SMALIGHT アプリケーションノート

ＪＭＣＲ２００９レポート
Smalight OS搭載マイコンカーラリーの開発

2009 年 1 月 11 日、札幌国際情報高校で開催された JMCR2009(ジャパンマイコンカーラリー2009)にて、Smalight OS 搭載のマイコンカー『はしれ早川』号を出走させました。本ドキュメントでは Smalight OS 搭載マイコンカーについて説明します。

目 次

図・表・リスト目次

1. 背景

1.1 マイコンカーラリーと RTOS(リアルタイム OS)

マイコンカーラリーで採用されるマイコンボードには、ルネサステクノロジ製 H8/3048F-one が搭載されております。H8/3048F-one は 16 ビットシングルチップマイコンで、内蔵フラッシュメモリ(128K バイト)にプログラムを書き込み、内蔵 RAM(4K バイト)上に変数などのデータを配置して動作します。

マイコンカーラリーでは RTOS(リアルタイム OS)はほとんど普及していないのが現状ですが、その要因としていくつかの理由が考えられます。

- 対象マイコン H8/3048F-one の搭載メモリが少ない (RTOS 導入するとメモリ不足となる)
- 高速で制御するマイコンカーでは OS オーバーヘッドが障害になる
- 商用 RTOS のほとんどが高価であった（最近は利用可能な RTOS が増えてきました）

1.2 Smalight OS

前述した RTOS 導入デメリット 1 つ目のメモリ不足に対して、Smalight OS は小メモリで動作可能な RTOS で、H8/3048F-one の少ないメモリ容量でも RTOS を導入した制御プログラムを動作させることができます。

RTOS 導入デメリット 2 つ目の OS オーバーヘッドは OS を導入した全てのシステムで必ず発生します。アプリケーションをタスク(仕事の単位)分割することで開発効率、保守性の向上などのメリットを享受することができますが、複数のタスクが動作するとき、タスクの切り替えや、OS サービスコールを発行することで OS が実行されることを OS オーバーヘッドといいます。

Smalight OS も例外なく OS オーバーヘッドが存在します。マイコンカーの開発において、Smalight OS の OS オーバーヘッドを考慮した設計を行う必要があります。その上で致命的な性能問題になるか検証していく必要があります。

RTOS 導入デメリット 3 つ目の価格についての考察は本ドキュメントでは割愛させて頂きます。

2. 開発コンセプト

2.1 Smalight OS 搭載マイコンカー

Smalight OS を導入してマイコンカー制御プログラムを開発します。

適切なタスク設計によりマイコンカー制御の機能を分割することで、見やすく、デバッグしやすいプログラム構造とします。また、OS オーバーヘッドを意識して、使用する OS 機能の選択、タスク構成を検討することで、RTOS 導入マイコンカーのソフトウェア構成の雛形となる構成を目標に開発を進めていきます。

2.2 ハード構成

マイコンカーを速く走らせるためには(ソフトウェアより)ハードウェアが占めるウェイトが大きい。Smalight OS 搭載マイコンカーとして、ハードウェア偏重では RTOS を導入したソフトウェアの評価がぼやけてくるため、基本をマイコンカーラリー公式標準キットとする。

3. ハードウェア

3.1 ハードウェア仕様

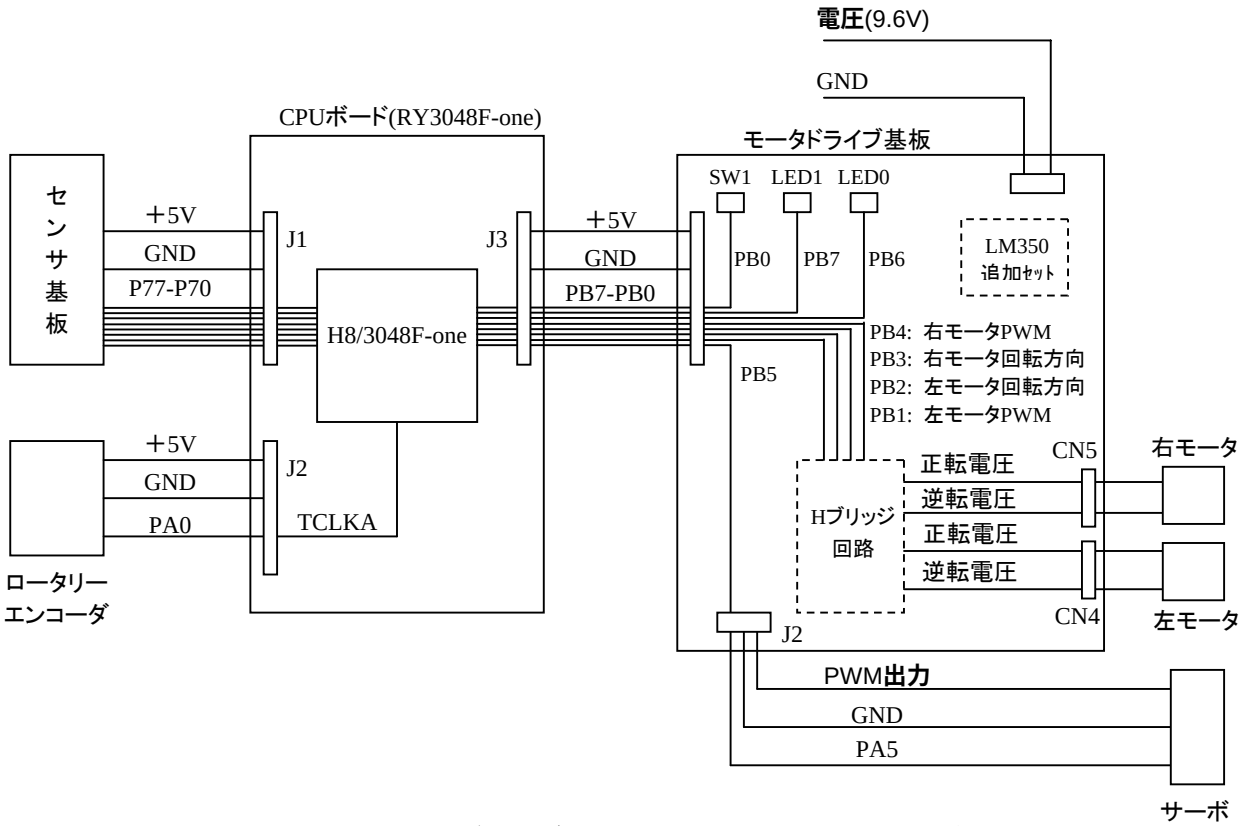
マイコンカーのハードウェア仕様を以下に示します。

表3-1 ハードウェア仕様

項番	項目	内容	備考
1	**基本構成**	**マイコンカーラリー公式標準キット**	
2	**全長・幅・高さ**	**全長** : 440 mm **幅** : 190 mm **高さ** : 70 mm	
3	**重量**	900 g (**電池含む**)	
4	**バッテリ**	GP2500 × 8 **本** **ニッケル水素電池 単** 3 **型** 1.2V 2500mAh	
5	**タイヤ**	**スポーツタイヤセット**×2 **セット**(2 **本/セット**) (**楽しい工作シリーズ** No.111) **タイヤサイズ：直径** 56mm、**幅**: 25mm (**シリコンシートによるグリップ強化**)	(株)タミヤ
6	CPU **ボード**	RY3048F-ONE CPU : H8/3048F-one CLOCK: 24.567MHz	
7	**センサ基板**	**センサ基板** Ver.4 **白線センサ**: 7 **個** **スタートバー検出センサ：**1 **個**	
8	**モータドライブ基板**	**モータドライブ基板** Vol 3 (LM350 **追加セットによるモータ電圧** VUP)	
9	**駆動モータ**	RC-260RA18130 × 2 **個(後輪左右)** 6V **カーボンブラシ**	(株)マブチ
10	**ギヤーボックス**	**ハイスピードギヤボックス** HE × 2 **個(後輪左右)** **ギヤ比** : 11.6:1	(株)タミヤ
11	**サーボ**	ERG-WRZ **重量**: 60g **寸法**: 39.0×20.0×37.4mm **速度**: 0.06sec/60 ° **トルク**: 8.1Kg.cm	(株)サンワ **デジタルサーボ**
12	**エンコーダ**	**ロータリーエンコーダセット** Ver.2 **相数**: 1 **相** 1 **回転当りのパルス数**: 36	

3.2 配線図

マイコンカーの配線図を以下に示します。



※ LCD、EEPROM を接続できるデバッグ用拡張基板を自作しておりますが、その部分の配線は省略しております。

図3-1 配線図

3.3 マイコンカー外観

マイコンカー外観を以下に示します。

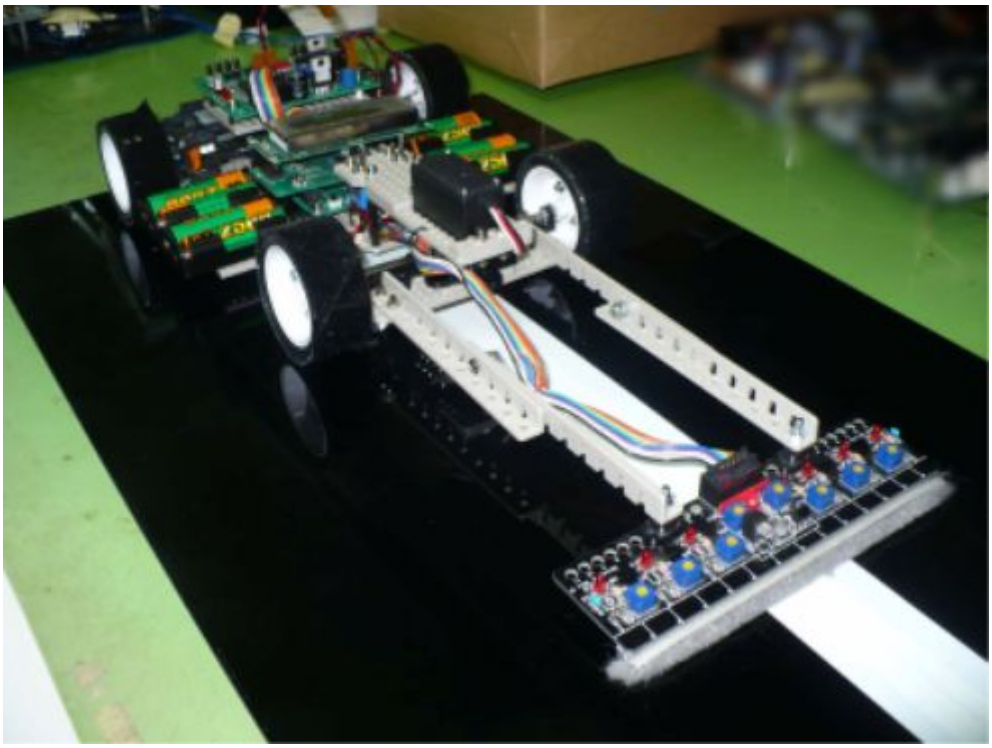


図3-2 マイコンカー外観

3.4 マイコンカーラリー公式標準キットからの変更点

3.4.1 サーボの変更

マイコンカーラリー公式標準キットにて採用されるサーボ HS-425BB の PWM 周期は 16(m 秒)となります。PWM 周期が 16(m 秒)の場合、デューティー比を変更してサーボの角度を変更してから最悪値で 16(m 秒)後に PWM 出力に反映されます。また、サーボの物理的な反応時間(4.8V 60° 移動時 0.21(秒))も考慮すると更に遅れが生じます。

白線センサの入力からハンドル移動するまでの時間が大きくなると、カーブやクランクでコースアウトのリスクが高くなるため、全体的に走行速度を遅くする必要があります。

走行性能への影響が大きいサーボは、より高速動作可能なサーボに置き換えて、高速化を図りました。

表3-2 サーボ変更

項番	変更前	変更後
1	HS-425BB (株)HITEC **重量**: 45.5g **寸法**: 40.6×19.8×36.6mm **速度**: 0.21sec/60 ° (4.8V) **トルク**: 3.3Kg.cm (4.8V) PWM **周期**: 16(m **秒**) **備考**: **アナログサーボ**	ERG-WRZ (株)サンワ **重量**: 60g **寸法**: 39.0×20.0×37.4mm **速度**: 0.06sec/60 ° **トルク**: 8.1Kg.cm PWM **周期**: 7(m **秒**) **備考**: **デジタルサーボ**

3.4.2 モータドライブ基板の改造

マイコンカーラリー公式標準キットは電池 4 本(4.8V)ずつ CPU ボード、駆動モータに供給しております。LM350 追加セットを加えた改造により、電池 8 本(9.6V)を駆動モータに供給しモータ回転数を上げます。

CPU ボードの動作電圧は 4.5～5.5 V、サーボの動作電圧は 6V 以下ですので、LM350 追加セットによる改造で定格にします。

3.4.3 ロータリーエンコーダ取付け

クランク、レーンチェンジ制御では大きくハンドルをきってから、しばらく走ってハンドルを戻します。ハンドルを戻すときコース中央の白線を見つけて戻しますが、ハンドルを大きくきってから安定するまでの間、センサ読み取りを無視します。その間、ロータリーエンコーダが付いていないサンプルプログラムでは安定するまでの判断を一定の時間としています。そのとき、止まっていても早く動いていても一定の時間経過すると、コース中央の白線をチェックしてハンドルを戻すタイミングを決定しますが、コース両端にも白線があるため、走行速度により、誤った判断をしてハンドルを戻すリスクが高くなります。

アナログセンサでコース中央の灰色識別している場合、この問題は回避可能です。

また、センサの状態を見ながら駆動モータに対して PWM 制御のデューティ比を設定しますが、ロータリーエンコーダが付いていないサンプルプログラムでは速度を認識できません。長い直線など、想定した速度より早くなった場合には次のカーブ、クランク、レーンチェンジで、コースアウトしてしまいます。

ロータリーエンコーダを付けることで、距離と速度を把握することができます。前述の問題を解決することが可能です。

3.4.4 センサ基板取り付け位置

カーブ、クランク、レーンチェンジをスムーズに行うため、早くその事象に気付くことが重要になります。センサ基板と前輪支持板とのアーム接続位置を、本体組み立てマニュアルより、4 箇所前に出します(40(mm)前方にセンサ位置が移動します)。

4. ソフトウェア機能設計

4.1 システム定義

4.1.1 システムの定義

以下の図はシステム定義を示します。基本構成としては、入力２種類、出力２種類とシンプルに表現できます。

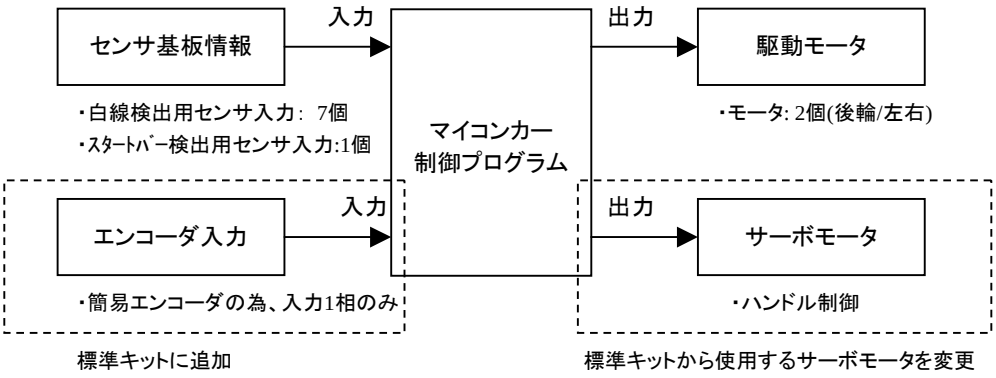


図4-1 システム定義

4.1.2 論理モデルの定義

以下の図は論理モデルを示します(この時点では、まだタスク構成等は決定しておりません)。

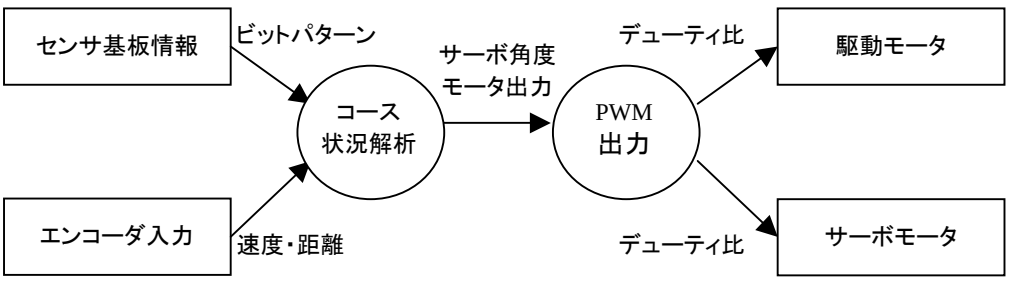


図4-2 論理モデル

4.2 要求されるリアルタイム性の検討

4.2.1 入出力の確認

各入出力のリアルタイム性について以下に示します。

表4-1 各入出力のリアルタイム性

項番	項目	リアルタイム性		備考
1	**センサ入力(白線)**	◎	**要求されるリアルタイム性は高い**	
2	**センサ入力(スタートバー)**	◎	**要求されるリアルタイム性は高い**	
3	**ロータリーエンコーダ入力**	○	**要求されるリアルタイム性は高いが、内蔵 ITU で処理されるため、CPU 負荷は低い。**	
4	**サーボ PWM 出力**	△	**使用するサーボモータの PWM 周期は 7(m 秒)**	
5	**右モータ PWM 出力**	△	**サーボ PWM に合わせて制御**	
6	**右モータ回転方法出力**	△	**サーボ PWM に合わせて制御**	
7	**左モータ PWM 出力**	△	**サーボ PWM に合わせて制御**	
8	**左モータ回転方法出力**	△	**サーボ PWM に合わせて制御**	
9	SW1	×	**人による操作**	
10	LED0	×	**目視**	
11	LED1	×	**目視**	

センサ入力(白線、スタートバー)に要求されるリアルタイム性は高く、目標性能を明確にする必要があります。次章で目標性能を検討していきます。

スタートバーのセンサ入力は、スタートバーが開くことを少しでも早く認識して走行を開始したい。しかし、主に走行開始前のチェックのため、リアルタイム性の実現は容易である(他の処理と並行して行う必要がないため、スタートバーのセンサ入力だけを行っても良い)。

サーボ PWM 出力、左右モータ PWM 出力は、リセット同期 PWM 制御を行っており PWM 周期は共通です。サーボ PWM 出力は PWM 周期 7(m 秒)以下にするとサーボが正しく動作できないため、左右モータ PWM 出力も PWM 周期 7(m/秒)に合わせて動作させます。

4.2.2 白線認識センサの読み取り間隔

白線センサの読み取りを見落としすることなく行うとは、一番速度が出ている状態で以下の判断が正しくできることです。

- **直線**
- **カーブ**
- **クランク予告**
- **クランク開始**
- **レーンチェンジ予告**
- **レーンチェンジ開始位置**

性能制限が一番厳しいと思われるクランク予告(レーンチェンジ予告)での目標性能を考えてみます。

(1) **直線での最高速度を想定し、読み取り時間間隔を計算する。**

直線での最高速度を 5 .0(m/秒)と仮定します(実際には Max 2.7(m/秒)程度で走行しました)。
クランク予告(レーンチェンジ予告) 白線の幅は 20.0(mm)です。

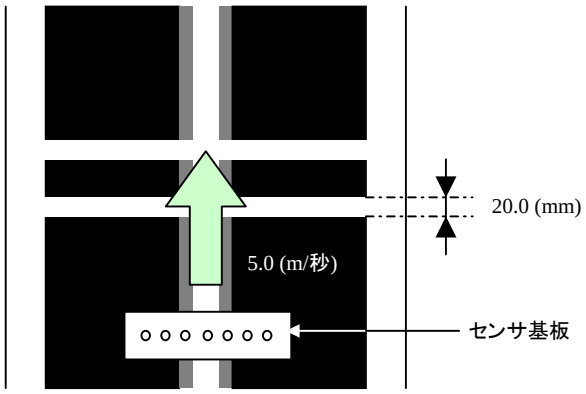


図4-3 白線読み取り条件

読み取り時間間隔(m 秒) = 20.0(mm) / 5.0(m/秒)
= 4.0 (m 秒)

但し、4.0 (m 秒)間隔での読み取りでは、20(mm)毎に 1 回のみとなります。汚れや振動による誤認識の可能性を考えると 20(mm)に 5 回は読み込みを行いたい。つまり、4(mm)毎に読込みを行います。

読み取り時間間隔(m 秒) = 4.0(mm) / 5.0(m/秒)
= 0.8 (m 秒)

(2) **クランク予告(レーンチェンジ予告) 突入角度を想定した読み取り時間間隔を計算する。**

前述の計算では、必ずマイコンカーが直線をまっすぐ走行することを想定しましたが、マイコンカーは白線の読み取り結果をフィードバックしながら走行するため、クランク予告(レーンチェンジ予告)突入角度が 0°とは限りません。突入時の角度が 0°以外のときを想定した読み取り時間間隔を検討します。

まずは使用するセンサ基板の寸法を確認すると、センサ範囲(右端～左端センサの距離)は、124.46mm となります。

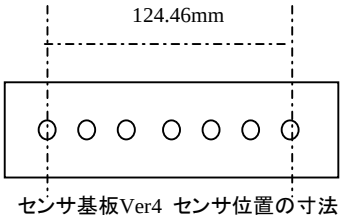


図4-4 センサ基板 寸法(右端～左端センサ距離)

クランク予告(レーンチェンジ予告)を正しく認識できる突入角度の限界値を計算します。

$\sin\theta$ = 20.0(mm) / 124.46(mm)
θ = 9.3°

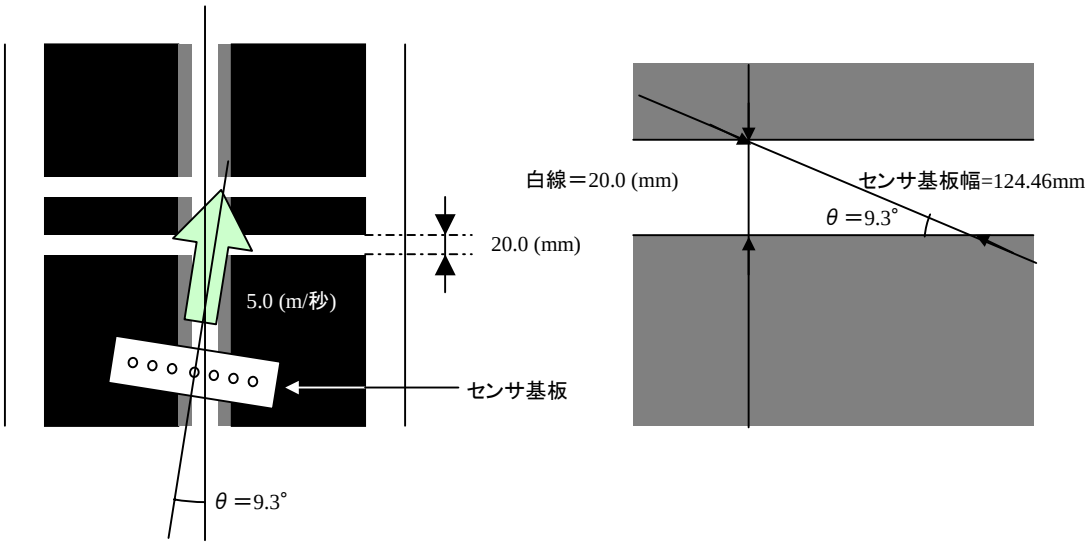


図4-5 クランク突入角度による誤認識の可能性

クランク予告(レーンチェンジ予告)突入角度が、9.3°を越える場合、クランク予告を見落としてしまいます。また、9.3°のとき、正しく認識できるのは一瞬で 0.8(m 秒)のセンサ読み取り間隔では、読み落とす可能性が高くなります。

クランク予告(レーンチェンジ予告)を通過するときの突入角度が大きいとき (9.3°を越える)、正しく検知できない問題があるため、事前の直線で正しく突入角度が 0.0°に限りなく近づける様フィードバック制御を行う必要があることを認識できました。

クランク予告(レーンチェンジ予告)を通過するときの突入角度 6.0°以内であれば、正しく認識できる様にするための読み取り時間間隔を計算します。

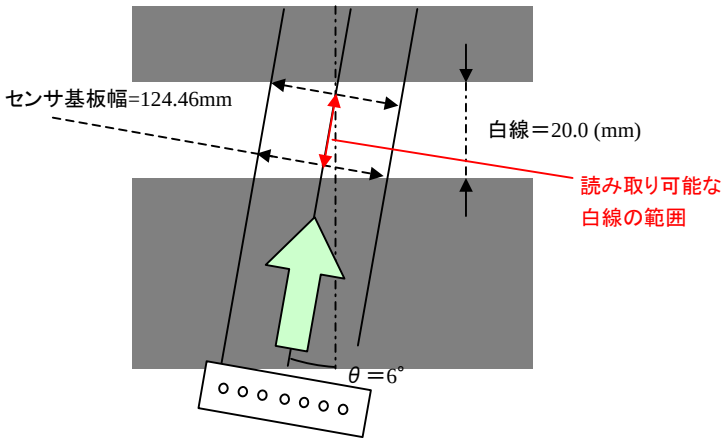


図4-6 クランク突入角度6.0°のとき読み取り可能な白線範囲

読み取り可能な白線の範囲 = (20.0(mm) / cos 6.0°) ー （ 124.46(mm) × tan 6.0°）
= 7.03 (mm)

同様に 7.03(mm)移動中に 5 回読み込むためには、1.41(mm)に一回の読み込みが必要です。

読み取り時間間隔(u 秒) = 1.41(mm) ÷ 5.0(m/秒)
= 281.2 (u 秒)

前述の検討結果から、読み取り時間間隔の目標性能を 250 (u 秒)と設定することで読み落としの無いプログラムを作成していきます。

4.3 Smalight OS 実装検討

本章では、Smalight OS 実装方法についての検討内容について説明しております。最終的な実装に基づき記載しておりますが、いきなり実装を決定した訳ではなく、4.3.1～4.3.3 の検討と評価を繰り返した上で、最終的な判断をしております。

4.3.1 タスク構成

前述の論理モデルの検討結果から、タスク構成を以下の２個に決定します。参考まで、タスク間通信部、ドライバの定義についても記述しております。

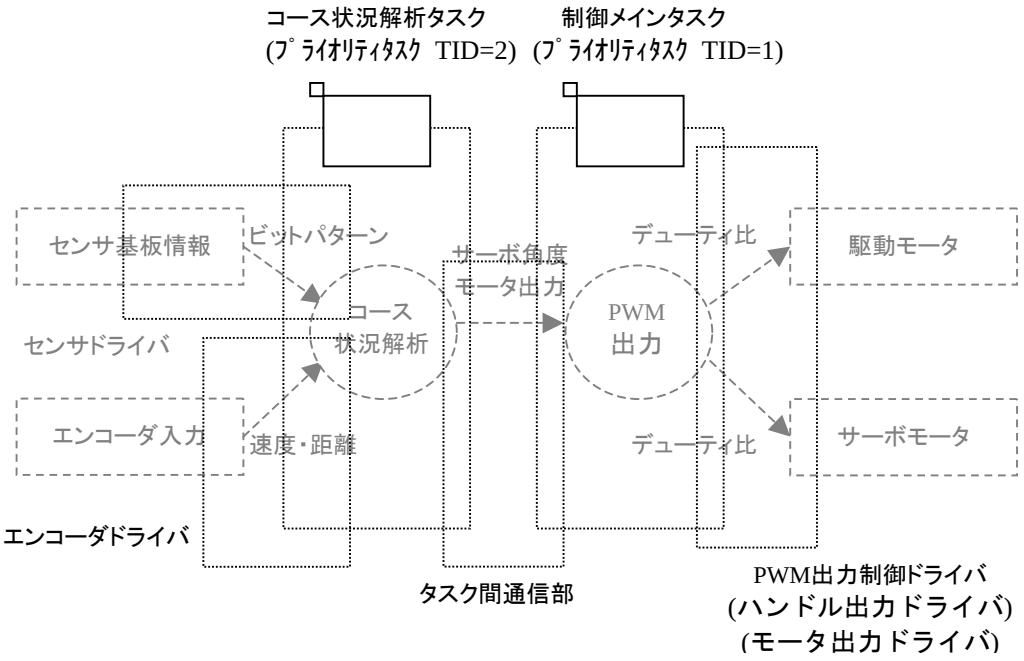


図4-7 タスク構成の検討

白線センサ読込みの目標性能 250(u 秒)は、動作周波数 24.567MHz の CPU に RTOS を搭載した場合、厳しい条件となります。タスク構成を更に細分化することも可能ですが、タスク数増加は OS オーバーヘッド増、メモリ増(タスクスタック)に影響を与えます。タスクの細分化は、OS オーバーヘッド、メモリ容量とトレードオフになります。

Smalight OS で使用可能なタスク間通信機能はイベントフラグとデータキューがあります。どういった方法で、タスク間通信を行うかは別途検討します。

ドライバとは入出力や機能を抽象化したインタフェースのことを指します。システム内の入出力や機能を適切にドライバ定義することで、汎用性の高いプログラムを開発できます。例えば、開発中に使用しているサーボを HS-255BB から ERG-WRZ に変更しても､アプリケーション部分はそのままで、ドライバ部分の修正のみで対応できます。

4.3.2 時間管理

通常、Smalight OS の時間管理は 1(m 秒)単位で行われます。白線センサ読込みの目標性能 250(u 秒)を実現するために、基準時間を 1(u 秒)と解釈して、周期ハンドラの周期時間を 250(u 秒)として実装します。その実装で問題なく動作可能か検討していきます。

本実装は裏技的な方法となります。厳密にシステム全体のパフォーマンスを確認した上で実装しないと、沈み込み(優先度の低いタスクが実行されない)などの問題が発生する可能性があります。

(1) 周期タイマハンドラの実行時間

まずは周期タイマハンドラ(ITU2)の実行時間を測定します。下記の表は 2 つのタスクが、tslp_tsk サービスコールで実行動作タイミングをとりながら動作するプログラムを想定しています。各条件下で周期タイマハンドラ割込みが発生してから、タスクが Run 状態になる、またはシステム状態がアイドルとなるまでの時間を計測します。

計測はシミュレータデバッガのトレース機能を利用して実行サイクル数を測定したものです。

表4-2 周期タイマハンドラの実行時間

項番	周期ﾀｲﾏﾊﾝﾄﾞﾗ割込み発生条件			周期ﾀｲﾏﾊﾝﾄﾞﾗ割込み発生後の状態			実行時間
	ﾀｽｸ 1	ﾀｽｸ 2	割込発生元	ﾀｽｸ 1	ﾀｽｸ 2	割込み後の動作	
1	RUN	RDY	**タスク 1**	RUN	RDY	**タスク 1 実行**	49.1 (u **秒**)
2	TSLP	RUN	**タスク 2**	TSLP	RDY	**タスク 2 実行**	51.9 (u **秒**)
3	TSLP	TSLP	**アイドル**	TSLP	TSLP	**アイドル**	51.9 (u **秒**)
4	TSLP	TSLP	**アイドル**	RUN	TSLP	**タスク 1 実行**	77.5 (u **秒**)
5	TSLP	TSLP	**アイドル**	TSLP	RUN	**タスク 2 実行**	77.5 (u **秒**)
6	TSLP	TSLP	**アイドル**	RUN	RDY	**タスク 1 実行**	104.0 (u **秒**)

※ RUN:Running 状態 RDY:Ready 状態 TSLP: Waiting(T 付き起床待ち)

上記測定結果から、周期タイマハンドラの周期時間 250(u 秒) に対して、周期タイマハンドラの実行時間が最悪値で 104.0(u 秒)です。そのとき、タスクが実行可能な時間（周期タイマハンドラの周期時間—周期タイマハンドラの実行時間）は 146.0(u 秒)となります。その割合が十分か否かの検討をする必要があります。

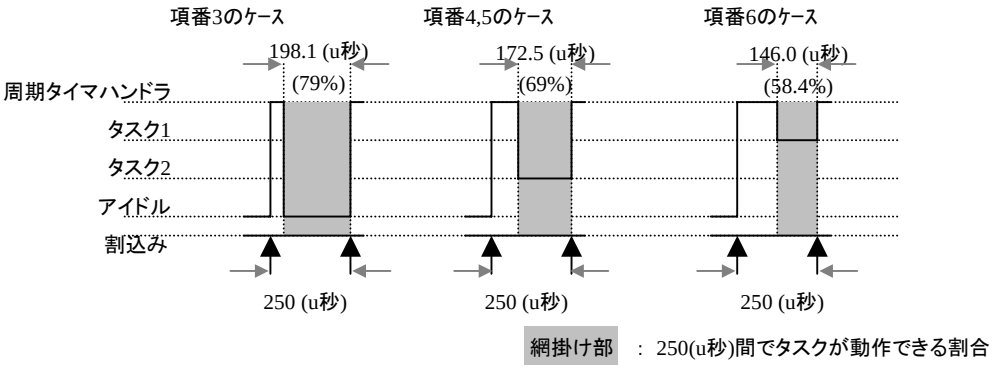


図4-8 周期ハンドラとタスクの実行時間

(2) Smalight OS ライブラリリビルドによる OS オーバーヘッドの削減

OS オーバーヘッド、メモリ制限等の制約が大きい場合、Smalight OS ライブラリリビルドで不要な処理、メモリの節約が可能です。具体的には標準で実装される Smalight OS 機能のうち、不要な機能を外して OS ライブラリを再構築することを言います。

今回、使用しない機能は①ウェイクアップカウント ②周期ハンドラ ③ セマフォ ④データキューです(イベントフラグはデバッグ用のシリアル通信にて実装しました)。再構築時の Smalight OS ヘッダファイル(slos.h)リストを以下に示します。OS 再構築の詳細は『Smalight OS V3 リファレンスマニュアル』をご参照ください。

リスト4-1 Smalight OS再構築(slos.h)

```
                    :
(22 行目)   #include "slos.def"
(23 行目)
(24 行目)   /*------ Building block --------------------*/
(25 行目)   #define KNL_BB_PRIORITY
(26 行目)   //#define KNL_BB_WUPCNT           … コメントアウト
(27 行目)   #define KNL_BB_SYSTIME
(28 行目)   #ifdef KNL_BB_SYSTIME
(29 行目)   //#define KNL_BB_CYCHDR           … コメントアウト
(30 行目)   #endif
(31 行目)   #define KNL_BB_EVTFLG
(32 行目)   //#define KNL_BB_SEMAPHORE        … コメントアウト
(33 行目)   //#define KNL_BB_DATAQUE          … コメントアウト
                    :
```

再構築後の周期タイマハンドラの実行時間を以下に示します。多少、オーバーヘッドが削減されました。

表4-3 再構築後の周期タイマハンドラの実行時間

項番	周期ﾀｲﾏﾊﾝﾄﾞﾗ割込み発生条件			周期ﾀｲﾏﾊﾝﾄﾞﾗ割込み発生後の状態			実行時間	
	ﾀｽｸ 1	ﾀｽｸ 2	割込発生元	ﾀｽｸ 1	ﾀｽｸ 2	割込み後の動作	再構築前	再構築後
1	RUN	RDY	**タスク 1**	RUN	RDY	**タスク 1 実行**	49.1 (u 秒)	47.9(u 秒)
2	TSLP	RUN	**タスク 2**	TSLP	RDY	**タスク 2 実行**	51.9 (u 秒)	50.8(u 秒)
3	TSLP	TSLP	**アイドル**	TSLP	TSLP	**アイドル**	51.9 (u 秒)	50.8(u 秒)
4	TSLP	TSLP	**アイドル**	RUN	TSLP	**タスク 1 実行**	77.5 (u 秒)	74.6(u 秒)
5	TSLP	TSLP	**アイドル**	TSLP	RUN	**タスク 2 実行**	77.5 (u 秒)	74.6(u 秒)
6	TSLP	TSLP	**アイドル**	RUN	RDY	**タスク 1 実行**	104.0 (u 秒)	100.5(u 秒)

※ RUN:Running 状態 RDY:Ready 状態 TSLP: Waiting(T 付き起床待ち)

(3) タスク実行時間の十分性の検討

タスク 1 は､コース状況に合わせて PWM 制御のデューティ比を設定するのが仕事となります(駆動モータ、サーボモータの PWM 制御)。

PWM 制御は、H8/3048F-one の内蔵タイマ ITU(リセット同期式 PWM モード)にて実現しております。駆動モータ(左右)とサーボモータの PWM 周期は共通となりますので、サーボモータ制御が可能な PWM 周期 7(m 秒)に決定します。

H8/3048F-one の内蔵タイマ ITU に設定することで PWM 出力されますが、こまめにデューティ比を変更しても、実際に PWM 出力に反映されるタイミングは、PWM 周期 7(m 秒)毎に 1 回だけです。例えば、PWM 周期 7(m 秒)の間、100(u 秒)毎にセンサチェックとデューティ比変更を行うと 70 回 ITU 設定が行えますが、有効な変更は最後の 1 回だけで 69 回は無駄な設定となります。

タスク 1 は上記理由から 3.5(m 秒)に一回動作すれば十分です。

タスク 2 はセンサを確認して、コース状況解析することが仕事となります。目標性能であるセンサ読み取り間隔 250(u 秒) に合わて動作する必要があります。

前述の周期タイマハンドラのオーバーヘッドを考慮すると、タスク 2 でセンサ読み取りをするのは危険です。タスク 1 も実行する必要があるため、タスクスケジューリングされず 250(u 秒)に 1 回の実行時間が保障されない可能性があるからです。

そこで、センサ読み取りは周期タイマハンドラにて実施し、タスク 2 は、周期タイマハンドラで読み取ったセンサ情報を受け取ることにします。それにより、センサ読み取り間隔 250(u 秒)の目標性能はクリアできます。

タスク 2 は 250(u 秒)毎に周期タイマハンドラにて収集されるセンサ情報を処理する必要があるため、周期タイマハンドラはセンサ情報を FIFO(First IN First OUT)方式のバッファに格納し、タスク 2 では 500(u 秒)毎に FIFO バッファに蓄積されたセンサ情報を取り出して、コース状況解析を行うこととします。

以下にタスク関連図と動作フローを示します。現時点ではタスク 1,2 間のデータ通信については未定です。

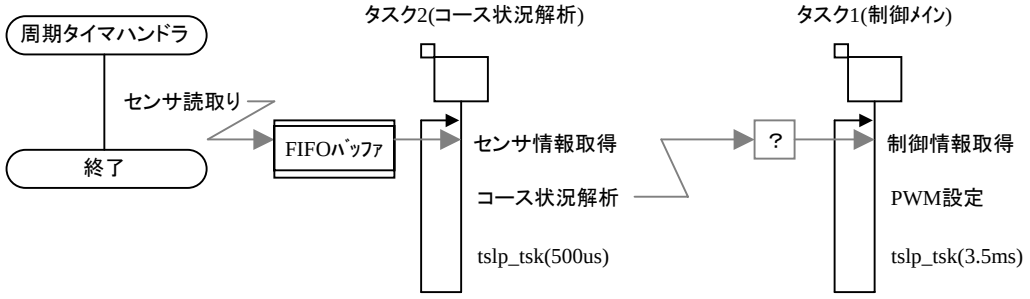


図4-9 タスク関連図と動作フロー

タスク 1,2 の動作タイミングから、3.5(m 秒)毎に以下の周期タイマハンドラのオーバーヘッドが想定されます。

a) 3.5(m 秒)間に実行される周期タイマハンドラの回数 :
3.5(m 秒) ÷ 250(u 秒) = 14 回

b) タスク 1,タスク 2 の実行タイミング
タスク 1 は 3.5(m 秒)毎に起床, タスク 2 は 500(u 秒)毎に起床します。
14 回の周期タイマハンドラ実行後の状態は以下の通りです。

表4-4 周期タイマハンドラ実行後のタスク状態

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
タスク 1	R	W	W	W	W	W	W	W	W	W	W	W	W	W
タスク 2	R	W	R	W	R	W	R	W	R	W	R	W	R	W

※ R: Run または Ready 状態, W: Waiting 状態

c) 上記表から、14 回の周期タイマハンドラ実行に対し、7 回はタスク 1,2 共に TSLP(Waiting 状態)で周期タイマハンドラが実行され、その結果、タスク 1,2 共に TSLP(Waiting 状態)である。

d) 上記表から、14 回の周期タイマハンドラ実行に対し、6 回はタスク 1,2 共に TSLP(Waiting 状態)で周期タイマハンドラが実行され、その結果、タスク 2 が RUN(Running 状態)である。

e) 上記表から、14 回の周期タイマハンドラ実行に対し、1 回はタスク 1,2 共に TSLP(Waiting 状態)で周期タイマハンドラが実行され、その結果、タスク 1 が RUN(Running 状態), タスク 2 が RDY(Ready 状態)である。

前述の c)〜e)と周期タイマハンドラのオーバーヘッドを以下の表に纏めることできます。

表4-5 周期タイマハンドラのオーバーヘッドとタスク実行時間

状態	実行回数	f)時間合計	g)オーバーヘッド	f)-g)タスク実行時間
c)	7 回	250(u 秒)×7 回=1750(u 秒)	50.8(u 秒)×7 回=355.6(u 秒)	1394.4(u 秒)
d)	6 回	250(u 秒)×6 回=1500(u 秒)	74.6(u 秒)×6 回=447.6(u 秒)	1052.4(u 秒)
e)	1 回	250(u 秒)×1 回=250(u 秒)	100.5(u 秒)×1 回=100.5(u 秒)	149.5(u 秒)
計	14 回	3500(u 秒)	903.7(u 秒)	2596.3(u 秒)

3.5(m 秒)あたりのタスクが実行できる時間が 2.6(m 秒)と予測できました。その時間内にタスク 1 が 1 回、タスク 2 が 7 回動作できれば、タスクの沈み込みなど発生せずに動作可能です。最終的に、各タスクの実行時間も計測して想定した時間内で動作できることを検証する必要があります。

4.3.3 タスク間データ通信制御

タスク間通信として、タスク 2(コース状況解析)からタスク 1(制御タスク)へデータを通信する必要があります。通信データとして考えられる情報は以下の通りです。

- ハンドル角度
- モータ速度(モード)

Smalight OS が提供するタスク間データ通信機能は①イベントフラグ②データキューの 2 つの機能になります。

(1) イベントフラグの可能性

Smalight OS のイベントフラグのサイズは 16 ビットです。イベントフラグをデータ通信として利用する場合、待ち解除属性を OR に設定した上で、受信側が待ちビットバターンを H'FFFF とすることで、16 ビットデータの受け渡しが可能となります。

ハンドル角度、モータ速度という 2 つのデータをイベントフラグで行うのは、データサイズが不足していることから適していないと判断します。

(2) データキューの可能性

Smalight OS のデータキューは 32 ビットのデータ受け渡しが可能です。データ転送の案として以下の方法が考えられます。

- 32 ビットの上位 16 ビットをハンドル角度、下位 16 ビットをモータ速度と意味付けする。
- ハンドル角度、モータ速度を格納したテーブルアドレスをデータ送信する。

では、データキューを使った場合のオーバーヘッドはどの様になるか検証してみます。タスク 1 がデータキューからの受信待ち、タスク 2 がデータキューへの送信をした場合、約 76(u 秒)のオーバーヘッドとなりました。

計測はシミュレータデバッガのトレース機能を利用して実行サイクル数を測定したものです。

データキュー利用も可能かもしれませんが、周期タイマハンドラのオーバーヘッドと、データキューのオーバーヘッドを合わせると少し性能的に厳しい印象があります。

(3) グローバル変数を利用したデータ通信

前述の検討から、Smalight OS のタスク間データ通信を利用せず、グローバル変数を用いてデータの受け渡しを行うこととします。それにより、データ通信による負荷はほとんど気にする必要がありません。

但し、データ更新を行う箇所とタイミング、データ参照を行う箇所とタイミングを整理しておく必要があるでしょう(排他制御)。

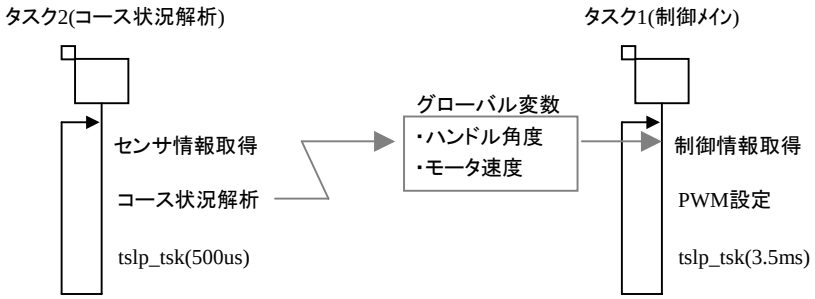


図4-10 タスク間データ通信(グローバル変数)

今回、性能面の検討から、グローバル変数を用いましたが性能的に問題なければ、極力、OS 提供のデータ通信を利用すべきです。

理由として、グローバル変数を用いるとモジュール間結合度が高くなり、モジュールの独立性が損なわれるからです。モジュール間結合度が高いと品質評価が困難になったり、移植性、保守性が低くなり結果として生産性が悪くなります。

OS 提供のデータ通信機能を用いることで、一般的には疎結合で独立性の高いモジュール設計が可能と言われております。

組込みシステムの開発では、量産コスト低減のため、安い部品(遅い CPU etc)を選択し、ギリギリの性能制約の中開発することが数多くあります。理想的な設計と妥協する部分のバランスも大切になります。

4.3.4 Smalight OS コンフィギュレーション一覧

以上の検討結果から、最終的な Smalight OS コンフィギュレーションを以下に示します。

表4-6 Smalight OSコンフィギュレーション一覧

項番	項目	設定				備考
1	CPU	H8/300H(Advanced mode)				
2	**カーネルマスクレベル**	CCR.UI=**マスクビット**(**割込みレベル**=3) **カーネルマスクレベル**=**レベル** 1				
3	**タスク**	**タスク数**=4				
		ID	**種別**	**関数名**	**説明**	
		1	**ﾌﾟﾗｲｵﾘﾃｨﾀｽｸ(優先度**=1)	tsk_handle	**制御メイン**	
		2	**ﾌﾟﾗｲｵﾘﾃｨﾀｽｸ(優先度**=2)	tsk_hayakawa	**コース状況解析**	
		3	**ﾌﾟﾗｲｵﾘﾃｨﾀｽｸ(優先度**=3)	tsk_debug_lcd	**デバッグ用**	(*1)
		4	**ﾌﾟﾗｲｵﾘﾃｨﾀｽｸ(優先度**=4)	tsk_turning	**デバッグ用**	(*1)
4	**イベントフラグ**	**有効：イベントフラグ数**=1				
		ID	**属性**	**説明**		
		1	OR	**シリアル送受信フラグ**		(*2)
5	**セマフォ**	**無効**				
6	**データキュー**	**無効**				
7	**時間管理**	**有効： 周期タイマハンドラの周期時間** = 250 (u **秒**)				(*3)
8	**データキュー**	**無効**				

(*1) デバッグ用タスク。初期状態=Waiting 状態(起床待ち)で基本的には動作しない。
(*2) デバッグ用シリアル通信にて使用。基本的に使用されない。
(*3) 周期タイマハンドラの周期時間は通常(m 秒)単位であるが、(u 秒)単位と解釈する。

5. ソフトウェア詳細設計

5.1 駆動モータ・サーボの出力

5.1.1 PWM 制御

PWM 制御はドライバとして実装します。PWM 制御ドライバ関数一覧を以下に示します。

表5-1 PWM制御ドライバ関数一覧

項番	関数名	説明	備考
1	void pwm_init(void)	PWM **初期化** **引数**: **なし** **戻り値**: **なし**	
2	void set_handle(W angle)	**サーボに対して、**PWM **出力のデューティ比を設定します。** **引数**: W angle = **ハンドル角度** **戻り値**: **なし**	
3	void mtr_set_rate(W rrate, W lrate)	**駆動モータに対して、正回転** PWM **出力のデューティ比を設定します。** **引数**: W rrate = **右モータデューティ比** W lrate = **左モータデューティ比** **戻り値**: **なし**	
4	W mtr_break_mode(W rrate, W lrate, W speed)	**駆動モータに対して、指定した速度以上のとき、逆回転** PWM **出力し急ブレーキをかけます。そのときのデューティ比は** 100%**です。指定した速度より遅い場合は引数で指定したデューティ比を出力します。** **引数**: W rrate = **右モータデューティ比** W lrate = **左モータデューティ比** W speed = **速度** **戻り値**: **なし**	(*1)
5	W curve_get_insp(W angle, W rate)	**ハンドル角度に応じた内輪側駆動モータの** PWM **出力デューティ比を計算します。** **引数**: W angle = **ハンドル角度** W rrate =**外輪側モータデューティ比** **戻り値：内輪側モータデューティ比**	(*2)

(注) PWM 制御を定義したソースファイル pwm.c, pwm.h では表に記載されていない関数もありますが、アプリケーション側から呼び出されないプライベート関数のため記載しておりません。

(*1) 詳細は 6.4.1 章にて説明しております。

(*2) 詳細は 5.1.2 章にて説明しております。

ドライバとは、ハードウェア依存する処理を API としてインタフェース仕様を定義するものです。例えば接続しているサーボを変更したとき、ソフトウェアの変更はドライバ部分のみに局所化可能となり、アプリケーション(タスク)側への影響を最小限に抑えることが可能になります。

5.1.2 内輪差・外輪差

駆動モータの出力は、左右個別の PWM 出力を行っております。カーブなどでは内側のタイヤと外側のタイヤでは回転スピードが異なります。よりスムーズにカーブを曲がるため、左右の PWM 出力のデューティ比を調整します。

速度の調整は、ハンドル角度と外輪側の速度(デューティ比)を入力に、内輪側の速度(デューティ比)を計算して設定します。計算方法の詳細はマイコンカーラリー事務局から提供されるマニュアルをご参照ください。

5.2 エンコーダ入力

マイコンカーラリー公式標準キットに、ロータリーエンコーダセット Ver.2 を取り付けることにより、速度と距離を監視します。

ロータリーエンコーダの入力を ITU0 の TCLK に接続します。ITU0 はクロックエッジを両エッジとすることで、ロータリーエンコーダが 1 周 72 回カウントアップします。エンコーダ入力ドライバ関数一覧を以下に示します。

表5-2 エンコーダ関数一覧

項番	関数名	説明	備考
1	void encoder_init(void)	**エンコーダ初期化** **引数: なし　　戻り値: なし**	
2	void get_encoder_count(void)	10(m 秒)**毎に呼び出すことで、速度・距離情報を更新します。** **引数: なし　　戻り値: なし**	
3	W get_encoder_total_len(void)	**累計の走行距離を取得します。** **引数: なし** **戻り値: 累計の走行距離(単位**:mm)	
4	void encoder_len_clr(void)	**指定位置からの距離用のカウンタをクリアします。本関数を発行した瞬間からの距離を測定します。** **引数: なし　　戻り値: なし**	
5	W get_encoder_len(void)	**指定位置からの距離を取得します** **引数: なし** **戻り値: 累計の走行距離(単位**:mm)	
6	W get_encoder_speed(void)	**速度を取得します。** **引数: なし** **戻り値: 累計の走行距離(単位**:mm/秒)	

速度を監視することで、スピードオーバーによるコースアウトのリスクを低減できます。コース状態(直線、カーブ、クランク、レーンチェンジ)とその速度上限を設定して、速すぎる場合、駆動モータへの出力を抑える様にします。

距離を監視することで、累積の走行距離、および、現時点の位置からの距離を把握できる様にします。標準サンプルプログラムではクランクでハンドルを大曲げして安定するまでの時間経過を待ってから、クランクの抜けを判断しております。安定するまでの時間で確認すると停止していても、高速で動作していても、クランク抜けの判断処理に入ります(条件が異なる)。距離を基準にすることで、より確実にコース状況把握が可能となります。

注意点として、使用するロータリーエンコーダセット Ver.2 の相数が 1 相のため回転方向が認識できません(相数を 2 相にすることで正回転／逆回転の認識が可能となります)。そのため、逆方向へ回転した場合も距離・速度が正方向へ進んだと誤った解釈となることを認識する必要があります。

5.3 スタートバーセンサ入力

スタートバーセンサ入力は、I/O ポート P74 に接続されます(“1”=停止、”0”=スタート)。基本的には走行開始の合図のため、走行前にチェックすれば OK ですが、スタートバーセンサ入力が”1”のとき、走行中であっても駆動モータへの出力を停止する様にします。

5.4 白線センサ入力の解析

5.4.1 直線・カーブ

直線・カーブのコース状況認識と、そのときのモード、および、速度設定を以下に示します。各々のコース状態(モード)に合わせて、ハンドル角度、駆動モータ速度、駆動モータ上限速度を設けます。駆動モータ速度・駆動モータ上限速度はモードに対応して設定する様にします。駆動モータ速度・駆動モータ上限速度は、実際に走行評価することで決定していきます。

表5-3 直線・カーブのセンサ解析とPWM出力

センサ状態	P77-0	説明(モード)	ハンドル角度	速度	速度上限	備考
●●●○●●●	0xF7	**直線**	0 °	**検討中**	**検討中**	
●●○○●●●	0xD7	**カーブ左** 0	5 °	**検討中**	**検討中**	
●●○●●●●	0xDF	**カーブ左** 1	10 °	**検討中**	**検討中**	
●○○●●●●	0x9F	**カーブ左** 2	15 °	**検討中**	**検討中**	
●○●●●●●	0xBF	**カーブ左** 3	20 °	**検討中**	**検討中**	
○○●●●●●	0x3F	**カーブ左** 4	25 °	**検討中**	**検討中**	
○●●●●●●	0x7F	**カーブ左** 5	30 °	**検討中**	**検討中**	
●●●○○●●	0xF3	**カーブ右** 0	-5 °	**検討中**	**検討中**	
●●●●○●●	0xFB	**カーブ右** 1	-10 °	**検討中**	**検討中**	
●●●●○○●	0xF9	**カーブ右** 2	-15 °	**検討中**	**検討中**	
●●●●●○●	0xFD	**カーブ右** 3	-20 °	**検討中**	**検討中**	
●●●●●○○	0xFC	**カーブ右** 4	-25 °	**検討中**	**検討中**	
●●●●●●○	0xFE	**カーブ右** 5	-30 °	**検討中**	**検討中**	

速度上限以下で走行している場合、モード毎に定めるデューティ比を設定します。速度上限を超えた速度で走行している場合、デューティ比を 0%に設定して速度超過を防止します。

5.4.2 レーンチェンジ予告・クランク予告の判定

レーンチェンジはハーフライン、クランクはクロスラインで予告されます。ハーフライン、クロスラインの判定をするとき、突入角度が 0° 以外のとき誤認識となる可能性があります。

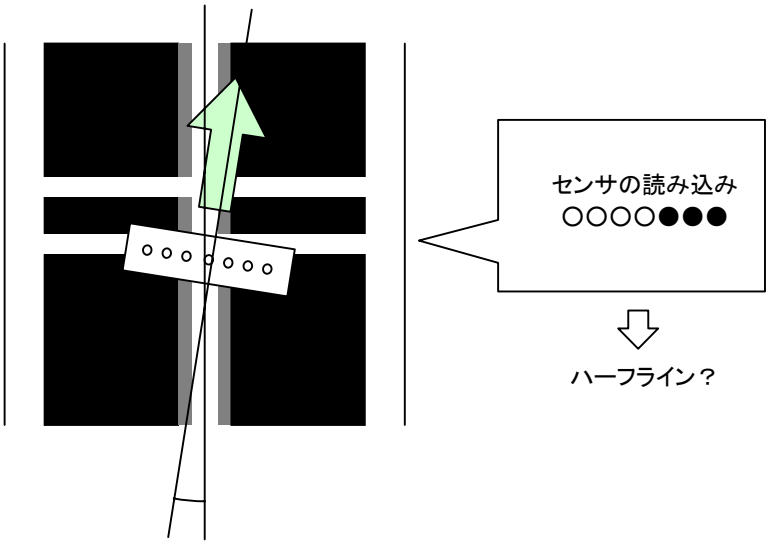


図5-1 レーンチェンジ予告・クランク予告の誤認識

上図はクロスライン(クランク予告)ですが、突入角度 0° 以外のため、瞬間的にハーフラインと誤認識する例です。そのため、ハーフラインと認識したケースにおいても、継続してセンサのチェックを続けて、ハーフライン・クロスラインを通過して、通常の直線、カーブの状態になったことを確認した時点で、はじめてレーンチェンジ・クランクであることを確定させます。

また、直線・カーブのセンサ解析は、クランク、レーンチェンジの以下の部分でも実施します(その間、直線ラインからズレていくと正しくクランク、レーンチェンジの処理ができなくなるからです)。そのため、本処理は関数として定義します。

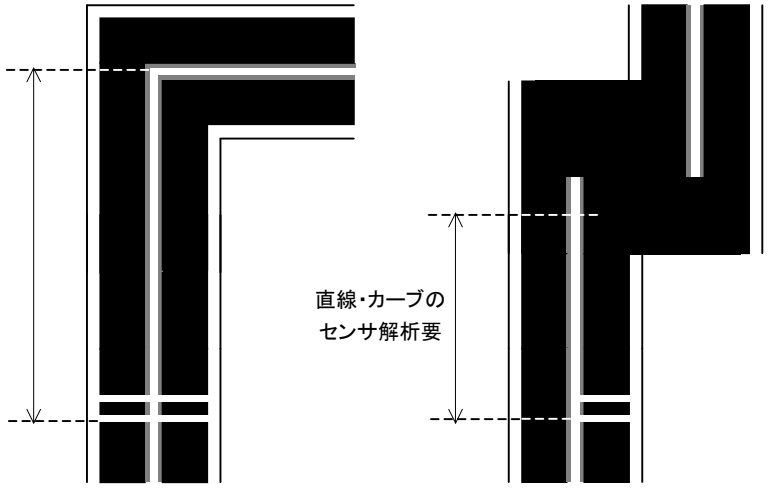


図5-2 クランク･レーンチェンジでの直線･カーブ センサ解析区間

5.4.3 レーンチェンジ

レーンチェンジは以下のモードに分割して制御します。一見、①～③までは同じコース状態ですがあえてモード分けることで、きめ細やかな速度調整をしております。

表5-4 レーンチェンジのモード一覧

No	説明(モード)	説明	ハンドル角度	速度	速度上限
①	**レーンチェンジ 0**	**レーンチェンジ確定**	-	**検討中**	**検討中**
②	**レーンチェンジ 1**	**①～250mm 移動まで**	-	**検討中**	**検討中**
③	**レーンチェンジ 2**	**②～ブラックアウト移動まで**	-	**検討中**	**検討中**
④	**レーンチェンジ 3**	**③～55mm 移動まで**	10 °	**検討中**	**検討中**
⑤	**レーンチェンジ 4**	④～220mm 移動まで	22 °	**検討中**	**検討中**
⑥	**レーンチェンジ 5**	⑤～中央白線認識まで	0 °	**検討中**	**検討中**
⑦	**通常モード**	**直線・カーブ認識**	-	**検討中**	**検討中**

(注)記載ハンドル角度で”-“は、直線・カーブと同じコース状況解析をしてハンドル角度を決定します。

(注)記載ハンドル角度は左斜線への変更時です。右折時はマイナスの値となります。

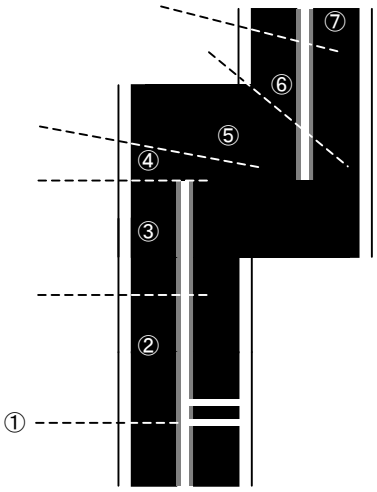


図5-3 レーンチェンジのモードイメージ

5.4.4 クランク

クランクは以下のモードに分割して制御します。一見、①～④までは同じコース状態ですがあえてモード分けることで、きめ細やかな速度調整をしております。

表5-5 クランクのモード一覧

No	説明(モード)	説明	ハンドル角度	速度	速度上限
①	**クランク** 0	**クランク確定**	-	**検討中**	**検討中**
②	**クランク** 1	①～100mm **移動まで**	-	**検討中**	**検討中**
③	**クランク** 2	②～300mm **移動まで**	-	**検討中**	**検討中**
④	**クランク** 3	③～クランク方向確定まで	-	**検討中**	**検討中**
⑤	**クランク** 4	④～急ﾌﾞﾚｰｷ(逆回転)	36 °	**検討中**	**検討中**
⑥	**クランク** 5	④～285mm 移動まで	36 °	**検討中**	**検討中**
⑦	**クランク** 6	⑤～中央白線認識まで	32 °	**検討中**	**検討中**
⑧	**クランク** 7	⑥～300mm 移動まで	-	**検討中**	**検討中**
⑨	**通常モード**	**直線・カーブ認識**	-	**検討中**	**検討中**

(注)記載ハンドル角度で”-“は、直線・カーブと同じコース状況解析をしてハンドル角度を決定します。

(注)記載ハンドル角度は左斜線への変更時です。右折時はマイナスの値となります。

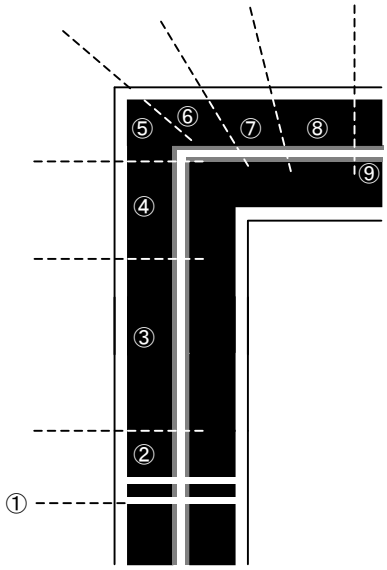


図5-4 クランクのモードイメージ

5.5 坂道

坂道は上り、下りで走行速度に影響を及ぼします。本マイコンカーは最終的に最高速度が2.7(m/秒)設定とかなり遅かったこと、また、速度監視により坂道でも一定の速度を保てることから、坂道を認識する機構がなくても影響なく動作できます。

6. 評価･デバッグ

6.1 各モードと速度

4、5 章にて、基本設計と詳細設計を行い基本的な構造が固まりました(注 1)。これから、実際にコース走行と評価を繰り返しながら、モード毎に適切な速度を検証していきます。

(注 1) 実際には 4～6 章の作業を繰り返し行うことで、最終的な実装が固まっております。

6.2 直線・カーブ

評価結果から、速度と速度上限値を以下の様に設定しました。

表6-1 直線･カーブの速度設定

説明(モード)	ハンドル角度	速度(デューティ比)	速度上限
直線	0 °	100%	2.718(m/**秒**)
カーブ左 0	5 °	90%	2.715(m/**秒**)
カーブ左 1	10 °	90%	2.360(m/**秒**)
カーブ左 2	15 °	75%	2.050(m/**秒**)
カーブ左 3	20 °	70%	1.650(m/**秒**)
カーブ左 4	25 °	65%	1.550(m/**秒**)
カーブ左 5	30 °	40%	1.450(m/**秒**)
カーブ右 0	-5 °	90%	2.715(m/**秒**)
カーブ右 1	-10 °	90%	2.360(m/**秒**)
カーブ右 2	-15 °	75%	2.050(m/**秒**)
カーブ右 3	-20 °	70%	1.650(m/**秒**)
カーブ右 4	-25 °	65%	1.550(m/**秒**)
カーブ右 5	-30 °	40%	1.450(m/**秒**)

6.3 レーンチェンジ

評価結果から、レーンチェンジでの速度と速度上限値を以下の様に設定しました。

表6-2 レーンチェンジの速度設定

説明(モード)	ハンドル角度	速度(デューティ比)	速度上限
レーンチェンジ 0	-	70%	2.650(m/**秒**)
レーンチェンジ 1	-	60%	2.400(m/**秒**)
レーンチェンジ 2	-	60%	2.100(m/**秒**)
レーンチェンジ 3	10 °	40%	1.450(m/**秒**)
レーンチェンジ 4	22 °	50%	2.100(m/**秒**)
レーンチェンジ 5	0 °	70%	2.650(m/**秒**)

(注)記載ハンドル角度で”-“は、直線・カーブと同じコース状況解析をしてハンドル角度を決定します。

(注)記載ハンドル角度は左斜線への変更時です。右折時はマイナスの値となります。

6.4 クランク

評価結果から、クランクでの速度と速度上限値を以下の様に設定しました。

表6-3 クランクの速度設定

説明(モード)	ハンドル角度	速度(デューティ比)	速度上限	備考
クランク 0	-	40%	1.250(m/**秒**)	
クランク 1	-	35%	1.200(m/**秒**)	
クランク 2	-	35%	1.100(m/**秒**)	
クランク 3	-	30%	1.000(m/**秒**)	
クランク 4	36 °	-100%	0.210(m/**秒**)	(*1)
クランク 5	36 °	75%	2.000(m/**秒**)	
クランク 6	32 °	70%	2.600(m/**秒**)	
クランク 7	-	70%	3.200(m/**秒**)	

(注)記載ハンドル角度で"-"は、直線・カーブと同じコース状況解析をしてハンドル角度を決定します。

(注)記載ハンドル角度は左斜線への変更時です。右折時はマイナスの値となります。

(*1) 急ブレーキのため、モータを逆回転させております。

6.4.1 クランクでの急ブレーキ

前述のクランクモード：クランク 4 にて、駆動モータを逆回転させて急ブレーキ動作を実現します。一般的にブレーキは駆動モータのデューティ比 0%にすることで実現しておりますが、本マイコンカーではブレーキが弱いため、クランクを走行するためには、全体的な速度を遅くする必要があります。そこで駆動モータの回転方向を逆回転させることで急ブレーキを実現します。

注意点として、本マイコンカーに取り付けている速度・距離計測用のエンコーダは 1 相のみのため回転方向がわからないという問題があります。回転方向を正回転から逆回転に変更し、実際に逆方向に走行を始めても速度は 0 以上の値を示すため、停止したことを正しく認識できません。そこで十分減速された速度 0.210(m/秒)未満に落ちるまで、逆回転(デューティ比=100%)を行い、正回転に駆動モータの回転方向を戻します。

２相式エンコーダでは問題になりません。

7. 主要ソースコード解説

7.1 共通ヘッダ

7.1.1 ry3048f_one.h

リスト7-1 共通ヘッダ(ry3048f_one.h)

```
(1 行目)   #ifndef __RY3048FONE_H
(2 行目)   #define __RY3048FONE_H

(15 行目)  /* ------------------------------------------------------------
(16 行目)         Define
(17 行目)     ------------------------------------------------------------*/
(18 行目)  /*******************************/
(19 行目)  /***    ﾓｰﾄﾞ                  ***/
(20 行目)  /*******************************/
(21 行目)  #define MD_STRAIGHT        0L              /* 直進 */
(22 行目)  #define MD_C_LEFT0         1L              /* 左 0 */
(23 行目)  #define MD_C_LEFT1         2L              /* 左 1 */
(24 行目)  #define MD_C_LEFT2         3L              /* 左 2 */
(25 行目)  #define MD_C_LEFT3         4L              /* 左 3 */
(26 行目)  #define MD_C_LEFT4         5L              /* 左 4 */
(27 行目)  #define MD_C_LEFT5         6L              /* 左 5 */
(28 行目)  #define MD_C_RIGHT0        7L              /* 右 0 */
(29 行目)  #define MD_C_RIGHT1        8L              /* 右 1 */
(30 行目)  #define MD_C_RIGHT2        9L              /* 右 2 */
(31 行目)  #define MD_C_RIGHT3       10L              /* 右 3 */
(32 行目)  #define MD_C_RIGHT4       11L              /* 右 4 */
(33 行目)  #define MD_C_RIGHT5       12L              /* 右 5 */
(34 行目)
(35 行目)  #define MD_NORMAL         12L              /* 12 未満: 通常ﾓｰﾄﾞ  */
(36 行目)
(37 行目)  #define MD_CRANK0         20L              /* ｸﾗﾝｸ確定(前) */
(38 行目)  #define MD_CRANK1         21L              /* ｸﾗﾝｸ(確定～100mm) */
(39 行目)  #define MD_CRANK2         22L              /* ｸﾗﾝｸ(100～400mm) */
(40 行目)  #define MD_CRANK3         23L              /* ｸﾗﾝｸ(方向確認) */
(41 行目)  #define MD_CRANK4         24L              /* ｸﾗﾝｸ(急ﾌﾞﾚｰｷ) */
(42 行目)  #define MD_CRANK5         25L              /* ｸﾗﾝｸ(方向変更～285mm) */
(43 行目)  #define MD_CRANK6         26L              /* ｸﾗﾝｸ(方向変更 285mm～中央白線認識) */
(44 行目)  #define MD_CRANK7         27L              /* ｸﾗﾝｸ(中央白線認識～300mm) */
(45 行目)
(46 行目)  #define MD_CC_LEFT0       40L              /* ｺｰｽ変更左(確定前) */
(47 行目)  #define MD_CC_LEFT1       41L              /* ｺｰｽ変更左(確定～250mm) */
(48 行目)  #define MD_CC_LEFT2       42L              /* ｺｰｽ変更左(250mm～ﾌﾞﾗｯｸｱｳﾄ) */
(49 行目)  #define MD_CC_LEFT3       43L              /* ｺｰｽ変更左(斜変～55mm(10°)) */
(50 行目)  #define MD_CC_LEFT4       44L              /* ｺｰｽ変更左(斜変 55～275mm(22°)) */
(51 行目)  #define MD_CC_LEFT5       45L              /* ｺｰｽ変更左(斜変 275～抜け(0°) */
```

<<続く>>

<<続き>>

```
(53 行目)   #define MD_CC_RIGHT0      50L                /* ｺｰｽ変更右(確定前) */
(54 行目)   #define MD_CC_RIGHT1      51L                /* ｺｰｽ変更右(確定～250mm) */
(55 行目)   #define MD_CC_RIGHT2      52L                /* ｺｰｽ変更右(250mm～ﾌﾞﾗｯｸｱｳﾄ) */
(56 行目)   #define MD_CC_RIGHT3      53L                /* ｺｰｽ変更右(斜変～55mm(10° )) */
(57 行目)   #define MD_CC_RIGHT4      54L                /* ｺｰｽ変更右(斜変 55～275mm(22° )) */
(58 行目)   #define MD_CC_RIGHT5      55L                /* ｺｰｽ変更右(斜変 275～抜け(0° ) */
(59 行目)
(60 行目)   /*******************************/
(61 行目)   /***    ｸﾗﾝｸ                 ***/
(62 行目)   /*******************************/
(63 行目)   #define DIR_NON           0L                 /* ｸﾗﾝｸ方向確定前 */
(64 行目)   #define DIR_RIGHT         1L                 /* ｸﾗﾝｸ方向右 */
(65 行目)   #define DIR_LEFT         -1L                 /* ｸﾗﾝｸ方向左 */
(66 行目)
(67 行目)   /*******************************/
(68 行目)   /***    ﾓｰﾀ関連              ***/
(69 行目)   /*******************************/
(70 行目)   #define RMTR_SET_FRONT    PB.DR.BIT.B3=0u    /* 右ﾓｰﾀ正転 */
(71 行目)   #define RMTR_SET_BACK     PB.DR.BIT.B3=1u    /* 右ﾓｰﾀ逆転 */
(72 行目)
(73 行目)   #define LMTR_SET_FRONT    PB.DR.BIT.B2=0u    /* 左ﾓｰﾀ正転 */
(74 行目)   #define LMTR_SET_BACK     PB.DR.BIT.B2=1u    /* 左ﾓｰﾀ逆転 */
(75 行目)
(76 行目)   #define MDB_LED3_ON       PB.DR.BIT.B6=0u    /* ﾓｰﾀﾄﾞﾗｲﾊﾞ基盤:LED3=ON  */
(77 行目)   #define MDB_LED3_OFF      PB.DR.BIT.B6=1u    /* ﾓｰﾀﾄﾞﾗｲﾊﾞ基盤:LED3=OFF */
(78 行目)
(79 行目)   #define MDB_LED2_ON       PB.DR.BIT.B7=0u    /* ﾓｰﾀﾄﾞﾗｲﾊﾞ基盤:LED2=ON  */
(80 行目)   #define MDB_LED2_OFF      PB.DR.BIT.B7=1u    /* ﾓｰﾀﾄﾞﾗｲﾊﾞ基盤:LED2=OFF */
(81 行目)
(82 行目)   #define MDB_SW1           PB.DR.BIT.B0       /* ﾓｰﾀﾄﾞﾗｲﾊﾞ基盤:SW1("0"=ON/"1"=OFF) */
(83 行目)
(84 行目)   /*******************************/
(85 行目)   /***    ｾﾝｻ関連              ***/
(86 行目)   /*******************************/
(87 行目)   #define START_SENSOR      P7.DR.BIT.B4       /* ｽﾀｰﾄﾊﾞｰ検知("1"=STOP/"0"=GO) */
(88 行目)
(89 行目)   #define COURSE_SENSOR     P7.DR.BYTE         /* ｺｰｽｾﾝｻ入力("0"=白/"1"=黒) */

(97 行目)   /*******************************/
(98 行目)   /***    ﾎﾞｰﾄﾞ資源            ***/
(99 行目)   /*******************************/
(100 行目)  #define SW1              (P6.DR.BYTE&0x0Fu) /* SW1 */
(101 行目)
(102 行目)  #define SW1_1             P6.DR.BIT.B0       /* SW1-1 ON="0" OFF="1" */
(103 行目)  #define SW1_2             P6.DR.BIT.B0       /* SW1-2 ON="0" OFF="1" */
(104 行目)  #define SW1_3             P6.DR.BIT.B0       /* SW1-3 ON="0" OFF="1" */
(105 行目)  #define SW1_4             P6.DR.BIT.B0       /* SW1-4 ON="0" OFF="1" */
(106 行目)
(107 行目)  /*******************************/
(108 行目)  /***    ｾﾝｻ用ﾊﾞｯﾌｧ           ***/
(109 行目)  /*******************************/
(110 行目)  #define FIFO_SENS_SIZE   2L
```

<<続く>>

<<続き>>

```
(112 行目)  /*******************************/
(113 行目)  /***    ｾﾝｻ周期              ***/
(114 行目)  /*******************************/
(115 行目)  #define SNS_INTERVAL     250L              /* 250us */
(116 行目)  #define MTSK_INTERVAL    500L              /* 500us */
(117 行目)
(118 行目)  /* ------------------------------------------------------------------
(119 行目)         Extern
(120 行目)     ------------------------------------------------------------------*/
(121 行目)  extern UB fifo_sensor_data[];
(122 行目)  extern H fifo_sensor_p;
(123 行目)
(124 行目)  /* ------------------------------------------------------------------
(125 行目)         Prototype
(126 行目)     ------------------------------------------------------------------*/
(127 行目)  void ry3048fone_init(void);
(128 行目)
(129 行目)  #endif /* __RY3048FONE_H */
```

7.2 PWM 制御ドライバ

7.2.1 pwm.h

リスト7-2 PWM制御ドライバヘッダ(pwm.h)

```
(1 行目)   #ifndef __PWM_H
(2 行目)   #define __PWM_H

(15 行目)  /* ------------------------------------------------------------------
(16 行目)         Define
(17 行目)     ------------------------------------------------------------------*/
(18 行目)  #define PWM_CYCTIME       0x53FFu       /* Pφ=24.567MHz φ/8 ==>  7.0msec */
(19 行目)
(20 行目)  #define SERVO_CENTER      3124u         /* ｻｰﾎﾞｾﾝﾀ  SANWA */
(21 行目)  #define SERVO_1DEGREE     26u           /* ｻｰﾎﾞ PWM 1° 分移動量  */
(22 行目)
(23 行目)  #define SERVO_RMAX        2084u         /* 切角右 SANWA */
(24 行目)  #define SERVO_LMAX        4164u         /* 切角左 SANWA */
(25 行目)
(26 行目)  /* ------------------------------------------------------------------
(27 行目)         Prototype
(28 行目)     ------------------------------------------------------------------*/
(29 行目)  W curve_get_insp(W angle, W rate);
(30 行目)  void srv_set_pmw(UH data);
(31 行目)  void pwm_init(void);
(32 行目)  void set_handle(W angle);
(33 行目)  void srv_set_pmw(UH data);
(34 行目)  UH srv_get_pwmdata(void);
(35 行目)  void srv_set_center(void);
(36 行目)  void rmtr_set_pmw(UH data);
(37 行目)  UH rmtr_get_pwmdata(void);
(38 行目)  void lmtr_set_pmw(UH data);
(39 行目)  UH lmtr_get_pwmdata(void);
(40 行目)  void mtr_set_rate(W rrate, W lrate);
(41 行目)  void mtr_break(W speed);
(42 行目)  W mtr_break_mode(W rrate, W lrate, W speed);
(43 行目)
(44 行目)  void srv_add_angle(UH degree); /* for DEBUG */
(45 行目)  void srv_del_angle(UH degree); /* for DEBUG */
(46 行目)  void srv_add_count(UH count);  /* for DEBUG */
(47 行目)  void srv_del_count(UH count);  /* for DEBUG */
(48 行目)
(49 行目)  #endif /* __PWM_H */
```

7.2.2 pwm.c

リスト7-3 PWM制御ドライバ (pwm.c)

```
(12 行目)   #include "slos.h"
(13 行目)   #include "iodefine.h"
(14 行目)   #include "pwm.h"
(15 行目)   #include "ry3048f_one.h"
(16 行目)    #include "encoder.h"

(18 行目)   const W innersp_rate[] = {
(19 行目)       /*  0 */   1000L,
(20 行目)       /*  1 */    983L,
(21 行目)       /*  2 */    966L,
(22 行目)       /*  3 */    949L,

                    (中略)

(61 行目)       /* 42 */    370L,
(62 行目)       /* 43 */    335L,
(63 行目)       /* 44 */    339L,
(64 行目)       /* 45 */    322L,
(65 行目)   };

(75 行目)   W curve_get_insp(W angle, W rate)
(76 行目)   {
(77 行目)       W data;
(78 行目)
(79 行目)       data  = rate * innersp_rate[angle];
(80 行目)       data /= 1000L;
(81 行目)
(82 行目)       return(data);
(83 行目)   }

(93 行目)   void srv_set_pmw(UH data)
(94 行目)   {
(95 行目)       ITU4.BRB = data;                /* ｻｰﾎﾞ PWM 設定    */
(96 行目)   }
```

外輪/内輪差 計算用の変換テーブル
配列の添字はハンドル角度に相応します。

ハンドル角度と外輪のデューティ比から、
内輪のデューティ比を求めます。

プライベート関数です。
サーボ用PWMデューティ比を設定します。

<<続く>>

<<続き>>

```
(105 行目)    void pwm_init(void)
(106 行目)    {
(107 行目)        /*
(108 行目)        /*    PB5(O) = ｻｰﾎﾞ PWM (TIOCXB4)
(109 行目)         *    PB4(O) = 右ﾓｰﾀ PWM(TIOXA4)
(110 行目)         *    PB1(O) = 左ﾓｰﾀ PWM(TIOCB3)
(111 行目)         */
(112 行目)        ITU.TSTR.BIT.STR3      = 0u;       /* ITU3 停止 */
(113 行目)
(114 行目)        ITU3.TCR.BIT.CCLR      = 1u;       /* GRA ｺﾝﾍﾟｱﾏｯﾁで TCNT ｸﾘｱ */
(115 行目)        ITU3.TCR.BIT.CKEG      = 0u;       /* ｸﾘｯｸｴｯｼﾞ=立上り */
(116 行目)        ITU3.TCR.BIT.TPSC      = 3u;       /* ｸﾛｯｸｿｰｽ=Pφ/8 */
(117 行目)
(118 行目)        ITU.TFCR.BIT.CMD       = 3u;       /* ch3-4 組合 ﾘｾｯﾄ同期 PWM ﾓｰﾄﾞ */
(119 行目)        ITU.TFCR.BIT.BFB4      = 1u;       /* GRB4-BRB4 はﾊﾞｯﾌｧ動作 */
(120 行目)        ITU.TFCR.BIT.BFA4      = 1u;       /* GRA4-BRA4 はﾊﾞｯﾌｧ動作 */
(121 行目)        ITU.TFCR.BIT.BFB3      = 1u;       /* GRB3-BRB3 はﾊﾞｯﾌｧ動作 */
(122 行目)        ITU.TFCR.BIT.BFA3      = 0u;       /* GRA3 は通常動作 */
(123 行目)
(124 行目)        ITU3.TCNT              = 0u;       /* TCNT ｸﾘｱ */
(125 行目)        ITU3.GRA               = PWM_CYCTIME;  /* Pφ=24.567MHz φ/8 ==> 16msec */
(126 行目)
(127 行目)        ITU3.GRB               = 0u;       /* 左ﾓｰﾀ PWM 設定  */
(128 行目)        ITU3.BRB               = 0u;
(129 行目)
(130 行目)        ITU4.GRA               = 0u;       /* 右ﾓｰﾀ PWM 設定  */
(131 行目)        ITU4.BRA               = 0u;
(132 行目)
(133 行目)        ITU4.GRB               = SERVO_CENTER; /* ｻｰﾎﾞ PWM 設定   */
(134 行目)        ITU4.BRB               = SERVO_CENTER;
(135 行目)
(136 行目)        ITU3.TIER.BIT.IMIEA    = 0u;       /* ITU3 ｺﾝﾍﾟｱﾏｯﾁ A 割込(基準) =Disable */
(137 行目)        ITU3.TIER.BIT.IMIEB    = 0u;       /* ITU3 ｺﾝﾍﾟｱﾏｯﾁ B 割込(左ﾓｰﾀ)=Disable */
(138 行目)        ITU4.TIER.BIT.IMIEA    = 0u;       /* ITU4 ｺﾝﾍﾟｱﾏｯﾁ A 割込(右ﾓｰﾀ)=Disable */
(139 行目)        ITU4.TIER.BIT.IMIEB    = 0u;       /* ITU4 ｺﾝﾍﾟｱﾏｯﾁ B 割込(ｻｰﾎﾞ) =Disable */
(140 行目)
(141 行目)        INTC.IPRB.BIT._ITU3    = 1u;       /* 割込ﾌﾟﾗｲｵﾘﾃｨﾚﾍﾞﾙ=1 */
(142 行目)        INTC.IPRB.BIT._ITU4    = 1u;       /* 割込ﾌﾟﾗｲｵﾘﾃｨﾚﾍﾞﾙ=1 */
(143 行目)
(144 行目)        ITU.TOER.BIT.EXB4      = 1u;       /* TOCXB4 出力=Enable */
(145 行目)        ITU.TOER.BIT.EXA4      = 1u;       /* TOCXA4 出力=Enable */
(146 行目)        ITU.TOER.BIT.EB3       = 1u;       /* TIOCB3 出力=Enable */
(147 行目)        ITU.TOER.BIT.EB4       = 0u;       /* TIOCB4 出力=Disable */
(148 行目)        ITU.TOER.BIT.EA4       = 0u;       /* TIOCA4 出力=Disable */
(149 行目)        ITU.TOER.BIT.EA3       = 0u;       /* TIOCA3 出力=Disable */
(150 行目)
(151 行目)        ITU.TSTR.BIT.STR3      = 1u;       /* ITU3 ｽﾀｰﾄ */
(152 行目)    }
```

PWM制御の初期化をします。

<<続く>>

<<続き>>

```
(162 行目)    void set_handle(W angle)
(163 行目)    {
(164 行目)        W wdata;
(165 行目)
(166 行目)        if(angle == 0L) {
(167 行目)            srv_set_pmw((UH)SERVO_CENTER); /* ｻｰﾎﾞ PWM 設定    */
(168 行目)        } else {
(169 行目)            wdata  = (W)SERVO_CENTER;

(171 行目)            wdata += ((W)SERVO_1DEGREE * angle);    /* ERG-WZ */
(172 行目)            wdata  = (W)((UW)wdata & 0x0000FFFFuL);

(180 行目)            if((wdata < (W)SERVO_RMAX) || (wdata > (W)SERVO_LMAX)) {
(181 行目)                if(angle > 0L) {
(182 行目)                    wdata = (W)SERVO_LMAX;
(183 行目)                } else {
(184 行目)                    wdata = (W)SERVO_RMAX;
(185 行目)                }
(186 行目)            }
(187 行目)            srv_set_pmw((UH)wdata);         /* ｻｰﾎﾞ PWM 設定    */
(188 行目)        }
(189 行目)    }

(222 行目)    #pragma inline(rmtr_set_pmw)
(223 行目)    void rmtr_set_pmw(UH data)
(224 行目)    {
(225 行目)        ITU4.BRA = data;                /* 右ﾓｰﾀ PWM 設定    */
(226 行目)    }

(247 行目)    #pragma inline(lmtr_set_pmw)
(248 行目)    void lmtr_set_pmw(UH data)
(249 行目)    {
(250 行目)        ITU3.BRB = data;                /* 左ﾓｰﾀ PWMPWM 設定    */
(251 行目)    }

(272 行目)    void mtr_set_rate(W rrate, W lrate)
(273 行目)    {
(274 行目)        W rwdata, lwdata;
(275 行目)
(276 行目)        if(rrate >= 100L) {
(277 行目)            rwdata = (W)PWM_CYCTIME -1L;
(278 行目)        } else {
(279 行目)            rwdata = (W)PWM_CYCTIME * rrate / 100L;
(280 行目)        }
(281 行目)        if(lrate >= 100L) {
(282 行目)            lwdata = (W)PWM_CYCTIME -1L;
(283 行目)        } else {
(284 行目)            lwdata = (W)PWM_CYCTIME * lrate / 100L;
(285 行目)        }
(286 行目)
(287 行目)        RMTR_SET_FRONT;
(288 行目)        LMTR_SET_FRONT;
(289 行目)        rmtr_set_pmw((UH)rwdata);       /* 右ﾓｰﾀ PWM 設定    */
(290 行目)        lmtr_set_pmw((UH)lwdata);       /* 左ﾓｰﾀ PWM 設定    */
(291 行目)    }
```

ハンドル角度を設定します。

ハンドル角度ら、
ITU設定値を求めます。

切れ角範囲でPWM
設定します。

プライベート関数です。駆動モータ用
PWMデューティ比(右)を設定します。

プライベート関数です。駆動モータ用
PWMデューティ比(左)を設定します。

左右駆動モータのPWMデューティ比
を設定します。

デューティ比から、
ITU設定値を求めます。

モータ回転方向を正回転とします。

<<続く>>

<<続き>>

```
(331 行目)    W mtr_break_mode(W rrate, W lrate, W speed)
(332 行目)    {
(333 行目)        W ercd;
(334 行目)
(335 行目)        rrate = - rrate;
(336 行目)        lrate = -–lrate;
(337 行目)
(338 行目)        if(rrate >= 100L) {
(339 行目)            rrate = (W)PWM_CYCTIME -1L;
(340 行目)        } else {
(341 行目)            rrate = (W)PWM_CYCTIME * rwdata / 100L;
(342 行目)        }
(343 行目)        if(lrate >= 100L) {
(344 行目)            lrate = (W)PWM_CYCTIME -1L;
(345 行目)        } else {
(346 行目)            lrate = (W)PWM_CYCTIME * lwdata / 100L;
(347 行目)        }
(348 行目)
(349 行目)        if(get_encoder_speed() > speed) {
(350 行目)            RMTR_SET_BACK;                      /* 逆回転 */
(351 行目)            LMTR_SET_BACK;                      /* 逆回転 */
(352 行目)
(353 行目)            rmtr_set_pmw((UH)(PWM_CYCTIME-1u));  /* ｻｰﾎﾞ PWM 設定(100%) */
(354 行目)            lmtr_set_pmw((UH)(PWM_CYCTIME-1u));  /* ｻｰﾎﾞ PWM 設定(100%) */
(355 行目)            ercd = 0L;
(356 行目)        } else {
(357 行目)            RMTR_SET_FRONT;                      /* 正回転 */
(358 行目)            LMTR_SET_FRONT;                      /* 正回転 */
(359 行目)
(360 行目)            rmtr_set_pmw((UH)rrate);             /* ｻｰﾎﾞ PWM 設定   */
(361 行目)            lmtr_set_pmw((UH)lrate);             /* ｻｰﾎﾞ PWM 設定   */
(362 行目)
(363 行目)            ercd = -1L;
(364 行目)        }
(365 行目)        return(ercd);
(366 行目)    }
```

急ブレーキ(逆回転)モード設定します。
上限速度と正回転時の左右デューティ比が引数です。

正回転時のデューティ比からITU設定値を求めます。

上限速度を超えるとき、逆回転

上限速度以下のとき、正回転

7.3 エンコーダドライバ

7.3.1 encoder.h

リスト7-4 エンコーダドライバヘッダ(encoder.h)

```
(1 行目)    #ifndef __ENCODER_H
(2 行目)    #define __ENCODER_H
(3 行目)    /***********************************************************************
(4 行目)     * Micom Car Rally Program
(5 行目)     *
(6 行目)     * Copyright (c) Renesas Northern Japan Semiconductor,Inc. 2008
(7 行目)     * Licensed Material of Renesas Northern Japan Semiconductor,Inc.
(8 行目)     ***********************************************************************
(9 行目)     * File:         encoder.h
(10 行目)    * Description: encode input
(11 行目)    * Created:     Aug 30, 2008   V0.00.00
(12 行目)    ***********************************************************************
(13 行目)    */
(14 行目)
(15 行目)    /* ---------------------------------------------------------------------
(16 行目)           Extern
(17 行目)       ---------------------------------------------------------------------*/
(18 行目)    extern UW  gEncTotal;          /* ﾄｰﾀﾙｶｳﾝﾄ数 */
(19 行目)    extern UW  gEncLen;            /* 距離 */
(20 行目)    extern W   gEncCur;            /* 単位時間辺りｶﾚﾝﾄ回転数 */
(21 行目)    extern UH  gEncBtcnt;          /* 前回測定時ﾄｰﾀﾙｶｳﾝﾄ数 */
(22 行目)    extern W   gEncSpeed;          /* 現在の速度 */
(23 行目)    /* ---------------------------------------------------------------------
(24 行目)           Prototype
(25 行目)       ---------------------------------------------------------------------*/
(26 行目)    void encoder_init(void);
(27 行目)    void get_encoder_count(void);
(28 行目)    W get_encoder_total_len(void);
(29 行目)    W get_encoder_len(void);
(30 行目)    void encoder_len_clr(void);
(31 行目)    W get_encoder_speed(void);
(32 行目)
(33 行目)    #endif /* __ENCODER_H */
```

7.3.2 encoder.c

リスト7-5 エンコーダドライバ(encoder.c)

```
(12 行目)   #include <machine.h>
(13 行目)   #include "slos.h"
(14 行目)   #include "iodefine.h"
(15 行目)
(16 行目)   #include "ry3048f_one.h"
(17 行目)   #include "encoder.h"
(18 行目)
(19 行目)   UW  gEncTotal;          /* ﾄｰﾀﾙｶｳﾝﾄ数 */
(20 行目)   UW  gEncLen;            /* 距離 */
(21 行目)   W   gEncCur;            /* 単位時間辺りｶﾚﾝﾄ回転数 */
(22 行目)   UH  gEncBtcnt;          /* 前回測定時ﾄｰﾀﾙｶｳﾝﾄ数 */
(23 行目)   W   gEncSpeed;          /* 現在の速度 */

(32 行目)   void encoder_init(void)
(33 行目)   {
(34 行目)       gEncTotal  = 0uL;
(35 行目)       gEncLen    = 0uL;
(36 行目)       gEncCur    = 0L;
(37 行目)       gEncBtcnt  = 0u;
(38 行目)
(39 行目)       ITU.TSTR.BIT.STR0      = 0u;        /* ITU0 停止 */
(40 行目)
(41 行目)       ITU0.TCR.BIT.CCLR      = 0u;        /* TCNT ｸﾘｱ禁止 */
(42 行目)       ITU0.TCR.BIT.CKEG      = 2u;        /* ｸﾘｯｸｴｯｼﾞ=両ｴｯｼﾞ */
(43 行目)       ITU0.TCR.BIT.TPSC      = 4u;        /* ｸﾛｯｸｿｰｽ=TCLKA */
(44 行目)
(45 行目)       ITU0.TCNT              = 0u;        /* TCNT ｸﾘｱ */
(46 行目)
(47 行目)       ITU.TSTR.BIT.STR0      = 1u;        /* ITU0 ｽﾀｰﾄ */
(48 行目)   }

(57 行目)   void get_encoder_count(void)
(58 行目)   {
(59 行目)       /* 単位時間 = 10msec */
(60 行目)       UH tcnt;
(61 行目)
(62 行目)       tcnt     = ITU0.TCNT;
(63 行目)       gEncCur  = (W)((W)tcnt - gEncBtcnt);
(64 行目)       if(gEncCur < 0L) {
(65 行目)           gEncCur = -gEncCur;
(66 行目)       }
(67 行目)       gEncTotal += (UW)gEncCur;
(68 行目)       gEncLen   += (UW)gEncCur;
(69 行目)       gEncBtcnt  = tcnt;
(70 行目)   }
```

エンコーダ入力の初期化

10(m秒)毎に呼び出すことで、
速度・距離情報を更新

<<続く>>

<<続き>>

```
(80 行目)    W get_encoder_total_len(void)
(81 行目)    {
(82 行目)        UW data;
(83 行目)
(84 行目)        data   = gEncTotal;
(85 行目)        data  *= 9165uL;
(86 行目)        data  /= 10000uL;
(87 行目)
(88 行目)        return((W)data);
(89 行目)    }

(98 行目)    W get_encoder_len(void)
(99 行目)    {
(100 行目)       UW data;

(108 行目)       data   = gEncLen;
(109 行目)       data  *= 9165uL;
(110 行目)       data  /= 10000uL;
(111 行目)
(112 行目)       return((W)data);
(113 行目)   }

(122 行目)   void encoder_len_clr(void)
(123 行目)   {
(124 行目)       UB ccr;
(125 行目)
(126 行目)       ccr = get_ccr();
(127 行目)       set_ccr(ccr | 0xC0);
(128 行目)
(129 行目)       gEncLen = 0uL;
(120 行目)
(121 行目)       set_ccr(ccr);
(122 行目)   }

(142 行目)    W get_encoder_speed(void)
(143 行目)   {
(144 行目)       UW data;
(145 行目)
(146 行目)       data   = gEncCur;
(147 行目)       data  *= 9165uL;
(148 行目)       data  /= 100uL;
(159 行目)
(150 行目)       return((W)data);
(151 行目)   }
```

累計距離を取得します

指定位置からの距離を取得します

指定位置からの距離用のカウンタを
クリアします。

速度を取得します

7.4 周期タイマハンドラ(ITU2)

7.4.1 itu2.h

リスト7-6 周期タイマハンドラヘッダ(itu2.h)

```
 (1 行目)   #ifndef __ITU2_H
 (2 行目)   #define __ITU2_H

(15 行目)   /* ------------------------------------------------------------------
(16 行目)          Prototype
(17 行目)      ------------------------------------------------------------------*/
(18 行目)   void itu2_init(void);
(19 行目)   void intITU2(H mode);
(20 行目)
(21 行目)   #endif /* __ITU2_H */
```

7.4.2 itu2.c

リスト7-7 周期タイマハンドラ(itu2.c)

```
(14 行目)   #include "slos.h"
(15 行目)   #include "iodefine.h"

(17 行目)   #include "ry3048f_one.h"
(18 行目)   #include "itu2.h"
(19 行目)    #include "encoder.h"

(30 行目)   void itu2_init(void)
(31 行目)   {
(32 行目)       ITU.TSTR.BIT.STR2       = 0u;       /* ITU2 停止 */
(33 行目)
(34 行目)       ITU2.TCR.BIT.CCLR       = 1u;       /* GRA ｺﾝﾍﾟｱﾏｯﾁで TCNT ｸﾘｱ */
(35 行目)       ITU2.TCR.BIT.CKEG       = 0u;       /* ｸﾘｯｸｴｯｼﾞ=立上り */
(36 行目)        ITU2.TCR.BIT.TPSC       = 3u;       /* ｸﾛｯｸｿｰｽ=Pφ/8 */
(37 行目)
(38 行目)       ITU2.TCNT               = 0u;       /* TCNT ｸﾘｱ */

(40 行目)       ITU2.GRA                = 767u;     /* Pφ=24.567MHz φ/8 ==> 250usec */
(41 行目)
(42 行目)       ITU2.TIER.BIT.OVIE      = 0u;       /* ｵｰﾊﾞｰﾌﾛｰ割込=Disable */
(43 行目)       ITU2.TIER.BIT.IMIEB     = 0u;       /* ｺﾝﾍﾟｱﾏｯﾁB 割込=Disable */
(44 行目)       ITU2.TIER.BIT.IMIEA     = 1u;       /* ｺﾝﾍﾟｱﾏｯﾁB 割込=Enable */
(45 行目)
(46 行目)       INTC.IPRA.BIT._ITU2     = 0u;       /* 割込ﾌﾟﾗｲｵﾘﾃｨﾚﾍﾞﾙ=0 */
(47 行目)
(48 行目)       ITU.TSTR.BIT.STR2       = 1u;       /* ITU2 ｽﾀｰﾄ */
(49 行目)    }

(58 行目)   #pragma noregsave(intITU2)
(59 行目)    void intITU2(H mode)
(60 行目)   {
(61 行目)       ITU2.TSR.BIT.IMFA       = 0u;       /* 割込要因ｸﾘｱ */
(62 行目)
(63 行目)       /* ｾﾝｻﾃﾞｰﾀ読込 */
(64 行目)       fifo_sensor_p = (fifo_sensor_p + 1) % FIFO_SENS_SIZE;
(65 行目)       fifo_sensor_data[fifo_sensor_p] = COURSE_SENSOR;
(66 行目)
(67 行目)       slos_cyclic_timer();
(68 行目)
(69 行目)       if((knl_systim % (10L*1000L)) == 0uL) {
(70 行目)           get_encoder_count();
(71 行目)       }
(72 行目)       disp(mode);
(73 行目)   }
```

ITU2を初期化します

周期=250(u秒)

周期タイマハンドラ(250u秒周期)

割込要因クリア

センサ入力を
FIFOバッファ保存

周期タイマハンドラ処理(OSサービスコール)

エンコーダ入力監視(10m秒周期)

割込みの終了(OSサービスコール)

7.5 マイコンカー制御プログラム

7.5.1 user.c

リスト7-8 マイコンカー制御プログラム(user.c)

```
(12 行目)   #include <machine.h>
(13 行目)   #include <string.h>
(14 行目)   #include "slos.h"
(15 行目)   #include "iodefine.h"
(16 行目)   #include "ry3048f_one.h"
(17 行目)   #include "pwm.h"
(18 行目)   #include "sci.h"
(19 行目)   #include "itu2.h"
(20 行目)   #include "encoder.h"
(21 行目)
(22 行目)   #include "lcd.h"
(23 行目)   #include "eeprom.h"
(24 行目)
(25 行目)   #include "common.h"

(45 行目)   W gHandle_Cur;      /* ﾊﾝﾄﾞﾙ角度(ｶﾚﾝﾄ) */
(46 行目)   W gHandle_Next;     /* ﾊﾝﾄﾞﾙ角度(次設定値) */
(47 行目)   W gSpeed_Cur;       /* 速度(ﾃﾞｭｰﾃｨ) */
(48 行目)   W gMode;            /* ﾓｰﾄﾞ */
(49 行目)   W gSpeed_Max;       /* 最高速度 */

(52 行目)   typedef struct _spset {
(53 行目)       W speed;
(54 行目)       W over;
(55 行目)       W brk;
(56 行目)   } SPSET;
(57 行目)
(58 行目)   const SPSET spset[] = {
(59 行目)       {   100L,     2718L,     0L  },          /* 直進 */
(60 行目)
(61 行目)       {    90L,     2715L,     0L  },          /* 左0 */
(62 行目)       {    90L,     2360L,     0L  },          /* 左1 */
(63 行目)       {    75L,     2050L,     0L  },          /* 左2 */
(64 行目)       {    70L,     1650L,     0L  },          /* 左3 */
(65 行目)       {    65L,     1550L,     0L  },          /* 左4 */
(66 行目)       {    45L,     1450L,     0L  },          /* 左5 */

                    (中略)

(126 行目)      {    0L,       0L,     0L  }
(127 行目)  };
```

グローバル変数

モード毎の速度(デューティー比)，上限速度，
急ブレーキ状態を定義するテーブル

モード毎の速度定義

<<続く>>

<<続き>>

```
(136 行目)    void uinit(void)
(137 行目)    {
(138 行目)        /* initialize */
(139 行目)        ry3048fone_init();              /* H8/3048 初期化 */
(140 行目)        sci_init((B)1);                 /* H8/3048 ｼﾘｱﾙ初期化 */
(141 行目)
(142 行目)        initEeprom();                   /* 拡張ﾎﾞｰﾄﾞ EEPROM 初期化 */
(143 行目)        lcd_init();                     /* 拡張ﾎﾞｰﾄﾞ LCD 初期化 */
(144 行目)
(145 行目)        RMTR_SET_FRONT;
(146 行目)        LMTR_SET_FRONT;
(147 行目)
(148 行目)        MDB_LED2_OFF;
(149 行目)        MDB_LED3_OFF;
(150 行目)
(151 行目)        pwm_init();                     /* H8/3048 PWM 初期化 */
(152 行目)        itu2_init();                    /* H8/3048 ITU2 初期化(ｼｽﾃﾑﾀｲﾏ) */
(153 行目)        encoder_init();                 /* ｴﾝｺｰﾀﾞ初期化 */
(154 行目)
(155 行目)        gHandle_Cur  = 0L;
(156 行目)        gHandle_Next = 0L;
(157 行目)        mtr_set_rate(0L, 0L);
(158 行目)
(159 行目)        gSpeed_Cur = 0L;
(160 行目)    }

(169 行目)    #pragma noregsave(tsk_handle)
(170 行目)    void tsk_handle(void)
(171 行目)    {
(172 行目)        W speed_cur;
(173 行目)        W flg;
(174 行目)        W i;
(175 行目)
(176 行目)        gSpeed_Max = 0L;
(177 行目)        set_handle(0L);
(178 行目)        mtr_set_rate(0L, 0L);
(179 行目)        /*************************/
(180 行目)        /* ｽﾀﾝﾊﾞｲ                */
(181 行目)        /*************************/
(182 行目)        flg = 0L;
(183 行目)        while(flg==0L) {
(184 行目)            MDB_LED2_ON;
(185 行目)            MDB_LED3_OFF;
(186 行目)            if(flg == 0L) {
(187 行目)                for(i=0L;i<50L;i++) {
(188 行目)                    if(MDB_SW1 == 0u) {
(189 行目)                        flg++;
(190 行目)                        i=50L;
(191 行目)                    }
(192 行目)                    tslp_tsk(10L*1000L);
(193 行目)                }
(194 行目)            }
```

ユーザ初期化コールバック

デバッグ用

制御メインタスク

サーボセンタリング、駆動モータ停止

SW1=ONまでLED2,3点滅(500m秒)

<<続く>>

《続き》

```
(195 行目)            MDB_LED2_OFF;
(196 行目)            MDB_LED3_ON;
(197 行目)            if(flg == 0L) {
(198 行目)                for(i=0L;i<50L;i++) {
(199 行目)                    if(MDB_SW1 == 0u) {
(200 行目)                        flg++;
(201 行目)                        i=50L;
(202 行目)                    }
(203 行目)                    tslp_tsk(10L*1000L);
(204 行目)                }
(205 行目)            }
(206 行目)        }
(207 行目)
(208 行目)        /*************************/
(209 行目)        /* ｽﾀｰﾄﾊﾞｰﾁｪｯｸ            */
(210 行目)        /*************************/
(211 行目)        wup_tsk((UB)2u);
(212 行目)         MDB_LED2_OFF;
(213 行目)        MDB_LED3_OFF;
(214 行目)        while(START_SENSOR == 0u) {
(215 行目)            tslp_tsk(25OL);           /* Wait */
(216 行目)        }
(217 行目)        /*************************/
(218 行目)        /* ｽﾀｰﾄ直後                */
(219 行目)        /*************************/
(220 行目)        mtr_set_rate(70L, 70L);
(221 行目)        tslp_tsk(150L);
(222 行目)
(223 行目)        /*************************/
(224 行目)        /* ﾒｲﾝﾙｰﾌﾟ                 */
(225 行目)        /*************************/
(226 行目)        while(get_encoder_total_len() <  80L*1000L) {  /*  80m 間動作 */

(229 行目)            /*************************/
(230 行目)            /* ﾊﾝﾄﾞﾙ PWM 設定          */
(231 行目)            /*************************/
(232 行目)            if(gHandle_Cur != gHandle_Next) {
(233 行目)                gHandle_Cur = gHandle_Next;
(234 行目)                 set_handle(gHandle_Cur);
(235 行目)            }
(236 行目)            /*************************/
(237 行目)            /* ﾓｰﾀ PWM 設定            */
(238 行目)            /*************************/
(239 行目)             if(START_SENSOR != 0u) {
(240 行目)                /* 最高速度ﾁｪｯｸ(for Debug) */
(241 行目)                speed_cur = get_encoder_speed();
(242 行目)                if(speed_cur >  gSpeed_Max) {
(243 行目)                    gSpeed_Max = speed_cur;
(244 行目)                }
```

コース状況解析タスク起動
(LED2,3=OFF)

スタートバーチェック

スタートバーオープン直後は
駆動モータ 70%

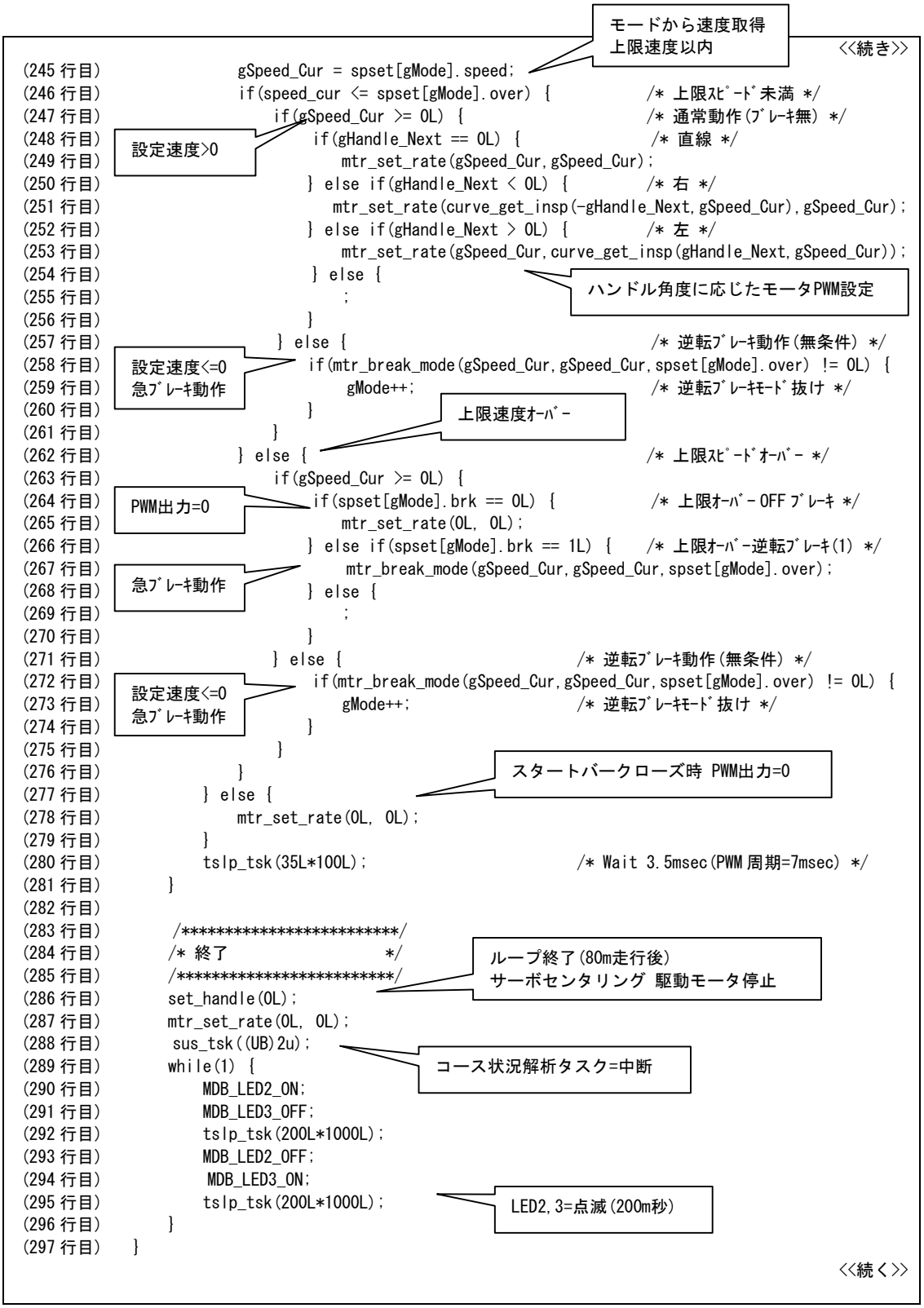
コース全長 80m間
メインループ開始

ハンドル角度変更有のとき
ハンドル角度設定

スタートバーオープン時の処理

最高速度チェック
(評価用)

《続く》




```
(245 行目)              gSpeed_Cur = spset[gMode].speed;
(246 行目)              if(speed_cur <= spset[gMode].over) {          /* 上限ｽﾋﾟｰﾄﾞ未満 */
(247 行目)                  if(gSpeed_Cur >= 0L) {                    /* 通常動作(ﾌﾞﾚｰｷ無) */
(248 行目)                      if(gHandle_Next == 0L) {               /* 直線 */
(249 行目)                          mtr_set_rate(gSpeed_Cur,gSpeed_Cur);
(250 行目)                      } else if(gHandle_Next < 0L) {         /* 右 */
(251 行目)                         mtr_set_rate(curve_get_insp(-gHandle_Next,gSpeed_Cur),gSpeed_Cur);
(252 行目)                      } else if(gHandle_Next > 0L) {         /* 左 */
(253 行目)                          mtr_set_rate(gSpeed_Cur,curve_get_insp(gHandle_Next,gSpeed_Cur));
(254 行目)                      } else {
(255 行目)                          ;
(256 行目)                      }
(257 行目)                  } else {                                   /* 逆転ﾌﾞﾚｰｷ動作(無条件) */
(258 行目)                      if(mtr_break_mode(gSpeed_Cur,gSpeed_Cur,spset[gMode].over) != 0L) {
(259 行目)                          gMode++;                           /* 逆転ﾌﾞﾚｰｷﾓｰﾄﾞ抜け */
(260 行目)                      }
(261 行目)                  }
(262 行目)              } else {                                       /* 上限ｽﾋﾟｰﾄﾞｵｰﾊﾞｰ */
(263 行目)                  if(gSpeed_Cur >= 0L) {
(264 行目)                      if(spset[gMode].brk == 0L) {           /* 上限ｵｰﾊﾞｰOFF ﾌﾞﾚｰｷ */
(265 行目)                          mtr_set_rate(0L, 0L);
(266 行目)                      } else if(spset[gMode].brk == 1L) {    /* 上限ｵｰﾊﾞｰ逆転ﾌﾞﾚｰｷ(1) */
(267 行目)                          mtr_break_mode(gSpeed_Cur,gSpeed_Cur,spset[gMode].over);
(268 行目)                      } else {
(269 行目)                          ;
(270 行目)                      }
(271 行目)                  } else {                                   /* 逆転ﾌﾞﾚｰｷ動作(無条件) */
(272 行目)                      if(mtr_break_mode(gSpeed_Cur,gSpeed_Cur,spset[gMode].over) != 0L) {
(273 行目)                          gMode++;                           /* 逆転ﾌﾞﾚｰｷﾓｰﾄﾞ抜け */
(274 行目)                      }
(275 行目)                  }
(276 行目)              }
(277 行目)          } else {
(278 行目)              mtr_set_rate(0L, 0L);
(279 行目)          }
(280 行目)          tslp_tsk(35L*100L);                               /* Wait 3.5msec(PWM 周期=7msec) */
(281 行目)      }
(282 行目)
(283 行目)      /*************************/
(284 行目)      /* 終了                  */
(285 行目)      /*************************/
(286 行目)      set_handle(0L);
(287 行目)      mtr_set_rate(0L, 0L);
(288 行目)      sus_tsk((UB)2u);
(289 行目)      while(1) {
(290 行目)          MDB_LED2_ON;
(291 行目)          MDB_LED3_OFF;
(292 行目)          tslp_tsk(200L*1000L);
(293 行目)          MDB_LED2_OFF;
(294 行目)          MDB_LED3_ON;
(295 行目)          tslp_tsk(200L*1000L);
(296 行目)      }
(297 行目)  }
```

<<続く>>

<<続き>>

```
(306 行目)    void check_curve(UB sensor, W *mode, W *handle)
(307 行目)    {
(308 行目)        *mode   = -1L;
(309 行目)        *handle = -180L;
(310 行目)        switch(sensor) {
(311 行目)          /* 直線 ----------------------------------*/
(312 行目)          case 0xF7u:  /* 直線 */
(313 行目)            *mode  = MD_STRAIGHT;
(314 行目)            *handle= 0L;
(315 行目)            break;
(316 行目)          /* カーブ左 ----------------------------------*/
(317 行目)          case 0x7Fu:  /* カーブ左 5 */
(318 行目)            *mode  = MD_C_LEFT5;
(319 行目)            *handle= 30L;
(320 行目)            break;
(321 行目)          case 0x3Fu:  /* カーブ左 4 */
(322 行目)            *mode  = MD_C_LEFT4;
(323 行目)            *handle= 25L;
(324 行目)           break;
(325 行目)          case 0xBFu:  /* カーブ左 3 */
(326 行目)            *mode  = MD_C_LEFT3;
(327 行目)            *handle= 20L;
(328 行目)            break;
(329 行目)          case 0x9Fu:  /* カーブ左 2 */
(330 行目)            *mode  = MD_C_LEFT2;
(331 行目)            *handle= 15L;
(332 行目)            break;
(333 行目)          case 0xDFu:  /* カーブ左 1 */
(334 行目)            *mode  = MD_C_LEFT1;
(335 行目)            *handle= 10L;
(336 行目)            break;
(337 行目)          case 0xD7u:  /* カーブ左 0 */
(338 行目)            *mode  = MD_C_LEFT0;
(339 行目)            *handle= 5L;
(340 行目)            break;

                      (中略)

(366 行目)          default:
(367 行目)            break;
(368 行目)        }
(369 行目)    }
```

直線・カーブのセンサ解析

該当センサパターン無時の戻り値
モード=-1,ハンドル角度=-180に設定

<<続く>>

<<続き>>

```
(378 行目)    void check_cross_half(UB sensor, W *mode)
(379 行目)    {
(380 行目)        *mode = -1L;
(381 行目)        /*************************************/
(382 行目)        /* クロスライン・ハーフラインチェック  */
(383 行目)        /*************************************/
(384 行目)        switch(sensor) {
(385 行目)          /* クランク検知 ------------------------------*/
(386 行目)          case 0x10u:  /* クランク予告 */
(387 行目)            *mode = MD_CRANKO;
(388 行目)            break;
(389 行目)          /* 斜変検知左 --------------------------------*/
(390 行目)          case 0x13u:  /* 斜変予告:左 */
(391 行目)          case 0x17u:  /* 斜変予告:左 */
(392 行目)        //case 0x1Fu:  /* 斜変予告:左 */
(393 行目)            if(gMode != MD_CRANKO) {
(394 行目)                *mode = MD_CC_LEFTO;
(395 行目)            }
(396 行目)            break;
(397 行目)          /* 斜変検知右 --------------------------------*/
(398 行目)          case 0xD0u:  /* 斜変予告:右 */
(399 行目)          case 0xF0u:  /* 斜変予告:右 */
(400 行目)        //case 0xF8u:  /* 斜変予告:右 */
(401 行目)            if(gMode != MD_CRANKO) {
(402 行目)                *mode = MD_CC_RIGHTO;
(403 行目)            }
(404 行目)            break;
(405 行目)          default:
(406 行目)            break;
(407 行目)        }
(408 行目)    }

(417 行目)    void mode_sens_ignore(W len)
(418 行目)    {
(419 行目)        W handle;
(420 行目)        W mode;
(421 行目)        volatile UB sensor;
(422 行目)        W i;
(423 行目)
(424 行目)        encoder_len_clr();              /* ｴﾝｺｰﾀﾞ距離ｸﾘｱ */
(425 行目)        while(get_encoder_len() < len) {
(426 行目)            for(i=0L;i<FIFO_SENS_SIZE;i++) {
(427 行目)                sensor = fifo_sensor_data[i];
(428 行目)
(429 行目)                check_curve(sensor, &mode, &handle);
(430 行目)                if(mode != -1L) {       /* 通常ﾗｲﾝ検知 */
(431 行目)                    gHandle_Next = handle;
(432 行目)                }
(433 行目)            }
(434 行目)            tslp_tsk(MTSK_INTERVAL);    /* Wait Main-Task Interval */
(435 行目)        }
(436 行目)    }
```

クロスハーフラインチェック

指定距離 進むまでWaitするルーチン

指定距離 進むまでループ

直線・カーブ解析を実施

<<続く>>

指定距離 進むまでWaitするルーチン

<<続き>>

```
(445 行目)    void mode_sens_ignore_ignore(W len)
(446 行目)    {
(447 行目)        encoder_len_clr();              /* ｴﾝｺｰﾀﾞ距離ｸﾘｱ */
(448 行目)        while(get_encoder_len() < len) {
(449 行目)            tslp_tsk(MTSK_INTERVAL);    /* Wait Main-Task Interval */
(450 行目)        }
(451 行目)    }

(460 行目)    void mode_wait_cc(void)
(461 行目)    {
(462 行目)        W handle;
(463 行目)        W mode;
(464 行目)        volatile UB sensor;
(465 行目)        W blkout;
(466 行目)        W i;
(467 行目)
(468 行目)        blkout = 0L;
(469 行目)        while(blkout < 4L) {
(470 行目)            for(i=0L;i<FIFO_SENS_SIZE;i++) {
(471 行目)                sensor = fifo_sensor_data[i];
(472 行目)
(473 行目)                if(sensor == 0xFFu) {
(474 行目)                    blkout++;
(475 行目)                } else {
(476 行目)                    blkout = 0L;
(477 行目)
(478 行目)                    check_curve(sensor, &mode, &handle);
(479 行目)                    if(mode != -1L) {       /* 通常ﾗｲﾝ検知 */
(480 行目)                        gHandle_Next = handle;
(481 行目)                    }
(482 行目)                }
(483 行目)            }
(484 行目)            tslp_tsk(MTSK_INTERVAL);    /* Wait Main-Task Interval */
(485 行目)        }
(486 行目)    }

(495 行目)    void mode_wait_cc_cr_end(void)
(496 行目)    {
(497 行目)        volatile UB sensor;
(498 行目)        W blkout;
(499 行目)        W i;
(500 行目)
(501 行目)        blkout = 0L;
(502 行目)        while(blkout < 2L) {
(503 行目)            for(i=0L;i<FIFO_SENS_SIZE;i++) {
(504 行目)                sensor = fifo_sensor_data[i];
(505 行目)
(506 行目)                switch(sensor) {
(507 行目)                  case 0xF7u:  /* 直線 */
(508 行目)                  case 0xDFu:  /* カーブ左１ */
(509 行目)                  case 0xD7u:  /* カーブ左０ */
```

直線・カーブ解析は実施しない

レーンチェンジ開始までWaitするルーチン

中央白線無が、レーンチェンジ開始条件

クランク・レーンチェンジ
モード終了までWaitするルーチン

モード抜け条件は5種類

●●●○●●●
●●○○●●●
●●○●●●●
●●●○○●●
●●●●○●●

<<続く>>

<<続き>>

```
(510 行目)                  case 0xFBu:  /* カーブ右１ */
(511 行目)                  case 0xF3u:  /* カーブ右０ */
(512 行目)                    blkout++;
(513 行目)                    break;
(514 行目)                  default:
(515 行目)                    blkout = 0L;
(516 行目)                    break;
(517 行目)                }
(518 行目)            }
(519 行目)            tslp_tsk(MTSK_INTERVAL);    /* Wait Main-Task Interval */
(520 行目)        }
(521 行目)    }

(530 行目)    W mode_wait_crankin(void)
(531 行目)    {
(532 行目)        W handle;
(533 行目)        W mode;
(534 行目)        W dir;
(535 行目)        volatile UB sensor;
(536 行目)        W blkout;
(537 行目)        W i;
(538 行目)
(539 行目)        dir = DIR_NON;
(540 行目)        blkout = 0L;
(541 行目)        while(blkout < 2L) {
(542 行目)            for(i=0L;i<FIFO_SENS_SIZE;i++) {
(543 行目)                sensor = fifo_sensor_data[i];
(544 行目)
(545 行目)                switch(sensor) {
(546 行目)                  case 0x13u:  /* クランク開始左 */
(547 行目)                  case 0x17u:  /* クランク開始左 */
(548 行目)                  case 0x1Fu:  /* クランク開始左 */
(549 行目)                    if(dir != DIR_RIGHT) {
(550 行目)                        blkout++;
(551 行目)                    } else {
(552 行目)                        blkout = 1L;
(553 行目)                    }
(554 行目)                    dir = DIR_LEFT;
(555 行目)                    break;
(556 行目)                  case 0xD0u:  /* クランク開始右 */
(557 行目)                  case 0xF0u:  /* クランク開始右 */
(558 行目)                  case 0xF8u:  /* クランク開始右 */
(559 行目)                    if(dir != DIR_LEFT) {
(560 行目)                        blkout++;
(561 行目)                    } else {
(562 行目)                        blkout = 1L;
(563 行目)                    }
(564 行目)                    dir = DIR_RIGHT;
(565 行目)                    break;
(566 行目)                  default:
(567 行目)                    blkout = 0L;
(568 行目)                    dir = DIR_NON;
(569 行目)                    check_curve(sensor, &mode, &handle);
(570 行目)                    if(mode != -1L) {       /* 通常ﾗｲﾝ検知 */
(571 行目)                        gHandle_Next = handle;
(572 行目)                    }
(573 行目)
```

クランク開始までWaitするルーチン
（クランク方向確認含む）

左クランク条件は3種類
○○○○○●●
○○○○●●●
○○○●●●●

右クランク条件は3種類
●●○○○○○
●●●○○○○
●●●●○○○

直線・カーブ解析を実施

<<続く>>

《続き》

```
(574 行目)                  break;
(575 行目)              }
(576 行目)          }
(577 行目)          tslp_tsk(MTSK_INTERVAL);    /* Wait Main-Task Interval */
(578 行目)      }
(579 行目)      return(dir);
(580 行目)  }

(589 行目)  void mode_crank(void)
(590 行目)  {
(591 行目)      W dir;
(592 行目)
(593 行目)      /* MD_CRANK1 */
(594 行目)      mode_sens_ignore(100L);         /* 確定～100mm 間ｾﾝｻ無視 */
(595 行目)      gMode++;
(596 行目)
(597 行目)      /* MD_CRANK2 */
(598 行目)      mode_sens_ignore(300L);         /* 100mm～400mm ｾﾝｻ無視 */
(599 行目)      gMode++;
(600 行目)
(601 行目)      /* MD_CRANK3 */
(602 行目)      dir = mode_wait_crankin();      /* 400mm～ｸﾗﾝｸ IN 待ち */
(603 行目)      gMode++;
(604 行目)
(605 行目)      /* MD_CRANK4-MD_CRANK5  */
(606 行目)      /* MD_CRANK4 is Break !! */
(607 行目)      RMTR_SET_BACK;                  /* 逆回転 */
(608 行目)      LMTR_SET_BACK;                  /* 逆回転 */
(609 行目)      if(dir == DIR_LEFT) {
(610 行目)          gHandle_Next = 36L;         /* 36° ﾊﾝﾄﾞﾙ切る */
(611 行目)      } else {
(612 行目)          gHandle_Next = -36L;        /* 36° ﾊﾝﾄﾞﾙ切る */
(613 行目)      }
(614 行目)      encoder_len_clr();              /* ｴﾝｺｰﾀﾞ距離ｸﾘｱ */
(615 行目)      while(get_encoder_len()<285L) {/* ｸﾗﾝｸ IN～285mm ｾﾝｻ無視 */
(616 行目)          tslp_tsk(MTSK_INTERVAL);    /* Wait Main-Task Interval */
(617 行目)      }
(618 行目)
(619 行目)      /* MD_CRANK6 */
(620 行目)      gMode = MD_CRANK6;
(621 行目)      if(dir == DIR_LEFT) {
(622 行目)          gHandle_Next = 32L;         /* 32° ﾊﾝﾄﾞﾙ切る */
(623 行目)      } else {
(624 行目)          gHandle_Next = -32L;        /* 32° ﾊﾝﾄﾞﾙ切る */
(625 行目)      }
(626 行目)      mode_wait_cc_cr_end();
(627 行目)      gMode++;
(628 行目)
(629 行目)      /* MD_CRANK7  */
(630 行目)      mode_sens_ignore(300L);         /* ｸﾗﾝｸ IN400～700mm */
(631 行目)      gMode = MD_STRAIGHT;
(632 行目)
(633 行目)      MDB_LED2_OFF;
(634 行目)      MDB_LED3_OFF;
(635 行目)  }
```

戻り値にクランク方向

クランクモードでの動作制御

《続く》

<<続き>>

左レーンチェンジモードでの動作制御

```
(644 行目)    void mode_cc_left(void)
(645 行目)    {
(646 行目)        /* MD_CC_LEFT1 */
(647 行目)        mode_sens_ignore(250L);          /* 確定～250mm 間ｾﾝｻ無視 */
(648 行目)        gMode++;
(649 行目)
(650 行目)        /* MD_CC_LEFT2 */
(651 行目)        mode_wait_cc();                  /* 250mm～ﾌﾞﾗｯｸｱｳﾄ待ち */
(652 行目)        gMode++;
(653 行目)
(654 行目)        /* MD_CC_LEFT3 */
(655 行目)        gHandle_Next = 10L;              /* 10° 左ﾊﾝﾄﾞﾙ切る */
(656 行目)        mode_sens_ignore_ignore(55L);    /* 55mm 間ｾﾝｻ無視 */
(657 行目)        gMode++;
(658 行目)
(659 行目)        /* MD_CC_LEFT4 */
(660 行目)        gHandle_Next = 22L;              /* 22° 左ﾊﾝﾄﾞﾙ切る */
(661 行目)        mode_sens_ignore_ignore(220L);   /* 220mm 間ｾﾝｻ無視 */
(662 行目)        gMode++;
(663 行目)
(664 行目)        /* MD_CC_LEFT5 */
(665 行目)        gHandle_Next = 0L;               /* ﾊﾝﾄﾞﾙ戻す */
(666 行目)        mode_wait_cc_cr_end();           /* ﾓｰﾄﾞ抜け判断(直線認識) */
(667 行目)        gMode = MD_STRAIGHT;
(669 行目)        MDB_LED2_OFF;
(670 行目)        MDB_LED3_OFF;
(671 行目)    }
```

右レーンチェンジモードでの動作制御

```
(680 行目)    void mode_cc_right(void)
(681 行目)    {
(682 行目)        /* MD_CC_RIGHT1 */
(683 行目)        mode_sens_ignore(250L);          /* 確定～150mm 間ｾﾝｻ無視 */
(684 行目)        gMode++;
(685 行目)
(686 行目)        /* MD_CC_RIGHT2 */
(687 行目)        mode_wait_cc();                  /* 250mm～ﾌﾞﾗｯｸｱｳﾄ待ち */
(688 行目)        gMode++;
(689 行目)
(690 行目)        /* MD_CC_RIGHT3 */
(691 行目)        gHandle_Next = -10L;             /* 10° 右ﾊﾝﾄﾞﾙ切る */
(692 行目)        mode_sens_ignore_ignore(55L);    /* 55mm 間ｾﾝｻ無視 */
(693 行目)        gMode++;
(694 行目)
(695 行目)        /* MD_CC_RIGHT4 */
(696 行目)        gHandle_Next = -22L;             /* 22° 右ﾊﾝﾄﾞﾙ切る */
(697 行目)        mode_sens_ignore_ignore(220L);   /* 220mm 間ｾﾝｻ無視 */
(698 行目)        gMode++;
(699 行目)
(700 行目)        /* MD_CC_RIGHT5 */
(701 行目)        gHandle_Next = 0L;               /* ﾊﾝﾄﾞﾙ戻す */
(702 行目)        mode_wait_cc_cr_end();           /* ﾓｰﾄﾞ抜け判断(直線認識) */
(703 行目)        gMode = MD_STRAIGHT;
(705 行目)        MDB_LED2_OFF;
(706 行目)        MDB_LED3_OFF;
(707 行目)    }
```

<<続く>>

《続き》

```
(716 行目)  #pragma noregsave(tsk_hayakawa)
(717 行目)  void tsk_hayakawa(void)
(718 行目)  {
(719 行目)      W handle;
(720 行目)      W mode;
(721 行目)      volatile UB sensor;
(722 行目)      W i;
(723 行目)      W mcnt;
(724 行目)
(725 行目)      gMode = MD_STRAIGHT;
(726 行目)      while(1) {
(727 行目)          for(i=0L;i<FIFO_SENS_SIZE;i++) {
(728 行目)              sensor = fifo_sensor_data[i];
(729 行目)              /*************************************/
(730 行目)              /* 通常ﾓｰﾄﾞ                          */
(731 行目)              /*************************************/
(732 行目)              if(gMode <= MD_NORMAL) {
(733 行目)                  MDB_LED2_OFF;
(734 行目)                  MDB_LED3_OFF;
(735 行目)                  check_curve(sensor, &mode, &handle);
(736 行目)                  if(mode != -1L) {          /* 通常ﾗｲﾝ検知 */
(737 行目)                      gMode       = mode;
(738 行目)                      gHandle_Next = handle;
(739 行目)                  } else {                   /* 通常ﾗｲﾝ以外(ｸﾗﾝｸ予告,ｺｰｽ変更予告,その他) */
(740 行目)                      check_cross_half(sensor,&mode);
(741 行目)                      if(mode != -1L) {      /* ｸﾗﾝｸ予告,ｺｰｽ変更予告 検知 */
(742 行目)                          gMode = mode;
(743 行目)                          mcnt  = 0L;
(744 行目)                      }
(745 行目)                  }
(746 行目)              } else if(gMode == MD_CRANK1) {
(747 行目)                  /*************************************/
(748 行目)                  /* ｸﾗﾝｸ確定                          */
(749 行目)                  /*************************************/
(750 行目)                  MDB_LED2_ON;
(751 行目)                  MDB_LED3_ON;
(752 行目)                  mode_crank();
(753 行目)              } else if(gMode == MD_CC_LEFT1) {
(754 行目)                  /*************************************/
(755 行目)                  /* ｺｰｽ変更左(確定)                   */
(756 行目)                  /*************************************/
(757 行目)                  MDB_LED2_OFF;
(758 行目)                  MDB_LED3_ON;
(759 行目)                  mode_cc_left();
(760 行目)              } else if(gMode == MD_CC_RIGHT1) {
(761 行目)                  /*************************************/
(762 行目)                  /* ｺｰｽ変更右(確定)                   */
(763 行目)                  /*************************************/
(764 行目)                  MDB_LED2_ON;
(765 行目)                  MDB_LED3_OFF;
(766 行目)                  mode_cc_right();
(767 行目)              } else {
```

コース状況解析タスク

FIFOバッファからセンサ情報取出し

直線・カーブの解析

直線・カーブ以外のとき、
クランクレーンチェンジのチェック

クランクモード処理へ

左レーンチェンジモード処理へ

右レーンチェンジモード処理へ

《続く》

<<続き>>

```
(767 行目)                } else {
(768 行目)                    /****************************************/
(769 行目)                    /* ｸﾗﾝｸ予告,ｺｰｽ変更予告 中                */
(770 行目)                    /****************************************/
(771 行目)                    check_cross_half(sensor,&mode);
(772 行目)                    if(mode != -1L) {          /* ｸﾗﾝｸ予告,ｺｰｽ変更予告 検知 */
(773 行目)                        gMode = mode;
(774 行目)                        mcnt  = 0L;
(775 行目)                    } else {                   /* ｸﾗﾝｸ予告,ｺｰｽ変更以外 */
(776 行目)                        check_curve(sensor, &mode, &handle);
(777 行目)                        if(mode != -1L) {      /* ｸﾗﾝｸ予告,ｺｰｽ変更 ⇒ 通常ﾗｲﾝ :  */
(778 行目)                            mcnt++;
(779 行目)                            if(mcnt > 2L) {
(780 行目)                                gMode++;       /* ｸﾗﾝｸ,ｺｰｽ変更 決定 */
(781 行目)                            }
(782 行目)                        }
(783 行目)                    }
(784 行目)                }
(785 行目)            }
(786 行目)            tslp_tsk(MTSK_INTERVAL);    /* Wait Main-Task Interval */
(787 行目)        }
(788 行目)    }
```

クランク・レーンチェンジチェック

クランク・レーンチェンジモード確定

8. 全国大会リザルト

JMCR2009(ジャパンマイコンカーラリー2009) 一般の部 予選での結果を以下に示します。

表8-1『はしれ早川』号 JMCR2009 一般の部 予選結果

順位	タイム	備考
37 **位**	00’31’’57	**予選敗退** **参加台数**:104 **台** **完走台数**:40 **台** **予選上位者タイム：**00’16’’48

9. 総括･課題

マイコンカーラリー公式標準キットをベースに開発した『はしれ早川』号は、Smalight OS を搭載してソフトウェア重視の開発を行いました。本大会予選にて無事完走できたことで、RTOS を搭載したプログラムでもマイコンカーの制御が可能であることを証明できたと思われます。

内容的には OS が提供するタスク間通信機能を利用すると OS オーバーヘッドが性能上のネックになる可能性も高く、十分に OS 機能を活用しているとはいい難いものでした。但し、Smalight OS を導入し機能をタスク分割することで、ハードウェア(CPU)の相違を隠蔽し、各機能(タスク)をシンプルにソースコードの記載ができました。移植性、開発効率(デバッグ効率)、保守性などの RTOS 導入メリットは享受できると考えられます。

今回、マイコンカーラリー公式標準キットを利用しましたが、現在多くの参加者で採用される自作サーボ、センサの A/D 入力化など、高速化したマイコンカーでも Smalight OS が導入可能であるかの検証は、今後の課題となります。

10. 最後に

本開発にあたり、山形県立米沢工業高等学校の方々には多大なるご協力を頂きました。心より感謝の意を表します。

11. 付録

11.1 Smalight OS データ型について

本ドキュメント内のソースリストでは、変数の型宣言を以下の定義に従い記載しております。本表記方法は ITRON 仕様に準じた型宣言となります。

表11-1 データ型一覧

型	意味
B	符号付き8ビット整数
H	符号付き16ビット整数
W	符号付き32ビット整数
UB	符号無し8ビット整数
UH	符号無し16ビット整数
UW	符号無し32ビット整数
FP	void型関数へのポインタ
CFP	void型関数へのポインタ(ベクタ用)

ホームページとサポート窓口

(株)ルネサス北日本セミコンダクタ ホームページ
http://www.kitasemi.renesas.com/

Smalight ホームページ
http://www.kitasemi.renesas.com/product/smalight/

お問合せ先
soft.support@kitasemi.renesas.com

改定記録

Rev	発行日	改定内容	
		ページ	ポイント
1.00	2009.4.16	―	**初版発行**
1.00.1	2009.4.18	58	**誤記修正**(表 8-1 ﾀｲﾑ)

安全設計に関するお願い

1. 弊社は品質、信頼性の向上に努めておりますが、半導体製品は故障が発生したり、誤動作する場合があります。弊社の半導体製品の故障又は誤動作によって結果として、人身事故、火災事故、社会的損害などを生じさせないような安全性を考慮した冗長設計、延焼対策設計、誤動作防止設計などの安全設計に十分ご留意ください。

本資料ご利用に際しての留意事項

1. 本資料は、お客様が用途に応じた適切な弊社製品をご購入いただくための参考資料であり、本資料中に記載の技術情報について弊社、および、第三者が所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
2. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、弊社は責任を負いません。
3. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、弊社は、予告なしに、本資料に記載した製品または仕様を変更することがあります。
4. 本資料に記載した情報は、正確を期すため、慎重に制作したものですが万一本資料の記述誤りに起因する損害がお客様に生じた場合には、弊社はその責任を負いません。
5. 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。弊社は、適用可否に対する責任は負いません。
6. 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、弊社へご照会ください。
7. 本資料の転載、複製については、文書による弊社の事前の承諾が必要です。
8. 本資料に関し詳細についてのお問い合わせ、その他お気付きの点がございましたら弊社までご照会ください。