LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY

LIGO Laboratory / LIGO Scientific Collaboration

LIGO- T1100607-v1 *advanced LIGO* 12/2/11

EtherCAT Setup of Modbus Devices

Daniel Sigg

Distribution of this document:
LIGO Scientific Collaboration

This is an internal working note
of the LIGO Laboratory.

California Institute of Technology
LIGO Project – MS 18-34
1200 E. California Blvd.
Pasadena, CA 91125
Phone (626) 395-2129
Fax (626) 304-9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project – NW22-295
185 Albany St
Cambridge, MA 02139
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

LIGO Hanford Observatory
P.O. Box 159
Richland WA 99352
Phone 509-372-8106
Fax 509-372-8137

LIGO Livingston Observatory
P.O. Box 940
Livingston, LA 70754
Phone 225-686-3100
Fax 225-686-7189

http://www.ligo.caltech.edu/

1 Introduction

This document describes the setup of the D1100251, the 384 Channel Acromag Binary Output chassis, and the HMS AB9000, Anybus X-gateway Modbus-TCP for EtherCAT. The D1100251 contains 4 Acromag ES2113-0100 units. These are 96 channel binary input/output modules that are controlled through a Modbus-TCP interface. Combining these units with the Anybus X-gateway makes the IO channels transparently accessible through EtherCAT. Look for data sheets, manuals, application notes and setup software in C1107420.

2 Setting up the ES2113

The first step is to set up the IP address. Look up the available addresses in E1101144. Looking from the front into the D1100251 chassis, the PCB 1 is on the top left. The PCB 2 is on the top right, the PCB 3 is on the bottom left and PCB 4 is on the bottom right. Locate the manual and the application note for the ES2113 in C1107420.

By default each ES2113 has an IP address of 128.1.1.100. Hook up the Ethernet of the first unit to a computer and make sure its IP address is 128.1.1.111. Try to open a web page with http://128.1.1.100. If this doesn't work, one may have to factory reset the unit. For this turn it off, pull the toggle switch to the up position and turn on the power. Now hold the toggle switch in the up position for at least 10 seconds. After releasing the unit should have reset itself to 128.1.1.100.

No make sure the network configuration page looks like Figure 1. The default user name and password are User/password. Do not change this. Make sure you have the correct IP address and subnet mask. For the gateway use the "X.X.X.1" address. We are using a static IP address and ports are operated in hub configuration. You can use the wink on/off button to turn on/off a blinking green LED at the front of the Acromag unit. This is to make sure that the correct PCB is selected.

The IO configuration page should be left in the default position after factory reset as seen in Figure 2.

Continue this procedure for all 4 PCBs. Finally, daisy chain the Ethernet of all 4 PCBs and hook it to the rear panel CAT5 feedthroughs.

Add a label to the D1100251 that lists the IP addresses as configured.

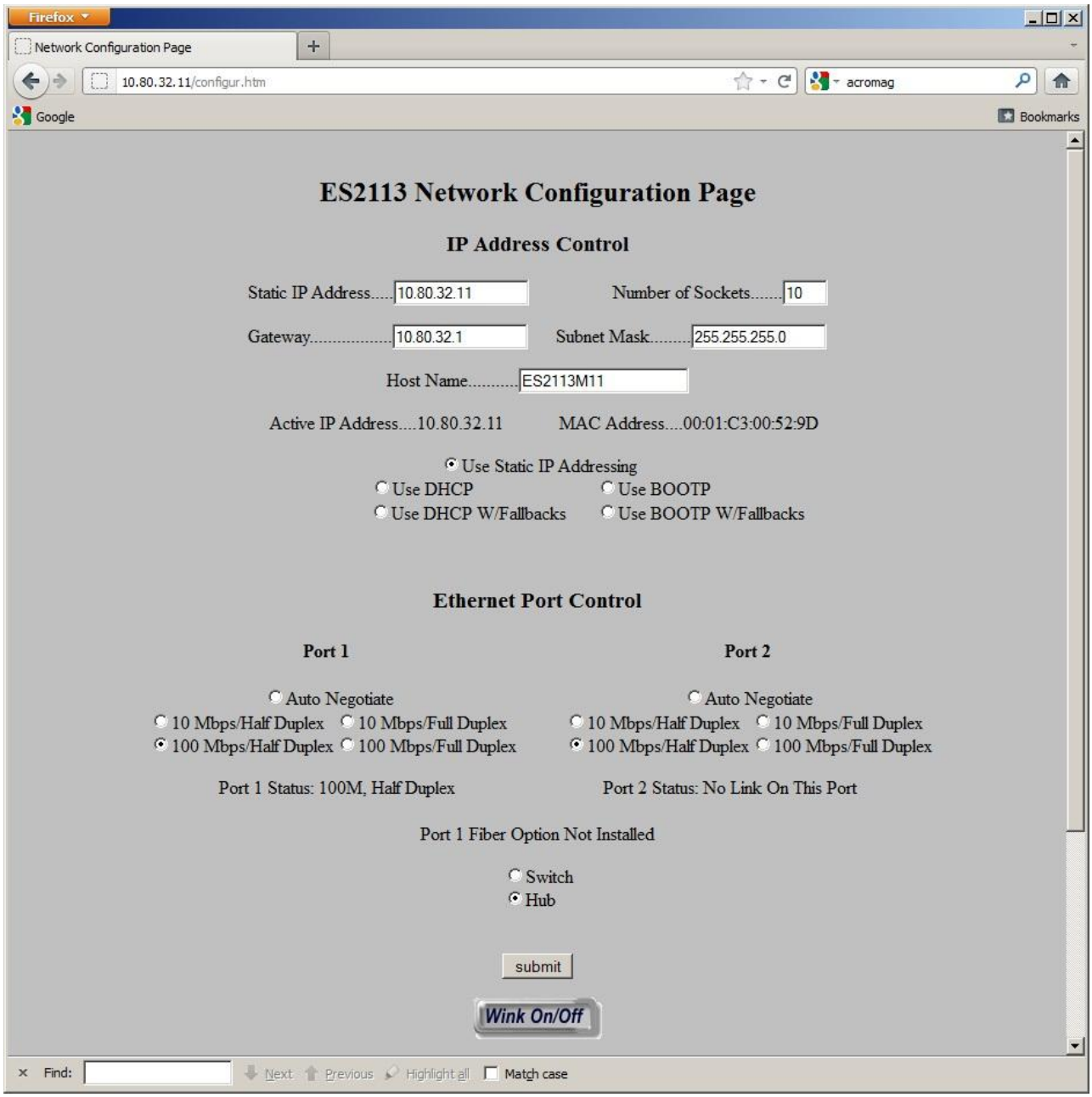

Figure 1: ES2113 network configuration.

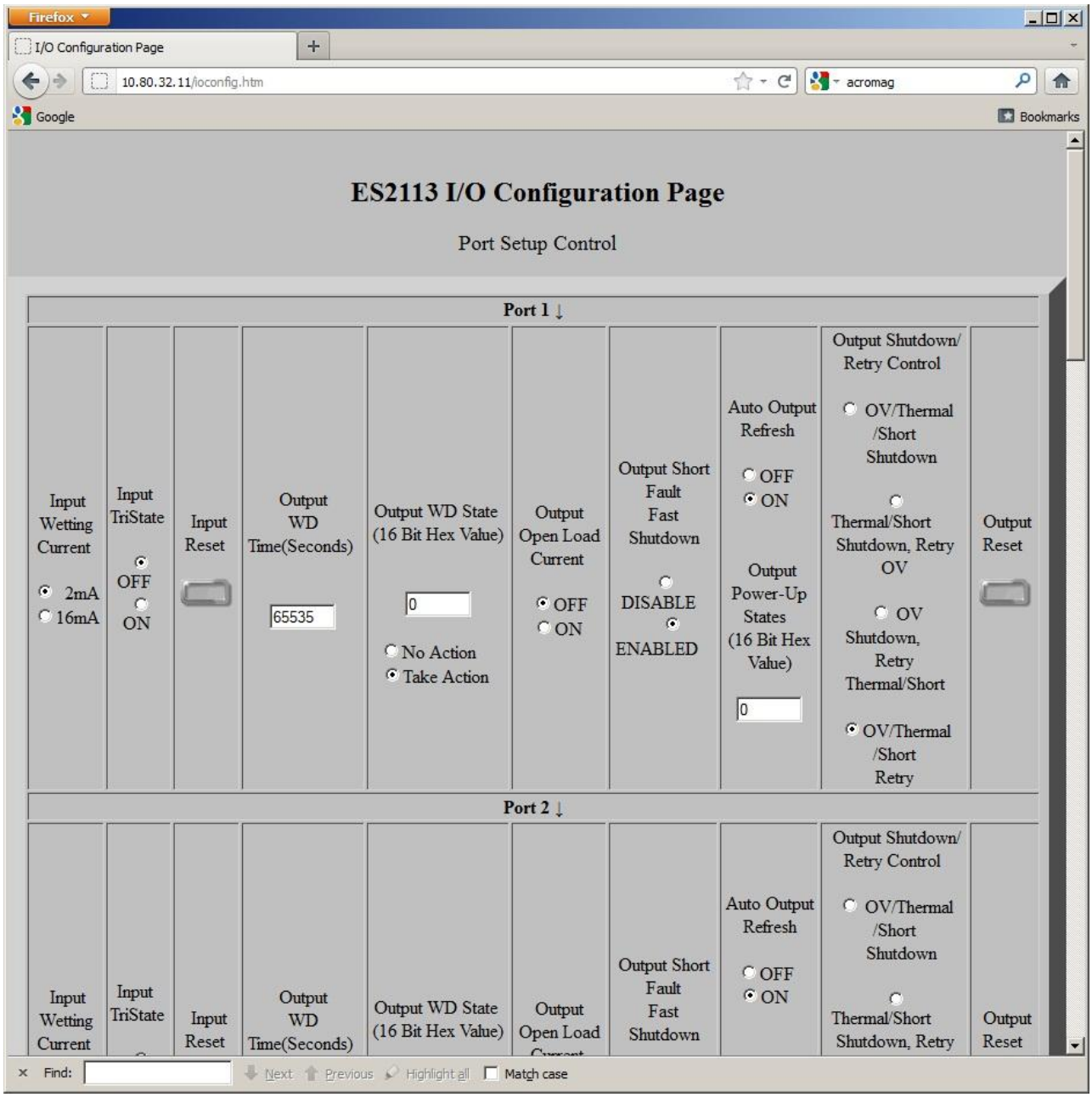

Figure 2: ES2113 IO Configuration page.

3 Setting up the AB9000

Locate the user manual for the AB9000 in C1107420. Make sure the AB9000 contains a SD card. Then, hook up the Ethernet of the Modbus side of the AB9000 to a computer. By default its IP address should be set to 192.168.0.100. You can use the Anybus IPconfig utility, to locate this address (part of C1107420). This utility may also allow you to change the IP address of the unit temporarily. If so, set it to the same subnet as the ES2113 units but with the address ".1". If not set the computer network port to the same subnet as the AB9000 but a different address, say

192.168.0.111. Now, point a web browser to 192.168.0.100. It should somewhat similar to Figure 3.

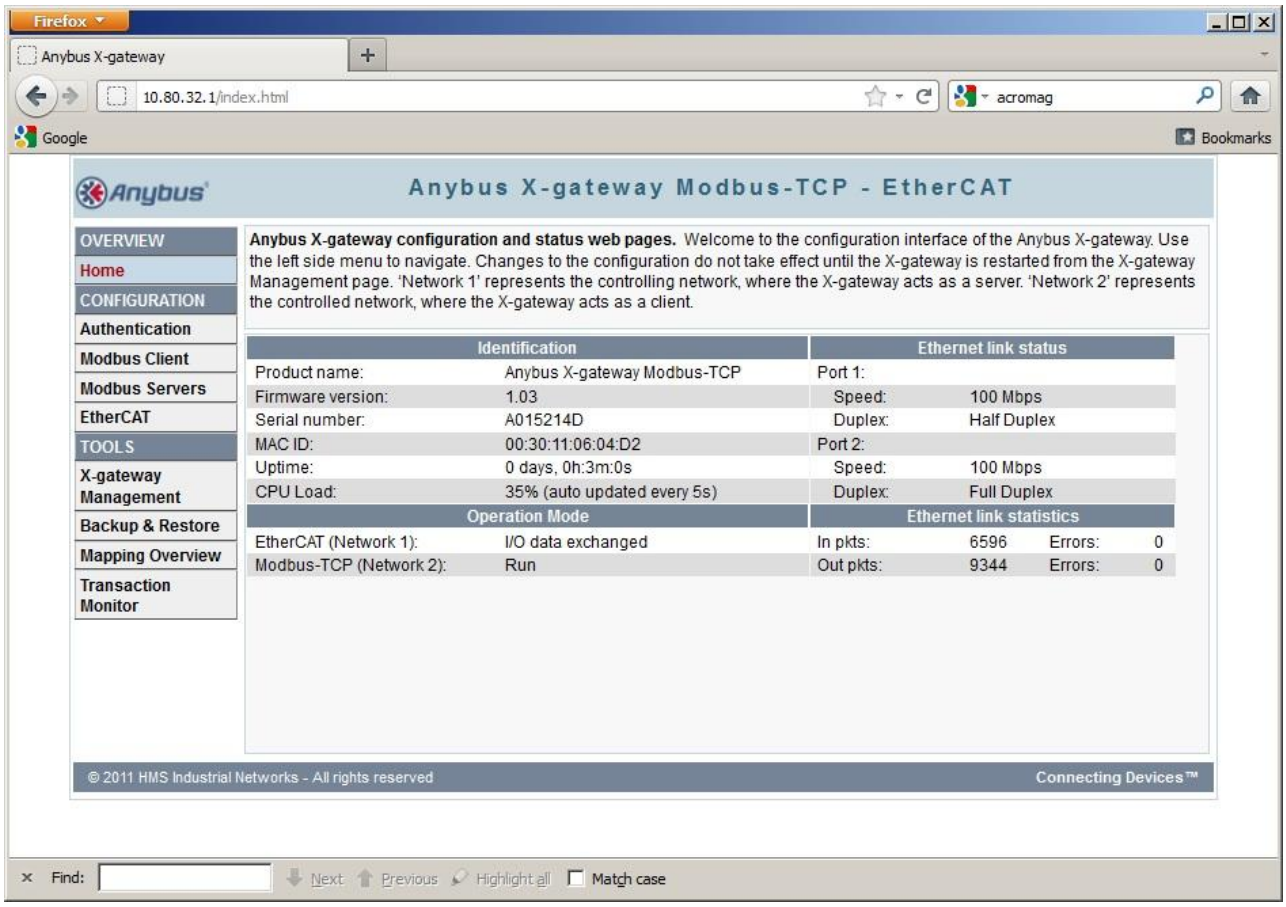

Figure 3: Home page of the AB9000.

Select the Modbus client page and make sure to change the IP address, subnet mask, DHCP setting, HICP, start-up operation mode and freeze setting so it looks like in Figure 4 but with the correct IP address. Save the settings.

Select the Modbus server page. Add the first Modbus server which should correspond to PCB 1 of D1100251. Leave the port and protocol at 502 and TCP, respectively. Now select the transaction link and add 3 transaction as shown in Figure 6. Go back to the server list and add the next server for PCB 2. Add the same transactions and repeat adding servers for PSB 3 and 4. The final page should look similar to Figure 5. Save the settings.

Select the EtherCAT page and enable the mapped live list. The setup should look like Figure 7.

Finally, go to the X-gateway management page and store the settings and reboot. If the IP address was changed, the new IP address will go into effect after the reboot of the AB9000. Make sure the computer network port is set accordingly. Now, connect the D1100251 Modbus Ethernet port to the second Modbus port of the AB9000. It should now be possible to look at the web pages of all 5 devices.

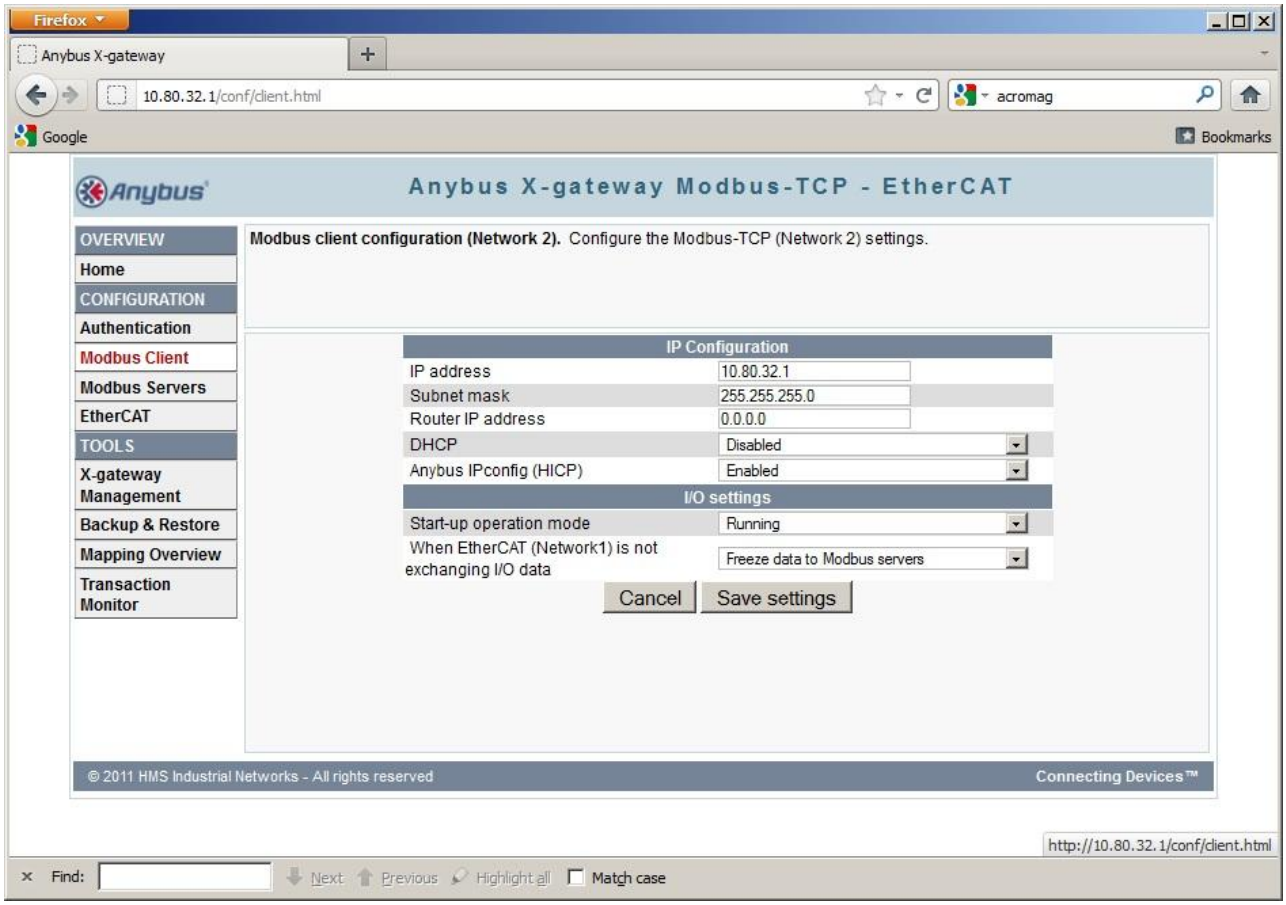

Figure 4: AB9000 Modbus client setup.

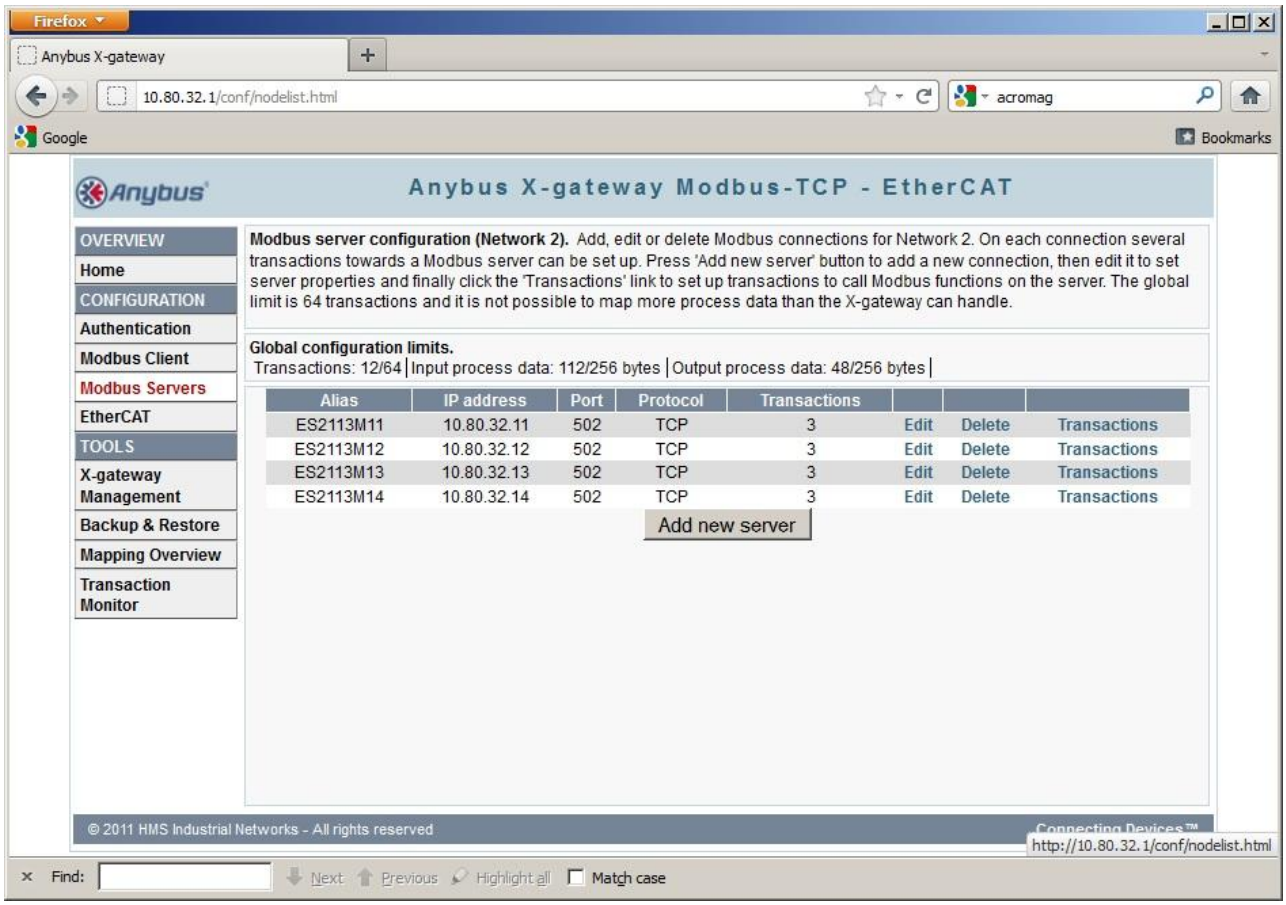

Figure 5: AB9000 Modbus server setup.

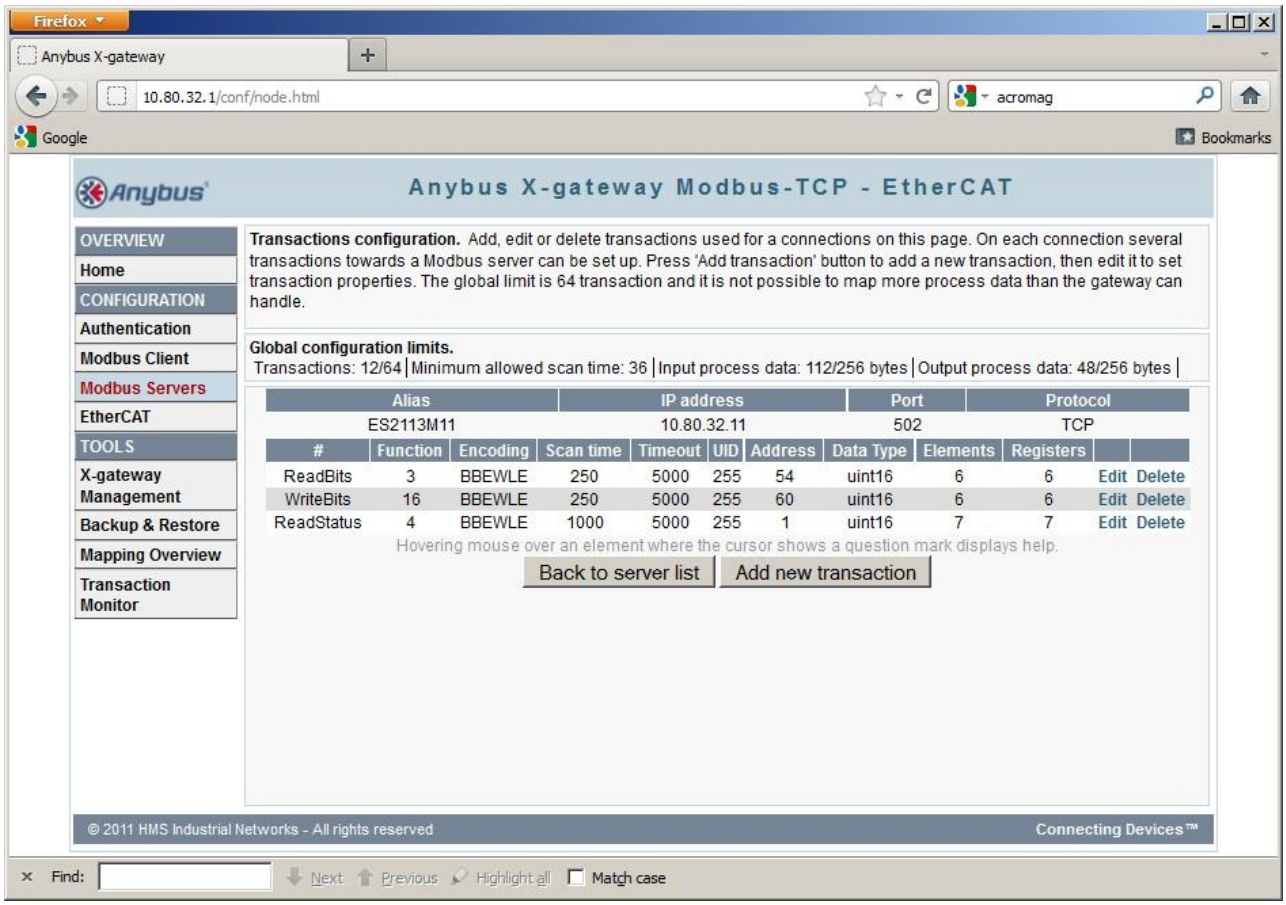

Figure 6: AB9000 Modbus server transaction setup.

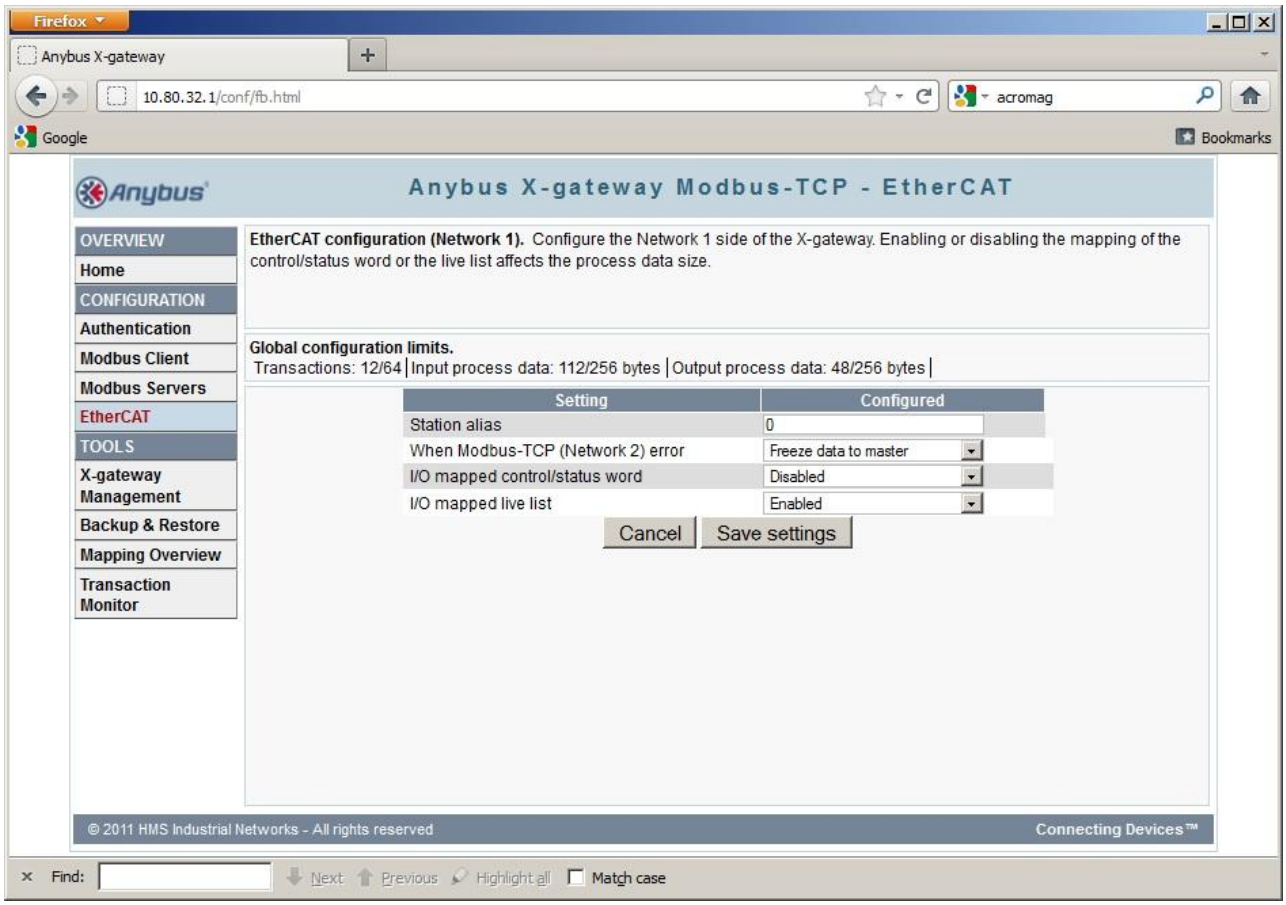

Figure 7: AB9000 EtherCAT setup.

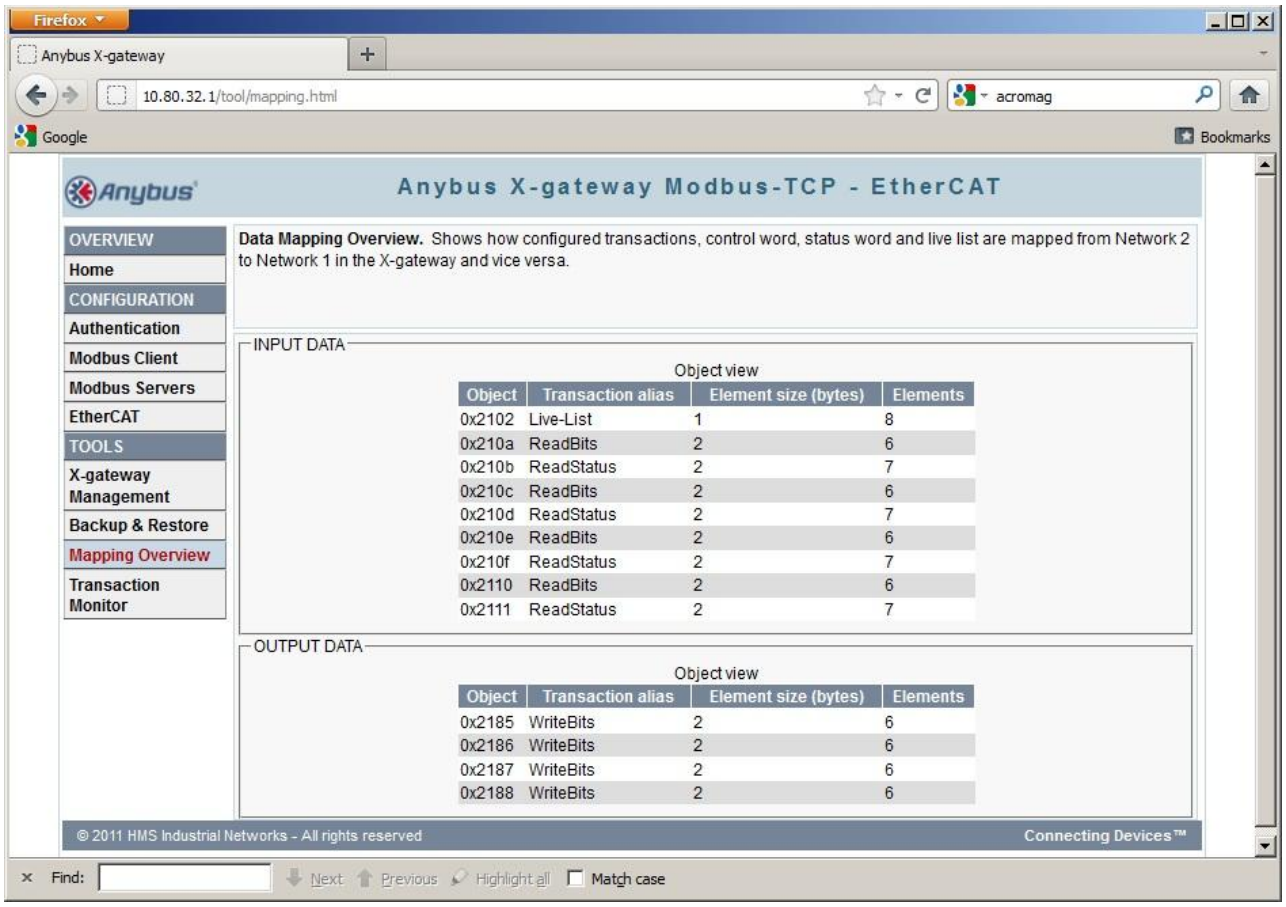

Figure 8: AB9000 Mapping.

Check the mapping on the AB9000 using the mapping overview page. It should look like Figure 8.

Up to 2 D1100251 can be connected to a single AB9000. Repeat the setup for the D1100251 but make sure to use a different set of IP addresses on the same subnet. Then, move to the Modbus server page on the AB9000 and add 4 corresponding server entries.

4 Setting up TwinCAT

Locate the EtherCAT slave TwinCAT application note as well as the ABXS_ECT file in C1107420. The ABXS_ECT file can also be downloaded from the HMS web page. This may be necessary if an AB9000 with a more recent firmware release is used. Copy the ABXS_ECT file to "C:\Program files\TwinCAT\Io\EtherCAT\" before you start the TwinCAT system manager. Start the TwinCAT system manager.

Connect a EtherCAT output port of the computer to the EtherCAT input port of the AB9000. Make sure the EtherCAT NIC is recognized by the TwinCAT system manager. Now you should be able to scan for new devices and recognize the AB9000. Go to its process data tab and load the PDO information from the device. Now the input and output variables should be available. They have intuitive names likes SubIndex 001, etc. Try to go to active run. This should start the data transfer.

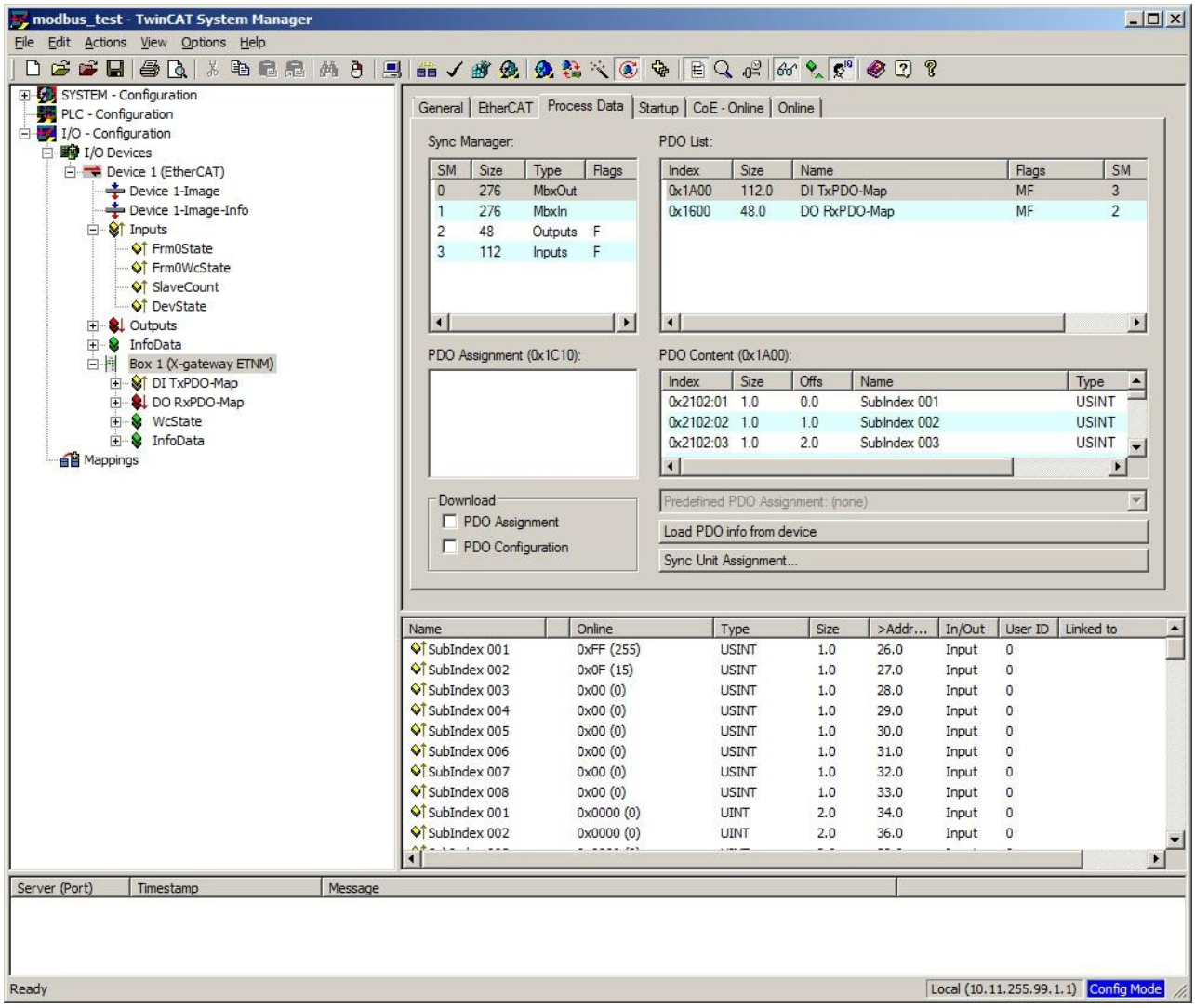

Figure 9: TwinCAT system manager.

Starting (active) run mode sometimes fails after the AB9000 has been reconfigured. The easiest remedy is to delete the X-gateway box and rescan the EtherCAT chain.

Finally, go back to the AB9000 web page and check the home page for errors. When working, it should look like Figure 3. You can also look the transaction monitor page and check for errors, see Figure 10.

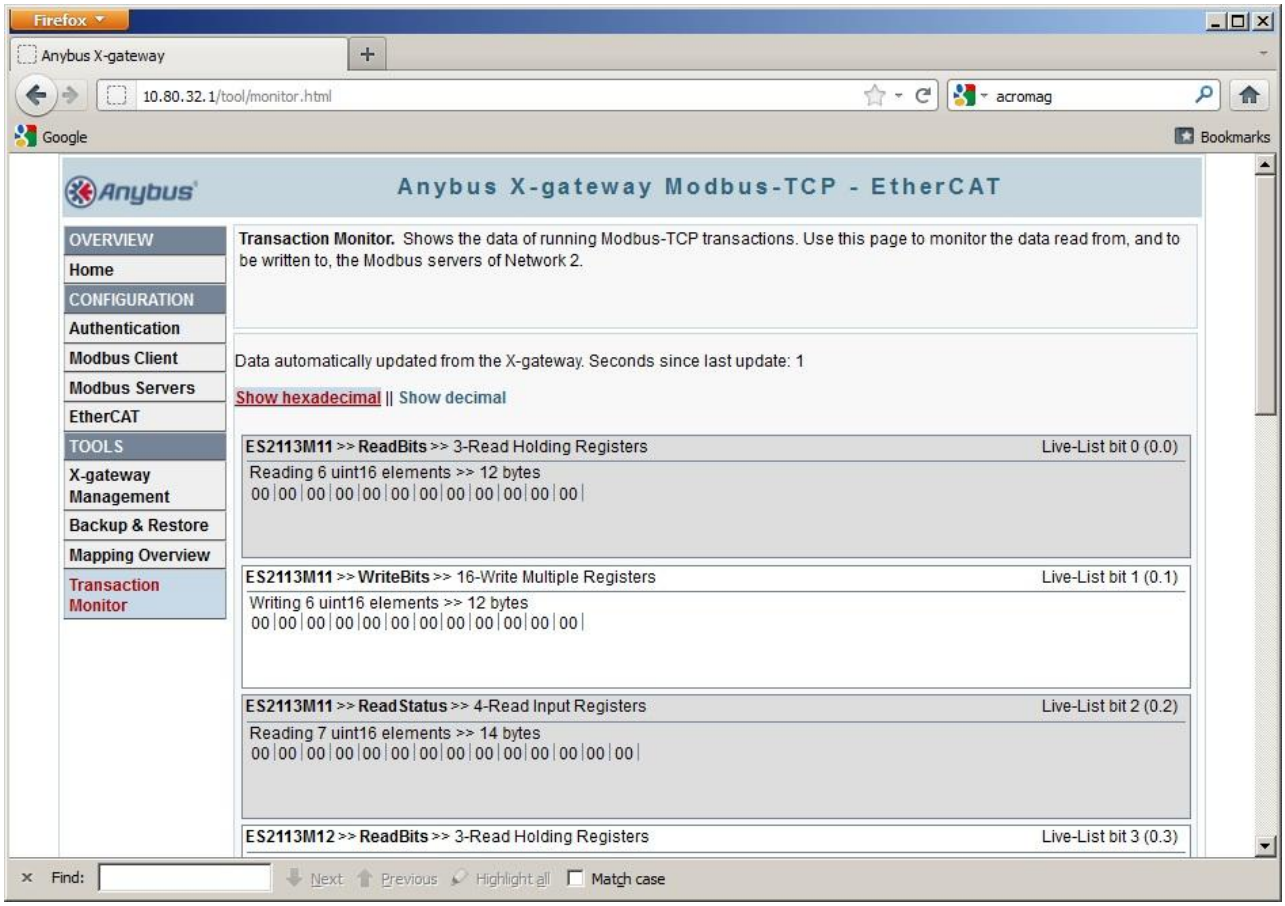

Figure 10: AB9000 transaction monitor.

5 PLC Programming with TwinCAT

A TwinCAT example project with a single AB9000 and a single D1100251 can be found in the zip associated with this document. The example code uses three main structures for the hardware input, the hardware outputs and the user interface, respectively.

```
TYPE _IscWhiteningIn :
STRUCT
      LiveList:         ARRAY[1..8] OF BYTE;
      PCB:              ARRAY[1..4,1..13] OF WORD;
      InfoDataState:    WORD;
END_STRUCT
END_TYPE

TYPE _IscWhiteningOut :
STRUCT
      PCB:              ARRAY[1..4,1..6] OF WORD;
END_STRUCT
END_TYPE

TYPE IscWhitening :
STRUCT
      Chassis:          ARRAY[1..6] OF IscWhiteningChassis;
END_STRUCT
END_TYPE

TYPE IscWhiteningChannel :
STRUCT
      Valid:            BOOL; (* read only *)
      Switches:         BYTE; (* read only *)
      Toggle:           BYTE;
      SetVal:           BYTE;
END_STRUCT
END_TYPE

TYPE IscWhiteningChassis :
STRUCT
      Channels:         ARRAY[1..8] OF IscWhiteningChannel;
END_STRUCT
END_TYPE
```

Figure 11: Program structures and sub structures.

A function block is used to transfer data from the input and output structures to the user structure. The mapping of IO channels to the variables is described in T1100195-v1. Two functions are used to obtain the channel and chassis index, respectively. The main program is straight forward and just calls the function block using global variables for the main structures.

```
FUNCTION_BLOCK IscWhiteningFB
VAR_INPUT
      In:         _IscWhiteningIn;
END_VAR
VAR_OUTPUT
      Out:        _IscWhiteningOut;
END_VAR
VAR_IN_OUT
      Val:        IscWhitening;
END_VAR
VAR_TEMP
      pcb:        INT;
      port:       INT;
      chassis:    INT;
      chn:        INT;
      LiveVal:    WORD;
      OutVal:     WORD;
END_VAR

FOR pcb := 1 TO 4 DO
      FOR port := 1 TO 6 DO
            OutVal := 0;
            LiveVal := SHR (BYTE_TO_WORD (In.LiveList[1]) OR
                        SHL (BYTE_TO_WORD (In.LiveList[2]), 8),
                        3*(pcb-1)) AND 16#0007;
            chassis := IscWhiteningChassisIndex (pcb, port, TRUE);
            chn := IscWhiteningChannelIndex (pcb, port, TRUE);
            Val.Chassis[chassis].Channels[chn].Switches :=
                  WORD_TO_BYTE (In.PCB[pcb,port] AND 16#00FF);
            Val.Chassis[chassis].Channels[chn].Valid :=
                  (LiveVal = 16#0007) AND
                  ((In.InfoDataState AND 16#3F) = 8) AND
                  ((In.PCB[pcb,port+7] AND 16#0002) = 0);
            Val.Chassis[chassis].Channels[chn].SetVal :=
                  Val.Chassis[chassis].Channels[chn].SetVal XOR
                  Val.Chassis[chassis].Channels[chn].Toggle;
            Val.Chassis[chassis].Channels[chn].Toggle := 0;
            OutVal := BYTE_TO_WORD
                  (Val.Chassis[chassis].Channels[chn].SetVal);
            chassis := IscWhiteningChassisIndex (pcb, port, FALSE);
            chn := IscWhiteningChannelIndex (pcb, port, FALSE);
            Val.Chassis[chassis].Channels[chn].Switches :=
                  WORD_TO_BYTE (SHR (In.PCB[pcb,port] AND 16#FF00, 8));
            Val.Chassis[chassis].Channels[chn].Valid :=
                  (LiveVal = 16#0007) AND
                  ((In.InfoDataState AND 16#3F) = 8) AND
                  ((In.PCB[pcb,port+7] AND 16#0002) = 0);
            Val.Chassis[chassis].Channels[chn].SetVal :=
                  Val.Chassis[chassis].Channels[chn].SetVal XOR
                  Val.Chassis[chassis].Channels[chn].Toggle;
            Val.Chassis[chassis].Channels[chn].Toggle := 0;
            OutVal := OutVal OR SHL (BYTE_TO_WORD
                  (Val.Chassis[chassis].Channels[chn].SetVal), 8);
            Out.PCB[pcb,port] := OutVal;
      END_FOR;
END_FOR;
END_FUNCTION_BLOCK
```

Figure 12: Function block.

```
FUNCTION IscWhiteningChannelIndex : INT
VAR_INPUT
      PCB:        INT;
      Port:       INT;
      LSB:        BOOL;
END_VAR
VAR
END_VAR

CASE PCB OF
      1:    CASE Port OF
                  1 : IscWhiteningChannelIndex := 1;
                  2 : IscWhiteningChannelIndex := 3;
                  3 : IscWhiteningChannelIndex := 5;
                  4 : IscWhiteningChannelIndex := 1;
                  5 : IscWhiteningChannelIndex := 3;
                  6 : IscWhiteningChannelIndex := 5;
            ELSE
                  IscWhiteningChannelIndex := 0;
            END_CASE;
      2:    CASE Port OF
                  1 : IscWhiteningChannelIndex := 5;
                  2 : IscWhiteningChannelIndex := 7;
                  3 : IscWhiteningChannelIndex := 1;
                  4 : IscWhiteningChannelIndex := 5;
                  5 : IscWhiteningChannelIndex := 7;
                  6 : IscWhiteningChannelIndex := 1;
            ELSE
                  IscWhiteningChannelIndex := 0;
            END_CASE;
      3:    CASE Port OF
                  1 : IscWhiteningChannelIndex := 7;
                  2 : IscWhiteningChannelIndex := 1;
                  3 : IscWhiteningChannelIndex := 3;
                  4 : IscWhiteningChannelIndex := 7;
                  5 : IscWhiteningChannelIndex := 1;
                  6 : IscWhiteningChannelIndex := 3;
            ELSE
                  IscWhiteningChannelIndex := 0;
            END_CASE;
      4:    CASE Port OF
                  1 : IscWhiteningChannelIndex := 3;
                  2 : IscWhiteningChannelIndex := 5;
                  3 : IscWhiteningChannelIndex := 7;
                  4 : IscWhiteningChannelIndex := 3;
                  5 : IscWhiteningChannelIndex := 5;
                  6 : IscWhiteningChannelIndex := 7;
            ELSE
                  IscWhiteningChannelIndex := 0;
            END_CASE;
      ELSE
            IscWhiteningChannelIndex := 0;
END_CASE;
IF (NOT LSB AND (IscWhiteningChannelIndex > 0)) THEN
      IscWhiteningChannelIndex := IscWhiteningChannelIndex + 1;
END_IF;
END_FUNCTION
```

Figure 13: Channel index function.

```
FUNCTION IscWhiteningChassisIndex : INT
VAR_INPUT
      PCB:        INT;
      Port:       INT;
      LSB:        BOOL;
END_VAR
VAR
END_VAR

CASE PCB OF
      1 :   CASE Port  OF
                  1..3 : IscWhiteningChassisIndex := 4;
                  4..6 : IscWhiteningChassisIndex := 1;
            ELSE
                  IscWhiteningChassisIndex := 0;
            END_CASE;
      2 :   CASE Port  OF
                  1..2 : IscWhiteningChassisIndex := 5;
                  3 :    IscWhiteningChassisIndex := 6;
                  4..5 : IscWhiteningChassisIndex := 2;
                  6 :    IscWhiteningChassisIndex := 3;
            ELSE
                  IscWhiteningChassisIndex := 0;
            END_CASE;
      3 :   CASE Port  OF
                  1 :    IscWhiteningChassisIndex := 4;
                  2..3 : IscWhiteningChassisIndex := 5;
                  4 :    IscWhiteningChassisIndex := 1;
                  5..6 : IscWhiteningChassisIndex := 2;
            ELSE
                  IscWhiteningChassisIndex := 0;
            END_CASE;
      4:    CASE Port  OF
                  1..3 : IscWhiteningChassisIndex := 6;
                  4..6 : IscWhiteningChassisIndex := 3;
            ELSE
                  IscWhiteningChassisIndex := 0;
            END_CASE;
      ELSE
            IscWhiteningChassisIndex := 0;
END_CASE;
END_FUNCTION
```

Figure 14: Chassis index function.

```
VAR_GLOBAL
      Whitening1In      AT %IB0:    IscWhiteningIn;
      Whitening1Out     AT %QB0:    IscWhiteningOut;
      Whitening1:                   IscWhitening;
END_VAR

PROGRAM MAIN
VAR
      Whitening1FB:     IscWhiteningFB;
      Counter:          INT := 200;
END_VAR

Whitening1FB (In := _Whitening1In, Out => _Whitening1Out,
              Val := Whitening1);

(* do some testing *)
IF (Counter <= 0) THEN
      Whitening1.Chassis[1].Channels[1].Toggle := 1;
      Counter := 200;
ELSIF (Counter = 100) THEN
      Whitening1.Chassis[5].Channels[7].Toggle := 16#F0;
      Counter := Counter - 1;
ELSE
      Counter := Counter - 1;
END_IF;

END_PROGRAM
```

Figure 15: Main program.