AVR313 : PC ATキーボードのインターフェース法

要点

- 標準PC ATキーボードのインターフェース
- 2つの入出力ピンだけが必要（それらの1つは外部割り込みピンでなければなりません。）
- 外部ハードウェア必要なし
- キーボード⇒シリアル変換器を実装するC言語での完全な例

序説

殆どのマイクロ コントローラはいくつかの種類の対人インターフェースを必要とします。この応用記述は標準PC ATキーボードを使用してこれを行う1つの方法を記述します。

物理インターフェース

キーボードとホスト間の物理インターフェースが図1.で示されます。クロックとデータの2つの信号線が使用されます。信号線はキーボード内に配置されたプルアップ抵抗を持つオープン コレクタです。これはホストまたはキーボードのどちらでもLowレベルの強制を可能にします。"5D"型式の5ピンDINコネクタとより小さな6ピンのミニDINの2つのコネクタが利用可能です。ピン割り当ては表1.で示されます。

図1. インターフェース

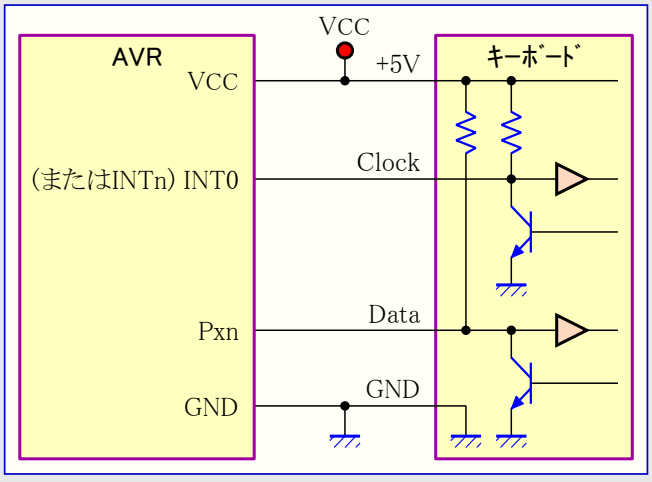
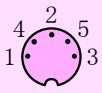
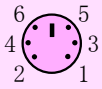


表1. ATキーボード コネクタ ピン割り当て

ATコンピュータ		
信号	DIN41524(PC側メス)5ピンDIN 180°	PS2形式6ピン ミニDIN(PC側メス)
Clock	1	5
Data	2	1
N.C.	3	2,6
GND	4	3
+5V	5	4
遮蔽	外殻	外殻

8ビット AVR® マイクロ コントローラ

応用記述

本書は一般の方々の便宜のため有志により作成されたもので、ATMEL社とは無関係であることを御承知ください。しおりのはじめにでの内容にご注意ください。

Rev. 1235B-05/02, 1235BJ1-12/13

ﾀｲﾐﾝｸﾞ

ｷｰﾎﾞｰﾄﾞからﾎｽﾄへのﾃﾞｰﾀ転送は図2.で示されます。規約は1開始ﾋﾞｯﾄ(常に0)、8ﾃﾞｰﾀ ﾋﾞｯﾄ、奇数ﾊﾟﾘﾃｨ ﾋﾞｯﾄ、1停止ﾋﾞｯﾄ(常に1)です。ﾃﾞｰﾀはｸﾛｯｸ ﾊﾟﾙｽのLow区間の間で有効です。ｷｰﾎﾞｰﾄﾞがｸﾛｯｸ信号を生成し、ｸﾛｯｸ ﾊﾟﾙｽは代表的に30～50μsのLowと30～50μsのHighです。

ﾎｽﾄ ｼｽﾃﾑはｸﾛｯｸ線をLowに強制することによってｷｰﾎﾞｰﾄﾞへ命令を送ることができます。そしてﾃﾞｰﾀ線をLowに引き込みます(開始ﾋﾞｯﾄ)。直ぐにｸﾛｯｸ線は(ｷｰﾎﾞｰﾄﾞに於いて)開放されなければなりません。ｷｰﾎﾞｰﾄﾞは10個のｸﾛｯｸ ﾊﾟﾙｽを数えます。ﾃﾞｰﾀ線はｸﾛｯｸ ﾊﾟﾙｽの後行端の前にﾎｽﾄによって正しいﾚﾍﾞﾙに設定されなければなりません。第10ﾋﾞｯﾄ後、ｷｰﾎﾞｰﾄﾞはﾃﾞｰﾀ線上のHighﾚﾍﾞﾙ(停止ﾋﾞｯﾄ)を調べ、それがHighなら、ﾃﾞｰﾀ線をLowに強制します。これはｷｰﾎﾞｰﾄﾞによってﾃﾞｰﾀが受信されたことをﾎｽﾄに知らせます。この設計記述に於けるｿﾌﾄｳｪｱはｷｰﾎﾞｰﾄﾞへどんな命令も送りません。

走査符号

ATｷｰﾎﾞｰﾄﾞは各ｷｰに関連する走査符号を持ちます。ｷｰが押されると、この符号が送信されます。ｷｰが一定の間、押さえ付けられると、繰り返しを始めます。繰り返し速度は代表的に1秒間に10回です。ｷｰが開放されると、ｷｰ走査符号が後続する"break"符号($F0)が送信されます。ｷｰの殆どに関して走査符号は1ﾊﾞｲﾄです。*Home,Insert,Delete* ｷｰのようないくつかのｷｰは2～5ﾊﾞｲﾄの拡張走査符号を持ちます。(これらの)先頭ﾊﾞｲﾄは常に$E0です。これは"break"手順(例えば$E0,$F0,$xx,...)についても変わりません。

ATｷｰﾎﾞｰﾄﾞは3組の走査符号を扱う能力があり、そして第2組が既定です。本例は第2組だけを使用します。

方法

ｷｰﾎﾞｰﾄﾞ受信はINT0_interrupt割り込み関数によって処理されます。受信は残りのﾌﾟﾛｸﾞﾗﾑと無関係に動作します。

方法は全く単純で、ｸﾛｯｸ ﾊﾟﾙｽの先行端でﾃﾞｰﾀ線の値を格納します。これはｸﾛｯｸ線がINT0またはINTnﾋﾟﾝに接続されている場合、容易に扱われます。割り込み関数はｸﾛｯｸ周期のｴｯｼﾞ毎に実行され、ﾃﾞｰﾀは下降端で格納されます。全てのﾋﾞｯﾄが受信された後、ﾃﾞｰﾀを復号することができます。これはdecode関数を呼び出すことによって行われます。文字ｷｰに関してこの関数はASCII文字を緩衝部内に格納します。ｷｰが押された時にｼﾌﾄ ｷｰが押さえ付けられている場合を考慮します。機能ｷｰ、誘導ｷｰ(矢印ｷｰ、Page Up/Downｷｰなど)ような他のｷｰとCtrlやAltのような変更ｷｰは無視されます。

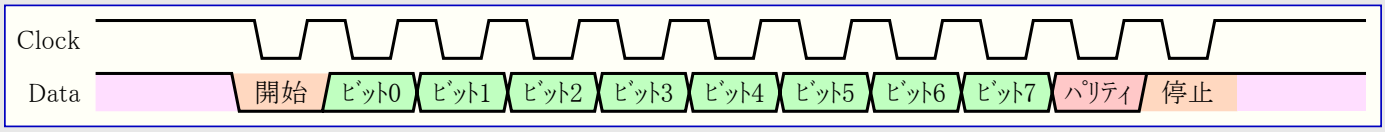
走査符号からASCII文字への割り当てはｼﾌﾄされた文字に対する1つの表とｼﾌﾄされないものに対する1つの参照表で処理されます。

変更と改良

ﾎｽﾄがｷｰﾎﾞｰﾄﾞとの同期を外した場合、後続する全ての受信ﾃﾞｰﾀが不正になります。これを解決する1つの方法は時間超過を用いることです。1.5ms以内に11ﾋﾞｯﾄが受信されない場合、何らかの異常が起きています。ﾋﾞｯﾄ計数器がﾘｾｯﾄされ、誤ったﾃﾞｰﾀは破棄されるべきです。

自動繰り返し(typematic)の速度と遅延のようなｷｰﾎﾞｰﾄﾞのﾊﾟﾗﾒｰﾀが設定される場合、ﾃﾞｰﾀがｷｰﾎﾞｰﾄﾞへ送られなければなりません。これは前で記述されるように行うことができます。命令についてはｷｰﾎﾞｰﾄﾞ製造業者の仕様をご覧ください。

図2. ｷｰﾎﾞｰﾄﾞからﾎｽﾄへの転送ﾀｲﾐﾝｸﾞ



Main.c

```
#include <pgmspace.h>
#include <stdio.h>
#include <stdlib.h>
#include "io8515.h"

#include "serial.h"
#include "gpr.h"
#include "kb.h"

void main(void)
{
  unsigned char key;

  init_uart();          // UART送信緩衝部初期化
  init_kb();            // ｷｰﾎﾞｰﾄﾞ受信初期化
  while(1)
  {
    key=getchar();      // ｷｰﾎﾞｰﾄﾞからの受信文字取得
    putchar(key);       // 受信文字をUART送信
    delay(100);
  }
}
```

Low_level_init.c

```
#include <ina90.h>
#include <io8515.h>

int __low_level_init(void)
{
  UBRR = 12;            // 19200bps/4 MHz
  UCR = 0x08;           // UART送信許可
  GIMSK= 0x40;          // INT0割り込み許可

  _SEI();               // 全体割り込み許可
  return 1;
}
```

Serial.c

```
#include <stdio.h>
#include <pgmspace.h>
#include <io8515.h>          /* I/Oレジスタ宣言 */
#include "serial.h"

#define ESC 0x1b
#define BUFF_SIZE 64

flash char CLR[] = {ESC, '[','H', ESC, '[', '2', 'J',0};

unsigned char UART_buffer[BUFF_SIZE];
unsigned char *inptr, *outptr;
unsigned char buff_cnt;

void init_uart(void)
{
  inptr = UART_buffer;
  outptr = UART_buffer;
  buff_cnt = 0;
}

void clr(void)
{
  puts_P(CLR);        // VT100端末へ'画面消去'を送信
}

int putchar(int c)
{
  if (buff_cnt<BUFF_SIZE)
  {                   // 緩衝部内空き有りで、
    *inptr = c;       // 緩衝部内に文字設定
    inptr++;          // 緩衝部書き込み位置指示子進行

    buff_cnt++;       // 緩衝部内文字バイト数増加

    if (inptr >= UART_buffer + BUFF_SIZE)  // 緩衝部書き込み位置指示子上限丸め
       inptr = UART_buffer;

    UCR = (1<<UDRIE) | (1<<TXEN);          // UARTデータレジスタ空割り込み許可

    return 1;         // 成功符号で復帰
  } else {            // 緩衝部内空きなしで、
    return 0;         // 緩衝部満杯符号で復帰
  }
}

/* 割り込み駆動UART送信 */
interrupt [UART_UDRE_vect] void UART_UDRE_interrupt(void)
{
  UDR = *outptr;      // 緩衝部から1バイト送信開始
  outptr++;           // 緩衝部読み込み位置指示子進行

  if (outptr >= UART_buffer + BUFF_SIZE)   // 緩衝部読み込み位置指示子上限丸め
     outptr = UART_buffer;

  if (--buff_cnt == 0)                     // 緩衝部内文字バイト数減少、緩衝部が空らで、
     UCR = UCR && (1<<UDRIE);              // UARTデータレジスタ空割り込み禁止
}
```

Kb.c

```
#include <pgmspace.h>
#include "kb.h"
#include "serial.h"
#include "gpr.h"

#include "scancodes.h"

#define BUFF_SIZE 64

unsigned char edge, bitcount;        // 0=下降端,1=上昇端

unsigned char kb_buffer[BUFF_SIZE];
unsigned char *inpt, *outpt;
unsigned char buffcnt;

void init_kb(void)
{
  inpt = kb_buffer;          // 緩衝部初期化
  outpt = kb_buffer;
  buffcnt = 0;

  MCUCR = 2;                 // INT0割り込みを下降端に設定
  edge = 0;                  // 下降端ﾌﾗｸﾞ設定(0=下降端,1=上昇端)
  bitcount = 11;             // 受信ﾋﾞｯﾄ計数器初期化
}

interrupt [INT0_vect] void INT0_interrupt(void)
{
  static unsigned char data;         // 受信した走査符号ﾊﾞｲﾄ

  if (!edge)                         // 下降端割り込みで開始
  {
    if(bitcount < 11 && bitcount > 2)       // 10～3がﾃﾞｰﾀ、開始/ﾊﾟﾘﾃｨ/停止ﾋﾞｯﾄは無視
    {
      data = (data >> 1);            // 現受信ﾃﾞｰﾀを1ﾋﾞｯﾄ下位側へ移動(MSB=0)
      if(PIND & 8)                   // Data線=High(1)で、
      data = data | 0x80;            // 現受信ﾃﾞｰﾀのMSBを1に設定
    }

    MCUCR = 3;               // INT0割り込みを上昇端に設定
    edge = 1;                // 上昇端ﾌﾗｸﾞ設定(0=下降端,1=上昇端)

  } else {                   // 上昇端での割り込みで、

    MCUCR = 2;               // INT0割り込みを下降端に設定
    edge = 0;                // 下降端ﾌﾗｸﾞ設定(0=下降端,1=上昇端)

    if(--bitcount == 0)      // 受信ﾋﾞｯﾄ計数器減数、受信完了で、
    {
      decode(data);          // 受信走査符号をASCII符号に変換
      bitcount = 11;         // 受信ﾋﾞｯﾄ計数器初期化
    }
  }
}
```

```
void decode(unsigned char sc)
{
  static unsigned char is_up=0, shift = 0, mode = 0;
  unsigned char i;
  if (!is_up)          // 直前がキー開放(break)符号でなければ、
  {
    switch (sc)
    {
      case 0xF0 :      // キー開放(break)符号なら、
      is_up = 1;       // キー開放(break)符号ﾌﾗｸﾞ設定
      break;

      case 0x12 :      // 左側Shiftキーなら、
      shift = 1;       // ｼﾌﾄ ﾌﾗｸﾞ設定
      break;

      case 0x59 :      // 右側Shiftキーなら、
      shift = 1;       // ｼﾌﾄ ﾌﾗｸﾞ設定
      break;

      case 0x05 :      // F1キーで、
      if(mode == 0) mode = 1;        // ASCII動作開放状態ならば走査符号動作に変更
      if(mode == 2) mode = 3;        // 走査符号動作開放状態ならばASCII動作に変更
      break;

      default:
      if(mode == 0 || mode == 3)     // ASCII動作なら、
      {
        if(!shift)                   // ｼﾌﾄなしで、
        {
          for(i = 0; unshifted[i][0]!=sc && unshifted[i][0]; i++);      // 対応表検索
          if (unshifted[i][0] == sc) {      // 有効符号なら、対応文字をｷｰ緩衝部へ格納
             put_kbbuff(unshifted[i][1]);
          }
        } else {                     // ｼﾌﾄ有りで、
          for(i = 0; shifted[i][0]!=sc && shifted[i][0]; i++);   // 対応表検索
          if (shifted[i][0] == sc) {        // 有効符号なら、対応文字をｷｰ緩衝部へ格納
             put_kbbuff(shifted[i][1]);
          }
        }
      } else{                        // 走査符号動作なら、
        print_hexbyte(sc);           // 走査符号を16進文字で直接出力
        put_kbbuff(' ');             // 緩衝部に空白文字を設定
        put_kbbuff(' ');
      }
      break;
    }
  } else {             // 直前がキー開放(break)符号で、
    is_up = 0;         // キー開放(break)符号ﾌﾗｸﾞ解除($F0連続は禁止)
    switch (sc)
    {
      case 0x12 :      // 左側Shiftキーなら、
      shift = 0;       // ｼﾌﾄ ﾌﾗｸﾞ解除
      break;

      case 0x59 :      // 右側Shiftキーなら、
      shift = 0;       // ｼﾌﾄ ﾌﾗｸﾞ解除
      break;

      case 0x05 :      // F1キーで、
      if(mode == 1) mode = 2;        // 走査符号動作ならば走査符号動作開放状態に変更
      if(mode == 3) mode = 0;        // ASCII動作ならばASCII動作開放状態に変更
      break;

      case 0x06 :      // F2キーで、
      clr();           // 画面消去符号列送信
      break;
    }
  }
}
```

```
void put_kbbuff(unsigned char c)
{
  if (buffcnt<BUFF_SIZE)     // ｷｰ緩衝部空き有りで、
  {
    *inpt = c;               // ｷｰ緩衝部内に文字ﾊﾞｲﾄ設定
    inpt++;                  // ｷｰ緩衝部書き込み位置指示子進行
    buffcnt++;               // ｷｰ緩衝部内文字ﾊﾞｲﾄ数計数値増加

    if (inpt >= kb_buffer + BUFF_SIZE)      // ｷｰ緩衝部書き込み位置指示子丸め
       inpt = kb_buffer;
  }
}

int getchar(void)
{
  int byte;
  while(buffcnt == 0);       // ｷｰ緩衝部文字有りまで待機

  byte = *outpt;             // ｷｰ緩衝部から文字ﾊﾞｲﾄ取得
  outpt++;                   // ｷｰ緩衝部読み込み位置指示子進行

  if (outpt >= kb_buffer + BUFF_SIZE)       // ｷｰ緩衝部読み込み位置指示子丸め
  outpt = kb_buffer;
  buffcnt--;                 // ｷｰ緩衝部内文字ﾊﾞｲﾄ数計数値減少

  return byte;
}
```

Gpr.c

```
#include "gpr.h"

void print_hexbyte(unsigned char i)
{
  unsigned char h, l;

  h = i & 0xF0;              // 上位ﾆﾌﾞﾙ抽出
  h = h>>4;                  // 上位ﾆﾌﾞﾙを下位ﾆﾌﾞﾙ位置へ移動
  h = h + '0';               // 16進文字符号変換
  if (h > '9') h = h + 7;    // '9'越えで、'A'からに変更

  l = (i & 0x0F)+'0';        // 下位ﾆﾌﾞﾙ16進文字符号変換
  if (l > '9') l = l + 7;    // '9'越えで、'A'からに変更

  putchar(h);                // 上位ﾆﾌﾞﾙ16進文字直接出力
  putchar(l);                // 下位ﾆﾌﾞﾙ16進文字直接出力
}

void delay(char d)
{
  char i,j,k;
  for(i=0; i<d; i++)
    for(j=0; j<40; j++)
      for(k=0; k<176; k++);
}
```

Pindefs.h

```
//**********************
// ﾋﾟﾝ定義ﾌｧｲﾙ
//**********************

// ｷｰﾎﾞｰﾄﾞ接続
#define PIN_KB PIND
#define PORT_KB PORTD
#define CLOCK 2
#define DATAPIN 3
```

Scancodes.h

```
// シフトなし符号-文字対応表
flash unsigned char unshifted[][2] = {
0x0d,9,
0x0e,'|',
0x15,'q',
0x16,'1',
0x1a,'z',
0x1b,'s',
0x1c,'a',
0x1d,'w',
0x1e,'2',
0x21,'c',
0x22,'x',
0x23,'d',
0x24,'e',
0x25,'4',
0x26,'3',
0x29,' ',
0x2a,'v',
0x2b,'f',
0x2c,'t',
0x2d,'r',
0x2e,'5',
0x31,'n',
0x32,'b',
0x33,'h',
0x34,'g',
0x35,'y',
0x36,'6',
0x39,',',
0x3a,'m',
0x3b,'j',
0x3c,'u',
0x3d,'7',
0x3e,'8',
0x41,',',
0x42,'k',
0x43,'i',
0x44,'o',
0x45,'0',
0x46,'9',
0x49,'.',
0x4a,'-',
0x4b,'l',
0x4c,'o',
0x4d,'p',
0x4e,'+',
0x52,'a',
0x54,'a',
0x55,'¥¥',
0x5a,13,
0x5b,'¨',
0x5d,'¥',
0x61,'<',
0x66,8,
0x69,'1',
0x6b,'4',
0x6c,'7',
0x70,'0',
0x71,',',
0x72,'2',
0x73,'5',
0x74,'6',
0x75,'8',
0x79,'+',
0x7a,'3',
0x7b,'-',
0x7c,'*',
0x7d,'9',
0,0
};
```

```
// シフト有り符号-文字対応表
flash unsigned char shifted[][2] = {
0x0d,9,
0x0e,'§',
0x15,'Q',
0x16,'!',
0x1a,'Z',
0x1b,'S',
0x1c,'A',
0x1d,'W',
0x1e,'"',
0x21,'C',
0x22,'X',
0x23,'D',
0x24,'E',
0x25,'.',
0x26,'#',
0x29,' ',
0x2a,'V',
0x2b,'F',
0x2c,'T',
0x2d,'R',
0x2e,'%',
0x31,'N',
0x32,'B',
0x33,'H',
0x34,'G',
0x35,'Y',
0x36,'&',
0x39,'L',
0x3a,'M',
0x3b,'J',
0x3c,'U',
0x3d,'/',
0x3e,'(',
0x41,';',
0x42,'K',
0x43,'I',
0x44,'O',
0x45,'=',
0x46,')',
0x49,':',
0x4a,'_',
0x4b,'L',
0x4c,'O',
0x4d,'P',
0x4e,'?',
0x52,'A',
0x54,'A',
0x55,'`',
0x5a,13,
0x5b,'^',
0x5d,'*',
0x61,'>',
0x66,8,
0x69,'1',
0x6b,'4',
0x6c,'7',
0x70,'0',
0x71,',',
0x72,'2',
0x73,'5',
0x74,'6',
0x75,'8',
0x79,'+',
0x7a,'3',
0x7b,'-',
0x7c,'*',
0x7d,'9',
0,0
};
```

本社

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131, USA
TEL 1(408) 441-0311
FAX 1(408) 487-2600

国外営業拠点

Atmel Asia
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
TEL (852) 2245-6100
FAX (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-Yvelines
Cedex
France
TEL (33) 1-30-60-70-00
FAX (33) 1-30-60-71-11

Atmel Japan
104-0033 東京都中央区
新川1-24-8
東熱新川ﾋﾞﾙ 9F
ｱﾄﾒﾙ ｼﾞｬﾊﾟﾝ株式会社
TEL (81) 03-3523-3551
FAX (81) 03-3523-7581

製造拠点

Memory
2325 Orchard Parkway
San Jose, CA 95131, USA
TEL 1(408) 441-0311
FAX 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131, USA
TEL 1(408) 441-0311
FAX 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3
France
TEL (33) 2-40-18-18-18
FAX (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex
France
TEL (33) 4-42-53-60-00
FAX (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR
Scotland
TEL (44) 1355-803-000
FAX (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn
Germany
TEL (49) 71-31-67-0
FAX (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Biometrics
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex
France
TEL (33) 4-76-58-47-50
FAX (33) 4-76-58-47-60

文献請求
www.atmel.com/literature

本応用記述はATMELのAVR313応用記述(doc1235.pdf Rev.1235B-05/02)の翻訳日本語版です。日本語では不自然となる重複する形容表現は省略されている場合があります。日本語では難解となる表現は大幅に意訳されている部分もあります。必要に応じて一部加筆されています。頁割の変更により、原本より頁数が少なくなっています。

必要と思われる部分には( )内に英語表記や略称などを残す形で表記しています。

青字の部分はﾘﾝｸとなっています。一般的に赤字の0,1は論理0,1を表します。その他の赤字は重要な部分を表します。