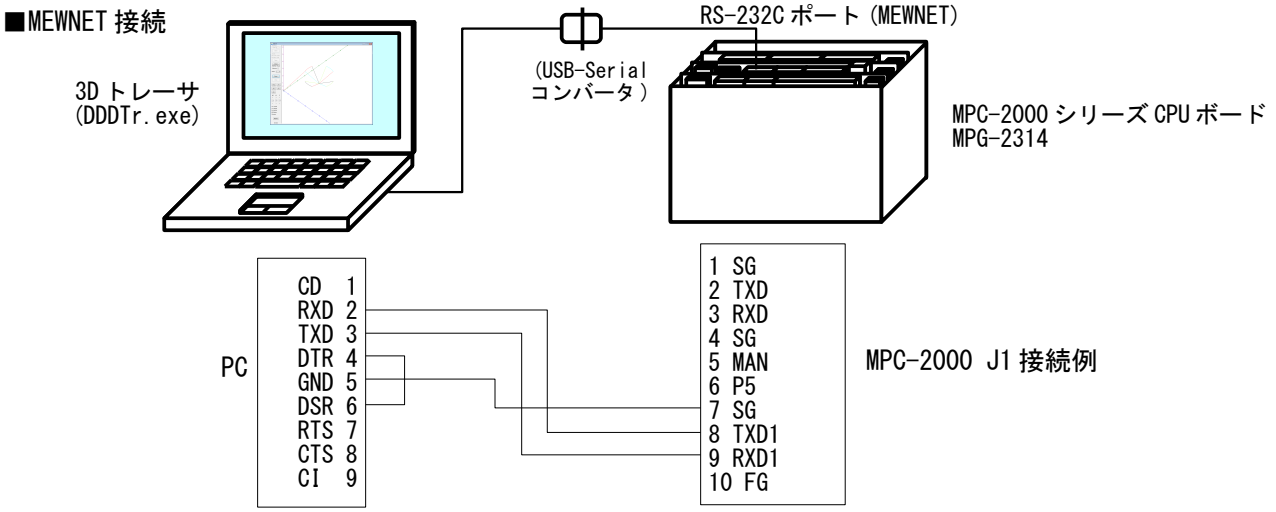
Technical Information	Ref No: ti2k-131031	Last Modify 131031
Title	3D トレーサを使った MPG-2314 動作シミュレーション	

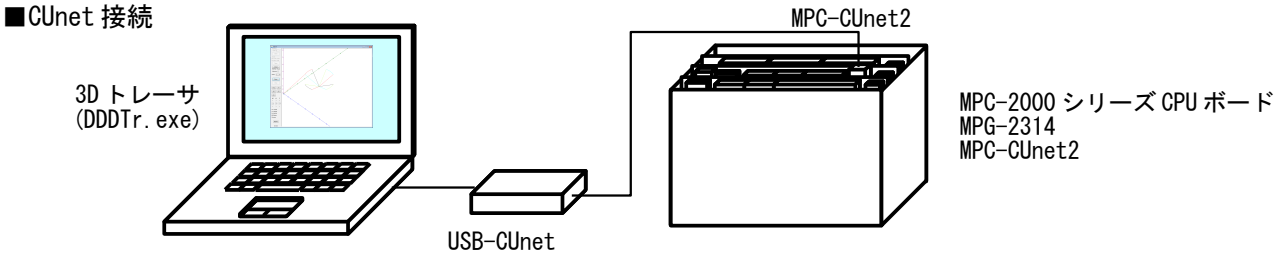
概要

- 3D トレーサは MPG-2314 の 3 軸動作を視覚的に表すアプリケーションです。
- 特に MOVT と RMVT の連続補間はコマンドを先読みするため、動作中の速度変更や I/O 操作のタイミングに注意が必要ですが、トレーサは工程を色分けして確認できます。
- シリアル通信のリアルタイム処理なので、追従速度は速くありませんが、机上で実機が無くても動作をシミュレーションすることができます。(MEWNET より CUnet の方が高速です)
- DDDTr.exe は MS VC++2008 Express 製なので、実行には MS .NET Framework3.5 が必要です。

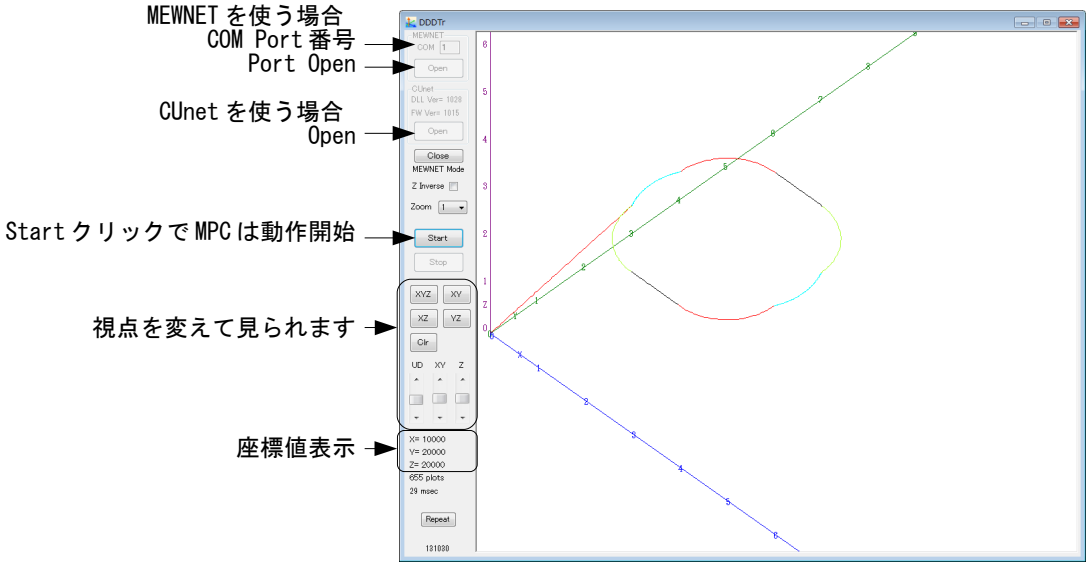
■MEWNET 接続



■CUnet 接続



■ トレーサの操作



MPC サンプルプログラム

```
/*********************************
/* 3D トレーサ(DDDTr.exe)サンプル
/*********************************

  QUIT_FORK 1 *MAIN
  END                                   /* メインタスク終了(実行中でもターミナル操作可能)

*MAIN

  CONST BLACK 0
  CONST RED 1
  CONST AQUA 2
  CONST GREEN 3
  CONST EMPTY 255
  MEWNET 38400                          /* Ch1 MEWNET 宣言
  CUNET  2 2 3                          /* CUNET グローバルエリア獲得
  PG 0                                  /* MPG-2314 DSW=0
  TIME 500
  QUIT_FORK 2 *COMMU_MEWNET             /* MEWNET で PC と通信
  cu_exist=@SW(2000)                    /* MPC-CUnet2 有無チェック、有れば cu_exist=1
  IF cu_exist THEN
    QUIT_FORK 3 *COMMU_CUNET            /* CUnet で PC と通信
  END_IF
  OFF 70000                             /* MEWNET のスタート
  OFF 2000                              /* CUNET のスタート
  PT=2                                  /* 動作パターン
  DO
    PRINT "Waiting Start"
    CLRPOS
    WAIT SW(70000)|(SW(2000)&cu_exist) /* PC からのスタート
    IF SW(70000)==1 THEN
      PR "MEWNET Mode"
      COMMU_MODE=0
      ach=10000
    ELSE
      PR "CUnet Mode"
      COMMU_MODE=1
      ach=20000
    END_IF

    SELECT_CASE VOID
      CASE SW(192)
        GOSUB *PATTERN_1
      CASE SW(193)
        GOSUB *PATTERN_2
      CASE SW(194)
        GOSUB *PATTERN_3
      CASE SW(195)
        GOSUB *PATTERN_4
      CASE_ELSE
        PR "???"
    END_SELECT

    TIME 100
    FIN=1                               /* PC へ動作終了フラグ
    ON 2255
    WAIT @SW(70000)&(@(SW(2000)&cu_exist))  /* PC のスタート OFF 待ち
    FIN=0
    OFF 2255
  LOOP
  END

/*************************************
/* MOVL による直線補間
/*  最もシンプルな動作
/*************************************
```

```
*PATTERN_1
  PR "PATTERN_1"

  pnt=0
  pnt=pnt+1 : SETP pnt 10000 20000 0 20000
  pnt=pnt+1 : SETP pnt 25000 25000 0 10000
  pnt=pnt+1 : SETP pnt 20000 10000 0 20000
  pnt=pnt+1 : SETP pnt 30000 10000 0 20000
  pnt=pnt+1 : SETP pnt 25000 25000 0 10000
  pnt=pnt+1 : SETP pnt 40000 20000 0 20000
  pnt=pnt+1 : SETP pnt 40000 30000 0 20000
  pnt=pnt+1 : SETP pnt 25000 25000 0 10000
  pnt=pnt+1 : SETP pnt 30000 40000 0 20000
  pnt=pnt+1 : SETP pnt 20000 40000 0 20000
  pnt=pnt+1 : SETP pnt 25000 25000 0 10000
  pnt=pnt+1 : SETP pnt 10000 30000 0 20000
  pnt=pnt+1 : SETP pnt 10000 20000 0 20000

  ACCEL ALL_A ach 1000 1000
  COLOR=RED
  MOVL P(1)                              /* 開始点に移動
  WAIT RR(ALL_A)==0

  ACCEL ALL_A ach 1000 ach               /* Max=Min で加減速無し
  FOR i=2 TO pnt
    COLOR=i%4
    MOVL P(i)
    WAIT RR(ALL_A)==0
  NEXT

  RETURN

/****************************************
/* MOVL の直線補間と MOVT の円弧補間を組み合わせた動作
/*  1ラインずつパルス発生終了を確認しています。
/****************************************
*PATTERN_2
  PR "PATTERN_2"

  pnt=0
  pnt=pnt+1 : SETP pnt 10000 20000 VOID  20000
  pnt=pnt+1 : SETP pnt 25000 25000 VOID  10000
  pnt=pnt+1 : SETP pnt 20000 10000 VOID  20000 /* 直線
  pnt=pnt+1 : SETP pnt 30000 10000 25000 25000 /* 円弧終点X,円弧終点Y,円弧中心X,円弧中心Y
  pnt=pnt+1 : SETP pnt 25000 25000 VOID  10000
  pnt=pnt+1 : SETP pnt 40000 20000 VOID  20000
  pnt=pnt+1 : SETP pnt 40000 30000 25000 25000
  pnt=pnt+1 : SETP pnt 25000 25000 VOID  10000
  pnt=pnt+1 : SETP pnt 30000 40000 VOID  20000
  pnt=pnt+1 : SETP pnt 20000 40000 25000 25000
  pnt=pnt+1 : SETP pnt 25000 25000 VOID  10000
  pnt=pnt+1 : SETP pnt 10000 30000 VOID  20000
  pnt=pnt+1 : SETP pnt 10000 20000 25000 25000

  ACCEL ALL_A ach 1000 1000
  COLOR=RED
  MOVL P(1)
  WAIT RR(ALL_A)==0

  ACCEL ALL_A ach 1000 ach
  axis=X_A|Y_A
  DS_DACL
  FOR i=2 TO pnt
   COLOR=i%4
   IF U(i)<>VOID THEN
      MOVT axis P(i) CCW                 /* 円弧 反時計回り
    ELSE
      MOVL P(i)                          /* 直線
    END_IF
    WAIT RR(ALL_A)==0                    /* 1ライン終了待ち
```

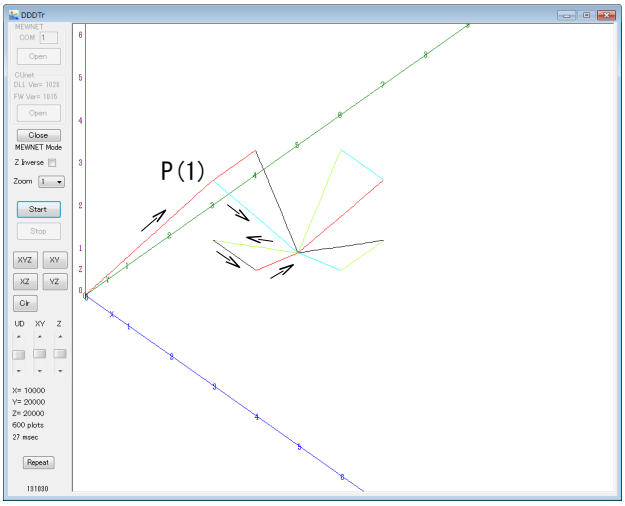
```
  NEXT
  EN_DACL

  RETURN

/*************************************
/* MOVTによる円弧と直線の連続補間
/*  [Note]
/*  *PATTERN_2のように、1ライン終了待ちのWAIT RR()を入れないと
/*  MOVTを1つ先読みして、2つ先のMOVTで待ち状態になります。
/*    MOVT 動作中  <--  ここを実行するとき COLOR=3 となる
/*    COLOR=2%4       |
/*    MOVT 先読み     |
/*    COLOR=3%4 ------
/*    MOVT ここで待ち状態
/*    COLOR=4%4
/*************************************
*PATTERN_3
  PR "PATTERN_3"

  pnt=0
  pnt=pnt+1 : SETP pnt 10000 20000 0     20000
  pnt=pnt+1 : SETP pnt 20000 10000 20000 20000 /* 円弧終点X,円弧終点Y,円弧中心X,円弧中心Y
  pnt=pnt+1 : SETP pnt 30000 10000 VOID  VOID  /* ここは直線
  pnt=pnt+1 : SETP pnt 40000 20000 30000 20000
  pnt=pnt+1 : SETP pnt 40000 30000 30000 25000
  pnt=pnt+1 : SETP pnt 30000 40000 30000 30000
  pnt=pnt+1 : SETP pnt 20000 40000 VOID  VOID
  pnt=pnt+1 : SETP pnt 10000 30000 20000 30000
  pnt=pnt+1 : SETP pnt 10000 20000 20000 25000
  pnt=pnt+1 : SETP pnt 10000 20000 VOID  VOID   /* ダミー終点

  ACCEL ALL_A ach 1000 1000
  COLOR=RED
  MOVL P(1)
  WAIT RR(ALL_A)==0

  axis=X_A|Y_A
  DS_DACL
  FOR i=2 TO pnt
    IF U(i)<>VOID THEN
      MOVT axis P(i) CCW                /* 円弧 反時計回り
    ELSE
      MOVT axis P(i)                    /* 直線
    END_IF
    COLOR=i%4
    IF COLOR==AQUA THEN                 /* 特定の場所でスピードを変える
      ACCEL ALL_A 2000 1000 2000        /* 低速部分
    ELSE
      ACCEL ALL_A ach 1000 ach
    END_IF
  NEXT
  WAIT RR(ALL_A)==0
  EN_DACL

  RETURN

/*************************************
/* RMVTによる3軸連続直線補間
/*   RMVTの先読みは *PATTERN_3 と同じです。
/*   3軸の円弧補間はできません。
/*************************************
*PATTERN_4
  PR "PATTERN_4"

  pnt=0
  pnt=pnt+1 : SETP pnt 10000 20000 0 20000 /* U軸は速度のパラメータとして使用
  pnt=pnt+1 : SETP pnt 25000 25000 0 10000
  pnt=pnt+1 : SETP pnt 20000 10000 0 20000
  pnt=pnt+1 : SETP pnt 30000 10000 1 20000
```

```
  pnt=pnt+1 : SETP pnt 25000 25000 0 10000
  pnt=pnt+1 : SETP pnt 40000 20000 0 20000
  pnt=pnt+1 : SETP pnt 40000 30000 1 20000
  pnt=pnt+1 : SETP pnt 25000 25000 0 10000
  pnt=pnt+1 : SETP pnt 30000 40000 0 20000
  pnt=pnt+1 : SETP pnt 20000 40000 1 20000
  pnt=pnt+1 : SETP pnt 25000 25000 0 10000
  pnt=pnt+1 : SETP pnt 10000 30000 0 20000
  pnt=pnt+1 : SETP pnt 10000 20000 1 20000
  pnt=pnt+1 : SETP pnt 10000 20000 0 20000  /* ダミー終点

  ACCEL ALL_A ach 1000 1000
  COLOR=RED
  MOVL P(1)
  WAIT RR(ALL_A)==0

  axis=X_A|Y_A|Z_A
  DS_DACL
  FOR i=2 TO pnt
    dx=X(i)-X(i-1)
    dy=Y(i)-Y(i-1)
    dz=Z(i)-Z(i-1)
    RMVT axis dx dy dz
    COLOR=i%4
    IF U(i-1)==1 THEN                  /* U(n)=1 の線を描くときはスピードを変える
      ACCEL ALL_A 2000 1000 2000       /* 低速部分
    ELSE
      ACCEL ALL_A ach 1000 ach
    END_IF
  NEXT
  WAIT RR(ALL_A)==0
  EN_DACL

  RETURN

/**************************************
/*  各座標範囲 0～65535(&HFFFF)
/*  2Byte 以上の場合は縮小する
/**************************************
*COMMU_MEWNET
  PG 0
  REDUCTION=1                /* 縮小率
  DO
      X_=(X(0)/REDUCTION)&&HFFFF
      Y_=((Y(0)/REDUCTION)&&HFFFF)<<16
      Z_=(Z(0)/REDUCTION)&&HFFFF
      C_=(COLOR&&HFFFF)<<16
      C_=(FIN<<24)+C_        /* 作業終了フラグ
      MBK(1000~Lng)=X_+Y_    /* 下位 2byte:X 座標, 上位 2byte:Y 座標
      MBK(1002~Lng)=Z_+C_    /* 下位 2byte:Z 座標, 上位 2byte:色,動作終了フラグ
      SWAP
  LOOP

/**************************************
/* 各座標 4バイト長(マイナス座標も可。ただし PC 側の表示範囲に制約有り)
/* CUnet フローバルメモリ引き当て
/* PC  SA0,SA1
/* MPC SA2,SA3
/**************************************
*COMMU_CUNET
  PG 0
  REDUCTION=1                /* 縮小率
  DO
      OUT COLOR 2028
      OUT X(0)/REDUCTION 2016~Lng
      OUT Y(0)/REDUCTION 2020~Lng
      OUT Z(0)/REDUCTION 2024~Lng
      SWAP
  LOOP
```

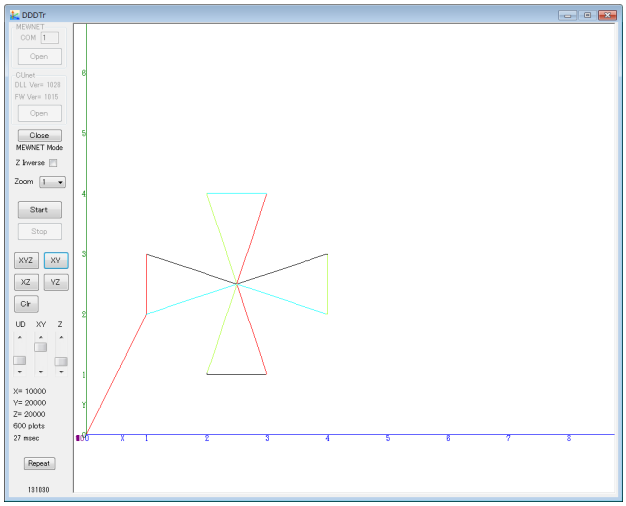
トレース画面

■ PATTERN_1　MOVL による直線補間

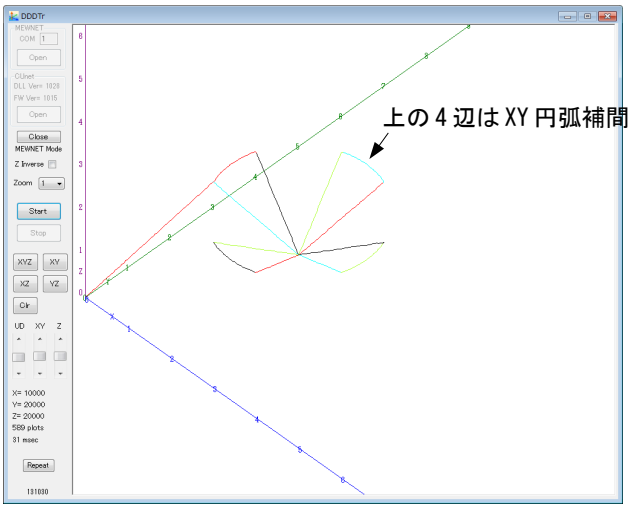
XYZ

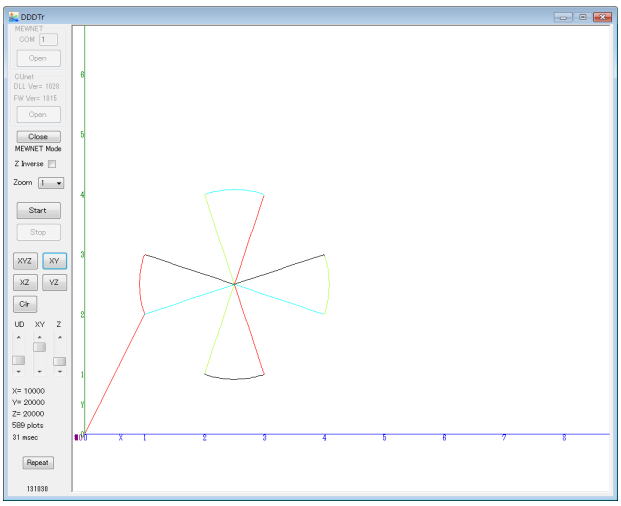


XY

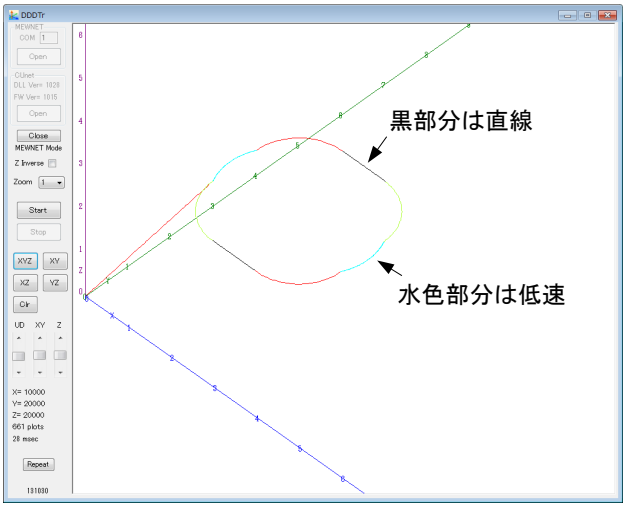


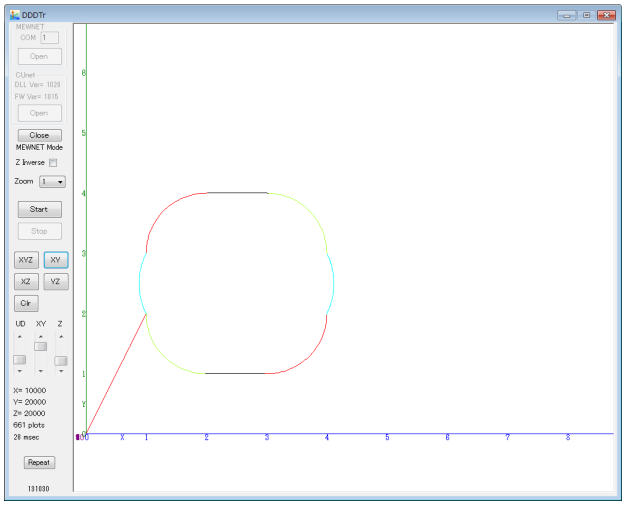
■ PATTERN_2　MOVL の直線補間と MOVT の円弧補間を組み合わせた動作



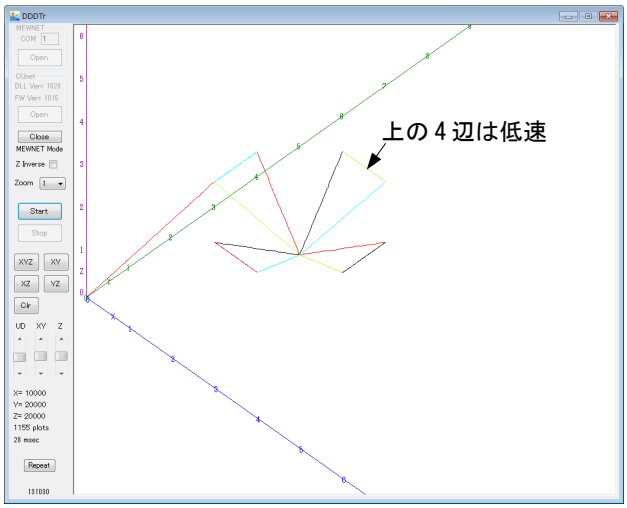


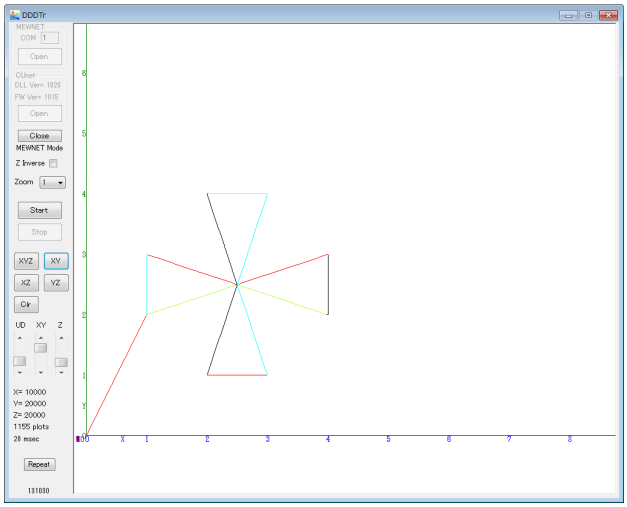
■ PATTERN_3　MOVT による円弧と直線の連続補間



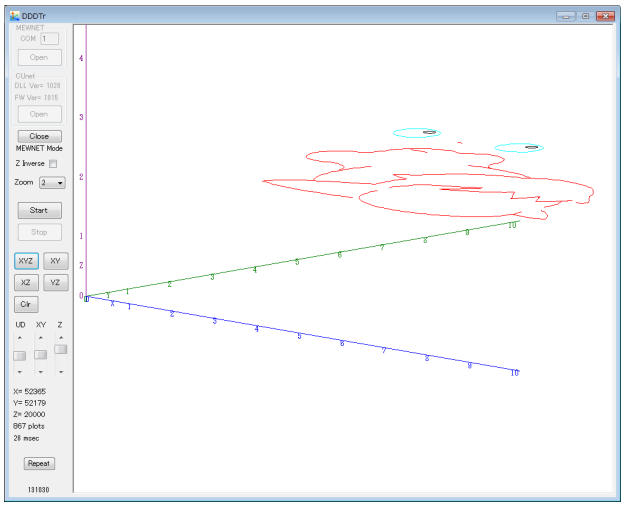


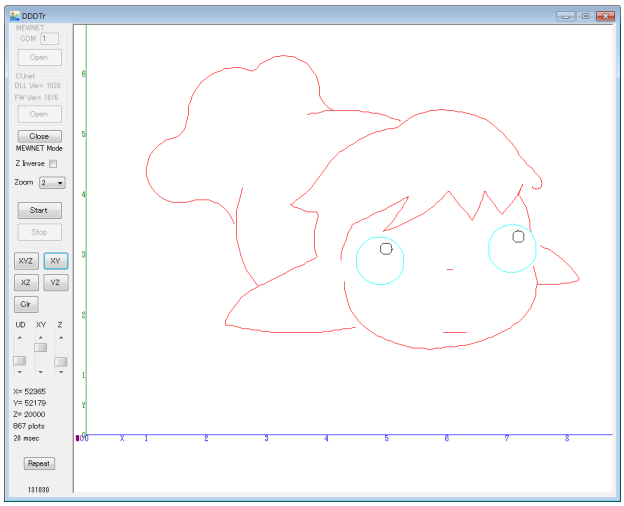
■ PATTERN_4　RMVT による 3 軸連続直線補間





■ サンプルプログラムの中にはありませんが、円弧補間の組み合わせで絵も描けます。





サンプルプログラムと動画

MPC サンプルプログラムと動画は、MPC-2000 BBS
http://acceleng.jp/_bbs35_2000/bbs35.cgi?article=158
からご覧いただけます。