

BCL for beginners

BCL is the **BARIX Control Language**, it is a simple (like Basic) but very powerful programming language to control Barix devices like the BARIX Barionet. In the meantime there is also a BCL interpreter firmware existing for the Barix Audio products – the Annunicom, the Instreamer and the Exstreamer.

BCL gives you access to almost all interfaces of the BARIX devices. You can create your own applications which fulfill your special requirements.

For BCL development you need :

- a normal text editor, to create a **BAS** file with the BCL commands
Only in this BAS file (e.g. Barionet.bas) you have write your BCL commands !!
- the file “tokenizer.exe”, to convert the BAS file to the right format that the BARIX device can interpret the BCL commands (creates a **TOK** file)
- the file “web2cob.exe”, this adds all from your program needed files (e.g. html, tok or MP3) to a archive with a special file structure to store it into the BARIX device’s web server (creates a **COB** file)

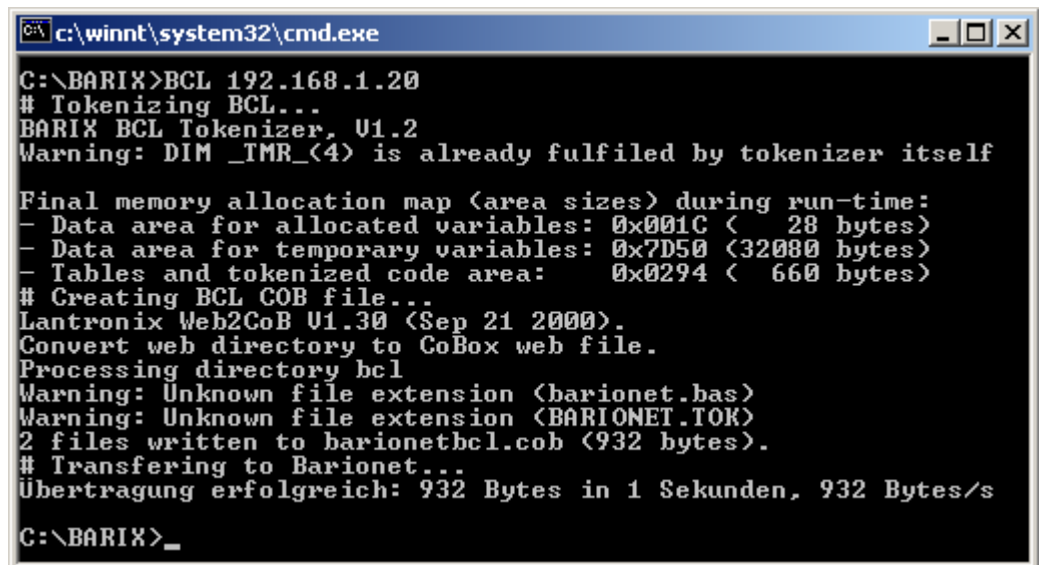
On Barix Audio devices you can use the “advanced Update” function to load the COB file to the web server. On the Barionet you have to use the TFTP command to upload the COB file .

In this beginner guide I continue with some Barionet specific examples .

In the Barionet development kit is a file “bcl.bat” contained which executes all the steps for you. There you only need to use the following command on the command prompt :

BCL <IP address of the Barionet> e.g. BCL 192.168.1.20

- command →
- automatic tokenizing of the barionet.bas →
- automatic binding the content of folder “BCL” into the COB file →
- automatic file transfer to the Barionet’s web server (to WEB4) →



```

C:\winnt\system32\cmd.exe
C:\BARIX>BCL 192.168.1.20
# Tokenizing BCL...
BARIX BCL Tokenizer, U1.2
Warning: DIM _TMR_(4) is already fulfilled by tokenizer itself

Final memory allocation map (area sizes) during run-time:
- Data area for allocated variables: 0x001C ( 28 bytes)
- Data area for temporary variables: 0x7D50 (32080 bytes)
- Tables and tokenized code area: 0x0294 ( 660 bytes)
# Creating BCL COB file...
Lantronix Web2CoB U1.30 (Sep 21 2000).
Convert web directory to CoBox web file.
Processing directory bcl
Warning: Unknown file extension (barionet.bas)
Warning: Unknown file extension (BARIONET.TOK)
2 files written to barionetbcl.cob (932 bytes).
# Transferring to Barionet...
Übertragung erfolgreich: 932 Bytes in 1 Sekunden, 932 Bytes/s

C:\BARIX>_
  
```

Everything you need (above mentioned) for the BCL development is contained in the Barionet development kit. Ok, maybe some more thing you probably need, the Barionet BCL manual and the Barionet Development Kit manual, both you can download from : www.barix.com (under DOWNLOADS – BARIONET)

BCL programing

The screenshot on the first page shows how to convert and to load the program to the Barionet. But before you have to write your program. On the following pages we make some simple programing examples. These BCL commands must be written in a BAS file (e.g. Barionet.bas) with a normal editor !

Note, if you need more details to the commands used below, then please read the BCL manual !

In the following example (content of the Barionet.bas) we can switch Relay1 to the same state as digital Input5. If Input5 is changing the state (e.g. to 1) then also Relay1 is changing his state (also to 1). Additional the program sends network messages to inform about the state changes.

(on the right side is a explanation for the commands)

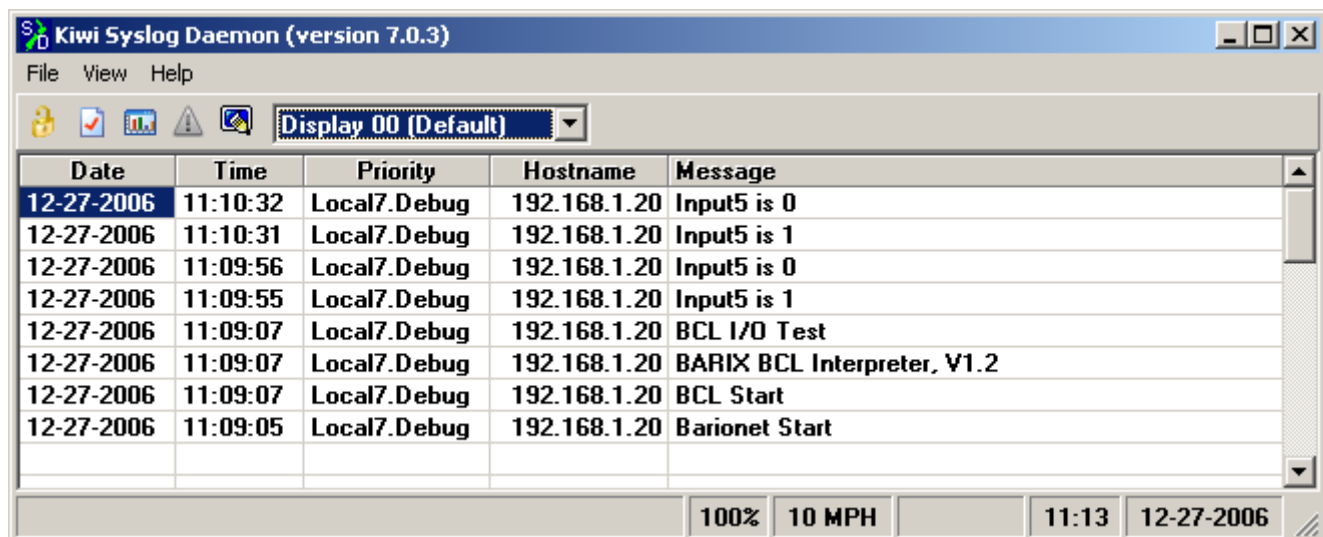
```
SYSLOG " BCL I/O Test",2          ' network message for the BCL program start ( not really needed )
DIM i5                            ' variable declaration
DIM i5old                          ' variable declaration

i5old = 0                          ' initial state assignment
1000 i5 = IOSTATE(205)             ' here begins the loop, variable i5 get the current state of Input5
IF i5<>i5old THEN                  ' comparing the old state with current state
IOCTL 1,i5                         ' if state is different then switch Relay1 to same state then Input5
SYSLOG "Input5 is " + SPRINTF$("%u",i5),1      'network message :  Input5 is 1 (or 0)
i5old = i5                         ' the current state is assigned to the old state for the next loop
ENDIF                               ' ends the "IF" command
DELAY 10                          ' important in endless loops, the little delay gives the Barionet time to
                                   ' do some other tasks

GOTO 1000                          ' goes to point 1000, back to the begin for the next round
```

The SYSLOG commands are not really needed, but they can be used to inform about any activity on the Barionet . **For bigger BCL applications it is urgently recommended to use SYSLOG for debugging purposes (failure analysis).**

There are some SYSLOG receivers on the market to receive and read the messages on your PC, the most popularest is probably the "Kiwi SYSLOG Daemon", which you can find on the internet. Following you can see a screenshot from Kiwi SYSLOG Daemon with messages from the program above:



Date	Time	Priority	Hostname	Message
12-27-2006	11:10:32	Local7.Debug	192.168.1.20	Input5 is 0
12-27-2006	11:10:31	Local7.Debug	192.168.1.20	Input5 is 1
12-27-2006	11:09:56	Local7.Debug	192.168.1.20	Input5 is 0
12-27-2006	11:09:55	Local7.Debug	192.168.1.20	Input5 is 1
12-27-2006	11:09:07	Local7.Debug	192.168.1.20	BCL I/O Test
12-27-2006	11:09:07	Local7.Debug	192.168.1.20	BARIX BCL Interpreter, V1.2
12-27-2006	11:09:07	Local7.Debug	192.168.1.20	BCL Start
12-27-2006	11:09:05	Local7.Debug	192.168.1.20	Barionet Start

As above mentioned you can also access almost all interfaces of the Barionet, also serial or network. Here is simple examples for TCP/serial communication. If one of the Inputs goes active then it will make a TCP connection and send a message (e.g. Input 3 =1). If the network connection fails then it will send the message as backup on the Barionet's serial port (RS232) .

(on the right side is a explanation for the commands)

```

SYSLOG " BCL I/O -TCP Test",2
DIM COM$(8) ' string variable declaration, max. 8 characters long
DIM TCP$(26) ' string variable declaration, max. 26 characters long
DIM ios(8) ' array variable declaration, max. 8 fields
DIM old(8) ' array variable declaration, max. 8 fields
DIM i ' variable declaration
DIM s$ ' string variable declaration

FOR i = 1 TO 8 ' FOR loop fills old array with 0 (a valid value)
old(i)=0
NEXT i

COM$ = "COM::1"
OPEN COM$ as 0 ' opens COM1 of the Barionet as handle 0

1000 ' Main program loop start
FOR i = 1 TO 8
ios(i)=IOSTATE(200 + i) ' reads IO state to ios array
IF ios(i)<>old(i) THEN ' compares ios value with old value, if different
    TCP$ = "TCP:192.168.1.10:5000"
    OPEN TCP$ as 1 ' then open TCP port as handle 1
    s$="Input"+ SPRINTF$("%u",i) =" "+ SPRINTF$("%u",ios(i))+ chr$(10)+chr$(13)
    IF NOT(Connected(1)) THEN ' if TCP connection (handle 1) fails
        SYSLOG "Connection failed",2 ' then send a SYSLOG message
        WRITE 0, s$, 0 ' and write s$ sting (Input x=0/1 ) to COM1 (handle 0)
    ELSE ' if TCP connection does not fail
        WRITE 1, s$, 0 ' then write s$ string to TCP connection (handle 1)
        CLOSE 1 ' close TCP connection
    ENDIF
ENDIF
old(i) = ios(i) ' assigns ios value to old value
DELAY 15 ' the delay gives the Barionet time to do other tasks
NEXT i ' increases variable i one step
GOTO 1000 ' goes to point 1000, back to begin for the next round

```

Following is a example for a UDP receiver.

If the Barionet receives a UDP message containing a number 1 until 6, then it will switch/pulse the according relay or digital output (1 and 2 for relay / 3 - 6 for digital output) for 10 seconds. Additional it will send a network Syslog message when it turns on a relay/output.

(on the right side is a explanation for the commands)

```
SYSLOG "BARIX BCL UDP-I/O Test"
```

```
DIM UDP$(17)           ' string variable declaration, max. 17 characters long
DIM c$                 ' string variable declaration
DIM ioa                ' variable declaration

UDP$ = "UDP:0.0.0.0:5001"
OPEN UDP$ as 1         'opens the local UDP listen port 5001

1000 DELAY 15          ' main program loop, the delay gives the Barionet time for other tasks
read 1,c$,1           ' reads the UDP socket
If lastlen(1)<1 Then GoTo 1000 ' If no data are received, then go back to start (1000)
ioa=Val(c$)           ' else convert the string to a number
  IF ioa = 0 then GOTO 1000 ' if the number = 0 then go to start (1000)
  IF ioa > 7 then GOTO 1000 ' if number is bigger than 6 go to start (1000)
  IF or(ioa = 1,ioa=2) then ' if the number is 1 or 2
    SYSLOG "Station " + SPRINTF$("%u",ioa) + " is calling",2 'Syslog message
    IOCTL ioa,100 ' then switch/pulse relay (1 or 2) for 10 seconds
    ELSE
    ioa = ioa + 98 ' if not, add 98 to value to change to output registers
    SYSLOG "Station " + SPRINTF$("%u",ioa) + " is calling",2 'Syslog message
    IOCTL ioa,100 ' switch/pulse output (1,2,3 or 4) for 10 seconds
  ENDIF
ENDIF
GOTO 1000             ' goes to point 1000, back to begin for the next round
```

Each of the examples you can copy in your Barionet.bas and it will work, they are all complete !

Here ends the first lesson.

Much more BCL programing examples you can find in the Barionet forum's files section.

The Barionet forum is also recommended for all BCL developers, there you will meet all the Barionet- and BCL- experts !

<http://tech.groups.yahoo.com/group/Barionet/>