

ABCL FIRMWARE

ABCL Firmware

Basic like programming language firmware for
Barix Audio products



Technical Documentation

Version V0.17
Released 29.09.2008



Revision History

Revision	Date	Initials	Notes
1.0	16/05/07	PK	Initial Working Draft
1.1	07/08/07	PK	Added BCL Development Kit
1.2	31/10/07	PK	BCL I/O Map
1.3	09/11/07	PK	I/O Map for IPAM
1.4	25/04/08	PK	Version A0.13: updated Setup record (syslog address)
1.5	10/07/08	PK	Version A0.15: hardware identification added to IO map
	21/07/08	PK	Corrected error in Syslog destination address (EEPROM layout), 4 bytes are used
1.6	17/09/08	PK	Added hardware capabilities overview Added PS16 Paging Station
	26/09/08	PK	Added hardware type 25 for the PS16 Paging Station

References

	Document	Date	Author
[1]	EEPROM Setup Record For “Barix ABCL Applications”	12.03.2007	P.Kulhavý
[2]	Barix Control Language (BCL v1.3) Programmers Manual 2.6	17.05.2007	P.Kulhavý, J.J.Zvánovec, J. Rietschel

© 2007 Barix AG, Zurich, Switzerland.

All rights reserved. All information is subject to change without notice.

All mentioned trademarks belong to their respective owners and are used for reference only.

Barix, Annunicom, Barionet, Exstreamer, Instreamer, SonicIP and IPzator are trademarks of Barix AG, Switzerland and are registered in certain countries.

Table of Contents

I INTRODUCTION.....	5
1.1 ABOUT THE ABCL SOFTWARE.....	5
1.2 ABOUT THIS TECHNICAL DOCUMENTATION.....	5
Links to chapters.....	5
Bookmarks pane in Adobe Acrobat.....	5
Chapter overview.....	5
2 BCL DEVELOPMENT KIT.....	6
2.1 REQUIREMENTS.....	6
2.2 ABCL ARCHITECTURE.....	6
2.3 SOURCE FILES.....	6
Sample applications.....	6
Custom applications.....	7
File Naming Conventions.....	7
2.4 COMPILATION TOOLS.....	8
Building a custom application.....	8
3 APPLICATION REMOTE CONTROL INTERFACE.....	9
3.1 CONTROL INTERFACE DESCRIPTION.....	9
3.2 LIST OF COMMANDS.....	9
3.3 PRINCIPLES OF THE CGI WEB INTERFACE.....	10
4 WEB USER INTERFACE.....	11
4.1 USER INTERFACE DEVELOPMENT KIT	11
Web2cob tool.....	11
Original UI Files.....	12
4.2 DYNAMIC WEB PAGES.....	13
Initial Dynamic Mark.....	13
Syntax of Dynamic Marks.....	14
List of Dynamic Mark IDs for &LSetup.....	14
Dynamic Marks For Group &LState:.....	16
State Variables:.....	16
4.4 CONFIGURATION VIA HTML PAGES.....	17
Examples.....	17
Form element names.....	19
4.5 WEB UI CONFIGURATION LOGOUT.....	20
5 MEMORY ORGANISATION.....	21
5.1 SERIAL RESCUE KIT / WEB UPDATE.....	21
5.2 FLASH MEMORY USAGE.....	21

Flash memory usage table (Note: this may be subject to change if the applications.cob requires more than 3 pages).....	21
5.3 CONFIGURATION STORAGE (EEPROM).....	22
EEPROM Layout.....	22
Default settings.....	22
Factory defaults using the Serial Rescue Kit.....	22
Factory defaults using the reset button.....	22
Configuration storage usage.....	23
General Terms (EEPROM Organization).....	23
List of Configuration parameters.....	23
6 AUTONOMOUS OPERATION.....	26
6.1 THE RESET BUTTON.....	26
6.2 DEVICE AVAILABILITY WITH THE GREEN AND RED STATUS LEDs.....	26
No Application loaded (only bootloader) or started with hold reset button during power up.....	26
Application starts (Barix boot-up sequence):.....	26
Application running.....	26
6.3 YELLOW AND GREEN NETWORK INTERFACE LEDs.....	27
7 FIRMWARE AND STANDARD FILE UPLOAD.....	28
7.1 ADVANCED UPDATE.....	29
APPENDIX A: BCL IO MAP.....	32
Appendix A: BCL I/O Address Map.....	32
Hardware identification.....	32
Device overview.....	32
I/O registers.....	33
1...100 Relay outputs (Read, Write).....	33
201...300 Digital inputs (Read only).....	37
401...500 Reserved.....	40
501...600 Analog inputs (Read only).....	40
601...700 Reserved.....	40
Instreamer Legacy/100, Exstreamer Legacy/100/200.....	42
Exstreamer 110.....	42
Annuncicom Legacy/100/200.....	42
IPAM/IPAM Carrier Board.....	43
Exstreamer 1000.....	44
Annuncicom 1000.....	45
PS16 Paging Station.....	46

I Introduction

This Technical Documentation describes the Non Volatile Databases, External Interfaces, Development Kit and Autonomous Operation of the ABCL .

I.1 About the ABCL Software

The ABCL Software is a user programmable audio platform supporting a variety of functions and interfaces like filesystem, network access, serial interface, audio input/output or direct IO control. The platform is programmable in the Barix Control Language (BCL, see the language reference [2]) and runs on the standard Barix audio hardware (Instreamer, Exstreamer, Annunicom series and the Barix IPAM).

The goal of the ABCL is to allow system integrators, distributors and users to develop custom specific applications running on Barix hardware.

I.2 About this Technical Documentation

Links to chapters

References to chapters (e.g. [X Chapter name](#)) are red and underlined and serve as direct links when viewed in Adobe Acrobat Viewer. Click on the link to jump to the referenced chapter, click on the left arrow icon to jump back to where you came from.

Bookmarks pane in Adobe Acrobat

The complete “Table of Contents” is available in Adobe Acrobat Viewer. Click on the “Bookmarks” pane tab on the left side of Adobe Acrobat Viewer to open it. Click on any bookmark to directly jump to the corresponding part of the manual.

Chapter overview

This technical documentation is divided into the following chapters:

- [2 BCL Development Kit](#) (describing the demo programs and explaining how to use the provided tools to compile custom applications)
- [3 Application Remote Control Interface](#) (explaining how to control the device using the command suite)
- [4 WEB User interface](#) (explaining the User Interface functionality and how to customize it)
- [5 Memory Organisation](#) (explaining the use of the Flash memory and the EEPROM configuration memory)
- [6 Autonomous Operation](#) (explaining the Reset Button functionality and the status information provided by the LEDs)
- [7 Firmware and standard File Upload](#) (explaining the standard Barix method for updating the units Software)

2 BCL Development Kit

2.1 Requirements

The ABCL rescuekit is distributed with necessary tools and scripts for compilation running on both Linux and Windows platforms.

However on Linux operating systems other external programs are required:

- the DOS emulator “dosemu” (see <http://www.dosemu.org/>)
- the Windows emulator “wine” (see <http://www.wine.org/>)
- the Advanced TFTP client “atftp” (see <http://freshmeat.net/projects/atftp/>)

All the mentioned programs are expected to be in the system path. If they are not, modify the scripts accordingly.

2.2 ABCL Architecture

The ABCL platform consists of the BCL interpreter (firmware and extensions), the WEB UI and the BCL programs with sources.

Before loading into the device the BCL sources are converted into a binary form (tokenized) and packed (usually with sources) into a `.cob` file.

See [2] for more details about tokenization. Programs are identified by the filename in the `.cob` file.

Typically more programs are loaded into the device, but only one can be interpreted at a time.

The 15-character name of the BCL program to be interpreted is stored in the Setup memory at position `S180` (see section 5.3 for more details) and can be selected in the “Application” field in the configuration WEB page of the device.

2.3 Source files

Sample applications

The ABCL platform is delivered with a set of sample BCL programs (applications) including their sources, the content of the set may vary between different versions of the rescuekit (more applications may be added).

The compiled applications are stored in `applications.cob` file located in the “update_rescue” directory and loaded into the FLASH memory of the device starting from page `WEB10`, see section 5.2 for more details.

To recreate `applications.cob` call the `applctns.bat` script; on Linux call `applications.sh`

Sample list of applications (version 1.05 of the WEB application) :

annunfdx	Full-duplex Annuncicom
stacker	Audio Stacker
ftpusb	FTP-USB recorder
iclient	Intercom Client
imaster	Intercom Master Station
sip	SIP Client
ftp_mp3	FTP/HTTP store and forward MP3 player
custom1	Custom Application 1
custom2	Custom Application 2

Sources of the sample programs are available in the “bcldevkit” directory in subdirectories of the respective name.

Custom applications

There're two placeholders for user-modifiable applications called **custom1** and **custom2**. Currently only **custom1** is present and depending on the rescue-kit is either loaded with a dummy program or an OEM application. The custom application (**custom1.cob**) is loaded into the FLASH memory of the device starting from page **WEB5** (see the memory organization in chapter 5.2).

The sample programs can be modified by the user and using the provided tools compiled and loaded into the device (see 2.4). If a program with the same name already exists among the sample programs, the user program (which is at lower address in memory) takes precedence. After deleting or overwriting the user area with a program of a different name the original sample application is available again.

File Naming Conventions

Since the files in the FLASH memory of the device are not structured into directories (even though they might be in different **.cob** files), there are standard file naming conventions to prevent mixing of files between different applications. The “<basename>” in the following text refers to the application name as listed in the above table. The basic rule (with the exception of “readme” files) is that names of all files belonging to the same application start with the basename.

File	Name
program source	<basename>.bas
tokenized program	<basename>.tok
WEB UI frameset	<basename>.html
UI configuration	<basename>_config.html
UI help	<basename>_help.html
program description	README_<basename>

2.4 Compilation Tools

To compile and load a program into the FLASH memory of a device use the provided `bcl.bat` (`bcl.sh` on Linux) script in the “bcldevkit” directory. It calls the `tokenizer.exe`, creates a `.cob` file calling the `web2cob.exe` and optionally loads the `.cob` file into the device using the TFTP protocol. Syntax of the program is the following:

```
bcl.bat <basename> [ <address of the unit> ]
```

On Linux:

```
bcl.sh <basename> [ <address of the unit> ]
```

The address is either the IP address or the DNS address. If it is omitted, only the `.cob` file is created.

Building a custom application

To build a custom program and load into the device, do the following:

1. Create a subdirectory of the “bcldevkit” directory with the basename of your program
2. Create the source `.bas` file, UI files and optionally other files belonging to the application using the naming convention mentioned above
3. Switch the device into the update mode (either over the WEB interface or using the reset button)
4. Run the `bcl.bat` script (`bcl.sh` on Linux) with the basename and address of the unit as described above
5. Reset the device

The program will be loaded into the FLASH memory starting from page **WEB5**. **Please note that the above procedure will overwrite the original memory content!** Therefore only ONE custom application can be loaded into memory at a time with this procedure.

Alternatively you can call `tokenizer.exe` and `web2cob.exe` manually to tokenize the source and create the `.cob` file, see [2] for further details.

If you're creating a new application don't forget to add a new entry to the list of applications in `uisettings.html` in the WEB UI as well(see 4.1).

3 Application Remote Control Interface

A command suite, based on AnnunicomIC technology, is provided to manage the V0.17 via the WEB HTTP Interface.

For remote control via HTTP, the CGI WEB script rc.cgi is used. For example <http://x.x.x.x/rc.cgi?c=99> (DEVICERESET- Hard reboot of device.)

3.1 Control Interface Description

- 0xnn means a hexadecimal number.
- means 0x0D 0x0A 0x00 on answers. On requests ↵ could be one or more of the following codes/bytes: 0x0D, 0x0A, 0x00.
- The answers are only echoed to the origin source of the command (not to the other interfaces).
- An answer can be selected by concatenating the L command. If no special answer is requested empty page is returned.
- The answer files can be edited and changed to your needs (see 4).
- The standard answers are designed as XML.
- All strings and everything else are case sensitive.
- All commands are asynchronous to the stream.
- One command mustn't exceed 1024 bytes even it is concatenated.
- To concatenate control commands use & (Ampersand, ASCII:38).
- If password is set, it must be sent either with "a=" concatenated with the command, e.g. <http://x.x.x.x/rc.cgi?c=99&a=password> or within the HTTP request header ("WWW Authentication Basic").

3.2 List of commands

Element	Description	command
<i>Reserved</i>		c=0 to c=98
DEVICERESET	Hard reboot of device.	c=99
BOOTLOADER	Starts the bootloader. The application will be left. It isn't running until the next reboot.	c=100
PASSWORD	If password is set and not send within the HTTP request header, a= must be used in concatenation with the command to execute, see 3.1	a=...
GETDYNFILE	The response is the dynamic file stored in a cob file (see 4.1) with that name. Example: L=index.html	L=...

3.3 Principles of the CGI WEB interface

- The browser should support frames.
- GET method should be used in forms.
- Respect the common character set for URLs.
- Example of CGI WEB commands: <http://x.x.x.x/rc.cgi?c=99> (command for RESET on ABCL unit with IP address x.x.x.x)
- If "L=" is not used, blank page is sent as an answer.
- All strings and everything else are case sensitive.
- A CGI request should not exceed 1024 bytes.
- To concatenate control commands use & (Ampersand, ASCII:38).
- If a password is set, it must be sent either with "a=" concatenated with the command, e.g. <http://x.x.x.x/rc.cgi?c=99&a=password> or within the HTTP request header ("WWW Authentication Basic")

4 WEB User interface

4.1 User Interface Development Kit

With the “User Interface Development Kit” you can design your own web pages (skin) and modify the answers to your needs.

The Development Kit holds the original HTML files you need for the web pages, the answer text files, lookup files (ini), graphics and sounds as well as the default configuration file `config.bin`. You can simply edit these files and/or add new ones.

Note: Filenames must not start with `rc.cgi`, `basic.cgi`, `BAS.cgi` or `setup.cgi`.

Web2cob tool

To generate the application cob file start the batch `abclapp.bat` which uses the packaging tool `web2cob.exe`.

For the upload of the `.cob` file to the device, go to the configuration page of the device and click on the button “Update”.

After the device has rebooted and the update page appears, click on “Advanced Update”.

Enter the correct Target (check the flash memory usage table in 5.2) in upper case letters.

Select the cob file you want to upload and hit the “OK” button.

Click on the “Upload” button.

Rules:

- If you upload a `.cob` file to pages already used, the current contents will be overwritten
- The web server in the device sees all the targets (`.cob` files) as one directory
- If two files in different `.cob` files have the same name then the one from the lower page is chosen.

After the upload reboot the device and reload the modified page in the browser to see the changes.

Depending on the browser's cache strategy, sometimes it's needed to close and reopen the browser to see the changes.

Original UI Files

The web interface (and the firmware) need at least the following files (more example files might be included):

Type	Filename.extension	Description
Binary	config.bin	System area factory default settings. The file is binary and it is an exact mirror of the system area settings for the EEPROM. See 5.3 for further details.
HTML	index.html	main page of the web server, included the five frames: info, menu, settings, empty. empty is a hidden frame that receives the answer of the CGI commands.
HTML	rebooting.html	displayed after the user settings are changed, redirects to the main page
HTML	status	shows the actual states of the device
HTML	uifapplication.html	frame page for BCL application configuration, links to the currently selected program in the setup (see 2.2) the actual configuration pages are stored in applications.cob file (see 2.3)
HTML	uifloader.html uifsettings.html uifreboot.html uifupdate.html	frameset for the corresponding pages
HTML	uifmenu.html	shows the Barix logo and the system version info
HTML	uihloader.html uihreboot.html uihsettings.html uihupdate.html	help for the corresponding pages
HTML	uilogout.html	logout page
HTML	uimenuline.html	menu buttons for configuration, reboot and update
HTML	uireboot.html	reboot the device
HTML	uirloader.html	showed when the device is rebooting into the bootloader mode
HTML	uirrebootI.html	showed after the device is rebooted and has then successfully rebooted
HTML	uirreboot.html	Shown when the device is rebooting ("Reboot" button was pressed and confirmed)
HTML	uirupdate.html	forwarding page to hide the command for the update
HTML	uiupdate.html	update the device, appears after clicking "Update" in the menu
HTML	update.html	frameset for uirupdate.html

Type	Filename.extension	Description
HTML	uisettings.html	main configuration page, contains the system settings
Image	4to0.gif	needed for apply the configuration for waiting for the reboot of the device
Image	barix.gif	used in uicfg.html
Image	menu.gif	picture for the menu buttons in the configuration, used in uicfg.html
Java Script	util.js	javascript functions for the HTML configuration pages (range checks)
text files	ABCLAPPVERSION	for the version number and the history
text files	ERRORS.HLP	table of textual error messages for the BCL interpreter
text files	SONICIPVERSION	for the version number of SonicIP implementation
Sound	baring.wav	Barix ringtone, used by BCL applications
Sound	n.mp3, dot.mp3	spoken "n" (where n:= 0-9) , spoken "dot"

4.2 Dynamic Web Pages

Web pages can include dynamic values. Dynamic Web Pages are built in HTML or XML or in an other text file format that exclude the binary character 0x00, i.e. the dynamic page can be an HTML file. It's possible to use scripts or everything else allowed in the given document's file format.

Initial Dynamic Mark

In order to indicate that Web page is dynamic, it has to contain the special initial dynamic mark `&L(0,"*")` ; in the first 500 Bytes and before any other dynamic value is used. The initial mark can also have decimal number as its optional third parameter. Example of such initial mark is `&L(0,"*",1)` ;.

The third parameter is parsed bitwise and has the following meaning:

- If bit 7 is set then the code page IBM437 will be used instead of the standard HTML code page.
- If bit 4 is set the access will be exclusive (only one user at a time, tested by its IP address). The user has to logout or the software does an automatic logoff 20 min after the last access to such a page. Only one password level can have the exclusive feature (doesn't matter which one).
- Bits 1-3 are used as password level (1-6) for the file corresponding to the password level parameters in the configuration.
Example for level 5: `(&L(0,"*",10);)`.
- If bit 0 is set, then the content length will not be included in the HTTP header. Page is sent faster by saving the time needed to calculate the content length.

Syntax of Dynamic Marks

Dynamic marks can be used to put dynamic values in Web pages. All dynamic marks have the following syntax: **&L<name>(<id>,<format>[,par]) ;**

A dynamic mark always starts with &L and it is always case sensitive.

- **<name>** selects a group of dynamic values. Defined is the "Setup" group for all configuration parameters, the "State" group for actual parameter states and the "BAS" group to interact with the BCL program (see [2]). Remaining parameters are included in parentheses, with the right parenthesis followed by a semicolon.
- **<id>** determines the desired function.
- **<format>** is a C-style format string (refer to the ANSI documentation).
- **<par>** are optional additional parameters. If additional parameters are needed, it is mentioned in the function lists below.

Note: The string ") ;" is not allowed inside a dynamic mark.

To have this construct inside the format string, use ") \ ;" (in an unknown escape sequence, only the '\' will be removed).

To have a "%" sign (percent sign) inside the format string, use "%%" (two signs without space).

The whole mark is replaced by the dynamic value formatted with the **<format>** string. Only one value is allowed per dynamic mark. The length of the dynamic mark mustn't exceed 500 characters. The resulting string from the dynamic mark must not exceed 500 characters.

A dynamic mark can be contained in an another dynamic mark. Only one recursion step is allowed and correct "escaping" has to be applied. Example:

```
&LSetup(3,"%s",419,B,!0,"<meta http-equiv=refresh content=\"&LSetup(1,\"%u\",419)\ ; url=info.html\">");
```

Note the special "\" before the semicolon of the dynamic mark inside. This is because the escape sequence is interpreted as only a semicolon and is needed in order to include the prohibited sequence ") ;" inside a dynamic mark.

List of Dynamic Mark IDs for &LSetup

ID	Type	Description
1	Function	Print setup value 3. [par]: Address (decimal) of the value in the setup 4. [par]: Type of the value (B for unsigned byte, W for word, D for double word, c for char/signed byte, b for bit numbered from 0 to 7, e.g. b3 for the fourth bit). If this parameter isn't available the type will be B . e.g. &LSetup(1, "%08lx", 315, D) ; as hexadecimal value with 8 characters and leading zeros e.g. &LSetup(1, "%1u", 311, D) ; as unsigned long decimal value
2	Function	Print Netmask Byte 3. [par]: Address (decimal) of the value in the setup 4. [par]: Byte number of the Netmask IP address byte starting with 0 for the first left byte and incremented by one for the next bytes

ID	Type	Description
3	Function	Print string if equal 3. [par]: Address (decimal) of the value in the setup 4. [par]: Type (see id 1 above) 5. [par]: value to compare. The prefixes !, > or < are allowed to change the comparison (no spaces between) 6. [par]: string for output if value at address is equal to 5. [par]
4	Function	Print string 3. [par]: Address (decimal) of the value in the setup
5	Byte (integer)	Firmware Version Major
6	Byte (integer)	Firmware Version Minor
7	Byte (integer)	Bootloader Version Major
8	Byte (integer)	Bootloader Version Minor
9	Function	Prints the version out of a standard version file in a *.cob application 3. [par]: name of the version file 4. [par]: 1 for major version number (byte), 0 for minor version number (byte)
10	Byte (integer)	year of the firmware build (only decade)
11	Byte (integer)	month of the firmware build
12	Byte (integer)	day of the firmware build
13	Byte (integer)	sg.bin (Audio and Utility library) Version Major
14	Byte (integer)	sg.bin (Audio and Utility library) Version Minor
15	Byte (integer)	Filesystem Version Major
16	Byte (integer)	Filesystem Version Minor
17	Byte (integer)	sg.bin (Audio and Utility library) date string
18	Byte (integer)	reserved
19	Byte (integer)	reserved
20	Byte (integer)	Filesystem build date - year only decade
21	Byte (integer)	Filesystem build date - month
22	Byte (integer)	Filesystem build date - day

Dynamic Marks For Group &LState:

ID	Type	Description
1	Function	Print state variable 3. [par]: state variable index, see the table below eg. <code>&LState(1, "%s", 12)</code> ; prints out device's MAC address
2	Function	Print string if condition is true 3. [par]: Index of the state variable to be compared, see the table below 4. [par]: value to compare. Variable is compared with the value "if equals", the prefixes !, > or < can be used to change the comparison (no spaces between allowed). When comparing variable with a string, the string has to be quoted (e.g. "string") 5. [par]: string to output output if condition is true. The string has to be quoted.

State Variables:

ID	Type	Description
0	Bool (Integer)	Filesystem present (1 means present)
1	Integer	Filesystem type: 0: unknown 1: FAT12 2: FAT16 4: VFAT 8: FAT32
2	Integer	Filesystem serial number
3-11		reserved
12	String	MAC address of the unit
13	String	Current IP address
14	Integer	USB device vendor ID
15	Integer	USB device product ID
16	Integer	USB device class
17	Integer	USB device subclass
18	Integer	USB interface class

ID	Type	Description
19	Integer	USB interface subclass
20	Integer	USB device's max. power consumption in mA
21	Bool (Integer)	USB device attached (1 means attached)
22	Integer	USB device capacity in kB
23-38		reserved
39	Integer	System uptime in milliseconds
40	Integer	System uptime in seconds

4.3

4.4 Configuration via HTML Pages

The HTML pages for the device configuration use the functionality for dynamic web pages (see 4.2). All of the configuration parameters are placed in HTML forms and are transferred by the method GET. Some of the values are checked by java script to prevent wrong values. Not all of the configuration parameters have to be present in a form. It is possible to have only a part of the configuration on a web page. The form has to start with the following two tags:

```
<form method=GET action=setup.cgi target="answer"><input type="hidden" type="text" name=L value=rebooting.html>
```

The target of the form could be changed.

The answer after transmitting the form will be the HTML page `rebooting.html`. For another HTML page change this value. If this value isn't available only the HTTP status 200 OK will be sent back.

Examples

The following example shows how to implement a form field for the configuration value of the highest byte in the 'own IP address'.

The input element name is a defined string, which has to be handled with care. The type character **B** stands for an unsigned value. 0 is the address of the expected configuration parameter. The value is a dynamic mark. The string `onChange=IPCheck(this)` will call the Javascript `util.js` to check if the value entered is in the range of 0 to 255.

```
<input name=B0 size=3 maxLength=3 value=&LSetup(1,"%u",0) ; onChange=IPCheck(this)>
```

In the next example the name selects the configuration parameter "DHCP Host Name".

```
<input name=S98 size=15 maxLength=15 value="&LSetup(4,"%s",98) ;">
```

This example shows how to implement a form field for the configuration of the Netmask. The names for the bytes of the Netmask are **N8B0**, **N8B1**, **N8B2** and **N8B3**. 8 is the address of the Netmask in the configuration memory. The value after the **B** is the byte number of the byte in the Netmask starting with 0 for the first byte at the left. This special handling for Netmask is needed because the Netmask is stored in one byte and not like the IP address in 4 bytes. The string `onChange=netMaskCheck(this)` will call the javascript `util.js` to check if the value entered is in the correct range.

```
<input name=N8B0 size=3 maxlength=3 value=&LSetup(2,"%u",8,0); onChange=netMaskCheck(this)>
```

The next example shows how to implement a form field for the configuration of the parameter 'Flow control' as a selection. If the value of the configuration parameter is equal to the second last parameter in the dynamic mark it will be replaced by the last parameter of the dynamic mark.

```
<select size=1 name=B82>
  <option value=0 &LSetup(3,"%s",82,B,0,"selected");>none</option>
  <option value=1 &LSetup(3,"%s",82,B,1,"selected");>Software (XON/XOFF)</option>
  <option value=2 &LSetup(3,"%s",82,B,2,"selected");>Hardware (RTS/CTS)</option>
</select>
```

This example shows how to implement radio buttons for the configuration parameter 'Sonic IP'. The functions of the dynamic marks are equal to the example above.

```
<input type=radio name=B277b7 value=0&LSetup(3,"%s",277,b7,0," checked");>Yes
<input type=radio name=B277b7 value=1&LSetup(3,"%s",277,b7,1," checked");>No
```

To transmit the new configuration data to the device the submit input type of the form is used.

```
<input type=submit value=" Apply ">
```

By pressing the Apply button the new configuration data will be transferred to the device. It will store the new data to its configuration memory (EEPROM). After this it sends the answer (see above) to the browser and reboots itself to apply the new configuration.

Passwords are stored in memory in hashed format (MD5) and set using the name **Px**, where **x** stands for the password level.

If the password is set already, the old password must also be supplied (with the name **Px**) together with the new password using the name **Px.1** (P level dot one).

```
<tr>
  &LSetup(3,"%s",130,D,0,"
  <td><b><font size=2>Set Password</font></b></td>
  <td><input name=P1 size=18 maxlength=25 type=password value=></td>
  ">
  &LSetup(3,"%s",130,D,!0,"
  <td><b><font size=2>Old Password</font></b></td>
  <td><input name=P1 size=18 maxlength=25 type=password value=></td>
</tr>
```

```
<tr>
    <td><b><font size=2>New Password</font></b></td>
    <td><input name=P1.1 size=18 maxlength=25 type=password value=></td>
</tr>
");
```

Px and **Px.1** can also be used for remote configuration.

Form element names

- If the value is an integer (1 byte) the first character is a **B**.
- If the value is an IP address the first character is an **I**, the complete IP address can be set as a string at once e.g.:
I0=192.168.1.2 (same as **B0=192 B1=168 B2=1 B3=2**) for IP address
I4=192.168.1.1 (same as **B4=192 B5=168 B6=1 B7=1**) for Gateway IP address
- If the value is a Netmask the first character is an **N**, e.g.:
N8=255.255.255.0 (same as **N8B0=255 N8B1=255 N8B2=255 N8B3=0**)
- If the value is a string the first character is an **S**.
- If the value is a word (2 bytes) the character is a **W**.
- If the value is a double word (4 bytes) the first character is a **D**.
- To set a password, use **P** as described above

The following decimal value in the name is the address of the configuration parameter (see chapter 5.3 Configuration storage (EEPROM)).

To set a bit in a configuration parameter (e.g. Media Configuration) add the character **b** followed by the number of the bit (starting at 0), e.g.: **b7** for the 8. bit in the byte.

Examples of names:

- **B0** first (left) byte of the configuration parameter 'own IP address'
- **B1** second byte of the configuration parameter 'own IP address'
- **N8B0** first (left) byte of the Netmask
- **N8B1** name of the second byte of the Netmask
- **N8** Netmask
- **S98** DHCP Host Name
- **B198b7** Sonic IP

4.5 WEB UI Configuration Logout

The logout is placed in an HTML form and is transferred by the method GET. The form has to contain an element named **L** with the value for the answer page and a second element with the name **D**. This element is the indication for the logout.

```
<form action=setup.cgi method=get target=_top>  
  <input type=hidden name=L value=logout.html><input type=hidden name=D><input type=submit value=" Logout ">  
</form>
```

The target of the form could be changed. The answer after transmitting the form will be the HTML page `logout.html`. For another HTML page change this value. If this value isn't available only the HTTP status 200 OK will be sent back.

5 Memory Organisation

5.1 Serial Rescue Kit / Web Update

Two different procedures exist to upload the “V0.17” firmware into the device:

The “Serial Rescue Kit” using the serial cable will upload the firmware files, the boot loader and the “factory defaults configuration” which will erase the current configuration.

The “Web update” using a browser will upload the firmware files and the “factory defaults configuration” but will not alter the current configuration.

For factory defaults and memory usage details see the following two sections.

5.2 Flash Memory usage

The “V0.17” firmware is using the built-in Flash memory as described in the table below.

Flash memory usage table (*Note: this may be subject to change if the applications.cob requires more than 3 pages*)

Page / Target	Content	Address for Rescuekit
8K (WEB0)	abclw.rom (Firmware)	0xF00000
WEB1	fs.bin (USB Filesystem FW Extension)	0xF10000
WEB2	sg.bin (FW Extension – Util library)	0xF20000
WEB3	abclapp.cob (Web Application and Sonic IP Resources)	0xF30000
WEB4	abclapp.cob continued (Web Application and Sonic IP Resources)	0xF40000
WEB5	customI.cob (Custom Application Program)	0xF50000
WEB6-8	Reserved for continuation of customI.cob and custom files	0xF60000-0xF80000
WEB9	bclio.bin (FW Extension – IO Driver)	0xF90000
WEB10	applications.cob (BCL Application Programs)	0xFA0000
WEB11	applications.cob continued	0xFB0000
WEB12	applications.cob continued	0xFC0000
WEB13	free	0xFD0000
WEB14	free	0xFE0000
No symbol	Boot Loader	0xFF0000

A page uses 64 kilobytes of flash memory. (Note: 0xC00000 = 0xD00000 = 0xE00000 = 0xF00000)

Both update procedures (Web update & Serial Rescue Kit) respect the above memory usage.

The boot loader is always on the last page at absolute address 0xFF0000 (offset 0xFF00). The rest of the software is loaded at locations compatible with the symbolic locations: 8K, WEB1, WEB2 etc. 8K is always the first page (page 0), WEB1 page 1, etc. The target has to be in capital letters (i.e. WEB4).

5.3 Configuration storage (EEPROM)

The current configuration is stored in a non-volatile memory (EEPROM). To change the current configuration use the web user interface and hit the “Apply” button to store it into the EEPROM.

EEPROM Layout

The EEPROM is logically split into two sequential areas, the system area (0-499) and the application area (500-1499). The first part contains the minimal set of parameters required by the BCL interpreter. The application area is used by BCL applications and does not have a fixed layout, nevertheless there is a recommended layout [1] for the demo applications delivered in the BCL rescue-kit.

There is no physical separation of the mentioned areas and the BCL programs can access the whole setup area of the EEPROM.

Default settings

The system area defaults are stored in the `config.bin` file in the folder “webuidevkit/abclapp”, whereas the application area defaults are stored in the `appconfig.bin` file in the folder “bcldevkit”. The complete setup image is a concatenation of these two files and is stored in binary file `config.bin` which is stored in the folder “update_rescue”. The concatenation can be done with any suitable PC tool, e.g. the UNIX “cat” or Windows “copy” command.

Factory defaults using the Serial Rescue Kit

The EEPROM is overwritten by the “factory defaults configuration” when applying the “Serial Rescue Kit” using the binary file `config.bin` (complete setup image) which is stored in the folder “update_rescue”. This file can be edited with a hex editor. Consult the “configuration memory usage” table carefully before you make any changes.

Factory defaults using the reset button

The “system defaults configuration” binary file `config.bin` (contained in `abclapp.cob`) and the “application defaults configuration” binary file `appconfig.bin` (contained in `applications.cob`) are loaded into the flash memory (not the EEPROM!) when performing the firmware update.

To apply the “factory defaults configuration” (both the system and the application factory defaults) the reset button has to be pushed for about 10 seconds, see chapter 6.1 for the usage of the reset button.

The file `appconfig.bin` as well as the `config.bin` can be edited with a hex editor. Consult [1] and the “configuration memory usage” table carefully before you make any changes.

Before uploading, folders containing the binary files have to be packed into the `abclapp.cob` and `applications.cob` files using the provided scripts. For more details see chapters 4.1 and 2.3.

Configuration storage usage

The following table shows where the configuration is stored in the EEPROM. The column “Byte” shows the offset as a decimal number. The column “Len” shows the length in Bytes. The column “Default” shows the default value as stored in the original “factory defaults configuration”.

General Terms (EEPROM Organization)

- IP addresses are always stored with the highest byte at the lowest address.
- Strings are coded in ASCII and terminated with 0x00. The Length includes the termination.
- Values are stored in little endian format (Intel) (low byte first)
- All Values are integer.
- Signed values are stored in 2-complement.
- Unused bytes must be set to 0x00.

List of Configuration parameters

Parameter	Byte	Dynamic Name	Len	Default	Short Description
Own IP	0	B0,B1, B2,B3	4	0.0.0.0	Static IP address of the device. 0.0.0.0 for automatic assignment 0.0.1.0 to disable AutoIP 0.0.2.0 to disable BOOTP 0.0.4.0 to disable DHCP 0.0.8.0 to disable IPzator add these special IP addresses to disable multiple protocols
Gateway IP	4	B4, B5, B6, B7	4	0.0.0.0	Gateway IP address. 0.0.0.0 for no gateway
Netmask	8	N8B0, N8B1, N8B2, N8B3	1	0	Subnetmask. The value is the count of the zero bits counted from the lowest byte. (ex. 8 for 255.255.255.0)
DNS 1	64	B64,B65, B66, B67	4	0.0.0.0	Primary DNS IP address. Set to 0.0.0.0 to get primary DNS from DHCP, if DHCP is configured, or to disable DNS, if DHCP is not configured.
DNS 2	68	B68, B69, B70, B71	4	0.0.0.0	Alternative DNS IP address. 0.0.0.0 here always disables secondary DNS

Parameter	Byte	Dynamic Name	Len	Default	Short Description								
IFMODE0	80	B80b0-1, B80b2-3, B80b4-5, B80b6-7 or B80	1	0x4C	Mode parameters for serial port 0. Bit definitions:								
					Function	7	6	5	4	3	2	1	0
					RS232-C							0	0
					7 Bit					1	0		
					8 Bit					1	1		
					no parity			0	0				
					even parity			1	1				
					odd parity			0	1				
					1 Stopbit	0	1						
					2 Stopbit	1	1						
BAUDRATE0	81	B81	1	2	Baudrate for the serial port 0. (7 = 300, 6 = 600, 5 = 1200, 4 = 2400, 3 = 4800, 2 = 9600, 1 = 19200, 0 = 38400, 9 = 57600, 11=76800, 8 = 115200, 10=230400)								
FLOWCONTROLO	82	B82	1	0	Flow control for the serial port 0. (0 = no, 1= Software XON/XOFF, 2 = Hardware RTS/CTS, 8=RS485 Direction Control)								
Reserved	86	V86	2										
Reserved	88	B88, B89, B90, B91	4										
Reserved	92	V92	2										
Security Settings	97	B97	1	0	bit0: 0 - reset function enabled, 1- reset function disabled bit1: 0 - factory defaults enabled, 1 - factory defaults disabled bit2:0 – web update enabled, 1 – web update disabled								
DHCP Host Name	98	S98	16		Name of the device sent in DHCP request. If not set, automatically generated name based on device's MAC address is sent. The string includes terminating zero.								
Version Major	116	B116	1	1	Version Major value (do not change)								
Version Minor	117	B117	1	4	Version Minor value (do not change)								
Setupex Length	120	V120	2	894	Length of the extended setup (always 894)								
Password Level 1	122	S122	8		Password stored as a MD5 hash (first 8 bytes) used for viewing and changing the configuration, all 0 means no password								
Passwords Level 2-6 Reserved	130-179												
Application Name	180	S180	16	annunfdx	BCL application name (without the extension) including the terminating NULL character.								

Parameter	Byte	Dynamic Name	Len	Default	Short Description
Web Server Port	196	W196	2	0	Port on which built-in webserver is running. Range: 1...65535
Media Configuration	198	B198b0, B198b1, B198b2, B198b3, B198b4, B198b5, B198b6, B198b7	1	0x00	Function is activated by setting the appropriate bit: 0x01: not used 0x02: not used 0x04: not used 0x08: not used 0x10: not used 0x20: not used 0x40: not used 0x80: 0 – SonicIP on, 1 – SonicIP off
Reserved	199		1		
Syslog Address	200	B200,B201, B202,B203	4	0	Destination address for syslog messages. If set to 0 (default), syslog is broadcasted.
Reserved	204- 499				
Application Config	500- 1499		1000		Assigned to user applications (see [1] for more details)

6 Autonomous Operation

This section describes the Reset Button functionality and the status information provided by the LEDs.

6.1 The Reset Button

A short press of the Reset Button resets the device. This can be disabled in the Web Configuration Interface.

In case the Reset Button is held for more than 10 seconds, factory default settings is restored. This is indicated by green LED being on and red LED blinking. Factory defaults feature can be disabled in the Web Configuration Interface.

Note: During power-up, the reset button has also another function. If reset button is held during power-up, it will bring the device to the Boot Menu mode. If that wasn't your intention, just reset the device again.

6.2 Device Availability with the Green and Red Status LEDs

During the initial boot up sequence the 2 status indicator LEDs are used as for standard Barix products.

No Application loaded (only bootloader) or started with hold reset button during power up

The green LED is on and the red LED blinks.

Application starts (Barix boot-up sequence):

First the red goes on and the green LED blinks once. Then during the startup the green and red LEDs blink.

During DHCP the red LED blinks with a continuous cycle. The green LED blinks five times and then pause four times.

If an error is detected the red led remains on and the device resets itself after the green LED has indicated the error as follows:

Error

Corrupt application or IP address conflict

The Network hardware could not be initialized or a Corrupt MAC address

Green LED blinks

five times.

three times.

Application running

After the boot-up when the application starts red and green LED indicate the following states:

Green LED	Red LED	State
OFF	BLINKING	Application is detecting USB devices or announcing the assigned IP address using SonicIP technology
ON	OFF	The application is operational.

Green LED	Red LED	State
ON	ON	Reset Button is being pressed, the unit will reboot after releasing the button.
ON	BLINKING	Reset Button has been kept pressed and the unit is ready to set factory defaults after releasing the button.

6.3 Yellow and Green Network Interface LEDs

The yellow LED (being on) indicates ACTIVITY on the network interface.

The green LED indicates the LINK status as described in the following table:

7 Firmware and standard File Upload

The standard Barix method is available to update the units with Firmware, the WEB UI, default configuration and language files. This method is summarized..

Update via a standard web browser

Please Note:

The update described below will NOT change the current configuration settings!

Reverting to Factory Defaults is not needed but recommended.

1. Open a browser and type the announced IP address into the URL field and hit the ENTER key.
2. Click on the CONFIGURATION button to enter the configuration pages.
3. Click on the UPDATE button to enter the update page.
4. Click on "Please click here to continue" to launch the update process.
The device will restart in a special mode called Bootloader showing a number counting down. Upon start up the following screen appears ready for the update process. (The word Browse may differ depending on browser and language).

Update	

Resource	Browse
	Upload
	Reboot

Advanced Update	

5. To upload an update click on "Browse" to locate the file you want to update. Browse to the folder "update_rescue" and choose the file compound.bin.
6. Once selected, click on "Upload". This process can take a few minutes. After a successful upload click on the "update" link and when the Update window reappears click the "Reboot" button or if there is no button, click on Browse and select the file "reboot". The device will reboot with the new firmware.

7.1 Advanced update

Individual files may be loaded using the Advanced Update.

1-4 Steps 1-4 above

5. Click on "Advanced Update" and the following screen appears ready for the advanced update process. (The word Browse may differ).

Advanced Update	

Target	Browse
Resource	Upload
	Reboot

Update	

6. For advanced updates an additional parameter is required called the "Target". This specifies where in the Flash to load the file as follows:

Type	Resource	Target
Firmware	abclw.rom	8K
WEB UI and Sonic IP	abclapp.cob	WEB3
FW Ext. 1	fs.bin	WEB1
FW Ext. 2	sg.bin	WEB2
FW Ext. 3	bclio.bin	WEB9
Application programs	applications.cob	WEB10
Custom application	custom1.cob	WEB5

Bootloader UNIFULL.SPB Loading the Boot Loader over the WEB is not normally recommended, please ask Barix for advice.

7. To make an update type in the Target and then click on "Browse" to locate the corresponding file in the "update_rescue" folder.

8. Once selected, click on "Upload". This process can take up to one minute. After a successful upload click on the "update" link and when the Update

window reappears click the "Reboot" button.
The device will reboot with the new resource file.

Appendix A: BCL I/O Address Map

ABCL provides access to peripherals of the hardware through I/O registers. They are numbered in ascending sequence starting from 1 and can hold an integer number (value range depends on the hardware and is defined below). I/O registers can be set using the **IOCTL** call and read using the **IOSTATE** call.

I/O registers are split into groups by their addresses, the groups are associated with particular inputs or outputs (e.g. relays, digital outputs, digital inputs etc.). Some of the registers are only virtual and are not mapped to any hardware, these can be used by the application to store an information which can be then retrieved e.g. via SNMP.

Hardware identification

Virtual registers starting from number 60000 describe the hardware ABCL runs on. These registers are read-only and have the following meaning:

Register	Description
60000	Hardware type identifier
60001	Number of serial ports
60002	Number of relays
60003	Number of digital outputs (excluding RTS/CTS)
60004	Number of digital inputs (excluding RTS/CTS)
60005	Number of analog outputs
60006	Number of analog inputs
60007	Number of keys if the hardware features a keyboard; 0 if no keyboard is available

Device overview

This section gives a hardware capabilities overview of Barix devices supported by ABCL.

Device name	HW ID	Serial ports	Relays	Digital outputs	Digital inputs	Analog outputs	Analog inputs	Display	Keyboard
Exstreamer Red Box	1	1	0	0	0	0	0	none	none
Annunicom legacy/100	7	1	1	0	2	0	0	none	2 keys (digital inputs)
Instreamer/Instreamer 100	8	1	0	0	0	0	0	none	none
Exstreamer Digital	9	1	0	0	0	0	0	none	none
IPAM	13	2	0	8	8	0	0	none	none

Device name	HW ID	Serial ports	Relays	Digital outputs	Digital inputs	Analog outputs	Analog inputs	Display	Keyboard
Exstreamer 200	15	1	0	0	0	0	0	none	none
IPAM Carrier Board	16	2	0	7	7	0	0	none	none
Annunicom 1000	17	1	9	0	9	0	2	none	8 keys (digital inputs)
Annunicom 200	19	1	1	0	1	0	0	none	2 keys (digital inputs)
Exstreamer 110	20	1	1	0	0	0	0	2x16	none
Exstreamer 1000	21	1	4	0	4	0	1	none	4 keys
PSI6 Paging Station	25	0	0	16	16	0	0	2x24	16 keys (keypad)

I/O registers

This section describes the layout of the I/O hardware registers.

The input and output capabilities depend on the hardware the ABCL runs on and not all I/Os may be present on all hardware (e.g. there are no relays available on Exstreamer 100). Write to these registers has no effect and read always returns 0. List of available registers for each hardware is defined below.

I/Os are always numbered from the lowest number in ascending order up to the number of relays, inputs, etc.

E.g. a device with 4 digital inputs uses the following registers:

- 201 - 1st digital input
- 202 - 2nd digital input
- 203 - 3rd digital input
- 204 - 4th digital input

Exception:

RTS and CTS on serial ports are accessible through digital outputs 200, 199, 198, ... (RTS1, RTS2, RTS3, ...) and through inputs 300, 299, 298, ... (CTS1, CTS2, CTS3, ...)

1...100 Relay outputs (Read, Write)

I/O	Output
1	Relay 1
2	Relay 2

value	Function
0	Set output to inactive (relay contact open)
1	Set output to active (relay contact closed)

I/O	Output
...	...
32	Relay 32
33..100	Reserved for future use

value	Function
999	Toggle output
n	Pulse output for n*100ms, valid values 2-998

Read from a relay output returns the current state of the relay. The default state of relay outputs (after reset) is **inactive**.

If set with the register write command to "999", the output toggles. If set with the register write command to a value >1, output generates a pulse of duration n*100 milliseconds. The pulse appears on the output within maximum 20ms time.

The registers 1 to 32 map to the relays, the addresses 33 to 100 are reserved for future internal use. The below table shows availability of relays on Barix hardware.

Hardware	Relays
Legacy Exstreamer devices	N/A
Exstreamer 100	N/A
Exstreamer 110	1
Exstreamer 200	N/A
Exstreamer 1000	1..4
Legacy Instreamer	N/A
Instreamer 100	N/A
Legacy Annuncicom	1
Annuncicom 100	1
Annuncicom 200	1
Annuncicom 1000	1..9
IPAM board	N/A
PSI6 Paging Station	N/A

101...200 Digital outputs (Read, Write)

I/O	Output
101	Output 1
102	Output 2
...	...
132	Output 32
133..196	Reserved
197..198	Reserved for further serial ports
199	RTS out on 2. serial port
200	RTS out on 1. serial port

value	Function
0	Set output to inactive
1	Set output to active
999	Revert (toggle) output
n	Pulse output for n*100ms, valid values 2-998

Read from a digital output returns the current state of the output.

If set with the register write command to “999”, the output toggles. If set with the register write command to a value >1, output generates a pulse of duration $n \times 100$ milliseconds. The pulse appears on the output within maximum 20ms time.

Active means “high” state on the output (on serial port RTS asserted). **Inactive** means “low” state on the output (on serial port RTS not asserted).

Important note: The RTS signal is by default (after reset) **active** as it is expected to be used by the flow-control. If used for other purposes than signaling (e.g. to control external hardware) this fact should be taken into account when connecting the external hardware (and use e.g. a signal inverter) to avoid problems like door being open after reset.

The addresses 101 to 132 are mapped to the digital outputs on the hardware, the addresses 133 to 196 are reserved for future use.

Addresses 199 and 200 are mapped to the RS-232 RTS output signal (on serial 2 and serial 1 respective) and should only be addressed if HW flow control is disabled. The addresses 197 and 198 are not used and reserved for the future for serial ports 4 and 3. Read from an RTS output is valid **only if used as I/O** and not when used for the flow-control.

The below table lists available digital outputs on Barix hardware.

Hardware	Digital Outputs	RTS Outputs
Legacy Exstreamer devices	N/A	200
Exstreamer 100	N/A	200
Exstreamer 110	N/A	200
Exstreamer 200	N/A	200
Exstreamer 1000	N/A	200
Legacy Instreamer	N/A	200
Instreamer 100	N/A	200
Legacy Annuncicom	N/A	200
Annuncicom 100	N/A	200
Annuncicom 200	N/A	200
Annuncicom 1000	N/A	200
IPAM board	101..108	199, 200
PS16 Paging Station	101..116 (extensible)	N/A

On the IPAM board the digital outputs and inputs are mapped directly to PIOs of the board. Read from a register (**IOSTATE**) switches automatically the PIO to input mode, writing to a register (**IOCTL**) switches the PIO to output mode. The below table lists the assignment of PIOs to digital outputs on IPAM. PIOs 8, 24 and 25 are used by the ABCL firmware (software reset, red and green LEDs) and can not be controlled by the BCL application.

By default all PIOs are initialised as inputs with weak pull-up.

On the IPAM Carrier Board the PIO17 is not available as it is used by RTC.

Digital Output	PIO
101	PIO11
102	PIO16
103	PIO17

Digital Output	PIO
I04	PIO20
I05	PIO22
I06	PIO23
I07	PIO29
I08	PIO30
Software reset	PIO8
Green LED	PIO24
Red LED	PIO25

201...300 Digital inputs (Read only)

I/O	Input
201	Input 1
202	Input 2
...	...
232	Input 32
233..396	Reserved
297..298	Reserved for further serial ports
299	CTS input on 2. serial port
300	CTS input on 1. serial port

value	Default function (low-active)
0	Input is active
1	Input is inactive

Input values are hardware dependent, most devices use low-active logic, however more than two values can be defined (error states). See the table below and the hardware specific sections at the end of this document.

For CTS inputs **value 1** means signal asserted, whereas **value 0** means signal not asserted.

If a CTS input is used for flow-control, read is valid and reflects the current state of CTS.

The addresses 201 to 232 are assigned to digital inputs, the addresses 233 to 296 are reserved for future use. The addresses 299 and 300 are assigned to CTS inputs on the second and the first serial port respectively. The addresses 297 and 298 are reserved for future use for serial ports 3 and 4.

The below table shows the assignment of digital inputs on Barix hardware.

Hardware	Digital Inputs	Value	CTS Inputs
Legacy Exstreamer devices	N/A	N/A	300
Exstreamer 100	N/A	N/A	300
Exstreamer 110	N/A	N/A	300
Exstreamer 200	N/A	N/A	300
Exstreamer 1000	201..204	Low active (0=active)	300
Legacy Instreamer	N/A	N/A	300
Instreamer 100	N/A	N/A	300
Legacy Annuncicom	201..202	Low active (0=active)	300
Annuncicom 100	201..202	Low active (0=active)	300
Annuncicom 200	201..202	Low active (0=active)	300

Hardware	Digital Inputs	Value	CTS Inputs
Annunicom 1000	201..208	0 = input activated 1 = input not activated 2 = short circuit 3 = input open (not connected)	300
IPAM board	201..208	Low active (0=active)	299, 300
PSI6 Paging Station	201..216 (extensible)	Low active (0=active)	N/A

On the IPAM board the digital outputs and inputs are mapped directly to PIOs of the board. Read from a register (**IOSTATE**) switches automatically the PIO to input mode, writing to a register (**IOCTL**) switches the PIO to output mode. The below table lists the assignment of PIOs to digital inputs on IPAM. PIOs 8, 24 and 25 are used by the ABCL firmware (software reset, red and green LEDs) and can not be controlled by the BCL application.

By default all PIOs are initialised as inputs with weak pull-up.

On the IPAM Carrier Board the PIO17 is not available as it is used by RTC.

Digital Output	PIO
I01	PIO11
I02	PIO16
I03	PIO17
I04	PIO20
I05	PIO22
I06	PIO23
I07	PIO29
I08	PIO30
Software reset	PIO8
Green LED	PIO24
Red LED	PIO25

301...400 Virtual I/O bits (Read, Write)

I/O	Function
301...400	Virtual bits

value	Function
0	Bit is inactive
1	Bit is active

The addresses 301 to 400 are mapped to virtual 1-bit registers (implemented in memory only). They can be used as a memory storage or to pass an information to external devices e.g. through SNMP.

401...500 Reserved

These registers are reserved for future use. Write to these registers has no effect and read always returns 0.

501...600 Analog inputs (Read only)

The analog inputs are hardware specific.

Hardware	Analog inputs	Values
Legacy Exstreamer devices	N/A	N/A
Exstreamer 100	N/A	N/A
Exstreamer 110	N/A	N/A
Exstreamer 200	N/A	N/A
Exstreamer 1000	501 – temperature sensor	In 0.01°C units; value 0 represents 0°C
Legacy Instreamer	N/A	N/A
Instreamer 100	N/A	N/A
Legacy Annuncicom	N/A	N/A
Annuncicom 100	N/A	N/A
Annuncicom 200	N/A	N/A
Annuncicom 1000	501 – internal temperature 502 – battery voltage	In 0.01°C units; value 0 represents 0°C In 0.01V units; value 0 represents 0V
IPAM board	N/A	N/A
PS16 Paging Station	N/A	N/A

601...700 Reserved

These registers are reserved for future use. Write to these registers has no effect and read always returns 0.

701...1000 Virtual 16bit registers (Read, Write)

I/O	Function
701...1000	Virtual 16bit registers

value	Function
0...65535	Register value

The addresses 701 to 1000 are mapped to virtual 16-bit registers (implemented in memory only). They can be used as a memory storage or to pass an information to external devices e.g. through SNMP.

Instreamer Legacy/I00, Exstreamer Legacy/I00/200

IO	Type	Name	Note
200	Digital output	RTS	
300	Digital input	CTS	

Exstreamer I10

IO	Type	Name	Note
I	Relay output	Relay I	
200	Digital output	RTS	
300	Digital input	CTS	

Exstreamer I10 features a 2x16 character LCD display.

Annunicom Legacy/I00/200

IO	Type	Name	Note
I	Relay output	Annunicom relay	
200	Digital output	RTS	
201	Digital input	IN0 / AI Phone button (Ann200)	Low active
202	Digital input	INI	Low active
300	Digital input	CTS	

Digital inputs I and 2 can be read as key events.

IPAM/IPAM Carrier Board

On the IPAM board the digital outputs and inputs are mapped directly to PIOs of the board. Read from a register (**IOSTATE**) switches automatically the PIO to input mode, writing to a register (**IOCTL**) switches the PIO to output mode. By default all PIOs are initialised as inputs with weak pull-up.

On the IPAM Carrier Board the PIO17 is not available as it is used by RTC.

IO	Type	Name	Note
101	Digital output	PIO11	
102	Digital output	PIO16	
103	Digital output	PIO17	Not available on IPAM Carrier Board as PIO17 is attached to RTC
104	Digital output	PIO20	
105	Digital output	PIO22	
106	Digital output	PIO23	
107	Digital output	PIO29	
108	Digital output	PIO30	
199	Digital output	RTS on serial 2	
200	Digital output	RTS on serial 1	
201	Digital input	PIO11	Low active
202	Digital input	PIO16	Low active
203	Digital input	PIO17	Low active Not available on IPAM Carrier Board as PIO17 is attached to RTC
204	Digital input	PIO20	Low active
205	Digital input	PIO22	Low active
206	Digital input	PIO23	Low active
207	Digital input	PIO29	Low active
208	Digital input	PIO30	Low active
299	Digital input	CTS on serial 2	
300	Digital input	CTS on serial 1	

Digital inputs 1-8 can be read as key events.

Exstreamer 1000

IO	Type	Name	Note
1	Relay output	Relay 1	
2	Relay output	Relay 2	
3	Relay output	Relay 3	
4	Relay output	Relay 4	
200	Digital output	RTS	
201	Digital input	Input 1	Low active
202	Digital input	Input 2	Low active
203	Digital input	Input 3	Low active
204	Digital input	Input 4	Low active
300	Digital input	CTS	
501	Analog input	Internal temperature	In 0.01 °C units; value 0 represents 0 °C

Digital inputs 1-4 can be read as key events.

Annunicom 1000

IO	Type	Name	Note
1	Relay output	Relay 1	
2	Relay output	Relay 2	
3	Relay output	Relay 3	
4	Relay output	Relay 4	
5	Relay output	Relay 5	
6	Relay output	Relay 6	
7	Relay output	Relay 7	
8	Relay output	Relay 8	
9	Relay output	Relay 9	Default state is "closed" (value=1)
200	Digital output	RTS	
201	Digital input	Input 1	0 = input activated 1 = input not activated 2 = short circuit 3 = input open (not connected)
202	Digital input	Input 2	
203	Digital input	Input 3	
204	Digital input	Input 4	
205	Digital input	Input 5	
206	Digital input	Input 6	
207	Digital input	Input 7	
208	Digital input	Input 8	
209	Digital input	Main input power status	Low – Main Supply $\geq 17V$ High – Main Supply $< 17V$
300	Digital input	CTS	
501	Analog input	Internal temperature	In 0.01°C units; value 0 represents 0°C
502	Analog input	Battery input voltage	In 0.01V units; value 0 represents 0V

Digital inputs 1-8 can be read as key events. States 0 and 2 are treated as key pressed, states 1 and 3 as key released.

PSI6 Paging Station

IO	Type	Name	Note
I01	Digital output	LED 0	
I02	Digital output	LED 1	
I03	Digital output	LED 2	
I04	Digital output	LED 3	
I05	Digital output	LED 4	
I06	Digital output	LED 5	
I07	Digital output	LED 6	
I08	Digital output	LED 7	
I09	Digital output	LED 8	
I10	Digital output	LED 9	
I11	Digital output	LED 10	
I12	Digital output	LED 11	
I13	Digital output	LED 12	
I14	Digital output	LED 13	
I15	Digital output	LED 14	
I16	Digital output	LED 15	
201	Digital input	Key 0 (lower left)	0 = key pressed 1 = key not pressed
202	Digital input	Key 1	
203	Digital input	Key 2	
204	Digital input	Key 3	
205	Digital input	Key 4	
206	Digital input	Key 5	
207	Digital input	Key 6	
208	Digital input	Key 7 (lower right)	
209	Digital input	Key 8 (upper left)	
210	Digital input	Key 9	
211	Digital input	Key 10	
212	Digital input	Key 11	
213	Digital input	Key 12	
214	Digital input	Key 13	
215	Digital input	Key 14	
216	Digital input	Key 15 (upper right)	

The Paging console features a 2x24 character LCD display and a 16-key keyboard.

The keyboard can be either polled directly through digital inputs or read as key events (key presses and releases). Every key on the keyboard has a LED, the LEDs are mapped to digital outputs corresponding to the respective digital input of the key.

The keyboard can be extended up to 48 keys, the number of digital inputs and outputs then increases.