```
variables
{
  //Vers. 9.4
    message DTOOL_to_USM msgTesterToEcu;
  message USM_to_DTOOL msgEcuToTester;


        ////////////////////////////////////////////////////////////////////
  ////////////////////////////////////////////////////////////////////////////
    const long VARS_SIZE = 300;              // Maximum number of variables
    const long MAX_VAR_LENGTH = 100;         // Maximum length of a
variable name
    const long MAX_VAR_SIZE = 100;           // Maximum number of
varibles read with an identifier
    const BYTE SEP = ';';                    // Separator used in
config files (0x3B)


    // EMC32
    const long NCycles= 500;
    const long NCyclesCalib=50;
    const long NSTEPS =8;
    long arrayError[VARS_SIZE][2];         //Capture flag up or down.
    long eflag[VARS_SIZE];
    long errorCounter=0;
    long startaux=0;
    long indexDelay=0;
    long indexStop=0;
    long CANSTATEflag[2];
    long NewStep[VARS_SIZE];
    long senueloA=0;
    long senueloB=0;
    long senueloC=0;
    int controlSimulador=0;
    WORD DID_LIST_ID[VARS_SIZE][100]; // List of ID readed with the
identifier


    //mstimer tEMC32;
    message m  mEMC;
    mstimer DelayCalibration;
    mstimer DelayCycling;
    mstimer DelayStop;
    mstimer tflagA, tflagB, tflagC;
    mstimer FailureCAN;

    struct ErrorImmClass
    {
    double maxVal;
    double minVal;
    double Value;
    double Frequenz;
    double ImmunityField;
    double Threshold;

```

```
    long State;
    long Ciclat;
    long Varid;
    int Error;
    char Names[MAX_VAR_LENGTH];


    };
    struct ErrorImmClass ErrorImm[NCycles][NSTEPS][VARS_SIZE];
    struct ErrorImmClass ErrorCalib[NCyclesCalib][NSTEPS][VARS_SIZE];

    // Read from configuration file
  WORD VARID[VARS_SIZE];          // Identifier of each variable (relation out/diag)
  WORD TYPES[VARS_SIZE];          // 0: DI, 1: AI, 2: DO, 3: PO, 4: AD, 5: DD, 6: ROUT
  WORD INVERT[VARS_SIZE];         // 1: Variable is inverted (only for 1 bit variables)
  char NAMES[VARS_SIZE][MAX_VAR_LENGTH];  // Name of the variable
  WORD DID[VARS_SIZE];          // DID for read/write the variable
  WORD NBYTES[VARS_SIZE];         // Number of bytes of data in CAN MSG
  WORD OFFSET[VARS_SIZE];         // Offset in CAN MSG
  WORD NBITS[VARS_SIZE];          // Number of bits of the variable in CAN MSG
    BYTE VARIANT_L0[VARS_SIZE];             // 0: Yes, 1: No, 2: Mounted

    // Internally used
  long VALUES[VARS_SIZE];         // Value of the variable
    WORD tmp[VARS_SIZE];                    // Used to detect changes in output variables
  WORD CHANGE[VARS_SIZE];         // Programmed change of an output, 1: active
  long NVARS = 0;             // Number of variables

  // Generated from the configuration
  WORD DID_LIST[VARS_SIZE];       // List of different DIDs
  BYTE DID_LIST_NUM_INPUTS[VARS_SIZE];  // Number of input vars for this DID
  BYTE DID_LIST_NUM_OUTPUTS[VARS_SIZE]; // Number of output vars for this DID
  BYTE DID_LIST_NUM_DIAGS[VARS_SIZE];   // Number of diag vars for this DID
  BYTE DID_LIST_NUM[VARS_SIZE];     // Number of vars for this DID
  WORD DID_LIST_NUMBER[VARS_SIZE][100]; // List of variables readed with the identifier
  long NDID = 0;              // Number of differente DIDs
  ////////////////////////////////////////////////////////////////////////////
  ////////////////////////////////////////////////////////////////////////////

      // All the times in milliseconds
  const int USER_SCAN_RATE=50;
    const int DELAY_SESSION=100;
    const int DELAY_SECURITY_ACCESS=500;
    const int DELAY_INPUT_READING=200;
    const int PERIOD_WATCHDOG=1000;
```

```
97    const int PERIOD_TESTER_PRESENT=2500;
98    const int PERIOD_SCHEDULER=1;
99
100   char ActiveSessionStr[4][100] = {"No Session", "Default Session", "Extended Session", "Extended Session (Secured)"};
101   BYTE ActiveSession = 0;
102
103   // Used for Tx/Rx data
104   const int BUFFER_SIZE=4098;
105 byte gRxDataBuffer[BUFFER_SIZE];
106 byte gTxDataBuffer[BUFFER_SIZE];
107   byte Aux_DataBuffer[BUFFER_SIZE];
108   char txArraySize;
109   char variant[4];
110
111   // Used for Security Access
112   BYTE ALG_eese_data[3];
113
114   mstimer UserScan;
115   mstimer StartDefaultSession;
116   mstimer StartExtendedSession;
117   mstimer Scheduler;
118   mstimer Watchdog;
119   mstimer ControlOutputs;
120   mstimer GetLINTestResults;
121
122   long sendCounter = 0;       // Max: NDID
123   long sendCounterAdd = 0;            // To periodically read other variables
124 long writeIndex = 0;          // Max: NVARS
125   BYTE controlOutputsActive = 0;      // In order to avoid timer overlap
126
127 DWORD timeStart = 0;
128 DWORD timeStop = 0;
129   BYTE schedulerStarted = 0;
130   BYTE state = 0;
131
132   // PROGRAMMED MESSAGE SENDINGS
133   BYTE hasToSendGetVariant = 0;
134   BYTE hasToSendGetSoftwareVersionS12Lear = 0;
135   BYTE hasToSendGetSoftwareVersionS12Jlr = 0;
136   BYTE hasToSendGetSoftwareVersionAtmel = 0;
137   BYTE hasToSendSetVariant = 0;
138   BYTE hasToSendReadStartLogicControl = 0;
139   BYTE hasToDisableProtections = 0;
140   BYTE hasToEnableProtections = 0;
141   BYTE hasToSendGetPowerMode = 0;
142   BYTE hasToSendChangePowerMode = 0;
143   BYTE hasToChangeWatchdogState = 0;
144   BYTE hasToSendFastGoToSleep = 0;
145   BYTE hasToSendLINTestStart = 0;
146   BYTE hasToSendLINTestGetResults = 0;
147   BYTE hasToSendWakeupTestStart = 0;
148   BYTE hasToSendWakeupTestGetResults = 0;
149   BYTE hasToSendGetSerialNumber = 0;
150   BYTE hasToSendReadColumnConfig = 0;
```

```
    BYTE hasToSendWriteColumnConfig = 0;
    BYTE hasToSendWriteColumnDefault = 0;

    BYTE StartLogicControl[6];
    BYTE PowerMode = 0;
    BYTE StopWatchdog = 0;
    BYTE ColumnState = 0;
    DWORD ColumnParamVal[16];
    BYTE ColumnParamId[16];
    BYTE ColumnCycleState;

    BYTE SvpasPwm = 0;

    // AUTO TESTING
    mstimer AT_Delay;
    const long AT_MAX_STEPS = 300;
    const long AT_MAX_LINE_CHARS = 1024;
    WORD AT_TYPES[VARS_SIZE];
    char AT_NAMES[VARS_SIZE][MAX_VAR_LENGTH];
    WORD AT_TOLERANCE[VARS_SIZE];
    WORD AT_VALUES[VARS_SIZE][AT_MAX_STEPS];
    WORD AT_MEASURED_VALUES[VARS_SIZE][AT_MAX_STEPS];
    WORD AT_VALIDATION_RESULTS[VARS_SIZE][AT_MAX_STEPS];    // 0: Not
Tested, 1: Not Applicable, 2: OK, 3: NOK

    BYTE AT_State;      // 0: Write Outputs, 1: Wait Stabilisation, 2:
Read Inputs & Diags
    BYTE AT_Mode;       // 0: Playing, 1: Recording
    BYTE AT_IncMode;    // 0: Decreasing Steps, 1: Increasing Steps
    BYTE AT_OneStep;
  long AT_SeqIndex;   // Sequential output index
  BYTE AT_SeqState;   // 0: Activation, 1: Deactivation
    long AT_OUT_INDEX[VARS_SIZE];
    long AT_NUM_OUTPUTS;

    //DTC's variables
    int DTRL_DTC=0;
    int x;

    //Variables de rutines
    mstimer Routine_T;
    byte RoutineDataBuffer[BUFFER_SIZE];
    char RoutineArraySize;

    // Char
  char buffer[64];
  char DiagInfoASCII_[3];
  int Asciifound=0;
    char inhibit;
    char Auto_Test=0;

    //BSS Test
    int sec,min,hour;
    int increase;
    float   Voltage;
    char    aux[6];
    mstimer TimerCount;
```

```
  Timer   VoltageScheduler;
dword glbHandle = 0;

  //Syst.Ident.
  int SwNum1,SwNum2;
  int EdNum1,EdNum2;
  char fileUsed[32];
  byte CheckingSwNum, CheckingEditNum;

  //AC Valve test
  Timer Test_Time;
msTimer timeT;
  msTimer ReadT;
  msTimer StopRT;
  int AC_Valve_DC;
int TestingTime;
  int t=0,s=0,m=0,ms=0;
  int up,ACV_T=0;
  int Offset;

  int LINErrors_Inicial;

  //Control_Lighting_Test
  mstimer t_ControlLighting;
  int     CL_x=0;
  int     SPI_Check=0;
  mstimer t_ReadFeedbacks;

  //Iocab DO_0 pulse generator
  mstimer t_PulseGenerator;
  int     T_On,T_Off,iter;
  int     ciclesCount;

  byte    HW_Testing=0;
}

on start
{
  OSEKTL_SetTxId(msgTesterToEcu.id);
  OSEKTL_SetRxId(msgEcuToTester.id);

  LoadConfigFromFile();
  LoadSystemIdent();
ProcessDIDs();

  //PrintConfig();
//PrintDIDs();
//LoadCyclingFromFile();
//PrintCycling();
  //
LoadDTCsFromFi


  CopyNameToDataBase();
  InitializeValues();
  putValue(VAR_DiagSession, ActiveSessionStr[ActiveSession]);

```

```
  settimer(UserScan, USER_SCAN_RATE);
    putValue(AT_Continous,0);
    putValue(EBCM_WakeUpSleepCommand_,3);
    setTimer(DelayCalibration,1000);

}

LoadConfigFromFile ()
{
  dword readHandle = 0;
  char readBuffer[255];
  char dataBuffer[255];
  long i = 0;
  long j = 0;
  long k = 0;
  long index = 0;

  write("Opening Config File");
  readHandle = openFileRead ("X10_Signals.csv", 0);
  //if(readHandle != 0) fileRewind(readHandle);
  if (readHandle != 0 && fileGetString(readBuffer, elcount(readBuffer),
readHandle) != 0)
  {
    while(fileGetString(readBuffer, elcount(readBuffer), readHandle) !=
0 )
    {
      // Read VARID
      j = 0; i = 0;
      while (readBuffer[i] != SEP && readBuffer[i] != 0 && i<=elcount(
readBuffer))
      {
        dataBuffer[j] = readBuffer[i];
        j++; i++;
      };
      dataBuffer[j] = 0;
      VARID[index] = atol(dataBuffer);

            //if(readBuffer[i] != SEP) write("Reading Error");

      // Read TYPE
      j = 0; i++;
      while (readBuffer[i] != SEP && readBuffer[i] != 0 && i<=elcount(
readBuffer))
      {
        dataBuffer[j] = readBuffer[i];
        j++; i++;
      };
      dataBuffer[j] = 0;
      if(strncmp(dataBuffer, "DI", strlen(dataBuffer))==0)
        TYPES[index] = 0;
      else if(strncmp(dataBuffer, "AI", strlen(dataBuffer))==0)
        TYPES[index] = 1;
      else if(strncmp(dataBuffer, "DO", strlen(dataBuffer))==0)
        TYPES[index] = 2;
      else if(strncmp(dataBuffer, "PO", strlen(dataBuffer))==0)
        TYPES[index] = 3;
      else if(strncmp(dataBuffer, "AD", strlen(dataBuffer))==0)

```

```
        TYPES[index] = 4;
313     else if(strncmp(dataBuffer, "DD", strlen(dataBuffer))==0)
314       TYPES[index] = 5;
315           else if(strncmp(dataBuffer, "ROUT", strlen(dataBuffer))==0)
316       TYPES[index] = 6;
317           //if(readBuffer[i] != SEP) write("Reading Error");
318
319     // Read NAME
320     j = 0; i++;
321     while (readBuffer[i] != SEP && readBuffer[i] != 0 && i<=elcount(
322 readBuffer))
      {
323       dataBuffer[j] = readBuffer[i];
324       j++; i++;
325     };
326     dataBuffer[j] = 0;
327     for(k=0; k<=j & k<MAX_VAR_LENGTH; k++)
328       NAMES[index][k]=dataBuffer[k];
329           //if(readBuffer[i] != SEP) write("Reading Error");
330
331     // Read INVERT
332     j = 0; i++;
333     while (readBuffer[i] != SEP && readBuffer[i] != 0 && i<=elcount(
334 readBuffer))
      {
335       dataBuffer[j] = readBuffer[i];
336       j++; i++;
337     };
338     dataBuffer[j] = 0;
339     INVERT[index] = atol(dataBuffer);
340           //if(readBuffer[i] != SEP) write("Reading Error");
341
342
343           //write("=%s,varid=%d, index =%d",NAMES[index],VARID[index],
344 index);

345     // Read DID
346     j = 2; i++;
347     while (readBuffer[i] != SEP && readBuffer[i] != 0 && i<=elcount(
348 readBuffer))
      {
349       dataBuffer[j] = readBuffer[i];
350       j++; i++;
351     };
352     dataBuffer[j] = 0;
353     dataBuffer[0] = '0';
354     dataBuffer[1] = 'x';
355     DID[index] = atol(dataBuffer);
356           //if(readBuffer[i] != SEP) write("Reading Error");
357
358     // Read NBYTES
359     j = 0; i++;
360     while (readBuffer[i] != SEP && readBuffer[i] != 0 && i<=elcount(
361 readBuffer))
      {
362       dataBuffer[j] = readBuffer[i];
363       j++; i++;
364
```

```
      };
      dataBuffer[j] = 0;
      NBYTES[index] = atol(dataBuffer);
            //if(readBuffer[i] != SEP) write("Reading Error");

      // Read OFFSET
      j = 0; i++;
      while (readBuffer[i] != SEP && readBuffer[i] != 0 && i<=elcount(readBuffer))
      {
        dataBuffer[j] = readBuffer[i];
        j++; i++;
      };
      dataBuffer[j] = 0;
      OFFSET[index] = atol(dataBuffer);
            //if(readBuffer[i] != SEP) write("Reading Error");

      // Read NBITS
      j = 0; i++;
      while (readBuffer[i] != SEP && readBuffer[i] != 0 && i<=elcount(readBuffer))
      {
        dataBuffer[j] = readBuffer[i];
        j++; i++;
      };
      dataBuffer[j] = 0;
      NBITS[index] = atol(dataBuffer);
            //if(readBuffer[i] != SEP) write("Reading Error");

            // Read L0 Population
      j = 0; i++;
      while (readBuffer[i] != SEP && readBuffer[i] != 0 && i<=elcount(readBuffer))
      {
        dataBuffer[j] = readBuffer[i];
        j++; i++;
      };
      dataBuffer[j] = 0;
      if(strncmp(dataBuffer, "Y", strlen(dataBuffer))==0)
        VARIANT_L0[index] = 0;
      else if(strncmp(dataBuffer, "N", strlen(dataBuffer))==0)
        VARIANT_L0[index] = 1;
            //if(readBuffer[i] != SEP) write("Reading Error");

      index++;
    };
    NVARS = index;
    write("Readed %d variables", NVARS);
    write("Closing Config File");
    fileClose(readHandle);
  }
  else
  {
    write("Config file cannot be opened");
  }

}
```

```
ProcessDIDs ()
{
  long i, j;
  int found;

  NDID = 0;
  for(i=0; i<NVARS; i++)
  {
    found = 0;
    for(j=0; j<NDID; j++)
    {
      if(DID[i] == DID_LIST[j])
      {
        DID_LIST_NUMBER[j][DID_LIST_NUM[j]] = i;//VARID[i];
                DID_LIST_ID[j][DID_LIST_NUM[j]] = VARID[i];//VARID[i];

                //write("\nDID_LIST_NUMBER[j][DID_LIST_NUM[j]]=%d    ,DID_LIST_NUMBER= %d,    ID=%d VAR=%d",DID_LIST_NUMBER[j][DID_LIST_NUM[j]],DID_LIST_NUM[j],j,i);
                //write("DID_LIST_ID[j][DID_LIST_NUM[j]]=%d    ,DID_LIST_NUMBER= %d,    j=%d i=%d",DID_LIST_ID[j][DID_LIST_NUM[j]],DID_LIST_NUM[j],j,VARID[i]);
                //write("%s",NAMES[i]);


                DID_LIST_NUM[j]++;
        if(TYPES[i] == 0) DID_LIST_NUM_INPUTS[j]++;
        else if(TYPES[i] == 1) DID_LIST_NUM_INPUTS[j]++;
        else if(TYPES[i] == 2) DID_LIST_NUM_OUTPUTS[j]++;
                else if(TYPES[i] == 3) DID_LIST_NUM_OUTPUTS[j]++;
                else if(TYPES[i] == 4) DID_LIST_NUM_DIAGS[j]++;
                else if(TYPES[i] == 5) DID_LIST_NUM_DIAGS[j]++;

        found = 1;
        break;
      }
    }
    if(found == 0)
    {
      DID_LIST_NUMBER[NDID][DID_LIST_NUM[NDID]] = i;//VARID[i];
            DID_LIST_ID[NDID][DID_LIST_NUM[NDID]] = VARID[i];//VARID[i];

            //write("DID_LIST_NUMBER=%d    ,DID_LIST_NUMBER= %d,    NDID=%d VAR=%d",DID_LIST_NUMBER[NDID][DID_LIST_NUM[NDID]],DID_LIST_NUM[NDID],NDID,i);
      //write("DID_LIST_NUMBER[NDID][DID_LIST_NUM[NDID]]=%s", NAMES[DID_LIST_NUMBER[NDID][DID_LIST_NUM[NDID]]]);

            DID_LIST_NUM[NDID]++;
      if(TYPES[i] == 0) DID_LIST_NUM_INPUTS[j]++;
      else if(TYPES[i] == 1) DID_LIST_NUM_INPUTS[j]++;
      else if(TYPES[i] == 2) DID_LIST_NUM_OUTPUTS[j]++;
            else if(TYPES[i] == 3) DID_LIST_NUM_OUTPUTS[j]++;
            else if(TYPES[i] == 4) DID_LIST_NUM_DIAGS[j]++;
            else if(TYPES[i] == 5) DID_LIST_NUM_DIAGS[j]++;


```

```
      DID_LIST[NDID] = DID[i];
      NDID++;
    }
  }
  write("Number of DIDs: %d", NDID);
}

PrintConfig ()
{
  long i;
  for(i=0; i<NVARS; ++i)
  {
    write("index: %d, i VARID: %d, TYPE: %d, INVERT: %d, NAME: %s, DID:
0x%x, NBYTES: %d, OFFSET: %d, NBITS: %d",i, VARID[i], TYPES[i], INVERT[i
], NAMES[i], DID[i], NBYTES[i], OFFSET[i], NBITS[i]);
  }
}

OSEKTL_DataCon (long txLength)
{
}

OSEKTL_DataInd (long rxLength)
{
    int count;

    OSEKTL_GetRxData(gRxDataBuffer,rxLength);

  count=rxLength;

  if((gRxDataBuffer[0]!=0x7F)&&(HW_Testing==1))
  {
        // EXTENDED SESSION ESTABLISHED
        if((gRxDataBuffer[0]==0x50)&&(gRxDataBuffer[1]==0x86))
        {
            ActiveSession = 2;
            putValue(VAR_DiagSession, ActiveSessionStr[ActiveSession]);
            write("Extended Session Established");

            // Start Input Reading
            putvalue(VAR_RefreshInputs, 1);
            putvalue(VAR_RefreshOutputs, 0);
            putvalue(VAR_RefreshDiags, 1);
            sendCounter = 0;
            sendCounterAdd = 0;
            writeIndex = NVARS;
            schedulerStarted = 1;
            hasToSendGetSerialNumber = 1;
            setTimer(Scheduler,DELAY_INPUT_READING);
          settimer(Watchdog, PERIOD_WATCHDOG);
            write("Starting Scheduler");
            write("Starting Watchdog");
        }
        // WRITE RESPONSE
        if((gRxDataBuffer[0]==0x7B)&&(gRxDataBuffer[1]==0x06))
        {
            if(inhibit==1)
```

```
            {
                putValue(Status_Inhibit_DTCs,1);
            }
            else if(inhibit==0)
            {
                putValue(Status_Enabled_DTCs,1);
            }
            putValue(VAR_RefreshInputs,1);
            putValue(VAR_RefreshDiags,1);
        }
        // RESET RECOVERY
        if((gRxDataBuffer[0]==0x51)&&(gRxDataBuffer[1]==0x01))
        {
            putValue(VAR_RefreshInputs,1);
            putValue(VAR_RefreshDiags,1);
        }
        // TESTER PRESENT
        if(gRxDataBuffer[0]==0x7E)
        {
            UpdateTimer();
        }
        // INPUTS READING
        if(gRxDataBuffer[0]==0x61)
        {
            if(gRxDataBuffer[1]==0x06)
            {
                for(x=0;x<21;x++)
                {
                    Aux_DataBuffer[x]=gRxDataBuffer[x+2];
                }
                WriteCfgParam();
            }
            else if(gRxDataBuffer[1]==0x91)
            {
                putValue(Param_A,  gRxDataBuffer[2]);
                putValue(Param_B1, gRxDataBuffer[3]);
                putValue(Param_B2, gRxDataBuffer[4]);
                putValue(Param_C,  gRxDataBuffer[5]);
            }
            if(gRxDataBuffer[1]==0x98)
            {
                putValue(BattDiscon_CNT,((gRxDataBuffer[2]*256)
+gRxDataBuffer[3]));

                putValue(VAR_RefreshInputs,1);
                putValue(VAR_RefreshDiags,1);
            }
            else if(gRxDataBuffer[1]==0x99)
            {
                putValue(B_First_GND_1,(gRxDataBuffer[2]>>7)&0x01);
                putValue(B_First_GND_2,(gRxDataBuffer[2]>>6)&0x01);
            }
            else if(gRxDataBuffer[1]==0x80) //System Ident.
            {
                interpreteAscii(gRxDataBuffer[2]);
            putValue(PartNumberLow_1_R,DiagInfoASCII_);
            interpreteAscii(gRxDataBuffer[3]);

```

```
            putValue(PartNumberLow_2_R,DiagInfoASCII_);
            interpreteAscii(gRxDataBuffer[4]);
            putValue(PartNumberLow_3_R,DiagInfoASCII_);
            interpreteAscii(gRxDataBuffer[5]);
            putValue(PartNumberLow_4_R,DiagInfoASCII_);
            interpreteAscii(gRxDataBuffer[6]);
            putValue(PartNumberLow_5_R,DiagInfoASCII_);

            putValue(CodeIdDiag_R,gRxDataBuffer[7]);

            interpreteAscii(gRxDataBuffer[8]);
            putValue(CodeITG_1_R,DiagInfoASCII_);
            interpreteAscii(gRxDataBuffer[9]);
            putValue(CodeITG_2_R,DiagInfoASCII_);
            interpreteAscii(gRxDataBuffer[10]);
            putValue(CodeITG_3_R,DiagInfoASCII_);

            interpreteAscii(gRxDataBuffer[11]);
            putValue(HardwareNumber_1_R,DiagInfoASCII_);
            interpreteAscii(gRxDataBuffer[12]);
            putValue(HardwareNumber_2_R,DiagInfoASCII_);
            interpreteAscii(gRxDataBuffer[13]);
            putValue(HardwareNumber_3_R,DiagInfoASCII_);
            interpreteAscii(gRxDataBuffer[14]);
            putValue(HardwareNumber_4_R,DiagInfoASCII_);
            interpreteAscii(gRxDataBuffer[15]);
            putValue(HardwareNumber_5_R,DiagInfoASCII_);

            putValue(ID_Sw1_R,gRxDataBuffer[16]);
            putValue(ID_Sw2_R,gRxDataBuffer[17]);

            putValue(ID_Release1_R,gRxDataBuffer[18]);
            putValue(ID_Release2_R,gRxDataBuffer[19]);

            putValue(ID_Calib1_R,gRxDataBuffer[20]);
            putValue(ID_Calib2_R,gRxDataBuffer[21]);

            putValue(PartNumberBasic_R,gRxDataBuffer[22]);

            putValue(HwNumberBasicPart_R,gRxDataBuffer[23]);

            putValue(ApprovalNumberBasicPart_R,gRxDataBuffer[24]);

            putValue(idManufacture_R,gRxDataBuffer[25]);

                if(Auto_Test==1){Auto_Test=0;Check_Traceability();}

                putValue(VAR_RefreshInputs,1);
                putValue(VAR_RefreshDiags,1);
            }
            else{OnReadResponse();}
        }
    // OUTPUTS WRITING ACK
        if(gRxDataBuffer[0]==0x70)
        {
            OnIOCtlResponse();
        }

```

```
        /*********************/
637     if((gRxDataBuffer[0]==0x7B)&&(gRxDataBuffer[1]==0x98))
638     {
639         putValue(BattDiscon_CNT,0);
640     }
641
642     if((gRxDataBuffer[0]==0x7B)&&(gRxDataBuffer[1]==0x91))
643     {
644         putValue(WriteResult,1);
645     }
646     /************************/
647     //SPI Test
648     if((gRxDataBuffer[0]==0x7B)&&(gRxDataBuffer[1]==0x9F)&&(
649 SPI_Check==1))
        {
650         //putValue(VAR_RefreshDiags,1);
651         CL_x=5;
652         settimer(t_ControlLighting,2900);
653     }
654     /************************/
655     //DTC's Routine
656     if (gRxDataBuffer[0]==0x54)
657   {
658     putValue(DTC_Cleared,1);
659     putValue(DTRL_Dtc_Status,0);
660     ClearDTCs ();
661         putValue(VAR_RefreshInputs,1);
662         putValue(VAR_RefreshDiags,1);
663   }
664
665   if (gRxDataBuffer[0]==0x59)
666   {
667     putValue(DTC_Cleared,0);
668     if (gRxDataBuffer[1]==0x02)
669     {
670       if(count>4)
671       {
672         if(DTRL_DTC==0)
673         {
674           ReportDTC(gRxDataBuffer,count);
675         }
676       }
677     }
678       putValue(VAR_RefreshInputs,1);
679         putValue(VAR_RefreshDiags,1);
680     }
681   }
682 else
683 {
684   // ERROR
685   if(gRxDataBuffer[1]==0x21)
686   {
687     //PrintErrorCode(gRxDataBuffer[2]);
688     UpdateTimer();
689   }
690   else if(gRxDataBuffer[1]==0x30)
691   {
692
```

```
      //PrintErrorCode(gRxDataBuffer[2]);
693     OnIOCtlResponse();
694   }
695     else if((gRxDataBuffer[1]==0x3B)&&(gRxDataBuffer[2]==0x12))
696   {
697     //SPI test: 3s counter begins
698     CL_x=4;
699           settimer(t_ControlLighting,30);
700   }
701   else
702   {
703     //PrintErrorCode(gRxDataBuffer[2]);
704     UpdateTimer();
705   }
706 }
707 }
708
709 OSEKTL_ErrorInd (int error)
710 {
711 }
712
713 OSEKTL_FirstFrameIndication (long source,long target,long length)
714 {
715 }
716
717 PrintDIDs ()
718 {
719   long i,j;
720   char list[500];
721   char number[10];
722
723   for(j=0; j<NDID; j++)
724   {
725     list[0] = 0;
726     for(i=0; i<DID_LIST_NUM[j]; ++i)
727     {
728       ltoa(DID_LIST_NUMBER[j][i], number, 10);
729       strncat(list, number, 500);
730       strncat(list, " ", 500);
731     }
732     write("DID: %x, I:%d, O:%d, D:%d, T:%d, %s", DID_LIST[j],
733 DID_LIST_NUM_INPUTS[j], DID_LIST_NUM_OUTPUTS[j], DID_LIST_NUM_DIAGS[j],
    DID_LIST_NUM[j], list);

734   }
735 }
736
737 on timer UserScan
738 {
739     long var;
740
741     CopyValueFromDataBase();
742
743   for(var=0; var<NVARS; ++var)
744   {
745     if((TYPES[var] == 2 || TYPES[var] == 3) && tmp[var] != VALUES[var])
746     {
747
```

```
            write("Change requested in %s, old: %d, new: %d", NAMES[var]
748 , VALUES[var], tmp[var]);
        VALUES[var] = tmp[var];
749     CHANGE[var] = 1;
750     writeIndex = var;
751   }
752 }
753
754 if(getvalue(VAR_RefreshInputs) == 0 && getvalue(VAR_RefreshOutputs)
755 == 0 && getvalue(VAR_RefreshDiags) == 0)
    UpdateTimer();
756
757
758   settimer(UserScan, USER_SCAN_RATE);
759 }
760
761 on envVar VAR_Start
762 {
763 if(getValue(this))
764 {
765   putValue(this,0);
766       HW_Testing=1;
767       setTimer(StartExtendedSession,DELAY_SESSION);
768 }
769 }
770
771 on envVar VAR_Stop
772 {
773 if(getValue(this))
774 {
775
776       if(getValue(CALIBRATION_MODE))
777       {
778       putValue(Ciclat, 0);
779       putValue(STATE, 0);
780       putValue(AT_Step,0);
781       }
782
783       putValue(this,0);
784       HW_Testing=0;
785       write("Stopping...");
786       putvalue(VAR_RefreshInputs, 0);
787       putvalue(VAR_RefreshOutputs, 0);
788       putvalue(VAR_RefreshDiags, 0);
789       schedulerStarted = 0;
790
791       canceltimer(Watchdog);
792   canceltimer(Scheduler);
793 }
794 }
795
796 on timer StartExtendedSession
797 {
798   long i;
799 for (i=0; i<BUFFER_SIZE; i++) gTxDataBuffer[i]=0;
800
801 gTxDataBuffer[0]=0x10;
802
```

```
  gTxDataBuffer[1]=0x86;
803   txArraySize=2;
804   OSEKTL_DataReq(gTxDataBuffer, txArraySize);
805 }
806
807 UpdateTimer ()
808 {
809     if(schedulerStarted == 1)
810     {
811         /*if(hasToSendGetVariant == 1) settimer(SendMessages,
812 PERIOD_SCHEDULER);
        else if(hasToSendSetVariant == 1) settimer(SendMessages,
813 PERIOD_SCHEDULER);
        else if(hasToSendGetSoftwareVersionS12Lear == 1) settimer(
814 SendMessages, PERIOD_SCHEDULER);
        else if(hasToSendGetSoftwareVersionS12Jlr == 1) settimer(
815 SendMessages, PERIOD_SCHEDULER);
        else if(hasToSendGetSoftwareVersionAtmel == 1) settimer(
816 SendMessages, PERIOD_SCHEDULER);
        else if(hasToSendReadStartLogicControl == 1) settimer(
817 SendMessages, PERIOD_SCHEDULER);
        else if(hasToDisableProtections == 1) settimer(SendMessages,
818 PERIOD_SCHEDULER);
        else if(hasToEnableProtections == 1) settimer(SendMessages,
819 PERIOD_SCHEDULER);
        else if(hasToSendGetPowerMode == 1) settimer(SendMessages,
820 PERIOD_SCHEDULER);
        else if(hasToSendChangePowerMode == 1) settimer(SendMessages,
821 PERIOD_SCHEDULER);
        else if(hasToChangeWatchdogState == 1) settimer(SendMessages,
822 PERIOD_SCHEDULER);
        else if(hasToSendFastGoToSleep == 1) settimer(SendMessages,
823 PERIOD_SCHEDULER);
        else if(hasToSendLINTestStart == 1) settimer(SendMessages,
824 PERIOD_SCHEDULER);
        else if(hasToSendLINTestGetResults == 1) settimer(SendMessages,
825 PERIOD_SCHEDULER);
        else if(hasToSendWakeupTestStart == 1) settimer(SendMessages,
826 PERIOD_SCHEDULER);
        else if(hasToSendWakeupTestGetResults == 1) settimer(
827 SendMessages, PERIOD_SCHEDULER);
        else if(hasToSendGetSerialNumber == 1) settimer(SendMessages,
828 PERIOD_SCHEDULER);
        else if(hasToSendReadColumnConfig == 1) settimer(SendMessages,
829 PERIOD_SCHEDULER);
        else if(hasToSendWriteColumnConfig == 1) settimer(SendMessages,
830 PERIOD_SCHEDULER);
        else if(hasToSendWriteColumnDefault == 1) settimer(SendMessages
831 , PERIOD_SCHEDULER);*/

832       if(writeIndex == NVARS) // No signal to write
833         {
834           if(getvalue(VAR_RefreshInputs) != 0 || getvalue(
835 VAR_RefreshOutputs) != 0 || getvalue(VAR_RefreshDiags) != 0)
            {
836           settimer(Scheduler, PERIOD_SCHEDULER);
837           }
838
```

```
        }
839
840       else if(controlOutputsActive == 0)
841       {
842         controlOutputsActive = 1;
843         settimer(ControlOutputs, PERIOD_SCHEDULER);
844       }
845     }
846 }
847
848 OnReadResponse ()
849 {
850   long identifier;
851   long index,i,var;
852   BYTE found;
853
854     identifier=gRxDataBuffer[1];
855
856     found = 0;
857   for(index=0; index<NDID; index++)
858   {
859     if(DID_LIST[index]==identifier)
860     {
861       found = 1;
862       break;
863     }
864   }
865
866   if(found)
867   {
868     for(i=0; i<DID_LIST_NUM[index]; i++)
869     {
870             var = (long)(DID_LIST_NUMBER[index][i]);
871
872             if((TYPES[var] != 2) && (TYPES[var] != 3))
873             {
874                 VALUES[var] = MsgGetValue(OFFSET[var], NBITS[var])
875 ;
876
877                 if(INVERT[var]==1)
878           {
879             if(VALUES[var]==0) VALUES[var]=1;
880             else VALUES[var]=0;
881           }
882             }
883     }
884   }
885   else
886     OnNotFoundDID(identifier);
887
888   if(ACV_T==0){UpdateTimer();}
889     else if(ACV_T==1){ReportAC(m,s,ms,AC_Valve_DC,VALUES[76]);}
890 }
891
892 OnIOCtlResponse ()
893 {
894   long i;
```

```
  controlOutputsActive = 0;

  for(i=0; i<NVARS; ++i)
    //for(i=NVARS; i>0; --i)
  {
    if(CHANGE[i]==1)
    {
            CHANGE[i]=0;
      writeIndex = i;
      break;
    }
  }

    //for(i=writeIndex; i<NVARS; ++i)
    /*for(i=writeIndex; i>0; --i)
  {
    if(CHANGE[i]==1)
    {
      writeIndex = i;
      break;
    }
  }*/

    if(i==NVARS) writeIndex=NVARS;
    //if(i==0) writeIndex=NVARS;

  UpdateTimer();
}

PrintErrorCode (BYTE errorID)
{
  switch(errorID)
  {
    case 0x11:
    {
      write("Service Not Supported");
      break;
    }
    case 0x12:
    {
      write("Subfunction Not Supported / Invalid Format");
      break;
    }
    case 0x13:
    {
      write("Incorrect Message Length or Invalid Format");
      break;
    }
    case 0x21:
        case 0x78:
    {
      write("Busy Repeat Request");
      break;
    }
    case 0x22:
    {

```

```
      write("Conditions Not Correct");
      break;
    }
    case 0x23:
    {
      write("Routine Not Complete");
      break;
    }
    case 0x31:
    {
      write("Request Out of Range");
      break;
    }
    case 0x33:
    {
      write("Security Access Denied");
      break;
    }
    case 0x35:
    {
      write("Invalid Key");
      break;
    }
    case 0x72:
    {
      write("General Programming Failure");
      break;
    }
    case 0x7E:
    {
      write("Subfunction Not Supported in Active Session");
      break;
    }
    case 0x7F:
    {
      write("Service Not Supported in Active Session");
      break;
    }
    default:
    {
      write("Unknown Error Code (%x)", errorID);
      break;
    }
  }
}

on timer StartDefaultSession
{
    long i;
  for (i=0; i<BUFFER_SIZE; i++) gTxDataBuffer[i]=0;

  gTxDataBuffer[0]=0x10;
  gTxDataBuffer[1]=0x86;
  txArraySize=2;
  OSEKTL_DataReq(gTxDataBuffer, txArraySize);
}
```

```
on timer Watchdog
{
  DWORD timeWD;

  timeWD = timeNow();
  if( (timeWD-timeStart)/100 > PERIOD_TESTER_PRESENT )
  {
    putvalue(VAR_RefreshRate, (timeWD-timeStart)/100);  // UPDATE REFRESH RATE COUNTER
    timeStart = timeWD;
    sendCounter = 0;
        sendCounterAdd = 0;
    controlOutputsActive = 0;
    SendTesterPresent();
  }

  settimer(Watchdog, PERIOD_WATCHDOG);
}

on timer Scheduler
{
  switch(state)
  {
    case 0:
    {
      // READ INPUT, OUTPUT, DIAGS
      SendReadDataById(DID_LIST[sendCounter]);
      IncSendCounter();
      if(sendCounter==0) state=1;
      break;
    }
        case 1:
    {
      // SEND TESTER PRESENT
      SendTesterPresent();

      // UPDATE REFRESH RATE COUNTER
      timeStop = timeNow();
      if(timeStart != 0) putvalue(VAR_RefreshRate, (timeStop-timeStart)/100);
      timeStart = timeNow();

            // UPDATE PANELS
            CopyInputsAndDiagsToDataBase();

            // UPDATE BATTERY VOLTAGE
            //putValue(VAR_BatteryVoltage, ((double)VALUES[249])/100);

      state=0;
      break;
    }
        case 2:
        {
            // AUTO-TEST MODE
            AT_Scheduler();
            break;
        }

```

```
  }
}

SendTesterPresent ()
{
  gTxDataBuffer[0]=0x3E;
  gTxDataBuffer[1]=0x01;
  txArraySize=2;
  OSEKTL_DataReq(gTxDataBuffer, txArraySize);
}

SendReadDataById(int value)
{
    long i;
    for (i=0; i<BUFFER_SIZE; i++) gTxDataBuffer[i]=0;

    gTxDataBuffer[0]=0x21;
  gTxDataBuffer[1]=value;
  txArraySize=2;

  OSEKTL_DataReq(gTxDataBuffer, txArraySize);
}

IncSendCounter ()
{
  sendCounter++;

    if(sendCounter==NDID)
    sendCounter=0;
    else
  {
    // Skip Inputs if necessary
    if(getvalue(VAR_RefreshInputs) == 0)
    {
      while(DID_LIST_NUM_INPUTS[sendCounter] != 0 &&
DID_LIST_NUM_OUTPUTS[sendCounter] == 0 && DID_LIST_NUM_DIAGS[sendCounter
] == 0)
      {
        sendCounter++;
        if(sendCounter==NDID) sendCounter=0;
      }
    }

    // Skip Outputs if necessary
    if(getvalue(VAR_RefreshOutputs) == 0)
    {
      while(DID_LIST_NUM_INPUTS[sendCounter] == 0 &&
DID_LIST_NUM_OUTPUTS[sendCounter] != 0 && DID_LIST_NUM_DIAGS[sendCounter
] == 0)
      {
        sendCounter++;
        if(sendCounter==NDID) sendCounter=0;
      }
    }

    // Skip Diags if necessary
    if(getvalue(VAR_RefreshDiags) == 0)

```

```
    {
      while(DID_LIST_NUM_INPUTS[sendCounter] == 0 &&
DID_LIST_NUM_OUTPUTS[sendCounter] == 0 && DID_LIST_NUM_DIAGS[sendCounter
] != 0)
      {
        sendCounter++;
        if(sendCounter==NDID) sendCounter=0;
      }
    }
  }
}

on timer ControlOutputs
{
  SendIOControlById(DID[writeIndex]);
}

void SendIOControlById (int identifier)
{
    long i;

    for (i=0; i<BUFFER_SIZE; i++) gTxDataBuffer[i]=0;

    gTxDataBuffer[0]=0x30;
  gTxDataBuffer[1]=identifier;
    gTxDataBuffer[3]=0xFF;

    txArraySize = 2 + MsgIOControl(identifier);


    OSEKTL_DataReq(gTxDataBuffer, txArraySize);
}

BYTE MsgIOControl (WORD identifier)
//MsgIOControl (WORD identifier)
{

    long index, i,var;
  WORD value;

  for(index=0; index<NDID; ++index)
    if(DID_LIST[index]==identifier) break;

  for(i=0; i<DID_LIST_NUM[index]; ++i)
  {
        var = DID_LIST_NUMBER[index][i];
        ErrorImm[0][0][var].Varid= DID_LIST_ID[index][i];

        value = VALUES[var];
        //write("%s, varid=%d, DID=%x",AT_NAMES[var], ErrorImm[0][0][var
].Varid, DID[var]);

    if(INVERT[var]==1)
    {
      if(value==0) value=1;
      else value=0;
    }
```

```
    if((TYPES[var]==2)||(TYPES[var]==3))
    {
      if((var==138)&&(value==1)&&(getValue(AT_Play)==1))  //LimpHome aprofitant el EXTERNAL_WATCHDOG
      {
        putValue(Env_SBCForceSafeActiveLow,1);
      }
      MsgSetValue(OFFSET[var], NBITS[var], value);
      //return  NBYTES[var];

    }
    CHANGE[var]=0;
  }




    ////write("=%s,AT_VARID=%d",AT_NAMES[i],AT_VARID[i]);


    //write("%x %x",NBYTES[var],var);
    return  NBYTES[var];
}

WORD MsgSetMask (WORD identifier)
{
  long index, var;
  WORD i, dataLength, addedBytes;
  WORD maskOn;
//write("Mask %x", identifier);
  maskOn = 1;
  addedBytes = 0;

  if(maskOn)
  {
    for(index=0; index<NDID; ++index)
      if(DID_LIST[index]==identifier) break;

    var = DID_LIST_NUMBER[index][0];
    dataLength = NBYTES[var];

//write("DataLength %d", dataLength);
    if(NBYTES[var]*8 != NBITS[var])
    {
      for(i=0; i<dataLength; ++i)
        gTxDataBuffer[4+dataLength+i] = 0xFF;
      addedBytes = dataLength;
    }
  }
  return addedBytes;
}

MsgSetValue (BYTE offset, BYTE size, int value)
{
    if(size<8)
    {
```

```
        if(value>0)
        {
            gTxDataBuffer[2] = 0xFB;
        }
        else if(value==0)
        {
            gTxDataBuffer[2] = 0xFC;
        }
    }
    else if (size==8)
    {
        if(value>0)
        {
            gTxDataBuffer[2] = 0xFB;
            gTxDataBuffer[4] = value;
        }
        else if(value==0)
        {
            gTxDataBuffer[2] = 0xFC;
            gTxDataBuffer[4] = value;
        }
    }
    else
        value = -1;
}

word MsgGetValue (DWORD offset, DWORD size)
{
    word value;

    if(size<8)
    {
        value = (gRxDataBuffer[(offset/8)+2]>>(7-(offset%8)-size+1))&(0xFF>>(8-size));
    }
    else if (size==8)
    {
        value = (gRxDataBuffer[(offset/8)+2])&0xFF;
    }
    else if (size==16)
    {
        value = (gRxDataBuffer[(offset/8)+2])*256+gRxDataBuffer[(offset/8)+3];
    }
    else
        value = -1;

    return value;
}

void OnNotFoundDID (long did)
{
  long index;
  float value;
  int aux;
    char SoftwareVersionS12Lear[25];
    char SoftwareVersionS12Jlr[17];
```

```
    char SoftwareVersionAtmel[100];
    char SerialNumber[17];

  switch(did)
  {
    /*case 0xF1F5:    // Variant
    {
            putValue(VAR_Variant, gRxDataBuffer[3]);
      break;
    }
    case 0xFD98:    // Software Version S12 Lear
    {
      for(index=0; index<24; ++index)
        SoftwareVersionS12Lear[index] = gRxDataBuffer[3+index];
      SoftwareVersionS12Lear[24] = 0;
      putvalue(VAR_SoftwareVersionS12Lear, SoftwareVersionS12Lear);
      break;
    }
    case 0xF188:    // Software Version S12 JLR
    {
      for(index=0; index<16; ++index)
        SoftwareVersionS12Jlr[index] = gRxDataBuffer[3+index];
      SoftwareVersionS12Jlr[16] = 0;
      putvalue(VAR_SoftwareVersionS12JLR, SoftwareVersionS12Jlr);
      break;
    }
    case 0xF1F0:    // Software Version Atmel
    {
            snprintf(SoftwareVersionAtmel, elcount(SoftwareVersionAtmel)
, "%x %x %x %x %x %x %x %x %x", gRxDataBuffer[3], gRxDataBuffer[4],
gRxDataBuffer[5], gRxDataBuffer[6], gRxDataBuffer[7], gRxDataBuffer[8],
gRxDataBuffer[9], gRxDataBuffer[10], gRxDataBuffer[11]);
      putvalue(VAR_SoftwareVersionAtmel, SoftwareVersionAtmel);
      break;
    }
    case 0xFEE7:    // Start Logic Control
    {
      for(index=0; index<6; ++index)
        StartLogicControl[index] = gRxDataBuffer[3+index];
            putValue(VAR_PowerMode, StartLogicControl[3]);
      break;
    }
    case 0xF18C:    // Serial Number
    {
      for(index=0; index<16; ++index)
        SerialNumber[index] = gRxDataBuffer[3+index];
      SerialNumber[16] = 0;
      putvalue(VAR_SerialNumber, SerialNumber);
      break;
    }*/
    default:
    {
      write("Received unknown DID 0x%4x", did);
      break;
    }
  }
}
```

```
CopyInputsAndDiagsToDataBase ()
{
    if(TYPES[0] == 0 || TYPES[0] == 1 || TYPES[0] == 4 || TYPES[0] == 5 ) putValue(VAR000, VALUES[0]);
    if(TYPES[1] == 0 || TYPES[1] == 1 || TYPES[1] == 4 || TYPES[1] == 5 ) putValue(VAR001, VALUES[1]);
    if(TYPES[2] == 0 || TYPES[2] == 1 || TYPES[2] == 4 || TYPES[2] == 5 ) putValue(VAR002, VALUES[2]);
    if(TYPES[3] == 0 || TYPES[3] == 1 || TYPES[3] == 4 || TYPES[3] == 5 ) putValue(VAR003, VALUES[3]);
    if(TYPES[4] == 0 || TYPES[4] == 1 || TYPES[4] == 4 || TYPES[4] == 5 ) putValue(VAR004, VALUES[4]);
    if(TYPES[5] == 0 || TYPES[5] == 1 || TYPES[5] == 4 || TYPES[5] == 5 ) putValue(VAR005, VALUES[5]);
    if(TYPES[6] == 0 || TYPES[6] == 1 || TYPES[6] == 4 || TYPES[6] == 5 ) putValue(VAR006, VALUES[6]);
    if(TYPES[7] == 0 || TYPES[7] == 1 || TYPES[7] == 4 || TYPES[7] == 5 ) putValue(VAR007, VALUES[7]);
    if(TYPES[8] == 0 || TYPES[8] == 1 || TYPES[8] == 4 || TYPES[8] == 5 ) putValue(VAR008, VALUES[8]);
    if(TYPES[9] == 0 || TYPES[9] == 1 || TYPES[9] == 4 || TYPES[9] == 5 ) putValue(VAR009, VALUES[9]);
    if(TYPES[10] == 0 || TYPES[10] == 1 || TYPES[10] == 4 || TYPES[10] == 5) putValue(VAR010, VALUES[10]);
    if(TYPES[11] == 0 || TYPES[11] == 1 || TYPES[11] == 4 || TYPES[11] == 5) putValue(VAR011, VALUES[11]);
    if(TYPES[12] == 0 || TYPES[12] == 1 || TYPES[12] == 4 || TYPES[12] == 5) putValue(VAR012, VALUES[12]);
    if(TYPES[13] == 0 || TYPES[13] == 1 || TYPES[13] == 4 || TYPES[13] == 5) putValue(VAR013, VALUES[13]);
    if(TYPES[14] == 0 || TYPES[14] == 1 || TYPES[14] == 4 || TYPES[14] == 5) putValue(VAR014, VALUES[14]);
    if(TYPES[15] == 0 || TYPES[15] == 1 || TYPES[15] == 4 || TYPES[15] == 5) putValue(VAR015, VALUES[15]);
    if(TYPES[16] == 0 || TYPES[16] == 1 || TYPES[16] == 4 || TYPES[16] == 5) putValue(VAR016, VALUES[16]);
    if(TYPES[17] == 0 || TYPES[17] == 1 || TYPES[17] == 4 || TYPES[17] == 5) putValue(VAR017, VALUES[17]);
    if(TYPES[18] == 0 || TYPES[18] == 1 || TYPES[18] == 4 || TYPES[18] == 5) putValue(VAR018, VALUES[18]);
    if(TYPES[19] == 0 || TYPES[19] == 1 || TYPES[19] == 4 || TYPES[19] == 5) putValue(VAR019, VALUES[19]);
    if(TYPES[20] == 0 || TYPES[20] == 1 || TYPES[20] == 4 || TYPES[20] == 5) putValue(VAR020, VALUES[20]);
    if(TYPES[21] == 0 || TYPES[21] == 1 || TYPES[21] == 4 || TYPES[21] == 5) putValue(VAR021, VALUES[21]);
    if(TYPES[22] == 0 || TYPES[22] == 1 || TYPES[22] == 4 || TYPES[22] == 5) putValue(VAR022, VALUES[22]);
    if(TYPES[23] == 0 || TYPES[23] == 1 || TYPES[23] == 4 || TYPES[23] == 5) putValue(VAR023, VALUES[23]);
    if(TYPES[24] == 0 || TYPES[24] == 1 || TYPES[24] == 4 || TYPES[24] == 5) putValue(VAR024, VALUES[24]);
    if(TYPES[25] == 0 || TYPES[25] == 1 || TYPES[25] == 4 || TYPES[25] == 5) putValue(VAR025, VALUES[25]);
    if(TYPES[26] == 0 || TYPES[26] == 1 || TYPES[26] == 4 || TYPES[26] == 5) putValue(VAR026, VALUES[26]);
```

```
1366     if(TYPES[27] == 0 || TYPES[27] == 1 || TYPES[27] == 4 || TYPES[27] == 5) putValue(VAR027, VALUES[27]);
1367     if(TYPES[28] == 0 || TYPES[28] == 1 || TYPES[28] == 4 || TYPES[28] == 5) putValue(VAR028, VALUES[28]);
1368     if(TYPES[29] == 0 || TYPES[29] == 1 || TYPES[29] == 4 || TYPES[29] == 5) putValue(VAR029, VALUES[29]);
1369     if(TYPES[30] == 0 || TYPES[30] == 1 || TYPES[30] == 4 || TYPES[30] == 5) putValue(VAR030, VALUES[30]);
1370     if(TYPES[31] == 0 || TYPES[31] == 1 || TYPES[31] == 4 || TYPES[31] == 5) putValue(VAR031, VALUES[31]);
1371     if(TYPES[32] == 0 || TYPES[32] == 1 || TYPES[32] == 4 || TYPES[32] == 5) putValue(VAR032, VALUES[32]);
1372     if(TYPES[33] == 0 || TYPES[33] == 1 || TYPES[33] == 4 || TYPES[33] == 5) putValue(VAR033, VALUES[33]);
1373     if(TYPES[34] == 0 || TYPES[34] == 1 || TYPES[34] == 4 || TYPES[34] == 5) putValue(VAR034, VALUES[34]);
1374     if(TYPES[35] == 0 || TYPES[35] == 1 || TYPES[35] == 4 || TYPES[35] == 5) putValue(VAR035, VALUES[35]);
1375     if(TYPES[36] == 0 || TYPES[36] == 1 || TYPES[36] == 4 || TYPES[36] == 5) putValue(VAR036, VALUES[36]);
1376     if(TYPES[37] == 0 || TYPES[37] == 1 || TYPES[37] == 4 || TYPES[37] == 5) putValue(VAR037, VALUES[37]);
1377     if(TYPES[38] == 0 || TYPES[38] == 1 || TYPES[38] == 4 || TYPES[38] == 5) putValue(VAR038, VALUES[38]);
1378     if(TYPES[39] == 0 || TYPES[39] == 1 || TYPES[39] == 4 || TYPES[39] == 5) putValue(VAR039, VALUES[39]);
1379     if(TYPES[40] == 0 || TYPES[40] == 1 || TYPES[40] == 4 || TYPES[40] == 5) putValue(VAR040, VALUES[40]);
1380     if(TYPES[41] == 0 || TYPES[41] == 1 || TYPES[41] == 4 || TYPES[41] == 5) putValue(VAR041, VALUES[41]);
1381     if(TYPES[42] == 0 || TYPES[42] == 1 || TYPES[42] == 4 || TYPES[42] == 5) putValue(VAR042, VALUES[42]);
1382     if(TYPES[43] == 0 || TYPES[43] == 1 || TYPES[43] == 4 || TYPES[43] == 5) putValue(VAR043, VALUES[43]);
1383     if(TYPES[44] == 0 || TYPES[44] == 1 || TYPES[44] == 4 || TYPES[44] == 5) putValue(VAR044, VALUES[44]);
1384     if(TYPES[45] == 0 || TYPES[45] == 1 || TYPES[45] == 4 || TYPES[45] == 5) putValue(VAR045, VALUES[45]);
1385     if(TYPES[46] == 0 || TYPES[46] == 1 || TYPES[46] == 4 || TYPES[46] == 5) putValue(VAR046, VALUES[46]);
1386     if(TYPES[47] == 0 || TYPES[47] == 1 || TYPES[47] == 4 || TYPES[47] == 5) putValue(VAR047, VALUES[47]);
1387     if(TYPES[48] == 0 || TYPES[48] == 1 || TYPES[48] == 4 || TYPES[48] == 5) putValue(VAR048, VALUES[48]);
1388     if(TYPES[49] == 0 || TYPES[49] == 1 || TYPES[49] == 4 || TYPES[49] == 5) putValue(VAR049, VALUES[49]);
1389     if(TYPES[50] == 0 || TYPES[50] == 1 || TYPES[50] == 4 || TYPES[50] == 5) putValue(VAR050, VALUES[50]);
1390     if(TYPES[51] == 0 || TYPES[51] == 1 || TYPES[51] == 4 || TYPES[51] == 5) putValue(VAR051, VALUES[51]);
1391     if(TYPES[52] == 0 || TYPES[52] == 1 || TYPES[52] == 4 || TYPES[52] == 5) putValue(VAR052, VALUES[52]);
1392     if(TYPES[53] == 0 || TYPES[53] == 1 || TYPES[53] == 4 || TYPES[53] == 5) putValue(VAR053, VALUES[53]);
1393     if(TYPES[54] == 0 || TYPES[54] == 1 || TYPES[54] == 4 || TYPES[54] == 5) putValue(VAR054, VALUES[54]);
1394     if(TYPES[55] == 0 || TYPES[55] == 1 || TYPES[55] == 4 || TYPES[55]
```

```
== 5) putValue(VAR055, VALUES[55]);
     if(TYPES[56] == 0 || TYPES[56] == 1 || TYPES[56] == 4 || TYPES[56]
1395 == 5) putValue(VAR056, VALUES[56]);
     if(TYPES[57] == 0 || TYPES[57] == 1 || TYPES[57] == 4 || TYPES[57]
1396 == 5) putValue(VAR057, VALUES[57]);
     if(TYPES[58] == 0 || TYPES[58] == 1 || TYPES[58] == 4 || TYPES[58]
1397 == 5) putValue(VAR058, VALUES[58]);
     if(TYPES[59] == 0 || TYPES[59] == 1 || TYPES[59] == 4 || TYPES[59]
1398 == 5) putValue(VAR059, VALUES[59]);
     if(TYPES[60] == 0 || TYPES[60] == 1 || TYPES[60] == 4 || TYPES[60]
1399 == 5) putValue(VAR060, VALUES[60]);
     if(TYPES[61] == 0 || TYPES[61] == 1 || TYPES[61] == 4 || TYPES[61]
1400 == 5) putValue(VAR061, VALUES[61]);
     if(TYPES[62] == 0 || TYPES[62] == 1 || TYPES[62] == 4 || TYPES[62]
1401 == 5) putValue(VAR062, VALUES[62]);
     if(TYPES[63] == 0 || TYPES[63] == 1 || TYPES[63] == 4 || TYPES[63]
1402 == 5) putValue(VAR063, VALUES[63]);
     if(TYPES[64] == 0 || TYPES[64] == 1 || TYPES[64] == 4 || TYPES[64]
1403 == 5) putValue(VAR064, VALUES[64]);
     if(TYPES[65] == 0 || TYPES[65] == 1 || TYPES[65] == 4 || TYPES[65]
1404 == 5) putValue(VAR065, VALUES[65]);
     if(TYPES[66] == 0 || TYPES[66] == 1 || TYPES[66] == 4 || TYPES[66]
1405 == 5) putValue(VAR066, VALUES[66]);
     if(TYPES[67] == 0 || TYPES[67] == 1 || TYPES[67] == 4 || TYPES[67]
1406 == 5) putValue(VAR067, VALUES[67]);
     if(TYPES[68] == 0 || TYPES[68] == 1 || TYPES[68] == 4 || TYPES[68]
1407 == 5) putValue(VAR068, VALUES[68]);
     if(TYPES[69] == 0 || TYPES[69] == 1 || TYPES[69] == 4 || TYPES[69]
1408 == 5) putValue(VAR069, VALUES[69]);
     if(TYPES[70] == 0 || TYPES[70] == 1 || TYPES[70] == 4 || TYPES[70]
1409 == 5) putValue(VAR070, VALUES[70]);
     if(TYPES[71] == 0 || TYPES[71] == 1 || TYPES[71] == 4 || TYPES[71]
1410 == 5) putValue(VAR071, VALUES[71]);
     if(TYPES[72] == 0 || TYPES[72] == 1 || TYPES[72] == 4 || TYPES[72]
1411 == 5) putValue(VAR072, VALUES[72]);
     if(TYPES[73] == 0 || TYPES[73] == 1 || TYPES[73] == 4 || TYPES[73]
1412 == 5) putValue(VAR073, VALUES[73]);
     if(TYPES[74] == 0 || TYPES[74] == 1 || TYPES[74] == 4 || TYPES[74]
1413 == 5) putValue(VAR074, VALUES[74]);
     if(TYPES[75] == 0 || TYPES[75] == 1 || TYPES[75] == 4 || TYPES[75]
1414 == 5) putValue(VAR075, VALUES[75]);
     if(TYPES[76] == 0 || TYPES[76] == 1 || TYPES[76] == 4 || TYPES[76]
1415 == 5) putValue(VAR076, VALUES[76]);
     if(TYPES[77] == 0 || TYPES[77] == 1 || TYPES[77] == 4 || TYPES[77]
1416 == 5) putValue(VAR077, VALUES[77]);
     if(TYPES[78] == 0 || TYPES[78] == 1 || TYPES[78] == 4 || TYPES[78]
1417 == 5) putValue(VAR078, VALUES[78]);
     if(TYPES[79] == 0 || TYPES[79] == 1 || TYPES[79] == 4 || TYPES[79]
1418 == 5) putValue(VAR079, VALUES[79]);
     if(TYPES[80] == 0 || TYPES[80] == 1 || TYPES[80] == 4 || TYPES[80]
1419 == 5) putValue(VAR080, VALUES[80]);
     if(TYPES[81] == 0 || TYPES[81] == 1 || TYPES[81] == 4 || TYPES[81]
1420 == 5) putValue(VAR081, VALUES[81]);
     if(TYPES[82] == 0 || TYPES[82] == 1 || TYPES[82] == 4 || TYPES[82]
1421 == 5) putValue(VAR082, VALUES[82]);
     if(TYPES[83] == 0 || TYPES[83] == 1 || TYPES[83] == 4 || TYPES[83]
1422 == 5) putValue(VAR083, VALUES[83]);
```

```
1423     if(TYPES[84] == 0 || TYPES[84] == 1 || TYPES[84] == 4 || TYPES[84]
 == 5) putValue(VAR084, VALUES[84]);
1424     if(TYPES[85] == 0 || TYPES[85] == 1 || TYPES[85] == 4 || TYPES[85]
 == 5) putValue(VAR085, VALUES[85]);
1425     if(TYPES[86] == 0 || TYPES[86] == 1 || TYPES[86] == 4 || TYPES[86]
 == 5) putValue(VAR086, VALUES[86]);
1426     if(TYPES[87] == 0 || TYPES[87] == 1 || TYPES[87] == 4 || TYPES[87]
 == 5) putValue(VAR087, VALUES[87]);
1427     if(TYPES[88] == 0 || TYPES[88] == 1 || TYPES[88] == 4 || TYPES[88]
 == 5) putValue(VAR088, VALUES[88]);
1428     if(TYPES[89] == 0 || TYPES[89] == 1 || TYPES[89] == 4 || TYPES[89]
 == 5) putValue(VAR089, VALUES[89]);
1429     if(TYPES[90] == 0 || TYPES[90] == 1 || TYPES[90] == 4 || TYPES[90]
 == 5) putValue(VAR090, VALUES[90]);
1430     if(TYPES[91] == 0 || TYPES[91] == 1 || TYPES[91] == 4 || TYPES[91]
 == 5) putValue(VAR091, VALUES[91]);
1431     if(TYPES[92] == 0 || TYPES[92] == 1 || TYPES[92] == 4 || TYPES[92]
 == 5) putValue(VAR092, VALUES[92]);
1432     if(TYPES[93] == 0 || TYPES[93] == 1 || TYPES[93] == 4 || TYPES[93]
 == 5) putValue(VAR093, VALUES[93]);
1433     if(TYPES[94] == 0 || TYPES[94] == 1 || TYPES[94] == 4 || TYPES[94]
 == 5) putValue(VAR094, VALUES[94]);
1434     if(TYPES[95] == 0 || TYPES[95] == 1 || TYPES[95] == 4 || TYPES[95]
 == 5) putValue(VAR095, VALUES[95]);
1435     if(TYPES[96] == 0 || TYPES[96] == 1 || TYPES[96] == 4 || TYPES[96]
 == 5) putValue(VAR096, VALUES[96]);
1436     if(TYPES[97] == 0 || TYPES[97] == 1 || TYPES[97] == 4 || TYPES[97]
 == 5) putValue(VAR097, VALUES[97]);
1437     if(TYPES[98] == 0 || TYPES[98] == 1 || TYPES[98] == 4 || TYPES[98]
 == 5) putValue(VAR098, VALUES[98]);
1438     if(TYPES[99] == 0 || TYPES[99] == 1 || TYPES[99] == 4 || TYPES[99]
 == 5) putValue(VAR099, VALUES[99]);
1439     if(TYPES[100] == 0 || TYPES[100] == 1 || TYPES[100] == 4 || TYPES[
100] == 5) putValue(VAR100, VALUES[100]);
1440     if(TYPES[101] == 0 || TYPES[101] == 1 || TYPES[101] == 4 || TYPES[
101] == 5) putValue(VAR101, VALUES[101]);
1441     if(TYPES[102] == 0 || TYPES[102] == 1 || TYPES[102] == 4 || TYPES[
102] == 5) putValue(VAR102, VALUES[102]);
1442     if(TYPES[103] == 0 || TYPES[103] == 1 || TYPES[103] == 4 || TYPES[
103] == 5) putValue(VAR103, VALUES[103]);
1443     if(TYPES[104] == 0 || TYPES[104] == 1 || TYPES[104] == 4 || TYPES[
104] == 5) putValue(VAR104, VALUES[104]);
1444     if(TYPES[105] == 0 || TYPES[105] == 1 || TYPES[105] == 4 || TYPES[
105] == 5) putValue(VAR105, VALUES[105]);
1445     if(TYPES[106] == 0 || TYPES[106] == 1 || TYPES[106] == 4 || TYPES[
106] == 5) putValue(VAR106, VALUES[106]);
1446     if(TYPES[107] == 0 || TYPES[107] == 1 || TYPES[107] == 4 || TYPES[
107] == 5) putValue(VAR107, VALUES[107]);
1447     if(TYPES[108] == 0 || TYPES[108] == 1 || TYPES[108] == 4 || TYPES[
108] == 5) putValue(VAR108, VALUES[108]);
1448     if(TYPES[109] == 0 || TYPES[109] == 1 || TYPES[109] == 4 || TYPES[
109] == 5) putValue(VAR109, VALUES[109]);
1449     if(TYPES[110] == 0 || TYPES[110] == 1 || TYPES[110] == 4 || TYPES[
110] == 5) putValue(VAR110, VALUES[110]);
1450     if(TYPES[111] == 0 || TYPES[111] == 1 || TYPES[111] == 4 || TYPES[
111] == 5) putValue(VAR111, VALUES[111]);
1451     if(TYPES[112] == 0 || TYPES[112] == 1 || TYPES[112] == 4 || TYPES[
```

```
112] == 5) putValue(VAR112, VALUES[112]);
    if(TYPES[113] == 0 || TYPES[113] == 1 || TYPES[113] == 4 || TYPES[
1452 113] == 5) putValue(VAR113, VALUES[113]);
    if(TYPES[114] == 0 || TYPES[114] == 1 || TYPES[114] == 4 || TYPES[
1453 114] == 5) putValue(VAR114, VALUES[114]);
    if(TYPES[115] == 0 || TYPES[115] == 1 || TYPES[115] == 4 || TYPES[
1454 115] == 5) putValue(VAR115, VALUES[115]);
    if(TYPES[116] == 0 || TYPES[116] == 1 || TYPES[116] == 4 || TYPES[
1455 116] == 5) putValue(VAR116, VALUES[116]);
    if(TYPES[117] == 0 || TYPES[117] == 1 || TYPES[117] == 4 || TYPES[
1456 117] == 5) putValue(VAR117, VALUES[117]);
    if(TYPES[118] == 0 || TYPES[118] == 1 || TYPES[118] == 4 || TYPES[
1457 118] == 5) putValue(VAR118, VALUES[118]);
    if(TYPES[119] == 0 || TYPES[119] == 1 || TYPES[119] == 4 || TYPES[
1458 119] == 5) putValue(VAR119, VALUES[119]);
    if(TYPES[120] == 0 || TYPES[120] == 1 || TYPES[120] == 4 || TYPES[
1459 120] == 5) putValue(VAR120, VALUES[120]);
    if(TYPES[121] == 0 || TYPES[121] == 1 || TYPES[121] == 4 || TYPES[
1460 121] == 5) putValue(VAR121, VALUES[121]);
    if(TYPES[122] == 0 || TYPES[122] == 1 || TYPES[122] == 4 || TYPES[
1461 122] == 5) putValue(VAR122, VALUES[122]);
    if(TYPES[123] == 0 || TYPES[123] == 1 || TYPES[123] == 4 || TYPES[
1462 123] == 5) putValue(VAR123, VALUES[123]);
    if(TYPES[124] == 0 || TYPES[124] == 1 || TYPES[124] == 4 || TYPES[
1463 124] == 5) putValue(VAR124, VALUES[124]);
    if(TYPES[125] == 0 || TYPES[125] == 1 || TYPES[125] == 4 || TYPES[
1464 125] == 5) putValue(VAR125, VALUES[125]);
    if(TYPES[126] == 0 || TYPES[126] == 1 || TYPES[126] == 4 || TYPES[
1465 126] == 5) putValue(VAR126, VALUES[126]);
    if(TYPES[127] == 0 || TYPES[127] == 1 || TYPES[127] == 4 || TYPES[
1466 127] == 5) putValue(VAR127, VALUES[127]);
    if(TYPES[128] == 0 || TYPES[128] == 1 || TYPES[128] == 4 || TYPES[
1467 128] == 5) putValue(VAR128, VALUES[128]);
    if(TYPES[129] == 0 || TYPES[129] == 1 || TYPES[129] == 4 || TYPES[
1468 129] == 5) putValue(VAR129, VALUES[129]);
    if(TYPES[130] == 0 || TYPES[130] == 1 || TYPES[130] == 4 || TYPES[
1469 130] == 5) putValue(VAR130, VALUES[130]);
    if(TYPES[131] == 0 || TYPES[131] == 1 || TYPES[131] == 4 || TYPES[
1470 131] == 5) putValue(VAR131, VALUES[131]);
    if(TYPES[132] == 0 || TYPES[132] == 1 || TYPES[132] == 4 || TYPES[
1471 132] == 5) putValue(VAR132, VALUES[132]);
    if(TYPES[133] == 0 || TYPES[133] == 1 || TYPES[133] == 4 || TYPES[
1472 133] == 5) putValue(VAR133, VALUES[133]);
    if(TYPES[134] == 0 || TYPES[134] == 1 || TYPES[134] == 4 || TYPES[
1473 134] == 5) putValue(VAR134, VALUES[134]);
    if(TYPES[135] == 0 || TYPES[135] == 1 || TYPES[135] == 4 || TYPES[
1474 135] == 5) putValue(VAR135, VALUES[135]);
    if(TYPES[136] == 0 || TYPES[136] == 1 || TYPES[136] == 4 || TYPES[
1475 136] == 5) putValue(VAR136, VALUES[136]);
    if(TYPES[137] == 0 || TYPES[137] == 1 || TYPES[137] == 4 || TYPES[
1476 137] == 5) putValue(VAR137, VALUES[137]);
    if(TYPES[138] == 0 || TYPES[138] == 1 || TYPES[138] == 4 || TYPES[
1477 138] == 5) putValue(VAR138, VALUES[138]);
    if(TYPES[139] == 0 || TYPES[139] == 1 || TYPES[139] == 4 || TYPES[
1478 139] == 5) putValue(VAR139, VALUES[139]);
    if(TYPES[140] == 0 || TYPES[140] == 1 || TYPES[140] == 4 || TYPES[
1479 140] == 5) putValue(VAR140, VALUES[140]);
```

```
1480     if(TYPES[141] == 0 || TYPES[141] == 1 || TYPES[141] == 4 || TYPES[
    141] == 5) putValue(VAR141, VALUES[141]);
1481     if(TYPES[142] == 0 || TYPES[142] == 1 || TYPES[142] == 4 || TYPES[
    142] == 5) putValue(VAR142, VALUES[142]);
1482     if(TYPES[143] == 0 || TYPES[143] == 1 || TYPES[143] == 4 || TYPES[
    143] == 5) putValue(VAR143, VALUES[143]);
1483     if(TYPES[144] == 0 || TYPES[144] == 1 || TYPES[144] == 4 || TYPES[
    144] == 5) putValue(VAR144, VALUES[144]);
1484     if(TYPES[145] == 0 || TYPES[145] == 1 || TYPES[145] == 4 || TYPES[
    145] == 5) putValue(VAR145, VALUES[145]);
1485     if(TYPES[146] == 0 || TYPES[146] == 1 || TYPES[146] == 4 || TYPES[
    146] == 5) putValue(VAR146, VALUES[146]);
1486     if(TYPES[147] == 0 || TYPES[147] == 1 || TYPES[147] == 4 || TYPES[
    147] == 5) putValue(VAR147, VALUES[147]);
1487     if(TYPES[148] == 0 || TYPES[148] == 1 || TYPES[148] == 4 || TYPES[
    148] == 5) putValue(VAR148, VALUES[148]);
1488     if(TYPES[149] == 0 || TYPES[149] == 1 || TYPES[149] == 4 || TYPES[
    149] == 5) putValue(VAR149, VALUES[149]);
1489     if(TYPES[150] == 0 || TYPES[150] == 1 || TYPES[150] == 4 || TYPES[
    150] == 5) putValue(VAR150, VALUES[150]);
1490     if(TYPES[151] == 0 || TYPES[151] == 1 || TYPES[151] == 4 || TYPES[
    151] == 5) putValue(VAR151, VALUES[151]);
1491     if(TYPES[152] == 0 || TYPES[152] == 1 || TYPES[152] == 4 || TYPES[
    152] == 5) putValue(VAR152, VALUES[152]);
1492     if(TYPES[153] == 0 || TYPES[153] == 1 || TYPES[153] == 4 || TYPES[
    153] == 5) putValue(VAR153, VALUES[153]);
1493     if(TYPES[154] == 0 || TYPES[154] == 1 || TYPES[154] == 4 || TYPES[
    154] == 5) putValue(VAR154, VALUES[154]);
1494     if(TYPES[155] == 0 || TYPES[155] == 1 || TYPES[155] == 4 || TYPES[
    155] == 5) putValue(VAR155, VALUES[155]);
1495     if(TYPES[156] == 0 || TYPES[156] == 1 || TYPES[156] == 4 || TYPES[
    156] == 5) putValue(VAR156, VALUES[156]);
1496     if(TYPES[157] == 0 || TYPES[157] == 1 || TYPES[157] == 4 || TYPES[
    157] == 5) putValue(VAR157, VALUES[157]);
1497     if(TYPES[158] == 0 || TYPES[158] == 1 || TYPES[158] == 4 || TYPES[
    158] == 5) putValue(VAR158, VALUES[158]);
1498     if(TYPES[159] == 0 || TYPES[159] == 1 || TYPES[159] == 4 || TYPES[
    159] == 5) putValue(VAR159, VALUES[159]);
1499     if(TYPES[160] == 0 || TYPES[160] == 1 || TYPES[160] == 4 || TYPES[
    160] == 5) putValue(VAR160, VALUES[160]);
1500     if(TYPES[161] == 0 || TYPES[161] == 1 || TYPES[161] == 4 || TYPES[
    161] == 5) putValue(VAR161, VALUES[161]);
1501     if(TYPES[162] == 0 || TYPES[162] == 1 || TYPES[162] == 4 || TYPES[
    162] == 5) putValue(VAR162, VALUES[162]);
1502     if(TYPES[163] == 0 || TYPES[163] == 1 || TYPES[163] == 4 || TYPES[
    163] == 5) putValue(VAR163, VALUES[163]);
1503     if(TYPES[164] == 0 || TYPES[164] == 1 || TYPES[164] == 4 || TYPES[
    164] == 5) putValue(VAR164, VALUES[164]);
1504     if(TYPES[165] == 0 || TYPES[165] == 1 || TYPES[165] == 4 || TYPES[
    165] == 5) putValue(VAR165, VALUES[165]);
1505     if(TYPES[166] == 0 || TYPES[166] == 1 || TYPES[166] == 4 || TYPES[
    166] == 5) putValue(VAR166, VALUES[166]);
1506     if(TYPES[167] == 0 || TYPES[167] == 1 || TYPES[167] == 4 || TYPES[
    167] == 5) putValue(VAR167, VALUES[167]);
1507     if(TYPES[168] == 0 || TYPES[168] == 1 || TYPES[168] == 4 || TYPES[
    168] == 5) putValue(VAR168, VALUES[168]);
1508     if(TYPES[169] == 0 || TYPES[169] == 1 || TYPES[169] == 4 || TYPES[
```

```
169] == 5) putValue(VAR169, VALUES[169]);
    if(TYPES[170] == 0 || TYPES[170] == 1 || TYPES[170] == 4 || TYPES[
1509 170] == 5) putValue(VAR170, VALUES[170]);
    if(TYPES[171] == 0 || TYPES[171] == 1 || TYPES[171] == 4 || TYPES[
1510 171] == 5) putValue(VAR171, VALUES[171]);
    if(TYPES[172] == 0 || TYPES[172] == 1 || TYPES[172] == 4 || TYPES[
1511 172] == 5) putValue(VAR172, VALUES[172]);
    if(TYPES[173] == 0 || TYPES[173] == 1 || TYPES[173] == 4 || TYPES[
1512 173] == 5) putValue(VAR173, VALUES[173]);
    if(TYPES[174] == 0 || TYPES[174] == 1 || TYPES[174] == 4 || TYPES[
1513 174] == 5) putValue(VAR174, VALUES[174]);
    if(TYPES[175] == 0 || TYPES[175] == 1 || TYPES[175] == 4 || TYPES[
1514 175] == 5) putValue(VAR175, VALUES[175]);
    if(TYPES[176] == 0 || TYPES[176] == 1 || TYPES[176] == 4 || TYPES[
1515 176] == 5) putValue(VAR176, VALUES[176]);
    if(TYPES[177] == 0 || TYPES[177] == 1 || TYPES[177] == 4 || TYPES[
1516 177] == 5) putValue(VAR177, VALUES[177]);
    if(TYPES[178] == 0 || TYPES[178] == 1 || TYPES[178] == 4 || TYPES[
1517 178] == 5) putValue(VAR178, VALUES[178]);
    if(TYPES[179] == 0 || TYPES[179] == 1 || TYPES[179] == 4 || TYPES[
1518 179] == 5) putValue(VAR179, VALUES[179]);
    if(TYPES[180] == 0 || TYPES[180] == 1 || TYPES[180] == 4 || TYPES[
1519 180] == 5) putValue(VAR180, VALUES[180]);
    if(TYPES[181] == 0 || TYPES[181] == 1 || TYPES[181] == 4 || TYPES[
1520 181] == 5) putValue(VAR181, VALUES[181]);
    if(TYPES[182] == 0 || TYPES[182] == 1 || TYPES[182] == 4 || TYPES[
1521 182] == 5) putValue(VAR182, VALUES[182]);
    if(TYPES[183] == 0 || TYPES[183] == 1 || TYPES[183] == 4 || TYPES[
1522 183] == 5) putValue(VAR183, VALUES[183]);
    if(TYPES[184] == 0 || TYPES[184] == 1 || TYPES[184] == 4 || TYPES[
1523 184] == 5) putValue(VAR184, VALUES[184]);
    if(TYPES[185] == 0 || TYPES[185] == 1 || TYPES[185] == 4 || TYPES[
1524 185] == 5) putValue(VAR185, VALUES[185]);
    if(TYPES[186] == 0 || TYPES[186] == 1 || TYPES[186] == 4 || TYPES[
1525 186] == 5) putValue(VAR186, VALUES[186]);
    if(TYPES[187] == 0 || TYPES[187] == 1 || TYPES[187] == 4 || TYPES[
1526 187] == 5) putValue(VAR187, VALUES[187]);
    if(TYPES[188] == 0 || TYPES[188] == 1 || TYPES[188] == 4 || TYPES[
1527 188] == 5) putValue(VAR188, VALUES[188]);
    if(TYPES[189] == 0 || TYPES[189] == 1 || TYPES[189] == 4 || TYPES[
1528 189] == 5) putValue(VAR189, VALUES[189]);
    if(TYPES[190] == 0 || TYPES[190] == 1 || TYPES[190] == 4 || TYPES[
1529 190] == 5) putValue(VAR190, VALUES[190]);
    if(TYPES[191] == 0 || TYPES[191] == 1 || TYPES[191] == 4 || TYPES[
1530 191] == 5) putValue(VAR191, VALUES[191]);
    if(TYPES[192] == 0 || TYPES[192] == 1 || TYPES[192] == 4 || TYPES[
1531 192] == 5) putValue(VAR192, VALUES[192]);
    if(TYPES[193] == 0 || TYPES[193] == 1 || TYPES[193] == 4 || TYPES[
1532 193] == 5) putValue(VAR193, VALUES[193]);
    if(TYPES[194] == 0 || TYPES[194] == 1 || TYPES[194] == 4 || TYPES[
1533 194] == 5) putValue(VAR194, VALUES[194]);
    if(TYPES[195] == 0 || TYPES[195] == 1 || TYPES[195] == 4 || TYPES[
1534 195] == 5) putValue(VAR195, VALUES[195]);
    if(TYPES[196] == 0 || TYPES[196] == 1 || TYPES[196] == 4 || TYPES[
1535 196] == 5) putValue(VAR196, VALUES[196]);
    if(TYPES[197] == 0 || TYPES[197] == 1 || TYPES[197] == 4 || TYPES[
1536 197] == 5) putValue(VAR197, VALUES[197]);
```

```
1537     if(TYPES[198] == 0 || TYPES[198] == 1 || TYPES[198] == 4 || TYPES[198] == 5) putValue(VAR198, VALUES[198]);
1538     if(TYPES[199] == 0 || TYPES[199] == 1 || TYPES[199] == 4 || TYPES[199] == 5) putValue(VAR199, VALUES[199]);
1539     if(TYPES[200] == 0 || TYPES[200] == 1 || TYPES[200] == 4 || TYPES[200] == 5) putValue(VAR200, VALUES[200]);
1540     if(TYPES[201] == 0 || TYPES[201] == 1 || TYPES[201] == 4 || TYPES[201] == 5) putValue(VAR201, VALUES[201]);
1541     if(TYPES[202] == 0 || TYPES[202] == 1 || TYPES[202] == 4 || TYPES[202] == 5) putValue(VAR202, VALUES[202]);
1542     if(TYPES[203] == 0 || TYPES[203] == 1 || TYPES[203] == 4 || TYPES[203] == 5) putValue(VAR203, VALUES[203]);
1543     if(TYPES[204] == 0 || TYPES[204] == 1 || TYPES[204] == 4 || TYPES[204] == 5) putValue(VAR204, VALUES[204]);
1544     if(TYPES[205] == 0 || TYPES[205] == 1 || TYPES[205] == 4 || TYPES[205] == 5) putValue(VAR205, VALUES[205]);
1545     if(TYPES[206] == 0 || TYPES[206] == 1 || TYPES[206] == 4 || TYPES[206] == 5) putValue(VAR206, VALUES[206]);
1546     if(TYPES[207] == 0 || TYPES[207] == 1 || TYPES[207] == 4 || TYPES[207] == 5) putValue(VAR207, VALUES[207]);
1547     if(TYPES[208] == 0 || TYPES[208] == 1 || TYPES[208] == 4 || TYPES[208] == 5) putValue(VAR208, VALUES[208]);
1548     if(TYPES[209] == 0 || TYPES[209] == 1 || TYPES[209] == 4 || TYPES[209] == 5) putValue(VAR209, VALUES[209]);
1549     if(TYPES[210] == 0 || TYPES[210] == 1 || TYPES[210] == 4 || TYPES[210] == 5) putValue(VAR210, VALUES[210]);
1550     if(TYPES[211] == 0 || TYPES[211] == 1 || TYPES[211] == 4 || TYPES[211] == 5) putValue(VAR211, VALUES[211]);
1551     if(TYPES[212] == 0 || TYPES[212] == 1 || TYPES[212] == 4 || TYPES[212] == 5) putValue(VAR212, VALUES[212]);
1552     if(TYPES[213] == 0 || TYPES[213] == 1 || TYPES[213] == 4 || TYPES[213] == 5) putValue(VAR213, VALUES[213]);
1553     if(TYPES[214] == 0 || TYPES[214] == 1 || TYPES[214] == 4 || TYPES[214] == 5) putValue(VAR214, VALUES[214]);
1554     if(TYPES[215] == 0 || TYPES[215] == 1 || TYPES[215] == 4 || TYPES[215] == 5) putValue(VAR215, VALUES[215]);
1555     if(TYPES[216] == 0 || TYPES[216] == 1 || TYPES[216] == 4 || TYPES[216] == 5) putValue(VAR216, VALUES[216]);
1556     if(TYPES[217] == 0 || TYPES[217] == 1 || TYPES[217] == 4 || TYPES[217] == 5) putValue(VAR217, VALUES[217]);
1557     if(TYPES[218] == 0 || TYPES[218] == 1 || TYPES[218] == 4 || TYPES[218] == 5) putValue(VAR218, VALUES[218]);
1558     if(TYPES[219] == 0 || TYPES[219] == 1 || TYPES[219] == 4 || TYPES[219] == 5) putValue(VAR219, VALUES[219]);
1559     if(TYPES[220] == 0 || TYPES[220] == 1 || TYPES[220] == 4 || TYPES[220] == 5) putValue(VAR220, VALUES[220]);
1560     if(TYPES[221] == 0 || TYPES[221] == 1 || TYPES[221] == 4 || TYPES[221] == 5) putValue(VAR221, VALUES[221]);
1561     if(TYPES[222] == 0 || TYPES[222] == 1 || TYPES[222] == 4 || TYPES[222] == 5) putValue(VAR222, VALUES[222]);
1562     if(TYPES[223] == 0 || TYPES[223] == 1 || TYPES[223] == 4 || TYPES[223] == 5) putValue(VAR223, VALUES[223]);
1563     if(TYPES[224] == 0 || TYPES[224] == 1 || TYPES[224] == 4 || TYPES[224] == 5) putValue(VAR224, VALUES[224]);
1564     if(TYPES[225] == 0 || TYPES[225] == 1 || TYPES[225] == 4 || TYPES[225] == 5) putValue(VAR225, VALUES[225]);
1565     if(TYPES[226] == 0 || TYPES[226] == 1 || TYPES[226] == 4 || TYPES[226]
```

```
226] == 5) putValue(VAR226, VALUES[226]);
    if(TYPES[227] == 0 || TYPES[227] == 1 || TYPES[227] == 4 || TYPES[
1566 227] == 5) putValue(VAR227, VALUES[227]);
    if(TYPES[228] == 0 || TYPES[228] == 1 || TYPES[228] == 4 || TYPES[
1567 228] == 5) putValue(VAR228, VALUES[228]);
    if(TYPES[229] == 0 || TYPES[229] == 1 || TYPES[229] == 4 || TYPES[
1568 229] == 5) putValue(VAR229, VALUES[229]);
    if(TYPES[230] == 0 || TYPES[230] == 1 || TYPES[230] == 4 || TYPES[
1569 230] == 5) putValue(VAR230, VALUES[230]);
    if(TYPES[231] == 0 || TYPES[231] == 1 || TYPES[231] == 4 || TYPES[
1570 231] == 5) putValue(VAR231, VALUES[231]);
    if(TYPES[232] == 0 || TYPES[232] == 1 || TYPES[232] == 4 || TYPES[
1571 232] == 5) putValue(VAR232, VALUES[232]);
    if(TYPES[233] == 0 || TYPES[233] == 1 || TYPES[233] == 4 || TYPES[
1572 233] == 5) putValue(VAR233, VALUES[233]);
    if(TYPES[234] == 0 || TYPES[234] == 1 || TYPES[234] == 4 || TYPES[
1573 234] == 5) putValue(VAR234, VALUES[234]);
    if(TYPES[235] == 0 || TYPES[235] == 1 || TYPES[235] == 4 || TYPES[
1574 235] == 5) putValue(VAR235, VALUES[235]);
    if(TYPES[236] == 0 || TYPES[236] == 1 || TYPES[236] == 4 || TYPES[
1575 236] == 5) putValue(VAR236, VALUES[236]);
    if(TYPES[237] == 0 || TYPES[237] == 1 || TYPES[237] == 4 || TYPES[
1576 237] == 5) putValue(VAR237, VALUES[237]);
    if(TYPES[238] == 0 || TYPES[238] == 1 || TYPES[238] == 4 || TYPES[
1577 238] == 5) putValue(VAR238, VALUES[238]);
    if(TYPES[239] == 0 || TYPES[239] == 1 || TYPES[239] == 4 || TYPES[
1578 239] == 5) putValue(VAR239, VALUES[239]);
    if(TYPES[240] == 0 || TYPES[240] == 1 || TYPES[240] == 4 || TYPES[
1579 240] == 5) putValue(VAR240, VALUES[240]);
    if(TYPES[241] == 0 || TYPES[241] == 1 || TYPES[241] == 4 || TYPES[
1580 241] == 5) putValue(VAR241, VALUES[241]);
    if(TYPES[242] == 0 || TYPES[242] == 1 || TYPES[242] == 4 || TYPES[
1581 242] == 5) putValue(VAR242, VALUES[242]);
    if(TYPES[243] == 0 || TYPES[243] == 1 || TYPES[243] == 4 || TYPES[
1582 243] == 5) putValue(VAR243, VALUES[243]);
    if(TYPES[244] == 0 || TYPES[244] == 1 || TYPES[244] == 4 || TYPES[
1583 244] == 5) putValue(VAR244, VALUES[244]);
    if(TYPES[245] == 0 || TYPES[245] == 1 || TYPES[245] == 4 || TYPES[
1584 245] == 5) putValue(VAR245, VALUES[245]);
    if(TYPES[246] == 0 || TYPES[246] == 1 || TYPES[246] == 4 || TYPES[
1585 246] == 5) putValue(VAR246, VALUES[246]);
    if(TYPES[247] == 0 || TYPES[247] == 1 || TYPES[247] == 4 || TYPES[
1586 247] == 5) putValue(VAR247, VALUES[247]);
    if(TYPES[248] == 0 || TYPES[248] == 1 || TYPES[248] == 4 || TYPES[
1587 248] == 5) putValue(VAR248, VALUES[248]);
    if(TYPES[249] == 0 || TYPES[249] == 1 || TYPES[249] == 4 || TYPES[
1588 249] == 5) putValue(VAR249, VALUES[249]);
    if(TYPES[250] == 0 || TYPES[250] == 1 || TYPES[250] == 4 || TYPES[
1589 250] == 5) putValue(VAR250, VALUES[250]);
    if(TYPES[251] == 0 || TYPES[251] == 1 || TYPES[251] == 4 || TYPES[
1590 251] == 5) putValue(VAR251, VALUES[251]);
    if(TYPES[252] == 0 || TYPES[252] == 1 || TYPES[252] == 4 || TYPES[
1591 252] == 5) putValue(VAR252, VALUES[252]);
    if(TYPES[253] == 0 || TYPES[253] == 1 || TYPES[253] == 4 || TYPES[
1592 253] == 5) putValue(VAR253, VALUES[253]);
    if(TYPES[254] == 0 || TYPES[254] == 1 || TYPES[254] == 4 || TYPES[
1593 254] == 5) putValue(VAR254, VALUES[254]);
```

```
1594     if(TYPES[255] == 0 || TYPES[255] == 1 || TYPES[255] == 4 || TYPES[
     255] == 5) putValue(VAR255, VALUES[255]);
1595     if(TYPES[256] == 0 || TYPES[256] == 1 || TYPES[256] == 4 || TYPES[
     256] == 5) putValue(VAR256, VALUES[256]);
1596     if(TYPES[257] == 0 || TYPES[257] == 1 || TYPES[257] == 4 || TYPES[
     257] == 5) putValue(VAR257, VALUES[257]);
1597     if(TYPES[258] == 0 || TYPES[258] == 1 || TYPES[258] == 4 || TYPES[
     258] == 5) putValue(VAR258, VALUES[258]);
1598     if(TYPES[259] == 0 || TYPES[259] == 1 || TYPES[259] == 4 || TYPES[
     259] == 5) putValue(VAR259, VALUES[259]);
1599     if(TYPES[260] == 0 || TYPES[260] == 1 || TYPES[260] == 4 || TYPES[
     260] == 5) putValue(VAR260, VALUES[260]);
1600     if(TYPES[261] == 0 || TYPES[261] == 1 || TYPES[261] == 4 || TYPES[
     261] == 5) putValue(VAR261, VALUES[261]);
1601     if(TYPES[262] == 0 || TYPES[262] == 1 || TYPES[262] == 4 || TYPES[
     262] == 5) putValue(VAR262, VALUES[262]);
1602     if(TYPES[263] == 0 || TYPES[263] == 1 || TYPES[263] == 4 || TYPES[
     263] == 5) putValue(VAR263, VALUES[263]);
1603     if(TYPES[264] == 0 || TYPES[264] == 1 || TYPES[264] == 4 || TYPES[
     264] == 5) putValue(VAR264, VALUES[264]);
1604     if(TYPES[265] == 0 || TYPES[265] == 1 || TYPES[265] == 4 || TYPES[
     265] == 5) putValue(VAR265, VALUES[265]);
1605     if(TYPES[266] == 0 || TYPES[266] == 1 || TYPES[266] == 4 || TYPES[
     266] == 5) putValue(VAR266, VALUES[266]);
1606     if(TYPES[267] == 0 || TYPES[267] == 1 || TYPES[267] == 4 || TYPES[
     267] == 5) putValue(VAR267, VALUES[267]);
1607     if(TYPES[268] == 0 || TYPES[268] == 1 || TYPES[268] == 4 || TYPES[
     268] == 5) putValue(VAR268, VALUES[268]);
1608     if(TYPES[269] == 0 || TYPES[269] == 1 || TYPES[269] == 4 || TYPES[
     269] == 5) putValue(VAR269, VALUES[269]);
1609     if(TYPES[270] == 0 || TYPES[270] == 1 || TYPES[270] == 4 || TYPES[
     270] == 5) putValue(VAR270, VALUES[270]);
1610     if(TYPES[271] == 0 || TYPES[271] == 1 || TYPES[271] == 4 || TYPES[
     271] == 5) putValue(VAR271, VALUES[271]);
1611     if(TYPES[272] == 0 || TYPES[272] == 1 || TYPES[272] == 4 || TYPES[
     272] == 5) putValue(VAR272, VALUES[272]);
1612     if(TYPES[273] == 0 || TYPES[273] == 1 || TYPES[273] == 4 || TYPES[
     273] == 5) putValue(VAR273, VALUES[273]);
1613     if(TYPES[274] == 0 || TYPES[274] == 1 || TYPES[274] == 4 || TYPES[
     274] == 5) putValue(VAR274, VALUES[274]);
1614     if(TYPES[275] == 0 || TYPES[275] == 1 || TYPES[275] == 4 || TYPES[
     275] == 5) putValue(VAR275, VALUES[275]);
1615     if(TYPES[276] == 0 || TYPES[276] == 1 || TYPES[276] == 4 || TYPES[
     276] == 5) putValue(VAR276, VALUES[276]);
1616     if(TYPES[277] == 0 || TYPES[277] == 1 || TYPES[277] == 4 || TYPES[
     277] == 5) putValue(VAR277, VALUES[277]);
1617     if(TYPES[278] == 0 || TYPES[278] == 1 || TYPES[278] == 4 || TYPES[
     278] == 5) putValue(VAR278, VALUES[278]);
1618     if(TYPES[279] == 0 || TYPES[279] == 1 || TYPES[279] == 4 || TYPES[
     279] == 5) putValue(VAR279, VALUES[279]);
1619     if(TYPES[280] == 0 || TYPES[280] == 1 || TYPES[280] == 4 || TYPES[
     280] == 5) putValue(VAR280, VALUES[280]);
1620     if(TYPES[281] == 0 || TYPES[281] == 1 || TYPES[281] == 4 || TYPES[
     281] == 5) putValue(VAR281, VALUES[281]);
1621     if(TYPES[282] == 0 || TYPES[282] == 1 || TYPES[282] == 4 || TYPES[
     282] == 5) putValue(VAR282, VALUES[282]);
1622     if(TYPES[283] == 0 || TYPES[283] == 1 || TYPES[283] == 4 || TYPES[
```

```
283] == 5) putValue(VAR283, VALUES[283]);
    if(TYPES[284] == 0 || TYPES[284] == 1 || TYPES[284] == 4 || TYPES[
1623 284] == 5) putValue(VAR284, VALUES[284]);
    if(TYPES[285] == 0 || TYPES[285] == 1 || TYPES[285] == 4 || TYPES[
1624 285] == 5) putValue(VAR285, VALUES[285]);
    if(TYPES[286] == 0 || TYPES[286] == 1 || TYPES[286] == 4 || TYPES[
1625 286] == 5) putValue(VAR286, VALUES[286]);
    if(TYPES[287] == 0 || TYPES[287] == 1 || TYPES[287] == 4 || TYPES[
1626 287] == 5) putValue(VAR287, VALUES[287]);
    if(TYPES[288] == 0 || TYPES[288] == 1 || TYPES[288] == 4 || TYPES[
1627 288] == 5) putValue(VAR288, VALUES[288]);
    if(TYPES[289] == 0 || TYPES[289] == 1 || TYPES[289] == 4 || TYPES[
1628 289] == 5) putValue(VAR289, VALUES[289]);
    if(TYPES[290] == 0 || TYPES[290] == 1 || TYPES[290] == 4 || TYPES[
1629 290] == 5) putValue(VAR290, VALUES[290]);
    if(TYPES[291] == 0 || TYPES[291] == 1 || TYPES[291] == 4 || TYPES[
1630 291] == 5) putValue(VAR291, VALUES[291]);
    if(TYPES[292] == 0 || TYPES[292] == 1 || TYPES[292] == 4 || TYPES[
1631 292] == 5) putValue(VAR292, VALUES[292]);
    if(TYPES[293] == 0 || TYPES[293] == 1 || TYPES[293] == 4 || TYPES[
1632 293] == 5) putValue(VAR293, VALUES[293]);
    if(TYPES[294] == 0 || TYPES[294] == 1 || TYPES[294] == 4 || TYPES[
1633 294] == 5) putValue(VAR294, VALUES[294]);
    if(TYPES[295] == 0 || TYPES[295] == 1 || TYPES[295] == 4 || TYPES[
1634 295] == 5) putValue(VAR295, VALUES[295]);
    if(TYPES[296] == 0 || TYPES[296] == 1 || TYPES[296] == 4 || TYPES[
1635 296] == 5) putValue(VAR296, VALUES[296]);
    if(TYPES[297] == 0 || TYPES[297] == 1 || TYPES[297] == 4 || TYPES[
1636 297] == 5) putValue(VAR297, VALUES[297]);
    if(TYPES[298] == 0 || TYPES[298] == 1 || TYPES[298] == 4 || TYPES[
1637 298] == 5) putValue(VAR298, VALUES[298]);
    if(TYPES[299] == 0 || TYPES[299] == 1 || TYPES[299] == 4 || TYPES[
1638 299] == 5) putValue(VAR299, VALUES[299]);
}
1639
1640 CopyValueFromDataBase ()
1641 {
1642     if(TYPES[0] == 2 || TYPES[0] == 3) tmp[0] = getValue(VAR000);
1643     if(TYPES[1] == 2 || TYPES[1] == 3) tmp[1] = getValue(VAR001);
1644     if(TYPES[2] == 2 || TYPES[2] == 3) tmp[2] = getValue(VAR002);
1645     if(TYPES[3] == 2 || TYPES[3] == 3) tmp[3] = getValue(VAR003);
1646     if(TYPES[4] == 2 || TYPES[4] == 3) tmp[4] = getValue(VAR004);
1647     if(TYPES[5] == 2 || TYPES[5] == 3) tmp[5] = getValue(VAR005);
1648     if(TYPES[6] == 2 || TYPES[6] == 3) tmp[6] = getValue(VAR006);
1649     if(TYPES[7] == 2 || TYPES[7] == 3) tmp[7] = getValue(VAR007);
1650     if(TYPES[8] == 2 || TYPES[8] == 3) tmp[8] = getValue(VAR008);
1651     if(TYPES[9] == 2 || TYPES[9] == 3) tmp[9] = getValue(VAR009);
1652     if(TYPES[10] == 2 || TYPES[10] == 3) tmp[10] = getValue(VAR010);
1653     if(TYPES[11] == 2 || TYPES[11] == 3) tmp[11] = getValue(VAR011);
1654     if(TYPES[12] == 2 || TYPES[12] == 3) tmp[12] = getValue(VAR012);
1655     if(TYPES[13] == 2 || TYPES[13] == 3) tmp[13] = getValue(VAR013);
1656     if(TYPES[14] == 2 || TYPES[14] == 3) tmp[14] = getValue(VAR014);
1657     if(TYPES[15] == 2 || TYPES[15] == 3) tmp[15] = getValue(VAR015);
1658     if(TYPES[16] == 2 || TYPES[16] == 3) tmp[16] = getValue(VAR016);
1659     if(TYPES[17] == 2 || TYPES[17] == 3) tmp[17] = getValue(VAR017);
1660     if(TYPES[18] == 2 || TYPES[18] == 3) tmp[18] = getValue(VAR018);
1661     if(TYPES[19] == 2 || TYPES[19] == 3) tmp[19] = getValue(VAR019);
1662
```

```
    if(TYPES[20] == 2 || TYPES[20] == 3) tmp[20] = getValue(VAR020);
1663 if(TYPES[21] == 2 || TYPES[21] == 3) tmp[21] = getValue(VAR021);
1664 if(TYPES[22] == 2 || TYPES[22] == 3) tmp[22] = getValue(VAR022);
1665 if(TYPES[23] == 2 || TYPES[23] == 3) tmp[23] = getValue(VAR023);
1666 if(TYPES[24] == 2 || TYPES[24] == 3) tmp[24] = getValue(VAR024);
1667 if(TYPES[25] == 2 || TYPES[25] == 3) tmp[25] = getValue(VAR025);
1668 if(TYPES[26] == 2 || TYPES[26] == 3) tmp[26] = getValue(VAR026);
1669 if(TYPES[27] == 2 || TYPES[27] == 3) tmp[27] = getValue(VAR027);
1670 if(TYPES[28] == 2 || TYPES[28] == 3) tmp[28] = getValue(VAR028);
1671 if(TYPES[29] == 2 || TYPES[29] == 3) tmp[29] = getValue(VAR029);
1672 if(TYPES[30] == 2 || TYPES[30] == 3) tmp[30] = getValue(VAR030);
1673 if(TYPES[31] == 2 || TYPES[31] == 3) tmp[31] = getValue(VAR031);
1674 if(TYPES[32] == 2 || TYPES[32] == 3) tmp[32] = getValue(VAR032);
1675 if(TYPES[33] == 2 || TYPES[33] == 3) tmp[33] = getValue(VAR033);
1676 if(TYPES[34] == 2 || TYPES[34] == 3) tmp[34] = getValue(VAR034);
1677 if(TYPES[35] == 2 || TYPES[35] == 3) tmp[35] = getValue(VAR035);
1678 if(TYPES[36] == 2 || TYPES[36] == 3) tmp[36] = getValue(VAR036);
1679 if(TYPES[37] == 2 || TYPES[37] == 3) tmp[37] = getValue(VAR037);
1680 if(TYPES[38] == 2 || TYPES[38] == 3) tmp[38] = getValue(VAR038);
1681 if(TYPES[39] == 2 || TYPES[39] == 3) tmp[39] = getValue(VAR039);
1682 if(TYPES[40] == 2 || TYPES[40] == 3) tmp[40] = getValue(VAR040);
1683 if(TYPES[41] == 2 || TYPES[41] == 3) tmp[41] = getValue(VAR041);
1684 if(TYPES[42] == 2 || TYPES[42] == 3) tmp[42] = getValue(VAR042);
1685 if(TYPES[43] == 2 || TYPES[43] == 3) tmp[43] = getValue(VAR043);
1686 if(TYPES[44] == 2 || TYPES[44] == 3) tmp[44] = getValue(VAR044);
1687 if(TYPES[45] == 2 || TYPES[45] == 3) tmp[45] = getValue(VAR045);
1688 if(TYPES[46] == 2 || TYPES[46] == 3) tmp[46] = getValue(VAR046);
1689 if(TYPES[47] == 2 || TYPES[47] == 3) tmp[47] = getValue(VAR047);
1690 if(TYPES[48] == 2 || TYPES[48] == 3) tmp[48] = getValue(VAR048);
1691 if(TYPES[49] == 2 || TYPES[49] == 3) tmp[49] = getValue(VAR049);
1692 if(TYPES[50] == 2 || TYPES[50] == 3) tmp[50] = getValue(VAR050);
1693 if(TYPES[51] == 2 || TYPES[51] == 3) tmp[51] = getValue(VAR051);
1694 if(TYPES[52] == 2 || TYPES[52] == 3) tmp[52] = getValue(VAR052);
1695 if(TYPES[53] == 2 || TYPES[53] == 3) tmp[53] = getValue(VAR053);
1696 if(TYPES[54] == 2 || TYPES[54] == 3) tmp[54] = getValue(VAR054);
1697 if(TYPES[55] == 2 || TYPES[55] == 3) tmp[55] = getValue(VAR055);
1698 if(TYPES[56] == 2 || TYPES[56] == 3) tmp[56] = getValue(VAR056);
1699 if(TYPES[57] == 2 || TYPES[57] == 3) tmp[57] = getValue(VAR057);
1700 if(TYPES[58] == 2 || TYPES[58] == 3) tmp[58] = getValue(VAR058);
1701 if(TYPES[59] == 2 || TYPES[59] == 3) tmp[59] = getValue(VAR059);
1702 if(TYPES[60] == 2 || TYPES[60] == 3) tmp[60] = getValue(VAR060);
1703 if(TYPES[61] == 2 || TYPES[61] == 3) tmp[61] = getValue(VAR061);
1704 if(TYPES[62] == 2 || TYPES[62] == 3) tmp[62] = getValue(VAR062);
1705 if(TYPES[63] == 2 || TYPES[63] == 3) tmp[63] = getValue(VAR063);
1706 if(TYPES[64] == 2 || TYPES[64] == 3) tmp[64] = getValue(VAR064);
1707 if(TYPES[65] == 2 || TYPES[65] == 3) tmp[65] = getValue(VAR065);
1708 if(TYPES[66] == 2 || TYPES[66] == 3) tmp[66] = getValue(VAR066);
1709 if(TYPES[67] == 2 || TYPES[67] == 3) tmp[67] = getValue(VAR067);
1710 if(TYPES[68] == 2 || TYPES[68] == 3) tmp[68] = getValue(VAR068);
1711 if(TYPES[69] == 2 || TYPES[69] == 3) tmp[69] = getValue(VAR069);
1712 if(TYPES[70] == 2 || TYPES[70] == 3) tmp[70] = getValue(VAR070);
1713 if(TYPES[71] == 2 || TYPES[71] == 3) tmp[71] = getValue(VAR071);
1714 if(TYPES[72] == 2 || TYPES[72] == 3) tmp[72] = getValue(VAR072);
1715 if(TYPES[73] == 2 || TYPES[73] == 3) tmp[73] = getValue(VAR073);
1716 if(TYPES[74] == 2 || TYPES[74] == 3) tmp[74] = getValue(VAR074);
1717 if(TYPES[75] == 2 || TYPES[75] == 3) tmp[75] = getValue(VAR075);
1718 if(TYPES[76] == 2 || TYPES[76] == 3) tmp[76] = getValue(VAR076);
1719
```

```
    if(TYPES[77] == 2 || TYPES[77] == 3) tmp[77] = getValue(VAR077);
1720 if(TYPES[78] == 2 || TYPES[78] == 3) tmp[78] = getValue(VAR078);
1721 if(TYPES[79] == 2 || TYPES[79] == 3) tmp[79] = getValue(VAR079);
1722 if(TYPES[80] == 2 || TYPES[80] == 3) tmp[80] = getValue(VAR080);
1723 if(TYPES[81] == 2 || TYPES[81] == 3) tmp[81] = getValue(VAR081);
1724 if(TYPES[82] == 2 || TYPES[82] == 3) tmp[82] = getValue(VAR082);
1725 if(TYPES[83] == 2 || TYPES[83] == 3) tmp[83] = getValue(VAR083);
1726 if(TYPES[84] == 2 || TYPES[84] == 3) tmp[84] = getValue(VAR084);
1727 if(TYPES[85] == 2 || TYPES[85] == 3) tmp[85] = getValue(VAR085);
1728 if(TYPES[86] == 2 || TYPES[86] == 3) tmp[86] = getValue(VAR086);
1729 if(TYPES[87] == 2 || TYPES[87] == 3) tmp[87] = getValue(VAR087);
1730 if(TYPES[88] == 2 || TYPES[88] == 3) tmp[88] = getValue(VAR088);
1731 if(TYPES[89] == 2 || TYPES[89] == 3) tmp[89] = getValue(VAR089);
1732 if(TYPES[90] == 2 || TYPES[90] == 3) tmp[90] = getValue(VAR090);
1733 if(TYPES[91] == 2 || TYPES[91] == 3) tmp[91] = getValue(VAR091);
1734 if(TYPES[92] == 2 || TYPES[92] == 3) tmp[92] = getValue(VAR092);
1735 if(TYPES[93] == 2 || TYPES[93] == 3) tmp[93] = getValue(VAR093);
1736 if(TYPES[94] == 2 || TYPES[94] == 3) tmp[94] = getValue(VAR094);
1737 if(TYPES[95] == 2 || TYPES[95] == 3) tmp[95] = getValue(VAR095);
1738 if(TYPES[96] == 2 || TYPES[96] == 3) tmp[96] = getValue(VAR096);
1739 if(TYPES[97] == 2 || TYPES[97] == 3) tmp[97] = getValue(VAR097);
1740 if(TYPES[98] == 2 || TYPES[98] == 3) tmp[98] = getValue(VAR098);
1741 if(TYPES[99] == 2 || TYPES[99] == 3) tmp[99] = getValue(VAR099);
1742 if(TYPES[100] == 2 || TYPES[100] == 3) tmp[100] = getValue(VAR100);
1743 if(TYPES[101] == 2 || TYPES[101] == 3) tmp[101] = getValue(VAR101);
1744 if(TYPES[102] == 2 || TYPES[102] == 3) tmp[102] = getValue(VAR102);
1745 if(TYPES[103] == 2 || TYPES[103] == 3) tmp[103] = getValue(VAR103);
1746 if(TYPES[104] == 2 || TYPES[104] == 3) tmp[104] = getValue(VAR104);
1747 if(TYPES[105] == 2 || TYPES[105] == 3) tmp[105] = getValue(VAR105);
1748 if(TYPES[106] == 2 || TYPES[106] == 3) tmp[106] = getValue(VAR106);
1749 if(TYPES[107] == 2 || TYPES[107] == 3) tmp[107] = getValue(VAR107);
1750 if(TYPES[108] == 2 || TYPES[108] == 3) tmp[108] = getValue(VAR108);
1751 if(TYPES[109] == 2 || TYPES[109] == 3) tmp[109] = getValue(VAR109);
1752 if(TYPES[110] == 2 || TYPES[110] == 3) tmp[110] = getValue(VAR110);
1753 if(TYPES[111] == 2 || TYPES[111] == 3) tmp[111] = getValue(VAR111);
1754 if(TYPES[112] == 2 || TYPES[112] == 3) tmp[112] = getValue(VAR112);
1755 if(TYPES[113] == 2 || TYPES[113] == 3) tmp[113] = getValue(VAR113);
1756 if(TYPES[114] == 2 || TYPES[114] == 3) tmp[114] = getValue(VAR114);
1757 if(TYPES[115] == 2 || TYPES[115] == 3) tmp[115] = getValue(VAR115);
1758 if(TYPES[116] == 2 || TYPES[116] == 3) tmp[116] = getValue(VAR116);
1759 if(TYPES[117] == 2 || TYPES[117] == 3) tmp[117] = getValue(VAR117);
1760 if(TYPES[118] == 2 || TYPES[118] == 3) tmp[118] = getValue(VAR118);
1761 if(TYPES[119] == 2 || TYPES[119] == 3) tmp[119] = getValue(VAR119);
1762 if(TYPES[120] == 2 || TYPES[120] == 3) tmp[120] = getValue(VAR120);
1763 if(TYPES[121] == 2 || TYPES[121] == 3) tmp[121] = getValue(VAR121);
1764 if(TYPES[122] == 2 || TYPES[122] == 3) tmp[122] = getValue(VAR122);
1765 if(TYPES[123] == 2 || TYPES[123] == 3) tmp[123] = getValue(VAR123);
1766 if(TYPES[124] == 2 || TYPES[124] == 3) tmp[124] = getValue(VAR124);
1767 if(TYPES[125] == 2 || TYPES[125] == 3) tmp[125] = getValue(VAR125);
1768 if(TYPES[126] == 2 || TYPES[126] == 3) tmp[126] = getValue(VAR126);
1769 if(TYPES[127] == 2 || TYPES[127] == 3) tmp[127] = getValue(VAR127);
1770 if(TYPES[128] == 2 || TYPES[128] == 3) tmp[128] = getValue(VAR128);
1771 if(TYPES[129] == 2 || TYPES[129] == 3) tmp[129] = getValue(VAR129);
1772 if(TYPES[130] == 2 || TYPES[130] == 3) tmp[130] = getValue(VAR130);
1773 if(TYPES[131] == 2 || TYPES[131] == 3) tmp[131] = getValue(VAR131);
1774 if(TYPES[132] == 2 || TYPES[132] == 3) tmp[132] = getValue(VAR132);
1775 if(TYPES[133] == 2 || TYPES[133] == 3) tmp[133] = getValue(VAR133);
1776
```

```
    if(TYPES[134] == 2 || TYPES[134] == 3) tmp[134] = getValue(VAR134);
1777 if(TYPES[135] == 2 || TYPES[135] == 3) tmp[135] = getValue(VAR135);
1778 if(TYPES[136] == 2 || TYPES[136] == 3) tmp[136] = getValue(VAR136);
1779 if(TYPES[137] == 2 || TYPES[137] == 3) tmp[137] = getValue(VAR137);
1780 if(TYPES[138] == 2 || TYPES[138] == 3) tmp[138] = getValue(VAR138);
1781 if(TYPES[139] == 2 || TYPES[139] == 3) tmp[139] = getValue(VAR139);
1782 if(TYPES[140] == 2 || TYPES[140] == 3) tmp[140] = getValue(VAR140);
1783 if(TYPES[141] == 2 || TYPES[141] == 3) tmp[141] = getValue(VAR141);
1784 if(TYPES[142] == 2 || TYPES[142] == 3) tmp[142] = getValue(VAR142);
1785 if(TYPES[143] == 2 || TYPES[143] == 3) tmp[143] = getValue(VAR143);
1786 if(TYPES[144] == 2 || TYPES[144] == 3) tmp[144] = getValue(VAR144);
1787 if(TYPES[145] == 2 || TYPES[145] == 3) tmp[145] = getValue(VAR145);
1788 if(TYPES[146] == 2 || TYPES[146] == 3) tmp[146] = getValue(VAR146);
1789 if(TYPES[147] == 2 || TYPES[147] == 3) tmp[147] = getValue(VAR147);
1790 if(TYPES[148] == 2 || TYPES[148] == 3) tmp[148] = getValue(VAR148);
1791 if(TYPES[149] == 2 || TYPES[149] == 3) tmp[149] = getValue(VAR149);
1792 if(TYPES[150] == 2 || TYPES[150] == 3) tmp[150] = getValue(VAR150);
1793 if(TYPES[151] == 2 || TYPES[151] == 3) tmp[151] = getValue(VAR151);
1794 if(TYPES[152] == 2 || TYPES[152] == 3) tmp[152] = getValue(VAR152);
1795 if(TYPES[153] == 2 || TYPES[153] == 3) tmp[153] = getValue(VAR153);
1796 if(TYPES[154] == 2 || TYPES[154] == 3) tmp[154] = getValue(VAR154);
1797 if(TYPES[155] == 2 || TYPES[155] == 3) tmp[155] = getValue(VAR155);
1798 if(TYPES[156] == 2 || TYPES[156] == 3) tmp[156] = getValue(VAR156);
1799 if(TYPES[157] == 2 || TYPES[157] == 3) tmp[157] = getValue(VAR157);
1800 if(TYPES[158] == 2 || TYPES[158] == 3) tmp[158] = getValue(VAR158);
1801 if(TYPES[159] == 2 || TYPES[159] == 3) tmp[159] = getValue(VAR159);
1802 if(TYPES[160] == 2 || TYPES[160] == 3) tmp[160] = getValue(VAR160);
1803 if(TYPES[161] == 2 || TYPES[161] == 3) tmp[161] = getValue(VAR161);
1804 if(TYPES[162] == 2 || TYPES[162] == 3) tmp[162] = getValue(VAR162);
1805 if(TYPES[163] == 2 || TYPES[163] == 3) tmp[163] = getValue(VAR163);
1806 if(TYPES[164] == 2 || TYPES[164] == 3) tmp[164] = getValue(VAR164);
1807 if(TYPES[165] == 2 || TYPES[165] == 3) tmp[165] = getValue(VAR165);
1808 if(TYPES[166] == 2 || TYPES[166] == 3) tmp[166] = getValue(VAR166);
1809 if(TYPES[167] == 2 || TYPES[167] == 3) tmp[167] = getValue(VAR167);
1810 if(TYPES[168] == 2 || TYPES[168] == 3) tmp[168] = getValue(VAR168);
1811 if(TYPES[169] == 2 || TYPES[169] == 3) tmp[169] = getValue(VAR169);
1812 if(TYPES[170] == 2 || TYPES[170] == 3) tmp[170] = getValue(VAR170);
1813 if(TYPES[171] == 2 || TYPES[171] == 3) tmp[171] = getValue(VAR171);
1814 if(TYPES[172] == 2 || TYPES[172] == 3) tmp[172] = getValue(VAR172);
1815 if(TYPES[173] == 2 || TYPES[173] == 3) tmp[173] = getValue(VAR173);
1816 if(TYPES[174] == 2 || TYPES[174] == 3) tmp[174] = getValue(VAR174);
1817 if(TYPES[175] == 2 || TYPES[175] == 3) tmp[175] = getValue(VAR175);
1818 if(TYPES[176] == 2 || TYPES[176] == 3) tmp[176] = getValue(VAR176);
1819 if(TYPES[177] == 2 || TYPES[177] == 3) tmp[177] = getValue(VAR177);
1820 if(TYPES[178] == 2 || TYPES[178] == 3) tmp[178] = getValue(VAR178);
1821 if(TYPES[179] == 2 || TYPES[179] == 3) tmp[179] = getValue(VAR179);
1822 if(TYPES[180] == 2 || TYPES[180] == 3) tmp[180] = getValue(VAR180);
1823 if(TYPES[181] == 2 || TYPES[181] == 3) tmp[181] = getValue(VAR181);
1824 if(TYPES[182] == 2 || TYPES[182] == 3) tmp[182] = getValue(VAR182);
1825 if(TYPES[183] == 2 || TYPES[183] == 3) tmp[183] = getValue(VAR183);
1826 if(TYPES[184] == 2 || TYPES[184] == 3) tmp[184] = getValue(VAR184);
1827 if(TYPES[185] == 2 || TYPES[185] == 3) tmp[185] = getValue(VAR185);
1828 if(TYPES[186] == 2 || TYPES[186] == 3) tmp[186] = getValue(VAR186);
1829 if(TYPES[187] == 2 || TYPES[187] == 3) tmp[187] = getValue(VAR187);
1830 if(TYPES[188] == 2 || TYPES[188] == 3) tmp[188] = getValue(VAR188);
1831 if(TYPES[189] == 2 || TYPES[189] == 3) tmp[189] = getValue(VAR189);
1832 if(TYPES[190] == 2 || TYPES[190] == 3) tmp[190] = getValue(VAR190);
1833
```

```
    if(TYPES[191] == 2 || TYPES[191] == 3) tmp[191] = getValue(VAR191);
1834 if(TYPES[192] == 2 || TYPES[192] == 3) tmp[192] = getValue(VAR192);
1835 if(TYPES[193] == 2 || TYPES[193] == 3) tmp[193] = getValue(VAR193);
1836 if(TYPES[194] == 2 || TYPES[194] == 3) tmp[194] = getValue(VAR194);
1837 if(TYPES[195] == 2 || TYPES[195] == 3) tmp[195] = getValue(VAR195);
1838 if(TYPES[196] == 2 || TYPES[196] == 3) tmp[196] = getValue(VAR196);
1839 if(TYPES[197] == 2 || TYPES[197] == 3) tmp[197] = getValue(VAR197);
1840 if(TYPES[198] == 2 || TYPES[198] == 3) tmp[198] = getValue(VAR198);
1841 if(TYPES[199] == 2 || TYPES[199] == 3) tmp[199] = getValue(VAR199);
1842 if(TYPES[200] == 2 || TYPES[200] == 3) tmp[200] = getValue(VAR200);
1843 if(TYPES[201] == 2 || TYPES[201] == 3) tmp[201] = getValue(VAR201);
1844 if(TYPES[202] == 2 || TYPES[202] == 3) tmp[202] = getValue(VAR202);
1845 if(TYPES[203] == 2 || TYPES[203] == 3) tmp[203] = getValue(VAR203);
1846 if(TYPES[204] == 2 || TYPES[204] == 3) tmp[204] = getValue(VAR204);
1847 if(TYPES[205] == 2 || TYPES[205] == 3) tmp[205] = getValue(VAR205);
1848 if(TYPES[206] == 2 || TYPES[206] == 3) tmp[206] = getValue(VAR206);
1849 if(TYPES[207] == 2 || TYPES[207] == 3) tmp[207] = getValue(VAR207);
1850 if(TYPES[208] == 2 || TYPES[208] == 3) tmp[208] = getValue(VAR208);
1851 if(TYPES[209] == 2 || TYPES[209] == 3) tmp[209] = getValue(VAR209);
1852 if(TYPES[210] == 2 || TYPES[210] == 3) tmp[210] = getValue(VAR210);
1853 if(TYPES[211] == 2 || TYPES[211] == 3) tmp[211] = getValue(VAR211);
1854 if(TYPES[212] == 2 || TYPES[212] == 3) tmp[212] = getValue(VAR212);
1855 if(TYPES[213] == 2 || TYPES[213] == 3) tmp[213] = getValue(VAR213);
1856 if(TYPES[214] == 2 || TYPES[214] == 3) tmp[214] = getValue(VAR214);
1857 if(TYPES[215] == 2 || TYPES[215] == 3) tmp[215] = getValue(VAR215);
1858 if(TYPES[216] == 2 || TYPES[216] == 3) tmp[216] = getValue(VAR216);
1859 if(TYPES[217] == 2 || TYPES[217] == 3) tmp[217] = getValue(VAR217);
1860 if(TYPES[218] == 2 || TYPES[218] == 3) tmp[218] = getValue(VAR218);
1861 if(TYPES[219] == 2 || TYPES[219] == 3) tmp[219] = getValue(VAR219);
1862 if(TYPES[220] == 2 || TYPES[220] == 3) tmp[220] = getValue(VAR220);
1863 if(TYPES[221] == 2 || TYPES[221] == 3) tmp[221] = getValue(VAR221);
1864 if(TYPES[222] == 2 || TYPES[222] == 3) tmp[222] = getValue(VAR222);
1865 if(TYPES[223] == 2 || TYPES[223] == 3) tmp[223] = getValue(VAR223);
1866 if(TYPES[224] == 2 || TYPES[224] == 3) tmp[224] = getValue(VAR224);
1867 if(TYPES[225] == 2 || TYPES[225] == 3) tmp[225] = getValue(VAR225);
1868 if(TYPES[226] == 2 || TYPES[226] == 3) tmp[226] = getValue(VAR226);
1869 if(TYPES[227] == 2 || TYPES[227] == 3) tmp[227] = getValue(VAR227);
1870 if(TYPES[228] == 2 || TYPES[228] == 3) tmp[228] = getValue(VAR228);
1871 if(TYPES[229] == 2 || TYPES[229] == 3) tmp[229] = getValue(VAR229);
1872 if(TYPES[230] == 2 || TYPES[230] == 3) tmp[230] = getValue(VAR230);
1873 if(TYPES[231] == 2 || TYPES[231] == 3) tmp[231] = getValue(VAR231);
1874 if(TYPES[232] == 2 || TYPES[232] == 3) tmp[232] = getValue(VAR232);
1875 if(TYPES[233] == 2 || TYPES[233] == 3) tmp[233] = getValue(VAR233);
1876 if(TYPES[234] == 2 || TYPES[234] == 3) tmp[234] = getValue(VAR234);
1877 if(TYPES[235] == 2 || TYPES[235] == 3) tmp[235] = getValue(VAR235);
1878 if(TYPES[236] == 2 || TYPES[236] == 3) tmp[236] = getValue(VAR236);
1879 if(TYPES[237] == 2 || TYPES[237] == 3) tmp[237] = getValue(VAR237);
1880 if(TYPES[238] == 2 || TYPES[238] == 3) tmp[238] = getValue(VAR238);
1881 if(TYPES[239] == 2 || TYPES[239] == 3) tmp[239] = getValue(VAR239);
1882 if(TYPES[240] == 2 || TYPES[240] == 3) tmp[240] = getValue(VAR240);
1883 if(TYPES[241] == 2 || TYPES[241] == 3) tmp[241] = getValue(VAR241);
1884 if(TYPES[242] == 2 || TYPES[242] == 3) tmp[242] = getValue(VAR242);
1885 if(TYPES[243] == 2 || TYPES[243] == 3) tmp[243] = getValue(VAR243);
1886 if(TYPES[244] == 2 || TYPES[244] == 3) tmp[244] = getValue(VAR244);
1887 if(TYPES[245] == 2 || TYPES[245] == 3) tmp[245] = getValue(VAR245);
1888 if(TYPES[246] == 2 || TYPES[246] == 3) tmp[246] = getValue(VAR246);
1889 if(TYPES[247] == 2 || TYPES[247] == 3) tmp[247] = getValue(VAR247);
1890
```

```
    if(TYPES[248] == 2 || TYPES[248] == 3) tmp[248] = getValue(VAR248);
1891    if(TYPES[249] == 2 || TYPES[249] == 3) tmp[249] = getValue(VAR249);
1892    if(TYPES[250] == 2 || TYPES[250] == 3) tmp[250] = getValue(VAR250);
1893    if(TYPES[251] == 2 || TYPES[251] == 3) tmp[251] = getValue(VAR251);
1894    if(TYPES[252] == 2 || TYPES[252] == 3) tmp[252] = getValue(VAR252);
1895    if(TYPES[253] == 2 || TYPES[253] == 3) tmp[253] = getValue(VAR253);
1896    if(TYPES[254] == 2 || TYPES[254] == 3) tmp[254] = getValue(VAR254);
1897    if(TYPES[255] == 2 || TYPES[255] == 3) tmp[255] = getValue(VAR255);
1898    if(TYPES[256] == 2 || TYPES[256] == 3) tmp[256] = getValue(VAR256);
1899    if(TYPES[257] == 2 || TYPES[257] == 3) tmp[257] = getValue(VAR257);
1900    if(TYPES[258] == 2 || TYPES[258] == 3) tmp[258] = getValue(VAR258);
1901    if(TYPES[259] == 2 || TYPES[259] == 3) tmp[259] = getValue(VAR259);
1902    if(TYPES[260] == 2 || TYPES[260] == 3) tmp[260] = getValue(VAR260);
1903    if(TYPES[261] == 2 || TYPES[261] == 3) tmp[261] = getValue(VAR261);
1904    if(TYPES[262] == 2 || TYPES[262] == 3) tmp[262] = getValue(VAR262);
1905    if(TYPES[263] == 2 || TYPES[263] == 3) tmp[263] = getValue(VAR263);
1906    if(TYPES[264] == 2 || TYPES[264] == 3) tmp[264] = getValue(VAR264);
1907    if(TYPES[265] == 2 || TYPES[265] == 3) tmp[265] = getValue(VAR265);
1908    if(TYPES[266] == 2 || TYPES[266] == 3) tmp[266] = getValue(VAR266);
1909    if(TYPES[267] == 2 || TYPES[267] == 3) tmp[267] = getValue(VAR267);
1910    if(TYPES[268] == 2 || TYPES[268] == 3) tmp[268] = getValue(VAR268);
1911    if(TYPES[269] == 2 || TYPES[269] == 3) tmp[269] = getValue(VAR269);
1912    if(TYPES[270] == 2 || TYPES[270] == 3) tmp[270] = getValue(VAR270);
1913    if(TYPES[271] == 2 || TYPES[271] == 3) tmp[271] = getValue(VAR271);
1914    if(TYPES[272] == 2 || TYPES[272] == 3) tmp[272] = getValue(VAR272);
1915    if(TYPES[273] == 2 || TYPES[273] == 3) tmp[273] = getValue(VAR273);
1916    if(TYPES[274] == 2 || TYPES[274] == 3) tmp[274] = getValue(VAR274);
1917    if(TYPES[275] == 2 || TYPES[275] == 3) tmp[275] = getValue(VAR275);
1918    if(TYPES[276] == 2 || TYPES[276] == 3) tmp[276] = getValue(VAR276);
1919    if(TYPES[277] == 2 || TYPES[277] == 3) tmp[277] = getValue(VAR277);
1920    if(TYPES[278] == 2 || TYPES[278] == 3) tmp[278] = getValue(VAR278);
1921    if(TYPES[279] == 2 || TYPES[279] == 3) tmp[279] = getValue(VAR279);
1922    if(TYPES[280] == 2 || TYPES[280] == 3) tmp[280] = getValue(VAR280);
1923    if(TYPES[281] == 2 || TYPES[281] == 3) tmp[281] = getValue(VAR281);
1924    if(TYPES[282] == 2 || TYPES[282] == 3) tmp[282] = getValue(VAR282);
1925    if(TYPES[283] == 2 || TYPES[283] == 3) tmp[283] = getValue(VAR283);
1926    if(TYPES[284] == 2 || TYPES[284] == 3) tmp[284] = getValue(VAR284);
1927    if(TYPES[285] == 2 || TYPES[285] == 3) tmp[285] = getValue(VAR285);
1928    if(TYPES[286] == 2 || TYPES[286] == 3) tmp[286] = getValue(VAR286);
1929    if(TYPES[287] == 2 || TYPES[287] == 3) tmp[287] = getValue(VAR287);
1930    if(TYPES[288] == 2 || TYPES[288] == 3) tmp[288] = getValue(VAR288);
1931    if(TYPES[289] == 2 || TYPES[289] == 3) tmp[289] = getValue(VAR289);
1932    if(TYPES[290] == 2 || TYPES[290] == 3) tmp[290] = getValue(VAR290);
1933    if(TYPES[291] == 2 || TYPES[291] == 3) tmp[291] = getValue(VAR291);
1934    if(TYPES[292] == 2 || TYPES[292] == 3) tmp[292] = getValue(VAR292);
1935    if(TYPES[293] == 2 || TYPES[293] == 3) tmp[293] = getValue(VAR293);
1936    if(TYPES[294] == 2 || TYPES[294] == 3) tmp[294] = getValue(VAR294);
1937    if(TYPES[295] == 2 || TYPES[295] == 3) tmp[295] = getValue(VAR295);
1938    if(TYPES[296] == 2 || TYPES[296] == 3) tmp[296] = getValue(VAR296);
1939    if(TYPES[297] == 2 || TYPES[297] == 3) tmp[297] = getValue(VAR297);
1940    if(TYPES[298] == 2 || TYPES[298] == 3) tmp[298] = getValue(VAR298);
1941    if(TYPES[299] == 2 || TYPES[299] == 3) tmp[299] = getValue(VAR299);
1942}
1943
1944CopyNameToDataBase ()
1945{
1946    putValue(STR000, NAMES[0]);
1947
```

```
    putValue(STR001, NAMES[1]);
1948 putValue(STR002, NAMES[2]);
1949 putValue(STR003, NAMES[3]);
1950 putValue(STR004, NAMES[4]);
1951 putValue(STR005, NAMES[5]);
1952 putValue(STR006, NAMES[6]);
1953 putValue(STR007, NAMES[7]);
1954 putValue(STR008, NAMES[8]);
1955 putValue(STR009, NAMES[9]);
1956 putValue(STR010, NAMES[10]);
1957 putValue(STR011, NAMES[11]);
1958 putValue(STR012, NAMES[12]);
1959 putValue(STR013, NAMES[13]);
1960 putValue(STR014, NAMES[14]);
1961 putValue(STR015, NAMES[15]);
1962 putValue(STR016, NAMES[16]);
1963 putValue(STR017, NAMES[17]);
1964 putValue(STR018, NAMES[18]);
1965 putValue(STR019, NAMES[19]);
1966 putValue(STR020, NAMES[20]);
1967 putValue(STR021, NAMES[21]);
1968 putValue(STR022, NAMES[22]);
1969 putValue(STR023, NAMES[23]);
1970 putValue(STR024, NAMES[24]);
1971 putValue(STR025, NAMES[25]);
1972 putValue(STR026, NAMES[26]);
1973 putValue(STR027, NAMES[27]);
1974 putValue(STR028, NAMES[28]);
1975 putValue(STR029, NAMES[29]);
1976 putValue(STR030, NAMES[30]);
1977 putValue(STR031, NAMES[31]);
1978 putValue(STR032, NAMES[32]);
1979 putValue(STR033, NAMES[33]);
1980 putValue(STR034, NAMES[34]);
1981 putValue(STR035, NAMES[35]);
1982 putValue(STR036, NAMES[36]);
1983 putValue(STR037, NAMES[37]);
1984 putValue(STR038, NAMES[38]);
1985 putValue(STR039, NAMES[39]);
1986 putValue(STR040, NAMES[40]);
1987 putValue(STR041, NAMES[41]);
1988 putValue(STR042, NAMES[42]);
1989 putValue(STR043, NAMES[43]);
1990 putValue(STR044, NAMES[44]);
1991 putValue(STR045, NAMES[45]);
1992 putValue(STR046, NAMES[46]);
1993 putValue(STR047, NAMES[47]);
1994 putValue(STR048, NAMES[48]);
1995 putValue(STR049, NAMES[49]);
1996 putValue(STR050, NAMES[50]);
1997 putValue(STR051, NAMES[51]);
1998 putValue(STR052, NAMES[52]);
1999 putValue(STR053, NAMES[53]);
2000 putValue(STR054, NAMES[54]);
2001 putValue(STR055, NAMES[55]);
2002 putValue(STR056, NAMES[56]);
2003 putValue(STR057, NAMES[57]);
2004
```

```
    putValue(STR058, NAMES[58]);
2005 putValue(STR059, NAMES[59]);
2006 putValue(STR060, NAMES[60]);
2007 putValue(STR061, NAMES[61]);
2008 putValue(STR062, NAMES[62]);
2009 putValue(STR063, NAMES[63]);
2010 putValue(STR064, NAMES[64]);
2011 putValue(STR065, NAMES[65]);
2012 putValue(STR066, NAMES[66]);
2013 putValue(STR067, NAMES[67]);
2014 putValue(STR068, NAMES[68]);
2015 putValue(STR069, NAMES[69]);
2016 putValue(STR070, NAMES[70]);
2017 putValue(STR071, NAMES[71]);
2018 putValue(STR072, NAMES[72]);
2019 putValue(STR073, NAMES[73]);
2020 putValue(STR074, NAMES[74]);
2021 putValue(STR075, NAMES[75]);
2022 putValue(STR076, NAMES[76]);
2023 putValue(STR077, NAMES[77]);
2024 putValue(STR078, NAMES[78]);
2025 putValue(STR079, NAMES[79]);
2026 putValue(STR080, NAMES[80]);
2027 putValue(STR081, NAMES[81]);
2028 putValue(STR082, NAMES[82]);
2029 putValue(STR083, NAMES[83]);
2030 putValue(STR084, NAMES[84]);
2031 putValue(STR085, NAMES[85]);
2032 putValue(STR086, NAMES[86]);
2033 putValue(STR087, NAMES[87]);
2034 putValue(STR088, NAMES[88]);
2035 putValue(STR089, NAMES[89]);
2036 putValue(STR090, NAMES[90]);
2037 putValue(STR091, NAMES[91]);
2038 putValue(STR092, NAMES[92]);
2039 putValue(STR093, NAMES[93]);
2040 putValue(STR094, NAMES[94]);
2041 putValue(STR095, NAMES[95]);
2042 putValue(STR096, NAMES[96]);
2043 putValue(STR097, NAMES[97]);
2044 putValue(STR098, NAMES[98]);
2045 putValue(STR099, NAMES[99]);
2046 putValue(STR100, NAMES[100]);
2047 putValue(STR101, NAMES[101]);
2048 putValue(STR102, NAMES[102]);
2049 putValue(STR103, NAMES[103]);
2050 putValue(STR104, NAMES[104]);
2051 putValue(STR105, NAMES[105]);
2052 putValue(STR106, NAMES[106]);
2053 putValue(STR107, NAMES[107]);
2054 putValue(STR108, NAMES[108]);
2055 putValue(STR109, NAMES[109]);
2056 putValue(STR110, NAMES[110]);
2057 putValue(STR111, NAMES[111]);
2058 putValue(STR112, NAMES[112]);
2059 putValue(STR113, NAMES[113]);
2060 putValue(STR114, NAMES[114]);
2061
```

```
    putValue(STR115, NAMES[115]);
    putValue(STR116, NAMES[116]);
    putValue(STR117, NAMES[117]);
    putValue(STR118, NAMES[118]);
    putValue(STR119, NAMES[119]);
    putValue(STR120, NAMES[120]);
    putValue(STR121, NAMES[121]);
    putValue(STR122, NAMES[122]);
    putValue(STR123, NAMES[123]);
    putValue(STR124, NAMES[124]);
    putValue(STR125, NAMES[125]);
    putValue(STR126, NAMES[126]);
    putValue(STR127, NAMES[127]);
    putValue(STR128, NAMES[128]);
    putValue(STR129, NAMES[129]);
    putValue(STR130, NAMES[130]);
    putValue(STR131, NAMES[131]);
    putValue(STR132, NAMES[132]);
    putValue(STR133, NAMES[133]);
    putValue(STR134, NAMES[134]);
    putValue(STR135, NAMES[135]);
    putValue(STR136, NAMES[136]);
    putValue(STR137, NAMES[137]);
    putValue(STR138, NAMES[138]);
    putValue(STR139, NAMES[139]);
    putValue(STR140, NAMES[140]);
    putValue(STR141, NAMES[141]);
    putValue(STR142, NAMES[142]);
    putValue(STR143, NAMES[143]);
    putValue(STR144, NAMES[144]);
    putValue(STR145, NAMES[145]);
    putValue(STR146, NAMES[146]);
    putValue(STR147, NAMES[147]);
    putValue(STR148, NAMES[148]);
    putValue(STR149, NAMES[149]);
    putValue(STR150, NAMES[150]);
    putValue(STR151, NAMES[151]);
    putValue(STR152, NAMES[152]);
    putValue(STR153, NAMES[153]);
    putValue(STR154, NAMES[154]);
    putValue(STR155, NAMES[155]);
    putValue(STR156, NAMES[156]);
    putValue(STR157, NAMES[157]);
    putValue(STR158, NAMES[158]);
    putValue(STR159, NAMES[159]);
    putValue(STR160, NAMES[160]);
    putValue(STR161, NAMES[161]);
    putValue(STR162, NAMES[162]);
    putValue(STR163, NAMES[163]);
    putValue(STR164, NAMES[164]);
    putValue(STR165, NAMES[165]);
    putValue(STR166, NAMES[166]);
    putValue(STR167, NAMES[167]);
    putValue(STR168, NAMES[168]);
    putValue(STR169, NAMES[169]);
    putValue(STR170, NAMES[170]);
    putValue(STR171, NAMES[171]);

```

```
    putValue(STR172, NAMES[172]);
    putValue(STR173, NAMES[173]);
    putValue(STR174, NAMES[174]);
    putValue(STR175, NAMES[175]);
    putValue(STR176, NAMES[176]);
    putValue(STR177, NAMES[177]);
    putValue(STR178, NAMES[178]);
    putValue(STR179, NAMES[179]);
    putValue(STR180, NAMES[180]);
    putValue(STR181, NAMES[181]);
    putValue(STR182, NAMES[182]);
    putValue(STR183, NAMES[183]);
    putValue(STR184, NAMES[184]);
    putValue(STR185, NAMES[185]);
    putValue(STR186, NAMES[186]);
    putValue(STR187, NAMES[187]);
    putValue(STR188, NAMES[188]);
    putValue(STR189, NAMES[189]);
    putValue(STR190, NAMES[190]);
    putValue(STR191, NAMES[191]);
    putValue(STR192, NAMES[192]);
    putValue(STR193, NAMES[193]);
    putValue(STR194, NAMES[194]);
    putValue(STR195, NAMES[195]);
    putValue(STR196, NAMES[196]);
    putValue(STR197, NAMES[197]);
    putValue(STR198, NAMES[198]);
    putValue(STR199, NAMES[199]);
    putValue(STR200, NAMES[200]);
    putValue(STR201, NAMES[201]);
    putValue(STR202, NAMES[202]);
    putValue(STR203, NAMES[203]);
    putValue(STR204, NAMES[204]);
    putValue(STR205, NAMES[205]);
    putValue(STR206, NAMES[206]);
    putValue(STR207, NAMES[207]);
    putValue(STR208, NAMES[208]);
    putValue(STR209, NAMES[209]);
    putValue(STR210, NAMES[210]);
    putValue(STR211, NAMES[211]);
    putValue(STR212, NAMES[212]);
    putValue(STR213, NAMES[213]);
    putValue(STR214, NAMES[214]);
    putValue(STR215, NAMES[215]);
    putValue(STR216, NAMES[216]);
    putValue(STR217, NAMES[217]);
    putValue(STR218, NAMES[218]);
    putValue(STR219, NAMES[219]);
    putValue(STR220, NAMES[220]);
    putValue(STR221, NAMES[221]);
    putValue(STR222, NAMES[222]);
    putValue(STR223, NAMES[223]);
    putValue(STR224, NAMES[224]);
    putValue(STR225, NAMES[225]);
    putValue(STR226, NAMES[226]);
    putValue(STR227, NAMES[227]);
    putValue(STR228, NAMES[228]);

```

```
    putValue(STR229, NAMES[229]);
    putValue(STR230, NAMES[230]);
    putValue(STR231, NAMES[231]);
    putValue(STR232, NAMES[232]);
    putValue(STR233, NAMES[233]);
    putValue(STR234, NAMES[234]);
    putValue(STR235, NAMES[235]);
    putValue(STR236, NAMES[236]);
    putValue(STR237, NAMES[237]);
    putValue(STR238, NAMES[238]);
    putValue(STR239, NAMES[239]);
    putValue(STR240, NAMES[240]);
    putValue(STR241, NAMES[241]);
    putValue(STR242, NAMES[242]);
    putValue(STR243, NAMES[243]);
    putValue(STR244, NAMES[244]);
    putValue(STR245, NAMES[245]);
    putValue(STR246, NAMES[246]);
    putValue(STR247, NAMES[247]);
    putValue(STR248, NAMES[248]);
    putValue(STR249, NAMES[249]);
    putValue(STR250, NAMES[250]);
    putValue(STR251, NAMES[251]);
    putValue(STR252, NAMES[252]);
    putValue(STR253, NAMES[253]);
    putValue(STR254, NAMES[254]);
    putValue(STR255, NAMES[255]);
    putValue(STR256, NAMES[256]);
    putValue(STR257, NAMES[257]);
    putValue(STR258, NAMES[258]);
    putValue(STR259, NAMES[259]);
    putValue(STR260, NAMES[260]);
    putValue(STR261, NAMES[261]);
    putValue(STR262, NAMES[262]);
    putValue(STR263, NAMES[263]);
    putValue(STR264, NAMES[264]);
    putValue(STR265, NAMES[265]);
    putValue(STR266, NAMES[266]);
    putValue(STR267, NAMES[267]);
    putValue(STR268, NAMES[268]);
    putValue(STR269, NAMES[269]);
    putValue(STR270, NAMES[270]);
    putValue(STR271, NAMES[271]);
    putValue(STR272, NAMES[272]);
    putValue(STR273, NAMES[273]);
    putValue(STR274, NAMES[274]);
    putValue(STR275, NAMES[275]);
    putValue(STR276, NAMES[276]);
    putValue(STR277, NAMES[277]);
    putValue(STR278, NAMES[278]);
    putValue(STR279, NAMES[279]);
    putValue(STR280, NAMES[280]);
    putValue(STR281, NAMES[281]);
    putValue(STR282, NAMES[282]);
    putValue(STR283, NAMES[283]);
    putValue(STR284, NAMES[284]);
    putValue(STR285, NAMES[285]);
```

```
    putValue(STR286, NAMES[286]);
    putValue(STR287, NAMES[287]);
    putValue(STR288, NAMES[288]);
    putValue(STR289, NAMES[289]);
    putValue(STR290, NAMES[290]);
    putValue(STR291, NAMES[291]);
    putValue(STR292, NAMES[292]);
    putValue(STR293, NAMES[293]);
    putValue(STR294, NAMES[294]);
    putValue(STR295, NAMES[295]);
    putValue(STR296, NAMES[296]);
    putValue(STR297, NAMES[297]);
    putValue(STR298, NAMES[298]);
    putValue(STR299, NAMES[299]);
}

InitializeValues ()
{
    long i;

    putValue(Threshold,5);
    putValue(CALIBRATION_MODE, 1);
    putValue(CYCLING_MODE, 0);


    //fileNameAT=("ImmunityCycle.xml");
    //strncpy(fileNameAT,"ImmunityCycle.xml%d",1,MAX_VAR_LENGTH);




    for(i=0; i<VARS_SIZE; ++i)
    {
        VALUES[i] = 0;
        tmp[i] = 0;
        arrayError[i][0]=0;
        arrayError[i][1]=0;
        NewStep[i]= AT_VALUES[i][0];


    }


}

WORD MsgSetMaskT5MY10 (WORD identifier)
{
  WORD i, dataLength, addedBytes;

    switch(identifier)
    {
        case 0xFD01:
        {
            dataLength = 9;
            addedBytes = 9;
            break;
        }

```

```
        case 0xFD04:
        {
            dataLength = 14;
            addedBytes = 2;
            break;
        }
        case 0xFD08:
        {
            dataLength = 3;
            addedBytes = 2;
            break;
        }
        case 0xFD09:
        {
            dataLength = 1;
            addedBytes = 1;
            break;
        }
    }

  for(i=0; i<addedBytes; ++i)
    gTxDataBuffer[4+dataLength+i] = 0xFF;

  return addedBytes;
}

/*on timer SendMessages
{
    // GET VARIANT
    if(hasToSendGetVariant == 1)
    {
        hasToSendGetVariant = 0;

        gTxDataBuffer[0]=0x22;
      gTxDataBuffer[1]=0xF1;
      gTxDataBuffer[2]=0xF5;
      txArraySize=3;
      OSEKTL_DataReq(gTxDataBuffer, txArraySize);
    }
    // GET SOFTWARE VERSION S12 LEAR
    else if(hasToSendGetSoftwareVersionS12Lear == 1)
    {
        hasToSendGetSoftwareVersionS12Lear = 0;
    gTxDataBuffer[0]=0x22;
      gTxDataBuffer[1]=0xFD;
      gTxDataBuffer[2]=0x98;
      txArraySize=3;
      OSEKTL_DataReq(gTxDataBuffer, txArraySize);
    }
    // GET SOFTWARE VERSION S12 JLR
    else if(hasToSendGetSoftwareVersionS12Jlr == 1)
    {
        hasToSendGetSoftwareVersionS12Jlr = 0;
    gTxDataBuffer[0]=0x22;
      gTxDataBuffer[1]=0xF1;
      gTxDataBuffer[2]=0x88;
      txArraySize=3;
```

```
  OSEKTL_DataReq(gTxDataBuffer, txArraySize);
}
// GET SOFTWARE VERSION ATMEL
else if(hasToSendGetSoftwareVersionAtmel == 1)
{
    hasToSendGetSoftwareVersionAtmel = 0;
gTxDataBuffer[0]=0x22;
  gTxDataBuffer[1]=0xF1;
  gTxDataBuffer[2]=0xF0;
  txArraySize=3;
  OSEKTL_DataReq(gTxDataBuffer, txArraySize);
}
// SET VARIANT
else if(hasToSendSetVariant == 1)
{
    hasToSendSetVariant = 0;
    gTxDataBuffer[0] = 0x2E;
  gTxDataBuffer[1] = 0xF1;
  gTxDataBuffer[2] = 0xF5;
    gTxDataBuffer[3] = getValue(VAR_Variant);

    txArraySize = 4;
  OSEKTL_DataReq(gTxDataBuffer, txArraySize);
}
// READ START LOGIC CONTROL
else if(hasToSendReadStartLogicControl == 1)
{
    hasToSendReadStartLogicControl = 0;
    gTxDataBuffer[0] = 0x22;
  gTxDataBuffer[1] = 0xFE;
  gTxDataBuffer[2] = 0xE7;

    txArraySize = 3;
  OSEKTL_DataReq(gTxDataBuffer, txArraySize);
}
// DISABLE PROTECTIONS
else if(hasToDisableProtections == 1)
{
    hasToDisableProtections = 0;
    gTxDataBuffer[0] = 0x2E;
  gTxDataBuffer[1] = 0xFD;
  gTxDataBuffer[2] = 0x81;
    gTxDataBuffer[3] = 0x00;
    gTxDataBuffer[4] = 0x00;
    gTxDataBuffer[5] = 0x00;
    gTxDataBuffer[6] = 0x00;
    gTxDataBuffer[7] = 0x00;
    gTxDataBuffer[8] = 0x00;
    gTxDataBuffer[9] = 0x00;
    gTxDataBuffer[10] = 0x00;
    gTxDataBuffer[11] = 0x00;

    txArraySize = 12;
  OSEKTL_DataReq(gTxDataBuffer, txArraySize);
}
// ENABLE PROTECTIONS
else if(hasToEnableProtections == 1)

```

```
{
    hasToEnableProtections = 0;
    gTxDataBuffer[0] = 0x2E;
  gTxDataBuffer[1] = 0xFD;
  gTxDataBuffer[2] = 0x81;
    gTxDataBuffer[3] = 0xFF;
    gTxDataBuffer[4] = 0xFF;
    gTxDataBuffer[5] = 0xFF;
    gTxDataBuffer[6] = 0xFF;
    gTxDataBuffer[7] = 0xFF;
    gTxDataBuffer[8] = 0xFF;
    gTxDataBuffer[9] = 0xFF;
    gTxDataBuffer[10] = 0xFF;
    gTxDataBuffer[11] = 0xFF;

    txArraySize = 12;
  OSEKTL_DataReq(gTxDataBuffer, txArraySize);
}
// GET POWER MODE
else if(hasToSendGetPowerMode == 1)
{
    hasToSendGetPowerMode = 0;
    gTxDataBuffer[0] = 0x22;
  gTxDataBuffer[1] = 0xFE;
  gTxDataBuffer[2] = 0xE7;

    txArraySize = 3;
  OSEKTL_DataReq(gTxDataBuffer, txArraySize);
}
// CHANGE POWER MODE
else if(hasToSendChangePowerMode == 1)
{
    hasToSendChangePowerMode = 0;
    gTxDataBuffer[0] = 0x2E;
  gTxDataBuffer[1] = 0xFE;
  gTxDataBuffer[2] = 0xE7;
    gTxDataBuffer[3] = 0x00;
    gTxDataBuffer[4] = StartLogicControl[0];
    gTxDataBuffer[5] = StartLogicControl[1];
    gTxDataBuffer[6] = StartLogicControl[2];
    gTxDataBuffer[7] = PowerMode;
    gTxDataBuffer[8] = StartLogicControl[4];
    if(PowerMode == 0x06)
        gTxDataBuffer[9] = StartLogicControl[5] | 0x04;
    else if(PowerMode == 0x04)
        gTxDataBuffer[9] = StartLogicControl[5] & 0xFB;

    txArraySize = 10;
  OSEKTL_DataReq(gTxDataBuffer, txArraySize);
}
// CHANGE WATCHDOG STATE
else if(hasToChangeWatchdogState == 1)
{
    hasToChangeWatchdogState = 0;
    gTxDataBuffer[0] = 0x31;
  gTxDataBuffer[1] = 0x01;
  gTxDataBuffer[2] = 0xFD;
```

```
    gTxDataBuffer[3] = 0x04;
    gTxDataBuffer[4] = StopWatchdog;    // Disabled: 0x01

    txArraySize = 5;
  OSEKTL_DataReq(gTxDataBuffer, txArraySize);
}
// FAST GO TO SLEEP
else if(hasToSendFastGoToSleep == 1)
{
    hasToSendFastGoToSleep = 0;
    gTxDataBuffer[0] = 0x31;
  gTxDataBuffer[1] = 0x01;
  gTxDataBuffer[2] = 0xFD;
    gTxDataBuffer[3] = 0x01;
    gTxDataBuffer[4] = 0x01;    // Fast Go To Sleep

    txArraySize = 5;
  OSEKTL_DataReq(gTxDataBuffer, txArraySize);
}
// LIN TEST START
else if(hasToSendLINTestStart == 1)
{
    hasToSendLINTestStart = 0;
    gTxDataBuffer[0] = 0x31;
  gTxDataBuffer[1] = 0x01;
  gTxDataBuffer[2] = 0xFD;
    gTxDataBuffer[3] = 0x03;

    txArraySize = 4;
  OSEKTL_DataReq(gTxDataBuffer, txArraySize);
}
// LIN TEST GET RESULTS
else if(hasToSendLINTestGetResults == 1)
{
    hasToSendLINTestGetResults = 0;
    gTxDataBuffer[0] = 0x31;
  gTxDataBuffer[1] = 0x03;
  gTxDataBuffer[2] = 0xFD;
    gTxDataBuffer[3] = 0x03;

    txArraySize = 4;
  OSEKTL_DataReq(gTxDataBuffer, txArraySize);
}
// WAKE-UP TEST START
else if(hasToSendWakeupTestStart == 1)
{
    hasToSendWakeupTestStart = 0;
    gTxDataBuffer[0] = 0x31;
  gTxDataBuffer[1] = 0x01;
  gTxDataBuffer[2] = 0xFD;
    gTxDataBuffer[3] = 0x05;

    txArraySize = 4;
  OSEKTL_DataReq(gTxDataBuffer, txArraySize);
}
// WAKE-UP TEST GET RESULTS
else if(hasToSendWakeupTestGetResults == 1)
```

```
    {
        hasToSendWakeupTestGetResults = 0;
        gTxDataBuffer[0] = 0x31;
      gTxDataBuffer[1] = 0x03;
      gTxDataBuffer[2] = 0xFD;
        gTxDataBuffer[3] = 0x05;

        txArraySize = 4;
      OSEKTL_DataReq(gTxDataBuffer, txArraySize);
    }
    // GET SERIAL NUMBER
    else if(hasToSendGetSerialNumber == 1)
    {
        hasToSendGetSerialNumber = 0;
        gTxDataBuffer[0] = 0x22;
      gTxDataBuffer[1] = 0xF1;
      gTxDataBuffer[2] = 0x8C;

        txArraySize = 3;
      OSEKTL_DataReq(gTxDataBuffer, txArraySize);
    }
    // READ COLUMN EEPROM CONFIG PARAMETERS
    else if(hasToSendReadColumnConfig == 1)
    {
        ColumnConfigRead();
    }
    // WRITE COLUMN EEPROM CONFIG PARAMETERS
    else if(hasToSendWriteColumnConfig == 1)
    {
        ColumnConfigWrite();
    }
    // WRITE COLUMN DEFAULT CONFIG
    else if(hasToSendWriteColumnDefault == 1)
    {
        hasToSendWriteColumnDefault = 0;
        gTxDataBuffer[0] = 0x31;
      gTxDataBuffer[1] = 0x01;
      gTxDataBuffer[2] = 0x04;
        gTxDataBuffer[3] = 0x0E;
        gTxDataBuffer[4] = 0x0F;

        txArraySize = 5;
      OSEKTL_DataReq(gTxDataBuffer, txArraySize);
    }
}*/

on envVar VAR_ALL_OFF
{
  if(getValue(this))
  {
    SetStateOFF();
  }
}

on envVar VAR_ALL_ON
{
  if(getValue(this))

```

```
  {
    SetStateON();
  }
}

SetStateOFF ()
{
  long i;

  for(i=0; i<NVARS; ++i)
  {
        if( i==124 || i==164 )
        {
            CHANGE[i]=0;    // SVPAS OUTPUTS ARE NOT DEACTIVATED (T5 SPECIFIC)
        }
        else
        {
        if(TYPES[i]==2 || TYPES[i]==3)
        {
          VALUES[i]=0;
          CHANGE[i]=1;
          writeIndex=i;
        }
        }
  }
  CopyOutputsToDataBase();
}

SetStateON()
{
  long i;

  for(i=0; i<NVARS; ++i)
  {
    if(TYPES[i]==2 || TYPES[i]==3)
    {
            if( i==143 || i==144 || i==145 || i==146 || i==147 || i==148 )
                VALUES[i]=0;    // COLUMN OUTPUTS ARE NOT ACTIVATED (T5 SPECIFIC)
            else
            {
          if(NBITS[i]==1)
            VALUES[i]=0x01;
          else if(NBITS[i]==8)
            VALUES[i]=0xFF;
          else if(NBITS[i]==16)
            VALUES[i]=0xFFFF;
            }
      CHANGE[i]=1;
      if(writeIndex == NVARS) writeIndex = i;
    }
  }
  CopyOutputsToDataBase();
}
```

```
CopyOutputsToDataBase()
{
    if(TYPES[0] == 2 || TYPES[0] == 3) putValue(VAR000, VALUES[0]);
    if(TYPES[1] == 2 || TYPES[1] == 3) putValue(VAR001, VALUES[1]);
    if(TYPES[2] == 2 || TYPES[2] == 3) putValue(VAR002, VALUES[2]);
    if(TYPES[3] == 2 || TYPES[3] == 3) putValue(VAR003, VALUES[3]);
    if(TYPES[4] == 2 || TYPES[4] == 3) putValue(VAR004, VALUES[4]);
    if(TYPES[5] == 2 || TYPES[5] == 3) putValue(VAR005, VALUES[5]);
    if(TYPES[6] == 2 || TYPES[6] == 3) putValue(VAR006, VALUES[6]);
    if(TYPES[7] == 2 || TYPES[7] == 3) putValue(VAR007, VALUES[7]);
    if(TYPES[8] == 2 || TYPES[8] == 3) putValue(VAR008, VALUES[8]);
    if(TYPES[9] == 2 || TYPES[9] == 3) putValue(VAR009, VALUES[9]);
    if(TYPES[10] == 2 || TYPES[10] == 3) putValue(VAR010, VALUES[10]);
    if(TYPES[11] == 2 || TYPES[11] == 3) putValue(VAR011, VALUES[11]);
    if(TYPES[12] == 2 || TYPES[12] == 3) putValue(VAR012, VALUES[12]);
    if(TYPES[13] == 2 || TYPES[13] == 3) putValue(VAR013, VALUES[13]);
    if(TYPES[14] == 2 || TYPES[14] == 3) putValue(VAR014, VALUES[14]);
    if(TYPES[15] == 2 || TYPES[15] == 3) putValue(VAR015, VALUES[15]);
    if(TYPES[16] == 2 || TYPES[16] == 3) putValue(VAR016, VALUES[16]);
    if(TYPES[17] == 2 || TYPES[17] == 3) putValue(VAR017, VALUES[17]);
    if(TYPES[18] == 2 || TYPES[18] == 3) putValue(VAR018, VALUES[18]);
    if(TYPES[19] == 2 || TYPES[19] == 3) putValue(VAR019, VALUES[19]);
    if(TYPES[20] == 2 || TYPES[20] == 3) putValue(VAR020, VALUES[20]);
    if(TYPES[21] == 2 || TYPES[21] == 3) putValue(VAR021, VALUES[21]);
    if(TYPES[22] == 2 || TYPES[22] == 3) putValue(VAR022, VALUES[22]);
    if(TYPES[23] == 2 || TYPES[23] == 3) putValue(VAR023, VALUES[23]);
    if(TYPES[24] == 2 || TYPES[24] == 3) putValue(VAR024, VALUES[24]);
    if(TYPES[25] == 2 || TYPES[25] == 3) putValue(VAR025, VALUES[25]);
    if(TYPES[26] == 2 || TYPES[26] == 3) putValue(VAR026, VALUES[26]);
    if(TYPES[27] == 2 || TYPES[27] == 3) putValue(VAR027, VALUES[27]);
    if(TYPES[28] == 2 || TYPES[28] == 3) putValue(VAR028, VALUES[28]);
    if(TYPES[29] == 2 || TYPES[29] == 3) putValue(VAR029, VALUES[29]);
    if(TYPES[30] == 2 || TYPES[30] == 3) putValue(VAR030, VALUES[30]);
    if(TYPES[31] == 2 || TYPES[31] == 3) putValue(VAR031, VALUES[31]);
    if(TYPES[32] == 2 || TYPES[32] == 3) putValue(VAR032, VALUES[32]);
    if(TYPES[33] == 2 || TYPES[33] == 3) putValue(VAR033, VALUES[33]);
    if(TYPES[34] == 2 || TYPES[34] == 3) putValue(VAR034, VALUES[34]);
    if(TYPES[35] == 2 || TYPES[35] == 3) putValue(VAR035, VALUES[35]);
    if(TYPES[36] == 2 || TYPES[36] == 3) putValue(VAR036, VALUES[36]);
    if(TYPES[37] == 2 || TYPES[37] == 3) putValue(VAR037, VALUES[37]);
    if(TYPES[38] == 2 || TYPES[38] == 3) putValue(VAR038, VALUES[38]);
    if(TYPES[39] == 2 || TYPES[39] == 3) putValue(VAR039, VALUES[39]);
    if(TYPES[40] == 2 || TYPES[40] == 3) putValue(VAR040, VALUES[40]);
    if(TYPES[41] == 2 || TYPES[41] == 3) putValue(VAR041, VALUES[41]);
    if(TYPES[42] == 2 || TYPES[42] == 3) putValue(VAR042, VALUES[42]);
    if(TYPES[43] == 2 || TYPES[43] == 3) putValue(VAR043, VALUES[43]);
    if(TYPES[44] == 2 || TYPES[44] == 3) putValue(VAR044, VALUES[44]);
    if(TYPES[45] == 2 || TYPES[45] == 3) putValue(VAR045, VALUES[45]);
    if(TYPES[46] == 2 || TYPES[46] == 3) putValue(VAR046, VALUES[46]);
    if(TYPES[47] == 2 || TYPES[47] == 3) putValue(VAR047, VALUES[47]);
    if(TYPES[48] == 2 || TYPES[48] == 3) putValue(VAR048, VALUES[48]);
    if(TYPES[49] == 2 || TYPES[49] == 3) putValue(VAR049, VALUES[49]);
    if(TYPES[50] == 2 || TYPES[50] == 3) putValue(VAR050, VALUES[50]);
    if(TYPES[51] == 2 || TYPES[51] == 3) putValue(VAR051, VALUES[51]);
    if(TYPES[52] == 2 || TYPES[52] == 3) putValue(VAR052, VALUES[52]);
    if(TYPES[53] == 2 || TYPES[53] == 3) putValue(VAR053, VALUES[53]);
    if(TYPES[54] == 2 || TYPES[54] == 3) putValue(VAR054, VALUES[54]);

```

```
    if(TYPES[55] == 2 || TYPES[55] == 3) putValue(VAR055, VALUES[55]);
2686 if(TYPES[56] == 2 || TYPES[56] == 3) putValue(VAR056, VALUES[56]);
2687 if(TYPES[57] == 2 || TYPES[57] == 3) putValue(VAR057, VALUES[57]);
2688 if(TYPES[58] == 2 || TYPES[58] == 3) putValue(VAR058, VALUES[58]);
2689 if(TYPES[59] == 2 || TYPES[59] == 3) putValue(VAR059, VALUES[59]);
2690 if(TYPES[60] == 2 || TYPES[60] == 3) putValue(VAR060, VALUES[60]);
2691 if(TYPES[61] == 2 || TYPES[61] == 3) putValue(VAR061, VALUES[61]);
2692 if(TYPES[62] == 2 || TYPES[62] == 3) putValue(VAR062, VALUES[62]);
2693 if(TYPES[63] == 2 || TYPES[63] == 3) putValue(VAR063, VALUES[63]);
2694 if(TYPES[64] == 2 || TYPES[64] == 3) putValue(VAR064, VALUES[64]);
2695 if(TYPES[65] == 2 || TYPES[65] == 3) putValue(VAR065, VALUES[65]);
2696 if(TYPES[66] == 2 || TYPES[66] == 3) putValue(VAR066, VALUES[66]);
2697 if(TYPES[67] == 2 || TYPES[67] == 3) putValue(VAR067, VALUES[67]);
2698 if(TYPES[68] == 2 || TYPES[68] == 3) putValue(VAR068, VALUES[68]);
2699 if(TYPES[69] == 2 || TYPES[69] == 3) putValue(VAR069, VALUES[69]);
2700 if(TYPES[70] == 2 || TYPES[70] == 3) putValue(VAR070, VALUES[70]);
2701 if(TYPES[71] == 2 || TYPES[71] == 3) putValue(VAR071, VALUES[71]);
2702 if(TYPES[72] == 2 || TYPES[72] == 3) putValue(VAR072, VALUES[72]);
2703 if(TYPES[73] == 2 || TYPES[73] == 3) putValue(VAR073, VALUES[73]);
2704 if(TYPES[74] == 2 || TYPES[74] == 3) putValue(VAR074, VALUES[74]);
2705 if(TYPES[75] == 2 || TYPES[75] == 3) putValue(VAR075, VALUES[75]);
2706 if(TYPES[76] == 2 || TYPES[76] == 3) putValue(VAR076, VALUES[76]);
2707 if(TYPES[77] == 2 || TYPES[77] == 3) putValue(VAR077, VALUES[77]);
2708 if(TYPES[78] == 2 || TYPES[78] == 3) putValue(VAR078, VALUES[78]);
2709 if(TYPES[79] == 2 || TYPES[79] == 3) putValue(VAR079, VALUES[79]);
2710 if(TYPES[80] == 2 || TYPES[80] == 3) putValue(VAR080, VALUES[80]);
2711 if(TYPES[81] == 2 || TYPES[81] == 3) putValue(VAR081, VALUES[81]);
2712 if(TYPES[82] == 2 || TYPES[82] == 3) putValue(VAR082, VALUES[82]);
2713 if(TYPES[83] == 2 || TYPES[83] == 3) putValue(VAR083, VALUES[83]);
2714 if(TYPES[84] == 2 || TYPES[84] == 3) putValue(VAR084, VALUES[84]);
2715 if(TYPES[85] == 2 || TYPES[85] == 3) putValue(VAR085, VALUES[85]);
2716 if(TYPES[86] == 2 || TYPES[86] == 3) putValue(VAR086, VALUES[86]);
2717 if(TYPES[87] == 2 || TYPES[87] == 3) putValue(VAR087, VALUES[87]);
2718 if(TYPES[88] == 2 || TYPES[88] == 3) putValue(VAR088, VALUES[88]);
2719 if(TYPES[89] == 2 || TYPES[89] == 3) putValue(VAR089, VALUES[89]);
2720 if(TYPES[90] == 2 || TYPES[90] == 3) putValue(VAR090, VALUES[90]);
2721 if(TYPES[91] == 2 || TYPES[91] == 3) putValue(VAR091, VALUES[91]);
2722 if(TYPES[92] == 2 || TYPES[92] == 3) putValue(VAR092, VALUES[92]);
2723 if(TYPES[93] == 2 || TYPES[93] == 3) putValue(VAR093, VALUES[93]);
2724 if(TYPES[94] == 2 || TYPES[94] == 3) putValue(VAR094, VALUES[94]);
2725 if(TYPES[95] == 2 || TYPES[95] == 3) putValue(VAR095, VALUES[95]);
2726 if(TYPES[96] == 2 || TYPES[96] == 3) putValue(VAR096, VALUES[96]);
2727 if(TYPES[97] == 2 || TYPES[97] == 3) putValue(VAR097, VALUES[97]);
2728 if(TYPES[98] == 2 || TYPES[98] == 3) putValue(VAR098, VALUES[98]);
2729 if(TYPES[99] == 2 || TYPES[99] == 3) putValue(VAR099, VALUES[99]);
2730 if(TYPES[100] == 2 || TYPES[100] == 3) putValue(VAR100, VALUES[100]);
2731 if(TYPES[101] == 2 || TYPES[101] == 3) putValue(VAR101, VALUES[101]);
2732 if(TYPES[102] == 2 || TYPES[102] == 3) putValue(VAR102, VALUES[102]);
2733 if(TYPES[103] == 2 || TYPES[103] == 3) putValue(VAR103, VALUES[103]);
2734 if(TYPES[104] == 2 || TYPES[104] == 3) putValue(VAR104, VALUES[104]);
2735 if(TYPES[105] == 2 || TYPES[105] == 3) putValue(VAR105, VALUES[105]);
2736 if(TYPES[106] == 2 || TYPES[106] == 3) putValue(VAR106, VALUES[106]);
2737 if(TYPES[107] == 2 || TYPES[107] == 3) putValue(VAR107, VALUES[107]);
2738 if(TYPES[108] == 2 || TYPES[108] == 3) putValue(VAR108, VALUES[108]);
2739 if(TYPES[109] == 2 || TYPES[109] == 3) putValue(VAR109, VALUES[109]);
2740 if(TYPES[110] == 2 || TYPES[110] == 3) putValue(VAR110, VALUES[110]);
2741 if(TYPES[111] == 2 || TYPES[111] == 3) putValue(VAR111, VALUES[111]);
2742
```

```
    if(TYPES[112] == 2 || TYPES[112] == 3) putValue(VAR112, VALUES[112]);
2743 if(TYPES[113] == 2 || TYPES[113] == 3) putValue(VAR113, VALUES[113]);
2744 if(TYPES[114] == 2 || TYPES[114] == 3) putValue(VAR114, VALUES[114]);
2745 if(TYPES[115] == 2 || TYPES[115] == 3) putValue(VAR115, VALUES[115]);
2746 if(TYPES[116] == 2 || TYPES[116] == 3) putValue(VAR116, VALUES[116]);
2747 if(TYPES[117] == 2 || TYPES[117] == 3) putValue(VAR117, VALUES[117]);
2748 if(TYPES[118] == 2 || TYPES[118] == 3) putValue(VAR118, VALUES[118]);
2749 if(TYPES[119] == 2 || TYPES[119] == 3) putValue(VAR119, VALUES[119]);
2750 if(TYPES[120] == 2 || TYPES[120] == 3) putValue(VAR120, VALUES[120]);
2751 if(TYPES[121] == 2 || TYPES[121] == 3) putValue(VAR121, VALUES[121]);
2752 if(TYPES[122] == 2 || TYPES[122] == 3) putValue(VAR122, VALUES[122]);
2753 if(TYPES[123] == 2 || TYPES[123] == 3) putValue(VAR123, VALUES[123]);
2754 if(TYPES[124] == 2 || TYPES[124] == 3) putValue(VAR124, VALUES[124]);
2755 if(TYPES[125] == 2 || TYPES[125] == 3) putValue(VAR125, VALUES[125]);
2756 if(TYPES[126] == 2 || TYPES[126] == 3) putValue(VAR126, VALUES[126]);
2757 if(TYPES[127] == 2 || TYPES[127] == 3) putValue(VAR127, VALUES[127]);
2758 if(TYPES[128] == 2 || TYPES[128] == 3) putValue(VAR128, VALUES[128]);
2759 if(TYPES[129] == 2 || TYPES[129] == 3) putValue(VAR129, VALUES[129]);
2760 if(TYPES[130] == 2 || TYPES[130] == 3) putValue(VAR130, VALUES[130]);
2761 if(TYPES[131] == 2 || TYPES[131] == 3) putValue(VAR131, VALUES[131]);
2762 if(TYPES[132] == 2 || TYPES[132] == 3) putValue(VAR132, VALUES[132]);
2763 if(TYPES[133] == 2 || TYPES[133] == 3) putValue(VAR133, VALUES[133]);
2764 if(TYPES[134] == 2 || TYPES[134] == 3) putValue(VAR134, VALUES[134]);
2765 if(TYPES[135] == 2 || TYPES[135] == 3) putValue(VAR135, VALUES[135]);
2766 if(TYPES[136] == 2 || TYPES[136] == 3) putValue(VAR136, VALUES[136]);
2767 if(TYPES[137] == 2 || TYPES[137] == 3) putValue(VAR137, VALUES[137]);
2768 if(TYPES[138] == 2 || TYPES[138] == 3) putValue(VAR138, VALUES[138]);
2769 if(TYPES[139] == 2 || TYPES[139] == 3) putValue(VAR139, VALUES[139]);
2770 if(TYPES[140] == 2 || TYPES[140] == 3) putValue(VAR140, VALUES[140]);
2771 if(TYPES[141] == 2 || TYPES[141] == 3) putValue(VAR141, VALUES[141]);
2772 if(TYPES[142] == 2 || TYPES[142] == 3) putValue(VAR142, VALUES[142]);
2773 if(TYPES[143] == 2 || TYPES[143] == 3) putValue(VAR143, VALUES[143]);
2774 if(TYPES[144] == 2 || TYPES[144] == 3) putValue(VAR144, VALUES[144]);
2775 if(TYPES[145] == 2 || TYPES[145] == 3) putValue(VAR145, VALUES[145]);
2776 if(TYPES[146] == 2 || TYPES[146] == 3) putValue(VAR146, VALUES[146]);
2777 if(TYPES[147] == 2 || TYPES[147] == 3) putValue(VAR147, VALUES[147]);
2778 if(TYPES[148] == 2 || TYPES[148] == 3) putValue(VAR148, VALUES[148]);
2779 if(TYPES[149] == 2 || TYPES[149] == 3) putValue(VAR149, VALUES[149]);
2780 if(TYPES[150] == 2 || TYPES[150] == 3) putValue(VAR150, VALUES[150]);
2781 if(TYPES[151] == 2 || TYPES[151] == 3) putValue(VAR151, VALUES[151]);
2782 if(TYPES[152] == 2 || TYPES[152] == 3) putValue(VAR152, VALUES[152]);
2783 if(TYPES[153] == 2 || TYPES[153] == 3) putValue(VAR153, VALUES[153]);
2784 if(TYPES[154] == 2 || TYPES[154] == 3) putValue(VAR154, VALUES[154]);
2785 if(TYPES[155] == 2 || TYPES[155] == 3) putValue(VAR155, VALUES[155]);
2786 if(TYPES[156] == 2 || TYPES[156] == 3) putValue(VAR156, VALUES[156]);
2787 if(TYPES[157] == 2 || TYPES[157] == 3) putValue(VAR157, VALUES[157]);
2788 if(TYPES[158] == 2 || TYPES[158] == 3) putValue(VAR158, VALUES[158]);
2789 if(TYPES[159] == 2 || TYPES[159] == 3) putValue(VAR159, VALUES[159]);
2790 if(TYPES[160] == 2 || TYPES[160] == 3) putValue(VAR160, VALUES[160]);
2791 if(TYPES[161] == 2 || TYPES[161] == 3) putValue(VAR161, VALUES[161]);
2792 if(TYPES[162] == 2 || TYPES[162] == 3) putValue(VAR162, VALUES[162]);
2793 if(TYPES[163] == 2 || TYPES[163] == 3) putValue(VAR163, VALUES[163]);
2794 if(TYPES[164] == 2 || TYPES[164] == 3) putValue(VAR164, VALUES[164]);
2795 if(TYPES[165] == 2 || TYPES[165] == 3) putValue(VAR165, VALUES[165]);
2796 if(TYPES[166] == 2 || TYPES[166] == 3) putValue(VAR166, VALUES[166]);
2797 if(TYPES[167] == 2 || TYPES[167] == 3) putValue(VAR167, VALUES[167]);
2798 if(TYPES[168] == 2 || TYPES[168] == 3) putValue(VAR168, VALUES[168]);
2799
```

```
    if(TYPES[169] == 2 || TYPES[169] == 3) putValue(VAR169, VALUES[169]);
2800 if(TYPES[170] == 2 || TYPES[170] == 3) putValue(VAR170, VALUES[170]);
2801 if(TYPES[171] == 2 || TYPES[171] == 3) putValue(VAR171, VALUES[171]);
2802 if(TYPES[172] == 2 || TYPES[172] == 3) putValue(VAR172, VALUES[172]);
2803 if(TYPES[173] == 2 || TYPES[173] == 3) putValue(VAR173, VALUES[173]);
2804 if(TYPES[174] == 2 || TYPES[174] == 3) putValue(VAR174, VALUES[174]);
2805 if(TYPES[175] == 2 || TYPES[175] == 3) putValue(VAR175, VALUES[175]);
2806 if(TYPES[176] == 2 || TYPES[176] == 3) putValue(VAR176, VALUES[176]);
2807 if(TYPES[177] == 2 || TYPES[177] == 3) putValue(VAR177, VALUES[177]);
2808 if(TYPES[178] == 2 || TYPES[178] == 3) putValue(VAR178, VALUES[178]);
2809 if(TYPES[179] == 2 || TYPES[179] == 3) putValue(VAR179, VALUES[179]);
2810 if(TYPES[180] == 2 || TYPES[180] == 3) putValue(VAR180, VALUES[180]);
2811 if(TYPES[181] == 2 || TYPES[181] == 3) putValue(VAR181, VALUES[181]);
2812 if(TYPES[182] == 2 || TYPES[182] == 3) putValue(VAR182, VALUES[182]);
2813 if(TYPES[183] == 2 || TYPES[183] == 3) putValue(VAR183, VALUES[183]);
2814 if(TYPES[184] == 2 || TYPES[184] == 3) putValue(VAR184, VALUES[184]);
2815 if(TYPES[185] == 2 || TYPES[185] == 3) putValue(VAR185, VALUES[185]);
2816 if(TYPES[186] == 2 || TYPES[186] == 3) putValue(VAR186, VALUES[186]);
2817 if(TYPES[187] == 2 || TYPES[187] == 3) putValue(VAR187, VALUES[187]);
2818 if(TYPES[188] == 2 || TYPES[188] == 3) putValue(VAR188, VALUES[188]);
2819 if(TYPES[189] == 2 || TYPES[189] == 3) putValue(VAR189, VALUES[189]);
2820 if(TYPES[190] == 2 || TYPES[190] == 3) putValue(VAR190, VALUES[190]);
2821 if(TYPES[191] == 2 || TYPES[191] == 3) putValue(VAR191, VALUES[191]);
2822 if(TYPES[192] == 2 || TYPES[192] == 3) putValue(VAR192, VALUES[192]);
2823 if(TYPES[193] == 2 || TYPES[193] == 3) putValue(VAR193, VALUES[193]);
2824 if(TYPES[194] == 2 || TYPES[194] == 3) putValue(VAR194, VALUES[194]);
2825 if(TYPES[195] == 2 || TYPES[195] == 3) putValue(VAR195, VALUES[195]);
2826 if(TYPES[196] == 2 || TYPES[196] == 3) putValue(VAR196, VALUES[196]);
2827 if(TYPES[197] == 2 || TYPES[197] == 3) putValue(VAR197, VALUES[197]);
2828 if(TYPES[198] == 2 || TYPES[198] == 3) putValue(VAR198, VALUES[198]);
2829 if(TYPES[199] == 2 || TYPES[199] == 3) putValue(VAR199, VALUES[199]);
2830 if(TYPES[200] == 2 || TYPES[200] == 3) putValue(VAR200, VALUES[200]);
2831 if(TYPES[201] == 2 || TYPES[201] == 3) putValue(VAR201, VALUES[201]);
2832 if(TYPES[202] == 2 || TYPES[202] == 3) putValue(VAR202, VALUES[202]);
2833 if(TYPES[203] == 2 || TYPES[203] == 3) putValue(VAR203, VALUES[203]);
2834 if(TYPES[204] == 2 || TYPES[204] == 3) putValue(VAR204, VALUES[204]);
2835 if(TYPES[205] == 2 || TYPES[205] == 3) putValue(VAR205, VALUES[205]);
2836 if(TYPES[206] == 2 || TYPES[206] == 3) putValue(VAR206, VALUES[206]);
2837 if(TYPES[207] == 2 || TYPES[207] == 3) putValue(VAR207, VALUES[207]);
2838 if(TYPES[208] == 2 || TYPES[208] == 3) putValue(VAR208, VALUES[208]);
2839 if(TYPES[209] == 2 || TYPES[209] == 3) putValue(VAR209, VALUES[209]);
2840 if(TYPES[210] == 2 || TYPES[210] == 3) putValue(VAR210, VALUES[210]);
2841 if(TYPES[211] == 2 || TYPES[211] == 3) putValue(VAR211, VALUES[211]);
2842 if(TYPES[212] == 2 || TYPES[212] == 3) putValue(VAR212, VALUES[212]);
2843 if(TYPES[213] == 2 || TYPES[213] == 3) putValue(VAR213, VALUES[213]);
2844 if(TYPES[214] == 2 || TYPES[214] == 3) putValue(VAR214, VALUES[214]);
2845 if(TYPES[215] == 2 || TYPES[215] == 3) putValue(VAR215, VALUES[215]);
2846 if(TYPES[216] == 2 || TYPES[216] == 3) putValue(VAR216, VALUES[216]);
2847 if(TYPES[217] == 2 || TYPES[217] == 3) putValue(VAR217, VALUES[217]);
2848 if(TYPES[218] == 2 || TYPES[218] == 3) putValue(VAR218, VALUES[218]);
2849 if(TYPES[219] == 2 || TYPES[219] == 3) putValue(VAR219, VALUES[219]);
2850 if(TYPES[220] == 2 || TYPES[220] == 3) putValue(VAR220, VALUES[220]);
2851 if(TYPES[221] == 2 || TYPES[221] == 3) putValue(VAR221, VALUES[221]);
2852 if(TYPES[222] == 2 || TYPES[222] == 3) putValue(VAR222, VALUES[222]);
2853 if(TYPES[223] == 2 || TYPES[223] == 3) putValue(VAR223, VALUES[223]);
2854 if(TYPES[224] == 2 || TYPES[224] == 3) putValue(VAR224, VALUES[224]);
2855 if(TYPES[225] == 2 || TYPES[225] == 3) putValue(VAR225, VALUES[225]);
2856
```

```
    if(TYPES[226] == 2 || TYPES[226] == 3) putValue(VAR226, VALUES[226]);
2857 if(TYPES[227] == 2 || TYPES[227] == 3) putValue(VAR227, VALUES[227]);
2858 if(TYPES[228] == 2 || TYPES[228] == 3) putValue(VAR228, VALUES[228]);
2859 if(TYPES[229] == 2 || TYPES[229] == 3) putValue(VAR229, VALUES[229]);
2860 if(TYPES[230] == 2 || TYPES[230] == 3) putValue(VAR230, VALUES[230]);
2861 if(TYPES[231] == 2 || TYPES[231] == 3) putValue(VAR231, VALUES[231]);
2862 if(TYPES[232] == 2 || TYPES[232] == 3) putValue(VAR232, VALUES[232]);
2863 if(TYPES[233] == 2 || TYPES[233] == 3) putValue(VAR233, VALUES[233]);
2864 if(TYPES[234] == 2 || TYPES[234] == 3) putValue(VAR234, VALUES[234]);
2865 if(TYPES[235] == 2 || TYPES[235] == 3) putValue(VAR235, VALUES[235]);
2866 if(TYPES[236] == 2 || TYPES[236] == 3) putValue(VAR236, VALUES[236]);
2867 if(TYPES[237] == 2 || TYPES[237] == 3) putValue(VAR237, VALUES[237]);
2868 if(TYPES[238] == 2 || TYPES[238] == 3) putValue(VAR238, VALUES[238]);
2869 if(TYPES[239] == 2 || TYPES[239] == 3) putValue(VAR239, VALUES[239]);
2870 if(TYPES[240] == 2 || TYPES[240] == 3) putValue(VAR240, VALUES[240]);
2871 if(TYPES[241] == 2 || TYPES[241] == 3) putValue(VAR241, VALUES[241]);
2872 if(TYPES[242] == 2 || TYPES[242] == 3) putValue(VAR242, VALUES[242]);
2873 if(TYPES[243] == 2 || TYPES[243] == 3) putValue(VAR243, VALUES[243]);
2874 if(TYPES[244] == 2 || TYPES[244] == 3) putValue(VAR244, VALUES[244]);
2875 if(TYPES[245] == 2 || TYPES[245] == 3) putValue(VAR245, VALUES[245]);
2876 if(TYPES[246] == 2 || TYPES[246] == 3) putValue(VAR246, VALUES[246]);
2877 if(TYPES[247] == 2 || TYPES[247] == 3) putValue(VAR247, VALUES[247]);
2878 if(TYPES[248] == 2 || TYPES[248] == 3) putValue(VAR248, VALUES[248]);
2879 if(TYPES[249] == 2 || TYPES[249] == 3) putValue(VAR249, VALUES[249]);
2880 if(TYPES[250] == 2 || TYPES[250] == 3) putValue(VAR250, VALUES[250]);
2881 if(TYPES[251] == 2 || TYPES[251] == 3) putValue(VAR251, VALUES[251]);
2882 if(TYPES[252] == 2 || TYPES[252] == 3) putValue(VAR252, VALUES[252]);
2883 if(TYPES[253] == 2 || TYPES[253] == 3) putValue(VAR253, VALUES[253]);
2884 if(TYPES[254] == 2 || TYPES[254] == 3) putValue(VAR254, VALUES[254]);
2885 if(TYPES[255] == 2 || TYPES[255] == 3) putValue(VAR255, VALUES[255]);
2886 if(TYPES[256] == 2 || TYPES[256] == 3) putValue(VAR256, VALUES[256]);
2887 if(TYPES[257] == 2 || TYPES[257] == 3) putValue(VAR257, VALUES[257]);
2888 if(TYPES[258] == 2 || TYPES[258] == 3) putValue(VAR258, VALUES[258]);
2889 if(TYPES[259] == 2 || TYPES[259] == 3) putValue(VAR259, VALUES[259]);
2890 if(TYPES[260] == 2 || TYPES[260] == 3) putValue(VAR260, VALUES[260]);
2891 if(TYPES[261] == 2 || TYPES[261] == 3) putValue(VAR261, VALUES[261]);
2892 if(TYPES[262] == 2 || TYPES[262] == 3) putValue(VAR262, VALUES[262]);
2893 if(TYPES[263] == 2 || TYPES[263] == 3) putValue(VAR263, VALUES[263]);
2894 if(TYPES[264] == 2 || TYPES[264] == 3) putValue(VAR264, VALUES[264]);
2895 if(TYPES[265] == 2 || TYPES[265] == 3) putValue(VAR265, VALUES[265]);
2896 if(TYPES[266] == 2 || TYPES[266] == 3) putValue(VAR266, VALUES[266]);
2897 if(TYPES[267] == 2 || TYPES[267] == 3) putValue(VAR267, VALUES[267]);
2898 if(TYPES[268] == 2 || TYPES[268] == 3) putValue(VAR268, VALUES[268]);
2899 if(TYPES[269] == 2 || TYPES[269] == 3) putValue(VAR269, VALUES[269]);
2900 if(TYPES[270] == 2 || TYPES[270] == 3) putValue(VAR270, VALUES[270]);
2901 if(TYPES[271] == 2 || TYPES[271] == 3) putValue(VAR271, VALUES[271]);
2902 if(TYPES[272] == 2 || TYPES[272] == 3) putValue(VAR272, VALUES[272]);
2903 if(TYPES[273] == 2 || TYPES[273] == 3) putValue(VAR273, VALUES[273]);
2904 if(TYPES[274] == 2 || TYPES[274] == 3) putValue(VAR274, VALUES[274]);
2905 if(TYPES[275] == 2 || TYPES[275] == 3) putValue(VAR275, VALUES[275]);
2906 if(TYPES[276] == 2 || TYPES[276] == 3) putValue(VAR276, VALUES[276]);
2907 if(TYPES[277] == 2 || TYPES[277] == 3) putValue(VAR277, VALUES[277]);
2908 if(TYPES[278] == 2 || TYPES[278] == 3) putValue(VAR278, VALUES[278]);
2909 if(TYPES[279] == 2 || TYPES[279] == 3) putValue(VAR279, VALUES[279]);
2910 if(TYPES[280] == 2 || TYPES[280] == 3) putValue(VAR280, VALUES[280]);
2911 if(TYPES[281] == 2 || TYPES[281] == 3) putValue(VAR281, VALUES[281]);
2912 if(TYPES[282] == 2 || TYPES[282] == 3) putValue(VAR282, VALUES[282]);
2913
```

```
    if(TYPES[283] == 2 || TYPES[283] == 3) putValue(VAR283, VALUES[283]);
    if(TYPES[284] == 2 || TYPES[284] == 3) putValue(VAR284, VALUES[284]);
    if(TYPES[285] == 2 || TYPES[285] == 3) putValue(VAR285, VALUES[285]);
    if(TYPES[286] == 2 || TYPES[286] == 3) putValue(VAR286, VALUES[286]);
    if(TYPES[287] == 2 || TYPES[287] == 3) putValue(VAR287, VALUES[287]);
    if(TYPES[288] == 2 || TYPES[288] == 3) putValue(VAR288, VALUES[288]);
    if(TYPES[289] == 2 || TYPES[289] == 3) putValue(VAR289, VALUES[289]);
    if(TYPES[290] == 2 || TYPES[290] == 3) putValue(VAR290, VALUES[290]);
    if(TYPES[291] == 2 || TYPES[291] == 3) putValue(VAR291, VALUES[291]);
    if(TYPES[292] == 2 || TYPES[292] == 3) putValue(VAR292, VALUES[292]);
    if(TYPES[293] == 2 || TYPES[293] == 3) putValue(VAR293, VALUES[293]);
    if(TYPES[294] == 2 || TYPES[294] == 3) putValue(VAR294, VALUES[294]);
    if(TYPES[295] == 2 || TYPES[295] == 3) putValue(VAR295, VALUES[295]);
    if(TYPES[296] == 2 || TYPES[296] == 3) putValue(VAR296, VALUES[296]);
    if(TYPES[297] == 2 || TYPES[297] == 3) putValue(VAR297, VALUES[297]);
    if(TYPES[298] == 2 || TYPES[298] == 3) putValue(VAR298, VALUES[298]);
    if(TYPES[299] == 2 || TYPES[299] == 3) putValue(VAR299, VALUES[299]);
}

on timer GetLINTestResults
{
    hasToSendLINTestGetResults=1;
}

AT_LoadConfig ()
{
  dword readHandle;
  char readBuffer[1024];
  char dataBuffer[255];
  long i;
  long j;
  long k;
  long index;
    char fileNameAT[255];
    BYTE headerOK;
    long AT_STEPS_tmp;

    readHandle = 0;
    i = 0;
    j = 0;
    k = 0;
    index = 0;
    putValue(AT_NVARS, 0);
    putValue(AT_STEPS, 0);
    getValue(AT_File, fileNameAT);
    AT_STEPS_tmp = 0;

  write("Opening AT File");
  readHandle = openFileRead (fileNameAT, 0);
  //if(readHandle != 0) fileRewind(readHandle);
  if (readHandle != 0 && fileGetString(readBuffer, elcount(readBuffer),
readHandle) != 0)
  {
        // READ HEADER
        headerOK = 1;
        {
```

```
            // Read TYPE
      j = 0; i = 0;
      while (readBuffer[i] != SEP && readBuffer[i] != 0 && i<=elcount(readBuffer))
      {
        dataBuffer[j] = readBuffer[i];
        j++; i++;
      }
      dataBuffer[j] = 0;
      if(strncmp(dataBuffer, "TYPE", strlen(dataBuffer))!=0) headerOK = 0;
            if(readBuffer[i] != SEP) headerOK = 0;

      // Read NAME
      j = 0; i++;
      while (readBuffer[i] != SEP && readBuffer[i] != 0 && i<=elcount(readBuffer))
      {
        dataBuffer[j] = readBuffer[i];
        j++; i++;
      }
      dataBuffer[j] = 0;
      if(strncmp(dataBuffer, "NAME", strlen(dataBuffer))!=0) headerOK = 0;
            if(readBuffer[i] != SEP) headerOK = 0;

      // Read TOLERANCE
      j = 0; i++;
      while (readBuffer[i] != SEP && readBuffer[i] != 0 && i<=elcount(readBuffer))
      {
        dataBuffer[j] = readBuffer[i];
        j++; i++;
      }
      dataBuffer[j] = 0;
      if(strncmp(dataBuffer, "TOL", strlen(dataBuffer))!=0) headerOK = 0;
            if(readBuffer[i] != SEP) headerOK = 0;

      // Read Number of Steps
            do
            {
          j = 0; i++;
          while (readBuffer[i] != SEP && readBuffer[i] != 0 && i<=elcount(readBuffer))
          {
            dataBuffer[j] = readBuffer[i];
            j++; i++;
          }
                AT_STEPS_tmp++;
            } while(readBuffer[i] == SEP);
            putValue(AT_STEPS, AT_STEPS_tmp);
            if(readBuffer[i-1] != 0x0A) headerOK = 0;
            if(readBuffer[i] != 0x00) headerOK = 0;
        }
        if(headerOK == 0) write("Header Incorrect");

        // READ ROWS

```

```
    while(fileGetString(readBuffer, elcount(readBuffer), readHandle) !=
0 & headerOK)
    {
      // Read TYPE
      j = 0; i = 0;
      while (readBuffer[i] != SEP && readBuffer[i] != 0 && i<=elcount(
readBuffer))
      {
        dataBuffer[j] = readBuffer[i];
        j++; i++;
      }
      dataBuffer[j] = 0;
      if(strncmp(dataBuffer, "DI", strlen(dataBuffer))==0)
        AT_TYPES[index] = 0;
      else if(strncmp(dataBuffer, "AI", strlen(dataBuffer))==0)
        AT_TYPES[index] = 1;
      else if(strncmp(dataBuffer, "DO", strlen(dataBuffer))==0)
        AT_TYPES[index] = 2;
      else if(strncmp(dataBuffer, "PO", strlen(dataBuffer))==0)
        AT_TYPES[index] = 3;
      else if(strncmp(dataBuffer, "AD", strlen(dataBuffer))==0)
        AT_TYPES[index] = 4;
      else if(strncmp(dataBuffer, "DD", strlen(dataBuffer))==0)
        AT_TYPES[index] = 5;
            //if(readBuffer[i] != SEP) write("Reading Error");

      // Read NAME
      j = 0; i++;
      while (readBuffer[i] != SEP && readBuffer[i] != 0 && i<=elcount(
readBuffer))
      {
        dataBuffer[j] = readBuffer[i];
        j++; i++;
      }
      dataBuffer[j] = 0;
      for(k=0; k<=j & k<MAX_VAR_LENGTH; k++)
        AT_NAMES[index][k]=dataBuffer[k];
            //if(readBuffer[i] != SEP) write("Reading Error");

      // Read TOLERANCE
      j = 0; i++;
      while (readBuffer[i] != SEP && readBuffer[i] != 0 && i<=elcount(
readBuffer))
      {
        dataBuffer[j] = readBuffer[i];
        j++; i++;
      };
      dataBuffer[j] = 0;
      AT_TOLERANCE[index] = (WORD)atol(dataBuffer);
            //if(readBuffer[i] != SEP) write("Reading Error");

      // Read Step Values
            for(k=0; k<getValue(AT_STEPS); k++)
            {
          j = 0; i++;
          while (readBuffer[i] != SEP && readBuffer[i] != 0 &&
readBuffer[i] != 0x0A && i<=elcount(readBuffer))
```

```
            {
3073          dataBuffer[j] = readBuffer[i];
3074          j++; i++;
3075        };
3076        dataBuffer[j] = 0;
3077        AT_VALUES[index][k] = atol(dataBuffer);
3078              //write("k=%d  =%s      value=%d",k, AT_NAMES[index],
3079 AT_VALUES[index][k]);
              //if(readBuffer[i] != SEP & readBuffer[i] != 0x0A) write
3080 ("Reading Error");
            }
3081
3082         if(k==getValue(AT_STEPS))
3083         {
3084             AT_VALUES[index][k] = AT_VALUES[index][0];
3085             //write("k=%d  =%s      value=%d",k, AT_NAMES[index],
3086 AT_VALUES[index][k]);
            }
3087
3088
3089
3090
3091      index++;
3092    };
3093        putValue(AT_NVARS, index);
3094    write("Readed %d variables", getValue(AT_NVARS));
3095    write("Closing AT File");
3096    fileClose(readHandle);
3097  }
3098  else
3099  {
3100    write("AT file cannot be opened");
3101  }
3102 }
3103
3104 AT_PrintConfig ()
3105 {
3106  long i;
3107  for(i=0; i<getValue(AT_NVARS); ++i)
3108  {
3109    write("TYPE: %d, NAME: %s, TOL: %d, S0: %d, S1: %d, S2: %d",
3110 AT_TYPES[i], AT_NAMES[i], AT_TOLERANCE[i], AT_VALUES[i][0], AT_VALUES[i]
   [1], AT_VALUES[i][2]);
  }
3111    write("Variables: %d", AT_NVARS);
3112    write("Steps %d", AT_STEPS);
3113 }
3114
3115 on envVar AT_Load
3116 {
3117  if(getValue(this))
3118  {
3119    putValue(this,0);
3120        AT_LoadConfig();
3121        //AT_PrintConfig();
3122
3123        state = 0;      // Normal Mode
3124
```

```
        AT_State = 0;   // Write Outputs
        AT_Mode = 0;    // Playing
        AT_IncMode = 1; // Increasing
        AT_SeqIndex = 0;
        AT_SeqState = 0;
        AT_OneStep = 0;
        putValue(AT_Step, 0);
  }
}

AT_SaveConfig ()
{
    dword writeHandle;
  char writeBuffer[AT_MAX_LINE_CHARS];
  char tmp[255];
    char EOL[2];
    char sep_str[2];
  long i;
  long j;
  long k;
  long index;
    char fileNameAT[255];
    BYTE headerOK;

    writeHandle = 0;
    i = 0;
    j = 0;
    k = 0;
    index = 0;
    EOL[0] = 0x0A; EOL[1] = 0x00;
    sep_str[0] = SEP; sep_str[1] = 0x00;
    getValue(AT_File, fileNameAT);

  write("Opening AT File");
  writeHandle = openFileWrite (fileNameAT, 0);
  if (writeHandle != 0 )
  {
        // HEADER
        snprintf(writeBuffer, elcount(writeBuffer), "%s", "TYPE");
        strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
        strncat(writeBuffer, "NAME", AT_MAX_LINE_CHARS);
        strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
        strncat(writeBuffer, "TOL", AT_MAX_LINE_CHARS);
        strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
        for(i=0; i< getValue(AT_STEPS); ++i)
        {
            snprintf(tmp, elcount(tmp), "%d", i+1);
            strncat(writeBuffer, tmp, AT_MAX_LINE_CHARS);
            if(i != getValue(AT_STEPS)-1) strncat(writeBuffer, sep_str,
AT_MAX_LINE_CHARS);
        }
        strncat(writeBuffer, EOL, AT_MAX_LINE_CHARS);
        filePutString(writeBuffer, strlen(writeBuffer), writeHandle);

        // VARIABLES
        for(index=0; index<getValue(AT_NVARS); ++index)
        {
```

```
3181            switch(AT_TYPES[index])
3182            {
3183                case 0:
3184                {
3185                    snprintf(writeBuffer, elcount(writeBuffer), "%s", "DI");
3186                    break;
3187                }
3188                case 1:
3189                {
3190                    snprintf(writeBuffer, elcount(writeBuffer), "%s", "AI");
3191                    break;
3192                }
3193                case 2:
3194                {
3195                    snprintf(writeBuffer, elcount(writeBuffer), "%s", "DO");
3196                    break;
3197                }
3198                case 3:
3199                {
3200                    snprintf(writeBuffer, elcount(writeBuffer), "%s", "PO");
3201                    break;
3202                }
3203                case 4:
3204                {
3205                    snprintf(writeBuffer, elcount(writeBuffer), "%s", "AD");
3206                    break;
3207                }
3208                case 5:
3209                {
3210                    snprintf(writeBuffer, elcount(writeBuffer), "%s", "DD");
3211                    break;
3212                }
3213            }
3214            strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
3215            strncat(writeBuffer, AT_NAMES[index], AT_MAX_LINE_CHARS);
3216            strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
3217            snprintf(tmp, elcount(tmp), "%d", AT_TOLERANCE[index]);
3218            strncat(writeBuffer, tmp, AT_MAX_LINE_CHARS);
3219            strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
3220            for(i=0; i< getValue(AT_STEPS); ++i)
3221            {
3222                snprintf(tmp, elcount(tmp), "%d", AT_VALUES[index][i]);
3223                strncat(writeBuffer, tmp, AT_MAX_LINE_CHARS);
3224                if(i != getValue(AT_STEPS)-1) strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
3225            }
3226            strncat(writeBuffer, EOL, AT_MAX_LINE_CHARS);
3227            filePutString(writeBuffer, strlen(writeBuffer), writeHandle);
3228        }
3229
3230    write("Closing AT File");
```

```
    fileClose(writeHandle);
  }
  else
  {
    write("AT file cannot be opened");
  }
}

on envVar AT_Save
{
  if(getValue(this))
  {
    AT_SaveConfig();
  }
}

on envVar AT_Reset
{
  if(getValue(this))
  {
       AT_ResetConfig();
  }
}

on envVar AT_Record
{
  if(getValue(this))
  {
       AT_RecordState();
  }
}

AT_RecordState ()
{
    long i;

    putValue(AT_NVARS ,NVARS);
    for(i=0; i<getValue(AT_NVARS); ++i)
    {
        AT_TYPES[i] = TYPES[i];
        strncpy(AT_NAMES[i], NAMES[i], strlen(NAMES[i])+1);
        if(AT_TYPES[i] == 0) AT_TOLERANCE[i] = 0;
        else if(AT_TYPES[i] == 1) AT_TOLERANCE[i] = 10;
        else if(AT_TYPES[i] == 2) AT_TOLERANCE[i] = 0;
        else if(AT_TYPES[i] == 3) AT_TOLERANCE[i] = 0;
        else if(AT_TYPES[i] == 4) AT_TOLERANCE[i] = 50;
        else if(AT_TYPES[i] == 5) AT_TOLERANCE[i] = 0;
        AT_VALUES[i][getValue(AT_STEPS)] = VALUES[i];
    }
    putValue(AT_STEPS, getValue(AT_STEPS)+1);
}

on envVar AT_Play
{
  if(getValue(this))
  {
       writeClear(1);  // Clear Write window (1=>CAPL)
```

```
        Auto_Test = 1;
3288    //putValue(SystId,1);
3289
3290    putValue(BSS_Read,1);
3291
3292    //LINErrors_Inicial=getValue(VAR_ErrCountLIN);
3293
3294    state = 2;      // AT MODE
3295    AT_State = 0;   // Write Outputs
3296    AT_Mode = 0;    // Playing
3297    AT_IncMode = 1; // Increasing
3298    AT_SeqIndex = 0;
3299    AT_SeqState = 0;
3300    AT_OneStep = 0;
3301    Auto_Test = 1;
3302    putValue(AT_Step, 0);
3303  }
3304  else
3305  {
3306    state = 0;      // NORMAL MODE
3307  }
3308}
3309
3310AT_SetOutputState (WORD AT_Step)
3311{
3312  long i;
3313
3314  for(i=0; i<getValue(AT_NVARS); ++i)
3315  {
3316
3317
3318
3319
3320
3321
3322    if(TYPES[i]==2 || TYPES[i]==3)
3323  {
3324    VALUES[i]=AT_VALUES[i][AT_Step];
3325    CHANGE[i]=1;
3326    writeIndex=i;
3327  }
3328  }
3329  CopyOutputsToDataBase();
3330    //msgBeep(4);
3331}
3332
3333on timer AT_Delay
3334{
3335  SendTesterPresent();
3336  AT_State=2;
3337}
3338
3339AT_Scheduler ()
3340{
3341  long i;
3342
3343  // UPDATE REFRESH RATE COUNTER
3344
```

```
  timeStop = timeNow();
3345  if(timeStart != 0) putvalue(VAR_RefreshRate, (timeStop-timeStart)/100);
3346  timeStart = timeNow();
3347
3348    switch(AT_State)
3349    {
3350        case 0:
3351        {
3352            // AUTO-TEST: WRITE OUTPUTS
3353            //DbgMsg("AT WRITE");
3354            //write("1-%lf",(timenow()/100000.0));
3355            if(AT_Mode==0) AT_SetOutputState(getValue(AT_Step));    //
3356  Playing
                else if(AT_Mode==1) AT_SeqActivation();                 //
3357  Recording
                UpdateTimer();
3358             AT_State=1;
3359             break;
3360         }
3361         case 1:
3362         {
3363             // AUTO-TEST: WAIT FOR DIAGNOSTICS STABILISATION
3364             //DbgMsg("AT WAIT");
3365             /*if((getValue(AT_Step)%2)==0)
3366             {
3367                 settimer(AT_Delay, getValue(AT_Interval));
3368             }
3369             else
3370             {
3371                 settimer(AT_Delay, getValue(AT_Interval));
3372             }*/
3373             //write("2-%lf",(timeNow()/100000.0));
3374             settimer(AT_Delay, getValue(AT_Interval));
3375             sendCounter = 0;
3376             //AT_State=2;
3377             // If it is playing and no read and check is requested
3378             if(getValue(AT_ReadAndCheck) == 0 & AT_Mode == 0)
3379             {
3380                 AT_State=0;
3381                 AT_UpdateStep();
3382                 if(AT_OneStep == 1) // FOR STEP FW & STEP BW
3383                 {
3384                     state = 0;
3385                     AT_OneStep = 0;
3386                 }
3387             }
3388             break;
3389         }
3390         case 2:
3391         {
3392             // AUTO-TEST: READ INPUTS & DIAGNOSTICS
3393             //if(sendCounter == 0) DbgMsg("AT READ");
3394       //write("3-%lf",(timeNow()/100000.0));
3395             SendReadDataById(DID_LIST[sendCounter]);
3396       IncSendCounter();
3397       if(sendCounter==0)  // LAST SIGNAL WAS READ
3398             {
3399
```

```
                AT_State=0;
3400            if(AT_Mode == 0)        // PLAYING
3401            {
3402
3403                //BUS CAN connecting back
3404                /*if((arrayError[144][1] < arrayError[144][0])&&
3405 getValue(Ciclat)>1)
                    {
3406                write("recuperem la comunicació en AT_scheduler");
3407                setTimer(FailureCAN,250);
3408                }*/
3409
3410
3411                CopyInputsAndDiagsToDataBase();
3412
3413                if (getValue(StateCycle) && startaux)
3414                {
3415                 AT_CheckValues(getValue(AT_Step));
3416                }
3417
3418                if (getValue(CALIBRATION_MODE))
3419                {
3420                 AT_CheckValues(getValue(AT_Step));
3421                }
3422
3423            }
3424            else if(AT_Mode == 1)   // RECORDING
3425            {
3426                if(AT_SeqState == 1)    // After Output Activation
3427                {
3428                    for(i=0; i<getValue(AT_NVARS); ++i)
3429                        AT_VALUES[i][getValue(AT_STEPS)] = VALUES[i];
3430                    putValue(AT_STEPS, getValue(AT_STEPS)+1);
3431                }
3432
3433                if(AT_SeqIndex == AT_NUM_OUTPUTS)
3434                {
3435                    putValue(AT_Auto, 0);   // Finish Recording
3436                }
3437            }
3438            // UPDATE AT_Step
3439            if(AT_Mode == 0)    // PLAYING
3440            {
3441                AT_UpdateStep();
3442                if(AT_OneStep == 1) // FOR STEP FW & STEP BW
3443                {
3444                    state = 0;
3445                    AT_OneStep = 0;
3446                }
3447            }
3448            else                // RECORDING
3449            {
3450                putValue(AT_Step, AT_STEPS);
3451            }
3452        }
3453        break;
3454    }
3455
```

```
    }
}

DbgMsg (char msg[])
{
    long timeArray[9];
    char str[1024];
    getLocalTime(timeArray);
    snprintf(str, elcount(str), "%d:%d:%d %s", timeArray[2], timeArray[1], timeArray[0], msg);
    write(str);
}

AT_CheckValues (WORD AT_Step)
{
    int contaux;

    long i;
    WORD minVal, maxVal;
    BYTE hasErrors;
    char msgString[AT_MAX_LINE_CHARS];
    char tmpString[AT_MAX_LINE_CHARS];
    long timeArray[9];
    char EOL[2];
    EOL[0] = 0x0A; EOL[1] = 0x00;
    contaux=0;

    hasErrors = 0;
    getLocalTime(timeArray);


    //snprintf(msgString, elcount(msgString), "%d:%d:%d STEP NUMBER %d", timeArray[2], timeArray[1], timeArray[0], AT_Step);
    //strncat(msgString, EOL, AT_MAX_LINE_CHARS);

    for(i=0; i<getValue(AT_NVARS); ++i)
    {

                    AT_VALIDATION_RESULTS[i][AT_Step] = 1;  // Not Applicable

                    if( AT_CheckVariant(i) )
                    {
                        AT_VALIDATION_RESULTS[i][AT_Step] = 2;  // OK
                        eflag[i]=0;

                        // Calculate MIN and MAX values
                        if(AT_VALUES[i][AT_Step] < AT_TOLERANCE[i])
                        minVal = 0;
                        else minVal = AT_VALUES[i][AT_Step] - AT_TOLERANCE[i];
                        maxVal = AT_VALUES[i][AT_Step] + AT_TOLERANCE[i];

                       //write("estem dintre del checkvalue");

                        // Check intervals
                        if(VALUES[i] < minVal | VALUES[i] > maxVal)

```

```
                        {
3508
3509                    eflag[i]=1;
3510
3511                    hasErrors = 1;
3512                    contaux++;
3513
3514                    /*snprintf(tmpString, elcount(tmpString), "-
3515 ERROR! VAR: %s / MIN: %d / MAX: %d / VAL: %d", AT_NAMES[i], minVal,
    maxVal, VALUES[i]);
                        strncat(msgString, tmpString, AT_MAX_LINE_CHARS);
3516                    strncat(msgString, EOL, AT_MAX_LINE_CHARS);*/
3517                    AT_VALIDATION_RESULTS[i][AT_Step] = 3;  // NOK
3518                    }
3519
3520                }
3521
3522
3523     // AD and DD just are displayed, Immunity cycle routine start on
3524 Cycle 1 in order to have a known state

3525     if(getValue(CYCLING_MODE))
3526     {
3527         if (TYPES[i]==4 || TYPES[i]==5)
3528         {
3529             if(getValue(Ciclat)>1) ImmunityCycle(i,AT_Step);
3530         }
3531     }
3532     if (getValue(CALIBRATION_MODE))
3533     {
3534
3535         Calibration(i,AT_Step);
3536
3537     }
3538
3539     AT_MEASURED_VALUES[i][AT_Step] = VALUES[i];
3540
3541     }
3542     putValue(Env_ErrorCounter,contaux);
3543     //write("contadoraux %d",contaux);
3544     //write("ErrorCounter %d",errorCounter);
3545     contaux=0;
3546
3547
3548     //BSS Value
3549     AT_MEASURED_VALUES[i+1][AT_Step] = getValue(EAlternatorLoadD);
3550     //LIN Communication
3551     AT_MEASURED_VALUES[i+2][AT_Step] = getValue(LIN_Status);
3552     AT_MEASURED_VALUES[i+3][AT_Step] = getValue(VAR_ErrCountLIN);
3553
3554     if(hasErrors) write(msgString);
3555
3556     if((getValue(VAR_ErrCountLIN)-LINErrors_Inicial)>0)
3557     {
3558         snprintf(tmpString, elcount(tmpString), "- LIN ERRORS! NUM: %d"
3559 , (getValue(VAR_ErrCountLIN)-LINErrors_Inicial));
         write(tmpString);
3560
```

```
    }

    if(getValue(LIN_Status)!=0)
    {
        snprintf(tmpString, elcount(tmpString), "- NOT LIN COMMUNICATION
");
        write(tmpString);
    }

    LINErrors_Inicial=getValue(VAR_ErrCountLIN);
}

on envVar AT_Auto
{
  if(getValue(this))
  {
        AT_GenOutputList();
        AT_DefaultConfig();
        state = 2;      // AT MODE
        AT_State = 0;   // Write Outputs
        AT_Mode = 1;    // Recording
        AT_IncMode = 1; // Increasing
        AT_SeqIndex = 0;
        AT_SeqState = 0;
        AT_OneStep = 0;
        putValue(AT_Step, 0);
    }
    else
    {
        state = 0;      // NORMAL MODE
    }
}

AT_SeqActivation ()
{
    long index;

  switch(AT_SeqState)
  {
    // ACTIVATION
    case 0:
    {
            index = AT_OUT_INDEX[AT_SeqIndex];

      // Only if the output is inactive
      if(VALUES[index] == 0)
      {
        // Activate output
          if(NBITS[index]==1)
            VALUES[index]=0x01;
          else if(NBITS[index]==8)
            VALUES[index]=0xFF;
          else if(NBITS[index]==16)
            VALUES[index]=0xFFFF;

        CHANGE[index] = 1;
        writeIndex = index;
```

```
3617                CopyOutputsToDataBase();
3618                //msgBeep(4);
3619
3620        AT_SeqState = 1;
3621
3622                putValue(AT_Message, NAMES[index]);
3623      }
3624      else
3625      {
3626        // Go to next output
3627                AT_SeqIndex++;
3628                //if(AT_SeqIndex==AT_NUM_OUTPUTS) AT_SeqIndex=0;
3629      }
3630      break;
3631    }
3632    // DEACTIVATION
3633    case 1:
3634    {
3635            index = AT_OUT_INDEX[AT_SeqIndex];
3636
3637      // Deactivate output
3638      VALUES[index]=0;
3639      CHANGE[index] = 1;
3640      writeIndex = index;
3641      CopyOutputsToDataBase();
3642
3643      // Go to next output
3644            AT_SeqIndex++;
3645            //if(AT_SeqIndex==AT_NUM_OUTPUTS) AT_SeqIndex=0;
3646
3647      AT_SeqState = 0;
3648      break;
3649    }
3650  }
3651}
3652
3653AT_GenOutputList ()
3654{
3655    long i;
3656
3657    AT_NUM_OUTPUTS = 0;
3658    for(i=0; i<NVARS; ++i)
3659    {
3660        if(TYPES[i] == 2 | TYPES[i] == 3)
3661        {
3662            AT_OUT_INDEX[AT_NUM_OUTPUTS] = i;
3663            AT_NUM_OUTPUTS++;
3664        }
3665    }
3666    AT_SeqIndex = 0;
3667    write("Number of Outputs used: %d", AT_NUM_OUTPUTS);
3668}
3669
3670AT_ResetConfig ()
3671{
3672    long i, j;
3673
```

```
    for(i=0; i<VARS_SIZE; ++i)
    {
        AT_TYPES[i] = 0;
        for(j=0; j<MAX_VAR_LENGTH; ++j)
            AT_NAMES[i][j] = 0;
        AT_TOLERANCE[i] = 0;
        for(j=0; j<AT_MAX_STEPS; ++j)
        {
            AT_VALUES[i][j] = 0;
            AT_MEASURED_VALUES[i][j] = 0;
            AT_VALIDATION_RESULTS[i][j] = 0;
        }
    }
    putValue(AT_NVARS, 0);
    putValue(AT_STEPS, 0);
    putValue(AT_Step, 0);
}

AT_DefaultConfig ()
{
    long i,j;

    putValue(AT_NVARS, NVARS);
    for(i=0; i<getValue(AT_NVARS); ++i)
    {
        AT_TYPES[i] = TYPES[i];
        strncpy(AT_NAMES[i], NAMES[i], strlen(NAMES[i])+1);
        if(AT_TYPES[i] == 0) AT_TOLERANCE[i] = 0;
        else if(AT_TYPES[i] == 1) AT_TOLERANCE[i] = 10;
        else if(AT_TYPES[i] == 2) AT_TOLERANCE[i] = 0;
        else if(AT_TYPES[i] == 3) AT_TOLERANCE[i] = 0;
        else if(AT_TYPES[i] == 4) AT_TOLERANCE[i] = 50;
        else if(AT_TYPES[i] == 5) AT_TOLERANCE[i] = 0;
        for(j=0; j<AT_MAX_STEPS; ++j)
            AT_VALUES[i][j] = 0;
    }
    putValue(AT_STEPS, 0);
}

on envVar AT_FW
{
  if(getValue(this))
  {
        if(AT_IncMode == 0)
        {
            AT_IncMode = 1; // Increasing
            AT_UpdateStep();
            AT_UpdateStep();
        }
        state = 2;      // AT MODE
        AT_State = 0;   // Write Outputs
        AT_Mode = 0;    // Playing
        AT_SeqIndex = 0;
        AT_SeqState = 0;
        AT_OneStep = 1;
    }
}
```

```
on envVar AT_BW
{
  if(getValue(this))
  {
        if(AT_IncMode == 1)
        {
            AT_IncMode = 0; // Decreasing
            AT_UpdateStep();
            AT_UpdateStep();
        }
        state = 2;      // AT MODE
        AT_State = 0;   // Write Outputs
        AT_Mode = 0;    // Playing
        AT_SeqIndex = 0;
        AT_SeqState = 0;
        AT_OneStep = 1;
    }
}

AT_UpdateStep ()
{
    if(AT_IncMode == 1) // Increasing
    {


        if ((errorCounter<=0  && startaux && getValue(StateCycle)))
        {
            putValue(AT_Step, getValue(AT_Step)+1);
            putValue(STATE, getValue(STATE)+1);
        }

        if (getValue(CALIBRATION_MODE))
        {
            putValue(AT_Step, getValue(AT_Step)+1);
            putValue(STATE, getValue(STATE)+1);

            if(getValue(AT_Step) == getValue(AT_STEPS))   // Last Step
            {
            putValue(AT_Step,0);
            putValue(StateCycle,0);
            putValue(STATE,0);
            putValue(Ciclat,getValue(Ciclat)+1);

                if(getValue(Ciclat)>=4)
                {
                 CreateXMLCalibration();
                 write("genero el report de calibratge");
                 putValue(Ciclat,0);
                }
            }



        }

        //// Immunity Cycle.
```

```
        if(getValue(AT_Step) > getValue(AT_STEPS))   // Last Step
        {
            putValue(AT_Step,0);
            putValue(StateCycle,0);
            putValue(STATE,0);
            putValue(Ciclat,getValue(Ciclat)+1);
            if(getValue(AT_Continous) == 0) putValue(AT_Play, 0);
        }


    }
    else  // Decreasing
    {
        if(getValue(AT_Step) == 0)   // Last Step
        {
            putValue(AT_Step,getValue(AT_STEPS)-1);
            putValue(STATE, getValue(STATE)-1);
            if(getValue(AT_Continous) == 0) putValue(AT_Play, 0);
        }
        else
        {
            putValue(AT_Step, getValue(AT_Step)-1);
            putValue(STATE, getValue(STATE)-1);
        }
    }
}

AT_UpdateInpAndDiag ()
{
    // Update Inputs and Diagnostics from Validation Results

    long i, s;
    for(s=0; s<getValue(AT_STEPS); ++s)
    {
        for(i=0; i<getValue(AT_NVARS); ++i)
        {
            if( AT_TYPES[i] == 0 | AT_TYPES[i] == 1 | AT_TYPES[i] == 4
| AT_TYPES[i] == 5)
            {
                AT_VALUES[i][s] = AT_MEASURED_VALUES[i][s];
            }
        }
    }
}

on envVar AT_Update
{
  if(getValue(this))
  {
        AT_UpdateInpAndDiag();
    }
}

AT_SaveValidationReport ()
{
    dword writeHandle;
  char writeBuffer[AT_MAX_LINE_CHARS];

```

```
  char tmp[255];
3844    char EOL[2];
3845    char sep_str[2];
3846  long i, j, k;
3847  long index;
3848    char fileNameReport[255];
3849
3850    long timeArray[9];
3851    char timeStr[1024];
3852
3853    BYTE globalResult;
3854    BYTE stepResult[AT_MAX_STEPS];
3855
3856    writeHandle = 0;
3857    i = 0;
3858    j = 0;
3859    k = 0;
3860    index = 0;
3861    EOL[0] = 0x0A; EOL[1] = 0x00;
3862    sep_str[0] = SEP; sep_str[1] = 0x00;
3863
3864    // INITIALIZE GLOBAL RESULT AND STEP RESULT
3865    globalResult = 2; // OK
3866    for(i=0; i< getValue(AT_STEPS); ++i)
3867    {
3868        stepResult[i] = 2; // OK
3869    }
3870    for(i=0; i< getValue(AT_STEPS); ++i)
3871    {
3872        for(index=0; index<getValue(AT_NVARS); ++index)
3873        {
3874            if( AT_TYPES[index] == 0 | AT_TYPES[index] == 1 | AT_TYPES[
3875index] == 4 | AT_TYPES[index] == 5)
                {
3876                if(AT_VALIDATION_RESULTS[index][i] == 3) stepResult[i]
3877= 3;
                    if(AT_VALIDATION_RESULTS[index][i] == 0 & stepResult[i]
3878!= 3) stepResult[i] = 0;

3879                  if(AT_VALIDATION_RESULTS[index][i] == 3) globalResult =
3880 3;
                    if(AT_VALIDATION_RESULTS[index][i] == 0 & globalResult !
3881= 3) globalResult = 0;
                }
3882        }
3883    }
3884
3885    // FILE NAME
3886    getValue(VAR_SerialNumber, fileNameReport);
3887    getLocalTime(timeArray);
3888    switch(getValue(VAR_Variant))
3889    {
3890        case 0:
3891            snprintf(timeStr, elcount(timeStr), "%s", "L0");
3892        break;
3893    }
3894
3895
```

```
    if(globalResult==0)
    {
        snprintf(tmp, elcount(tmp), "_%d-%d-%d_%d.%d_NT.csv", timeArray[5]+1900, timeArray[4]+1, timeArray[3], timeArray[2], timeArray[1]);
    }
    else if(globalResult==2)
    {
        snprintf(tmp, elcount(tmp), "_%d-%d-%d_%d.%d_OK.csv", timeArray[5]+1900, timeArray[4]+1, timeArray[3], timeArray[2], timeArray[1]);
    }
    else if(globalResult==3)
    {
        snprintf(tmp, elcount(tmp), "_%d-%d-%d_%d.%d_NOK.csv", timeArray[5]+1900, timeArray[4]+1, timeArray[3], timeArray[2], timeArray[1]);
    }
    strncat(timeStr, tmp, 255);
    strncat(fileNameReport, timeStr, 255);

 writeHandle = openFileWrite (fileNameReport, 0);
 if (writeHandle != 0 )
 {
       // HEADER
       snprintf(writeBuffer, elcount(writeBuffer), "%s", "TYPE");
       strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
       strncat(writeBuffer, "NAME", AT_MAX_LINE_CHARS);
       strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
       strncat(writeBuffer, "TOL", AT_MAX_LINE_CHARS);
       strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
       for(i=0; i< getValue(AT_STEPS); ++i)
       {
           if(stepResult[i]==0)
               snprintf(tmp, elcount(tmp), "%d (NT)", i+1);
           if(stepResult[i]==2)
               snprintf(tmp, elcount(tmp), "%d", i+1);
           if(stepResult[i]==3)
               snprintf(tmp, elcount(tmp), "%d (NOK)", i+1);
           strncat(writeBuffer, tmp, AT_MAX_LINE_CHARS);
           if(i != getValue(AT_STEPS)-1) strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
       }
       strncat(writeBuffer, EOL, AT_MAX_LINE_CHARS);
       filePutString(writeBuffer, strlen(writeBuffer), writeHandle);

       // VARIABLES
       for(index=0; index<getValue(AT_NVARS); ++index)
       {
           switch(AT_TYPES[index])
           {
               case 0:
               {
                   snprintf(writeBuffer, elcount(writeBuffer), "%s", "DI");
                   break;
               }
               case 1:
               {
                   snprintf(writeBuffer, elcount(writeBuffer), "%s", "
```

```
AI");
                    break;
3948            }
3949            case 2:
3950            {
3951                snprintf(writeBuffer, elcount(writeBuffer), "%s", "
3952 DO");
                    break;
3953            }
3954            case 3:
3955            {
3956                snprintf(writeBuffer, elcount(writeBuffer), "%s", "
3957 PO");
                    break;
3958            }
3959            case 4:
3960            {
3961                snprintf(writeBuffer, elcount(writeBuffer), "%s", "
3962 AD");
                    break;
3963            }
3964            case 5:
3965            {
3966                snprintf(writeBuffer, elcount(writeBuffer), "%s", "
3967 DD");
                    break;
3968            }
3969        }
3970        strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
3971        strncat(writeBuffer, AT_NAMES[index], AT_MAX_LINE_CHARS);
3972        strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
3973        snprintf(tmp, elcount(tmp), "%d", AT_TOLERANCE[index]);
3974        strncat(writeBuffer, tmp, AT_MAX_LINE_CHARS);
3975        strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
3976        for(i=0; i< getValue(AT_STEPS); ++i)
3977        {
3978            if( AT_TYPES[index] == 0 | AT_TYPES[index] == 1 |
3979 AT_TYPES[index] == 4 | AT_TYPES[index] == 5)
            {
3980                if(AT_VALIDATION_RESULTS[index][i] == 0)    // Not
3981 Tested
                    snprintf(tmp, elcount(tmp), "NT");
3982                if(AT_VALIDATION_RESULTS[index][i] == 1)    // Not
3983 Applicable
                    snprintf(tmp, elcount(tmp), "NA");
3984                if(AT_VALIDATION_RESULTS[index][i] == 2)    // OK
3985                    snprintf(tmp, elcount(tmp), "%d",
3986 AT_MEASURED_VALUES[index][i]);
                if(AT_VALIDATION_RESULTS[index][i] == 3)    // NOK
3987                    snprintf(tmp, elcount(tmp), "%d (NOK)",
3988 AT_MEASURED_VALUES[index][i]);
            }
3989            else
3990                snprintf(tmp, elcount(tmp), "%d", AT_VALUES[index][i
3991 ]);

3992            strncat(writeBuffer, tmp, AT_MAX_LINE_CHARS);
3993
```

```
3994                if(i != getValue(AT_STEPS)-1) strncat(writeBuffer,
sep_str, AT_MAX_LINE_CHARS);
            }
3995            strncat(writeBuffer, EOL, AT_MAX_LINE_CHARS);
3996            filePutString(writeBuffer, strlen(writeBuffer), writeHandle);
3997        }
3998
3999        //BSS
4000        snprintf(writeBuffer, elcount(writeBuffer), "%s", "COM");
4001        strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
4002        strncat(writeBuffer, "BSS", AT_MAX_LINE_CHARS);
4003        strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
4004        snprintf(tmp, elcount(tmp), "%d", 0);
4005        strncat(writeBuffer, tmp, AT_MAX_LINE_CHARS);
4006        strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
4007        for(i=0; i< getValue(AT_STEPS); ++i)
4008        {
4009            snprintf(tmp, elcount(tmp), "%d", AT_MEASURED_VALUES[index+1
4010 ][i]);
            strncat(writeBuffer, tmp, AT_MAX_LINE_CHARS);
4011            if(i != getValue(AT_STEPS)-1) strncat(writeBuffer, sep_str,
4012 AT_MAX_LINE_CHARS);
        }
4013        strncat(writeBuffer, EOL, AT_MAX_LINE_CHARS);
4014        filePutString(writeBuffer, strlen(writeBuffer), writeHandle);
4015
4016        //LIN - State
4017        snprintf(writeBuffer, elcount(writeBuffer), "%s", "COM");
4018        strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
4019        strncat(writeBuffer, "LIN State", AT_MAX_LINE_CHARS);
4020        strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
4021        snprintf(tmp, elcount(tmp), "%d", 0);
4022        strncat(writeBuffer, tmp, AT_MAX_LINE_CHARS);
4023        strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
4024        for(i=0; i< getValue(AT_STEPS); ++i)
4025        {
4026            snprintf(tmp, elcount(tmp), "%d", AT_MEASURED_VALUES[index+2
4027 ][i]);
            strncat(writeBuffer, tmp, AT_MAX_LINE_CHARS);
4028            if(AT_MEASURED_VALUES[index+2][i]==2)
4029            {
4030                snprintf(tmp, elcount(tmp), "(NOK)");
4031                strncat(writeBuffer, tmp, AT_MAX_LINE_CHARS);
4032            }
4033            strncat(writeBuffer, tmp, AT_MAX_LINE_CHARS);
4034            if(i != getValue(AT_STEPS)-1) strncat(writeBuffer, sep_str,
4035 AT_MAX_LINE_CHARS);
        }
4036        strncat(writeBuffer, EOL, AT_MAX_LINE_CHARS);
4037        filePutString(writeBuffer, strlen(writeBuffer), writeHandle);
4038
4039        //LIN - Errors
4040        snprintf(writeBuffer, elcount(writeBuffer), "%s", "COM");
4041        strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
4042        strncat(writeBuffer, "LIN errors", AT_MAX_LINE_CHARS);
4043        strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
4044        snprintf(tmp, elcount(tmp), "%d", 0);
4045
```

```
        strncat(writeBuffer, tmp, AT_MAX_LINE_CHARS);
4046    strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
4047    for(i=0; i< getValue(AT_STEPS); ++i)
4048    {
4049        snprintf(tmp, elcount(tmp), "%d", AT_MEASURED_VALUES[index+3
4050 ][i]);
            strncat(writeBuffer, tmp, AT_MAX_LINE_CHARS);
4051        if(i != getValue(AT_STEPS)-1) strncat(writeBuffer, sep_str,
4052 AT_MAX_LINE_CHARS);
        }
4053    strncat(writeBuffer, EOL, AT_MAX_LINE_CHARS);
4054    filePutString(writeBuffer, strlen(writeBuffer), writeHandle);
4055    //*********************
4056
4057    //Traceability
4058    snprintf(writeBuffer, elcount(writeBuffer), "%s", "Traceability"
4059 );
        strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
4060    strncat(writeBuffer, "Soft.Num", AT_MAX_LINE_CHARS);
4061    strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
4062
4063    if(CheckingSwNum==1){snprintf(tmp, elcount(tmp), "%s", "OK");
4064 strncat(writeBuffer, tmp, AT_MAX_LINE_CHARS);}
        else if(CheckingSwNum==2){snprintf(tmp, elcount(tmp), "%s", "NOK
4065 ");strncat(writeBuffer, tmp, AT_MAX_LINE_CHARS);}
        CheckingSwNum=0;
4066    strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
4067
4068    snprintf(tmp, elcount(tmp), "%x", SwNum2);
4069    strncat(writeBuffer, tmp, AT_MAX_LINE_CHARS);
4070    snprintf(tmp, elcount(tmp), "%x", SwNum1);
4071    strncat(writeBuffer, tmp, AT_MAX_LINE_CHARS);
4072    strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
4073    snprintf(tmp, elcount(tmp), "%x", getValue(ID_Sw1_R));
4074    strncat(writeBuffer, tmp , AT_MAX_LINE_CHARS);
4075    snprintf(tmp, elcount(tmp), "%x", getValue(ID_Sw2_R));
4076    strncat(writeBuffer, tmp, AT_MAX_LINE_CHARS);
4077    strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
4078    strncat(writeBuffer, EOL, AT_MAX_LINE_CHARS);
4079    filePutString(writeBuffer, strlen(writeBuffer), writeHandle);
4080
4081    snprintf(writeBuffer, elcount(writeBuffer), "%s", "Traceability"
4082 );
        strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
4083    strncat(writeBuffer, "Edit.Num", AT_MAX_LINE_CHARS);
4084    strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
4085
4086    if(CheckingEditNum==1){snprintf(tmp, elcount(tmp), "%s", "OK");
4087 strncat(writeBuffer, tmp, AT_MAX_LINE_CHARS);}
        else if(CheckingEditNum==2){snprintf(tmp, elcount(tmp), "%s", "
4088 NOK");strncat(writeBuffer, tmp, AT_MAX_LINE_CHARS);}
        CheckingEditNum=0;
4089    strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
4090
4091    snprintf(tmp, elcount(tmp), "%x", EdNum2);
4092    strncat(writeBuffer, tmp, AT_MAX_LINE_CHARS);
4093    snprintf(tmp, elcount(tmp), "%x", EdNum1);
4094
```

```
4095        strncat(writeBuffer, tmp, AT_MAX_LINE_CHARS);
4096        strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
4097        snprintf(tmp, elcount(tmp), "%x", getValue(ID_Release1_R));
4098        strncat(writeBuffer, tmp, AT_MAX_LINE_CHARS);
4099        snprintf(tmp, elcount(tmp), "%x", getValue(ID_Release2_R));
4100        strncat(writeBuffer, tmp, AT_MAX_LINE_CHARS);
4101        strncat(writeBuffer, sep_str, AT_MAX_LINE_CHARS);
4102        strncat(writeBuffer, EOL, AT_MAX_LINE_CHARS);
4103        filePutString(writeBuffer, strlen(writeBuffer), writeHandle);
4104        //*********************
4105
4106    write("Validation Report Created");
4107    fileClose(writeHandle);
4108  }
4109  else
4110  {
4111    write("Validation Report Could not be Created");
4112  }
4113}
4114
4115on envVar AT_Report
4116{
4117  if(getValue(this))
4118  {
4119        AT_SaveValidationReport();
4120    }
4121}
4122
4123on envVar ReadBattDiscon
4124{
4125    if(getValue(this)==1)
4126    {
4127        putValue(VAR_RefreshInputs,0);
4128        putValue(VAR_RefreshDiags,0);
4129
4130        RoutineDataBuffer[0]=0x21;
4131    RoutineDataBuffer[1]=0x98;
4132    RoutineArraySize=2;
4133        settimer(Routine_T,200);
4134    }
4135}
4136
4137on envVar ResetBattDiscon
4138{
4139    if(getValue(this)==1)
4140    {
4141        putValue(VAR_RefreshInputs,0);
4142        putValue(VAR_RefreshDiags,0);
4143
4144        RoutineDataBuffer[0]=0x3B;
4145    RoutineDataBuffer[1]=0x98;
4146    RoutineDataBuffer[2]=0x00;
4147    RoutineArraySize=3;
4148        settimer(Routine_T,200);
4149    }
4150}
4151
```

```
on envVar Read_LossGND
{
    if(getValue(this)==1)
    {
        putValue(VAR_RefreshInputs,0);
        putValue(VAR_RefreshDiags,0);

        RoutineDataBuffer[0]=0x21;
    RoutineDataBuffer[1]=0x99;
    RoutineArraySize=2;
        settimer(Routine_T,200);
    }
}

ClearDTCs ()
{
  putValue(DTC1,0);
  putValue(DTC2_1,0);
  putValue(DTC2_2,0);
  putValue(DTC2_3,0);
  putValue(DTC2_4,0);
  putValue(DTC4,0);
  putValue(DTC5_1,0);
  putValue(DTC5_2,0);
  putValue(DTC6_1,0);
  putValue(DTC6_2,0);
  putValue(DTC7_1,0);
  putValue(DTC7_2,0);
  putValue(DTC7_3,0);
  putValue(DTC8_1,0);
  putValue(DTC8_2,0);
  putValue(DTC8_3,0);
  putValue(DTC9_1,0);
  putValue(DTC9_2,0);
  putValue(DTC9_3,0);
  putValue(DTC10_1,0);
  putValue(DTC10_2,0);
  putValue(DTC11_1,0);
  putValue(DTC11_2,0);
  putValue(DTC11_3,0);
  putValue(DTC12_1,0);
  putValue(DTC12_2,0);
  putValue(DTC12_3,0);
  putValue(DTC13_1,0);
  putValue(DTC13_2,0);
  putValue(DTC13_3,0);
  putValue(DTC14,0);
  putValue(DTC15_1,0);
  putValue(DTC15_2,0);
  putValue(DTC17,0);
  putValue(DTC18,0);
  putValue(DTC19_1,0);
    putValue(DTC19_2,0);
  putValue(DTC20_1,0);
  putValue(DTC20_2,0);
  putValue(DTC21_1,0);
  putValue(DTC21_2,0);

```

```
putValue(DTC22_1,0);
putValue(DTC22_2,0);
putValue(DTC23_1,0);
putValue(DTC23_2,0);
putValue(DTC24_1,0);
  putValue(DTC24_2,0);
putValue(DTC25,0);
putValue(DTC26,0);
putValue(DTC27_1,0);
putValue(DTC27_2,0);
putValue(DTC27_3,0);
putValue(DTC28_1,0);
putValue(DTC28_2,0);
putValue(DTC28_3,0);
putValue(DTC29_1,0);
putValue(DTC29_2,0);
putValue(DTC29_3,0);
putValue(DTC30_1,0);
putValue(DTC30_2,0);
putValue(DTC31,0);
putValue(DTC32_1,0);
putValue(DTC32_2,0);
putValue(DTC33_1,0);
  putValue(DTC33_2,0);
  putValue(DTC33_3,0);
  putValue(DTC33_4,0);
putValue(DTC34_1,0);
putValue(DTC34_2,0);
putValue(DTC34_3,0);
putValue(DTC35_1,0);
putValue(DTC35_2,0);
putValue(DTC35_3,0);
putValue(DTC36_1,0);
putValue(DTC36_2,0);
putValue(DTC36_3,0);
putValue(DTC37_1,0);
putValue(DTC37_2,0);
putValue(DTC38,0);
putValue(DTC39,0);
putValue(DTC40_1,0);
putValue(DTC40_2,0);
putValue(DTC41_1,0);
  putValue(DTC41_2,0);
putValue(DTC42,0);
putValue(DTC43_1,0);
  putValue(DTC43_2,0);

  putValue(DTC44_1,0);
putValue(DTC44_2,0);

putValue(DTC45_1,0);
putValue(DTC45_2,0);
putValue(DTC45_3,0);
  putValue(DTC46_1,0);
putValue(DTC46_2,0);
putValue(DTC47,0);
putValue(DTC48_1,0);

```

```
    putValue(DTC48_2,0);
    putValue(DTC48_3,0);
    putValue(DTC48_4,0);
    putValue(DTC48_5,0);
    putValue(DTC48_6,0);
    putValue(DTC48_7,0);
    putValue(DTC49_1,0);
    putValue(DTC49_2,0);
    putValue(DTC49_3,0);
    putValue(DTC49_4,0);
    putValue(DTC49_5,0);
    putValue(DTC49_6,0);
    putValue(DTC49_7,0);
  putValue(DTC50_1,0);
  putValue(DTC50_2,0);
  putValue(DTC51_1,0);
    putValue(DTC51_2,0);
  putValue(DTC52_1,0);
    putValue(DTC52_2,0);
    putValue(DTC52_3,0);
  putValue(DTC53_1,0);
    putValue(DTC53_2,0);
    putValue(DTC54_1,0);
    putValue(DTC54_2,0);
    putValue(DTC54_3,0);
}

ReportDTC(byte RxBuffer[],int length)
{
  int n;

  for(n=3;n<length;n=n+4)
  {
    if(RxBuffer[n]==0x92)
    {
      switch(RxBuffer[n+1])
      {
        case 0x0E:
          if(RxBuffer[n+2]==0x04) putValue(DTC1,1);
        break;

                case 0x0F:
          if(RxBuffer[n+2]==0x16) putValue(DTC33_1,1);
          if(RxBuffer[n+2]==0x17) putValue(DTC33_2,1);
          if(RxBuffer[n+2]==0x81) putValue(DTC33_3,1);
                    if(RxBuffer[n+2]==0x96) putValue(DTC33_4,1);
                    if(RxBuffer[n+2]==0x64) putValue(DTC33_5,1);
        break;

        case 0x11:
          if(RxBuffer[n+2]==0x14) putValue(DTC4,1);
        break;

        case 0x12:
          if(RxBuffer[n+2]==0x11) putValue(DTC5_1,1);
          if(RxBuffer[n+2]==0x15) putValue(DTC5_2,1);
                    if(RxBuffer[n+2]==0x12) putValue(DTC5_3,1);

```

```
        break;
4323
4324
        case 0x13:
4325      if(RxBuffer[n+2]==0x11) putValue(DTC6_1,1);
4326      if(RxBuffer[n+2]==0x15) putValue(DTC6_2,1);
4327                if(RxBuffer[n+2]==0x12) putValue(DTC6_3,1);
4328    break;
4329
4330    case 0x14:
4331      if(RxBuffer[n+2]==0x11) putValue(DTC7_1,1);
4332      if(RxBuffer[n+2]==0x15) putValue(DTC7_2,1);
4333                if(RxBuffer[n+2]==0x12) putValue(DTC7_3,1);
4334    break;
4335
4336    case 0x15:
4337      if(RxBuffer[n+2]==0x11) putValue(DTC8_1,1);
4338      if(RxBuffer[n+2]==0x15) putValue(DTC8_2,1);
4339                if(RxBuffer[n+2]==0x12) putValue(DTC8_3,1);
4340    break;
4341
4342    case 0x16:
4343      if(RxBuffer[n+2]==0x11) putValue(DTC9_1,1);
4344      if(RxBuffer[n+2]==0x12) putValue(DTC9_2,1);
4345      if(RxBuffer[n+2]==0x13) putValue(DTC9_3,1);
4346    break;
4347
4348    case 0x17:
4349      if(RxBuffer[n+2]==0x11) putValue(DTC10_1,1);
4350      if(RxBuffer[n+2]==0x12) putValue(DTC10_2,1);
4351    break;
4352
4353    case 0x18:
4354      if(RxBuffer[n+2]==0x11) putValue(DTC11_1,1);
4355      if(RxBuffer[n+2]==0x15) putValue(DTC11_2,1);
4356                if(RxBuffer[n+2]==0x12) putValue(DTC11_3,1);
4357    break;
4358
4359    case 0x19:
4360      if(RxBuffer[n+2]==0x11) putValue(DTC12_1,1);
4361      if(RxBuffer[n+2]==0x15) putValue(DTC12_2,1);
4362                if(RxBuffer[n+2]==0x12) putValue(DTC12_3,1)
4363;
    break;
4364
4365    case 0x1A:
4366      if(RxBuffer[n+2]==0x11) putValue(DTC13_1,1);
4367      if(RxBuffer[n+2]==0x15) putValue(DTC13_2,1);
4368                if(RxBuffer[n+2]==0x12) putValue(DTC13_3,1)
4369;
    break;
4370
4371    case 0x1C:
4372      if(RxBuffer[n+2]==0x11) putValue(DTC14,1);
4373    break;
4374
4375    case 0x1D:
4376      if(RxBuffer[n+2]==0x11) putValue(DTC15_1,1);
4377
```

```
                if(RxBuffer[n+2]==0x64) putValue(DTC15_2,1);
      break;

      case 0x1E:
        if(RxBuffer[n+2]==0x11) putValue(DTC16_1,1);
        if(RxBuffer[n+2]==0x12) putValue(DTC16_2,1);
      break;

      case 0x1F:
        if(RxBuffer[n+2]==0x12) putValue(DTC18,1);
      break;

      case 0x20:
        if(RxBuffer[n+2]==0x11) putValue(DTC19_1,1);
                if(RxBuffer[n+2]==0x12) putValue(DTC19_2,1);
      break;

      case 0x21:
        if(RxBuffer[n+2]==0x11) putValue(DTC20_1,1);
        if(RxBuffer[n+2]==0x12) putValue(DTC20_2,1);
      break;

      case 0x22:
        if(RxBuffer[n+2]==0x11) putValue(DTC21_1,1);
        if(RxBuffer[n+2]==0x12) putValue(DTC21_2,1);
      break;

      case 0x23:
        if(RxBuffer[n+2]==0x11) putValue(DTC22_1,1);
        if(RxBuffer[n+2]==0x12) putValue(DTC22_2,1);
      break;

      case 0x24:
        if(RxBuffer[n+2]==0x11) putValue(DTC23_1,1);
        if(RxBuffer[n+2]==0x12) putValue(DTC23_2,1);
      break;

      case 0x26:
        if(RxBuffer[n+2]==0x11) putValue(DTC25,1);
      break;

      case 0x27:
        if(RxBuffer[n+2]==0x11) putValue(DTC26,1);
      break;

      case 0x29:
        if(RxBuffer[n+2]==0x11) putValue(DTC27_1,1);
        if(RxBuffer[n+2]==0x12) putValue(DTC27_2,1);
        if(RxBuffer[n+2]==0x13) putValue(DTC27_3,1);
      break;

      case 0x2A:
        if(RxBuffer[n+2]==0x11) putValue(DTC28_1,1);
        if(RxBuffer[n+2]==0x12) putValue(DTC28_2,1);
        if(RxBuffer[n+2]==0x13) putValue(DTC28_3,1);
      break;

```

```
        case 0x2C:
4435      if(RxBuffer[n+2]==0x12) putValue(DTC30_1,1);
4436      if(RxBuffer[n+2]==0x64) putValue(DTC30_2,1);
4437    break;
4438
4439    case 0x2D:
4440      if(RxBuffer[n+2]==0x15) putValue(DTC31,1);
4441    break;
4442
4443           case 0x2E:
4444      if(RxBuffer[n+2]==0x12) putValue(DTC36_1,1);
4445                if(RxBuffer[n+2]==0x64) putValue(DTC36_2,1);
4446    break;
4447
4448           case 0x2F:
4449      if(RxBuffer[n+2]==0x12) putValue(DTC44_1,1);
4450                if(RxBuffer[n+2]==0x64) putValue(DTC44_2,1);
4451    break;
4452
4453    case 0x30:
4454      if(RxBuffer[n+2]==0x11) putValue(DTC34_1,1);
4455      if(RxBuffer[n+2]==0x12) putValue(DTC34_2,1);
4456      if(RxBuffer[n+2]==0x13) putValue(DTC34_3,1);
4457                if(RxBuffer[n+2]==0x92) putValue(DTC34_4,1);
4458    break;
4459
4460    case 0x31:
4461      if(RxBuffer[n+2]==0x11) putValue(DTC35_1,1);
4462      if(RxBuffer[n+2]==0x15) putValue(DTC35_2,1);
4463                if(RxBuffer[n+2]==0x12) putValue(DTC35_3,1)
4464 ;
    break;
4465
4466    case 0x33:
4467      if(RxBuffer[n+2]==0x12) putValue(DTC37_1,1);
4468      if(RxBuffer[n+2]==0x14) putValue(DTC37_2,1);
4469    break;
4470
4471    case 0x35:
4472      if(RxBuffer[n+2]==0x14) putValue(DTC38,1);
4473    break;
4474
4475    case 0x36:
4476      if(RxBuffer[n+2]==0x14) putValue(DTC39,1);
4477    break;
4478
4479    case 0x37:
4480      if(RxBuffer[n+2]==0x12) putValue(DTC40_1,1);
4481      if(RxBuffer[n+2]==0x14) putValue(DTC40_2,1);
4482                if(RxBuffer[n+2]==0x97) putValue(DTC40_3,1);
4483    break;
4484
4485    case 0x38:
4486      if(RxBuffer[n+2]==0x12) putValue(DTC41_1,1);
4487      if(RxBuffer[n+2]==0x14) putValue(DTC41_2,1);
4488    break;
4489
4490
```

```
case 0x39:
  if(RxBuffer[n+2]==0x97) putValue(DTC42,1);
            if(RxBuffer[n+2]==0x64) putValue(DTC42_2,1);
break;

        case 0x3A:
  if(RxBuffer[n+2]==0x55) putValue(DTC55_1,1);
            if(RxBuffer[n+2]==0x92) putValue(DTC55_2,1);
            if(RxBuffer[n+2]==0x67) putValue(DTC55_3,1);
break;

case 0x40:
  if(RxBuffer[n+2]==0x12) putValue(DTC43_1,1);
  if(RxBuffer[n+2]==0x14) putValue(DTC43_2,1);
break;

        case 0x41:
  if(RxBuffer[n+2]==0x11) putValue(DTC2_1,1);
  if(RxBuffer[n+2]==0x15) putValue(DTC2_2,1);
            if(RxBuffer[n+2]==0x12) putValue(DTC2_3,1);
break;

        case 0x52:
  if(RxBuffer[n+2]==0x13) putValue(DTC3,1);
break;

        case 0x55:
  if(RxBuffer[n+2]==0x13) putValue(DTC17,1);
break;

        case 0x56:
  if(RxBuffer[n+2]==0x11) putValue(DTC24_1,1);
  if(RxBuffer[n+2]==0x15) putValue(DTC24_2,1);
            if(RxBuffer[n+2]==0x12) putValue(DTC24_3,1);
break;

        case 0x57:
  if(RxBuffer[n+2]==0x11) putValue(DTC29_1,1);
  if(RxBuffer[n+2]==0x15) putValue(DTC29_2,1);
            if(RxBuffer[n+2]==0x12) putValue(DTC29_3,1);
break;

        case 0x58:
  if(RxBuffer[n+2]==0x11) putValue(DTC32_1,1);
  if(RxBuffer[n+2]==0x15) putValue(DTC32_2,1);
            if(RxBuffer[n+2]==0x12) putValue(DTC32_3,1);
break;

        case 0x60:
  if(RxBuffer[n+2]==0x11) putValue(DTC45_1,1);
  if(RxBuffer[n+2]==0x15) putValue(DTC45_2,1);
            if(RxBuffer[n+2]==0x15) putValue(DTC45_3,1);
break;

        case 0x61:
  if(RxBuffer[n+2]==0x11) putValue(DTC46_1,1);
  if(RxBuffer[n+2]==0x15) putValue(DTC46_2,1)
```

```
;
        break;
4548
4549            case 0x62:
4550      if(RxBuffer[n+2]==0x97) putValue(DTC47,1);
4551    break;
4552
4553            case 0x63:
4554      if(RxBuffer[n+2]==0x11) putValue(DTC48_1,1);
4555      if(RxBuffer[n+2]==0x12) putValue(DTC48_2,1);
4556                if(RxBuffer[n+2]==0x62) putValue(DTC48_3,1);
4557                if(RxBuffer[n+2]==0x64) putValue(DTC48_4,1);
4558      if(RxBuffer[n+2]==0x81) putValue(DTC48_5,1);
4559                if(RxBuffer[n+2]==0x92) putValue(DTC48_6,1);
4560                if(RxBuffer[n+2]==0x96) putValue(DTC48_7,1);
4561    break;
4562
4563            case 0x64:
4564      if(RxBuffer[n+2]==0x11) putValue(DTC49_1,1);
4565      if(RxBuffer[n+2]==0x12) putValue(DTC49_2,1);
4566                if(RxBuffer[n+2]==0x62) putValue(DTC49_3,1);
4567                if(RxBuffer[n+2]==0x64) putValue(DTC49_4,1);
4568      if(RxBuffer[n+2]==0x81) putValue(DTC49_5,1);
4569                if(RxBuffer[n+2]==0x92) putValue(DTC49_6,1);
4570                if(RxBuffer[n+2]==0x96) putValue(DTC49_7,1);
4571    break;
4572
4573            case 0x65:
4574      if(RxBuffer[n+2]==0x11) putValue(DTC50_1,1);
4575      if(RxBuffer[n+2]==0x12) putValue(DTC50_2,1)
4576;
        break;
4577
4578            case 0x66:
4579      if(RxBuffer[n+2]==0x81) putValue(DTC51_1,1);
4580      if(RxBuffer[n+2]==0x96) putValue(DTC51_2,1)
4581;
        break;
4582
4583            case 0x67:
4584      if(RxBuffer[n+2]==0x55) putValue(DTC52_1,1);
4585      if(RxBuffer[n+2]==0x67) putValue(DTC52_2,1)
4586;
                if(RxBuffer[n+2]==0x68) putValue(DTC52_2,1)
4587;
                if(RxBuffer[n+2]==0x41) putValue(DTC52_4,1);
4588            if(RxBuffer[n+2]==0x64) putValue(DTC52_5,1);
4589    break;
4590
4591            case 0x68:
4592      if(RxBuffer[n+2]==0x55) putValue(DTC53_1,1);
4593      if(RxBuffer[n+2]==0x68) putValue(DTC53_2,1)
4594;
                if(RxBuffer[n+2]==0x64) putValue(DTC53_3,1)
4595;
        break;
4596
4597
```

```
                case 0x69:
4598              if(RxBuffer[n+2]==0x11) putValue(DTC54_1,1);
4599              if(RxBuffer[n+2]==0x12) putValue(DTC54_2,1)
4600 ;
                                  if(RxBuffer[n+2]==0x13) putValue(DTC54_3,1);
4601            break;
4602
4603                    case 0x70:
4604              if(RxBuffer[n+2]==0x67) putValue(DTC56_1,1);
4605            break;
4606
4607                    case 0x71:
4608              if(RxBuffer[n+2]==0x1D) putValue(DTC57_1,1);
4609              if(RxBuffer[n+2]==0x92) putValue(DTC57_2,1)
4610 ;
                                  if(RxBuffer[n+2]==0x98) putValue(DTC57_3,1);
4611            break;
4612
4613                    case 0x72:
4614              if(RxBuffer[n+2]==0x64) putValue(DTC58_1,1);
4615            break;
4616
4617          }
4618        }
4619      }
4620 }
4621
4622 on envVar ReadDTC
4623 {
4624   if(getValue(this)==1)
4625   {
4626     putValue(ReadDTC,0);
4627     putValue(VAR_RefreshInputs,0);
4628         putValue(VAR_RefreshDiags,0);
4629
4630         RoutineDataBuffer[0]=0x19;
4631     RoutineDataBuffer[1]=0x02;
4632     RoutineDataBuffer[2]=0xEF;
4633     RoutineArraySize=3;
4634         settimer(Routine_T,200);
4635     DTRL_DTC=0;
4636   }
4637 }
4638
4639 on envVar ClearDTC
4640 {
4641   if(getValue(this)==1)
4642   {
4643     putValue(ClearDTC,0);
4644     putValue(VAR_RefreshInputs,0);
4645         putValue(VAR_RefreshDiags,0);
4646
4647         RoutineDataBuffer[0]=0x14;
4648     RoutineDataBuffer[1]=0xFF;
4649     RoutineDataBuffer[2]=0xFF;
4650         RoutineDataBuffer[3]=0xFF;
4651     RoutineArraySize=4;
4652
```

```
        settimer(Routine_T,200);
  }
}

on envVar ExtWatchdog_Ctrl
{
    if(getValue(this)==1)
  {
    putValue(VAR_RefreshInputs,0);
        putValue(VAR_RefreshDiags,0);

        RoutineDataBuffer[0]=0x3B;
    RoutineDataBuffer[1]=0x97;
    RoutineDataBuffer[2]=0x00;
    RoutineArraySize=3;
        settimer(Routine_T,200);
  }
}

on timer Routine_T
{
    for(x=0;x<RoutineArraySize;x++)
    {
        gTxDataBuffer[x]=RoutineDataBuffer[x];
    }
    txArraySize=RoutineArraySize;
    OSEKTL_DataReq(gTxDataBuffer, txArraySize);

    putValue(VAR_RefreshInputs,1);
    putValue(VAR_RefreshDiags,1);
}

on envVar Env_SBCReleaseSafeInactiveHigh
{
    if(getValue(this)==1)
    {
        putValue(VAR_RefreshInputs,0);
        putValue(VAR_RefreshDiags,0);

        RoutineDataBuffer[0]=0x3B;
    RoutineDataBuffer[1]=0x96;
    RoutineDataBuffer[2]=0x01;
    RoutineArraySize=3;
        settimer(Routine_T,200);
    }
}

on envVar Env_SBCForceSafeActiveLow
{
    if(getValue(this)==1)
    {
        putValue(this,0);
        putValue(VAR_RefreshInputs,0);
        putValue(VAR_RefreshDiags,0);

        RoutineDataBuffer[0]=0x3B;
    RoutineDataBuffer[1]=0x96;
```

```
    RoutineDataBuffer[2]=0x00;
    RoutineArraySize=3;
        settimer(Routine_T,200);
    }
}

char interpreteAscii(byte asciicode)
{
  Asciifound=0;
  strncpy(DiagInfoASCII_," ",3);
  switch(asciicode)
  {
    case 0x30: //--
             strncpy(DiagInfoASCII_,"0",3);
            Asciifound=1; break;
    case 0x31: //--
               strncpy(DiagInfoASCII_, "1",3);
        Asciifound=1; break;
    case 0x32: //--
               strncpy(DiagInfoASCII_, "2",3);
        Asciifound=1; break;
    case 0x33: //--
               strncpy(DiagInfoASCII_, "3",3);
        Asciifound=1; break;
    case 0x34: //--
               strncpy(DiagInfoASCII_, "4",3);
        Asciifound=1; break;
    case 0x35: //--
               strncpy(DiagInfoASCII_, "5",3);
        Asciifound=1; break;
    case 0x36: //--
               strncpy(DiagInfoASCII_, "6",3);
        Asciifound=1; break;
    case 0x37: //--
               strncpy(DiagInfoASCII_, "7",3);
        Asciifound=1; break;
    case 0x38: //--
               strncpy(DiagInfoASCII_, "8",3);
        Asciifound=1; break;
    case 0x39: //--
               strncpy(DiagInfoASCII_, "9",3);
        Asciifound=1; break;
    case 0x41: //--
               strncpy(DiagInfoASCII_, "A",3);
        Asciifound=1; break;
    case 0x42: //--
               strncpy(DiagInfoASCII_, "B",3);
        Asciifound=1; break;
    case 0x43: //--
               strncpy(DiagInfoASCII_, "C",3);
        Asciifound=1; break;
    case 0x44: //--
               strncpy(DiagInfoASCII_, "D",3);
        Asciifound=1; break;
    case 0x45: //--
               strncpy(DiagInfoASCII_, "E",3);
        Asciifound=1; break;
```

```
case 0x46: //--
4767            strncpy(DiagInfoASCII_, "F",3);
4768        Asciifound=1; break;
4769    case 0x47: //--
4770            strncpy(DiagInfoASCII_, "G",3);
4771        Asciifound=1; break;
4772    case 0x48: //--
4773            strncpy(DiagInfoASCII_, "H",3);
4774        Asciifound=1; break;
4775    case 0x49: //--
4776            strncpy(DiagInfoASCII_, "I",3);
4777        Asciifound=1; break;
4778    case 0x4A: //--
4779            strncpy(DiagInfoASCII_, "J",3);
4780        Asciifound=1; break;
4781    case 0x4B: //--
4782            strncpy(DiagInfoASCII_, "K",3);
4783        Asciifound=1; break;
4784    case 0x4C: //--
4785            strncpy(DiagInfoASCII_, "L",3);
4786        Asciifound=1; break;
4787    case 0x4D: //--
4788            strncpy(DiagInfoASCII_, "M",3);
4789        Asciifound=1; break;
4790    case 0x4E: //--
4791            strncpy(DiagInfoASCII_, "N",3);
4792        Asciifound=1; break;
4793    case 0x4F: //--
4794            strncpy(DiagInfoASCII_, "O",3);
4795        Asciifound=1; break;
4796    case 0x50: //--
4797            strncpy(DiagInfoASCII_, "P",3);
4798        Asciifound=1; break;
4799    case 0x51: //--
4800            strncpy(DiagInfoASCII_, "Q",3);
4801        Asciifound=1; break;
4802    case 0x52: //--
4803            strncpy(DiagInfoASCII_, "R",3);
4804        Asciifound=1; break;
4805    case 0x53: //--
4806            strncpy(DiagInfoASCII_, "S",3);
4807        Asciifound=1; break;
4808    case 0x54: //--
4809            strncpy(DiagInfoASCII_, "T",3);
4810        Asciifound=1; break;
4811    case 0x55: //--
4812            strncpy(DiagInfoASCII_, "U",3);
4813        Asciifound=1; break;
4814    case 0x56: //--
4815            strncpy(DiagInfoASCII_, "V",3);
4816        Asciifound=1; break;
4817    case 0x57: //--
4818            strncpy(DiagInfoASCII_, "W",3);
4819        Asciifound=1; break;
4820    case 0x58: //--
4821            strncpy(DiagInfoASCII_, "X",3);
4822        Asciifound=1; break;
4823
```

```
  case 0x59: //--
4824              strncpy(DiagInfoASCII_, "Y",3);
4825      Asciifound=1; break;
4826  case 0x5A: //--
4827              strncpy(DiagInfoASCII_, "Z",3);
4828      Asciifound=1; break;
4829 case 0x61: //--
4830              strncpy(DiagInfoASCII_, "a",3);
4831      Asciifound=1; break;
4832  case 0x62: //--
4833              strncpy(DiagInfoASCII_, "b",3);
4834      Asciifound=1; break;
4835  case 0x63: //--
4836              strncpy(DiagInfoASCII_, "c",3);
4837      Asciifound=1; break;
4838  case 0x64: //--
4839              strncpy(DiagInfoASCII_, "d",3);
4840      Asciifound=1; break;
4841  case 0x65: //--
4842              strncpy(DiagInfoASCII_, "e",3);
4843      Asciifound=1; break;
4844  case 0x66: //--
4845              strncpy(DiagInfoASCII_, "f",3);
4846      Asciifound=1; break;
4847  case 0x67: //--
4848              strncpy(DiagInfoASCII_, "g",3);
4849      Asciifound=1; break;
4850  case 0x68: //--
4851              strncpy(DiagInfoASCII_, "h",3);
4852      Asciifound=1; break;
4853  case 0x69: //--
4854              strncpy(DiagInfoASCII_, "i",3);
4855      Asciifound=1; break;
4856  case 0x6A: //--
4857              strncpy(DiagInfoASCII_, "j",3);
4858      Asciifound=1; break;
4859  case 0x6B: //--
4860              strncpy(DiagInfoASCII_, "k",3);
4861      Asciifound=1; break;
4862  case 0x6C: //--
4863              strncpy(DiagInfoASCII_, "l",3);
4864      Asciifound=1; break;
4865  case 0x6D: //--
4866              strncpy(DiagInfoASCII_, "m",3);
4867      Asciifound=1; break;
4868  case 0x6E: //--
4869              strncpy(DiagInfoASCII_, "n",3);
4870      Asciifound=1; break;
4871  case 0x6F: //--
4872              strncpy(DiagInfoASCII_, "o",3);
4873      Asciifound=1; break;
4874  case 0x70: //--
4875              strncpy(DiagInfoASCII_, "p",3);
4876      Asciifound=1; break;
4877  case 0x71: //--
4878              strncpy(DiagInfoASCII_, "q",3);
4879      Asciifound=1; break;
4880
```

```
    case 0x72: //--
4881                strncpy(DiagInfoASCII_, "r",3);
4882        Asciifound=1; break;
4883    case 0x73: //--
4884                strncpy(DiagInfoASCII_, "s",3);
4885        Asciifound=1; break;
4886    case 0x74: //--
4887                strncpy(DiagInfoASCII_, "t",3);
4888        Asciifound=1; break;
4889    case 0x75: //--
4890                strncpy(DiagInfoASCII_, "u",3);
4891        Asciifound=1; break;
4892    case 0x76: //--
4893                strncpy(DiagInfoASCII_, "v",3);
4894        Asciifound=1; break;
4895    case 0x77: //--
4896                strncpy(DiagInfoASCII_, "w",3);
4897        Asciifound=1; break;
4898    case 0x78: //--
4899                strncpy(DiagInfoASCII_, "x",3);
4900        Asciifound=1; break;
4901    case 0x79: //--
4902                strncpy(DiagInfoASCII_, "y",3);
4903        Asciifound=1; break;
4904    case 0x7A: //--
4905                strncpy(DiagInfoASCII_, "z",3);
4906        Asciifound=1; break;
4907  default: //--
4908                strncpy(DiagInfoASCII_, "?",3);
4909                Asciifound=1;
4910                break;
4911   }
4912   return(DiagInfoASCII_[0]);
4913}
4914
4915on envVar SystId
4916{
4917  if(getValue(this)==1)
4918    {
4919        putValue(this,0);
4920      putValue(VAR_RefreshInputs,0);
4921        putValue(VAR_RefreshDiags,0);
4922
4923        RoutineDataBuffer[0]=0x21;
4924        RoutineDataBuffer[1]=0x80;
4925      RoutineArraySize=2;
4926        settimer(Routine_T,200);
4927    }
4928}
4929
4930WriteCfgParam ()
4931{
4932    gTxDataBuffer[0]=0x3B;
4933    gTxDataBuffer[1]=0x06;
4934
4935    for(x=0;x<21;x++)
4936    {
4937
```

```
        gTxDataBuffer[x+2]=Aux_DataBuffer[x];
    }

    if(inhibit==1)
    {
        gTxDataBuffer[7]=0xFF;
        gTxDataBuffer[8]=0xFF;
        gTxDataBuffer[9]=0xFF;
        gTxDataBuffer[10]=0xFF;
        gTxDataBuffer[11]=0xFF;
        gTxDataBuffer[12]=0xFF;
        gTxDataBuffer[13]=0xFF;
        gTxDataBuffer[14]=0xFF;
        gTxDataBuffer[15]=0xFF;
        gTxDataBuffer[16]=0xFF;
    }
    else if(inhibit==0)
    {
        gTxDataBuffer[7]=0x00;
        gTxDataBuffer[8]=0x00;
        gTxDataBuffer[9]=0x00;
        gTxDataBuffer[10]=0x00;
        gTxDataBuffer[11]=0x00;
        gTxDataBuffer[12]=0x00;
        gTxDataBuffer[13]=0x00;
        gTxDataBuffer[14]=0x00;
        gTxDataBuffer[15]=0x00;
        gTxDataBuffer[16]=0x00;
    }
    txArraySize=23;
    OSEKTL_DataReq(gTxDataBuffer, txArraySize);
}

on envVar Enable_DTCs
{
    if(getValue(this))
    {
        putValue(VAR_RefreshInputs,0);
        putValue(VAR_RefreshDiags,0);

        RoutineDataBuffer[0]=0x21;
    RoutineDataBuffer[1]=0x06;
    RoutineArraySize=2;
        settimer(Routine_T,200);
        inhibit=0;

        putvalue(Status_Enabled_DTCs,0);
        putvalue(Status_Inhibit_DTCs,0);
    }
}

on envVar Inhibit_DTCs
{
    if(getValue(this))
    {
        putValue(VAR_RefreshInputs,0);
        putValue(VAR_RefreshDiags,0);
```

```
4995            RoutineDataBuffer[0]=0x21;
4996        RoutineDataBuffer[1]=0x06;
4997        RoutineArraySize=2;
4998            settimer(Routine_T,200);
4999            inhibit=1;
5000
5001            putvalue(Status_Enabled_DTCs,0);
5002            putvalue(Status_Inhibit_DTCs,0);
5003        }
5004    }
5005
5006    on envVar SwReset
5007    {
5008        if(getValue(this))
5009        {
5010            putValue(this,0);
5011            putValue(VAR_RefreshInputs,0);
5012            putValue(VAR_RefreshDiags,0);
5013
5014            RoutineDataBuffer[0]=0x11;
5015        RoutineDataBuffer[1]=0x01;
5016        RoutineArraySize=2;
5017            settimer(Routine_T,200);
5018        }
5019    }
5020
5021    on timer TimerCount
5022    {
5023        settimer(TimerCount,1000);
5024        sec++;
5025        if(sec>59)
5026        {
5027            sec=0;
5028            min++;
5029        }
5030        if(min>59)
5031        {
5032            min=0;
5033            hour++;
5034        }
5035        putValue(TimeInSec,sec);
5036        putValue(TimeInMin,min);
5037        putValue(TimeInHour,hour);
5038
5039        if(getValue(EAlternatorLoadD)==0xFF)
5040        {
5041            putValue(BSS_Alarm,1);
5042            ReportError(getValue(EAlternatorLoadD),Voltage,hour,min,sec);
5043        }
5044        else{putValue(BSS_Alarm,0);}
5045    }
5046
5047    on timer VoltageScheduler
5048    {
5049        if(increase==1)
5050        {
5051
```

```
        Voltage = Voltage + getValue(Stepvolt);
    }
    else if(increase==0)
    {
        Voltage = Voltage - getValue(Stepvolt);
    }

    snprintf(aux, 6, "%.2f", Voltage);

    putValue(Source_Voltage,aux);

    if((Voltage==16.00)&&(increase==1))
    {
        increase=0;
    }
    else if((Voltage==8.00)&&(increase==0))
    {
        increase=1;
    }

    putValue(BSSCurrentVolt,Voltage);
    setTimer(VoltageScheduler,getValue(TimeStep));
}

on envVar StartBSSTest
{
    if(getValue(this))
    {
        increase = 1;
        Voltage = 8.00;
        snprintf(aux, 6, "%.2f", Voltage);

        putValue(Source_Voltage,aux);
        putValue(BSSCurrentVolt,Voltage);
        setTimer(VoltageScheduler,getValue(TimeStep));
        sec=0;
        min=0;
        hour=0;
        settimer(TimerCount,1000);
        glbHandle = OpenFileWrite ("ErrorReport_BSS_Test.txt",0);
    }
}

on envVar StopBSSTest
{
    if(getValue(this))
    {
        increase = 1;
        Voltage = 0.00;
        snprintf(aux, 6, "%.2f", Voltage);
        putValue(Source_Voltage,aux);

        putValue(BSSCurrentVolt,Voltage);
        cancelTimer(VoltageScheduler);
        canceltimer(TimerCount);
        fileClose (glbHandle);
    }
```

```
}

ReportError(int AltLoad,float Volt,int hh,int mm,int ss)
{
  if ( glbHandle!=0 )
  {
        snprintf (buffer,elcount(buffer),"
**********\n");
    filePutString (buffer, elcount(buffer),glbHandle);
        snprintf (buffer,elcount(buffer),"ERROR - BSS_Value: %ld,
Voltage: %.2f V, Time: %ld:%ld:%ld\n",AltLoad,Volt,hh,mm,ss);
    filePutString (buffer, elcount(buffer),glbHandle);
    }
    else write ("File 'ErrorReport_BSS_Test.txt' was not opened for
write access.");
}

Check_Traceability ()
{
    char tmp[255];

    if((getValue(ID_Sw1_R)==SwNum2)&&(getValue(ID_Sw2_R)==SwNum1))
    {
        CheckingSwNum=1;
    }
    else{CheckingSwNum=2;}

    if((getValue(ID_Release1_R)==EdNum2)&&(getValue(ID_Release2_R)
==EdNum1))
    {
        CheckingEditNum=1;
    }
    else{CheckingEditNum=2;}

    if((CheckingSwNum == 1)&&(CheckingEditNum == 1))    //Software &
Hardware Test OK
    {
        AT_Play(); //Start auto_test
    }
    else
    {
        /*Report Error on screen AT_Message*/
        snprintf(tmp, elcount(tmp), "%s", "Software Version not Correct"
);
        putValue(AT_Message, tmp);
    }
}

LoadSystemIdent ()
{
    int aux;

    strncpy(fileUsed,"Traceability.sid",32);

    aux=0;
    aux=getProfileInt("Traceability","Soft_Num",0,fileUsed);
    SwNum1=(aux & 0xFF);

```

```
    SwNum2=(((aux)>>8) & 0xFF);

    aux=getProfileInt("Traceability","Edit_Num",0,fileUsed);
    EdNum1=(aux & 0xFF);
    EdNum2=(((aux)>>8) & 0xFF);
}

AT_Play ()
{
    state = 2;      // AT MODE
    AT_State = 0;   // Write Outputs
    AT_Mode = 0;    // Playing
    AT_IncMode = 1; // Increasing
    AT_SeqIndex = 0;
    AT_SeqState = 0;
    AT_OneStep = 0;
    Auto_Test = 1;
    putValue(AT_Step, 0);
}

on envVar VAR036
{
    if(getValue(this)==1)
    {
        putValue(EStartingMode_BCM_R_,3);
    }
    else if(getValue(this)==0)
    {
        putValue(EStartingMode_BCM_R_,0);
    }
}

on envVar VAR_FastGoToSleep
{
    if(getValue(this)==1)
    {
        putValue(EBCM_WakeUpSleepCommand_,0);
    }
}

on envVar VAR_WakeUp
{
    if(getValue(this)==1)
    {
        putValue(EBCM_WakeUpSleepCommand_,3);
    }
}

on envVar Apply_Param
{
  if(getValue(this)==1)
  {
    putValue(VAR_RefreshInputs,0);
        putValue(VAR_RefreshDiags,0);
        putValue(WriteResult,0);

        gTxDataBuffer[0]=0x3B;
```

```
    gTxDataBuffer[1]=0x91;
    gTxDataBuffer[2]=getValue(Param_A);
    gTxDataBuffer[3]=getValue(Param_B1);
    gTxDataBuffer[4]=getValue(Param_B2);
    gTxDataBuffer[5]=getValue(Param_C);
    txArraySize=6;

    OSEKTL_DataReq(gTxDataBuffer, txArraySize);
  }
}

on envVar AC_Variant
{
    switch(getValue(this))
    {
        case 1:
            //Gold AC Valve
            putValue(Param_A, 10);
            putValue(Param_B1, 73);
            putValue(Param_B2, 112);
            putValue(Param_C, 100); //99.7138
        break;

        case 2:
            //Silver AC Valve
            putValue(Param_A, 30);
            putValue(Param_B1, 69);
            putValue(Param_B2, 136);
            putValue(Param_C, 115); //114.838
        break;

        case 3:
            //Black AC Valve
            putValue(Param_A, 0);
            putValue(Param_B1, 78);
            putValue(Param_B2, 32);
            putValue(Param_C, 75); //75.45
        break;

        default:
            //No AC Valve
            putValue(Param_A, 0);
            putValue(Param_B1, 0);
            putValue(Param_B2, 0);
            putValue(Param_C, 0);
        break;
    }
}

on envVar Read_AC
{
  if(getValue(this)==1)
  {
    putValue(VAR_RefreshInputs,0);
        putValue(VAR_RefreshDiags,0);

        gTxDataBuffer[0]=0x21;
```

```
    gTxDataBuffer[1]=0x91;
    txArraySize=2;

    OSEKTL_DataReq(gTxDataBuffer, txArraySize);
  }
}

on envVar Start_AC_Valve_Test
{
  if(getValue(this) == 1)
  {
    putValue(VAR_RefreshInputs,0);
        putValue(VAR_RefreshDiags,0);

        putValue(WriteResult,0);
      up=1;
        ACV_T=1;

        if(getValue(Choose_Method)==0)
        {
            AC_Valve_DC=getValue(AC_DutyC);
            Offset=getValue(AC_Offset);
        }
        else if(getValue(Choose_Method)==1)
        {
            AC_Valve_DC=getValue(Initialcurr);
            Offset=getValue(StepCurrent);

            putvalue(EMinimumVoltagebyAC_, 1);
            putvalue(EAC_StopAutoForbidden_, 1);
            putvalue(EPumpActivationRequest_, 1);
            putvalue(EACCompClutchRequest_, 1);
            putvalue(EACCompWorkingMode_, 1);
            putvalue(EFrontDefrostRequest_, 1);
            putValue(EACCompClutchActivation_,1);
        }

        TestingTime=(getValue(Time_Test));

        glbHandle = OpenFileWrite ("Values_ACValve_Test.txt",0);

        //Ignition ON
//        putValue(VAR046,0);
        putValue(VAR045,1);
        settimer(ReadT,200);
        setTimer(Test_Time,TestingTime);
        setTimer(timeT,1);
  }
}

on timer timeT  //comptador de temps
{
  ms++;
    if(ms>999)
    {
        s++;
        ms=0;
```

```
    }
  if(s>59)
  {
    s=0;
    m++;
  }
  putValue(min,m);
  putValue(sec,s);
  setTimer(timeT,1);
}

on timer Test_Time
{
  cancelTimer(ReadT);
    cancelTimer(Watchdog);

    if(up==1)
  {
     AC_Valve_DC = AC_Valve_DC + Offset;
  }
  else if(up==0)
  {
     AC_Valve_DC = AC_Valve_DC - Offset;
  }

    if(getValue(Choose_Method)==0)
    {
        //*********************
        //putValue(VAR054,AC_Valve_DC);
        gTxDataBuffer[0]=0x30;
      gTxDataBuffer[1]=0x21;
        gTxDataBuffer[2]=0xfb;
        gTxDataBuffer[3]=0xFF;
        gTxDataBuffer[4]=(AC_Valve_DC*2);

        txArraySize = 5;

        OSEKTL_DataReq(gTxDataBuffer, txArraySize);
        //***************************
  }
    else if(getValue(Choose_Method)==1)
    {
        putValue(EACCompValveRequest_,AC_Valve_DC);
    }

  if(AC_Valve_DC==100)
  {
      up=0;
  }
  else if(AC_Valve_DC==0)
  {
      up=1;
    }
  setTimer(Test_Time,TestingTime);
    settimer(ReadT,25);
    setTimer(StopRT,((TestingTime*1000)-100));
    settimer(Watchdog,((TestingTime*1000)-500));
```

```
    write("dc: %d",AC_Valve_DC);
}

on envVar StopACValveTest
{
  if(getValue(this) == 1)
  {
    putValue(VAR_RefreshInputs,0);
        putValue(VAR_RefreshDiags,0);

        ACV_T=0;
        cancelTimer(Test_Time);
        cancelTimer(timeT);

        //Ignition OFF
        putValue(VAR045,0);
        putValue(VAR046,1);
        putValue(VAR054,0);
        fileClose (glbHandle);

        putValue(EACCompValveRequest_,0);

        putvalue(EMinimumVoltagebyAC_, 0);
        putvalue(EAC_StopAutoForbidden_, 0);
        putvalue(EPumpActivationRequest_, 0);
        putvalue(EACCompClutchRequest_, 0);
        putvalue(EACCompWorkingMode_, 0);
        putvalue(EFrontDefrostRequest_, 0);
        putValue(EACCompClutchActivation_,0);
    }
}

on timer ReadT
{
    SendReadDataById(0x02);
    settimer(ReadT,200);
}

ReportAC(int min,int sec,int ms,int ACV_Value,int ACV_Diag)
{
  if ( glbHandle!=0 )
  {
        //snprintf (buffer,elcount(buffer),"
**********\n");
    //filePutString (buffer, elcount(buffer),glbHandle);
        snprintf (buffer,elcount(buffer),"Time: %ld:%ld:%ld; %ld; %ld\n"
,min,sec,ms,ACV_Value,ACV_Diag);
    filePutString (buffer, elcount(buffer),glbHandle);
    }
    else write ("File 'Values_ACValve_Test.txt' was not opened for
write access.");
}

on timer StopRT
{
    cancelTimer(ReadT);
}
```

```
on timer t_ControlLighting
{
    switch(CL_x)
    {
        case 1:
            //IGNITION ON
            putValue(EStartingMode_BCM_R_,2);   //L1,L2,L3 (USM2010)
            //putValue(EnvStartingMode_BCM_,2);   //L4 (X61)
            CL_x=2;
            settimer(t_ControlLighting,200);
        break;

        case 2:
            //Lights ON
            //putValue(EnvFrontFogLightsRequest_,1);   //High Beam
            putValue(EFrontFogLightsRequest_,1);
            //putValue(EnvLowBeamRequest_UCH__,1);   //Low Beam
            putValue(ELowBeamRequest_,1);
            CL_x=3;
            settimer(t_ControlLighting,1000);
        break;

        case 3:
            //Enable SPI Test
            gTxDataBuffer[0]=0x3B;
          gTxDataBuffer[1]=0x9F;
            txArraySize = 6;
            OSEKTL_DataReq(gTxDataBuffer, txArraySize);
        break;

        case 4:
            //Begins of SPI Test
            SPI_Check=1;
            gTxDataBuffer[0]=0x3B;
          gTxDataBuffer[1]=0x9F;
            gTxDataBuffer[2]=0x03;
          gTxDataBuffer[3]=0x01;
            txArraySize = 4;
            OSEKTL_DataReq(gTxDataBuffer, txArraySize);
            //settimer(t_ReadFeedbacks,200);
        break;

        case 5:
            //End of SPI Test
            //putValue(VAR_RefreshDiags,0);
            canceltimer(t_ReadFeedbacks);
            SPI_Check=0;
            gTxDataBuffer[0]=0x3B;
          gTxDataBuffer[1]=0x9F;
            gTxDataBuffer[2]=0x03;
          gTxDataBuffer[3]=0x02;
            txArraySize = 4;
            OSEKTL_DataReq(gTxDataBuffer, txArraySize);
            //Lights OFF
            //putValue(EnvFrontFogLightsRequest_,0); //High Beam
            putValue(EFrontFogLightsRequest_,0);

```

```
            //putValue(EnvLowBeamRequest_UCH__,0);   //Low Beam
            putValue(ELowBeamRequest_,0);
            //IGNITION OFF
            putValue(EStartingMode_BCM_R_,0);       //L1,L2,L3 (USM2010)
            //putValue(EnvStartingMode_BCM_,0);     //L4 (X61)
            CL_x=6;
            settimer(t_ControlLighting,1000);
        break;

        case 6:
            putValue(SwReset,1);
        break;
    }
}

on envVar Control_Lighting_Test
{
    if(getValue(this)==1)
    {
        putValue(VAR_RefreshInputs,0);
        putValue(VAR_RefreshDiags,0);
        CL_x=1;
        settimer(t_ControlLighting,200);
    }
}

on timer t_ReadFeedbacks
{
    SendReadDataById(0x02);
    settimer(t_ReadFeedbacks,30);
}

on envVar Run_DO_0_IOcab
{
    if(getValue(this)==1)
    {
        T_On = getValue(Ton);
        T_Off = getValue(Toff);
        putValue(RemainCycles,getValue(Num_Cicles));
        iter=1;
        settimer(t_PulseGenerator,100);
    }
    else if(getValue(this)==0)
    {
        iter=10;
        canceltimer(t_PulseGenerator);
    }
}

on timer t_PulseGenerator
{
    switch(iter)
    {
        case 1:
            ciclesCount++;
            if(ciclesCount>getValue(Num_Cicles))
            {

```

```
                putValue(Run_DO_0_IOcab,0);
            }
            else
            {
                putValue(DO_0_IOcab,1);
                iter=2;
                settimer(t_PulseGenerator,T_On);
            }
        break;

        case 2:
            putValue(DO_0_IOcab,0);
            iter=1;
            settimer(t_PulseGenerator,T_Off);
            putValue(RemainCycles,getValue(RemainCycles)-1);
        break;
    }
}

on envVar VAR_Variant
{
    long i;
    char varname[100];
    long colourY;
    long colourN;
    long colourM;

    colourY = makeRGB(255,255,255);
    colourN = makeRGB(255,100,100);

    switch(getValue(this))
    {
        case 0: // L0
        {
            for(i=0; i<NVARS; ++i)
            {
                if(i<10) snprintf(varname,elcount(varname), "STR00%d", i
);
                else if(i<100) snprintf(varname,elcount(varname), "STR0
%d", i);
                else snprintf(varname,elcount(varname), "STR%d", i);

                if(VARIANT_L0[i] == 0)        // Yes
                    setControlBackColor("X10 Functionality", varname,
colourY);
                else if(VARIANT_L0[i] == 1)   // No
                    setControlBackColor("X10 Functionality", varname,
colourN);
            }
            putValue(AT_File, "USM2010_L0.csv");
            snprintf(variant, elcount(variant), "%s", "L0");
            break;
        }

        default:    //None
        {
            for(i=0; i<NVARS; ++i)
```

```
            {
5609            if(i<10) snprintf(varname,elcount(varname), "STR00%d", i
5610);
                else if(i<100) snprintf(varname,elcount(varname), "STR0
5611%d", i);
                else snprintf(varname,elcount(varname), "STR%d", i);
5612
5613            setControlBackColor("X10 Functionality", varname,
5614colourY);
            }
5615        putValue(AT_File, "None");
5616        break;
5617      }
5618    }
5619}
5620
5621BYTE AT_CheckVariant (long index)
5622{
5623    // Checks if a signal is present in a variant
5624    BYTE variant;
5625    BYTE isOK;
5626
5627    variant = getValue(VAR_Variant);
5628    isOK = 1;
5629    if(variant == 0 & VARIANT_L0[index] != 0) isOK = 0;
5630
5631    return isOK;
5632}
5633
5634on envVar EMV_Frequenz
5635{
5636
5637    if(getValue(this)) PutValue(StateCycle,1);
5638
5639    putValue(EMV_Frequenz_aux,getValue(EMV_Frequenz)/1000000);
5640    controlSimulador=1;
5641
5642}
5643
5644on envVar EMV_Test_Start
5645{
5646
5647 if(getValue(this))
5648     {
5649           //Delay(2);
5650           startaux=1;
5651     }
5652}
5653
5654on envVar EMV_Test_Stop
5655{
5656
5657    SetTimer(DelayStop,100);
5658    //AT_Scheduler();
5659    Delay(6);
5660
5661}
5662
```

```
Delay (int timeD)
{
long i;

for(i=0; i<timeD*1e8; ++i)
    {

    }


}

on timer tflagA
{

    senueloA=1;

}

on timer DelayStop
{

    switch(indexStop)
      {
        case 0:
        {
          startaux=0;
                putValue(AT_Play,0);
          indexStop=1;
                setTimer(DelayStop,1000);
          break;
        }
           case 1:
        {
          putValue(VAR046,1);

          indexStop=2;
                setTimer(DelayStop,2000);
          break;
        }
            case 2:
        {
                if (getValue(CYCLING_MODE)) putValue(CYCLING_XML,1);
          indexStop=3;
                setTimer(DelayStop,3000);
          break;
        }


            case 3:
        {
           putValue(VAR_ALL_OFF,1);
                setTimer(DelayStop,1000);
          //break;
        }


```

```
5720        }
5721  }
5722
5723  CreateXMLCycling()
5724  {
5725      long i, j, z;
5726      long tm[9];
5727      dword glbHandle = 0;
5728      char timebuffer[255];
5729    char buffer[255];
5730      char fileNameAT[255]="ImmunityCycle";
5731      char fileFormatAT[255]=".xml";
5732
5733
5734          getLocalTime(tm);
5735
5736
5737
5738          ltoa(tm[3],timebuffer,10);
5739          strncat(fileNameAT,timebuffer,MAX_VAR_LENGTH);
5740          strncat(fileNameAT,"_",MAX_VAR_LENGTH);
5741
5742          ltoa((tm[4]+1),timebuffer,10);
5743          strncat(fileNameAT,timebuffer,MAX_VAR_LENGTH);
5744          strncat(fileNameAT,"_",MAX_VAR_LENGTH);
5745
5746          ltoa((tm[5]+1900),timebuffer,10);
5747          strncat(fileNameAT,timebuffer,MAX_VAR_LENGTH);
5748          strncat(fileNameAT,"_",MAX_VAR_LENGTH);
5749
5750          ltoa(tm[2],timebuffer,10);
5751          strncat(fileNameAT,timebuffer,MAX_VAR_LENGTH);
5752          strncat(fileNameAT,"_",MAX_VAR_LENGTH);
5753          //write("1%s",fileNameAT);
5754
5755          ltoa(tm[1],timebuffer,10);
5756          strncat(fileNameAT,timebuffer,MAX_VAR_LENGTH);
5757          strncat(fileNameAT,"_",MAX_VAR_LENGTH);
5758          //write("2%s",fileNameAT);
5759
5760          ltoa(tm[0],timebuffer,10);
5761          strncat(fileNameAT,timebuffer,MAX_VAR_LENGTH);
5762          strncat(fileNameAT,fileFormatAT,MAX_VAR_LENGTH);
5763          //write("3%s",fileNameAT);
5764
5765          glbHandle = OpenFileWrite (fileNameAT,0);
5766
5767          if ( glbHandle!=0 )
5768        {
5769
5770
5771              snprintf (buffer,elcount(buffer),"<?xml version='1.0'
5772  encoding='UTF-8'?>\n");
          filePutString (buffer, elcount(buffer),glbHandle);
5773
5774              snprintf (buffer,elcount(buffer),"<?xml-stylesheet type='
5775
```

```
text/xsl' href='ImmunityCycle.xsl'?>\n");
        filePutString (buffer, elcount(buffer),glbHandle);
5776
5777
            snprintf (buffer,elcount(buffer),"<ImmunityCycle>");
5778    filePutString (buffer, elcount(buffer),glbHandle);
5779
5780
            snprintf (buffer,elcount(buffer),"\n\t<header>\n\t<test
5781 -title>Lab Technician</test-title>\n\t<test-date>%d/%d/%d %d:%d:%d</test
-date>\n\t<test-project>USM2010_Functional_Panel v9.4</test-project>\n</
header>",(tm[3]),(tm[4]+1),(tm[5]+1900), tm[2],tm[1],tm[0]);
        filePutString (buffer, elcount(buffer),glbHandle);
5782
5783
5784
5785
5786            for (i=0;i<=getValue(Ciclat);i++)
5787             {
5788
5789                 for (j=0;j<=getValue(AT_STEPS);j++)
5790                 {
5791
5792                     for (z=0;z<getValue(AT_NVARS);z++)
5793                     {
5794                         if (ErrorImm[i][j][z].Error)
5795                         {
5796
5797                             if (ErrorImm[i][j][z].State== getValue(
5798 AT_STEPS) || z==144)
                             {
5799
5800                                 snprintf (buffer,elcount(buffer),"\n
5801 \t<error>");
                             filePutString (buffer, elcount(buffer),
5802 glbHandle);

5803                                 snprintf (buffer,elcount(buffer),"\n
5804 \t\t<frequency> %f </frequency>",ErrorImm[i][j][z].Frequenz);
                             filePutString (buffer, elcount(buffer),
5805 glbHandle);

5806                                 snprintf (buffer,elcount(buffer),"\n
5807 \t\t<Cycle> %d </Cycle>", ErrorImm[i][j][z].Ciclat);
                             filePutString (buffer, elcount(buffer),
5808 glbHandle);

5809                                 snprintf (buffer,elcount(buffer),"\n
5810 \t\t<State> %d </State>",ErrorImm[i][j][z].State);
                             filePutString (buffer, elcount(buffer),
5811 glbHandle);

5812                                 snprintf (buffer,elcount(buffer),"\n
5813 \t\t<Name> %s </Name>",ErrorImm[i][j][z].Names);
                             filePutString (buffer, elcount(buffer),
5814 glbHandle);

5815
5816
                                 snprintf (buffer,elcount(buffer),"\n
5817
```

```
\t\t<Min> %f </Min>",ErrorImm[i][j][z].minVal);
                                    filePutString (buffer, elcount(buffer),
5818 glbHandle);

5819                                    snprintf (buffer,elcount(buffer),"\n
5820 \t\t<Max> %f </Max>",ErrorImm[i][j][z].maxVal);
                                    filePutString (buffer, elcount(buffer),
5821 glbHandle);

5822                                    snprintf (buffer,elcount(buffer),"\n
5823 \t\t<Value> %f </Value>",ErrorImm[i][j][z].VALUE);
                                    filePutString (buffer, elcount(buffer),
5824 glbHandle);

5825
5826
5827
5828                                    if(ErrorImm[i][j][z].ImmunityField<
5829 = ErrorImm[i][j][z].Threshold)
                                    {
5830                                    snprintf (buffer,elcount(buffer),"\n
5831 \t\t<immunityfield> %f </immunityfield>",ErrorImm[i][j][z].Threshold);
                                    filePutString (buffer, elcount(buffer),
5832 glbHandle);

5833                                    snprintf (buffer,elcount(buffer),"\n
5834 \t\t<Threshold> %f </Threshold>",ErrorImm[i][j][z].Threshold);
                                    filePutString (buffer, elcount(
5835 buffer),glbHandle);

5836                                    }
5837
5838                                    else
5839                                    {
5840                                    snprintf (buffer,elcount(buffer),"\n
5841 \t\t<immunityfield> %f </immunityfield>",ErrorImm[i][j][z].ImmunityField
    );
                                    filePutString (buffer, elcount(buffer),
5842 glbHandle);

5843                                    snprintf (buffer,elcount(buffer),"\n
5844 \t\t<Threshold> %f </Threshold>",ErrorImm[i][j][z].Threshold);
                                    filePutString (buffer, elcount(
5845 buffer),glbHandle);
                                    }
5846
5847                                snprintf (buffer,elcount(buffer),"\n\t</
5848 error>");
                               filePutString (buffer, elcount(buffer),glbHandle
5849 );

5850                                    }
5851
5852
5853
5854
5855                                }
5856
```

```
                        }
                    }
                }


            snprintf (buffer,elcount(buffer),"\n</ImmunityCycle>\n");
            filePutString (buffer, elcount(buffer),glbHandle);

            FileClose(glbHandle);
            strncpy(fileNameAT,"ImmunityCycle",MAX_VAR_LENGTH);



    }









}
on timer DelayCalibration
{

char fileNameAT[255];
char buffer[255];

    switch(indexDelay)
    {
            case 0:
        {

                putValue(VAR_Start,1);
          indexDelay=1;
                setTimer(DelayCalibration,1000);
          break;
        }

            case 1:
        {

                putValue(AT_Continous,1);
          indexDelay=2;
                setTimer(DelayCalibration,1000);
          break;
        }




            case 2:
        {
```

```
5914                if(getValueSize(AT_File)>1)
5915                {
5916                //SetControlColors ("EMC32","DisplayFile",  MakeRGB(255,
5917 255,255), MakeRGB(0,255,0));
                    //putValueToControl("EMC32","DisplayFile"," File is
5918 loaded");
                    putValue(AT_Load,1);
5919                indexDelay=3;
5920                setTimer(DelayCalibration,1000);
5921          break;
5922                }
5923
5924                else
5925                {
5926                indexDelay=2;
5927                setTimer(DelayCalibration,1000);
5928                break;
5929                }
5930
5931
5932        }
5933            case 3:
5934            {
5935
5936                //putValue(AT_Load,1);
5937
5938                if(getValue(AT_NVARS)>0 && getValue(AT_STEPS)>0)
5939                {
5940                SetControlColors ("EMC32","DisplayFile",  MakeRGB(255,
5941 255,255), MakeRGB(0,255,0));
                    putValueToControl("EMC32","DisplayFile"," File is loaded
5942 ");
                    indexDelay=4;
5943                setTimer(DelayCalibration,1000);
5944          break;
5945                }
5946
5947                else
5948                {
5949                SetControlColors ("EMC32","DisplayFile",  MakeRGB(255,
5950 255,255), MakeRGB(255,0,0));
                    putValueToControl("EMC32","DisplayFile"," Incorrect File
5951 ");
                    indexDelay=2;
5952                setTimer(DelayCalibration,1000);
5953                break;
5954
5955                }
5956
5957
5958            }
5959            case 4:
5960            {
5961                putValue(AT_Play,1);
5962                break;
5963            }
5964
```

```
    }
5965
5966 }
5967
5968 on busOff
5969 {
5970     putValue(CAN_STATE_LED,2);
5971     /*putValue(CAN_State_Green,0);
5972     putValue(CAN_State_Red,1);*/
5973 }
5974
5975 on errorActive
5976 {
5977
5978     putValue(CAN_STATE_LED,1);
5979     putValue(CANSTATE,0);
5980     arrayError[144][0] = arrayError[144][1];
5981     arrayError[144][1] = getValue(CANSTATE);
5982     VALUES[144]=0;
5983
5984     // Show the error features to panell canoe.
5985
5986     if(getValue(CYCLING_MODE))
5987     {
5988
5989         if((arrayError[144][1] < arrayError[144][0]) && getValue(Ciclat)
5990 >1)
                 {
5991
5992              write("recuperem la comunicació");
5993              setTimer(FailureCAN,300);
5994
5995              ErrorImm[getValue(Ciclat)][getValue(STATE)][144].Ciclat=
5996 getValue(Ciclat);
              ErrorImm[getValue(Ciclat)][getValue(STATE)][144].State=
5997 getValue(STATE);
              strncpy(ErrorImm[getValue(Ciclat)][getValue(STATE)][144].
5998 Names,AT_NAMES[144],MAX_VAR_LENGTH);
              ErrorImm[getValue(Ciclat)][getValue(STATE)][144].Frequenz=
5999 getValue(EMV_Frequenz_aux);
              ErrorImm[getValue(Ciclat)][getValue(STATE)][144].
6000 ImmunityField= getValue(EMV_Feld);
              ErrorImm[getValue(Ciclat)][getValue(STATE)][144].Threshold
6001 =getValue(Threshold);
              ErrorImm[getValue(Ciclat)][getValue(STATE)][144].Error= 1;
6002
6003              SetControlColors ("EMC32","Display",  MakeRGB(255,255,255),
6004 MakeRGB(0,0,0));
              putValueToControl("EMC32","Display","\nCycle: ");
6005              putValueToControl("EMC32","Display",getValue(Ciclat),0);
6006              putValueToControl("EMC32","Display","        State: ");
6007              putValueToControl("EMC32","Display",getValue(STATE),0);
6008              putValueToControl("EMC32","Display","\nNAME: ");
6009              putValueToControl("EMC32","Display",AT_NAMES[144]);
6010              putValueToControl("EMC32","Display","\nFrequency: ");
6011              putValueToControl("EMC32","Display",getValue(
6012 EMV_Frequenz_aux),0);
```

```
            putValueToControl("EMC32","Display"," MHz");
            putValueToControl("EMC32","Display","\nImmunity Field: ");
            putValueToControl("EMC32","Display",getValue(EMV_Feld),0);
            putValueToControl("EMC32","Display"," V/m");
            putValueToControl("EMC32","Display","\n                    ");

            }
     }
}

on errorPassive
{
    putValue(CAN_STATE_LED,2);
    putValue(CANSTATE,1);
    arrayError[144][0] = arrayError[144][1];
    arrayError[144][1] = getValue(CANSTATE);
    VALUES[144]=1;




}
ImmunityCycle (int i ,WORD AT_Step)
{

 WORD minVal, maxVal;
 long j;


        //Initialitze New Step;
        if (AT_Step==0) AT_VALUES[i][getValue(AT_STEPS)]= AT_VALUES[i][AT_Step];



        // Detection error signal flag
        arrayError[i][0] = arrayError[i][1];
        arrayError[i][1] = eflag[i];

        // Display the checksum of monitorization signals
        mEMC.EMC1=errorCounter;
        output(mEMC);


        // Calculate MIN and MAX values
        if(AT_VALUES[i][AT_Step] < AT_TOLERANCE[i]) minVal = 0;
        else minVal = AT_VALUES[i][AT_Step] - AT_TOLERANCE[i];
        maxVal = AT_VALUES[i][AT_Step] + AT_TOLERANCE[i];

        if ((AT_VALIDATION_RESULTS[i][AT_Step] == 3))
        {
        findID(i,ErrorImm[0][0][i].Varid,AT_Step);
        }
```

```
6068                // This function just detect when a failure occurs but
6069 not when it dissapear.
                    if(( getValue(STATE)<(getValue(AT_STEPS))) && (
6070 AT_VALIDATION_RESULTS[i][AT_Step] == 3))
                    {
6071
6072
6073                ErrorImm[getValue(Ciclat)][getValue(STATE)][i].Error= 0;
6074                }
6075
6076
6077                //In case the failure don't disappear when the immunity
6078 field down.
                    //ArrayError 0 and 1, just show the signals with
6079 contain an error.

6080                if ((getValue(EMV_Feld)<getValue(Threshold)) && (
6081 arrayError[i][1] == 1) && (arrayError[i][0] == 1))
                    {
6082
6083                         ErrorImm[getValue(Ciclat)][getValue(STATE)][i].
6084 Ciclat= getValue(Ciclat);
                         ErrorImm[getValue(Ciclat)][getValue(STATE)][i].
6085 State= getValue(STATE);
                         ErrorImm[getValue(Ciclat)][getValue(STATE)][i].
6086 minVal= minVal;
                         ErrorImm[getValue(Ciclat)][getValue(STATE)][i].
6087 maxVal= maxVal;
                         ErrorImm[getValue(Ciclat)][getValue(STATE)][i].
6088 VALUE= VALUES[i];
                         strncpy(ErrorImm[getValue(Ciclat)][getValue(
6089 STATE)][i].Names,AT_NAMES[i],MAX_VAR_LENGTH);
                         ErrorImm[getValue(Ciclat)][getValue(STATE)][i].
6090 Frequenz=  getValue(EMV_Frequenz_aux);
                         ErrorImm[getValue(Ciclat)][getValue(STATE)][i].
6091 ImmunityField= getValue(Threshold);
                         ErrorImm[getValue(Ciclat)][getValue(STATE)][i].
6092 Threshold=getValue(Threshold);
                         ErrorImm[getValue(Ciclat)][getValue(STATE)][i].
6093 Error= 1;

6094                      if(getValue(STATE)==(getValue(AT_STEPS)))
6095                      {
6096
6097                      SetControlColors ("EMC32","Display",  MakeRGB(
6098 255,255,255), MakeRGB(0,0,0));
                         putValueToControl("EMC32","Display","\nCycle: ");
6099                      putValueToControl("EMC32","Display",getValue(
6100 Ciclat),0);
                         putValueToControl("EMC32","Display","
6101 State: ");
                         putValueToControl("EMC32","Display",getValue(
6102 STATE),0);
                         putValueToControl("EMC32","Display","\nNAME: ");
6103                      putValueToControl("EMC32","Display",AT_NAMES[i]);
6104                      putValueToControl("EMC32","Display","\nmin: ");
6105
```

```
                    putValueToControl("EMC32","Display",minVal,0);
6106                putValueToControl("EMC32","Display"," max: ");
6107                putValueToControl("EMC32","Display",maxVal,0);
6108                putValueToControl("EMC32","Display"," Value: ");
6109                putValueToControl("EMC32","Display",VALUES[i],0);
6110                putValueToControl("EMC32","Display","\nFrequency
6111 : ");
                    putValueToControl("EMC32","Display",getValue(
6112 EMV_Frequenz_aux),0);
                    putValueToControl("EMC32","Display"," MHz");
6113                putValueToControl("EMC32","Display","\nThe
6114 failure remains below ");
                    SetControlColors ("EMC32","Display",  MakeRGB(
6115 255,255,255), MakeRGB(255,0,0));
                    putValueToControl("EMC32","Display",ErrorImm[
6116 getValue(Ciclat)][getValue(STATE)][i].Threshold,0);
                    SetControlColors ("EMC32","Display",  MakeRGB(
6117 255,255,255), MakeRGB(0,0,0));
                    putValueToControl("EMC32","Display"," V/m");
6118                putValueToControl("EMC32","Display","\
6119 n                  ");
                    errorCounter=0;
6120                 arrayError[i][0]=0;
6121                 arrayError[i][1]=0;
6122
6123                 }
6124
6125             }
6126
6127             // provoquem un flag de baixada (un error desapareix
6128 quan baixem el camp)
                 if(i ==144)
6129              {
6130                  if (senueloC)
6131                  {
6132                  arrayError[i][1]=0;
6133                  write("tflag  ha sigut executat, %s", AT_NAMES[i]);
6134
6135                  }
6136              }
6137
6138              if(i ==80)
6139              {
6140                  if (senueloA)
6141                  {
6142                  arrayError[i][1]=0;
6143                  write("tflag  ha sigut executat, %s", AT_NAMES[i]);
6144
6145                  }
6146              }
6147
6148              if(i ==76)
6149              {
6150
6151                  if (senueloB)
6152                  {
6153                  arrayError[i][1]=1;
6154
```

```
                    write("tflag B ha sigut executat, %s", AT_NAMES[i]);
6155                }
6156            }
6157
6158            if (controlSimulador)
6159            {
6160            senueloA=0;
6161            senueloB=0;
6162            controlSimulador=0;
6163            }
6164
6165
6166            // flag down when a failure disappears.
6167
6168            if (arrayError[i][1] < arrayError[i][0])
6169            {
6170
6171                    ErrorImm[getValue(Ciclat)][getValue(STATE)][i].
6172Ciclat= getValue(Ciclat);
                        ErrorImm[getValue(Ciclat)][getValue(STATE)][i].
6173State= getValue(STATE);
                        ErrorImm[getValue(Ciclat)][getValue(STATE)][i].
6174minVal= minVal;
                        ErrorImm[getValue(Ciclat)][getValue(STATE)][i].
6175maxVal= maxVal;
                        ErrorImm[getValue(Ciclat)][getValue(STATE)][i].
6176VALUE= VALUES[i];
                        strncpy(ErrorImm[getValue(Ciclat)][getValue(
6177STATE)][i].Names,AT_NAMES[i],MAX_VAR_LENGTH);
                        ErrorImm[getValue(Ciclat)][getValue(STATE)][i].
6178Frequenz=  getValue(EMV_Frequenz_aux);
                        ErrorImm[getValue(Ciclat)][getValue(STATE)][i].
6179ImmunityField= getValue(EMV_Feld);
                        ErrorImm[getValue(Ciclat)][getValue(STATE)][i].
6180Threshold=getValue(Threshold);
                        ErrorImm[getValue(Ciclat)][getValue(STATE)][i].
6181Error= 1;

6182                        if(getValue(STATE)==(getValue(AT_STEPS)))
6183                        {
6184
6185                            SetControlColors ("EMC32","Display",
6186MakeRGB(255,255,255), MakeRGB(0,0,0));
                                putValueToControl("EMC32","Display","\
6187nCycle: ");
                                putValueToControl("EMC32","Display",
6188getValue(Ciclat),0);
                                putValueToControl("EMC32","Display",
6189"        State: ");
                                putValueToControl("EMC32","Display",
6190getValue(STATE),0);
                                putValueToControl("EMC32","Display","\
6191nNAME: ");
                                putValueToControl("EMC32","Display",
6192AT_NAMES[i]);
                                putValueToControl("EMC32","Display","\
6193nFrequency: ");
```

```
                              putValueToControl("EMC32","Display",
6194 getValue(EMV_Frequenz_aux),0);
                              putValueToControl("EMC32","Display","
6195 MHz");
                              putValueToControl("EMC32","Display","\
6196 nImmunity Field: ");
                              putValueToControl("EMC32","Display",
6197 getValue(EMV_Feld),0);
                              putValueToControl("EMC32","Display"," V/
6198 m");
                              putValueToControl("EMC32","Display","\
6199 n                        ");

6200
6201                          arrayError[i][0]=0;
6202                          arrayError[i][1]=0;
6203
6204                          write("Step %d",getValue(AT_Step));
6205
6206                          if (errorCounter>0) errorCounter--;
6207                     }
6208
6209               }
6210
6211                 // flag up when a new failure occurs
6212               if ((arrayError[i][1] > arrayError[i][0]))
6213               {
6214                     arrayError[i][0]=0;
6215                     arrayError[i][1]=0;
6216
6217                     if (getValue(STATE)==(getValue(AT_STEPS)))
6218                     {
6219                     arrayError[i][0]=1;
6220                     arrayError[i][1]=1;
6221                     errorCounter++;
6222                     }
6223
6224                }
6225
6226                     if (getValue(STATE)==(getValue(AT_STEPS)) && (
    getValue(EMV_Feld)==ErrorImm[getValue(Ciclat)][getValue(STATE)][i].
    Threshold))
6227                     {
6228                     errorCounter=0;
6229                     }
6230
6231 }
6232
6233 on key 'a'
6234 {
6235
       settimer(tflagA,750);
6236
6237 }
6238
6239 // Find the ID correspond to PO -> AD, AI
6240 // Find the ID correspond to DO -> DD
6241
6242
```

```
FindID (long index,long varIDaux, word AT_Step)
{

  long i;
  long IDfound;

    IDfound=0;

    for (i=0; i<getValue(AT_NVARS); i++)
    {

      if(TYPES[i] == 2 || TYPES[i] == 3) ///2: DO, 3: PO
      {
            if(varIDaux == ErrorImm[0][0][i].Varid)
            {
            //AT_VALIDATION_RESULTS[index][AT_Step] = 3;
            AT_VALUES[i][getValue(AT_STEPS)]= AT_VALUES[i][AT_Step];
            IDfound=1;
            }
      }

    }



     if(TYPES[index] == 4 || TYPES[index] == 5) ///4: AD, 5: DD
     {
            ;
            if (!IDfound) AT_VALUES[index][getValue(AT_STEPS)]=
AT_VALUES[index][AT_Step];
     }

      AT_VALIDATION_RESULTS[index][AT_Step] = 3;



}

on key 'b'
{
settimer(tflagB,750);
}

on timer tflagB
{

    senueloB=1;



}

Calibration (int i ,WORD AT_Step)
{

 WORD minVal, maxVal;

```

```
6299        // Detection Flag
6300        arrayError[i][0] = arrayError[i][1];
6301        arrayError[i][1] = eflag[i];
6302
6303
6304        // Calculate MIN and MAX values
6305        if(AT_VALUES[i][AT_Step] < AT_TOLERANCE[i]) minVal = 0;
6306        else minVal = AT_VALUES[i][AT_Step] - AT_TOLERANCE[i];
6307        maxVal = AT_VALUES[i][AT_Step] + AT_TOLERANCE[i];
6308
6309
6310
6311
6312
6313                // This function just detect when a failure occurs but
6314 not when it dissapear.
6315                if(( getValue(STATE)<(getValue(AT_STEPS))) && (
6316 AT_VALIDATION_RESULTS[i][AT_Step] == 3))
                {
6317
6318                        ErrorCalib[getValue(Ciclat)][getValue(STATE)][i]
6319 .Ciclat= getValue(Ciclat);
                        ErrorCalib[getValue(Ciclat)][getValue(STATE)][i]
6320 .State= getValue(STATE);
                        ErrorCalib[getValue(Ciclat)][getValue(STATE)][i]
6321 .minVal= minVal;
                        ErrorCalib[getValue(Ciclat)][getValue(STATE)][i]
6322 .maxVal= maxVal;
                        ErrorCalib[getValue(Ciclat)][getValue(STATE)][i]
6323 .VALUE= VALUES[i];
                        strncpy(ErrorCalib[getValue(Ciclat)][getValue(
6324 STATE)][i].Names,AT_NAMES[i],MAX_VAR_LENGTH);
                        ErrorCalib[getValue(Ciclat)][getValue(STATE)][i]
6325 .Error= 1;
6326
6327                        SetControlColors ("EMC32","DisplayCalib",
6328 MakeRGB(255,255,255), MakeRGB(0,0,0));
                        putValueToControl("EMC32","DisplayCalib","\
6329 nCycle: ");
                        putValueToControl("EMC32","DisplayCalib",
6330 getValue(Ciclat),0);
                        putValueToControl("EMC32","DisplayCalib",
6331 "        State: ");
                        putValueToControl("EMC32","DisplayCalib",
6332 getValue(STATE),0);
                        putValueToControl("EMC32","DisplayCalib","\nNAME
6333 : ");
                        putValueToControl("EMC32","DisplayCalib",
6334 AT_NAMES[i]);
                        putValueToControl("EMC32","DisplayCalib","\nmin
6335 : ");
                        putValueToControl("EMC32","DisplayCalib",minVal,
6336 0);
                        putValueToControl("EMC32","DisplayCalib"," max:
6337
```

```
");
                        putValueToControl("EMC32","DisplayCalib",maxVal,
0);
                        putValueToControl("EMC32","DisplayCalib"," Value
: ");
                        putValueToControl("EMC32","DisplayCalib",VALUES[
i],0);
                        putValueToControl("EMC32","DisplayCalib","\
n                          ");


                }
}

CreateXMLCalibration()
{
    long i, j, z;
    long tm[9];
    dword glbHandle = 0;
  char buffer[255];


        glbHandle = OpenFileWrite ("Calibration.xml",0);

        if ( glbHandle!=0 )
      {

            getLocalTime(tm);
            snprintf (buffer,elcount(buffer),"<?xml version='1.0'
encoding='UTF-8'?>\n");
        filePutString (buffer, elcount(buffer),glbHandle);

            snprintf (buffer,elcount(buffer),"<?xml-stylesheet type='
text/xsl' href='Calibration.xsl'?>\n");
        filePutString (buffer, elcount(buffer),glbHandle);

            snprintf (buffer,elcount(buffer),"<ImmunityCycle>");
        filePutString (buffer, elcount(buffer),glbHandle);

            snprintf (buffer,elcount(buffer),"\n\t<header>\n\t<test
-title>Lab Technician</test-title>\n\t<test-date>%d/%d/%d %d:%d:%d</test
-date>\n\t<test-project>USM2010_Functional_Panel v9.4</test-project>\n</
header>",(tm[3]),(tm[4]+1),(tm[5]+1900), tm[2],tm[1],tm[0]);
        filePutString (buffer, elcount(buffer),glbHandle);


              for (i=0;i<=getValue(Ciclat);i++)
               {

                   for (j=0;j<=getValue(AT_STEPS);j++)
                   {

                       for (z=0;z<getValue(AT_NVARS);z++)
                       {
```

```
6385                            if (ErrorCalib[i][j][z].Error)
6386                            {
6387
6388
6389                                snprintf (buffer,elcount(buffer),"\n\t<
6390 error>");
                                filePutString (buffer, elcount(buffer),
6391 glbHandle);

6392                                snprintf (buffer,elcount(buffer),"\n\t\t
6393 <frequency> %f </frequency>",ErrorCalib[i][j][z].Frequenz);
                                filePutString (buffer, elcount(buffer),
6394 glbHandle);

6395                                snprintf (buffer,elcount(buffer),"\n\t\t
6396 <Cycle> %d </Cycle>", ErrorCalib[i][j][z].Ciclat);
                                filePutString (buffer, elcount(buffer),
6397 glbHandle);

6398                                snprintf (buffer,elcount(buffer),"\n\t\t
6399 <State> %d </State>",ErrorCalib[i][j][z].State);
                                filePutString (buffer, elcount(buffer),
6400 glbHandle);

6401                                snprintf (buffer,elcount(buffer),"\n\t\t
6402 <Name> %s </Name>",ErrorCalib[i][j][z].Names);
                                filePutString (buffer, elcount(buffer),
6403 glbHandle);

6404
6405                                snprintf (buffer,elcount(buffer),"\n\t\t
6406 <Min> %f </Min>",ErrorCalib[i][j][z].minVal);
                                filePutString (buffer, elcount(buffer),
6407 glbHandle);

6408                                snprintf (buffer,elcount(buffer),"\n\t\t
6409 <Max> %f </Max>",ErrorCalib[i][j][z].maxVal);
                                filePutString (buffer, elcount(buffer),
6410 glbHandle);

6411                                snprintf (buffer,elcount(buffer),"\n\t\t
6412 <Value> %f </Value>",ErrorCalib[i][j][z].VALUE);
                                filePutString (buffer, elcount(buffer),
6413 glbHandle);

6414
6415                                snprintf (buffer,elcount(buffer),"\n\t</
6416 error>");
                                filePutString (buffer, elcount(buffer),
6417 glbHandle);

6418
6419                            }
6420
6421                        }
6422                    }
6423
```

```
                }
6424
6425
6426            snprintf (buffer,elcount(buffer),"\n</ImmunityCycle>\n");
6427            filePutString (buffer, elcount(buffer),glbHandle);
6428
6429            FileClose(glbHandle);
6430
6431
6432
6433    }
6434
6435
6436
6437
6438
6439
6440
6441
6442 }
6443
6444 // CYCLING mode.
6445 on envVar CYCLING_MODE
6446 {
6447     if(getValue(this))
6448   {
6449     //Set the default background and text color for a specific control
6450 of a panel.
     //SetDefaultControlColors("EMC32","Display");
6451
6452     putValue(CALIBRATION_MODE,0);
6453     putValue(Ciclat, 0);
6454     putValue(STATE, 0);
6455     putValue(AT_Step,0);
6456     putValue(VAR_Start,1);
6457     }
6458
6459 }
6460
6461 on envVar CALIBRATION_MODE
6462 {
6463
6464     if(getValue(this))
6465   {
6466     putValue(AT_Step,0);
6467     putValue(STATE,0);
6468     putValue(Ciclat,0);
6469     putValue(CYCLING_MODE,0);
6470     }
6471 }
6472
6473 on envVar CALIBRATION_XML
6474 {
6475     if(getValue(this))
6476   {
6477     CreateXMLCalibration();
6478     }
6479
```

```
}

on envVar CYCLING_XML
{
putValue(CYCLING_XML,0);
CreateXMLCycling();
}

on key 'c'
{
settimer(tflagC,750);
//VALUES[144]=0;
}

on timer tflagC
{

    senueloC=1;



}

on envVar CANSTATE
{

    if(getValue(this))
    {

    //write("arrayError[144][0]=%d",arrayError[144][0]);
    //write("arrayError[144][1]=%d",arrayError[144][1]);


        /*if((arrayError[144][1] < arrayError[144][0]) && getValue(Ciclat)>1)
        {

        write("recuperem la comunicació");
        setTimer(FailureCAN,270);

        ErrorImm[getValue(Ciclat)][getValue(STATE)][144].Ciclat= getValue(Ciclat);
        ErrorImm[getValue(Ciclat)][getValue(STATE)][144].State= getValue(STATE);
        strncpy(ErrorImm[getValue(Ciclat)][getValue(STATE)][144].Names, AT_NAMES[144],MAX_VAR_LENGTH);
        ErrorImm[getValue(Ciclat)][getValue(STATE)][144].Frequenz= getValue(EMV_Frequenz_aux);
        ErrorImm[getValue(Ciclat)][getValue(STATE)][144].ImmunityField= getValue(EMV_Feld);
        ErrorImm[getValue(Ciclat)][getValue(STATE)][144].Threshold =getValue(Threshold);
        ErrorImm[getValue(Ciclat)][getValue(STATE)][144].Error= 1;

        SetControlColors ("EMC32","Display",  MakeRGB(255,255,255), MakeRGB(0,0,0));
        putValueToControl("EMC32","Display","\nCycle: ");
```

```
        putValueToControl("EMC32","Display",getValue(Ciclat),0);
        putValueToControl("EMC32","Display","        State: ");
        putValueToControl("EMC32","Display",getValue(STATE),0);
        putValueToControl("EMC32","Display","\nNAME: ");
        putValueToControl("EMC32","Display",AT_NAMES[144]);
        putValueToControl("EMC32","Display","\nFrequency: ");
        putValueToControl("EMC32","Display",getValue(EMV_Frequenz_aux),0
);
        putValueToControl("EMC32","Display"," MHz");
        putValueToControl("EMC32","Display","\nImmunity Field: ");
        putValueToControl("EMC32","Display",getValue(EMV_Feld),0);
        putValueToControl("EMC32","Display"," V/m");
        putValueToControl("EMC32","Display","\
n                        ");

        }*/
    }

}

on timer FailureCAN
{
long indexCAN;

    switch(indexCAN)
      {
        case 0:
        {
            putValue(VAR_Stop,1);
      indexCAN=1;
            setTimer(FailureCAN,200);
      break;

        }
            case 1:
        {
            putValue(VAR_Start,1);
      indexCAN=2;
            setTimer(FailureCAN,200);
      break;

        }
            case 2:
        {
            putValue(AT_Load,1);
      indexCAN=3;
            setTimer(FailureCAN,200);
      break;

        }

            case 3:
        {
            putValue(AT_Play,0);
      indexCAN=4;
            setTimer(FailureCAN,200);
      break;

```

```
        }
6584
6585            case 4:
6586        {
6587            putValue(AT_Play,1);
6588            indexCAN=0;
6589            //setTimer(FailureCAN,100);
6590            write("ejecutamos todo el proceso");
6591      break;
6592        }
6593
6594      }
6595
6596
6597 }
6598
6599 on warningLimit
6600 {
6601 putValue(CAN_STATE_LED,0);
6602 }
6603
6604
6605
```