カッターヘッド制御技術の開発　（第２報）

加工技術部　山田　圭、三谷和弘

1. はじめに

本研究は、横編地の裁断で行われているハサミを用いた手裁断を自動化するため、横編地自動裁断装置を開発するものである。今年度は、前年度試作した編地裁断用カッターヘッドを直交座標系ロボットに組込み、その制御方法について検討し、性能評価を行った。

2. 内　容

2.1　使用機器及び材料

(1)　カッターヘッド改造に係わるもの

○モータ：日本サーボ㈱製

直流モータ　DME44BB

- 出　力　　15 w
- 電　圧　　24 v
- トルク　　400 gcm
- 回転数　　3600 rpm

○エア関係：8φシリコンチューブ、8φアルミパイプ、8φアクリルパイプ他

○本体：アルミ板　他

(2)　カッターヘッド制御装置

○制御装置

- パーソナルコンピュータ：㈱ＮＥＣ製 PC-9821AS2/U8W
- ディスプレイ：㈱ＮＥＣ製 PC-KM153
- プリンタ：セイコーエプソン MJ-1000V2

○制御ソフト作成ツール

- Ｃ－コンパイラ：マイクロソフト㈱製 MS-C／C++

○インターフェイスユニット

- モータコントローラ：㈱インターフェイス製　AZI-7205

○モータ

- パルスモータ：オリエンタルモーター㈱製 PH544TG07-N
- パルスモータコントローラ：オリエンタルモーター㈱製　UPD5107N

(3)　直交座標系ロボット

- 直交座標系ロボット：㈱芝浦製作所製 MA3-A1R6050-31J （動作範囲 600㎜ × 500㎜）

2.2　直交座標系ロボットへのカッターヘッド組込み

直交座標系ロボットへのカッターヘッド組込みに際し、新たな機構の付加及びより適切な性能をもたらすための改造を行った。その結果を図１に示す。

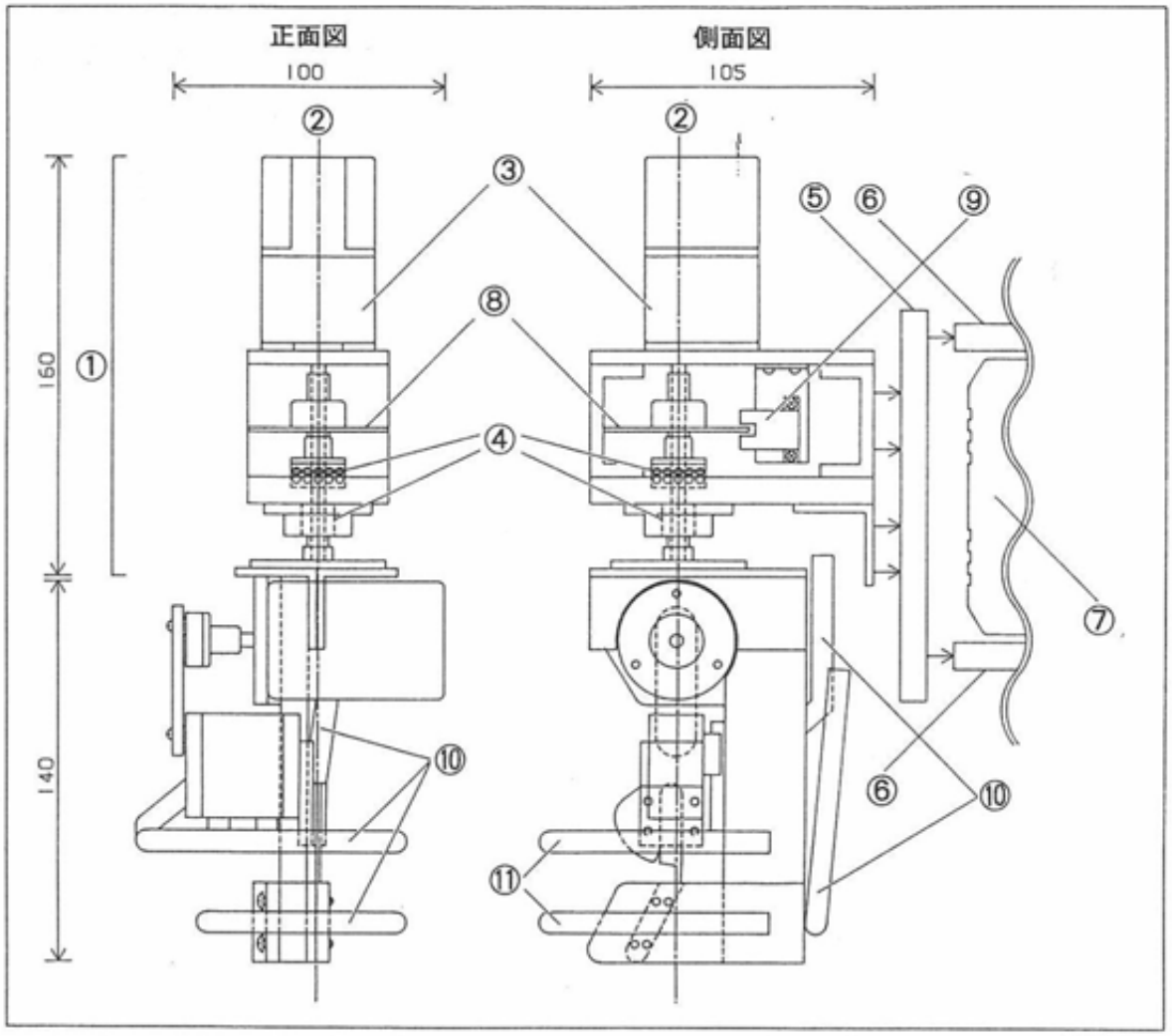
2.2.1　カッターヘッドの改造

昨年度開発したカッターヘッドの上メス駆動用モータはＡＣギヤードモータであったが、より小型で回転数の設定が容易に行えるＤＣモータに変更した。動力の伝達方式については、前年度のものはカッターヘッド本体をコンパクトにするためモータをクランクに直結せずに垂直に設置し、傘歯車で回転を変換させる方式としたが、これを取りやめ、モータをクランクに直結させることで高速化に対応した。

また、前年度開発したカッターヘッドは、Ｒ軸がカッターヘッド本体の端に位置しており、重心との隔たりが大きく慣性モーメントが大きいので、パルスモータの負担を柔らげる意味で、重心とＲ軸とのずれが小さくなるよう改造を行った。

2.2.2　Ｒ軸回転機構り設置

このカッターヘッドはオーバーロックミシンに使用されている上下メスによって裁断を行っており、裁断中はこのメスが常に裁断方向を向くよう



①R軸回転機構　②R軸　③パルスモータ　④ベアリング　⑤カッターヘッド取付板
⑥直交座標系ロボットスライダ　⑦直交座標系ロボットフレーム（Y軸）　⑧遮光板
⑨フォトセンサ　⑩エアパイプ　⑪エアノズル

図1　試作カッターヘッドNo.5及びR軸回転機構

制御する必要がある。そのためカッターヘッドの上部にパルスモータを用いたR軸回転機構を付加した。

2.2.3　R軸の原点復帰機構の設置

裁断装置始動時、カッターヘッドは必ずしも初期設定状態（メスが直交座標系ロボットのX軸に対し平行かつ正の方向を向いている状態）にあるとは限らないため、裁断作業開始前にカッターヘッドの原点復帰操作を行う必要がある。そのためR軸回転機構下部にフォトセンサと遮光板を用いたR軸の原点復帰機構を付加した。

2.3　横編地把持機構

カッターヘッドは上下メス式を採用しているため、裁断点は下メス把持部の高さだけ高くなる。そのため裁断しようとする横編地は一定の高さで保持されねばならない。その方法として、横編地を裁断する形よりひとまわり小さい形の台の上に乗せ、台からはみ出して垂れ下がった部分を裁断することとした。このとき、垂れ下がった編地を裁断点まで持ち上げる装置が必要となり、エアを用いた横編地把持機構を前年度作成した。しかし、直交座標系ロボットへの組み込みに際しての改造により昨年のガイドをそのまま使うことはできなくなり、新たにアクリルパイプを用いた横編地把持機構を作成した。また、横編地把持機構はエアで吹きとばすことによる裁断屑の除去機能をも有

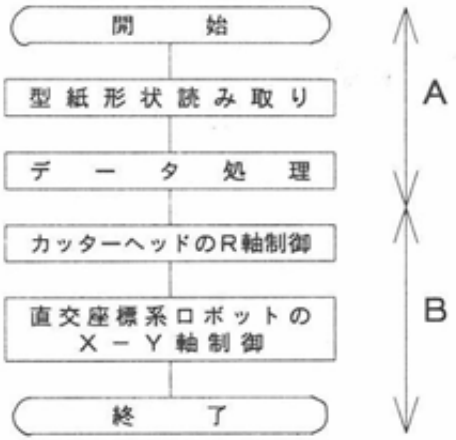


図2　ゼネラルフロー

している。

2.4　カッターヘッド制御ソフトの開発

2.4.1　カッターヘッドの位置及び方向制御

横編地自動裁断装置は座標読み取り装置から入力した型紙情報をパソコンで加工して、カッターヘッドが進むべきポイントの座標データを得る。そして、そのデータにより進行方向が決まってからカッターヘッドが移動していくという順序を保ちながら裁断を行う。

その概略のフローは図2のとおりで、区分Aはデータの入力及び加工、区分Bは裁断の実行を行

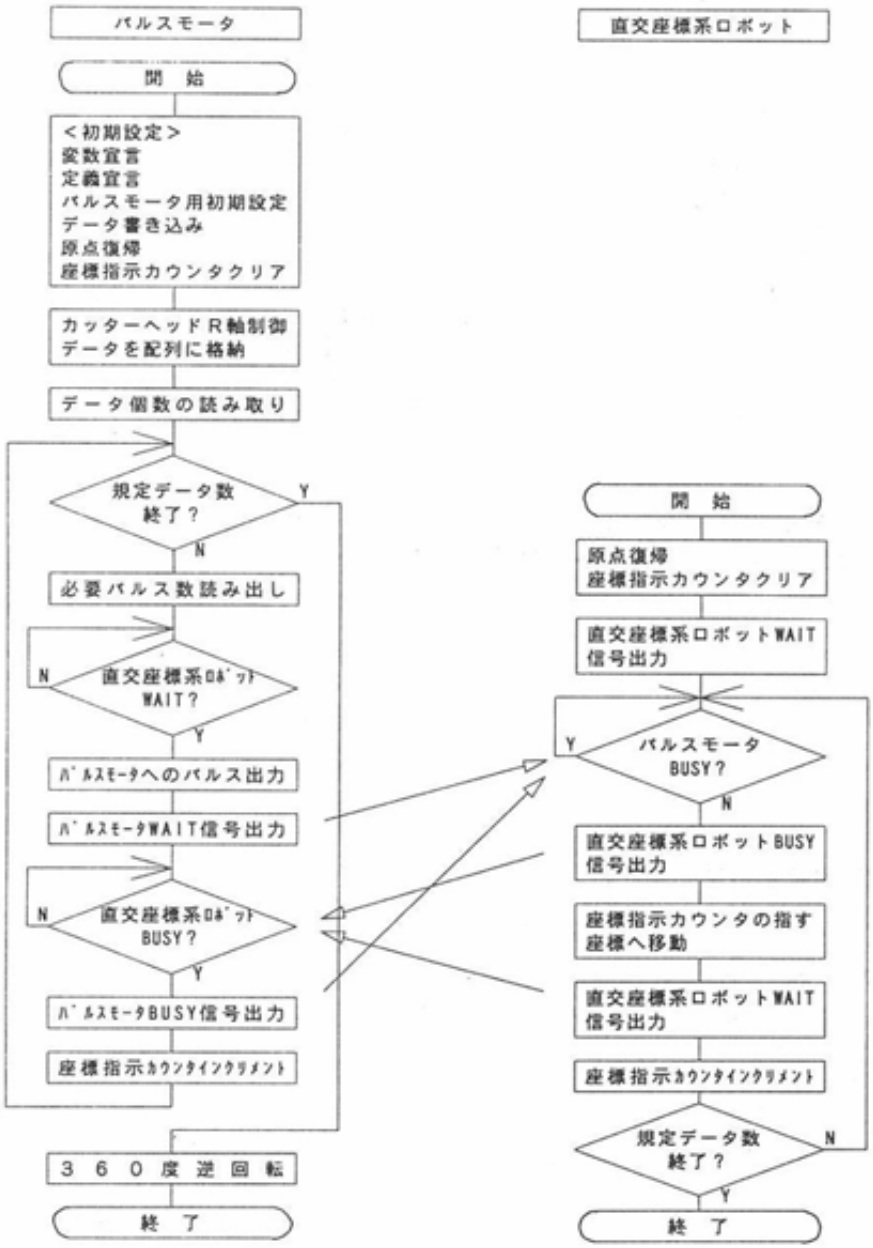


図3　パルスモータと直交座標系ロボットのシェークハンドによる制御のフロー

うが、今年度は区分Bについてのソフト開発を行った。裁断の実行に関する制御は、直交座標系ロボットの制御（X－Y軸制御）とパルスモータの回転制御（R軸制御）に分かれる。裁断にあたっては、パルスモータにパルスが送り出されている間、直交座標系ロボットは停止しており、パルスモータへのパルスの送り出しが終了してから直交座標系ロボットの移動に移る。その間パルスモータへのパルス送り出しは停止しているようR軸制御とX－Y軸制御はシェークハンドでコントロールされている。その様子を図3のフローに示す。そして図8にR軸制御プログラムのリストを、図9に直交座標系ロボットのX－Y軸制御プログラムのリストを示す。

2.4.2　座標データの書き込み

座標読み取り装置はまだ設置されてないので、裁断すべき形状の座標データは任意の形状を想定し、手作りで作成した。一例として直径24cmの円の外周を36等分したデータを図4に示す。1行目はデータ個数、2行目からが座標データで左側がX座標値、右側がY座標値である。ポイント数は36個であるが、スタートポイントからスタートして、再びスタートポイントに戻ってくるためデータ個数はポイント数＋1の37個となる。

```
0037
0065.10,0065.10
0081.10,0051.70
0099.30,0041.20
0118.90,0034.10
0139.50,0030.50
0160.50,0030.50
0181.10,0034.10
0200.70,0041.20
0218.80,0051.70
0234.90,0065.10
0248.30,0081.20
0269.50,01134.5
      :
      :
0030.50,0160.50
0030.50,0139.50
0034.10,0118.90
0041.20,0099.30
0051.70,0081.20
0065.10,0065.10
```

図4　裁断用座標テーブルの一例

これらのデータはパソコンでエディタにより書き込み、「裁断用座標テーブル」へ保存した。また、直交座標系ロボット側でも同じデータを必要とするので、シリアル通信回線によりパソコンから直交座標系ロボットのコントローラへ送信して記憶させた。

2.4.3　カッターヘッドの方向制御

カッターヘッドが現在のポイントから次のポイントへ向かうとき、その進行方向を次のポイントへ向かわせるようR軸に直結したパルスモータでその進行方向を決めている。進行方向を変えるときパルスモータへ送るパルス数の計算は次のようにして行った。

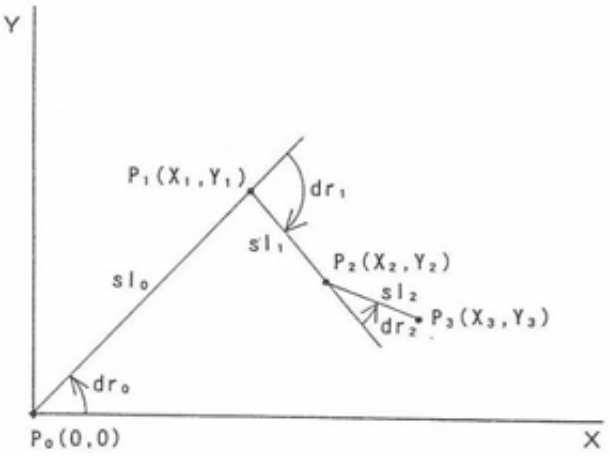


図5　R軸回転角度の算出方法

図5のようにカッターヘッドが例えば$P_1(X_1, Y_1)$から$P_2(X_2, Y_2)$へ向かうとき、その直線のX軸に対する勾配sl_1(deg)は次のようになる。

$$sl_1 = \tan^{-1} \frac{Y_2 - Y_1}{X_2 - X_1}$$

また、P_2から$P_3(X_3, Y_3)$へ向かうとき、その直線の勾配sl_2(deg)は次のようになる。

$$sl_2 = \tan^{-1} \frac{Y_3 - Y_2}{X_3 - X_2}$$

カッターヘッドがP_1からP_2に到着し、次にP_3へ向かうとき、その回転角度dr_2(deg) は次のように求められる。　$dr_2 = sl_2 - sl_1$

使用したパルスモータは0.1deg／stepであるため、dr_2(deg)だけ回転させるためには$dr_2 \times 10$個のパルスを送るようパルスモータインターフェイスへ指示すればよい。

2.4.4　R軸の原点復帰

前述のようにR軸は直交座標系ロボットX軸の正方向を基準に選び、その時の角度を0°とした。

裁断開始時にはカッターヘッドをX－Y軸及びR軸の原点に復帰させる。X－Y軸の原点復帰は

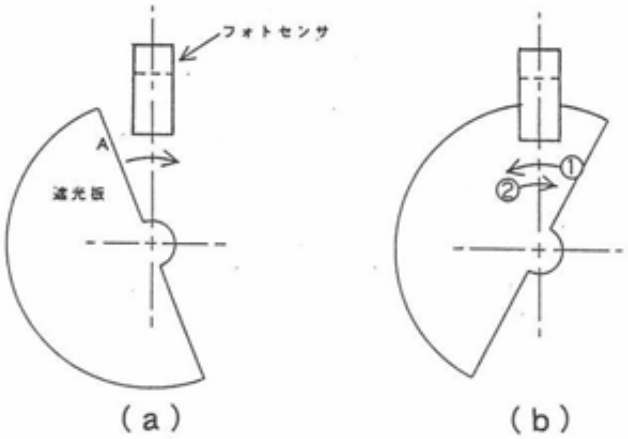


図6　R軸原点復帰の確認方法

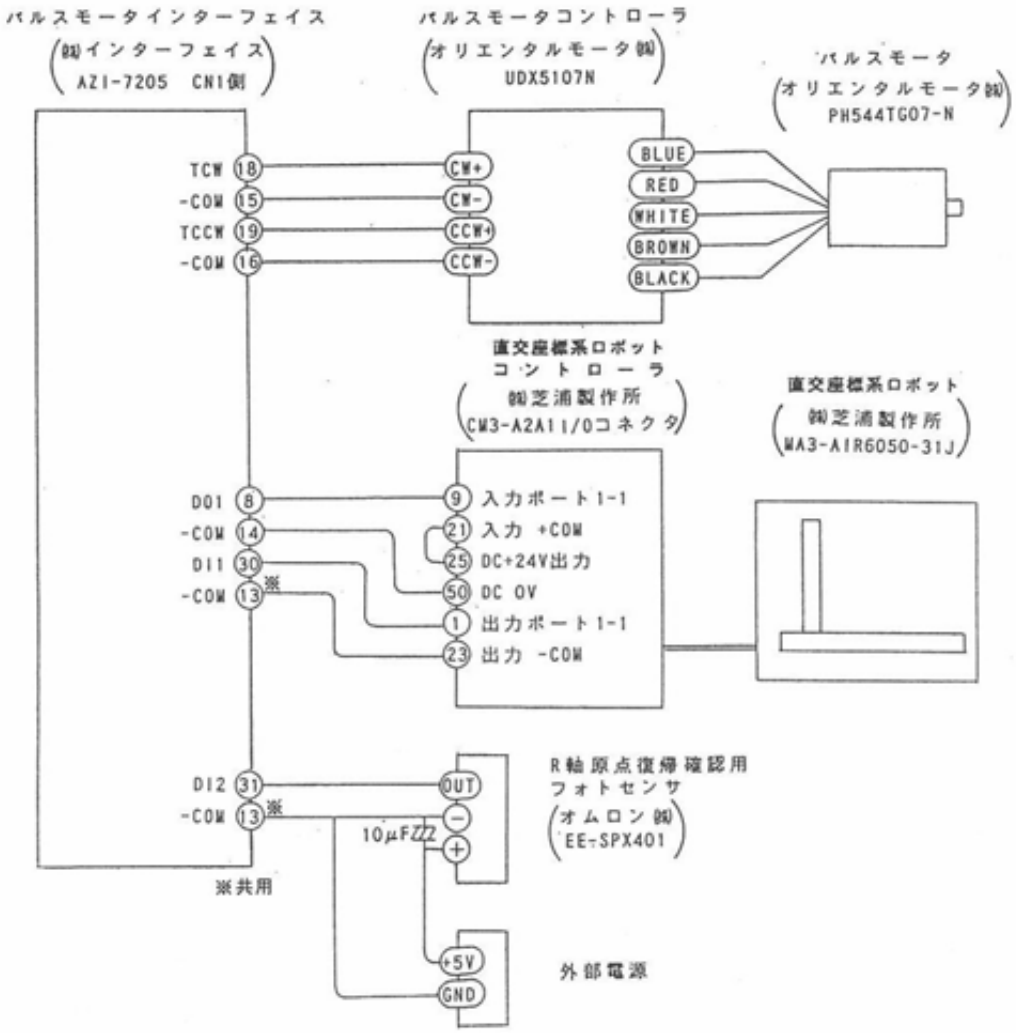


図7　パルスモータインターフェイスとパルスモータコントローラ、直交座標系ロボットコントローラ及びフォトセンサとの接続

直交座標系ロボットのコントローラの原点復帰コマンドによる。

またR軸の原点復帰は透過型のフォトセンサと遮光板を用いて次のように行った。

すなわち図6(a)のように遮光板がフォトセンサを遮光していないときは、遮光板を時計回りに回転させ、遮光板がフォトセンサを遮光したときをR軸の原点とした。一方、同図(b)のように遮光板がフォトセンサを遮光しているときは、遮光板を反時計回りに回転させ、遮光板がフォトセンサの光軸を通り過ぎ、光が透過するようになっても遮光板を止めないで、なおそのまま回転を続けさせる。光が透過して数ステップ回転させた後、今度は反転して時計回りに回転させ、(a)と同じように遮光板がフォトセンサを遮光したときを原点とした。

このとき、遮光板の前端Aとカッターヘッドの角度(進行方向)は一致するよう細かく調整しておく必要がある。

2.4.5　接　　続

パルスモータインターフェイスとパルスモータコントローラ、直交座標系ロボットコントローラ及びR軸原点復帰確認用フォトセンサとの接続は図7のとおりである。

2.5　実働試験

前述の直径24cmの円を36等分したデータを用い、実働試験を行った。試験には大まかに切り取った編地を使用し、その編地を、周囲にずれ止めの鋲を配した直径21cmの円の塩ビ板で挟み、さらにその上に重りを乗せ、裁断台に配置した。

試験の結果、原点から最初の点まで移動し、裁断を終えて再び原点に戻るまでにおよそ56秒要した。円や曲線を細かく分割したデータでは、直交座標系ロボット及びパルスモータが加減速運転を繰り返すこととなり、同じ距離を直線で裁断する場合と比較してロスが多いと考えられる。

裁断結果は良好で、均一かつ安定した裁断片が得られた。

3. ま と め

本年度は、横編地裁断の自動化技術に関する研究の2年目として、カッターヘッド制御技術の開発を行うとともに、仮想データによる実働試験を行った。この結果、手裁断に変わる自動機として十分な性能を有する装置が開発できた。しかし、一方で裁断形状によりカッターヘッド本体が裁断作業に支障をきたす場合があることも発見された。これは、内側に凹んだ形状を裁断する際、その形状が急な曲線、あるいは鋭角である場合、下メス把持部を支える支柱が編地もしくは編地を乗せている台に接触する現象で、これにはカッターヘッドのメスから支柱後部までの長さが関与している。あらゆる形状に対応させるためには、メスから支柱後部までの長さは可能な限り小さくした方が良いといえる。

今後の課題としては、以下の項目があげられる。

ア) パターンデータ入力ソフトの開発

デジタイザから型紙形状を効率よく入力するためのソフトを開発する。

イ) パターンデータの直交座標系ロボット用データへの変換方法検討。

デジタイザから入力された座標データを、直交座標系ロボットのX－Y軸制御ソフトに出力するためのフォーマットに加工する。

ウ) ディスプレイへの表示方法の検討。

入力されたデータと型紙との整合性を、カッターヘッド制御装置のディスプレイ上において確認する。

エ) カッターヘッド制御ソフトとパターンデータ入力ソフトとの調整。

データ入力から裁断までの作業が確実に行えるようカッターヘッド制御ソフトとパターンデータ入力ソフトとの調整を行う。

オ) 横編地裁断装置の総合調整

```
/*****************************************************************************
*   Y7205YPW5.C         ・座標データファイルから配列へ格納                      *
*                       ・配列から動作パルス数を読みとる                        *
*                       ・パルスモータ用初期設定書き込み及び定速動作を関数化    *
*                       ・パルス送信後のmotor停止のチェック                     *
*                       ・WA3とのハンドシェーク                                 *
*                       ・原点復帰                                              *
*                       ・座標データファイルは回転方向、送り出しパルス数        *
*                       ・.pedファイルからの読み出し                            *
*****************************************************************************/
#include        <stdio.h>
#include        <dos.h>
int dousa_h,dousa_m,dousa_l;/* パルスモータへのパルス出力数が入る */
int pntno;/* 座標データ個数 */
int pntcnt=1;/* 座標指示カウンタ */
int dr;       /*パルスモータ回転方向  dr>0→cw dr<0→ccw  0,1にしない理由:
                角度計算結の正、負を利用することによる*/
int pullsesuu;/* 送り出すパルス数 */
void initial_set(void);
void orgn(void);
void seqread(void);
int dec_hex(int *,int *,int *,int *);
int pullse(int);
int delay(int);
#define ADR     0x00d0
#define IBF     0x02
#define OBF     0x01
#define BUSY    0x04
#define CW_1    0x23            /* 初期設定コマンド    */
#define RA_WAI  255             /* 起動時パルスレート */
#define RA_WIN  50              /* 高速時パルスレート */
#define KAGEN_L 232             /* 加減速時パルス数（下位バイト）*/
#define KAGEN_H 3               /* 加減速時パルス数（上位バイト）*/
#define CW_6CW  0x44            /* 定速動作コマンド      CW */
#define CW_6CCW 0x4c            /* 定速動作コマンド CCW */
#define TEISOKU 30              /* 定速パルスレート      */
#define INPDT   0x01            /* 汎用入力ポートDI1=ON */
#define TURNDR  0x02            /* 汎用入力ポートDI2用 マスク */
struct point
        {       float x;        /* 回転方向 */
                int   y;        /* パルス数 */
        } pnt[512];

main()
{
/* パルスモータ用初期設定データ書き込み */
        initial_set();
/* 原点復帰 */
        printf("pulse mortor origin¥n");
        orgn();
      . printf("origin ok!");
/* 座標データファイルからデータを配列へ格納 */
        seqread();
/*
        /* データ個数の読み取り */
        pntno=pnt[0].y;
        printf("ポイント個数 - %d¥n",pntno);
/* 既定データ数終了？ */
        while ( pntcnt <= pntno )
        {
        /* ①パルス数読み出し */
                dr=pnt[pntcnt].x;
                pullsesuu=pnt[pntcnt].y;
                printf("¥nＰＯＩＮＴ Ｎｏ ＝［%d］ 回転方向(%d) パルス数(%d)¥n",p
ntcnt,dr,pullsesuu);
/* 10進→ 16進への変換及び出力すべきパルス数の書き込み */
                dec_hex(&pullsesuu,&dousa_h,&dousa_m,&dousa_l);
/* ②２軸ロボット作動中？ */
                while((inp(ADR+4) & INPDT) != 0);
/* ③パルスモータへのパルス出力 */
                pullse(dr);
                printf("pullse sending¥n");
/* パルスモーター作動終了？ */
                while((inp(ADR+2) & BUSY) != 0 );
                printf("motor stop ! ¥n");
/* ④パルスモータ停止信号出力 */
                outp(ADR+4,01);
                delay(200);
/* ⑥ */
                while((inp(ADR+4) & INPDT) == 01);
/* ⑤ */
                outp(ADR+4,0);
/* 座標指示カウンターインクリメント */
                pntcnt++;
        }
        outp(ADR+4,00);/*************************************************/
}
/*****************************************************************************
* パルスモータ用初期設定データ書き込み
*****************************************************************************/
void initial_set(void)
{
        /* 初期設定コマンド書き込み */
        while (((inp(ADR+2) & IBF) !=0)||(inp(ADR+2) & BUSY) !=0);
        outp(ADR+2,CW_1);
        /* 起動時パルスレートデータ書き込み */
        while ((inp(ADR+2) &IBF ) !=0);
        outp(ADR+0,RA_WAI);
        /* 高速時パルスレートデータ書き込み */
        while ((inp(ADR+2) &IBF ) !=0);
        outp(ADR+0,RA_WIN);
        /* 加減速パルス数（下位バイト）データ書き込み */
        while ((inp(ADR+2) &IBF ) !=0);
        outp(ADR+0,KAGEN_L);
        /* 加減速パルス数（上位バイト）データ書き込み */
        while ((inp(ADR+2) &IBF ) !=0);
        outp(ADR+0,KAGEN_H);
}
/*****************************************************************************
* 原点復帰
*****************************************************************************/
void orgn(void)
{
        int cond,i;
        cond=(inp(ADR+4) & TURNDR);
        printf("<1> %x¥n",inp(ADR+4),cond);
        if(cond == 0)/* 遮光ならif以下を実行*/
        {
                /*printf("遮光モードで%x¥n",cond);*/
                dr=1;
                for(i=1;i<=3600;i++)
                {
                        pullsesuu=1;
                        dec_hex(&pullsesuu,&dousa_h,&dousa_m,&dousa_l);
                        pullse(dr);
                        cond=(inp(ADR+4) & TURNDR);
                        if(cond == 2 ) break;
                        /*printf("しゃ光%x¥n",cond);*/
                }
                /*printf("追加ｃｃｗ¥n");*/
                pullsesuu=50;
                /*printf("a ");*/
                dec_hex(&pullsesuu,&dousa_h,&dousa_m,&dousa_l);
                /*printf("b¥n");*/
                pullse(dr);
/* パルスモーター作動終了？ */
                while ((inp(ADR+2) & BUSY) != 0 );
                /*printf("motor stop ! ¥n¥n¥n");*/
        }
                /*printf("入光モードで¥n");*/
                dr=-1;
                for(i=1;i<=3600;i++)
                {
                cond=(inp(ADR+4) & TURNDR);
                if(cond == 0 ) break;
                /*printf("入光%x¥n",cond);*/
                pullsesuu=1;
                dec_hex(&pullsesuu,&dousa_h,&dousa_m,&dousa_l);
                pullse(dr);
                }
}
/*****************************************************************************
* 座標データファイルからデータを配列に格納                                    *
*****************************************************************************/
void seqread(void)
{
char name[80];
char filename[80];
const char *d_drive = "d:";
char pedname[80];
FILE *fpped;
float fpx;
int   fpy;
int   i,ii;
        strcpy(filename,d_drive);
        printf("filename = %s¥n",filename);
```

図８－１　Ｒ軸制御プログラム

```
        printf("座標読み取りデータファイル名( .pedは不要)を入力してください¥n");
        scanf("%s",name);
        printf("name = %s¥n",name);
        strcat(filename,name);
        printf("filename = %s¥n",filename);
        strcpy(pedname,filename);
        printf("*0 pedname = %s¥n",pedname);
        strcat(pedname,".ped");
        printf("*1 pedname= %s¥n",pedname);
/*ファイルからの入力*/
    /* char *name = "d:point.dat"; */
    fpped = fopen(pedname,"r");
    if (fpped == NULL)
       printf("The file %s was not opened¥n",name);
    else
    {
        i = 0;
        while (fscanf(fpped,"%f,%d",&fpx,&fpy) != EOF)
        {
                    printf("%f¥t%d¥n",fpx,fpy);
           pnt[i].x=fpx;
           pnt[i].y=fpy;
                    i++;
        }
        fclose(fpped);
    }
}
/*********************************************************************
* 動作パルス数の dec→hex 変換及び出力すべきパルス数の書き込み
*********************************************************************/
int dec_hex(int *n,int *dousa_h,int *dousa_m,int *dousa_l)
{
        *dousa_h=(*n / 256)/256;
        *dousa_m=(*n - *dousa_h*256*256)/256;
        *dousa_l=*n - *dousa_h*256*256 - *dousa_m*256;
        return(*n,*dousa_h,*dousa_m,*dousa_l);
}
/*********************************************************************
*パルスモータへのパルス数出力(回転方向・定速動作・動作パルス数の書き込み)
*********************************************************************/
int pulse(int dr)
{
        /* 定速動作書き込み */
        if( dr >= 0 )
        {
        while ((inp(ADR+2) &IBF ) !=0);
        outp(ADR+2,CW_6CW);
        printf("dr>=0,cw¥n");
        }
        else
        {
        while ((inp(ADR+2) &IBF ) !=0);
        outp(ADR+2,CW_6CCW);
        printf("dr<0,ccw¥n");
        }
    /*  printf("e¥n");*/
        /* 定速パルスレート書き込み     */
        while ((inp(ADR+2) &IBF ) !=0);
        outp(ADR+0,TEISOKU);
    /*  printf("f¥n");*/
        /* 動作パルス数書き込み(下位バイト) */
        while ((inp(ADR+2) &IBF ) !=0);
        outp(ADR+0,dousa_l);
        /*printf("doudsa_l %x¥n",dousa_l);*/
        /* 動作パルス数書き込み(中位バイト) */
        while ((inp(ADR+2) &IBF ) !=0);
        outp(ADR+0,dousa_m);
        /*printf("dousa_m %x¥n",dousa_m);*/
        /* 動作パルス数書き込み(上位バイト) */
        while ((inp(ADR+2) &IBF ) !=0);
        outp(ADR+0,dousa_h);
        /*printf("dousa_h %x¥n",dousa_h);*/
}
/*********************************************************************
* DELAY
*********************************************************************/
int delay(int dly)
{
        int i;
        for(i=1;i<=dly;i++)
        printf("-");
}
```

図８－２　R軸制御プログラム続き

```
0001 SPD V=01
0002 CNT CN01=0001
0003 OUT PN1=.......0
0004 TAG TAG=001
0005 IN PN1=.......1
0006 OUT PN1=.......1
0007 MOVP a PT=000 CN=01 S V=00 POST
0008 OUT PN1=.......0
0009 CNT+ CN01=0001
0010 JMPC TAG=001 CN01<=0037
0011 MOV a X=+0000.00 Y=+0000.00 Z=+0000.00 R=+0000.00 S V=00 POST
0012 END
```

図９　直交座標系ロボットＸ－Ｙ軸制御プログラム