```
1   PROGRAM PLC_PRG
2   VAR
3       state       : UINT ;
4       RFID        : DC_ECP . Dtm425 ;
5       timer       : TON ;
6       RFIDdata    : DC_ECP . st_CPF_RfidData ;
7       correctID   : UINT ;
8       tcpClient   : DC_ECP . TcpClientByteStream ;
9       varDataIn   : STRING ;
10      timeData    : DATE_AND_TIME ;
11      getTime     : DTU . GetDateAndTime ;
12      timeString  : STRING ;
13      xmlString   : STRING [ 511 ] ;
14      waitTime    : TIME ;
15      delayString : STRING ;
16      iAlive      : INT ;
17      bBool       : BOOL := TRUE ;
18      prevID      : UINT ;
19
20      bBool2      : BOOL := TRUE ;
21  END_VAR
22  VAR RETAIN
23      testString  : STRING ;
24  END_VAR
25
```

```
1   //0. Init connect til server og start bånd etter
2   //1. Wait for carrier
3   //2. Når funnet stop carrier
4   //3. læs RFID and get time and date
5   //4. skriv XML String
6   //5. When ready send XML String
7   //6. Vent på svar fra TCP
8   //7. Læs svar (Det er ventetid i ms)
9   //8. Vent antal ms og kør
10  //9. bonks?
11
12
13  CASE state OF
14      0 :
15
16      // Here the PLC connects to the Server, make sure to change the ip and port
    to correct ones
17      // we only want to run this once otherwise the connection will be messed up
18      IF bBool THEN
19
20          tcpClient . Connect ( sIP := '172.20.66.106' , uiPort := 6666 ) ;
21          bBool := FALSE ;
22      END_IF
23
```

```
24
25          io . xQA1_RIGHT  :=  TRUE ;  // Start the track
26          io . xMB20  :=  FALSE ;  // Pull the stopprt up
27          RFID . ClearError ( ) ;  // RFID stuff, don't know if it's important
28
29          // Here we are checking whether we made the connection to the TCP server or
        not
30          // We also check whether the connection is ready to send/recieve
31          IF  tcpClient . xConnected  AND  tcpClient . xReady  THEN
32              state  :=  1 ;
33          END_IF
34
35
36
37
38          1 :
39          // Here we are checking if the induction sensor can see the carrier
40          // If it cant the stopper is pulled up and the state changes
41          IF  io . xBG21  =  TRUE  THEN
42              io . xMB20  :=  FALSE ;
43              state  := 2 ;
44          END_IF
45
46          // Here we connect to the RFID
47          // The Node id can be found under CAN/CANopen/DTM425
48          // The NetworkID can be found under CAN
49          // The channel can be found under CAN/CANopen/DTM425, Press the SDO Channels
        button and look for the channel number
50
51          2 :
52          IF  RFID . xReady  THEN
53              RFID . Connect ( usiNodeId  :=  32 , usiNetworkID  :=  0 ,  usiChannel  :=  1 )
        ;
54              state  :=  25 ;
55          END_IF
56
57          // This state makes sure the RFID is ready to be read before it switches to
        the next state
58          25 :
59              IF  RFID . xReady  THEN
60                  state  :=  3 ;
61              END_IF
62
63          // This state reads the RFID on the carrier and places the data into the
        RFIDdata variable
64          3 :
65          IF  RFID . xReady  THEN
66              RFID . ReadTag ( uiStartAddress  :=  0 ,  uiDataLength  :=  SIZEOF ( RFIDdata
        ) ,  pData := ADR ( RFIDdata ) ) ;
67              state  :=  35 ;
68
```

```
69          END_IF
70
71          // The carrier id intially saved is not the numerically correct number
72          // Therefore we are using this SwapWORD function to get the correct carrier
        ID
73          35 :
74              IF  RFID . xReady  THEN
75                  correctID  :=  DC_ECP . SwapWORD ( RFIDdata . uiCarrierID ) ;
76                  state  :=  375 ;
77              END_IF
78
79          // Here the save the current time and date into timeData
80          // timeData is a time data type so we convert it to a string (timeString) so
         we can send it with the XML
81          375 :
82              getTime . xExecute  :=  TRUE ;
83              IF  getTime . xDone  THEN
84                  timeData  :=  getTime . dtDateAndTime ;
85                  timeString  :=  DT_TO_STRING ( timeData ) ;
86                  state  :=  4 ;
87              END_IF
88
89          // In this state we are concatenating our XML file
90          // We are sending our carrier ID, station ID and time and date
91          // note that the station ID is manually coded and not something we detect
92          4 :
93              xmlString  :=  '<?xml version="1.0" encoding="UTF-8" ?><RFID><carrierID>'
        ;
94              xmlString :=  CONCAT ( xmlString , UINT_TO_STRING ( correctID ) ) ;
95              xmlString  :=  CONCAT ( xmlString ,
        '</carrierID><stationID>STPLC_12</stationID><dateAndTime>' ) ;

96              xmlString  :=  CONCAT ( xmlString , timeString ) ;
97              xmlString  :=  CONCAT ( xmlString , '</dateAndTime></RFID>end' ) ;
98
99              state  :=  5 ;
100
101         5 :
102         // In this state we are sending the XML string to the TCP server
103         // So when the Client is ready it sends the data.
104         // Note that even if not all the bytes allocated is used they still gets
        send with the XML string
105         IF  tcpClient . xReady  THEN
106
107         tcpClient . Send ( ADR ( xmlString ) , SIZEOF ( xmlString ) ) ;
108         state  :=  6 ;
109         END_IF
110
111         // In this state we are waiting for the server to respond to our message
112         // When something is recieved, the date is put into the varDataIn string
113         6 :
```

```
114         IF tcpClient.xReady THEN
115             tcpClient.Receive(ADR(varDataIn),SIZEOF(varDataIn));
116             state := 65;
117         END_IF
118
119         // This is debug code that i'm to afraid to remove in case it kills the
        program
120         // but most likely it does nothing
121         65:
122
123             IF tcpClient.xReady THEN
124                 //varDataIn := tcpClient.sReceived;
125                 IF bBool2 THEN
126                     testString := varDataIn;
127                     bBool2 := FALSE;
128                 END_IF
129                 state := 7;
130             END_IF
131         // Here we have recieved a reply from the server in form of a string with
        the number of ms the timer should wait
132         // Then the delayString is created in the format of a time datatype
        (T#<number>ms)
133         7:
134         delayString := CONCAT('T#',varDataIn);
135         delayString := CONCAT(delayString,'ms');
136
137         state := 8;
138
139         // The delay string is then converted from string to time data type and used
         in our timer
140         // when the timer is done the program swtiches states
141         8:
142             waitTime := STRING_TO_TIME(delayString);
143
144             timer(IN:=TRUE,PT:=waitTime);
145             iAlive := iAlive+1;
146             IF timer.Q = TRUE THEN
147                 timer(IN := FALSE);
148                 state := 9;
149             END_IF
150
151         // Now the delay is done and the stopper is released
152         // After the stopper is pulled down the system waits until it can't detect
        the carrier anymore
153         // When the carrier is gone the stopper is pulled up again and the system
        goes back to state 1
154         9:
155         io.xMB20 := TRUE;
156         IF NOT io.xBG21 THEN
157             io.xMB20 := FALSE;
158             state := 1;
```

```
159             END_IF
160
161
162
163         END_CASE
164         // Cyclic process needed for the different things
165         timer ( ) ;
166         RFID ( ) ;
167         tcpClient ( ) ;
168         getTime ( ) ;
169
```