```
/* Parallel port:
The following definitions might be helpfull for the actuall device, but isn't implemented in the prototype.
% inputs Parallel port -> PIC: (digital I/O only, allowing for TTL characteristics)
*/

#include <p18f67j60.h>

#define UILED    PORTBbits.RB4  //LED on PIC-WEB board
#define UIBUTTON PORTBbits.RB0  //Button on PIC-WEB board

#define PLP1  PORTEbits.RE1  //X-step
#define PLP2  PORTEbits.RE0  //X-direction
#define PLP3  PORTEbits.RE3  //Y-step
#define PLP4  PORTEbits.RE2  //Y-direction
#define PLP5  PORTGbits.RG4  //Z-step
#define PLP6  PORTEbits.RE4  //Z-direction

#define M1ap  PORTEbits.RE5  //Sig1p = RE5
#define M1an  PORTFbits.RF1  //Sig1n = RF1
#define M1bp  PORTAbits.RA5  //M1b   = RA5
#define M1bn  PORTFbits.RF2  //M1a   = RF2

#define M2ap  PORTBbits.RB3  //Sig2p = RB3
#define M2an  PORTDbits.RD0  //Sig2n = RD0
#define M2bp  PORTAbits.RA3  //M2b   = RA3
#define M2bn  PORTFbits.RF5  //M2a   = RF5

#define M3an  PORTDbits.RD2  //M3b   = RD2 (old S3)
#define M3ap  PORTBbits.RB2  //Sig3n = RB2 (old M3n)
#define M3bn  PORTCbits.RC2  //M3a   = RC2
#define M3bp  PORTBbits.RB1  //Sig3p = RB1 (old M3p)

//Global Variables
unsigned int MOTOR1POS = 1;
unsigned int MOTOR2POS = 1;
unsigned int MOTOR3POS = 1;

void           init ( void );
void           x_motor( unsigned short step, unsigned short direction);
void           y_motor( unsigned short step, unsigned short direction);
void           z_motor( unsigned short step, unsigned short direction);


void main(void) {

   unsigned int blinkduration = 60000;
   unsigned int ia = 0;
   unsigned int ib = 0;
   unsigned int ic = 0;
   unsigned int oldPLP1 = 0;
   unsigned int oldPLP3 = 0;
   unsigned int oldPLP5 = 0;

   init();
   M1ap = 0;
   M1an = 0;
   M1bp = 0;
   M1bn = 0;

   M2ap = 0;
   M2an = 0;
   M2bp = 0;
   M2bn = 0;

   M3ap = 0;
   M3an = 0;
   M3bp = 0;
   M3bn = 0;


//   x_motor(1,0);
```

```
//   x_motor(1,1); //Move twice, to ensure the Motor is in a deffined state
//   y_motor(1,0);
//   y_motor(1,1); //Move twice, to ensure the Motor is in a deffined state
//   z_motor(1,0);
//   z_motor(1,1); //Move twice, to ensure the Motor is in a deffined state


   for ( ia = 0; 1; ia = ia + 1 ) {


      //X-Motor controll
      if ( PLP1 != oldPLP1 ) { //Check if the last step signal of this motor has finished.

         if ( PLP1 == 1 ) { //Turn motor

            x_motor( 1, PLP2 ); //PLP2 determines direction

         }

         oldPLP1 = PLP1;

      }


      //Y-Motor controll
      if ( PLP3 != oldPLP3 ) { //Check if the last step signal of this motor has finished.

         if ( PLP3 == 1 ) { //Turn motor

            y_motor( 1, PLP4 ); //PLP4 determines direction

         }

         oldPLP3 = PLP3;

      }


      //X-Motor controll
      if ( PLP5 != oldPLP5 ) { //Check if the last step signal of this motor has finished.

         if ( PLP5 == 1 ) { //Turn motor

            z_motor( 1, PLP6 ); //PLP2 determines direction

         }

         oldPLP5 = PLP5;

      }


//Optical response to see if it is still running
      if ( ia >= blinkduration ) {

         ia = 0;
         if ( UILED ) {

            UILED = 0;  //Set LED

         } else {

            UILED = 1;  //Clear LED

         }

      }


      if ( UIBUTTON == 0 ) {    //If Button is pressed, turn off motors
```

```
            x_motor(0,0);
            y_motor(0,0);
            z_motor(0,0);


      }

   }

}



void           x_motor( unsigned short step, unsigned short direction) {

        unsigned int M1p, M1n, M2p, M2n;

        M1ap = 0;                       //First reset all
        M1an = 0;                       //First reset all
        M1bp = 0;                       //First reset all
        M1bn = 0;                       //First reset all

        if ( step == 1 && direction == 0 ) { // Turning CW
/*/ //Single Stepping
                if ( MOTOR1POS == 1 ) { //Enable the motor required
                   M1ap = 1;
                   MOTOR1POS = MOTOR1POS + 1;
                } else if ( MOTOR1POS == 2 ) {
                   M1bp = 1;
                   MOTOR1POS = MOTOR1POS + 1;
                } else if ( MOTOR1POS == 3 ) {
                   M1an = 1;
                   MOTOR1POS = MOTOR1POS + 1;
                } else {
                   M1bn = 1;
                   MOTOR1POS = 1;
                }
//*/
//*/ //Micro Stepping
                if ( MOTOR1POS == 1 ) { //Enable the motor required
                   M1ap = 1;
                   MOTOR1POS = MOTOR1POS + 1;
                } else if ( MOTOR1POS == 2 ) {
                   M1ap = 1;
                   M1bp = 1;
                   MOTOR1POS = MOTOR1POS + 1;
                } else if ( MOTOR1POS == 3 ) {
                   M1bp = 1;
                   MOTOR1POS = MOTOR1POS + 1;
                } else if ( MOTOR1POS == 4 ) {
                   M1bp = 1;
                   M1an = 1;
                   MOTOR1POS = MOTOR1POS + 1;
                } else if ( MOTOR1POS == 5 ) {
                   M1an = 1;
                   MOTOR1POS = MOTOR1POS + 1;
                } else if ( MOTOR1POS == 6 ) {
                   M1an = 1;
                   M1bn = 1;
                   MOTOR1POS = MOTOR1POS + 1;
                } else if ( MOTOR1POS == 7 ) {
                   M1bn = 1;
                   MOTOR1POS = MOTOR1POS + 1;
                } else if ( MOTOR1POS == 8 ) {
                   M1bn = 1;
                   M1ap = 1;
                   MOTOR1POS = 1;
                }
//*/
        } else if ( step == 1 && direction == 1 ) { // Turning CCW
/*/ //Single Stepping
```

```
                if ( MOTOR1POS == 1 ) { //Enable the motor required
                   M1an = 1;
                   MOTOR1POS = 4;
                } else if ( MOTOR1POS == 2 ) {
                   M1bn = 1;
                   MOTOR1POS = MOTOR1POS - 1;
                } else if ( MOTOR1POS == 3 ) {
                   M1ap = 1;
                   MOTOR1POS = MOTOR1POS - 1;
                } else {
                   M1bp = 1;
                   MOTOR1POS = MOTOR1POS - 1;
                }
//*/
//*/ //Micro Stepping
                if ( MOTOR1POS == 1 ) { //Enable the motor required
                   M1an = 1;
                   MOTOR1POS = 8;
                } else if ( MOTOR1POS == 2 ) {
                   M1an = 1;
                   M1bn = 1;
                   MOTOR1POS = MOTOR1POS - 1;
                } else if ( MOTOR1POS == 3 ) {
                   M1bn = 1;
                   MOTOR1POS = MOTOR1POS - 1;
                } else if ( MOTOR1POS == 4 ) {
                   M1bn = 1;
                   M1ap = 1;
                   MOTOR1POS = MOTOR1POS - 1;
                } else if ( MOTOR1POS == 5 ) {
                   M1ap = 1;
                   MOTOR1POS = MOTOR1POS - 1;
                } else if ( MOTOR1POS == 6 ) {
                   M1ap = 1;
                   M1bp = 1;
                   MOTOR1POS = MOTOR1POS - 1;
                } else if ( MOTOR1POS == 7 ) {
                   M1bp = 1;
                   MOTOR1POS = MOTOR1POS - 1;
                } else {
                   M1bp = 1;
                   M1an = 1;
                   MOTOR1POS = MOTOR1POS - 1;
                }
//*/
        }

}


void            y_motor( unsigned short step, unsigned short direction) {

        unsigned int M1p, M1n, M2p, M2n;

    M2ap = 0;                   //First reset all
    M2an = 0;                   //First reset all
    M2bp = 0;                   //First reset all
    M2bn = 0;                   //First reset all

        if ( step == 1 && direction == 0 ) { // Turning CW
/*/ //Single Stepping
                if ( MOTOR2POS == 1 ) { //Enable the motor required
                   M2ap = 1;
                   MOTOR2POS = MOTOR2POS + 1;
                } else if ( MOTOR2POS == 2 ) {
                   M2bp = 1;
                   MOTOR2POS = MOTOR2POS + 1;
                } else if ( MOTOR2POS == 3 ) {
                   M2an = 1;
                   MOTOR2POS = MOTOR2POS + 1;
                } else {
```

```
                M2bn = 1;
                MOTOR2POS = 1;
            }
//*/
//*/ //Micro Stepping
            if ( MOTOR2POS == 1 ) { //Enable the motor required
                M2ap = 1;
                MOTOR2POS = MOTOR2POS + 1;
            } else if ( MOTOR2POS == 2 ) {
                M2ap = 1;
                M2bp = 1;
                MOTOR2POS = MOTOR2POS + 1;
            } else if ( MOTOR2POS == 3 ) {
                M2bp = 1;
                MOTOR2POS = MOTOR2POS + 1;
            } else if ( MOTOR2POS == 4 ) {
                M2bp = 1;
                M2an = 1;
                MOTOR2POS = MOTOR2POS + 1;
            } else if ( MOTOR2POS == 5 ) {
                M2an = 1;
                MOTOR2POS = MOTOR2POS + 1;
            } else if ( MOTOR2POS == 6 ) {
                M2an = 1;
                M2bn = 1;
                MOTOR2POS = MOTOR2POS + 1;
            } else if ( MOTOR2POS == 7 ) {
                M2bn = 1;
                MOTOR2POS = MOTOR2POS + 1;
            } else if ( MOTOR2POS == 8 ) {
                M2bn = 1;
                M2ap = 1;
                MOTOR2POS = 1;
            }
//*/
        } else if ( step == 1 && direction == 1 ) { // Turning CCW
/*/ //Single Stepping
            if ( MOTOR2POS == 1 ) { //Enable the motor required
                M2an = 1;
                MOTOR2POS = 4;
            } else if ( MOTOR2POS == 2 ) {
                M2bn = 1;
                MOTOR2POS = MOTOR2POS - 1;
            } else if ( MOTOR2POS == 3 ) {
                M2ap = 1;
                MOTOR2POS = MOTOR2POS - 1;
            } else {
                M2bp = 1;
                MOTOR2POS = MOTOR2POS - 1;
            }
//*/
//*/ //Micro Stepping
            if ( MOTOR2POS == 1 ) { //Enable the motor required
                M2an = 1;
                MOTOR2POS = 8;
            } else if ( MOTOR2POS == 2 ) {
                M2an = 1;
                M2bn = 1;
                MOTOR2POS = MOTOR2POS - 1;
            } else if ( MOTOR2POS == 3 ) {
                M2bn = 1;
                MOTOR2POS = MOTOR2POS - 1;
            } else if ( MOTOR2POS == 4 ) {
                M2bn = 1;
                M2ap = 1;
                MOTOR2POS = MOTOR2POS - 1;
            } else if ( MOTOR2POS == 5 ) {
                M2ap = 1;
                MOTOR2POS = MOTOR2POS - 1;
            } else if ( MOTOR2POS == 6 ) {
                M2ap = 1;
```

```
                    M2bp = 1;
                    MOTOR2POS = MOTOR2POS - 1;
                } else if ( MOTOR2POS == 7 ) {
                    M2bp = 1;
                    MOTOR2POS = MOTOR2POS - 1;
                } else {
                    M2bp = 1;
                    M2an = 1;
                    MOTOR2POS = MOTOR2POS - 1;
                }
//*/
        }

}


void            z_motor( unsigned short step, unsigned short direction) {

        unsigned int M1p, M1n, M2p, M2n;

    M3ap = 0;                   //First reset all
    M3an = 0;                   //First reset all
    M3bp = 0;                   //First reset all
    M3bn = 0;                   //First reset all

        if ( step == 1 && direction == 0 ) { // Turning CW
/*/ //Single Stepping
                if ( MOTOR3POS == 1 ) { //Enable the motor required
                    M3ap = 1;
                    MOTOR3POS = MOTOR3POS + 1;
                } else if ( MOTOR3POS == 2 ) {
                    M3bp = 1;
                    MOTOR3POS = MOTOR3POS + 1;
                } else if ( MOTOR3POS == 3 ) {
                    M3an = 1;
                    MOTOR3POS = MOTOR3POS + 1;
                } else {
                    M3bn = 1;
                    MOTOR3POS = 1;
                }
//*/
//*/ //Micro Stepping
                if ( MOTOR3POS == 1 ) { //Enable the motor required
                    M3ap = 1;
                    MOTOR3POS = MOTOR3POS + 1;
                } else if ( MOTOR3POS == 2 ) {
                    M3ap = 1;
                    M3bp = 1;
                    MOTOR3POS = MOTOR3POS + 1;
                } else if ( MOTOR3POS == 3 ) {
                    M3bp = 1;
                    MOTOR3POS = MOTOR3POS + 1;
                } else if ( MOTOR3POS == 4 ) {
                    M3bp = 1;
                    M3an = 1;
                    MOTOR3POS = MOTOR3POS + 1;
                } else if ( MOTOR3POS == 5 ) {
                    M3an = 1;
                    MOTOR3POS = MOTOR3POS + 1;
                } else if ( MOTOR3POS == 6 ) {
                    M3an = 1;
                    M3bn = 1;
                    MOTOR3POS = MOTOR3POS + 1;
                } else if ( MOTOR3POS == 7 ) {
                    M3bn = 1;
                    MOTOR3POS = MOTOR3POS + 1;
                } else if ( MOTOR3POS == 8 ) {
                    M3bn = 1;
                    M3ap = 1;
                    MOTOR3POS = 1;
                }
```

```
//*/

        } else if ( step == 1 && direction == 1 ) { // Turning CCW
/*/ //Single Stepping
                if ( MOTOR3POS == 1 ) { //Enable the motor required
                   M3an = 1;
                   MOTOR3POS = 4;
                } else if ( MOTOR3POS == 2 ) {
                   M3bn = 1;
                   MOTOR3POS = MOTOR3POS - 1;
                } else if ( MOTOR3POS == 3 ) {
                   M3ap = 1;
                   MOTOR3POS = MOTOR3POS - 1;
                } else {
                   M3bp = 1;
                   MOTOR3POS = MOTOR3POS - 1;
                }
//*/
//*/ //Micro Stepping
                if ( MOTOR3POS == 1 ) { //Enable the motor required
                   M3an = 1;
                   MOTOR3POS = 8;
                } else if ( MOTOR3POS == 2 ) {
                   M3an = 1;
                   M3bn = 1;
                   MOTOR3POS = MOTOR3POS - 1;
                } else if ( MOTOR3POS == 3 ) {
                   M3bn = 1;
                   MOTOR3POS = MOTOR3POS - 1;
                } else if ( MOTOR3POS == 4 ) {
                   M3bn = 1;
                   M3ap = 1;
                   MOTOR3POS = MOTOR3POS - 1;
                } else if ( MOTOR3POS == 5 ) {
                   M3ap = 1;
                   MOTOR3POS = MOTOR3POS - 1;
                } else if ( MOTOR3POS == 6 ) {
                   M3ap = 1;
                   M3bp = 1;
                   MOTOR3POS = MOTOR3POS - 1;
                } else if ( MOTOR3POS == 7 ) {
                   M3bp = 1;
                   MOTOR3POS = MOTOR3POS - 1;
                } else {
                   M3bp = 1;
                   M3an = 1;
                   MOTOR3POS = MOTOR3POS - 1;
                }
//*/
        }

}


void init( void ) {

//    WDT = OFF;    //WatchDogTimer
//    STVR = ON;    //StackOver/UnderflowReset
//    XINST = OFF;  //Extended Instruction Set
//    CP0 = OFF;    //Code Protection
 //Oscillator settings
//    FOSC = HS;    //HighSpeed oscillator
//    FOSC = HSPLL; //HS+PLL oscillator, PLL enabled and under software control
//    FOSC = EC;    //ExternalClock oscillator, CLKO function on OSC2
//    FOSC = ECPLL; //EC+PLL oscillator, PLL enabled and under software control, CLKO function on OSC2
    //Runs on internal Oscillator
//    SCS0 = 1;
//    SCS1 = 1;

//    IESO = OFF; //Two-Speed Start-up
```

```
//Ports ( 0 = output, 1 = input )
        TRISA  = 0b00000000; //Analogue input bit0
        TRISB  = 0b00000001; //LED on bit4; Button on bit0
        TRISC  = 0b00000000;
        TRISD  = 0b00000000;
        TRISE  = 0b00011111;
        TRISF  = 0b00000000;
        TRISG  = 0b00010000;

//AD-converter
        ADCON0 = 0b00000000; // Turn off AD-converter to use Port F as normal input
        ADCON1 = 0b11111111; // Turn off AD-converter to use Port F as normal input

//Comperator
        CMCON  = 0b00110110; //Enable comperator
//              .C2OUT  = 0;            .C1OUT  = 0;            .C2INV  = 1;            .C1INV  = 1;
//              .CIS    = 0;            .CM<2>  = 1;            .CM<1>  = 1;            .CM<0>  = 0;
        CVRCON = 0b10100001; //Enable voltage refference for comperator
//              .CVREN  = 1;            .CVROE  = 0;            .CVRR   = 1;            .CVRSS  = 0;
//              .CVR<3> = 0;//Sets the reference voltage to 1 ~ 0.1375V; 2 ~ 0.275V
//              .CVR<2> = 0;            .CVR<1> = 0;            .CVR<0> = 1;

}









/*/***************************Start Test Part*************************************

        for (ia = 0; 1; ia = ia + 1) {

          if (ia > blinkduration) {

                Nop();
                Nop();

        if ( PORTBbits.RB4 ) {
                        PORTAbits.RA1 = 0;      //Clear LED
                PORTBbits.RB4 = 0;      //Set LED
        } else {
                        PORTAbits.RA1 = 1;      //Set LED
                PORTBbits.RB4 = 1;      //Clear LED
                }

        if ( PORTBbits.RB0 == 0 ) {     //If Button is pressed, the green Ethernet LED is switched on

            Nop();
            Nop();

          if ( PLP2 ) { //If Step is recognized, turn:
            if ( PLP1 ) {       //cw
                        PORTAbits.RA0 = 0;   //Clear LED

                    x_motor(1,0);
```

```
                        y_motor(1,0);

                        Nop();
                        Nop();

            } else {    //ccw
                           PORTAbits.RA0 = 1;   //Set LED

                        x_motor(1,1);
                        y_motor(1,1);

                        Nop();
                        Nop();
            }

          }

                } else { // If Button is not pressed, turn off motors

            x_motor(0,0);
            y_motor(0,0);

        }

            ia = 0;
          }

        }

*/
//*****************************End Test Part*************************************
```

```
:020000040000FA
:0600000084EF04F0120081
:0A0006000100260900000A0F0000A7
:0400100006000000E6
:0C001400D9CFE6FFE1CFD9FF0E0EE126A8
:10002000600EDE6EEA0EDD6E020EDB6A030EDB6A28
:10003000040EDB6A050EDB6A060EDB6A070EDB6A5E
:10004000080EDB6A090EDB6A0A0EDB6A0B0EDB6A3E
:100050000C0EDB6A0D0EDB6AE1DB849A8592809AD6
:10006000859481968390809685 9A819483948192D9
:10007000829 4020EDB6A030EDB6A8450020B01E0FD
:10008000010E006E016A080EDB50001803E1090E34
:10009000DB5001181CE084A20CD08450010BE66EEA
:1000A000E66A010EE66EE66ABFD8E552E552E55211
:1000B000E5528450020B01E0010EE66ED950080FA4
:1000C000E96EDACFEAFFE552E750EE6EED6A845052
:1000D000080B01E0010E006E016A0A0EDB500018E9
:1000E00003E10B0EDB5001181EE084A60ED08450F5
:1000F000040B01E0010EE66EE66A010EE66EE66AAA
:1001000091D9E552E552E552E5528450080B01E0E1
:10011000010EE66ED9500A0FE96EDACFEAFFE5521A
:10012000E750EE6EED6A8650100B01E0010E006E96
:10013000016A0C0EDB50001803E10D0EDB500118B4
:100140001EE086A80ED08450100B01E0010EE66E72
:10015000E66A010EE66EE66A63DAE552E552E552BA
:10016000E5528650100B01E0010EE66ED9500C0FDF
:10017000E96EDACFEAFFE552E750EE6EED6A020E65
:10018000DBCF00F0030EDBCF01F0DECF02F0DDCFDE
:1001900003F00250005C0350015809E3020EDB6AD1
:1001A000030EDB6A81A802D0819801D0818881B0DA
:1001B0001ED0000EE66EE66AE66EE66A35D8E552B7
:1001C000E552E552E552000EE66EE66AE66EE66A34
:1001D00029D9E552E552E552E552000EE66EE66A8F
:1001E000E66EE66A1DDAE552E552E552E552010E89
:1001F000E76E020EDBCF00F0030EDBCF01F0E7501D
:10020000026000E0122020E00C0DBFF030E01C01B
:10021000DBFF33D70E0EE15C02E2E16AE552E16EEC
:10022000E552E7CFD9FF1200D9CFE6FFE1CFD9FFE2
:10023000080EE126849A8592809A8594D950FD0F04
:10024000E96EFF0EDA20EA6E010EEE1801E1ED50C4
:1002500073E1D950FB0FE96EFF0EDA20EA6EEE5023
:10026000ED106AE10F01010E0A1901E10B5108E1DD
:10027000848A010E0A250A6F000E0B210B6F5BD0DA
:10028000020E0A1901E10B5109E1848A808A010EEC
:100290000A250A6F000E0B210B6F4DD0030E0A19B1
:1002A00001E10B5108E1808A010E0A250A6F000E58
:1002B0000B210B6F40D0040E0A1901E10B5109E12B
:1002C000808A8582010E0A250A6F000E0B210B6FB2
:1002D00032D0050E0A1901E10B5108E18582010EA9
:1002E0000A250A6F000E0B210B6F25D0060E0A1986
:1002F00001E10B5109E185828584010E0A250A6F0F
:10030000000E0B210B6F17D0070E0A1901E10B51DC
:1003100008E18584010E0A250A6F000E0B210B6F80
:100320000AD0080E0A1901E10B5105E18584848A7F
:10033000010E0A6F0B6B6CD0D950FD0FE96EFF0EEA
:10034000DA20EA6E010EEE1801E1ED5061E1D950BC
:10035000FB0FE96EFF0EDA20EA6E010EEE1801E1E6
:10036000ED5056E10F01010E0A1901E10B5105E1B3
:1003700082080E0A6F0B6B4BD0020E0A1901E141
:100380000B5107E185828584010E0A5F000E0B5B2D
:100390003FD0030E0A1901E10B5106E18584010EDD
:1003A0000A5F000E0B5B34D0040E0A1901E10B51F9
:1003B00007E18584848A010E0A5F000E0B5B28D05A
:1003C000050E0A1901E10B5106E1848A010E0A5F4C
:1003D000000E0B5B1DD0060E0A1901E10B5107E15F
:1003E000848A808A010E0A5F000E0B5B11D0070E13
:1003F0000A1901E10B5106E1808A010E0A5F000E25
:100400000B5B06D0808A8582010E0A5F000E0B5BB3
:10041000080EE15C02E2E16AE552E16EE552E7CFE7
:10042000D9FF1200D9CFE6FFE1CFD9FF080EE126B0
:1004300081968390809685 9AD950FD0FE96EFF0EC4
```

```
:10044000DA20EA6E010EEE1801E1ED5073E1D950A9
:10045000FB0FE96EFF0EDA20EA6EEE50ED106AE156
:100460000F01010E0C1901E10D5108E18186010E09
:100470000C250C6F000E0D210D6F5BD0020E0C19B8
:1004800001E10D5109E181868086010E0C250C6F7A
:10049000000E0D210D6F4DD0030E0C1901E10D5111
:1004A00008E18086010E0C250C6F000E0D210D6FEA
:1004B00040D0040E0C1901E10D5109E180868380C2
:1004C000010E0C250C6F000E0D210D6F32D0050EA4
:1004D0000C1901E10D5108E18380010E0C250C6F10
:1004E000000E0D210D6F25D0060E0C1901E10D51E6
:1004F00009E18380858A010E0C250C6F000E0D2109
:100500000D6F17D0070E0C1901E10D5108E1858A16
:10051000010E0C250C6F000E0D210D6F0AD0080E78
:100520000C1901E10D5105E1858A8186010E0C6FE0
:100530000D6B6CD0D950FD0FE96EFF0EDA20EA6E1C
:10054000010EEE1801E1ED5061E1D950FB0FE96EAB
:10055000FF0EDA20EA6E010EEE1801E1ED5056E1D1
:100560000F01010E0C1901E10D5105E18380080E08
:100570000C6F0D6B4BD0020E0C1901E10D5107E110
:10058000838085 8A010E0C5F000E0D5B3FD0030E49
:100590000C1901E10D5106E1858A010E0C5F000E78
:1005A0000D5B34D0040E0C1901E10D5107E1858A71
:1005B0008186010E0C5F000E0D5B28D0050E0C1914
:1005C00001E10D5106E18186010E0C5F000E0D5B0D
:1005D0001DD0060E0C1901E10D5107E181868086C0
:1005E000010E0C5F000E0D5B11D0070E0C1901E11E
:1005F0000D5106E18086010E0C5F000E0D5B06D0EA
:10060000808683800 10E0C5F000E0D5B080EE15C9E
:1006100002E2E16AE552E16EE552E7CFD9FF12004E
:10062000D9CFE6FFE1CFD9FF080EE126819483946C
:1006300081928294D950FD0FE96EFF0EDA20EA6EA6
:10064000010EEE1801E1ED5073E1D950FB0FE96E98
:10065000FF0EDA20EA6EEE50ED106AE10F01010E96
:100660000E1901E10F5108E18184010E0E250E6F74
:10067000000E0F210F6F5BD0020E0E1901E10F511A
:1006800009E181848182010E0E250E6F000E0F217B
:100690000F6F4DD0030E0E1901E10F5108E1818259
:1006A000010E0E250E6F000E0F210F6F40D0040EAD
:1006B0000E1901E10F5109E181828384010E0E259B
:1006C0000E6F000E0F210F6F32D0050E0E1901E1D3
:1006D0000F5108E18384010E0E250E6F000E0F21CD
:1006E0000F6F25D0060E0E1901E10F5109E1838429
:1006F0008284010E0E250E6F000E0F210F6F17D092
:10070000070E0E1901E10F5108E18284010E0E253A
:100710000E6F000E0F210F6F0AD0080E0E1901E1A7
:100720000F5105E182848184010E0E6F0F6B6CD036
:10073000D950FD0FE96EFF0EDA20EA6E010EEE18B9
:1007400001E1ED5061E1D950FB0FE96EFF0EDA20B7
:10075000EA6E010EEE1801E1ED5056E10F01010EB7
:100760000E1901E10F5105E18384080E0E6F0F6B26
:100770004BD0020E0E1901E10F5107E183848284F0
:10078000010E0E5F000E0F5B3FD0030E0E1901E14C
:100790000F5106E18284010E0E5F000E0F5B34D014
:1007A000040E0E1901E10F5107E182848184010ECC
:1007B0000E5F000E0F5B28D0050E0E1901E10F51E0
:1007C00006E18184010E0E5F000E0F5B1DD0060E48
:1007D0000E1901E10F5107E181848182010E0E5F44
:1007E000000E0F5B11D0070E0E1901E10F5106E14B
:1007F0008182010E0E5F000E0F5B06D08182838422
:10080000010E0E5F000E0F5B080EE15C02E2E16A72
:10081000E552E16EE552E7CFD9FF1200926A010E70
:10082000936E946A956A1F0E966E976A100E986E74
:0E083000C26AC168360EB46EA10EB56E12001B
:02083E00060EA4
:10084000F66E000EF76E000EF86E0F010900F550FF
:10085000056F0900F550066F03E1056701D03DD033
:100860000900F550006F0900F550016F0900F550BF
:10087000026F09000900F550E96E0900F550EA6EB3
:1008800009000900900F550036F0900F550046FD5
:1008900009000900F6CF07FFF7CF08FFF8CF09FFDF
```

```
:1008A00000CFF6FF01CFF7FF02CFF8FF0F01035390
:1008B00002E1045307E00900F550EE6E0307F8E289
:1008C0000407F9D707CFF6FF08CFF7FF09CFF8FFE6
:0C08D0000F010507000E065BBFD71200E9
:0408DC00800EF36E29
:1008E00000EE00F00E0E07D8600EF36E0FEE00F073
:1008F0000F0E01D81200EA6002D0EE6AFCD7F35066
:08090000E9601200EE6AFCD769
:080908001DEE00F02DEE00F0E1
:10091000F86A059C1FEC04F092EC04F00AEC00F07D
:04092000FBD71200EF
:020924001200BF
:0609260001000100100C8
:00000001FF
```