Appendix 4 加速度センサを使って基板を傾けた方向を検出

付属基板をジョイスティックとして使おう

久詰 祐和

● 概要

本記事は本誌付属 ColdFire マイコン基板を使用した応用事例です．付属 ColdFire マイコン基板を左右に傾けることにより，別のシステム上で動作しているゲームを操作します．イメージとして近いのは，Ethernet 接続のジョイスティックでしょうか．

ゲームは，ColdFire プロセッサを搭載したワンボード・コンピュータ「CoFilint」（コフィリント，**写真 A**，プロファイア社製）上で動作し，サンプルとしてソース・コードが公開されているミニ・ゲーム「BungeePaPa」（バンジー・パパ）を使用します．CoFilint は Linux が稼働するパソコンとして使用できるように作られており，BungeePaPa はソース・コードを公開しているので，ほかの Linux が稼働するマシンでも比較的簡単に BungeePaPa を動作させられると思います．

付属 ColdFire マイコン基板と CoFilint を Ethernet で接続し，UDP（User Datagram Protocol）で通信を行います．

● 付属 ColdFire マイコン基板

付属 ColdFire マイコン基板を部品面が上になるように水平にします．そして左側に電源コネクタ（DC ジャック），右側に Ethernet コネクタ（RJ-45）がくるようにして，横長に持ちます．このとき，基板の左下に「CQ 出版社」の文字が正立（倒立の反対．つまり，ちゃんと読める状態）して見えます．この状態で加速度センサの軸は，左が－X，右が＋ X，手前が－ Y，奥が＋ Y，下が－ Z，上が＋ Z になります．

今回は，左右に動かす操作へ対応させるので，*X* 軸の加速度データを利用します．付属 ColdFire マイコン基板を左側が下，右側が上になるように傾けると，重力加速度がかかるため加速度センサの出力電圧が上がります．逆に，左側が上，右側が下になるように傾けると，重力加速度が逆にかかるため加速度センサの出力電圧が下がります．

● 付属 ColdFire マイコン基板側のプログラム

付属 ColdFire マイコン基板には ColdFire マイクロコントローラが搭載され，SilentC が走るようになっています．小規模なシステムであれば，この SilentC 上で簡単に素早くプログラムを作成し，実行できます．

リスト A と**リスト B** が付属 ColdFire マイコン基板側のプログラムです．**リスト A** は，加速度センサから加速度データを取得し，UDP で送信する関数 `udpsend` です．これを `ad` というファイル名で，付属 ColdFire マイコン基板側に格納します．**リスト B** は，関数 `udpsend` を呼び出すメイン関数 `main` です．送信先の IP アドレスはこの関数 `main` の引き数で指定します．実際のターゲット・マシンの IP アドレスを設定してください．同じように，これも `Main` というファイル名で付属 ColdFire マイコン基板側に格納します．

電源 ON と同時にこのプログラムを実行するように，レジストリ・ファイル `SilentC_Registry` に以下の 1 行を追加します．また既に `AutoRun` の行が存在する場合は，次のように変更します．

```
AutoRun = 1
```

これで，付属 ColdFire マイコン基板の電源を ON にすれば，加速度データを連続的にターゲット・マシンの 30049 番ポートに UDP で送信します．

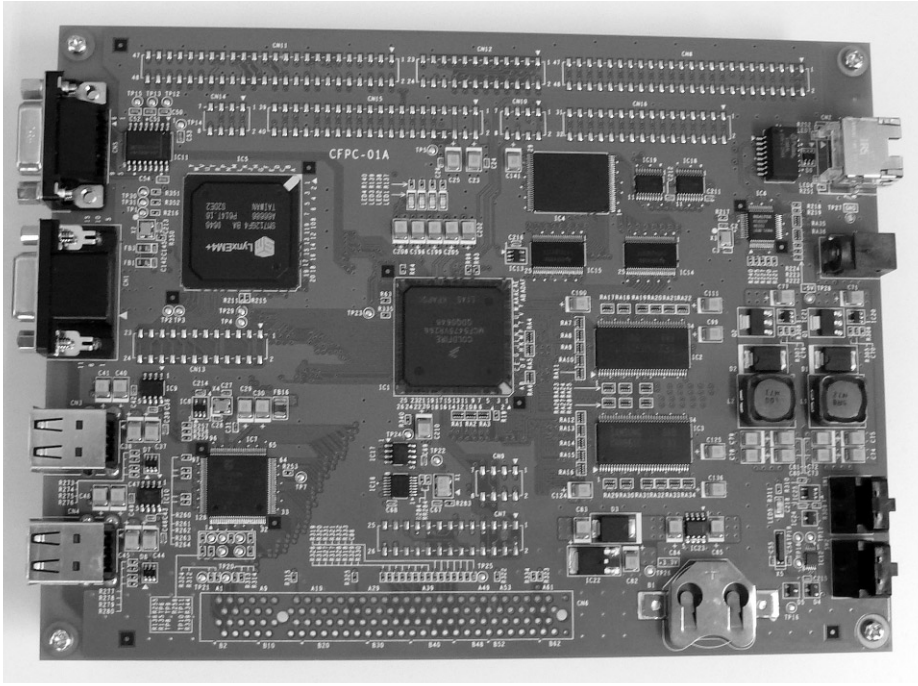

写真 A　ColdFire V4 コア・プロセッサ MCF5475 搭載ワンボード・コンピュータ「CoFilint」の外観

KEYWORD ―― 加速度センサ，ジョイスティック，CoFilint，UDP

```
udpsend(char *str)
{
  char *soc,*buf;                    //
  long ip;                           //
  ip=GetIP(str);                     // ターゲット・マシンのIPアドレスを取得する
  if(ip==0)return;                   // IPアドレスが取得できなければ，終了する
  buf=MemoryAlloc(10);               // 送信用バッファを10バイト確保する
  soc=CreateSocket(0);               // UDPソケットを生成する
  InitAd(0x70);                      // AN4-AN6をA/D変換として使用する
#stop 0                              // ブレーク・センスを禁止する(文字抜け防止)
  for(;;){                           //
    GetDigit(GetAd(4),buf);
                                     // AN4のA/D変換値を読み出し、文字列に変換する
    SendTo(soc,ip,30049,buf,StrLen(buf)+1);
                                     // ターゲット・マシンの30049番ポートに送信する
    if(Getc(0)=='q')break;           // qキーが押されていれば，終了する
  }                                  //
  CloseSocket(soc);                  // UDPソケットを破棄する
  MemoryFree(buf);                   // 送信用バッファを解放する
}
```

◀リストA
SilentCプログラム
～UDP送信のudpsend関数～(ad)

リストB　SilentCプログラム～メイン関数～(Main)

```
main()
{
  ad::udpsend("192.168.1.2");
                // ターゲット・マシンのIPアドレスを指定して
                // udpsendを実行する
}
```

● CoFilint側のプログラム

サンプル・プログラムとして，CoFilintにはBungeePaPaというミニ・ゲームが付いています．前述のようにこのゲームのソース・コードは公開されているので，これを元に改造します．付属ColdFireマイコン基板を左右に傾けることで，自キャラクタが左右に動くようにしましょう．

オリジナルではキー・スキャンを行い自キャラクタの座標を更新している個所を，UDPで受信した加速度データを付属ColdFireマイコン基板の左右の傾きとして捉え，その傾きに応じて自キャラクタの座標を更新するように変更します．

付属ColdFireマイコン基板から送られてくる加速度データを受け取るために必要なUDP通信の関数群を，新しいファイルudp.cppに作ります．**リストC**がそのプログラムになります．UDPソケットの生成の関数udpinit，終了の関数udpclose，UDPソケットから加速度データを受け取る関数udprecvを準備します．

これらの関数をゲームのメイン・ルーチンがあるmain.cppに追加します．**リストD**にその追加個所を示します．

リストC　CoFilint側プログラム(udp.cpp)

```
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/ioctl.h>
#include <netinet/in.h>

/* ポート番号 */
#define PORT 30049

/* バッファ・サイズ */
#define BUFFER_SIZE 2048

/* ソケット */
static int recvSocket;

void udpinit( void )
{
  struct sockaddr_in recvSockAddr;
  int val;

  /* sockaddr_in構造体の設定 */
  memset( &recvSockAddr, 0, sizeof( recvSockAddr ) );
  recvSockAddr.sin_port = htons( PORT );
  recvSockAddr.sin_family = AF_INET;
  recvSockAddr.sin_addr.s_addr = htonl( INADDR_ANY );

  /* ソケット生成 */
  recvSocket = socket( AF_INET, SOCK_DGRAM, 0 );

  /* バインド */
  bind( recvSocket, ( struct sockaddr * ) &recvSockAddr, sizeof(
recvSockAddr ) );

  /* 非ブロック・モードに設定 */
  val = 1;
  ioctl( recvSocket, FIONBIO, &val );
}

int udprecv( void )
{
  int pos = 0;
  int numrecv;
  char buffer[ BUFFER_SIZE ];
  char buffer2[ BUFFER_SIZE ];

  /* データ受信 */
  numrecv = recvfrom( recvSocket, buffer, BUFFER_SIZE, 0, NULL,
NULL );

  /* 受信バッファを空にする */
  while ( recvfrom( recvSocket, buffer2, BUFFER_SIZE, 0, NULL,
NULL ) > 0 );

  /* 数値に変換 */
  if ( numrecv > 0 ) {
    pos = strtol( buffer, NULL, 10 );
  } else {
    pos = -1;
  }

  return pos;
}

void udpclose( void )
{
  /* ソケット終了 */
  close( recvSocket );
}
```