

# ***STX Driver Description***

# Table of Content

1 LIST OF NEW COMMANDS.....	4
1.1 MAIN COMMANDS.....	5
1.1.1 STX2Activate.....	5
1.1.2 STX2Deactivate.....	6
1.1.3 STX2Reset.....	7
1.1.4 STX2ReadActualClimate.....	8
1.1.5 STX2WriteSetClimate.....	9
1.1.6 STX2ReadSetClimate.....	10
1.1.7 STX2ActivateShaker.....	11
1.1.8 STX2DeactivateShaker.....	12
1.1.9 STX2ReadSetShakerSpeed.....	13
1.1.10 STX2SwapIn.....	14
1.1.11 STX2SwapOut.....	15
1.1.12 STX2Lock.....	16
1.1.13 STX2UnLock.....	17
1.1.14 STX2AbandonAccess.....	18
1.1.15 STX2ContinueAccess.....	19
1.1.16 STX2GetSysStatus.....	20
1.1.17 STX2ServiceReadBarcode.....	21
1.1.18 STX2Inventory.....	22
1.1.19 STX2ServiceMovePlate.....	23
1.2 ADDITIONAL COMMANDS.....	25
1.2.1 STX2LoadPlate.....	25
1.2.2 STX2UnloadPlate.....	26
1.2.3 STX2IsOperationRunning.....	27
1.2.4 STX2ReadErrorCode.....	28
1.2.5 STX2SoftReset.....	29
1.2.6 STX2ReadUserDoorFlag.....	30
1.2.7 STX2ReadShovelDetector.....	31
1.2.8 STX2ReadXferStationDetector1.....	32
1.2.9 STX2ReadXferStationDetector2.....	33
1.2.10 STX2BeeperOn.....	34
1.2.11 STX2BeeperOff.....	35
2 STOREX NETWORK COMMUNICATION.....	36
2.1 CLIENT SOCKET EXAMPLE (C#).....	37
2.1.1 Client Socket class.....	37
2.1.2 Using an Client Socket.....	40



## 1 List of new Commands

Each command is a set of characters, consisting of name and parameters. All command have at least one parameter, that is ID of device and ended with Carriage Return ASCII 0Dh (CR).

The ID of each device is assigned in a configuration file (ParameterGroup - "Device", Parameter Name - "ID"). All parameters are enclosing in brackets and separated by coma. Each command is prompted by a Response string. Response is an ASCII string sent by device.

In case of syntax error the SoreX TCP/IP Server returns "E1", "E2" or "E3" values.

"E1"+CR - invalid command or command was sent without CR.

"E2"+CR - Invalid ID of a device configuration files was not loaded and Server can't recognise any ID

"E3"+CR - Invalid parameter or error of parsing a command (type cast error etc.).

Each response is ended with (each response consists of) two characters: Carriage Return ASCII 0Dh (CR) + Line Feed ASCII 0Ah (LF).

## 1.1 Main Commands

### 1.1.1 STX2Activate

"STX2Activate(ID)" – Opens Serial Communication and Initialises the StoreX (opens the PLC connection, initialises the handling, reads StoreX system constants).

Parameter:

ID – Identifier of a device.

Return values: the reply consists of one or two characters (separated by semicolon) and CR+LF.

The first character is the device initialisation state:

- 1 - Communication is opened and device is initialised.
- 1 - Error of opening Serial Port.
- 2 - Error of opening Serial Port (serial Port is already opened).
- 3 - No communication.
- 4 - Communication Error.
- 5 - System Error (System Error Flag is true).
- 6 - User Door is opened (or cannot read User Door status).
- 7 - System is in undefined status (Error and Ready Flags are false)

The second character depends on Barcode Reader presence and shows it initialisation state :

- 1 - BCR Serial Port is successfully opened.
- 1 - Error of opening BCR port.
- 2 - wrong value of BCR port.

### 1.1.2 STX2Deactivate

"STX2Deactivate(ID)" - Closes serial communication through the active Serial Port.  
This function also closes Serial communication for Barcode Reader.

Parameter: ID – Identifier of a device.

Return values: CR+LF.

### 1.1.3 STX2Reset

"STX2Reset(ID)" - Reset the StoreX after the error. Puts the StoreX in the idle state. The user should call the STX2Reset method after any error of the machine. The user should call STX2Activate again to continue operations, or press manually the "Reset" button of the machine.

Parameter: ID – Identifier of a device.

Return values: CR+LF.

#### 1.1.4 STX2ReadActualClimate

"STX2ReadActualClimate(ID)" – Returns actual climate values separated by semicolon.

Parameter: ID – Identifier of a device.

Return values: climate values separated by semicolon and CR+LF.

The first is a value of temperature in °C, the second is a value of relative humidity in percent, the third is a value of CO2 concentration in percent, the fourth is a value of N2 concentration in percent.

### 1.1.5 STX2WriteSetClimate

"STX2WriteSetClimate(ID,T,H,CO2,N2)" – sets the climate values.

Parameter:

ID – Identifier of a device.

T – target temperature in °C.

H – target relative humidity in percent.

CO2 – the target CO2 concentration in percent.

N2 – the target N2 concentration in percent.

Return values: CR+LF.

### 1.1.6 STX2ReadSetClimate

"STX2ReadSetClimate(ID)" – Returns the target of climate values separated by semicolon.

Parameter: ID – Identifier of a device.

Return values: target climate values separated by semicolon and CR+LF.

The first is a target value of temperature in °C, the second is a target value of relative humidity in percent, the third is a target value of CO2 concentration in percent, the fourth is a target value of N2 concentration in percent.

### 1.1.7 STX2ActivateShaker

"STX2ActivateShaker(ID,Speed)" - Writes shaker speed settings value and switches shaker on.

Parameter:

ID - Identifier of a device.

Speed - Shaker speed (range 1...50).

Return values: CR+LF.

### 1.1.8 STX2DeactivateShaker

"STX2DeactivateShaker(ID)" - Switches shaker off.

Parameter: ID – Identifier of a device.

Return value: CR+LF.

### 1.1.9 STX2ReadSetShakerSpeed

"STX2ReadSetShakerSpeed(ID)" - Returns shaker speed value,

Parameter: ID – Identifier of a device.

Return values: shaker speed value or "-1" in case of an error and CR+LF.

### 1.1.10 STX2SwapIn

"STX2SwapIn(ID)" - Rotates the swap station on 180 degree.

Parameter: ID – Identifier of a device.

Return values: Result of operation and CR+LF.

1 - operation has been completed.

-1 - Error.

### 1.1.11 STX2SwapOut

"STX2SwapOut(ID)" - Rotates the swap station back to home position..

Parameter: ID – Identifier of a device.

Return values: Result of operation and CR+LF.

1 - operation has been completed.

-1 - Error.

### 1.1.12 STX2Lock

"STX2Lock(ID)" - Locks the door and reads User Door Switch

Parameter: ID - Identifier of a device.

Return values: User door status and CR+LF.

"1" - User's door is opened.  
"0" - User's door is closed.  
"-1" - Error.

### 1.1.13 STX2UnLock

"STX2UnLock(ID)" - Unlock the door.

Parameter: ID - Identifier of a device.

Return values: Result of operation and CR+LF.

"1" - operation has been completed.

"-1" - Error.

#### **1.1.14 STX2AbandonAccess**

"STX2AbandonAccess(ID)" – Abandons to perform the prior Load Plate Operation

Parameter: ID – Identifier of a device.

Return values: CR+LF.

### **1.1.15 STX2ContinueAccess**

"STX2ContinueAccess(ID)" – Continues to perform the prior Load Plate Operation

Parameter: ID – Identifier of a device.

Return values: CR+LF.

### 1.1.16 STX2GetSysStatus

"STX2GetSysStatus(ID)" – Returns value of Status Register (DM202).

Parameter: ID – Identifier of a device.

Return value: DM202 (Status Register) or "-1" in case of an error and CR+LF.

Bit	Comment
00	System Ready
01	Plate Ready
02	System Initialized
03	XferStn status change
04	Gate closed
05	User door
06	Warning
07	Error
08	
09	
10	
11	
12	
13	
14	
15	

### 1.1.17 STX2ServiceReadBarcode

"STX2ServiceReadBarcode(ID,Slot,Level)" - Reads the barcode of a plate at specified location.

Parameters:

ID - Identifier of a device.

Slot - plate slot position.

Level - plate level position.

Return values: Result of operation and CR+LF.

"BCRError" - Barcode reader is not initialised.

"InitError" - StoreX is not initialized.

"Device Status Error" - Device has the Error Status.

"Device not Ready" - Device not ready.

"Error" - Cassette or Level value not set or previous long operation is not finished.

"No Plate" - There is no Plate at the specified position.

"No Barcode" - There is no Barcode on the Plate.

### 1.1.18 STX2Inventory

"STX2Inventory(ID,InvFileName,PPD,BCR)" - Implements inventory of entire unit, the result is saved in the file– InvFileName. If the name of the file is not assigned, it will be generated automatically. The name of the file consists of a Serial number of device, date (e.g. "3298\_A7010101.inv", where last two digits are number of the file).

#### Parameters:

ID – Identifier of a device.

FileName – name of file for saving results of inventory.

PPD – {0,1} sets whether Plate Present Detector will be used for Inventory.

1 – Inventory with Plate Presents Detector.

0 – Inventory without Plate Presents Detector.

BCR – {0,1} sets whether Barcode Reader will be used for Inventory.

1 – Inventory with Barcode Reader.

0 – Inventory without Barcode Reader.

#### Return values:

1 – STX2InventoryCassets operation is started.

-1 – Device is not initialised.

-2 – Previous long operation is not finished.

-3 – Device not ready.

-4 – Device has the Error Status.

The result of the STX2Inventory is a file which consists of four columns separated by coma.

The first column is number of Cassette, the second column is a value of Level, the third column is a value of Plate Present Detector (1 - plate is present; 0 - plate is not present or Plate Present Detector is off), the fourth column is a value of a Barcode (<null> - No Barcode or Barcode Reader is switched off ).

### 1.1.19 STX2ServiceMovePlate

"STX2ServiceMovePlate(

SrcInstrID,SrcPos,SrcSlot,SrcLevel,TransSrcSlot,SrcPltType,  
TrgInstrID,TrgPos,TrgSlot,TrgLevel,TransTrgSlot,TrgPltType)"

Moves a plate from position SrcPos to position TrgPos. This operation allows to move the plate within the bound of one device or between the cascader system.

Parameters:

SrcInstr - Identifier of a source Device.  
SrcPos - Source position {1-TransferStation, 2-Slot-Level Position, 3 - Shovel, 4-Tunnel, 5-Tube Picker}.  
SrcSlot - plate slot position of source.  
SrcLevel - plate Level position of source.  
TransSrcSlot - number of a transport slot of a source device. It is obligatory for moving a plate between devices Base, Extended in Cascader. This Parameter is even for Extended Device and odd for Base Device.  
In case of Tube Picker this parameter defines a Source Plate Position on a Tube Picker Device {0,1}  
0-Target Position.  
1 - Source Position.  
  
SrcPltType - Type of plate of source position {0-MTP, 1-DWP, 3-P28}.  
  
TrgInstr - Identifier of a target device.  
TrgPos - Target position {1-TransferStation, 2-Slot-Level Position, 3 - Shovel, 4-Tunnel, 5-Tube Picker}.  
TrgSlot - plate slot position of target.  
TrgLevel - plate level position of target.  
TransTrgSlot - number of a transport slot of a Target Device. It is obligatory for moving a plate between devices Base, Extended in Cascader. This Parameter is even for Extended Device and odd for Base Device.  
In case of Tube Picker this parameter defines a Target Plate Position on a Tube Picker Device {0,1}  
0-Target Position.  
1 - Source Position.  
TrgPltType - Type of plate of source position {0-MTP, 1-DWP, 3-P28}.

Return value: result of operation and CR+LF.

1 - Function STX2ServiceMovePlate has been completed.

Returning values because of parameters error:

- 1 - Previous long operation is not finished.
- 2 - One of input parameters is not a valid integer value.
- 3 - A Source or a Target Device is not specified or not initialised.
- 4 - One or more of devices is not defined in a system.
- 5 - One of transport slot is not specified.
- 6 - wrong value of a target transport slot.
- 7 - wrong value of a source transport slot.
- 8 - wrong value of a source position.
- 9 - wrong value of a target position.

Returning values because of internal device error:

This value consists of more than one character and separated by semicolon. First character returns Identifier of a Device where the error has been taken place and last character indicates a step of the error.

- ID;1 - Error during LoadPlate operation, device "ID".
- ID;2 - Error during UnloadPlate operation.
- ID;3 - Error during PickPlate operation.
- ID;4 - Error during PlacePlate operation.
- ID;5 - Error during SetPlate operation.
- ID;6 - Error during GetPlate operation.
- ID;7 - Device is not ready.
- ID;8 - Devece has the Error Status.

## 1.2 Additional Commands

### 1.2.1 STX2LoadPlate

"STX2LoadPlate(ID,Slot,Level)" – Loads plate from Transfer Station to specified target location.

#### Parameters:

ID - Identifier of a device.  
Slot - Target slot position.  
Level - Target level position.

#### Return values:

1 - Plate has been loaded.  
-1 - Previous long operation is not finished.  
-2 - Device is not initialised.  
-3 - Device has the Error Status.  
-4 - Wrong value of a target position.  
-5 - Error of loading a plate.

### 1.2.2 STX2UnloadPlate

"STX2UnloadPlate(ID,Slot,Level)" – Unloads plate from specified source location to Transfer Station.

Parameters:

ID - Identifier of a device.

Slot - source slot position.

Level - source level position.

Return values:

1 - Plate has been unloaded.

-1 - Previous long operation is not finished.

-2 - Device is not initialised.

-3 - Device has the Error Status.

-4 - Wrong value of a target position.

-5 - Error of unloading a plate.

### 1.2.3 STX2IsOperationRunning

"STX2IsOperationRunning(ID)" - Checks whether previous long operation is running. The user can check whether long operations like STX2ServiceMovePlate, STX2Inventory are running by means of usage this method.

Parameter: ID – Identifier of a device.

Return values: Result of operation and CR+LF.

"1" - Previous long operation is still running

"0" - Previous long operation is completed or cancelled (or Device has the Error Status).

#### 1.2.4 STX2ReadErrorCode

"STX2ReadErrorCode(ID)" - Interrogates Error Flag and Error code of System.

Parameter: ID – Identifier of a device.

Return values: Result of operation and CR+LF.

"0" – No Errors.

If returns the System status code then see descriptions of error codes.

"-1" – Error.

### 1.2.5 STX2SoftReset

"STX2SoftReset(ID)" - Implements Soft Reset command.

Parameter: ID – Identifier of a device.

Return values: CR+LF.

### 1.2.6 STX2ReadUserDoorFlag

"STX2ReadUserDoorFlag(ID)" - Reads whether door is opened.

Parameter: ID - Identifier of a device.

Return values: Result of operation and CR+LF.

"1" - User's door is closed.

"0" - User's door is opened.

"-1" - Error.

### 1.2.7 STX2ReadShovelDetector

"STX2ReadShovelDetector(ID)" - Reads whether plate is present on the Shovel. If Plate Shovel Detector is not assigned in Unit configuration file section Sensor Configuration value PlateShovelSensor=1 this function returns value "0" by default.

Parameter: ID – Identifier of a device.

Return values: Result of operation and CR+LF.

"1" - Plate presents on the Shovel.

"0" - Plate doesn't present on the Shovel.

"-1" - Error.

### 1.2.8 STX2ReadXferStationDetector1

"STX2ReadXferStationDetector1(ID)" - Reads whether plate is present on the Transfer Station. If Plate Station Detector is not assigned in Unit configuration file section Sensor Configuration value PlateXferStSensor1=1 this function returns value "0" by default.

Parameter: ID – Identifier of a device.

Return values: Result of operation and CR+LF.

"1" - Plate is on the Transfer Station.

"0" - Plate is not on the Transfer Station.

"-1" - Error.

### 1.2.9 STX2ReadXferStationDetector2

"STX2ReadXferStationDetector2(ID)" - Reads whether plate is present on the second Transfer Station. If Second Plate Station Detector is not assigned in Unit configuration file section Sensor Configuration value PlateXferStSensor2=1 this function returns value "0" by default.

Parameter: ID – Identifier of a device.

Return values: result of operation and CR+LF.

"1" - Plate is on the Transfer Station.

"0" - Plate is not on the Transfer Station.

"-1" - Error.

### 1.2.10 STX2BeeperOn

"STX2BeeperOn(ID)" – Turns Beeper Alarm on.

Parameter: ID – Identifier of a device.

Return values: CR+LF.

### 1.2.11 STX2BeeperOff

"STX2BeeperOff(ID)" – Turns Bepper Alarm off.

Parameter: ID – Identifier of a device.

Return values: CR+LF

### 1.2.12 STX2ReadBarcodeAtTransferStation

"STX2ReadBarcodeAtTransferStation(ID)" – Reads barcode of the Plate at TransferStation.

Parameter: ID – Identifier of a device.

Return values: Result of operation and CR+LF.

"BCRError" – Barcode reader is not initialised.

"InitError" – StoreX is not initialized.

"Device Status Error" – Device has the Error Status.

"Device not Ready" – Device not ready.

"Error" – Error at the time of reading Barcode.

"No Barcode" – There is no Barcode on the Plate.

## **2 StoreX Network Communication**

## 2.1 Client Socket example (C#)

### 2.1.1 Client Socket class

```
public class StateObject
{
    public Socket workSocket = null;
    public const int BufferSize = 256;
    public byte[] buffer = new byte[BufferSize];
    public StringBuilder sb = new StringBuilder();
}

public class AsynchronousClient
{
    private Socket client;

    // The port number for the remote device.
    //private const int port = 3336;

    // ManualResetEvent instances signal completion.
    private ManualResetEvent connectDone = new ManualResetEvent(false);
    private ManualResetEvent sendDone = new ManualResetEvent(false);
    private ManualResetEvent receiveDone = new ManualResetEvent(false);

    // The response from a remote Storex Server.
    private String response = String.Empty;

    public Socket getSocket()
    {
        return client;
    }

    public void startClient(String STXIPAddress, int STXport)
    {
        // Connect to a remote device.
        try
        {
            // Establish the remote endpoint for the socket.
            IPHostEntry ipHostInfo = Dns.Resolve(STXIPAddress);
            IPAddress ipAddress = ipHostInfo.AddressList[0];
            IPEndPoint remoteEP = new IPEndPoint(ipAddress, STXport);

            // Create a TCP/IP socket.
            client = new Socket(AddressFamily.InterNetwork,
                                SocketType.Stream, ProtocolType.Tcp);

            // Connect to the remote endpoint.
            client.BeginConnect(remoteEP,
                                new AsyncCallback(ConnectCallback), client);
            connectDone.WaitOne();
        }
        catch (Exception e)
        {
            Console.WriteLine(e.ToString());
        }
    }

    public void stopClient()
```

```
{
    // Release the socket.
    client.Shutdown(SocketShutdown.Both);
    client.Close();
}

private void ConnectCallback(IAsyncResult ar)
{
    try
    {
        // Retrieve the socket from the state object.
        Socket client = (Socket) ar.AsyncState;

        // Complete the connection.
        client.EndConnect(ar);

        Console.WriteLine("Socket connected to {0}",
            client.RemoteEndPoint.ToString());

        // Signal that the connection has been made.
        connectDone.Set();
    }
    catch (Exception e)
    {
        Console.WriteLine(e.ToString());
    }
}

private void Receive(Socket client)
{
    try
    {
        // Create the state object.
        StateObject state = new StateObject();
        state.workSocket = client;

        // Begin receiving the data from the remote device.
        client.BeginReceive( state.buffer, 0, StateObject.BufferSize, 0,
            new AsyncCallback(ReceiveCallback), state);
    }
    catch (Exception e)
    {
        Console.WriteLine(e.ToString());
    }
}

private void ReceiveCallback( IAsyncResult ar )
{
    if (!this.client.Connected) return;

    try
    {
        StateObject state = (StateObject) ar.AsyncState;
        Socket client = state.workSocket;

        int bytesRead = client.EndReceive(ar);

        if (bytesRead > 0)
        {
            // There might be more data, so store the data received so far.
            state.sb.Append(Encoding.ASCII.GetString(state.buffer,0,
                bytesRead));
        }
    }
}
```

```
// Get the rest of the data.
client.BeginReceive(state.buffer,0,StateObject.BufferSize,
    0,new AsyncCallback(ReceiveCallback), state);

response = ncoding.ASCII.GetString(state.buffer,0,bytesRead);
Console.WriteLine("Received from STX: "+response);
receiveDone.Set();
}
else
{
    // All the data has arrived; put it in response.
    if (state.sb.Length > 1)
    {
        response = state.sb.ToString();
    }

    // Signal that all bytes have been received.
    receiveDone.Set();
}
}
catch (Exception e)
{
    Console.WriteLine(e.ToString());
}
}

public String Send(String data)
{
    response = "";

    // Convert the string data to byte data using ASCII encoding.
    byte[] byteData = Encoding.ASCII.GetBytes(data);

    // Begin sending the data to the remote device.
    client.BeginSend(byteData, 0, byteData.Length, 0,
        new AsyncCallback(SendCallback), client);

    Console.WriteLine("Sent STRING to STX server: "+data);

    Receive(client);
    receiveDone = new ManualResetEvent(false);
    receiveDone.WaitOne();

    return response;
}

private void SendCallback(IAsyncResult ar)
{
    try
    {
        // Retrieve the socket from the state object.
        Socket client = (Socket) ar.AsyncState;

        // Complete sending the data to the remote device.
        int bytesSent = client.EndSend(ar);

        Console.WriteLine("Sent {0} bytes to server.", bytesSent);

        // Signal that all bytes have been sent.
        sendDone.Set();
    }
}
```

```
        }  
        catch (Exception e)  
        {  
            Console.WriteLine(e.ToString());  
        }  
    }  
} // End of Socket's client class
```

### 2.1.2 Using an Client Socket

```
String STXId = "STX";  
String STXAddress = "192.168.1.1";  
int STXPort = 3336;  
  
AsynchronousClient STXSCK = new STX_TCP_Client.AsynchronousClient();  
STXSCK.StartClient(STXAddress, STXPort);  
String ResActivate = STXSCK.Send("STX2Activate("+STXId+")"+(char)13);  
if ((ResActivate == "1;1") || (ResActivate == "1"))  
{  
    MessageBox.Show("Device is initialised!", "1", MessageBoxButtons.OK,  
        MessageBoxIcon.Information);  
}  
else  
{  
    MessageBox.Show("Initialisation Error!", "1", MessageBoxButtons.OK,  
        MessageBoxIcon.Error);  
}  
  
STXSCK.StopClient();
```