E U R O P E A N S O U T H E R N O B S E R V A T O R Y

Organisation Européenne pour des Recherches Astronomiques dans l'Hémisphère Austral

Europäische Organisation für astronomische Forschung in der südlichen Hemisphäre

VERY LARGE TELESCOPE

VLT Software

Environments

Common Configuration

User Manual

Doc.No. VLT-MAN-ESO-17210-0855

Issue 3.0

Date 30.10.1998

Prepared F.Carbognani ....
Name Date Signature

Approved G.Filippi ....
Name Date Signature

Released G.Raffi ....
Name Date Signature

Change Record

Issue/Rev.	Date	Section/Page affected	Reason/Initiation/Document/Remarks
1.0	11.06.1996	All	First Release, except the following chapters, in which change-bars indicate modifications:
		5, 7	Taken from VLT-MAN-ESO-17210-0375, “Driver Development Guide & User Manual”
		7.2.2	Error definition files are now loaded on LCU.
		7.4.1	Editing of *bootScript* no longer recommended.
2.0	05.11.1997		**vcc multi-network + multi-CPU extension**
		2.3	Updated location of RtapEnvList.
		3.7	Added off-line environment creation.
		5	Updated figures.
		5.4.1	Extended for multi-CPU systems.
		5.4.3	Extended for multi-network systems.
		8	Updated man-pages.
3.0	30.10.1998	2.3, 3.5, 8	Updated for new CCS-Lite on OCT98

The information contained in this manual is intended to be used in the ESO VLT project by ESO and authorized external contractors only.

While every precaution has been taken in the development of the software and in the preparation of this documentation, ESO assumes no responsibility for errors or omissions, or for damage resulting from the use of the software or of the information contained herein.

TABLE OF CONTENTS

1 INTRODUCTION 9
1.1 Purpose 9
1.2 Scope 9
1.3 Reference Documents 9
1.4 Abbreviations And Acronyms 10
1.5 Stylistic Conventions 10
1.6 Naming Conventions 10
1.7 Problem Reporting and Change Request 10

2 CONFIGURATION DATABASE 11
2.1 Purpose of the VCCDB 11
2.2 On-line Configuration Access 11
2.3 System Configuration Files 12
2.3.1 exports (HP) or dfstab (SUN) 12
2.3.2 hosts 12
2.3.3 services 12
2.3.4 logLCU.config 13
2.3.5 lqs.boot 13
2.3.6 RtapEnvList 13
2.3.7 CcsEnvList 13
2.3.8 .rhosts 13

3 UNDERSTANDING ENVIRONMENTS 15
3.1 Types of Environments 15
3.2 Tools to Manage Environments 15
3.3 Operations on Environments 16
3.4 To Setup an RTAP Environment 16
3.5 To Setup a QSEMU (CCS-Lite) Environment 18
3.6 To Setup an LCU Environment 20
3.7 To Create an Environment Off-Line 22

4 ENVIRONMENT SETUP - vccEnv 23
4.1 Overview 23
4.2 Starting from the Command Line 23
4.3 Panel Description 24
4.4 Actions 24

5 CONFIGURATION OF LCU ENVIRONMENTS - vccConfigLcu 25
5.1 Overview 25
5.2 Starting from the Command Line 25
5.3 Example of Usage 26
5.4 Panel Description 27
5.4.1 Environments Selection 27

5.4.2 Root Configuration 27
5.4.3 Network Configuration 28
5.4.4 Modules Configuration. 28
5.4.5 Devices Configuration 29
5.4.6 Processes 29
5.4.7 Target Files 29
5.4.8 Actions. 30
5.5 Known Problems 30

6 CONFIGURATION ACCESS FOR PROGRAMMERS 32
6.1 Panel Mega-Wigets 32
6.2 Programmatic Access from Seqencer-Scripts. 32

7 THE LCU BOOT ENABLER - lcuboot 33
7.1 Purpose of lcuboot 33
7.2 Available lcuboot Functions. 33
7.2.1 lcuboot Automatic Environment Setup 33
7.2.2 lcuboot General Module Support 34
7.2.3 lcuboot Driver and Device Installation 34
7.2.4 lcuboot LCC Support 34
7.2.5 lcuboot File Access Support 35
7.3 How to create LCU Module-Boot-Scripts. 35
7.3.1 Module-Boot-Scripts for General Module Installation. 35
7.3.2 Module-Boot-Scripts for Automatic Driver and Device Installation 36
7.4 How to configure LCUs with lcuboot. 37
7.4.1 To create a boot-script suitable for lcuboot 37
7.4.2 To configure module-boot-scripts for your site 38
7.4.3 To configure module-boot-scripts for a specific LCU 38

8 REFERENCE 39
8.1 User Commands Reference 39
8.1.1 envsCreate(1) 40
8.1.2 envsKill(1). 42
8.1.3 vccConfigLcu(1). 43
8.1.4 vccEnv(1). 50
8.1.5 vccEnvCheck(1) 53
8.1.6 vccEnvCreate(1). 55
8.1.7 vccEnvDelete(1) 59
8.1.8 vccEnvInit(1). 61
8.1.9 vccEnvStart(1) 63
8.1.10 vccEnvStop(1). 65
8.1.11 vccShow(1) 67
8.2 Functions Reference. 68
8.2.1 lcubootAutoDrv(1) 69

8.2.2 lcubootAutoEnv(1) 73
8.2.3 lcubootAutoGen(1) 75
8.2.4 lcubootAutoLcc(1) 78
8.2.5 lcubootError(1) 79
8.2.6 lcubootFile(1) 81
8.2.7 lcubootLog(1) 82
8.3 Directory Reference 84
8.3.1 ENVIRONMENTS_LCU(5) 85
8.3.2 ENVIRONMENTS_QSEMU(5) 87
8.3.3 ENVIRONMENTS_RTAP(5) 89
8.3.4 lcuboot(5) 91
8.4 Files Reference 93
8.4.1 bootScript(5) 94
8.4.2 devicesFile(5) 97
8.4.3 userScript(5) 98
8.5 Panel Widgets and Libraries 99

1 INTRODUCTION

1.1 Purpose

This document describes how to create and maintain environments of any type (Rtap, Qsemu, LCU) with respect to the VLT standards. It contains the instructions for the VLT-tools to support these tasks, which are part of the workstation-based ***vcc*** and ***envs*** modules, and the LCU-based boot enabler, named ***lcuboot***.

The environment setup and configuration features are provided as:

1. a set of **GUI panels** for manual control of environments.

   These interactive panels make use of:
2. a set of **WS commands** to create, start, stop, delete, and check all types of environments. These commands are also used by higher level procedures (e.g. automatic test support tool kit).

   For LCUs, these commands can generate files which are compatible with:
3. a set of **LCU library functions** that facilitate the boot procedure.

1.2 Scope

The User Manual part of this document describes the following VLT software modules:

- **lcuboot** from version 1.29 for LCU booting functions
- **vcc** from version 2.7 for environment setup and configuration
- **envs** from version 1.17 for basic environment creation from templates

The document assumes that the reader has a good knowledge of UNIX and the VxWorks operating systems.

1.3 Reference Documents

The following documents contain additional information and are referenced in the text.

[1]	VxWorks Version 5.3 Programmer's Guide Wind River Systems
[2]	VxWorks Version 5.3 Reference Manual Wind River Systems
[3]	HOS/ACC - User Manual VLT-MAN-ESO-17230-1023, 1.1
[4]	HOS/ACC - Software User Manual VLT-MAN-ESO-17230-1024, 1.1
[5]	Q-Server Emulator User Manual VLT-MAN-ESO-17210-0422, 3.0
[6]	VLT Software Problem Report Change Request User Manual VLT-MAN-ESO-17200-0981, 2.0 15/01/96

1.4 Abbreviations And Acronyms

The following abbreviations and acronyms are used in this document:

ACC	Access and Configuration Control
CCS	Central Control Software
HW	Hardware
I/O	Input/Output
LAN	Local Area Network
LCC	LCU Common Software
LCU	Local Control Unit
N/A	Not Applicable
OS	Operating System
SW	Software
TBD	To Be Defined
VCCDB	VLT Common Configuration Database
VLT	Very Large Telescope
WS	Workstation

1.5 Stylistic Conventions

The following styles are used:

bold - in the text, for commands, filenames, prefixes/suffixes as they have to be typed.

italic - in the text, for parts that have to be substituted with the real content before typing.

`teletype` - for examples.

`<name>` in the examples, for parts that have to be substituted with the real content before typing.

bold and *italic* are also used to highlight words.

Items which are subject to change in future versions are marked in this way (AuthorRemark).

✗Very important items are marked in this way (Warning).

1.6 Naming Conventions

This implementation follows the naming conventions as outlined in the VLT Programming Standards.

1.7 Problem Reporting and Change Request

See [6] for instructions.

2 CONFIGURATION DATABASE

2.1 Purpose of the VCCDB

The main idea of a VLT common configuration database (**VCCDB**) is to have a single centralized configuration definition for the whole network, from which all host-based files can be derived, and from which configuration data can be queried by engineering interfaces and applications at run-time.

The SQL-based database of the Access and Configuration Control (ACC) module has been chosen for this purpose, since it contains - apart from other data - all information that are relevant for environment configuration. [1]

To set up such a database, please refer to the dedicated ACC documentation for details [4].

2.2 On-line Configuration Access

The vcc tools can work with or without a VCCDB, but the functionality without it will be very limited. Some of the tools and programs will not work at all. The ACC database is queried for configuration information at run-time.

In order to access the VCCDB at run-time, you have to set the shell-variable **ACC_HOST** to the host on which the database server is running, e.g.:

```
setenv ACC_HOST te49
```

After that the configuration information is taken from that VCCDB.

If you wish to work without VCCDB, then you must not define the shell-variable ACC_HOST and, so that no access will take place.

Note that none of the tools and functions described in this manual ever *writes* into the VCCDB. Only ACC facitlities are intended to write to the database.

1. In the JUL95/DEC95 release a prototype of the VCCDB was implemented as Rtap database. Such implementation is no more available. The shell-variable *VLT_VCCENV* shall not be defined any longer.

2.3 System Configuration Files

✗Global configuration files can be derived automatically from the VCCDB. Such a feature will be added in a future release. For the time being, these files shall still be edited by hand.

The following table shows which files are needed for which type of environment, and in which manual pages detailed information about their contents can be found:

File	Man-Page	Needed for type

HP-UX: /etc/exports
Sun-Solaris: /etc/dfs/dfstab
	exports(4)
	dfstab(4)
		LCU

/etc/hosts	hosts(4)	RTAP, QSEMU
/etc/services	services(4)	RTAP
$VLTDATA/config/logLCU.config	logLCU.config(5)	RTAP, QSEMU
$VLTDATA/config/lqs.boot	lqs.boot(5)	LCU
$VLTDATA/config/CcsEnvList	not jet avalaible	QSEMU
/etc$RTAPROOT/RtapEnvList	RtapEnvList(4)	RTAP
~vx/.rhosts	remsh(1) or rsh(1)	LCU

2.3.1 exports (HP) or dfstab (SUN)

All LCUs that boot from the WS must have an entry in this file to allow NFS mounting. Note that an additional command is necessary to commit any changes in the file.

2.3.2 hosts

All nodes, WS and LCU, must have an entry in this file to establish the mapping between host-names and IP-addresses.

2.3.3 services

From the communication point of view, each environment is identified by the node on which is running and a TCP/IP port number. The same number can be used on different nodes for the same type of environment. Currently we use:

- 2160 for LCU environments
- 2223 for QSEMU environments
- one number in the range 2001-2999 for each RTAP environment

2.3.4 logLCU.config

To establish the assignment between LCU and WS environment to which logs shall be sent.

2.3.5 lqs.boot

Each LCU needs to know from which host it has to boot and which are the other environments: CCSs and QSEMUs (an LCU should not talk directly to other LCUs). If LCU(s) need to communicate with more WS environments than the assigned boot environment, then the file "`$VLTDATA/config/lqs.boot`" must be created and edited. The standard file in "`$VLTROOT/vw/bin/$CPU/lqs.boot`" can be taken as template. Otherwise the installed default is enough.

2.3.6 RtapEnvList

Each CCS (RTAP) environment needs to know where other CCS, QSEMU, LCC environments are located, they can be both local or remote to the WS. `RtapEnvList` provides such a mapping.

2.3.7 CcsEnvList

This is the equivalent of RtapEnvList for CCS-Lite Environment. It also uses the same sintax. A template is installed in `$VLTROOT/config/CcsEnvList`.

2.3.8 .rhosts

The LCU performs during start-up remote-shell accesses to the booting WS under the user-name 'vx'. In order to enable that, the node-name of the LCU must be stated in the file `~vx/.rhosts`, i.e. in the home-directory of user vx..

3 UNDERSTANDING ENVIRONMENTS

3.1 Types of Environments

The VLTSW is based on the concept of environments where processes can run and exchange messages with other processes, either in the same environment or in a remote environment, on the same machine or in other machines.

Environment can be:

CCS environment (on WS)
: based on an RTAP environment, providing database and communication facilities (q-server), and used to build WS applications. There can be one or more such environments per WS.

CCS-Lite (or QSEMU) environment (on WS)
: a limited environment used when no RTAP is installed (CCS-Lite). The present version provides both database and communication facilities.

LCC environment (on LCU)
: provides database and communication facilities (lcu-Qserver) to build WS applications. Maximum one environment per LCU. There can be one or more such environments per WS

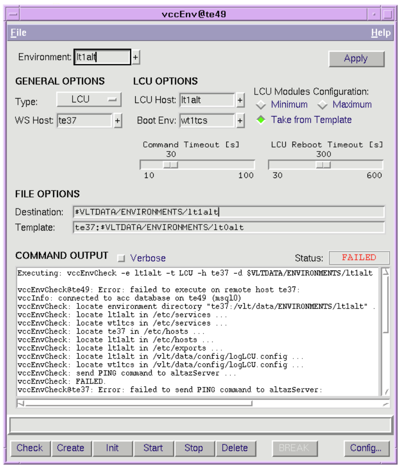
An environment is uniquely identified by the environment name. Remember that environment names are limited to 7 chars and the first letter is mandatory "w" for WS and "l" for LCU.

REMARK: Real VLT environment names must follows the conventions defined by the applicable version of "VLT LAN's Specification". In the following example generic names are used.

3.2 Tools to Manage Environments

The following WS-based tools are available for the user:

- ***vccEnv**** - a set of commands to deal with all types of environments (see 8.1)
- ***vccEnv*** - a panel that allows interactive use of the *vccEnv** commands (see 4)
- ***vccConfigLcu*** - a panel specifically intended for configuration of LCU environments (see 5)



Environment: lt1alt

GENERAL OPTIONS
Type: LCU
WS Host: te37

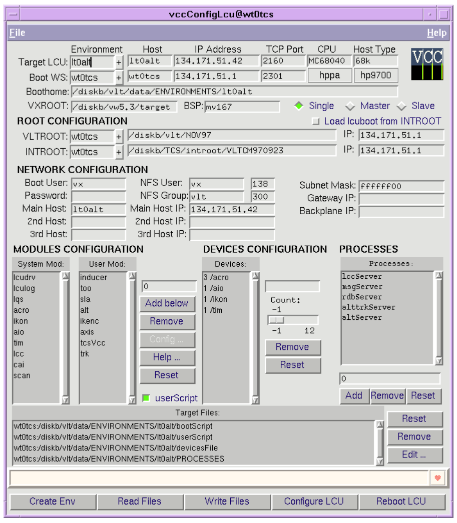
LCU OPTIONS
LCU Host: lt1alt
Boot Env: wt1tcs
LCU Modules Configuration: Take from Template
Command Timeout [s] 30 (10–100)
LCU Reboot Timeout [s] 300 (30–600)

FILE OPTIONS
Destination: $VLTDATA/ENVIRONMENTS/lt1alt
Template: te37:$VLTDATA/ENVIRONMENTS/lt0alt

COMMAND OUTPUT Verbose Status: FAILED

```
Executing: vccEnvCheck -e lt1alt -t LCU -h te37 -d $VLTDATA/ENVIRONMENTS/lt1alt

vccEnvCheck@te49: Error: failed to execute on remote host te37:
vccInfo: connected to acc database on te49 (msql0)
vccEnvCheck: locate environment directory "te37:/vlt/data/ENVIRONMENTS/lt1alt" .
vccEnvCheck: locate lt1alt in /etc/services ...
vccEnvCheck: locate wt1tcs in /etc/services ...
vccEnvCheck: locate te37 in /etc/hosts ...
vccEnvCheck: locate lt1alt in /etc/hosts ...
vccEnvCheck: locate lt1alt in /etc/exports ...
vccEnvCheck: locate lt1alt in /vlt/data/config/logLCU.config ...
vccEnvCheck: locate wt1tcs in /vlt/data/config/logLCU.config ...
vccEnvCheck: send PING command to altazServer ...
vccEnvCheck: FAILED.
vccEnvCheck@te37: Error: failed to send PING command to altazServer:
```



	Environment	Host	IP Address	TCP Port	CPU	Host Type
Target LCU:	lt0alt	lt0alt	134.171.51.42	2160	MC68040	68k
Boot WS:	wt0tcs	wt0tcs	134.171.51.1	2301	hppa	hp9700

Boothome: /diskb/vlt/data/ENVIRONMENTS/lt0alt
VXROOT: /diskb/vw5.3/target BSP: mv167 Single

ROOT CONFIGURATION
VLTROOT: wt0tcs /diskb/vlt/NOV97 IP: 134.171.51.1
INTROOT: wt0tcs /diskb/TCS/introot/VLTCM970923 IP: 134.171.51.1

NETWORK CONFIGURATION
Boot User: vx NFS User: vx 138 Subnet Mask: ffffff00
Password: NFS Group: vlt 300 Gateway IP:
Main Host: lt0alt Main Host IP: 134.171.51.42 Backplane IP:
2nd Host: 2nd Host IP:
3rd Host: 3rd Host IP:

MODULES CONFIGURATION
System Mod: lcudrv, lculog, lqs, acro, ikon, aio, tim, lcc, cai, scan
User Mod: inducer, too, sla, alt, ikenc, axis, tcsVcc, trk
0
userScript

DEVICES CONFIGURATION
Devices: 3 /acro, 1 /aio, 1 /ikon, 1 /tim
Count: -1 (-1–12)

PROCESSES
Processes: lccServer, msgServer, rdbServer, alttrkServer, altServer
0

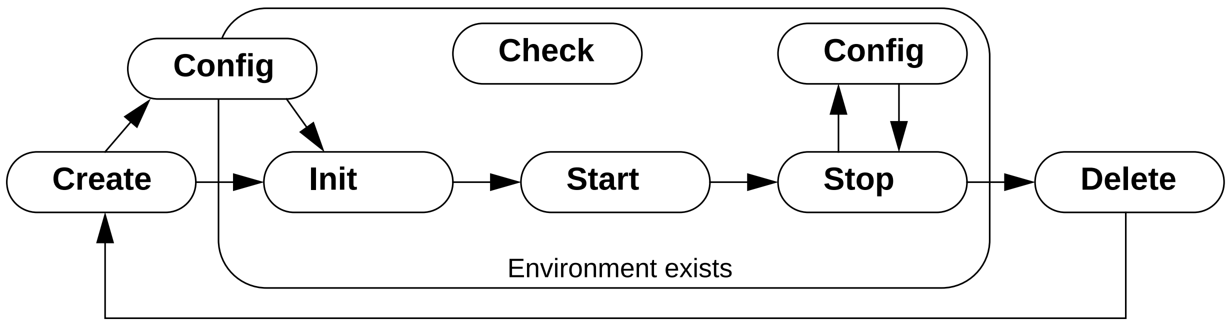
Target Files:
wt0tcs:/diskb/vlt/data/ENVIRONMENTS/lt0alt/bootScript
wt0tcs:/diskb/vlt/data/ENVIRONMENTS/lt0alt/userScript
wt0tcs:/diskb/vlt/data/ENVIRONMENTS/lt0alt/devicesFile
wt0tcs:/diskb/vlt/data/ENVIRONMENTS/lt0alt/PROCESSES

Their usage is described by examples in the next sections.

3.3 Operations on Environments

The provided tools implement some standard operations on environments.

The diagram shows which operations are defined. They exist for each type of environment, although the underlying steps are type-dependent: .



The following examples will explain the meaning of the operations for each environment type.

In order to execute the examples, please first start the *vccEnv* panel from the command line. Refer to chapter 4 for full instructions.

3.4 To Setup an RTAP Environment

The following steps demonstrate the possible actions with an RTAP environment named ***wtest***, and assigned to host *te49*.

- **Create the environment:**

  Enter ***wtest*** as environment-name and press <*Return*>.
  Then press the ***Create*** action-button.

```
Executing: vccEnvCreate -e wtest -t RTAP -h te49 -d $VLTDATA/ENVIRONMENTS/wtest

vccEnvCreate: copy standard template to “te49:/vlt/data/ENVIRONMENTS/wtest” ...
vccEnvCreate: make database from “te49:/vlt/data/ENVIRONMENTS/wtest/dbl” ...
vccEnvCreate: make env. directory world writable ...
```

  Application-specific databases must be added manually in the *dbl* subdirectory now, inclusive the *make db*. Alternatively, a template can be specified in the panel before *Create*, which then overrides the defaults, see chapter 4.

- **Configure the environment:**

  At this point the environment usually needs to be further configured for your specific case, e.g. the *RtapEnvTable* must be modified to add more application processes. Do do this, press the ***Config*** action-button to start Rtap’s configuration facilities, or edit the file *$VLTDATA/ENVIRONMENTS/wtest/RtapEnvTable.normal* directly.
  Note that *RtapEnvTable.normal* will be copied to *RtapEnvTable* during the *Init* operation.

- **Initialize the environment:**

  Press the ***Init*** action-button and confirm the dialog.

```
Executing: vccEnvInit -e wtest -t RTAP -h te49 -d $VLTDATA/ENVIRONMENTS/wtest

RTAP/Plus Copyright (c) Hewlett-Packard (Canada) Ltd. 1988-1994
All rights reserved.
Real-Time Applications Platform (RTAP/Plus) release A.06.60

vccEnvInit: create empty database in "te49:/vlt/data/ENVIRONMENTS/wtest" ...
vccEnvInit: load database from DB branch ...
vccEnvInit: restore normal RtapEnvTable ...
```

- **Check the environment before start:**

  Press the ***Check*** action-button. All checks should be OK until the sequence comes to processes, which are not running at this point.

```
Executing: vccEnvCheck -e wtest -t RTAP -h te49 -d $VLTDATA/ENVIRONMENTS/wtest

vccEnvCheck: locate environment directory "te49:/vlt/data/ENVIRONMENTS/wtest" ...
vccEnvCheck: locate wtest in /etc/services ...
vccEnvCheck: locate te49 in /etc/hosts ...
vccEnvCheck: locate wtest in /usr/rtap/etc/RtapEnvList ...
vccEnvCheck: send PING command to cmdManager ...
vccEnvCheck@te49: Error: failed to send PING command to cmdManager:

 ---------------- Error Structure ----------------
Time Stamp     : 96-06-10  13:31:58.419563
Process Number : 66       Process Name : msgSend
Environment    : wtest  StackId  : 2         Sequence : 4
Error Number   : 24       Severity : W
Module         : ccs      Location : ccsInit.c
Error Text     : ccsERR_ENV_NOT_ACTIVE : Environment wtest not active

child process exited abnormally
vccEnvCheck: FAILED.
```

- **Start the environment:**

  Press the ***Start*** action-button.

```
Executing: vccEnvStart -e wtest -t RTAP -h te49 -d $VLTDATA/ENVIRONMENTS/wtest

vccEnvStart: store log in "/vlt/data/ENVIRONMENTS/wtest/.RtapScheduler.log" ...
vccEnvStart: run RtapScheduler in background ...
```

- **Check the environment after start:**

  Press the ***Check*** action-button a second time. Now all checks should be OK.

```
Executing: vccEnvCheck -e wtest -t RTAP -h te49 -d $VLTDATA/ENVIRONMENTS/wtest

vccEnvCheck: locate environment directory "te49:/vlt/data/ENVIRONMENTS/wtest" ...
vccEnvCheck: locate wtest in /etc/services ...
vccEnvCheck: locate te49 in /etc/hosts ...
vccEnvCheck: locate wtest in /usr/rtap/etc/RtapEnvList ...
vccEnvCheck: send PING command to cmdManager ...
vccEnvCheck: send PING command to logManager ...
vccEnvCheck: send PING command to msgServer ...
```

- **Stop the environment:**

  Press the ***Stop*** action-button.

```
Executing: vccEnvStop -e wtest -t RTAP -h te49 -d $VLTDATA/ENVIRONMENTS/wtest
```

```
vccEnvStop: update snapshots with RtapForceSnap ...
vccEnvStop: stop wtest with RtapShutdown ...
```

- **Delete the environment:**

  Press the ***Delete*** action-button and confirm the dialog.

```
Executing: vccEnvDelete -e wtest -t RTAP -h te49 -d $VLTDATA/ENVIRONMENTS/wtest

vccEnvDelete@te49: Warning: failed to unprotect environment files under \
 "te49:/vlt/data/ENVIRONMENTS/wtest":
chmod: can't change .licensed: Not owner
vccEnvDelete: locate environment directory "te49:/vlt/data/ENVIRONMENTS/wtest" ...
vccEnvDelete: unprotect environment files under "te49:/vlt/data/ENVIRONMENTS/wtest"
vccEnvDelete: delete environment files under "te49:/vlt/data/ENVIRONMENTS/wtest"
```

3.5 To Setup a QSEMU (CCS-Lite) Environment

The following steps demonstrate the possible actions with a QSEMU environment named ***wtestqs***, running on host *te49*.

- **Create the environment:**

  Enter ***wte49qs*** as environment-name and press <*Return*>. Then press the ***Create*** action-button.

```
Executing: vccEnvCreate -e wte49qs -t QSEMU -h te49 -d $VLTDATA/ENVIRONMENTS/wte49qs
-s '' -l '' -w 'wte49qs' -m 'minimum' -j
vccEnvCreate: copy standard template to "te49:/vlt/data/ENVIRONMENTS/wte49qs" ...
vccEnvCreate:  locate  environment  directory  "te49:/diska/vlt/data/ENVIRONMENTS/
wte49qs" ...
vccEnvCreate: make database from "te49:/diska/vlt/data/ENVIRONMENTS/wte49qs/dbl" ...
vccEnvCreate: make env. directory world writable ...xecuting: vccEnvCreate -e wtestqs
-t QSEMU -h te49 -d $VLTDATA/ENVIRONMENTS/wtestqs
```

- **Check the environment before start:**

  Press the ***Check*** action-button. All checks should be OK until the sequence comes to processes, which are not running at this point.

```
Executing: vccEnvCheck -e wte49qs -t QSEMU -h te49 -d $VLTDATA/ENVIRONMENTS/wte49qs -
s '' -l '' -w 'wte49qs' -m 'minimum' -j

vccEnvCheck: locate environment directory "te49:/diska/vlt/data/ENVIRONMENTS/wte49qs"
...
vccEnvCheck: locate wte49qs in /etc/services ...
vccEnvCheck: locate te49 in /etc/hosts ...
vccEnvCheck: locate wte49qs in /vlt/data/config/CcsEnvList ...
vccEnvCheck: send PING command to cmdManager ...
vccEnvCheck@te49: Error: failed to send PING command to cmdManager:

 ---------------- Error Structure ----------------
Time Stamp     : 98-11-02  11:45:01.888001
Process Number : 74       Process Name : msgSend
Environment    : wte49qs  StackId  : 2         Sequence : 2
Error Number   : 24       Severity : W
Module         : ccs      Location : ccsInit.c
Error Text     : ccsERR_ENV_NOT_ACTIVE : Environment wte49qs not active
```

```
child process exited abnormally
vccEnvCheck: FAILED.
```

- **Start the environment:**

  Press the ***Start*** action-button.

```
Executing: vccEnvStart -e wte49qs -t QSEMU -h te49 -d $VLTDATA/ENVIRONMENTS/wte49qs
-s ‘’ -l ‘’ -w ‘wte49qs’ -m ‘minimum’ -j

vccEnvStart:  store  log  in  “/diska/vlt/data/ENVIRONMENTS/wte49qs/.ccsScheduler.log”
...
vccEnvStart: run ccsScheduler in background ...
vccEnvStart:  verify  start-up  log  /diska/vlt/data/ENVIRONMENTS/wte49qs/.ccsSchedul-
er.log ...
vccEnvStart: register to wte49qs and verify database access ...
```

- **Check the environment after start:**

  Press the ***Check*** action-button a second time. Now all checks should be OK.

```
Executing: vccEnvCheck -e wte49qs -t QSEMU -h te49 -d $VLTDATA/ENVIRONMENTS/wte49qs
-s ‘’ -l ‘’ -w ‘wte49qs’ -m ‘minimum’ -j

vccEnvCheck:   locate   environment   directory   “te49:/diska/vlt/data/ENVIRONMENTS/
wte49qs” ...
vccEnvCheck: locate wte49qs in /etc/services ...
vccEnvCheck: locate te49 in /etc/hosts ...
vccEnvCheck: locate wte49qs in /vlt/data/config/CcsEnvList ...
vccEnvCheck: send PING command to cmdManager ...
vccEnvCheck: send PING command to logManager ...
vccEnvCheck: send PING command to msgServer ...
```

- **Stop the environment:**

  Press the ***Stop*** action-button.

```
Executing: vccEnvStop -e wte49qs -t QSEMU -h te49 -d $VLTDATA/ENVIRONMENTS/wte49qs -
s ‘’ -l ‘’ -w ‘wte49qs’ -m ‘minimum’ -j

vccEnvStop: update snapshots with dbForceSnap ...
vccEnvStop: stop wte49qs with ccsShutdown ...
```

- **Delete the environment:**

  Press the ***Delete*** action-button and confirm the dialog.

```
Executing: vccEnvDelete -e wte49qs -t QSEMU -h te49 -d $VLTDATA/ENVIRONMENTS/wte49qs
-s ‘’ -l ‘’ -w ‘wte49qs’ -m ‘minimum’ -j

vccEnvDelete:   locate   environment   directory   “te49:/diska/vlt/data/ENVIRONMENTS/
wte49qs” ...
vccEnvDelete: unprotect environment files under “te49:/diska/vlt/data/ENVIRONMENTS/
wte49qs” ...
vccEnvDelete:  delete  environment  files  under  “te49:/diska/vlt/data/ENVIRONMENTS/
wte49qs” ...
```

3.6 To Setup an LCU Environment

The following steps demonstrate the possible actions with an LCU environment named ***ltest***, which is assigned to the LCU-node *te41* and shall boot from *wtest*.

- **Create the environment:**

  Enter ***ltest*** as environment-name and press <*Return*>. Adjust the fields *LCU-host* and *Boot-Env* in the *LCU Options* and select *Maximum* for module-configuration. Then press the ***Create*** action-button.

```
Executing: vccEnvCreate -e ltest -t LCU -h te49 -d $VLTDATA/ENVIRONMENTS/ltest \
 -s ‘’ -l ‘te41’ -w ‘wtest’ -m ‘maximum’

vccEnvCreate@te49: Warning: VccInfo: no valid result from query: GetByEnv wtest host-
Name
vccEnvCreate@te49: Error: failed to generate bootScript using: maximum:
VccInfo: no valid result from query: GetByEnv ltest {hostName tcpPort}
vccEnvCreate: copy standard template to “te49:/vlt/data/ENVIRONMENTS/ltest” ...
vccEnvCreate: make database from “te49:/vlt/data/ENVIRONMENTS/ltest/dbl” ...
vccEnvCreate: make env. directory world writable ...
vccEnvCreate: generate bootScript using: maximum ...
vccEnvCreate: FAILED.
```

  In this example no VCCDB is used, therefore some values could not be queried, and the procedure failed to create a fully configured *bootScript*. In such a case, all missing configuration parameters **must** be added manually in the next step. Otherwise further configuration is optional.

  Application-specific databases must be added manually in the *dbl* subdirectory now, inclusive the *make db*. Alternatively, a template can be specified in the panel before *Create*, which then overrides the defaults, see chapter 4.

- **Configure the environment:**

  We assume now that the previous step failed (partly) due to the missing VCCDB.

  Press the ***Config*** action-button to start *vccConfigLcu* (see chapter 5). Enter *ltest* again and press <*Return*>, then enter all missing parameters in order.

  The module configuration can now be adjusted according to your specific case.

  Finally press the ***Write Files*** button to regenerate the target files.

- **Initialize the environment:**

  Press the ***Init*** action-button and confirm the dialog.

```
Executing: vccEnvInit -e ltest -t LCU -h te49 -d $VLTDATA/ENVIRONMENTS/ltest -l ‘te41’

vccEnvInit@te49: Warning: VccInfo: no valid result from query: GetByEnv wtest hostName
vccEnvInit: execute bootChange sequence on LCU te41 and reboot ...
```

- **Check the environment before start:**

  Press the ***Check*** action-button. All checks should be OK until the sequence comes to processes, which are not running at this point.

```
Executing: vccEnvCheck -e ltest -t LCU -h te49 -d $VLTDATA/ENVIRONMENTS/ltest \
 -l ‘te41’ -w ‘wtest’

vccEnvCheck: Warning: VccInfo: no valid result from query: GetByEnv wtest hostName
vccEnvCheck: locate environment directory “te49:/vlt/data/ENVIRONMENTS/ltest” ...
vccEnvCheck: locate ltest in /etc/services ...
vccEnvCheck: locate wtest in /etc/services ...
vccEnvCheck: locate te49 in /etc/hosts ...
```

```
vccEnvCheck: locate te41 in /etc/hosts ...
vccEnvCheck: locate te41 in /etc/exports ...
vccEnvCheck: locate ltest in /vlt/data/config/logLCU.config ...
vccEnvCheck: locate wtest in /vlt/data/config/logLCU.config ...
vccEnvCheck: send PING command to inducerServer ...
vccEnvCheck@te49: Error: failed to send PING command to inducerServer:

 ---------------- Error Structure ----------------
Time Stamp     : 96-06-10  14:47:14.101329
Process Number : 66       Process Name : msgSend
Environment    : wtest    StackId  : 2094      Sequence : 2
Error Number   : 24       Severity : W
Module         : msg      Location : msgSetFilter.c
Error Text     : ccsERR_ENV_NOT_ACTIVE : Environment ltest not active

child process exited abnormally
vccEnvCheck: FAILED.
```

- **Start the environment:**

  Adjust the timeout scales in the LCU Options according to the estimated values (the default settings will usually be sufficient for standard database and module configuration). Then press the ***Start*** action-button. After a few seconds a pop-up window appears that displays the output during the reboot of the LCU.

```
Executing: vccEnvStart -e ltest -t LCU -h te49 -d $VLTDATA/ENVIRONMENTS/ltest \
 -l ‘te41’ -w ‘wtest’

vccEnvStart@te49: Warning: VccInfo: no valid result from query: GetByEnv wtest host-
Name
vccEnvStart: reboot LCU te41 with VxWorks only ...
vccEnvStart: install bootScript in boot-line on LCU te41 ...
vccEnvStart: store boot output in “/vlt/data/ENVIRONMENTS/ltest/.reboot.log” ...
vccEnvStart: execute bootScript on LCU te41 (timeout: 300 s) ...
```

- **Check the environment after start:**

  Press the ***Check*** action-button a second time. Now all checks should be OK.

```
Executing: vccEnvCheck -e ltest -t LCU -h te49 -d $VLTDATA/ENVIRONMENTS/ltest \
 -l ‘te41’ -w ‘wtest’

vccEnvCheck@te49: Warning: VccInfo: no valid result from query: GetByEnv wtest host-
Name
vccEnvCheck: locate environment directory “te49:/vlt/data/ENVIRONMENTS/ltest” ...
vccEnvCheck: locate ltest in /etc/services ...
vccEnvCheck: locate wtest in /etc/services ...
vccEnvCheck: locate te49 in /etc/hosts ...
vccEnvCheck: locate te41 in /etc/hosts ...
vccEnvCheck: locate te41 in /etc/exports ...
vccEnvCheck: locate ltest in /vlt/data/config/logLCU.config ...
vccEnvCheck: locate wtest in /vlt/data/config/logLCU.config ...
vccEnvCheck: send PING command to inducerServer ...
vccEnvCheck: send PING command to lccServer ...
vccEnvCheck: send PING command to msgServer ...
vccEnvCheck: send PING command to rdbServer ...
```

- **Stop the environment:**

  Press the ***Stop*** action-button. After that the LCU will no longer be accessible via message system.

```
Executing: vccEnvStop -e ltest -t LCU -h te49 -d $VLTDATA/ENVIRONMENTS/ltest \
 -l ‘te41’ -w ‘wtest’

vccEnvStop@te49: Warning: VccInfo: no valid result from query: GetByEnv wtest hostName
vccEnvStop: uninstall bootScript and reboot te41 with VxWorks only ...
```

- **Delete the environment:**

  Press the ***Delete*** action-button and confirm the dialog.

```
Executing: vccEnvDelete -e ltest -t LCU -h te49 -d $VLTDATA/ENVIRONMENTS/ltest \
 -l ‘te41’ -w ‘wtest’ -m ‘maximum’ -j

vccEnvDelete: Warning: VccInfo: no valid result from query: GetByEnv wtest hostName
vccEnvDelete: locate environment directory “te49:/vlt/data/ENVIRONMENTS/ltest” ...
vccEnvDelete: unprotect environment files under “te49:/vlt/data/ENVIRONMENTS/ltest” ..
vccEnvDelete: delete environment files under “te49:/vlt/data/ENVIRONMENTS/ltest” ...
```

3.7 To Create an Environment Off-Line

In special cases it is necessary to create the enviornment files on a different (off-line) machine, and move the files later to their final target. The *vccEnvCreate* program supports that by means of the ‘-d’ option. The operation is only possible from the command line, not from the *vccEnv* panel. In this example the enviornment *lt0hb* is actually assigned to *wt0tcs*, but created on *te67*:

```
> vccEnvCreate -e lt0hb -d te67:
Warning: Files are created on te67, but only operational when moved to wt0tcs!
vccEnvCreate: copy standard template to “te67:/vlt/data/ENVIRONMENTS/lt0hb” ...
vccEnvCreate: locate environment directory “te67:/diskb/vlt/data/ENVIRONMENTS/lt0hb”
vccEnvCreate: generate bootScript using configuration-set: minimum ...
vccEnvCreate: generate bootChange sequence scripts ...
vccEnvCreate: make database from “te67:/diskb/vlt/data/ENVIRONMENTS/lt0hb/dbl” ...
```

Similar off-line operation is possible with *vccEnvDelete*, but not with the other *vccEnv** programs.

4 ENVIRONMENT SETUP - vccEnv

This chapter describes version 2.7 of the vcc module.

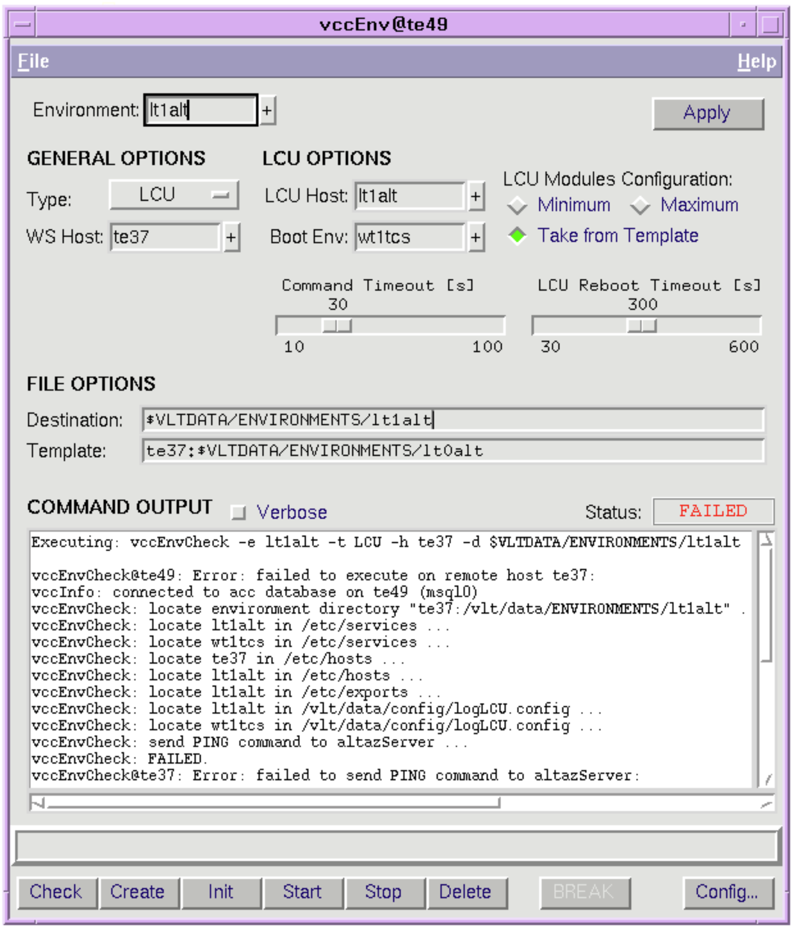
4.1 Overview

The ***vccEnv*** panel allows to perform common maintenance operations for a specific environment. All environments are handled in a uniform way, regardless of which type it is (Rtap, Qsemu, LCU). Moreover, it is automatically resolved on which host the environment is located, and correspondingly remote operation is selected.

4.2 Starting from the Command Line

```
% vccEnv &
```

The program then connects to the VCCDB and the following panel pops up:



Environment: lt1alt

Type: LCU

WS Host: te37

LCU Host: lt1alt

Boot Env: wt1tcs

LCU Modules Configuration: Minimum Maximum [✓ Take from Template]

Command Timeout [s] 30 (10–100)

LCU Reboot Timeout [s] 300 (30–600)

Destination: $VLTDATA/ENVIRONMENTS/lt1alt

Template: te37:$VLTDATA/ENVIRONMENTS/lt0alt

Status: FAILED

```
Executing: vccEnvCheck -e lt1alt -t LCU -h te37 -d $VLTDATA/ENVIRONMENTS/lt1alt

vccEnvCheck@te49: Error: failed to execute on remote host te37:
vccInfo: connected to acc database on te49 (msql0)
vccEnvCheck: locate environment directory "te37:/vlt/data/ENVIRONMENTS/lt1alt" .
vccEnvCheck: locate lt1alt in /etc/services ...
vccEnvCheck: locate wt1tcs in /etc/services ...
vccEnvCheck: locate te37 in /etc/hosts ...
vccEnvCheck: locate lt1alt in /etc/hosts ...
vccEnvCheck: locate lt1alt in /etc/exports ...
vccEnvCheck: locate lt1alt in /vlt/data/config/logLCU.config ...
vccEnvCheck: locate wt1tcs in /vlt/data/config/logLCU.config ...
vccEnvCheck: send PING command to altazServer ...
vccEnvCheck: FAILED.
vccEnvCheck@te37: Error: failed to send PING command to altazServer:
```

4.3 Panel Description

1. Choose an environment with the ***Environment*** selector; either select from the menu, or type the name in (with completion function) and press *<Return>* or the ***Apply*** button at the right.
2. Adjust the ***General Options*** and ***LCU Options*** if the selected defaults (partly taken from the database) do not fit. This, however, will rarely be necessary. Note also that some combinations, although they can be selected, can lead to an inconsistency with the information in the VCCDB, that is detected when an action is started.
3. For LCUs two timeouts are of interest in connection with the *Init*, *Start* and *Stop* actions: one for each shell command that is sent, and one to complete a reboot. Both can be adjusted with the two scales, in case that errors occur due to insufficient timeout values.
4. The ***File Options*** provide a way to specify another directory where the environment files are or shall be located (***Destination***), and which user-provided template directory to use (***Template***) during the *Create* action. All files in this template will override the defaults. The template directory can be located on another host, in which case the hostname and a colon must be given as prefix. See the man-page of *vccEnvCreate* for more, in particualar concerning *dbl* files and LCU *bootScript* handling.
5. Press an action-button to perform the operation. The result (*OK* or *FAILED*) is displayed in the **Status** output after completion; during execution *RUNNING* is displayed instead.

4.4 Actions

All actions can be applied to any type of environment.

- ***Check*** - Check existence and operation of selected environment
- ***Create*** - Create a new environment with selected name and options
- ***Init*** - Initialize selected environment (RTAP snapshots, LCU boot-line etc.)
- ***Start*** - Start selected environment with current configuration
- ***Stop*** - Stop selected environment so that it is not reachable
- ***Delete*** - Delete selected environment, all files will be removed

The name of the actions match the dedicated programs *vccEnv<Action>*, that are described in full detail in the reference chapter 8.

Since ***Init*** and ***Delete*** will overwrite or remove data, a pop-up dialog is presented for confirmation.

The Create actions makes use of the envsCreate program, see reference chapter 8.

Two additional action buttons are available:

- ***BREAK*** - Eventually break a currently running command
- ***Config*** - Configure selected environment with a dedicated tool, depending on its type:
  - LCU: vccConfigLcu, see chapter 5 in this document
  - RTAP: Rtap's own configuration panel

5 CONFIGURATION OF LCU ENVIRONMENTS - vccConfigLcu

This chapter describes version 2.7 of the vcc module.

vccConfigLcu@te49

5.1 Overview

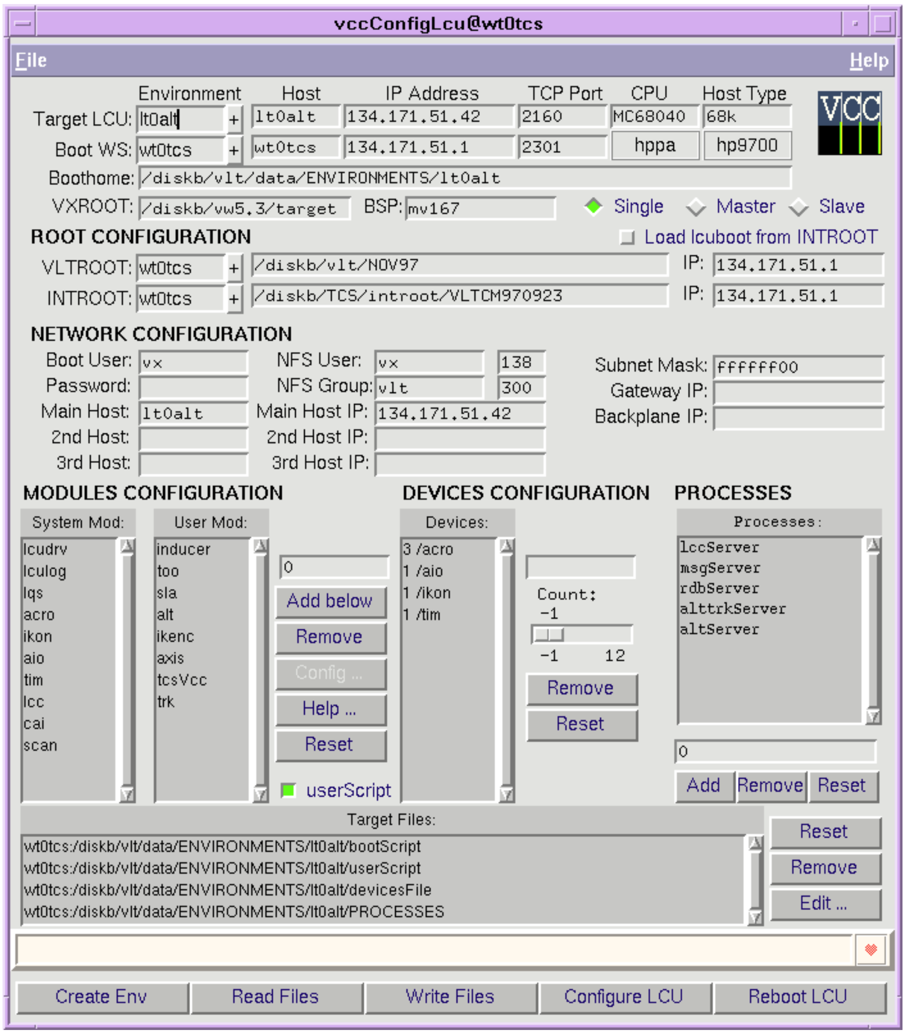
This ***vccConfigLcu*** panel allows to generate and configure the most important files needed for an LCU environment, in particular:

- the boot-script executed by the LCU upon start-up that is suitable for the LCU module *lcuboot*, see chapter 7.
- the processes-file used by engineering interfaces (ccsei and lccei)

5.2 Starting from the Command Line

```
% vccConfigLcu [<environment-name>] &
```

If an explicit LCU environment is given then the configuration data is immediately read from the corresponding target files (that must have been previously created), otherwise the environment can be selected interactively.



	Environment	Host	IP Address	TCP Port	CPU	Host Type
Target LCU:	lt0alt	lt0alt	134.171.51.42	2160	MC68040	68k
Boot WS:	wt0tcs	wt0tcs	134.171.51.1	2301	hppa	hp9700

Boothome: /diskb/vlt/data/ENVIRONMENTS/lt0alt

VXROOT: /diskb/vw5.3/target BSP: mv167 [✓ Single] Master Slave

ROOT CONFIGURATION

☐ Load lcuboot from INTROOT

VLTROOT: wt0tcs /diskb/vlt/NOV97 IP: 134.171.51.1

INTROOT: wt0tcs /diskb/TCS/introot/VLTCM970923 IP: 134.171.51.1

NETWORK CONFIGURATION

Boot User:	vx	NFS User:	vx 138	Subnet Mask:	ffffff00
Password:		NFS Group:	vlt 300	Gateway IP:	
Main Host:	lt0alt	Main Host IP:	134.171.51.42	Backplane IP:	
2nd Host:		2nd Host IP:			
3rd Host:		3rd Host IP:			

MODULES CONFIGURATION

System Mod: lcudrv, lculog, lqs, acro, ikon, aio, tim, lcc, cai, scan

User Mod: inducer, too, sla, alt, ikenc, axis, tcsVcc, trk

0

[✓ userScript]

DEVICES CONFIGURATION

Devices: 3 /acro, 1 /aio, 1 /ikon, 1 /tim

Count: -1

-1 12

PROCESSES

Processes: lccServer, msgServer, rdbServer, alttrkServer, altServer

0

Target Files:

```
wt0tcs:/diskb/vlt/data/ENVIRONMENTS/lt0alt/bootScript
wt0tcs:/diskb/vlt/data/ENVIRONMENTS/lt0alt/userScript
wt0tcs:/diskb/vlt/data/ENVIRONMENTS/lt0alt/devicesFile
wt0tcs:/diskb/vlt/data/ENVIRONMENTS/lt0alt/PROCESSES
```

The following section presents quick instructions how to operate the panel. A more detailed description of the usage of each corresponding part in the panel is given later in section 5.4.

5.3 Example of Usage

This describes how to modify a LCU Environment with vccConfigLcu.

1. Start the vccConfigLcu panel and select your LCU from the **'Target LCU'** selector.
2. Press the **'Read Files'** action button to parse the contents of existing target files.
3. If you want to remove some of the User Modules:
   a. select the module to be removed in the **'User Mod'** listbox
   b. press the **'Remove'** button on the right
   c. repeat this for each module
   d. eventually also adjust contents of the **'PROCESSES'** listbox

   Examples:
   - if you don't have motors, remove: *mac4, vme4sa, mot, motci*
   - if you have a TIM board and don't need NTP, remove: *xntpd*
4. If you don't want automatic device installation, but explicit installation checks:
   a. select each device in the **'Devices'** listbox
   b. set the **'Count'** scale to the expected number of devices
5. After your modifications, press the **'Write Files'** action button to generate the files listed in the **'Target Files'** box, and confirm overwriting for each file.
6. Press the **'Configure LCU'** action button.

   This will re-program the boot-line of the LCU. You can see the values in the terminal window.

   Note that no other session must lock the LCU's shell!
7. Press the **Reboot LCU** action button.

   This will reboot the LCU using the new bootScript. You can see the output during the reboot in the terminal window. A temporary pop-up window lets you adjust the connection time.

   Note that no other session must lock the LCU's shell! See also section 5.5 in case of problems.

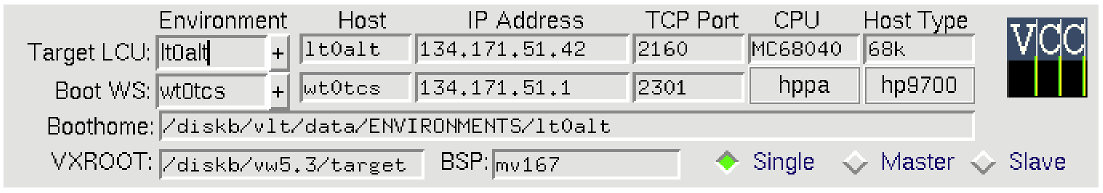
Near the end, "LCC INITIALISATION SUCCESSFULL" should be listed. A possible start-up error message about *timeOpenDevice* can be ignored if no TIM board is installed.

The bootScript will be aborted if anything goes wrong.

After the selected time the connection is closed. You cannot rlogin to the LCU's shell before that, unless you press the **'Cancel Connection'** button in the pop-up dialog.

5.4 Panel Description

5.4.1 Environments Selection



	Environment	Host	IP Address	TCP Port	CPU	Host Type
Target LCU:	lt0alt	lt0alt	134.171.51.42	2160	MC68040	68k
Boot WS:	wt0tcs	wt0tcs	134.171.51.1	2301	hppa	hp9700

Boothome: /diskb/vlt/data/ENVIRONMENTS/lt0alt

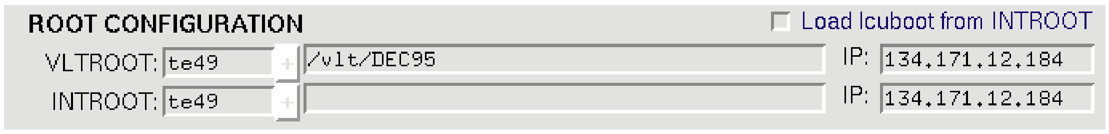
VXROOT: /diskb/vw5.3/target BSP: mv167 [✓ Single] Master Slave

The first action must always be the selection of the ***Target LCU Environment*** that shall be modified. This is done by selection from the menu at the top left in the panel. The derived values like node-name, IP-address and associated boot-host are the queried from the configuration database, provided it is used. Otherwise these values have to be typed in. The ***Boot WS Environment*** can be arbitrarily selected. Remote file operations are performed if the ***Boot WS Host*** is not identical to the workstation where this panel is running on. Note that these remote accesses are much slower than local accesses. ***Boothome*** specifies the home-directory of the LCU environment, and ***VXROOT*** the VxWorks root-directory.

The ***BSP*** value assigns the LCU environment to a specific board support package. Several of such BSP directories can exist under *$VXROOT/config*, which contain VxWorks kernels that are specifically configured for a certain (set of) environment(s). The selection for a ***Single***, ***Master***, or ***Slave*** CPU is indirectly represented by the BSP name, whereby multi-CPU systems are supported; see the man-page for details.

Note that the address parameters in the first line represent the LCU *booting* interface, which may not be the same as the CPU ethernet port. For slave CPUs this is the *backplane* interface address.

5.4.2 Root Configuration

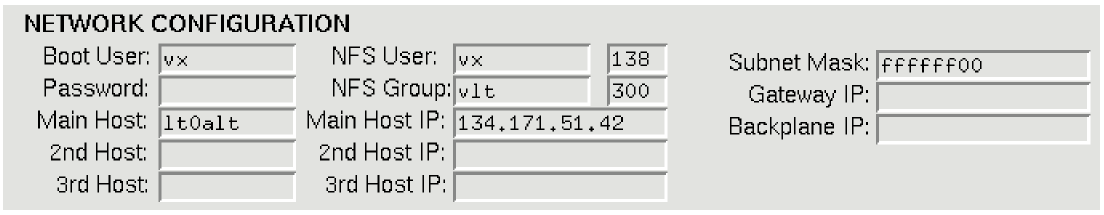


ROOT CONFIGURATION ☐ Load lcuboot from INTROOT

VLTROOT:	te49	/vlt/DEC95	IP:	134.171.12.184
INTROOT:	te49		IP:	134.171.12.184

The variables ***VLTROOT***, and optionally ***INTROOT*** must exist in the LCU environment. The contents of the entries is by default set to the same values of the corresponding shell variables on the selected "Boot Host". They may be edited by the user. Note that only ***physical*** path names without links must be given, as seen from the LCU. Separate NFS mounts will be done to VLTROOT and INTROOT, so that each can be located on a different host as the selected “Boot Host”. The checkbutton allows to load the ***lcuboot*** module (which is always implicitly loaded first) from INTROOT rather than VLTROOT, which is only useful during development on *lcuboot*.

5.4.3 Network Configuration



NETWORK CONFIGURATION

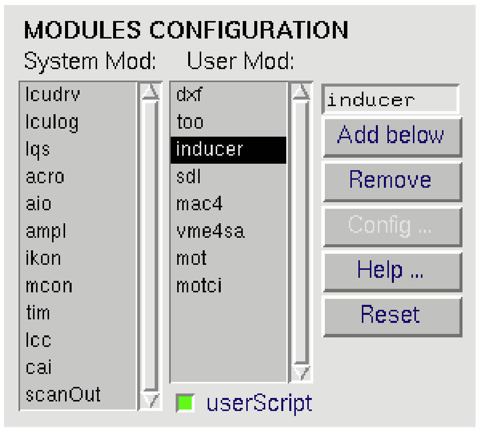
Boot User:	vx	NFS User:	vx	138	Subnet Mask:	ffffff00
Password:		NFS Group:	vlt	300	Gateway IP:	
Main Host:	lt0alt	Main Host IP:	134.171.51.42		Backplane IP:	
2nd Host:		2nd Host IP:				
3rd Host:		3rd Host IP:				

The defaults in these entries will fit in most cases. The ***Boot User*** specifies under which user-name the LCU is accessing the boot-script. The ***Password*** is usually empty, so that remote shell access is used; otherwise FTP is applied. The ***Gateway*** allows to specify an IP-address as gateway to the boot-host. The ***Subnet-Mask*** must be set to the network's value. The ***NFS User-ID*** and ***Group-ID*** is mandatory and must be set corresponding to a valid user on the boot-host. By default the user vx's IDs are taken.

In case of multi-network systems, the entries for ***Main/2nd/3rd Host*** become important. These determine the IP addresses of the CPU main ethernet port and up to two optional additional ethernet interface baords. For single and master CPUs the Main Host is always identical to the address entered in the *ENVIRONMENTS SELECTION*, while for slave CPUs booting over the backplane it is different.

For details about multi-network configuration see the man-page.

5.4.4 Modules Configuration



MODULES CONFIGURATION

System Mod:	User Mod:
lcudrv	dxf
lculog	too
lqs	inducer
acro	sdl
aio	mac4
ampl	vme4sa
ikon	mot
mcon	motci
tim	
lcc	
cai	
scanOut	

inducer

Add below

Remove

Config ...

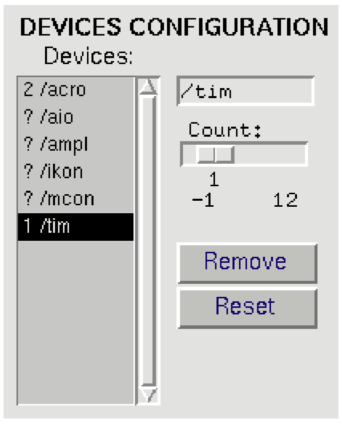
Help ...

Reset

🟩 userScript

The names of modules to be loaded must be specified in the two boxes. Modules in the ***System*** box are normally always needed and their order should not be changed (unless you know about possible side effects). Modules in the ***User*** box usually require all the system-modules and are not always needed, depending on the LCU hardware etc. Individual changes to the lists can be done by selecting an entry and using the ***Add below*** and ***Remove*** buttons on the right of the boxes. For each module listed, there must exist a module-boot-script named *module*.boot in one of the directories in the module-boot-script search-path, see 7.2.2. The ***Help*** button shows the man-page for a selected module-boot-script, provided it is available. The ***Reset*** button fills the two boxes with all modules for which module-boot-scripts are available. The ***userScript*** button enables the processing of the user-script.

5.4.5 Devices Configuration



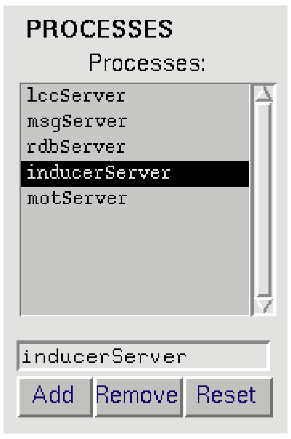
```
2 /acro
? /aio
? /ampl
? /ikon
? /mcon
1 /tim
```

/tim

1
-1 12

All hardware-devices on the LCU can be installed automatically, using the facilities of *lcuboot*. The devices configuration therefore consists only in checking that the required number of devices have actually been installed by that procedure. An explicit number can be given, or the special value **?** resp. **-1**, meaning that no explicit check shall be done ("Plug & Play"). This is useful when the board configuration in a LCU often changes. No modification in the boot-script would be necessary then. The ***Remove*** button completely removes a selected device name from the list. The ***Reset*** button returns to the default. Add new entries by editing the device name and moving the scale.

5.4.6 Processes



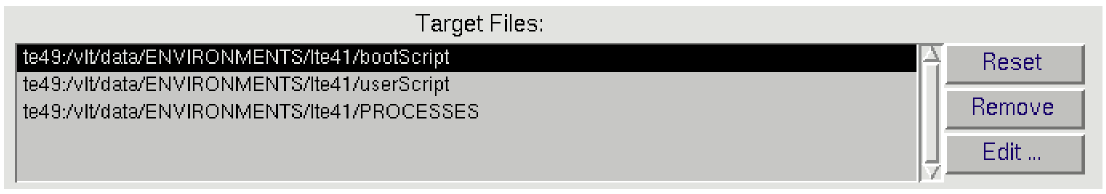
```
lccServer
msgServer
rdbServer
inducerServer
motServer
```

inducerServer

This box contains the names of the LCU processes that shall be accessible by commands via the message-system. The list is normally only used by engineering interfaces (lccei, ccsei) to provide user-menus, but not to restrict any access. The generated processes-file will be written to the *ENVIRONMENTS* sub-directory of *VLTDATA*.

The list can be modified by the ***Add*** and ***Remove*** buttons. The ***Reset*** button returns to the default.

5.4.7 Target Files

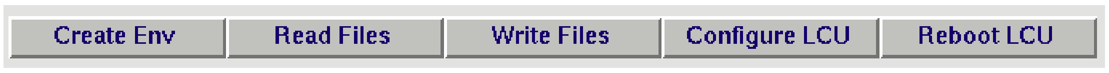


```
te49:/vlt/data/ENVIRONMENTS/lte41/bootScript
te49:/vlt/data/ENVIRONMENTS/lte41/userScript
te49:/vlt/data/ENVIRONMENTS/lte41/PROCESSES
```

This box contains the names of files that shall be processed when the action-button for ***Read Files*** or ***Write Files*** is pressed. A specific target can be selected and removed with the ***Remove*** button, so that it is not processed anymore (but not deleted, of course). An editor can be called with the selected file by the Edit button. The ***Reset*** button returns to the default.

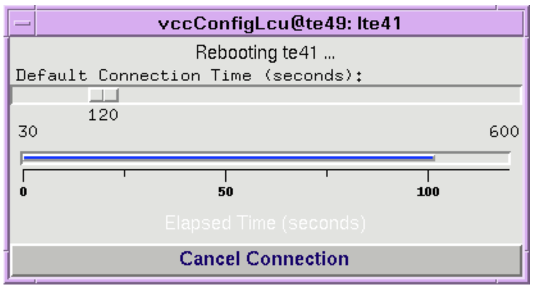
(Remark: Although the *userScript* and *devicesFile* are listed, they will not be altered by the tool.)

5.4.8 Actions



One of these action buttons must be pressed in order to perform a specific operation. All buttons are disabled until an LCU environment is selected.

- "**Create Env**" creates a new set of files under *$VLTDATA/ENVIRONMENTS* on the selected boot-host. The operation fails if one of the files to be created already exist.
- "**Read Files**" reads the files listed in the "Target Files" box and parses their contents, so that the sections in the panel are updated accordingly. Note that the behaviour for reading files not produced with this tool is undefined!
- "**Write Files**" writes the files listed in the "Target Files" box according to selections in the panel's sections. A confirmation is requested before overwriting any existing files.
- "**Configure LCU**" write the boot-line according to the selections in the panel, i.e. host-names, IP-addresses etc. See section 5.5 for potential failures.
- "**Reboot LCU**" restarts the LCU and executes the selected boot-script. The output during start-up is redirected to the terminal where the panel was started from. During booting, a temporary pop-up window allows to specify the default connection time, which must be at least the actually used time to complete the boot. The connection can be aborted explicitly with the ***Cancel Connection*** button.


vccConfigLcu@te49: lte41

Rebooting te41 ...

Default Connection Time (seconds): 120

5.5 Known Problems

The actions **'Configure LCU'** and **'Reboot LCU'** require a *rlogin* to the target LCU. The panel might be blocked when this remote login fails, e.g. when another session is active on the same LCU. To recover from this situation, the user must manually "kill" all active rlogin processes to this LCU node (e.g. with: "psg <LCU-name>" to find out the PID, then "kill <PID>" to stop the process).

Also 'crashed' LCUs - where the *rlogin* daemon is not working anymore - can cause the panel to block.

The rlogin done by **'Reboot LCU'** is terminated after a fixed time (by default a few minutes) and not synchronized with the boot-script execution, i.e. it cannot be detected when the boot process terminates to cancel the connection automatically. The shell variable **VLT_VCCBOOTTIME** specifies the time in seconds after that the connection to the LCU during booting is closed.

The LCU understands only **physical path-names** without links. Although the panel should automatically resolve links, the user should explicitly check this, particularly when paths are entered manually.

It is impossible to recover an LCU node after a **‘Configure LCU’** and **‘Reboot LCU’** sequence when wrong configuration data were entered. A serial terminal must be used as LCU console to re-adjust the boot-line.

The device installation checks are for some situations not rigid enough, so that the boot-script can run into a secondary error rather than stopping at the expected point. In case of troubles, the device’s jumper settings should be checked carefully.

6 CONFIGURATION ACCESS FOR PROGRAMMERS

6.1 Panel Mega-Wigets

TBD

6.2 Programmatic Access from Seqencer-Scripts

TBD

7 THE LCU BOOT ENABLER - lcuboot

✗This chapter presents configuration instructions for LCU modules and drivers

7.1 Purpose of lcuboot

The “LCU boot enabler” shall make the creation and maintenance of LCU boot-scripts easier. The main features are:

- Common module-specific sub-boot-scripts (Module-Boot-Scripts)
- Automatic driver and device installation
- 90% automatic LCU environment variables setup
- Implementation of a search-paths for module loading and configuration

In LCU systems applying this scheme, **lcuboot** is always the first module that is loaded during system start-up. After that several functions are available that facilitate the loading and installation of all further modules.

These functions in principle support the automatic installation of arbitrary LCU modules, but especially of drivers and devices. All functions are intended to be used directly from the VxWorks shell in a boot-script.

The script is aborted when a fatal error condition occurs, which is signalled as a log message in the form:

```
<tid> (tShell): lcuboot: <message>: <faulty item>:<errno | faulty item>
<tid> (tShell): --- SCRIPT ABORTED ---
```

Most of the functions in lcuboot are not reentrant and should therefore only be used from LCU boot-scripts, where reentrancy is not important!

The installed version of lcuboot can be printed with **lcubootVersion** from the VxWorks shell.

The functions available with *lcuboot* are briefly described in the next sections.

7.2 Available lcuboot Functions

7.2.1 lcuboot Automatic Environment Setup

- **lcubootAutoEnvInit** - initializes the standard LCC shell environment variables from the LCU boot-line settings and sets the VxWorks shell-prompt. VLTROOT and optionally INTROOT must be set before calling it. This makes the explicit definition of many variables in the boot-script redundant. It also assigns search-path variables for prioritized access of binary modules (*BINPATH*), module-boot-scripts (*BOOTPATH*), and configuration files (*CONFIGPATH*). Functions are provided to access files based on these search-paths, see section 7.2.5.

See the man-page of **lcubootAutoEnv** in section 8.2 for details and examples.

7.2.2 lcuboot General Module Support

- **lcubootAutoLoad** - auto-loads a binary module and error definition files from *MODROOT*, *INTROOT* (with priority) or *VLTROOT*. This can be used to conditionally load a module. By default a module is not loaded again when it already exists in the LCU memory.

  Do not use the VxWorks function *ld* to load a VLT module, since this will not handle the module's error definition files under *xxxROOT/ERRORS*.
- **lcubootAutoExec** - executes a function depending on a given condition. This can be used to conditionally / optionally install a module.
- **lcubootAutoSpawn** - spawns a function as a task depending on a given condition. This can be used to conditionally / optionally spawn a task.
- **lcubootAutoCd** - changes to the directory under *INTROOT* (with priority) or *VLTROOT* in which a file given by a relative pathname is found first.
- **lcubootAutoCdBoot** - dto., but specifically intended for module-boot-scripts. Only the module-name is needed, the search-path is fixed (see man-page).
- **lcubootAutoProbe** - tests whether any hardware is present at the given addresses. This can be used to determine the number of boards / devices that are present in the system.

See the man-page of **lcubootAutoGen** in section 8.2 for details and examples.

7.2.3 lcuboot Driver and Device Installation

- **lcubootAutoDevRegister** - checks for the presence of a device and registers it for later creation and registration in LCC. Subsequent calls of *lcubootAutoDrvInstall* must be used to install the driver and *lcubootAutoDevCreate* to create the devices automatically depending on their presence.
- **lcubootAutoDrvInstall** - conditionally loads and installs a driver from *INTROOT* (with priority) or *VLTROOT* and prepares for subsequent calls of *lcubootAutoDevCreate*.
- **lcubootAutoDevCreate** - performs the automatic installation of a device. Note that there must be as many calls of this function as the maximum number of devices that shall be supported.
- **lcubootAutoDevCheck** - checks whether the number of devices installed in the system are equal to a given value. This can be used to verify that the expected number of devices are actually installed; however, it works against the idea of automatic device installation.

See the man-page of **lcubootAutoDrv** in section 8.2 for details and examples.

7.2.4 lcuboot LCC Support

- **lcubootAutoLccRegisterDevs** - register all devices in LCC that have been previously announced with *lcubootAutoDevRegister*. This function is usually called during installation of LCC.

See the man-page of **lcubootAutoLcc** in section 8.2 for details and examples.

7.2.5 lcuboot File Access Support

- **lcubootFileOpen** - opens a file using a given search-path. This allows for instance to access configuration files at run-time, using the environment variable *CONFIGPATH* as search-path.

See the man-page of **lcubootFile** in section 8.2 for details and examples.

7.3 How to create LCU Module-Boot-Scripts

✗A “<module>.boot” file is mandatory for every LCU module!

As usual, there will be a main boot-script named ***bootScript*** that the LCU processes initially. From that, Module-Boot-Scripts named ***<module>.boot*** are invoked to load and initialize each module. These shall make use of the *lcuboot* functions, and apply the following guidelines:

- For each LCU module there **must be** one module-boot-script named ***<module>.boot*** that can be called from the main boot-script. It shall be installed during the *make* procedure of the module by specifying it under the *SCRIPTS* directive in the *Makefile*.
- For each LCU module there should be a man-page named ***<module>.boot(5)*** in the ‘File Formats’ section that explains the contents of the module-boot-script.
- For each LCU **driver** module there should be a man-page named ***<module>(4)*** in the ‘Devices’ section that explains the respective board-hardware (addresses, jumpers etc.).

See the man-pages of **lcuboot(5)** and **bootScript(5)** in the reference section 8 for details about the file and directory organization.

The following sections will present some more details about the recommended structure of module-boot-scripts, first for general LCU modules, then for drivers in particular.

7.3.1 Module-Boot-Scripts for General Module Installation

This section describes the recommended structure of standard module-boot-scripts for general modules. Module-Boot-Scripts are called from the main ***bootScript*** of the LCU when a module shall be initialized.

There are the following guidelines:

- The module-boot-script shall load the code of the module as well as all other modules that are needed for operation using the *lcubootAutoLoad* function, which will also take care of the error definition files. For user-application modules LCC can be expected to be already initialized, but driver boot-scripts should not expect that any other module they depend on has already been loaded.
- Functions needed to initialize the module shall be executed using the *lcubootAutoExec* function.
- Tasks needed for the module shall be spawned using the *lcubootAutoSpawn* function.

Some examples for module-boot-scripts:

1. Typical simple module installation where only loading is necessary:

```
lcubootAutoLoad 1,"too"
```

2. Typical more complex module installation:

```
lcubootAutoLoad 1,"lqs"
lcubootAutoExec 1,"lqsInit",10,"rtap"
lcubootAutoExec 1,"lqsAddEnvTbl","wte49","te49","134.171.12.184",2301
... (more) ...
lcubootAutoSpawn 1, "tLqs", 90, 0x18, 20000, "lqs", 3
```

3. Typical driver installation (see section 7.3.2 for more):

```
acroN = lcubootAutoDevRegister("/acro0",0x1000,1,1,1)
... (more) ...
lcubootAutoDrvInstall "acro"
lcubootAutoDevCreate "/acro0",0x1000,112,1
... (more) ...
lcubootAutoSpawn acroN,"tintACRO",200,0x18,2000,"acroInt"
```

See the man-page of **Module.boot(5)** for full details and examples. It is available on-line.

7.3.2 Module-Boot-Scripts for Automatic Driver and Device Installation

This section describes the recommended structure of standard module-boot-scripts for automatic loading and installation of the hardware drivers and devices. They provide a “Plug & Play” feature, so that hardware will be automatically recognized and appropriately installed.

The installation is done in four steps:

1. **Probing for existence of devices:**

   Driver and devices will only be installed when corresponding hardware is found in the VME system. The probing is done with a sequence of calls as follows:

   ```
   <drv>N = lcubootAutoDevRegister("<devX>",<AM>,<addrX>,<args>)

        <drv>   - name of driver, e.g. “acro”
        <devX>  - name of Xth device, e.g. “/acro0”
        <AM>    - VMEbus address modifier code for A16/A24/A32 space
        <addrX> - VMEbus address for Xth device
        <args>  - 3 integer arguments
   ```

   Possible values for the VMEbus address-modifiers AM are for example:

   - Short I/O (A16): 0x2d
   - Standard (A24): 0x3d
   - Extended (A32): 0x0d

   See the VxWorks header “$VXROOT/h/vme.h” for all available values, but in this context only the above stated *Supervisor/Data* modifier codes are of interest.

   The three integer arguments are lateron passed to *lccRegisterDevice*.

2. **Loading and installing of driver code:**

   This will be skipped by the function if there was no hardware found in the first step:

```
        lcubootAutoDrvInstall "<drv>"
```

3. **Installation of devices:**

   For each device to be created, there must be one line as shown below. Where no hardware was found in step 1, the creation will be skipped:

   ```
           lcubootAutoDevCreate "<devX>",<bus-addrX>,<params>,...
   ```

   The device creation usually takes VMEbus addresses as <bus-addrX>, which are different from CPU addresses. <params> are specific to each device family.

4. **Spawning of tasks:**

   Some drivers need specific interrupt handling tasks. They can be spawned with:

   ```
     lcubootAutoSpawn <drv>N,"<taskname>",<prio>,<flags>,<stack>,"<func>"
   ```

   All installed devices are automatically registered afterwards in LCC with the function *lcubootAutoLccRegisterDevs*, which is usually called at the end of the script “lcc.boot”.

The automatic installation depends totally on the usage of the VLT standard board address settings, as described in each driver’s man-page in the devices volume, e.g. *acro(4)*.

See the man-page of **Driver.boot(5)** for full details and examples. It is available on-line.

7.4 How to configure LCUs with lcuboot

In order to build a bootScript that is suitable for the lcuboot structure, it is recommendable to use the ***vccConfigLcu*** panel, see section 5. Hence the bootScript is a generated file and should not be edited by hand.

The ***userScript*** will still be processed as usual, its contents remains fully user-defined.

It may be necessary to configure some module-boot-scripts for your site, in particular *lqs* and *xntp*. Configuration can be done on a per-LCU basis, as well as on a per-host basis, where it applies to all LCUs booting from that host.

The following section describe these steps in more detail.

7.4.1 To create a boot-script suitable for lcuboot

It is assumed that all environment files have already been created as described in section 4.

1. Start the *vccConfigLcu* panel, see section 5.
2. Select the LCU environment
3. Press the *Read Files* action button to update the panel from the existing files
4. Modify the panel contents
5. Press the *Write Files* action button to commit the changes

6. Reboot the LCU.

   All devices should be installed automatically, provided that you follow the board setups as described in the man-pages *acro(4)*, *aio(4)*, etc. Check with *moduleShow* and *devs* that all modules and devices are existing as expected.

These steps cover only the most general case. The ***userScript*** will still be sourced, so that it might require some adjustment.

7.4.2 To configure module-boot-scripts for your site

Some module-specific boot-scripts (those with ".boot" extension) might require some site-specific adjustments that are common to all LCUs in it. The most probable candidates for such a customization are "lqs.boot" and "xntpd.boot", where some node-names and IP-addresses must be defined. This is usually common to all LCUs, so that the configuration should be done on a per-host basis.

1. Copy the respective module-boot-script to the host's configuration directory:

   ```
   cd $VLTDATA/config/
   cp $VLTROOT/vw/bin/<cpu>/<module>.boot .
   ```

2. Edit this copy according to your specific needs. It will automatically have priority over the installed defaults under *VLTROOT* and *INTROOT*.

Refer to the on-line man-pages <module>.boot(5) for more information, e.g. **lqs.boot(5)** respectively **xntpd.boot(5)**.

7.4.3 To configure module-boot-scripts for a specific LCU

If you have the case that one LCU needs a different configuration for a certain module than other LCUs, then do the following:

1. Copy the respective module-boot-script to the LCU environment's home-directory:

   ```
   cd $VLTDATA/ENVIRONMENTS/<LCU-Env-Name>
   cp $VLTDATA/config/<module>.boot .
   or: cp $VLTROOT/vw/bin/<cpu>/<module>.boot .
   ```

2. Edit this copy according to your specific needs. It will automatically have priority over the installed defaults under *VLTROOT* and *INTROOT*, as well as the per-host configuration in *VLTDATA/config*, if present.

Refer to the man-pages <module>.boot(5) for more information.

8 REFERENCE

8.1 User Commands Reference

The following sections contain - in alphabetical order - the manual pages for the user commands included in the interface of the modules:

8.1.1 envsCreate(1)

NAME

```
envsCreate - create an environment from its standard template
```

SYNOPSIS

```
envsCreate [-e <env>] [-t <type>] [-d <dest>]
```

DESCRIPTION

```
This command creates an environment directory from scratch.
All files are copied from a template corresponding to <type>.

However, the created files are not configured in any way.
This must be done explicitly with dedicated tools.
```

ARGUMENTS

```
Arguments can either be put on the command-line directly,
or will be asked interactively.

When prompted for input, the default will be shown in square brackets.

<env>   Name of the environment.
        If not given then interactive input is expected.

<type>  Type of the environment: LCU | RTAP | QSEMU
        Default: derived from <env> as follows:
                if      <env> matches "l*"   then LCU
                else if <env> matches "w*qs" then QSEMU
                else if <env> matches "w*"   then RTAP
                else error

<dest>  Directory under which environment files shall be created.
        This must not already exist!
        Default: $VLTDATA/ENVIRONMENTS/<env>
```

FILES

```
$VLTROOT/templates/forEnvs/<type> - source template directories
$VLTDATA/ENVIRONMENTS/<env>       - default target directory
```

ENVIRONMENT

```
VLTROOT - path to VLT constant code area
VLTDATA - path to VLT variable data area
```

EXAMPLES

```
> envsCreate -e wt1tcs

> envsCreate
Enter Environment Name []: wt1tcs
Enter Target Directory Name [/vlt/data/ENVIRONMENTS/wt1tcs]: ./wt1tcs
Enter Type of Environment (LCU,RTAP,QSEMU) [RTAP]:
creating RTAP environment wt1tcs under "./wt1tcs" ...
setting default protection of "./wt1tcs" ...
```

```
- - - - - -
Last change:  02/11/98-14:22
```

8.1.2 envsKill(1)

NAME

```
       envsKill - Kill an environment upon reception of special command
```

SYNOPSIS

```
       envsKill
```

DESCRIPTION

```
       This program has to be started up with every RTAP environment used for
       engineering purposes.
       When envsKill gets the Command KILLENV it will send a mail to the
       owner that started the environment and wait for 4 minutes.
       Within this grace period owner can inhibit the pending shutdown by simply
       killing envsKill
       After the 4 minutes the environment is shut down.

       KILLENV <user name>
               Username has to be a valid user name, which is propagated to
               the owner of the environment.
```

CAUTIONS

```
       Error handling should be improved!
       As the command KILLENV would take about 5 minutes if it was synchronous
       it returns immediately and doesn't wait until the shutdown.
       The killer has to watch completion of the environment shutdown otherwise.
```

```
- - - - - -
Last change:  02/11/98-14:22
```

8.1.3 vccConfigLcu(1)

NAME

```
vccConfigLcu - Configuration Panel for LCU Boot Files
```

SYNOPSIS

```
vccConfigLcu [<environment-name>] &
```

DESCRIPTION

```
This panel allows to generate the most important files needed
for an LCU environment, in particular:

        - the boot-script executed by the LCU upon start-up
        - the processes-file used by engineering interfaces

If an explicit <environment-name> is given at the command-line,
then it is immediately selected and the panel's contents is updated
from the current configuration files, when present.
Otherwise it can be selected interactively.

The following sections describe the usage of each corresponding
section in the panel.
```

MENU BAR

```
File menu:
        "Quit"          - exit the panel

Help menu:
        "Extended Help" - display this man-page
```

ENVIRONMENTS SELECTION

```
The first action must always be the selection of the LCU environment
that shall be modified. This is done by selection from the menu
at the top left in the panel. The derived values like node-name,
IP-address and associated boot-host are queried from the central
configuration database - see VccInfo(n) - provided it is active.
Otherwise these values have to be typed in.

The "Target LCU" parameters refer to the environment as a whole
and to the booting interface of it. In case of slave CPU systems
this may not be obvious; see MULTIPLE CPU SYSTEMS for more.

The "Boot WS" environment can be arbitrarily selected.
Remote file operations are performed if the "Boot Host" is not
identical to the workstation where this panel is running on.
Note that these remote accesses are much slower that local accesses.

"Target LCU"    - values related to the LCU to be configured
"Boot WS"       - values related to the WS the LCUs boots from

"Environment"   - name of the environment, as "lt1alt", "wt1tcs"
"Host"          - host where the environment is located
"IP Address"    - IP address of that host, eg. "11.22.33.44"
"TCP Port"      - assigned TCP port number of the environment
"CPU"           - CPU name, eg. "MC68040"; must match "vw/bin/<cpu>"
"Host Type"     - architecture of the host, eg. "68k" (for info only)

When a new "Target LCU Environment" is selected, the corresponding
```

values will be queried from the configuration database as far as available. If defined, also the "Boot WS" is automatically selected.

Similarly, when a new "Boot WS Environment" is selected, the values are queried, and the available modules are updated as if the "Reset" button in the MODULES CONFIGURATION section was pressed (see there).

```
"Boothome"      - directory where the environment resides
"VXROOT"        - VxWorks target root directory
"BSP"           - VxWorks board-support-package name, eg. "mv167".
                  Values as "mv167-p0" indicate master/slave versions.
                  See MULTIPLE CPU SYSTEMS for special issues.
```

BOOTHOME is derived from VLTDATA, meaning that the -->TARGET FILES will be read from resp. written to that area, and that the LCU will finally boot with the boot-script located there.

The type of system is indicated with the radio-buttons:

```
"Single"        - for a normal system with only one CPU
"Master"        - for the master in the system
"Slave"         - for slave #1 in the system (> 1 not supported)
```

The selection has only indirect effect on the BSP value, which in turn results in the dedicated VxWorks kernel being loaded. See MULTIPLE CPU SYSTEMS for more details.

ROOT CONFIGURATION

The variables BOOTHOME, VLTROOT, and optionally INTROOT must exist in the LCU environment. The contents of the entries is by default set to the same values of the corresponding shell variables on the selected "Boot WS" host. They may be edited by the user.

VLTROOT and INTROOT may reside on a different host than BOOTHOME. Implicit NFS mounts will be done on the LCU as appropriate.

Note that path names must be given as seen from the LCU. This might in some cases be different from the path names on the WS, eg. when there are links in the path. Only the physical path is allowed. Lateron, however, the LCU NFS-mount may use different local names.

```
"VLTROOT" - host name and physical path to the VLTROOT area
"INTROOT" - host name and physical path to the INTROOT area
```

The "IP" field must contain the IP address in dot-notation of the selected host, if it is different from the "Boot WS" host.

When a new host is selected, the available modules are updated as if the "Reset" button in the MODULES CONFIGURATION section was pressed.

NETWORK CONFIGURATION

The defaults in these entries will fit in most cases:

```
"Boot User"     specifies under which user-name the LCU is accessing
                the boot-script. It is usually `vx'.
"Password"      specifies the password of the "Boot User".
                It is usually empty, so that remote shell access is
                used; otherwise FTP is applied.

"NFS User"      specifies under which user ID the LCU is accessing
                files via NFS. It is usually `vx'.
                The first entry with the user name is only for
                convenience. The second entry is the important one,
                and it must contain the corresponding numeric ID.
"NFS Group"     specifies under which group ID the LCU is accessing
                files via NFS. It is usually the group of user `vx'.
                It applies the same as for "NFS User"

"Main Host(IP)" specifies the parameters of the on-board CPU network
                interface; this is usually the same as the booting
                interface specified above, except for slave CPUs
                which are booting over shared memory backplane.
"2nd Host (IP)" specifies the parameters of an optional network
                interface, eg. additional ethernet board.
"3rd Host (IP)" dto. for further interface.

"Subnet-Mask"   must be set to the network's inherent value.
"Gateway IP"    is the gateway IP address to the "Boot WS" host.
                It should be empty when both host are on the same net.
                This value is discarded in MULTIPLE CPU SYSTEMS.
"Backplane IP"  is the IP address assigned to the backplane interface.
                This value is discarded in MULTIPLE CPU SYSTEMS.
```

The user-ID and group-ID for NFS accesses is mandatory and must be set corresponding to a valid user on the boot-host. All valid user and groups are permitted, by default the IDs of user "vx" are taken.

MODULES CONFIGURATION

The names of modules to be loaded must be specified in the two boxes. Modules in the "System" box are normally always needed and their order should not be changed. Modules in the "User" box usually require all the system-modules and are not always needed, depending on the LCU hardware etc.

Individual changes to the lists can be done by selecting an entry and using the buttons on the right of the boxes.

For each module listed, there must exist a module-boot-script named "<module>.boot" in one of the directories listed in -->FILES, which should contain the auto-loading and initialization of the module, using the functions of lcuboot(1).

```
"Add below"     - add the module from the entry below the selection
"Remove"        - remove the selected module
"Config..."     - open panel for individual module configuration
"Help..."       - display help for selected module (if available)
"Reset..."      - fill listboxes with all available modules. This
                  implies that the available module-boot-scripts on
                  the selected hosts (Boot WS, VLTROOT, INTROOT)
                  are scanned to resolve module dependencies.
                  For remote accesses this can take a while.
```

DEVICES CONFIGURATION

All hardware-devices on the LCU can be installed automatically, using the facilities of lcuboot(1). The devices configuration therefore consists only in checking that the required number of devices have actually been installed by that procedure.

An explicit number can be given, or the special value `?' resp. `-1', meaning that no explicit check shall be done. This is useful when the board configuration in a LCU often changes. No modification in the boot-script would be necessary then.

```
"Count"  - the number of expected devices, or `-1' for auto-config
"Remove" - remove the selected device family from the listbox
"Reset"  - reset the listbox to the default
```

PROCESSES

This box contains the names of the LCU processes that shall be accessible by commands via the message-system. The list is normally only used by engineering interfaces (lccei, ccsei) to provide user-menus, but not to restrict any access.

```
"Add"    - add the process from the entry to the listbox
"Remove" - remove the selected process from the listbox
"Reset"  - reset the listbox to the default
```

TARGET FILES

This box contains the names of files that shall be processed when the action-button for "Read Files" or "Write Files" is pressed (see -->ACTIONS).

```
"Reset"   - reset the listbox to the default
"Remove"  - omit the selected target file from any processing
"Edit..." - open an editor for the selected target file
```

ACTIONS

All action buttons are disabled until an LCU environment is selected.

```
"Create Env"    create a new set of LCU environment files under
                VLTDATA on the selected boot-host. The operation
                fails if the environment to be created already exists.

"Read Files"    read the files listed in the "Target Files" box and
                parses their contents, so that the sections in the
                panel are updated accordingly.
                Note that the behaviour for reading files not produced
                with this tool is undefined!

"Write Files"   write the files listed in the "Target Files" box
                according to selections in the panel's sections.
                A confirmation is requested before overwriting any
                existing files.

"Configure LCU" write the boot-line according to the selections in the
                panel, i.e. host-names, IP-addresses etc.;
                it uses the VxWorks `bootChange' command.
                See -->DIAGNOSTICS for potential failures.

"Reboot LCU"    restart the LCU and execute the selected boot-script.
                A transient window appears displaying the elapsed
                boot time. The default waiting time before closing
                the connection can be adjusted with the scale.
                - NOTE THAT THE DEFAULT TIME MUST BE HIGHER THAN THE
                  ACTUAL BOOTING TIME, THERE IS NO SYNCHRONIZATION -
                The "Close Connection" button aborts the connection
                explicitly, eg. when the booting has completed.
                The output during start-up is redirected to the
                terminal where the panel was started from.
                See -->DIAGNOSTICS for potential failures.
```

MULTIPLE NETWORK INTERFACES

An LCU may contain additional "passive" network interface boards. Each of them must be assigned with a hostname and IP address.

In such case, there must be a list of hostnames assigned to this environment in the configuration database, where normally is only one single hostname. The first in the list is always the on-board main interface. The IP addresses are then resolved from the names.

Under the VxWorks shell there will be special variables defined, which can be used by the network driver to setup the interfaces. These are: LOCALHOST_n and LOCALIPADDRESS_n, with n = 1,2.
Up to two additional interfaces are supported.

MULTIPLE CPU SYSTEMS

The special case of multiple CPU systems is discussed here.

The master CPU will be configured to boot over its ethernet port, as for a normal single CPU system. A specilized VxWorks kernel is required in order to support booting of the slaves via the master as gateway, see below.

The slave CPUs will be configured to boot over the backplane; note that the hostname and IP addresses in the ENVIRONMENTS SELECTION have to reflect this! The booting interface, and not the on-board ethernet port, must be specified there.

By VLT convention multi-CPU systems shall use Proxy-ARP.
This implies different VxWorks kernels for master and slave CPUs, where the following kernel options must be set:

```
- master CPU with proxy-server:
  INCLUDE_SM_NET                include backplane net interface
  INCLUDE_PROXY_SERVER          proxy arp server (Master Board)
  INCLUDE_PROXY_DEFAULT_ADDR    ethernet addr to generate bp addrs
  INCLUDE_SM_SEQ_ADDR           SM network auto address setup

- slave CPU with proxy-client:
  INCLUDE_SM_NET                include backplane net interface
  INCLUDE_PROXY_CLIENT          proxy arp client (Slave Board)
  INCLUDE_SM_SEQ_ADDR           SM network auto address setup
```

The VCC configuration distinguishes between master and slave CPUs based on the value of the BSP field. If it has the form:

```
        <basename>-p<procnum>-s<memsize>
```

then <memsize> indicates the CPU's decimal memory-size in MB and <procnum> (0..15) indicates the processor-number, ie. zero for the master and 1..15 for the slaves.

The corresponding VxWorks kernel must then exist under the location derived from the full BSP value, see FILES.

Slave CPUs will be configured to boot over shared memory located in the master with the SM anchor set according to the memory-size, namely: anchor = 2 * memorysize + 0x600.
This requires corresponding and consistent configuration in sysLib.c in the struct sysPhysMemDesc.

Due to the above mentioned kernel options, inet addresses for backplane of master and slaves, as well as gateway of slaves will

```
be determined automatically by incrementing the masters ethernet
inet address. Explicit assignment of those is not supported.

When mv167 boards are booted over backplane, the on-board ethernet
interface is NOT automatically attached (although documented so).
Therefore explicit instructions are written into the bootScript
NETWORK section to do this. Note however that the ethernet i/f
will remain detached until the bootScript is executed.
```

ENVIRONMENT

```
VLTDATA         path to VLT data area
VLTROOT         path to VLT code area
INTROOT         path to private integration code area (optional)
VXROOT          path to VxWorks files

VLT_VCCENV      name of configuration database environment (optional)
VLT_VCCBOOTTIME time in seconds after which reboot session is closed
                (optional, default is 120 seconds)
VLT_VCCTIMEOUT  timeout in seconds for LCU reboot with VxWorks only
                (optional, default is 30 seconds)

Other variables see VccInfo(n).
```

FILES

```
Under VLTDATA the following files may be read and/or written:

ENVIRONMENTS/<env>/bootScript   - boot-script for <env>
ENVIRONMENTS/<env>/userScript   - user-script for <env>
ENVIRONMENTS/<env>/devicesFile  - LCC software-devices for <env>
ENVIRONMENTS/<env>/PROCESSES    - process-file for <env>
ENVIRONMENTS/<env>/*.boot       - module-boot-scripts

Under VLTROOT or INTROOT the following files may be read only:

vw/bin/*.boot                   - module-boot-scripts
vw/bin/<cpuName>/*.boot         - module-boot-scripts

Under VXROOT the following files may be read only:

config/<bsp>/vxWorks            - VxWorks kernel

See also VccInfo(n) and lcuboot(5) for accessed files.
```

DIAGNOSTICS

```
The -->ACTIONS "Configure LCU" and "Reboot LCU" require a `rlogin'
to the target LCU. The panel might be blocked when this remote login
fails, e.g. when another session is active on the same LCU.
To recover from this situation, the user must manually "kill" all
active rlogin processes to this LCU node.
This may also happen when the LCU has "crashed".
```

BUGS

```
The action-button "Reboot LCU" does an `rlogin' to the target-LCU.
This remote-login is terminated after the time set with
VLT_VCCBOOTTIME and is not synchronized with the boot-script execution.
A message appears on stdout when the connection is finally closed.
```

SEE ALSO

```
        vccEnv(1), vccEnvCreate(1),
        bootScript(5), userScript(5), module.boot(5),
        VccInfo(n),
        lcuboot(1)
```

```
- - - - - -
Last change:  02/11/98-13:00
```

8.1.4 vccEnv(1)

NAME

```
vccEnv - common configuration environment programs
```

SYNOPSIS

```
vccEnvCreate -e <env> ...
vccEnvDelete -e <env> ...
vccEnvInit   -e <env> ...
vccEnvStart  -e <env> ...
vccEnvStop   -e <env> ...
```

DESCRIPTION

```
The `vccEnv' program family allows to create, delete, and control
environments in a uniform way, so that the environment type can be
anything of LCU, RTAP, or QSEMU, and will be automatically handled
in the correct way. Also remote operation will be done automatically
depending on the host to which the environment is assigned to.

This is done based on the information in the configuration database,
where for each environment and host all relevant data are stored.

The usual sequence during an environment life-time is:

        Create --> Init --> Start <--> Stop --> Delete
                    ^                   |
                    |___________________|

where:
        Create: create the environment files from scratch
        Init:   initialize and reset the environment
        Start:  put the environment into running state
        Stop:   put the environment into stopped state
        Delete: remove the environment files

All programs will enter an interactive mode when called with no
arguments.

The man-pages of each individual program contain the detailed
steps that are performed, and the supported options.
```

ARGUMENTS

```
The following arguments are supported in principle everywhere,
but some of them will be ignored by several programs.

If no value is given, then the default will be taken from the
configuration database, or queried in some other way.

<env>   Name of environment, eg. lt1hb, wt1tcs, wte1qs
        If not given, then interactive input is expected.

<type>  Type of environment: LCU | RTAP | QSEMU
        default_1: as defined in configuration database
        default_2: derived from <env>

<host>  Name of the WS host where environment files shall reside.
        default_1: as assigned in configuration database
        default_2: local host
```

```
                For LCU environments this can also be a colon-separated
                list of not more than four hosts (all are optional),
                meaning "<boot-host>:<vlt-host>:<int-host>:<mod-host>".
                They refer to the host where VLTROOT, INTROOT, and MODROOT
                reside, and default to <boot-host> if omitted.
                All given hosts are then NFS-mounted in the LCU.

        <dest>  Name of destination directory,
                default: "$VLTDATA/ENVIRONMENTS/<env>" on <host>

        <src>   Name of directory to be used as user-template.
                Must be prefixed with "<host2>:" if not residing on <host>.
                All files and directories under this path will be merged
                with the standard template and have priority.
                At least one file must be in this directory.
                Note that there is a special handling of LCU bootScript files,
                and that dbl files are not modified to match for <env>!
                Default: no user-template

        <lcu-host> For LCU environments the node-name of the LCU host
                can be specified.
                Default: as assigned in configuration database

        <ws-env> For LCU environments the name of the assigned WS boot
                environment can be specified.
                Default: as assigned in configuration database

        <mods>  For LCU environments the user-modules to be loaded can be
                specified by a colon-separated list of module names,
                eg. "inducer:mot". System- and user-modules are automatically
                distinguished.
                The list can also contain the special keywords "minimum"
                or "maximum" to request a standard system modules
                configuration with either a minimal or maximal set of modules.
                The single keyword "template" causes the module-configuration
                to be propagated from the user-template <src>.
                The -m option can also be given more than once.
```

OPTIONS

```
        The following options may be given:

        -i(nteractive)  asks for input
        -j(nteractive)  "pseudo-interactive" on display, but no input required
        -q(uiet)        inhibits output to stdout
        -v(erbose)      outputs more information than normally
```

CAUTIONS

```
        For non-interactive calls stdin should be redirected to /dev/null.
```

RETURN VALUE

```
        exit status 0 (OK) or 1 (failed)
```

ENVIRONMENT

```
        All programs use rcp(1) and remsh(1)/rsh(1) for remote file access.
        It must be made sure that all relevant environment variables are
        defined while the r-command executes on the remote host.
        They should therefore be set in "~/.cshrc" or equivalent rc-files.
```

```
See VccInfo(n) for configuration database related environment.
```

SEE ALSO

```
vccEnvCreate(1), vccEnvDelete(1), vccEnvInit(1),
vccEnvStart(1), vccEnvStop(1),
VccInfo(n),
rcp(1), remsh(1), rsh(1)
```

```
- - - - - -
Last change:  02/11/98-13:00
```

8.1.5 vccEnvCheck(1)

NAME

```
vccEnvCheck - check environment directory and files
```

SYNOPSIS

```
vccEnvCheck -e <env> [-h <host>] [-d <dest>]
```

DESCRIPTION

```
Checks the environment <env> with the following steps:

1. check existence of directory <dest>
2. check essential global configuration files for <env>
   (this is only a rough check, no precise parsing!)
3. send PING command to all essential environment processes, i.e.
   for LCU:            lccServer msgServer rdbServer
   for RTAP and QSEMU:  cmdManager msgServer logManager

Note that an OK result cannot guarantee that the environment is
running properly in all respects.
Only a FAILED result ensures that the environment is NOT OK.
```

ARGUMENTS

```
<env>   Name of environment, eg. lt1hb, wt1tcs, wte1qs
        Mandatory, no default.

<host>  Name of the host where environment files reside,
        default_1: as assigned in configuration database
        default_2: local host

<dest>  Name of destination directory,
        default: "$VLTDATA/ENVIRONMENTS/<env>" on <host>
```

OPTIONS

```
The following option may be given in any combination:

-q(uiet)        inhibits output to stdout
-v(erbose)      outputs more information than normally
```

CAUTIONS

```
The system configuration files are not updated!
This should be done by updating the configuration database, and then
generating the configuration files using vccMake(1).
```

RETURN VALUE

```
exit status 0 (OK) or 1 (failed)
```

FILES

```
Files under the directory given by <dest> are read.
Depending on the type of environment also:

/etc/services
/etc/hosts
/etc/$RTAPROOT/RtapEnvList (Rtap >= 6.7) or $RTAPROOT/etc/RtapEnvList
$VLTDATA/config/logLCU.config
```

```
$VLTDATA/config/CcsEnvList
```

ENVIRONMENT

```
See vccEnv(1).
```

SEE ALSO

```
vccEnv(1), vccMake(1), msgSend(1)
```

```
- - - - - -
Last change:  02/11/98-13:00
```

8.1.6 vccEnvCreate(1)

NAME

```
vccEnvCreate - create environment directory and files
```

SYNOPSIS

```
vccEnvCreate -e <env> [-t <type>] [-h <host>] [-d <dest>] [-s <src>]

vccEnvCreate ... -l <lcu-host> -w <ws-env> -m <mods>:... (for type LCU)

vccEnvCreate -i                                         (interactive)
```

DESCRIPTION

```
Creates the environment directory and files for the given <type>.

All directories and files are created on <host> under the
directory <dest> from the standard template, optionally taking
<src> as additional higher prioritzed user template.

<dest> must not already exist.

Created files are not fully configured, ie. depending on <type>
they may have to be edited or processed with dedicated tools in
order to make them fully usable.

Detailed actions depending on <type> are:

LCU:    1. create files from standard template and set protections
        2. merge with user-defined template from <src>
        3. make database
        4. target-specific bootScript configuration

RTAP:   1. create files from standard template and set protections
        2. merge with user-defined template from <src>
        3. make database

QSEMU:  1. create files from standard template and set protections
        2. merge with user-defined template from <src>
        3. make database
```

ARGUMENTS

```
<env>   Name of environment, eg. lt1hb, wt1tcs, wte1qs
        If not given, then interactive input is expected.

<type>  Type of environment: LCU | RTAP | QSEMU
        default_1: as defined in configuration database
        default_2: derived from <env>

<host>  Name of the host where environment files shall reside,
        default_1: as assigned in configuration database
        default_2: local host

        For LCU environments this can also be a colon-separated
        list of not more than four hosts (all are optional),
        meaning "<boot-host>:<vlt-host>:<int-host>:<mod-host>".
        They refer to the host where VLTROOT, INTROOT, and MODROOT
        reside, and default to <boot-host> if omitted.
        All given hosts are then NFS-mounted in the LCU.
        See ENVIRONMENT for these variables, and CAUTIONS for MODROOT.
```

```
<dest>  Name of destination directory.
        For off-line creation (ie. on another machine than <host>)
        it can be prefixed with "<desthost>:" so that the files are
        created on <desthost> and can later be moved to <host>.
        A warning will be generated in such a case.
        If "<desthost>:" (remark the colon) is given, then the
        path may also be omitted so that the default is taken.
        Default: "$VLTDATA/ENVIRONMENTS/<env>" on <host>

<src>   Name of directory to be used as user-template.
        Must be prefixed with "<host2>:" if not residing on <host>.
        All files and directories under this path will be merged
        with the standard template and have priority.
        At least one file must be in this directory.
        Note that there is a special handling of LCU bootScript files,
        and that dbl files are not modified to match for <env>!
        Default: no user-template

<lcu-host> For LCU environments the node-name of the LCU host
        can be specified.
        Default: as assigned in configuration database

<ws-env> For LCU environments the name of the assigned WS boot
        environment can be specified.
        Default: as assigned in configuration database

<mods>  For LCU environments the user-modules to be loaded can be
        specified by a colon-separated list of module names,
        eg. "inducer:mot". System- and user-modules are automatically
        distinguished.
        The list can also contain the special keywords "minimum"
        or "maximum" to request a standard system modules
        configuration with either a minimal or maximal set of modules.
        The single keyword "template" causes the module-configuration
        to be propagated from the user-template <src>.
        The -m option can also be given more than once.
```

OPTIONS

```
The following option may be given in any combination:

-i(nteractive)  asks for input
-q(uiet)        inhibits output to stdout
-v(erbose)      outputs more information than normally
```

CAUTIONS

```
The system configuration files are not updated for the new environment!
This should be done by updating the configuration database, and then
generating the configuration files using vccMake(1).

MODROOT: it will be problematic to define MODROOT dynamically!
All xxxROOT/DATA environment variables shall be defined during
the shell rc routines (.cshrc etc.) to allow remote operation.
Otherwise they may be left undefined and produce unexpected results.
```

RETURN VALUE

```
exit status 0 (OK) or 1 (failed)
```

FILES

```
Files under the directory given by <src> are read.
Files under the directory given by <dest> are written.

You must have priviledge to write the specified target file(s),
otherwise the operation will fail.

The mode and owner of the generated file will be set to
world-readable and group-writable, unless otherwise noted.

See ENVIRONMENTS_<type>(5) for detailed target-file lists.
```

ENVIRONMENT

```
See vccEnv(1).
```

EXAMPLES

```
Create LCU environment in interactive mode (with error):

> vccEnvCreate
vccInfo: connected to VCC database at wte49
usage:  vccEnvCreate [-iqv] -e <env> [...]
(will use interactive mode now, enter -- to continue non-interactive)

Enter Environment Name []: lte41
Enter Type of lte41 (LCU,RTAP,QSEMU) [LCU]:
Enter LCU Host of lte41 [te41]:
Enter WS Boot Environment of lte41 [wte49]:
Enter WS Host of wte49 [te49]:
Enter Home Directory of lte41 on te49 [$VLTDATA/ENVIRONMENTS/lte41]:
Enter Template Directory for lte41 (host:path) []:
Enter that you are sure to continue (yes,no)? [no]: yes
vccEnvCreate: copy standard template to "/vlt/data/ENVIRONMENTS/lte41" ...
vccEnvCreate@te49: Error: failed to copy standard template to "..."
envsCreate: /vlt/data/ENVIRONMENTS/lte41 already exists


Create LCU environment named `lt0hb' on host `te49',
with NFS mount of VLTROOT on the same host, and INTROOT on te13:

> vccEnvCreate -e lt0hb -h te49::te13 -w wte49
vccInfo: connected to VCC database at wte49
vccEnvCreate: copy standard template to "/vlt/data/ENVIRONMENTS/lt0hb" ...
vccEnvCreate: make database from "/vlt/data/ENVIRONMENTS/lt0hb/dbl" ...
vccEnvCreate: make DB branch world writable ...
vccEnvCreate: generate bootScript using: minimum ...
vccEnvCreate: generate bootChange sequence scripts ...


Create RTAP environment named `wte4910' using `wte49' as template:

> vccEnvCreate -e wte4910 -s /vlt/data/ENVIRONMENTS/wte49
vccInfo: connected to VCC database at wte49
vccEnvCreate: copy standard template to "/vlt/data/ENVIRONMENTS/wte4910"
vccEnvCreate: merge with template from "/vlt/data/ENVIRONMENTS/wte49" ...
vccEnvCreate: make database from "/vlt/data/ENVIRONMENTS/wte4910/dbl" ...
vccEnvCreate: make DB branch world writable ...
```

SEE ALSO

```
vccEnv(1), vccMake(1), envsCreate(1),
ENVIRONMENTS_LCU(5), ENVIRONMENTS_RTAP(5), ENVIRONMENTS_QSEMU(5),
```

TODO

```
- parse the bootScript contained in the user-template and
  propagate its contents to the new one.
```

```
- - - - - -
Last change:  02/11/98-13:00
```

8.1.7 vccEnvDelete(1)

NAME
```
          vccEnvDelete - delete environment directory and files
```

SYNOPSIS
```
          vccEnvDelete -e <env> [-h <host>] [-d <dest>]
```

DESCRIPTION
```
          Deletes the environment directory and files for <env>.
          All files on <host> under the directory <dest> are deleted.

          To be sure that the environment is stopped before, use vccEnvStop(1).
```

ARGUMENTS
```
          <env>   Name of environment, eg. lt1hb, wt1tcs, wte1qs
                  Mandatory, no default.

          <host>  Name of the host where environment files reside,
                  default_1: as assigned in configuration database
                  default_2: local host

          <dest>  Name of destination directory.
                  For off-line deletion (ie. on another machine than <host>)
                  it can be prefixed with "<desthost>:" so that the files are
                  deleted on <desthost> rather than <host>.
                  A warning will be generated in such a case.
                  If "<desthost>:" (remark the colon) is given, then the
                  path may also be omitted so that the default is taken.
                  Default: "$VLTDATA/ENVIRONMENTS/<env>" on <host>
```

OPTIONS
```
          The following option may be given in any combination:

          -q(uiet)        inhibits output to stdout
          -v(erbose)      outputs more information than normally
```

CAUTIONS
```
          The system configuration files are not updated!
          This should be done by updating the configuration database, and then
          generating the configuration files using vccMake(1).
```

RETURN VALUE
```
          exit status 0 (OK) or 1 (failed)
```

FILES
```
          Files under the directory given by <dest> are deleted.

          You must have priviledge to write the target directory and file(s),
          otherwise the operation will fail.
```

ENVIRONMENT
```
          See vccEnv(1).
```

EXAMPLES

```
Delete RTAP environment named `wte49':

> vccEnvDelete -e wte49
vccInfo: connected to VCC database at wte49
vccEnvDelete: locate environment directory "/vlt/data/ENVIRONMENTS/wte49"
vccEnvDelete: unprotect env files under "/vlt/data/ENVIRONMENTS/wte49"
vccEnvDelete: delete env files under "/vlt/data/ENVIRONMENTS/wte49"
```

SEE ALSO

vccEnv(1), vccMake(1),

```
- - - - - -
Last change:  02/11/98-13:00
```

8.1.8 vccEnvInit(1)

NAME

```
vccEnvInit - initialize environment
```

SYNOPSIS

```
vccEnvInit -e <env> [-t <type>] [-h <host>] [-d <dest>]
```

DESCRIPTION

```
Initializes the environment <env> of the given <type>.

Detailed actions depending on <type> are:

LCU:    1. remove snapshot file
        2. configure boot-line with bootChange(1), reboot

RTAP:   1. create empty database from *.empty snapshots
        2. load database from DB branch into snapshots
        3. restore normal RtapEnvTable

QSEMU:  1. remove snapshot file
```

ARGUMENTS

```
<env>   Name of environment, eg. lt1hb, wt1tcs, wte1qs
        Mandatory, no default.

<type>  Type of environment: LCU | RTAP | QSEMU
        default_1: as defined in configuration database
        default_2: derived from <env>

<host>  Name of the host where environment files reside,
        default_1: as assigned in configuration database
        default_2: local host

<dest>  Name of destination directory,
        default: "$VLTDATA/ENVIRONMENTS/<env>" on <host>
```

OPTIONS

```
The following option may be given in any combination:

-q(uiet)        inhibits output to stdout
-v(erbose)      outputs more information than normally
```

CAUTIONS

```
This program calls vccRemExpect(1) to execute remote commands,
which requires the `expect' interpreter.
```

RETURN VALUE

```
exit status 0 (OK) or 1 (failed)
```

FILES

```
Files under the directory given by <dest> may be modified.

You must have priviledge to write the target directory and file(s),
otherwise the operation will fail.
```

ENVIRONMENT

See vccEnv(1).

EXAMPLES

```
Initialize LCU environment named `lte41' (also increase the rlogin
timeout to 20 seconds because of slow network or so):

> setenv VLT_VCCTIMEOUT 20
> vccEnvInit -e lte41
  vccInfo: connected to VCC database at wte49
  vccEnvInit: execute bootChange sequence on LCU te41 and reboot ...

Initialize RTAP environment named `wte49':

> vccEnvInit -e wte49
  vccInfo: connected to VCC database at wte49
  vccEnvInit: create empty database in "/vlt/data/ENVIRONMENTS/wte49"
  vccEnvInit: load database from DB branch ...
  vccEnvInit: restore normal RtapEnvTable ...
```

SEE ALSO

vccEnv(1), vccRemExpect(1)

```
- - - - - -
Last change:  02/11/98-13:00
```

8.1.9 vccEnvStart(1)

NAME

```
vccEnvStart - start environment
```

SYNOPSIS

```
vccEnvStart -e <env> [-t <type>] [-h <host>] [-d <dest>]
```

DESCRIPTION

```
Starts the environment <env> of the given <type>.

Detailed actions depending on <type> are:

LCU:    1. reboot LCU with VxWorks only
        2. install bootScript in boot-line with bootChange(1)
        3. execute bootScript

        In case of interactive mode (see OPTIONS -j), a pop-window
        is exposed to display the output during booting.

RTAP:   1. run RtapScheduler(1) in background
        2. wait and check for correct start-up of RtapScheduler
        3. wait and check for full database start-up

QSEMU:  1. run ccsScheduler(1) in background
        2. wait and check for correct start-up of ccsScheduler
        3. wait and check for full database start-up
```

ARGUMENTS

```
<env>   Name of environment, eg. lt1hb, wt1tcs, wte1qs
        Mandatory, no default.

<type>  Type of environment: LCU | RTAP | QSEMU
        default_1: as defined in configuration database
        default_2: derived from <env>

<host>  Name of the host where environment files reside,
        default_1: as assigned in configuration database
        default_2: local host

<dest>  Name of destination directory,
        default: "$VLTDATA/ENVIRONMENTS/<env>" on <host>
```

OPTIONS

```
The following option may be given in any combination:

-i(nteractive)  asks for input
-j(nteractive)  "pseudo-interactive" (no input, only pop-up window)
-j<timeout>     dto. but pop-up window is automatically deleted
                after <timeout> seconds after completion of start.

-q(uiet)        inhibits output to stdout
-v(erbose)      outputs more information than normally
```

CAUTIONS

```
This program calls vccRemExpect(1) to execute remote commands,
which requires the `expect' interpreter.

Do not try to start QSEMU environments on systems that are
not installed with CCS-lite!
```

RETURN VALUE

```
exit status 0 (OK) or 1 (failed)
```

FILES

```
Files under the directory given by <dest> are read.
A temporary file is used to store the LCU reboot output.
```

ENVIRONMENT

```
VLT_VCCBOOTTIME specifies the timeout in seconds to be used for
                LCUs while waiting for bootScript completion.
                If not defined, then the default is 120 seconds.

DISPLAY (optional) for display of pop-up window during LCU reboot.

See also vccEnv(1).
```

EXAMPLES

```
Start LCU environment named `lte41', also set timeout to complete
booting to 100 seconds:

> setenv VLT_VCCBOOTTIME 100
> vccEnvStart -e lte41
  vccInfo: connected to VCC database at wte49
  vccEnvStart: reboot LCU te41 with VxWorks only ...
  vccEnvStart: install bootScript in boot-line on LCU te41 ...
  vccEnvStart: execute bootScript on LCU te41 (timeout: 100 s) ...


Start RTAP environment named `wte49':

> vccEnvStart -e wte4910
  vccInfo: connected to VCC database at wte49
  vccEnvStart: run RtapScheduler in background ...
  RTAP/Plus Copyright (c) Hewlett-Packard (Canada) Ltd. 1988-1994
                All rights reserved.
  Real-Time Applications Platform (RTAP/Plus) release A.06.60
  CCS Shared Memory (Id = 828) : ACTIVE
```

SEE ALSO

```
vccEnv(1), vccRemExpect(1)
```

```
- - - - - -
Last change:  02/11/98-13:00
```

8.1.10 vccEnvStop(1)

NAME

```
vccEnvStop - stop environment
```

SYNOPSIS

```
vccEnvStop -e <env> [-t <type>] [-h <host>]
```

DESCRIPTION

```
Stops the environment <env> of the given <type>.

Detailed actions depending on <type> are:

LCU:    1. uninstall bootScript in boot-line with bootChange(1)
        2. reboot the system with VxWorks only

RTAP:   1. shutdown with RtapShutdown(1) or envsKill(1)

QSEMU:  1. shutdown with ccsShutdown
```

ARGUMENTS

```
<env>   Name of environment, eg. lt1hb, wt1tcs, wte1qs
        Mandatory, no default.

<type>  Type of environment: LCU | RTAP | QSEMU
        default_1: as defined in configuration database
        default_2: derived from <env>

<host>  Name of the host where environment files reside,
        default_1: as assigned in configuration database
        default_2: local host
```

OPTIONS

```
The following option may be given in any combination:

-q(uiet)        inhibits output to stdout
-v(erbose)      outputs more information than normally
```

CAUTIONS

```
This program calls vccRemExpect(1) to execute remote commands,
which requires the `expect' interpreter.
```

RETURN VALUE

```
exit status 0 (OK) or 1 (failed)
```

ENVIRONMENT

```
See vccEnv(1).
```

EXAMPLES

```
Stop LCU environment named `lte41':

> vccEnvStop -e lte41
  vccInfo: connected to VCC database at wte49
  vccEnvStop: uninstall bootScript and reboot te41 with VxWorks only
```

SEE ALSO

```
vccEnv(1), vccRemExpect(1)
```

```
- - - - - -
Last change:  02/11/98-13:00
```

8.1.11 vccShow(1)

NAME

```
          vccShow - Show contents of VCC database on screen
```

SYNOPSIS

```
          vccShow
```

DESCRIPTION

```
          Prints the contents of the VLT Central Configuration Database
          in tabular form to the screen.
```

ENVIRONMENT

```
          see VccInfo(n)
```

SEE ALSO

```
          VccInfo(n)
```

```
- - - - - -
Last change:  02/11/98-13:00
```

8.2 Functions Reference

The following sections contain - in alphabetical order - the manual pages for the functions and macros included in the programmatic interface of the modules.

8.2.1 lcubootAutoDrv(1)

NAME

```
lcubootAutoDrv,
lcubootAutoDevRegister,
lcubootAutoDrvInstall,
lcubootAutoDevCreate,
lcubootAutoDevCheck
- Automatic LCU Driver and Device Installation Facility
```

SYNOPSIS

```
int lcubootAutoDevRegister(const char *devName,
                           int addrSpace, void *busAddr,
                           int lccArg1, ... lccArg3)

int lcubootAutoDrvInstall(const char *moduleName,
                          int numDevices,
                          const char *drvFuncName,
                          const char *devFuncName,
                          int drvArg1, ... drvArg6)

int lcubootAutoDevCreate(const char *devName, int arg1, ... arg9)

int lcubootAutoDevCheck(const char *devBaseName, int nominalCount)
```

DESCRIPTION

```
These functions support the automatic installation of LCU
drivers and devices. All functions are intended to be
used directly from the VxWorks shell in a boot-script.

The script is aborted when a fatal error condition occurs, which
is signalled as a log message in the form:

<tid> (tShell): lcuboot: <message>: <faulty item>:<errno | faulty item>
<tid> (tShell): --- SCRIPT ABORTED ---

lcubootAutoDevRegister - registers a device for later creation and
        registration in LCC. Internal variables are set that are used
        in subsequent calls of `lcubootAutoDrvInstall' to install the
        driver and `lcubootAutoDevCreate' to create the devices.

        `busAddr' must be a valid VMEbus address in `addrSpace':

                VMEbus Address Modifier    addrSpace       busAddr
                ----------------------- --------------- -------------
                VME_AM_SUP_SHORT_IO            0x2d    0 .. 0xffff
                VME_AM_STD_SUP_DATA            0x3d    0 .. 0xffffff
                VME_AM_EXT_SUP_DATA            0x0d    0 .. 0xffffffff

        `lccArg1' ff. are are saved and passed to `lccRegisterDevice'
        in a subsequent call of `lcubootAutoLccRegisterDevs' to make
        all devices known to LCC, see lccRegisterDevice(3).

        The number of devices found so far is returned. This count is
        reset to zero by `lcubootAutoDrvInstall'.

        The script is aborted if the internal device table is full,
        or if the given address-space and bus-address could not be
        converted into a valid local address.
```

```
lcubootAutoDrvInstall - conditionally loads and installs a driver
        and prepares all data for subsequent device installation.

        `moduleName' is searched in BINPATH and loaded.
        The rest of the parameters are usually defaulted (see below).
        No operation is done and OK is returned if the number of
        devices evaluates to zero (see `numDevices' below),
        otherwise the installation is done accordingly.

        `drvFuncName' is the driver install function (e.g "xxxDrv").
        `devFuncName' is the device install function ("xxxDevCreate").
        `drvArg1' ff. are passed to the driver install function.

        The script is aborted in the following cases:
        - the driver module could not be found or loaded.
        - the driver installation failed, i.e. it did not return OK
        - the device installation function could not be found

        The following parameters are defaulted - when omitted - with:

          `numDevices' = effective count from `lcubootAutoDevRegister',
                or - if this is zero - from the variable
                `lcubootProbeCount' as set by `lcubootAutoProbe'.
                Both counts are reset to zero by the function.

          `drvFuncName' = "`moduleName'`lcubootAutoDrvFuncSuffix'"
          `devFuncName' = "`moduleName'`lcubootAutoDevFuncSuffix'"

          `drvArg1' = `numDevices' (after resolving as above)
          `drvArg2' = `lcubootAutoDrvArg2'
          `drvArg3' = `lcubootAutoDrvArg3'

        See VARIABLES for the contents of `lcubootAutoD...'.

        To install the corresponding devices, at least as many
        consecutive calls of `lcubootAutoDevCreate' must follow
        as the evaluated number of devices implies.

lcubootAutoDevCreate - performs the installation of a device using
        the function previously defined with `lcubootAutoDrvInstall'.
        The name of the device is given by `devName', which is passed
        with up to nine further arguments to the install-function.

        No operation is done and OK is returned if the number
        of devices to be installed as defined before with
        `lcubootAutoDrvInstall' has already been reached,
        or if the corresponding address given with
        `lcubootAutoDevRegister' is empty.

        It returns OK is the device installation is successful,
        otherwise the script is aborted.

        Note that there must be as many calls of this function stated
        in the boot-script as the maximum number of devices that shall
        be supported. However, usually fewer devices are actually
        installed.

        Note also that the base-addresses given as arguments to this
        function may be different from the probe-addresses stated
        in the call of `lcubootAutoDevRegister'.

lcubootAutoDevCheck - checks whether the number of devices installed
        in the system are equal to a given value.
```

```
                `devBaseName' denotes the name-prefix of the device family,
                e.g. "/acro" for digital I/O devices.
                `nominalCount' is the expected number of installed devices, a
                negative value means no check shall be done and returns OK.
                OK is returned if the actual and nominal count are equal,
                otherwise the script is aborted.

                This can be used at the end of the boot-script to verify that
                the expected number of devices are actually installed.
                Note however that the boot-script must then be updated
                according to the HW configuration.
```

VARIABLES

```
        The following global variables are provided and can be modified by
        the user in exceptional cases:

        lcubootAutoDrvFuncSuffix - Default suffix to the module-name to
                yield the driver install function.
                The variable is preset with: "Drv"
                The maximum length is 15 chars.

        lcubootAutoDevFuncSuffix - Default suffix to the module-name to
                yield the device install function.
                The variable is preset with: "DevCreate"
                The maximum length is 15 chars.

        lcubootAutoDrvArg2 - Default 2nd parameter for the driver install
                function, usually the number of channels.
                The variable is preset with: 25
                Note that that this parameter can be explicitly given as
                argument to `lcubootAutoDrvInstall'.

        lcubootAutoDrvArg3 - Default 3rd parameter for the driver install
                function, usually the timeout in ticks.
                The variable is preset with: 100
                Note that that this parameter can be explicitly given as
                argument to `lcubootAutoDrvInstall'.
```

ENVIRONMENT

```
        BINPATH - colon-separated directory searchpath for binary modules
```

CAUTIONS

```
        Most functions are not reentrant and should therefore only be used
        from LCU boot-scripts, where reentrancy is not important!

        Not more than 64 devices are supported in one LCU.

        The global variable `lcubootAutoDrvArg2' contains the default number
        of device channels that can be opened to all devices of a driver.
        When very many devices are present, this might have to be increased.
```

EXAMPLES

```
        This example automatically loads and installs the "acro" driver if
        at least one acro-device is present, and then installs the found
        number of devices, up to four.
        After that the found devices are registed in LCC.
```

```
acroN = lcubootAutoDevRegister("/acro0",0x2d,0x1000,1,1,1)
acroN = lcubootAutoDevRegister("/acro1",0x2d,0x1400,1,1,1)
acroN = lcubootAutoDevRegister("/acro2",0x2d,0x1800,1,1,1)
acroN = lcubootAutoDevRegister("/acro3",0x2d,0x1c00,1,1,1)

lcubootAutoDrvInstall "acro"

lcubootAutoDevCreate "/acro0",0x1000,0xb0,1
lcubootAutoDevCreate "/acro1",0x1400,0xb1,1
lcubootAutoDevCreate "/acro2",0x1800,0xb2,1
lcubootAutoDevCreate "/acro3",0x1c00,0xb3,1

lcubootAutoExec acroN,"sysIntEnable",1
lcubootAutoSpawn acroN,"tintACRO",200,0x18,2000,"acroInt"
...
lcubootAutoLccRegisterDevs


A similar - but maybe less elegant - implementation:

acroN = lcubootAutoProbe(0xffff1000,0xffff1400,0xffff1800,0xffff1c00)
lcubootAutoDrvInstall "acro",acroN,"acroDrv","acroDevCreate",acroN,5,80
# or with defaults simply: lcubootAutoDrvInstall "acro"
lcubootAutoDevCreate "/acro0",0x1000,0xb0,1
lcubootAutoDevCreate "/acro1",0x1400,0xb1,1
lcubootAutoDevCreate "/acro2",0x1800,0xb2,1
lcubootAutoDevCreate "/acro3",0x1c00,0xb3,1

e=calloc(512,1)
lcubootAutoExec acroN>0,"lccRegisterDevice","/acro0",0xffff1000,1,1,1,e
lcubootAutoExec acroN>1,"lccRegisterDevice","/acro1",0xffff1400,1,1,1,e
lcubootAutoExec acroN>2,"lccRegisterDevice","/acro2",0xffff1800,1,1,1,e
lcubootAutoExec acroN>3,"lccRegisterDevice","/acro3",0xffff1c00,1,1,1,e
free e
```

SEE ALSO

```
lcubootAutoEnv(1), lcubootAutoGen(1), lcubootAutoLcc(1),
loadLib(1), symLib(1),
ld(2)
```

```
- - - - - -
Last change:  05/11/97-10:58
```

8.2.2 lcubootAutoEnv(1)

NAME

```
          lcubootAutoEnv,
          lcubootAutoEnvInit
          - Automatic LCU Environment Setup Facility
```

SYNOPSIS

```
          int lcubootAutoEnvInit(void)
```

DESCRIPTION

```
          These functions support the automatic setup of LCU shell environment.
          All functions are intended to be used directly from the VxWorks shell
          in a boot-script.

          The script is aborted when a fatal error condition occurs, which
          is signalled as a log message in the form:

          <tid> (tShell): lcuboot: <message>: <faulty item>:<errno | faulty item>
          <tid> (tShell): --- SCRIPT ABORTED ---

          lcubootAutoEnvInit - initialize shell environment variable from the
                  LCU boot-line settings and set the shell-prompt.

                  The following variables are derived from the LCU boot-line
                  or derived from other variables, but only if they are not
                  already defined:

                          LOCALHOST   := <bootParams.targetName>
                          LOCALIPADDR := <bootParams.ead>
                          LOCALENV    := l${LOCALHOST}
                          HOSTNAME    := <bootParams.hostName>
                          HOSTIPADDR  := <bootParams.had>
                          HOSTENV     := w${HOSTNAME}

                          BOOTHOME    (see below)
                          BOOTROOT    (see below)
                          BOOTDB      := ${BOOTHOME}/DB
                          LOGFILE     := ${BOOTHOME}/logfile

                  BOOTHOME corresponds to the directory of the startup-script
                  as specified in the boot-line. BOOTROOT is derived from that
                  by going two level up in the directory hierarchy. This should
                  be the same then as VLTDATA on the host workstation.

                  Default search-paths are also set for binary files (BINPATH)
                  and module-boot-scripts (BOOTPATH). The parts with INTROOT
                  respectively MODROOT are omitted when not defined. The
                  CPU-dependent extension is omitted when CPU is not defined.

                  BINPATH  := $BOOTHOME:\
                              [$MODROOT/bin:]\             (if MODROOT defined)
                              [$INTROOT/vw/bin[/$CPU]:]\  (if INTROOT defined)
                              $VLTROOT/vw/bin[/$CPU]

                  BOOTPATH := $BOOTHOME:\
                              $BOOTROOT/config:\
                              [$MODROOT/bin:]\             (if MODROOT defined)
                              [$INTROOT/vw/bin/$CPU:]\    (if INTROOT/CPU def.)
                              [$INTROOT/vw/bin:]\         (if INTROOT defined)
```

```
                        $VLTROOT/vw/bin/$CPU:\      (if CPU defined)
                        $VLTROOT/vw/bin

            CONFIGPATH := $BOOTHOME:\
                          $BOOTROOT/config:\
                          [$MODROOT/config:]\       (if MODROOT defined)
                          [$INTROOT/config:]\       (if INTROOT defined)
                          $VLTROOT/config

            The shell-prompt is derived from the local environment name.

            Note that VLTROOT, INTROOT and MODROOT are not set!
```

ENVIRONMENT

```
    `lcubootAutoEnvInit' sets the above stated shell variables.
    VLTROOT and optionally INTROOT/MODROOT must be set before calling it.

    CPU can be optionally set to extend BINPATH and BOOTPATH.
    Values are the same as for the VxWorks compiler, e.g. MC68040.
```

CAUTIONS

```
    Most functions are not reentrant and should therefore only be used
    from LCU boot-scripts, where reentrancy is not important!
```

COMPATIBILITY

```
    Observe the following changes after the JUL95 VLT SW Release:
    - The structure under BOOTROOT has been changed.
    - The default HOSTENV is no longer appended by "qs".
```

SEE ALSO

```
    lcubootAutoGen(1), lcubootAutoDrv(1), lcubootAutoLcc(1),
    symLib(1)
```

```
- - - - - -
Last change:  05/11/97-10:58
```

8.2.3 lcubootAutoGen(1)

NAME

```
lcubootAutoGen,
lcubootAutoProbe,
lcubootAutoLoad,
lcubootAutoExec,
lcubootAutoSpawn,
lcubootAutoCd, lcubootAutoCdBoot
- Automatic LCU Module Installation Facility
```

SYNOPSIS

```
int lcubootAutoProbe(char *probeAddr0, ... *probeAddr9)
int lcubootAutoLoad(BOOL cond, const char *moduleName,
                    int reldFlag, int symFlag)
int lcubootAutoExec(BOOL cond, const char *funcName, int arg1, ...arg8)
int lcubootAutoSpawn(BOOL cond, const char *taskName,
                int priority, int options, int stackSize,
                const char *funcName, int arg1, ..arg4)
int lcubootAutoCd(BOOL cond, const char *searchPath,
                  const char *fileName)
int lcubootAutoCdBoot(BOOL cond, const char *scriptName)
```

DESCRIPTION

```
These functions support the automatic installation of all general
LCU modules. All functions are intended to be
used directly from the VxWorks shell in a boot-script.

The script is aborted when a fatal error condition occurs, which
is signalled as a log message in the form:

<tid> (tShell): lcuboot: <message>: <faulty item>:<errno | faulty item>
<tid> (tShell): --- SCRIPT ABORTED ---

lcubootAutoProbe - tests whether any hardware is present at the given
        addresses. Byte-read accesses are executed to verify this.
        Up to ten addresses can be given, the function stops after
        the first non-successful access.
        It returns the number of successful accesses.

        The counting begins normally from zero each time the function
        is called. If `probeAddr0' is NULL then the counting is
        continued from the last value, which is useful when more
        than ten addresses must be probed.

        This can be used to determine the number of boards/devices
        that are present in the system.

        Note that it cannot be checked whether the hardware found
        at a given address is actually the "correct" one.
        Standard pre-defined board addresses must be used as an
        essential precondition.

lcubootAutoLoad - if `cond' is TRUE, then it loads the binary module
        `moduleName', applying the search-path in the environment
        variable BINPATH.

        It also loads the module's error-definition files, provided
        they exist. It is silently ignored if no error-files are found.
        See lcubootErrorLoad(1) for details.
        No operation is done and OK is returned for FALSE `cond'.
        The same is true if a module with this name is already loaded
        and `reldFlag' is zero, otherwise the module is re-loaded.
        `symFlag' is usually zero and can be omitted, see loadLib(1).
        It returns OK if the module is found and loaded.
        It returns ERROR if the module is not found, however,
        the script is not aborted then. It aborts the script
        only if the module is found but cannot be loaded properly.

        This can be used to conditionally/optionally load a module.

lcubootAutoExec - if `cond' is TRUE, then it executes the given
        function with the given arguments.

        No operation is done and OK is returned for FALSE `cond'.
        It returns the called function's return-value, or ERROR
        if the function is not found in the symbol-table.

        This can be used to conditionally/optionally install a module.

lcubootAutoSpawn - if `cond' is TRUE, then it spawns `funcName' as a
        task with the name `taskName', using the given parameters,
        as in taskSpawn.

        No operation is done and OK is returned for FALSE `cond'.
        It returns the return-value of the taskSpawn call, or
        ERROR if the function is not found in the symbol-table.

        Note that fewer arguments can be passed to the task compared
        to a direct call of taskSpawn.

        This can be used to conditionally/optionally spawn a task.

lcubootAutoCd - if `cond' is TRUE, then it changes to the directory
        in which a given file is found. The file is searched applying
        the given colon-separated directory search-path.

        No operation is done and OK is returned for FALSE `cond'.
        It returns the OK, or ERROR in case of any failure.
        Note that after calling this function the current working
        directory is not reset to it's previous value.

lcubootAutoCdBoot - is similar to `lcubootAutoCd', but specifically
        intended for module-boot-scripts. It searches for `scriptName'
        applying the search-path in the environment variable BOOTPATH.

        The script is aborted when the module-boot-script could not
        be found in any of these directories.
```

VARIABLES

```
lcubootProbeCount - counts the valid addresses in `lcubootAutoProbe'
```

FILES

```
See lcubootErrorLoad(1) for error-definition files.
```

ENVIRONMENT

```
BINPATH  - colon-separated directory searchpath for binary modules
BOOTPATH - colon-separated directory searchpath for module-boot-scripts
```

CAUTIONS

```
Most functions are not reentrant and should therefore only be used
from LCU boot-scripts, where reentrancy is not important!
```

EXAMPLES

```
The first example searches for the module-boot-script of "lqs"
and executes it under the shell:

lcubootAutoCdBoot 1,"lqs.boot"
< lqs.boot

This module-boot-script could then have the following contents.
It automatically loads and sets up the "lqs" module.
After that the found lqs task is spawned.

lcubootAutoLoad 1,"lqs"

# Define Ping and Ack timeout(seconds)
lcubootAutoExec 1,"lqsInit",10,"rtap"

lcubootAutoExec 1,"lqsAddEnvTbl","wte13qs","te13","134.171.12.222",2223
lcubootAutoExec 1,"lqsAddEnvTbl","wte19qs","te19","134.171.12.228",2223
lcubootAutoExec 1,"lqsAddEnvTbl","wte13",  "te13","134.171.12.222",2155
lcubootAutoExec 1,"lqsAddEnvTbl","wte16",  "te16","134.171.12.225",2167

# Spawn the lqs task
lcubootAutoSpawn 1, "tLqs", 90, 0x18, 20000, "lqs"
```

SEE ALSO

```
lcubootAutoEnv(1), lcubootAutoDrv(1), lcubootAutoLcc(1),
lcubootErrorLoad(1),
loadLib(1), symLib(1),
ld(2)
```

```
- - - - - -
Last change:  05/11/97-10:58
```

8.2.4 lcubootAutoLcc(1)

NAME

```
lcubootAutoLcc,
lcubootAutoLccRegisterDevs
- Automatic LCU Installation Support Facility for LCC
```

SYNOPSIS

```
int lcubootAutoLccRegisterDevs(void)
```

DESCRIPTION

```
These functions support the automatic installation of LCC,
especially for devices. All functions are intended to be
used directly from the VxWorks shell in a boot-script.

The script is aborted when a fatal error condition occurs, which
is signalled as a log message in the form:

<tid> (tShell): lcuboot: <message>: <faulty item>:<errno | faulty item>
<tid> (tShell): --- SCRIPT ABORTED ---

lcubootAutoLccRegisterDevs - register all devices under LCC that were
        previously announced with `lcubootAutoDevRegister'.

        The number of registered devices is returned.
        The script is aborted when a call to `lccRegisterDevice' fails.
```

CAUTIONS

```
Most functions are not reentrant and should therefore only be used
from LCU boot-scripts, where reentrancy is not important!
```

SEE ALSO

```
lcubootAutoEnv(1), lcubootAutoGen(1), lcubootAutoDrv(1),
loadLib(1), symLib(1),
ld(2)
```

```
- - - - - -
Last change:  05/11/97-10:58
```

8.2.5 lcubootError(1)

NAME

```
lcubootErrorLoad,
lcubootErrorGetFormat
- LCU Error system functions.
```

SYNOPSIS

```
int lcubootErrorLoad(const char *modName)
int lcubootErrorGetFormat(const char *modName,
                          int errNumber,
                          char *severity,
                          char *format)
```

DESCRIPTION

```
This functions provides the facilities to load ERROR files and
retrieve the information about error text and format for a
given module and error number. lcubootErrorLoad is called automatically
at module loading, but can be used by itself for test purposes.

lcubootErrorLoad - load the ERROR definition file for the given
                   module, if the module does not have ERROR
                   files it just return. If the module has the
                   ERROR file (<mod>_ERRORS) but the index
                   file is not present (<mod>ERRORS.idx) the
                   script is aborted.

lcubootErrorGetFormat - retrieves severity and format
                        definition for the pair
                        (modName,errNumber).
        Returns values: lcubootOK if no problem
                        lcubootError_MOD_NOT_FOUND if the
                               module was not found
                        lcubootError_ERR_NOT_FOUND if the
                               error number was not found for
                               the module
                        lcubootError_INTERNAL if an internal
                               error occurs.
```

FILES

```
Error-defition files are accessed read-only under VLTROOT
respectively INTROOT:

        ERRORS/<mod>_ERRORS
        ERRORS/<mod>ERRORS.idx

where <mod> is the name of the module, as given by `modName'.
```

CAUTIONS

```
Note that the name of the module and the prefix of the
error-definition files must be the same, otherwise the
files will not be found!
```

EXAMPLES

```
lcubootErrorLoad("mod");
lcubootErrorLoad("motci");
```

```
status = lcubootErrorGeFormat("mod",mod_ERR_NOT_FOUND,
                              myFormat, mySeveritty);
if (status != lcudrvOK)
   switch (status) {
         case lcubootError_MOD_NOT_FOUND: ....
         case  lcubootError_ERR_NOT_FOUND: ...
         default: ...
   }
```

SEE ALSO

lcubootAutoEnv(1), lcubootAutoGen(1), lcubootAutoLcc(1), loadLib(1), symLib(1)

```
- - - - - -
Last change:  05/11/97-10:58
```

8.2.6 lcubootFile(1)

NAME

```
lcubootFile,
lcubootFileOpen - find and open a file
```

SYNOPSIS

```
int lcubootFileOpen(const char *searchPath,
                    const char *fileName,
                    int openMode)
```

DESCRIPTION

```
Search for a file with a given search path and open it.

searchPath - colon-separated list of directories
fileName - relative filename
openMode - mode passed as second parameter to `open'
```

RETURN VALUES

```
a file-descriptor (> 0) if the file is found and opened
lcubootERROR if the file is not found or cannot be opened
```

EXAMPLE

```
fd = lcubootFileOpen(getenv("BINPATH"), "lcc", O_RDONLY);
if (fd < 0) ...
...
close(fd);
```

CAUTIONS

```
The returned file-descriptor should be closed after operation.
```

SEE ALSO

```
open(2), close(2)
```

```
- - - - - -
Last change:  05/11/97-10:58
```

8.2.7 lcubootLog(1)

NAME

```
 lcubootLogInit
 lcubootLogFinish
 lcubootLogMsg

 - log of boot messages from lcuboot into a WS file.
```

SYNOPSIS

```
STATUS lcubootLogInit(void);
STATUS lcubootLogFinish(void);
STATUS lcubootLogMsg(char *fmt, int arg1, int arg2, int arg3,
                     int arg4, int arg5, int arg6)
```

DESCRIPTION

```
These functions supports logging of messages generated while
executing the LCU's bootScript.
Messages are logged into the file lcubootLogFile, this file is
located in the boot WS in the directory $VLTDATA/ENVIRONMENTS/<lcuEnv>.

lcubootLogInit -  Open the log file and adds its file
        descriptor to the VxWorks logging system. To avoid that the
        logfile size grows indefinetely, it is re-created
        every time the boot process is started and the file
        size is greater tha 0.

lcubootLogFinish - Logs "lcuboot end" in the logFile,
        closes it  and deletes the file descriptor from
        the VxWorks's logging system.
        If lcubootLogInit have not been called or if it have
        failed nothing is done.

lcubootLogMsg - Behaves exactly like vxWorks's logMsg, but
        it flushes the buffer of the logfile.
        If lcubootLogInit have not been called or if it have
        failed nothing is done.
```

RETURN VALUES

```
lcubootLogInit   -  lcubootERROR if it fails to open the
                    logFile, otherwise  lcubootOK.
lcubootLogMsg    -  lcubootOK.
lcubootLogFinish -  lcubootOK.
```

EXAMPLES

```
   lcubooLogInit
   lcubootLogMsg("hello %d %d %d testing ...",1,2,3,0,0,0)
   lcubootLogFinish
```

SEE ALSO

```
logLib
```

```
- - - - - -
Last change:  05/11/97-10:58
```

8.3 Directory Reference

The following sections contain the manual pages for the directory formats for environments.

8.3.1 ENVIRONMENTS_LCU(5)

NAME

ENVIRONMENTS_LCU - LCU standard environment directory

SYNOPSIS

```
$VLTROOT/templates/forEnvs/LCU/         (template)
$VLTDATA/ENVIRONMENTS/<env>/            (target)
```

DESCRIPTION

The directory contains the files needed by an LCU environment:

```
     $VLTDATA/
       |
       |--ENVIRONMENTS/
       |        |
       |        |--<env>/
       |        |    |
       |        |    |-- PROCESSES
       |        |    |-- bootScript
       |        |    |-- devicesFile
       |        |    |-- lcubootLogFile
       |        |    |-- logfile
       |        |    |-- rebootFile
       |        |    |-- userScript
       |        |    |
       |        |    |--dbl/
       |        |    |    |--Makefile
       |        |    |    |--DATABASE.db
       |        |    |    |--USER.db
       |        |    |
       |        |    |--DB/
       :        :
```

Each file is created from a template available in $VLTROOT.

<env> name of the LCU environment. Must begin with `l' (ell).

FILES

The standard template can be copied with envsCreate(1).

The actual target should be located under:

```
        $VLTDATA/ENVIRONMENTS/<env>/
```

The directory contains the following files and sub-directories:

Name	Purpose
PROCESSES	List of processes that accept commands
bootScript	Executed during start-up
devicesFile	Defines LCC software devices
logfile	Used by LCC
lcubootLogFile	Used by LCC
rebootFile	Used by LCC
userScript	For user definition during start-up
dbl/	Database source directory
dbl/DATABASE.db	Standard database
dbl/Makefile	Accepts `make db' to build database branch

```
dbl/USER.db              Configurable database part
DB/                      Directory branch generated by `make db'
```

ENVIRONMENT

```
VLTROOT - path to the VLT constant text area
VLTDATA - path to the VLT variable data area
```

SEE ALSO

```
envsCreate(1)
```

```
- - - - - -
Last change:  02/11/98-14:22
```

8.3.2 ENVIRONMENTS_QSEMU(5)

NAME

```
ENVIRONMENTS_QSEMU - QSEMU (CCS-Lite) standard environment directory
```

SYNOPSIS

```
$VLTROOT/templates/forEnvs/QSEMU/       (template)
$VLTDATA/ENVIRONMENTS/<env>/            (target)
```

DESCRIPTION

```
The directory contains the files needed by an QSEMU environment:

     $VLTDATA/
       |
       |--ENVIRONMENTS/
       |        |
       |        |--<env>/
       |        |    |
       |        |    |-- CcsEnvTable
       |        |    |-- PROCESSES
       |        |    |
       |        |    |--dbl/
       |        |    |    |--Makefile
       |        |    |    |--DATABASE.db
       |        |    |    |--USER.db
       |        |    |
       |        |    |--DB/
       :        :

Each file is created from a template available in $VLTROOT.

<env>  name of the QSEMU environment.
Must begin with `w' and should end with `qs'.
```

FILES

```
The standard template can be copied with envsCreate(1).

The actual target should be located under:

        $VLTDATA/ENVIRONMENTS/<env>/

The directory contains the following files and sub-directories:

Name                    Purpose
~~~~~~~~~~~~~~~~~       ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
PROCESSES               List of processes that accept commands
CcsEnvTable             Process definitions
dbl/                    Database source directory
dbl/DATABASE.db         Standard database
dbl/Makefile            Accepts `make db' to build database branch
dbl/USER.db             Configurable database part
DB/                     Directory branch generated by `make db'
```

ENVIRONMENT

```
VLTROOT - path to the VLT constant text area
VLTDATA - path to the VLT variable data area
```

SEE ALSO

```
        envsCreate(1)
```

```
- - - - - -
Last change:  02/11/98-14:22
```

8.3.3 ENVIRONMENTS_RTAP(5)

NAME

```
ENVIRONMENTS_RTAP - RTAP standard environment directory
```

SYNOPSIS

```
$VLTROOT/templates/forEnvs/RTAP/        (template)
$VLTDATA/ENVIRONMENTS/<env>/            (target)
```

DESCRIPTION

```
The directory contains the files needed by an RTAP environment:

     $VLTDATA/
       |
       |--ENVIRONMENTS/
       |        |
       |        |--<env>/
       |        |    |
       |        |    |-- PROCESSES
       |        |    |-- RtapEnvTable
       |        |    |
       |        |    |--dbl/
       |        |    |    |--Makefile
       |        |    |    |--DATABASE.db
       |        |    |    |--USER.db
       |        |    |
       |        |    |--DB/
       :        :

Each file is created from a template available in $VLTROOT.

<env>  name of the RTAP environment. Must begin with `w'.
```

FILES

```
The standard template can be copied with envsCreate(1).

The actual target should be located under:

        $VLTDATA/ENVIRONMENTS/<env>/

The directory contains the following files and sub-directories:

Name                     Purpose
~~~~~~~~~~~~~~~~~~~      ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
PROCESSES                List of processes that accept commands
RtapEnvTable.normal      Process definitions for normal operation
RtapEnvTable.startup     Process definitions for initial DB load
Rtap*Snap?               Rtap snapshot file for normal operation
Rtap*Snap?.empty         Rtap snapshot file for initial DB load
dbl/                     Database source directory
dbl/DATABASE.db          Standard database
dbl/Makefile             Accepts `make db' to build database branch
dbl/USER.db              Configurable database part
DB/                      Directory branch generated by `make db'
```

ENVIRONMENT

```
VLTROOT - path to the VLT constant text area
VLTDATA - path to the VLT variable data area
```

DATABASE CAPACITY POINTS LICENSE

```
Rtap*.empty files are configured to work with 1000 points capacity.
To change this, new *.empty files must be generated by :
- modifying RtapEnvTable to start RtapDbStartup manually:
  RtapDbStartup -r
- starting the environment (RtapScheduler -e <...>)
- choosing the option "Create empty database"
- modifying the field "Maximum number of points"
- choosing the option "Accept configuration: startup"
- stopping the environment (RtapShutdown -e <...>)
- renaming newly created files into .empty (RtapDiskDb, RtapHdr*, RtapRam*,
  RtapSnapCtrl)
- restoring RtapEnvTable

Note that when running ls_stat or RtapPerfMon, licenses are displayed in
64 points units (16 units = 1000 points; 125 units = 8000 points).
```

SEE ALSO

```
envsCreate(1)
```

```
- - - - - -
Last change:  02/11/98-14:22
```

8.3.4 lcuboot(5)

NAME

```
lcuboot - LCU boot-script organization
```

DESCRIPTION

```
After loading the VxWorks image, an LCU sources a boot-script
to load and initialize all further software it needs.

Module-specific sub-boot-scripts are called from the main
`bootScript' of the LCU for each module that shall be initialized.

For each module there must be a VxWorks boot-script named
"<module>.boot" in one of the directories stated in FILES
that loads and initializes the module.
See Module.boot(5) and Driver.boot(5) for the typical contents
of that file.

For each sub-boot-script there should be a man-page named
"<module>.boot(5)" that explains the contents of the boot-script.

For each driver sub-boot-script there should be a man-page
named "<module>(4)" that explains the respective board-hardware
(jumpers etc.).
```

FILES

```
The files "bootScript" and "userScript" must be located in:

        $VLTDATA/ENVIRONMENTS/<lcu-env>/

This directory is assigned on to the variable BOOTHOME on the LCU.
VLTDATA is only defined on the WS, and corresponds to BOOTROOT on LCU.

Binary files for all modules to be loaded are searched in
BINPATH, which consists by default of the following directories
(cpuName is for instance MC68040):

        1. $BOOTHOME
        2. $INTROOT/vw/bin/<cpuName>/<module>
        3. $VLTROOT/vw/bin/<cpuName>/<module>

The module-boot-scripts are searched in BOOTPATH, which has
the following order of priority:

        1. $BOOTHOME/<module>.boot
        2. $BOOTROOT/config/<module>.boot
        3. $INTROOT/vw/bin/<cpuName>/<module>.boot
        4. $INTROOT/vw/bin/<module>.boot
        5. $VLTROOT/vw/bin/<cpuName>/<module>.boot
        6. $VLTROOT/vw/bin/<module>.boot

All boot-scripts are initially installed under "$VLTROOT/vw/bin".

If you want to insert some changes that shall apply only to
one specific LCU-environment, then copy the respective script
from "$VLTROOT/vw/bin/<cpuName>" to "$VLTDATA/ENVIRONMENTS/<lcu-env>".

If you want to insert some changes that shall apply to all
LCU-environments not having the above mentioned private version,
then copy the respective script either to your INTROOT, or
```

```
          from "$VLTROOT/vw/bin/<cpuName>" to "$VLTDATA/config/".
```

ENVIRONMENT

```
          VLTROOT - path to global VLT installation area (mandatory)
          VLTDATA - path to global VLT data area (mandatory)
          INTROOT - path to private installation area (optional)
```

SEE ALSO

```
          bootScript(5), userScript(5), Module.boot(5), Driver.boot(5)
```

```
- - - - - -
Last change:  05/11/97-10:58
```

8.4 Files Reference

The following sections contain the manual pages for the file formats used in the modules.

8.4.1 bootScript(5)

NAME

```
          bootScript - LCU standard boot script
```

SYNOPSIS

```
          $VLTDATA/ENVIRONMENTS/<env>/bootScript
```

DESCRIPTION

```
      The LCU boot script allows to load and initialise all the software
      running on the LCU:

              . drivers and other support modules,
              . Q Server,
              . LCU Common Software,
              . user applications.

      Several configuration sections allow to set-up the boot script for a
      specific environment and target system.
      This version of the boot script is a generic version that must be
      customized for each LCU.
      Values to be given to customization parameters are indicated by
      letters between angle brackets (e.g. <hostName>) and must be replaced
      by the actual values.

      The following sections correspond to the recommented order of sections
      in the bootScript.
```

ROOT CONFIGURATION

```
      Configurable environment variables and parameters

      Mandatory variables:

          VLTROOT         Path of the VLTROOT area on the boot host.
                          Example:        /vlt/DEC95

      Optional variables:

          INTROOT         Path of the INTROOT area on the mounted host.
                          This variable is optional:
                          - if not defined then all files are accessed in VLTROOT
                          - if defined then INTROOT has priority to VLTROOT
                          Examples:       /diskb/userx/INTROOT
                                          /earth/projects/ESO_LCU/INTEGRATION

          MODROOT         Path of the MODROOT area on the mounted host.
                          This variable is optional:
                          - if defined then MODROOT has highest priority.

          BOOTHOME        Path of the home directory in which the LCU bootScript
                          etc. are located. The variable is optional, it can
                          be derived automatically by `lcubootAutoEnvInit'.

      Note that the local filenames of xxxROOT can also be fixed to for example
      "/VLTROOT" and "/INTROOT", and the assignment to remote filesystems be
      done via NFS mounting, see next section.
```

NETWORK CONFIGURATION

```
Files are accessed through NFS by the LCU boards during system loading and
initialisation and for database loading/unloading and backup/restore
operations.

By default, all exported filesystems of the boot server host are mounted
by VxWorks boot procedure. The NETWORK configuration section allows to
mount filesystems from other hosts on the LCU:

    nfsAuthUnixSet "<hostName>",<userId>,<groupId>
    hostAdd "<hostName>","<ipAddr>"
    nfsMount "<hostName>","<fileSystem>","<localName>"

    <hostName>      Name of remote system. Example: "earth"

    <userId>        Unix user id (*) to be used by the LCU for NFS access.
                    Example: 150

    <groupId>       Unix group id (*) to be used by the LCU for NFS access.
                    Example: 50

                    (*) Can be any existing UNIX user, but conventionally
                        it is the one used by the LCU to boot, namely "vx"

    <ipAddr>        Internet address of remote system.
                    Example: "192.9.200.2"

    <fileSystem>    Name of remote filesystem. Example: "/earth"

    <localName>     Local name for the filesystem. Example: "/earth"
```

LCUBOOT CONFIGURATION

```
The "lcuboot" module should be present under VLTROOT and is loaded
usually from there. If it is (also) present in INTROOT then it can
optionally be loaded from there, if explicitly stated.
The module provides the functions that are used during the
further execution of the boot-script.

No user-specific configuration is necessary in this section,
except that the loading from INTROOT can be enabled.
```

ENVIRONMENT CONFIGURATION

```
The following environment variables can be configured by the user.
A call of `lcubootAutoEnvInit' automatically defines all variables
that are left undefined by the user with default values, which will
fit in most cases.

LOCALHOST       Unix host name of LCU. Example: "moon"
LOCALENV        Environment name of LCU. Example: "lmoon"
LOCALIPADDR     Internet address of LCU. Example: "192.9.200.45"
LOCALTCPPORT    TCP port number of LCU. Example: 2160

HOSTNAME        Name of boot server host. Example: "earth"
HOSTENV         Environment name of boot server host. Example: "wearth"
HOSTIPADDR      Internet address of boot server host. Ex: "192.9.200.2"
HOSTTCPPORT     TCP port number of boot server host. Example: 2301

BOOTROOT        root-directory from which the LCU is booting
```

```
BOOTHOME        private LCU boot-directory under BOOTROOT
BOOTDB          directory of LCU boot-database
LOGFILE         name of LCU log-file
```

MODULES CONFIGURATION

Modules are divided into SYSTEM modules (that are usually always necessary and should always be loaded in correct order) and USER modules (that are normally not always necessary).

All modules should be loaded first, and initialized after all modules have been loaded. Module-specific sub-boot-scripts should be called to load (if not already done) and initialize each module.

The driver modules can normally always be included, because they install themselves automatically if corresponding devices are found.

See lcuboot(5) for the search directories of module images and boot-scripts.

DEVICES CONFIGURATION

All devices are normally automatically installed when corresponding hardware is found. To check whether the expected number of devices for each device-family have actually been installed, the respective line must be uncommented and the expected device-count must be stated.

USER CONFIGURATION

In this section the USER modules are initialized by calling their respective module-boot-scripts.

After that the "userScript" in the LCU's boot-directory is executed, which can have any user-defined contents.

TERMINATION

Bootstrap terminating actions of LCC.
No user configuration necessary.

FILES

See lcuboot(5) for general organization of files.

SEE ALSO

lcuboot(5), userScript(5), Module.boot(5)

- - - - - -
Last change: 02/11/98-14:22

8.4.2 devicesFile(5)

NAME

```
devicesFile - LCU device table
```

SYNOPSIS

```
    $VLTROOT/vw/BOOT/<lcuenv>/devicesFile
```

DESCRIPTION

```
The devicesFile lists the software devices controlled by LCC. It
contains the definition of a TABLE attribute of the database used
by LCC (:LCC:DEVICES:deviceTable). The table contains the following
fields:
  - name of software device (Bytes20),
  - name of software device control process (Bytes20),
  - initialisation status of software device (UInt32) [LCC internal use],
  - reply status of software device (Logical) [LCC internal use],
  - simulation status of software device (Logical) [LCC internal use,
    updated by lccDevEnterSim and lccDevExitSim],
  - state of software device (UInt32) [LCC internal use, updated by
    lccSetDeviceState].
The attribute is restored by LCC using the dbRestore function
when the corresponding LCU is rebooted.
```

EXAMPLES

```
   <CWP>: :LCC:DEVICES
   <ATTRIBUTE>: deviceTable <TYPE>: Table <REC>: 0 - 2 <FIELDS>: 0 - 5
            "motor1" "MotorControl1" 1 0 0 1
            "motor2" "MotorControl2" 1 0 0 1
            "motor3" "MotorControl3" 1 0 0 1

the values 0 - 2 and 0 - 5 in the above example give respectively the
records in the table and the fields of each record.

For an LCU configured without software devices, the devicesFile
shall contain the following line:

   <CWP>: :LCC:DEVICES
```

```
- - - - - -
Last change:  02/11/98-14:22
```

8.4.3 userScript(5)

NAME

```
userScript - LCU environment user-configurable initializations
```

SYNOPSIS

```
$VLTDATA/ENVIRONMENTS/<env>/userScript
```

DESCRIPTION

```
This script allows to load and initialise application software.
It is called from the LCU boot script (bootScript) after all
drivers and LCC Common Software have been loaded and initialised.
```

FILES

```
See lcuboot(5) for general organization of files.
```

SEE ALSO

```
lcuboot(5), bootScript(5), Module.boot(5)
```

```
- - - - - -
Last change:  02/11/98-14:22
```

8.5 Panel Widgets and Libraries

TBD

___oOo___