6.8.2003

NOKEVAL SCL

User Manual

PROTOCOL

Communication parameters

Every Nokeval device uses 8N1 bit protocol, i.e. 8 data bits, none parity, and one stop bit. Baud rates depend on device in concern, see its user manual. Recommended baud rate is 9600 bits per second.

The control characters presented below are one byte each, and every character (letter or number or space) in the command takes one byte as well.

Command frame

The actual command is composed of letters and numbers and is human readable. The commands recognized by a device are specified in its user manual; however some most common commands are represented in this manual also.

The command frame is obtained by adding some control bytes to the actual command:

ID	command	ETX	BCC

Start byte ID

The start byte is the only byte in SCL protocol that has the most significant bit set. This identifies that a completely new command frame is to come. The ID is formed from the device address (0-123) by setting the most significant bit, or by adding 128 (80h) to the address.

Address 126 (ID byte 126+128=254) is reserved for general call. The purpose of this is to provide a connection to a device the address of which is not known. When using this procedure, there should be only one device on the bus. This is supported by certain products only.

Note: The ID is one byte, not a series of separate numbers (e.g. ‘1’, ‘2’, and ‘8’).

Actual command

The command is sent as is. The command can be for example ”MEA CH 1 ?”.

End byte ETX

The command is terminated with single byte with value of 3 (03h), this is called ETX. Do not send Ascii character ‘3’.

Checksum byte BCC

The target device calculates the checksum and compares it to the BCC sent with the command to see if the command has transfer errors. The BCC is a single byte that is calculated using XOR operation (bitwise exclusive OR) on every byte in the actual command and ETX (ID is excluded!).

Some Nokeval devices have an option to disable the BCC, because it may be too difficult to calculate in some applications.

Example of a command frame

We want to send a command MEA CH 1 ? to a device with address 1. The command frame is represented below with the hexadecimal values of the bytes:

```
<ID> M  E  A     C  H     1     ? <ETX> <BCC>
 81  4D 45 41 20 43 48 20 31 20 3F  03   6F
```

BCC is calculated by applying XOR on all bytes except ID (× represents XOR operation):
4D×45×41×20×43×48×20×31×20×3F×03 = 6F

Response frame

When the device is commanded, it will send back a response frame. It can be either normal response (ACK) or error response (NAK).

Normal response

The response packet starts with response start byte ACK (06h), then comes the actual response, ETX (03h), and finally the BCC.

ACK	response	ETX	BCC

BCC is calculated by applying XOR on all response frame bytes, including the start byte ACK or NAK!

Assume the panel meter/transmitter has a result 21.3. It will answer to the MEA command with a response frame:

```
<ACK>  2 1  .  3  <ETX> <BCC>
 06   32 31 2E 33  03    1B
```

Note that some commands cause the device to send an empty response (<ACK><ETX><BCC>) to indicate that the command has been received but there is nothing to say (kind of OK response). DISP, OUT, and DO are this kind of commands.

Error response

If there is error in the transfer or in the command, the device sends an error response. It is started with NAK byte (15h) instead of ACK, and the actual response is an integer consisting of Ascii characters '0'-'9'.

NAK	error number	ETX	BCC

The precise meaning of the error number is specified in the device user manual. Most common error number meanings are:

0 Device busy – try again later
1 Buffer overflow – too long command
2 Timeout – command interrupted
3 BCC error in command frame – disturbance in transmission
4 Command not recognized
5 First parameter invalid
6 Second parameter invalid
7 etc

MOST COMMON COMMANDS

Device identification

TYPE ?

The device sends its type and software version, e.g. ”7100 V1.0” (without the quotes).

SN ?

The device sends its serial number, e.g. ”V203-12345”.

DR ?

The device sends information about its channels (“device resources”). The response is consisted of several integers. There may be any number of integers, but their meanings are in this order:

1. Number of measurement channels (MEA command)
2. Number of analog outputs (OUT command)
3. Number of digital inputs (DI command)
4. Number of digital outputs/alarms (DO command)
5. Number of externally readable keys (KEYB command)
6. Number of display characters (DISP command)
7. Reserved for future

Reading measurement value MEA

MEA CH 1 ?

The device returns the most recent result on channel 1 using Ascii characters 0-9, - (minus), . (decimal point). There may also be spaces before and after the reading. The channels are numbered from 1 on.

An example of actual reponse: ”21.3” (without quotes).

MEA SCAN 1 4

Returns results on channels 1-4 in one response frame. They are separated from each other with one or more spaces.

Example: ” 21.3 103.32 -938.89 1.2”

MEA LIST 3 1 2 7

Returns results from three channels – channels 1, 2, and 7, in one response frame. See MEA SCAN.

Controlling outputs OUT

OUT CH 1 100.0

Tell the device to set the output on channel 1 to value of 100.0. Most devices have scaling values that will select how the physical output corresponds to the engineering values. The device returns an empty response frame.

OUT SCAN 1 4 10 20 30 40

Set output on channels 1-4 to values 10, 20, 30, and 40 correspondingly.

Digital inputs and outputs DI and DO

DI CH 1 ?

Returns the state of digital/switch input 1. The response is either “0” or “1”.

DI SCAN 1 4

Returns the state of inputs 1-4 separated with spaces, for example ”1 1 0 1”.

DO CH 1 0

Sets the digital output 1 to state 0. An empty response is returned.

DO SCAN 1 4 0 0 0 0

Zeroes four first digital outputs.

Controlling the display DISP

DISP HELLO

Sends a string HELLO to the display. An empty response is returned.

MIXING SCL AND OTHER PROTOCOLS

It is not recommended to mix several protocols on one RS-485 bus. However this can be done if the protocols are examined and verified that there is no risk of misunderstandings.

Nokeval SCL frame can contain any byte 0…255, but bytes 128…255 are used as frame starters only. Note that the BCC can be any byte 0…127.

To receive a complete command frame, following conditions have to be met:

- There is a byte whose value is 128+device address
- After that, there is a byte whose value is 3 (ETX)
- After that, there is one more byte

When these conditions fulfill, the device will respond (probably with a NAK response) and mess the other traffic.

However some devices send a Timeout NAK response, and then the only requirement is that the device has seen 128+its address on the bus.

If you can ensure the other protocols do not contain bytes with value of 128+address of any Nokeval SCL device, you can be sure there is no problem on Nokeval side. However you have to ensure that the other devices do not get confused with SCL messages.

TROUBLESHOOTING

It is a common situation that a user has got new devices, connected them, and written a software of his own, and there is no any response from the devices. It is hard to say where the problem is.

To be sure the problem is not in the software, it is advisable to use software known to work. Nokeval distributes SCL test program called Sicala for free. It is easy to use: select the COM port, baud rate, and target device address. Then write a command and click Send. If there is still no response, start examining the wiring.

First, be sure there really is supply voltage present at the device. Use a multimeter to check the polarity. Then check the baud rate and address of the target device.

If using RS-232, ensure that you have connected TxD to RxD and vice versa. Measure the voltage on TxD against the RS-232 common: there should be negative voltage at least –3 V, typically –10 V. Then measure RxD, there should be negative voltage too.

Nokeval uses symbols A and B differently than most manufacturers. Consequently, if mixing Nokeval and other products on the same bus, Nokeval A connection should be routed to B connection on the other equipment and vice versa. Between Nokeval devices, A is connected to A and B to B.

In a RS-485 bus, measure the voltage between A (+) and B(-) on every device on the bus:

- More than 1 V
  - One or both of the wires to this device is broken.
  - Terminating resistors are not enabled on the bus. Switch on at the first and the last device.
  - AC termination is used, not a fault.
  - Some device has reserved the bus. Try to detach the devices one by one.
- About 0.2 V
  - Proper idle voltage, all is OK.
- 0.0...0.1 V
  - The terminating device (e.g. RS-232/485 converter) has no supply voltage.
  - There is no fail-safe resistors switched on on the bus. Switch them on in the terminating devices. Some newer Nokeval devices have a single jumper that will switch on both the terminating resistor and the fail-safe.
- About –0.2 V
  - A and B switched over.

HINTS FOR PROGRAMMER

Forming the command frame

It is easy to compose the command frame once you know how to calcualte the checksum. An example for C programming language:

```
int bcc( char *s ) {
           int b = 0;
           while( *s ) b = b ^ (*s++);
           return b;
}
```

Before sending, it is advisable to clear the input buffer.

Receiving

Receiving needs more careful design than sending: you have to decide when the response frame has been received completely, and you have to cancel if there is no response in appropriate time.

Most elegant way to implement the reception is to use a state variable that keeps account when progressing through the response frame. The algorithm represented below is quite tolerant for disturbances, e.g. hearing our own command as an echo, or receiving some disturbances before/after the response.

Define the state variable T with values:

0. Waiting for start byte
1. Reading the actual response, waiting for end byte
2. Waiting for checksum
3. Complete

Reserve a variable B for checksum calculation and E to indicate errors. Yet have a string for the response.

```
E:=0 //no errors yet
T:=0 //wait for start byte
Repeat
    If received byte A from the port:
        If (T=0 or T=1) and (A=6 or A=21): //start byte received
            T:=1 //advance to receive the response
            B:=A //start calculating BCC
            Clear response string
            If A=21: //NAK response
                E:=1 //indicate NAK was received
            Else
                E:=0 //forget the past, new frame begins
            End if
        Else if T=1:  //receiving the actual response
            B:=B XOR A     //update BCC
            If A=3:      //ETX received
                T:=2          //advance to wait the BCC
            Else:
                Append the byte A to the response string
            End if
        Else if T=2:  //receiving the BCC
            If B< >A:
                E:=2 //indicate BCC error
            End if
            T:=3 //frame complete (erroneous or not)
        End if
    If more than 2 seconds elapsed from the command:
        E:=3    //timeout error
    End if
Until T=3 or E=3
If E=0, we have a proper response.
```

This algorithm implemented on Visual Basic is in appendix A.

APPENDIX A: VB6 EXAMPLE

Here is three handy functions to use with Nokeval SCL devices.

Drag a serial port object (MSComm1 in our example) to any form (Form1 in this example).

First initiate the serial port using InitCom function. If your settings change, call InitCom again. There is no need to call CloseCom first. InitCom returns zero, if the port initiated successfully. If not, most probably there is no such port or it has been reserved by some other program.

To send a command to a device, simply call:
Dim r As String
r = SCL_command(“MEA CH 1 ?”, 0)

This function returns an empty string in case of an error. So it is not possible to see if there was a response to a command that has an empty response such as DISP. If using this kind of commands, you have to modify the end of SCL_command() to return something else or set a global variable to indicate communication error.

```
---------------------------------------------------------------------
'SCL serial functions.
'Uses Form1!MSComm1 object.
'Nokeval Oy / Juha Hämäläinen

'First initiate the port:
'   If InitCom(1, "9600") Then MsgBox "Serial port not available"
'Then send a command and wait for a response:
'   r = SCL_command( "MEA CH 1 ?", 0 )
```

Function InitCom(port As Integer, baud As String) As Integer

```
'Initiates a serial port. Port is COM port 1...
'Returns 0 if successful.

    'Prevent the program from crashing if initialisation fails
    On Error GoTo ComError

    'Close the port so that we can change the settings
    If Form1!MSComm1.PortOpen Then Form1!MSComm1.PortOpen = False

    'Change settings and open the port
    Form1!MSComm1.CommPort = port
    Form1!MSComm1.Settings = baud & ",N,8,1"
    Form1!MSComm1.InputLen = 1  'Read response byte by byte
    Form1!MSComm1.PortOpen = True

    InitCom = 0
    Exit Function

ComError:
    InitCom = 1
    Exit Function
End Function
```

Sub CloseCom()

```
'Closes (releases) the port.
    If Form1!MSComm1.PortOpen Then Form1!MSComm1.PortOpen = False
End Sub
```

Function SCL_command(cmd As String, addr As Integer) As String

```
'Sends a SCL command cmd to address addr and waits for a response.
'In case of error, returns an empty string.
'Example r = SCL_command( "MEA CH 1 ?", 0 )
'Timer function is used for timeout, but because it rolls over to 0
at midnight,
'we need to use MOD 86400.
Dim bcc As Integer      'Checksum calculation
Dim i As Integer        'General purpose counter / received byte
Dim recvstat            'Reception state
Dim t0                  'The moment (Timer) at which the command was
sent
Dim resp As String      'Response
Dim e As Integer        'Error (0 = OK)

    'Flush input buffer:
    Form1!MSComm1.InBufferCount = 0

    'Calculate command frame BCC:
    bcc = 3     'Contains ETX which is not included in the string yet
    For i = 1 To Len(cmd): bcc = bcc Xor Asc(Mid$(cmd, i, 1)): Next

    'Send the command frame:
    Form1!MSComm1.Output = Chr$(addr Or 128)&cmd&Chr$(3)&Chr$(bcc)

    'Start waiting the response:
    recvstat = 0
    t0 = Timer
    Do 'Repeat until the whole response has come
        If Form1!MSComm1.InBufferCount Then  'One byte received
            i = Asc(Form1!MSComm1.Input)  'Value of that byte
            If recvstat < 2 And (i = 6 Or i = 21) Then 'Start byte
                recvstat = 1    'Advance to receiving the response
                bcc = i
                resp = ””    ’Clear the reception string
                If i = 6 Then e = 0 Else e = 1  'ACK or NAK response?
            ElseIf recvstat = 1 Then 'Read the actual response
                bcc = bcc Xor i
                If i = 3 Then 'ETX received
                    recvstat = 2 'Advance to wait for BCC
                Else
                    resp = resp + Chr$(i)   'Collect the response
                End If
            ElseIf recvstat = 2 Then 'Receive the BCC
                If bcc <> i Then e = 2  'It was wrong!
                recvstat = 3 'However, the frame is complete, exit.
            End If
        End If
        If (Timer - t0 + 86400) Mod 86400 > 2 Then e = 3    'Timeout!
    Loop Until recvstat = 3 Or e = 3 or DoEvents() = 0

    'Set the response:
    If e = 0 Then
        SCL_command = resp
    Else
        SCL_command = "" 'Error
```

```
    End If

End Function
```