ELTEC

elektronik mainz

OS-9 V3.0 on EUROCOM-17

Installation Guide

Software Manual

Revision 2 A

Rev.	Changes	Date
1 A	First Edition valid for Software Revision 1.A	18.04.94, D.W.
2A	Second Edition valid for Software Revision 3.A	15.03.96, D.W.

ELTEC Elektronik AG

Galileo-Galilei-Str. 11	Postfach 42 13 63
D-55129 Mainz	D-55071 Mainz

Telephone	+49 (61 31) 9 18-0
Telefax	+49 (61 31) 9 18-1 99

Table of Contents

List of Tables

Scope of Delivery

Description:		Order No.:
OS-9 V3.0	Extended OS-9 V3.0.2 for EUROCOM-17 Industrial MGR	**W-O917-C505**

Options

Description:	Order No.:
None	

i

The last letter of the order numbers refers to the software revision and is subject to changes. Please contact ELTEC for information about valid order numbers.

Example: *W-O916-B105*

Revision number, subject to change!

Related Products

Description:		Order No.:
Documentation:	Software Manual OS-9 V3.0 on EUROCOM-17	H-O917-B509
	Hardware Manual EUROCOM-17	V-E17.-B991
	Software Manual RMon	W-FIRM-A209
Hardware:		
EUROCOM-17:	Single 68040, 25 MHz, VME32	V-E17.-A510
	Single 68040, 25 MHz, VME32, Graphics	V-E17.-A511
	Single 68040, 25 MHz, VME32, Network	V-E17.-A512
	Single 68040, 25 MHz, VME32, Graphics, Network	V-E17.-A515
	Single 68040, 25 MHz, VME32, SCSI	V-E17.-A516
	Single 68040, 25 MHz, VME32, SCSI, Graphics	V-E17.-A517
	Single 68040, 33 MHz, VME32	V-E17.-A540
	Single 68040, 33 MHz, VME32, Graphics	V-E17.-A541
	Single 68040, 33 MHz, VME32, Network	V-E17.-A542
	Single 68040, 33 MHz, VME32, Graphics, Network	V-E17.-A543
	Single 68040, 33 MHz, VME32, SCSI	V-E17.-A544
	Single 68040, 25 MHz, VME32, SCSI, Graphics	V-E17.-A545
	Double 68040, 33 MHz, VME32	V-E17.-A640
	Double 68040, 33 MHz, VME32, Graphics	V-E17.-A641
	Double 68040, 33 MHz, VME32, Network	V-E17.-A642
	Double 68040, 33 MHz, VME32, Graphics, Network	V-E17.-A643
	Double 68040, 33 MHz, VME32, SCSI	V-E17.-A644
	Double 68040, 25 MHz, VME32, SCSI, Graphics	V-E17.-A645

The last letter of the order numbers refers to the hardware revision and is subject to changes. Please contact ELTEC for information about valid order numbers.

Example: *V-E16.-B105*

Revision number, subject to change!

Conventions

If not otherwise specified, addresses are written in hexadecimal notation and identified by a leading dollar sign ("$").

Signal names preceded by a slash ("/"), indicate that this signal is either active low or that this signal becomes active with the trailing edge.

b	bit
B	byte
K	kilo, means the factor 400 in hex (1024 decimal)
M	mega, the multiplication with 100 000 in hex (1 048576 decimal)
MHz	1 000 000 Hertz

Software-specific abbreviations:

<BS>	Back Space ($8)
<CAN>	Control-X ($19)
<Ctrl>	Control
<CR> <ENTER>	Carriage Return ($D)
<ESC>	Escape Character ($2B)
<LF>	Line Feed ($A)
<SP>	Space ($20)
NMI	Non-maskable Interrupt

How to Use this Manual

Document Conventions

Font Types:

Font	Use
Helvetica, 8 Pt	Tables and drawings
Helvetica, 10 Pt	Signal names
Times, italic	*Notes*
`Courier, bold`	`Program code, function names, commands, file names, module names`
Times, bold	**Emphasized text**

Other conventions:

Indicates information that requires close attention.

Indicates critical information that is essential to read.

Indicates information that is imperative to read. Skipping this material, possibly causes damage to the system.

1 Getting Started

This manual contains informations about the implementation dependent part of OS-9/68K ELTEC systems.

We recommend, that you are familiar with the following documentations:

- EUROCOM-17 Hardware Manual
- RMon Manual
- OS-9/68K Users Manual
- Using UMACS

1.1 Pre-Installed System

If you have ordered a complete system, the operating system will already be installed completely on the harddisk, when shipped off. You do not need the floppy disks delivered with the system and you are also not concerned with the whole installing procedure (hardware configuration, formatting, copying, ...) as described in this manual (... until the next OS-9 update arrives).

Only in case of a total crash of the system these disks are used to reconfigure the system as described - so keep these disks in a save place!

1.2 No Pre-Installed System

If you want to use a serial line terminal, configure it as follows:

9600 baud,
1 start bit,
8 data bits,
1 stop bit,
no parity.

Then connect the RJ11 socket and the terminal I/O socket. Set hex switch S2 on front panel to 3 for serial I/O with RMon. If you are not using an ELTEC cable, refer to the hardware manual for correct connection of signals.

If you want to use a VGA monitor and MF-2 keyboard, set hex switch S2 to '0'.

Connect the floppy drive(s) and harddisk(s) according to the description in the hardware manual and jumper them as mentioned in Section 4.2 'Fixed Device Parameters' of this manual.

Now switch on the terminal and the EUROCOM-17. The LED display now indicates 'F', and you should get the power-on message of the RMon monitor from the terminal.

The monitor program on EUROCOM-17 has a **`SCSI`** command (check SCSI bus). This command can be used to check connected devices on the SCSI bus. It checks SCSI IDs 0 to 6 and on each ID logical unit number (LUN) 0 to 3. For ever ready device a '+' is issued.

At this point boot OS-9/68K (see Section 2.1 'Booting from Floppy Disk').

2 Installing OS-9 V3.0

This chapter explains the installation of the new OS-9 release on a harddisk.

In some cases the installation program creates a directory README. This directory contains further information about this distribution which is not covered by this documentation.

In some cases the OS-9 was delivered with an addendum disk with new or bugfixed modules. See the file **`ReadMe.Addendum`** *on this disk for further details.*

2.1 Booting from Floppy Disk

After power-up, the EUROCOM-17 prompts with its monitor program. All commands described in the RMon manual are ready to be used. Insert disk labelled 'OS9 Bootdisk' into drive 0 if you wish to boot from floppy.

If you ordered a system with harddisk, the whole operating system and the boot file are installed on the harddisk by ELTEC. In this case skip the steps 1 to 4 and boot directly.

In the following items enclosed in " " are monitor prompts. Your response are enclosed in ' '.

The description of the **`set`** *command refers to RMon Version 2.0 or higher.*

The answers to the prompts given below correspond to the OS-9 as delivered by ELTEC. Selecting other SCSI IDs or LUNs means that you first have to configure your own corresponding OS-9 system.

"[***]>"

1. Type 'set<CR>' to enter the configuration set.
   "Select item: "
   Type 'e' for boot configuration.

2. Type 'b' to change boot device.
   "Boot device: Harddisk"
   Press '<SP>' to select floppy as boot device.
   Press '<CR>' to accept.

   Type 'c' to change boot controller.
   "Controller: TEAC"
   Press '<SP>' to select SCFL as boot device.
   Press '<CR>' to accept.

3. Type 'x' to exit boot menu.

4. Type 'x' to exit setup menu.
   "Save parameters (y/n)?"
   Type 'y' to save the selected parameters.
   "[***]>"

5. Type 'boot<CR>' to boot OS-9/68K.

6. The boot device is selected and the operating system comes up.

2.2 Installing OS-9 V3.0 on an Empty Harddisk

It is recommended to install the operating system on a fresh formatted device.

1. Boot the system from floppy disk (see Section 2.1 'Booting from Floppy Disk').

2. Enter `install` at shell prompt to start the installation program.

   ```
   $ install
   ```

3. The following text is now displayed:

   ```
   Loading installation modules ...
   ```

   After loading the necessary installation modules, you have to enter your destination:

   ```
   Loading installation modules ... done.

   h0 Harddisk with embedded SCSI (ID 6)
   h1 Harddisk with embedded SCSI (ID 5)
   h2 Harddisk with embedded SCSI (ID 2)
   h3 Harddisk with embedded SCSI (ID 3)
   h4 Harddisk with embedded SCSI (ID 4)

   Select the destination drive [h0]:
   ```

   Press <ENTER> to accept the default drive /h0 or enter the name of the drive without leading slash.

   We assume that your drive is /h0.

   ```
   Format /h0? y/[n]/a:
   ```

   If you want to format your drive enter now `y`, press <ENTER> for no formatting, or `a` to abort the installation program.

   ```
   Do you want to copy CMDS, SYS and ETC directories to the
   root directory to create a self-hosted development envi-
   ronment? [y]/n
   ```

   If your development system is a EUROCOM-17, you have to answer with `y`. If you do not develop on a EUROCOM-17, you should not create CMDS, SYS and ETC directories, because these directories are already created for your specific system.

```
You can rename the existing directories to <DIR>.old
(any already existing <DIR.old will be removed.

If you do not rename the directories, the existing files
will be overwritten with files from distribution !!!

Rename [y]/n/a:
```

Enter `y` for renaming, `n` for not renaming or `a` to abort the installation.

The whole OS-9 is delivered as compressed files. First, all disks are copied to the destination harddisk.

```
Insert disk labelled SYS 1 ... <ENTER>

Insert disk labelled SYS 2 ... <ENTER>

Insert disk labelled ISP   ... <ENTER>
```

Now the installation program will decompress the archives. This will take a while.

```
Do you want to create a new OS9Boot file for drive /h0? y/[n]
```

You should only answer with `y` if you want to create a self-hosted development environment. In all other cases answer with `n`.

i

A sample bootlist is located in MWOS/OS9/68020/PORTS/E17/BOOTLISTS. If you want to change the bootfile, edit the sample bootlist called `dd.bl`*. Then change current directory to MWOS/OS9/68020/PORTS/E17 and create bootfile with OS9GEN:*

```
$ chd MWOS/OS9/68020/PORTS/E17
$ dmode /dd format=on
$ os9gen /dd -z=bootlists/dd.bl -eb=300
```

4. Reset the system and reboot the operating system.

2.3 Updating OS-9 V2.X to V3.0

Installing the software on a harddisk already containing an OS-9 (older version or version for a different CPU) may cause trouble if modules are mixed between different releases. The following procedure is a save way to avoid this:

0. Make a complete backup of the harddisk.
1. Generate a bootable floppy with the old version of the operating system.
2. Insert boot disk and change current directory to floppy. We assume that the floppy drive is named /d0.

   ```
   $ chd /d0
   ```

   Now start the installation program:

   ```
   $ install
   ```

If you choose 'rename exisiting directories' in the installation program the CMDS, SYS, ETC, and MWOS directories and the **`startup`** *file are renamed to CMDS.OLD, SYS.OLD, ETC.OLD, MWOS.OLD and* **`startup.old`**. *If CMDS.OLD, SYS.OLD, ETC.OLD, MWOS.OLD and* **`startup.old`** *already exist, they will be deleted!*

3. Same as step 3 in Section 2.2 'Installing OS-9 V3.0 on an Empty Harddisk'.
4. Reset the system and reboot the operating system.

2.4 68040 Cache Configuration

ELTEC supplies a modified **`cache040`** module which configures the address space of the EUROCOM-17 as follows (for a 32 MB type):

\$0000.0000 - \$1FFF.FFFF	- data cache enabled, copy back mode
\$0200.0000 - \$FFFF.FFFF	- data cache disabled, all serialized
\$0000.0000 - \$FFFF.FFFF	- instruction cache enabled, write through mode

Using data cache in copy back mode will result in best CPU performance.

If the EUROCOM-17 has an MC68040 CPU (no 68EC040), you may change cache mode for user state (see chapter 'System Memory Cache Lists' on page 2-24 in the OS-9 Technical Manual for details).

The default values for the Transparent Translation Registers are located in the ***`INIT`*** *module. If you want to change these values, change file* ***`systype.d`*** *in MWOS/OS9/68020/PORTS/E17 and rebuild the* ***`INIT`*** *module.*

3 Drivers and Descriptors

This chapter describes the drivers and descriptors for clock, RBF, SCF and SBF type devices. Your delivery may contain more drivers and descriptors as mentioned here, since ELTEC feels free to add devices as they become ready, without changing the documentation. All makefiles and system dependent files (like **`systype.d`** or descriptors) are located in MWOS/OS9/68020/PORTS/E17.
System independent files (like kernel, RBF, SBF or SCF) are located in MWOS/OS9/68000/CMDS/BOOTOBJS or
MWOS/OS9/68020/CMDS/BOOTOBJS.

If you have created a self-hosted development system, all drivers and descriptors are also located in /dd/CMDS/BOOTOBJS.

3.1 Clock Modules

For time slicing, OS-9 needs a real-time clock that periodically interrupts the CPU and an appropriate clock driver module to handle the interrupt. For the EUROCOM-17, the VIC068 internal timer is used as clock interrupter.

- **`tk17`**
  Clock driver module: uses VIC068 internal timer feature for time slice interrupt. It also handles the watchdog.
- **`tkscpu17`**
  Clock driver module for secondary CPU.
- **`rtc17`**
  Subroutine module used by both clock driver modules to read the board real-time clock.

Directory MWOS/OS9/SRC/SYSMODS/GCLOCK contains the sources for the clock modules:

`tkvic.a`	- clock driver module using the VIC068 timer
`tkscpu.a`	- clock driver for secondary CPU
`tickgeneric.a`	- hardware-independent part of the clock driver module
`rtc48t02.a`	- subroutine module which reads the real-time clock of the EUROCOM-17

3.2 RBF Drivers

The RBF drivers are structured in a physical and logical part.

- **`scsi17`**
  is the physical driver which deals the NCR53C720 SCSI I/O controller and has to be loaded if any SCSI I/O is desired.
- **`rbvccs`** / **`rbcvccs`**
  is the logical driver for all harddisks with embedded SCSI controller.
- **`rbteac`** / **`rbcteac`**
  is the logical driver for TEAC FC-1 floppy disk with integrated SCSI controller.
- **`rbscfl`** / **`rbcscfl`**
  is the logical driver for ELTEC's SCFL floppy controller. The centronics port is not supported on EUROCOM-17 and there is no disconnect/ reselect capability.

*SCSI commands can be directly sent using the **`DoDirect SetStat`** (**`SS_DCmd`**) call of the appropriate driver. The file **`dodi.c`** in MWOS/OS9/ SRC/IO/SCSI/DODI contains the C interface to this **`SetStat`** entry.*

`Rbcvccs`, **`rbcteac`** and **`rbcscfl`** are dummy drivers for Snowtops disk caching software. The disk cache, called **`DCH`**, requires two additional **`SetStat`** calls (**`SS_Cache`** and **`SS_CacheXfr`**) which are supplied by the dummy drivers. If the dummy driver was invoked with a **`SS_Cache`** or **`SS_CacheXfr`**, the request is routed to the cache file manager (**`CFM`**). If there is no request to **`CFM`** or **`DCH`** is not enabled, the request is directly passed to the logical driver.

You always have to load the dummy drivers together with the appropriate logical driver, otherwise your SCSI I/O fails with error 000:221.

Never use any descriptors referring to the same logical driver without using the dummy driver. Otherwise you risk having two drivers simultaneously trying to control the same interface.

3.3 RBF Descriptors

The sources of these descriptors are located in directory MWOS/OS9/IO/RBF/DESC.

3.3.1 Descriptors for Harddisk with embedded SCSI

All descriptors for embedded SCSI disk are located in CMDS/BOOTOBJS/VCCS and have the logical unit number (LUN) 0.

`h0           dd.h0`
The h0 descriptors refer to SCSI ID 6.

`h1           dd.h1`
The h1 descriptors refer to SCSI ID 5.

`h2           dd.h2`
The h2 descriptors refer to SCSI ID 2.

`h3           dd.h3`
The h3 descriptors refer to SCSI ID 3.

`h4           dd.h4`
The h4 descriptors refer to SCSI ID 4.

For disks with wide SCSI the following descriptors are located in CMDS/BOOTOBJS/VCCS.

`h0_wide      dd.h0_wide`
The h0 descriptors refer to SCSI ID 6.

`h1_wide      dd.h1_wide`
The h1 descriptors refer to SCSI ID 5.

`h2_wide      dd.h2_wide`
The h2 descriptors refer to SCSI ID 2.

`h3_wide      dd.h3_wide`
The h3 descriptors refer to SCSI ID 3.

`h4_wide      dd.h4_wide`
The h4 descriptors refer to SCSI ID 4.

Do not use wide-descriptors with 8 bit targets because this crashes some targets which do not reject the wide-transfer-message sent by the driver.

The embscsi descriptors support variable sector sizes, i.e. they adapt automatically to the physical sector size of the harddisk.

3.3.2 Descriptors for TEAC FC-1 Controller

All descriptors are located in CMDS/BOOTOBJS/TEACFC1 and are for SCSI ID 3.

- **`d0/d1`**
  Descriptor for drive select 0 (LUN 0) / drive select 1 (LUN 1). It is for the Microware 38W7 floppy disk format.
- **`u0/u1`**
  Descriptor for drive select 0 (LUN 0) / drive select 1 (LUN 1). It is for the Microware universal floppy disk format.
- **`s0/s1`**
  Descriptor for drive select 0 (LUN 0) / drive select 1 (LUN 1). The descriptor supports HD disk format with 32 sectors per track. This is a special ELTEC floppy disk format.
- **`d0h/d1h`**
  Descriptor for drive select 0 (LUN 0) / drive select 1 (LUN 1). The descriptor supports HD disk format with 18 sectors à 512 bytes per track.

3.3.3 Descriptors for SCFL Controller

All descriptors are located in CMDS/BOOTOBJS/SCFL and are for SCSI ID 1.

- **`d0/d1`**
  Descriptors for drive select 0 (LUN 0)/ drive select 1 (LUN 1). They are for the Microware 58W7 (38W7) floppy disk format.
- **`u0/u1`**
  Descriptors for drive select 0 (LUN 0) / drive select 1 (LUN 1). They are for the Microware universal floppy disk format.
- **`s0/s1`**
  Descriptor for drive select 0 (LUN 0) / drive select 1 (LUN 1). The descriptor supports HD disk format with 32 sectors per track.
- **`d0h/d1h`**
  Descriptor for drive select 0 (LUN 0) / drive select 1 (LUN 1). The descriptor supports HD disk format with 18 sectors à 512 bytes per track.

3.4 SCF Drivers and Descriptors

- **`scrmon`**
  Depending on the setup of the RMon, the VGA monitor/AT-keyboard or one of the serial channels may be used as console. The console is implemented as SCF device with the device driver **`scrmon`**, which operates in **polling** mode. The console device descriptor name is always term.
  Using **`scrmon`** makes sure that the OS-9 configuration adapts to the RMon configuration.

i

The device driver **`scrmon`** *cannot receive more than three characters per 20 ms. If the setup feature of the RMon is not needed, the interrupt driven device driver* **`sc17cons`** *with* **`term.cons`** *for VGA monitor/AT-keyboard or the device driver* **`sccd2401`** *with* **`term.t0`** *should be used.*

- **`sc17cons`**
  This is the **interrupt driven** driver for the MF-2 keyboard and the graphics. The driver needs a **`kbset`** module, so one of the following modules from CMDS/BOOTOBJS has to be loaded:

`kbset_us`	standard US keyboard set
`kbset_ger`	german keyboard set
`kbset_mgr`	keyboard set for MGR US layout
`kbset_mger`	keyboard set for MGR german layout

  Additionally, the driver needs a font module. Select one of the following fonts and load it to your module directory from CMDS/BOOTOBJS:

`font_9x18`	for 800 x 600 video resolution
`font_7x14`	for 640 x 480 video resolution
`font_11x20`	for 1024 x 768 or 1152 x 900 video resolution

  The resolution and refresh rate of the graphic interface are taken from RMon 's parameter RAM.

  This driver should be default.

i

See appendix A for a detailed description of the control sequence codes.

- **`sc8x36`**
  Centronics driver for CIO (z8536) on EUROCOM-17.
- **`sccd2401`**
  Driver for the four serial channels on the EUROCOM-17.

Table 1: Available SCF Descriptors and Appropriate Drivers

Descriptor	**Driver**	**Description**
term.rmon	scrmon	Console device depending on RMon setup
term.cons	sc17cons	Interrupt-driven driver for keyboard and graphics
term.t0	sccd2401	Serial channel 0 (RJ11 on front panel)
t0	sccd2401	CD2401 port A (RJ11 on front panel)
t1	sccd2401	CD2401 port B (CHAN.2 on CONV-300)
t2	sccd2401	CD2401 port C (CHAN.3 on CONV-300)
t3	sccd2401	CD2401 port D (CHAN.4 on CONV-300)
p1	sccd2401	CD2401 port B (for serial printers)
pp	sc8x36	Centronics port on z8536
cons	sc17cons	Same as term.cons

The sources of the descriptors may be found in directory MWOS/OS9/IO/SCF/DESC.

3.5 SBF Drivers and Descriptors

For each supported streamer there is a driver and an appropriate descriptor called **`mt0`**. The following streamers are supported:

Table 2: Available SBF Drivers and Descriptor Locations

Driver	**Directory for `mt0`**	**Type**
sbtandberg	TAND	TANDBERG 36xx
sbgiga	WANG	WangDAT 2600
sbviper	VIPER	ARCHIVE VIPER
sbgiga	EXABYTE	EXABYTE
sbteac	TEACMT2	TEAC MT-2
sbhpdat	HPDAT	HP DAT 35470A/35480A

The sources of the descriptors are located in directory MWOS/OS9/SRC/IO/SBF/DESC. All **`mt0`** descriptors are for SCSI ID 2.

i

The **`back`** *utility does not work with HP DAT tapes.*

3.6 PCF - The PC File Manager

Load the following modules into memory:

1. **`PCF`** - the PC file manager (located in CMDS/BOOTOBJS).
2. RBF driver **`rbteac/rbcteac`**, **`rbscfl/rbcscfl`** or **`rbvccs/rbcvccs`**.
3. The descriptor for your PC-DOS format.

   The following descriptors are available:

Name	File	Format
pcd0	SCFL/pcd0.360KB	40 tracks, 9 sectors, 5.25"
pcd0	SCFL/pcd0.720KB	80 tracks, 9 sectors, 3.5"
pcd0h	SCFL/pcd0h.120MB	80 tracks,15 sectors, 5.25"
pcd0h	SCFL/pcd0h.144MB	80 tracks,18 sectors, 3.5"
pcd0	TEACFC1/pcd0.720KB	80 tracks, 9 sectors, 3.5"
pcd0	TEACFC1/pcd0.144MB	80 tracks, 18 sectors, 3.5"
pch0	VCCS/pch0	Harddisk, SCSI ID 6, 512 B/sector
pch1	VCCS/pch1	Harddisk, SCSI ID 5, 512 B/sector
pch2	VCCS/pch2	Harddisk, SCSI ID 2, 512 B/sector
pch3	VCCS/pch3	Harddisk, SCSI ID 3, 512 B/sector
pch4	VCCS/pch4	Harddisk, SCSI ID 4, 512 B/sector

   SCFL and TEAC descriptors are also available for drive 1.

`pcd0.360KB` *enables double stepping on SCFL, so only a 80 track floppy drive can be used.*
Use the **`partdgen`** *utility to create a descriptor for partitioned harddisks (see chapter 1-12 in PCF documentation).*

Restrictions:

- The **`DCH`** disk cache can not be used on **`/pcxx`** devices.
- ELTEC's **`back`** utility works correctly for **`/pcxx`** as destination and file names restricted to eight characters plus three for extension.

4 Installing New RBF Devices

4.1 Installing New Harddisks with Embedded SCSI Controllers

Using the **`hx`** descriptor, the system will adapt automatically to the new harddisk's sector size. No further action is required.

If you wish to change the sector size, the harddisk has to be reformatted physically.

Use **`dmode`** utility for changing the sector from default value 0 (for embedded harddisks) to fixed new value:

```
$ dmode /hx ssize=512
$ dmode /hx format=on
```

This example changes the sector size to 512 B/sector and clears format inhibit bit. Now reformat the harddisk with the **`format`** command.

4.2 Fixed Device Parameters

Table 3: Used SCSI IDs on EUROCOM-17

SCSI-ID	LUN	Descriptor	Type
0		<none>	
1	0	d0	SCFL (ELTEC format)
	1	d1	SCFL (ELTEC format)
	0	d0h	SCFL (PC HD format)
	1	d1h	SCFL (PC HD format)
	0	s0	SCFL (ELTEC high density format)
	1	s1	SCFL (ELTEC high density format)
	0	u0	SCFL (universal format)
	1	u1	SCFL (universal format)
2	0	mt0	Streamer
		h2	Harddisk
3	0	d0	TEAC (ELTEC format)
	1	d1	TEAC (ELTEC format)
	0	d0h	TEAC (PC HD format)
	1	d1h	TEAC (PC HD format)
	0	s0	TEAC (high density format)
	1	s1	TEAC (high density format)
	0	u0	TEAC (universal format)
	1	u1	TEAC (universal format)
	0	h3	Harddisk
4	0	h4	Harddisk
5	0	h1	Harddisk
6	0	h0	Harddisk

5 Features and Enhancements

This chapter describes the changes in the ELTEC dependent part of the software since the V2.3 release.

5.1 Sysgo

The **`sysgo`**/**`systs`** modules are rewritten in C. **`Sysgo`** now uses the **`profile`** utility to proceed the **`startup`** file. As a consequence the environment variables for the initial **`shell`** may be set in the **`startup`** file. **`Sysgo`** also executes the script file /dd/SYS/**`.login_default`** to set the default environment.

5.2 Init

- System Identification -

Some software packages developed by ELTEC (OS9TCP, COMU-200) used to read the MainFram string (M$Instal) located in the **`init`** module in order to identify the CPU board. Problems arose, when customers began to change this string, so ELTEC had to find a solution to overcome potential problems: the **`init`** module contains a field named M$Site, which was not used in the past, but now holds a unique board identification code. This field must not be changed!!!

The SiteCode for the EUROCOM-17 is $45313700.

ELTEC's software packages now proceed by the following strategy:
The software reads the MainFram string, which defaults to 'ELTEC Eurocom X'. If the string has not been changed, it sufficiently identifies the board. If it has been changed, the software reads the 'Site' code to identify the board. This way there will be no problems in terms of compatibility with older versions of OS-9.

6 Additional Utilities

6.1 Dmode Utility

As an addition to the Microware utilities, ELTEC delivers this utility to examine or change RBF descriptors in runtime (like `xmode` for SCF descriptors).

- **Syntax:**
  `dmode [<opts>] /<device> [<parameters>] [<opts>]`

- **Options:**

-?	List usage

- **Parameters:** (prefix hex values with $)

drive=<n>	RBF logical drive number
type=hard|floppy	drive type
size=5|8	disk size (use 5 for 3.5)
dens=s|d	data density (single or double)
tk0dens=s|d	data density on track 0 (single or double)
heads=<n>	number of data surfaces
cyls=<n>	number of cylinders (including spares)
trkdens=s|d	track density (on floppies)
scttrk=<n>	physical sectors per track (including spares)
scttk0=<n>	sectors per track on track 0, if tk0dns=s
ssize=<n>	physical sector size
unit=<n>	unit number (used by controller)
ctlid=<n>	SCSI controller ID
step=<n>	step rate code
verify=on|off	verify by read after write
seg=<n>	minimum segment allocation size
ilv=<n>	physical interleave factor
toffs=<n>	track base offset
soffs=0|1	first physical sector on each track
format=on|off	enable/inhibit logical and physical formatting
multsct=on|off	enable/disable multi-sector transfers
autosiz=on|off	use/don't use SS_DSize GetStat call during format
trkfmt=on|off	enable/inhibit single track formatting
tries=<n>	number of attempts on read/write (1 = no retries)
wpc=<n>	first cylinder with write precompensation
rwr=<n>	first cylinder with reduced write current
park=<n>	cylinder to park heads on

lsnoffs=<n>	offset to first logical sector
disconn=on|off	enable/disable SCSI disconnect/reselect
sync=on|off	enable/disable synchronous transfer
maxcnt=<n>	max. transfer count (0 = default = 64K)

6.2 Back Utility

`Back` is ELTEC's special backup utility for general backup purposes.

o **Syntax:**
`back [<opts>] <source> [<destination>] [<opts>]`

o **Description:**
`Back` is used to backup/restore directory trees to/from disk or tape. `Back` may be used instead of `dsave` and `fsave`/`frestore`.

If the destination device is a disk, you may backup the directories either in OS-9 structure (like using the `dsave` utility) or in a streamer like structure called saveset, using a special raw mode, which works much faster.

The '-l' (list) and '-v' (verify) options are useful to manage the saveset. The '-z' (exclude), '-o' (include) and '-i' (date) options avoid useless copying, by defining special conditions for the files to be copied.

If there is no more space on the device, `back` will prompt for a new disk/tape to continue. The name of saveset's parts on the additional output volumes are labeled with a sequence number, which is not considered as a part of the name.

When using a streamer, `back` needs a device descriptor loaded into memory before operation. `Back` uses the `mt0` SBF descriptor as default tape descriptor.

The following streamer types are supported:

`TAND/mt0`	for the Tandberg 3620/40/60 Streamers
`EXABYTE/mt0`	for the ExaByte 8200
`TEACMT2/mt0`	for the Teac MT-2ST Streamer
`VIPER/mt0`	for the Archive Viper Tapes
`WANG/mt0`	for the WangDat 2600

Default destination device can be set with the **`shell`** environment parameter BACK_DEV, otherwise it is **`/mt0`**.

o **Caveats:**
If the source is a saveset, the destination must **not** be a saveset. If the output device runs out of free space during a saveset restore operation **`back`** cannot call for a new volume to continue.

There are problems with Tandberg TDC 3660 streamers with ROM Revision 4.00 and writing more than one saveset to tape. In some cases the streamer will hang.

The underscore character '_' is not allowed in saveset name.

Be sure that the specified volume size is smaller than or equal to volume capacity.

i

There are two different versions of **`back`**, *one for old ELTEC drivers and one for original Microware drivers. At this time Microware drivers are used with OS-9 V3.0 only.*

o **Options:**

-s=<filename>	Specifies source/destination saveset name for disk.
-t[=<filename>]	Specifies source/destination saveset name for tape. If no filename/device is specified, **`back`** uses the default tape device mt0 and the name save_1.
-i=<date>	Only files with a more recent date than specified in <date> are treated. Format of <date>: dd.mm.yy[-hh.mm]

-z[=<filename>]	Reads an exclude list. None of the files/ subdirectories in this list will be treated. Note that the full pathname of each file/directory is required. Wildcards ('*','?') are accepted. If <filename> is given, **`back`** will read the exclude list from <filename>, otherwise it will be read from the standard input path. Input from standard input path can be terminated by <ESC>.
-o[=<filename>]	Reads a select list. Only the files/subdirectories included in this list will be treated. For more details, see the '-z' option.
-f[=<filename>]	Formats destination device. <Filename> is the name of a command file, which is forked by **`back`**. This file has to contain a command line like: **`format /d0 -npnvnfr`**. The default command file is /dd/SYS/**`format.back.`** Tapes should always be erased prior writing first saveset.
-b=<num>	Allocates <num> KB of memory for copying. **`Back`** uses 100 KB by default.
-p	Asks before copying.
-q	Doesn't ask before copying (default).
-x	Debug mode
-l	Lists names of the files in the saveset. (Works only on saveset)
-v	Verifies the files in the saveset. (Works only on saveset)
-a	Writes a saveset in block mode.
-c	Fills last block to complete buffer size.
-k	Doesn't overwrite existing files (copy and rename first).

The following options only make sense if destination is not a saveset.

-r	Writes over existing destination file with same name without asking.

-n	Asks if existing destination file with same name shall be overwritten. (default)
-u	Update mode. Only sources with a more recent creation date than existing destination are treated. Add '-r' option if for automatic update.

The following options only work on tapes.

-w	Rewinds the tape before reading or writing. Use this if you are not sure about the saveset position on the tape. If this option is not specified, the saveset will be written behind the last saveset on tape.

-e=<volume_size> Specifies volume size for tape in KB.
Defaults are:
120 MB for Tandberg 36XX
2048 MB for ExaByte 8200
120 MB for Teac MT-2ST
120 MB for Archive Viper
3000 MB for WangDat 2600

o **Examples:**

a) Make backups on disks maintaining the OS-9 file structure:

```
$ back /h0/SOURCES  /d0/SOURCES.back   -r -u
```

`Back` copies all files and directories from /h0/SOURCES to /d0/SOURCES.back and writes over existing files with same names if they are older than the source files to copy ('-r' '-u'). The file structure of /d0/SOURCES.back will be the same as in /h0/SOURCES.

```
$ back /h0/SOURCES  /d0/SOURCES.back -f -i=17.10.86
```

`Back` formats the disk in /d0 ('-f'), using the /dd/SYS/**`format.back`** command file and then copy all files/directories which have a more recent date than 17.10.86 ('-i=') from /h0/SOURCES to /d0/SOURCES.back.

b) Make backups on disks using a saveset:

```
$ back /h0/SOURCES -s=/d0/savesource >/h0/backout&
```

`Back` copies all files from /h0/SOURCES into one saveset named savesource on disk. The saveset will be copied into the root directory of /d0. In this case `back` will work in the background and redirect standard output to /h0/backout.

```
$ back /h0/SOURCES  -s=/d0/savesource -z=exclude
```

The file `exclude` may contain the following lines:

```
"/h0/oldprog"
"/h0/PROGS/*.c"
"*/CMDS/*"
```

`Back` copies all files from /h0/SOURCES into savesource, except for /h0/oldprog, all C files in PROGS and the files in all CMDS subdirectories.

c) Make backups on tapes:

```
$ back /h0 -wft
```

`Back` copies all files from device /h0 into the saveset save_1 on streamer device mt0. Prior to writing to the tape, it will be erased. That is the way to backup to a tape without data on it.

```
$ back /h0/SOURCES -t=/h3/savesource
```

`Back` copies all files from /h0/SOURCES into the saveset on tape. The file is written to the end of data area.

```
$ back -t=/h3/savesource  /h1/SOURCES  -w
```

`Back` rewinds the tape ('-w') and restores the files on /h1/SOURCES. The original file structure will be rebuilt.

```
$ back /dd -t=/mt0/harddisk -wv
```

`Back` rewinds the tape and compares all files stored in saveset 'harddisk' with files on device /dd.

6.3 Rawcopy Utility

This utility copies a disk image from/to a file or disk. `Rawcopy` can be used to create a RAM-disk image for ROMbased OS-9 systems. The C-sources of this program can be found in directory MWOS/OS9/SRC/UTILS.

- **Syntax:**
  `rawcopy [<opts>] <source> <destination> [<opts>]`

- **Options:**

-b=<num>	Set buffer size to <num> (default is $2000)
-q	Quit mode
-nv	No verify

- **Caveats:**
  Iniz ever the source and target device descriptor or rawcopy will fail.

- **Examples:**
  a) Copy RAM-disk image to file `ramdisk`.

  ```
  $ rawcopy /r0 ramdisk
  ```

  b) Copy disk /d0 to /d1.

  ```
  $ rawcopy /d0 /d1
  ```

6.4 Eput Utility

To put a merged OS-9 bootfile into the EUROCOM-17's RAM, there is an utility called **`eput`**. This utility can also be used to generate a local RESET on the EUROCOM-17 with the '-r' option, to generate a local NMI with the '-a' option (abort) and to start the downloaded OS-9 with the '-s' option.

The '-h' option is used if the EUROCOM-17's VMEbus base address (extended and short I/O) is set with hex switch S1, located at the front panel. A special case of this option ('-h=0') is used to start the OS-9 on the local secondary CPU.

The C-sources of this program can be found in directory MWOS/OS9/SRC/UTILS.

- **Syntax:**
  `eput [<opts>] {<path> [<opts>]}`

- **Options:**

-d=<addr>	Set destination address (default: $C002.0000)
-h=<num>	Set address for hex switch position
-b=<num>	Set buffer size (default: $2.0000)
-r	Generate local RESET before download
-a	Generate Level 7 IRQ (abort)
-s[=<addr>]	Start program at loacal address <addr> (default: longword at destination address + 4)

- **Examples:**
  a) Download the file **`rommer`** to the VMEbus address $5002.0000 and start it using the longword at address $5002.0004.

```
$ eput -sh=5 cmds/bootobjs/nobug/rommer
```

  b) Move the file **`rombugger`** to the local address $0102.0000 and start the local secondary CPU.

```
$ eput -s -h=0 -d=0x01020000 cmds/bootobjs/romscpu/rombugger
```

6.5 Flash Programming Utility

This utility is a sample how to program the local or the Flash EPROM on the MMC1 memory module. **`flprg`** writes a complete binary image to the Flash EPROM using the **`systrap`** module (see Section 7.1 'The F$System System Call ' for details).

- **Syntax:**
  `flprg [<opts>] [<path>] [<opts>]`

- **Options:**

-a=<addr>	Set base address (default: $FE80.0000)
-e	Erase Flash EPROM
-i	Read Flash EPROM Identification Code
-q	Don't print header message

- **Example:**
  Write a ROMbased OS-9 to Flash EPROM on MMC1.

```
$ flprg -e  -a=0x0d000000 rombugger
```

7 Additional Libraries

7.1 The F$System System Call

The F$System system call has been added by ELTEC to provide board-hardware specific functions to the user. The functions are available to the members of group 0 only.

To install the F$System call, add the module **`systrap`** located in MWOS/OS9/68020/PORTS/E17/CMDS/BOOTOBJS to your bootlist, create a new bootfile with **`os9gen`** and reboot your system.

To keep the number of new system calls to a minimum, all ELTEC-specific functions are accessible through the F$System call. A function code is passed in register d0.w to indicate the operation desired. Specific parameters and functions of each system operation are discussed in the following sections. Actual values are resolved by linking with the library in directory MWOS/OS9/68000/LIB named **`libeltec.l`** for programs written in assembly language or **`clibeltec.l`** for programs written in C.

i

When the system comes up after booting, the **`SysTrap`** *module checks if it runs on the right hardware. It does this by analyzing the SiteCode located in the* **`init`** *module. For correct functioning this field must contain the unique board identification code of the CPU board, i.e. $45313700.*

7.2 The Assembler Library LIBELTEC

The following section contains the complete description of the functions included with the **`F$System`** system call:

`Sys$IOS`	Get mmu-protected I/O segment
`Sys$VMECCtl`	Enable/Disable caching of VMEbus read cycles
`Sys$DSCtrl0`	Data size control 0 (A32)
`Sys$DSCtrl1`	Data size control 1 (A24)
`Sys$ASCtrl0`	Select VMEbus AM source
`Sys$BlkDisp`	Enable/disable hex display
`Sys$EnSemIRQH`	Enable/disable semaphore interrupt at $7C
`Sys$SlavAddr`	Set VMEbus slave base address
`Sys$BlkMove`	VMEbus block transfer via VIC
`Sys$AlignPtr`	Align pointer to 256 byte boundary
`Sys$SetDisp`	Set digit of hex display
`Sys$GetSwt`	Get contents of hex switches

The default configuration after RESET is indicated by a (*) where appropriate.

Sys$IOS Get mmu-protected I/O segment

- **Input:**

d0.w =	0 (`Sys$IOS` function code)
d1.l =	1: request I/O segment
	0: return I/O segment
d2.l =	size of I/O segment
(a0) =	address of segment requested

- **Output:**
  none

- **Error Output:**

cc =	carry bit set
d1.w =	error code if error

- **Possible Errors:**
  E$Permit - you must belong to group 0 to use this function
  E$MemFull, E$NoRAM

- **Function:**
  `Sys$IOS` is used in systems equipped with a paged memory management unit (PMMU) and thereby using the system security module (`SSM`). This function enables group 0 user programs to perform memory mapped I/O, i.e. writing patterns into a video RAM located outside the process memory.

- **Cross Reference:**
  see F$Permit

Sys$VMECCtl Enable/disable caching of VMEbus read cycles

- **Input:**
  d0.w = 1 (`Sys$VMECCtl` function code)
  d1.l = 1: enables caching
  0: disables caching (*)
  -1: read status only

- **Output:**
  d0.l = status

- **Error Output:**
  cc = carry bit set
  d1.w = error code if error

- **Possible Errors:**
  E$Permit - you must belong to group 0 to use this function

- **Function:**
  `Sys$VMECCtl` enables or disables the cache for VMEbus longword read cycles. If d1.l equals -1, no action takes place. The status of this function is always returned in d0.

- **Note:**
  The VMEbus caching is allowed for aligned longword read cycles (A32, D32) only.

Sys$DSCtrl0 Data size control 0 (A32)

- **Input:**
  d0.w = 2 (`Sys$DSCtrl0` function code)
  d1.l = 1: A32/D16 transfers
  0: A32/D32 transfers (*)
  -1: read status only

- **Output:**
  d0.l = Status

- **Error Output:**
  cc = carry bit set
  d1.w = error code if error

- **Possible Errors:**
  E$Permit - you must belong to group 0 to use this function

- **Function:**
  `Sys$DSCtrl0` sets the data size on the VMEbus during master access at the address range $0040.0000 - $EFFF.FFFF.
  If d1.l equals -1, no action takes place. The status of this function is always returned in d0.

Sys$DSCtrl1 Data size control 1 (A24)

- **Input:**
  d0.w = 3 (`Sys$DSCtrl1` function code)
  d1.l = 1: A24/D32 transfers
  0: A24/D16 transfers (*)
  -1: read status only

- **Output:**
  d0.l = Status

- **Error Output:**
  cc = carry bit set
  d1.w = error code if error

- **Possible Errors:**
  E$Permit - you must belong to group 0 to use this function

- **Function:**
  `Sys$DSCtrl1` sets the data size on the VMEbus during master access at the address range $FF00.0000 - FFFE.FFFF.
  If d1.l equals -1, no action takes place. The status of this function is always returned in d0.

Sys$ASCtrl0 Select VMEbus AM source

- **Input:**
  d0.w = 4 (`Sys$ASCtrl0` function code)
  d1.l = 1: the AM source register of the VIC is used to generate the address modifier code on the VMEbus.
  0: extended AM code is generated for address range from $0040.0000 - $EFFF.FFFF, standard AM code at the address range $FF00.0000 - $FFFE.FFFF and a short AM code at addresses $FFFF.0000 - $FFFF.FFFF. (*)
  -1: read status only

- **Output:**
  d0.l = Status

- **Error Output:**
  cc = carry bit set
  d1.w = error code if error

- **Possible Errors:**
  E$Permit - you must belong to group 0 to use this function

- **Function:**
  If d1.l equals -1, no action takes place. The status of this function is always returned in d0.

Sys$BlkDisp Enable/disable hex display

- **Input:**
  d0.w = 6 (`Sys$BlkDisp` function code)
  d1.l = 1: enables hex display (*)
  0: disables hex display
  -1: read status only

- **Output:**
  d0.l = Status

- **Error Output:**
  cc = carry bit set
  d1.w = error code if error

- **Possible Errors:**
  E$Permit - you must belong to group 0 to use this function

- **Function:**
  `Sys$BlkDisp` controls the Blank input of the hex display at the front panel.
  If d1.l equals -1, no action takes place. The status of this function is always returned in d0.

Sys$EnSemIRQH Enable/disable semaphore interrupt at $7C

- **Input:**
  d0.w = 7 (`Sys$EnSemIRQH` function code)
  d1.l = 1: enables semaphore IRQ at address $7C (*)
  0: disables semaphore IRQ at address $7C
  -1: read status only

- **Output:**
  d0.l = Status

- **Error Output:**
  cc = carry bit set
  d1.w = error code if error

- **Possible Errors:**
  E$Permit - you must belong to group 0 to use this function

- **Function:**
  `Sys$EnSemIRQH` only controls the hardware to enable/disable the semaphore IRQ at address $7C. However, the user still is responsible for programming any associated port hardware and/or IRQ handlers. If d1.l equals -1, no action takes place. The status of this function is always returned in d0.

Sys$SlavAddr Set VMEbus slave base address

- **Input:**
  d0.w = 10 (`Sys$SlavAddr` function code)
  d1.l = VMEbus slave address for standard access
  d2.l = VMEbus slave address for extended access

- **Output:**
  none

- **Error Output:**
  cc = carry bit set
  d1.w = error code if error

- **Possible Errors:**
  E$Permit - you must belong to group 0 to use this function.
  E$Param - impossible address given

- **Function:**
  `Sys$SlavAddr` sets the VMEbus slave base address for both standard and extended addressing.

Sys$BlkMove VMEbus block transfer via VIC

- **Input:**

d0.w =	11 (`Sys$BlkMove` function code)
d1.l =	transfer length in bytes
d2.l =	bit 0 = 0: write to slave
	bit 0 = 1: read from slave
a0.l =	pointer to local buffer
a1.l =	pointer to target buffer

- **Output:**
  none

- **Error Output:**

cc =	carry bit set
d1.w =	error code if error

- **Possible Errors:**

E$Permit -	you must belong to group 0 to use this function
E$Param -	either of the given addresses is not aligned properly (see below)
E$BadSiz -	The transfer count is not divisible by 4.
E$BusErr -	A bus error occurred on local or VMEbus

- **Function:**
  This function initiates a block transfer between a VMEbus master and slave. Both master and slave MUST be supplied with a VIC068 chip. The data width for block transfers is 32-bit (longword) only, so the given transfer length must be a number divisible by four. To minimize software overhead, both pointers MUST be 256 byte aligned.

Sys$AlignPtr Align pointer to 256 byte boundary

- **Input:**
  d0.w = 12 (`Sys$AlignPtr` function code)
  d1.l = pointer to memory block

- **Output:**
  d0.l = the given pointer aligned to the next 256 byte boundary

- **Error Output:**
  cc = carry bit set
  d1.w = error code if error

- **Possible Errors:**
  E$Permit - you must belong to group 0 to use this function

- **Function:**
  Normally, a pointer to a memory block is returned by a 'Request Memory' function. It will be aligned in any way suitable for the operating system. Some applications (i.e. `Sys$BlkMove`) require a 256 byte alignment of all pointers. To do this, the user should issue a memory request of the amount needed PLUS 256 bytes used for the alignment. The pointer returned by the OS is then used by `Sys$AlignPtr`.

Sys$SetDisp Set digit of hex display

- **INPUT:**
  d0.w = 13 (`Sys$SetDisp` function code)
  d1.l = Digit for hex display
- **Output:**
  none
- **Error Output:**
  cc = carry bit set
  d1.w = error code if error
- **Possible Errors:**
  E$Permit - you must belong to group 0 to use this function
  E$Param - impossible value for hex display
- **Function:**
  `Sys$SetDisp` writes the value of d1.l into the boards hex display, which is located at the front panel. This function returns -1 if the board does not have a hex display.

Sys$GetSwt Get contents of hex switches

- **Input:**
  d0.l = 13 (`Sys$SetDisp` function code)
- **Output:**
  d0.l = contents of two hex switches
- **Error Output:**
  cc = carry bit set
  d1.w = error code if error
- **Possible Errors:**
  E$Permit - you must belong to group 0 to use this function
- **Function:**
  `Sys$GetSwt` reads the contents of the hex switches of the board, which are located at the front panel. The lower switch is located in the lower nibble of the long word.

7.3 The C Library CLIBELTEC

The following section contains complete description of the C functions included with the F$System system call:

`get_ios()`	Get mmu-proctected I/O segment
`vme_cctl()`	Enable/disable caching of VMEbus read cycles
`ds_cntrl0()`	Data size control 0 (A32)
`ds_cntrl1()`	Data size control 1 (A24)
`as_cntrl0()`	Select VMEbus AM source
`blk_disp()`	Enable/disable hex display
`en_sem_irq()`	Enable/disable semaphore interrupt at $7C
`slave_addr()`	Set VMEbus slave base address
`blk_move()`	VMEbus block transfer via VIC
`align_ptr()`	Align pointer to 256 byte boundary
`set_disp()`	Set the digit of the hex display
`get_swt()`	Get the contents of the hex switches

The default configuration after RESET is indicated by a (*) where appropriate.

get_ios()

Get mmu-protected I/O segment

o **Synopsis:**
```
int get_ios (cntrl, size, address)
int cntrl;              /* 1: request I/O segment */
                        /* 0: return I/O segment */
long size;              /* size of segment requested */
char *address;          /* ptr to segment beginning */
```

o **Usage:**
The `get_ios()` function is used in OS-9 systems protected by the system security module (`SSM`) to enable user programs to perform memory mapped I/O, i.e. accessing a video RAM located outside the process memory. If an error occurs, `get_ios()` returns -1 and the appropriate error code is placed in the global variable errno. If no error occurs, `get_ios()` returns zero.

o **See Also:**
F$System system call, F$Permit

vme_cctl()

Enable/disable caching of VMEbus READ cycles

- **Synopsis:**

```
int vme_cctl (cntrl)
int cntrl;            /*  1: enables caching */
                      /*  0: disables caching (*) */
                      /* -1: read status only */
```

- **Usage:**
  The `vme_cctl()` function controls the cache for VMEbus longword read cycles. This is allowed for aligned longword read cycles (A32, D32) only. If an error occurs, `vme_cctl()` returns -1 and the appropriate error code is placed in the global variable errno. If no error occurs, `vme_cctl()` returns the current status.

- **See Also:**
  F$System system call

ds_cntrl0()

Data size control 0 (A32)

- **Synopsis:**

```
int ds_cntrl0 (cntrl)
int cntrl;            /*  1: A32/D32 transfers (*) */
                      /*  0: A32/D16 transfers     */
                      /* -1: read status only      */
```

- **Usage:**
  The `ds_cntrl0()` function is used to control the data size on the VMEbus during master access at the address range from $0040.0000 - $EFFF.FFFF.
  If an error occurs, `ds_cntrl0()` returns -1 and the appropriate error code is placed in the global variable errno. If no error occurs, `ds_cntrl0()` returns the current status.

- **See Also:**
  F$System system call

ds_cntrl1()

Data size control 1 (A24)

- **Synopsis:**

```
int ds_cntrl1 (cntrl)
int cntrl;            /*  1: A24/D32 transfers     */
                      /*  0: A24/D16 transfers (*) */
                      /* -1: read status only      */
```

- **Usage:**
  The `ds_cntrl1()` function sets the data size on the VMEbus during master access at the address range from $FF00.0000 - $FFFE.FFFF.
  If an error occurs, `ds_cntrl1()` returns -1 and the appropriate error code is placed in the global variable errno. If no error occurs, `ds_cntrl1()` returns the current status.

- **See Also:**
  F$System system call

as_cntrl0()

Select VMEbus AM source

- **Synopsis:**

```
int as_cntrl0 (cntrl)
int cntrl;            /* see below */
```

- **Usage:**
  The `as_cntrl0()` function selects the source for generation of the AM code during VMEbus access.

cntrl =	1:	the AM source register of the VIC is used to generate the AM code on the VMEbus
cntrl =	0: (*)	extended AM code is generated for address range from $0040.0000 - $EFFF.FFFF, standard AM code at the address range $FF00.0000 - $FFFE.FFFF and short AM code at $FFFF.0000 - $FFFF.FFFF.
cntrl =	-1:	read status only

  If an error occurs, `as_cntrl0()` returns -1 and the appropriate error code is placed in the global variable errno. If no error occurs, `as_cntrl0()` returns the current status.

- **See Also:**
  F$System system call

blk_disp()

Enable/disable hex display

- **Synopsis:**

```
int blk_disp (cntrl)
int cntrl;            /*  1: enables hex display (*) */
                      /*  0: disables hex display    */
                      /* -1: read status only        */
```

- **Usage:**
  The `blk_disp()` function enables or disables the hex display at the front panel.
  If an error occurs, `blk_disp()` returns -1 and the appropriate error code is placed in the global variable errno. If no error occurs, `blk_disp()` returns the current status.

- **See Also:**
  F$System system call

en_sem_irq()

Enable/disable semaphore interrupt at $7C

- **Synopsis:**

```
int en_sem_irq (cntrl)
int cntrl;      /*  1: enables semaphore interrupt (*) */
                /*  0: disables semaphore interrupt */
                /* -1: read status only */
```

- **Usage:**
  The `en_sem_irq()` function enables or disables semaphore interrupts at address $7C.
  If an error occurs, `en_sem_irq()` returns -1 and the appropriate error code is placed in the global variable errno. If no error occurs, `en_sem_irq()` returns the current status.

- **See Also:**
  F$System system call

slave_addr()

Set VMEbus slave base address

- **Synopsis:**

```
int slave_addr (std_addr, ext_addr)
unsigned std_addr;  /* VMEbus addr. for standard access */
unsigned ext_addr;  /* VMEbus addr. for extended access */
```

- **Usage:**
  The `slave_addr()` function sets the VMEbus slave base address for both standard and extended addressing.

  If an error occurs, `slave_addr()` returns -1 and the appropriate error code is placed in the global variable errno. If no error occurs, `slave_addr()` returns zero.

- **See Also:**
  F$System system call

blk_move()

VMEbus block transfer via VIC

- **Synopsis:**

```
int blk_move (count, mode, mbuf, sbuf)

unsigned count;         /* Transfer length in bytes    */
short mode;             /* 0 = write to slave   */
                        /* 1 = read from slave   */
long *mbuf;             /* pointer to buffer on master */
long *sbuf;             /* pointer to buffer on slave  */
```

- **Usage:**
  The `blk_move()` function initiates a block transfer between a VMEbus master and slave. Both master and slave MUST be supplied with a VIC068 chip. The data width for block transfers is 32 bit (longword) only, so the given transfer length must be a number divisible by four. To minimize software overhead, both pointers MUST be 256 byte aligned. If an error occurs, `blk_move()` returns -1 and the appropriate error code is placed in the global variable errno. If no error occurs, `blk_move()` returns zero.

- **See Also:**
  F$System system call

align_ptr()

Align a given pointer to a 256 byte boundary

- **Synopsis:**

```
long *align_ptr (pointer)

long *pointer;
```

- **Usage:**
  Normally, a pointer to a memory block is returned by a 'Request Memory' function, e.g. `malloc()`. It will be aligned in any way suitable for the operating system. Some applications (i.e. `blk_move()`) require a 256-byte alignment of all pointers. To do this, the user should issue a memory request of the amount needed PLUS 256 bytes used for the alignment. The pointer returned by the OS is then used by `align_ptr()`.

- **See Also:**
  F$System system call

set_disp()

Set digit of hex display

- **Synopsis:**

```
int blk_move (digit)

unsigned digit;       /* Digit for hex display */
```

- **Usage:**
  This function writes the value of 'digit' into the boards hex display.

- **See Also:**
  F$System system call

get_swt()

Return the contents of the boards hex switches

- **Synopsis:**

```
int get_swt()
```

- **Usage:**
  Returns the contents of the boards hex switches, which are located at the front panel. The contents of the lower hex switch is located in the lower nibble of the return value.

- **See Also:**
  F$System system call

8 ROMbased OS-9

8.1 Getting Started

The EUROCOM-17 board support package contains all you need to configure an OS-9 to be downloaded to the EUROCOM-17 or burned into an EPROM.

There are ready-to-use makefiles provided to generate a ROMbased OS-9. Change the data directory to MWOS/OS9/68020/PORTS/E17 and enter the command

```
$ make -f=rombug.make
```

to generate a file containing the OS-9 for the EUROCOM-17.

Assuming hex switch S1 on the front panel of the EUROCOM-17 is set to position 'C' you may use the **`eput`** utility to download and start the ROMbased OS-9. After the command

```
eput -s cmds/bootobjs/rombug/rombugger
```

is entered, the following message should appear on the EUROCOM-17 console:

```
OS-9/68K System Bootstrap
Now searching memory ($00040000 - $0005FFFF) for an OS-9 Kernel...

An OS-9 kernel module was found at $00040000
A valid OS-9 bootfile was found.
OS-9/68K V3.0.2   ELTEC Eurocom 17 - 68040
sysgo: no system-device selected
OS-9:
```

Now the EUROCOM-17 runs under OS-9. To provide an application on the EUROCOM-17, add the filenames to the mergelist (**`rom_utilities.ml`** in MWOS/OS9/68020/PORTS/E17/BOOTLISTS).

To generate an OS-9 for the User EPROM some changes must be made in the following file:

```
MWOS/OS9/68020/PORTS/E17/rombug.make
```

1. Change the link address from $0002.0000 to $FEA0.0000. The link address is defined with the symbol 'LADDR'.

```
Download:      LADDR   = "LADDR=-r=00020000"
   EPROM:      LADDR   = "LADDR=-r=FEA00000"
```

2. Remove the definition of the symbol 'RAMLOAD'.

```
Download:      RAMLOAD  = "RAMLOAD=-aRAMLOAD"
   EPROM:      RAMLOAD  = #"RAMLOAD=-aRAMLOAD"
```

8.2 The Mergelists

Generally, a new ROMbased OS-9 for the EUROCOM-17 is created by simply merging the modules required for the desired application into one file using one of the following makefiles:

`rombug.make`	Generate a ROMbased OS-9 with ROM-debugger included
`rom.make`	Generate a ROMbased OS-9 without ROM-debugger
`romscpu.make`	Generate a ROMbased OS-9 with ROM-debugger included for the secondary CPU.

Once you have created your own mergelist file, to generate a ROMbased OS-9, type:

```
   $ make -f=rombug.make
or $ make -f=rom.make
or $ make -f=romscpu.make
```

Make sure that the size of your file does not exceed the size of the EPROM.

8.3 Supported Modifications

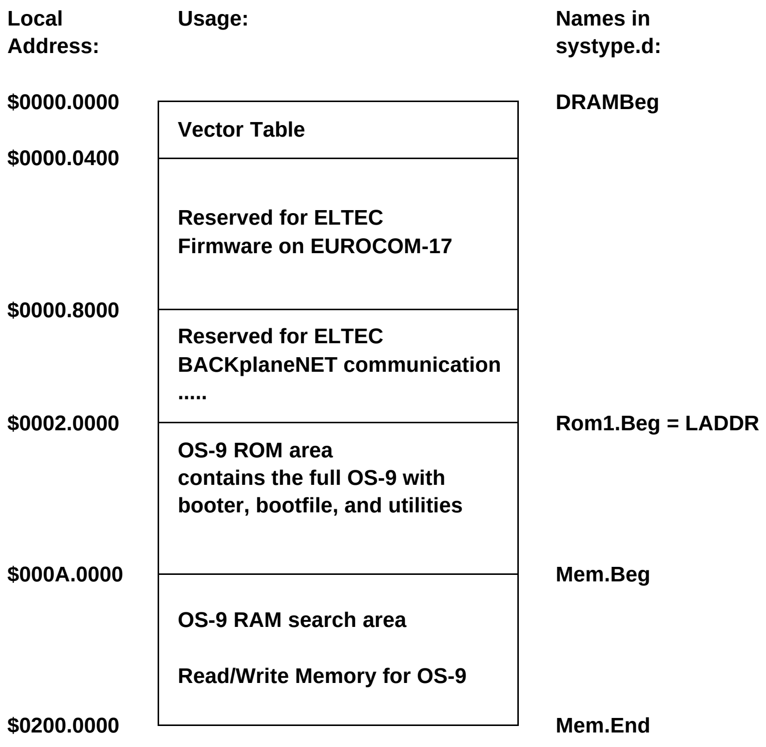
The EUROCOM-17 board support package contains all source files needed to completely reconfigure the system.

Yet, for most applications, only a few of all the possible modifications will be required. The following list gives an overview of the available modifications that were designed to be easy-to-do:

Item to modify	Files to change	Files to re-make	See
Memory layout	systype.d	rommer, rombugger file to prom	8.3.1
System level debugger available/not available	none	file to prom	8.3.2
Initial program to execute	sysgo.c	sysgo file to prom	8.3.3

8.3.1 Changing the Memory Layout

The following memory map is recommended for the OS-9 downloaded onto an EUROCOM-17 with 32 MB RAM (symbol 'RAMLOAD' defined in makefile and LADDR set to $0002.0000):


Local Address:
Usage:
Names in systype.d:
$0000.0000
Vector Table
DRAMBeg
$0000.0400
Reserved for ELTEC Firmware on EUROCOM-17
$0000.8000
Reserved for ELTEC BACKplaneNET communication .....
$0002.0000
Rom1.Beg = LADDR
OS-9 ROM area contains the full OS-9 with booter, bootfile, and utilities
$000A.0000
Mem.Beg
OS-9 RAM search area
Read/Write Memory for OS-9
$0200.0000
Mem.End

The following memory map is recommended for the OS-9 located within the User EPROM of an EUROCOM-17 with 32 MB RAM (definition of symbol 'RAMLOAD' removed with comment and LADDR set to $FEA0.0000):

Local Address:	Usage:	Names in systype.d:
$0000.0000	Vector Table	DRAMBeg
$0000.0400	Reserved for ELTEC Firmware on EUROCOM-17	
$0000.8000	Reserved for ELTEC BACKplaneNET communication	
$0002.0000	OS-9 RAM search area Read/Write Memory for OS-9	Mem.Beg
$0200.0000		Mem.End
$FEA0.0000	OS-9 ROM area contains the full OS-9 with booter, bootfile, and utilities	Rom1.Beg = LADDR
$FEA8.0000		

All memory areas declared as ROM are searched for valid OS-9 modules during system startup, all RAM is initialized (see OS-9 Technical Manual). The RAM between $8000 and $20000 is reserved for download purposes such as the nodes module used by BACKplaneNET.

8.3.2 Including/ Excluding the System Level Debugger

OS-9's system level debugger is a useful tool when developing software that shall execute in system state (i.e. drivers, system modules, file managers), but for ready-debugged applications it is usually obsolete. (The debugger increases the bootstrap size by about 70 kB). Therefore, two bootstraps are provided to activate/deactivate the debugger:

Capability	File in CMDS/BOOTOBJS
no debugger included	NOBUG/rommer
debugger included	ROMBUG/rombugger

To disable the debugger prompt at startup, use the RMon's setup command set enable the quiet mode.

8.3.3 Initial Program

The first program executed after reset is **`sysgo`**. **`Sysgo`** itself forks another program (which is usually **`shell`** to allow the user to enter commands). If the program terminates, **`sysgo`** immediately restarts it.

For dedicated applications, it is often desired to have **`sysgo`** execute an arbitrary program other than **`shell`**. In this case, the EUROCOM-17 immediately runs the desired program without any interaction and automatically restarts it if it terminates due to any reason.

For stand-alone applications without VMEbus, the default initial program name must be changed:
The initial program's name and (optional) command line arguments are defined as strings in the file **`sysgo.c`** by the macro **`SHELL_PARAMS`**.

After editing **`sysgo.c`** according to your requirements a new **`sysgo`** can be created by changing the current data directory to MWOS/OS9/SRC/SYSMODS/SYSGO and typing:

```
$ make sysgo
```

The output file will be named **`sysgo`** and will be created in MWOS/OS9/68020/CMDS/BOOTOBJS, overwriting any old version with the same name.

Appendix A: Control Sequence Codes

ANSI Standard Terminal Emulation

The **`sc17cons`** output character functions for the graphic interface emulates a subset of a standard ANSI X3.64 terminal.

The **`sc17cons`** displays 24 lines of 80 ASCII characters per line (default setting), with scrolling, (x, y) cursor addressability, and some other control functions. The non-blinking block cursor marks the current line and character position on the screen. When one of the ASCII characters between $20 (space) and $FF are written to the screen by calling the **`sc17cons`** (and the character is not part of an escape sequence), it is displayed at the current cursor position and the cursor moves one position to the right on the current line. If the cursor is already at the right edge of the screen, it moves to the first character position on the next line. If the cursor is already at the right edge of the screen on the bottom line, the screen is scrolls up by one line, before moving the cursor to the first character position on the next line.

Control Sequence Syntax

The **`sc17cons`** output function defines a number of control sequences which may occur in its input. When such a sequence is written to the **`sc17cons`** output function, it is not displayed on the screen, but effects some control function as described below.

Some of the control sequences consist of a single character. The notation
<CTRL>-x
for some character x, represents a control character.

Other ANSI control sequences are of the form
CSI <params> <char> or < ESC> [<params> char

Spaces are included only for readability. These characters must occur in the given sequence without the intervening spaces.

<ESC>	represents the ASCII escape character (<ESC>, <Ctrl>-[, $1B).
[	The next character is a left square bracket '[' ($5B).
<params>	are a sequence of zero or more decimal numbers made up of digits between 0 and 9, separated by semicolons.
<char>	represents a function character which is different for each control sequence.
CSI	represents the ANSI control sequence introducer ($9B).

'<ESC> [' and CSI are alternate representations of the ANSI 'Control Sequence Introducer' and may replace each other in any situation.

Some examples of syntactically valid escape sequences are:

<ESC> [m	select graphic rendition with default parameter
<ESC> [2 A	moves cursor 2 lines up
<ESC> [10;5 H	set cursor position

Supported Control Codes

- <CTRL>-H ($08) Backspace <BS>
  The cursor moves one position to the left on the current line. If it is already at the left edge of the screen, nothing happens.
- <CTRL>-J ($0A) Line-feed <LF>
  The cursor moves down one line, remaining at the same character position on the line. If the cursor is already at the bottom line, the screen scrolls up one line.
- <CTRL>-M ($0D) Return <CR>
  The cursor moves to the leftmost character position on the current line.

Supported ANSI Control Sequences

The syntax of the sequences follows the ANSI terminal standard, i.e. arguments are to be given as readable ASCII strings, using decimal notation, and are to be separated by semicolons. In the following arguments will be indicated by short names enclosed in angle brackets.

Printing characters in the range '@'..'~' are regarded as terminating codes. If they are defined in the following, they start processing the respective function. Undefined terminating codes simply abort the sequence without any action taken.

If a syntactical error is found within a sequence, the **`sc17cons`** output function skips all input until a terminating code is encountered, which results in a return to the normal not-in-sequence state.

- <ESC> [<n> A Cursor Up (CUU)
  Takes one parameter, <n> (default 1). Moves the cursor up <n> lines. If the cursor is fewer than <n> lines from the top of the screen, moves the cursor to the topmost line on the screen. The character position of the cursor on the line is unchanged.
- <ESC> [<n> B Cursor Down (CUD)
  Takes one parameter, <n> (default 1). Moves the cursor down <n> lines. If the cursor is fewer than <n> lines from the bottom of the screen, moves the cursor to the last line on the screen. The character position of the cursor on the line is unchanged.
- <ESC> [<n> C Cursor Forward (CUF)
  Takes one parameter, <n> (default 1). Moves the cursor right by <n> character positions on the current line. If the cursor is fewer than <n> positions from the right edge of the screen, moves the cursor to the rightmost position on the current line.
- <ESC [<n> D Cursor Backward (CUB)
  Takes one parameter, <n> (default 1). Moves the cursor left by <n> character positions on the current line. If the cursor is fewer than <n> positions from the left edge of the screen, moves the cursor to the leftmost position on the current line.
- <ESC> [<lin>;<col> H Cursor Position (CUP)
  Takes two parameters, <lin> and <col> (default 1, 1). Moves the cursor to the line <lin> and the character position <col>. Character positions are numbered from 1 at the left edge of the screen; line positions are numbered from 1 at the top of the screen.
- <ESC> [J Erase in Display (ED)
  Takes no parameter. Erases from the current cursor position inclusive to the end of the screen. The cursor position is unchanged.
- <ESC> [K Erase in Line (EL)
  Takes no parameters. Erases from the current cursor position inclusive to the end of the current line. The cursor position is unchanged.
- <ESC> [<n> L Insert Line (IL)
  Takes one parameter, <n> (default 1). Makes room for <n> new lines starting at the current line by scrolling down by <n> lines the portion of the screen from the current line inclusive to the bottom. The <n> new lines at the cursor are filled with spaces. The bottom <n> lines shift off the bottom of the screen and are lost. The position of the cursor on the screen is unchanged.

- <ESC> [<n> M Delete Line (DL)
  Takes one parameter, <n> (default 1). Delete <n> beginning with the current line. The portion of the screen from the current line inclusive to the bottom is scrolled upwards by <n> lines. The <n> new lines scrolling onto the bottom of the screen are filled with spaces. The <n> old lines beginning at the cursor line are deleted. The position of the cursor on the screen is unchanged.
- <ESC> [<sel> m Select Graphic Rendition (SGR)
  Takes one parameter, <sel> (default 0). Invokes the graphic rendition specified by the parameter. All following printing characters in the data stream are rendered according to the parameter until the next occurrence of this escape sequence. Only two graphic renditions are defined.

0	Normal rendition
7	Reverse video mode on

- <ESC> ? <sel> h
  Takes one parameter, <sel>. Turns on private feature specified by the parameter. Only one private feature is defined.
  25 Cursor On (visible)
- <ESC> ? <sel> l
  Takes one parameter, <sel>. Turns off private feature specified by the parameter. Only one private feature is defined.
  25 Cursor Off (invisible)

Appendix B: Used Interrupt Vectors

Vector numbers $1C to $1F and $40 to $4F are fixed due to VIC programming by the firmware. Do not change these vector numbers.

Example:

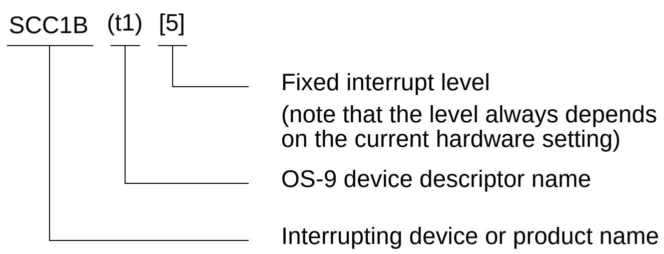

SCC1B (t1) [5]
Fixed interrupt level
(note that the level always depends on the current hardware setting)
OS-9 device descriptor name
Interrupting device or product name

Table B.1: Used Interrupt Vectors

Vector	Usage
$1C (28)	BACKNET (IRQ via VIC ICMS0) / Doubler
$1D (29)	BACKNET (IRQ via VIC ICMS1) / Doubler
$1F (31)	BACKNET (IRQ via VIC ICMS3) / Doubler
$40 (64)	VICBase
$41 (65)	Keyboard (term,cons)
$42 (66)	SystemTick
$43 (67)	SCSI (h0,d0,...)
$44 (68)	
$45 (69)	
$46 (70)	
$47 (71)	local ILACC
$48 (72)	
$49 (73)	
$4A (74)	
$4B (75)	
$4C (76)	
$4D (77)	
$4E (78)	
$4F (79)	

Table B.1: Used Interrupt Vectors (Continued)

Vector	Usage
$50 (80)	CD2401 ch0 (t0)
$51 (81)	CD2401 ch0 (t0)
$52 (82)	CD2401 ch0 (t0)
$53 (83)	CD2401 ch0 (t0)
$54 (84)	CD2401 ch1 (t1)
$55 (85)	CD2401 ch1 (t1)
$56 (86)	CD2401 ch1 (t1)
$57 (87)	CD2401 ch1 (t1)
$58 (88)	CD2401 ch2 (t2)
$59 (89)	CD2401 ch2 (t2)
$5A (90)	CD2401 ch2 (t2)
$5B (91)	CD2401 ch2 (t2)
$5C (92)	CD2401 ch3 (t3)
$5D (93)	CD2401 ch3 (t3)
$5E (94)	CD2401 ch3 (t3)
$5F (95)	CD2401 ch3 (t3)
$60 (96)	CIO2 (pp)
$61 (97)	
$62 (98)	
$63 (99)	
$64 (100)	WinDOS-RT - RFCdemon * cannot be changed
$65 (101)	WinDOS-RT - SCATV6 scat
$66 (102)	WinDOS-RT - SCATV5 scat
$67 (103)	WinDOS-RT - SCATV4 scat
$68 (104)	WinDOS-RT - SCATV3 scat /t2
$69 (105)	WinDOS-RT - SCATV2 scat /p2
$6A (106)	WinDOS-RT - SCATV1 scat
$6B (107)	WinDOS-RT - SCWDW16 scat /wdw16
$6C (108)	WinDOS-RT - SCWDW15 scat /wdw15
$6D (109)	WinDOS-RT - SCWDW14 scat /wdw14
$6E (110)	WinDOS-RT - SCWDW13 scat /wdw13
$6F (111)	WinDOS-RT - SCWDW12 scat /wdw12

Table B.1: Used Interrupt Vectors (Continued)

Vector	Usage				
$70 (112)	WinDOS-RT	- SCWDW11	scat	/wdw11	
$71 (113)	WinDOS-RT	- SCWDW10	scat	/wdw10	
$72 (114)	WinDOS-RT	- SCWDW9	scat	/wdw9	
$73 (115)	WinDOS-RT	- SCWDW8	scat	/wdw8	
$74 (116)	WinDOS-RT	- SCWDW7	scat	/wdw7	
$75 (117)	WinDOS-RT	- SCWDW6	scat	/wdw6	
$76 (118)	WinDOS-RT	- SCWDW5	scat	/wdw5	
$77 (119)	WinDOS-RT	- SCWDW4	scat	/wdw4	
$78 (120)	WinDOS-RT	- SCWDW3	scat	/wdw3	
$79 (121)	WinDOS-RT	- SCWDW2	scat	/wdw2	
$7A (122)	WinDOS-RT	- SCWDW1	scat	/wdw1, /term	
$7B (123)	WinDOS-RT	- SIGVECT	scat	/wdw1, /term	*cannot be changed
$7C (124)	WinDOS-RT	- SUPERV.	scat	/wdw1, /term	*cannot be changed
$7D (125)	WinDOS-RT	- HD	rbint13n	/h0, /h1, /h0p, /h1p	
$7E (126)	WinDOS-RT	- FLOPPY	rbint13n	/d0, /d1, /u0, /u1, /s0, /s1 ...	
$7F (127)	WinDOS-RT	- OS9SYS	rbrandat	/dd, /c0, /c1, /c2, /c3, /c4 ...	
$80 (128)					
$81 (129)					
$82 (130)					
$83 (131)					
$84 (132)					
$85 (133)					
$86 (134)					
$87 (135)					
$88 (136)					
$89 (137)					
$8A (138)					
$8B (139)					
$8C (140)					
$8D (141)					
$8E (142)					
$8F (143)					

Table B.1: Used Interrupt Vectors (Continued)

Vector	Usage	
$90 (144)		
$91 (145)		
$92 (146)		
$93 (147)		
$94 (148)		
$95 (149)		
$96 (150)		
$97 (151)		
$98 (152)		
$99 (153)		
$9A (154)		
$9B (155)		
$9C (156)		
$9D (157)	IPIN-100 on IPCA LEB1 IRQ level 1	
$9E (158)	IPIN-100 on IPCA LEB2 IRQ level 3	
$9F (159)	IPIN-100 on IPCA LEB3 IRQ level 5	
$A0 (160)	IPIN-300 SCC1B (t5)	IPIN-1100 CIO (pp1)
$A1 (161)	IPIN-300 SCC2B (t7)	
$A2 (162)	IPIN-300 SCC1B (t5)	
$A3 (163)	IPIN-300 SCC2B (t7)	
$A4 (164)	IPIN-300 SCC1B (t5)	
$A5 (165)	IPIN-300 SCC2B (t7)	
$A6 (166)	IPIN-300 SCC1B (t5)	
$A7 (167)	IPIN-300 SCC2B (t7)	
$A8 (168)	IPIN-300 SCC1A (t4)	IPIN-1100 CIO (pp2)
$A9 (169)	IPIN-300 SCC2A (t6)	
$AA (170)	IPIN-300 SCC1A (t4)	
$AB (171)	IPIN-300 SCC2A (t6)	
$AC (172)	IPIN-300 SCC1A (t4)	
$AD (173)	IPIN-300 SCC2A (t6)	
$AE (174)	IPIN-300 SCC1A (t4)	
$AF (175)	IPIN-300 SCC2A (t6)	

Table B.1: Used Interrupt Vectors (Continued)

Vector	Usage
$B0 (176)	IPIN-300 SCC3B (t9)
$B1 (177)	
$B2 (178)	IPIN-300 SCC3B (t9)
$B3 (179)	
$B4 (180)	IPIN-300 SCC3B (t9)
$B5 (181)	
$B6 (182)	IPIN-300 SCC3B (t9)
$B7 (183)	
$B8 (184)	IPIN-300 SCC3A (t8)
$B9 (185)	
$Ba (186)	IPIN-300 SCC3A (t8)
$BB (187)	
$BC (188)	IPIN-300 SCC3A (t8)
$BD (189)	
$BE (190)	IPIN-300 SCC3A (t8)
$BF (191)	
$C0 (192)	
$C1 (193)	
$C2 (194)	
$C3 (195)	
$C4 (196)	
$C5 (197)	
$C6 (198)	
$C7 (199)	
$C8 (200)	
$C9 (201)	
$CA (202)	
$CB (203)	
$CC (204)	
$CD (205)	
$CE (206)	
$CF (207)	

Table B.1: Used Interrupt Vectors (Continued)

Vector	Usage
$D0 (208) $D1 (209) $D2 (210) $D3 (211) $D4 (212) $D5 (213) $D6 (214) $D7 (215) $D8 (216) $D9 (217) $DA (218) $DB (219) $DC (220) $DD (221) $DE (222) $DF (223)	
$E0 (224) $E1 (225) $E2 (226) $E3 (227) $E4 (228) $E5 (229) $E6 (230) $E7 (231) $E8 (232) $E9 (233) $EA (234) $EB (235) $EC (236) $ED (237) $EE (238) $EF (239)	

Table B.1: Used Interrupt Vectors (Continued)

Vector	Usage
$F0 (240)	
$F1 (241)	
$F2 (242)	
$F3 (243)	
$F4 (244)	
$F5 (245)	
$F6 (246)	
$F7 (247)	
$F8 (248)	
$F9 (249)	
$FA (250)	
$FB (251)	
$FC (252)	
$FD (253)	
$FE (254)	
$FF (255)	

The local SCCs (Z8530) and also the SCCs located on the IPIN-300 use 8 vectors per SCC (one SCC contain two serial channels). Starting at the base vector (Bits 1 to 3 are zero) every second vector number is used. Changing the base vector in the device descriptor of both channels (i.e. t2 and t3 !!!) results in a change of all eight vectors.

The actual used vectors are described in the file `vectors.ELTEC` *which is located in MWOS/OS9/68020/PORTS/E17.*

Appendix C: Indivisible Cycle Operation

When the CPU performs a locked cycle to the '020 bus (e.g. TAS to the VMEbus) and someone wants to access the '040 bus from the '020 bus (e.g. slave access from VMEbus to local RAM or network activity) there is a deadlock situation. On normal reads or writes such a deadlock is resolved by sending a retry acknowledge to the CPU. On locked cycles this does not work because the arbiter does not grant the bus from the current bus master as long as /LOCK is active. Such deadlocks are resolved by sending an error acknowledge to the CPU, which produces a BUS ERROR exception. The user program has to install an exception handler which traps the BUS ERROR and retries the operation. The ULTRA-C library offers a function called **`_os9_strap()`** which can be used to install such a trap handler as follows:

```
#define tas(addr) _asm(" tas.b %0",__obj_modify(*addr));

/* service request initialization table */
_asm("ExcpIn:  dc.w  %0,BusErr-*-4", T_BUSERR << 2);
_asm("         dc.w  -1,-1");

/* pointer to service request initialization table */
extern void *ExcpIn;
os9strap  *in_tbl = (os9strap *)ExcpIn;

jmp_buf env;             /* program environment */

/* BUS-ERROR handler */
void BusErr()
{
    int err;/* return value from longjmp() */

    longjmp(env,err); /* make retry */
}

/* M A I N */
main()
{
    error_code error;    /* return value of _os9_strap */

    /* install trap handler */
    if ((error=_os9_strap (0, (os9strap *)in_tbl))!=0)  exit (errno);

    /* store program environment */
    setjmp(env);

    tas (0xc0100000);
}
```

A complete sample program called **`tasdemo.c`** is located in **`MWOS/OS9/SRC/UTILS`**.