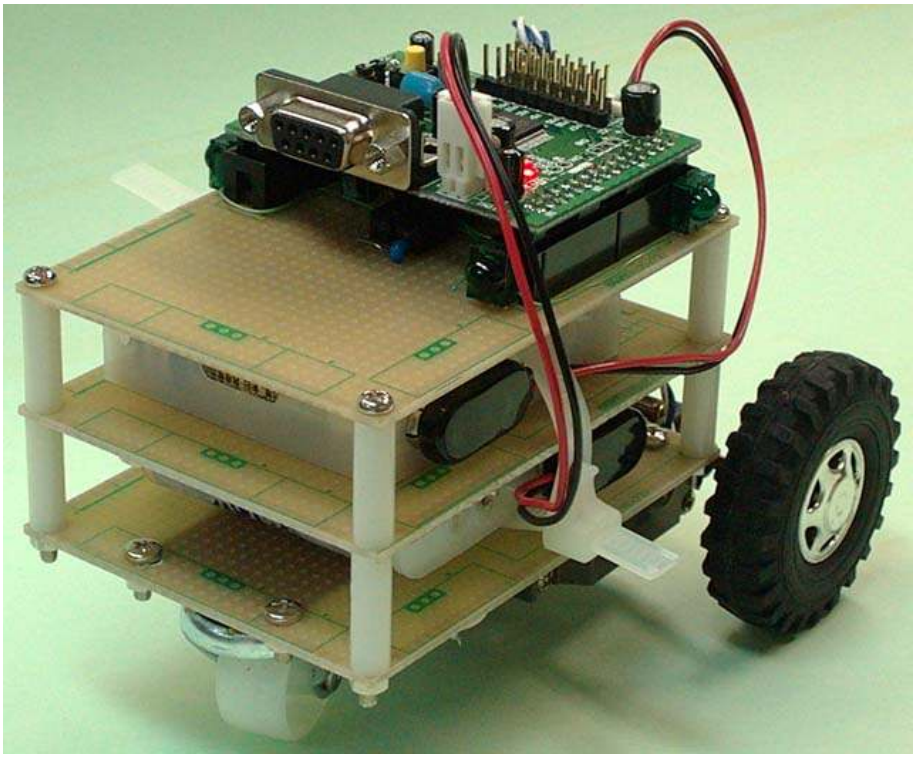
TK-3687mini オプション

赤外リモコンカー“F1μ”の作成

Version_110114



「F1μ」のプログラムファイルは付属 CD の「f1micro」フォルダの中にあります。
詳細は「CD:¥f1micro¥readme.txt」をご覧ください。

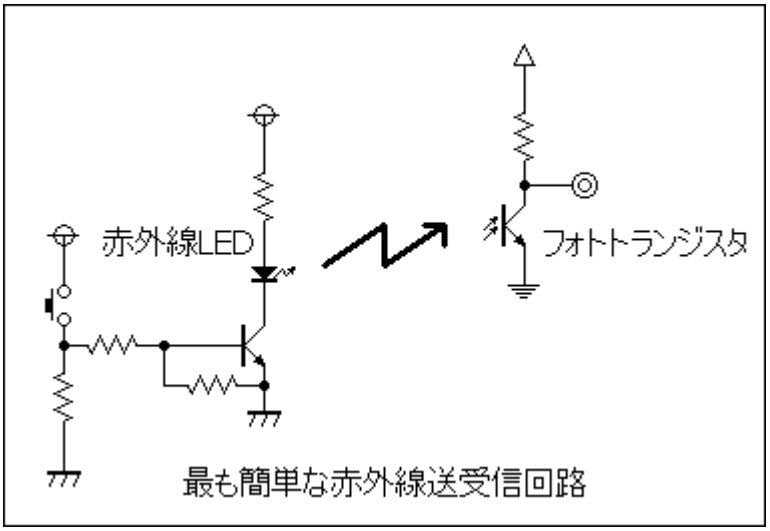
(株)東洋リンクス

第1章 赤外線送受信回路

1-1. 赤外線送受信回路

1-1. 赤外線送受信回路

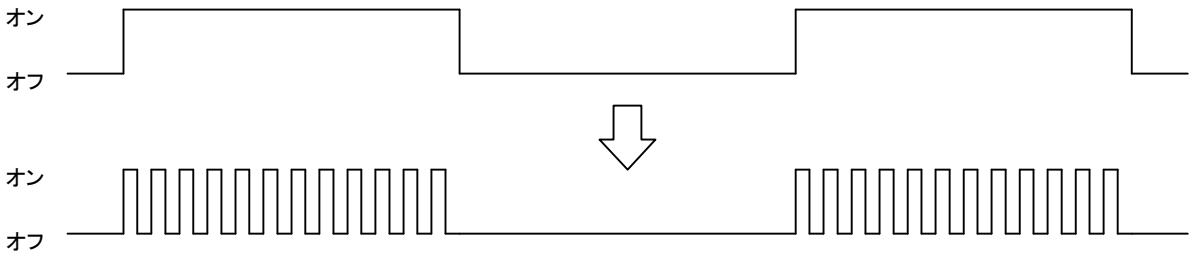
赤外線送受信回路の基本的な形は，赤外線 LED とフォトトランジスタ(もしくはフォトダイオード)を使った右のような回路になります。(スイッチをオンすると赤外線 LED が発光し，フォトトランジスタがオンする)



最も簡単な赤外線送受信回路

ただ，実際にやってみるとわかりますが，この回路はまったくと言っていいほど使い物になりません。おそらく数センチ離しただけで届かなくなりますし，送信側のスイッチを押していないにもかかわらず受信側は頻繁にオンします。

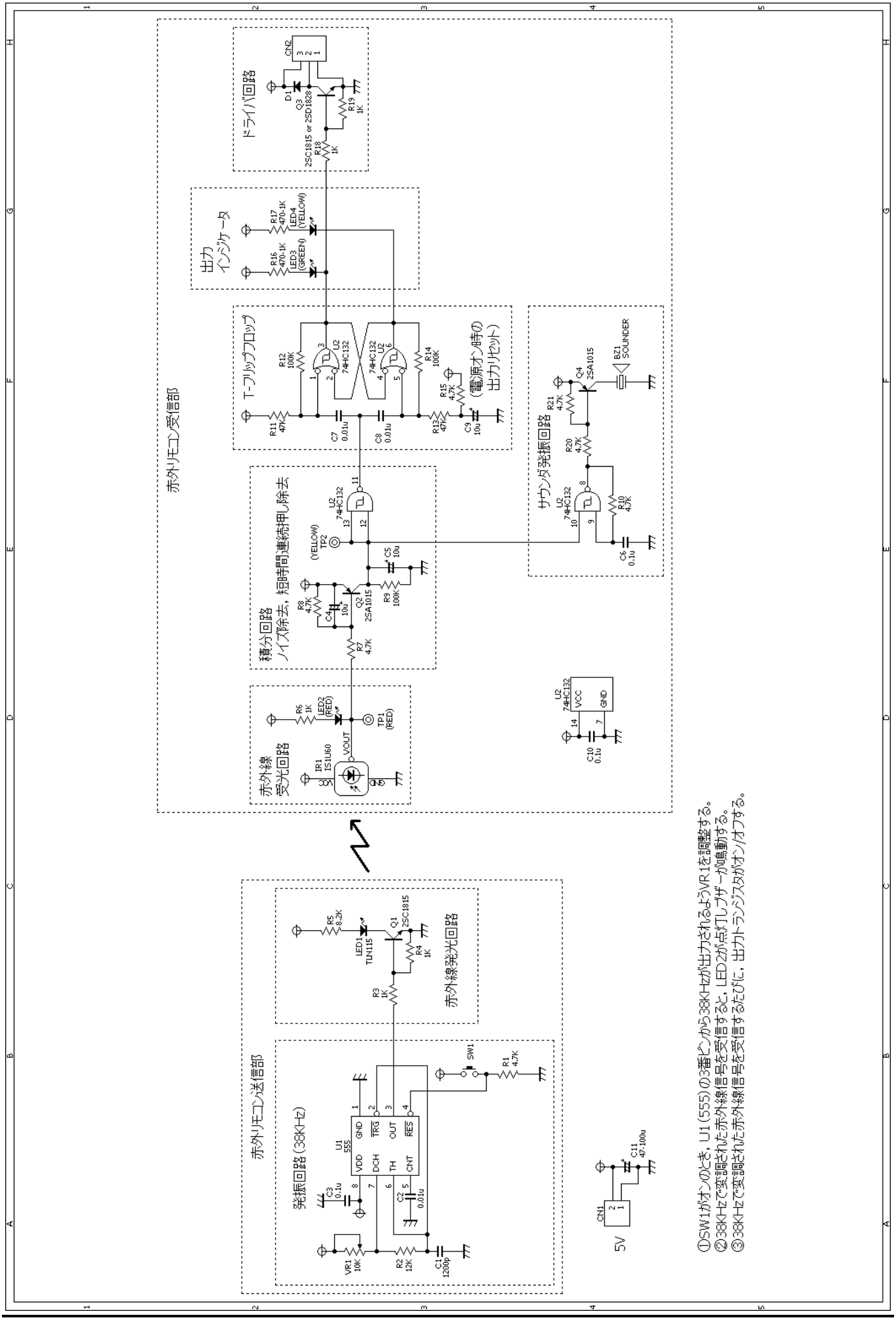
原因は，わたしたちの周囲に赤外線を出すものが氾濫していることです(太陽光線，白熱電球，蛍光灯など)。そのため，それらがノイズとなって通信を邪魔します。そこで，伝えたいデータをキャリアで変調する方法が使われています(下図参照，この図は正論理で描いています)。なぜ，キャリアで変調するとノイズに強くなるのでしょうか。

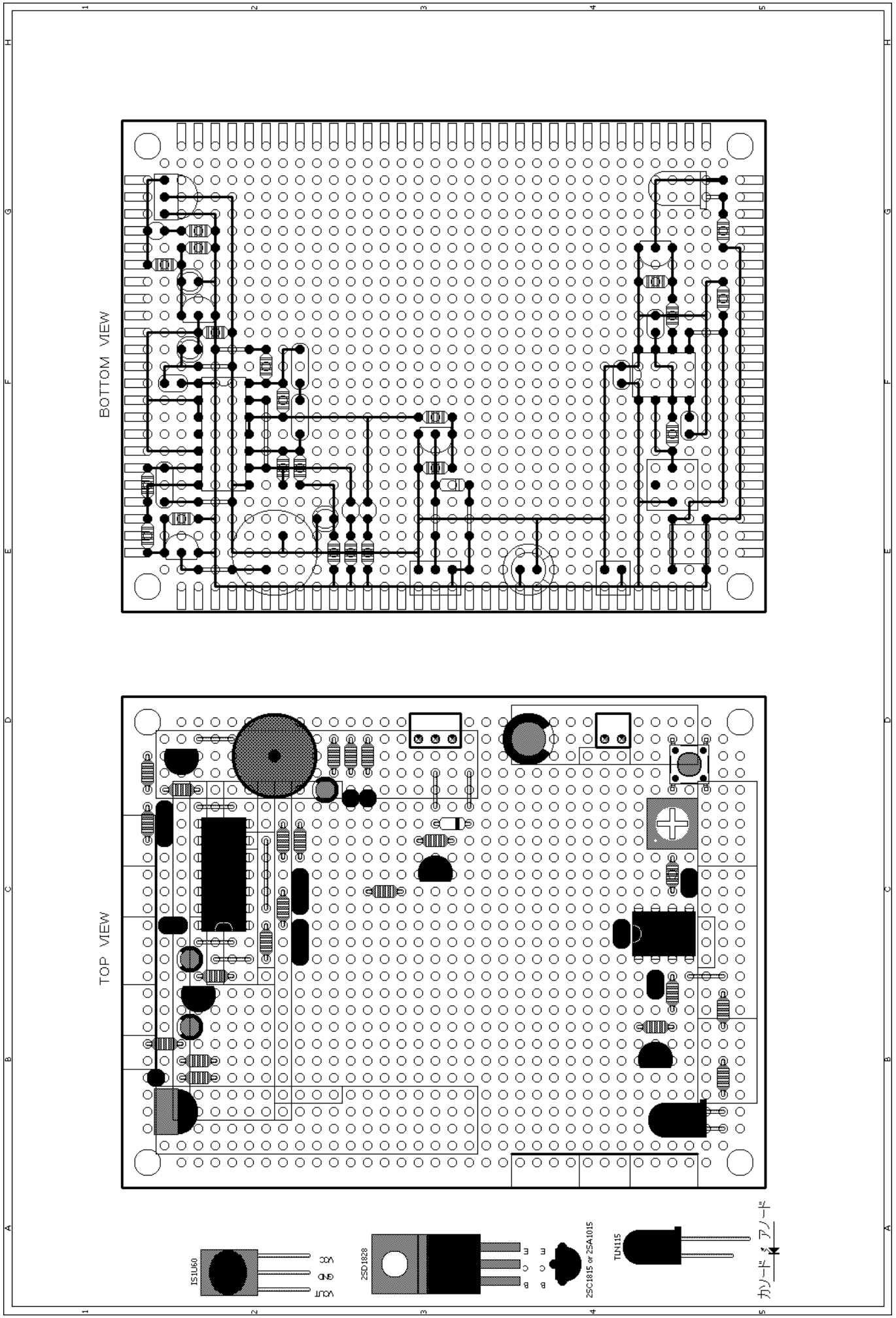


ノイズに強くするためには，強力な赤外線を発光してデータの強さがノイズレベル以上になるようにする必要があります。そのために赤外発光 LED に大きな電流を流したいところです。ところが，LED の最大定格を見てみると(TLN115 の場合)直流順電流は 100mA までです。これだと弱すぎて遠くまで信号を伝えることができません。もちろん，最大定格を超える電流を流しつづけることはできません。しかし，よく見るとデータシートにはもう一つ，パルス順電流というのが載せられていて，TLN115 の場合 1A です。これは，ごく短い時間であれば 1A まで流すことができることを示しています。そこで，変調することによって(つまりパルス波形にして＝短い時間だけオンするようにして)大きな電流を流し，より強い赤外線を発光するようにします。

さらに，変調することにより，受信側にキャリア周波数だけ通すバンドパスフィルタ回路を追加することでノイズを除去することができます。多くの赤外リモコンはキャリア周波数として 38KHz を採用しています。赤外信号受光 IC (IS1U60, TSOP1738, 等)は 38KHz だけを通すバンドパスフィルタが内蔵されており，キャリアを除去したあとの復調された信号が**負論理**で出力されます。

ではここで，これまでのことを踏まえて検討したハードウェアだけで構成した赤外リモコン実験回路を次ページから紹介します。1 枚の基板に送信回路と受信回路を実装した実験回路です。SW1 をオンすると 38KHz で変調された赤外線信号が出力され，変調された赤外線信号を受信すると LED2 が点灯，ブザーが鳴動し，さらに，SW をオンするたびにトランジスタスイッチがオン/オフします。もし興味があるなら部品を集めて組み立ててみてください。


赤外リモコン受信部
ドライバ回路
CN2
D1
Q3
2SC1815 or 2SD1828
R19 1K
R18 1K
出力
インジケータ
R17 470-1K
LED4 (YELLOW)
R16 470-1K
LED3 (GREEN)
T-フリップフロップ
R12 100K
U2 74HC132
R14 100K
R11 47K
R13 47K
R15 4.7K
C7 0.01u
C8 0.01u
C9 10u
(電源オン時の出力リセット)
積分回路
ノイズ除去，短時間連続押し除去
U2 74HC132
(YELLOW) TP2
C5 10u
R8 4.7K
C4 10u
Q2 2SA1015
R9 100K
R7 4.7K
サウンド発振回路
R21 4.7K
Q4 2SA1015
BZ1 SOUNDER
R20 4.7K
R10 4.7K
C6 0.1u
赤外線
受光回路
R6 1K
LED2 (RED)
TP1 (RED)
IR1 1S1U60
VOUT
U2 74HC132
VCC
GND
C10 0.1u
赤外リモコン送信部
発振回路(38KHz)
U1 555
GND
TRG
OUT
RES
VDD
DCH
TH
CNT
C3 0.1u
C2 0.01u
C1 1200p
VR1 10K
R2 12K
SW1
R1 4.7K
赤外線発光回路
R5 8.2K
LED1 TLN115
Q1 2SC1815
R4 1K
R3 1K
CN1
5V
C11 47-100u
①SW1がオンのとき，U1(555)の3番ピンから38KHzが出力されるようVR1を調整する。
②38KHzで変調された赤外線信号を受信すると，LED2が点灯しブザーが鳴動する。
③38KHzで変調された赤外線信号を受信するたびに，出力トランジスタがオン/オフする。


BOTTOM VIEW
TOP VIEW
IS1U60
2SD1828
2SC1815 or 2SA1015
TUN115
カソード
アノード

赤外リモコン実験回路

	部品番号	型名，規格	メーカー	数量	付属数量	備考
1	**■赤外線信号送信部**					
2	LED1	TLN115	東芝	1	1	*1，赤外LED
3	SW1	SKHHAK/AM/DC	ALPS	1	1	*1
4	R1	4．7KΩ		1	1	
5	R2	12KΩ		1	1	
6	R3，4	1KΩ		2	2	
7	R5	8．2KΩ		1	1	
8	VR1	10KΩ		1	1	
9	C1	1200pF（フィルム）		1	1	
10	C2	0．01μF（フィルム）		1	1	
11	C3	0．1μF（セラ）		1	1	
12	Q1	2SC1815		1	1	*1
13	U1	555		1	1	*1
14						
15	**■赤外線信号受信部**					
16	IR1	IS1U60	SHARP	1	1	*1
17	LED2			1	1	*1，赤
18	LED3			1	1	*1，緑
19	LED4			1	1	*1，黄
20	U2	74HC132		1	1	
21	R6，18，19	1KΩ		3	3	
22	R7，8，10，15，R20，21	4．7KΩ		6	6	
23	R9，12，14	100KΩ		3	3	
24	R11，13	47KΩ		2	2	
25	R16，17	470Ω～1KΩ		2	2	LEDの輝度によって選択
26	C4，5，9	10μF/16V		3	3	
27	C6	0．1μF（フィルム）		1	1	
28	C7，8	0．01μF（フィルム）		2	2	
29	C10	0．1μF（セラ）		1	1	
30	Q2，4	2SA1015		2	2	*1
31	Q3	2SC1815 2SD1828		1	1	*1（負荷によって選択）
32	D1			1	1	逆起電力吸収用
33	BZ1	QMX-05	スター精密	1	1	*1，サウンダ
34	CN2	B3P-SHF-1AA	JST	1	1	
35						
36	**■その他**					
37	CN1	B2P-SHF-1AA	JST	1	1	
35	C11	47～100μF/16V		1	1	
38	ユニバーサル基板	B6093(95×72㎜)	東洋リンクス	1	1	
39	ラッピングケーブル	50cm		1	1	*2，メッキ線として使用
40	メッキ線			0	0	ハンダ面結線用　*2
41						

(*1)相当品を使用することがあります。
(*2)ラッピングケーブルの被覆をはがし2本をよじって使用します。また，抵抗やコンデンサの足も流用できます。

◆

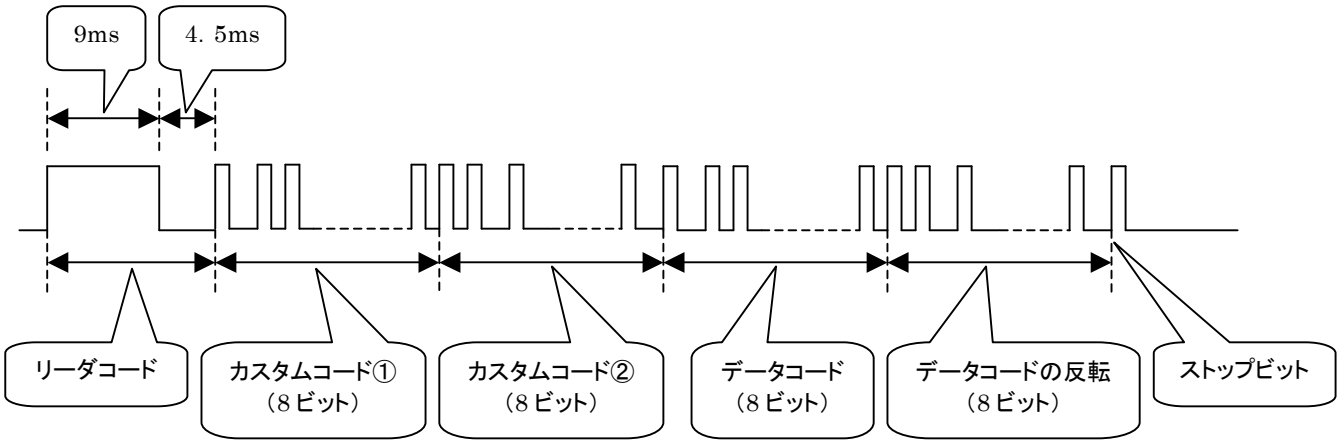
動かしてみるとわかりますが，実際にはこれでも誤動作します。赤外信号受信 IC のバンドパスフィルタでも除去しきれずに出力されてしまったり，他の赤外線が重なることで信号が欠けてしまったりするためです。それで，通常はマイコンと組み合わせてプログラムでさらにノイズを除去し，正しいデータを取り出すようにします。次に，赤外リモコンの信号フォーマットを見てみましょう。

第2章 赤外リモコンの信号フォーマット

2-1. NEC フォーマット
2-2. 受信プログラムの考え方

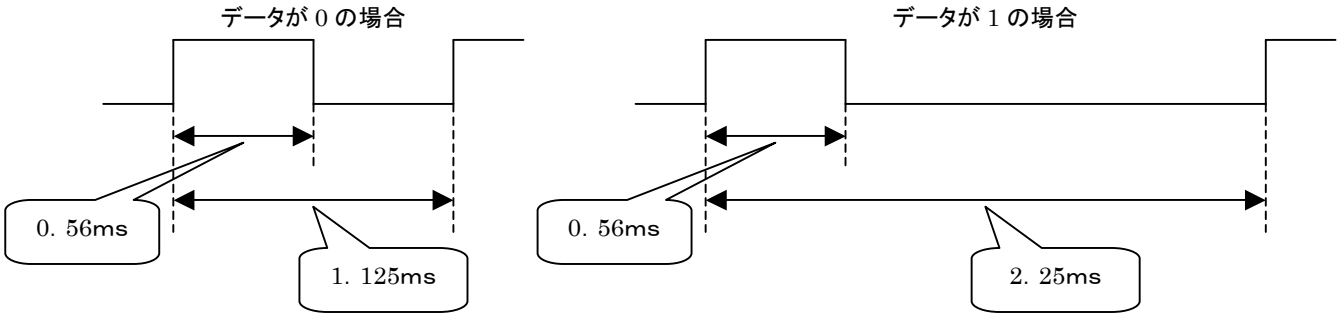
2-1. NEC フォーマット

赤外リモコンでは，赤外線を利用してデータを低速で送信し，受信側は赤外線を検知してデータを受け取ります。リモコン信号はシリアル信号なので，受信プログラムでは，どこからデータが始まったのか，本当に正しく受信できたのか，判断する必要があります。そのためのデータフォーマットが定められています。世の中でよく使われている標準的なフォーマットが何種類か存在しますが，その一つ，NEC フォーマットを見てみましょう。NEC フォーマットは 1 バイト（＝8 ビット）のデータを送受信するフォーマットで，次のような構成になっています（この図は正論理で描いています，また，実際は送信時に変調されます）。



リーダコードはこれからデータが始まることを示すコードです。9ms の期間オンの状態が続き，その後 4. 5ms の期間オフ状態になります。他のコード（カスタムコードやデータコード）と比較して波形が大きく異なるため，容易にリーダコードであることがわかります。

続く，カスタムコードやデータコードが 0/1 のデータを含む部分です。各部分は下位ビットから送信されます。そして，0/1 は赤外線の有無ではなく，下図のように信号の長さで区別します（この図は正論理で描いています）。

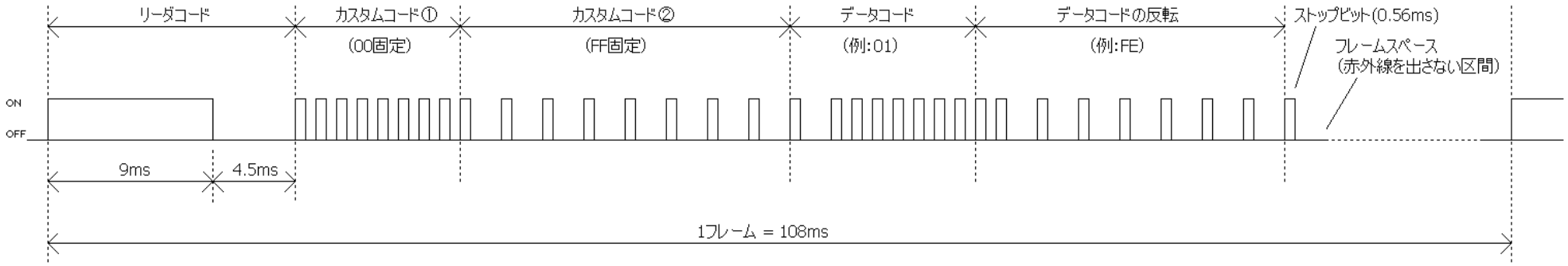


カスタムコード①，②は，メーカによって誤動作しないよう区別するコードで，メーカごとに NEC から割り当てられます。NEC に登録しなくてもメーカに関係なく自由に使えるコードもあり，今回は“00”，“FF”を使っています（注意：このカスタムコードを使っている他の赤外リモコン機器があれば動作してしまう可能性があります）。

次に，データコードを送信します。データコードは 8 ビットなので 0～255（00h～FFh）まで 256 種類のデータを送信することができます。続いて，データコードの全てのビットを反転したデータを送信します。受信プログラムでは，受信したデータコードと続けて受信した反転データコードを比較して正しい値か確認し，間違ったデータを受信したときはそのデータを捨て，正しいデータだけを採用します。

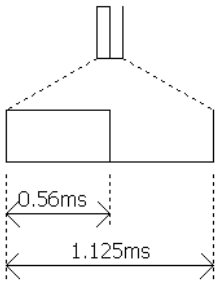
次のページにデータ“01”を NEC フォーマットで送信する場合のタイミングチャートを示します。カスタムコードは“00”，“FF”にしました。

NECフォーマットの赤外線リモコン信号
(データ＝01を送信する場合)

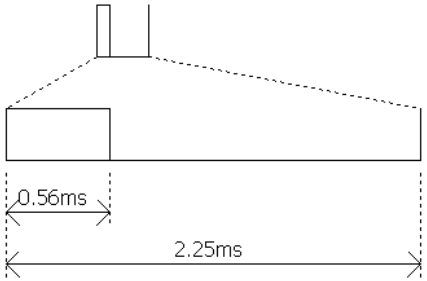


■リモコン信号のビット=0/1の違い

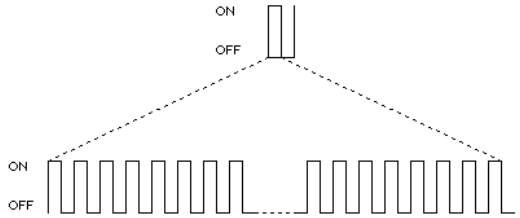
<ビット＝0>



<ビット＝1>



■赤外線ONの期間は38KHzで変調する



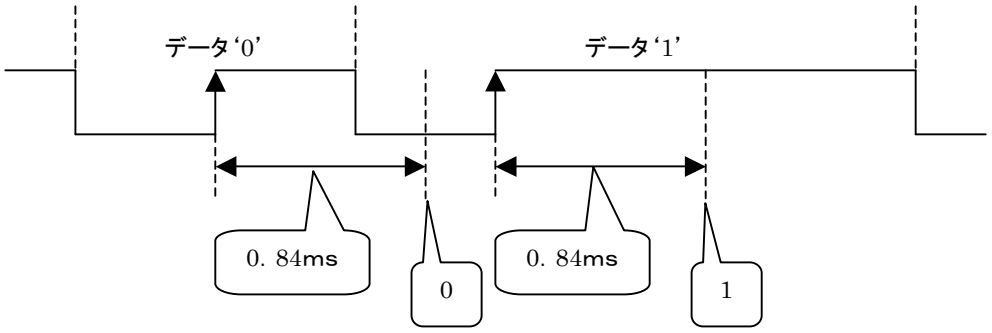
■データは下位ビットから送信する

例1:01(16進数)を送る場合は10000000の順番。
例2:FE(16進数)を送る場合は01111111の順番。

2-2. 受信プログラムの考え方

受信プログラムは，まず赤外線信号が入力されたかチェックし，入力されたならばその信号がリーダコードかチェックします。リーダコードかどうかは赤外線信号のオン時間とオフ時間がある範囲に収まるかで判断します。許容範囲をどれくらいにするかは使用する環境にもよりますが，今回は±10%（オン時間=8．1ms～9．9ms，オフ時間=4．05ms～4．95ms）にしました。

リーダコードを受信したら，続いて受信プログラムは，赤外線信号のオン時間とオフ時間をチェックして 0/1 を判断していきます。下図のように，赤外信号受光 IC の負論理出力の立ち上がりから 0．84ms 後のポートの状態で判断しています（この図は赤外信号受光 IC の出力，つまり負論理で描いています）。



受信したカスタムコードが自分のコードであればデータを採用します。次に，受信したデータコードと続けて受信した反転データコードを比較し正しいデータだけ採用します。

ところで，信頼性を向上させるために，通常のデータ通信では正しく伝わったか互いに確認しながらデータを送受信します。しかし，赤外リモコンの場合，送信器から一方的にデータを送るだけなので（垂れ流しと言います），正しく受け取ったか確認したり，もう一度送るよう要求したりすることができません。そこで，スイッチが押されている間は繰り返し同じデータを送信し続け，多少データを取りこぼしたとしても動作に影響しないように受信側もプログラムします。

◆

NEC フォーマットについての詳しい説明は次の Web サイトをご覧ください。（2008 年 11 月 19 日現在）

http://www.necel.com/ja/faq/mi_com/__com_remo.html

第3章 赤外リモコンカーの組み立て

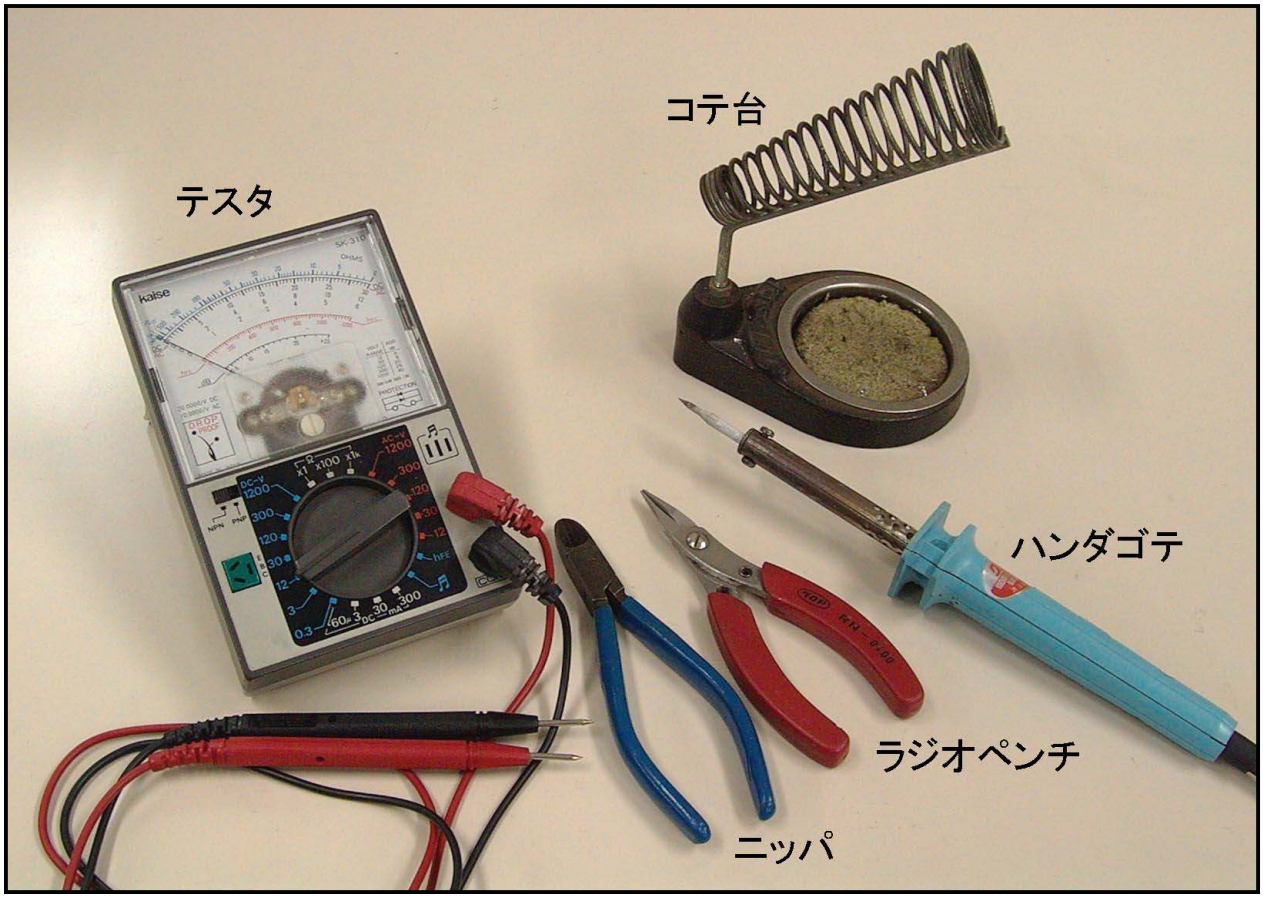
3-1. 工具の準備

■ **モータドライバ基板と CPU 基板の組み立て**

ハンダゴテは 30～40W 程度のものを用意して下さい。キットにハンダは入っていません。太さ 1mm 程度のハンダを用意してください。余分なリード線を切るためのニッパ，リード線を折り曲げるためのラジオペンチ，ラッピングケーブルの被覆をむくためのワイヤストリッパ，細かい配線に使用するピンセット，コテ台などを用意します。また，電圧や抵抗値を計るためにテスタが必要です。

■ **機構部品の組み立て**

＋ドライバとナット回しを用意します。ツインモータギヤボックスの組み立てにはカッター，またはニッパが必要です。モータへの結線のために 30～40W 程度のハンダゴテとワイヤストリッパを用意します。



3-2. 部品の確認

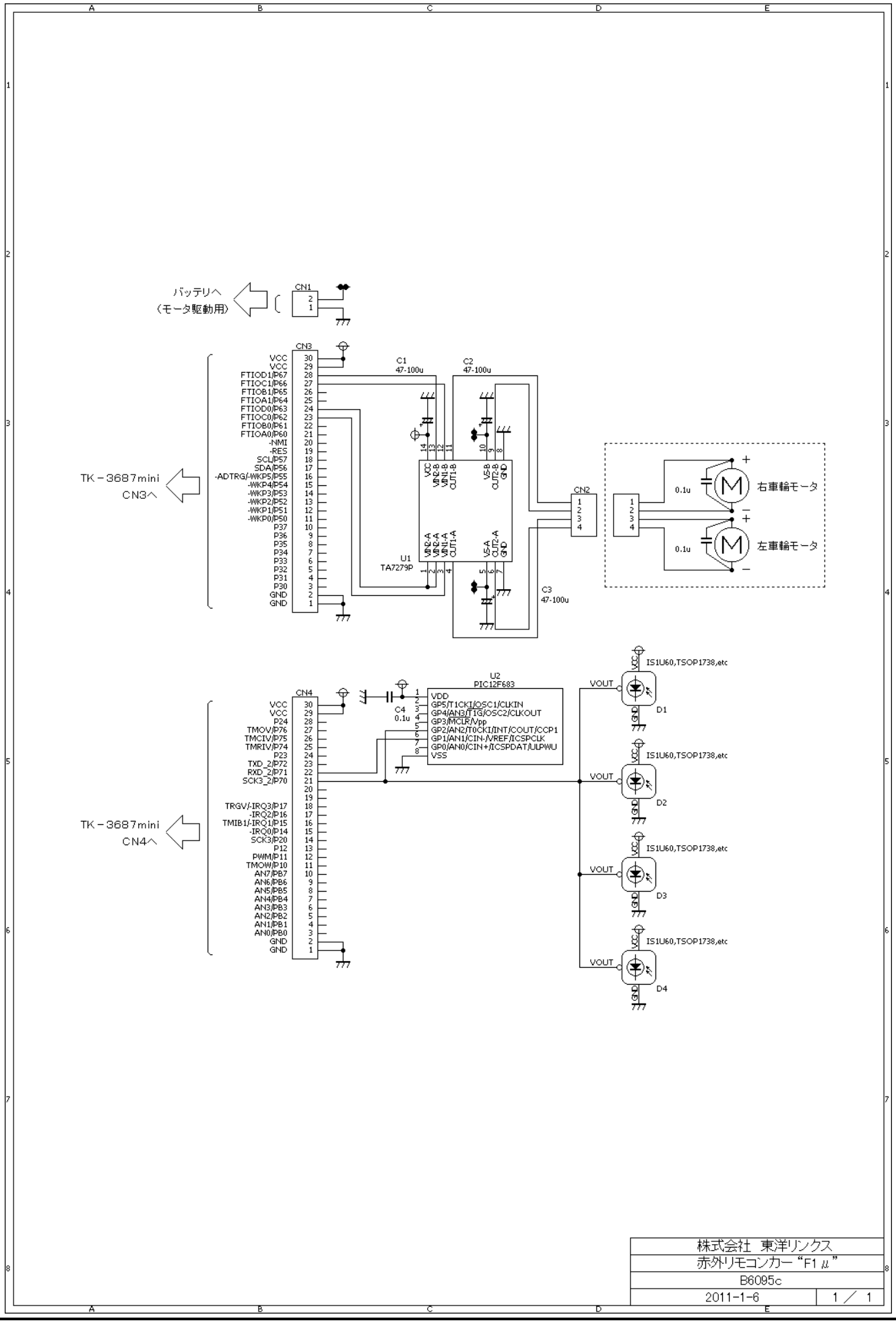
最初に，部品表と比較して部品が全てそろっているか確認しましょう。部品によっては相当品使用の場合もあります。（部品が足りないときは巻末記載の連絡先までお問い合わせください。）

赤外リモコンカー

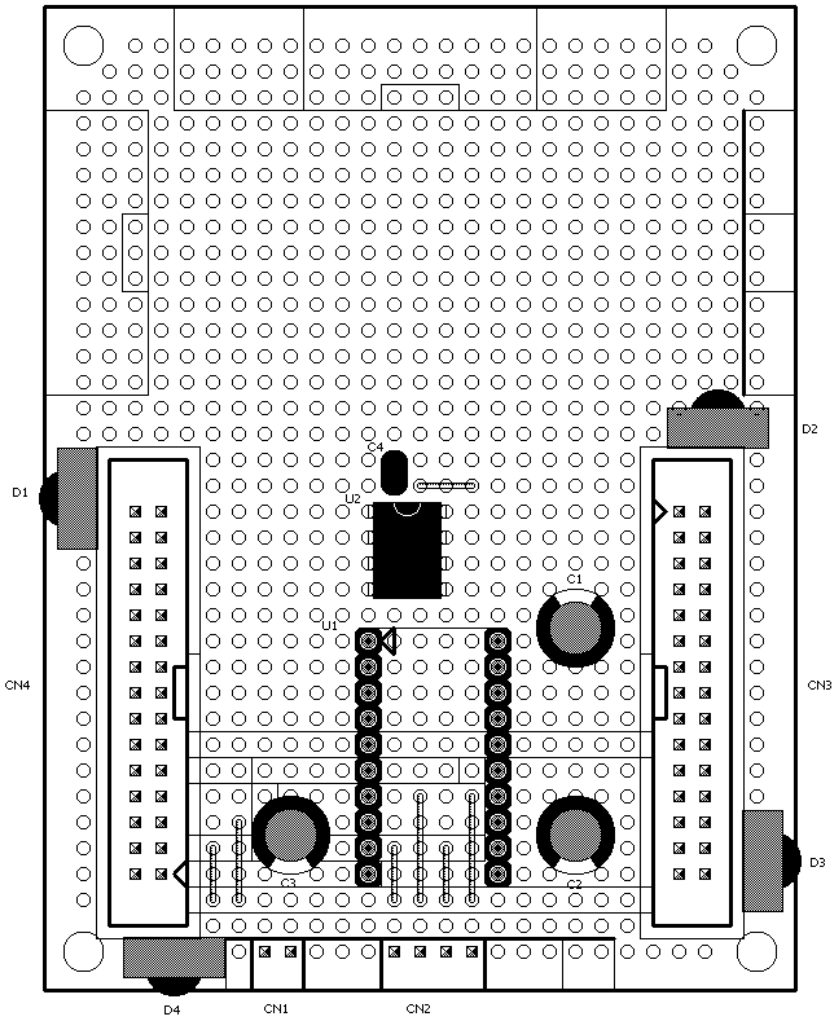
	部品番号	型名，規格	メーカー	数量	付属数量	備考
1	**■モータドライバ基板**					
2	U1&ICソケット	TA7279P	東芝	1	1	ICソケットは二つに切って使用
3	U2&ICソケット	PIC12F683	マイクロチップテクノロジ	1	1	
4	D1,2,3,4	IS1U60 TSOP1738 PL-IRM2121-A538	SHARP Vishay Telefunken PARA LIGHT	4	4	*1
5	C1～3	47～100μF/16V		3	3	
6	C4	0.1μF（セラ）		1	1	
7	CN3,4	HIF3FC-30PA-2.54DSA	HRS	2	2	*1
8	CN2	B4P-SHF-1AA	JST	1	1	
9	CN1	B2P-SHF-1AA	JST	1	1	
10	ラッピングケーブル	1m		1	1	
11	メッキ線			0	0	ハンダ面結線用 *2
12	ユニバーサル基板	B6093(95×72mm)	東洋リンクス	1	1	
13						
14	**■CPUボード**					
15	CPUボード	TK-3687mini	東洋リンクス	1	1	フラットパッケージ実装済み
16	REG1	TA48M05F(S)	東芝	1	1	
17	X1	20MHz		1	1	メインクロック
18	X2	32.768KHz		1	1	サブクロック
19	D3	1SS133-T72	ROHM	1	1	*1
20	LED1	HLMP-6300#A04	HP	1	1	*1
21	C3,19	47～100μF/16V		2	2	
22	C4,6,17,18,20	10μF/16V		5	5	
23	SW1	SKHHAK/AM/DC	ALPS	1	1	*1
24	CN1	B2P-SHF-1AA	JST	1	1	電源用
25	CN3,4	HIF3FB-30DA-2.54DSA	HRS	2	2	基板間コネクタ，ハンダ面に実装
26	CN5	D-Sub9pin		1	1	
27	JP1	2pin		1	1	ピンとソケットのセット
28						
29	**■機構部品**					
30	ノイズ吸収用コンデンサ	0.1～0.47μF		2	2	
31	モータ接続用ケーブル	4芯×16cm		1	1	
32	ネジ	3-8mm		6	6	
33	ネジ	3-45mm～3-50mm		4	4	
34	ナット	3φ		10	10	
35	スペーサ	H=20mm		8	8	
36	ユニバーサル基板	B6093(95×72mm)	東洋リンクス	1	1	ギアボックス，キャスター取付用，穴開け加工済み
37	ユニバーサル基板	B6093(95×72mm)	東洋リンクス	1	1	電池ボックス固定用
38						
39	**■その他**					
40	タイヤ	50φ		2	2	
41	キャスター			1	1	
42	ギアボックス	ツインモータギアボックス	田宮模型	1	1	
43	電池ボックス			2	2	単3×4本用，ケーブル付
44	結束バンド			1	1	電池ボックス固定用
45						

(*1)相当品を使用することがあります。
(*2)ラッピングケーブルの被覆をはがし2本をよじって使用します。また，抵抗やコンデンサの足も流用できます。

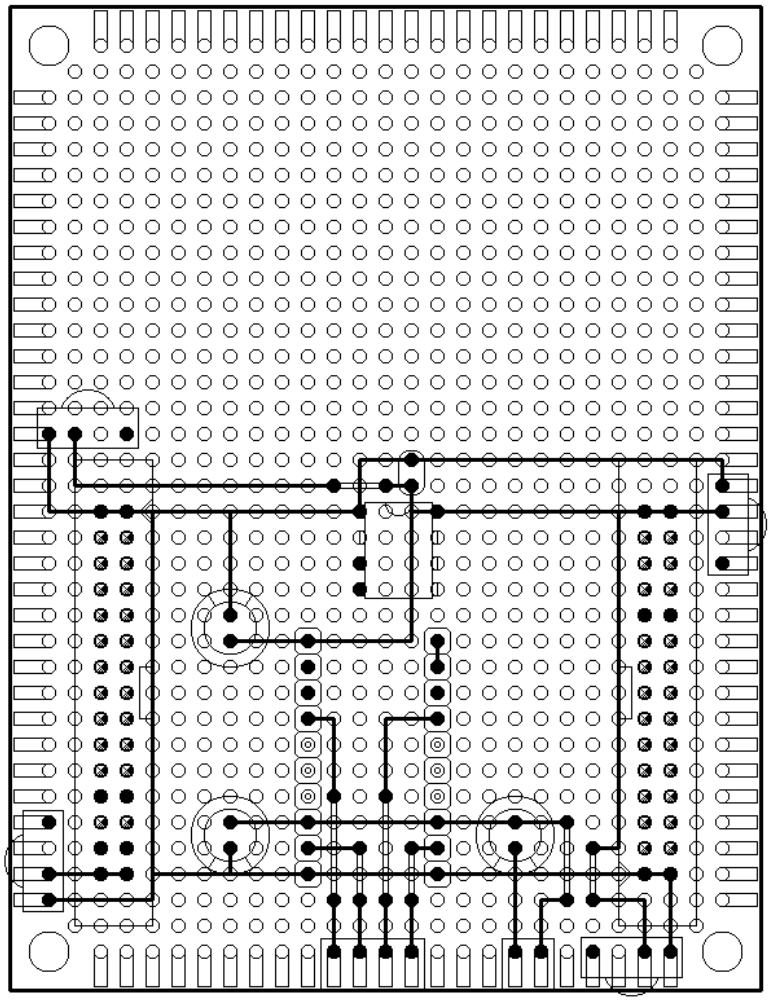
3-3. 回路図と実装図


バッテリへ
（モータ駆動用）
CN1
CN3
TK－3687mini
CN3へ
C1
47-100u
C2
47-100u
U1
TA7279P
C3
47-100u
CN2
0.1u
右車輪モータ
0.1u
左車輪モータ
CN4
TK－3687mini
CN4へ
C4
0.1u
U2
PIC12F683
VDD
GP5/T1CKI/OSC1/CLKIN
GP4/AN3/T1G/OSC2/CLKOUT
GP3/MCLR/Vpp
GP2/AN2/T0CKI/INT/COUT/CCP1
GP1/AN1/CIN-/VREF/ICSPCLK
GP0/AN0/CIN+/ICSPDAT/ULPWU
VSS
IS1U60,TSOP1738,etc
VOUT
D1
D2
D3
D4
株式会社 東洋リンクス
赤外リモコンカー“F1μ”
B6095c
2011-1-6
1 ／ 1

表面



裏面



この図の赤外リモコン受信モジュールは"TSOP1738"です。

株式会社　東洋リンクス	
赤外リモコンカー　実装図（TSOP1738版）	
B6095c	
2011-1-5	1 ／ 1

3-4. 部品の実装

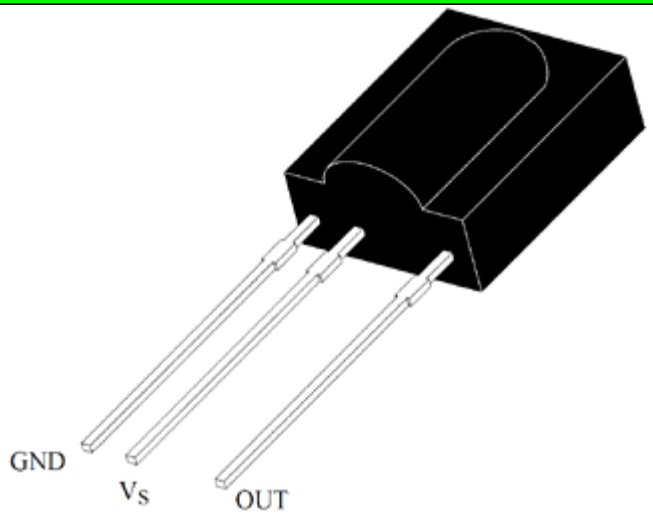
■ モータドライバ基板

最初に，実装図の表面（部品面）を見てユニバーサル基板に部品を載せます。次に，実装図の裏面（半田面）を見てメッキ線での配線を済ませてください。最後に，回路図を見ながらラッピングケーブルで残りの配線を行ないます。なお，PIC12F683 は 7 章まで使いませんので，IC ソケットのみ実装してください。

◆赤外リモコンモジュールについて◆

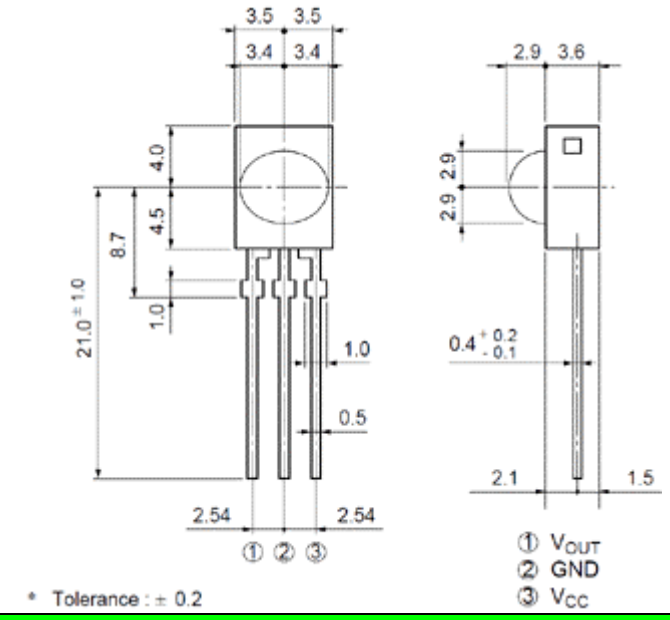
前ページの実装図は「TSOP1738」を使用した図です。次ページの写真は「IS1U60」を使用しています。入手状況により入っている部品が異なります。VCC，GND，VOUT のピン配置が異なるので組立の際はご注意ください。参考までに，下記に使用する可能性のある赤外リモコンモジュールのピン配置図を示します。

TSOP1738

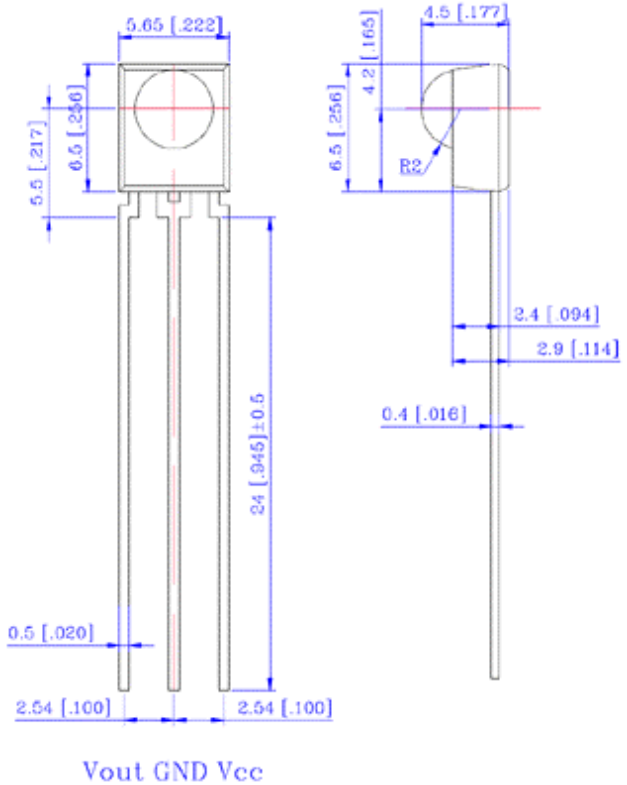


Vs＝Vcc，OUT=Vout のこと

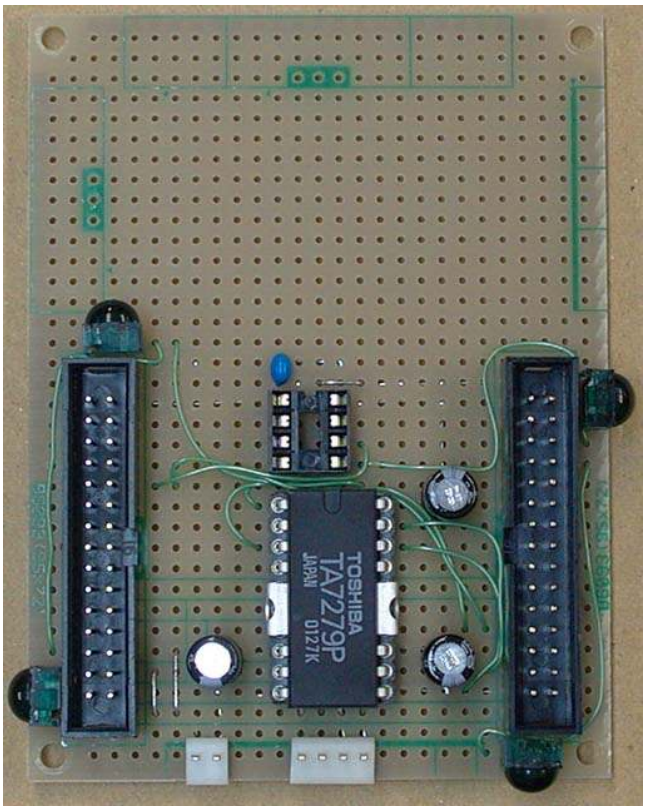
IS1U60

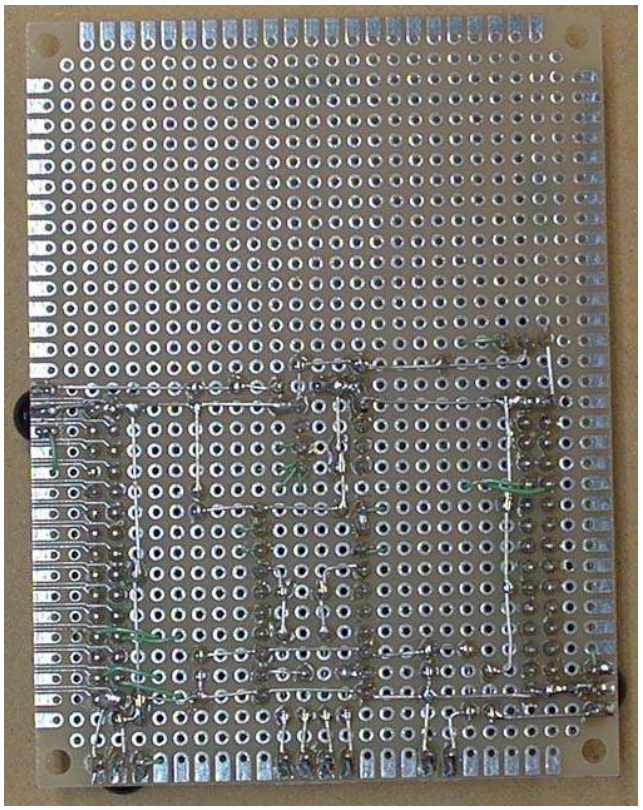


PL-IRM2121-A538



配線が終了したら，回路図どおり配線されているかもう一度確認して下さい。確認方法は，テスタで部品面の端子間の抵抗を測り，導通があるか，すなわち 0Ω か否かで判断します。また，半田付けがきちんと行なわれているか見ておきましょう。動かない原因の大部分は配線ミスと半田付け不良です。





この写真の赤外リモコンモジュールは“IS1U60”です。

■ CPU ボード

付属 CD の「¥TK-3687mini¥ﾏﾆｭｱﾙ¥」フォルダの中に「TK-3687mini 組み立て手順書」があります。このマニュアルを見ながら TK-3687mini を組立ててください。ただし，F1μ 付属の TK-3687mini の部品にはピンヘッダーは入っていません。

組み立てが終わったら，TK-3687mini にプログラムを書き込みます。付属 CD の中の **‘ir_remocon_car_01.mot’** を，FDT を使って TK-3687mini にダウンロードしてください。FDT の基本的な使い方は「TK-3687mini 組み立て手順書」に記されています。これを参考にして下さい。

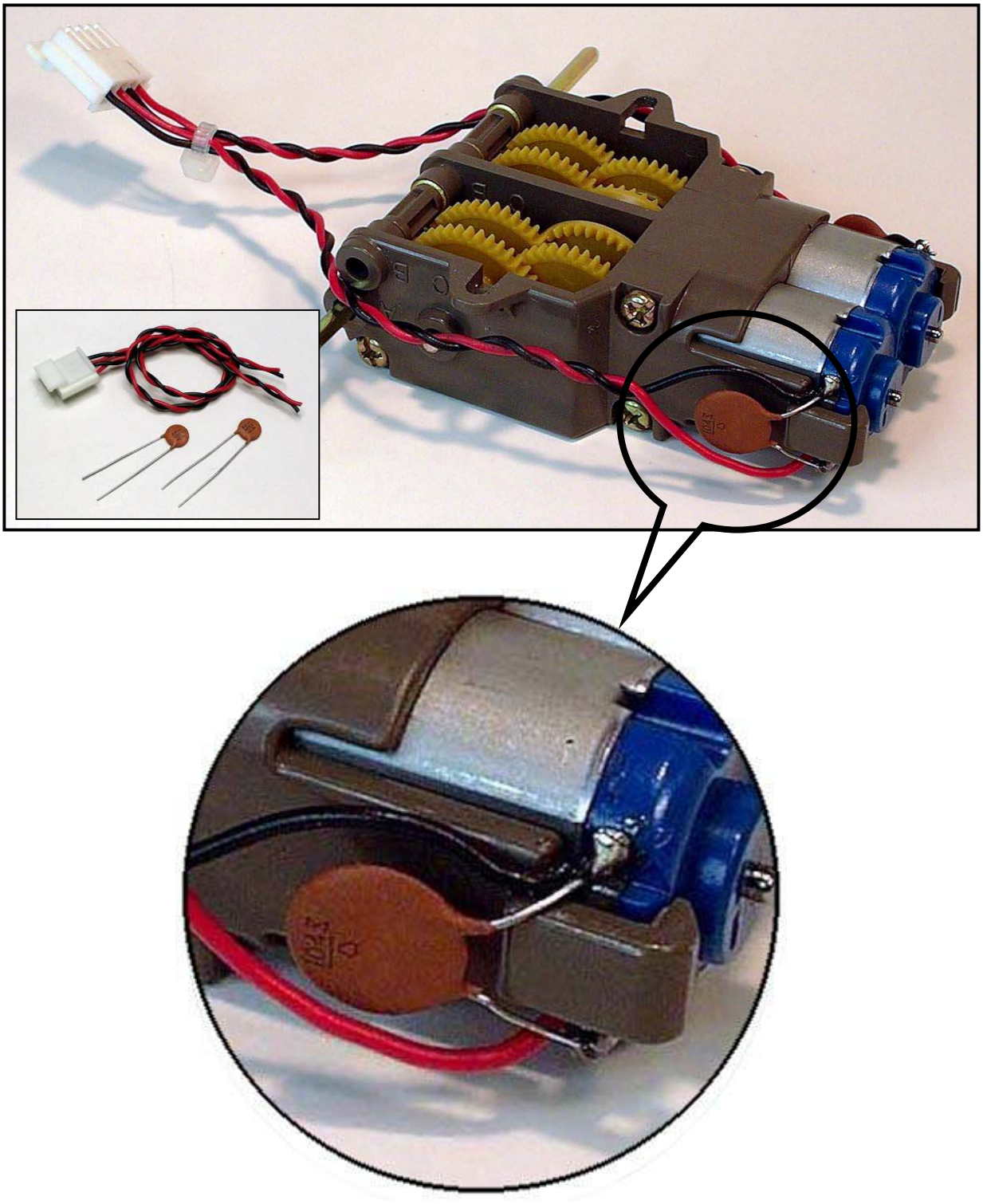
3-5. 組み立て

■ ツインモータギヤボックスの組み立て

ツインモータギヤボックス内の組み立て図を見て，**標準仕様（A タイプ）**で組立ててください。

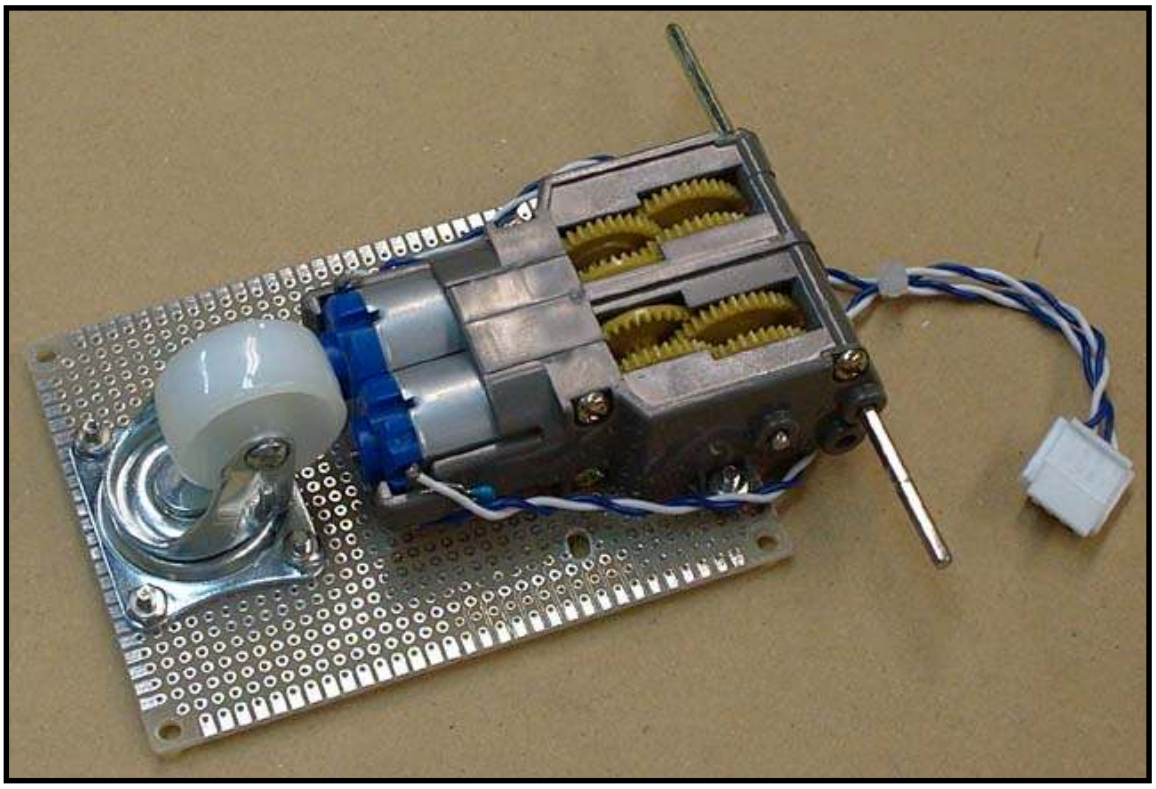
■ モータの結線

4 ピンコネクタがついているケーブルの被服をワイヤストリッパでむいてモータにハンダ付けします。コネクタの 1，2 番が右車輪用のモータに，3，4 番が左車輪用のモータにつながります。下の写真ではケーブルの色が赤黒になっていますが，製品では白青になっているものもあります。その場合は，白を赤，青を黒として結線してください。ケーブルと一緒に付属のコンデンサ（0．1μF）もハンダ付けします。コンデンサはモータの発生ノイズを低減するのに必要ですから必ず取り付けてください。



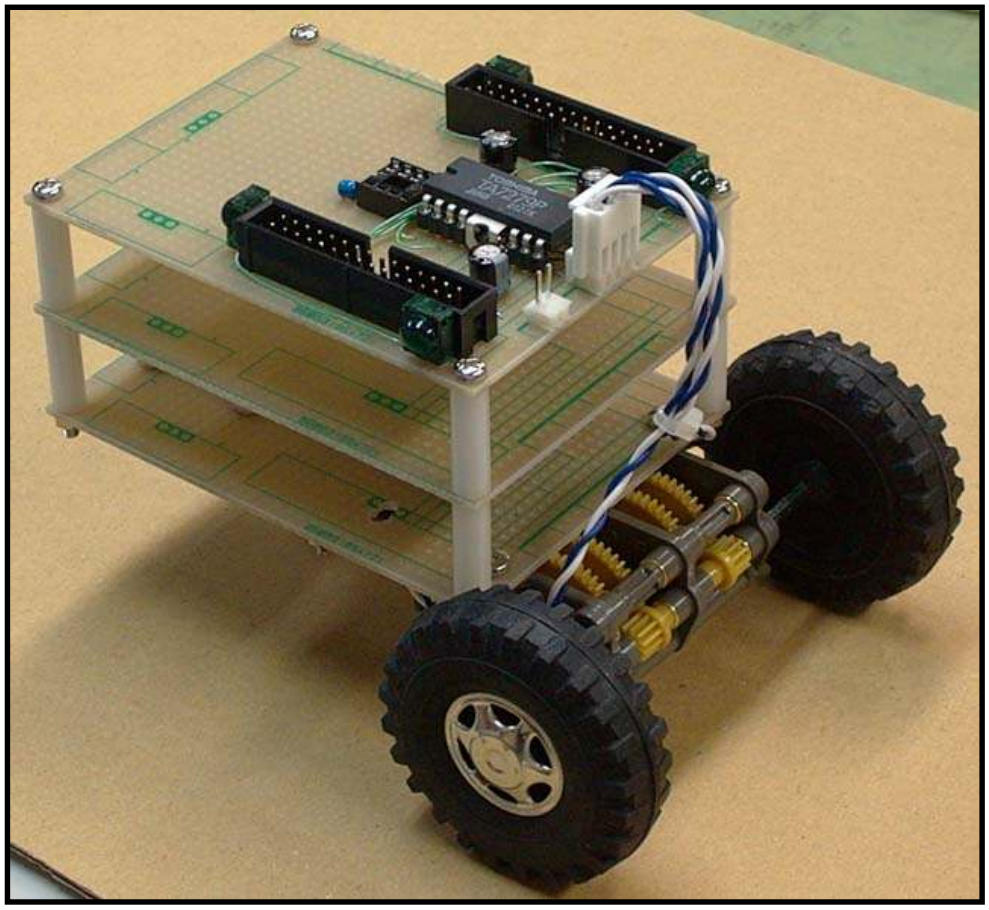
■　**ギヤボックスとキャスターの取り付け**

穴が開けてあるユニバーサル基板に，ギヤボックスとキャスターを 8mm のネジとナットで取り付けます。



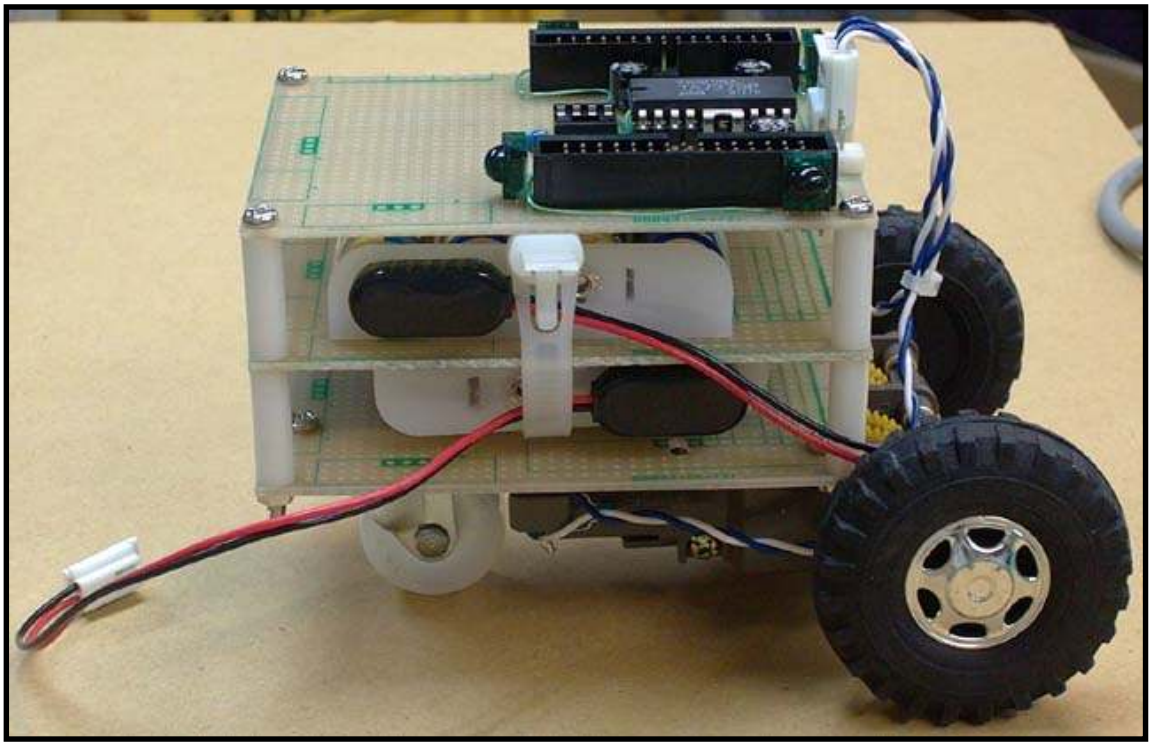
■　**ユニバーサル基板のスタッキング**

キャスターとギヤボックスを取り付けた基板，ユニバーサル基板，モータドライバ基板を 20mm のスペーサを介して写真のように 45mm のネジとナットでスタッキングします。組み立て終わったら後輪をシャフトに挿します。



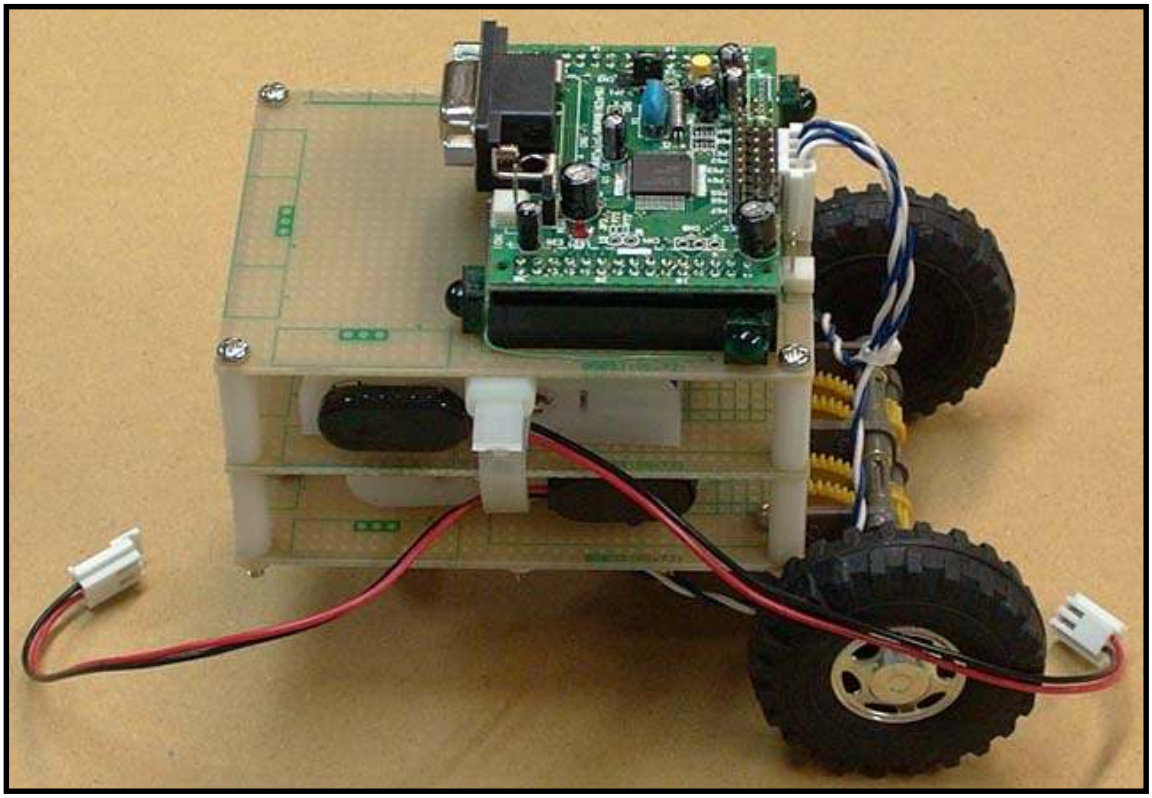
■　**電池ボックスの取り付け**

電池ボックスに電池をセットしてください。そのあと，結束バンドを使い，二つの電池ボックスで真中のユニバーサル基板をはさむような形で固定します。



■　**マイコンボードの取り付け**

マイコンボードをモータドライバ基板に取り付ければ完成です。方向を間違えないようにしてください。なお，電池が入っていますので，使うとき以外は電源ケーブルを抜いてください。



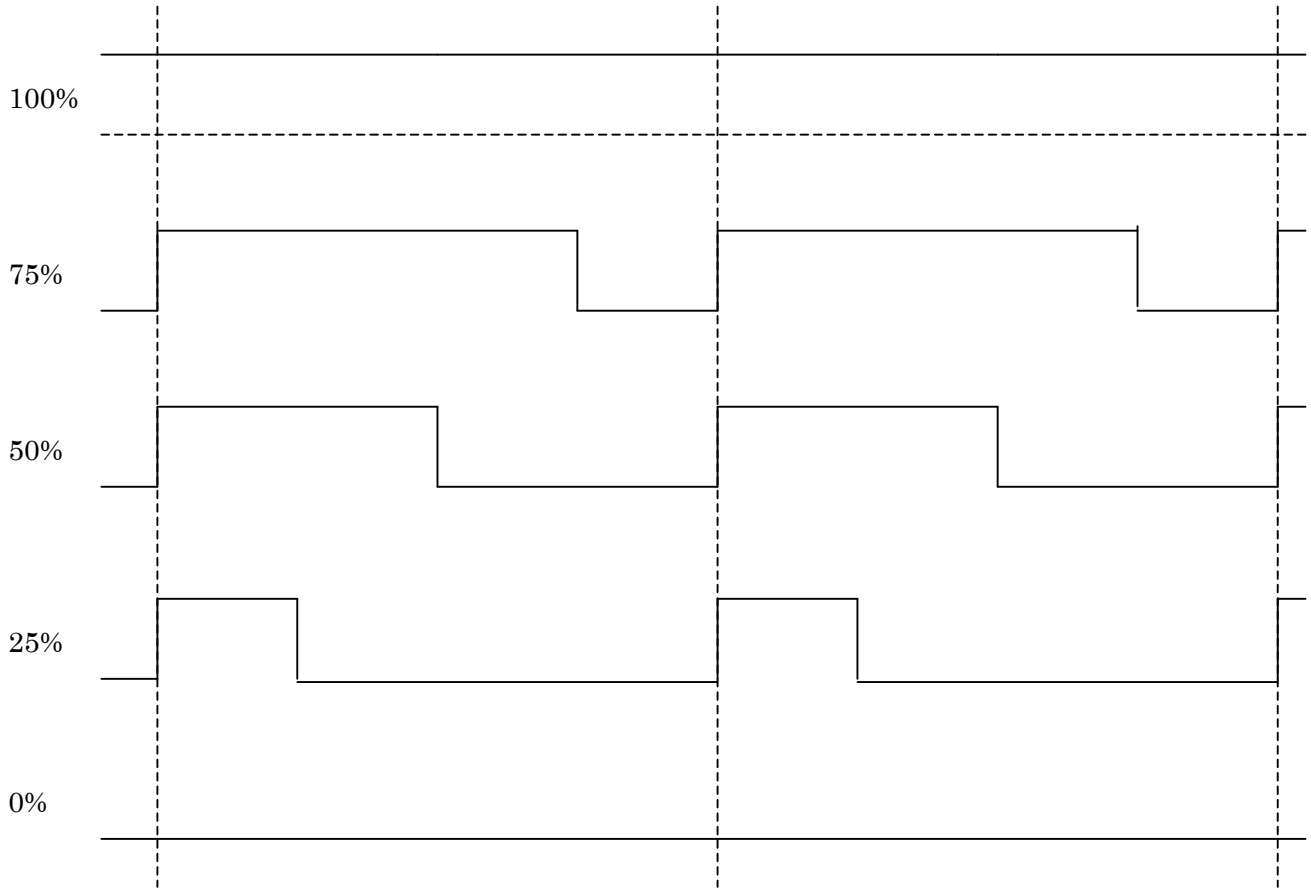
第4章 赤外リモコンカーのプログラム

4-1. モータ制御の考え方

DC モータは電圧をかけると回転し，電圧をかけないと止まります。しかし，これでは回転するか，回転しないかの 2 種類で，モータの回転数を変化させることはできません。では，どのようにして変化させているのでしょうか。

DC モータは電圧をかけたからといってその瞬間すぐに 100%で回転するわけではなく，また，電圧をかけるのをやめたからといってすぐに止まるものでもありません。徐々に回転が変化していきます。では，電圧のオン・オフをすばやく繰り返したらどうなるでしょうか。電圧が加わると，徐々にモータが回りだします。100%に向かって回転数が上がっていきますが，すぐに電圧がオフになります。すると今度は徐々にモータが止まろうとします。止まる前に再び電圧が加わるので，また回りだします。これを繰り返せば一定の回転数でモータを回すことができます。また，オン・オフの時間を変化させれば任意の回転数でまわすこともできます。

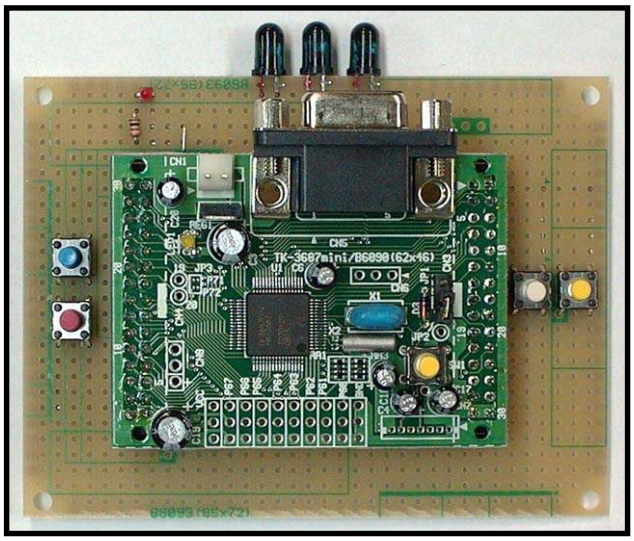
通常，オンとオフを足した時間を一定にし，オンとオフの比（デューティ）を変化させます。このような制御方法を PWM（Pulse Width Modulation：パルス幅変調）といいます。



赤外リモコンカーの場合，タイマ Z を PWM モードで使用しています。そして，左右のモータを別々に PWM で制御し，スピードと共に方向も変えています。また，プログラムでは PWM のデューティをセットします。

4-2. リモコンコード

赤外リモコン送信機はオプションなので詳細については 6 章で説明します。しかし，家にあるリモコンを利用するにしても，受信プログラムのベースはこのリモコンにします。ここでは，NEC フォーマットの受信プログラムを考えるうえで，想定している送信機の外観，どのようなコードが送られてくるかを説明します。



赤外リモコン送信機は右の写真のようなものを想定しています。(注意:タクトスイッチの色は一例です。説明の都合上，右写真の色を使って説明していますが，実際のキットでは異なる色が入っていることがありますし，すべてのスイッチの色が同じこともあります。)

リモコン送信機はいずれかのスイッチが押されたときに，カスタムコード“00”，“FF”に続いて，押されたスイッチに応じてつぎのようなコードを NEC フォーマットで送信します。

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
	固定データ				青	赤	黄	白
1	0	1	0	0	オン	オン	オン	オン
0					オフ	オフ	オフ	オフ

赤外リモコンカーは受信したコードに基づき次のように動作します。

リモコンの「青」スイッチオン	→	前進
リモコンの「赤」スイッチオン	→	後退
リモコンの「青」「黄」スイッチ同時オン	→	前進右旋回
リモコンの「青」「白」スイッチ同時オン	→	前進左旋回
リモコンの「赤」「黄」スイッチ同時オン	→	後退右旋回
リモコンの「赤」「白」スイッチ同時オン	→	後退左旋回
リモコンの「黄」スイッチオン	→	右回転
リモコンの「白」スイッチオン	→	左回転
リモコンの「青」「赤」スイッチ同時オン	→	停止
0. 3 秒以上，赤外リモコン信号を受信しない	→	停止

4-3. 赤外リモコンカーのプログラム

```
/*********************************************************************/
/*                                                                   */
/*  FILE        :ir_remocon_car_01.c                                 */
/*  DATE        :Wed, Aug 26, 2009                                   */
/*  DESCRIPTION :Main Program                                        */
/*  CPU TYPE    :H8/3687                                             */
/*                                                                   */
/*  This file is programed by TOYO-LINX Co.,Ltd. / yKikuchi          */
/*                                                                   */
/*********************************************************************/

/*********************************************************************
    プログラムの内容
*********************************************************************/
/*
2009-08-26 : プログラム開始
*/

/*********************************************************************
    リモコンスイッチの並び方と赤外線コードの対応
---------------------------------------------------------------------
        +-----+
        | SW4 |
        |     |
        +-----+                   +-----+ +-----+
                                  | SW1 | | SW2 |
                                  |     | |     |
        +-----+                   +-----+ +-----+
        | SW3 |
        |     |
        +-----+
---------------------------------------------------------------------

        +-----+-----+-----+-----+-----+-----+-----+-----+
        |bit7 |bit6 |bit5 |bit4 |bit3 |bit2 |bit1 |bit0 |
        +-----+-----+-----+-----+-----+-----+-----+-----+
        |  0  |  1  |  0  |  0  | SW4 | SW3 | SW2 | SW1 |
        |     |     |     |     |     |     |     |     |
        +-----+-----+-----+-----+-----+-----+-----+-----+
                                    0=Off / 1=On
*********************************************************************/

/*********************************************************************
    インクルードファイル
*********************************************************************/
#include<machine.h>      //H8特有の命令を使う
#include"iodefine.h"     //内蔵I/Oのラベル定義
#include"binary.h"       //Cで2進数を使うための定義

/*********************************************************************
    定数の定義（直接指定）
*********************************************************************/
#define     OK          0       //戻り値
#define     NG          -1      //戻り値
```

```
//リモコン用,PWMのパーセント ----------------------------------------
//  モータに電力を供給する時間率で指定
#define     TL3LR       20      //左旋回,急,左モータ,%
#define     TL3RR       50      //左旋回,急,右モータ,%
#define     TL2LR       30      //左旋回,中,左モータ,%
#define     TL2RR       50      //左旋回,中,右モータ,%
#define     TL1LR       40      //左旋回,緩,左モータ,%
#define     TL1RR       50      //左旋回,緩,右モータ,%

#define     CW4LR       50      //直進,4速,左モータ,%
#define     CW4RR       50      //直進,4速,右モータ,%
#define     CW3LR       40      //直進,3速,左モータ,%
#define     CW3RR       40      //直進,3速,右モータ,%
#define     CW2LR       30      //直進,2速,左モータ,%
#define     CW2RR       30      //直進,2速,右モータ,%
#define     CW1LR       20      //直進,1速,左モータ,%
#define     CW1RR       20      //直進,1速,右モータ,%

#define     TR1LR       50      //右旋回,緩,左モータ,%
#define     TR1RR       40      //右旋回,緩,右モータ,%
#define     TR2LR       50      //右旋回,中,左モータ,%
#define     TR2RR       30      //右旋回,中,右モータ,%
#define     TR3LR       50      //右旋回,急,左モータ,%
#define     TR3RR       20      //右旋回,急,右モータ,%

//ソフトウェアタイマ --------------------------------------------------
#define     T0Const     300     //T0(300ms)  通信が途絶えてから停止するまでの時間（リモコンモード）
#define     T1Const     0       //T1(ms)
#define     T2Const     0       //T2(ms)
#define     T3Const     0       //T3(ms)
#define     T4Const     0       //T4(ms)
#define     T5Const     0       //T5(ms)
#define     T6Const     0       //T6(ms)
#define     T7Const     0       //T7(ms)

/*************************************************************************
    定数エリアの定義(ROM)
*************************************************************************/

/*************************************************************************
    グローバル変数の定義とイニシャライズ(RAM)
*************************************************************************/
// 走行に関係した変数 ------------------------------------------------
//左旋回,急,左モータ
unsigned int TurnLeft3L   =   0xffff - ((unsigned long)0xffff * TL3LR / 100);
//左旋回,急,右モータ
unsigned int TurnLeft3R   =   0xffff - ((unsigned long)0xffff * TL3RR / 100);
//左旋回,中,左モータ
unsigned int TurnLeft2L   =   0xffff - ((unsigned long)0xffff * TL2LR / 100);
//左旋回,中,右モータ
unsigned int TurnLeft2R   =   0xffff - ((unsigned long)0xffff * TL2RR/ 100);
//左旋回,緩,左モータ
unsigned int TurnLeft1L   =   0xffff - ((unsigned long)0xffff * TL1LR/ 100);
//左旋回,緩,右モータ
unsigned int TurnLeft1R   =   0xffff - ((unsigned long)0xffff * TL1RR/ 100);

//直進,4速,左モータ
```

```
unsigned int Cw4L        =   0xffff - ((unsigned long)0xffff * CW4LR/ 100);
//直進,4速,右モータ
unsigned int Cw4R        =   0xffff - ((unsigned long)0xffff * CW4RR/ 100);
//直進,3速,左モータ
unsigned int Cw3L        =   0xffff - ((unsigned long)0xffff * CW3LR/ 100);
//直進,3速,右モータ
unsigned int Cw3R        =   0xffff - ((unsigned long)0xffff * CW3RR/ 100);
//直進,2速,左モータ
unsigned int Cw2L        =   0xffff - ((unsigned long)0xffff * CW2LR/ 100);
//直進,2速,右モータ
unsigned int Cw2R        =   0xffff - ((unsigned long)0xffff * CW2RR/ 100);
//直進,1速,左モータ
unsigned int Cw1L        =   0xffff - ((unsigned long)0xffff * CW1LR/ 100);
//直進,1速,右モータ
unsigned int Cw1R        =   0xffff - ((unsigned long)0xffff * CW1RR/ 100);

//右旋回,緩,左モータ
unsigned int TurnRight1L =   0xffff - ((unsigned long)0xffff * TR1LR/ 100);
//右旋回,緩,右モータ
unsigned int TurnRight1R =   0xffff - ((unsigned long)0xffff * TR1RR/ 100);
//右旋回,中,左モータ
unsigned int TurnRight2L =   0xffff - ((unsigned long)0xffff * TR2LR/ 100);
//右旋回,中,右モータ
unsigned int TurnRight2R =   0xffff - ((unsigned long)0xffff * TR2RR/ 100);
//右旋回,急,左モータ
unsigned int TurnRight3L =   0xffff - ((unsigned long)0xffff * TR3LR/ 100);
//右旋回,急,右モータ
unsigned int TurnRight3R =   0xffff - ((unsigned long)0xffff * TR3RR/ 100);

// ソフトウェアタイマに関係した変数 ----------------------------------
struct SoftTimer{                    //ソフトウェアタイマの構造体タグ
    unsigned char    Status;         //タイマステータス
                                     // 0:停止中(タイマ未使用)
                                     // 1:スタート指令
                                     // 2:カウント中
                                     // 3:カウント終了
    unsigned long    Count;          //タイマカウンタ
};
struct SoftTimer TimT0;              //T0タイマ
struct SoftTimer TimT1;              //T1タイマ
struct SoftTimer TimT2;              //T2タイマ
struct SoftTimer TimT3;              //T3タイマ
struct SoftTimer TimT4;              //T4タイマ
struct SoftTimer TimT5;              //T5タイマ
struct SoftTimer TimT6;              //T6タイマ
struct SoftTimer TimT7;              //T7タイマ

unsigned long    T0 = T0Const;       //T0タイマカウンタ初期値
unsigned long    T1 = T1Const;       //T1タイマカウンタ初期値
unsigned long    T2 = T2Const;       //T2タイマカウンタ初期値
unsigned long    T3 = T3Const;       //T3タイマカウンタ初期値
unsigned long    T4 = T4Const;       //T4タイマカウンタ初期値
unsigned long    T5 = T5Const;       //T5タイマカウンタ初期値
unsigned long    T6 = T6Const;       //T6タイマカウンタ初期値
unsigned long    T7 = T7Const;       //T7タイマカウンタ初期値

/************************************************************************
    関数の定義
```

```
*****************************************************************************/
void        ccw1(void);             //後退,1速
void        ccw2(void);             //後退,2速
void        ccw3(void);             //後退,3速
void        ccw4(void);             //後退,4速
void        cw1(void);              //前進,1速
void        cw2(void);              //前進,2速
void        cw3(void);              //前進,3速
void        cw4(void);              //前進,4速
void        init_io(void);          //IOポートイニシャライズ
void        init_tmb1(void);        //タイマB1イニシャライズ
void        init_tmz(void);         //タイマZイニシャライズ
void        intprog_tmb1(void);     //タイマB1割込み
void        main(void);             //メインプログラム
void        stop(void);             //停止
void        turn_left_1(void);      //左旋回（大）
void        turn_left_2(void);      //左旋回（中）
void        turn_left_3(void);      //左旋回（小）
void        turn_right_1(void);     //右旋回（大）
void        turn_right_2(void);     //右旋回（中）
void        turn_right_3(void);     //右旋回（小）

void        run(void);              //走行制御

void        rev_turn_left_1(void);  //後退左旋回（大）
void        rev_turn_left_2(void);  //後退左旋回（中）
void        rev_turn_left_3(void);  //後退左旋回（小）
void        rev_turn_right_1(void); //後退右旋回（大）
void        rev_turn_right_2(void); //後退右旋回（中）
void        rev_turn_right_3(void); //後退右旋回（小）

void        spin_right_1(void);     //右回転-1
void        spin_right_2(void);     //右回転-2
void        spin_right_3(void);     //右回転-3
void        spin_right_4(void);     //右回転-4
void        spin_left_1(void);      //左回転-1
void        spin_left_2(void);      //左回転-2
void        spin_left_3(void);      //左回転-3
void        spin_left_4(void);      //左回転-4

void        init_soft_timer(void);
void        dec_soft_timer(struct SoftTimer *,unsigned long);

unsigned int IR_rcv(void);          //赤外線受信

/*****************************************************************************
    メインプログラム
*****************************************************************************/
void main(void)
{
    // イニシャライズ ------------------------------------------------
    init_io();
    init_soft_timer();
    init_tmb1();
    init_tmz();

    // メインループ --------------------------------------------------
    while(1){
```

```
        run();
    }
}

/*****************************************************************************
    走行制御
*****************************************************************************/
void run(void)
{
    unsigned int ans;    //赤外線受信ルーチンの戻り値
    unsigned char dt;    //リモコン受信コマンド

    ans = IR_rcv();

    if ((ans & 0xff00)==0x0000){
        TimT0.Status = 0;    //タイマストップ
        dt = (unsigned char)(ans & 0x00ff);
        switch(dt){
            case _01001000B: //前進
                cw4();
                break;
            case _01001010B: //前進&右
                turn_right_3();
                break;
            case _01001001B: //前進&左
                turn_left_3();
                break;
            case _01000100B: //後退
                ccw4();
                break;
            case _01000110B: //後退&右
                rev_turn_right_3();
                break;
            case _01000101B: //後退&左
                rev_turn_left_3();
                break;
            case _01000010B: //右回転
                spin_right_3();
                break;
            case _01000001B: //左回転
                spin_left_3();
                break;
            case _01001100B: //前後同時オン→停止
                stop();
                break;
        }
        TimT0.Status = 1;    //タイマスタート
    }

    if (TimT0.Status==3){    //通信が途絶えた
        TimT0.Status = 0;    //タイマストップ
        stop(); //停止
    }
}

/*****************************************************************************
    モータスピードセット
*****************************************************************************/
```

```
// 前進,1速 ------------------------------------------------------------
void cw1(void)
{
    TZ0.GRC = Cw1L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = Cw1R;
    TZ1.GRD = 0xffff;
}

// 前進,2速 ------------------------------------------------------------
void cw2(void)
{
    TZ0.GRC = Cw2L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = Cw2R;
    TZ1.GRD = 0xffff;
}

// 前進,3速 ------------------------------------------------------------
void cw3(void)
{
    TZ0.GRC = Cw3L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = Cw3R;
    TZ1.GRD = 0xffff;
}

// 前進,4速 ------------------------------------------------------------
void cw4(void)
{
    TZ0.GRC = Cw4L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = Cw4R;
    TZ1.GRD = 0xffff;
}

// 後退,1速 ------------------------------------------------------------
void ccw1(void)
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = Cw1L;
    TZ1.GRC = 0xffff;
    TZ1.GRD = Cw1R;
}

// 後退,2速 ------------------------------------------------------------
void ccw2(void)
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = Cw2L;
    TZ1.GRC = 0xffff;
    TZ1.GRD = Cw2R;
}

// 後退,3速 ------------------------------------------------------------
void ccw3(void)
{
    TZ0.GRC = 0xffff;
```

```
    TZ0.GRD = Cw3L;
    TZ1.GRC = 0xffff;
    TZ1.GRD = Cw3R;
}

// 後退,4速 ----------------------------------------------------------
void ccw4(void)
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = Cw4L;
    TZ1.GRC = 0xffff;
    TZ1.GRD = Cw4R;
}

// 右旋回（小） ----------------------------------------------------------
void turn_right_1(void)
{
    TZ0.GRC = TurnRight1L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = TurnRight1R;
    TZ1.GRD = 0xffff;
}

// 右旋回（中） ----------------------------------------------------------
void turn_right_2(void)
{
    TZ0.GRC = TurnRight2L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = TurnRight2R;
    TZ1.GRD = 0xffff;
}

// 右旋回（大） ----------------------------------------------------------
void turn_right_3(void)
{
    TZ0.GRC = TurnRight3L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = TurnRight3R;
    TZ1.GRD = 0xffff;
}

// 左旋回（小） ----------------------------------------------------------
void turn_left_1(void)
{
    TZ0.GRC = TurnLeft1L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = TurnLeft1R;
    TZ1.GRD = 0xffff;
}

// 左旋回（中） ----------------------------------------------------------
void turn_left_2(void)
{
    TZ0.GRC = TurnLeft2L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = TurnLeft2R;
    TZ1.GRD = 0xffff;
}
```

```
// 左旋回（大） ----------------------------------------------------
void turn_left_3(void)
{
    TZ0.GRC = TurnLeft3L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = TurnLeft3R;
    TZ1.GRD = 0xffff;
}

// 後退右旋回（小） ----------------------------------------------------
void rev_turn_right_1(void)
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = TurnRight1L;
    TZ1.GRC = 0xffff;
    TZ1.GRD = TurnRight1R;
}

// 後退右旋回（中） ----------------------------------------------------
void rev_turn_right_2(void)
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = TurnRight2L;
    TZ1.GRC = 0xffff;
    TZ1.GRD = TurnRight2R;
}

// 後退右旋回（大） ----------------------------------------------------
void rev_turn_right_3(void)
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = TurnRight3L;
    TZ1.GRC = 0xffff;
    TZ1.GRD = TurnRight3R;
}

// 後退左旋回（小） ----------------------------------------------------
void rev_turn_left_1(void)
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = TurnLeft1L;
    TZ1.GRC = 0xffff;
    TZ1.GRD = TurnLeft1R;
}

// 後退左旋回（中） ----------------------------------------------------
void rev_turn_left_2(void)
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = TurnLeft2L;
    TZ1.GRC = 0xffff;
    TZ1.GRD = TurnLeft2R;
}

// 後退左旋回（大） ----------------------------------------------------
void rev_turn_left_3(void)
{
```

```
    TZ0.GRC = 0xffff;
    TZ0.GRD = TurnLeft3L;
    TZ1.GRC = 0xffff;
    TZ1.GRD = TurnLeft3R;
}

// 停止 ----------------------------------------------------------------
void stop()
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = 0xffff;
    TZ1.GRC = 0xffff;
    TZ1.GRD = 0xffff;
}

// 右回転-1 ------------------------------------------------------------
void spin_right_1(void)
{
    TZ0.GRC = Cw1L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = 0xffff;
    TZ1.GRD = Cw1R;
}

// 右回転-2 ------------------------------------------------------------
void spin_right_2(void)
{
    TZ0.GRC = Cw2L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = 0xffff;
    TZ1.GRD = Cw2R;
}

// 右回転-3 ------------------------------------------------------------
void spin_right_3(void)
{
    TZ0.GRC = Cw3L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = 0xffff;
    TZ1.GRD = Cw3R;
}

// 右回転-4 ------------------------------------------------------------
void spin_right_4(void)
{
    TZ0.GRC = Cw4L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = 0xffff;
    TZ1.GRD = Cw4R;
}

// 左回転-1 ------------------------------------------------------------
void spin_left_1(void)
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = Cw1L;
    TZ1.GRC = Cw1R;
    TZ1.GRD = 0xffff;
```

```
}

// 左回転-2 ------------------------------------------------------------
void spin_left_2(void)
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = Cw2L;
    TZ1.GRC = Cw2R;
    TZ1.GRD = 0xffff;
}

// 左回転-3 ------------------------------------------------------------
void spin_left_3(void)
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = Cw3L;
    TZ1.GRC = Cw3R;
    TZ1.GRD = 0xffff;
}

// 左回転-4 ------------------------------------------------------------
void spin_left_4(void)
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = Cw4L;
    TZ1.GRC = Cw4R;
    TZ1.GRD = 0xffff;
}

/*************************************************************************
    I/Oポート イニシャライズ
*************************************************************************/
void init_io(void)
{
    IO.PCR6         =   _00000000B; //ポート6,P60-67入力

    IO.PCR7         =   _00000000B; //ポート7,P77-70入力
}

/*************************************************************************
    タイマZ イニシャライズ
*************************************************************************/
void init_tmz(void)
{
    TZ.TSTR.BYTE    =   0x00;   //TCNT0,1 停止

    TZ0.TCR.BYTE    =   0x20;   //GRAのコンペアマッチでTCNT=0,φ/1
    TZ1.TCR.BYTE    =   0x20;   //GRAのコンペアマッチでTCNT=0,φ/1

    TZ.TPMR.BYTE    =   0x66;   //FTIOC0,FTIOD0,FTIOC1,FTIOD1 PWMモード
    TZ.TOCR.BYTE    =   0x00;   //初期出力,All Low

    TZ0.POCR.BYTE   =   0x06;   //FTIOC0,FTIOD0 出力ハイアクティブ
    TZ1.POCR.BYTE   =   0x06;   //FTIOC1,FTIOD1 出力ハイアクティブ

    TZ0.GRA         =   0xfffe; //PWM 周期(65534*(1/20MHz)=3.2767ms)
    TZ1.GRA         =   0xfffe; //PWM 周期(65534*(1/20MHz)=3.2767ms)
    TZ0.GRC         =   0xffff; //FTIOC0=Low
```

```
    TZ1.GRC          =   0xffff; //FTIOC1=Low
    TZ0.GRD          =   0xffff; //FTIOD0=Low
    TZ1.GRD          =   0xffff; //FTIOD1=Low

    TZ.TOER.BYTE     =   0x33;   //FTIOC0,DTIOD0.FTIOC1,FTIOD1 出力許可

    TZ0.TCNT         =   0x0000; //TCNT0=0
    TZ1.TCNT         =   0x0000; //TCNT1=0
    TZ.TSTR.BYTE     =   0x03;   //TCNT0 カウントスタート
}

/************************************************************************
    タイマB1 イニシャライズ
************************************************************************/
void init_tmb1(void)
{
    TB1.TMB1.BYTE    =   0xf9;   //オートリロード, 内部クロックφ/2048
    TB1.TLB1         =   0-97;   //周期=10ms(100Hz)
    IRR2.BIT.IRRTB1  =   0;      //タイマB1割込み要求フラグ クリア
    IENR2.BIT.IENTB1 =   1;      //タイマB1割込み要求イネーブル
}

/************************************************************************
    タイマB1 割込み(10ms)
************************************************************************/
#pragma regsave (intprog_tmb1)
void intprog_tmb1(void)
{
    //タイマB1割込み要求フラグ クリア
    IRR2.BIT.IRRTB1 = 0;

    //ソフトウェアタイマ T0
    if (TimT0.Status==1 || TimT0.Status==2){
        dec_soft_timer(&TimT0,T0/10);
    }
    //ソフトウェアタイマ T1
    if (TimT1.Status==1 || TimT1.Status==2){
        dec_soft_timer(&TimT1,T1/10);
    }
    //ソフトウェアタイマ T2
    if (TimT2.Status==1 || TimT2.Status==2){
        dec_soft_timer(&TimT2,T2/10);
    }
    //ソフトウェアタイマ T3
    if (TimT3.Status==1 || TimT3.Status==2){
        dec_soft_timer(&TimT3,T3/10);
    }
    //ソフトウェアタイマ T4
    if (TimT4.Status==1 || TimT4.Status==2){
        dec_soft_timer(&TimT4,T4/10);
    }
    //ソフトウェアタイマ T5
    if (TimT5.Status==1 || TimT5.Status==2){
        dec_soft_timer(&TimT5,T5/10);
    }
    //ソフトウェアタイマ T6
    if (TimT6.Status==1 || TimT6.Status==2){
        dec_soft_timer(&TimT6,T6/10);
```

```
    }
    //ソフトウェアタイマ T7
    if (TimT7.Status==1 || TimT7.Status==2){
        dec_soft_timer(&TimT7,T7/10);
    }
}

/************************************************************************
    ソフトウェアタイマのデクリメント
------------------------------------------------------------------------
    引数     *pst     ソフトウェアタイマ構造体のポインタ
             initial      タイマカウンタの初期値
************************************************************************/
void dec_soft_timer(struct SoftTimer *pst,unsigned long initial)
{
    if (pst->Status==1){              //タイマスタート指令
        pst->Status  =   2;           //カウント中セット
        pst->Count   =   initial;     //タイマカウンタ初期化
    }
    pst->Count--;    //カウンタ-1
    if (pst->Count==0)                //カウンタが0になった
        pst->Status  =   3;           //カウント終了セット
}

/************************************************************************
    ソフトウェアタイマのイニシャライズ
************************************************************************/
void init_soft_timer(void)
{
    TimT0.Status = 0; TimT1.Status = 0; TimT2.Status = 0; TimT3.Status = 0;
    TimT4.Status = 0; TimT5.Status = 0; TimT6.Status = 0; TimT7.Status = 0;
}
```

このプログラムは割込みを使っているので,「intprg. c」を追加・修正します。

```
/************************************************************************/
/*                                                                      */
/*  FILE        :intprg.c                                               */
/*  DATE        :Wed, Aug 26, 2009                                      */
/*  DESCRIPTION :Interrupt Program                                      */
/*  CPU TYPE    :H8/3687                                                */
/*                                                                      */
/*  This file is generated by Renesas Project Generator (Ver.4.9).      */
/*                                                                      */
/************************************************************************/



#include <machine.h>

extern void intprog_tmb1(void);

#pragma section IntPRG
//  vector 1 Reserved

//  vector 2 Reserved
```

```
//  vector 3 Reserved

//  vector 4 Reserved

//  vector 5 Reserved

//  vector 6 Reserved

//  vector 7 NMI
__interrupt(vect=7) void INT_NMI(void) {/* sleep(); */}
//  vector 8 TRAP #0
__interrupt(vect=8) void INT_TRAP0(void) {/* sleep(); */}
//  vector 9 TRAP #1
__interrupt(vect=9) void INT_TRAP1(void) {/* sleep(); */}
//  vector 10 TRAP #2
__interrupt(vect=10) void INT_TRAP2(void) {/* sleep(); */}
//  vector 11 TRAP #3
__interrupt(vect=11) void INT_TRAP3(void) {/* sleep(); */}
//  vector 12 Address break
__interrupt(vect=12) void INT_ABRK(void) {/* sleep(); */}
//  vector 13 SLEEP
__interrupt(vect=13) void INT_SLEEP(void) {/* sleep(); */}
//  vector 14 IRQ0
__interrupt(vect=14) void INT_IRQ0(void) {/* sleep(); */}
//  vector 15 IRQ1
__interrupt(vect=15) void INT_IRQ1(void) {/* sleep(); */}
//  vector 16 IRQ2
__interrupt(vect=16) void INT_IRQ2(void) {/* sleep(); */}
//  vector 17 IRQ3
__interrupt(vect=17) void INT_IRQ3(void) {/* sleep(); */}
//  vector 18 WKP
__interrupt(vect=18) void INT_WKP(void) {/* sleep(); */}
//  vector 19 RTC
__interrupt(vect=19) void INT_RTC(void) {/* sleep(); */}
//  vector 20 Reserved

//  vector 21 Reserved

//  vector 22 Timer V
__interrupt(vect=22) void INT_TimerV(void) {/* sleep(); */}
//  vector 23 SCI3
__interrupt(vect=23) void INT_SCI3(void) {/* sleep(); */}
//  vector 24 IIC2
__interrupt(vect=24) void INT_IIC2(void) {/* sleep(); */}
//  vector 25 ADI
__interrupt(vect=25) void INT_ADI(void) {/* sleep(); */}
//  vector 26 Timer Z0
__interrupt(vect=26) void INT_TimerZ0(void) {/* sleep(); */}
//  vector 27 Timer Z1
__interrupt(vect=27) void INT_TimerZ1(void) {/* sleep(); */}
//  vector 28 Reserved

//  vector 29 Timer B1
__interrupt(vect=29) void INT_TimerB1(void) {intprog_tmb1();}
//  vector 30 Reserved

//  vector 31 Reserved
```

```
//  vector 32 SCI3_2
__interrupt(vect=32) void INT_SCI3_2(void) {/* sleep(); */}
```

赤外リモコン受信ルーチンはアセンブラで作りました。下記にソースリストを掲載しますが，アセンブラのプログラムはフローチャートにすると，処理の内容をより理解しやすくなります（プログラムの学習にも最適です）。がんばってフローチャートに直してみてください。ちなみに実際のプログラムのときは，フローチャートを書いてからソースリストにコーディングします。

```
;-----------------------------------------------------------------------
;                                                                      |
;   FILE        :ir_rcv.src                                            |
;   DATE        :Tue, Aug 25, 2009                                     |
;   DESCRIPTION :Sub Program                                           |
;   CPU TYPE    :H8/3687                                               |
;                                                                      |
;   This file is programed by TOYO-LINX Co.,Ltd. / yKikuchi            |
;                                                                      |
;-----------------------------------------------------------------------

    .export     _IR_rcv          ;Cからコールされる,"IR_rcv();"
    .section P,CODE,ALIGN=2

;************************************************************************
;   定数定義
;************************************************************************
PDR7    .equh'FFDA           ;ﾎﾟｰﾄﾃﾞｰﾀﾚｼﾞｽﾀ 7
PCR7    .equh'FFEA           ;ﾎﾟｰﾄｺﾝﾄﾛｰﾙﾚｼﾞｽﾀ 7

IR_port .equPDR7             ;赤外線受信ポート
IR_bit  .equb'00000001       ;赤外線受信ビット

;************************************************************************
;   赤外線リモコン信号受信（NECフォーマット準拠）
;------------------------------------------------------------------------
;   戻り値
;   R0 = 00xx -> 受信OK,xx=受信コマンド
;   R0 = FFFF -> 受信NG
;************************************************************************
_IR_rcv:
    stc.b   ccr,r1l              ;割り込みマスク
    push.l  er1
    ldc.b   #b'10000000,ccr

    push.l  er2 ;ER0,ER1はCからコールした段階で自動的にPUSHされる
    push.l  er3
    push.l  er4
    push.l  er5
    push.l  er6

    mov.b   @IR_port,r0l         ;赤外線受光?
    and.b   #IR_bit,r0l
    bne     IR_rcv_NG

    bsr     check_leader         ;リーダチェック
    bne     IR_rcv_NG
```

```
    bsr     getcode:16          ;カスタムコード(1)
    mov.b   @ircode,r0l
    cmp.b   #h'00,r0l
    bne     IR_rcv_NG

    bsr     getcode:16          ;カスタムコード(2)
    mov.b   @ircode,r0l
    cmp.b   #h'ff,r0l
    bne     IR_rcv_NG

    bsr     getcode:16          ;データ
    mov.b   @ircode,r0l
    mov.b   r0l,@ircmd

    bsr     getcode:16          ;反転データ
    mov.b   @ircode,r0l
    not.b   r0l
    mov.b   @ircmd,r0h
    cmp.b   r0l,r0h
    bne     IR_rcv_NG

IR_rcv_OK:
    mov.b   @ircmd,r0l
    mov.b   #h'00,r0h
    bra     IR_rcv_EXIT

IR_rcv_NG:
    mov.w   #h'ffff,r0

IR_rcv_EXIT:
    pop.l   er6
    pop.l   er5
    pop.l   er4
    pop.l   er3
    pop.l   er2 ;ER0,ER1はCにリターンするとき自動的にPOPされる

    pop.l   er1
    and.b   #B'10000000,r1l
    beq     IR_rcv_EXIT_01
    bra     IR_rcv_EXIT_02
IR_rcv_EXIT_01:
    andc    #B'01111111,ccr
IR_rcv_EXIT_02:

    rts

;*****************************************************************************
;   リーダコードチェック
;*****************************************************************************
check_leader:
    mov.l   #9000,er1           ;8.1ms
check_leader_01:
    mov.b   @IR_port,r0l
    and.b   #IR_bit,r0l
    bne     check_leader_NG:16  ;短すぎ
    dec.l   #1,er1
    bne     check_leader_01
```

```
    mov.l   #2000,er1           ;1.8ms
check_leader_11:
    mov.b   @IR_port,r0l
    and.b   #IR_bit,r0l
    bne     check_leader_12:16
    dec.l   #1,er1
    bne     check_leader_11
    bra     check_leader_NG     ;長すぎ
check_leader_12:

    mov.l   #4500,er1           ;4.05ms
check_leader_21:
    mov.b   @IR_port,r0l
    and.b   #IR_bit,r0l
    beq     check_leader_NG:16  ;短すぎ
    dec.l   #1,er1
    bne     check_leader_21

    mov.l   #1000,er1           ;0.9ms
check_leader_31:
    mov.b   @IR_port,r0l
    and.b   #IR_bit,r0l
    beq     check_leader_32:16
    dec.l   #1,er1
    bne     check_leader_31
    bra     check_leader_NG     ;長すぎ
check_leader_32:

check_leader_OK:
    mov.b   #h'00,r0l
    rts

check_leader_NG:
    mov.b   #h'ff,r0l
    rts

;*****************************************************************************
;   コード受信
;*****************************************************************************
getcode:
    mov.b   #8,r2l              ;ビット数
    mov.b   #0,r1l              ;受信コード

getcode_01:
    mov.b   @IR_port,r0l        ;立ち上がり検知
    and.b   #IR_bit,r0l
    beq     getcode_01

    bsr     wait15bit           ;0.84ms待つ

    mov.b   @IR_port,r0l
    and.b   #IR_bit,r0l
    bne     getcode_bit1
getcode_bit0:
    andc    #b'11111110,ccr
    rotxr.b r1l
    bra     getcode_03
```

```
getcode_bit1:
    orc     #b'00000001,ccr
    rotxr.b r1l
getcode_02:
    mov.b   @IR_port,r0l
    and.b   #IR_bit,r0l
    bne     getcode_02
getcode_03:

    dec.b   r2l
    bne     getcode_01

    mov.b   r1l,@ircode          ;赤外線コードストア

    rts

wait15bit:
    push.l  er0
    mov.l   #2795,er0
wait15bit_01:
    dec.l   #1,er0
    bne     wait15bit_01
    pop.l   er0
    rts

;=====================================================================
;       ワークエリア
;=====================================================================
        .section B,data,ALIGN=2

ircmd       .res.b      1   ;赤外線コマンド
ircode      .res.b      1   ;赤外線リモコンコード

    .end
```

◆

このプログラムは NEC フォーマットで送信する赤外リモコン送信機がないと遊ぶことができません。しかし，家の片隅に使わなくなったリモコンが一つや二つ転がっていると思います。そのリモコンで動かすことができたら送信機を作る手間が省けます。また，中身の分からない機械の仕組みを探るのは知的好奇心がくすぐられないでしょうか。というわけで，次の章では，この章で作った NEC フォーマットの受信プログラムをベースにして，テレビやビデオやラジカセのリモコンで赤外リモコンカーを動かすことを考えてみましょう。

第5章 家にあるリモコンで赤外リモコンカーを動かしてみよう

5-1. 解析ツール「LogicAnalyze」
5-2. National　HK9098T
5-3. SONY　RMT-V410B

市販の赤外リモコンを利用するには，どんな信号が出力されているか知っている必要があります。これまでNECフォーマットで考えてきましたが，持っている赤外リモコンが NEC フォーマットだとは限りません。しかも，どんな信号が出力されるのかについては，ほとんど情報が公開されていません。そのため，自分で調べることになります。ただ，オシロスコープかロジックアナライザを持っていればよいのですが，普通は持っていないと思います。そこで，赤外リモコンカーのハードウェアをそのまま使った解析用のツールを用意しましたので，これを利用することにします。

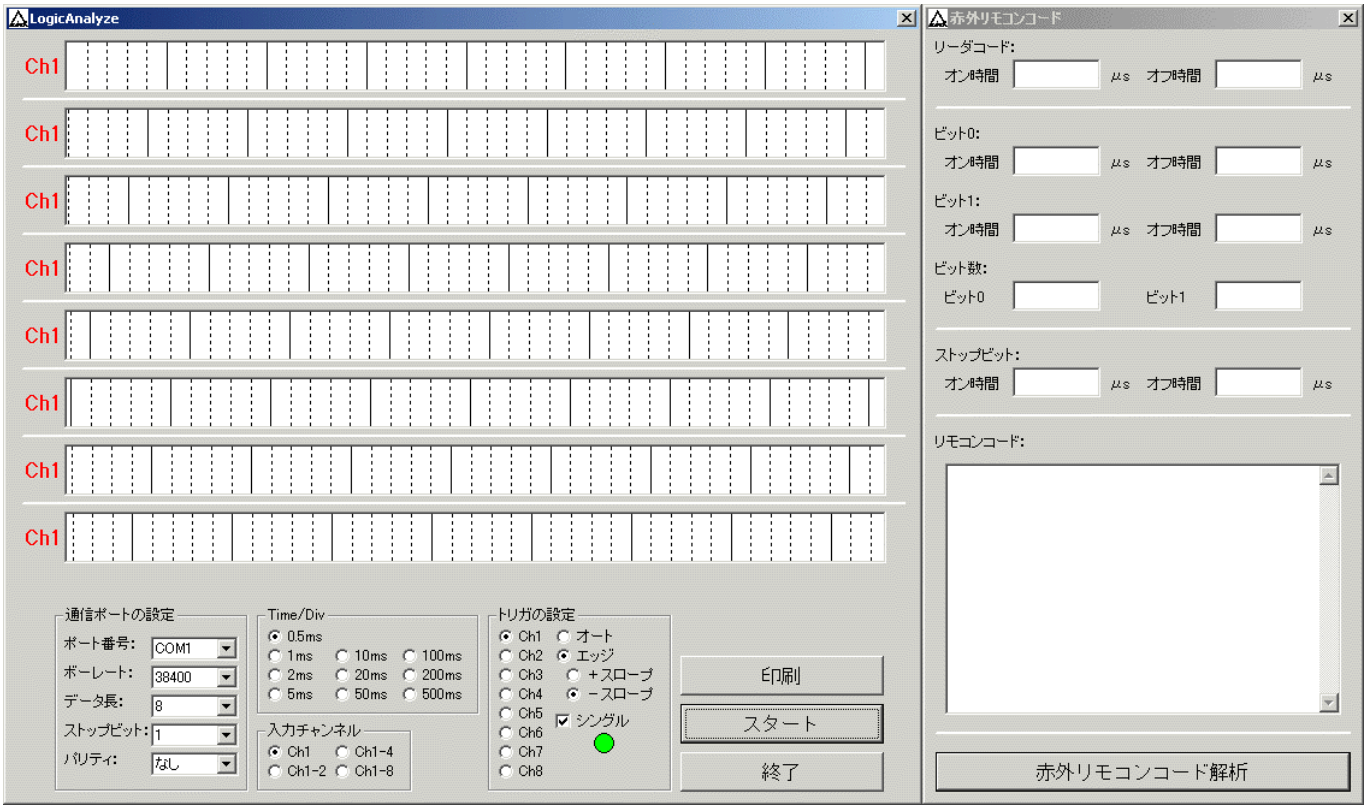
5-1. 解析ツール「LogicAnalyze」

「LogicAnalyze」は VisualBasic で作成した赤外リモコン信号解析ツールです。赤外リモコンカーに実装されている TK-3687mini と組み合わせて使います。もちろん，赤外線信号の受光部も赤外リモコンカーを使います。

最初に，パソコンのシリアルポートと赤外リモコンカーの TK-3687mini を，シリアルケーブルでつなぎます。そして，FDT を使って赤外リモコンカーの TK-3687mini に「**LogicAnalyze. mot**」をダウンロードしてください。

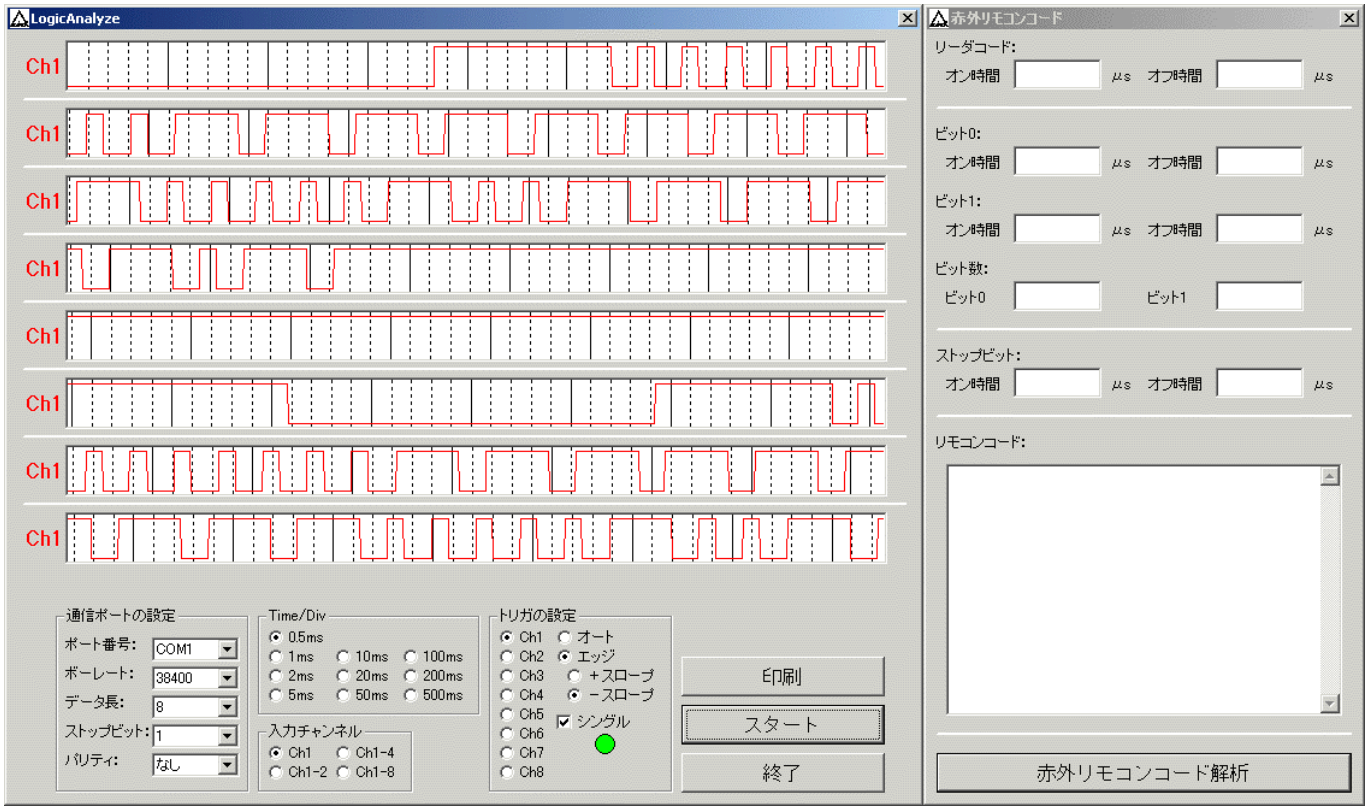
ダウンロードが終わったら，パソコンとつないだまま，赤外リモコンカーのマイコンの電源をつないで TK-3687mini のプログラムを立ち上げます。（モータの電源をつなぐ必要はありません）。

パソコンのプログラム「**LogicAnalyze. exe**」を実行します（ダブルクリック）。すると，次の画面が表示されます。

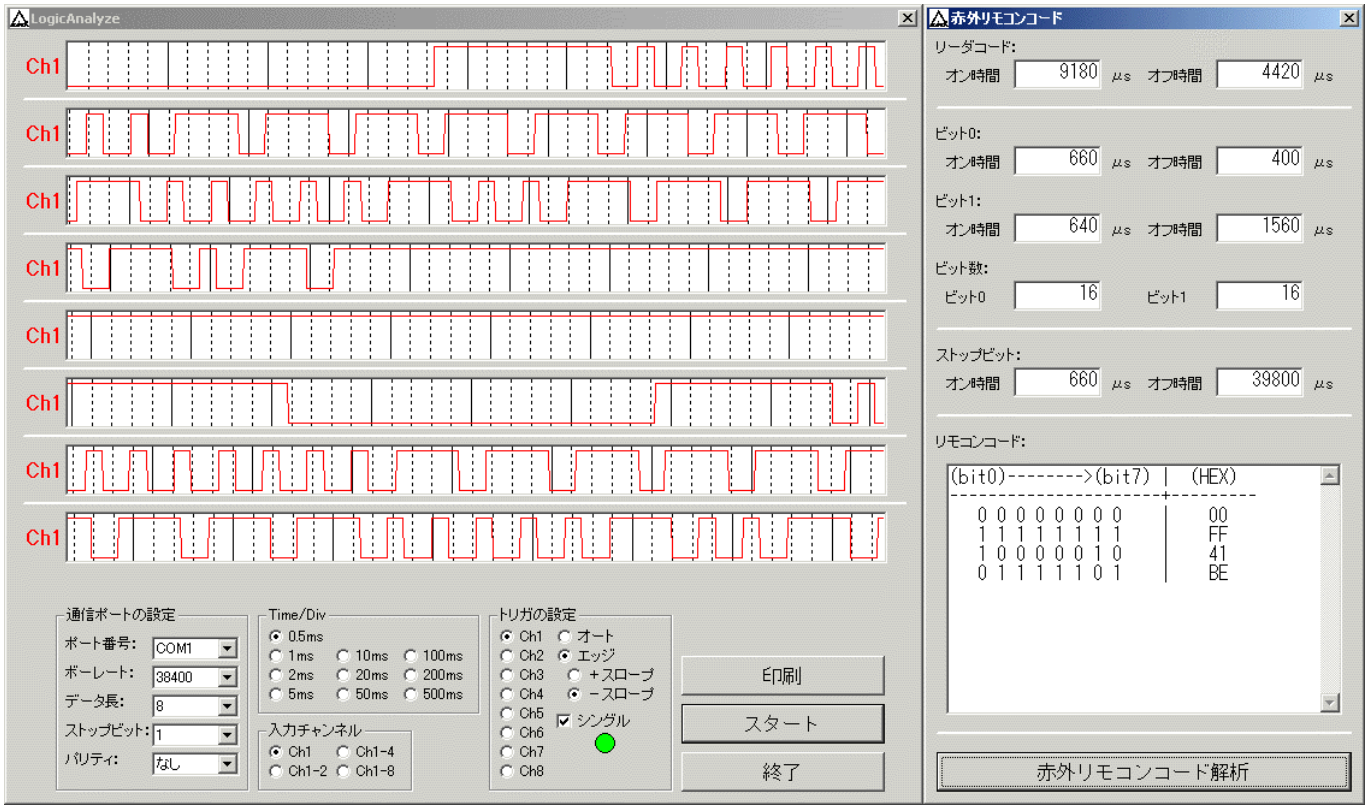


このプログラムは，もともと TK-3687mini を使ったロジックアナライザーで，赤外リモコン解析ツールの機能を追加したものです。プログラム名称や入力チャンネル数などにロジックアナライザーの名残があります。

「スタート」ボタンをクリックすると信号受信待ちになります。赤外リモコンカーの受光部に赤外リモコンを向けてコードを調べたいスイッチを押します。例として，このキットで作成する NEC フォーマットの赤外リモコン送信機の白スイッチ（SW1）を押したときの信号を見てみましょう。受信すると次のように表示されます。



次に，「赤外リモコンコード解析」をクリックすると，波形から読み取った赤外リモコンコードが表示されます。

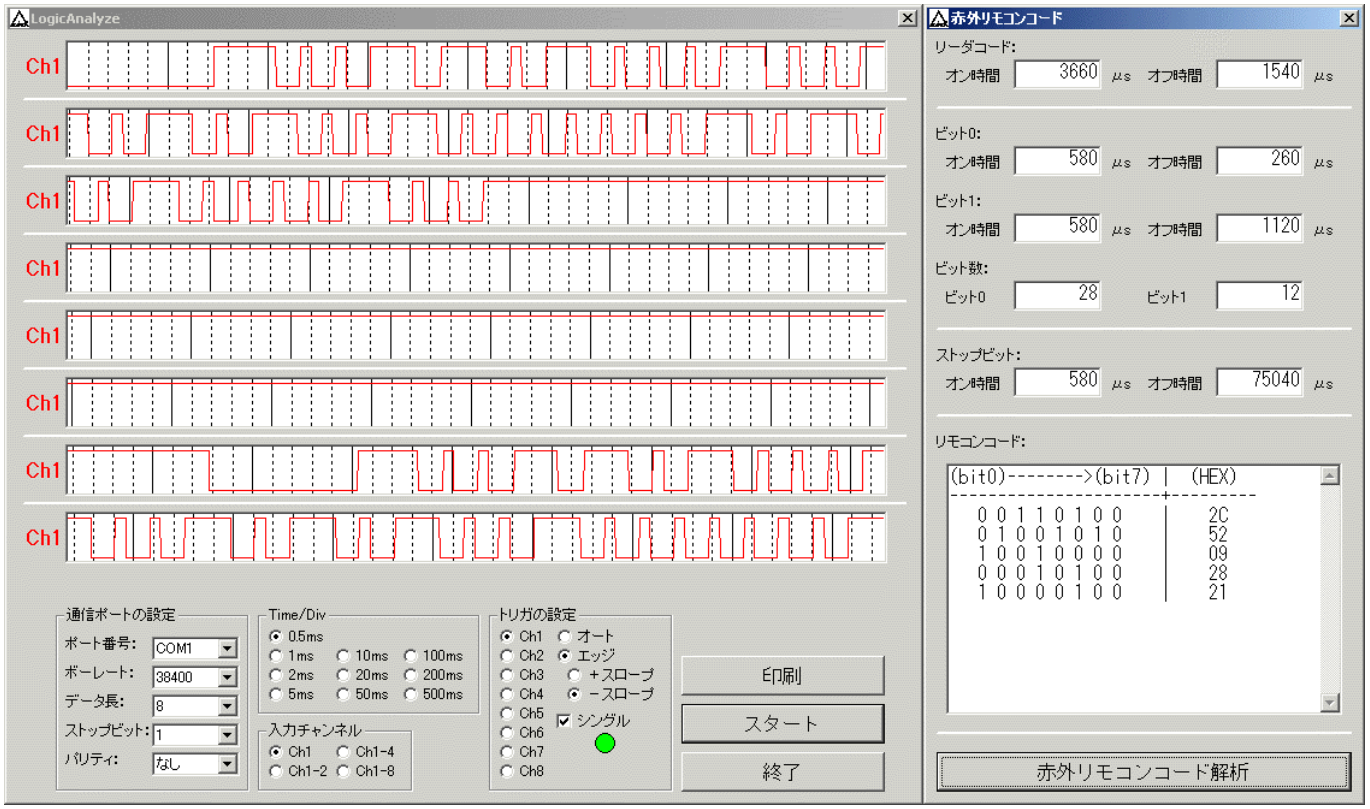


「4-2. リモコンコード」と比較すると，想定しているコードが送信されていることが分かります。ちなみに，リーダコードやビット 0/1 のオン/オフ時間は規格と誤差がありますが，これは空間を伝わってくることや受光素子の特性などにより生じます。つまり，受信プログラムはこれらの誤差も吸収できるように考える必要があります。

5-2. National HK9098T

一例として，今から 20 年近く前に使われていた松下電工株式会社（National）の白熱灯調光器の赤外リモコン（型番：HK9098T，右写真）で，赤外リモコンカーを動かしてみます。スイッチが 3 個しかないので，「明」スイッチで前進，「暗」スイッチで後退，「ON/OFF」スイッチで右回転としましょう。

最初に「LogicAnalyze」で赤外リモコンコードを調べます。例えば，「ON/OFF」スイッチの赤外リモコンコードは次のように解析されました。

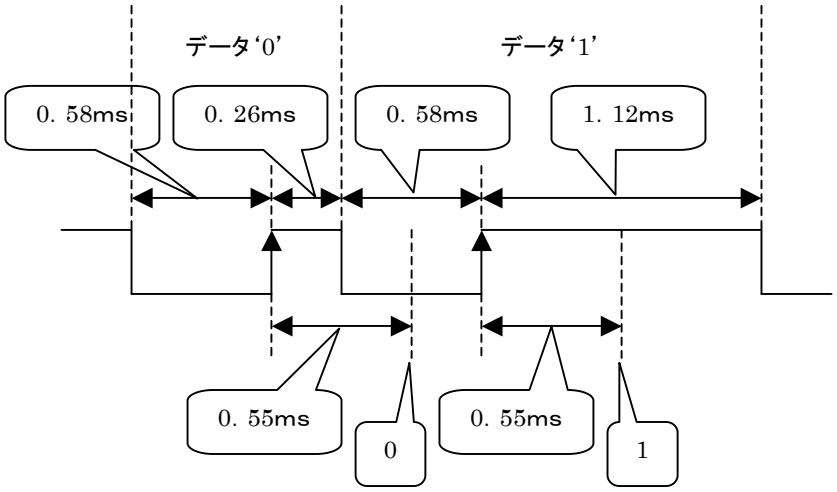


同じように，他のスイッチも調べると次のようになります。

		ON/OFF	明	暗
リーダコード	オン時間（μs）	3660	3660	3660
	オフ時間（μs）	1540	1540	1520
ビット 0	オン時間（μs）	580	580	580
	オフ時間（μs）	260	260	260
ビット 1	オン時間（μs）	580	580	600
	オフ時間（μs）	1120	1120	1120
ストップビット	オン時間（μs）	580	540	580
	オフ時間（μs）	75040	75080	75060
リモコンコード		2C 52 09 28 21	2C 52 09 2A 23	2C 52 09 2B 22

NEC フォーマットと比較して，プログラムをどのように修正するか検討します。大きな違いの一つはリーダコードの時間の長さです。NEC フォーマットに比べるとかなり短くなっています。それで，表からリーダコードのオン時間は 3. 66ms，オフ時間は 1. 53ms とし，その±10%を許容範囲とします。

ビット 0 とビット 1 は，オン時間は同じで，オフ時間の長さが異なる，という時間の関係は変わりません。時間の長さだけ調整して，赤外信号受光 IC の負論理出力の立ち上がりから 0. 55ms 後のポートの状態で判断します。



最後にリモコンコードを検討しましょう。NEC フォーマットの場合は 4 バイトでしたが，このリモコンは 5 バイトです。また，最初の 3 バイトが同じなので，おそらくこれがカスタムコード(もしくはそれに類したもの)なのでしょう。続く 2 バイトがデータですが，NEC フォーマットのようにデータコードとデータコードの反転という関係ではありません。それで，2 バイトデータとして考えることにしましょう。

では，実際にプログラムを修正します。アセンブラで作った赤外リモコン受信ルーチンは次のようになります。変更箇所は黄色でマークしています。

```
;-----------------------------------------------------------------------
;                                                                      |
;   FILE        :ir_rcv.src                                            |
;   DATE        :Tue, Aug 25, 2009                                     |
;   DESCRIPTION :Sub Program                                           |
;   CPU TYPE    :H8/3687                                               |
;                                                                      |
;   This file is programed by TOYO-LINX Co.,Ltd. / yKikuchi            |
;                                                                      |
;-----------------------------------------------------------------------

    .export     _IR_rcv         ;Cからコールされる,"IR_rcv();"
    .section    P,CODE,ALIGN=2

;************************************************************************
;   定数定義
;************************************************************************
PDR7    .equ    h'FFDA          ;ﾎﾟｰﾄﾃﾞｰﾀﾚｼﾞｽﾀ 7
PCR7    .equ    h'FFEA          ;ﾎﾟｰﾄｺﾝﾄﾛｰﾙﾚｼﾞｽﾀ 7

IR_port .equ    PDR7            ;赤外線受信ポート
IR_bit  .equ    b'00000001      ;赤外線受信ビット

;************************************************************************
;   赤外線リモコン信号受信 (HK9098T,National)
;-----------------------------------------------------------------------
```

```
;    戻り値
;    R0 = xxxx -> 受信OK,xxxx=受信コマンド
;    R0 = FFFF -> 受信NG
;*******************************************************************************
_IR_rcv:
    stc.b    ccr,r1l                ;割り込みマスク
    push.l   er1
    ldc.b    #b'10000000,ccr

    push.l   er2 ;ER0,ER1はCからコールした段階で自動的にPUSHされる
    push.l   er3
    push.l   er4
    push.l   er5
    push.l   er6

    mov.b    @IR_port,r0l           ;赤外線受光?
    and.b    #IR_bit,r0l
    bne      IR_rcv_NG

    bsr      check_leader           ;リーダチェック
    bne      IR_rcv_NG

    bsr      getcode:16             ;カスタムコード(1)
    mov.b    @ircode,r0l
    cmp.b    #h'2c,r0l
    bne      IR_rcv_NG

    bsr      getcode:16             ;カスタムコード(2)
    mov.b    @ircode,r0l
    cmp.b    #h'52,r0l
    bne      IR_rcv_NG

    bsr      getcode:16             ;カスタムコード(3)
    mov.b    @ircode,r0l
    cmp.b    #h'09,r0l
    bne      IR_rcv_NG

    bsr      getcode:16             ;データ(1)
    mov.b    @ircode,r0l
    mov.b    r0l,@ircmd0

    bsr      getcode:16             ;データ(2)
    mov.b    @ircode,r0l
    mov.b    r0l,@ircmd1

IR_rcv_OK:
    mov.b    @ircmd0,r0h
    bra      IR_rcv_EXIT

IR_rcv_NG:
    mov.w    #h'ffff,r0

IR_rcv_EXIT:
    pop.l    er6
    pop.l    er5
    pop.l    er4
    pop.l    er3
    pop.l    er2 ;ER0,ER1はCにリターンするとき自動的にPOPされる
```

```
    pop.l    er1
    and.b    #B'10000000,r1l
    beq      IR_rcv_EXIT_01
    bra      IR_rcv_EXIT_02
IR_rcv_EXIT_01:
    andc     #B'01111111,ccr
IR_rcv_EXIT_02:

    rts

;*********************************************************************
;    リーダコードチェック
;*********************************************************************
check_leader:
    mov.l    #3660,er1              ;3.294ms
check_leader_01:
    mov.b    @IR_port,r0l
    and.b    #IR_bit,r0l
    bne      check_leader_NG:16     ;短すぎ
    dec.l    #1,er1
    bne      check_leader_01

    mov.l    #813,er1               ;0.732ms
check_leader_11:
    mov.b    @IR_port,r0l
    and.b    #IR_bit,r0l
    bne      check_leader_12:16
    dec.l    #1,er1
    bne      check_leader_11
    bra      check_leader_NG        ;長すぎ
check_leader_12:

    mov.l    #1530,er1              ;1.377ms
check_leader_21:
    mov.b    @IR_port,r0l
    and.b    #IR_bit,r0l
    beq      check_leader_NG:16     ;短すぎ
    dec.l    #1,er1
    bne      check_leader_21

    mov.l    #340,er1               ;0.306ms
check_leader_31:
    mov.b    @IR_port,r0l
    and.b    #IR_bit,r0l
    beq      check_leader_32:16
    dec.l    #1,er1
    bne      check_leader_31
    bra      check_leader_NG        ;長すぎ
check_leader_32:

check_leader_OK:
    mov.b    #h'00,r0l
    rts

check_leader_NG:
    mov.b    #h'ff,r0l
    rts
```

```
;*************************************************************************
;    コード受信
;*************************************************************************
getcode:
    mov.b   #8,r2l              ;ビット数
    mov.b   #0,r1l              ;受信コード

getcode_01:
    mov.b   @IR_port,r0l        ;立ち上がり検知
    and.b   #IR_bit,r0l
    beq     getcode_01

    bsr     wait15bit           ;0.55ms待つ

    mov.b   @IR_port,r0l
    and.b   #IR_bit,r0l
    bne     getcode_bit1
getcode_bit0:
    andc#b'11111110,ccr
    rotxr.b r1l
    bra     getcode_03
getcode_bit1:
    orc     #b'00000001,ccr
    rotxr.b r1l
getcode_02:
    mov.b   @IR_port,r0l
    and.b   #IR_bit,r0l
    bne     getcode_02
getcode_03:

    dec.b   r2l
    bne     getcode_01

    mov.b   r1l,@ircode         ;赤外線コードストア

    rts

wait15bit:
    push.l  er0
    mov.l   #1828,er0
wait15bit_01:
    dec.l   #1,er0
    bne     wait15bit_01
    pop.l   er0
    rts

;=========================================================================
;       ワークエリア
;=========================================================================
        .section B,data,ALIGN=2

ircmd0      .res.b      1   ;赤外線コマンド(0)
ircmd1      .res.b      1   ;赤外線コマンド(1)
ircode      .res.b      1   ;赤外線リモコンコード

    .end
```

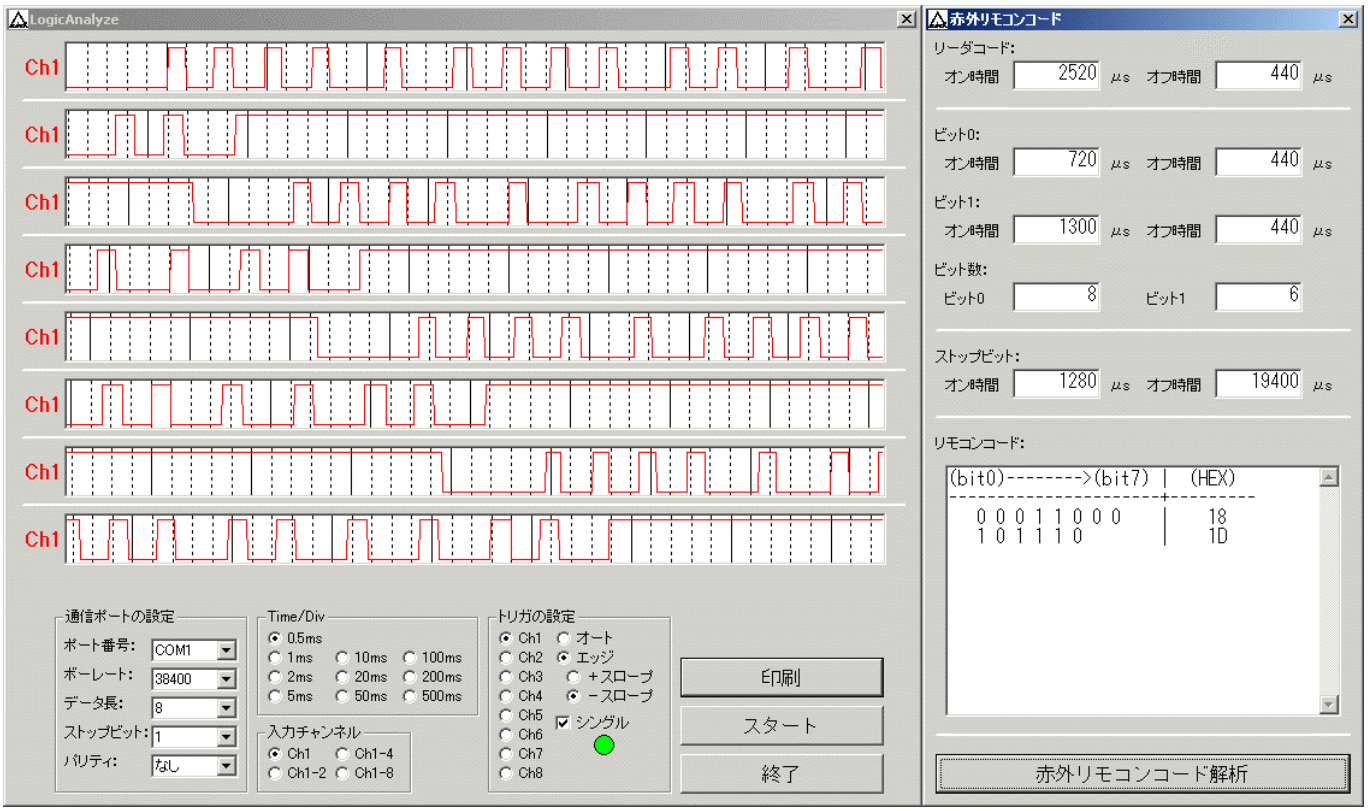
続いて，C のプログラム修正です。赤外リモコンコードが変更になるだけなので，「走行制御」関数を修正します。ソースファイルのファイル名は「ir_remocon_car_01_hk9098. c」です。同じように，変更箇所は黄色でマークしています。

```
/**************************************************************************
    走行制御
**************************************************************************/
void run(void)
{
    unsigned int ans;    //赤外線受信ルーチンの戻り値

    ans = IR_rcv();

    if (ans!=0xffff){
        TimT0.Status = 0;    //タイマストップ
        switch(ans){
            case 0x2a23:     //前進
                cw4();
                break;
            case 0x2b22:     //後退
                ccw4();
                break;
            case 0x2821:     //右回転
                spin_right_3();
                break;
        }
        TimT0.Status = 1;    //タイマスタート
    }

    if (TimT0.Status==3){    //通信が途絶えた
        TimT0.Status = 0;    //タイマストップ
        stop(); //停止
    }
}
```

5-3. SONY　RMT-V410B

別の例として，SONY の VHS ビデオデッキの赤外リモコン（型番：RMT-V410B，右写真）で，赤外リモコンカーを動かしてみます。スイッチの配置から，「一時停止」スイッチで前進，「停止」スイッチで後退，「早送り」スイッチで右旋回，「巻戻し」スイッチで左旋回，「再生」スイッチで右回転としましょう。

最初に「LogicAnalyze」で赤外リモコンコードを調べます。例えば，「再生」スイッチの赤外リモコンコードは次のように解析されました。

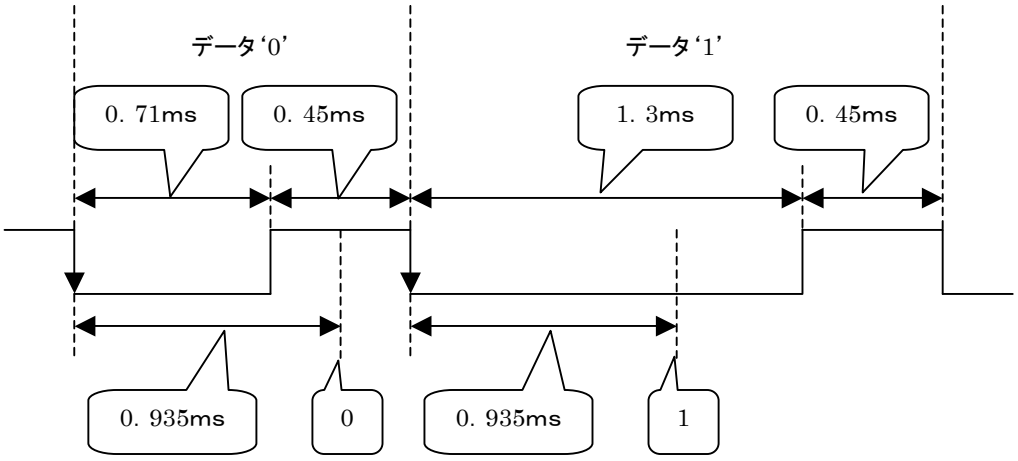


同じように，他のスイッチも調べると次のようになります。

		再生	停止	一時停止	早送り	巻戻し
リーダコード	オン時間（μs）	2520	2500	2480	2560	2500
	オフ時間（μs）	440	460	440	400	440
ビット 0	オン時間（μs）	720	700	700	720	720
	オフ時間（μs）	440	460	460	440	440
ビット 1	オン時間（μs）	1300	1300	1300	1280	1280
	オフ時間（μs）	440	460	460	460	460
ストップビット	オン時間（μs）	1280	1280	1280	1280	1280
	オフ時間（μs）	19400	18200	18780	19400	20000
リモコンコード		18 1D	0F 1D	0E 1D	11 1D	10 1D

NEC フォーマットと比較して，プログラムをどのように修正するか検討します。大きな違いの一つはリーダコードの時間の長さです。NEC フォーマットに比べるとかなり短くなっています。それで，表からリーダコードのオン時間は 2. 5ms，オフ時間は 0. 44ms とし，その±10%を許容範囲とします。

ビット 0 とビット 1 は，NEC フォーマットとは異なり，オフ時間が同じで，オン時間の長さが異なります。それで，赤外信号受光 IC の負論理出力の立ち下がりから 0. 935ms 後のポートの状態で判断します。



ビット 0/1 の判定の変更に伴い，リーダコードの判定にも多少の影響が生じます。リーダコードのオフ時間の判定をしていると，最初のビットの判定時間に影響してしまいます。それで，リーダコードの判定はオン時間のみにします。

最後にリモコンコードを検討しましょう。NEC フォーマットの場合は 32 ビット(4 バイト)でしたが，このリモコンは 14 ビットです。リモコンコード全体のビット数がバイト単位ではないので，上位 2 ビットは 0 固定にして 2 バイトデータとして扱うことにします。

ちなみに･･･。このリモコンで使おうとしているスイッチのリモコンコードは 14 ビットですが，他のスイッチは 11 ビットでした。おそらく，このリモコンのコードはビット数が可変長で，ビット数を決める何らかのデータがあると思われます。もし，それらのスイッチも使うとなると，14 ビットデータも 11 ビットデータも受信できるようにしなければなりません。今回は 14 ビットデータ専用です。

では，実際にプログラムを修正します。アセンブラで作った赤外リモコン受信ルーチンは次のようになります。変更箇所は黄色でマークしています。

```
;------------------------------------------------------------------------
;                                                                       |
;   FILE        :ir_rcv.src                                             |
;   DATE        :Tue, Aug 25, 2009                                      |
;   DESCRIPTION :Sub Program                                            |
;   CPU TYPE    :H8/3687                                                |
;                                                                       |
;   This file is programed by TOYO-LINX Co.,Ltd. / yKikuchi             |
;                                                                       |
;------------------------------------------------------------------------

    .export     _IR_rcv         ;Cからコールされる,"IR_rcv();"
    .section    P,CODE,ALIGN=2

;************************************************************************
;   定数定義
;************************************************************************
PDR7    .equ    h'FFDA          ;ﾎﾟｰﾄﾃﾞｰﾀﾚｼﾞｽﾀ 7
PCR7    .equ    h'FFEA          ;ﾎﾟｰﾄｺﾝﾄﾛｰﾙﾚｼﾞｽﾀ 7
```

```
IR_port .equ     PDR7              ;赤外線受信ポート
IR_bit  .equ     b'00000001        ;赤外線受信ビット

;*********************************************************************
;    赤外線リモコン信号受信 (RMT-V410B,SONY)
;---------------------------------------------------------------------
;    戻り値
;    R0 = xxxx -> 受信OK,xxxx=受信コマンド
;    R0 = FFFF -> 受信NG
;*********************************************************************
_IR_rcv:
    stc.b    ccr,r1l               ;割り込みマスク
    push.l   er1
    ldc.b    #b'10000000,ccr

    push.l   er2 ;ER0,ER1はCからコールした段階で自動的にPUSHされる
    push.l   er3
    push.l   er4
    push.l   er5
    push.l   er6

    mov.b    @IR_port,r0l          ;赤外線受光?
    and.b    #IR_bit,r0l
    bne      IR_rcv_NG

    bsr      check_leader          ;リーダチェック
    bne      IR_rcv_NG

    bsr      getcode:16            ;データ
    mov.w    @ircode,r0
    mov.w    r0,@ircmd

IR_rcv_OK:
    bra      IR_rcv_EXIT

IR_rcv_NG:
    mov.w    #h'ffff,r0

IR_rcv_EXIT:
    pop.l    er6
    pop.l    er5
    pop.l    er4
    pop.l    er3
    pop.l    er2 ;ER0,ER1はCにリターンするとき自動的にPOPされる

    pop.l    er1
    and.b    #B'10000000,r1l
    beq      IR_rcv_EXIT_01
    bra      IR_rcv_EXIT_02
IR_rcv_EXIT_01:
    andc     #B'01111111,ccr
IR_rcv_EXIT_02:

    rts

;*********************************************************************
;    リーダコードチェック
;*********************************************************************
```

```
check_leader:
    mov.l   #2500,er1           ;2.25ms
check_leader_01:
    mov.b   @IR_port,r0l
    and.b   #IR_bit,r0l
    bne     check_leader_NG:16  ;短すぎ
    dec.l   #1,er1
    bne     check_leader_01

    mov.l   #555,er1            ;0.5ms
check_leader_11:
    mov.b   @IR_port,r0l
    and.b   #IR_bit,r0l
    bne     check_leader_12:16
    dec.l   #1,er1
    bne     check_leader_11
    bra     check_leader_NG     ;長すぎ
check_leader_12:

check_leader_OK:
    mov.b   #h'00,r0l
    rts

check_leader_NG:
    mov.b   #h'ff,r0l
    rts

;*********************************************************************
;   コード受信
;*********************************************************************
getcode:
    mov.b   #14,r2l             ;ビット数
    mov.w   #0,r1               ;受信コード

getcode_01:
    mov.b   @IR_port,r0l        ;立ち下がり検知
    and.b   #IR_bit,r0l
    bne     getcode_01

    bsr     wait15bit           ;0.935ms待つ

    mov.b   @IR_port,r0l
    and.b   #IR_bit,r0l
    beq     getcode_bit1
getcode_bit0:
    andc    #b'11111110,ccr
    rotxr.w r1
    bra     getcode_03
getcode_bit1:
    orc     #b'00000001,ccr
    rotxr.w r1
getcode_02:
    mov.b   @IR_port,r0l
    and.b   #IR_bit,r0l
    beq     getcode_02
getcode_03:

    dec.b   r2l
```

```
    bne     getcode_01

    shlr.w  r1
    shlr.w  r1
    mov.w   r1,@ircode          ;赤外線コードストア

    rts

wait15bit:
    push.l  er0
    mov.l   #3116,er0
wait15bit_01:
    dec.l   #1,er0
    bne     wait15bit_01
    pop.l   er0
    rts

;=====================================================================
;       ワークエリア
;=====================================================================
    .section    B,data,ALIGN=2

ircmd       .res.w      1   ;赤外線コマンド
ircode      .res.w      1   ;赤外線リモコンコード

    .end
```

続いて，C のプログラム修正です。赤外リモコンコードが変更になるだけなので，「走行制御」関数を修正します。ソースファイルのファイル名は「ir_remocon_car_01_rmtv410b. c」です。同じように，変更箇所は黄色でマークしています。

```
/*********************************************************************
    走行制御
*********************************************************************/
void run(void)
{
    unsigned int ans;    //赤外線受信ルーチンの戻り値

    ans = IR_rcv();

    if (ans!=0xffff){
        TimTO.Status = 0;    //タイマストップ
        switch(ans){
            case 0x1d0e:     //前進
                cw4();
                break;
            case 0x1d11:     //前進&右
                turn_right_3();
                break;
            case 0x1d10:     //前進&左
                turn_left_3();
                break;
            case 0x1d0f:     //後退
                ccw4();
                break;
            case 0x1d18:     //右回転
```

```
                spin_right_3();
                break;
        }
        TimT0.Status = 1;    //タイマスタート
    }

    if (TimT0.Status==3){    //通信が途絶えた
        TimT0.Status = 0;    //タイマストップ
        stop(); //停止
    }
}
```

◆

使わなくなった赤外リモコンを利用する際の，コード判定プログラムの事例を取り上げてみました。解析ツールを使うことで，比較的簡単に実現することができます。もちろん，ここで取り上げたものと大きく異なるフォーマットもたくさんありますので，このツールでは解析できないものもあるはずです。また，キャリア周波数(38KHz)が異なる場合は，受光素子の選択などハードウェアの設計から考えないといけません。これを参考に，お持ちのリモコンの有効活用を考えてみてください。

ところで，このツールの解析結果(リーダコードの時間，ビット 0/1 の時間，ビット数，リモコンコードなど)は，あくまで読み取った波形から推測したものに過ぎません。それぞれのメーカは赤外リモコンの規格を定めているはずで，このツールの解析結果が規格とまったく同じであると保証することはできません。というより，違うと考えていたほうがよいでしょう。やはり一番よいのは規格を手に入れることです。入手が難しい場合に，参考にするためにツールを使い，とりあえず動くものを作ろう，というのがこのツールのふさわしい使い方です。

第6章 赤外リモコン送信機

4 章で考えた，NEC フォーマットの赤外リモコン送信機を自作します。

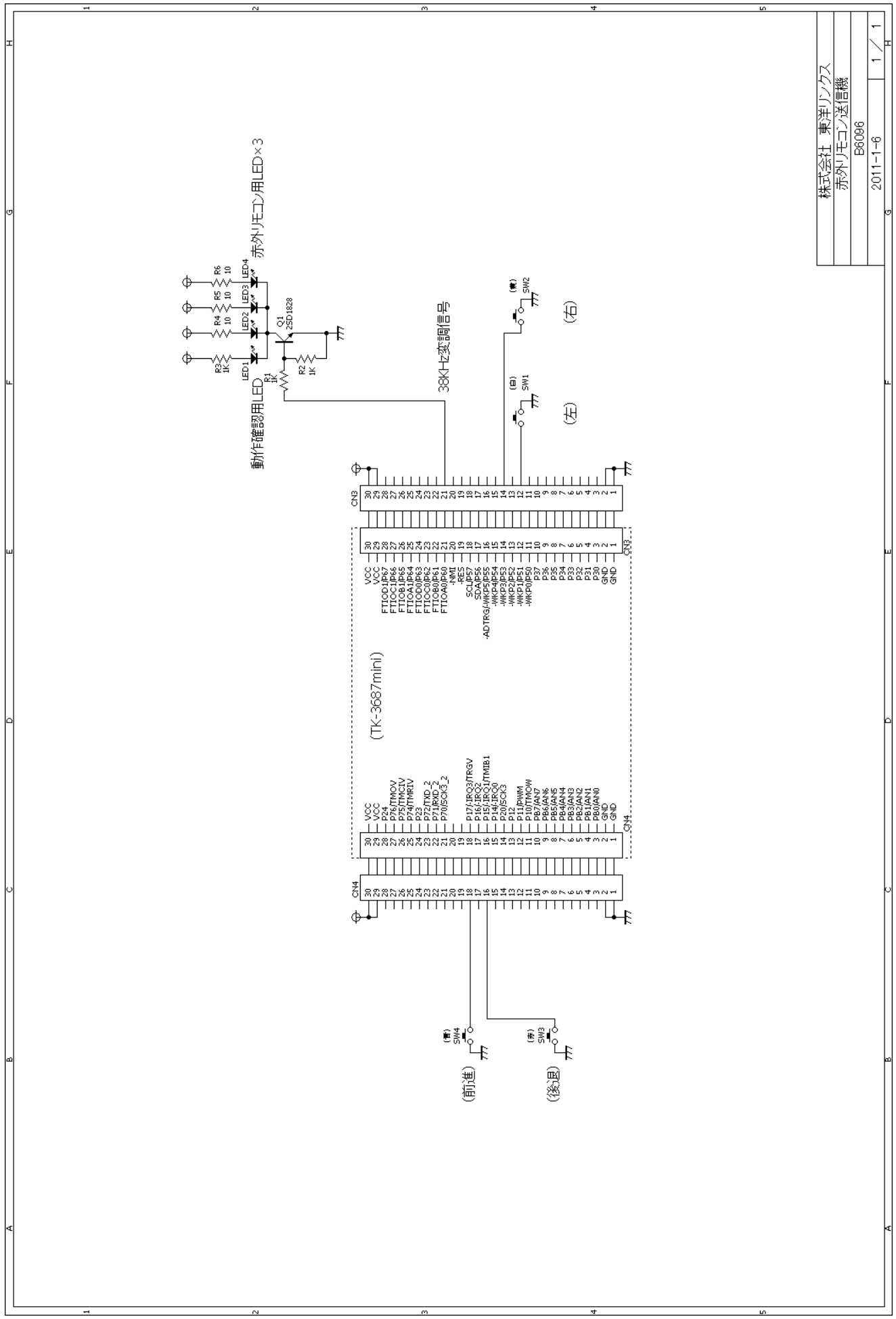
6-1. 部品の確認

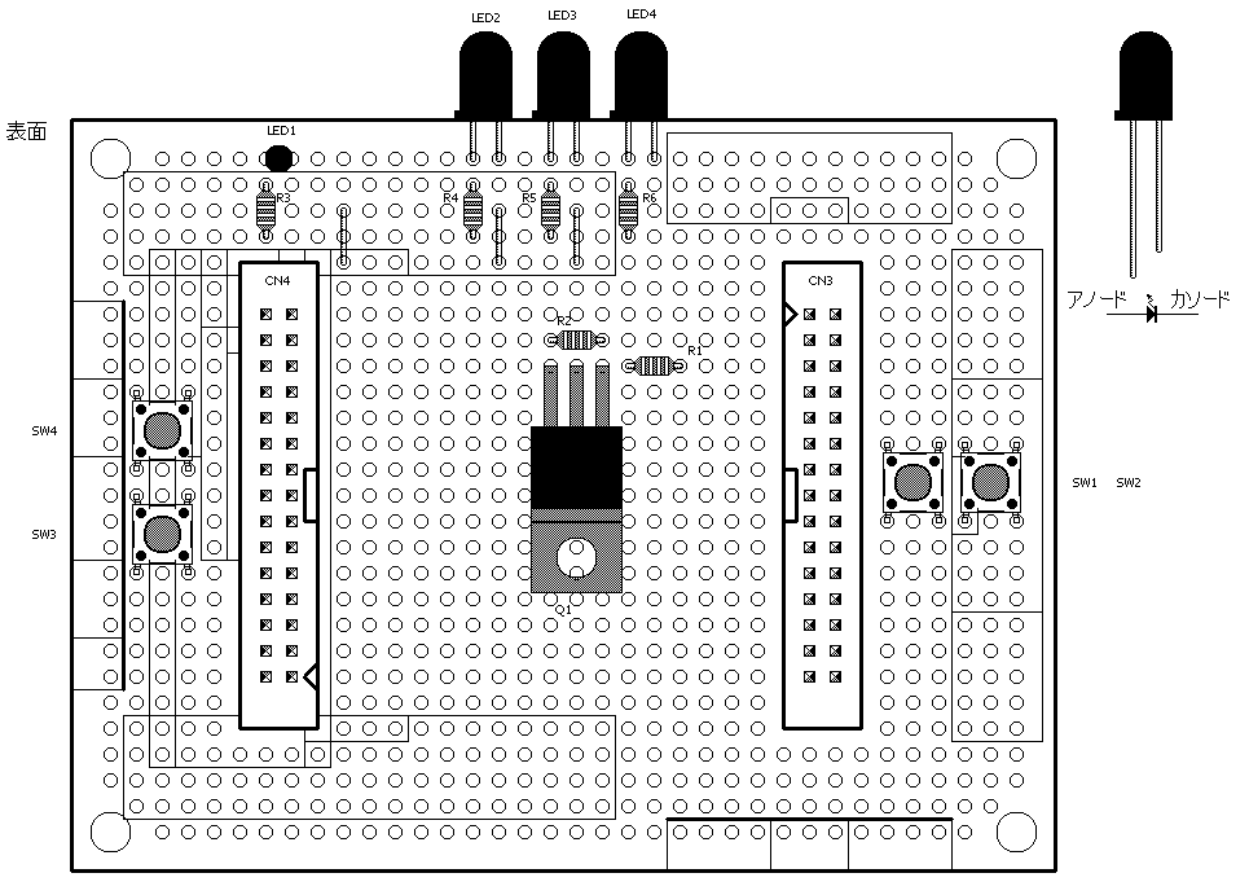
最初に，部品表と比較して部品が全てそろっているか確認しましょう。部品によっては相当品使用の場合もあります。（部品が足りないときは巻末記載の連絡先までお問い合わせください。）

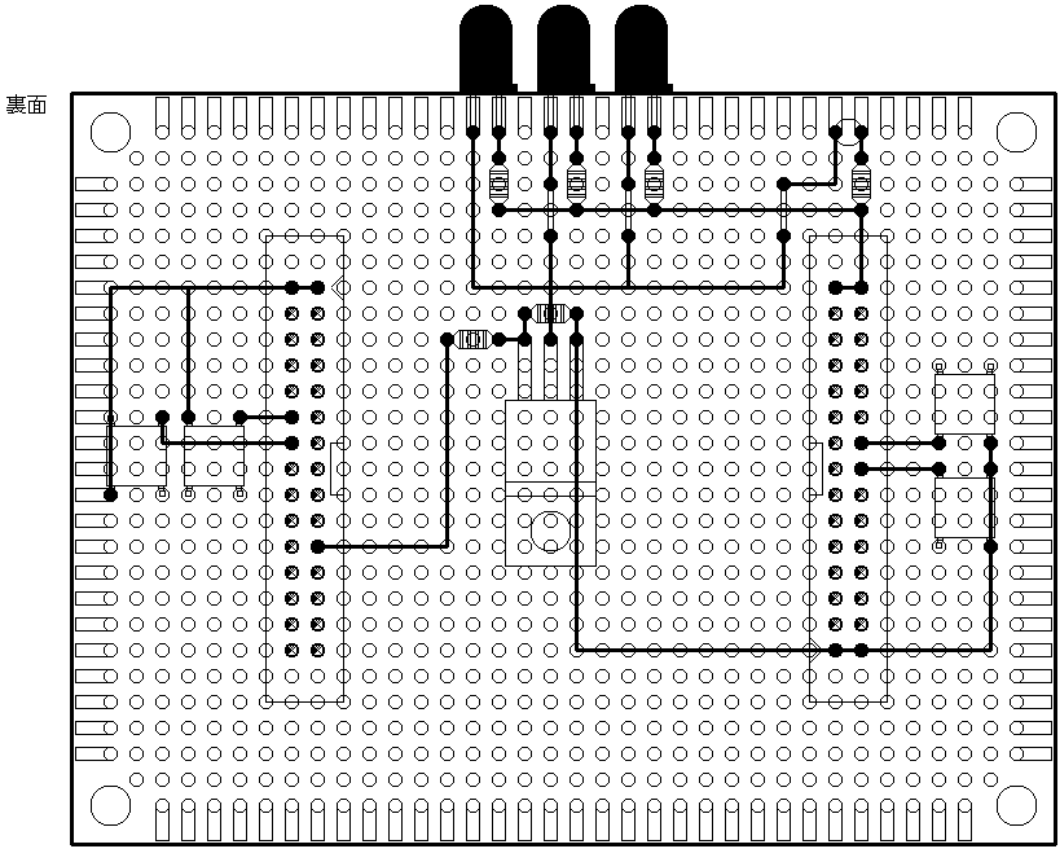
赤外リモコン送信機 組み立てキット

	部品番号	型名，規格	メーカー	数量	付属数量	備考
1	**■赤外リモコン送信機基板**					
2	Q1	2SD1828		1	1	*1，ダーリントントランジスタ
3	LED1	HLMP-6300#A04	HP	1	1	*1
4	LED2～4	TLN115	東芝	3	3	*1，赤外LED
5	SW1～4	SKHHAK/AM/DC	ALPS	4	4	*1，赤，青，黄，白
6	R1～3	1KΩ		3	3	
7	R4～6	10Ω		3	3	
8	CN3,4	HIF3FC-30PA-2.54DSA	HRS	2	2	*1，CN1,2は欠番
9	ラッピングケーブル	50cm		1	1	*2，メッキ線として使用
10	メッキ線			0	0	ハンダ面結線用 *2
11	ユニバーサル基板	B6093(95×72㎜)	東洋リンクス	1	1	
12						
13	**■CPUボード（組立キットの場合）**					
14	CPUボード	TK-3687mini	東洋リンクス	1	1	フラットパッケージ実装済み
15	REG1	TA48M05F(S)	東芝	1	1	
16	X1	20MHz		1	1	メインクロック
17	X2	32.768KHz		1	1	サブクロック
18	D3	1SS133-T72	ROHM	1	1	*1
19	LED1	HLMP-6300#A04	HP	1	1	*1
20	C3,19	47～100μF/16V		2	2	
21	C4,6,17,18,20	10μF/16V		5	5	
22	SW1	SKHHAK/AM/DC	ALPS	1	1	*1
23	CN1	B2P-SHF-1AA	JST	1	1	電源用
24	CN3,4	HIF3FB-30DA-2.54DSA	HRS	2	2	基板間コネクタ，ハンダ面に実装
25	CN5	D-Sub9pin		1	1	ストレート
26	JP1	2pin		1	1	ピンとソケットのセット
27						
28	**■その他**					
29	電池ボックス			1	1	単3×4本用，ケーブル付
30						
31						

(*1)相当品を使用することがあります。
(*2)ラッピングケーブルの被覆をはがし2本をよじって使用します。また，抵抗やコンデンサの足も流用できます。


株式会社 東洋ソケクス
赤外リモコン送信機
B6096
2011-1-6
1／1
赤外リモコン用LED×3
動作確認用LED
38KHz変調信号
(TK-3687mini)
(前進)
(後退)
(左)
(右)





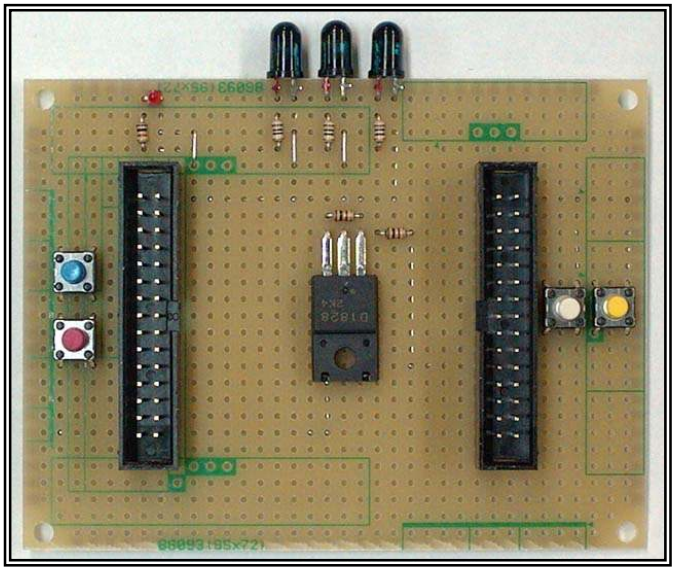
株式会社 東洋リンクス	
リモコン送信機 実装図	
B6096	
2009-9-15	1 ／ 1

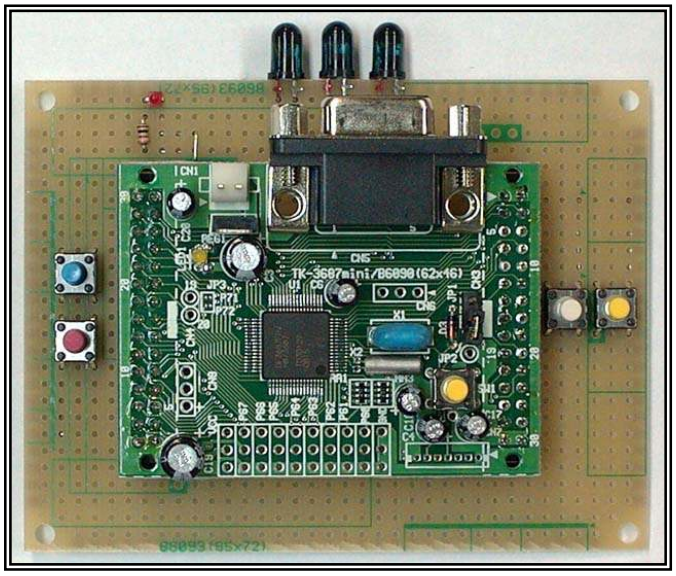
6-3. 組み立て

実装図の表面（部品面）を見てユニバーサル基板に部品を載せます。次に，実装図の裏面（半田面）を見てメッキ線での配線を済ませてください。実装図どおりに作ればラッピングケーブルの結線は必要ありません。

配線が終了したら，回路図どおり配線されているかもう一度確認して下さい。確認方法は，テスタで部品面の端子間の抵抗を測り，導通があるか，すなわち 0Ω か否かで判断します。また，半田付けがきちんと行なわれているか見ておきましょう。動かない原因の大部分は配線ミスと半田付け不良です。

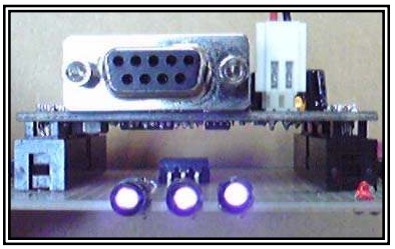
問題なければ，TK-3687mini を取り付けて組み立て終了です。TK-3687mini に FDT で**“ir_remocon_send.mot”**をダウンロードします。



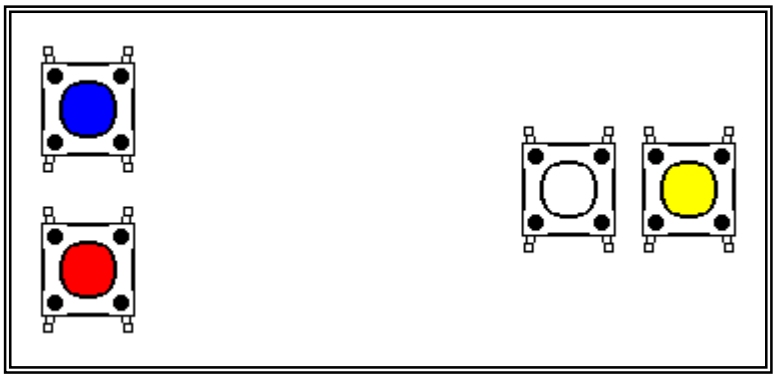


■動作確認の方法

どれかスイッチを押してみてください。押している間，動作確認用の LED が点滅すれば，まずは OK です。4 つのスイッチのいずれを押したときにも LED が点滅することを確認してください。また，本当に赤外線が発光されているか確認したい場合は，カメラ付き携帯電話か PHS，もしくはデジカメで，スイッチを押しているときの赤外 LED の様子をファインダで見てください。大抵の場合，LED が光っている様子を見ることができます（カメラに使われているイメージセンサが可視光線だけでなく赤外線にも反応するものが多いため，但し，赤外線フィルタが入っていると見えません）。



6-4. 操作方法



リモコンの「青」スイッチオン	→	前進
リモコンの「赤」スイッチオン	→	後退
リモコンの「青」「黄」スイッチ同時オン	→	前進右旋回
リモコンの「青」「白」スイッチ同時オン	→	前進左旋回
リモコンの「赤」「黄」スイッチ同時オン	→	後退右旋回
リモコンの「赤」「白」スイッチ同時オン	→	後退左旋回
リモコンの「黄」スイッチオン	→	右回転
リモコンの「白」スイッチオン	→	左回転
リモコンの「青」「赤」スイッチ同時オン	→	停止
0．3 秒以上，赤外リモコン信号を受信しない	→	停止

（注意：タクトスイッチの色は一例です。説明の都合上，上の図の色を使って説明していますが，実際のキットでは異なる色が入っていることがありますし，すべてのスイッチの色が同じこともあります。）

◆

文章にすると複雑に見えますが，なるべく感覚的に動かせるようにスイッチを割り当てました。おそらく，動かしているうちに自然に覚えられると思います。

6-5. プログラム

```
/************************************************************************/
/*                                                                      */
/*  FILE        :ir_remocon_send.c                                      */
/*  DATE        :Wed, Sep 09, 2009                                      */
/*  DESCRIPTION :Main Program                                           */
/*  CPU TYPE    :H8/3687                                                */
/*                                                                      */
/*  This file is programed by TOYO-LINX Co.,Ltd. / yKikuchi             */
/*                                                                      */
/************************************************************************/

/************************************************************************
    スイッチの並び方とポート番号の対応
------------------------------------------------------------------------
        +-----+
        | SW4 |
        |(P17)|
        +-----+                     +-----+ +-----+
                                    | SW1 | | SW2 |
                                    |(P51)| |(P53)|
        +-----+                     +-----+ +-----+
        | SW3 |
        |(P15)|
        +-----+
************************************************************************/

/************************************************************************
    インクルードファイル
************************************************************************/
#include<machine.h>      //H8特有の命令を使う
#include"iodefine.h"     //内蔵I/Oのラベル定義
#include"binary.h"       //Cで2進数を使うための定義

/************************************************************************
    定数の定義（直接指定）
************************************************************************/
#define     OK          0       //戻り値
#define     NG          -1      //戻り値

#define     SW_MODE     1       //スイッチ入力による赤外出力モード
                                // 0:単独スイッチ入力
                                //  SW1-31h,SW2-32h,SW3-33h,SW4-34h
                                // 1:複数スイッチ入力(ID固定=4xh)
                                //  bit0=SW1
                                //  bit1=SW2
                                //  bit2=SW3
                                //  bit3=SW4
                                //  bit4=0
                                //  bit5=0
                                //  bit6=1
                                //  bit7=0
                                // 2:複数スイッチ入力(ID可変)
                                //  bit0=SW1
                                //  bit1=SW2
```

```
                                    //  bit2=SW3
                                    //  bit3=SW4
                                    //  bit4=ID_bit0
                                    //  bit5=ID_bit1
                                    //  bit6=ID_bir2
                                    //  bit7=ID_bit3

/*************************************************************************
    定数エリアの定義(ROM)
*************************************************************************/
//                                    0         1         2         3
//                                    01234567890123456789012345678901
const char StartChkCnstData[32]   =   {"TOYO-LINX Co.,Ltd. IR-Send     "};

/*************************************************************************
    グローバル変数の定義とイニシャライズ(RAM)
*************************************************************************/
// スイッチ入力に関係した変数 ---------------------------------------
unsigned char    SwData1      =   0;  //ファーストリード
unsigned char    SwData2      =   0;  //ダブルリードにより決定したデータ
unsigned char    SwData3      =   0;  //前回のダブルリードで決定したデータ
unsigned char    SwData4      =   0;  //0→1に変化したデータ
unsigned char    SwStatus     =   0;  //スイッチ入力ステータス
                                      // 0:ファーストリード
                                      // 1:ダブルリード

// IDに関係した変数 -------------------------------------------------
unsigned char    IDNo         =   0;  //ID番号

// スタート識別に関係した変数 ---------------------------------------
char         StartChkData[32];        //スタート識別データ

/*************************************************************************
    関数の定義
*************************************************************************/
void    button(unsigned char);
void    button1(void);
void    button2(void);
void    button3(void);
void    button4(void);
void    buttonID(unsigned char);
void    id_set(void);
void    init_io(void);
void    init_tmv(void);
void    intprog_tmv(void);
void    led_disp(unsigned char);
void    main(void);
char    start_check(void);
void    switch_in(void);

void    IR_sig_out(char);    //アセンブラ関数の定義

/*************************************************************************
    メインプログラム
*************************************************************************/
void main(void)
{
    // イニシャライズ ---------------------------------------------
```

```
    init_io();
    init_tmv();

    id_set();

    if (SW_MODE==2){
        while((IO.PDR1.BIT.B7==0)
            ||(IO.PDR1.BIT.B5==0)
            ||(IO.PDR5.BIT.B3==0)
            ||(IO.PDR5.BIT.B1==0))    {}  //スイッチ全オフまで待つ
    }

    // メインループ ----------------------------------------------------
    while(1){
        if (SW_MODE==0){     //単独スイッチ入力モード
            if      ((SwData2 & 0x02)==0x02){     //ボタン1が押された
                button1();
                SwData4 = 0;
            }
            else if ((SwData2 & 0x08)==0x08){     //ボタン2が押された
                button2();
                SwData4 = 0;
            }
            else if ((SwData2 & 0x20)==0x20){     //ボタン3が押された
                button3();
                SwData4 = 0;
            }
            else if ((SwData2 & 0x80)==0x80){     //ボタン4が押された
                button4();
                SwData4 = 0;
            }
        }
        else if (SW_MODE==1){     //複数スイッチ入力モード(ID固定)
            if ((SwData2 & _10101010B)!=0){
                button(SwData2);
                SwData4 = 0;
            }
            else{
                IO.PDR6.BYTE = _11111110B;
            }
        }
        else if (SW_MODE==2){     //複数スイッチ入力モード(ID可変)
            if ((SwData2 & _10101010B)!=0){
                buttonID(SwData2);
                SwData4 = 0;
            }
            else{
                IO.PDR6.BYTE = _11111110B;
            }
        }
    }
}

/************************************************************************
    ボタン１
************************************************************************/
void button1(void)
{
```

```
    IR_sig_out(0x31);
}

/*****************************************************************************
    ボタン２
*****************************************************************************/
void button2(void)
{
    IR_sig_out(0x32);
}

/*****************************************************************************
    ボタン３
*****************************************************************************/
void button3(void)
{
    IR_sig_out(0x33);
}

/*****************************************************************************
    ボタン４
*****************************************************************************/
void button4(void)
{
    IR_sig_out(0x34);
}

/*****************************************************************************
    ボタン/複数入力,固定ID
*****************************************************************************/
void button(unsigned char sw)
{
    unsigned char d = 0;

    if ((sw & 0x02)==0x02)    {d = d | _00000001B;}
    if ((sw & 0x08)==0x08)    {d = d | _00000010B;}
    if ((sw & 0x20)==0x20)    {d = d | _00000100B;}
    if ((sw & 0x80)==0x80)    {d = d | _00001000B;}
    IR_sig_out(d | 0x40);    //固定IDを付加する
    led_disp(d);
}

/*****************************************************************************
    ボタン/複数入力,可変ID
*****************************************************************************/
void buttonID(unsigned char sw)
{
    unsigned char d = 0;

    if ((sw & 0x02)==0x02)    {d = d | _00000001B;}
    if ((sw & 0x08)==0x08)    {d = d | _00000010B;}
    if ((sw & 0x20)==0x20)    {d = d | _00000100B;}
    if ((sw & 0x80)==0x80)    {d = d | _00001000B;}
    IR_sig_out(d | IDNo);    //IDを付加する
    led_disp(d);
}

/*****************************************************************************
```

```
    LED表示(Port6)
*****************************************************************************/
void led_disp(unsigned char sw)
{
    unsigned char pd6_temp1;
    unsigned char pd6_temp2;

    pd6_temp1 =  IO.PDR6.BYTE & _00000001B;
    pd6_temp2 = ~IO.PDR6.BYTE & _11111110B;

    if ((sw & 0x01)==0x01){
        if (pd6_temp2==0){
            pd6_temp2 = _00000010B;
        }
        else{
            pd6_temp2 = pd6_temp2 << 1;
            if (pd6_temp2==0){
                pd6_temp2 = _00000010B;
            }
        }
    }
    else if ((sw & 0x02)==0x02){
        if (pd6_temp2==0){
            pd6_temp2 = _10000000B;
        }
        else{
            pd6_temp2 = pd6_temp2 >> 1;
            if (pd6_temp2==_00000001B){
                pd6_temp2 = _10000000B;
            }
        }
    }
    else if ((sw & 0x04)==0x04)   {pd6_temp2 = _00000000B;}
    else if ((sw & 0x08)==0x08)   {pd6_temp2 = _00000000B;}
    else                          {pd6_temp2 = _00000000B;}

    IO.PDR6.BYTE = pd6_temp1 | (~pd6_temp2 & _11111110B);
}

/*****************************************************************************
    I/Oポート イニシャライズ
*****************************************************************************/
void init_io(void)
{
    IO.PMR1.BYTE     =   0x00;    //ポート1,汎用入出力ポート
    IO.PUCR1.BYTE    =   0xff;    //ポート1,内蔵プルアップオン
    IO.PCR1          =   0x00;    //ポート1,P10-P17入力

    IO.PMR5.BYTE     =   0x00;    //ポート5,汎用入出力ポート
    IO.PUCR5.BYTE    =   0xff;    //ポート5,内蔵プルアップオン
    IO.PCR5          =   0x00;    //ポート5,P50-P57入力

    IO.PCR6          =   0xff;    //ポート6,P60-P67出力
    IO.PDR6.BYTE     =   0xfe;    //ポート6,初期出力設定
}

/*****************************************************************************
    タイマV イニシャライズ
```

```
************************************************************************/
void init_tmv(void)
{
    TV.TCSRV.BYTE    =   0x00;    //TOMV端子は使わない
    TV.TCORA         =   156;     //周期=2ms(156/156.25KHz=1ms)
    TV.TCRV1.BYTE    =   0x01;    //TRGVトリガ入力禁止,
    TV.TCRV0.BYTE    =   0x4b;    //コンペアマッチA 割込みイネーブル
                                  //コンペアマッチA でTCNTVクリア
                                  //内部クロックφ/128(20MHz/128=156.25KHz)
}

/************************************************************************
    タイマV 割込み(1ms)
************************************************************************/
#pragma regsave (intprog_tmv)
void intprog_tmv(void)
{
    //コンペアマッチフラグA クリア
    TV.TCSRV.BIT.CMFA = 0;

    //スイッチ入力
    switch_in();
}

/************************************************************************
    スイッチ入力
************************************************************************/
void switch_in(void)
{
    switch(SwStatus){
        case 0:
            SwData1 = (~IO.PDR1.BYTE & _10100000B) | (~IO.PDR5.BYTE & _00001010B);
            if (SwData1!=0)  {SwStatus = 1;}
            else             {SwData2 = SwData3 =0;}
            break;
        case 1:
            if (SwData1==((~IO.PDR1.BYTE & _10100000B) | (~IO.PDR5.BYTE & _00001010B))){
                SwData2 = SwData1;
                SwData4 = SwData4 | (SwData2 & (~SwData3));
                SwData3 = SwData2;
            }
            SwStatus = 0;
            break;
    }
}

/************************************************************************
    ID番号セット
************************************************************************/
void id_set(void)
{
    if ((start_check()==-1)||(IDNo==0)){
        if      (IO.PDR1.BIT.B7==0)  {IDNo = _10000000B;}
        else if (IO.PDR1.BIT.B5==0)  {IDNo = _01000000B;}
        else if (IO.PDR5.BIT.B3==0)  {IDNo = _00100000B;}
        else if (IO.PDR5.BIT.B1==0)  {IDNo = _00010000B;}
        else                         {IDNo = _00000000B;}
    }
```

```
}

/************************************************************************
    スタートチェック
************************************************************************/
char start_check(void)
{
    char i,j;

    for(i=0; i<32; i++){
        if (StartChkData[i]=!StartChkCnstData[i]){
            for(j=0; j<32; j++) {StartChkData[j] = StartChkCnstData[j];}
            return -1;   //コールドスタート
        }
    }

    return 0;    //ウォームスタート
}
```

このプログラムは割込みを使っているので,「intprg. c」を追加・修正します。

```
/************************************************************************/
/*                                                                      */
/*  FILE        :intprg.c                                               */
/*  DATE        :Wed, Sep 09, 2009                                      */
/*  DESCRIPTION :Interrupt Program                                      */
/*  CPU TYPE    :H8/3687                                                */
/*                                                                      */
/*  This file is generated by Renesas Project Generator (Ver.4.9).      */
/*                                                                      */
/************************************************************************/



#include <machine.h>

extern void intprog_tmv(void);

#pragma section IntPRG
//  vector 1 Reserved

//  vector 2 Reserved

//  vector 3 Reserved

//  vector 4 Reserved

//  vector 5 Reserved

//  vector 6 Reserved

//  vector 7 NMI
__interrupt(vect=7) void INT_NMI(void) {/* sleep(); */}
//  vector 8 TRAP #0
__interrupt(vect=8) void INT_TRAP0(void) {/* sleep(); */}
//  vector 9 TRAP #1
```

```
__interrupt(vect=9) void INT_TRAP1(void) {/* sleep(); */}
//  vector 10 TRAP #2
__interrupt(vect=10) void INT_TRAP2(void) {/* sleep(); */}
//  vector 11 TRAP #3
__interrupt(vect=11) void INT_TRAP3(void) {/* sleep(); */}
//  vector 12 Address break
__interrupt(vect=12) void INT_ABRK(void) {/* sleep(); */}
//  vector 13 SLEEP
__interrupt(vect=13) void INT_SLEEP(void) {/* sleep(); */}
//  vector 14 IRQ0
__interrupt(vect=14) void INT_IRQ0(void) {/* sleep(); */}
//  vector 15 IRQ1
__interrupt(vect=15) void INT_IRQ1(void) {/* sleep(); */}
//  vector 16 IRQ2
__interrupt(vect=16) void INT_IRQ2(void) {/* sleep(); */}
//  vector 17 IRQ3
__interrupt(vect=17) void INT_IRQ3(void) {/* sleep(); */}
//  vector 18 WKP
__interrupt(vect=18) void INT_WKP(void) {/* sleep(); */}
//  vector 19 RTC
__interrupt(vect=19) void INT_RTC(void) {/* sleep(); */}
//  vector 20 Reserved

//  vector 21 Reserved

//  vector 22 Timer V
__interrupt(vect=22) void INT_TimerV(void) {intprog_tmv();}
//  vector 23 SCI3
__interrupt(vect=23) void INT_SCI3(void) {/* sleep(); */}
//  vector 24 IIC2
__interrupt(vect=24) void INT_IIC2(void) {/* sleep(); */}
//  vector 25 ADI
__interrupt(vect=25) void INT_ADI(void) {/* sleep(); */}
//  vector 26 Timer Z0
__interrupt(vect=26) void INT_TimerZ0(void) {/* sleep(); */}
//  vector 27 Timer Z1
__interrupt(vect=27) void INT_TimerZ1(void) {/* sleep(); */}
//  vector 28 Reserved

//  vector 29 Timer B1
__interrupt(vect=29) void INT_TimerB1(void) {/* sleep(); */}
//  vector 30 Reserved

//  vector 31 Reserved

//  vector 32 SCI3_2
__interrupt(vect=32) void INT_SCI3_2(void) {/* sleep(); */}
```

このプログラムは赤外線リモコン信号出力サブルーチンをアセンブラで作りました。下記にソースリストを掲載しますが，アセンブラのプログラムはフローチャートにすると，処理の内容をより理解しやすくなります(プログラムの学習にも最適です)。がんばってフローチャートに直してみてください。ちなみに実際のプログラムのときは，フローチャートを書いてからソースリストにコーディングします。

```
;-----------------------------------------------------------------------
;                                                                      |
;   FILE        :asmprg.src                                            |
```

```
;   DATE        :Wed, Sep 09, 2009                                      |
;   DESCRIPTION :Sub Program                                            |
;   CPU TYPE    :H8/3687                                                |
;                                                                       |
;   This file is programed by TOYO-LINX Co.,Ltd. / yKikuchi             |
;                                                                       |
;-----------------------------------------------------------------------

    .export     _IR_sig_out     ;Cからコールされる,"IR_sig_out(N);"
    .section    P,CODE,ALIGN=2

;***********************************************************************
;   定数定義
;***********************************************************************
PDR6    .equ    h'FFD9  ;ﾎﾟｰﾄﾃﾞｰﾀﾚｼﾞｽﾀ 6
PCR6    .equ    h'FFE9  ;ﾎﾟｰﾄｺﾝﾄﾛｰﾙﾚｼﾞｽﾀ 6

;***********************************************************************
;   赤外線リモコン信号出力（NECフォーマット準拠）
;-----------------------------------------------------------------------
;   ROL:リモコン信号データ
;***********************************************************************
_IR_sig_out:
    push.l  er2     ;ER0,ER1はCからコールした段階で自動的にPUSHされる
    push.l  er3
    push.l  er4
    push.l  er5
    push.l  er6

    mov.b   r0l,@IRBuf_2        ;データコード
    not.b   r0l                 ;データコードの反転
    mov.b   r0l,@IRBuf_3
    mov.b   #h'00,r0l           ;カスタムコード-0
    mov.b   r0l,@IRBuf_0
    mov.b   #h'ff,r0l           ;カスタムコード-1
    mov.b   r0l,@IRBuf_1

    mov.l   #IRBuf_0,er1
    mov.w   #4,r2
    bsr     IR_signal_out:16

    pop.l   er6
    pop.l   er5
    pop.l   er4
    pop.l   er3
    pop.l   er2     ;ER0,ER1はCにリターンするとき自動的にPOPされる
    rts

;***********************************************************************
;   赤外線リモコン信号出力
;***********************************************************************
IR_signal_out:
    push.l  er1
    push.l  er2

    orc     #h'80,ccr           ;割込み禁止

    bsr     IR_leader:16    ;リーダ
```

```
IR_signal_out_01:
    mov.b   @er1,r3l
    mov.b   #8,r3h
IR_signal_out_02:
    rotr.b  r3l
    bcs     IR_signal_out_03
    bsr     IR_bit0:16
    bra     IR_signal_out_04
IR_signal_out_03:
    bsr     IR_bit1:16
IR_signal_out_04:
    dec.b   r3h
    bne     IR_signal_out_02
    inc.l   #1,er1
    dec.w   #1,r2
    bne     IR_signal_out_01

    bsr     IR_trailer:16       ;トレーラ

    andc    #h'7f,ccr           ;割込み許可

    pop.l   er2
    pop.l   er1
    rts

;*************************************************************************
;   IR / 38KHzのパルス（1周期）
;*************************************************************************
IR_pulse_38khz:
    mov.b   @PDR6,r0l
    or.b    #h'01,r0l
    mov.b   r0l,@PDR6

    mov.b   #42,r0l
IR_pulse_38khz_01:
    dec.b   r0l
    bne     IR_pulse_38khz_01

    mov.b   @PDR6,r0l
    and.b   #h'fe,r0l
    mov.b   r0l,@PDR6

    mov.b   #39,r0l
IR_pulse_38khz_02:
    dec.b   r0l
    bne     IR_pulse_38khz_02

    rts

;*************************************************************************
;   IR / 38KHzのパルス（1周期）と同じ時間Low
;*************************************************************************
IR_non_pulse_38khz:
    mov.b   @PDR6,r0l
    and.b   #h'fe,r0l
    mov.b   r0l,@PDR6
```

```
    mov.b   #42,r0l
IR_non_pulse_38khz_01:
    dec.b   r0l
    bne     IR_non_pulse_38khz_01

    mov.b   @PDR6,r0l
    and.b   #h'fe,r0l
    mov.b   r0l,@PDR6

    mov.b   #39,r0l
IR_non_pulse_38khz_02:
    dec.b   r0l
    bne     IR_non_pulse_38khz_02

    rts

;*************************************************************************
;   IR / 赤外線信号 bit=0
;*************************************************************************
IR_bit0:
    mov.b   #21,r0h
IR_bit0_01:
    bsr     IR_pulse_38khz
    dec.b   r0h
    bne     IR_bit0_01

    mov.b   #21,r0h
IR_bit0_02:
    bsr     IR_non_pulse_38khz
    dec.b   r0h
    bne     IR_bit0_02

    rts

;*************************************************************************
;   IR / 赤外線信号 bit=1
;*************************************************************************
IR_bit1:
    mov.b   #21,r0h
IR_bit1_01:
    bsr     IR_pulse_38khz
    dec.b   r0h
    bne     IR_bit1_01

    mov.b   #64,r0h
IR_bit1_02:
    bsr     IR_non_pulse_38khz
    dec.b   r0h
    bne     IR_bit1_02

    rts

;*************************************************************************
;   IR / リーダ部
;*************************************************************************
IR_leader:
    mov.b   #171,r0h
IR_leader_01:
```

```
    bsr     IR_pulse_38khz
    dec.b   r0h
    bne     IR_leader_01

    mov.b   #171,r0h
IR_leader_03:
    bsr     IR_pulse_38khz
    dec.b   r0h
    bne     IR_leader_03

    mov.b   #171,r0h
IR_leader_02:
    bsr     IR_non_pulse_38khz
    dec.b   r0h
    bne     IR_leader_02

    rts

;*************************************************************************
;   IR / トレーラ部
;*************************************************************************
IR_trailer:
    mov.b   #21,r0h
IR_trailer_01:
    bsr     IR_pulse_38khz
    dec.b   r0h
    bne     IR_trailer_01

    mov.l   #133333,er0         ;40ms
IR_trailer_02:
    dec.l   #1,er0
    bne     IR_trailer_02

    rts

;=========================================================================
;       ワークエリア
;=========================================================================
        .section B,data,ALIGN=2

IRBuf_0     .res.b      1   ;カスタムコード-0
IRBuf_1     .res.b      1   ;カスタムコード-1
IRBuf_2     .res.b      1   ;データコード
IRBuf_3     .res.b      1   ;データコードの反転

    .end
```

第7章 赤外リモコン受信部のモジュール化

7-1. モジュール化する理由

前の章まで，赤外リモコンの受信プログラムは H8/3687 で実行していました。受信プログラムの原理を見ると分かるように，プログラムの実行時間を使ってリーダコードやビット 0/1 を判定しています。その結果，受信プログラム実行中は（判定ミスを防ぐため）ほかの処理を行なうわけにはいきません。実際，アセンブラルーチン「_IR_rcv」には次の処理がしてあります。

```
;*************************************************************************
;    赤外線リモコン信号受信（NECフォーマット準拠）
;-------------------------------------------------------------------------
;    戻り値
;    R0 = 00xx -> 受信OK,xx=受信コマンド
;    R0 = FFFF -> 受信NG
;*************************************************************************
_IR_rcv:
    stc.b    ccr,r1l                ;割り込みマスク
    push.l   er1
    ldc.b    #b'10000000,ccr

        .
        .
        .
        .

    pop.l    er1
    and.b    #B'10000000,r1l
    beq      IR_rcv_EXIT_01
    bra      IR_rcv_EXIT_02
IR_rcv_EXIT_01:
    andc     #B'01111111,ccr
IR_rcv_EXIT_02:

    rts
```

これは，サブルーチン内では割り込みをマスクして受信処理のみを行なうようにし，リターンする前に割り込みマスクの状態をコール前の状態に戻す処理です。この結果，受信中（NEC フォーマットの 1 フレーム 108ms のうち，68ms の間）は受信処理以外何も行なうことができません。赤外リモコンカーを走らせるだけなら問題にはなりませんが，さらに仕事を追加して応用しようとするとリアルタイム性が損なわれます。

そこで，より小さなマイコンに受信処理を任せてしまい，受信結果だけを H8/3687 が受け取るようにします。今回は，マイクロチップテクノロジの PIC12F683 を利用することにしました。パッケージが 8 ピン DIP で，外付けの部品も必要ないため，このような用途にはぴったりです。

7-2. PIC のプログラム

付属の PIC12F683 には出荷時に，NEC フォーマットの赤外リモコン送信機に対応した受信プログラムが書き込まれています。それで，すぐに利用することができます。赤外リモコンカーを組み立てたときに 8 ピンの IC ソケットのみ実装しました。そこに PIC12F683 を挿して下さい。

PIC12F683 に書き込まれているプログラムは，受信した赤外リモコン信号のデータ部を取り出し，そのデータを調歩同期式のシリアル信号で送信するものです。アセンブラで作りました。参考までにソースリストを掲載しますが，アセンブラのプログラムはフローチャートにすると，処理の内容をより理解しやすくなります（プログラムの学習にも最適です）。がんばってフローチャートに直してみてください。ちなみに実際のプログラムのときは，フローチャートを書いてからソースリストにコーディングします。

```
;-----------------------------------------------------------------------
;                                                                      |
;   FILE        :ir_rcv.asm                                            |
;   DATE        :Wed, Nov 26, 2008                                     |
;   DESCRIPTION :Main Program                                          |
;   CPU TYPE    :PIC12F683                                             |
;                                                                      |
;   This file is programed by TOYO-LINX Co.,Ltd. / yKikuchi            |
;                                                                      |
;-----------------------------------------------------------------------

    list      p=12f683
    #include <p12f683.inc>

    __CONFIG _FCMEN_ON & _IESO_ON & _BOD_ON & _CPD_OFF & _CP_OFF & _MCLRE_OFF & _PWRTE_ON & _WDT_OFF &
_INTRC_OSC_NOCLKOUT

    RADIX   DEC

;***********************************************************************
;   Constant Diffinition
;***********************************************************************
IR      equ     GP2     ;IR Signal Bit
TIMING  equ     GP0     ;Timig Check Signal

;***********************************************************************
;   Data Memory
;***********************************************************************
    UDATA   20h
ircmd       RES     1   ;IR Command
ircode      RES     1   ;IR Signal Code
ircnt       RES     1   ;IR Receive Counter

txdata      RES     1   ;Transimt Data
txcnt       RES     1   ;Transmit Counter
tx_tmp      RES     1   ;Test Program Data

wait_tmp    RES     1   ;Wait Counter
wait_tmp2   RES     1   ;Wait Counter No.2

;***********************************************************************
;   Program Memory
;***********************************************************************
    ORG 0000h
```

```
    GOTO    start

    ORG 0004h

;*************************************************************************
;   Initialize
;*************************************************************************
    ORG 0010h
start
    BSF     STATUS,RP0   ;Bank=1
    MOVLW   01111000b    ;Internal OSC 8MHz
    MOVWF   OSCCON
    BCF     STATUS,RP0   ;Bank=0
    MOVLW   000010b      ;GPIO Initial Out
    MOVWF   GPIO
    MOVLW   07h          ;Comparator Off(Use I/O Pin)
    MOVWF   CMCON0
    BSF     STATUS,RP0   ;Bank=1
    CLRF    ANSEL        ;Analog Input Off(Use I/O Pin)
    MOVLW   111100b      ;GP2,3,4,5=Input / GP0,1=Output
    MOVWF   TRISIO
    MOVLW   01111111b    ;GPIO Pull Up On
    MOVWF   OPTION_REG
    BCF     STATUS,RP0   ;Bank=0

;*************************************************************************
;   Main Program
;*************************************************************************
main
    CLRWDT

    BTFSC   GPIO,IR      ;Leader Code
    GOTO    main
    CALL    check_leader
    SUBLW   01h
    BTFSS   STATUS,Z
    GOTO    main

    CALL    getcode      ;Custom Code(1)
    MOVLW   000h
    SUBWF   ircode,W
    BTFSS   STATUS,Z
    GOTO    main

    CALL    getcode      ;Custom Code(2)
    MOVLW   0ffh
    SUBWF   ircode,W
    BTFSS   STATUS,Z
    GOTO    main

    CALL    getcode      ;Data
    MOVF    ircode,W
    MOVWF   ircmd

    CALL    getcode      ;Data(Compliment)
    COMF    ircode,W
    SUBWF   ircmd,W
    BTFSS   STATUS,Z
```

```
    GOTO    main

    MOVF    ircmd,W     ;Transmit Command Data
    MOVWF   txdata
    CALL    txone

    GOTO    main

;***************************************************************************
;   Check IR Leader Signal
;***************************************************************************
check_leader
    MOVLW   250
    MOVWF   wait_tmp
check_leader_01
    BTFSC   GPIO,IR
    GOTO    check_leader_40
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    DECFSZ  wait_tmp,F
    GOTO    check_leader_01

    CLRF    wait_tmp
check_leader_11
    BTFSC   GPIO,IR
    GOTO    check_leader_21
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
```

```
    GOTO    $+1
    GOTO    $+1
    DECFSZ  wait_tmp,F
    GOTO    check_leader_11
    GOTO    check_leader_40

check_leader_21
    MOVLW   250
    MOVWF   wait_tmp
check_leader_22
    BTFSS   GPIO,IR
    GOTO    check_leader_40
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    DECFSZ  wait_tmp,F
    GOTO    check_leader_22

    CLRF    wait_tmp
check_leader_31
    BTFSS   GPIO,IR
    RETLW   01h         ;IR Leader Signal Get
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    DECFSZ  wait_tmp,F
    GOTO    check_leader_31
    GOTO    check_leader_40

check_leader_40
    RETLW   00h         ;IR Leader Signal NG

;*********************************************************************
;   Get IR Signal Code
;*********************************************************************
getcode
    MOVLW   8
    MOVWF   ircnt
    CLRF    ircode

getcode_01
    BTFSS   GPIO,IR
    GOTO    $-1

    BSF     GPIO,TIMING
    CALL    wait15bit
    BCF     GPIO,TIMING
```

```
    BTFSC   GPIO,IR
    GOTO    getcode_03
    BCF     STATUS,C
    RRF     ircode,F
getcode_02
    DECFSZ  ircnt,F ; 12
    GOTO    getcode_01
    RETURN
getcode_03
    BSF     STATUS,C
    RRF     ircode,F

    BTFSC   GPIO,IR
    GOTO    $-1
    GOTO    getcode_02

;*************************************************************************
;   Wait 1.5bit(IR Signal)
;*************************************************************************
wait15bit
    MOVLW   209         ;IR Signal 1.5bit(0.84ms)
    MOVWF   wait_tmp
wait15bit_01
    CLRWDT
    GOTO    $+1
    GOTO    $+1
    DECFSZ  wait_tmp,F
    GOTO    wait15bit_01
    RETURN

;*************************************************************************
;   Transmit 1-Charactor
;*************************************************************************
TXD     equ     GP1     ;TXD Bit

txone
    MOVLW   8
    MOVWF   txcnt

    BCF     GPIO,TXD;Start Bit
    CALL    wait1bit

txone_01
    RRF     txdata,F;Data Bit
    BTFSC   STATUS,C
    GOTO    txone_02
    BCF     GPIO,TXD
    GOTO    txone_03
txone_02
    BSF     GPIO,TXD
    NOP
txone_03
    CALL    wait1bit
    DECFSZ  txcnt,F
    GOTO    txone_01

    BSF     GPIO,TXD;Stop Bit(2bit)
    CALL    wait1bit
```

```
    CALL    wait1bit

    RETURN

;*******************************************************************
;   Wait 1-bit
;*******************************************************************
wait1bit
    MOVLW   10          ;38400 baud
    MOVWF   wait_tmp
wait1bit_01
    CLRWDT
    DECFSZ  wait_tmp,F
    GOTO    wait1bit_01
    RETURN

;*******************************************************************
;   Wait Timer
;*******************************************************************
wait1ms
    MOVLW   249         ;1ms
    MOVWF   wait_tmp
wait1ms_01
    CLRWDT
    GOTO    $+1
    GOTO    $+1
    DECFSZ  wait_tmp,F
    GOTO    wait1ms_01
    RETURN

wait100ms
    MOVLW   100         ;100ms
wait100ms_00
    MOVWF   wait_tmp2
wait100ms_01
    CALL    wait1ms
    DECFSZ  wait_tmp2,F
    GOTO    wait100ms_01
    RETURN

wait200ms
    MOVLW   200         ;200ms
    GOTO    wait100ms_00

;*******************************************************************
;   Test Program 1
;*******************************************************************
test01
    BSF     GPIO,1
    BCF     GPIO,1
    GOTO    test01

;*******************************************************************
;   Test Program 2
;*******************************************************************
test02
    MOVLW   020h
    MOVWF   tx_tmp
```

```
test02_01
    MOVF    tx_tmp,W
    MOVWF   txdata
    call    txone
    INCF    tx_tmp,F
    BTFSS   tx_tmp,6
    GOTO    test02_01
    MOVLW   0dh
    MOVWF   txdata
    CALL    txone
test02_02
    MOVF    tx_tmp,W
    MOVWF   txdata
    call    txone
    INCF    tx_tmp,F
    BTFSS   tx_tmp,7
    GOTO    test02_02
    MOVLW   0dh
    MOVWF   txdata
    CALL    txone
    CALL    wait200ms
    CALL    wait200ms
    CALL    wait100ms
    GOTO    test02

;*******************************************************************************
    END
```

7-3. 赤外リモコンカーのプログラム

TK-3687mini にプログラムを書き込みます。付属 CD の中の **'ir_remocon_car_11. mot'** を, FDT を使って TK-3687mini にダウンロードしてください。FDT の基本的な使い方は「TK-3687mini 組み立て手順書」に記されています。これを参考にして下さい。

このプログラムは PIC12F683 から送られてくる受信したリモコンデータを, シリアルコミュニケーションインターフェース, SCI3_2 で受信します。受信処理は割り込みを使い, 走行と受信の並列処理を行なっています。プログラムのソースリストは次のとおりです。

```
/************************************************************************/
/*                                                                      */
/*  FILE        :ir_remocon_car_11.c                                    */
/*  DATE        :Mon, Sep 07, 2009                                      */
/*  DESCRIPTION :Main Program                                           */
/*  CPU TYPE    :H8/3687                                                */
/*                                                                      */
/*  This file is programed by TOYO-LINX Co.,Ltd. / yKikuchi             */
/*                                                                      */
/************************************************************************/

/************************************************************************
    プログラムの内容
************************************************************************/
/*
2009-09-07 : プログラム開始,赤外リモコン受信をPICで行なう
*/

/************************************************************************
    リモコンスイッチの並び方と赤外線コードの対応
------------------------------------------------------------------------
        +-----+
        | SW4 |
        |     |
        +-----+                  +-----+ +-----+
                                 | SW1 | | SW2 |
                                 |     | |     |
        +-----+                  +-----+ +-----+
        | SW3 |
        |     |
        +-----+
------------------------------------------------------------------------

        +-----+-----+-----+-----+-----+-----+-----+-----+
        |bit7 |bit6 |bit5 |bit4 |bit3 |bit2 |bit1 |bit0 |
        +-----+-----+-----+-----+-----+-----+-----+-----+
        |  0  |  1  |  0  |  0  | SW4 | SW3 | SW2 | SW1 |
        |     |     |     |     |     |     |     |     |
        +-----+-----+-----+-----+-----+-----+-----+-----+
                                  0=Off / 1=On
************************************************************************/

/************************************************************************
    インクルードファイル
************************************************************************/
#include<machine.h>      //H8特有の命令を使う
```

```
#include"iodefine.h"     //内蔵I/Oのラベル定義
#include"binary.h"       //Cで2進数を使うための定義

/************************************************************************
    定数の定義（直接指定）
************************************************************************/
#define     OK          0       //戻り値
#define     NG          -1      //戻り値

//リモコン用,PWMのパーセント ----------------------------------------
//  モータに電力を供給する時間率で指定
#define     TL3LR       20      //左旋回,急,左モータ,%
#define     TL3RR       50      //左旋回,急,右モータ,%
#define     TL2LR       30      //左旋回,中,左モータ,%
#define     TL2RR       50      //左旋回,中,右モータ,%
#define     TL1LR       40      //左旋回,緩,左モータ,%
#define     TL1RR       50      //左旋回,緩,右モータ,%

#define     CW4LR       50      //直進,4速,左モータ,%
#define     CW4RR       50      //直進,4速,右モータ,%
#define     CW3LR       40      //直進,3速,左モータ,%
#define     CW3RR       40      //直進,3速,右モータ,%
#define     CW2LR       30      //直進,2速,左モータ,%
#define     CW2RR       30      //直進,2速,右モータ,%
#define     CW1LR       20      //直進,1速,左モータ,%
#define     CW1RR       20      //直進,1速,右モータ,%

#define     TR1LR       50      //右旋回,緩,左モータ,%
#define     TR1RR       40      //右旋回,緩,右モータ,%
#define     TR2LR       50      //右旋回,中,左モータ,%
#define     TR2RR       30      //右旋回,中,右モータ,%
#define     TR3LR       50      //右旋回,急,左モータ,%
#define     TR3RR       20      //右旋回,急,右モータ,%

//通信 ----------------------------------------------------------------
#define     RXBUF_SIZE  256     //RxBufのサイズ

//ソフトウェアタイマ --------------------------------------------------
#define     T0Const     300     //T0(300ms)  通信が途絶えてから停止するまでの時間（リモコンモード）
#define     T1Const     0       //T1(ms)
#define     T2Const     0       //T2(ms)
#define     T3Const     0       //T3(ms)
#define     T4Const     0       //T4(ms)
#define     T5Const     0       //T5(ms)
#define     T6Const     0       //T6(ms)
#define     T7Const     0       //T7(ms)

/************************************************************************
    定数エリアの定義(ROM)
************************************************************************/

/************************************************************************
    グローバル変数の定義とイニシャライズ(RAM)
************************************************************************/
// 走行に関係した変数 ------------------------------------------------
//左旋回,急,左モータ
unsigned int TurnLeft3L   =   0xffff - ((unsigned long)0xffff * TL3LR / 100);
//左旋回,急,右モータ
```

```
unsigned int TurnLeft3R  =   0xffff - ((unsigned long)0xffff * TL3RR / 100);
//左旋回,中,左モータ
unsigned int TurnLeft2L  =   0xffff - ((unsigned long)0xffff * TL2LR / 100);
//左旋回,中,右モータ
unsigned int TurnLeft2R  =   0xffff - ((unsigned long)0xffff * TL2RR / 100);
//左旋回,緩,左モータ
unsigned int TurnLeft1L  =   0xffff - ((unsigned long)0xffff * TL1LR / 100);
//左旋回,緩,右モータ
unsigned int TurnLeft1R  =   0xffff - ((unsigned long)0xffff * TL1RR / 100);

//直進,4速,左モータ
unsigned int Cw4L        =   0xffff - ((unsigned long)0xffff * CW4LR / 100);
//直進,4速,右モータ
unsigned int Cw4R        =   0xffff - ((unsigned long)0xffff * CW4RR / 100);
//直進,3速,左モータ
unsigned int Cw3L        =   0xffff - ((unsigned long)0xffff * CW3LR / 100);
//直進,3速,右モータ
unsigned int Cw3R        =   0xffff - ((unsigned long)0xffff * CW3RR / 100);
//直進,2速,左モータ
unsigned int Cw2L        =   0xffff - ((unsigned long)0xffff * CW2LR / 100);
//直進,2速,右モータ
unsigned int Cw2R        =   0xffff - ((unsigned long)0xffff * CW2RR / 100);
//直進,1速,左モータ
unsigned int Cw1L        =   0xffff - ((unsigned long)0xffff * CW1LR / 100);
//直進,1速,右モータ
unsigned int Cw1R        =   0xffff - ((unsigned long)0xffff * CW1RR / 100);

//右旋回,緩,左モータ
unsigned int TurnRight1L =   0xffff - ((unsigned long)0xffff * TR1LR / 100);
//右旋回,緩,右モータ
unsigned int TurnRight1R =   0xffff - ((unsigned long)0xffff * TR1RR / 100);
//右旋回,中,左モータ
unsigned int TurnRight2L =   0xffff - ((unsigned long)0xffff * TR2LR / 100);
//右旋回,中,右モータ
unsigned int TurnRight2R =   0xffff - ((unsigned long)0xffff * TR2RR / 100);
//右旋回,急,左モータ
unsigned int TurnRight3L =   0xffff - ((unsigned long)0xffff * TR3LR / 100);
//右旋回,急,右モータ
unsigned int TurnRight3R =   0xffff - ((unsigned long)0xffff * TR3RR / 100);

// 通信に関係した変数 ----------------------------------------------
unsigned char    RxBuf[RXBUF_SIZE];   //受信バッファ
unsigned char    *RxBufWrPnt;         //受信バッファライトポインタ
unsigned char    *RxBufRdPnt;         //受信バッファリードポインタ
unsigned char    *RxBufMin;           //受信バッファの最初
unsigned char    *RxBufMax;           //受信バッファの最後

// ソフトウェアタイマに関係した変数 ------------------------------------
struct SoftTimer{                     //ソフトウェアタイマの構造体タグ
    unsigned char    Status;          //タイマステータス
                                      // 0:停止中(タイマ未使用)
                                      // 1:スタート指令
                                      // 2:カウント中
                                      // 3:カウント終了
    unsigned long    Count;           //タイマカウンタ
};
struct SoftTimer TimT0;               //T0タイマ
struct SoftTimer TimT1;               //T1タイマ
```

```
struct SoftTimer TimT2;              //T2タイマ
struct SoftTimer TimT3;              //T3タイマ
struct SoftTimer TimT4;              //T4タイマ
struct SoftTimer TimT5;              //T5タイマ
struct SoftTimer TimT6;              //T6タイマ
struct SoftTimer TimT7;              //T7タイマ

unsigned long    T0 = T0Const;       //T0タイマカウンタ初期値
unsigned long    T1 = T1Const;       //T1タイマカウンタ初期値
unsigned long    T2 = T2Const;       //T2タイマカウンタ初期値
unsigned long    T3 = T3Const;       //T3タイマカウンタ初期値
unsigned long    T4 = T4Const;       //T4タイマカウンタ初期値
unsigned long    T5 = T5Const;       //T5タイマカウンタ初期値
unsigned long    T6 = T6Const;       //T6タイマカウンタ初期値
unsigned long    T7 = T7Const;       //T7タイマカウンタ初期値

/*************************************************************************
    関数の定義
*************************************************************************/
void        ccw1(void);              //後退,1速
void        ccw2(void);              //後退,2速
void        ccw3(void);              //後退,3速
void        ccw4(void);              //後退,4速
void        cw1(void);               //前進,1速
void        cw2(void);               //前進,2速
void        cw3(void);               //前進,3速
void        cw4(void);               //前進,4速
void        init_io(void);           //IOポートイニシャライズ
void        init_tmb1(void);         //タイマB1イニシャライズ
void        init_tmz(void);          //タイマZイニシャライズ
void        intprog_tmb1(void);      //タイマB1割込み
void        main(void);              //メインプログラム
void        stop(void);              //停止
void        turn_left_1(void);       //左旋回（大）
void        turn_left_2(void);       //左旋回（中）
void        turn_left_3(void);       //左旋回（小）
void        turn_right_1(void);      //右旋回（大）
void        turn_right_2(void);      //右旋回（中）
void        turn_right_3(void);      //右旋回（小）

void        run(void);               //走行制御

void        rev_turn_left_1(void);   //後退左旋回（大）
void        rev_turn_left_2(void);   //後退左旋回（中）
void        rev_turn_left_3(void);   //後退左旋回（小）
void        rev_turn_right_1(void);  //後退右旋回（大）
void        rev_turn_right_2(void);  //後退右旋回（中）
void        rev_turn_right_3(void);  //後退右旋回（小）

void        spin_right_1(void);      //右回転-1
void        spin_right_2(void);      //右回転-2
void        spin_right_3(void);      //右回転-3
void        spin_right_4(void);      //右回転-4
void        spin_left_1(void);       //左回転-1
void        spin_left_2(void);       //左回転-2
void        spin_left_3(void);       //左回転-3
void        spin_left_4(void);       //左回転-4
```

```
int             put_rxbuf(unsigned char);
void            rxerr_sci3_2(void);
void            rxdata_sci3_2(void);
unsigned char   rxone(void);
void            txone(unsigned char);
int             get_rxbuf(unsigned char *);
void            init_rxbuf(void);
void            init_sci3_2(void);
void            intprog_sci3_2(void);

void            init_soft_timer(void);
void            dec_soft_timer(struct SoftTimer *,unsigned long);

/************************************************************************
    メインプログラム
************************************************************************/
void main(void)
{
    // イニシャライズ ---------------------------------------------------
    init_io();
    init_soft_timer();
    init_tmb1();
    init_tmz();
    init_rxbuf();
    init_sci3_2();

    // メインループ -----------------------------------------------------
    while(1){
        run();
    }
}

/************************************************************************
    走行制御
************************************************************************/
void run(void)
{
    unsigned char dt;    //リモコン受信データ

    if (get_rxbuf(&dt)==OK){
        if ((dt & 0xf0)==0x40){
            TimT0.Status = 0;    //タイマストップ
            switch(dt){
                case _01001000B: //前進
                    cw4();
                    break;
                case _01001010B: //前進&右
                    turn_right_3();
                    break;
                case _01001001B: //前進&左
                    turn_left_3();
                    break;
                case _01000100B: //後退
                    ccw4();
                    break;
                case _01000110B: //後退&右
                    rev_turn_right_3();
                    break;
```

```
                case _01000101B: //後退&左
                    rev_turn_left_3();
                    break;
                case _01000010B: //右回転
                    spin_right_3();
                    break;
                case _01000001B: //左回転
                    spin_left_3();
                    break;
                case _01001100B: //前後同時オン→停止
                    stop();
                    break;
                default:
                    stop();      //停止
            }
            TimT0.Status = 1;    //タイマスタート
        }
    }

    if (TimT0.Status==3){    //通信が途絶えた
        TimT0.Status = 0;    //タイマストップ
        stop(); //停止
    }
}

/*************************************************************************
    モータスピードセット
*************************************************************************/
// 前進,1速 ------------------------------------------------------------
void cw1(void)
{
    TZ0.GRC = Cw1L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = Cw1R;
    TZ1.GRD = 0xffff;
}

// 前進,2速 ------------------------------------------------------------
void cw2(void)
{
    TZ0.GRC = Cw2L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = Cw2R;
    TZ1.GRD = 0xffff;
}

// 前進,3速 ------------------------------------------------------------
void cw3(void)
{
    TZ0.GRC = Cw3L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = Cw3R;
    TZ1.GRD = 0xffff;
}

// 前進,4速 ------------------------------------------------------------
void cw4(void)
{
```

```
    TZ0.GRC = Cw4L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = Cw4R;
    TZ1.GRD = 0xffff;
}

// 後退,1速 ------------------------------------------------------------
void ccw1(void)
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = Cw1L;
    TZ1.GRC = 0xffff;
    TZ1.GRD = Cw1R;
}

// 後退,2速 ------------------------------------------------------------
void ccw2(void)
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = Cw2L;
    TZ1.GRC = 0xffff;
    TZ1.GRD = Cw2R;
}

// 後退,3速 ------------------------------------------------------------
void ccw3(void)
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = Cw3L;
    TZ1.GRC = 0xffff;
    TZ1.GRD = Cw3R;
}

// 後退,4速 ------------------------------------------------------------
void ccw4(void)
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = Cw4L;
    TZ1.GRC = 0xffff;
    TZ1.GRD = Cw4R;
}

// 右旋回（小） ---------------------------------------------------------
void turn_right_1(void)
{
    TZ0.GRC = TurnRight1L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = TurnRight1R;
    TZ1.GRD = 0xffff;
}

// 右旋回（中） ---------------------------------------------------------
void turn_right_2(void)
{
    TZ0.GRC = TurnRight2L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = TurnRight2R;
    TZ1.GRD = 0xffff;
```

```
}

// 右旋回（大） ------------------------------------------------------
void turn_right_3(void)
{
    TZ0.GRC = TurnRight3L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = TurnRight3R;
    TZ1.GRD = 0xffff;
}

// 左旋回（小） ------------------------------------------------------
void turn_left_1(void)
{
    TZ0.GRC = TurnLeft1L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = TurnLeft1R;
    TZ1.GRD = 0xffff;
}

// 左旋回（中） ------------------------------------------------------
void turn_left_2(void)
{
    TZ0.GRC = TurnLeft2L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = TurnLeft2R;
    TZ1.GRD = 0xffff;
}

// 左旋回（大） ------------------------------------------------------
void turn_left_3(void)
{
    TZ0.GRC = TurnLeft3L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = TurnLeft3R;
    TZ1.GRD = 0xffff;
}

// 後退右旋回（小） ------------------------------------------------------
void rev_turn_right_1(void)
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = TurnRight1L;
    TZ1.GRC = 0xffff;
    TZ1.GRD = TurnRight1R;
}

// 後退右旋回（中） ------------------------------------------------------
void rev_turn_right_2(void)
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = TurnRight2L;
    TZ1.GRC = 0xffff;
    TZ1.GRD = TurnRight2R;
}

// 後退右旋回（大） ------------------------------------------------------
void rev_turn_right_3(void)
```

```
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = TurnRight3L;
    TZ1.GRC = 0xffff;
    TZ1.GRD = TurnRight3R;
}

// 後退左旋回（小） ------------------------------------------------
void rev_turn_left_1(void)
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = TurnLeft1L;
    TZ1.GRC = 0xffff;
    TZ1.GRD = TurnLeft1R;
}

// 後退左旋回（中） ------------------------------------------------
void rev_turn_left_2(void)
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = TurnLeft2L;
    TZ1.GRC = 0xffff;
    TZ1.GRD = TurnLeft2R;
}

// 後退左旋回（大） ------------------------------------------------
void rev_turn_left_3(void)
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = TurnLeft3L;
    TZ1.GRC = 0xffff;
    TZ1.GRD = TurnLeft3R;
}

// 停止 ----------------------------------------------------------
void stop()
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = 0xffff;
    TZ1.GRC = 0xffff;
    TZ1.GRD = 0xffff;
}

// 右回転-1 ------------------------------------------------------
void spin_right_1(void)
{
    TZ0.GRC = Cw1L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = 0xffff;
    TZ1.GRD = Cw1R;
}

// 右回転-2 ------------------------------------------------------
void spin_right_2(void)
{
    TZ0.GRC = Cw2L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = 0xffff;
```

```
    TZ1.GRD = Cw2R;
}

// 右回転-3 ------------------------------------------------------------
void spin_right_3(void)
{
    TZ0.GRC = Cw3L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = 0xffff;
    TZ1.GRD = Cw3R;
}

// 右回転-4 ------------------------------------------------------------
void spin_right_4(void)
{
    TZ0.GRC = Cw4L;
    TZ0.GRD = 0xffff;
    TZ1.GRC = 0xffff;
    TZ1.GRD = Cw4R;
}

// 左回転-1 ------------------------------------------------------------
void spin_left_1(void)
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = Cw1L;
    TZ1.GRC = Cw1R;
    TZ1.GRD = 0xffff;
}

// 左回転-2 ------------------------------------------------------------
void spin_left_2(void)
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = Cw2L;
    TZ1.GRC = Cw2R;
    TZ1.GRD = 0xffff;
}

// 左回転-3 ------------------------------------------------------------
void spin_left_3(void)
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = Cw3L;
    TZ1.GRC = Cw3R;
    TZ1.GRD = 0xffff;
}

// 左回転-4 ------------------------------------------------------------
void spin_left_4(void)
{
    TZ0.GRC = 0xffff;
    TZ0.GRD = Cw4L;
    TZ1.GRC = Cw4R;
    TZ1.GRD = 0xffff;
}

/*********************************************************************
```

```
    I/Oポート イニシャライズ
************************************************************************/
void init_io(void)
{
    IO.PCR6         =   _00000000B; //ポート6,P60-67入力

    IO.PCR7         =   _00000000B; //ポート7,P77-70入力
}

/************************************************************************
    タイマZ イニシャライズ
************************************************************************/
void init_tmz(void)
{
    TZ.TSTR.BYTE    =   0x00;   //TCNT0,1 停止

    TZ0.TCR.BYTE    =   0x20;   //GRAのコンペアマッチでTCNT=0,φ/1
    TZ1.TCR.BYTE    =   0x20;   //GRAのコンペアマッチでTCNT=0,φ/1

    TZ.TPMR.BYTE    =   0x66;   //FTIOC0,FTIOD0,FTIOC1,FTIOD1 PWMモード
    TZ.TOCR.BYTE    =   0x00;   //初期出力,All Low

    TZ0.POCR.BYTE   =   0x06;   //FTIOC0,FTIOD0 出力ハイアクティブ
    TZ1.POCR.BYTE   =   0x06;   //FTIOC1,FTIOD1 出力ハイアクティブ

    TZ0.GRA         =   0xfffe; //PWM 周期(65534*(1/20MHz)=3.2767ms)
    TZ1.GRA         =   0xfffe; //PWM 周期(65534*(1/20MHz)=3.2767ms)
    TZ0.GRC         =   0xffff; //FTIOC0=Low
    TZ1.GRC         =   0xffff; //FTIOC1=Low
    TZ0.GRD         =   0xffff; //FTIOD0=Low
    TZ1.GRD         =   0xffff; //FTIOD1=Low

    TZ.TOER.BYTE    =   0x33;   //FTIOC0,DTIOD0.FTIOC1,FTIOD1 出力許可

    TZ0.TCNT        =   0x0000; //TCNT0=0
    TZ1.TCNT        =   0x0000; //TCNT1=0
    TZ.TSTR.BYTE    =   0x03;   //TCNT0 カウントスタート
}

/************************************************************************
    タイマB1 イニシャライズ
************************************************************************/
void init_tmb1(void)
{
    TB1.TMB1.BYTE   =   0xf9;   //オートリロード, 内部クロックφ/2048
    TB1.TLB1        =   0-97;   //周期=10ms(100Hz)
    IRR2.BIT.IRRTB1 =   0;      //タイマB1割込み要求フラグ クリア
    IENR2.BIT.IENTB1=   1;      //タイマB1割込み要求イネーブル
}

/************************************************************************
    タイマB1 割込み(10ms)
************************************************************************/
#pragma regsave (intprog_tmb1)
void intprog_tmb1(void)
{
    //タイマB1割込み要求フラグ クリア
    IRR2.BIT.IRRTB1 = 0;
```

```
    //ソフトウェアタイマ T0
    if (TimT0.Status==1 || TimT0.Status==2){
        dec_soft_timer(&TimT0,T0/10);
    }
    //ソフトウェアタイマ T1
    if (TimT1.Status==1 || TimT1.Status==2){
        dec_soft_timer(&TimT1,T1/10);
    }
    //ソフトウェアタイマ T2
    if (TimT2.Status==1 || TimT2.Status==2){
        dec_soft_timer(&TimT2,T2/10);
    }
    //ソフトウェアタイマ T3
    if (TimT3.Status==1 || TimT3.Status==2){
        dec_soft_timer(&TimT3,T3/10);
    }
    //ソフトウェアタイマ T4
    if (TimT4.Status==1 || TimT4.Status==2){
        dec_soft_timer(&TimT4,T4/10);
    }
    //ソフトウェアタイマ T5
    if (TimT5.Status==1 || TimT5.Status==2){
        dec_soft_timer(&TimT5,T5/10);
    }
    //ソフトウェアタイマ T6
    if (TimT6.Status==1 || TimT6.Status==2){
        dec_soft_timer(&TimT6,T6/10);
    }
    //ソフトウェアタイマ T7
    if (TimT7.Status==1 || TimT7.Status==2){
        dec_soft_timer(&TimT7,T7/10);
    }
}

/*************************************************************************
    ソフトウェアタイマのデクリメント
-------------------------------------------------------------------------
    引数    *pst    ソフトウェアタイマ構造体のポインタ
            initial     タイマカウンタの初期値
*************************************************************************/
void dec_soft_timer(struct SoftTimer *pst,unsigned long initial)
{
    if (pst->Status==1){            //タイマスタート指令
        pst->Status =   2;          //カウント中セット
        pst->Count  =   initial;    //タイマカウンタ初期化
    }
    pst->Count--;                   //カウンタ-1
    if (pst->Count==0)              //カウンタが0になった
        pst->Status =   3;          //カウント終了セット
}

/*************************************************************************
    ソフトウェアタイマのイニシャライズ
*************************************************************************/
void init_soft_timer(void)
{
    TimT0.Status = 0; TimT1.Status = 0; TimT2.Status = 0; TimT3.Status = 0;
```

```
    TimT4.Status = 0; TimT5.Status = 0; TimT6.Status = 0; TimT7.Status = 0;
}

/*************************************************************************
    RxBuf の初期化
*************************************************************************/
void init_rxbuf(void)
{
    RxBufRdPnt   =   RxBuf;              //受信バッファリードポインタセット
    RxBufWrPnt   =   RxBufRdPnt;         //受信バッファライトポインタセット
    RxBufMin     =   RxBuf;              //受信バッファの最初をセット
    RxBufMax     =   RxBuf+RXBUF_SIZE-1; //受信バッファの最後をセット
}

/*************************************************************************
    RxBuf にデータを格納する
-------------------------------------------------------------------------
    引数    data    RxBufに格納するデータ
-------------------------------------------------------------------------
    戻り値  OK          格納できた
            NG          エラー,バッファからあふれた
*************************************************************************/
int put_rxbuf(unsigned char data)
{
    int ret_code;    //OK or NG

    if ((RxBufRdPnt==RxBufMin && RxBufWrPnt==RxBufMax) || RxBufWrPnt==RxBufRdPnt-1)
        ret_code = NG;       //バッファサイズを越えた
    else{
        *RxBufWrPnt = data;
        RxBufWrPnt++;
        if (RxBufWrPnt>RxBufMax)
            RxBufWrPnt=RxBufMin; //ライトポインタを先頭に戻す
        ret_code = OK;
    }
    return ret_code;
}

/*************************************************************************
    RxBuf からデータを取り出す
-------------------------------------------------------------------------
    引数    *pd         取り出したデータをセットするポインタ
-------------------------------------------------------------------------
    戻り値  OK          取り出せた
            NG          エラー,バッファに何も入っていない
*************************************************************************/
int get_rxbuf(unsigned char *pd)
{
    int ret_code;

    if (RxBufWrPnt==RxBufRdPnt)
        ret_code = NG;       //バッファに何も入っていない
    else{
        *pd = *RxBufRdPnt;
        RxBufRdPnt++;
        if (RxBufRdPnt>RxBufMax)
            RxBufRdPnt=RxBufMin; //リードポインタを先頭に戻す
        ret_code = OK;
```

```
    }
    return ret_code;
}

/************************************************************************
    SCI3_2 イニシャライズ
************************************************************************/
void init_sci3_2(void)
{
    #define     MHz     20          // Clock=20MHz
    #define     BAUD    38400       // BaudRate
    #define     BITR(MHz*1000000)/BAUD/32-1
    #define     WAIT_1B (MHz*1000000)/6/BAUD

    unsigned long i;

//  IO.PMR1.BIT.TXD2    =   1;          //TxD_2端子イネーブル
    SCI3_2.SCR3.BYTE    =   0x00;       //動作停止
    SCI3_2.SMR .BYTE    =   0x00;       //調歩同期,8bit,NonParity,StopBit=1
    SCI3_2.BRR          =   BITR;       //ﾋﾞｯﾄﾚｰﾄ
    for (i=0;i<WAIT_1B;i++){};          //1bit期間 wait
    SCI3_2.SCR3.BYTE    =   0x50;       //受信イネーブル,受信割り込みイネーブル
}

/************************************************************************
    SCI3_2 割込み（割込み要因によって振り分ける）
************************************************************************/
#pragma regsave (intprog_sci3_2)
void intprog_sci3_2(void)
{
    if (SCI3_2.SSR.BIT.OER==1
     || SCI3_2.SSR.BIT.FER==1
     || SCI3_2.SSR.BIT.PER==1)          rxerr_sci3_2();     //受信エラー
    else if (SCI3_2.SSR.BIT.RDRF==1)    rxdata_sci3_2();    //受信
//  else if (SCI3_2.SSR.BIT.TEND==1)    txend_sci3_2();     //送信終了
//  else if (SCI3_2.SSR.BIT.TDRE==1)    txdata_sci3_2();    //送信
}

/************************************************************************
    SCI3_2 受信エラー割込み
************************************************************************/
void rxerr_sci3_2(void)
{
    unsigned char dmy;

    SCI3_2.SSR.BYTE = SCI3_2.SSR.BYTE & 0x87; //エラーフラグクリア
    dmy = SCI3_2.RDR;                         //ダミーリード
}

/************************************************************************
    SCI3_2 受信割込み
************************************************************************/
void rxdata_sci3_2(void)
{
    put_rxbuf(SCI3_2.RDR);    //データをRxBufにストア
}

/************************************************************************
```

```
    SCI3_2 1文字送信（ポーリング）
------------------------------------------------------------------------
    引数txdata      送信データ
************************************************************************/
void txone(unsigned char txdata)
{
    while(SCI3_2.SSR.BIT.TDRE == 0){}     //送信可能まで待つ
    SCI3_2.TDR = txdata;
}

/************************************************************************
    SCI3_2 1文字受信（ポーリング）
------------------------------------------------------------------------
    戻り値   受信データ
************************************************************************/
unsigned char rxone(void)
{
    while(SCI3_2.SSR.BIT.RDRF == 0){}     //受信するまで待つ
    return SCI3_2.RDR;
}
```

このプログラムは割込みを使っているので,「intprg. c」を追加・修正します。

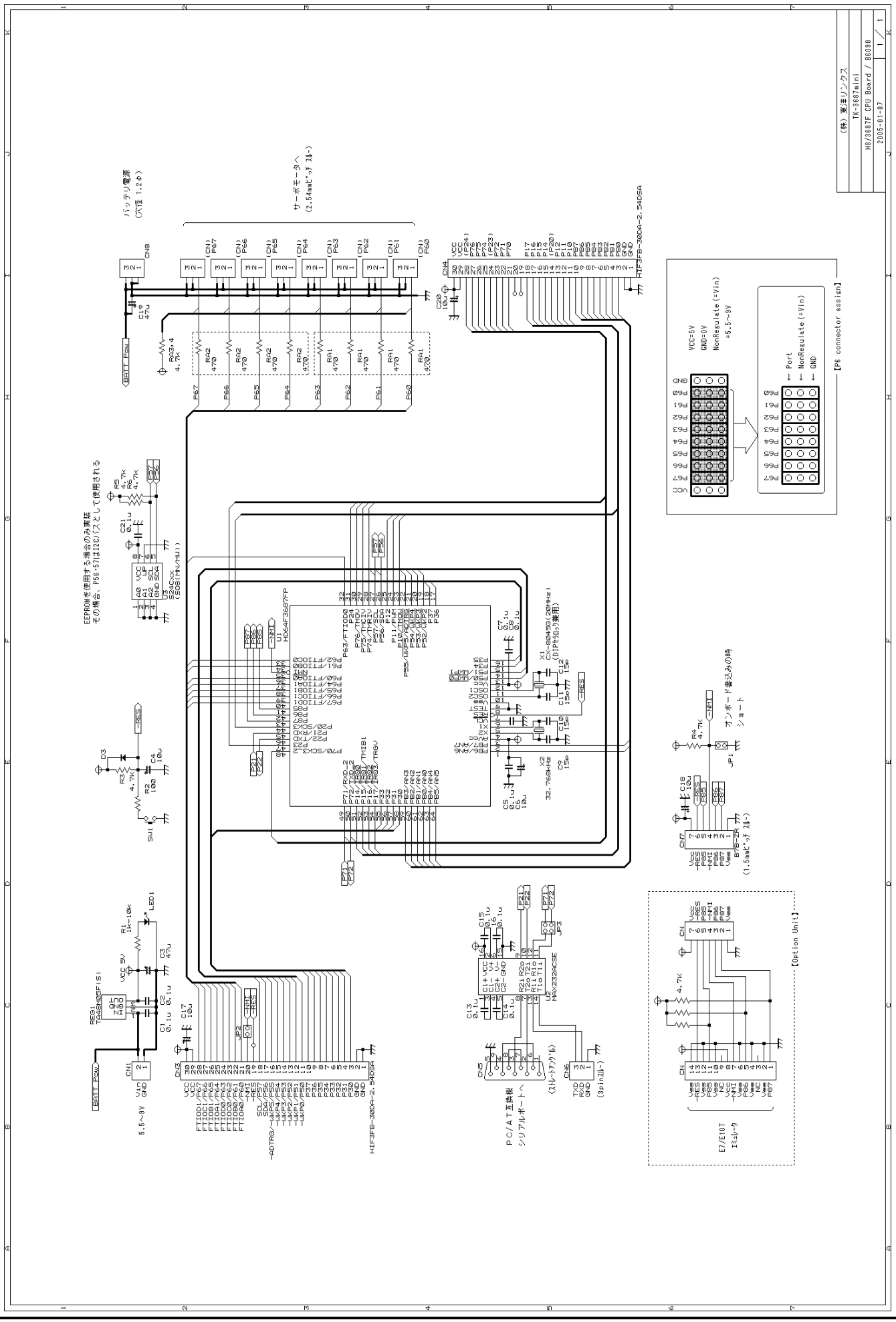
```
/************************************************************************/
/*                                                                      */
/*  FILE        :intprg.c                                               */
/*  DATE        :Mon, Sep 07, 2009                                      */
/*  DESCRIPTION :Interrupt Program                                      */
/*  CPU TYPE    :H8/3687                                                */
/*                                                                      */
/*  This file is generated by Renesas Project Generator (Ver.4.9).      */
/*                                                                      */
/************************************************************************/



#include <machine.h>

extern void intprog_tmb1(void);
extern void intprog_sci3_2(void);

#pragma section IntPRG
//  vector 1 Reserved

//  vector 2 Reserved

//  vector 3 Reserved

//  vector 4 Reserved

//  vector 5 Reserved

//  vector 6 Reserved

//  vector 7 NMI
__interrupt(vect=7) void INT_NMI(void) {/* sleep(); */}
```

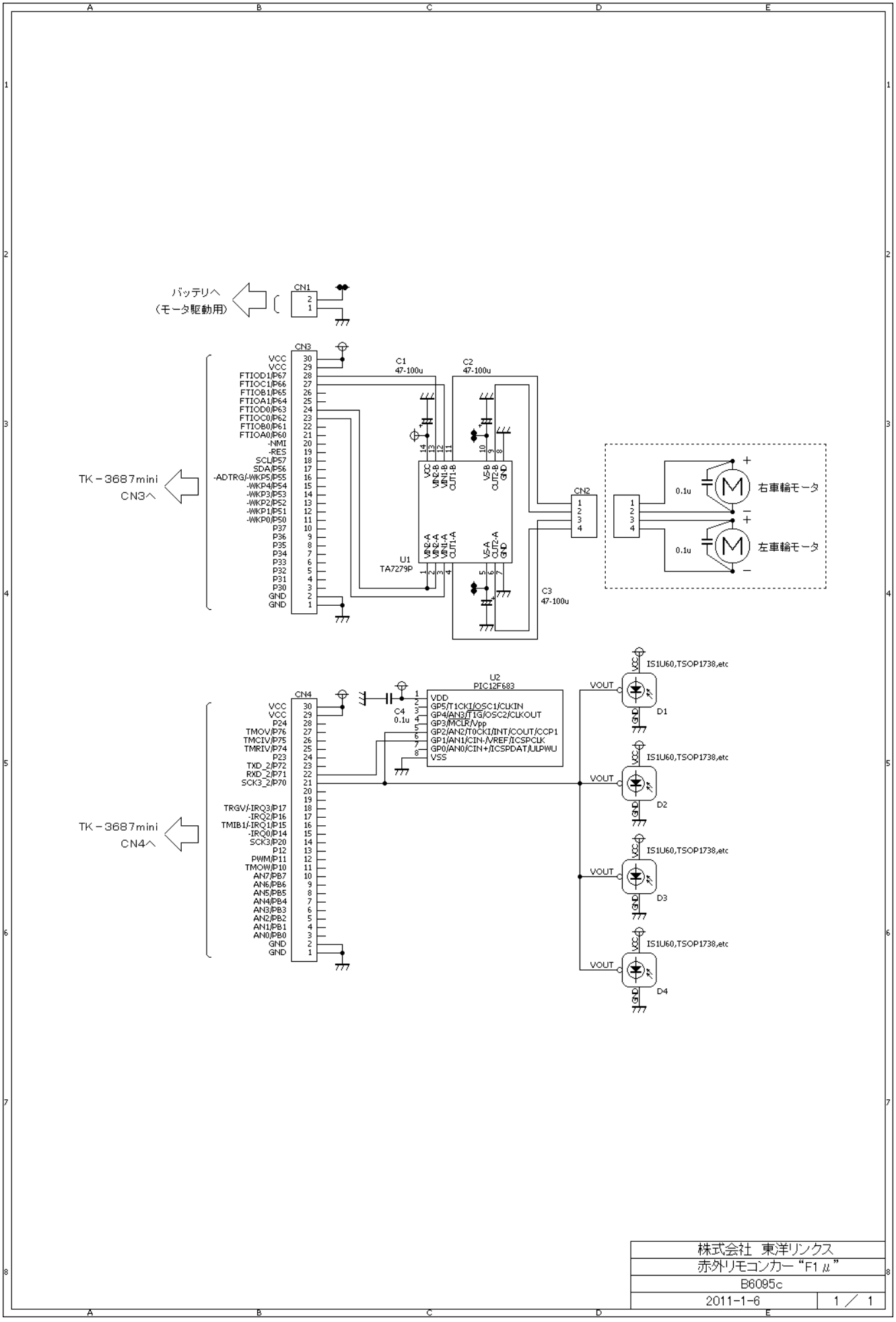
```
//  vector 8 TRAP #0
__interrupt(vect=8) void INT_TRAP0(void) {/* sleep(); */}
//  vector 9 TRAP #1
__interrupt(vect=9) void INT_TRAP1(void) {/* sleep(); */}
//  vector 10 TRAP #2
__interrupt(vect=10) void INT_TRAP2(void) {/* sleep(); */}
//  vector 11 TRAP #3
__interrupt(vect=11) void INT_TRAP3(void) {/* sleep(); */}
//  vector 12 Address break
__interrupt(vect=12) void INT_ABRK(void) {/* sleep(); */}
//  vector 13 SLEEP
__interrupt(vect=13) void INT_SLEEP(void) {/* sleep(); */}
//  vector 14 IRQ0
__interrupt(vect=14) void INT_IRQ0(void) {/* sleep(); */}
//  vector 15 IRQ1
__interrupt(vect=15) void INT_IRQ1(void) {/* sleep(); */}
//  vector 16 IRQ2
__interrupt(vect=16) void INT_IRQ2(void) {/* sleep(); */}
//  vector 17 IRQ3
__interrupt(vect=17) void INT_IRQ3(void) {/* sleep(); */}
//  vector 18 WKP
__interrupt(vect=18) void INT_WKP(void) {/* sleep(); */}
//  vector 19 RTC
__interrupt(vect=19) void INT_RTC(void) {/* sleep(); */}
//  vector 20 Reserved

//  vector 21 Reserved

//  vector 22 Timer V
__interrupt(vect=22) void INT_TimerV(void) {/* sleep(); */}
//  vector 23 SCI3
__interrupt(vect=23) void INT_SCI3(void) {/* sleep(); */}
//  vector 24 IIC2
__interrupt(vect=24) void INT_IIC2(void) {/* sleep(); */}
//  vector 25 ADI
__interrupt(vect=25) void INT_ADI(void) {/* sleep(); */}
//  vector 26 Timer Z0
__interrupt(vect=26) void INT_TimerZ0(void) {/* sleep(); */}
//  vector 27 Timer Z1
__interrupt(vect=27) void INT_TimerZ1(void) {/* sleep(); */}
//  vector 28 Reserved

//  vector 29 Timer B1
__interrupt(vect=29) void INT_TimerB1(void) {intprog_tmb1();}
//  vector 30 Reserved

//  vector 31 Reserved

//  vector 32 SCI3_2
__interrupt(vect=32) void INT_SCI3_2(void) {intprog_sci3_2();}
```

◆

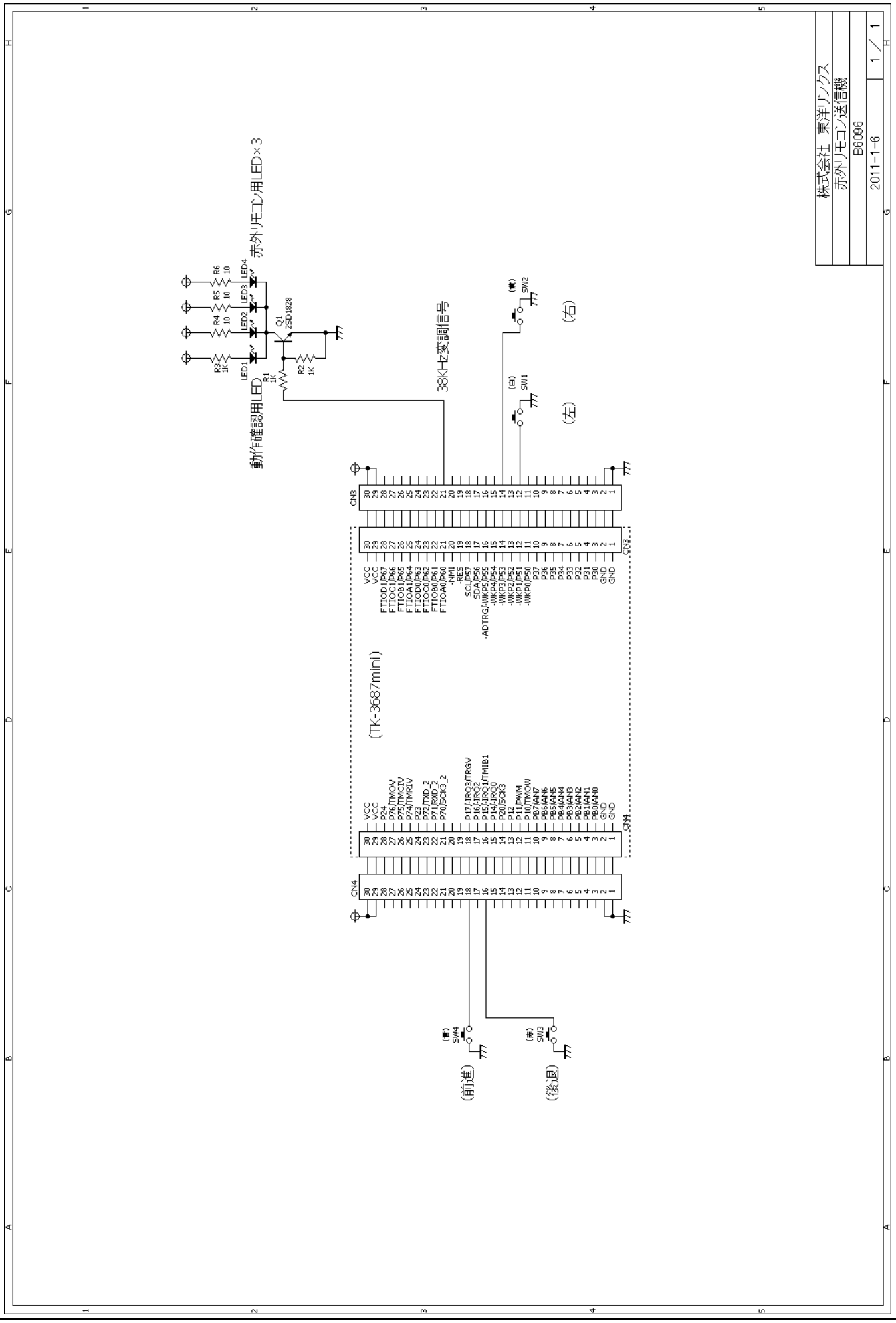
この PIC12F683 を利用した赤外線受信モジュールは，赤外リモコンカーだけでなく，いろいろな用途で使うことができます。ぜひ考えてみてください。（付録の「赤外リモコン受信部のモジュール化 Part-2」もご覧ください）

付録


(株)東洋リンクス
TK-3687mini
H8/3687F CPU Board / B8030
2005-01-07
1 / 1
バッテリ電源
(穴径 1.2φ)
サーボモータへ
(2.54mmピッチ 3ピン)
CN8
CN1
CN2
CN3
CN4
CN5
CN6
CN7
CN9
BATT Pow
C19 47u
C20 10u
RA3 4.7K
RA2 470
RA1 470
P67
P66
P65
P64
P63
P62
P61
P60
HIF3FB-30DA-2.54DSA
VCC=5V
GND=0V
NonRegulate(+Vin)
=5.5~9V
← Port
← NonRegulate(+Vin)
← GND
【P6 connector assign】
EEPROMを使用する場合のみ実装
その場合、P58/SCL1は I2C バスとして使用される
R5 4.7K
R6 4.7K
C21 0.1u
U3
24xCxx
(SO8(MN/MW))
U1
HD64F3687FP
X1
CST4.00MGW(CERAMIC)
C11 15p
C12 15p
C10 15p
C9 15p
X2
32.768kHz
C5 0.1u
C7 0.1u
C8 0.1u
D3
R3 4.7K
R2 100
C4 10u
SW1
-RES
-NMI
R4 4.7K
JP1
オンボード書込みの時
ショート
C18 10u
B7B-ZR
(1.5mmピッチ 7ピン)
REG1
TA48M05F(S)
C1 0.1u
C2 0.1u
C17 10u
R1 1K~10K
LED1
C3 47u
VCC 5V
Vin
GND
5.5~9V
JP2
JP3
C15 0.1u
C16 0.1u
C13 0.1u
C14 0.1u
U2
MAX232ACSE
C1+ C1- V+ V- C2+ C2- GND
T1i T2i R1o R2o R1i R2i Vcc
JP3
CN5
CN6
(3pinヨコ)
TXD
RXD
GND
(2ch トランザム)
PC/AT互換機
シリアルポートへ
【Option Unit】
CN
4.7K
E7/E10T
エミュレータ
H1F3FB-30DA-2.54DSA
FTIOD1/PB5
FTIOD1/PB4
FTIOD0/PB3
FTIOD0/PB2
FTIOD0/PB1
FTIOD0/PB0
-RES
SCL/P57
SDA/P56
-AOTRG/-WKP5/P55
-WKP4/P54
-WKP3/P53
-WKP2/P52
-WKP1/P51
-WKP0/P50
P36
P35
P34
P33
P32
P31
P30
GND
GND

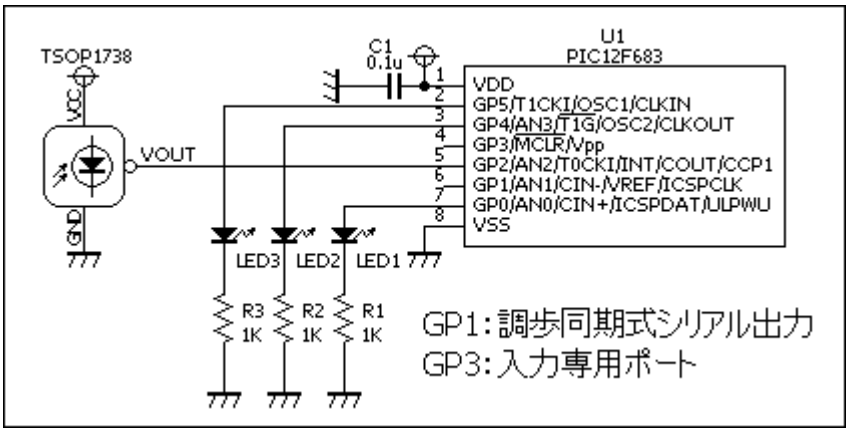


株式会社　東洋リンクス	
赤外リモコンカー“F1μ”	
B6095c	
2011-1-6	1 ／ 1


株式会社 東洋リンクス
赤外リモコン送信機
B6096
2011-1-6
1／1
赤外リモコン用LED×3
LED1
LED2
LED3
LED4
R1 1K
R2 1K
R3 1K
R4 10
R5 10
R6 10
Q1 2SD1828
動作確認用LED
38KHz変調信号
(TK-3687mini)
CN3
CN4
SW1 (白)
SW2 (黒)
SW3 (赤)
SW4 (青)
(左)
(右)
(前進)
(後退)

赤外リモコン受信部のモジュール化 Part-2

本文では PIC12F683 で赤外リモコン信号を受信し，調歩同期式のシリアルデータで TK-3687mini に送信しました。当然のこととして，TK-3687mini のようなマイコンで受信しないといけません。ただ，PIC12F683 には未使用の出力ポートがまだ 3 つあります。ここに，受信データに応じたコード(3 ビット→0～7)を出力すれば，ロジック回路だけでいろいろな応用ができます。次のような回路を考えてみました。(LED は出力のチェック用)



今回使用する赤外リモコンは NATIONAL の VTR に付属していた「VEQ0487」です。解析ツール「LogicAnalyze」でコードを調べてみると次のようになりました。

		早送り	再生	巻戻し	コマ送り	停止	一時停止
リーダコード	オン時間(μs)	3620	3640	3600	3640	3640	3620
	オフ時間(μs)	3300	3300	3340	3320	3280	3340
ビット 0	オン時間(μs)	980	1000	980	980	1000	980
	オフ時間(μs)	700	700	720	700	700	720
ビット 1	オン時間(μs)	980	980	980	980	980	980
	オフ時間(μs)	2480	2460	2480	2460	2460	2460
ストップビット	オン時間(μs)	980	980	980	1000	1000	1000
	オフ時間(μs)	38960	38940	38960	38940	38940	38960
リモコンコード		62 E8 3C	42 E9 35	42 E8 3D	82 E9 33	02 E8 3F	C2 E8 39

本文で作った PIC12F683 用のプログラム「ir_rcv. asm」をもとに改造します。それぞれのスイッチが押されたら(＝対応する赤外リモコンのデータを受信したら)，GP0, GP4, GP5 を次のように出力します。

	押されていない あるいは 右欄以外	早送り	再生	巻戻し	コマ送り	停止	一時停止
リモコンコード	―	62 E8 3C	42 E9 35	42 E8 3D	82 E9 33	02 E8 3F	C2 E8 39
GP0	0	1	0	1	0	1	0
GP4	0	0	1	1	0	0	1
GP5	0	0	0	0	1	1	1

ソースリストは次のようになります(ファイル名「akira_01. asm」)。アセンブラのプログラムはフローチャートにすると，処理の内容をより理解しやすくなります(プログラムの学習にも最適です)。がんばってフローチャートに直してみてください。ちなみに実際のプログラムのときは，フローチャートを書いてからソースリストにコーディングします。

```
;-------------------------------------------------------------------------
;                                                                        |
;   FILE        :akira_01asm                                           |
;   DATE        :Wed, Feb 10, 2010                                     |
;   DESCRIPTION :Main Program                                          |
;   CPU TYPE    :PIC12F683                                             |
;                                                                        |
;   This file is programed by TOYO-LINX Co.,Ltd. / yKikuchi            |
;                                                                        |
;-------------------------------------------------------------------------

;-------------------------------------------------------------------------
;赤外線リモコンデータを受信すると，受信スイッチに応じてLEDを点灯する。
;  リモコン：VEQ0487 (NATIONAL)
;              LED2 LED1 LED0
;    早送り     0    0    1
;    再生       0    1    0
;    巻戻し     0    1    1
;    コマ送り   1    0    0
;    停止       1    0    1
;    一時停止   1    1    0
;
;  I/Oピンの割り当て
;    GP0  LED0
;    GP1  シリアル出力
;    GP2  赤外リモコン入力
;    GP3  未使用,入力専用
;    GP4  LED1
;    GP5  LED2
;-------------------------------------------------------------------------

    list      p=12f683
    #include <p12f683.inc>

    __CONFIG _FCMEN_ON & _IESO_ON & _BOD_ON & _CPD_OFF & _CP_OFF & _MCLRE_OFF & _PWRTE_ON & _WDT_OFF &
_INTRC_OSC_NOCLKOUT

    RADIX   DEC

;*************************************************************************
;   Constant Diffinition
;*************************************************************************
IR      equ     GP2     ;IR Signal Bit
TIMING  equ     GP0     ;Timig Check Signal

;*************************************************************************
;   Data Memory
;*************************************************************************
    UDATA   20h
ircmd1      RES     1   ;IR Command
ircmd2      RES     1   ;IR Command
ircmd3      RES     1   ;IR Command
ircode      RES     1   ;IR Signal Code
```

```
ircnt       RES     1   ;IR Receive Counter

txdata      RES     1   ;Transimt Data
txcnt       RES     1   ;Transmit Counter
tx_tmp      RES     1   ;Test Program Data

wait_tmp    RES     1   ;Wait Counter
wait_tmp2   RES     1   ;Wait Counter No.2

cmdcnt      RES     1   ;Command Counter
tblcnt      RES     1   ;Table Counter

offcnt      RES     2   ;LED Off Counter

;*****************************************************************************
;   Program Memory
;*****************************************************************************
    ORG 0000h
    GOTOstart

    ORG 0004h

;*****************************************************************************
;   Initialize
;*****************************************************************************
    ORG 0010h
start
    BSF     STATUS,RP0  ;Bank=1
    MOVLW   01111000b       ;Internal OSC 8MHz
    MOVWF   OSCCON
    BCF     STATUS,RP0  ;Bank=0
    MOVLW   000010b     ;GPIO Initial Out
    MOVWF   GPIO
    MOVLW   07h         ;Comparator Off(Use I/O Pin)
    MOVWF   CMCON0
    BSF     STATUS,RP0  ;Bank=1
    CLRF    ANSEL       ;Analog Input Off(Use I/O Pin)
    MOVLW   001100b     ;GP2,3=Input / GP0,1,4,5=Output
    MOVWF   TRISIO
    MOVLW   01111111b   ;GPIO Pull Up On
    MOVWF   OPTION_REG
    BCF     STATUS,RP0  ;Bank=0

;*****************************************************************************
;   Main Program
;*****************************************************************************
main
    CLRWDT

    BTFSC   GPIO,IR     ;Leader Code
    GOTO    main_13
    CALL    check_leader
    SUBLW   01h
    BTFSS   STATUS,Z
    GOTO    main_13

    CALL    getcode     ;Data-1
    MOVF    ircode,W
```

```
    MOVWF   ircmd1

    CALL    getcode     ;Data-2
    MOVF    ircode,W
    MOVWF   ircmd2

    CALL    getcode2    ;Data-3
    BCF     STATUS,C
    RRF     ircode,F
    BCF     STATUS,C
    RRF     ircode,F
    MOVF    ircode,W
    MOVWF   ircmd3

main_10
    MOVLW   8
    MOVWF   cmdcnt
main_11
    MOVF    cmdcnt,W
    SUBLW   8
    MOVWF   tblcnt
    BCF     STATUS,C
    RLF     tblcnt,F
    BCF     STATUS,C
    RLF     tblcnt,F

    MOVF    tblcnt,W
    CALL    getcmd
    SUBWF   ircmd1,W
    BTFSS   STATUS,Z
    GOTO    main_12
    INCF    tblcnt,F
    MOVF    tblcnt,W
    CALL    getcmd
    SUBWF   ircmd2,W
    BTFSS   STATUS,Z
    GOTO    main_12
    INCF    tblcnt,F
    MOVF    tblcnt,W
    CALL    getcmd
    SUBWF   ircmd3,W
    BTFSS   STATUS,Z
    GOTO    main_12
    INCF    tblcnt,F
    MOVF    tblcnt,W
    CALL    getcmd
    MOVWF   txdata
    MOVF    GPIO,W
    ANDLW   b'000110'
    IORWF   txdata,W
    MOVWF   GPIO
    CALL    txone

    MOVLW   h'ff'
    MOVWF   offcnt
    MOVLW   h'80'
    MOVWF   offcnt+1
```

```
    GOTO    main

main_12
    DECF    cmdcnt,F
    btfss   STATUS,Z
    GOTO    main_11
main_13
    DECF    offcnt,F
    btfss   STATUS,Z
    GOTO    main
    DECF    offcnt+1,F
    btfss   STATUS,Z
    GOTO    main
    MOVLW   h'ff'
    MOVWF   offcnt
    MOVLW   h'80'
    MOVWF   offcnt+1
    BCF     GPIO,0
    BCF     GPIO,4
    BCF     GPIO,5
    GOTO    main

;-----------------------------------------------------------------------
main_20
    MOVF    ircmd1,W    ;Transmit Command Data-1
    MOVWF   txdata
    CALL    txone
    MOVF    ircmd2,W    ;Transmit Command Data-2
    MOVWF   txdata
    CALL    txone
    MOVF    ircmd3,W    ;Transmit Command Data-3
    MOVWF   txdata
    CALL    txone

    GOTO    main


;************************************************************************
;   Get Command Data
;************************************************************************
getcmd
    ADDWF   PCL,F

    RETLW   h'62'       ;早送り
    RETLW   h'E8'
    RETLW   h'3C'
    RETLW   b'000001'

    RETLW   h'42'       ;再生
    RETLW   h'E9'
    RETLW   h'35'
    RETLW   b'010000'

    RETLW   h'42'       ;巻戻し
    RETLW   h'E8'
    RETLW   h'3D'
    RETLW   b'010001'
```

```
    RETLW   h'82'       ;コマ送り
    RETLW   h'E9'
    RETLW   h'33'
    RETLW   b'100000'

    RETLW   h'02'       ;停止
    RETLW   h'E8'
    RETLW   h'3F'
    RETLW   b'100001'

    RETLW   h'C2'       ;一時停止/静止
    RETLW   h'E8'
    RETLW   h'39'
    RETLW   b'110000'

    RETLW   h'00'       ;
    RETLW   h'00'
    RETLW   h'00'
    RETLW   b'110001'

    RETLW   h'00'       ;
    RETLW   h'00'
    RETLW   h'00'
    RETLW   b'000000'


;*************************************************************************
;   Check IR Leader Signal
;*************************************************************************
check_leader
    MOVLW   100
    MOVWF   wait_tmp
check_leader_01
    BTFSC   GPIO,IR
    GOTO    check_leader_40
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
```

```
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    DECFSZ  wait_tmp,F
    GOTO    check_leader_01

    MOVLW   84
    MOVWF   wait_tmp
check_leader_11
    BTFSC   GPIO,IR
    GOTO    check_leader_21
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    DECFSZ  wait_tmp,F
    GOTO    check_leader_11
    GOTO    check_leader_40

check_leader_21
    MOVLW   237
    MOVWF   wait_tmp
check_leader_22
    BTFSS   GPIO,IR
    GOTO    check_leader_40
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    DECFSZ  wait_tmp,F
    GOTO    check_leader_22

    MOVLW   78
    MOVWF   wait_tmp
check_leader_31
    BTFSS   GPIO,IR
    RETLW   01h          ;IR Leader Signal Get
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    DECFSZ  wait_tmp,F
    GOTO    check_leader_31
    GOTO    check_leader_40
```

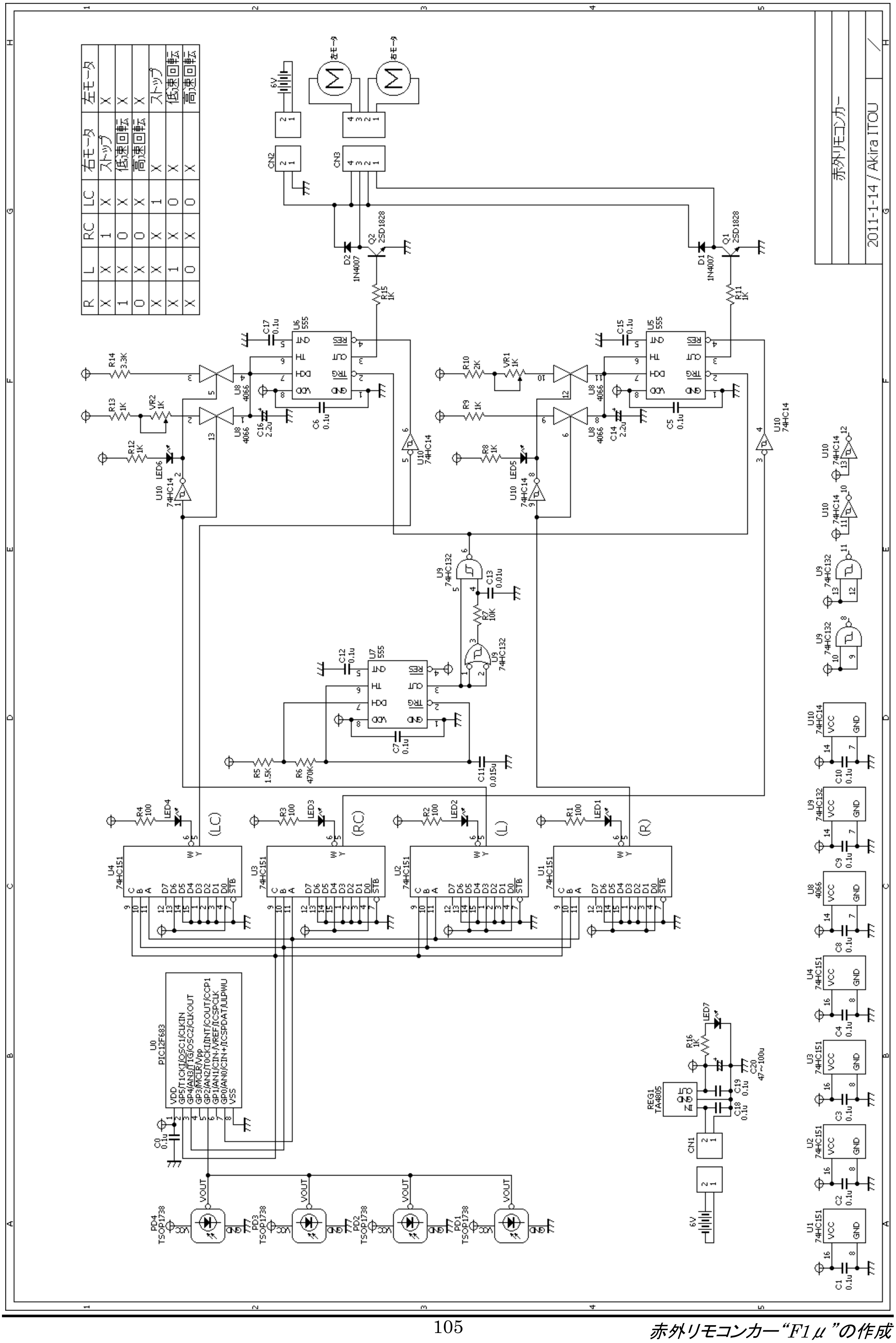
```
check_leader_40
    RETLW   00h         ;IR Leader Signal NG

;*************************************************************************
;   Get IR Signal Code
;*************************************************************************
getcode
    MOVLW   8
getcode_00
    MOVWF   ircnt
    CLRF    ircode

getcode_01
    BTFSS   GPIO,IR
    GOTO    $-1

;   BSF     GPIO,TIMING
    CALL    wait15bit
;   BCF     GPIO,TIMING

    BTFSC   GPIO,IR
    GOTO    getcode_03
    BCF     STATUS,C
    RRF     ircode,F
getcode_02
    DECFSZ  ircnt,F
    GOTO    getcode_01
    RETURN
getcode_03
    BSF     STATUS,C
    RRF     ircode,F

    BTFSC   GPIO,IR
    GOTO    $-1
    GOTO    getcode_02

getcode2
    MOVLW   6
    GOTO    getcode_00

;*************************************************************************
;   Wait 1.5bit(IR Signal)
;*************************************************************************
wait15bit
    MOVLW   120         ;IR Signal 1.5bit(1.2ms)
    MOVWF   wait_tmp
wait15bit_01
    CLRWDT
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    DECFSZ  wait_tmp,F
```

```
    GOTO    wait15bit_01
    RETURN

;*************************************************************************
;   Transmit 1-Charactor
;*************************************************************************
TXD     equ     GP1     ;TXD Bit

txone
    MOVLW   8
    MOVWF   txcnt

    BCF     GPIO,TXD    ;Start Bit
    CALL    wait1bit

txone_01
    RRF     txdata,F    ;Data Bit
    BTFSC   STATUS,C
    GOTO    txone_02
    BCF     GPIO,TXD
    GOTO    txone_03
txone_02
    BSF     GPIO,TXD
    NOP
txone_03
    CALL    wait1bit
    DECFSZ  txcnt,F
    GOTO    txone_01

    BSF     GPIO,TXD    ;Stop Bit(2bit)
    CALL    wait1bit
    CALL    wait1bit

    RETURN

;*************************************************************************
;   Wait 1-bit
;*************************************************************************
wait1bit
    MOVLW   10          ;38400 baud
    MOVWF   wait_tmp
wait1bit_01
    CLRWDT
    DECFSZ  wait_tmp,F
    GOTO    wait1bit_01
    RETURN

;*************************************************************************
;   Wait Timer
;*************************************************************************
wait1ms
    MOVLW   249         ;1ms
    MOVWF   wait_tmp
wait1ms_01
    CLRWDT
    GOTO    $+1
    GOTO    $+1
    DECFSZ  wait_tmp,F
```

```
    GOTO    wait1ms_01
    RETURN

wait100ms
    MOVLW   100         ;100ms
wait100ms_00
    MOVWF   wait_tmp2
wait100ms_01
    CALL    wait1ms
    DECFSZ  wait_tmp2,F
    GOTO    wait100ms_01
    RETURN

wait200ms
    MOVLW   200         ;200ms
    GOTO    wait100ms_00


;*****************************************************************************
    END
```

回路図を次ページに示します。赤外リモコンのスイッチに応じ，赤外リモコンカーは次のように動作します。

赤外リモコンのスイッチ	PIC の出力			74HC151(U1～4)の出力				モータの動作		赤外リモコンカーの動作
	GP5	GP4	GP0	RC	LC	R	L	右モータ	左モータ	
(押さない)	0	0	0	1	1	1	1	ストップ	ストップ	停止
早送り	0	0	1	0	0	1	0	低速回転	高速回転	右旋回
再生	0	1	0	0	0	0	0	高速回転	高速回転	高速直進
巻戻し	0	1	1	0	0	0	1	高速回転	低速回転	左旋回
コマ送り	1	0	0	1	0	0	0	ストップ	高速回転	右急旋回
停止	1	0	1	0	0	1	1	低速回転	低速回転	低速直進
一時停止	1	1	0	0	1	0	0	高速回転	ストップ	左急旋回


R L RC LC 右モータ 左モータ
X X 1 X ストップ X
1 X 0 X 低速回転 X
0 X 0 X 高速回転 X
X X X 1 X ストップ
X 1 X 0 X 低速回転
X 0 X 0 X 高速回転
赤外リモコンカー
2011-1-14 / Akira ITOU

株式会社 東洋リンクス

ユーザサポート係(10:00～17:00, 土日祭は除く)
〒102-0093 東京都千代田区平河町 1-2-2, 朝日ビル
TEL:03-3234-0559 / FAX:03-3234-0549
E-mail : toyolinx@va. u-netsurf. jp
http : //www2. u-netsurf. ne. jp/~toyolinx(最新情報はこちらからダウンロードできます)

20110114