赤外線リモコン送受信器を題材にした 組込みプログラム用電子教材の開発

Development of New Electronic Study Materials for Built-in-System Programming Education Based on Infra-Red Remote-Control Transceiver

玉真昭男*

Teruo TAMAMA*

Abstract: Recently programmers for built-in-systems are badly off. Thus, fostering of those human resources at universities is urgent. New electronic study materials have been developed for built-in-system programming education based on an IR remote-control transceiver. Using those materials, students are able to learn microcomputer programming as well as soldering technique and oscilloscope operation.

1．はじめに

近年、マイコンはデジタル家電、携帯電話、自動車などに数多く組み込まれているが、マイコンを動かすプログラムを開発できる組込みプログラマの不足が深刻な問題となっており、その育成が急務となっている。2005年、経済産業省と情報処理推進機構は「組込みソフトウェアスキル標準」を策定し、広く産業界や大学・高専に技術者・開発者の指導・育成を訴えた。組込みスキル基準は、必要なスキルを明確化・体系化したものであり、組込みソフトウェア開発者の人材育成・活用に有用な「ものさし」を提供する。

本学コンピュータシステム学科でも、それを受けて、H19年から組込みプログラム（ソフトウェア）を教えるカリキュラムを検討・準備し、本22年度より「コンピュータシステム実験」の中で15週×3コマを掛けて集中的に組込みプログラムを教えることになった。

画面の上だけのＰＣ上のプログラムと違い、マイコンのプログラムには「モノが動く」楽しさがある。ＬＥＤが点滅し、モーターが回り、ロボットが動く。「コンピュータシステム実験～組込みプログラム～」は３人の教員が担当し、それぞれ表示系、計測系、通信系の３つのカテゴリのプログラム課題を教えることにした。著者は通信系を担当し、赤外線リモコン送受信器とジャイロセンサーを用いた簡易Wii型リモコンの制御プログラム作成をメインテーマとした。

赤外線リモコン送受信器製作の課題は、マイコンのプログラミングが学べるだけでなく、電子部品のハンダ付けや基板の組立、オシロスコープの操作法などをしっかりと学習出来る内容になっている。その意味で、組込みプログラムを総合的に学習出来る教材になったと自負している。本学では、これを週3コマ（90分／コマ）×5週の実験講座の中で行っている。

赤外線リモコン送受信器を教材にした「組込みプログラム」教育は他所では行われていない。しかし、本稿に示すように、単にマイコンプログラムの演習になるだけでなく、ハンダ付けやオシロスコープの操作法を必然的にじっくりと学習することにもなるので、「組込みプログラム」教材として好適であると考える。演習問題として「学習リモコン」のプログラミングも含んでいるので、プログラミングの課題としても高度であると考えている。

2011年3月4日受理

* 総合情報学部　コンピューシステム学科

2． ハンダ付けの指導

本実験では、試作した赤外線リモコン送受信器のベアボードを学生に組み立てさせる。ハンダ付け未体験の学生に部品点数 25、総ピン数 124 のプリント基板一個のハンダ付けを２人一組でやらせて時間内に完成させさせるには、最初にコツを教え、また丁寧なマニュアルを用意するのが大切と考えた。

2．1 ハンダ付けのコツ

以下のような手順書を書いた。参考のため、一部掲載する。

- コテ置台の黄色いスポンジを水でぬらしておく。
- 温度調節器設定 350℃
- ハンダを適当な長さ(20cm 程度)に切っておく。
- ハンダのコテ先をヤスリで研ぎ、金属色を出す。
- そこにハンダメッキする。「光沢色(銀色)」になることを確認。
- コテ先にハンダが付いて玉状になったら、コテ置台のスポンジでこすって取り去る。(作業中も頻繁に行う)
- 糸ハンダの中にクリーム状のペーストが入っている。ハンダをコテに当てると金属ハンダが溶けるだけでなく、中のペーストも溶けて蒸気が上がる。
  - ➢ この「蒸気が上がっている」間にハンダ付け作業を完了することが重要！！
- Fig. 1 に示すように、コテ先をパッドとリードの両方に当て、7 つ数える。その後、ハンダをコテ先かリード線のどちらか(図中の赤矢印)に当て、ハンダを溶かし、流し込む。
  - ➢ ハンダが溶けて液状になり、「水がしみ渡るようにジワーと広がって行く」感じが良い(Fig. 2)。

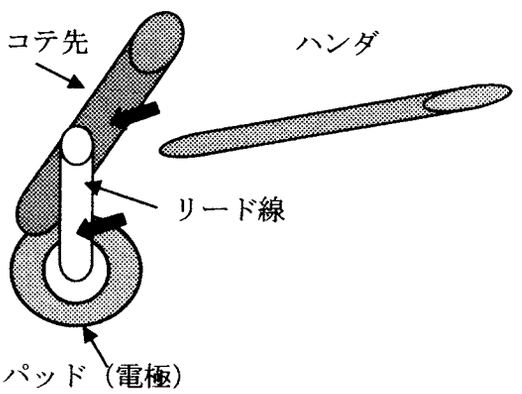

Fig. 1 ハンダ付けのコツ

「富士山型」が理想！
→静岡県民なら「富士山型」を目指せ！

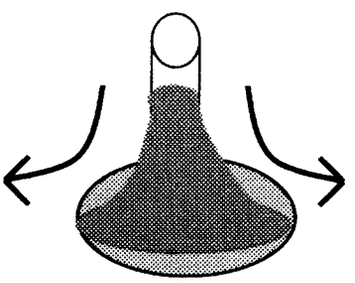

Fig. 2 富士山型についたハンダのイメージ

- 「団子型」はダメ
  - ➢ 時間を掛け過ぎるとハンダが団子状に固まり、中が「す」(空洞) になって、ちゃんと付かない。
  - ➢ 中のペーストの「蒸気が上がっている間」に作業を済ませないと「す」になり易い。

2．2 ハンダ付けの練習

汎用プリント基板を使用し、２行５列の 10 ピンコネクタ１個と抵抗２個（１個横向き、１個縦向き）をハンダ付けする練習をさせた。その出来具合を指導教員が目視でチェックし、問題がある学生にはその箇所を指摘して、やり直しをさせた。

２．３　赤外線リモコン送受信器（IrCOM）ボードのハンダ付け手順

赤外線リモコン送受信器全体の回路図は最後の Fig. 12 に示す。部品点数 25、総ピン数 124 の赤外線リモコン送受信器（Fig. 3）各一個のハンダ付けを２人一組でやらせた。

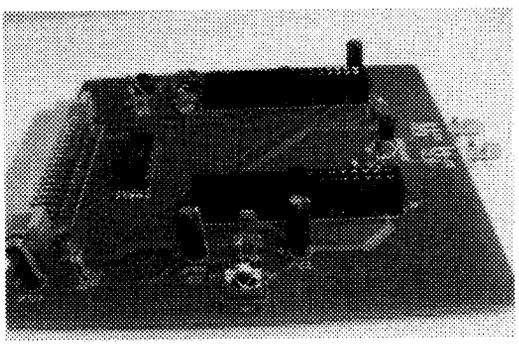

Fig. 3 試作赤外線リモコン送受信器ボード

使用するマイコンボードは北斗電子製の BB64E3687F で、この図の 17×2=34 ピンコネクタ２個の上に搭載する。

ハンダ付けは未体験の学生がほとんどであったため、丁寧な手順書を作った。以下はその一部の抜粋である。

(1) IC(74HC14)

- 両側２列の足(ピン)をテーブルに当てるなどして、ほぼ垂直になるように曲げる。
  - ➢ ボードに無理なく入るところまで曲げる。
  - ➢ 写真に従い、向きを間違えないように注意
  - ➢ IC が落ちないように抑えてボードを逆さにし、テーブルに置く。このとき IC が平らになるように注意
  - ➢ まず、両端の２ピンをハンダ付け　→ハンダを付け過ぎないこと！
  - ➢ 次に、残りのピン全部をハンダ付け　→隣のピンとショートしないように注意！

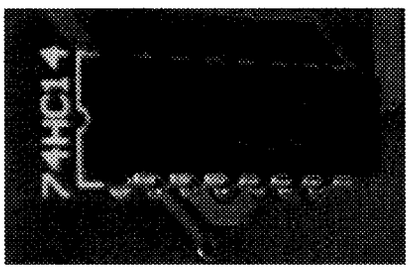

Fig. 4　IC(74HC14)のハンダ付け

(2) トランジスタ(2SC1815)

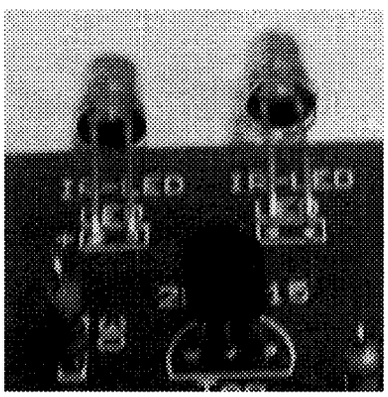

Fig. 5　2SC1815 のハンダ付け

- 写真を参考に、向きを間違えないように注意！
- 両側の２本の足を左右に曲げ、落ちないようにしておいて、真ん中の足をハンダ付け。
- 次に左右に曲げた足をやや真っ直ぐに戻してから、ハンダ付け
- ニッパーで足を 1.5mm 程度残して切断

実験前は、部品点数 25、総ピン数 124 のボードの組立は、ハンダ付け未体験の学生には荷が重く、「面倒くさい」「嫌だ」という感想を予想していた。実験後は、「非常に面白かった」「モノを作っている実感があり、楽しかった」という意見がほとんどで、むしろ驚いた。

実験後のアンケートに次の項目を入れた。

【①】ハンダ付けのコツは理解出来ましたか？

【②】部品点数 25、総ピン数 124 の赤外線リモコン送受信器のハンダ付けは面白かったですか？

30 人の平均値は、5 段階評価でそれぞれ 4.4、4 と高かった。

3．オシロスコープの操作法

電子計測器類を全く触ったことの無い学生達にオシロスコープの使い方を教えるのは厄介である。しかも、赤外線リモコンの波形を計測するためには、繰り返し現象ではなく、単発現象の計測が出来、しかもパルス幅を正確に測定できなければならない。そのため、オシロスコープの操作方法に関しても丁寧なマニュアルを用意した。なお、実験で使用したのはアジレント・テクノロジー社の DSO1002A(60MHz, 2 チャンネル)である。

3．1　基本設定

オシロスコープとは時間的に変化する電子信号(電圧波形)を計測するための装置であり、これを使用するためには次の 3 つの設定をする必要がある。

- 垂直軸(電圧レベル)設定、水平軸(時間単位)設定、トリガー設定(波形をどこで捕えるか)

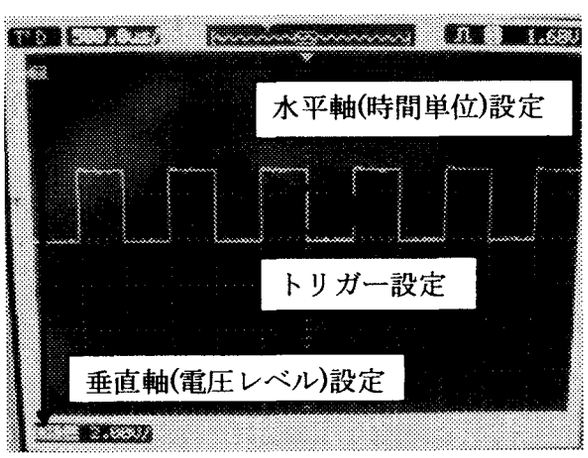

Fig. 6　オシロスコープ画面

3．2　単発波形の測定

- マイコンの動作波形には 1 度しか発生しない信号が多い。
- このときは Single モードで測定する。

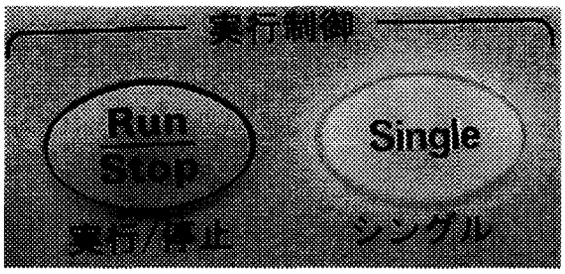

Fig. 7　Single モードボタン

- 波形を捕らえると、左の「Run/Stop」ボタンが赤に点灯する。
- 「7 インチ TV」用リモコンの信号を受信機で受けると、ボタン 1(ＣＨ１)の場合、次のような波形が観測される。

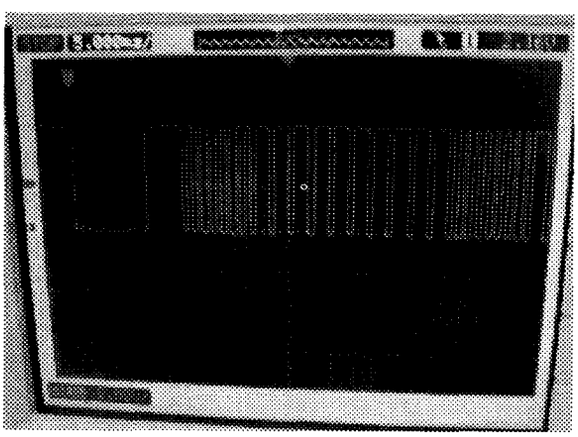

Fig. 8　CH1 の NEC 送信フォーマット

4．2 節で述べる「赤外線リモコン送信機プログラム」のベースプログラムの場合は次のような波形が観測される。(反転：ＯＦＦ)

この課題では、プログラムをいじりながら、この波形のパルス幅を何度も何度も測定することになる。完成まで、少なくとも数時間は HEW4 を使ったプログラムの変更・ビルドと、オシロスコープを使った単発波形計測、パルス幅測定と格闘することになる。2 つのツールの使い方がいつの間にか身に付く課題になっている。

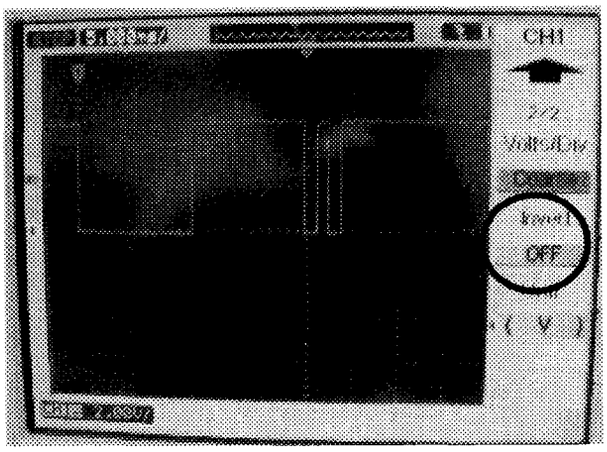

Fig. 9　サンプルプログラムの出力波形

4．赤外線リモコン送受信器

4．1　赤外線リモコンのデータ・フォーマット

よく使われている赤外線リモコンのデータ・フォーマットはＮＥＣ送信フォーマットと家電製品協会フォーマットである。本実験では赤外線リモコンの例として AVOX 社の７型ポータブルＴＶ用リモコンを採用した（Fig. 10)。TV 本体が小型軽量で持ち運びに便利なので、実験向きであること、価格も安いこと、などがその理由である。

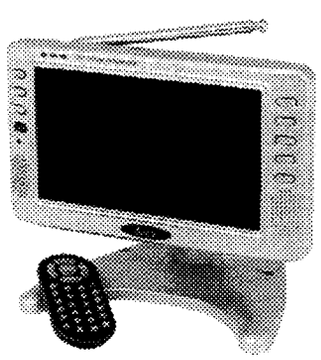

Fig. 10　７型ポータブルＴＶ（AVOX 社）

このＴＶはＮＥＣ送信フォーマットを使用しているので、これについて説明する。Fig. 11 のように、リーダー部とデータ部からなる [1]。データ部はカスタムコードとデータコードから構成されている。前者は会社ごとに違った番号を使用している。後者は、制御対象機器がＴＶならチャネル番号などになる。

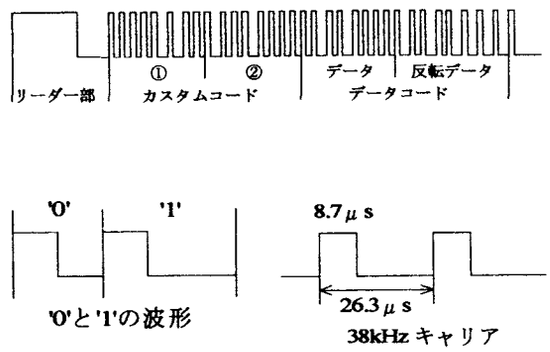

Fig.11 ＮＥＣ送信フォーマット

赤外線リモコンは 38kHz で変調された光パルスを赤外線 LED から発することで通信している。この 38KHz で変調された光パルスはキャリアと呼ばれるが、消費電力を抑えるため、図のようにデューティ比は 1:3 に設定されている。

リーダー部は、本機の場合、図のように HIGH 9ms、LOW 4.5ms となっており、HIGH 時間中のみ 38kHz のキャリアが送出される。

データ部は２進数0、1の組合せで構成されており、図のようにパルス 0 とパルス 1 は次のような HIGH（キャリア ON)、LOW（キャリア OFF）時間の組合せとなっている。

パルス０　：　HIGH 0.56ms　LOW 0.56ms
（計 1.12ms)
パルス１：　HIGH 0.56ms　LOW 1.69ms
（計 2.25ms)

３．２節で示した Fig. 8 は、本機のリモコンで CH1 を送った時のオシロスコープ波形（長いため全ビットは表示されていない）である。この写真からデータ部を解析すると次のようになっていることが分かる。

カスタムコード：　0000 0001 1111 1111
データコード：　1000 0000 0111 1111+0
(16 ビット＋ストップビット)

データコードは、10進値1を2進8ビットで表したものを LSB→MSB の順に並べたもの、及びその反転データになっていることが分かる。他のチャンネル制御信号も調べたが、まさにこのルールになっていることが分かった。

4．2　赤外線リモコン送信機のプログラミング

H8マイコンプログラミング環境HEW4の操作方法に従って新規プロジェクトを作る[2]。CPU を 300H/3687 にする。

- 付録を参照して main.c のプログラムを作成する。
  - ビルド、実行

- オシロスコープと赤外線受信機の準備
  - 上のプログラムが正しく出来ていれば、SW0 を押したとき、リーダー部のような波形がオシロスコープで観測されるはず
  - その後ろに、'0'の波形とストップビットも入れてある。但し、パルス幅は仮の値。
  - ①関数 RederCode()内の待ち時間発生用繰返しルーチン(for 文)を修正して、赤外線リモコンのリーダー部と同じ波形が出るように調整
  - ②関数 Send_0( )の中の待ち時間発生用繰返しルーチン(for 文)を修正して、'0' の波形が正しく出力されるようにする
  - ③Send_0( )にならって、関数 Send_1( )の中にコードを追加し、'1' の波形が正しく出力されるようにする

- オシロスコープで読み取った「7インチ TV リモコン」のボタン1の波形に従って、カスタムコード、データコードを関数 OnPressButtonSW1(void) の中に追加する。
- 例えば、カスタムコードは 00000001・・・、データコードは 1000・・・とする

5．学習リモコンのプログラミング

NEC フォーマットの赤外線リモコンの送信コードを自動的に読み込み、同じ信号を発生させる「学習リモコン」を作る。

◎プログラミングのポイント

- タイマーZ0 割込みを使い、0.1ms＝100μs ごとに受光器からの入力信号(PDR6)をチェックする。
  - OnTimer()関数を使用
- リーダー部では、Low Level が約 90 回(9.0ms)続き、次に High Level が約 45 回(4.5ms)続くはず。
  - 100μs ごとに Low Level と High Level の連続回数を数え、それぞれ約 90 回、約 45 回と続いたら、リーダー部が来たと判定
- カスタムコード部、データ部でも、100μs ごとに Low Level と High Level の連続回数を数え、ビットコードの判定を行う。

◎使用法

- SW0　→パラメータリセット
- リモコンの LED を IrCOM ボードの受光器に向け、どれかの CH ボタンを押す。
- 赤外線 LED を TV に向け、SW1 を押す。
  - これで CH が正しく変われば完成

6．まとめ

赤外線リモコン送受信器を題材にした組込みプログラム用電子教材の開発を開発した。本教材は、単にマイコンプログラムの演習になるだけでなく、ハンダ付けやオシロスコープの操作法を必然的にじっくりと学習することにもなるので、「組込みプログラム」教材として好適であると考える。

参考文献

1) http://www5b.biglobe.ne.jp/~YAUSI/gallery/electronics/041219/041219.htm.

2) 島田義人:「H8/Tiny マイコン完璧マニュアル」, CQ 出版社, 2007.

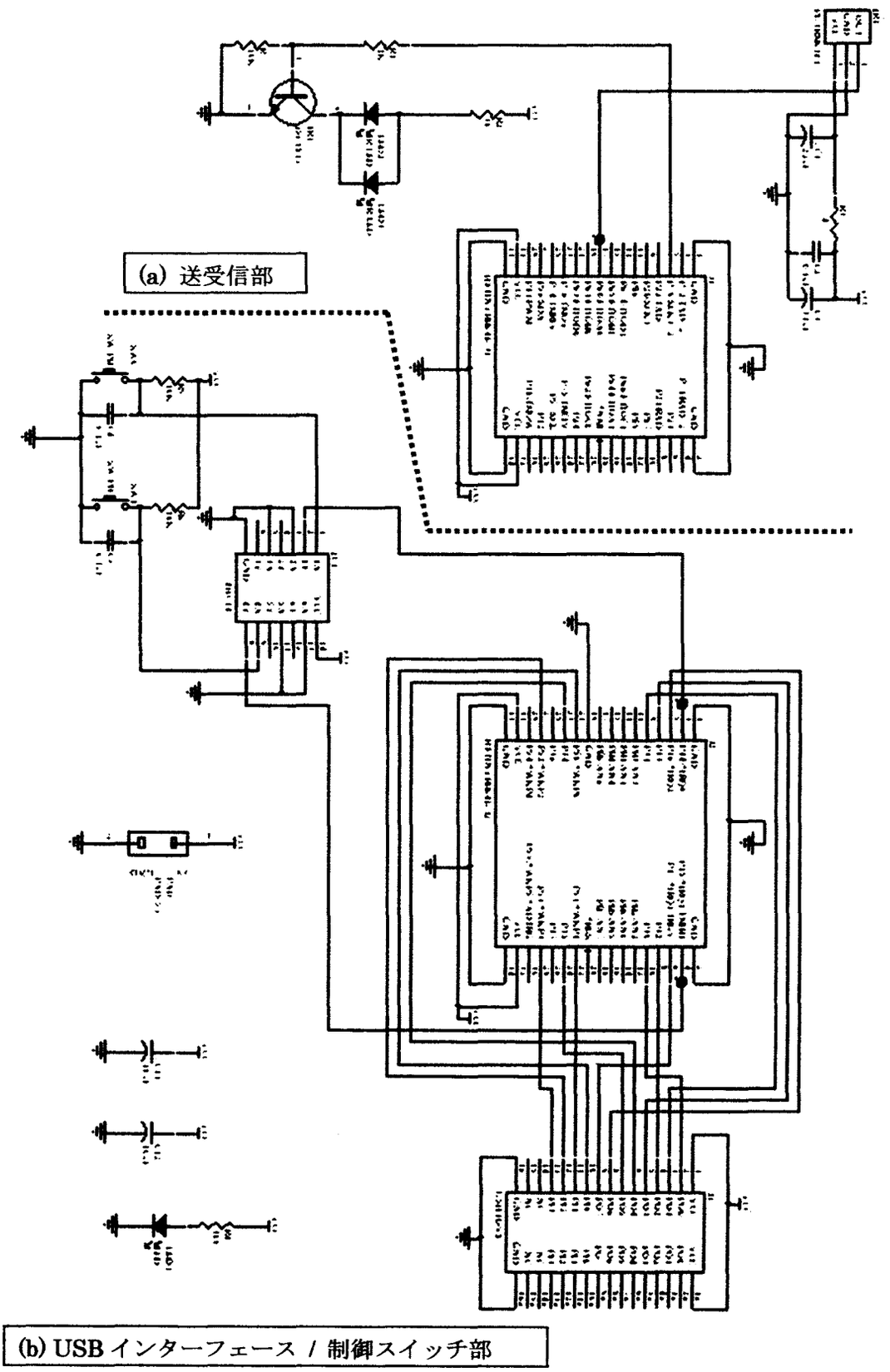

Fig. 12 赤外線リモコン送受信器全体回路図

◎付録　赤外線送信器のプログラム(H8/3687 用)

```
/*****************************************************************************
        main.c  (CPU TYPE : H8/3687)
*****************************************************************************/
#include<MACHINE.H>                                     /*組込み関数ヘッダ*/
#pragma interrupt(OnPressButtonSW1)                     /*IRQ0割り込み関数*/

//3687F用
#define PCR7 *((volatile unsigned char *)0xffea)        /* PCR7 */
#define PDR7 *((volatile unsigned char *)0xffda)        /* PDR7 */

//割込み設定
#define PUCR1 *((volatile unsigned char*) 0xffd0)       /*PUCR1*/
#define IEGR1 *((volatile unsigned char*) 0xfff2)        /*IEGR1  エッジ選択*/
#define IENR1 *((volatile unsigned char*) 0xfff4)        /*IENR1  割り込みイネーブル*/
#define IRR1 *((volatile unsigned char*) 0xfff6)        /*IRR1 割り込みステータスフラグ*/
#define PMR1 *((volatile unsigned char*) 0xffe0)        /*PMR1 モードレジスタ*/
#define PCR1 *((volatile unsigned char*) 0xffe4)        /*PCR1*/

void Sleep_ho(unsigned long lTime);
void Send_1(void);
void Send_0(void);
void SendByte(unsigned char cc);
void Carrier(void);
void CarrierOff(void);
void ReaderCode(void);

void OnPressButtonSW1(void)/*IRQ0*/
{
        volatile unsigned long i;
        volatile unsigned char DataCode;

        IRR1 = IRR1 & 0xfe; /*割込み要求フラグクリア*/

        ReaderCode();

        //Custom Code
        for(i=0;i<7; i++) Send_0(); Send_1();
        for(i=0;i<8; i++) Send_1();

        //Data Code - CH7
        DataCode=0x07;
        SendByte(DataCode);

        //Stop Bit
        Send_0();

        return;
}

#pragma section V1                              // CV1
// 仮想ベクターテーブル
void (*const VEC_TBL1[])(void) = {
        OnPressButtonSW1            //IRQ0
};
```

```
/*****************************************************************************
    時間待ちループ
*****************************************************************************/
void Sleep_ho(unsigned long lTime)
{
        volatile unsigned long l;
        for(l=0;l<lTime*1000;++l)
        {
                /*何もせずに時間稼ぎ*/
        }
        return;
}

void ReaderCode(void)
{
        volatile unsigned long i;
        for(i=0; i<340; i++) Carrier();              //9ms
        for(i=0; i<184; i++) CarrierOff();           //4.5ms
        return;
}

void Send_1(void)
{
        volatile unsigned long i;
        for(i=0; i<22; i++) Carrier();               //0.56ms(0.568ms)
        for(i=0; i<67; i++) CarrierOff();            //1.69ms(Pulse Width => 2.25ms)
        return;
}

void Send_0(void)
{
        volatile unsigned long i;
        for(i=0; i<22; i++) Carrier();               //0.56ms(0.568ms)
        for(i=0; i<22; i++) CarrierOff();            //0.56ms(Pulse Width => 1.12ms)
        return;
}

void SendByte(unsigned char cc)
{
        volatile unsigned char c1;
        volatile unsigned int k=8;

        while(k>0) {
                c1=cc & 0x01;

                if(c1==1) Send_1();
                else Send_0();

                cc = cc >> 1;
                k--;
        }

        return;
}
```

```
// 37.88kHz, 26.4us
void Carrier(void)
{
        volatile unsigned int k;
        PDR7 = 0x01; for(k=0; k<10; k++);
        PDR7 = 0x00; for(k=0; k<9; k++);
        return;
}

// 37.88kHz, 26.4us
void CarrierOff(void)
{
        volatile unsigned int k;
        PDR7 = 0x00; for(k=0; k<19; k++);
        return;
}

void main(void)
{
        //3687F用
        PCR7 = 0xff;                    /* PCR7 */
        set_imask_ccr(1);               /*割り込み不許可*/

        /*ボタンの使用 割り込みの初期設定 P14,P15使用*/
        PCR1 = 0xcf;                    /*P14,15をリードピン、残りをライトピンに明示指定*/
        PUCR1 = 0x38;                   /* P14, P15のMOSプルアップをオン*/
        Sleep_ho(100);                  /*51,52番ピンの初期電圧が安定するまで念のため時間稼ぎ*/
        PMR1 = 0x30;                    /*汎用ポートではなくIRQ0,IRQ1機能を選択*/
        Sleep_ho(10);                   /*ハードウエアマニュアルに従い、時間稼ぎ*/
        IRR1 = IRR1 & 0x00;             /*割込み要求フラグクリア*/
        IEGR1 = IEGR1 | 0x03;           /*IRQ0,IRQ1立ち上がりエッジ検出を選択*/
        IENR1 = IENR1 | 0x03;           /*IRQ0,IRQ1 イネーブル*/
        set_imask_ccr(0);               /*割り込み許可*/

        while(1)
        {
                PDR7 = 0x00;
        }

        return;
}
```