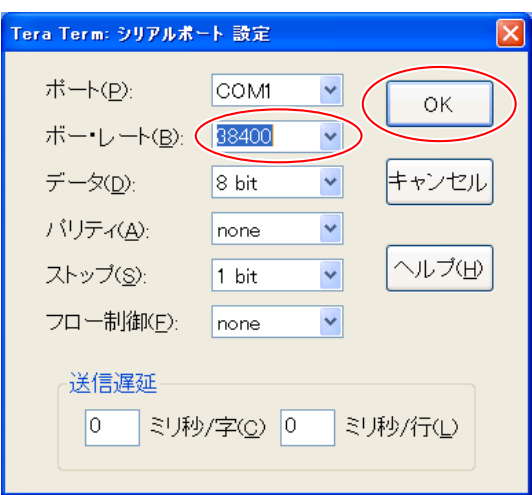
番号	ボー・レート	シンボル定義
1	4800bps	SPEED_4800
2	9600bps	SPEED_9600
3	19200bps	SPEED_19200
4	38400bps	SPEED_38400

※シンボル定義は、「printf\_lib.h」で行っています。

今回は、38400bps に設定しますので、59 行目の init\_uart0\_printf 関数の引数部分に、「**SPEED\_38400**」と設定します。

```
59 :      init_uart0_printf( SPEED_38400 );      /* UART0とprintf関連の初期化      */
```

また、TeraTerm 側も、通信速度を変更する必要があります。

1		<p>「設定」→「シリアルポート」をクリックし、「シリアルポート設定」を開きます。</p>
2		<p>上から2つ目の「ボー・レート」を「38400」に設定し、<b>OK</b>をクリックします。</p> <p>設定ができれば、RY_R8C38 ボードの電源を入れて、試してみてください。</p>



## 8. プロジェクト「kit12msd01\_38a」 走行データを microSD に記録

### 8.1 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

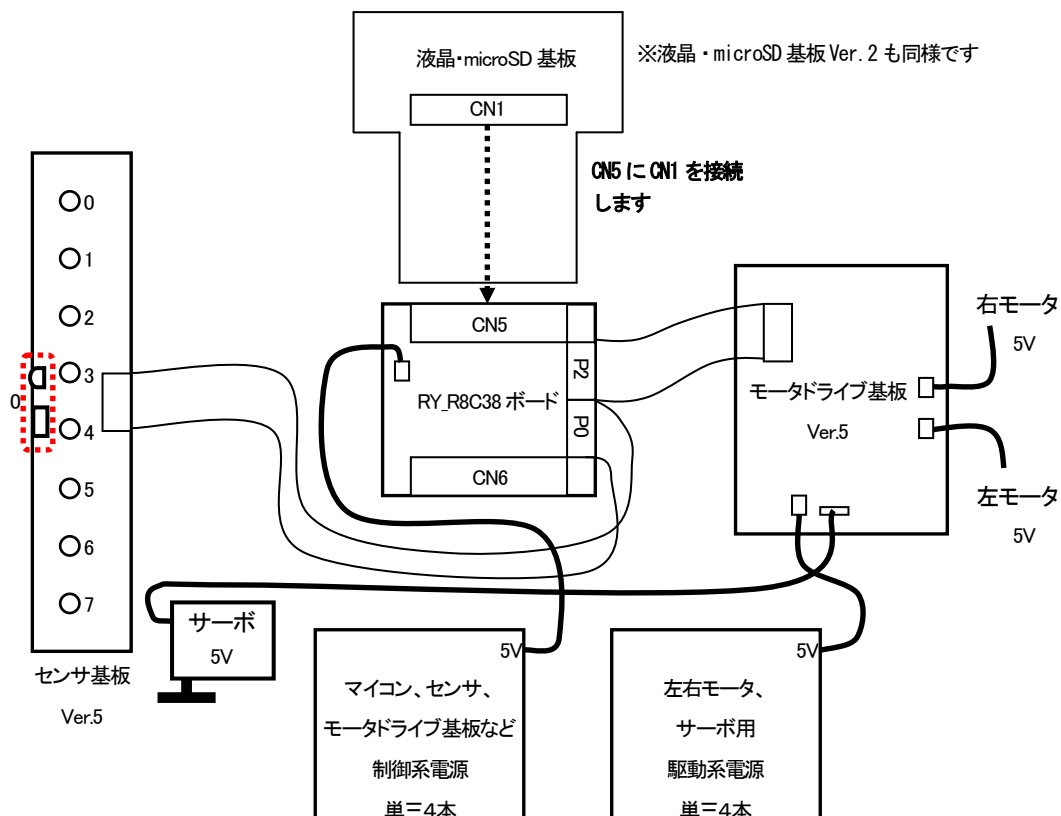
これらのデータを、走行開始から 10ms ごとに 60 秒間記録します。60 秒間経った場合は、データの記録は止めますが、走行はそのまま行きます。

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。

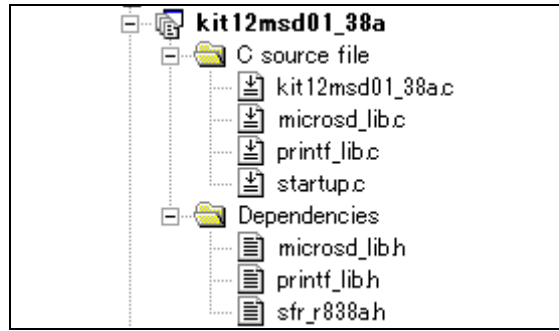
### 8.2 マイコンカーの構成

マイコンカーキット Ver.5.1 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。

- ・RY\_R8C38 ボードのポート 0 と、センサ基板 Ver.5 を接続します。
- ・RY\_R8C38 ボードのポート 2 と、モータドライブ基板 Ver.5 を接続します。
- ・RY\_R8C38 ボードの CN5 と液晶・microSD 基板の CN1 を接続します。



### 8.3 プロジェクトの構成



	ファイル名	内容
1	kit12msd01_38a.c	実際に制御するプログラムが書かれています。R8C/38A 内蔵周辺機能(SFR)の初期化も行います。 ファイルの位置→C:\¥Workspace¥kit12msd_38a¥kit12msd01_38a¥kit12msd01_38a.c
1	microsd_lib.c	microSD 制御ライブラリです。microSD を使用する場合は、このファイルを追加します。 ファイルの位置→C:\¥Workspace¥common_r8c38a¥microsd_lib.c
3	printf_lib.c	通信をするための設定、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。 ファイルの位置→C:\¥Workspace¥common_r8c38a¥printf_lib.c
4	startup.c	固定割り込みベクタアドレスの設定、スタートアッププログラム、RAM の初期化(初期値のないグローバル変数、初期値のあるグローバル変数の設定)などを行います。 ファイルの位置→C:\¥Workspace¥kit12msd_38a¥ki127msd01_38a¥startup.c
5	microsd_lib.h	microSD 制御ライブラリのヘッダファイルです。 ファイルの位置→C:\¥Workspace¥common_r8c38a¥microsd_lib.h
6	printf_lib.h	printf、scanf 制御ライブラリのヘッダファイルです。 ファイルの位置→C:\¥Workspace¥common_r8c38a¥printf_lib.h
7	sfr_r838a.h	R8C/38A マイコンの内蔵周辺機能を制御するためのレジスタ(Special Function Register)を定義したファイルです。 ファイルの位置→C:\¥Workspace¥common_r8c38a¥sfr_r838a.h

### 8.4 プログラム

プログラムのゴシック体部分が、「kit12\_38a.c」から microSD 制御、走行データを記録できるように追加した部分です。

```

1 : /*****
2 : /* 対象マイコン R8C/38A */
3 : /* ファイル内容  microSDを使ったマイコンカートレースプログラム(R8C/38A版) */
4 : /* バージョン  Ver. 1. 00 */
5 : /* Date 2013. 04. 24 */
6 : /* Copyright ジャパンマイコンカーラー実行委員会 */
7 : /*****
8 :
9 : /*
10 : 本プログラムは、「kit12_38a.c」にmicroSDによる走行データ保存、転送を
11 : 追加したプログラムです。次のデータを保存、転送することができます。
12 : ・パターン番号 ・センサの状態
13 : ・ハンドル角度 ・左モータPWM値 ・右モータPWM値
14 : */
15 :
```

```

16 : /*=====*/
17 : /* インクルード */
18 : /*=====*/
19 : #include <stdio.h>
20 : #include "sfr_r838a.h" /* R8C/38A SFRの定義ファイル */
21 : #include "printf_lib.h" /* printf使用ライブラリ */
22 : #include "microsd_lib.h" /* microSD制御ライブラリ */
23 :
24 : /*=====*/
25 : /* シンボル定義 */
26 : /*=====*/
27 : /* 定数設定 */
28 : #define PWM_CYCLE 39999 /* モータPWMの周期 */
29 : #define SERVO_CENTER 3750 /* サーボのセンタ値 */
30 : #define HANDLE_STEP 22 /* 1° 分の値 */
31 :
32 : /* マスク値設定 × : マスクあり(無効) ○ : マスク無し(有効) */
33 : #define MASK2_2 0x66 /* ×○○××○○× */
34 : #define MASK2_0 0x60 /* ×○○×××××× */
35 : #define MASK0_2 0x06 /* ×××××○○× */
36 : #define MASK3_3 0xe7 /* ○○○×○○○ */
37 : #define MASK0_3 0x07 /* ××××○○○ */
38 : #define MASK3_0 0xe0 /* ○○○××××× */
39 : #define MASK4_0 0xf0 /* ○○○○×××× */
40 : #define MASK0_4 0xf0 /* ××××○○○ */
41 : #define MASK4_4 0xff /* ○○○○○○○○ */
42 :
43 : /*=====*/
44 : /* プロトタイプ宣言 */
45 : /*=====*/
46 : void init( void );
47 : void timer( unsigned long timer_set );
48 : int check_crossline( void );
49 : int check_rightline( void );
50 : int check_leftline( void );
51 : unsigned char sensor_inp( unsigned char mask );
52 : unsigned char dipsw_get( void );
53 : unsigned char pushsw_get( void );
54 : unsigned char startbar_get( void );
55 : void led_out( unsigned char led );
56 : void motor( int accele_l, int accele_r );
57 : void handle( int angle );
58 : unsigned long convertBCD_CharToLong( unsigned char hex );
59 :
60 : /*=====*/
61 : /* グローバル変数の宣言 */
62 : /*=====*/
63 : unsigned long cnt0; /* timer関数用 */
64 : unsigned long cnt1; /* main内で使用 */
65 : int pattern; /* パターン番号 */
66 :
67 : /* microSD関連変数 */
68 : signed char msdBuff[ 512 ]; /* 一時保存バッファ */
69 : int msdBuffAddress; /* 一時記録バッファ書込アドレス */
70 : int msdFlag; /* 1:データ記録 0:記録しない */
71 : int msdTimer; /* 取得間隔計算用 */
72 : unsigned long msdStartAddress; /* 記録開始アドレス */
73 : unsigned long msdEndAddress; /* 記録終了アドレス */
74 : unsigned long msdWorkAddress; /* 作業用アドレス */
75 : int msdError; /* エラー番号記録 */
76 :
77 : /* 現在の状態保存用 */
78 : int handleBuff; /* 現在のハンドル角度記録 */
79 : int leftMotorBuff; /* 現在の左モータPWM値記録 */
80 : int rightMotorBuff; /* 現在の右モータPWM値記録 */
81 :
82 : /*=====*/
83 : /* メインプログラム */
84 : /*=====*/
85 : void main( void )
86 : {
87 :     int ret;
88 :
89 :     /* マイコン機能の初期化 */
90 :     init(); /* 初期化 */
91 :     init_uart0_printf( SPEED_9600 ); /* UART0とprintf関連の初期化 */
92 :     setMicroSDLedPort( &p6, &pd6, 0 ); /* microSD モニタLED設定 */
93 :     asm( " fset I " ); /* 全体の割り込み許可 */
94 :
95 :     // microSD 書き込み開始アドレス
96 :     // 512の倍数に設定する
97 :     msdStartAddress = 0;
98 :
99 :     // microSD 書き込み終了アドレス
100 :     // 書き込みしたい時間[ms] : x = 10[ms] : 64バイト
101 :     // 60000msなら、x = 60000 * 64 / 10 = 384000
102 :     // 結果は512の倍数になるように繰り上げる。
103 :     msdEndAddress = 384000;
104 :     msdEndAddress += msdStartAddress; /* スタート分足す */
105 :

```

## 8. プロジェクト「kit12msd01\_38a」 走行データを microSD に記録

```

106 :      /* microSD初期化 */
107 :      ret = initMicroSD();
108 :      if( ret != 0x00 ) {
109 :          msdError = 1;
110 :          /* 初期化できなければ3秒間、LEDの点灯方法を変える */
111 :          cnt1 = 0;
112 :          while( cnt1 < 3000 ) {
113 :              if( cnt1 % 200 < 100 ) {
114 :                  led_out( 0x3 );
115 :              } else {
116 :                  led_out( 0x0 );
117 :              }
118 :          }
119 :      }
120 :
121 :      /* 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 */
122 :      if( pushsw_get() ) {
123 :          pattern = 71;
124 :      }
125 :
126 :      /* マイコンカーの状態初期化 */
127 :      handle( 0 );
128 :      motor( 0, 0 );
129 :
130 :      while( 1 ) {
131 :          switch( pattern ) {
132 :
133 :          /******
134 :          パターンについて
135 :          0: スイッチ入力待ち
136 :          1: スタートバーが開いたかチェック
137 :          11: 通常トレース
138 :          12: 右へ大曲げの終わりのチェック
139 :          13: 左へ大曲げの終わりのチェック
140 :          21: クロスライン検出時の処理
141 :          22: クロスラインを読み飛ばす
142 :          23: クロスライン後のトレース、クランク検出
143 :          31: 左クランククリア処理 安定するまで少し待つ
144 :          32: 左クランククリア処理 曲げ終わりのチェック
145 :          41: 右クランククリア処理 安定するまで少し待つ
146 :          42: 右クランククリア処理 曲げ終わりのチェック
147 :          51: 右ハーフライン検出時の処理
148 :          52: 右ハーフラインを読み飛ばす
149 :          53: 右ハーフライン後のトレース、レーンチェンジ
150 :          54: 右レーンチェンジ終了のチェック
151 :          61: 左ハーフライン検出時の処理
152 :          62: 左ハーフラインを読み飛ばす
153 :          63: 左ハーフライン後のトレース、レーンチェンジ
154 :          64: 左レーンチェンジ終了のチェック
155 :          *****/
156 :
157 :          case 0:
158 :              /* スイッチ入力待ち */
159 :              if( pushsw_get() ) {
160 :                  ret = eraseMicroSD( msdStartAddress, msdEndAddress-1 );
161 :                  if( ret != 0x00 ) {
162 :                      /* イレースできず */
163 :                      msdError = 2;
164 :                  }
165 :                  /* microSDProcess開始処理 */
166 :                  ret = microSDProcessStart( msdStartAddress );
167 :                  if( ret != 0x00 ) {
168 :                      /* 開始処理できず */
169 :                      msdError = 3;
170 :                  }
171 :                  pattern = 1;
172 :                  cnt1 = 0;
173 :                  break;
174 :              }
175 :              if( cnt1 < 100 ) {                /* LED点滅処理 */
176 :                  led_out( 0x1 );
177 :              } else if( cnt1 < 200 ) {
178 :                  led_out( 0x2 );
179 :              } else {
180 :                  cnt1 = 0;
181 :              }
182 :              break;
183 :
184 :          case 1:
185 :              /* スタートバーが開いたかチェック */
186 :              if( !startbar_get() ) {
187 :                  /* スタート!! */
188 :                  led_out( 0x0 );
189 :                  pattern = 11;
190 :                  msdBuffAddress = 0;
191 :                  msdWorkAddress = msdStartAddress;
192 :                  msdFlag = 1;                /* データ記録開始 */
193 :                  cnt1 = 0;
194 :                  break;
195 :              }

```

```
196 :         if( cnt1 < 50 ) {                               /* LED点滅処理          */
197 :             led_out( 0x1 );
198 :         } else if( cnt1 < 100 ) {
199 :             led_out( 0x2 );
200 :         } else {
201 :             cnt1 = 0;
202 :         }
203 :         break;
```

中略

```
539 :     case 71:
540 :         /* 走行データ転送準備 */
541 :         handle( 0 );
542 :         motor( 0, 0 );
543 :         msdFlag = 0;
544 :         if( msdError != 0 ) {
545 :             /* microSDに不具合があったなら終了 */
546 :             printf( "microSD Initialize Error!!\n" );
547 :             pattern = 99;
548 :         } else {
549 :             pattern = 72;
550 :             cnt1 = 0;
551 :         }
552 :         break;
553 :
554 :     case 72:
555 :         /* 最後のデータ書き込みまで待つ*/
556 :         if( checkMicroSDProcess() == 0 ) {
557 :             pattern = 73;          /* データ転送処理へ          */
558 :             break;
559 :         }
560 :         if( checkMicroSDProcess() == 11 ) {
561 :             microSDProcessEnd(); /* microSDProcess終了処理 */
562 :             while( checkMicroSDProcess() );
563 :             pattern = 73;          /* データ転送処理へ          */
564 :         }
565 :         break;
566 :
567 :     case 73:
568 :         /* スイッチが離されたかチェック */
569 :         if( !pushsw_get() ) {
570 :             pattern = 74;
571 :             cnt1 = 0;
572 :         }
573 :         break;
574 :
575 :     case 74:
576 :         /* 0.2s待ち */
577 :         if( cnt1 > 200 ) {
578 :             pattern = 75;
579 :             cnt1 = 0;
580 :             break;
581 :         }
582 :         if( pushsw_get() ) {
583 :             pattern = 73;
584 :         }
585 :         break;
586 :
587 :     case 75:
588 :         /* スイッチが押されたかチェック */
589 :         led_out( (cnt1/500) % 2 + 1 );
590 :         if( pushsw_get() ) {
591 :             pattern = 76;
592 :             cnt1 = 0;
593 :         }
594 :         break;
595 :
596 :     case 76:
597 :         /* タイトル転送、準備 */
598 :         printf( "\n" );
599 :         printf( "Your Car Name Data Out\n" );
600 :         printf( "Pattern, Sensor, ハンドル, 左モータ, 右モータ\n" );
601 :
602 :         msdWorkAddress = msdStartAddress; /* 読み込み開始アドレス */
603 :         pattern = 77;
604 :         break;
605 :
```

```

606 :     case 77:
607 :         /* microSDよりデータ読み込み */
608 :         if( msdWorkAddress >= msdEndAddress ) {
609 :             /* 書き込み終了アドレスになったら、終わり */
610 :             pattern = 99;
611 :             break;
612 :         }
613 :         ret = readMicroSD( msdWorkAddress , msdBuff );
614 :         if( ret != 0x00 ) {
615 :             /* 読み込みエラー */
616 :             printf( "\nmicroSD Read Error!!\n" );
617 :             pattern = 99;
618 :             break;
619 :         } else {
620 :             /* エラーなし */
621 :             msdWorkAddress += 512;
622 :             msdBuffAddress = 0;
623 :             pattern = 78;
624 :         }
625 :         break;
626 :
627 :     case 78:
628 :         /* データ転送 */
629 :         led_out( cnt1/100 ) % 2 + 1 ); /* LED点滅処理 */
630 :
631 :         if( msdBuff[msdBuffAddress+0] == 0 ) {
632 :             /* パターンが0なら終了 */
633 :             printf( "End.\n" );
634 :             pattern = 99;
635 :             break;
636 :         }
637 :
638 :         printf( "%d,%08ld",
639 :             msdBuff[msdBuffAddress+0], /* パターン */
640 :             convertBCD_CharToLong( msdBuff[msdBuffAddress+1] ), /* センサ */
641 :             msdBuff[msdBuffAddress+2], /* ハンドル */
642 :             msdBuff[msdBuffAddress+3], /* 左モータ */
643 :             msdBuff[msdBuffAddress+4] /* 右モータ */
644 :         );
645 :         msdBuffAddress += 64;
646 :
647 :         if( msdBuffAddress >= 512 ) {
648 :             pattern = 77;
649 :         }
650 :         break;
651 :
652 :     case 99:
653 :         /* 転送終了 */
654 :         led_out( 0x3 );
655 :         break;
656 :
657 :     default:
658 :         /* どれも無い場合は待機状態に戻す */
659 :         pattern = 0;
660 :         break;
661 :     }
662 : }
663 :
664 :
665 : /******
666 : /* R8C/38A スペシャルファンクションレジスタ(SFR)の初期化 */
667 : /******
668 : void init( void )
669 : {
670 :     int i;
671 :
672 :     /* クロックをXINクロック(20MHz)に変更 */
673 :     prc0 = 1; /* プロテクト解除 */
674 :     cm13 = 1; /* P4_6,P4_7をXIN-XOUT端子にする */
675 :     cm05 = 0; /* XINクロック発振 */
676 :     for(i=0; i<50; i++ ); /* 安定するまで少し待つ(約10ms) */
677 :     ocd2 = 0; /* システムクロックをXINにする */
678 :     prc0 = 0; /* プロテクトON */
679 :
680 :     /* ポートの入出力設定 */
681 :     prc2 = 1; /* PD0のプロテクト解除 */
682 :     pd0 = 0x00; /* 7-0:センサ基板Ver. 5 */
683 :     pd1 = 0xd0; /* 5:RXD0 4:TXD0 3-0:DIP SW */
684 :     p2 = 0xc0;
685 :     pd2 = 0xfe; /* 7-0:モータドライブ基板Ver. 5 */
686 :     pd3 = 0xff; /* */
687 :     p4 = 0x20; /* P4_5のLED:初期は点灯 */
688 :     pd4 = 0xb8; /* 7:XOUT 6:XIN 5:LED 2:VREF */
689 :     pd5 = 0x7f; /* 7-0:LCD/microSD基板 */
690 :     pd6 = 0xef; /* 4-0:LCD/microSD基板 */
691 :     pd7 = 0xff; /* */
692 :     pd8 = 0xff; /* */
693 :     pd9 = 0x3f; /* */
694 :     pur0 = 0x04; /* P1_3~P1_0のプルアップON */
695 :

```

```

696 : /* タイマRBの設定 */
697 : /* 割り込み周期 = 1 / 20[MHz] * (TRBPRE+1) * (TRBPR+1)
698 : = 1 / (20*10^6) * 200 * 100
699 : = 0.001[s] = 1[ms]
700 : */
701 : trbmr = 0x00; /* 動作モード、分周比設定 */
702 : trbpre = 200-1; /* プリスケーラレジスタ */
703 : trbpr = 100-1; /* プライマリレジスタ */
704 : trbic = 0x07; /* 割り込み優先レベル設定 */
705 : trbcr = 0x01; /* カウント開始 */
706 :
707 : /* タイマRD リセット同期PWMモードの設定*/
708 : /* PWM周期 = 1 / 20[MHz] * カウントソース * (TRDGRA0+1)
709 : = 1 / (20*10^6) * 8 * 40000
710 : = 0.016[s] = 16[ms]
711 : */
712 : trdpsr0 = 0x08; /* TRDIOB0, C0, D0端子設定 */
713 : trdpsr1 = 0x05; /* TRDIOA1, B1, C1, D1端子設定 */
714 : trdmr = 0xf0; /* バッファレジスタ設定 */
715 : trdfcr = 0x01; /* リセット同期PWMモードに設定 */
716 : trdcr0 = 0x23; /* ソースカウントの選択:f8 */
717 : trdgra0 = trdgrc0 = PWM_CYCLE; /* 周期 */
718 : trdgrb0 = trdgrd0 = 0; /* P2_2端子のON幅設定 */
719 : trdgral = trdgrcl = 0; /* P2_4端子のON幅設定 */
720 : trdgrbl = trdgrdl = SERVO_CENTER; /* P2_5端子のON幅設定 */
721 : trdoerl = 0xcd; /* 出力端子の選択 */
722 : trdstr = 0x0d; /* TRD0カウント開始 */
723 : }
724 :
725 : /*****
726 : /* タイマRB 割り込み処理
727 : /*****
728 : #pragma interrupt intTRB(vect=24)
729 : void intTRB( void )
730 : {
731 :     signed char *p;
732 :
733 :     cnt0++;
734 :     cnt1++;
735 :
736 :     /* microSD間欠書き込み処理(1msごとに実行) */
737 :     microSDProcess();
738 :
739 :     /* microSD記録処理 */
740 :     if( msdFlag == 1 ) {
741 :         /* 記録間隔のチェック */
742 :         msdTimer++;
743 :         if( msdTimer >= 10 ) {
744 :             msdTimer = 0;
745 :             p = msdBuff + msdBuffAddress;
746 :
747 :             /* バッファに記録 ここから */
748 :             *p++ = pattern; /* パターン */
749 :             *p++ = sensor_inp(0xff); /* センサ */
750 :             *p++ = handleBuff; /* ハンドル */
751 :             *p++ = leftMotorBuff; /* 左モータPWM値 */
752 :             *p++ = rightMotorBuff; /* 右モータPWM値 */
753 :             /* バッファに記録 ここまで */
754 :
755 :             msdBuffAddress += 64; /* RAMの記録アドレスを次へ */
756 :
757 :             if( msdBuffAddress >= 512 ) {
758 :                 /* 512個になったら、microSDに記録する */
759 :                 msdBuffAddress = 0;
760 :                 setMicroSDdata( msdBuff );
761 :                 msdWorkAddress += 512;
762 :                 if( msdWorkAddress >= msdEndAddress ) {
763 :                     /* 記録処理終了 */
764 :                     msdFlag = 0;
765 :                 }
766 :             }
767 :         }
768 :     }
769 : }
770 :

```

中略

## 8. プロジェクト「kit12msd01\_38a」 走行データを microSD に記録

```

904 : /*****/
905 : /* モータ速度制御 */
906 : /* 引数 左モータ:-100~100、右モータ:-100~100 */
907 : /* 0で停止、100で正転100%、-100で逆転100% */
908 : /* 戻り値 なし */
909 : /*****/
910 : void motor( int accele_l, int accele_r )
911 : {
912 :     int    sw_data;
913 :
914 :     sw_data = dipsw_get() + 5;
915 :     accele_l = accele_l * sw_data / 20;
916 :     accele_r = accele_r * sw_data / 20;
917 :
918 :     leftMotorBuff = accele_l;      /* バッファに保存 */
919 :     rightMotorBuff = accele_r;    /* バッファに保存 */
920 :
921 :     /* 左モータ制御 */
922 :     if( accele_l >= 0 ) {
923 :         p2 &= 0xfd;
924 :         trdgrd0 = (long)( PWM_CYCLE - 1 ) * accele_l / 100;
925 :     } else {
926 :         p2 |= 0x02;
927 :         trdgrd0 = (long)( PWM_CYCLE - 1 ) * ( -accele_l ) / 100;
928 :     }
929 :
930 :     /* 右モータ制御 */
931 :     if( accele_r >= 0 ) {
932 :         p2 &= 0xf7;
933 :         trdgrd1 = (long)( PWM_CYCLE - 1 ) * accele_r / 100;
934 :     } else {
935 :         p2 |= 0x08;
936 :         trdgrd1 = (long)( PWM_CYCLE - 1 ) * ( -accele_r ) / 100;
937 :     }
938 : }
939 :
940 : /*****/
941 : /* サーボハンドル操作 */
942 : /* 引数  サーボ操作角度 : -90~90 */
943 : /*      -90で左へ90度、0でまっすぐ、90で右へ90度回転 */
944 : /*****/
945 : void handle( int angle )
946 : {
947 :     handleBuff = angle;          /* バッファに保存 */
948 :
949 :     /* サーボが左右逆に動く場合は、「-」を「+」に替えてください */
950 :     trdgrd1 = SERVO_CENTER - angle * HANDLE_STEP;
951 : }
952 :
953 : /*****/
954 : /* char型データの値をlong型変数に2進数で変換 */
955 : /* 引数  unsigned char 変換元の8bitデータ */
956 : /* 戻り値 unsigned long 変換先の変数(0~11111111) ※0か1しかありません */
957 : /*****/
958 : unsigned long convertBCD_CharToLong( unsigned char hex )
959 : {
960 :     int    i;
961 :     unsigned long    l = 0;
962 :
963 :     for( i=0; i<8; i++ ) {
964 :         l *= 10;
965 :         if( hex & 0x80 ) l += 1;
966 :         hex <<= 1;
967 :     }
968 :
969 :     return l;
970 : }
971 :
972 : /*****/
973 : /* end of file */
974 : /*****/

```



## 8.5 プログラムの解説

### 8.5.1 変数

```

60 : /*=====*/
61 : /* グローバル変数の宣言 */
62 : /*=====*/
63 : unsigned long cnt0; /* timer関数用 */
64 : unsigned long cnt1; /* main内で使用 */
65 : int pattern; /* パターン番号 */
66 :
67 : /* microSD関連変数 */
68 : signed char msdBuff[ 512 ]; /* 一時保存バッファ */
69 : int msdBuffAddress; /* 一時記録バッファ書込アドレス */
70 : int msdFlag; /* 1:データ記録 0:記録しない */
71 : int msdTimer; /* 取得間隔計算用 */
72 : unsigned long msdStartAddress; /* 記録開始アドレス */
73 : unsigned long msdEndAddress; /* 記録終了アドレス */
74 : unsigned long msdWorkAddress; /* 作業用アドレス */
75 : int msdError; /* エラー番号記録 */
76 :
77 : /* 現在の状態保存用 */
78 : int handleBuff; /* 現在のハンドル角度記録 */
79 : int leftMotorBuff; /* 現在の左モータPWM値記録 */
80 : int rightMotorBuff; /* 現在の右モータPWM値記録 */
    
```

microSD に走行データを記録するにあたって、新たにグローバル変数を追加しています。各変数の役割を下表に示します。

変数名	内容
msdError	microSD 処理にエラーがあった場合は、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。

### 8.5.2 main 関数(初期化)

```

82 : /*****
83 : /* メインプログラム */
84 : /*****
85 : void main( void )
86 : {
87 :     int    ret;
88 :
89 :     /* マイコン機能の初期化 */
90 :     init();                /* 初期化 */
91 :     init_uart0_printf( SPEED_9600 ); /* UART0とprintf関連の初期化 */
92 :     setMicroSDLedPort( &p6, &pd6, 0 ); /* microSD モニタLED設定 */
93 :     asm(" fset I ");      /* 全体の割り込み許可 */
94 :
95 :     // microSD 書き込み開始アドレス
96 :     // 512の倍数に設定する
97 :     msdStartAddress = 0;
98 :
99 :     // microSD 書き込み終了アドレス
100 :    // 書き込みしたい時間[ms] : x = 10[ms] : 64バイト
101 :    // 60000msなら、x = 60000 * 64 / 10 = 384000
102 :    // 結果は512の倍数になるように繰り上げる。
103 :    msdEndAddress = 384000;
104 :    msdEndAddress += msdStartAddress; /* スタート分足す */
105 :
106 :    /* microSD初期化 */
107 :    ret = initMicroSD();
108 :    if( ret != 0x00 ) {
109 :        msdError = 1;
110 :        /* 初期化できなければ3秒間、LEDの点灯方法を変える */
111 :        cnt1 = 0;
112 :        while( cnt1 < 3000 ) {
113 :            if( cnt1 % 200 < 100 ) {
114 :                led_out( 0x3 );
115 :            } else {
116 :                led_out( 0x0 );
117 :            }
118 :        }
119 :    }
120 :
121 :    /* 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 */
122 :    if( pushsw_get() ) {
123 :        pattern = 71;
124 :    }
    
```

97 行目	microSD の書き込み開始アドレスを設定します。
103 行目 ~ 104 行目	microSD に書き込む容量を指定します。104 行目で、書き込む容量に書き込み開始アドレスを加えて終了アドレスに指定します。 今回、データの記録条件を次のようにしました。 <ul style="list-style-type: none"> <li>• データ記録の間隔 … 10ms ごと</li> <li>• データ記録数 …………… 64 バイト</li> <li>• データ記録時間 …… 60 秒(60000ms)</li> </ul> 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。

	<p>容量 = 記録したい時間[ms] ÷ 記録する間隔[ms] × 1 回に記録するバイト数</p> <p>よって、容量は次のとおりです。</p> $= 60000 \div 10 \times 64$ $= 384000$ <p>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。</p> $384000 \div 512 = 750 \text{ 余り } 0$ <p>割り切れますので、この値で OK です。103 行目に、「384000」を設定します。</p>
107 行目	microSD を初期化します。初期化エラーであれば、109 行目で msdError 変数に 1 を代入してエラー情報を保存します。111~119 行目で 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。
122 行目	プッシュスイッチが押されているかチェックします。起動時にプッシュ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)に移ります。

### 8.5.3 パターン 0: スイッチ入力待ち

パターン 0 は、プッシュスイッチ入力待ちで、プッシュスイッチが押されたら 160 行目以降を実行します。

157 :	case 0:
158 :	/* スイッチ入力待ち */
159 :	if( pushsw_get() ) {
<b>160 :</b>	<b>ret = eraseMicroSD( msdStartAddress, msdEndAddress-1 );</b>
161 :	if( ret != 0x00 ) {
162 :	/* イレーズできず */
163 :	msdError = 2;
164 :	}
165 :	/* microSDProcess開始処理 */
<b>166 :</b>	<b>ret = microSDProcessStart( msdStartAddress );</b>
167 :	if( ret != 0x00 ) {
168 :	/* 開始処理できず */
169 :	msdError = 3;
170 :	}
171 :	pattern = 1;
172 :	cnt1 = 0;
173 :	break;
174 :	}
175 :	if( cnt1 < 100 ) { /* LED点滅処理 */
176 :	led_out( 0x1 );
177 :	} else if( cnt1 < 200 ) {
178 :	led_out( 0x2 );
179 :	} else {
180 :	cnt1 = 0;
181 :	}
182 :	break;

160 行目	microSD の記録開始アドレスから終了アドレスまでをイレーズします。イレーズエラーなら、163 行目で msdError 変数に 2 を代入してエラー情報を保存します。
166 行目	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。イレーズ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。

### 8.5.4 パターン 1: スタートバーが開いたかチェック

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)、188 行目以降を実行します。

```

184 :     case 1:
185 :         /* スタートバーが開いたかチェック */
186 :         if( !startbar_get() ) {
187 :             /* スタート!! */
188 :             led_out( 0x0 );
189 :             pattern = 11;
190 :             msdBuffAddress = 0;
191 :             msdWorkAddress = msdStartAddress;
192 :             msdFlag = 1;                /* データ記録開始                */
193 :             cnt1 = 0;
194 :             break;
195 :         }
196 :         if( cnt1 < 50 ) {                /* LED点滅処理                */
197 :             led_out( 0x1 );
198 :         } else if( cnt1 < 100 ) {
199 :             led_out( 0x2 );
200 :         } else {
201 :             cnt1 = 0;
202 :         }
203 :         break;
    
```

190 行目	msdBuff 配列変数(RAM)を参照する変数を 0 にクリアしています。
191 行目	microSD の作業アドレスを書き込み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。
192 行目	msdFlag 変数を 1 にします。192 行目以降の 1ms ごとの割り込みから、記録が開始されます。

### 8.5.5 パターン 71: 走行データ転送準備

パターン 71 は、走行データの転送準備を行います。

```

539 :     case 71:
540 :         /* 走行データ転送準備 */
541 :         handle( 0 );
542 :         motor( 0, 0 );
543 :         msdFlag = 0;
544 :         if( msdError != 0 ) {
545 :             /* microSDに不具合があったなら終了 */
546 :             printf( "microSD Initialize Error!!\n" );
547 :             pattern = 99;
548 :         } else {
549 :             pattern = 72;
550 :             cnt1 = 0;
551 :         }
552 :         break;
    
```

544 行目	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 へ移り何もみません。エラーが特になければ、パターン 72 へ移ります。
--------	---

### 8.5.6 パターン 72:最後のデータ書き込むまで待つ

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。処理が終わってれば、パターン 73 へ移ります。

```

554 :     case 72:
555 :         /* 最後のデータ書き込むまで待つ*/
556 :         if( checkMicroSDProcess() == 0 ) {
557 :             pattern = 73;          /* データ転送処理へ          */
558 :             break;
559 :         }
560 :         if( checkMicroSDProcess() == 11 ) {
561 :             microSDProcessEnd();    /* microSDProcess終了処理    */
562 :             while( checkMicroSDProcess() );
563 :             pattern = 73;          /* データ転送処理へ          */
564 :         }
565 :         break;
    
```

今回のプログラムは、電源を入れたときにプッシュ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。

### 8.5.7 パターン 73、74:プッシュスイッチが離されたかチェック

パターン 73、74 は、プッシュスイッチが離されたかチェックします。

```

567 :     case 73:
568 :         /* スイッチが離されたかチェック */
569 :         if( !pushsw_get() ) {
570 :             pattern = 74;
571 :             cnt1 = 0;
572 :         }
573 :         break;
574 :
575 :     case 74:
576 :         /* 0.2s待ち */
577 :         if( cnt1 > 200 ) {
578 :             pattern = 75;
579 :             cnt1 = 0;
580 :             break;
581 :         }
582 :         if( pushsw_get() ) {
583 :             pattern = 73;
584 :         }
585 :         break;
    
```

567 行目 ～ 573 行目	パターン 73 でプッシュ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。
575 行目 ～ 585 行目	パターン 74 では、再度プッシュ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。

### 8.5.8 パターン 75:スイッチが押されたかチェック

パターン 75 は、プッシュ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。

```
587 :     case 75:
588 :         /* スイッチが押されたかチェック */
589 :         led_out( (cnt1/500) % 2 + 1 );
590 :         if( pushsw_get() ) {
591 :             pattern = 76;
592 :             cnt1 = 0;
593 :         }
594 :         break;
```

### 8.5.9 パターン 76:タイトル送信

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。

```
596 :     case 76:
597 :         /* タイトル転送、準備 */
598 :         printf( "\n" );
599 :         printf( "Your Car Name  Data Out\n" );
600 :         printf( "Pattern, Sensor, ハンドル, 左モータ, 右モータ\n" );
601 :
602 :         msdWorkAddress = msdStartAddress; /* 読み込み開始アドレス */
603 :         pattern = 77;
604 :         break;
```

### 8.5.10 パターン 77:microSD よりデータ読み込み

パターン 77 は、microSD からデータを読み込みます。

```
606 :     case 77:
607 :         /* microSDよりデータ読み込み */
608 :         if( msdWorkAddress >= msdEndAddress ) {
609 :             /* 書き込み終了アドレスになったら、終わり */
610 :             pattern = 99;
611 :             break;
612 :         }
613 :         ret = readMicroSD( msdWorkAddress , msdBuff );
614 :         if( ret != 0x00 ) {
615 :             /* 読み込みエラー */
616 :             printf( "\nmicroSD Read Error!!\n" );
617 :             pattern = 99;
618 :             break;
619 :         } else {
620 :             /* エラーなし */
621 :             msdWorkAddress += 512; 次にmicroSDから読み込むアドレスをセット
622 :             msdBuffAddress = 0; 今回読み込んだデータを参照する変数をクリア
623 :             pattern = 78;
624 :         }
625 :         break;
```

608 行目	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。
613 行目	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、621 行目で読み込みアドレスを+512 して、次に読み込むアドレスを設定しておきます。また 622 行目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。

### 8.5.11 パターン 78:データ転送

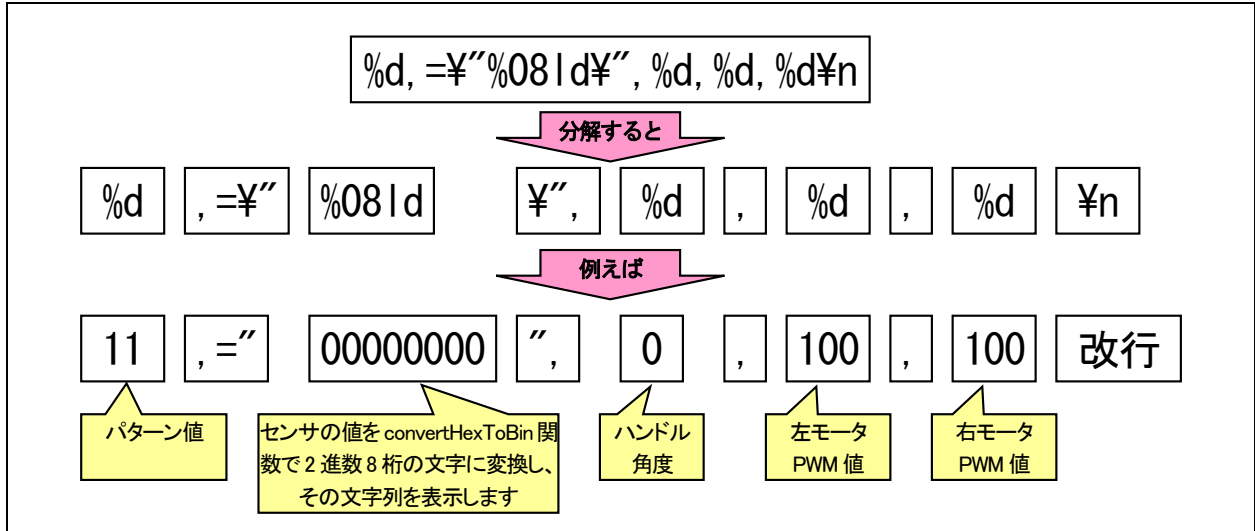
パターン 78 は、パソコンヘデータを転送する部分です。

```

627 :     case 78:
628 :         /* データ転送 */
629 :         led_out( (cnt1/100) % 2 + 1 ); /* LED点滅処理 */
630 :
631 :         if( msdBuff[msdBuffAddress+0] == 0 ) {
632 :             /* パターンが0なら終了 */
633 :             printf( "End. ¥n" );
634 :             pattern = 99;
635 :             break;
636 :         }
637 :
638 :         printf( "%d, =¥"%08ld¥", %d, %d, %d¥n",
639 :             msdBuff[msdBuffAddress+0], /* パターン */
640 :             convertBCD_CharToLong( msdBuff[msdBuffAddress+1] ), /* センサ*/
641 :             msdBuff[msdBuffAddress+2], /* ハンドル */
642 :             msdBuff[msdBuffAddress+3], /* 左モータ */
643 :             msdBuff[msdBuffAddress+4] /* 右モータ */
644 :         );
645 :         msdBuffAddress += 64;
646 :
647 :         if( msdBuffAddress >= 512 ) {
648 :             pattern = 77;
649 :         }
650 :         break;
    
```

631 行目	パターン番号をチェックし、0 ならデータはもうないと判断して転送を終了します。
638 行目 ～ 644 行目	パソコンヘデータを転送しています。
645 行目	次に転送する msdBuff 配列変数の位置をセットします。
647 行目	msdBuff 配列変数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。

パソコンへ送る形式を下記に示します。センサの値は、printf 文を実行する前に convertBCD\_CharToLong 関数で unsigned long 型の 2 進数("0"or"1")に変換し、0~11111111 の値を返します。



実際の転送データ例を下記に示します。

```
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
```

### 8.5.12 パターン 99: 転送終了

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。

```
652 :     case 99:
653 :         /* 転送終了 */
654 :         led_out( 0x3 );
655 :         break;
```



### 8.5.13 割り込み処理

```

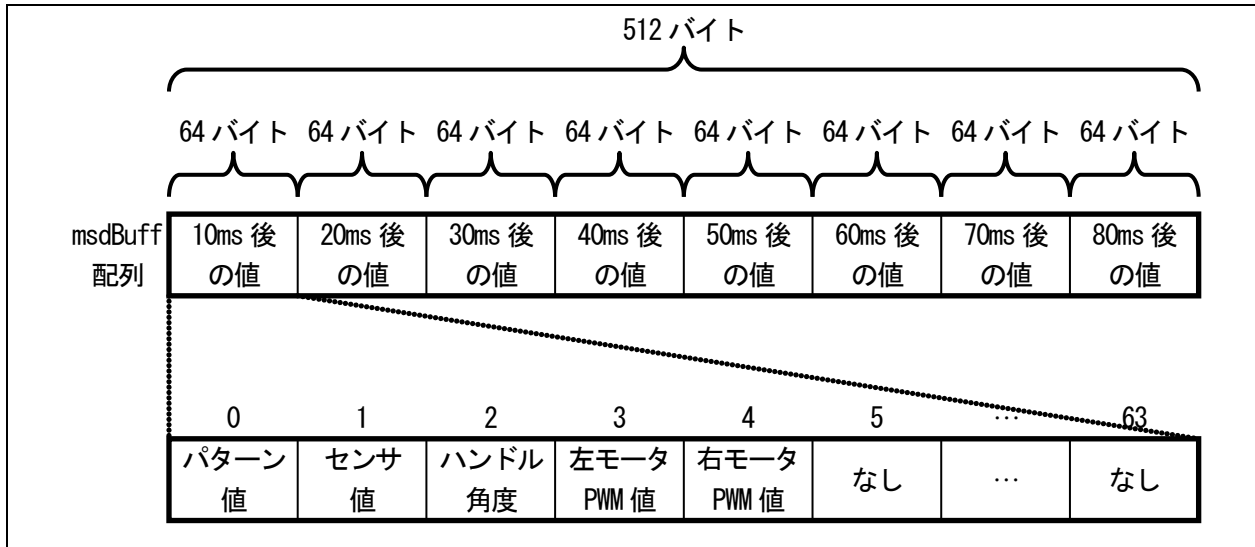
725 : /*****
726 : /* タイマRB 割り込み処理 */
727 : /*****
728 : #pragma interrupt intTRB(vect=24)
729 : void intTRB( void )
730 : {
731 :     signed char *p;
732 :
733 :     cnt0++;
734 :     cnt1++;
735 :
736 :     /* microSD間欠書き込み処理(1msごとに実行) */
737 :     microSDProcess();
738 :
739 :     /* microSD記録処理 */
740 :     if( msdFlag == 1 ) {
741 :         /* 記録間隔のチェック */
742 :         msdTimer++;
743 :         if( msdTimer >= 10 ) {
744 :             msdTimer = 0;
745 :             p = msdBuff + msdBuffAddress;
746 :
747 :             /* バッファに記録 ここから */
748 :             *p++ = pattern;          /* パターン */
749 :             *p++ = sensor_inp(0xff); /* センサ */
750 :             *p++ = handleBuff;      /* ハンドル */
751 :             *p++ = leftMotorBuff;   /* 左モータPWM値 */
752 :             *p++ = rightMotorBuff;  /* 右モータPWM値 */
753 :             /* バッファに記録 ここまで */
754 :
755 :             msdBuffAddress += 64;    /* RAMの記録アドレスを次へ */
756 :
757 :             if( msdBuffAddress >= 512 ) {
758 :                 /* 512個になったら、microSDに記録する */
759 :                 msdBuffAddress = 0;
760 :                 setMicroSDdata( msdBuff );
761 :                 msdWorkAddress += 512;
762 :                 if( msdWorkAddress >= msdEndAddress ) {
763 :                     /* 記録処理終了 */
764 :                     msdFlag = 0;
765 :                 }
766 :             }
767 :         }
768 :     }
769 : }
    
```

743 行目	記録間隔のチェックをしています。今回は、msdTimer 変数が 10 以上になったら、すなわち 10ms たったなら記録処理を行います。
755 行目	msdBuff 配列変数の番地を次に記録する番地に設定します。今回は、64 バイトごとに記録をしていますので、msdBuffAddress 変数に 64 を足します。
757 行目	msdBuffAddress 変数が 512 バイトになったかチェックします。512 バイトになったら msdBuff 配列変数に格納したデータが 512 バイト記録したと判断し、microSD へ書き込み処理をします。
760 行目	setMicroSDdata 関数で microSD に書き込む準備を行います。

8. プロジェクト「kit12msd01\_38a」 走行データを microSD に記録

762 行目 ～ 765 行目	msdWorkAddress 変数が書き込み終了アドレスより大きくなったかチェックします。大きくなったなら、書き込み終了アドレスまでデータを記録したと判断し、msdFlag 変数を 0 にして、記録処理を終了します。
-----------------------	--

748 行目～752 行目が msdBuff 配列変数に記録している内容です。記録イメージを下図に示します。



8.5.14 記録データをバッファに保存

motor 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。

```

910 : void motor( int accele_l, int accele_r )
911 : {
912 :     int    sw_data;
913 :
914 :     sw_data = dipsw_get() + 5;
915 :     accele_l = accele_l * sw_data / 20;
916 :     accele_r = accele_r * sw_data / 20;
917 :
918 :     leftMotorBuff = accele_l;          /* バッファに保存          */
919 :     rightMotorBuff = accele_r;        /* バッファに保存          */

```

中略

```

945 : void handle( int angle )
946 : {
947 :     handleBuff = angle;                /* バッファに保存          */
948 :
949 :     /* サーボが左右逆に動く場合は、「-」を「+」に替えてください */
950 :     trdgrd1 = SERVO_CENTER - angle * HANDLE_STEP;
951 : }

```

以下、略

## 8.6 プログラムの調整

### 8.6.1 自分のマイコンカーに合わせて調整

「kit12msd01\_38a.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
29	#define SERVO_CENTER <b>3750</b>	自分のマイコンカーのサーボセンタ値に変更します。
341	handle( <b>-38</b> );	自分のマイコンカーの左最大切れ角の値に変更します。
350	handle( <b>38</b> );	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「kit12msd01\_38a」をビルドして、「kit12msd01\_38a.mot」ファイルを RY\_R8C38 ボードに書き込みます。

### 8.6.2 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を下表に示します。

記録間隔	645 行目の 変更	647 行目の 変更	743 行目の 変更	755 行目の 変更	757 行目の 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

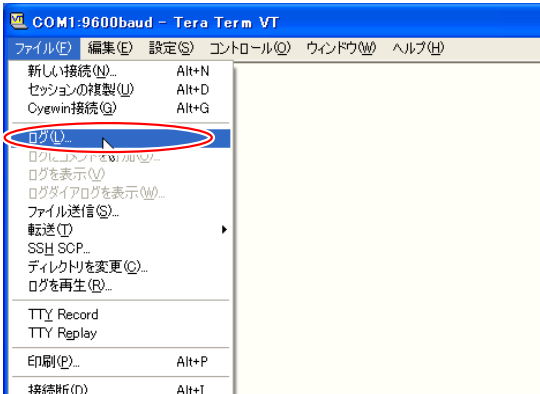
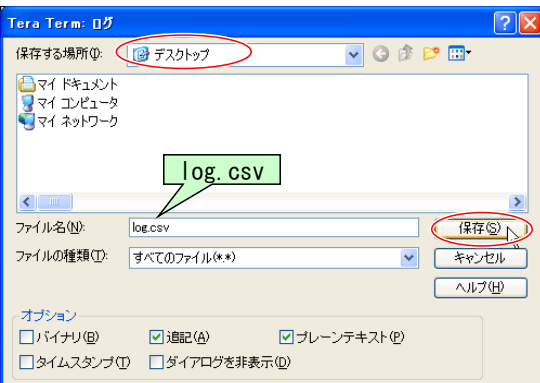
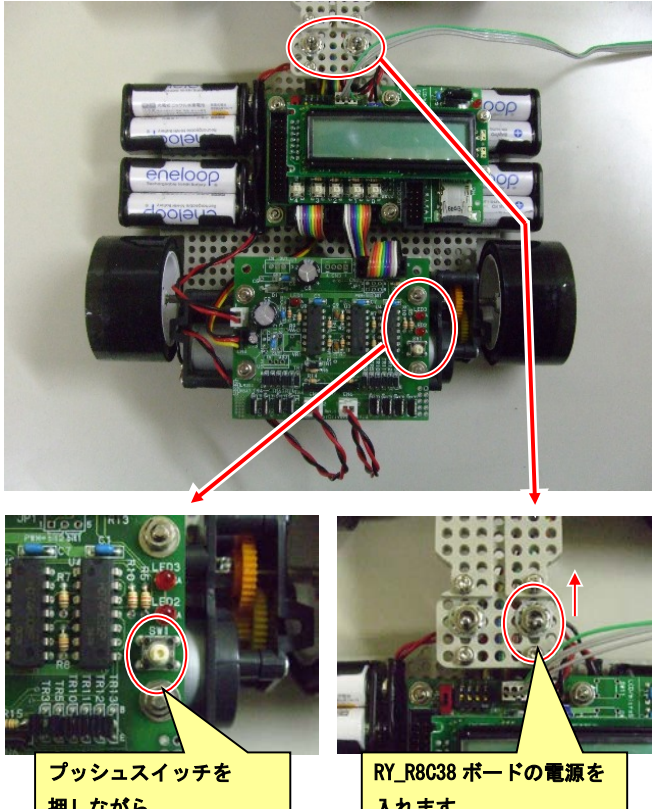
## 8.7 走行からデータ転送までの流れ

マイコンカーを走行させます。走行データが記録できるのは、スタートしてから 60 秒間です。

**走行後、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。**

### 8.7.1 走行データの取り込み

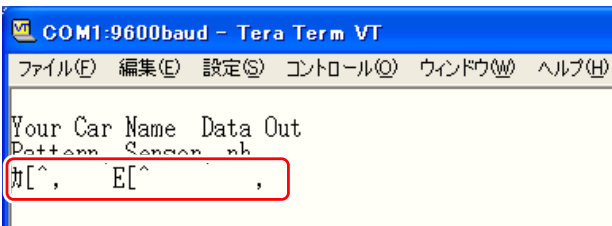
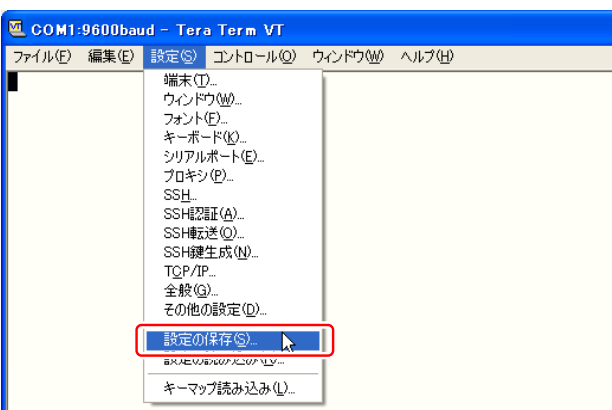
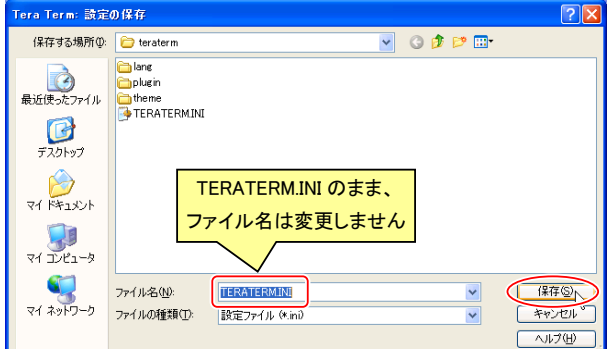
1	 <p>RY-WRITER 基板</p> <p>4ピンケーブル</p> <p>USBケーブル</p> <p>マイコンカーキット (RY_R8C38 ボード)</p>	<p>マイコンカー (RY_R8C38 ボード)とパソコン間を RY-WRITER 基板、USB ケーブル、4ピンケーブルで接続します</p>
2		<p>Tera Term を立ち上げます。  <b>※マイコンカーの電源は、まだ入れません。</b></p>
3		<p>①:「シリアルポート」を選択します。ポート番号は、R8C Writer で選択している番号と同じ番号にします。RY-WRITER 基板の場合は、「Prolific USB-to-Serial Com Port」と表示されている番号です。</p> <p>②: <b>OK</b> をクリックします。</p>
4		<p>立ち上がりました。</p>

5		<p>受信データをファイルに保存します。          「ファイル→ログ」を選択します。</p>
6		<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、「保存」をクリックします。  <b>※拡張子は必ず「csv」にします。</b></p>
7	 <p>プッシュスイッチを押しながら</p> <p>RY_R8C38 ボードの電源を入れます</p>	<p>マイコンカーは、モータドライブ基板のプッシュスイッチを押しながら、RY_R8C38 ボードの電源を ON にします。</p>

8. プロジェクト「kit12msd01\_38a」 走行データを microSD に記録

<p>8</p>		<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 を繰り返した場合は、microSD と接続できていませんので、接続を確認してください。</p>
<p>9</p>		<p>モータドライブ基板のプッシュスイッチを再度押します。 Tera Term 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、Tera Term は終了します。</p>
<p>10</p>		<p>エディタなどで「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。</p>

8.7.2 Tera Term の設定：文字化けに対する設定

1		<p>microSD からデータを表示すると、左図のように、□で囲んだ部分が文字化けしてしまいます。          このような文字化けをしないように設定します。</p>
2		<p>「設定→端末」をクリックします。</p>
3		<p>□で囲んだ部分2箇所を「UTF-8」から「SJIS」に設定を変えます。設定ができれば「OK」をクリックします。これで設定は完了です。</p>
4		<p>ここでは、Tera Term を毎回立ち上げるたびに設定しなければいけません。手間がかかります。この設定内容を保存することになります。          「設定→設定の保存」をクリックします。</p>
5		<p>ファイル名の「TERATERM.INI」を変更せずに、「保存」をクリックします。          これで、次回から Tera Term を立ち上げると、この設定のまま立ち上がります。</p>

### 8.8 エクセルへの取り込み方

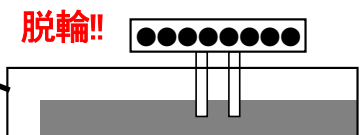
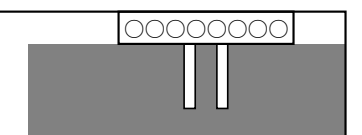
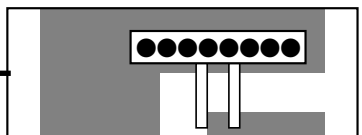
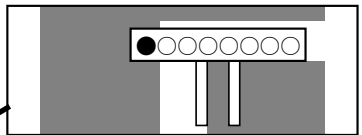
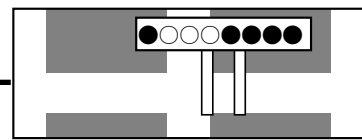
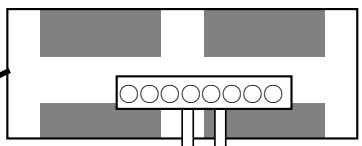
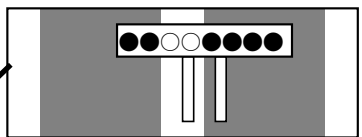
1	<table border="1"> <thead> <tr> <th></th> <th>A</th> <th>B</th> <th>C</th> <th>D</th> <th>E</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>1</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>2</td> <td colspan="2">Your Car Name Data Out</td> <td></td> <td></td> <td></td> <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></td> </tr> <tr> <td>4</td> <td>11</td> <td>00011000</td> <td>0</td> <td>85</td> <td>85</td> <td></td> </tr> <tr> <td>5</td> <td>11</td> <td>00011000</td> <td>0</td> <td>85</td> <td>85</td> <td></td> </tr> <tr> <td>6</td> <td>11</td> <td>00011000</td> <td>0</td> <td>85</td> <td>85</td> <td></td> </tr> <tr> <td>7</td> <td>11</td> <td>00011000</td> <td>0</td> <td>85</td> <td>85</td> <td></td> </tr> <tr> <td>8</td> <td>11</td> <td>00011000</td> <td>0</td> <td>85</td> <td>85</td> <td></td> </tr> <tr> <td>9</td> <td>11</td> <td>00011000</td> <td>0</td> <td>85</td> <td>85</td> <td></td> </tr> <tr> <td>10</td> <td>11</td> <td>00011000</td> <td>0</td> <td>85</td> <td>85</td> <td></td> </tr> <tr> <td>11</td> <td>11</td> <td>00011000</td> <td>0</td> <td>85</td> <td>85</td> <td></td> </tr> <tr> <td>12</td> <td>11</td> <td>00011000</td> <td>0</td> <td>85</td> <td>85</td> <td></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		4	11	00011000	0	85	85		5	11	00011000	0	85	85		6	11	00011000	0	85	85		7	11	00011000	0	85	85		8	11	00011000	0	85	85		9	11	00011000	0	85	85		10	11	00011000	0	85	85		11	11	00011000	0	85	85		12	11	00011000	0	85	85		<p>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。</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																																																																																																										
	4	11	00011000	0	85	85																																																																																																										
	5	11	00011000	0	85	85																																																																																																										
	6	11	00011000	0	85	85																																																																																																										
	7	11	00011000	0	85	85																																																																																																										
	8	11	00011000	0	85	85																																																																																																										
	9	11	00011000	0	85	85																																																																																																										
	10	11	00011000	0	85	85																																																																																																										
	11	11	00011000	0	85	85																																																																																																										
12	11	00011000	0	85	85																																																																																																											
2	<table border="1"> <thead> <tr> <th></th> <th>パターン</th> <th>センサ</th> <th>ハンドル</th> <th>左モータ</th> <th>右モータ</th> <th></th> </tr> </thead> <tbody> <tr><td>178</td><td>11</td><td>00011000</td><td>0</td><td>85</td><td>85</td><td></td></tr> <tr><td>179</td><td>11</td><td>00011000</td><td>0</td><td>85</td><td>85</td><td></td></tr> <tr><td>180</td><td>11</td><td>00011000</td><td>0</td><td>85</td><td>85</td><td></td></tr> <tr><td>181</td><td>11</td><td>00011000</td><td>0</td><td>85</td><td>85</td><td></td></tr> <tr><td>182</td><td>22</td><td>11111111</td><td>0</td><td>0</td><td>0</td><td></td></tr> <tr><td>183</td><td>22</td><td>11111111</td><td>0</td><td>0</td><td>0</td><td></td></tr> <tr><td>184</td><td>22</td><td>11111100</td><td>0</td><td>0</td><td>0</td><td></td></tr> <tr><td>185</td><td>22</td><td>00011000</td><td>0</td><td>0</td><td>0</td><td></td></tr> <tr><td>186</td><td>22</td><td>00011000</td><td>0</td><td>0</td><td>0</td><td></td></tr> <tr><td>187</td><td>22</td><td>00011000</td><td>0</td><td>0</td><td>0</td><td></td></tr> <tr><td>188</td><td>22</td><td>00011000</td><td>0</td><td>0</td><td>0</td><td></td></tr> <tr><td>189</td><td>22</td><td>00011000</td><td>0</td><td>0</td><td>0</td><td></td></tr> <tr><td>190</td><td>22</td><td>00011000</td><td>0</td><td>0</td><td>0</td><td></td></tr> </tbody> </table>							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		183	22	11111111	0	0	0		184	22	11111100	0	0	0		185	22	00011000	0	0	0		186	22	00011000	0	0	0		187	22	00011000	0	0	0		188	22	00011000	0	0	0		189	22	00011000	0	0	0		190	22	00011000	0	0	0		<p>1 行あたり、10ms の時間になります。クロスラインの検出幅は 3 行あります。よって、30ms かかってクロスラインを通過したことが分かります。</p>							
		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																																																																																																										
	183	22	11111111	0	0	0																																																																																																										
	184	22	11111100	0	0	0																																																																																																										
	185	22	00011000	0	0	0																																																																																																										
	186	22	00011000	0	0	0																																																																																																										
	187	22	00011000	0	0	0																																																																																																										
	188	22	00011000	0	0	0																																																																																																										
189	22	00011000	0	0	0																																																																																																											
190	22	00011000	0	0	0																																																																																																											
3	<table border="1"> <thead> <tr> <th></th> <th>パターン</th> <th>センサ</th> <th>ハンドル</th> <th>左モータ</th> <th>右モータ</th> <th></th> </tr> </thead> <tbody> <tr><td>233</td><td>23</td><td>00011000</td><td>0</td><td>34</td><td>34</td><td></td></tr> <tr><td>234</td><td>23</td><td>00011000</td><td>0</td><td>34</td><td>34</td><td></td></tr> <tr><td>235</td><td>23</td><td>00011000</td><td>0</td><td>34</td><td>34</td><td></td></tr> <tr><td>236</td><td>31</td><td>11111000</td><td>-38</td><td>8</td><td>42</td><td></td></tr> <tr><td>237</td><td>31</td><td>11111000</td><td>-38</td><td>8</td><td>42</td><td></td></tr> <tr><td>238</td><td>31</td><td>11111000</td><td>-38</td><td>8</td><td>42</td><td></td></tr> <tr><td>239</td><td>31</td><td>11111000</td><td>-38</td><td>8</td><td>42</td><td></td></tr> <tr><td>240</td><td>31</td><td>11111000</td><td>-38</td><td>8</td><td>42</td><td></td></tr> <tr><td>241</td><td>31</td><td>11111000</td><td>-38</td><td>8</td><td>42</td><td></td></tr> <tr><td>242</td><td>31</td><td>11111000</td><td>-38</td><td>8</td><td>42</td><td></td></tr> <tr><td>243</td><td>31</td><td>11000000</td><td>-38</td><td>8</td><td>42</td><td></td></tr> <tr><td>244</td><td>31</td><td>00000000</td><td>-38</td><td>8</td><td>42</td><td></td></tr> <tr><td>245</td><td>31</td><td>00000000</td><td>-38</td><td>8</td><td>42</td><td></td></tr> <tr><td>246</td><td>31</td><td>00000000</td><td>-38</td><td>8</td><td>42</td><td></td></tr> </tbody> </table>							パターン	センサ	ハンドル	左モータ	右モータ		233	23	00011000	0	34	34		234	23	00011000	0	34	34		235	23	00011000	0	34	34		236	31	11111000	-38	8	42		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		<p>左クラックを検出し、ハンドルを左へ曲げていることが分かります。</p>
		パターン	センサ	ハンドル	左モータ	右モータ																																																																																																										
	233	23	00011000	0	34	34																																																																																																										
	234	23	00011000	0	34	34																																																																																																										
	235	23	00011000	0	34	34																																																																																																										
	236	31	11111000	-38	8	42																																																																																																										
	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																																																																																																											
4	<table border="1"> <thead> <tr> <th></th> <th>パターン</th> <th>センサ</th> <th>ハンドル</th> <th>左モータ</th> <th>右モータ</th> <th></th> </tr> </thead> <tbody> <tr><td>339</td><td>32</td><td>10000011</td><td>-38</td><td>8</td><td>42</td><td></td></tr> <tr><td>340</td><td>32</td><td>10000001</td><td>-38</td><td>8</td><td>42</td><td></td></tr> <tr><td>341</td><td>32</td><td>10000001</td><td>-38</td><td>8</td><td>42</td><td></td></tr> <tr><td>342</td><td>32</td><td>10000001</td><td>-38</td><td>8</td><td>42</td><td></td></tr> <tr><td>343</td><td>32</td><td>11000001</td><td>-38</td><td>8</td><td>42</td><td></td></tr> <tr><td>344</td><td>32</td><td>11000001</td><td>-38</td><td>8</td><td>42</td><td></td></tr> <tr><td>345</td><td>32</td><td>11000000</td><td>-38</td><td>8</td><td>42</td><td></td></tr> <tr><td>346</td><td>32</td><td>11000000</td><td>-38</td><td>8</td><td>42</td><td></td></tr> <tr><td>347</td><td>32</td><td>11000000</td><td>-38</td><td>8</td><td>42</td><td></td></tr> <tr><td>348</td><td>32</td><td>11100000</td><td>-38</td><td>8</td><td>42</td><td></td></tr> <tr><td>349</td><td>32</td><td>11100000</td><td>-38</td><td>8</td><td>42</td><td></td></tr> <tr><td>350</td><td>11</td><td>01100000</td><td>-10</td><td>58</td><td>68</td><td></td></tr> <tr><td>351</td><td>11</td><td>01100000</td><td>-10</td><td>58</td><td>68</td><td></td></tr> <tr><td>352</td><td>11</td><td>01110000</td><td>-10</td><td>58</td><td>68</td><td></td></tr> </tbody> </table>							パターン	センサ	ハンドル	左モータ	右モータ		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		351	11	01100000	-10	58	68		352	11	01110000	-10	58	68		<p>中心線を検出し、パターン 11 へ復帰していることが分かります。</p>
		パターン	センサ	ハンドル	左モータ	右モータ																																																																																																										
	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																																																																																																											
351	11	01100000	-10	58	68																																																																																																											
352	11	01110000	-10	58	68																																																																																																											



## 8.9 データをエクセルで解析する

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

パターン	センサ 2進数
11	00110000
11	00110000
11	00110000
11	00110000
11	00110000
22	11111111
22	11111111
22	11111111
22	00110000
23	00110000
23	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
<b>23</b>	<b>01111111</b>
<b>23</b>	<b>01111111</b>
23	00000000
23	00000000
23	00000000
23	00000000
23	00000000
23	00000000
23	00000000
23	11100000
23	11111111
23	11111111
23	11111111
23	10000000
23	00000000
23	00000000



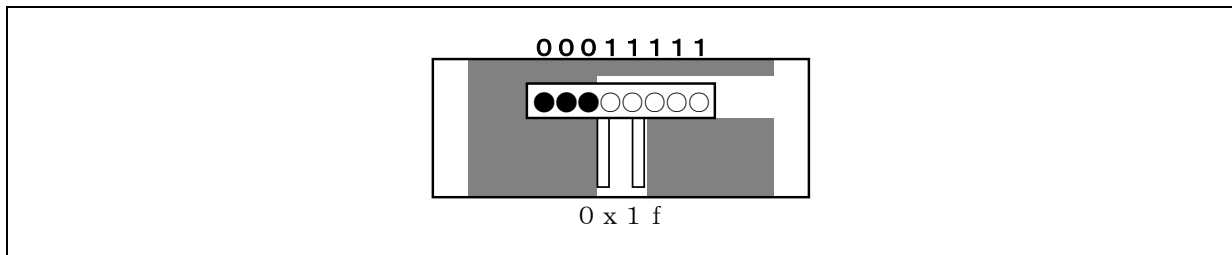
右クランクと判断するセンサ状態  
 である 0x1f ではないので、  
 そのまま進む!!

脱輪!!

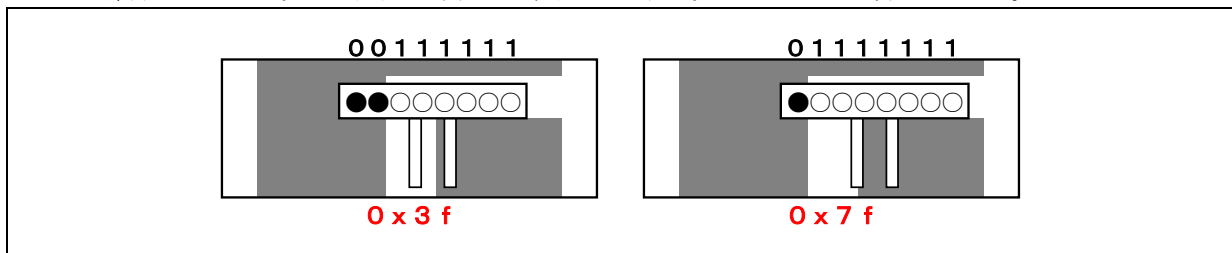
プログラムを見てみます。

```
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 );
    motor( 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 );
    motor( 50 , 10 );
    pattern = 41;
    cnt1 = 0;
    break;
  }
  switch( sensor_inp(MASK3_3) ) {
    case 0x00:
      /* センタ→まっすぐ */
      handle( 0 );
      motor( 40 , 40 );
      break;
    case 0x04:
    case 0x06:
    case 0x07:
    case 0x03:
      /* 左寄り→右曲げ */
      handle( 8 );
      motor( 40 , 35 );
      break;
    case 0x20:
    case 0x60:
    case 0xe0:
    case 0xc0:
      /* 右寄り→左曲げ */
      handle( -8 );
      motor( 35 , 40 );
      break;
  }
  break;
```

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



データ解析でセンサの状態を何度か確認して、下図のような状態があることが分かりました。



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

```

/*****
/* メインプログラム
/*****
void main( void )
{
    int    ret;
    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 );
            motor( 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 );
            motor( 50 ,10 );
            pattern = 41;
            cnt1 = 0;
            break;
        }
        switch( sensor_inp(MASK3_3) ) {
            case 0x00:
                /* センタ→まっすぐ */
                handle( 0 );
                motor( 40 ,40 );
                break;
            case 0x04:
            case 0x06:
            case 0x07:
            case 0x03:
                /* 左寄り→右曲げ */
                handle( 8 );
                motor( 40 ,35 );
                break;
            case 0x20:
            case 0x60:
            case 0xe0:
            case 0xc0:
                /* 右寄り→左曲げ */
                handle( -8 );
                motor( 35 ,40 );
                break;
        }
    }
    break;
}

```

この追加を行うことで、右クランクをクリアすることができました。

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

## 9. プロジェクト「msd\_fat11\_38a」 microSD にデータ記録(FAT32 版)

### 9.1 概要

このプログラムは、

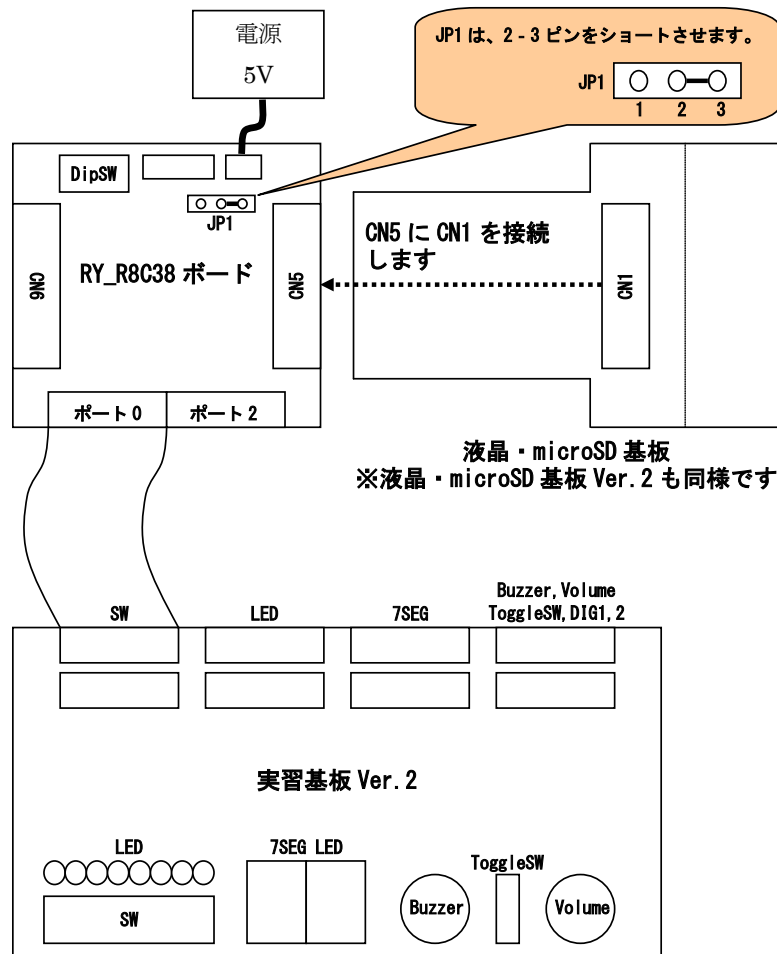
- ポート 0 に接続されているディップスイッチの値
- RY\_R8C38 ボードのディップスイッチの値

を、10ms ごとに **FAT32 形式で microSD へ書き込みます**。microSD に書き込み処理を行っていても 10ms ごとの記録は続けます。

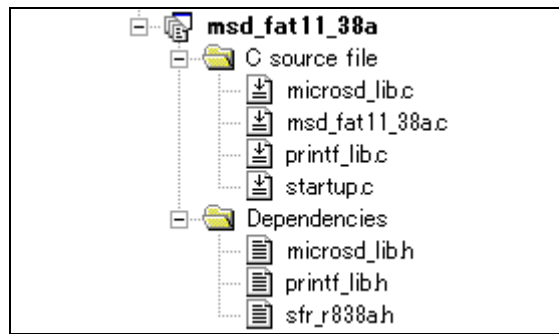
記録終了後、microSD を Windows などで読み込んで、書き込まれたファイルを開いてみてください。

### 9.2 接続

- RY\_R8C38 ボードの CN5(ポート 3、ポート 5、ポート 6)と、液晶・microSD 基板の CN1 のコネクタを重ね合わせて接続します。
- RY\_R8C38 ボードのポート 0 と、実習基板 Ver.2 のスイッチ部分をフラットケーブルで接続します。  
**※ポート 0 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。**



### 9.3 プロジェクトの構成



	ファイル名	内容
1	microsd_lib.c	microSD 制御ライブラリです。microSD を使用する場合は、このファイルを追加します。 ファイルの位置→C:\¥Workspace¥common_r8c38a¥microsd_lib.c
2	msd_fat11_38a.c	実際に制御するプログラムが書かれています。R8C/38A 内蔵周辺機能(SFR)の初期化も行います。 ファイルの位置→C:\¥Workspace¥kit12msd_38a¥msd_fat01_38a¥msd_fat01_38a.c
3	printf_lib.c	通信をするための設定、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。 ファイルの位置→C:\¥Workspace¥common_r8c38a¥printf_lib.c
4	startup.c	固定割り込みベクタアドレスの設定、スタートアッププログラム、RAM の初期化(初期値のないグローバル変数、初期値のあるグローバル変数の設定)などを行います。 ファイルの位置→C:\¥Workspace¥kit12msd_38a¥msd_fat01_38a¥startup.c
5	microsd_lib.h	microSD 制御ライブラリのヘッダファイルです。 ファイルの位置→C:\¥Workspace¥common_r8c38a¥microsd_lib.h
6	printf_lib.h	printf、scanf 制御ライブラリのヘッダファイルです。 ファイルの位置→C:\¥Workspace¥common_r8c38a¥printf_lib.h
7	sfr_r838a.h	R8C/38A マイコンの内蔵周辺機能を制御するためのレジスタ(Special Function Register)を定義したファイルです。 ファイルの位置→C:\¥Workspace¥common_r8c38a¥sfr_r838a.h

### 9.4 プログラム

プログラムのゴシック体部分が、microSD 書き込み(FAT32 対応版)の部分です。

```

1 : /*****
2 : /* 対象マイコン R8C/38A */
3 : /* ファイル内容 microSD基板の実験(microSDへファイルとして書き込み) */
4 : /* バージョン Ver. 1. 00 */
5 : /* Date 2013. 04. 24 */
6 : /* Copyright ジャパンマイコンカーラー実行委員会 */
7 : *****/
8 :
9 : /*
10 : 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。
11 : ・ポート0のデータ
12 : ・マイコンボード上のディップスイッチの値
13 : FAT32でフォーマットしたmicroSDに、ファイルとして書き込みます。
14 : */
15 :
```

## 9. プロジェクト「msd\_fat11\_38a」 microSD にデータ記録(FAT32 版)

```

16 : /*=====*/
17 : /* インクルード */
18 : /*=====*/
19 : #include <stdio.h>
20 : #include "sfr_r838a.h" /* R8C/38A SFRの定義ファイル */
21 : #include "printf_lib.h" /* printf使用ライブラリ */
22 : #include "microsd_lib.h" /* microSD制御ライブラリ */
23 :
24 : /*=====*/
25 : /* プロトタイプ宣言 */
26 : /*=====*/
27 : void init( void );
28 : unsigned char dipsw_get( void );
29 :
30 : /*=====*/
31 : /* グローバル変数の宣言 */
32 : /*=====*/
33 : const char *C_DATE = __DATE__; /* コンパイルした日付 */
34 : const char *C_TIME = __TIME__; /* コンパイルした時間 */
35 :
36 : unsigned long cnt1; /* 時間計測用 */
37 : int pattern; /* パターン番号 */
38 : int countDown; /* 表示作業用 */
39 :
40 : /* microSD関連変数 */
41 : int msdFlag; /* 1:データ記録 0:記録しない */
42 : int msdTimer; /* 取得間隔計算用 */
43 :
44 : /*=====*/
45 : /* メインプログラム */
46 : /*=====*/
47 : void main( void )
48 : {
49 :     int i, ret;
50 :     char fileName[ 8+1+3+1 ]; /* 名前8字+'.'+拡張子3字+'%' */
51 :
52 :     init(); /* SFRの初期化 */
53 :     init_uart0_printf( SPEED_9600 ); /* UART0とprintf関連の初期化 */
54 :     setMicroSDLedPort( &p6, &pd6, 0 ); /* microSD モニタLED設定 */
55 :     asm(" fset I "); /* 全体の割り込み許可 */
56 :
57 :     /* microSD初期化 */
58 :     ret = initMicroSD();
59 :     if( ret != 0x00 ) {
60 :         printf( "\n" );
61 :         printf( "microSD Initialize Error!!\n" ); /* 初期化できず */
62 :         printf( "(Error Code = %d)\n", ret );
63 :         pattern = 99;
64 :     } else {
65 :         printf( "\n" );
66 :         printf( "microSD Initialize OK!!\n" ); /* 初期化完了 */
67 :     }
68 :
69 :     /* FAT32でマウント */
70 :     if( ret == 0x00 ) {
71 :         ret = mountMicroSD_FAT32();
72 :         if( ret != 0x00 ) {
73 :             printf( "\n" );
74 :             printf( "microSDはFAT32のフォーマットではありません。 \n" );
75 :             printf( "FAT32でフォーマットしてください。 \n" );
76 :             printf( "(Error Code = %d)\n", ret );
77 :             pattern = 99;
78 :         } else {
79 :             printf( "\n" );
80 :             printf( "microSDはFAT32フォーマットです。 \n" );
81 :         }
82 :     }
83 :
84 :     /* 書き込みファイル名作成 */
85 :     if( ret == 0x00 ) {
86 :         i = readMicroSDNumber(); /* microSDの空き領域から番号読み込み */
87 :         if( i == -1 ) {
88 :             printf( "microSDから書き込み番号が読めません。 \n" );
89 :             ret = 1;
90 :             pattern = 99;
91 :         } else {
92 :             i++;
93 :             if( i >= 10000 ) i = 1;
94 :             writeMicroSDNumber( i ); /* microSDの空き領域へ番号書き込み */
95 :
96 :             /* ファイル名変換 */
97 :             sprintf( fileName, "test%04d.csv", i );
98 :         }
99 :     }
100 : }

```

```
101 : /* ファイル名のセット、領域確保 */
102 : if( ret == 0x00 ) {
103 :     /* ファイルのタイムスタンプセット */
104 :     setDateStamp( getCompileYear( C_DATE ),
105 :                  getCompileMonth( C_DATE ), getCompileDay( C_DATE ) );
106 :     setTimeStamp( getCompileHour( C_TIME ),
107 :                  getCompilerMinute( C_TIME ), getCompilerSecond( C_TIME ) );
108 :
109 :     /* ファイル名の確認 */
110 :     printf( "ファイルを開いて、領域の確保中です。:" );
111 :     for( i=0; fileName[i]!='\0'; i++) printf( "%c", fileName[i] );
112 :     printf( "\n" );
113 :
114 :     /* ファイル名のセット、領域確保 */
115 :     /* 書き込みしたい時間[ms] : x = 10[ms] : 23バイト(保存バイト数) */
116 :     /* 10000ms(10秒)なら、x = 10000 * 23 / 10 = 23000 */
117 :     /* 結果は512の倍数になるように繰り上げる。 */
118 :     /* また、最初のメッセージ分として+512しておく */
119 :     ret = writeFile( fileName, 23552 );
120 :     if( ret != 0x00 ) {
121 :         printf( "ファイルが開けません。(Error Code = %d)\n", ret );
122 :         pattern = 99;
123 :     } else {
124 :         printf( "ファイルを開きました。:\n" );
125 :         printf( "\n" );
126 :
127 :         /* microSD書き込み */
128 :         msdPrintf( "msd_fat11_38a Log Data\n" );
129 :         while( checkMsdPrintf() ); // msdPrintf処理完了待ち
130 :         msdPrintf( "Compile Date:" );
131 :         while( checkMsdPrintf() ); // msdPrintf処理完了待ち
132 :         msdPrintf( C_DATE );
133 :         while( checkMsdPrintf() ); // msdPrintf処理完了待ち
134 :         msdPrintf( " Time:" );
135 :         while( checkMsdPrintf() ); // msdPrintf処理完了待ち
136 :         msdPrintf( C_TIME );
137 :         while( checkMsdPrintf() ); // msdPrintf処理完了待ち
138 :         msdPrintf( "\n\nLineNo. ポート0, デイプススイッチ\n" );
139 :         while( checkMsdPrintf() ); // msdPrintf処理完了待ち
140 :     }
141 : }
142 :
143 : while( 1 ) {
144 :
145 :     switch( pattern ) {
146 :     case 0:
147 :         /* タイトル転送、準備 */
148 :         printf( "3秒後から、ポート0の値と"
149 :                " デイプススイッチの値を記録します。:\n" );
150 :         printf( "\n" );
151 :         printf( "Ready " );
152 :         pattern = 1;
153 :         cnt1 = 0;
154 :         break;
155 :
156 :     case 1:
157 :         /* カウントダウン表示 */
158 :         if( cnt1 / 1000 != countDown ) {
159 :             countDown = cnt1 / 1000;
160 :             if( cnt1 / 1000 == 4 ) { /* 4秒たったら開始 */
161 :                 pattern = 2;
162 :                 break;
163 :             }
164 :             printf( "%d ", 3 - countDown );
165 :         }
166 :         break;
167 :
168 :     case 2:
169 :         /* データ記録開始 */
170 :         printf( "\n" );
171 :         printf( "Data recording " );
172 :         msdFlag = 1; /* データ記録開始 */
173 :         pattern = 3;
174 :         cnt1 = 0;
175 :         break;
176 :
177 :     case 3:
178 :         /* データ記録中 記録は割り込みの中で行う */
179 :         /* 書き込み終了時間になると、割り込み内でmsdFlagが0になる */
180 :         if( msdFlag == 0 ) {
181 :             pattern = 4;
182 :             break;
183 :         }
184 :
185 :         /* 時間表示 */
186 :         if( cnt1 / 1000 != countDown ) {
187 :             countDown = cnt1 / 1000;
188 :             printf( "%d ", countDown );
189 :         }
190 :         break;
191 :     }
```

## 9. プロジェクト「msd\_fat11\_38a」 microSD にデータ記録(FAT32 版)

```

192 :     case 4:
193 :         /* 最後のデータが書き込まれるまで待つ*/
194 :         if( microSDProcessEnd() == 0 ) {
195 :             pattern = 5;
196 :         }
197 :         break;
198 :
199 :     case 5:
200 :         /* 終了メッセージ表示 */
201 :         printf( "%n\n" );
202 :         printf( "End. %n" );
203 :         pattern = 99;
204 :         break;
205 :
206 :     case 99:
207 :         /* 終了 */
208 :         break;
209 :
210 :     default:
211 :         /* どれでもない場合は待機状態に戻す */
212 :         pattern = 0;
213 :         break;
214 :     }
215 : }
216 : }
217 :
218 : /*****
219 : /* R8C/38A スペシャルファンクションレジスタ (SFR) の初期化 */
220 : /*****
221 : void init( void )
222 : {
223 :     int i;
224 :
225 :     /* クロックをXINクロック (20MHz)に変更 */
226 :     prc0 = 1; /* プロテクト解除 */
227 :     cm13 = 1; /* P4_6, P4_7をXIN-XOUT端子にする*/
228 :     cm05 = 0; /* XINクロック発振 */
229 :     for(i=0; i<50; i++ ); /* 安定するまで少し待つ(約10ms) */
230 :     ocd2 = 0; /* システムクロックをXINにする */
231 :     prc0 = 0; /* プロテクトON */
232 :
233 :     /* ポートの入出力設定 */
234 :     prc2 = 1; /* PD0のプロテクト解除 */
235 :     pd0 = 0x00; /* */
236 :     pd1 = 0xd0; /* 5:RXD0 4:TXD0 3-0:DIP SW */
237 :     pd2 = 0xff; /* 7-0:LED */
238 :     pd3 = 0xff; /* */
239 :     p4 = 0x20; /* P4_5のLED:初期は点灯 */
240 :     pd4 = 0xb8; /* 7:XOUT 6:XIN 5:LED 2:VREF */
241 :     pd5 = 0x7f; /* 7-0:LCD/microSD基板 */
242 :     pd6 = 0xef; /* 4-0:LCD/microSD基板 */
243 :     pd7 = 0xff; /* */
244 :     pd8 = 0xff; /* */
245 :     pd9 = 0x3f; /* */
246 :     pur0 = 0x04; /* P1_3~P1_0のプルアップON */
247 :
248 :     /* タイマRBの設定 */
249 :     /* 割り込み周期 = 1 / 20[MHz] * (TRBPRE+1) * (TRBPR+1)
250 :     = 1 / (20*10-6) * 200 * 100
251 :     = 0.001[s] = 1[ms]
252 :
253 :     /* 動作モード、分周比設定 */
254 :     trbmr = 0x00; /* プリスケールレジスタ */
255 :     trbpre = 200-1; /* */
256 :     trbpr = 100-1; /* プライマリレジスタ */
257 :     trbic = 0x07; /* 割り込み優先レベル設定 */
258 :     trbcr = 0x01; /* カウント開始 */
259 : }

```



```

260 : /******
261 : /* タイマRB 割り込み処理
262 : /******
263 : #pragma interrupt intTRB(vect=24)
264 : void intTRB( void )
265 : {
266 :     static int line_no;          /* 行番号
267 :     int ret;
268 :
269 :     cnt1++;
270 :
271 :     /* microSD間欠書き込み処理(1msごとに実行) */
272 :     microSDProcess();
273 :
274 :     /* microSD記録処理 */
275 :     if( msdFlag == 1 ) {
276 :         /* 記録間隔のチェック */
277 :         msdTimer++;
278 :         if( msdTimer >= 10 ) {
279 :             msdTimer = 0;
280 :
281 :             ret = msdPrintf( "%4d,%8b", %4x%r%r\n",
282 :                 line_no,          // 行番号
283 :                 p0,              // ポート0
284 :                 dipsw_get()      // ディップスイッチ
285 :             );
286 :             if( ret == 2 ) msdFlag = 0;
287 :
288 :             if( ++line_no >= 10000 ) line_no = 0;
289 :         }
290 :     }
291 : }
292 :
293 : /*
294 : ●msdPrintf使用の注意点
295 :
296 : ・microSDに展開される文字数は、1行(CR(%r), LF(%n)を含めて)64文字まで。
297 : ・引数は、20個程度まで。
298 : ・msdPrintf関数は、10msごとに実行する(10ms間で64文字まで)。
299 : ・10ms以下でログを記録したい場合は、変数に値を保存しておき、
300 :   msdPrintf関数を実行するときに、まとめて出力する。
301 :   例) msdPrintf( "%3d%3d%r%r%3d%3d%r%r\n", s1, m1, s2, m2 );
302 :       s1とm1:5ms前の値、s2とm2:今回の値
303 : ・msdPrintf関数の戻り値が0ならセット完了、0以外なら前のデータを
304 :   書き込み中で、今回のデータは書き込みできず。
305 :   例) ret = msdPrintf( "%5d%r%r\n", i );
306 :       while( checkMsdPrintf() ); // msdPrintf処理完了待ち
307 : */
308 :
309 : /******
310 : /* ディップスイッチ値読み込み
311 : /* 戻り値 スイッチ値 0~15
312 : /******
313 : unsigned char dipsw_get( void )
314 : {
315 :     unsigned char sw;
316 :
317 :     sw = p1 & 0x0f;          /* P1_3~P1_0読み込み
318 :
319 :     return sw;
320 : }
321 :
322 : /******
323 : /* end of file
324 : /******
    
```

## 9.5 FAT32 形式で microSD ヘデータを書き込む

※FAT については、ウィキペディア (Wikipedia): フリー百科事典「File Allocation Table」(最終更新 2011 年 9 月 6 日 (火) 12:48)を引用しています。

**今回、走行データを書き込んだ microSD を、パソコンからも読み込めるように FAT32 形式に対応しました。**

FAT(ファット)とは「ファイル・アロケーション・テーブル(File Allocation Table)」の略で、ファイルシステムにおけるディスク内のファイルの位置情報などを記録するための領域を指します。現在では、FAT を用いるファイルシステムとしても FAT と呼ぶようになっています。

Windows は FAT をサポートしているので、R8C/38A マイコンも FAT の形式に合わせて microSD にデータを書き込めば、Windows からデータを読めるようになります。

FAT には、クラスタ番号の管理ビット数によって「FAT12」、「FAT16」、「FAT32」、「exFAT」などがあります。**今回のプログラムで対応しているのは、「FAT32」のみです。それ以外の FAT はエラーになりますので、FAT32 で microSD をフォーマットして使用してください。**

microSD への書き込みで、microSD をメモリとして書き込む場合(FAT 非対応)と、FAT32 に対応させて書き込む場合の特徴を、下表に示します。

	メモリとして書き込む場合 (FAT 非対応)	※	FAT32 形式で書き込む場合
R8C/38A マイコンの プログラム容量	メモリとして書き込むだけなので、少ない容量で可能	>	FAT32 形式に対応させるプログラムが必要
走行中の microSD への 書き込み内容	データを無加工で書き込む (文字列に変換は、走行後とパソコンへの転送時に行う)	>	アスキーデータに変換する必要がある ので、負荷が大きい (割り込み内で処理を分散させて対応)
パソコンへの 転送作業	通信ケーブルを接続して、 TeraTerm などの通信ソフトの操作が必要 (ただし、プログラムの書き込みを行う場合は、データ転送後、すぐに書き込みができる)	<=	microSD の抜き差しが必要
通信ソフト	必要 通信ソフトが COM を使用していると、プログラムの書き込みができない	<	通信ソフトを使わない
パソコンへの 転送時間	シリアル通信で転送するので、転送に時間がかかる(最大で数分程度)	<<<	USB-SD カード変換アダプタを通して ファイルとして読み込むので時間は 0
ログの保存	TeraTerm などの通信ソフトのログ操作で、ファイル名を付けてデータを保存、管理	<	microSD に書き込まれたファイル名で管理

※A>B で「A の方が扱いやすい、性能がよい」、A<B で「B の方が扱いやすい、性能がよい」という意味です。

FAT32 形式に対応させると、プログラムの容量や、走行中の microSD への書き込み負荷が多くなりますが、それ以上にメリットが多くなります。

これから、microSD への書き込みを FAT32 に対応させたプログラムについて、説明します。

## 9.6 プログラムの解説

### 9.6.1 変数

30 :	/*=====*/		
31 :	/* グローバル変数の宣言		*/
32 :	/*=====*/		
33 :	const char *C_DATE = __DATE__;	/* コンパイルした日付	*/
34 :	const char *C_TIME = __TIME__;	/* コンパイルした時間	*/
35 :			
36 :	unsigned long cnt1;	/* 時間計測用	*/
37 :	int pattern;	/* パターン番号	*/
38 :	int countDown;	/* 表示作業用	*/
39 :			
40 :	/* microSD 関連変数 */		
41 :	int msdFlag;	/* 1:データ記録 0:記録しない	*/
42 :	int msdTimer;	/* 取得間隔計算用	*/

microSD にファイルとして書き込むにあたり、新たにグローバル変数を追加しています。各変数の役割を下表に示します。

変数名	内容
*C_DATE	<p>コンパイル(ビルド)した日付の文字列データを格納しています。                      microSD にファイルとして書き込むとき、マイコンカーはカレンダー情報を持っていません。そのため、ビルドしたときの日付を、microSD へ書き込むファイルの日付とします。                      変更する必要がないので、const を付けて、ROM に配置します(const を付けないと、RAM にも配置されて RAM 容量を使ってしまいます)。                      例えば、2012 年 4 月 23 日なら、下記のような文字列になります。</p> <p>「Apr_23_2012」 ※_は、スペース</p>
*C_TIME	<p>コンパイル(ビルド)した時刻の文字列データを格納しています。                      microSD にファイルとして書き込むとき、マイコンカーはカレンダー情報を持っていません。そのため、ビルドしたときの時刻を、microSD へ書き込むファイルの時刻とします。                      変更する必要がないので、const を付けて、ROM に配置します(const を付けないと、RAM にも配置されて RAM 容量を使ってしまいます)。                      例えば、12 時 34 分 56 秒なら、下記のような文字列になります。</p> <p>「12:34:56」</p>

### 9.6.2 main 関数 (microSD の初期化)

```

44 : /*****
45 : /* メインプログラム
46 : /*****
47 : void main( void )
48 : {
49 :     int    i, ret;
50 :     char   fileName[ 8+1+3+1 ];      /* 名前 8 字+'.'+拡張子 3 字+'¥0' */
51 :
52 :     init();                          /* SFR の初期化
53 :     init_uart0_printf( SPEED_9600 ); /* UART0 と printf 関連の初期化
54 :     setMicroSDLedPort( &p6, &pd6, 0 ); /* microSD モニタ LED 設定
55 :     asm( " fset I " );                /* 全体の割り込み許可
56 :
57 :     /* microSD 初期化 */
58 :     ret = initMicroSD();
59 :     if( ret != 0x00 ) { 0 以外ならエラー
60 :         printf( "¥n" );
61 :         printf( "microSD Initialize Error!!¥n" ); /* 初期化できず
62 :         printf( "(Error Code = %d)¥n", ret );
63 :         pattern = 99;
64 :     } else {
65 :         printf( "¥n" );
66 :         printf( "microSD Initialize OK!!¥n" ); /* 初期化完了
67 :     }
    
```

50 行目	microSD へファイルとして書き込むときの、ファイル名を格納する配列です。 名前 8 文字+ピリオド 1 文字+拡張子 3 文字+終端文字('¥0')の合計 13 文字分、確保します。
58 行目 ～ 67 行目	initMicroSD 関数で microSD を初期化します。 microSD と正常にやり取りができなければエラーメッセージを表示させて、パターン 99 へ移りプログラムを終了します。

### 9.6.3 main 関数 (FAT32 でマウント)

```

69 :     /* FAT32 でマウント */
70 :     if( ret == 0x00 ) {
71 :         ret = mountMicroSD_FAT32();
72 :         if( ret != 0x00 ) { 0 以外ならエラー
73 :             printf( "¥n" );
74 :             printf( "microSD は FAT32 のフォーマットではありません。¥n" );
75 :             printf( "FAT32 でフォーマットしてください。¥n" );
76 :             printf( "(Error Code = %d)¥n", ret );
77 :             pattern = 99;
78 :         } else {
79 :             printf( "¥n" );
80 :             printf( "microSD は FAT32 フォーマットです。¥n" );
81 :         }
82 :     }
83 :
84 :     /* 書き込みファイル名作成 */
85 :     if( ret == 0x00 ) {
86 :         i = readMicroSDNumber(); /* microSD の空き領域から番号読み込み*/
    
```

```
87 :         if( i == -1 ) { -1 なら以外ならエラー
88 :             printf( "microSD から書き込み番号が読めません。¥n" );
89 :             ret = 1;
90 :             pattern = 99;
91 :         } else {
92 :             i++;
93 :             if( i >= 10000 ) i = 1;
94 :             writeMicroSDNumber( i ); /* microSD の空き領域へ番号書き込み */
95 :
96 :             /* ファイル名変換 */
97 :             sprintf( fileName, "test%04d.csv", i ); ファイル名変換
98 :         }
99 :     }
100 :
101 :     /* ファイル名のセット、領域確保 */
102 :     if( ret == 0x00 ) {
103 :         /* ファイルのタイムスタンプセット */
104 :         setDateStamp( getCompileYear( C_DATE ),
105 :                     getCompileMonth( C_DATE ), getCompileDay( C_DATE ) );
106 :         setTimeStamp( getCompileHour( C_TIME ),
107 :                     getCompilerMinute( C_TIME ), getCompilerSecond( C_TIME ) );
108 :
109 :         /* ファイル名の確認 */
110 :         printf( "ファイルを開いて、領域の確保中です。:" );
111 :         for( i=0; fileName[i]!='¥0'; i++ ) printf( "%c", fileName[i] );
112 :         printf( "¥n" );
113 :
114 :         /* ファイル名のセット、領域確保 */
115 :         /* 書き込みしたい時間[ms] : x = 10[ms] : 23 バイト(保存バイト数) */
116 :         /* 10000ms(10 秒)なら、x = 10000 * 23 / 10 = 23000 */
117 :         /* 結果は 512 の倍数になるように繰り上げする。 */
118 :         /* また、最初のメッセージ分として+512 しておく */
119 :         ret = writeFile( fileName, 23552 );
120 :         if( ret != 0x00 ) { 0 以外ならエラー
121 :             printf( "ファイルが開けません。(Error Code = %d)¥n", ret );
122 :             pattern = 99;
123 :         } else {
124 :             printf( "ファイルを開きました。¥n" );
125 :             printf( "¥n" );
126 :
127 :             /* microSD 書き込み */
128 :             msdPrintf( "msd_fat11_38a Log Data¥n" );
129 :             while( checkMsdPrintf() ); // msdPrintf 処理完了待ち
130 :             msdPrintf( "Compile Date:" );
131 :             while( checkMsdPrintf() ); // msdPrintf 処理完了待ち
132 :             msdPrintf( C_DATE );
133 :             while( checkMsdPrintf() ); // msdPrintf 処理完了待ち
134 :             msdPrintf( " Time:" );
135 :             while( checkMsdPrintf() ); // msdPrintf 処理完了待ち
136 :             msdPrintf( C_TIME );
137 :             while( checkMsdPrintf() ); // msdPrintf 処理完了待ち
138 :             msdPrintf( "¥n¥nLineNo, ポート 0, ディップスイッチ¥n" );
139 :             while( checkMsdPrintf() ); // msdPrintf 処理完了待ち
140 :         }
141 :     }
```

9. プロジェクト「msd\_fat11\_38a」 microSD にデータ記録(FAT32 版)

70 行目 ～ 82 行目	mountMicroSD_FAT32 関数で、microSD から FAT32 情報を読み込みます。 <b>microSD が FAT32 以外でフォーマットされている場合はエラーになります。</b> Windows など、FAT 32 でフォーマットしてください。
85 行目 ～ 99 行目	ファイル名を連番にするために、前回書き込んだ番号を読み込み、今回の番号を保存します。 86 行の readMicroSDNumber 関数で、microSD の空き領域から前回書き込んだファイル番号を読み込みます。 92 行で 1 つ大きい値にして、今回のファイル名の番号にします。 94 行の writeMicroSDNumber 関数で、次に備えて今回の番号を保存しておきます。 97 行目で fileName 配列にファイル名を設定します。今回は「test0000.csv」で、「0000」部分の数字が、書き込むたびに増えていきます。ファイル名を変えたい場合はここで変えますが、 <b>ファイル名の長さは、8 文字以内+ピリオド+拡張子 3 文字以内にしてください。</b>
104 行目 ～ 107 行目	microSD にファイルとして書き込むとき、マイコンはカレンダー情報を持っていません。そのため、ビルドしたときの日付、時刻を、microSD へ書き込むファイルの日付、時刻とします。 setDateStamp 関数で、年月日を設定します。年月日情報が保存されている C_DATE 配列は、文字列として情報を持っているので、これらを int 型に変換する関数で変換して設定しています。 setTimeStamp 関数で、時分秒を設定します。時分秒情報が保存されている C_TIME 配列は、文字列として情報を持っているので、これらを int 型に変換する関数で変換して設定しています。
110 行目 ～ 112 行目	microSD に書き込むファイル名を printf 文で通信ソフトに表示しています。書き込むファイル名の確認用なので、この部分は無くても構いません。
115 行目 ～ 126 行目	119 行の writeFile 関数で、microSD に保存するファイル名と書き込む容量を指定して、FAT32 領域を確保します。 今回、データの記録条件を次のようにしました。  <ul style="list-style-type: none"> <li>・データ記録の間隔 … 10ms ごと</li> <li>・データ記録数 …………… 23 バイト (内容は後述します)</li> <li>・データ記録時間 ……… 10 秒(10000ms)</li> </ul> microSD に確保する容量は、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。  $= 10000 \div 10 \times 23$ $= 23000$ 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。  $23000 \div 512 = 44 \text{ 余り } 472$ 割り切れないので、512 の倍数で切り上げます。また、最初に分かりやすいようにコメントを書くのでその分を加えます。100 文字くらいですが、最小単位は 512 なので、512 を加えます。よって、  $512 \times 45 + 512 = 23552$ となります。 ファイル名は、fileName 配列に設定しているので、この配列名を writeFile 関数の引数にします。 writeFile 関数の戻り値が 0 なら、microSD にファイル名の登録、容量の確保が完了です。0 以外ならエラーとなります。
128 行目 ～ 139 行目	書き込むファイルの最初に、ビルドしたときの日付、時間、書き込む列の内容を書き込みます。 microSD への書き込みには msdPrintf 文を使います (詳しくは後述します)。 <b>msdPrintf 文で書き込んだ後、次に書き込むには、①最大時間の 10ms 待つ ②checkMsdPrintf 関数で書き込みが終わったか確認する の二通りの方法があります。</b> 今回は、②で確認します。 msdPrintfMode 関数の戻り値が 0 なら処理完了、0 以外は処理中です。よって、while 文で 0 になるまで待つようにします。while の行で最大 10ms 間止まるので走行中はこのような記述はできませんが、走行前なので 10ms 程度の待ち時間は問題ありません。

#### 9.6.4 パターン 0:タイトル表示

パターン 0 はメッセージを表示します。表示後、パターン 1 へ移ります。

```
143 :     while( 1 ) {
144 :
145 :         switch( pattern ) {
146 :             case 0:
147 :                 /* タイトル転送、準備 */
148 :                 printf( "3 秒後から、ポート 0 の値と"
149 :                     "ディップスイッチの値を記録します。¥n" );
150 :                 printf( "¥n" );
151 :                 printf( "Ready " );
152 :                 pattern = 1;
153 :                 cnt1 = 0;
154 :                 break;
```

#### 9.6.5 パターン 1:タイトル表示

3 秒待ち、3 秒経ったならパターン 2 に移ります。待っているだけだと、何もしていないように思えるので、1 秒ごとにカウントダウンして値を表示します。

```
156 :         case 1:
157 :             /* カウントダウン表示 */
158 :             if( cnt1 / 1000 != countDown ) {
159 :                 countDown = cnt1 / 1000;
160 :                 if( cnt1 / 1000 == 4 ) { /* 4 秒たったら開始 */
161 :                     pattern = 2;
162 :                     break;
163 :                 }
164 :                 printf( "%d ", 3 - countDown );
165 :             }
166 :             break;
```

#### 9.6.6 パターン 2:データ記録開始

msdFlag を 1 にして、データ記録を開始します。

```
168 :         case 2:
169 :             /* データ記録開始 */
170 :             printf( "¥n" );
171 :             printf( "Data recording " );
172 :             msdFlag = 1; /* データ記録開始 */
173 :             pattern = 3;
174 :             cnt1 = 0;
175 :             break;
```

### 9.6.7 パターン 3: データ記録中

データ記録中です。writeFile 関数で指定した容量分、書き込むと、割り込みプログラム内で msdFlag を 0 にするので、ここでは msdFlag が 0 になったかを確認します。0 になったなら、パターン 4 へ移ります。データ記録中は時間経過が分かるように、1 秒ごとに時間を表示します。

```
177 :     case 3:
178 :         /* データ記録中 記録は割り込みの中で行う */
179 :         /* 書き込み終了時間になると、割り込み内で msdFlag が 0 になる */
180 :         if( msdFlag == 0 ) {
181 :             pattern = 4;
182 :             break;
183 :         }
184 :
185 :         /* 時間表示 */
186 :         if( cnt1 / 1000 != countDown ) {
187 :             countDown = cnt1 / 1000;
188 :             printf( "%d ", countDown );
189 :         }
190 :         break;
```

### 9.6.8 パターン 4、5、99: 終了処理

msdFlag が 0 になった後、最後のデータを書き込むまで待ちます。この作業は microSDProcessEnd 関数を実行して、戻り値が 0 以外なら書き込み中、0 なら最後のデータ書き込み終了と判断してパターン 5 へ移ります。

パターン 5 では、printf 文で "End" を出力して終わったことを知らせ、パターン 99 に移ります。

パターン 99 は、無限ループで電源が切れるかリセットされるまで待ちます。

```
192 :     case 4:
193 :         /* 最後のデータが書き込まれるまで待つ*/
194 :         if( microSDProcessEnd() == 0 ) {
195 :             pattern = 5;
196 :         }
197 :         break;
198 :
199 :     case 5:
200 :         /* 終了メッセージ表示 */
201 :         printf( "\n\n" );
202 :         printf( "End. \n" );
203 :         pattern = 99;
204 :         break;
205 :
206 :     case 99:
207 :         /* 終了 */
208 :         break;
```



### 9.6.9 割り込み処理

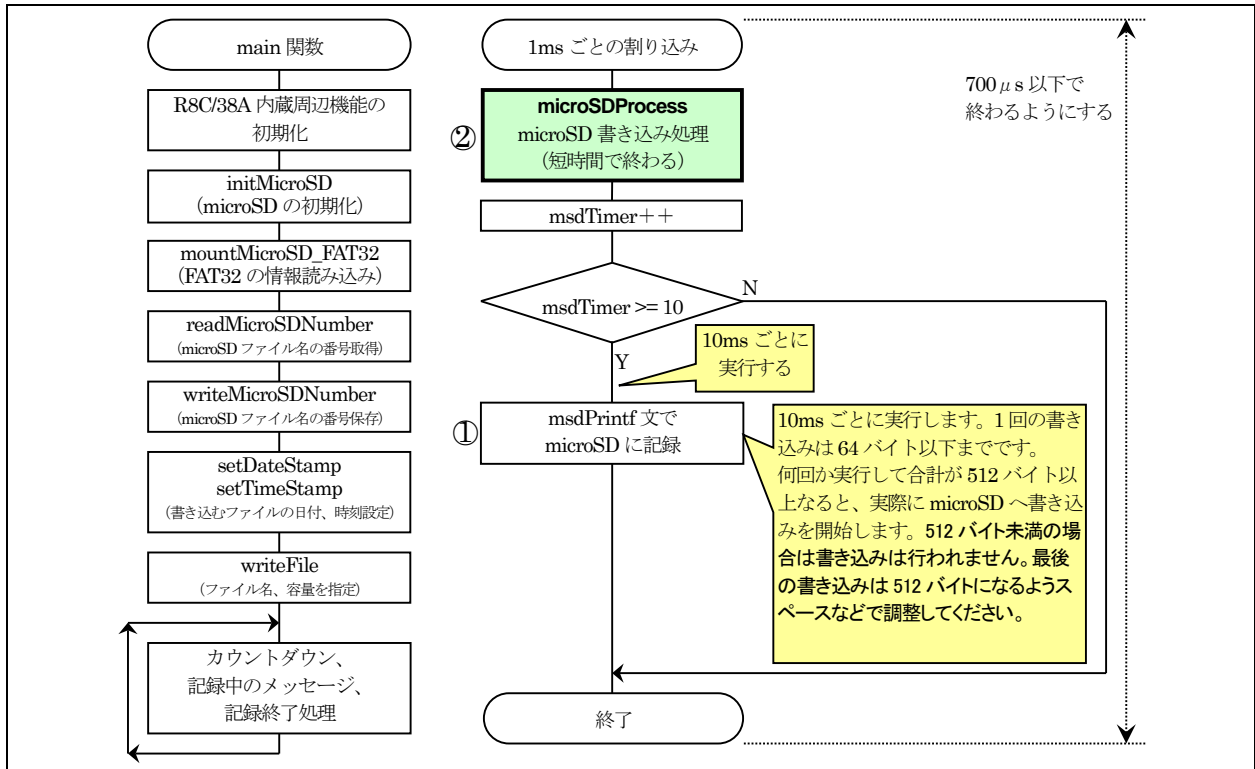
1ms ごとの割り込み処理です。この中で、microSD に記録する文字列を作り、microSD に記録します。

```

263 : #pragma interrupt intTRB(vect=24)
264 : void intTRB( void )
265 : {
266 :     static int line_no;           /* 行番号          */
267 :     int ret;
268 :
269 :     cnt1++;
270 :
271 :     /* microSD 間欠書き込み処理(1ms ごとに実行)  */
272 :     microSDProcess();
273 :
274 :     /* microSD 記録処理 */
275 :     if( msdFlag == 1 ) {
276 :         /* 記録間隔のチェック */
277 :         msdTimer++;
278 :         if( msdTimer >= 10 ) {
279 :             msdTimer = 0;           このカッコの中は 10ms に 1 回実行
280 :
281 :             ret = msdPrintf( "%4d,=%x%8b%", %4x%r%yn",
282 :                 line_no,           // 行番号
283 :                 p0,               // ポート 0
284 :                 dipsw_get()      // ディップスイッチ
285 :             );
286 :             if( ret != 0 ) msdFlag = 0;
287 :
288 :             if( ++line_no >= 10000 ) line_no = 0;
289 :         }
290 :     }
291 : }
    
```

266 行目	line_no という変数を設定しています。関数内で「static」を付けた変数を静的変数といい、関数が終了しても値を保持します。関数内のグローバル変数のようなイメージです。その関数内でしか使わな いけれども値を保持したい場合は、静的変数にします。
278 行目	今回のプログラムは、microSD への記録を 10ms ごとに行います。割り込みは 1ms ごとなので、 msdTimer 変数を割り込みごとに+1 して、10 になったなら 0 に戻します。
281 行目	msdPrintf 関数で microSD へデータを記録します。 今回は、①line_no ②ポート 0 の値 ③ディップスイッチの値 を記録します。
286 行目	msdPrintf 関数の戻り値をチェックします。 戻り値の内容を下記に示します。 0 : 成功 1 : 書き込み中で書き込みできず 2 : 書き込み中止(ファイルクローズ) writeFile 関数で指定した容量以上になると、ファイルをクローズするので、戻り値が 2 になります。記 録開始後、約 10 秒経つと指定容量になり戻り値が 2 になります。今回は戻り値が 0 以外なら、 msdFlag を 0 にします。 msdFlag が 0 になると、275 行が成り立たないので、次回以降は msdPrintf 文を実行しません。
288 行目	line_no 変数を+1 します。line_no 変数の値を microSD に書き込みますが、この変数は int 型なので 32767 以上にはできません。また今回 line_no 変数の記録は 4 桁なので、5 桁(10000 以上)になつた ら 0 に戻します。

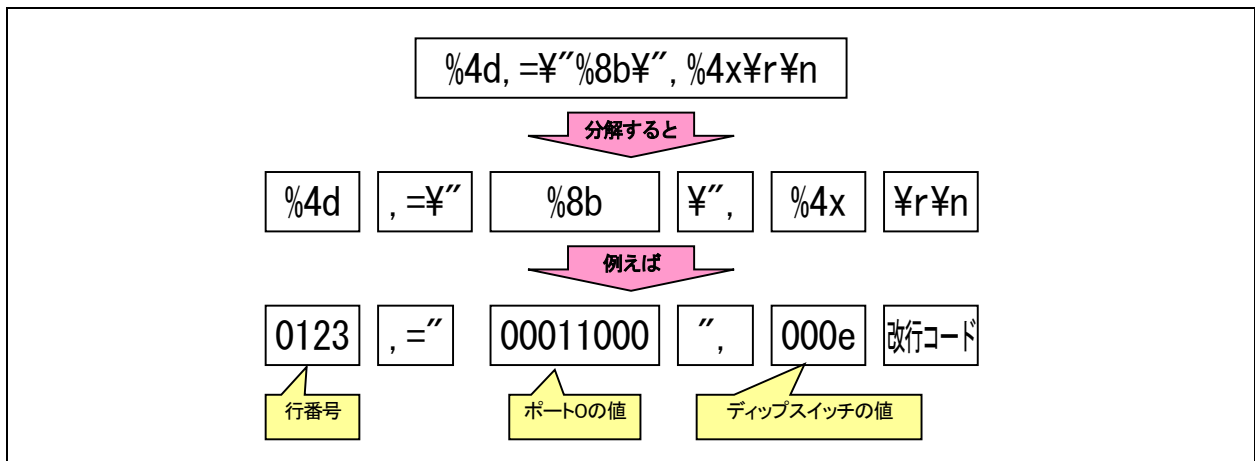
main 関数と割り込みプログラムのフローチャートを、下記に示します。



	msdTimer が 10(以上)なら、msdPrintf 関数で microSD にデータを記録します。 <b>正確には記録する準備をしているだけです。</b>
①	msdPrintf 関数を複数回実行して合計が 512 バイト以上になると、実際に microSD へ書き込みを開始します。 <b>512 バイト未満の場合は書き込みは行われません。最後の書き込みは 512 バイトになるようスペースなどで調整してください。</b>
②	msdPrintf 関数で記録する内容が合計 512 バイトになると <b>microSDProcess 関数</b> で実際の書き込み処理を行います。

### 9.6.10 記録する内容

microSD へ記録する書式を下記に示します。



今回のプログラムは、10ms ごとに 23 バイトの文字列を記録します。記録例を、下記に示します。

```
0001, ="00000001", 000f
0002, ="00000011", 000f
0003, ="00001110", 000e
0004, ="00111100", 000e
0005, ="11110000", 000d
0006, ="11100000", 000d
0007, ="10000000", 000c
0008, ="00000000", 000c
```

#### 9.6.11 記録できる文字数と記録間隔について

**microSD は、80ms 間隔で 512 バイト記録できます。**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。

ファイルとして保存する場合、1 行ごとに改行コードを入れます。改行コードは、CR コード(0x0d)と LF コード(0x0a)の 2 文字なので、この分を引いた残りが、記録できる文字列になります。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	実際の記録数	備考
80ms	512 バイト以下	510 バイト以下	
40ms	256 バイト以下	254 バイト以下	
20ms	128 バイト以下	126 バイト以下	
<b>10ms</b>	<b>64 バイト以下</b>	<b>62 バイト以下</b>	今回の記録間隔です。
5ms	32 バイト以下	30 バイト以下	
2ms	12 バイト以下	10 バイト以下	
1ms	6 バイト以下	4 バイト以下	

msdPrintf 関数の実行は、10ms 以上間隔を空けて実行してください。

記録間隔を 5ms にしたときのプログラム例を下記に示します。5ms 前の値を保存する変数を用意して、出力するときに 5ms 前と現在の値を出力するようにします。

```
#pragma interrupt intTRB(vect=24)
void intTRB( void )
{
    static int line_no;           /* 行番号 */
    static int line_no_old;       5ms 前の行番号を保存
    static unsigned char p0_old, dipsw_old; 5ms 前のポート0、ディップスイッチ値を保存
    int ret;

    cnt1++;

    /* microSD 間欠書き込み処理(1ms ごとに実行) */
    microSDProcess();

    /* microSD 記録処理 */
    if( msdFlag == 1 ) {
        /* 記録間隔のチェック */
        msdTimer++;
        if( msdTimer == 5 ) { 5ms 前の値として3つの値を保存する
            line_no_old = line_no;
            p0_old = p0;
            dipsw_old = dipsw_get();
            if( ++line_no >= 10000 ) line_no = 0;
        } else if( msdTimer >= 10 ) {
            msdTimer = 0;

            ↑この部分が5ms前の値をmicroSDに出力する部分です
            ret = msdPrintf( "%4d,=%¥"%8b¥", %4x¥r¥n",
                line_no_old, // 5ms 前の行番号
                p0_old, // 5ms 前のポート0
                dipsw_old, // 5ms 前のディップスイッチ
                line_no, // 行番号
                p0, // ポート0
                dipsw_get() // ディップスイッチ
            );
            if( ret == 2 ) msdFlag = 0; // ファイルクローズなら終了
            if( ++line_no >= 10000 ) line_no = 0;
        }
    }
}
```

## 10. プロジェクト「kit12msd\_fat11\_38a」 走行データを microSD に記録(FAT32 対応版)

### 10.1 概要

マイコンカーの走行中のデータを、microSD (FAT32 対応版) 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

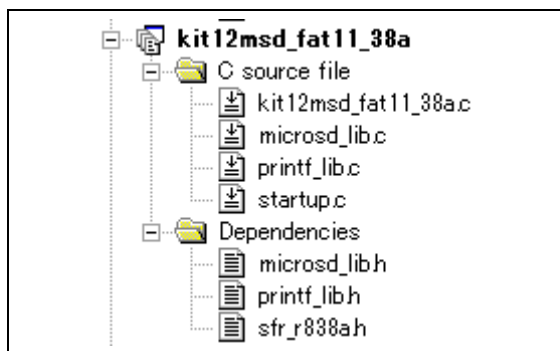
これらのデータを、走行開始から 10ms ごとに 60 秒間記録します。60 秒間経った場合は、データの記録は止めますが、走行はそのまま行います。

走行後、microSD をパソコンに挿して、Windows のメモ帳やエクセルで走行データを読み込むことができます。この情報を基に、プログラムのデバッグに役立てます。

### 10.2 接続

「8.2 マイコンカーの構成」と同じです。

### 10.3 プロジェクトの構成



	ファイル名	内容
1	microsd_lib.c	microSD 制御ライブラリです。microSD を使用する場合は、このファイルを追加します。 ファイルの位置→C:\¥Workspace¥common_r8c38a¥microsd_lib.c
2	kit12msd_fat01_38a.c	実際に制御するプログラムが書かれています。R8C/38A 内蔵周辺機能 (SFR) の初期化も行います。 ファイルの位置→C:\¥Workspace¥kit12msd_38a¥kit12msd_fat01_38a¥kit12msd_fat01_38a.c
3	printf_lib.c	通信をするための設定、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。 <b>※msdPrintf 文を使用する場合は、プロジェクトに「printf_lib.c」ファイルを追加してください。「printf_lib.c」が無い場合は、コンパイルエラーになります。</b> ファイルの位置→C:\¥Workspace¥common_r8c38a¥printf_lib.c

10. プロジェクト「kit12msd\_fat11\_38a」 走行データを microSD に記録(FAT32 対応版)

4	startup.c	固定割り込みベクタアドレスの設定、スタートアッププログラム、RAM の初期化(初期値のないグローバル変数、初期値のあるグローバル変数の設定)などを行います。 ファイルの位置→C:\¥Workspace¥kit12msd_38a¥kit12msd_fat01_38a¥startup.c
5	microsd_lib.h	microSD 制御ライブラリのヘッダファイルです。 ファイルの位置→C:\¥Workspace¥common_r8c38a¥microsd_lib.h
6	printf_lib.h	printf、scanf 制御ライブラリのヘッダファイルです。 ファイルの位置→C:\¥Workspace¥common_r8c38a¥printf_lib.h
7	sfr_r838a.h	R8C/38A マイコンの内蔵周辺機能を制御するためのレジスタ (Special Function Register)を定義したファイルです。 ファイルの位置→C:\¥Workspace¥common_r8c38a¥sfr_r838a.h

10.4 プログラム

プログラムのゴシック体部分が、microSD 書き込み(FAT32 版)の部分です。

```

1 : /*****
2 : /* 対象マイコン R8C/38A */
3 : /* ファイル内容 microSDを使ったマイコンカートレスプログラム msdPrintf使用版(R8C/38A版) */
4 : /* バージョン Ver. 1.00 */
5 : /* Date 2013.04.24 */
6 : /* Copyright ジャパンマイコンカーラー実行委員会 */
7 : *****/
8 :
9 : /*
10 : 本プログラムは、「kit12_38a.c」にmicroSDによる走行データ保存(ファイルとして)
11 : 追加したプログラムです。次のデータをファイルとしてmicroSDに保存します。
12 : ・パターン番号 ・センサの状態
13 : ・ハンドル角度 ・左モータPWM値 ・右モータPWM値
14 : */
15 :
16 : /*=====*/
17 : /* インクルード */
18 : /*=====*/
19 : #include <stdio.h>
20 : #include "sfr_r838a.h" /* R8C/38A SFRの定義ファイル */
21 : #include "printf_lib.h" /* printf使用ライブラリ */
22 : #include "microsd_lib.h" /* microSD制御ライブラリ */
23 :
24 : /*=====*/
25 : /* シンボル定義 */
26 : /*=====*/
27 : /* 定数設定 */
28 : #define PWM_CYCLE 39999 /* モータPWMの周期 */
29 : #define SERVO_CENTER 3750 /* サーボのセンタ値 */
30 : #define HANDLE_STEP 22 /* 1°分の値 */
31 :
32 : /* マスク値設定 ×:マスクあり(無効) ○:マスク無し(有効) */
33 : #define MASK2_2 0x66 /* ×○○××○○× */
34 : #define MASK2_0 0x60 /* ×○○××××× */
35 : #define MASK0_2 0x06 /* ×××××○○× */
36 : #define MASK3_3 0xe7 /* ○○○××○○○ */
37 : #define MASK0_3 0x07 /* ×××××○○○ */
38 : #define MASK3_0 0xe0 /* ○○○××××× */
39 : #define MASK4_0 0xf0 /* ○○○○×××× */
40 : #define MASK0_4 0x0f /* ××××○○○○ */
41 : #define MASK4_4 0xff /* ○○○○○○○○ */
42 :
43 : /*=====*/
44 : /* プロトタイプ宣言 */
45 : /*=====*/
46 : void init( void );
47 : void timer( unsigned long timer_set );
48 : int check_crossline( void );
49 : int check_rightline( void );
50 : int check_leftline( void );
51 : unsigned char sensor_inp( unsigned char mask );
52 : unsigned char dipsw_get( void );
53 : unsigned char pushsw_get( void );
54 : unsigned char startbar_get( void );
55 : void led_out( unsigned char led );
56 : void motor( int accele_l, int accele_r );
57 : void handle( int angle );
58 :

```

```

59 : /*=====*/
60 : /* グローバル変数の宣言 */
61 : /*=====*/
62 : const char *C_DATE = _DATE_; /* コンパイルした日付 */
63 : const char *C_TIME = _TIME_; /* コンパイルした時間 */
64 :
65 : unsigned long cnt0; /* timer関数用 */
66 : unsigned long cnt1; /* main内で使用 */
67 : int pattern; /* パターン番号 */
68 :
69 : /* microSD関連変数 */
70 : int msdFlag; /* 1:データ記録 0:記録しない */
71 : int msdTimer; /* 取得間隔計算用 */
72 : int msdError; /* エラー番号記録 */
73 :
74 : /* 現在の状態保存用 */
75 : int handleBuff; /* 現在のハンドル角度記録 */
76 : int leftMotorBuff; /* 現在の左モータPWM値記録 */
77 : int rightMotorBuff; /* 現在の右モータPWM値記録 */
78 :
79 : /*=====*/
80 : /* メインプログラム */
81 : /*=====*/
82 : void main( void )
83 : {
84 :     int i, ret;
85 :     char fileName[ 8+1+3+1 ]; /* 名前+'.'+拡張子+'%0' */
86 :
87 :     /* マイコン機能の初期化 */
88 :     init(); /* 初期化 */
89 :     init_uart0_printf( SPEED_9600 ); /* UART0とprintf関連の初期化 */
90 :     setMicroSDLedPort( &p6, &pd6, 0 ); /* microSD モニタLED設定 */
91 :     asm( " fset I " ); /* 全体の割り込み許可 */
92 :
93 :     /* microSD初期化 */
94 :     ret = initMicroSD();
95 :     if( ret != 0x00 ) msdError = 1;
96 :
97 :     /* FAT32でマウント */
98 :     if( msdError == 0 ) {
99 :         ret = mountMicroSD_FAT32();
100 :         if( ret != 0x00 ) msdError = 2;
101 :     }
102 :
103 :     if( msdError != 0 ) {
104 :         /* microSD処理にエラーがあれば3秒間、LEDの点灯方法を変える */
105 :         cnt1 = 0;
106 :         while( cnt1 < 3000 ) {
107 :             if( cnt1 % 200 < 100 ) {
108 :                 led_out( 0x3 );
109 :             } else {
110 :                 led_out( 0x0 );
111 :             }
112 :         }
113 :     }
114 :
115 :     /* マイコンカーの状態初期化 */
116 :     handle( 0 );
117 :     motor( 0, 0 );
118 :
119 :     while( 1 ) {
120 :         switch( pattern ) {
121 :
122 :         /*=====*/
123 :         パターンについて
124 :         0 : スイッチ入力待ち
125 :         1 : スタートバーが開いたかチェック
126 :         11 : 通常トレース
127 :         12 : 右へ大曲げの終わりのチェック
128 :         13 : 左へ大曲げの終わりのチェック
129 :         21 : クロスライン検出時の処理
130 :         22 : クロスラインを読み飛ばす
131 :         23 : クロスライン後のトレース、クランク検出
132 :         31 : 左クランククリア処理 安定するまで少し待つ
133 :         32 : 左クランククリア処理 曲げ終わりのチェック
134 :         41 : 右クランククリア処理 安定するまで少し待つ
135 :         42 : 右クランククリア処理 曲げ終わりのチェック
136 :         51 : 右ハーフライン検出時の処理
137 :         52 : 右ハーフラインを読み飛ばす
138 :         53 : 右ハーフライン後のトレース、レーンチェンジ
139 :         54 : 右レーンチェンジ終了のチェック
140 :         61 : 左ハーフライン検出時の処理
141 :         62 : 左ハーフラインを読み飛ばす
142 :         63 : 左ハーフライン後のトレース、レーンチェンジ
143 :         64 : 左レーンチェンジ終了のチェック
144 :         /*=====*/
145 :
    
```

## 10. プロジェクト「kit12msd\_fat11\_38a」 走行データを microSD に記録(FAT32 対応版)

```

146 :     case 0:
147 :         /* スイッチ入力待ち */
148 :         if( pushsw_get() ) {
149 :             led_out( 0x0 );
150 :
151 :             if( msdError == 0 ) {
152 :                 /* microSDの空き領域から読み込み */
153 :                 i = readMicroSDNumber();
154 :                 if( i == -1 ) {
155 :                     msdError = 3;
156 :                 }
157 :             }
158 :             if( msdError == 0 ) {
159 :                 /* microSDの空き領域へ書き込み */
160 :                 i++;
161 :                 if( i >= 10000 ) i = 1;
162 :                 ret = writeMicroSDNumber( i );
163 :                 if( ret == -1 ) {
164 :                     msdError = 4;
165 :                 } else {
166 :                     /* ファイル名変換 */
167 :                     sprintf( fileName, "log_%04d.csv", i );
168 :                 }
169 :             }
170 :             if( msdError == 0 ) {
171 :                 /* ファイルのタイムスタンプセット */
172 :                 setDateStamp( getCompileYear( C_DATE ),
173 :                               getCompileMonth( C_DATE ), getCompileDay( C_DATE ) );
174 :                 setTimeStamp( getCompileHour( C_TIME ),
175 :                               getCompilerMinute( C_TIME ), getCompilerSecond( C_TIME ) );
176 :
177 :                 /* 書き込みファイル名作成 */
178 :                 // 書き込みしたい時間[ms] : x = 10[ms] : 64バイト
179 :                 // 6000msなら、x = 60000 * 64 / 10 = 384000
180 :                 // 結果は512の倍数になるように繰り上げる。
181 :                 ret = writeFile( fileName, 384000 );
182 :                 if( ret != 0x00 ) msdError = 11;
183 :
184 :                 // microSD書き込み
185 :                 msdPrintf( "[Your Car Name] Log Data\n" );
186 :                 while( checkMsdPrintf() ); // msdPrintf処理完了待ち
187 :                 msdPrintf( "Compile Date:" );
188 :                 while( checkMsdPrintf() ); // msdPrintf処理完了待ち
189 :                 msdPrintf( C_DATE );
190 :                 while( checkMsdPrintf() ); // msdPrintf処理完了待ち
191 :                 msdPrintf( " Time:" );
192 :                 while( checkMsdPrintf() ); // msdPrintf処理完了待ち
193 :                 msdPrintf( C_TIME );
194 :                 while( checkMsdPrintf() ); // msdPrintf処理完了待ち
195 :                 msdPrintf( "\n\nLineNo, Pattern, Sensor,"
196 :                               "\nハンドル, 左モータ, 右モータ\n" );
197 :                 while( checkMsdPrintf() ); // msdPrintf処理完了待ち
198 :             }
199 :             pattern = 1;
200 :             cnt1 = 0;
201 :             break;
202 :         }
203 :
204 :         if( cnt1 < 100 ) { /* LED点滅処理 */ /*
205 :             led_out( 0x1 );
206 :         } else if( cnt1 < 200 ) {
207 :             led_out( 0x2 );
208 :         } else {
209 :             cnt1 = 0;
210 :         }
211 :         break;
212 :
213 :     case 1:
214 :         /* スタートバーが開いたかチェック */
215 :         if( !startbar_get() ) {
216 :             /* スタート!! */
217 :             led_out( 0x0 );
218 :             pattern = 11;
219 :             if( msdError == 0 ) msdFlag = 1; /* データ記録開始 */ /*
220 :             cnt1 = 0;
221 :             break;
222 :         }
223 :         if( cnt1 < 50 ) { /* LED点滅処理 */ /*
224 :             led_out( 0x1 );
225 :         } else if( cnt1 < 100 ) {
226 :             led_out( 0x2 );
227 :         } else {
228 :             cnt1 = 0;
229 :         }
230 :         break;

```

中略



```
566 :     case 101:
567 :         /* microSDの停止処理 */
568 :         /* 脱輪した際の自動停止処理後は、必ずこの処理を行ってください */
569 :         handle( 0 );
570 :         motor( 0, 0 );
571 :         msdFlag = 0;
572 :         pattern = 102;
573 :         break;
574 :
575 :     case 102:
576 :         /* 最後のデータが書き込まれるまで待つ */
577 :         if( microSDProcessEnd() == 0 ) {
578 :             pattern = 103;
579 :         }
580 :         break;
581 :
582 :     case 103:
583 :         /* 書き込み終了 */
584 :         led_out( 0x3 );
585 :         break;
586 :
587 :     default:
588 :         /* どれでもない場合は待機状態に戻す */
589 :         pattern = 0;
590 :         break;
591 :     }
592 : }
593 : }
594 :
```

中略

```
655 : /******
656 : /* タイマRB 割り込み処理 */
657 : /******
658 : #pragma interrupt intTRB(vect=24)
659 : void intTRB( void )
660 : {
661 :     static int line_no;          /* 行番号 */
662 :     int ret;
663 :
664 :     cnt0++;
665 :     cnt1++;
666 :
667 :     /* microSD間欠書き込み処理(1msごとに実行) */
668 :     microSDProcess();
669 :
670 :     /* microSD記録処理 */
671 :     if( msdFlag == 1 ) {
672 :         /* 記録間隔のチェック */
673 :         msdTimer++;
674 :         if( msdTimer >= 10 ) {
675 :             msdTimer = 0;
676 :
677 :             msdPrintf( "%4d,%3d,=%8b%", %3d, %4d, %4d%r%r",
678 :                 line_no,          // 行番号
679 :                 pattern,          // パターン番号
680 :                 sensor_inp(0xff), // センサ情報(8bit)
681 :                 handleBuff,      // ハンドル値
682 :                 leftMotorBuff,   // 左モータ値
683 :                 rightMotorBuff   // 右モータ値
684 :             );
685 :             if( ++line_no >= 10000 ) line_no = 0;
686 :         }
687 :     }
}
```

以下、略

## 10.5 プログラムの解説

### 10.5.1 変数

```

59 : /*=====*/
60 : /* グローバル変数の宣言 */
61 : /*=====*/
62 : const char *C_DATE = __DATE__; /* コンパイルした日付 */
63 : const char *C_TIME = __TIME__; /* コンパイルした時間 */
64 :
65 : unsigned long cnt0; /* timer 関数用 */
66 : unsigned long cnt1; /* main 内で使用 */
67 : int pattern; /* パターン番号 */
68 :
69 : /* microSD 関連変数 */
70 : int msdFlag; /* 1:データ記録 0:記録しない */
71 : int msdTimer; /* 取得間隔計算用 */
72 : int msdError; /* エラー番号記録 */
73 :
74 : /* 現在の状態保存用 */
75 : int handleBuff; /* 現在のハンドル角度記録 */
76 : int leftMotorBuff; /* 現在の左モータ PWM 値記録 */
77 : int rightMotorBuff; /* 現在の右モータ PWM 値記録 */
    
```

microSD にファイルとして書き込むにあたり、新たにグローバル変数を追加しています。各変数の役割を下表に示します。

変数名	内容
*C_DATE	コンパイル(ビルド)した日付の文字列データを格納しています。 microSD にファイルとして書き込むとき、マイコンカーはカレンダー情報を持っていません。そのため、ビルドしたときの日付を、microSD へ書き込むファイルの日付とします。変更する必要がないので、const を付けて、ROM に配置します(const を付けないと、RAM にも配置されて RAM 容量を使ってしまいます)。
*C_TIME	コンパイル(ビルド)した時刻の文字列データを格納しています。 microSD にファイルとして書き込むとき、マイコンカーはカレンダー情報を持っていません。そのため、ビルドしたときの時刻を、microSD へ書き込むファイルの時刻とします。変更する必要がないので、const を付けて、ROM に配置します(const を付けないと、RAM にも配置されて RAM 容量を使ってしまいます)。
handleBuff	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。

### 10.5.2 main 関数(microSD の初期化)

```

79 : /*****
80 : /* メインプログラム
81 : *****/
82 : void main( void )
83 : {
84 :     int    i, ret;
85 :     char   fileName[ 8+1+3+1 ];      /* 名前+'.'+拡張子+'¥0' */
86 :
87 :     /* マイコン機能の初期化 */
88 :     init();                          /* 初期化 */
89 :     init_uart0_printf( SPEED_9600 ); /* UART0 と printf 関連の初期化 */
90 :     setMicroSDLedPort( &p6, &pd6, 0 ); /* microSD モニタ LED 設定 */
91 :     asm( " fset I " );                /* 全体の割り込み許可 */
92 :
93 :     /* microSD 初期化 */
94 :     ret = initMicroSD();
95 :     if( ret != 0x00 ) msdError = 1;
96 :
97 :     /* FAT32 でマウント */
98 :     if( msdError == 0 ) {
99 :         ret = mountMicroSD_FAT32();
100 :         if( ret != 0x00 ) msdError = 2;
101 :     }
102 :
103 :     if( msdError != 0 ) {
104 :         /* microSD 処理にエラーがあれば 3 秒間、LED の点灯方法を変える */
105 :         cnt1 = 0;
106 :         while( cnt1 < 3000 ) {
107 :             if( cnt1 % 200 < 100 ) {
108 :                 led_out( 0x3 );
109 :             } else {
110 :                 led_out( 0x0 );
111 :             }
112 :         }
113 :     }

```

85 行目	microSD へファイルとして書き込むときの、ファイル名を格納する配列です。 名前 8 文字+ピリオド 1 文字+拡張子 3 文字+終端文字('¥0')の合計 13 文字分、確保します。
94 行目	initMicroSD 関数で microSD を初期化します。
99 行目	mountMicroSD_FAT32 関数で、microSD から FAT32 情報を読み込みます。 <b>microSD が FAT32 以外でフォーマットされている場合はエラーになります。</b> Windows などで FAT32 形式でフォーマットしてください。
103 行目 ～ 113 行目	microSD でエラーがあれば、モータドライブ基板の LED 2 個を 3 秒間点滅させ、エラーがあることを知らせます。microSD エラーがあっても、走りには影響ありません。

### 10.5.3 パターン 0:スイッチ入力待ち

```
146 :     case 0:
147 :         /* スイッチ入力待ち */
148 :         if( pushsw_get() ) {
149 :             led_out( 0x0 );
150 :
151 :             if( msdError == 0 ) {
152 :                 /* microSD の空き領域から読み込み */
153 :                 i = readMicroSDNumber();
154 :                 if( i == -1 ) {
155 :                     msdError = 3;
156 :                 }
157 :             }
158 :             if( msdError == 0 ) {
159 :                 /* microSD の空き領域へ書き込み */
160 :                 i++;
161 :                 if( i >= 10000 ) i = 1;
162 :                 ret = writeMicroSDNumber( i );
163 :                 if( ret == -1 ) {
164 :                     msdError = 4;
165 :                 } else {
166 :                     /* ファイル名変換 */
167 :                     sprintf( fileName, "log_%04d.csv", i );
168 :                 }
169 :             }
170 :             if( msdError == 0 ) {
171 :                 /* ファイルのタイムスタンプセット */
172 :                 setDateStamp( getCompileYear( C_DATE ),
173 :                     getCompileMonth( C_DATE ), getCompileDay( C_DATE ) );
174 :                 setTimeStamp( getCompileHour( C_TIME ),
175 :                     getCompilerMinute( C_TIME ), getCompilerSecond( C_TIME ) );
176 :
177 :                 /* 書き込みファイル名作成 */
178 :                 // 書き込みしたい時間[ms] : x = 10[ms] : 64 バイト
179 :                 // 60000ms なら、x = 60000 * 64 / 10 = 384000
180 :                 // 結果は 512 の倍数になるように繰り上げる。
181 :                 ret = writeFile( fileName, 384000 );
182 :                 if( ret != 0x00 ) msdError = 11;
183 :
184 :                 // microSD 書き込み
185 :                 msdPrintf( "[Your Car Name] Log Data\r\n" );
186 :                 while( checkMsdPrintf() ); // msdPrintf 処理完了待ち
187 :                 msdPrintf( "Compile Date:" );
188 :                 while( checkMsdPrintf() ); // msdPrintf 処理完了待ち
189 :                 msdPrintf( C_DATE );
190 :                 while( checkMsdPrintf() ); // msdPrintf 処理完了待ち
191 :                 msdPrintf( " Time:" );
192 :                 while( checkMsdPrintf() ); // msdPrintf 処理完了待ち
193 :                 msdPrintf( C_TIME );
194 :                 while( checkMsdPrintf() ); // msdPrintf 処理完了待ち
195 :                 msdPrintf( "\r\nLineNo, Pattern, Sensor, "
196 :                     "ハンドル, 左モータ, 右モータ\r\n" );
197 :                 while( checkMsdPrintf() ); // msdPrintf 処理完了待ち
198 :             }
```

```

199 :         pattern = 1;
200 :         cnt1 = 0;
201 :         break;
202 :     }
203 :
204 :     if( cnt1 < 100 ) {                /* LED 点滅処理                */
205 :         led_out( 0x1 );
206 :     } else if( cnt1 < 200 ) {
207 :         led_out( 0x2 );
208 :     } else {
209 :         cnt1 = 0;
210 :     }
211 :     break;
    
```

151 行目 ~ 169 行目	ファイル名を連番にするために、前回書き込んだ番号を読み込み、今回の番号を保存します。 153 行の readMicroSDNumber 関数で、microSD の空き領域から前回書き込んだファイル番号を読み込みます。 160 行で 1 つ大きい値にして、今回のファイル名の番号にします。 162 行の writeMicroSDNumber 関数で、次に備えて今回の番号を保存しておきます。 167 行目で fileName 配列にファイル名を設定します。今回は「log_0000.csv」で、「0000」部分の数字が、書き込むたびに増えていきます。ファイル名を変えたい場合はここで変えますが、 <b>ファイル名の長さは、8 文字以内+ピリオド+拡張子 3 文字以内にしてください。</b>
172 行目 ~ 175 行目	microSD にファイルとして書き込むとき、マイコンはカレンダー情報を持っていません。そのため、ビルドしたときの日付、時刻を、microSD へ書き込むファイルの日付、時刻とします。 setDateStamp 関数で、年月日を設定します。年月日情報が保存されている C_DATE 配列は、文字列として情報を持っているので、これらを int 型に変換する関数で変換して設定しています。 setTimeStamp 関数で、時分秒を設定します。時分秒情報が保存されている C_TIME 配列は、文字列として情報を持っているので、これらを int 型に変換する関数で変換して設定しています。
181 行目	writeFile 関数で、microSD に保存するファイル名と、書き込む容量を指定して FAT32 領域を確保します。 今回、データの記録条件を次のようにしました。 <ul style="list-style-type: none"> <li>・データ記録の間隔 … 10ms ごと</li> <li>・データ記録数 …………… 64 バイト (実際はもっと少ないです)</li> <li>・データ記録時間 ……… 60 秒(60000ms)</li> </ul> microSD に確保する容量は、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。 $\begin{aligned} &= 60000 \div 10 \times 64 \\ &= 384000 \end{aligned}$ 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。 $384000 \div 512 = 750 \text{ 余り } 0$ 割り切れますので、この値で問題ありません。 ファイル名は、fileName 配列に設定しているので、この配列名を writeFile 関数の引数にします。 writeFile 関数の戻り値が 0 なら、microSD にファイル名の登録、容量の確保が完了です。0 以外ならエラーとなります。

10. プロジェクト「kit12msd\_fat11\_38a」 走行データを microSD に記録(FAT32 対応版)

184 行目 ～ 197 行目	<p>走行データを記録する前に、</p> <ul style="list-style-type: none"> <li>・カーネーム</li> <li>・コンパイルした日時</li> <li>・列の内容</li> </ul> <p>を書き込んでおきます。 microSD への書き込みには msdPrintf 文を使います。 <b>msdPrintf 文で書き込んだ後、次に書き込むには、①最大時間の 10ms 待つ ②checkMsdPrintf 関数で書き込みが終わったか確認する の二通りの方法があります。</b>今回は、②で確認します。 msdPrintfMode 関数の戻り値が 0 なら処理完了、0 以外は処理中です。よって、while 文で 0 になるまで待つようにします。while の行で最大 10ms 間止まるので走行中はこのような記述はできませんが、走行前なので 10ms 程度の待ち時間は問題ありません。 今回、下記の内容を書き込みます。</p> <pre>[Your Car Name] Log Data Compile Date:xxx xx xxxx Time:xx:xx:xx</pre> <p style="text-align: right; border: 1px solid black; padding: 2px;">xx は日時で変わります</p> <p>LineNo, Pattern, Sensor, ハンドル, 左モータ, 右モータ</p>
-----------------------	---

10.5.4 パターン 1:スタートバーが開いたかチェック

スタートバーが開いたかチェックします。開いたならば 216～220 行を実行し、パターン 11 へ移ります。

213 : 214 : 215 : 216 : 217 : 218 : <b>219 :</b> 220 : 221 : 222 : 223 : 224 : 225 : 226 : 227 : 228 : 229 : 230 :	<pre>case 1:     /* スタートバーが開いたかチェック */     if( !startbar_get() ) {         /* スタート！！ */         led_out( 0x0 );         pattern = 11; <b>if( msdError == 0 ) msdFlag = 1; /* データ記録開始 */</b>         cnt1 = 0;         break;     }     if( cnt1 &lt; 50 ) { /* LED 点滅処理 */         led_out( 0x1 );     } else if( cnt1 &lt; 100 ) {         led_out( 0x2 );     } else {         cnt1 = 0;     }     break;</pre>
---	---

219 行目	msdError 変数が 0 なら、microSD の使用準備が整っていますので、msdFlag 変数を 1 にしてデータ記録処理を開始します。データの記録自体は、割り込みプログラム内で行います。
--------	--

### 10.5.5 パターン 101~103: microSD 終了処理

microSD へ記録中に電源を落とすと、書き込んだデータが microSD に保存されないことがあります。これは書き込みが終わった microSD は、書き込み終了処理をしなければいけないためです。

脱輪したら自動的に停止するなどの走行を終了させるプログラムを追加した場合、パターン 101 に移して、microSD の終了処理を必ず行ってください。

```
566 :     case 101:
567 :         /* microSD の停止処理 */
568 :         /* 脱輪した際の自動停止処理後は、必ずこの処理を行ってください */
569 :         handle( 0 );
570 :         motor( 0, 0 );
571 :         msdFlag = 0;
572 :         pattern = 102;
573 :         break;
574 :
575 :     case 102:
576 :         /* 最後のデータが書き込まれるまで待つ */
577 :         if( microSDProcessEnd() == 0 ) { 戻り値が0ならmicroSDへの書き込み完了
578 :             pattern = 103;
579 :         }
580 :         break;
581 :
582 :     case 103:
583 :         /* 書き込み終了 */
584 :         led_out( 0x3 );
585 :         break;
```

### 10.5.6 割り込み処理

1ms ごとの割り込み処理です。この中で、microSD に記録する文字列を作り、microSD に記録します。

```
658 : #pragma interrupt intTRB(vect=24)
659 : void intTRB( void )
660 : {
661 :     static int line_no;          /* 行番号          */
662 :     int ret;
663 :
664 :     cnt0++;
665 :     cnt1++;
666 :
667 :     /* microSD 間欠書き込み処理(1ms ごとに実行) */
668 :     microSDProcess();
669 :
670 :     /* microSD 記録処理 */
671 :     if( msdFlag == 1 ) {
672 :         /* 記録間隔のチェック */
673 :         msdTimer++;
674 :         if( msdTimer >= 10 ) { 10になったらmicroSDに記録するかチェック
675 :             msdTimer = 0;
676 :
```

10. プロジェクト「kit12msd\_fat11\_38a」 走行データを microSD に記録(FAT32 対応版)

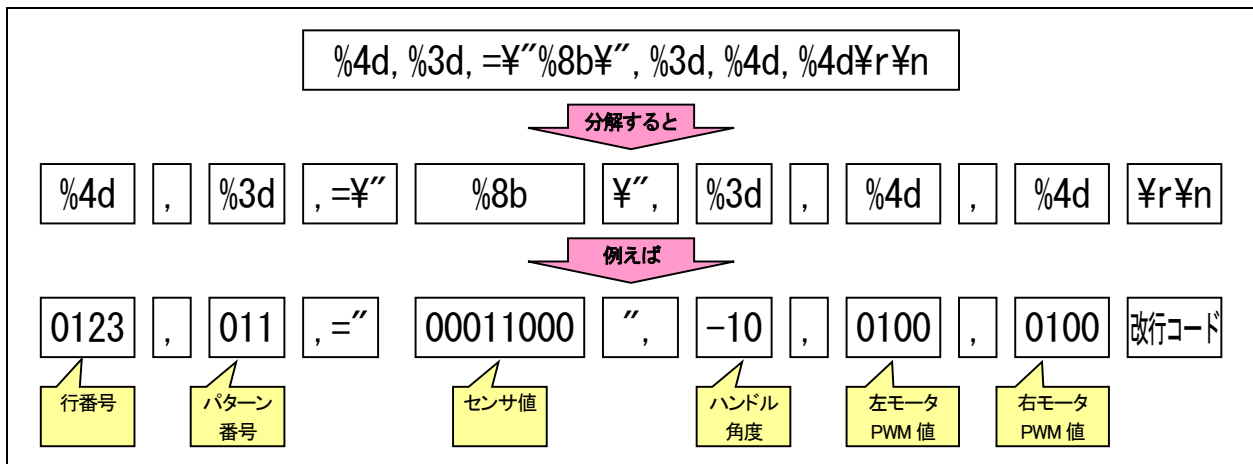
```

677 :         msdPrintf( "%4d, %3d, =¥"%8b¥", %3d, %4d, %4d¥r¥n",
678 :                   line_no,           // 行番号
679 :                   pattern,           // パターン番号
680 :                   sensor_inp(0xff),   // センサ情報(8bit)
681 :                   handleBuff,       // ハンドル値
682 :                   leftMotorBuff,    // 左モータ値
683 :                   rightMotorBuff    // 右モータ値
684 :         );
685 :         if( ++line_no >= 10000 ) line_no = 0;
686 :     }
687 : }
688 : }
    
```

661 行目	line_no という変数を設定しています。関数内で「static」を付けた変数を静的変数といい、関数が終了しても値を保持します。関数内のグローバル変数のようなイメージです。その関数内でしか使わないけれども値を保持したい場合は、静的変数にします。
674 行目	今回のプログラムは、microSD への記録を 10ms ごとに行います。割り込みは 1ms ごとなので、msdTimer 変数を割り込みごとに+1 して、10 になったなら 0 に戻します。
677 行目 ～ 684 行目	msdPrintf 関数で microSD へデータを記録します。今回は、次の 6 つの情報を記録します。 ①行番号 ②パターン番号 ③センサ情報(8bit) ④ハンドル値 ⑤左モータ値 ⑥右モータ値
685 行目	line_no 変数を+1 します。line_no 変数の値を microSD に書き込みますが、この変数は int 型なので、32767 以上にはできません。また今回 line_no 変数の記録は 4 桁なので、5 桁(10000 以上)になったら 0 に戻します。

10.5.7 記録する内容

microSD へ記録する書式を下記に示します。



記録例を、下記に示します。左から、パターン、センサ(2進数)、サーボハンドル角度、左モータ PWM、右モータ PWM 値です。1 行 37 文字(CR+LF を含む)です。

```

0998, 013, ="00000000", -25, 0011, 0018
0999, 013, ="00000000", -25, 0011, 0018
1000, 022, ="11111111", 000, 0000, 0000
1001, 022, ="11111111", 000, 0000, 0000
1002, 022, ="11111111", 000, 0000, 0000
    
```



## 11. 参考文献

- ルネサス エレクトロニクス(株)  
R8C/38C グループ ユーザーズマニュアル ハードウェア編 Rev.1.10
- ルネサス エレクトロニクス(株)  
M16C シリーズ,R8C ファミリー用 C/C++コンパイラパッケージ V.6.00  
C/C++コンパイラユーザーズマニュアル Rev.1.00
- ルネサス エレクトロニクス(株)  
High-performance Embedded Workshop V.4.09 ユーザーズマニュアル Rev.1.00
- 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
- 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) 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、販売部品についての詳しい情報は、マイコンカーラリー販売サイトをご覧ください。

<https://www2.himdx.net/mcr/>

R8C 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/>

の製品情報にある「マイコン」→「R8C」でご覧頂けます