PL Engineering Ltd.

API

PX5100.dll & LabVIEW

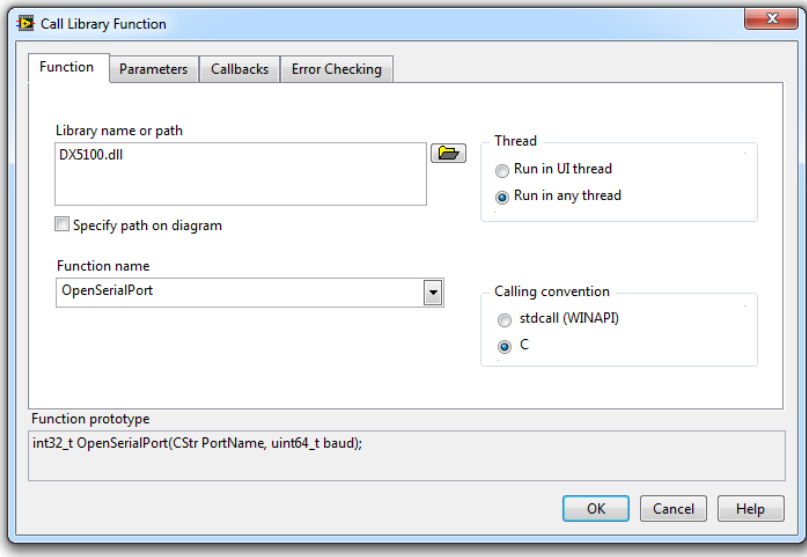
Moscow, 2016
Version 1.01

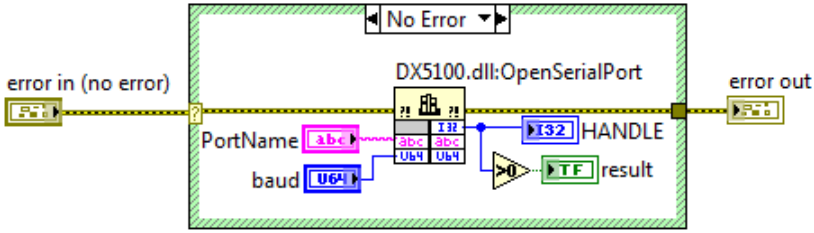
Contents

SERVICE FUNCTIONS

```
//------------------------------------------------------
```

OpenSerialPort

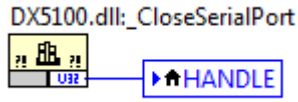





```
// Supported baud:
9600,19200(default),38400,57600,115200
```

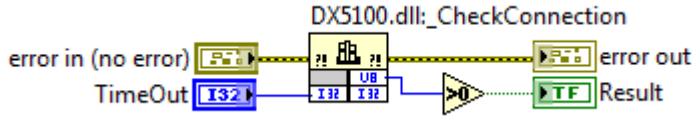
CloseSerialPort

```
uint32_t _CloseSerialPort(void );
```



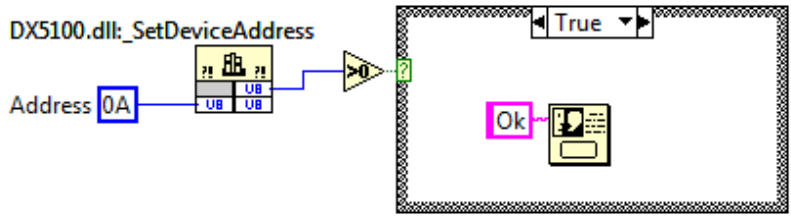
CheckConnection

```
uint8_t _CheckConnection(int32_t TimeOut);
```



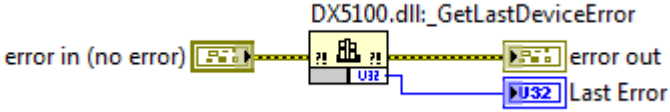
SetDeviceAddress

```
uint8_t _SetDeviceAddress(uint8_t Address);

// This address will be involved in the formation of a
query to the device
```



GetLastDeviceError

```
uint32_t _GetLastDeviceError(void );
```

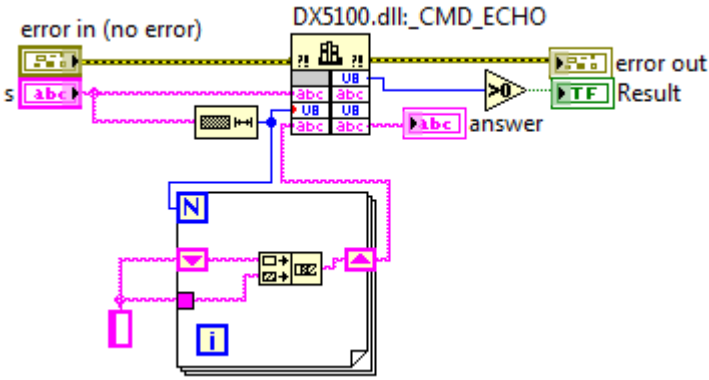

DX5100.dll:_GetLastDeviceError
error in (no error)
error out
U32
Last Error

SYSTEM COMMANDS

//--

0x02 CMD_ECHO

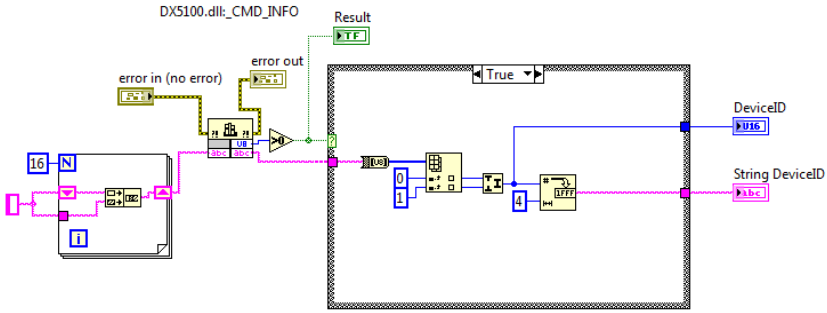
```
void  _CMD_ECHO(CStr s, uint8_t len, CStr answer);
```



//--

0x03 CMD_INFO

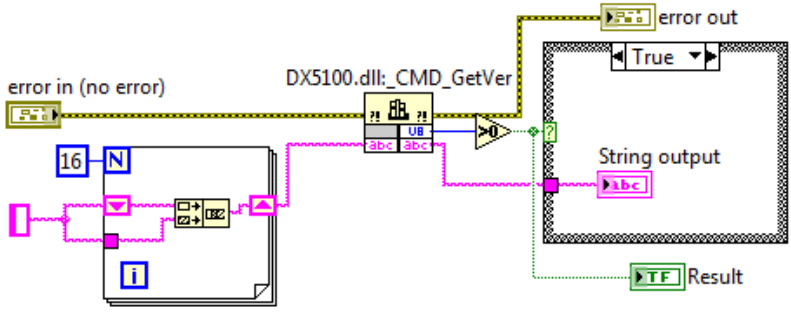
```
uint8_t _CMD_INFO(CStr DeviceID);
```



//--

0x04 CMD_GetVer

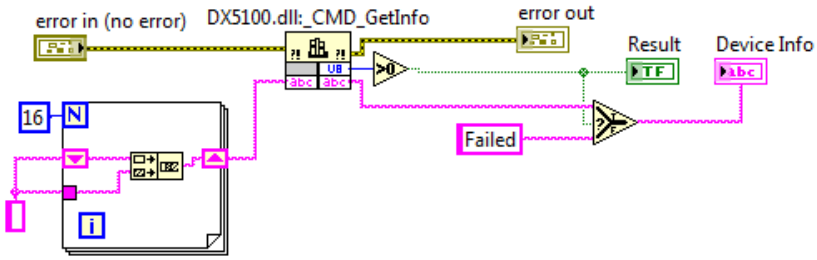
```
uint8_t _CMD_GetVer(CStr FirmwareVer);
```




```
//-------------------------------------------------------
```

0x05 CMD_GetInfo

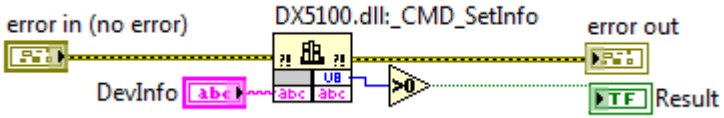
```
uint8_t _CMD_GetInfo(CStr DevInfo);
```




```
//-------------------------------------------------------
```

0x06 CMD_SetInfo

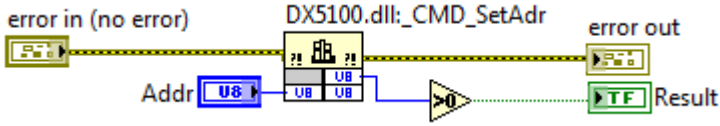
```
uint8_t _CMD_SetInfo(CStr DevInfo);
```




```
//-------------------------------------------------------
```

0x07 CMD_SetAdr

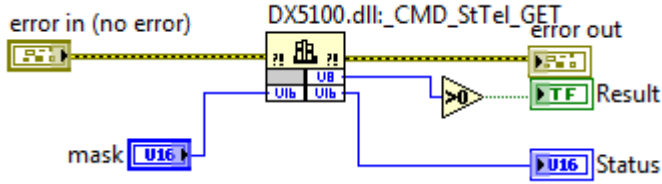
```
uint8_t _CMD_SetAdr(uint8_t Addr);
```




```
//-------------------------------------------------------
```

0x40 CMD_StTel_GET

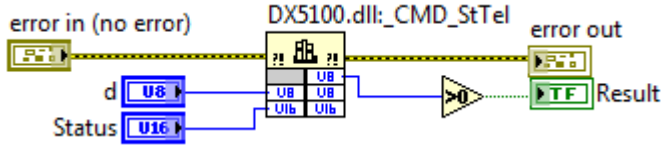
```
uint8 t  CMD StTel GET(uint16 t *Status);
```




```
//-------------------------------------------------------
```

0x40 CMD_StTel_SET

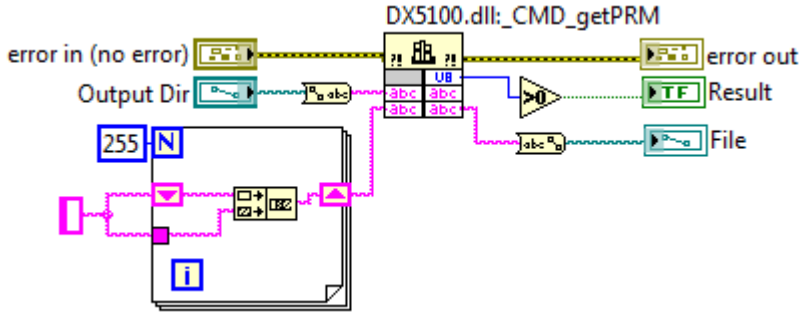
```
uint8_t _CMD_StTel(uint8_t d, uint16_t Status);
```




```
//-------------------------------------------------------
```

0x41 CMD_getPRM

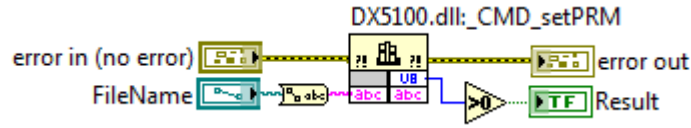
```
uint8_t _CMD_getPRM(CStr OutputFileDir);
```



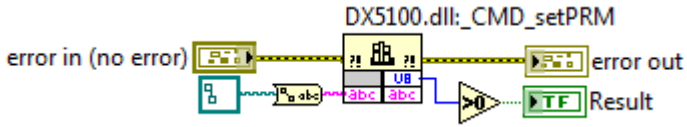

```
//-------------------------------------------------------
```

0x42 CMD_setPRM

```
uint8_t _CMD_setPRM(CStr FileName);
```




```
// In this case, you will be prompted to select a file:
```

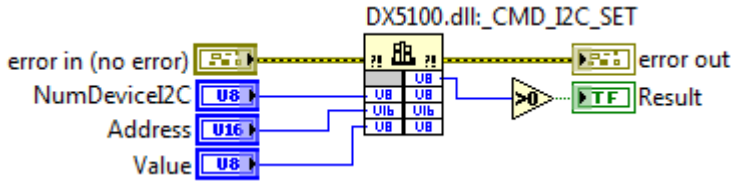



```
//-------------------------------------------------------
```

0x44 CMD_I2C

CMD_I2C_SET

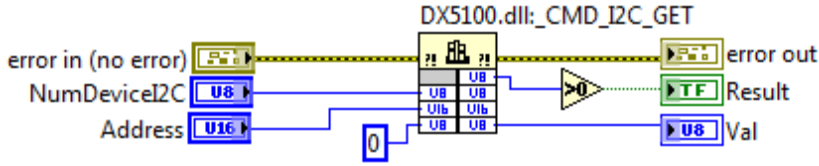
```
uint8_t _CMD_I2C_SET(uint8_t NumDeviceI2C, uint16_t
AddrDeviceI2C, uint8_t Val);
```




```
//-------------------------------------------------------
```

CMD_I2C_GET

```
uint8_t _CMD_I2C_GET(uint8_t NumDeviceI2C, uint16_t
AddrDeviceI2C, uint8 t *Val);
```



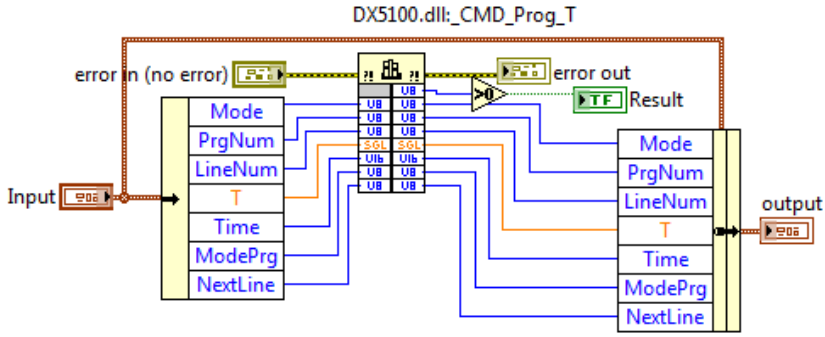

```
//-------------------------------------------------------
```

0x45 CMD_Prog_T

```
uint8_t _CMD_Prog_T(uint8_t *Mode, uint8_t *PrgNum,
uint8_t *LineNum, double *T, uint16_t *Time,
uint8_t *ModePrg, uint8_t *NextLine);
```

```
      char Mode = 1;            // 0-write, 1-read, 2-
set status, 3-get status
      char PrgNum = 0;          // number of program to
deal with
      char LineNum = 0;         // (0-49) for mode 0,1
or status(0-255) for mode 2,3
      float T = 0;              // Setpoint T[K] for
mode 2,3 and U(V) for mode 4
      WORD Time = 0;            // Time(s) to maintain
the setting value (setpoint)
      char ModePrg = 0;// Most significant nibble -
mode of current line
                                // 0  Interdiction of
regulation
                                // 2  T-regulation
                                // 3  Temperature
maintenance (PID)
```

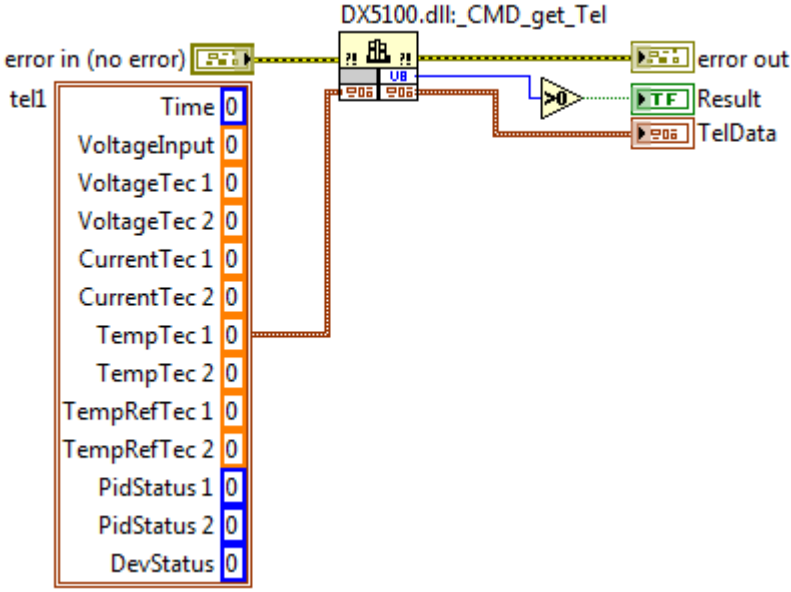
```
                                        // 4  Constant voltage
                                        // Least significant
nibble - number of
                                        // program to go (0-15)
after
        char NextLine = 0;              // number of line to go
(0-49)
```




```
//-------------------------------------------------------
```

0x46 CMD_get_Tel

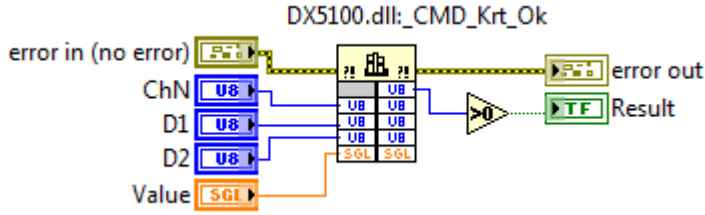
```
uint8_t _CMD_get_Tel(void *TelData);
```




```
//---------------------------------------------------------
```

0x49 CMD_Krt_Ok

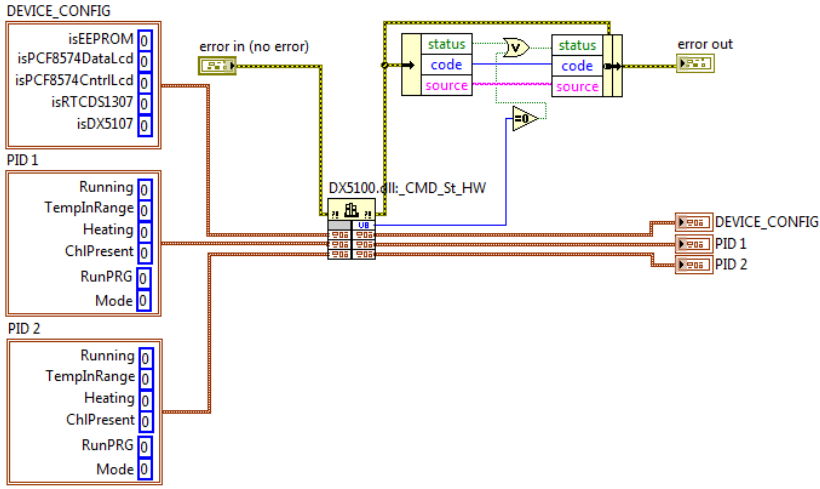
```
uint8_t _CMD_Krt_Ok(uint8_t ChN, uint8_t D1,
uint8_t D2, float Value);
```




```
//---------------------------------------------------------
```

0x4a CMD_St_HW

```
uint8_t _CMD_St_HW(void *dev, void *st1, void
*st2);
```

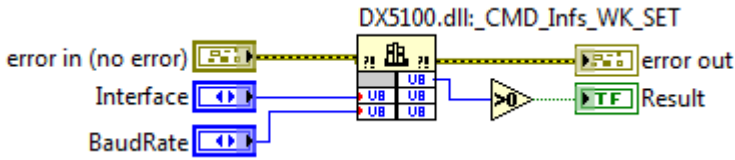



```
// NOTE: BOOL are equivalent I32 LabVIEW data type.

//-------------------------------------------------------
```

0x4b CMD_Infs_WK

CMD_Infs_WK_SET

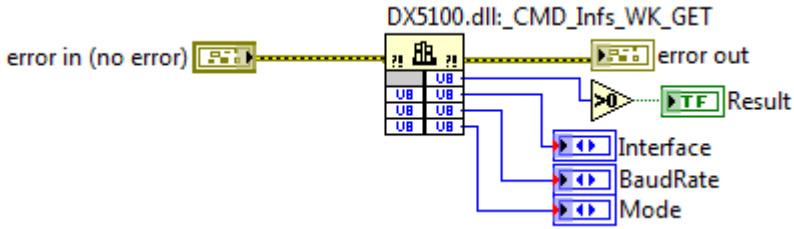
```
uint8_t _CMD_Infs_WK_SET(uint8_t interface, uint8_t
baud);
// interface:    0-RS232; 1-RS485
// baudrate:     0-9600; 1-19200; 2-38400; 3-57600; 4-
115200
```




```
//-------------------------------------------------------
```

CMD_Infs_WK_GET

```
uint8_t _CMD_Infs_WK_GET(uint8_t *interface,
uint8_t *baud, uint8_t *mode);
// interface:    0-RS232; 1-RS485
// baudrate:     0-9600; 1-19200; 2-38400; 3-57600; 4-
115200
// WAKE mode:    0-binary; 1-symbol
```

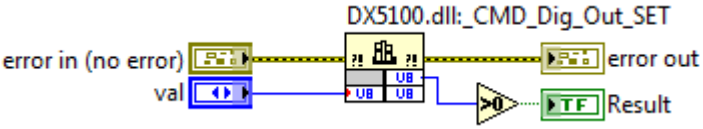



```
//-------------------------------------------------------
```

0x4d CMD_Dig_Out

CMD_Dig_Out_SET

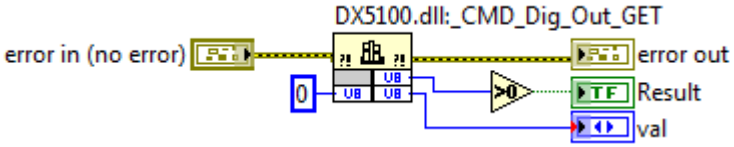
```
uint8_t _CMD_Dig_Out_SET(uint8_t val);
// 0-disable; 1-enable
```




```
//-------------------------------------------------------
```

CMD_Dig_Out_GET

```
uint8_t _CMD_Dig_Out_GET(uint8_t *val);
```

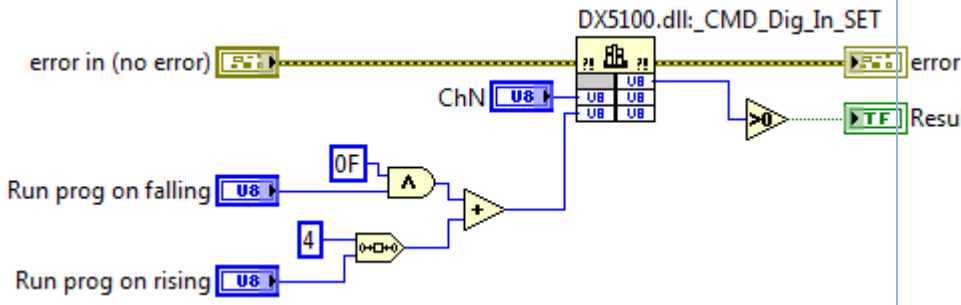



```
//-------------------------------------------------------
```

0x4e CMD_Dig_In

CMD_Dig_In_SET

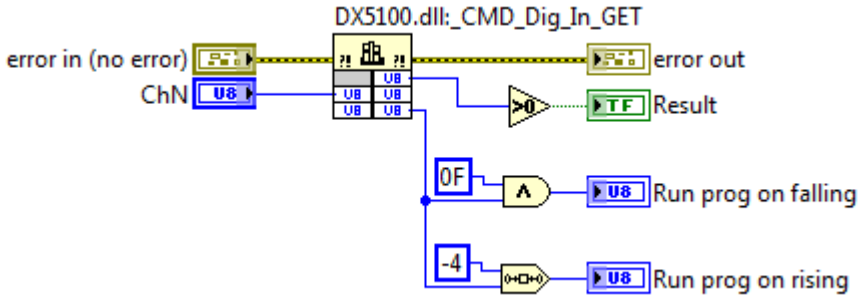
```
uint8_t _CMD_Dig_In_SET(uint8_t *ChN, uint8_t
*ProgNum);       {
// ChN: channel number
// ProgNum:
// D0..D3 number of program, which run on falling
// D4..D7 number of program, which run on rising
```




```
//-------------------------------------------------------
```

CMD_Dig_In_GET

```
uint8_t _CMD_Dig_In_GET(uint8_t *ChN, uint8_t
*ProgNum);       {
// ChN: channel number
// ProgNum:
// D0..D3 number of program, which run on falling
// D4..D7 number of program, which run on rising
```



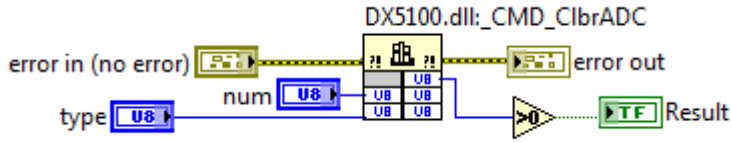
```
//-----------------------------------------------------
```

COMMANDS OF WORK WITH ADC

```
//-----------------------------------------------------
```

0x10 CMD_ClbrADC

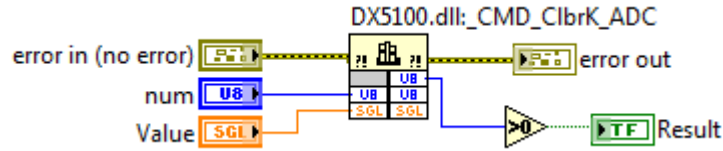
```
uint8_t _CMD_ClbrADC(uint8_t num, uint8_t type);
```




```
//-----------------------------------------------------
```

0x11 CMD_ClbrK_ADC

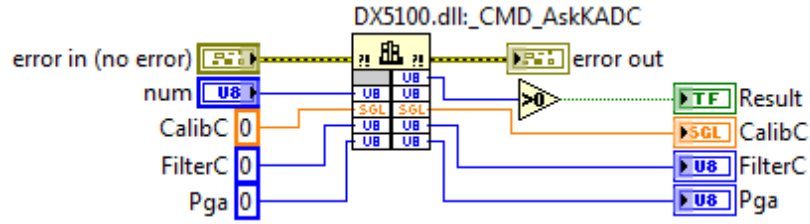
```
uint8_t _CMD_ClbrK_ADC(uint8_t num, float Value);
```




```
//-----------------------------------------------------
```

0x14 CMD_AskKADC

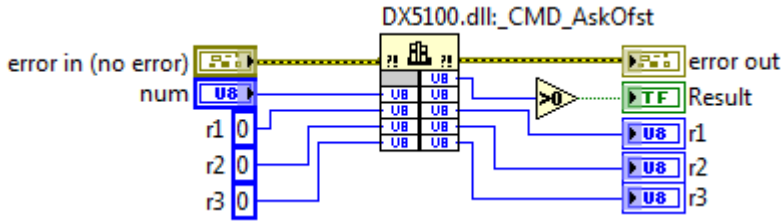
```
uint8_t _CMD_AskKDAC(uint8_t num, float *CalibC,
uint8_t *FilterC, uint8_t *Pga);
```



```
//-------------------------------------------------------
```

0x15 CMD_AskOfst

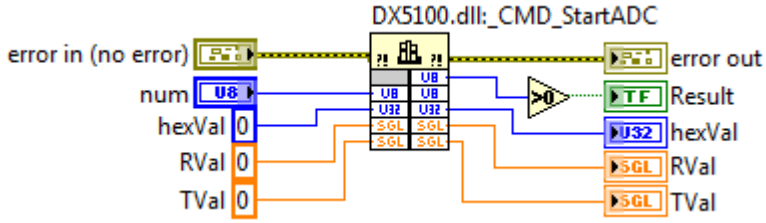
```
uint8_t _CMD_AskOfst(uint8_t num, uint8_t *r1,
uint8_t *r2, uint8_t *r3);
```




```
//-------------------------------------------------------
```

0x16 CMD_StartADC

```
uint8_t _CMD_StartADC(uint8_t ChN, uint32_t
*hexVal, float *RVal, float *TVal);
```




```
//-------------------------------------------------------
```

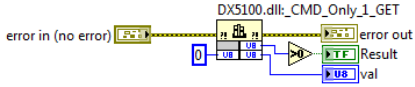
0x17 CMD_Only_1

CMD_Only_1_GET

```
uint8_t _CMD_Only_1_GET(uint8_t *val);
```

2.5.8. ADC One Channel Measurement
To increase the speed of digitalizing there is a command 17h that transfers ADC into a mode in which the channels are not switched, and the one chosen is measured. This mode can be used for measuring the object dynamic characteristics.

1 Byte with bit set in a position corresponding to a chosen channel, if the channel for quick measurement is chosen, otherwise - 0x00
ADC channel
01h supply voltage
02h TEC1 voltage
04h TEC2 voltage
08h TEC1 current
10h TEC2 current
20h TEC1 temperature
40h TEC2 temperature



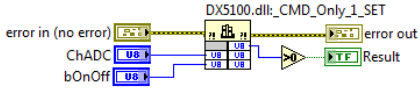

```
//-------------------------------------------------------
```

CMD_Only_1_SET

```
uint8_t _CMD_Only_1_SET(uint8_t ChADC, uint8_t
bOnOff);
```

2.5.8. ADC One Channel Measurement
To increase the speed of digitalizing there is a command 17h that transfers ADC into a mode in which the channels are not switched, and the one chosen is measured. This mode can be used for measuring the object dynamic characteristics.

1 Byte with bit set in a position corresponding to a chosen channel, if the channel for quick measurement is chosen, otherwise - 0x00
ADC channel
01h supply voltage
02h TEC1 voltage
04h TEC2 voltage
08h TEC1 current
10h TEC2 current
20h TEC1 temperature
40h TEC2 temperature



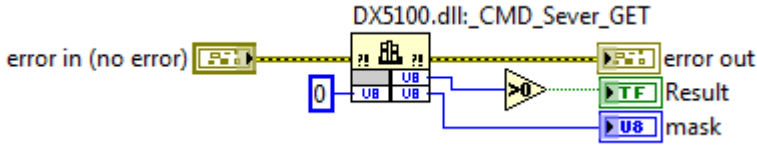

```
//-------------------------------------------------------
```

0x18 CMD_Sever

CMD_Sever_GET

```
uint8_t _CMD_Sever_GET(uint8_t *mask);

// mask     ADC channel
// ---------------------------------
```

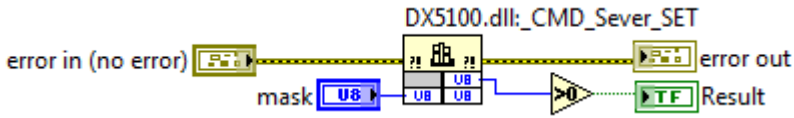
```
// 0x01     supply voltage
// 0x02     TEC 1 voltage
// 0x04     TEC 2 voltage
// 0x08     TEC 1 current
// 0x10     TEC 2 current
// 0x20     TEC 1 temperature
// 0x40     TEC 2 temperature
```




```
//-------------------------------------------------------
```

CMD_Sever_SET

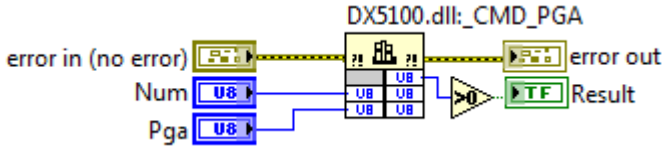
```
uint8_t _CMD_Sever_SET(uint8_t mask);

// mask     ADC channel
// ---------------------------------
// 0x01     supply voltage
// 0x02     TEC 1 voltage
// 0x04     TEC 2 voltage
// 0x08     TEC 1 current
// 0x10     TEC 2 current
// 0x20     TEC 1 temperature
// 0x40     TEC 2 temperature
```




```
//-------------------------------------------------------
```

0x19 CMD_PGA

```
uint8_t _CMD_PGA(uint8_t Num, uint8_t Pga);
// ADC channel number
// determines Programmable Gain Amplifier = 2hh
// 0-1 2-4    4-16   6-64
// 1-2 3-8    5-32   7-128
```

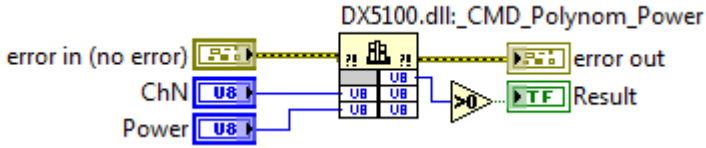



```
//-------------------------------------------------------
```

0x1a CMD_Polynom

CMD_Polynom_Power

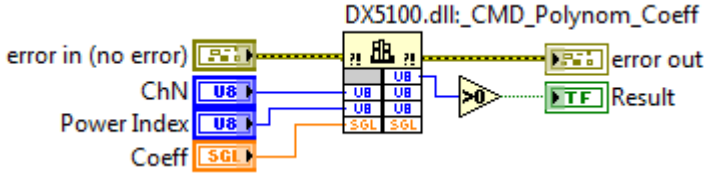
```
uint8_t _CMD_Polynom_Power(uint8_t ChN, uint8_t
Power);
```




```
//-------------------------------------------------------
```

CMD_Polynom_Coeff

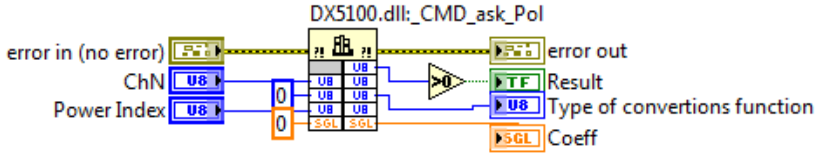
```
uint8_t _CMD_Polynom_Coeff(uint8_t ChN, uint8_t
Power Index, float Coeff);
```




```
//-------------------------------------------------------
```

0x1b CMD_ask_Pol

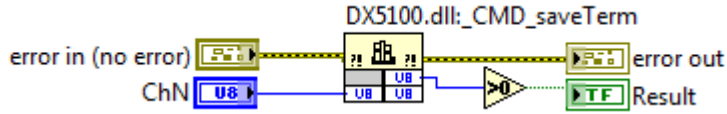
```
uint8_t _CMD_ask_Pol(uint8_t ChN, uint8_t *Type of
convertions function, uint8_t Power Index, float
*Coeff);
```




```
//-------------------------------------------------------
```

0x1c CMD_saveTerm

```
uint8_t _CMD_saveTerm(uint8_t ChN);
// save current settings into thermistors table
// The current calibration settings of thermistor
measurements input are stored in a// line,
// according to the parameter of the set current and ADC
gain.
// A table line stores: the conversion factor, the
values of calibration offset
// registers and sign validity data
```

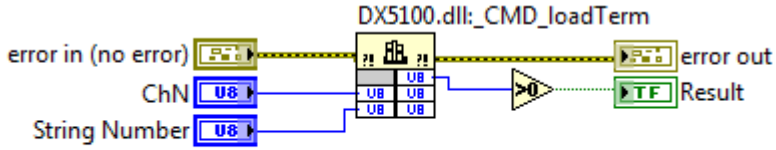



```
//-------------------------------------------------------
```

0x1d CMD_loadTerm

```
uint8_t _CMD_loadTerm(uint8_t ChN, uint8_t String
Number);
// select thermistor settings from table (line 0C) &
reset
// This command causes the reboot controller

// "String Number" Parameter corresponding to the set
characteristics of the
// calibration
// (to the maximum possible thermistor resistance)
// It is checked whether the data stored in the table
are correct (data saved by the// command 1C).
// If the data are correct, the calibration parameters,
the values of the measuring
// current and channel ADC gain
// are filled according to them.
```

```
//     The command execution result:
// 0x01- parameters are restored
// 0x00- requested parameters aren't stored by the
command1C.
```

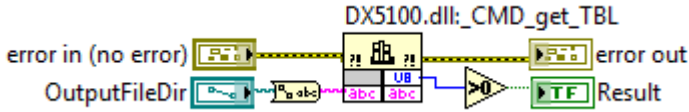


This command causes the reboot controller!

```
//-------------------------------------------------------
```

0x1e CMD_get_TBL

```
uint8_t _CMD_get_TBL(CStr OutputFileDir);
// save thermistor table to file
// After sending this command, the controller outputs
the contents of the table with// the parameters stored
in
// the nonvolatile memory into the command and non-
command interfaces.
// Each byte is transmitted as two characters of the
hexadecimal digits.
// This command is used for backup of parameters stored.
The output of parameters will// begin into
// both interfaces after reception of any character by
any interface.
// The data corresponding to one line are stored in 8
bytes.
```

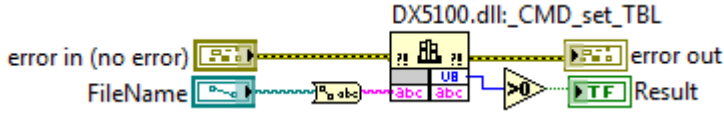



```
//-------------------------------------------------------
```

0x1f CMD_set_TBL

```
uint8_t _CMD_set_TBL(CStr FileName);
// save thermistor table file into device
```

```
// After receiving this command, the device expects data
to come by the command or
// non-command
// interface and interprets them as stored by the
command 1E.
// The data received are stored in the non-volatile
memory.
```

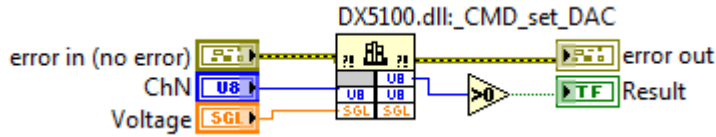


COMMANDS OF WORK WITH DAC

//---

0x21 CMD_set_DAC

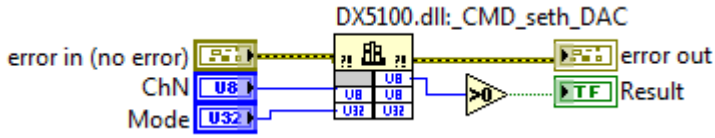
```
uint8_t _CMD_set_DAC(uint8_t ChN, float Voltage);
      // When the command is executed, the set voltage
does not exceed
      // maximal for a chosen channel.
```



//---

0x22 CMD_seth_DAC

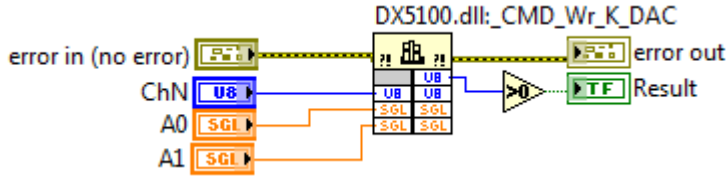
```
uint8_t _CMD_seth_DAC(uint8_t ChN, uint32_t Mode);

      // Attention! Check for excess of maximal voltage
is not done!
// The command is used for calibration.
// It should be used at voltage up to 8 V.
```



//---

0x23 CMD_Wr_K_DAC

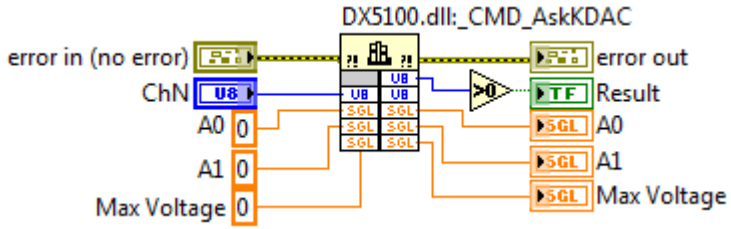
```
uint8_t _CMD_Wr_K_DAC(uint8_t ChN, float A0, float
A1);
// The coefficients determine the function by which a
value
// loaded to DAC is obtained, depending on voltage
needed.
```




```
//-------------------------------------------------------
```

0x24 CMD_AskKDAC

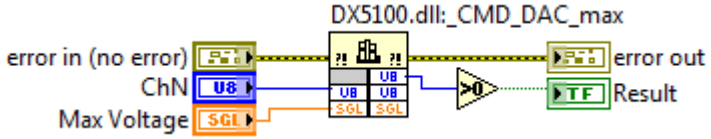
```
uint8_t _CMD_AskKDAC(uint8_t ChN, float *A0, float
*A1, float *Max);
// Sending Coefficient and DAC Maximal Values
```




```
//-------------------------------------------------------
```

0x25 CMD_DAC_max

```
uint8_t _CMD_DAC_max(uint8_t ChN, float Value);
// Writing Maximal Allowable Voltage
```

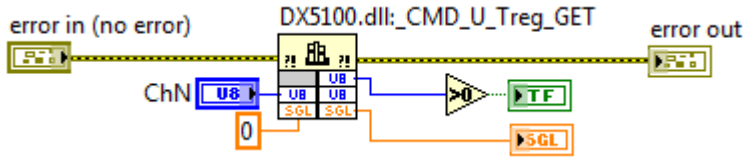



```
//-------------------------------------------------------
```

0x26 CMD_U_Treg

CMD_U_Treg_GET

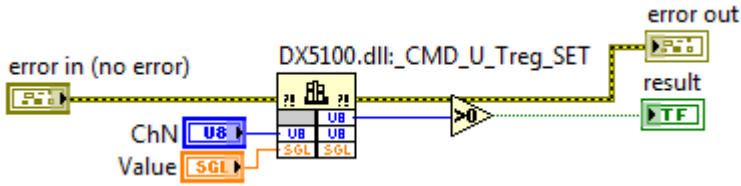
```
uint8_t _CMD_U_Treg_GET(uint8_t ChN, float *Value);
// Voltage of T-reg
```




```
//---------------------------------------------------------
```

CMD_U_Treg_SET

```
uint8_t _CMD_U_Treg_SET(uint8_t ChN, float Value);
// Voltage of T-reg
```

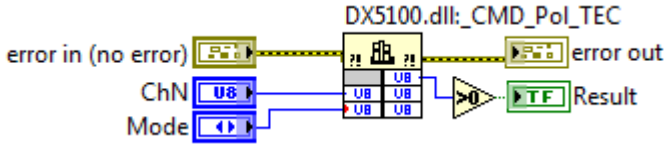



```
//---------------------------------------------------------
```

COMMANDS OF WORK WITH PID CONTROLLER

0x30 CMD_Pol_TEC

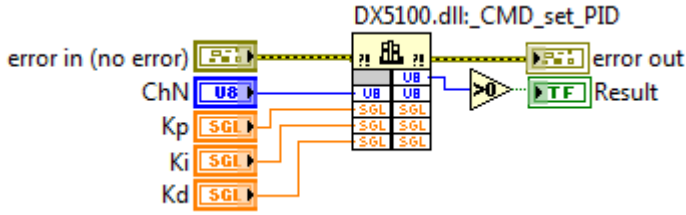
```
uint8_t _CMD_Pol_TEC(uint8_t ChN, uint8_t Mode);
// Setting of TEC Polarity
// 0- TEC is off
// 1- TEC is heating
// 2- TEC is cooling
```




```
//---------------------------------------------------------
```

0x31 CMD_set_PID

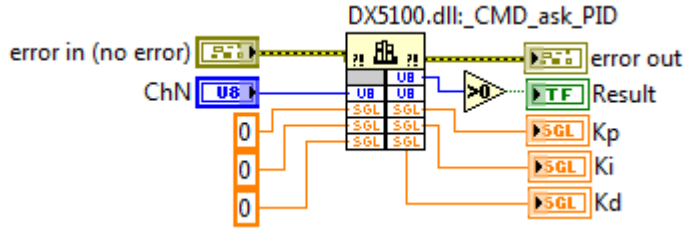
```
  uint8_t _CMD_set_PID(uint8_t ChN, float Kp, float Ki,
float Kd);
// Writing Parameters of PID Controller
```




```
//-------------------------------------------------------
```

0x32 CMD_ask_PID

```
uint8_t _CMD_ask_PID(uint8_t ChN, float *Kp, float
*Ki, float *Kd);
// Read Parameters of PID Controller
```

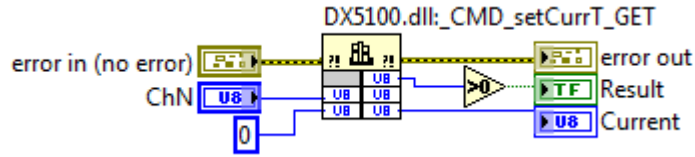



```
//-------------------------------------------------------
```

0x33 CMD_setCurrT

CMD_setCurrT_GET

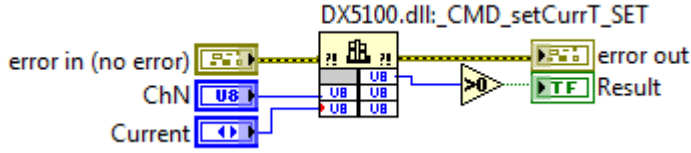
```
uint8_t _CMD_setCurrT_GET(uint8_t ChN, uint8_t
*Current);
// Get thermistor current
```




```
//-------------------------------------------------------
```

CMD_setCurrT_SET

```
uint8_t _CMD_setCurrT_SET(uint8_t ChN, uint8_t
Current);
// Set thermistor current
```

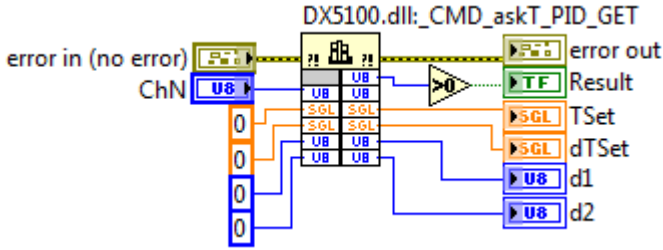



```
//---------------------------------------------------------
```

0x34 CMD_askT_PID

CMD_askT_PID_GET

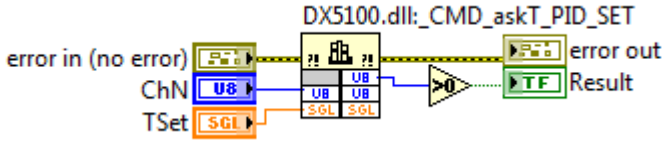
```
uint8_t _CMD_askT_PID_GET(uint8_t ChN, float *TSet,
float *dTSet, uint8_t *d1, uint8_t *d2);

// Sending/Set Setpoints of PID Controller
```




```
//---------------------------------------------------------
```

CMD_askT_PID_SET

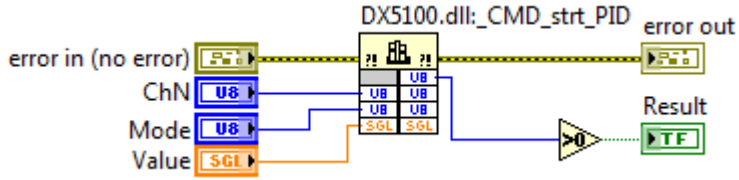
```
uint8_t _CMD_askT_PID_SET(uint8_t ChN, float TSet);
// Set Setpoints of PID Controller
```




```
//---------------------------------------------------------
```

0x35 CMD_strt_PID

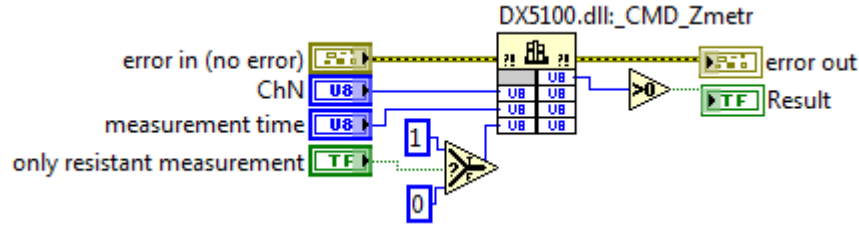
```
uint8_t _CMD_strt_PID(uint8_t ChN, uint8_t Mode,
float Value);
// Starting Controller

// 0 Regulation stop
// 1 Regulation according to program
// 2 T-regulation
// 3 Temperature maintenance – PID starting – setting
regulation
// 4 Constant voltage
```




```
//-------------------------------------------------------
```

0x37 CMD_Zmetr

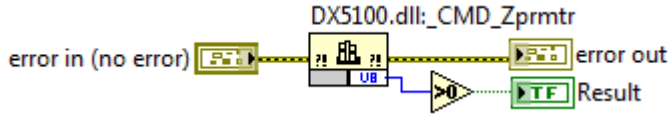
```
uint8_t _CMD_Zmetr(uint8_t ChN, uint8_t Mode,
uint8_t Value);
// Starting Z-meter

// ChN:  TEC channel number
// Time: Z-meter measurement time (s) 20…255
// AdvParam: if "1" – only resistant measurement
```




```
//-------------------------------------------------------
```

0x38 CMD_Zprmtr

```
uint8_t _CMD_Zprmtr(void );
// Storage of Z-Metering Parameters

// By this command the found parameters are stored as
// reference ones reference for the given object
```

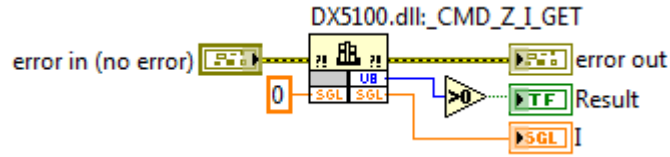



```
//-------------------------------------------------------
```

0x39 CMD_Z_I

CMD_Z_I_GET

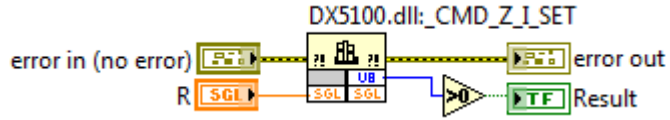
```
uint8_t _CMD_Z_I_GET(float *I);
// Get Z-Meter Current
```




```
//-------------------------------------------------------
```

CMD_Z_SET

```
uint8_t _CMD_Z_I_SET(float R);
// Set Z-Meter Current

// By this command the value of calculated current is
stored
// in the non-volatile memory and used for Z-metering
calculations
```

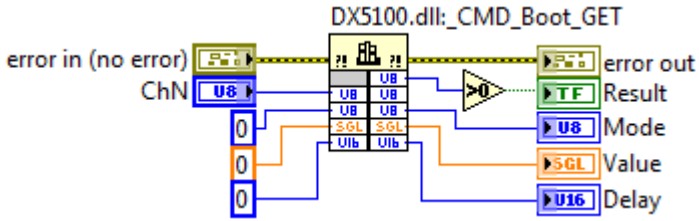



```
//-------------------------------------------------------
```

0x3b CMD_Boot

CMD_Boot_GET

```
uint8_t _CMD_Boot_GET(uint8_t ChN, uint8_t *Mode,
float *Value, uint16_t *Delay);
// Switching On Regulation after Restarting

// TEC channel number
// 3   4
// 0   Regulation stop                    Not present
// 1   Regulation according to program    0…15 program to
go to
// 2   T-regulation                       Temperature to
be maintained
// 3   Temperature maintenance -
// PID starting -                      Temperature to be
maintained
// setting regulation                 (setting value)
// 4   Constant voltage                   Voltage to be
maintained
//
// time (s) after which to proceed to the program
// (only for Regulation according to program)
```

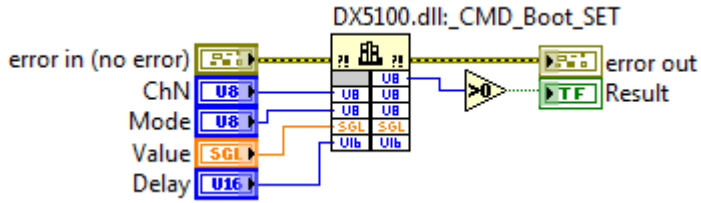



```
//-------------------------------------------------------
```

CMD_Boot_SET

```
uint8_t _CMD_Boot_SET(uint8_t ChN, uint8_t Mode,
float Value, uint16_t Delay);
// Switching On Regulation after Restarting

// TEC channel number
// 3   4
// 0   Regulation stop                  Not present
// 1   Regulation according to program  0…15 program to
go to
```

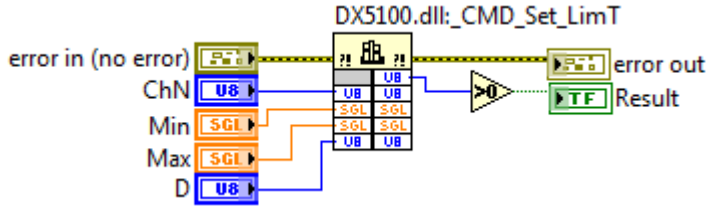
```
// 2   T-regulation                    Temperature to
be maintained
// 3   Temperature maintenance -
// PID starting -                   Temperature to be
maintained
// setting regulation               (setting value)
// 4   Constant voltage                Voltage to be
maintained
//
// time (s) after which to proceed to the program
// (only for Regulation according to program)
```




```
//-------------------------------------------------------
```

0x3c CMD_Set_LimT

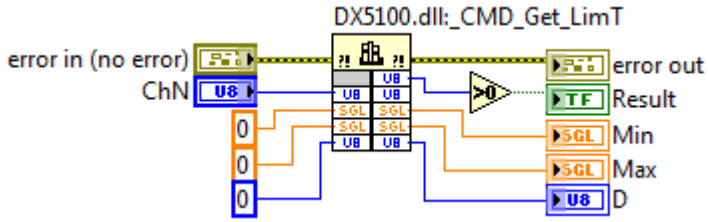
```
uint8_t _CMD_Set_LimT(uint8_t ChN, float Min, float
Max, uint8_t D);
// Writing Limiting Temperatures
```




```
//-------------------------------------------------------
```

0x3d CMD_Get_LimT

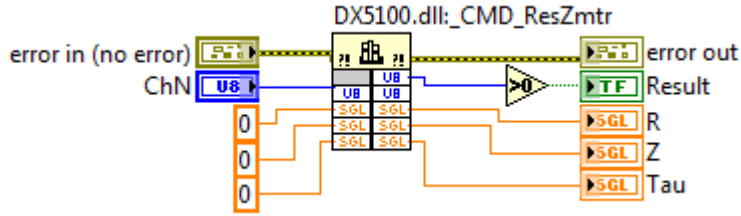
```
uint8_t _CMD_Get_LimT(uint8_t ChN, float *Min,
float *Max, uint8_t *D);
// Sending Limiting Temperatures
```




```
//-------------------------------------------------------
```

0x3e CMD_ResZmtr

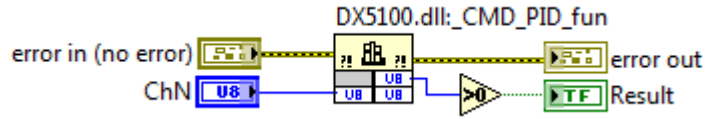
```
uint8_t _CMD_ResZmtr(uint8_t ChN, float *R, float
*Z, float *Tau);
// Sending Z-metering Results
```




```
//-------------------------------------------------------
```

0x51 CMD_PID_fun

```
uint8_t _CMD_PID_fun(uint8_t ChN);
// Autotuning PID

// The autotuning function searches the values of
proportional,
// integral and differential coefficients of the PID
algorithm.
```

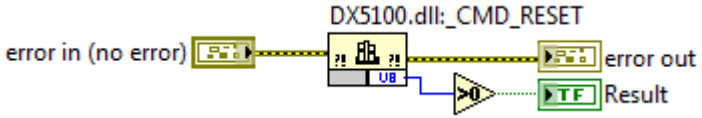


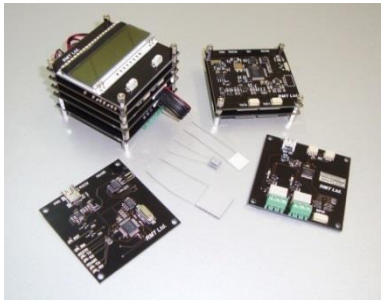

```
//-------------------------------------------------------
```

0x53 CMD_RESET

```
uint8_t _CMD_RESET(void );
```

```
// reset controller
```





PL Engineering Ltd.

46 Warshawskoe shosee
Moscow 115230 Russia
e-mail: info@promln.com
phone: +7-499-678-3231
fax: +7-499-678-3258
website: www.promln.ru

Overseas Sales representative

TEC Microsystems GmbH
Schwarzschildstrasse 8
Berlin 12489, Germany
phone: +49-(0)30-6789-3314
fax: +49-(0)30-6789-3315
e-mail: info@tec-microsystems.com