HSBRX63T-100RC

RC カー制御ソフトウェア編マニュアル

ルネサス エレクトロニクス社　RX63T(QFP-100 ピン)搭載
HSB シリーズマイコンボード

-本書を必ずよく読み、ご理解された上でご利用ください

株式会社北斗電子

REV.1.0.0.0

ー目　次ー

注意事項

本書を必ずよく読み、ご理解された上でご利用ください

【ご利用にあたって】

1. 本製品をご利用になる前には必ず取扱説明書をよく読んで下さい。また、本書は必ず保管し、使用上不明な点がある場合は再読し、よく理解して使用して下さい。
2. 本書は株式会社北斗電子製マイコンボードの使用方法について説明するものであり、ユーザシステムは対象ではありません。
3. 本書及び製品は著作権及び工業所有権によって保護されており、全ての権利は弊社に帰属します。本書の無断複写・複製・転載はできません。
4. 弊社のマイコンボードの仕様は全て使用しているマイコンの仕様に準じております。マイコンの仕様に関しましては製造元にお問い合わせ下さい。弊社製品のデザイン・機能・仕様は性能や安全性の向上を目的に、予告無しに変更することがあります。また価格を変更する場合や本書の図は実物と異なる場合もありますので、御了承下さい。
5. 本製品のご使用にあたっては、十分に評価の上ご使用下さい。
6. 未実装の部品に関してはサポート対象外です。お客様の責任においてご使用下さい。

【限定保証】

1. 弊社は本製品が頒布されているご利用条件に従って製造されたもので、本書に記載された動作を保証致します。
2. 本製品の保証期間は購入戴いた日から1年間です。

【保証規定】

保証期間内でも次のような場合は保証対象外となり有料修理となります

1. 火災・地震・第三者による行為その他の事故により本製品に不具合が生じた場合
2. お客様の故意・過失・誤用・異常な条件でのご利用で本製品に不具合が生じた場合
3. 本製品及び付属品のご利用方法に起因した損害が発生した場合
4. お客様によって本製品及び付属品へ改造・修理がなされた場合

【免責事項】

弊社は特定の目的･用途に関する保証や特許権侵害に対する保証等、本保証条件以外のものは明示・黙示に拘わらず一切の保証は致し兼ねます。また、直接的・間接的損害金もしくは欠陥製品や製品の使用方法に起因する損失金・費用には一切責任を負いません。損害の発生についてあらかじめ知らされていた場合でも保証は致し兼ねます。
ただし、明示的に保証責任または担保責任を負う場合でも、その理由のいかんを問わず、累積的な損害賠償責任は、弊社が受領した対価を上限とします。本製品は「現状」で販売されているものであり、使用に際してはお客様がその結果に一切の責任を負うものとします。弊社は使用または使用不能から生ずる損害に関して一切責任を負いません。
保証は最初の購入者であるお客様ご本人にのみ適用され、お客様が転売された第三者には適用されません。よって転売による第三者またはその為になすお客様からのいかなる請求についても責任を負いません。
本製品を使った二次製品の保証は致し兼ねます。

安全上のご注意

製品を安全にお使いいただくための項目を次のように記載しています。絵表示の意味をよく理解した上でお読み下さい。

表記の意味

警告 取扱を誤った場合、人が死亡または重傷を負う危険が切迫して生じる可能性がある事が想定される

注意 取扱を誤った場合、人が軽傷を負う可能性又は、物的損害のみを引き起こすが可能性がある事が想定される

絵記号の意味

	**一般指示** 使用者に対して指示に基づく行為を強制するものを示します		**一般禁止** 一般的な禁止事項を示します
	**電源プラグを抜く** 使用者に対して電源プラグをコンセントから抜くように指示します		**一般注意** 一般的な注意を示しています

警告

以下の警告に反する操作をされた場合、本製品及びユーザシステムの破壊・発煙・発火の危険があります。マイコン内蔵プログラムを破壊する場合もあります。

1. 本製品及びユーザシステムに電源が入ったままケーブルの抜き差しを行わないでください。
2. 本製品及びユーザシステムに電源が入ったままで、ユーザシステム上に実装されたマイコンまたはIC等の抜き差しを行わないでください。
3. 本製品及びユーザシステムは規定の電圧範囲でご利用ください。
4. 本製品及びユーザシステムは、コネクタのピン番号及びユーザシステム上のマイコンとの接続を確認の上正しく扱ってください。

発煙・異音・異臭にお気付きの際はすぐに使用を中止してください。

電源がある場合は電源を切って、コンセントから電源プラグを抜いてください。そのままご使用すると火災や感電の原因になります。

⚠注意

以下のことをされると故障の原因となる場合があります。

1. 静電気が流れ、部品が破壊される恐れがありますので、ボード製品のコネクタ部分や部品面には直接手を触れないでください。
2. 次の様な場所での使用、保管をしないでください。

   ホコリが多い場所、長時間直射日光があたる場所、不安定な場所、衝撃や振動が加わる場所、落下の可能性がある場所、水分や湿気の多い場所、磁気を発するものの近く
3. 落としたり、衝撃を与えたり、重いものを乗せないでください。
4. 製品の上に水などの液体や、クリップなどの金属を置かないでください。
5. 製品の傍で飲食や喫煙をしないでください。

ボード製品では、裏面にハンダ付けの跡があり、尖っている場合があります。

取り付け、取り外しの際は製品の両端を持ってください。裏面のハンダ付け跡で、誤って手など怪我をする場合があります。

CD メディア、フロッピーディスク付属の製品では、故障に備えてバックアップ(複製)をお取りください。

製品をご使用中にデータなどが消失した場合、データなどの保証は一切致しかねます。

アクセスランプがある製品では、アクセスランプが点灯中に電源の切断を行わないでください。

製品の故障や、データの消失の恐れがあります。

本製品は、医療、航空宇宙、原子力、輸送などの人命に関わる機器やシステム及び高度な信頼性を必要とする設備や機器などに用いられる事を目的として、設計及び製造されておりません。

医療、航空宇宙、原子力、輸送などの設備や機器、システムなどに本製品を使用され、本製品の故障により、人身や火災事故、社会的な損害などが生じても、弊社では責任を負いかねます。お客様ご自身にて対策を期されるようご注意ください。

概要

HSBRX63T-100RC ボードで、RC カーキットを制御するプログラムに関する説明を記載したマニュアルとなります。

本マニュアルで解説するプログラムは、主に 3 種の制御

・コントローラからの入力の処理[COMM]

・車体制御（エレクトリックスピードコントローラ(ESC), ステアリングサーボ）[CHASSIS]

・ボディ制御（各種 LED の制御、ブザー制御）[BODY]

を行っています。

RC カーキットには、HSBRX63T-100RC ボードを 1~3 台搭載可能となっており、1 台のボードで COMM, CHASSIS, BODY 3 種全ての制御を行う事も、3 台のボードで制御を分散して処理する事も可能です。

複数台のボードで RC カーを制御する場合、ボード間の通信は CAN インタフェースで行います。

1. 仕様

1.1. ボードの役割

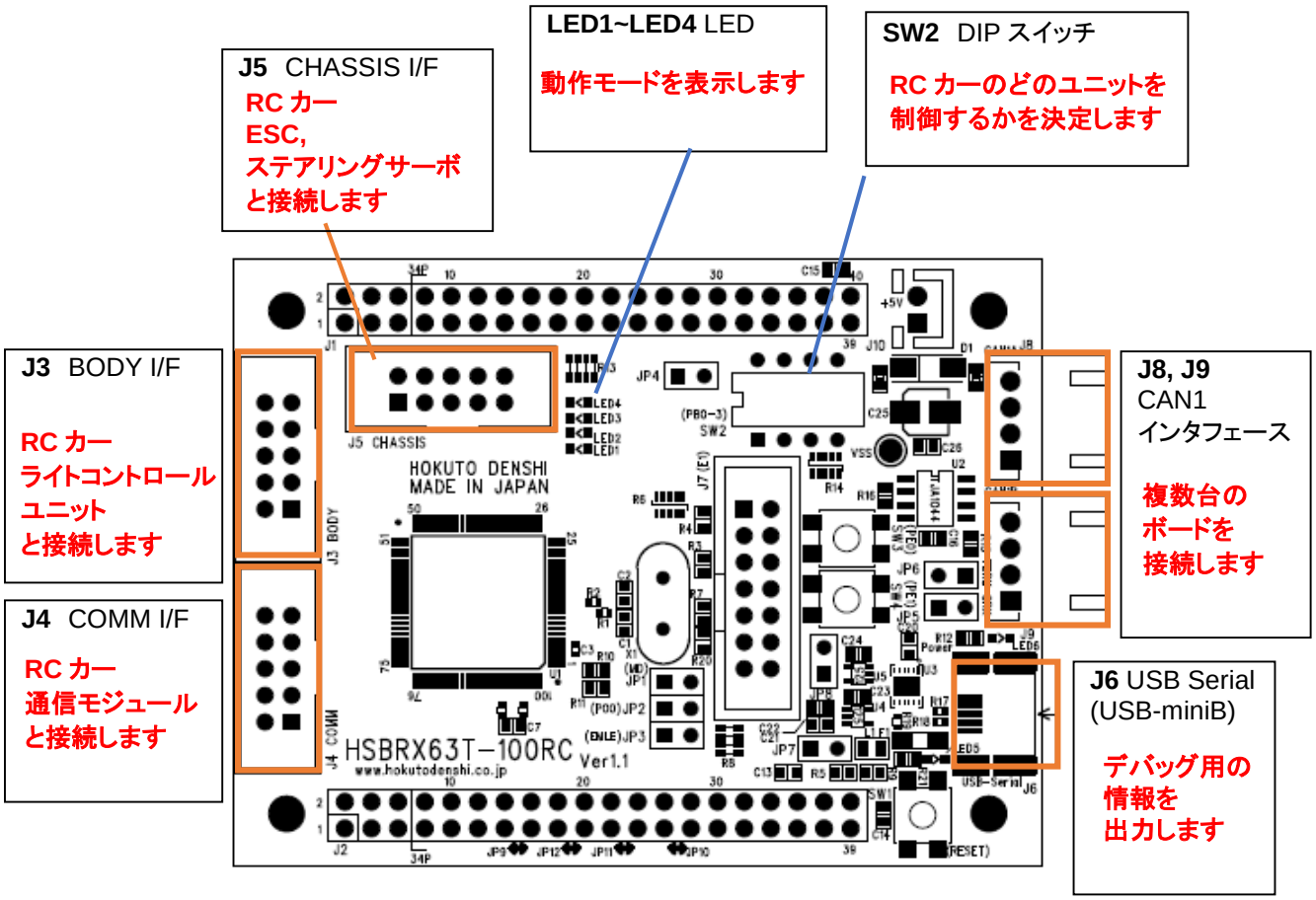


図 1-1 ボード配置図・役割

図 1-1 にボード配置図・役割を示します。RC カーとの接続では、J3~J5 を使用します。

SW3, SW4（プッシュスイッチ）は汎用入力として使用可能ですが、現状のプログラムでは使用していません。ユーザ側でプログラムを拡張し、任意の用途で使用する事が可能です。

本ボードは、J6(USB-miniB)からボードに給電する事が可能ですが、RC カーのボディ制御には USB(~2.0)で供給可能な 500mA を超える電流容量が必要になりますので、RC カー接続時の給電は RC カー側から行ってください。（RC カー接続時は、J5(CHASSIS)からの給電となります）

1.2. RC カーインタフェース

1.2.1. BODY インタフェース(J3)

J3 は、RC カーのボディ制御を行う端子で、ライトコントロールユニット（タミヤ製 TLU-01)と接続するコネクタです。

表 1-1 J3 [BODY] RC カー信号対応表

No	マイコン信号名	接続先	用途
1	VCC	TLU-01 - 1P	TLU-01 に対しての電源（＋）
2	VSS	TLU-01 - 2P	TLU-01 に対しての電源（－）
3	P30	TLU-01 - 3P	ブザー
4	P31	TLU-01 - 4P	(NC)
5	P32	TLU-01 - 5P	ブレーキランプ
6	P33	TLU-01 - 6P	バックランプ
7	P70	TLU-01 - 7P	左ウィンカ
8	P71	TLU-01 - 8P	フォグランプ
9	P72	TLU-01 - 9P	ヘッドランプ
10	P73	TLU-01 - 10P	右ウィンカ

*(NC)は未接続です。

※表 1-1 の用途は接続の代表例です。車種や、ボディの加工有無により、LED の数は異なる事もあります。本マニュアルで解説しているプログラムは、表 1-1 の対応に従い制御を行うプログラムとなっています。

P30~33, P70~73 の各端子は、汎用 I/O 出力に設定し、RC カーのライト(LED)、ブザーを制御します。

端子を L 制御すると、LED は消灯となります。H（または Hi-Z）で、LED は点灯します。

ブザーは、L/H を断続的に切り替えることにより鳴動します。RC カーキット付属のブザーは、4kHz 程度で駆動した場合、明瞭に鳴動します。

1.2.2. COMM インタフェース(J4)

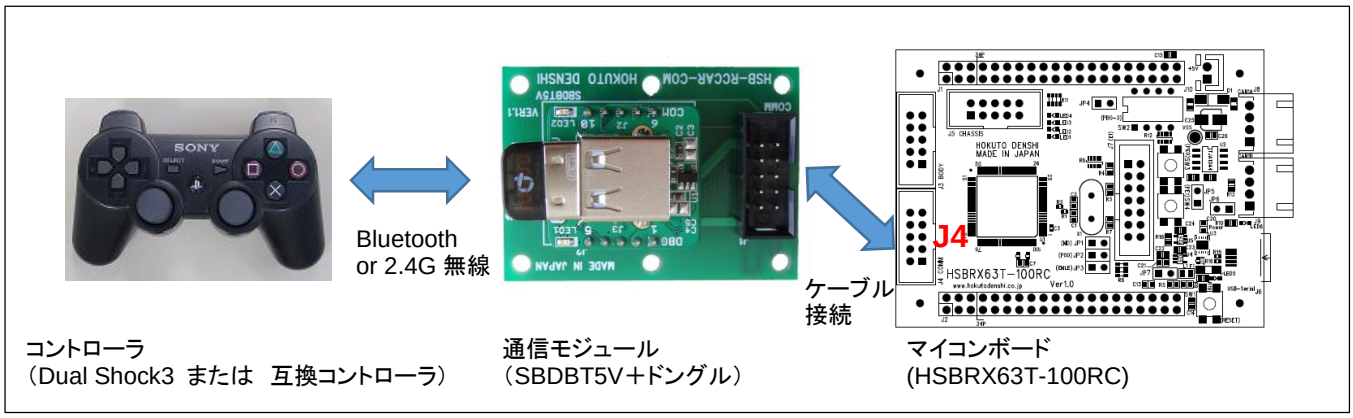
J4 は、RC カーの通信モジュール SBDBT5V と接続するコネクタです。コントローラの信号受け取る入力となります。

表 1-2 J4 [COMM] RC カー信号対応表

No	マイコン信号名	接続先	用途
1	VCC	SBDBT5V - 2P	SBDBT5V に対しての電源（＋）
2	(NC)		(NC)
3	(NC)		(NC)
4	(NC)		(NC)
5	(NC)		(NC)
6	(NC)		(NC)
7	(NC)		(NC)
8	P23/TXD0	SBDBT5V - 8P	
9	P22/RXD0	SBDBT5V - 7P	SBDBT5V からの制御信号入力
10	VSS	SBDBT5V - 3P	SBDBT5V に対しての電源（－）

マイコン側の制御としては、P22 を RXD0 に設定し、通信速度 115,200bps に設定し、データを受信します。P23/TXD0 は本プログラムでは未使用です（ハードウェアとしては配線がつながっていますので利用は可能です）。

通信モジュール(SBDBT5V)は、Bluetooth または 2.4GHz のドングルを経由してコントローラ（PS3 Dualshock 等）とリンクし、コントローラの入力をマイコンまで伝えます。



ーコントローラから送信される（マイコンボードで受信する）信号(hex)ー

80 X1 X2 Y1 Y2 Y3 Y4 SS

コントローラの入力が SBDBT5V で変換され、8 バイトの信号が、約 45ms に 1 回送られてきます。

X1 X2 ：ボタン押下の情報（デジタル信号） 2Bytes

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
			R2	R1	L2	L1	□		○	×	△	←	→	↓	↑

コントローラの各キーを押している間、上記に対応する箇所のビットが立つ（1 となる）

START は、↑と↓のビットが立つ(0x00 0x03)
SELECT は、→と左のビットが立つ(0x00 0x0C)

Y1 Y2 Y3 Y4 ：アナログスティックの情報 4Bytes

Y1 左スティック左右 RC カーではステアリングに使用
Y2 左スティック上下
Y3 右スティック左右
Y4 右スティック上下 RC カーではアクセル（バック）に使用

アナログスティックは、センターが 0x40。右・下に倒した場合、最大 0x7f。左・上に倒した場合、最小 0x00 になります。

左・上		センター		右・下
0x00	←	0x40	→	0x7f

SS ：チェックサム 1Bytes

先頭の 0x80 を除いた 6 バイトの単純加算

※現行のプログラムでは、チェックサム異常は判定していません

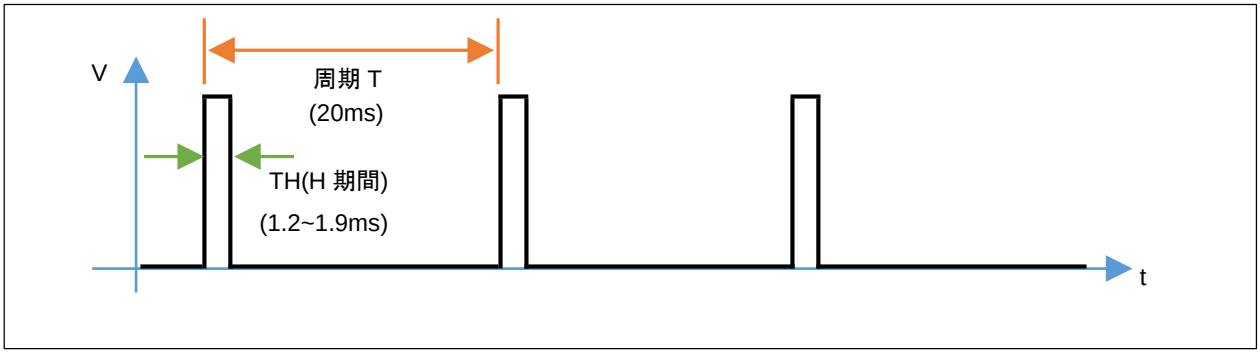
1.2.3. CHASSIS インタフェース(J5)

J5 は、RC カーの ESC（エレクトリックスピードコントローラ）、ステアリングサーボと接続するコネクタです。また、RC カーキットと接続する場合は、このコネクタから、マイコンボードを動作させる電源の供給を受けます。

表 1-3 J5 [CHASSIS] RC カー信号対応表

No	マイコン信号名	接続先	用途
1	VCC	HSB-RCCAR-SERVO - +5V	電源入力（＋）
2	(NC)		(NC)
3	(NC)		(NC)
4	(NC)		(NC)
5	(NC)		(NC)
6	(NC)		(NC)
7	(NC)		(NC)
8	P75(MTIOC4C)	HSB-RCCAR-SERVO - S-White	ステアリングサーボ制御
9	P91(MTIOC7C)	HSB-RCCAR-SERVO - E-White	ESC 制御
10	VSS	HSB-RCCAR-SERVO - GND	電源入力（－）

マイコンから側では、P75,P91 をタイマ出力(MTIOC)に設定し、PWM 信号を出力して ESC とステアリングサーボを制御します。



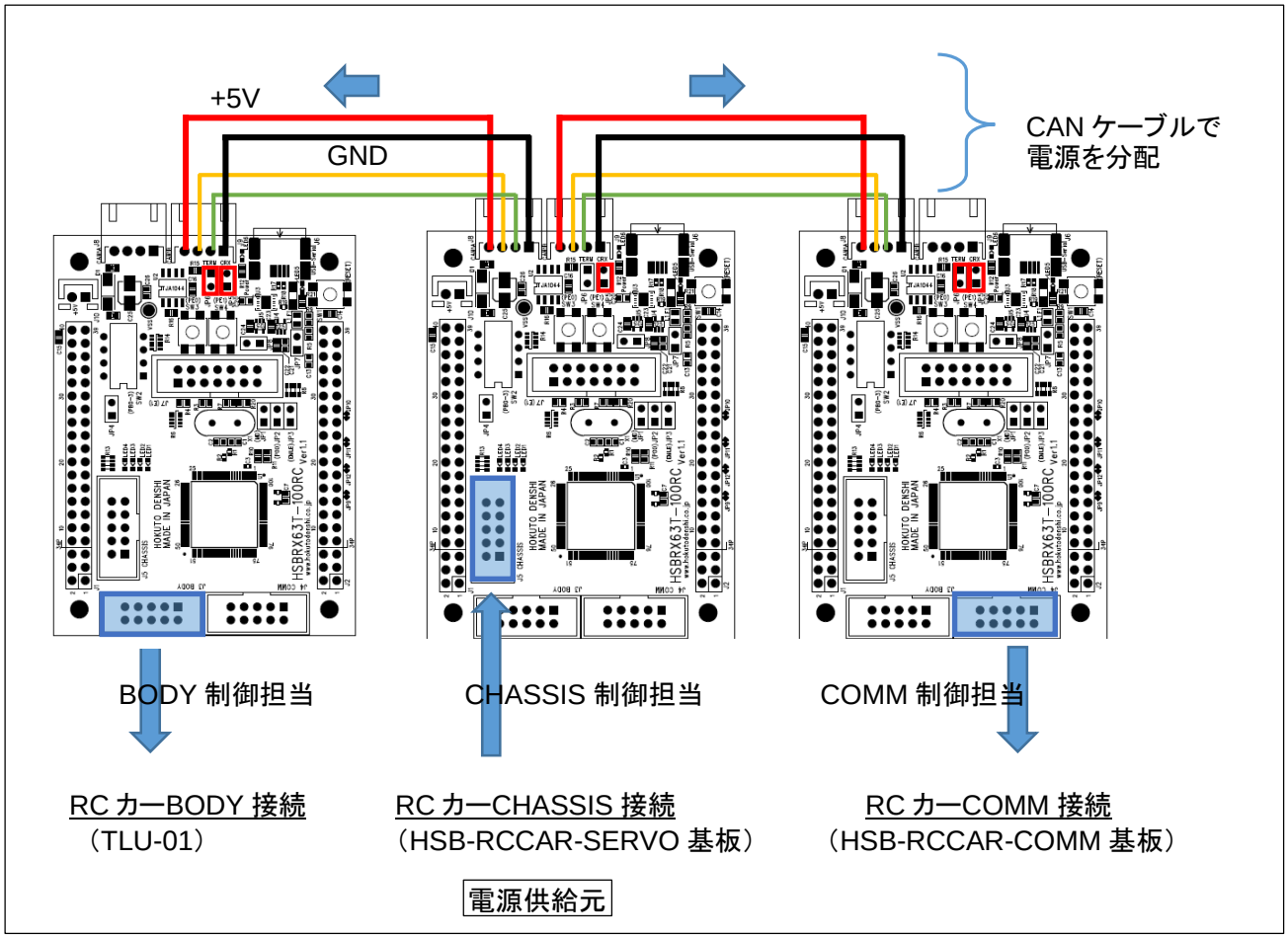
マイコン側から出力する信号としては、矩形波で、周期は固定、TH 期間を変更したものとなります。

TH	前進	静止	後進
ESC	1.4ms	1.55ms	1.7ms

TH	ハンドルを右	ハンドルセンター	ハンドルを左
ステアリングサーボ	1.9ms	1.55ms	1.2ms

上記は、タイミングパラメータの例となります。ESC やステアリングサーボの個体差で微調整が必要な場合があります。

・3 台のボードを接続する場合の給電方式



RC カーキット制御に使用する場合、J5(CHASSIS)コネクタに接続した、HSB-RCCAR-SERVO 基板が電源供給元となります。

複数台のボード同士は、CAN ケーブルで接続しますが、CAN ケーブルは電源を分配する役割を果たします。

※電源供給元が複数存在する事のない様にご注意ください

※上記の系で、
・J10（電源コネクタ）に電源を接続する
・J6(USB-miniB)に PC を接続し、JP7 をショートに設定する
事は禁止です。

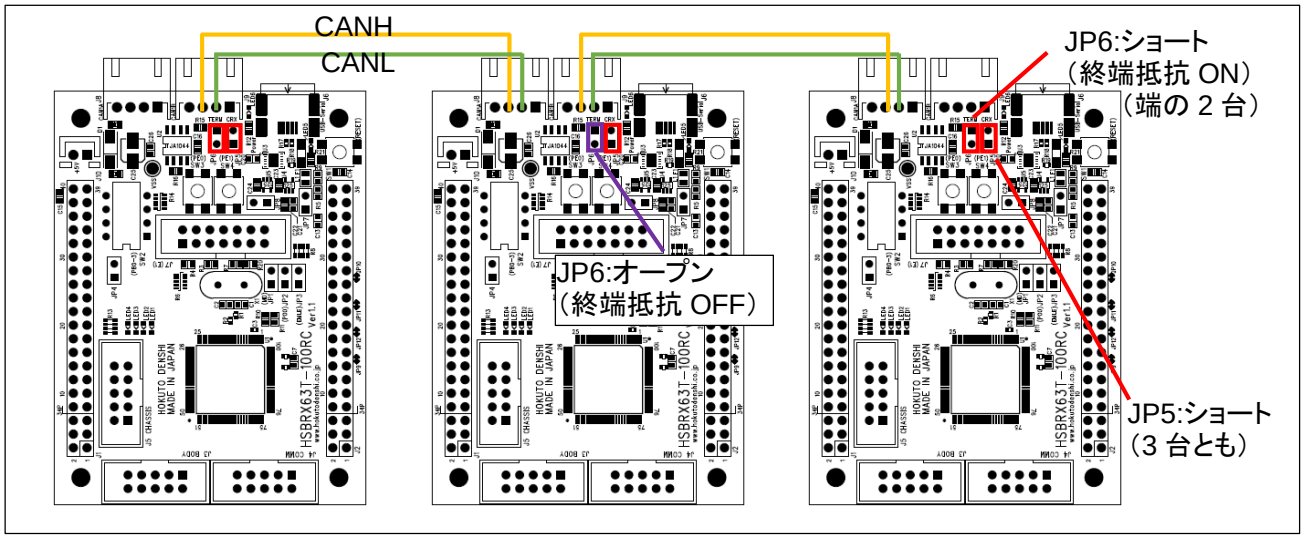
1.2.4. USB-Serial インタフェース(J6)

本コネクタと PC を接続し、マイコンからデバッグ情報を端末（仮想 COM ポート）に表示させることができます。
※初回接続時、PC にドライバソフトがインストールされていない場合は、FTDI 社の仮想 COM ポートドライバをインストールしてください
速度は、本プログラムでは、115,200bps に設定しています。

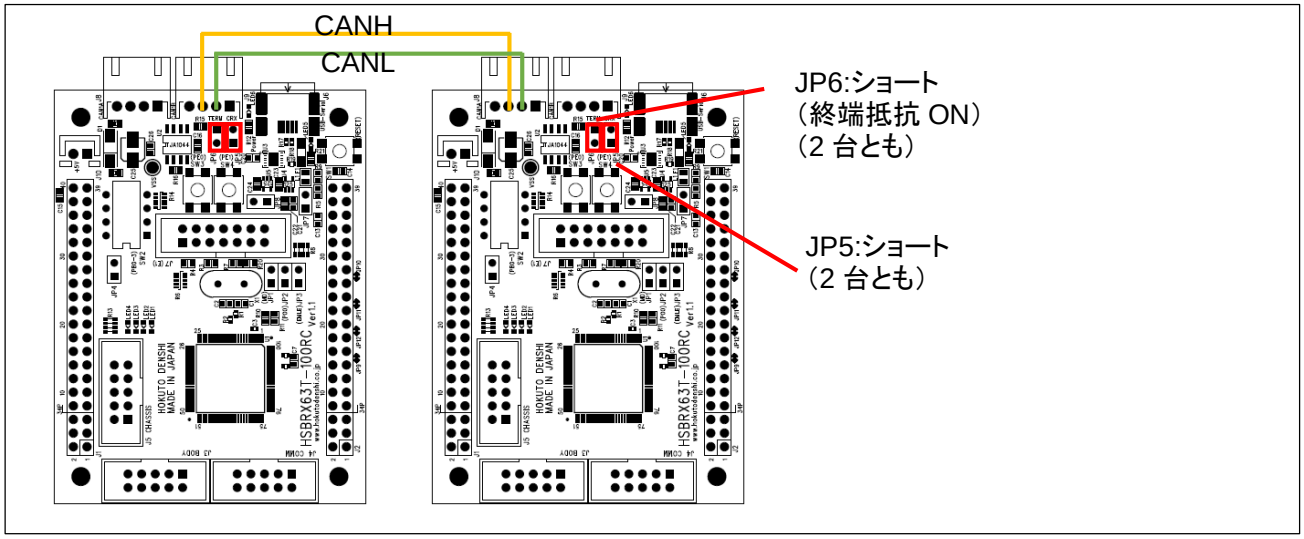
1.2.5. CAN インタフェース(J8,J9)

2 台以上のボードを RC カーキットに搭載する場合は、CAN インタフェースでボード同士を接続する必要があります。

・3 台のボードを接続する場合



・2 台のボードを接続する場合



CAN ケーブル(10cm)は、RC カーキットに付属しています。CAN バスの端に来るボードの終端抵抗を ON（JP6 ショート）としてください。

※CAN バスに他の（本ボード以外）CAN 機器を接続しても問題ありません。その場合は、終端抵抗の設定に注意願います。

※J8, J9 はコネクタの同一ピン同士が接続されていますので、どちらのコネクタを用いても同等です。

本プログラムでは、CAN は COMM に通信モジュールを挿しているボードから、他のボード（COMM に通信モジュールを接続していないボード）に、コントローラからの情報を受け渡すために使用しています。

1.2.6. DIP スイッチ(SW2)

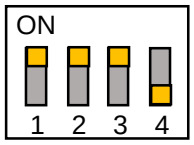
DIP スイッチは、ボードの動作を決めるための入力として使用しています。

動作モード設定

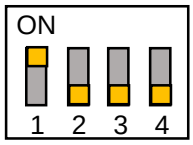
スイッチ	SW2-1	SW2-2	SW2-3	SW2-4
ON	BODY 制御を担当	CHASSIS 制御を担当	COMM 制御を担当	デバッグ情報を 出力

・モード設定例

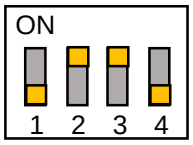
SW2 モード設定例



1 台のボードで
全処理を行う



BODY 制御
のみ行う



CHASSIS と COMM
制御を行う

SW2-1 を ON にしたボードは、J3(BODY)を RC カーと接続しください。
SW2-2 を ON にしたボードは、J5(CHASSIS)を RC カーと接続しください。
SW2-3 を ON にしたボードは、J4(COMM)を RC カーと接続しください。

モード設定の、SW2-1~SW2-3 は、起動時に読み込みます。（起動後に変更しても、変更は反映されません。）
デバッグ情報の表示を制御する SW2-4 は、起動後でも随時設定が切り替わります。

1.2.7. LED(LED1~4)

LED	LED1	LED2	LED3	LED4
初期状態	BODY 制御を担当時 点灯	CHASSIS 制御を担当時 点灯	COMM 制御を担当時 点灯	デバッグ情報 出力時 点灯
動作時	制御アクティブ時 点滅	制御アクティブ時 点滅	コントローラ情報 読み取り時 点滅	文字送出時点滅

LED は、ボードがどの制御を担当しているかを示しています。

LED1, LED2 は、RC カーと情報をやり取りしているタイミングで点滅（消灯）します。

LED3 は、COMM 制御を担当していないボードに関しても、CAN でコントローラ情報を受信した際に点滅します。

RC カー操作

(1)前進、後進、ステアリング、ブレーキ

ステアリング　　前進
後進

アナログ　**左スティック**でステアリング操作

右スティックで、前進、後進を制御する

×でブレーキ

(2)ライト制御

L1　左ウィンカ点滅／消灯
R1　右ウィンカ点滅／消灯
○ヘッドランプ点灯／消灯
□フォグランプ点灯／消灯
△ハザード ON／OFF

(3)ステアリング調整

← ステアリングのニュトラル位置を左に微調整
（キーを押している間左ウィンカ点灯）
→ ステアリングのニュトラル位置を右に微調整
（キーを押している間右ウィンカ点灯）
↑ ステアリングのニュトラル位置をデフォルトに戻す

※調整限界に達した場合は、ウィンカ点灯せず

(4)ゲイン調整

L2　ESC のゲイン（調整幅）を狭める
（キーを押している間左ウィンカ点灯）
R2　ESC のゲイン（調整幅）を広げる
（キーを押している間右ウィンカ点灯）
↓ ゲインをデフォルトに戻す

※調整限界に達した場合は、ウィンカ点灯せず

2. プログラムの説明

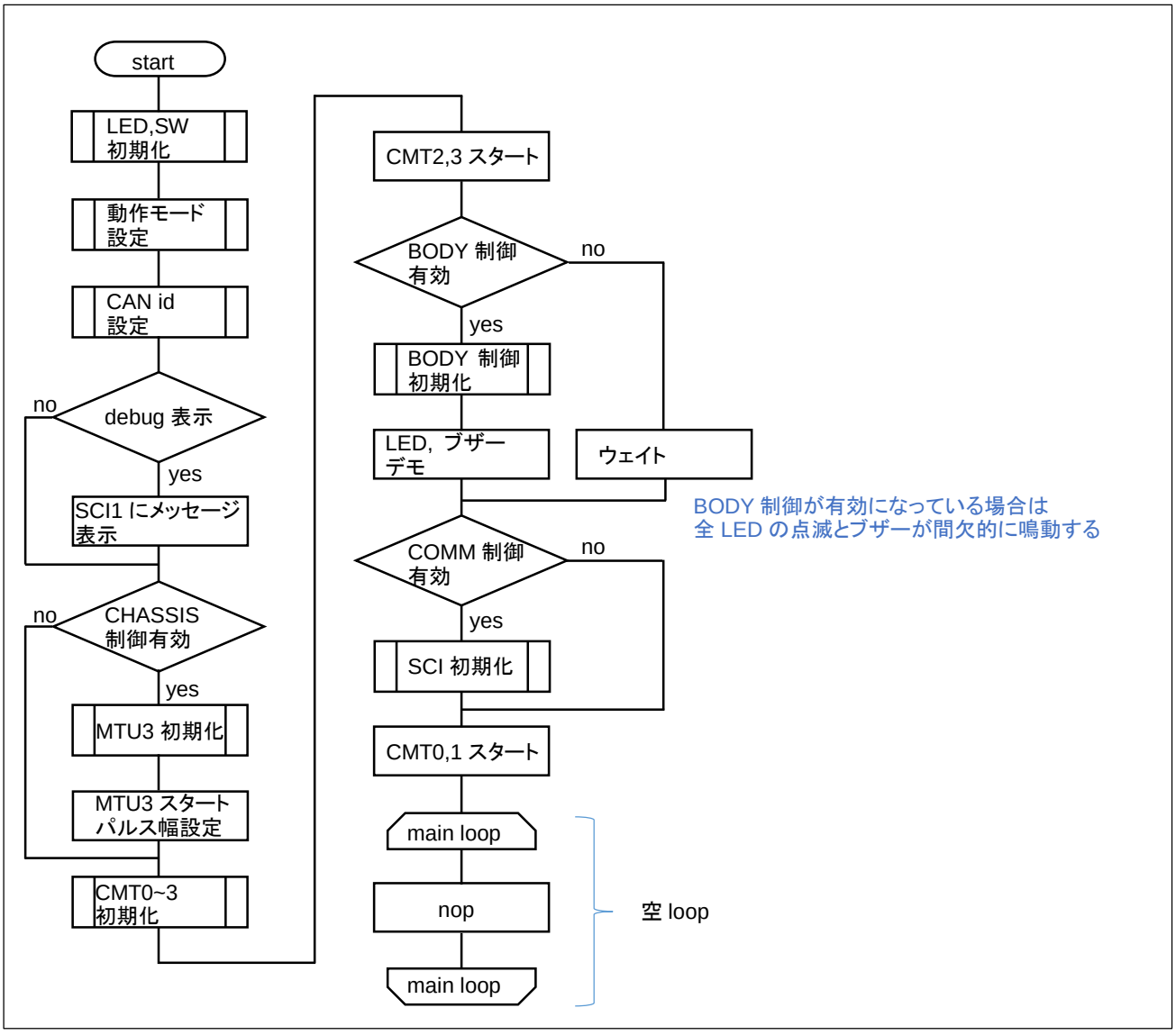
2.1. ソースファイルツリー

RX63T_RCCAR　以下

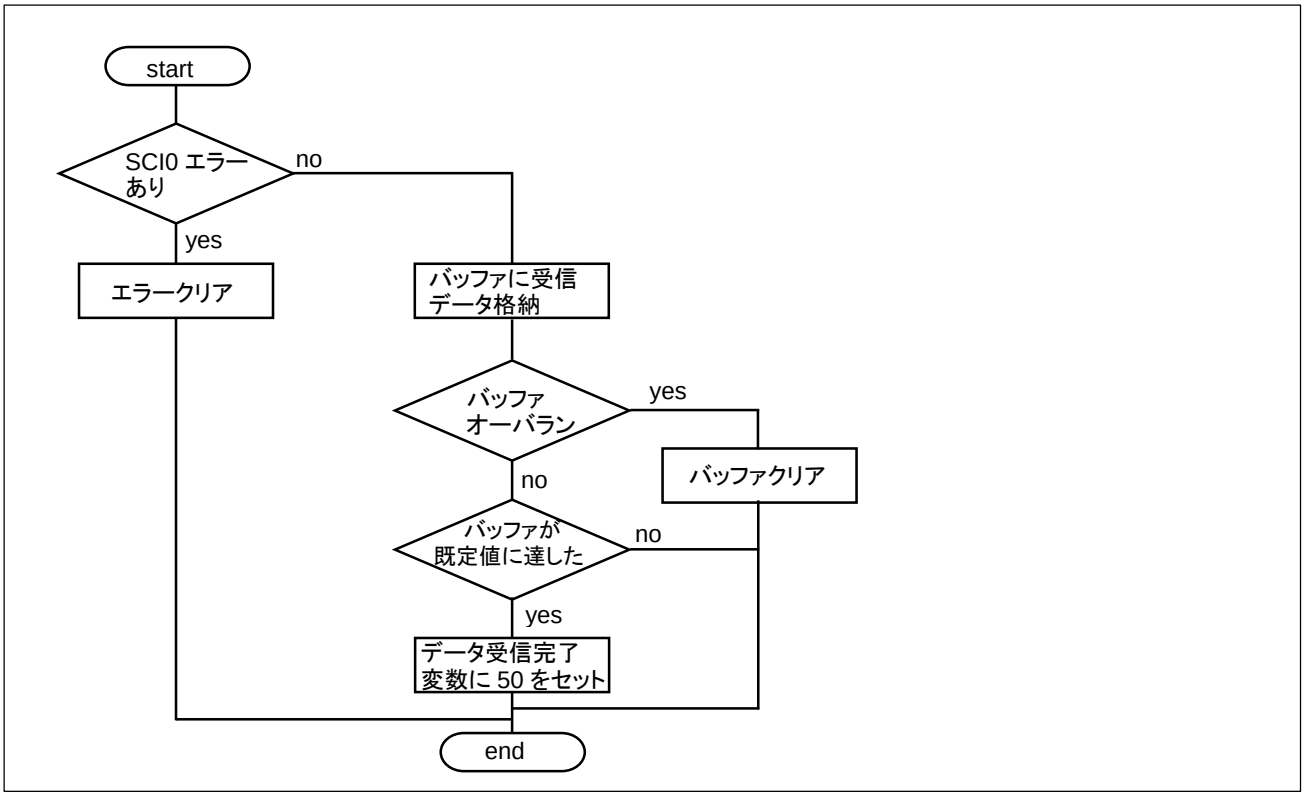
RX63T_RCCAR.c		メイン関数	
rccar/	rccar.c	RC カー制御	
can/	can1drv.c	CAN 部	
sci/	sci.c	SCI 部	
mtu3/	mtu3.c	タイマ部	

2.2. 処理フロー

メイン関数　main()

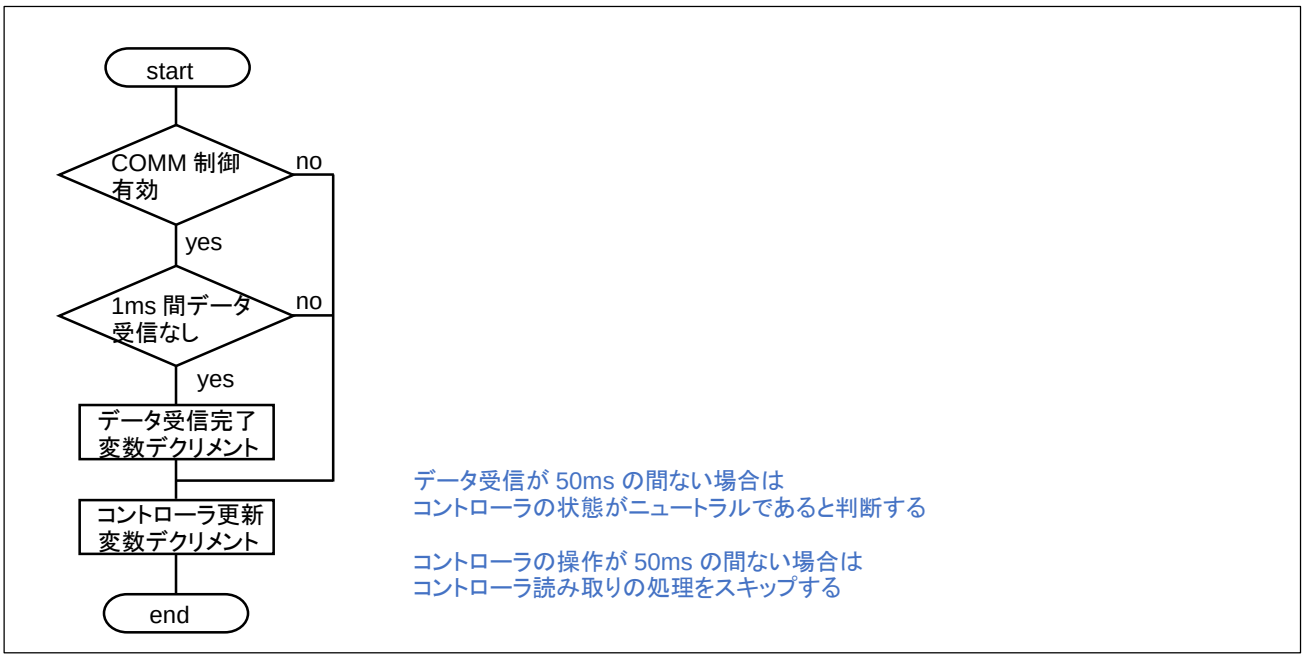


sci0 受信割り込み関数　Intr_SCI0_RXI0()



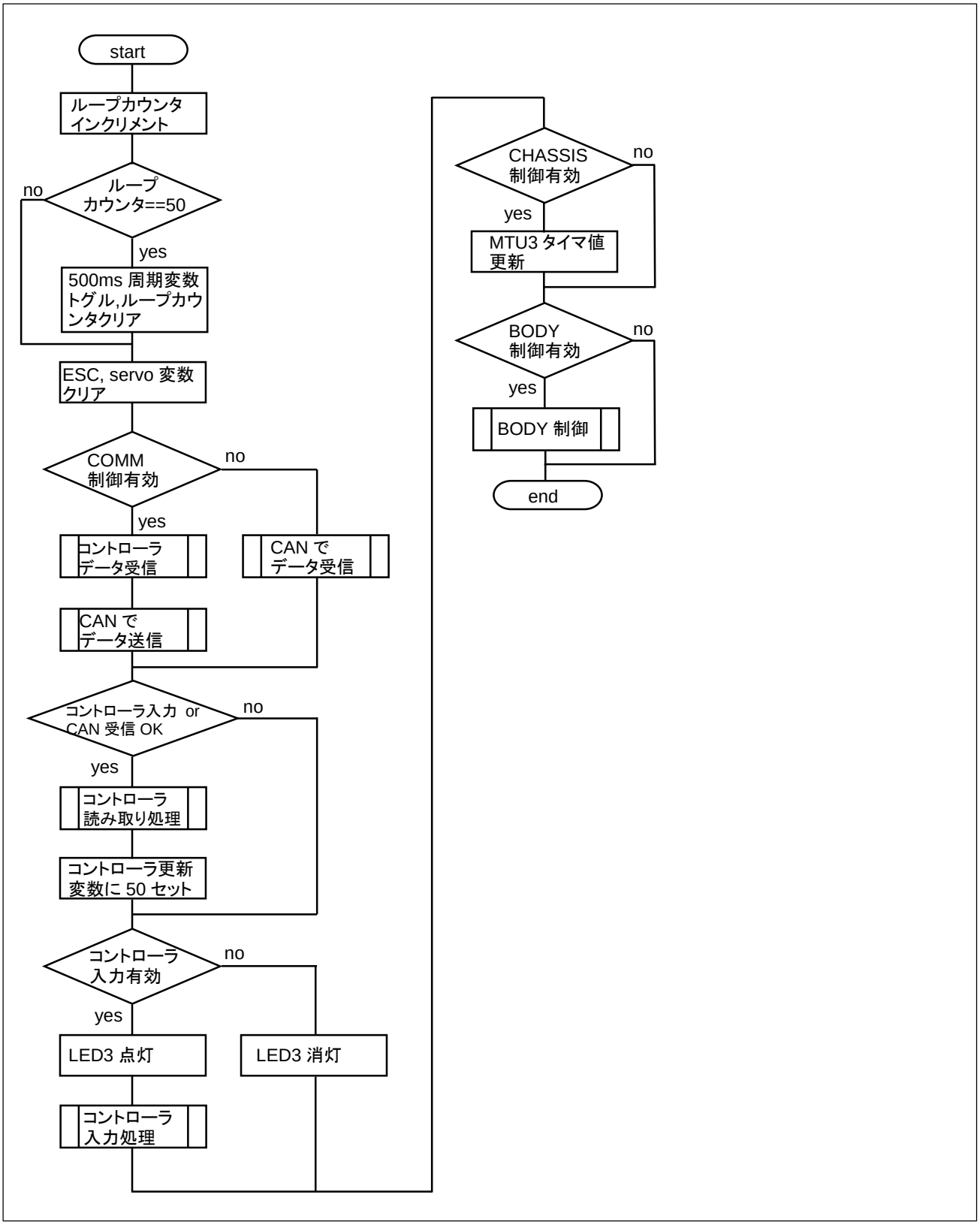
コントローラからの入力（マイコンから見た場合、SCI0 からの受信）を処理する割り込み関数。

1ms 周期割り込み関数　Intr_CMT0_CMI0()



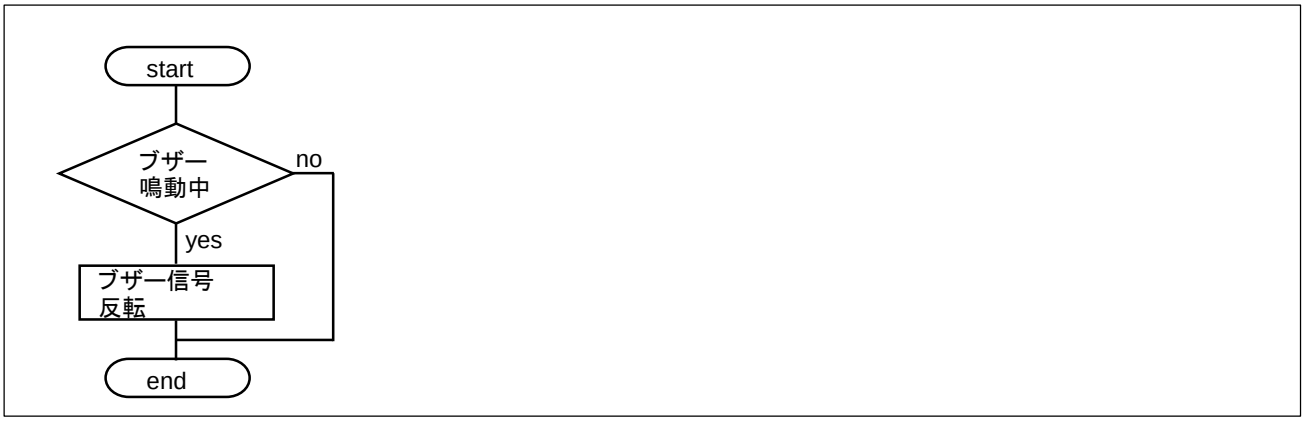
1ms 毎の周期割り込みを使い、50ms 以上の期間コントローラからの入力がない場合、キー入力なしと判断。

10ms 周期割り込み関数　Intr_CMT1_CMI1()



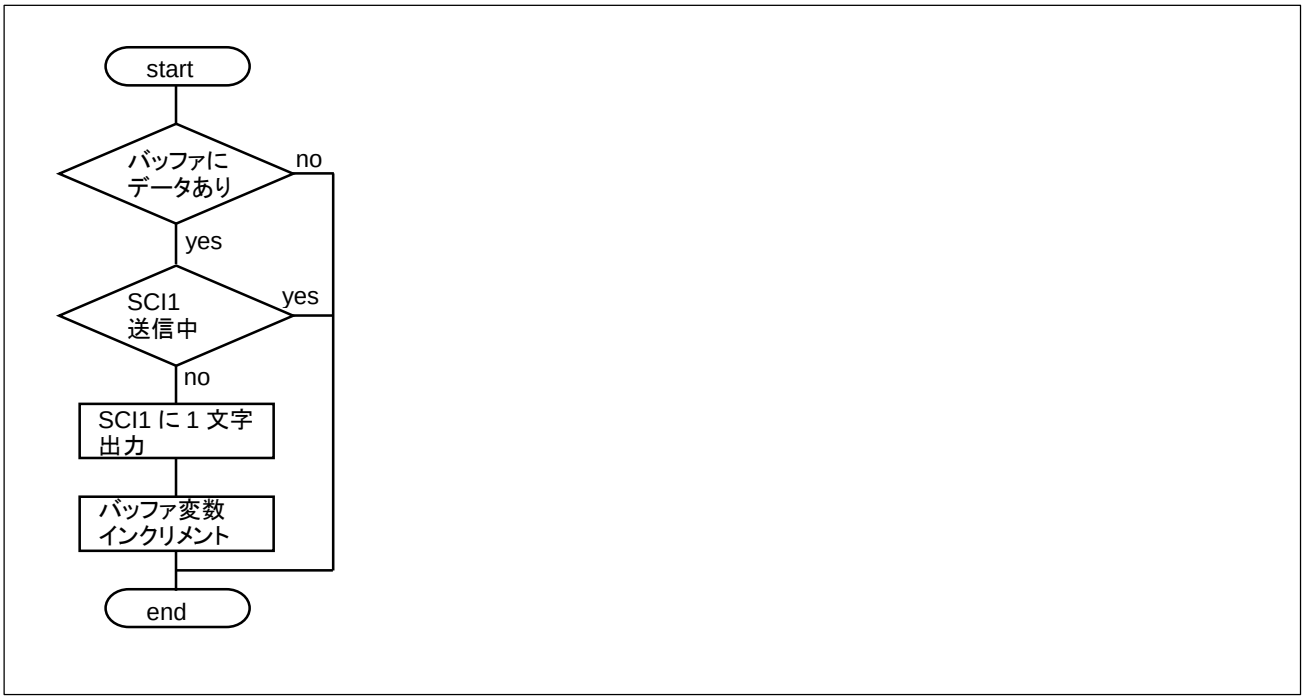
RC カー制御プログラム本体。10ms 毎に呼び出しを行い、コントローラの読み取り、CAN 通信、コントローラのキーの変化、アクセル・ステアリングに応じた車体制御、LED・ブザーの BODY 制御を行う。

8kHz(125us)周期割り込み関数　Intr_CMT2_CMI2()



ブザーを 4kHz で鳴動させるための、周期割り込み。

100us 周期割り込み関数　Intr_CMT3_CMI3()



デバッグ用の SCI1 出力を行う 100us の周期割り込み関数。

2.3. 関数仕様

body_control_init

概要:BODY 制御初期化
宣言:void body_control_init(void)
説明:
・RC カーボディに接続されている I/O ポートを出力制御
・RC カーボディの LED は消灯制御
引数:なし
戻り値:なし

led_sw_init

概要:LED,スイッチ制御初期化
宣言:void led_sw_init(void)
説明:
・マイコンボード本体の LED の I/O ポートを出力制御
・マイコンボード本体の DIP スイッチの I/O ポートを入力制御
引数:なし
戻り値:なし

mode_set

概要:動作モード設定
宣言:void mode_set(void)
説明:
・マイコンボード本体の DIP スイッチを読み取り動作モードを設定
・マイコンボード本体の LED(LED0~LED2)に動作モードを表示
引数:なし
戻り値:なし

canid_set

概要:CAN-ID 設定
宣言:void canid_set(void)
説明:
・動作モードに応じた CAN-ID を設定
引数:なし
戻り値:なし

CAN-ID は、COMM（通信）制御を担当する場合は、0x30 に設定されます。
COMM 制御以外の場合は、
BODY 制御　0x31
CHASSIS 制御　0x32
BODY 及び CHASSIS 制御　0x33
に設定されます。現バージョンでは、CAN-ID が、0x31, 0x32, 0x33 に設定されたボードは、CAN の送信を行いません。コントローラが接続される COMM 制御のボード(CAN-ID=0x30)のみ、コントローラの情報を CAN バスに送信します。

comm_read

概要：コントローラからの通信を読み込む関数
宣言：int comm_read(unsigned char *data)
説明：
・コントローラのデータを*data に格納
引数：
*data :コントローラのデータ(8bytes)
戻り値：
0 :正常終了
1 :入力データなし
2 :パケットエラー

controller_read

概要：コントローラからのデータを読み込む関数
宣言：int controller_read(unsigned char *data)
説明：
・*data から読み込んだ値をグローバル変数に格納
引数：
*data :コントローラのデータ(8bytes)
戻り値：
0 :正常終了

digital_key_state[] にデジタルのボタンデータ、accel, steering にアナログスティックのデータを格納。

body_control_val_change

概要:コントローラからのデータを読み込む関数

宣言:unsigned long body_control_val_change(unsigned long body_control_val, unsigned char no, unsigned char state)

説明:

・ボディ制御変数の値を設定する

引数:

body_control_val :設定前(現状)の変数値

no :変更箇所(0~7), TLU-01 の 1~8 に対応

state: 書き換え後の状態(0:消灯, 1:点灯, 2:点滅)

戻り値:

ボディ制御変数値(unsigned long, 32bit 変数)

	b31-28	b27-24	b23-20	b19-16	b15-12	b11-8	b7-4	b3-0
no	7	6	5	4	3	2	1	0
対応する TLU-01 ch	8	7	6	5	4	3	2	1
対応する I/O ポート	P73	P72	P71	P70	P33	P32	P31	P30
用途	右ウィンカ	ヘッドランプ	フォグランプ	左ウィンカ	バックランプ	ブレーキランプ	未使用	ブザー

ボディ制御変数は、32bit 変数を 4bit 単位で値を設定する

例えば

body_control_val_change(0x00000100, 4, 2) の戻り値は 0x00020100 となる。(no4(b19-16)に 2 をセット)

0x00020001 の場合は左ウィンカ点滅、ブレーキランプ点灯を意味する

→P70 は L/H を 0.5 秒単位で切り替え、P32 は H 制御、他(P73,P72,P71,P33,P31,P30)は L 制御

body_control

概要:ボディ制御関数

宣言:void body_control(unsigned long body_control, unsigned char blink)

説明:

・ボディ制御を行う

引数:

body_control :設定変数値

blink: LED 点滅変数

戻り値:なし

analog_stick_val

概要:アナログスティック関数
宣言:int analog_stick_val(unsigned char analog)
説明:
・アナログスティックの入力値を変換する
引数:
  analog :アナログスティック読み取り値
戻り値:
  変換後の値

  0x40 がセンターとなる値を 0 センターに変換する。

Can1Init

概要:CAN 初期化関数
宣言:void Can1Init(void)
説明:
・CAN(ch1)の初期化を行う
引数:なし
戻り値:なし

  マイコン周辺クロック 50MHz, CAN 通信速度 1Mbps 設定。

Can1Tx

概要:CAN 送信関数
宣言:Can1Tx(unsigned short box_no, unsigned char *pData, unsigned char len)
説明:
・CAN(ch1)からデータを送出する
引数:
  box_no: メールボックス NO
  *pData: 送信データ
  len: 送信バイト数(最大 8)
戻り値:
  0: 正常終了
  -1: 指定メールボックスのデータ送信未完了
  -2: メールボックス番号が正しくない

Can1Rx

概要:CAN 受信関数

宣言:Can1Rx(unsigned short box_no, unsigned char *pData)

説明:

・CAN(ch1)のデータを受信する

引数:

box_no: メールボックス NO

*pData: 送信データ

戻り値:

0<=x: 受信データバイト数(DLC)

-1:指定メールボックスにデータが受信されていない

-2:指定メールボックスのデータ更新中

2.4. グローバル変数

operation_mode

概要:動作モードを格納する変数
宣言:unsigned char operation_mode
説明:
b3: 1 のときデバッグモード（SCI1 に情報表示を行う）
b2: 1 のとき COMM 制御を行う
b1: 1 のとき CHASSIS 制御を行う
b0: 1 のとき BODY 制御を行う

can_id

概要:ボードの CAN-ID を格納する変数
宣言:unsigned short can_id
説明:
コントローラ接続時は、0x30、未接続時は 0x31~0x33 が設定される。

comm_data_update

概要:コントローラからの通信データの更新を表す変数
宣言:unsigned short comm_data_update
説明:
コントローラからの通信があった際、50 に設定され、1ms 毎にデクリメントされる

controller_data_update

概要:コントローラからのデータの更新を表す変数
宣言:unsigned short controller_data_update
説明:
コントローラからのデータがあった際、50 に設定され、1ms 毎にデクリメントされる

digital_key_state

概要:コントローラのキーの状態を示す変数
宣言:unsigned char digital_key_state[DIGITAL_KEY_NUM]
説明:
0:キーが押されていない、1:キーが押されている

digital_key_state_prev

概要:コントローラのキーの前状態を示す変数
宣言:unsigned char digital_key_state_prev[DIGITAL_KEY_NUM]
説明:
0:キーが押されていない、1:キーが押されている

digital_key_pressed

概要:コントローラのキーの変化を示す変数
宣言:unsigned char digital_key_pressed[DIGITAL_KEY_NUM]
説明:
0:キーの変化なし, 1:キーが押されてる状態に変化した

digital_key_released

概要:コントローラのキーの変化を示す変数
宣言:unsigned char digital_key_released[DIGITAL_KEY_NUM]
説明:
0:キーの変化なし, 1:キーが離されている状態に変化した

body_control_state

概要:ボディ制御の状態を表す変数
宣言:unsigned char body_control_state[TLU01_NUM]
説明:
0:消灯, 1:点灯, 2:点滅

body_control_hazard

概要:ハザード(左右のウィンカ点滅)の状態を表す変数
宣言:unsigned char body_control_hazard
説明:
0:ハザード OFF, 1:ハザード ON

controller_packet

概要:コントローラのパケットを保存する変数
宣言:unsigned char controller_packet[CONTROLLER_PACKET]
説明:
　コントローラのデータ(8bytes)

accel

概要:アクセルの状態を表す変数
宣言:int accel
説明:
　-0x40（後進）~ 0x3f（前進）, 0 がアクセル OFF

steering

概要:ステアリングの状態を表す変数
宣言:int steering
説明:
　-0x40（左）~ 0x3f（右）, 0 がステアリングセンター

esc

概要:ESC のパルス幅を表す変数
宣言:unsigned short esc
説明:
　us 単位（1.5ms→ 1500）

servo

概要:SERVO のパルス幅を表す変数
宣言:unsigned short servo
説明:
　us 単位（1.5ms→ 1500）

buzz_active

概要:ブザーの鳴動状態を表す変数
宣言:unsigned char buzz_active
説明:
　0:OFF, 1:鳴動

esc_gain

概要:ESC のゲイン（設定幅）を表す変数
宣言:unsigned short esc_gain
説明:
　初期値 150（ESC タイミングの設定振れ幅を 150us とする）

servo_idle

概要:ステアリングサーボのセンターを表す変数
宣言:unsigned short servo_idle
説明:
　初期値 1550（ステアリングサーボの初期パルス幅を 1550us にする）

break_valid1

概要:ブレーキの保持時間(1)変数
宣言:unsigned short break_valid1
説明:
　10ms 毎に減算されるブレーキ保持時間(1)

break_valid2

概要:ブレーキの保持時間(2)変数
宣言:unsigned short break_valid2
説明:
　10ms 毎に減算されるブレーキ保持時間(2)

　アクセルをバック方向に「ブレーキ保持時間(1)」入れた後、前進方向に「ブレーキ保持時間(2)」入れる事により、ESC でブレーキが有効となる。

break_on

概要:ブレーキの ON 状態を表す変数

宣言:unsigned char break_on

説明:

0:ブレーキは OFF, 1:ブレーキを ON にしている

2.5. 定数定義

```
#define OFF  0
#define ON  1
#define BLINK  2
```

LEDの点灯／消灯／点滅を定義する定数

```
#define MTU3_SERVO  4
#define MTU3_ESC  7
```

MTU3タイマのchを定義する変数
SERVOは、MTIOC4C(ch4), ESCはMTIOC7C(ch7)を使用している。

```
#define TLU01_NUM  8
```

TLU-01のch数を定義

```
#define BUZZER  0
#define NC  1
#define BREAK_LAMP  2
#define BACK_LAMP  3
#define LEFT_WINKER  4
#define FOG_LAMP  5
#define HEAD_LAMP  6
#define RIGHT_WINKER  7
```

名称をTLU-01のch番号にマッピング。
※TLU-01のch番号は1~8ですが、上記のマッピングでは0~7を使用しています

```
#define CONTROLLER_PACKET  8
```

コントローラの1パケットのバイト数

```
#define CONTROLLER_START_CODE  0x80
```

コントローラから送られてくるスタートコード

#define DIGITAL_KEY_NUM12

（本プログラムで使用している）コントローラのディジタルキー数の定義

```
#define DIGITAL_UP  0
#define DIGITAL_DOWN  1
#define DIGITAL_LEFT  2
#define DIGITAL_RIGHT  3
#define DIGITAL_SANKAKU  4
#define DIGITAL_MARU  5
#define DIGITAL_BATSU  6
#define DIGITAL_SHIKAKU  7
#define DIGITAL_L1  8
#define DIGITAL_L2  9
#define DIGITAL_R1  10
#define DIGITAL_R2  11
```

キー名称と番号のマッピング。
※送られてくるキーコードと対応している訳ではなく、便宜上割り振っている番号です

#define CYCLE 20000

ESC，サーボの、パルス波形の周期[us] (20ms)

#define ESC_IDLE 1550

コントローラのスティックをセンターにしているにも拘わらず、
前進・後進するようであれば、この値を増減させてください

ESCアイドル時のパルス幅[us] (1550us)

#define SERVO_IDLE 1550

この値を変更すると、初期のステアリングの中心値が変わります
（ステアリング調整は、コントローラの左右キーでも調整可能です）

ステアリングサーボセンターのパルス幅[us] (1550us)

```
#define SERVO_MAX  1900
#define SERVO_MIN  1200
```

ステアリングサーボの設定最大、最小値[us]

#define ESC_MAX 1800
#define ESC_MIN 1300

ESCの設定最大、最小値[us]

#define ESC_GAIN 150

ESCの初期設定幅[us]（±150us）

#define ESC_GAIN_MIN 100
#define ESC_GAIN_MAX 200

ESCの設定幅の最大、最小値[us]

#define SERVO_GAIN 300

ステアリングサーボの設定幅[us]（±300us）

#define SERVO_GAIN_FINE 9

ステアリングサーボの微調整刻み[us]

#define ESC_GAIN_FINE 5

ESC設定幅の微調整刻み[us]

#define CONTROLLER_RANGE 0x40

コントローラアナログスティックの入力幅

#define CONTROLLER_KEEP_TIME 50

コントローラの入力を有効とみなす期間[ms]

#define BREAK_TIME1 8
#define BREAK_TIME2 8

ブレーキの保持時間(1)(2) [ms]

```
#define BREAK_STRENGTH1  -80
#define BREAK_STRENGTH2  80
```

ブレーキの強さ[us]
ブレーキを掛ける際、BREAK_TIME1の時間、BREAK_STRENGTH1のタイミングでESCを制御し、その後BREAK_TIME2の時間、BREAK_STRENGTH2のタイミングでESCを制御する

```
#define  OPERATION_BODY  1
#define OPERATION_CHASSIS  2
#define OPERATION_COMM  4
#define OPERATION_DEBUG  8
```

動作モード定義

```
#define CAN_TIMEOUT  5
```

CAN受信タイムアウト
5回待って有効な受信データが得られないときは、受信エラーとして処理する

```
#defineCAN_WAIT  900
```

CAN受信タイムアウト時間の設定
900→約100us
最大の待ち時間は100us×5回(CAN_TIMEOUT)

```
#define CAN_RECEIVE_MB  0
```

CANの受信メールボックスとしては、0番を使用する

```
#define CAN_SEND_MB  4
```

CANの送信メールボックスとしては、4番を使用する

```
#define CAN_DLC  8
```

CAN のデータは、8byte とする

```
#define CAN_ID_BASE  0x30
```

CAN のベース ID は 0x30 とする

2.6. 使用しているマイコン機能

マイコン機能	用途	備考
MTU3	ESC, SERVO の PWM 波形生成	MTIOC4C MTIOC7C を使用
CMT0	1ms, コントローラ入力タイムアウト処理	
CMT1	10ms. メイン処理	
CMT2	8kHz, ブザー	
CMT3	100us, SCI1 バッファ出力	
SCI0	コントローラ入力	
SCI1	デバッグ情報出力	
CAN1	ボード間通信	MD[0]を受信 MD[4]を送信 に使用

2.7. デバッグ表示に関して

ボードの J6(USB-miniB)端子経由で、SCI(UART)による情報出力機能があります。

端末を 115,200bps に設定し、SW2-4 を ON にした状態で電源を投入してください。

```
Copyright (C) 2018 HokutoDenshi. All Rights Reserved.
HSBRX63T-100RC RC-CAR boot!
SCI ch-1 for debug

- control node information -
BODY : o
CHASSIS : o
COMM : o
CHASSIS -> ESC: 1550 SERVO: 1550
BODY -> val: 00000000
CTRL -> D:                              MARU                                    / A: 40 40 40 40
CTRL -> D:                              MARU                                    / A: 40 40 40 40
```

端末に上記のような表示が出力されます。

CHASSIS -> は、ESC に送信するパルス幅の情報（0.5s に 1 回）。

BODY -> は、ボディに送信する LED, ブザー制御の情報（0.5s に 1 回）。

CTRL -> は、コントローラから読み取った、キー押下、アナログスティック値が表示されます。

2.8. 本プログラムソースの環境

本プログラムは、ルネサスエレクトロニクス製 CS+向けのプロジェクトとなっています。
プログラムをビルド、改変する場合は、CS+で再コンパイル、ビルドを行ってください。

取扱説明書改定記録

バージョン	発行日	ページ	改定内容
REV.1.0.0.0	2018.5.7	—	初版発行

お問合せ窓口

最新情報については弊社ホームページをご活用ください。

ご不明点は弊社サポート窓口までお問合せください。

株式会社 北斗電子

〒060-0042 札幌市中央区大通西 16 丁目 3 番地 7

TEL 011-640-8800　FAX 011-640-8801

e-mail：support@hokutodenshi.co.jp (サポート用)、order@hokutodenshi.co.jp (ご注文用)

URL:http://www.hokutodenshi.co.jp

ルネサス エレクトロニクス RX63T(QFP-100 ピン)搭載
HSB シリーズマイコンボード

HSBRX63T-100RC
RC カー制御　ソフトウェア編　マニュアル

株式会社 北斗電子
 2018 年 5 月 7 日改訂 REV.1.0.0.0 (180507)