Library & variable declarations (1)

```
//========================= Libraries that must be included ============================

#include "TimerOne.h"
/*
    This library must be download from ARDUINO IDE
                http://playground.arduino.cc/Code/Timer1
                Download -> TimerOne Google Code download
    To install, simply unzip and put the files in Arduino/hardware/libraries/Timer1/
*/
#include <EEPROM.h>
/*
    This library is available in the ARDUINO IDE. It don't require any download;
*/
//======================================================================================
```

Library & variable declarations (2)

```
// VARIABLES FOR THE ROUTINE DECODING AND COUNTING PULSES

int PIN_CH1A = 12;        //Assigns the  PIN_CH1A  label at the input IO12
int PIN_CH1B = 11;        //Assigns the  PIN_CH1B  label at the input IO11

int CH1A = 0;             //Creates and sets to 0 the variable CH1A. Used to store status of PIN_CH1A
int CH1A_old = 0;         //Creates and sets to 0 the variable CH1A_old. Used to store status of CH1A variable

int CH1B = 0;             //Creates and sets to 0 the variable CH1B. Used to store status of PIN_CH1B

int CH1B_old = 0;         //Creates and sets to 0 the variable CH1B_old. Used to store status of CH1B variable

int ENCODER1_dec = 0;     //Creates and sets to 0 the variable ENCODER1_dec.
                          //Used to store the result of the decoding of the pulses

long Counter_ENC_1 = 2000; //Creates the variable Counter_ENC_1.
                          //This variable is used to hold counting of the pulses

//=============================================================================================
```

Library & variable declarations (3)

```
//=============================================================================================
// VARIABLES OF THE ACTUATOR's PARAMETERs

long Act_MaxSpeed = 4090;
long Act_MaxSpeed_MT = 4090;
int Act_MinSpeed = 5;
long Act_DecSpeed;
int Act_AccRamp = 25;
int Act_DecRamp = 200;
int Act_CalSpeed = 512;
unsigned long Act_dTarget = 0;
unsigned long Act_Resolution = 1053;
unsigned long Act_Position;
//=============================================================================================
```

Library & variable declarations (4)

```
//=============================================================================================
// VARIABLES FOR CONTROL MOTOR OUTPUTs

int PIN_FWD_OUT = 7;          //Assigns the  PIN_FWD_OUT label at the output D7 forward relay
int PIN_BKW_OUT = 6;          //Assigns the  PIN_BKW_OUT label at the output D6 backward relay
int PIN_PWM_OUT = 9;          //Assigns the  PIN_PWM_OUT label at the output D9 used as analog
                              // output to modulate the chopper of the motor current MOSFET

// VARIABLES FOR MotorControl FUNCTION (SETS SPEED AND COMMAND DIRECTION)

boolean MC_CMD_Dir = false;       //Creates and sets to false the logic variable MC_CMD_Dir 0 = Forward , 1 = Backward
boolean MC_CMD_Alarm = false;     //Creates and sets to false the logic variable MC_CMD_Alarm 1 = Alarm , 0 = Ready
boolean MC_ST_Start = false;      //Creates and sets to false the logic variable MC_ST_START 1 = START , 0 = STOP
boolean MC_ST_PWM = false;        //Creates and sets to false the logic variable MC_ST_PWM 1 = enable PWM , 0 = disable PWM

boolean MC_Out_FWD = false;       //Creates and sets to false the logic variable MC_Out_FWD
boolean MC_Out_BKW = false;       //Creates and sets to false the logic variable MC_Out_BKW
unsigned int MC_Out_PWM = 0;      //Creates and sets to 0 the integer variable MC_Out_PWM.
unsigned int MC_Out_PWM0 = 0;
unsigned int MC_Out_PWM1 = 0;
unsigned int MC_Out_PWM2 = 0;
unsigned int MC_Out_PWM3 = 0;
unsigned int MC_Out_PWM4 = 0;
unsigned int MC_Out_PWM5 = 0;
unsigned int MC_Out_PWM6 = 0;
unsigned int MC_Out_PWM7 = 0;
unsigned int MC_Out_PWM8 = 0;
unsigned int MC_Out_PWM9 = 0;

//=============================================================================================
```

Library & variable declarations (5)

```
//=========================================================================================
// VARIABLES FOR SpeedControl Selection Mode FUNCTION (Select souces of Speed command reference)

int SC_Out_Speed = 0;            //Creates and sets to 0 the integer variable

boolean SC_Out_Start = 0;      //Creates and sets to false the logic variable
boolean SC_Out_Dir = 0;        //Creates and sets to false the logic variable

// VARIABLES FOR SpeedTrapezioidalProfile FUNCTION (Generates a speed trapezioidal profile command reference)

int STP_Gdec = 10;               // This variable acts as gain to decrease speed when reaching the target position
                                 // STP_Dec_Speed = STP_Gdec * Act_dTarget;
long STP_Acc_Speed = 0;        // This variable indicates speed value during initial ramp till max speed
long STP_Dec_Speed = 0;        // This variable indicates speed value during final ramp till min speed
int STP_Out_Speed = 0;         // This variable indicates output speed reference from trapezoidal profile algoritm

boolean STP_Out_Start = false;
boolean STP_Out_Dir = false;
boolean STP_Out_Busy = false;
boolean STP_Out_Done = false;
//=========================================================================================
```

Library & variable declarations (6)

```
//=============================================================================
// VARIABLES FOR Motion Task FUNCTION (Generates a sequence of movements and pauses)

int MT_Step = 0;              // This variable indicates Step value of the motion task cycle
//int MT_Speed = 0;           // For next implementation
int MT_Speed1 = 500;          // This variable indicates Speed value programmed for motion N°1
int MT_Speed2 = 1000;         // This variable indicates Speed value programmed for motion N°2
int MT_Speed3 = 4000;         // This variable indicates Speed value programmed for motion N°3

boolean MT_Start = false;     // This variable indicates start of the motion task
boolean MT_Dir = false;       // This variable indicates direction of the motion task movement

long MT_Position;             // This variable contains Position value in count pulse units
long MT_Position1 = 2000;     // This variable indicates Position value programmed for motion N°1
long MT_Position2 = 3500;     // This variable indicates Position value programmed for motion N°2
long MT_Position3 = 1000;     // This variable indicates Position value programmed for motion N°3

int MT_Pause1 = 2000;         // This variable indicates Pause value programmed for motion N°1 (msec)
int MT_Pause2 = 2000;         // This variable indicates Pause value programmed for motion N°2 (msec)
int MT_Pause3 = 3000;         // This variable indicates Pause value programmed for motion N°3 (msec)

unsigned long time0 = 0;      // This variable is set at the beginnings of the motion task movement
unsigned long time = 0;       // Requaires control on overun value about one time every 49
days//=============================================================================
```

Library & variable declarations (7)

```
//=============================================================================================
// VARIABLES FOR In_Position FUNCTION (Actives flag when in position)

long Target_Position = 0;
boolean Pos_Window = false;


// VARIABLES FOR EndStrokeStop FUNCTION (Limits storke between 0 and PosEndStroke or when target position is reached)


unsigned long ES_OutMaxSpeed = 1000 ;
boolean Pos_Limit = false;                  // This variable is set when positive sw limit is reached
boolean Neg_Limit = false;                  // This variable is set when negative sw limit is reached
long Brake_Space = 1;                       // This variable acts a gain reduction of the final speed ramp
                                            // Act_DecSpeed = Act_dTarget * Act_MaxSpeed / Brake_Space
int Act_WindowPos = 10;                     // This variable indicates position target range reached

boolean StopMotor = false;                  //Creates and sets to false the logic variable StopMotor.
//=============================================================================================
```

Library & variable declarations (8)

```
//=============================================================================================
// VARIABLES FOR In_Position FUNCTION (Actives flag when in position)

long Target_Position = 0;
boolean Pos_Window = false;

// VARIABLES FOR EndStrokeStop FUNCTION (Limits storke between 0 and PosEndStroke or when target position is
reached)

unsigned long ES_OutMaxSpeed = 1000 ;
boolean Pos_Limit = false;                  // This variable is set when positive sw limit is reached
boolean Neg_Limit = false;                  // This variable is set when negative sw limit is reached
long Brake_Space = 1;                       // This variable acts a gain reduction of the final speed ramp
                                            // Act_DecSpeed = Act_dTarget * Act_MaxSpeed / Brake_Space;
int Act_WindowPos = 10;                     // This variable indicates position target range reached

boolean StopMotor = false;                  //Creates and sets to false the logic variable StopMotor.

// VARIABLES FOR Calibration Sequence FUNCTION

long Pos_End_Stroke = 10000 ;               // SW positive end stroke initilized at power on
int   CS_Step= 0;                           // This variable indicates Step value of the Calibration cycle
boolean   CS_REQ = false;                   // This variable indicates Calibration in progres
//=============================================================================================
```

Library & variable declarations (9)

```
//=============================================================================
//VARIABLES FOR MEASURE MOTOR CURRENT AND CONTROL LIMIT CURRENT
// The gain of the current amplifier, is 5V = 16A = 1024 unit


volatile int IMot = 0;          //Creates and sets to 0 the variable IMot. Used to store analog to digital value of A0 input
int ImaxMotFWD = 20;            //Creates and sets to ... the variable ImaxMotFWD.
                                //Used to store the max limit current, during forward of the actuator
int ImaxMotBKW = 10;            //Creates and sets to ... the variable ImaxMotBKW.
                                //Used to store the max limit current, during backward of the actuator
boolean Imax  = false;          //Creates and sets to false the logic variable IMax.This flag represent the  overcurrent reached

//VARIABLES FOR MEASURE MOTOR SPEED

 int ActSpeed = 0;                      // This variable indicates actual speed motor revolution (RPM)
 long Counter_ENC_1_old = 0;            // This variable indicates initial space point measurement
 int CountSpeedSample = 0;              // This variable count time evolving from start measurement
 int SpeedSample = 200;                 // This variable indicates time sample measurement

//VARIABLES USED TO TEST SW ROUTINE

int PIN_TEST_1 = 8;             //Assigns the PIN_TEST_1 label at the output IO8. Used to test sw
                                //On the board, this pin must be connected to an external LED with 330 ohm in serie(CN2/16)
int PIN_TEST_2 = 10;            //Assigns the PIN_TEST_1 label at the output IO10. Used to test sw
                                 //On the board, this pin must be connected to an external LED with 330 ohm in serie(CN2/16

//=============================================================================
```

Library & variable declarations (10)

```
//=============================================================================
/*  VARIABLES FOR THE POWER FAIL OPERATION
    This routine is necessary to save and recall the real position of the actuator
    The real position variable requires N°3 Bytes (24 bit). ARDUINO has an EEPROM.write function that save one byte for each
    operation.  It require 3,3 ms for each byte.   Counter position is saved in byte N°1,2,3  */

int PF_INT = 3;                 //Assigns the  PF_INT  label at the input D3
boolean POWER_ON = false;  // Status flag to identify the power up. After power up during INIT operations this bit is set false
                                // when recognized must be set true
int CountHigh = 0;              //Creates and sets to 0 the variable CountHigh. Used to store actual position value (High byte)
int CountMiddle = 0;            //Creates and sets to 0 the variable CountMiddle. Used to store actual position value (Middle byte)
int CountLow = 0;               //Creates and sets to 0 the variable CountLow. Used to store actual position value (Low byte)

byte ByteHigh = 0;              //Creates and sets to 0 the variable ByteHigh. Used to store actual position value (High byte)to be
                                //saved in EEPROM
byte ByteMiddle = 0;            //Creates and sets to 0 the variable ByteMiddle. Used to store actual position value (Middle byte)to
                                //be saved in EEPROM
byte ByteLow = 0;               //Creates and sets to 0 the variable ByteLow. Used to store actual position value (Low byte)to be
                                //saved in EEPROM

long TempVarEEPROM = 0;   // Temporary variable used to convert long format into 3 bytes

volatile int Save_PF = 0;

/*     VARIABLES FOR SAVE SW END STROKE
    This routine is necessary to save and recall the Sw end stroke of the actuator
    The real position variable requires N°3 Bytes (24 bit). ARDUINO has an EEPROM.write function that save one byte for each
    operation.   It require 3,3 ms for each byte.  SW end stroke position is saved in byte N°4,5,6     */

boolean Save_SW_End_Stoke = false;   // Status flag to identify the sw save end stroke operation.
//=============================================================================
```

Library & variable declarations (11)

```
//=============================================================================================
//VARIABLES FOR THE INPUT COMMAND MANUAL OPERATION

int PIN_FWD_IN = A2;     //Assigns the PIN_FWD_IN label at the input A2. Active when low
                  //On the board this pin must be connected to an external push button (CN2/4)
int PIN_BKW_IN = A3;     //Assigns the PIN_BKW_IN label at the input A3. Active when low
                  //On the board this pin must be connected to an external push button (CN2/5)
int PIN_LOCAL = A4;      //Assigns the PIN_BKW_IN label at the input A4. Active when low
                  //On the board this pin must be connected to an external switch to select Cycle operation (CN2/6)
int PIN_CAL = A5;        //Assigns the PIN_CAL label at the input A5. Active when low
                  //On the board this pin must be connected to an external switch to select Calibration operation (CN2/7)

/*   VARIABLES FOR POLLING INPUT COMMAND SIGNALS */
  boolean INPUT_FWD = false;      //Creates and sets to false the flag status INPUT_FWD (forward push botton)
  boolean INPUT_BKW = false;    //Creates and sets to false the flag status INPUT_BKW (backward push botton)
  boolean INPUT_MAN= false;     //Creates and sets to false the flag status INPUT_LOC (cycle switch)
  boolean INPUT_CAL = false;    //Creates and sets to false the flag status INPUT_CAL (calibration switch)

/*   For to have a minimun debounce of the transitions of the  commands inputs are used intermediate variables, where are stored
     the commands waiting the successive sample that must to do confirm with a logic AND operation (Actual_value && Old_value)*/
  boolean TSI_OLD_FWD = false;      //Creates and sets to false the flag status TSI_OLD_FWD (used to avoid false transition)
  boolean TSI_OLD_BKW = false;      //Creates and sets to false the flag status TSI_OLD_BKW (used to avoid false transition)
  boolean TSI_OLD_MAN = false;      //Creates and sets to false the flag status TSI_OLD_LOC (used to avoid false transition)
  boolean TSI_OLD_CAL = false;      //Creates and sets to false the flag status TSI_OLD_CAL (used to avoid false transition)

  // The result of the logic operation sets the active operating state
  boolean TSI_ST_FWD = false;       //Creates and sets to false the flag status TSI_ST_FWD (used to indicate active state forward)

  boolean TSI_ST_BKW = false;       //Creates and sets to false the flag status TSI_ST_BKW (used to indicate active state backward)

  boolean TSI_ST_MAN = false;       //Creates and sets to false the flag status TSI_ST_LOC (used to indicate active state cycle)

  boolean TSI_ST_CAL = false;       //Creates and sets to false the flag status TSI_ST_CAL (used to indicate active state calibration)
  boolean TSI_START_FWD = false;
  boolean TSI_START_BKW = false;
  boolean TSI_Out_Start = false;
  boolean TSI_Out_Dir = false;
  int TSI_Out_Sel_Mode = 0;         // This variable indicates actual speed mode active
//=============================================================================================
```