ec5554 Mon Documentation

Version 1.00

Ingenieurbüro Edwin Würz
Im Burgfeld 4
D- 35781 Weilburg
Tel: ++49 6471 629 884; Fax: ++49 6471 629 885
info@wuerz-elektronik.com
http://www.wuerz-elektronik.com

Content

1. Overview 6
  1.1 History 6
2. System Configuration 6
  2.1 Clock configuration 6
  2.2 External bus configuration 6
  2.3 Memory configuration 6
    2.3.1 Internal memory configuration 6
    2.3.2 Memory controller 7
      2.3.2.1 CS0 external flash 7
      2.3.2.2 CS1 external sram 7
    2.3.3 Address-location overview 7
      2.3.3.1 EC555x Physical address-locations 7
      2.3.3.2 EC555x Logical address-locations 8
    2.3.4 System protection configuration 8
3. Boot up sequence 8
4. Menu-Operation 9
  4.1 Handling and display of a menu 9
  4.2 Menu-Entries 9
    4.2.1 EC555x Main-Menu 9
    4.2.2 Tools List 10
    4.2.3 Ram-tool 10
      4.2.3.1 Show external ramsize 10
      4.2.3.2 Fill memory 10
      4.2.3.3 Find first differences 11
      4.2.3.4 Dump memory 11
      4.2.3.5 Summ memory 11
      4.2.3.6 Read functions 11
      4.2.3.7 Write functions 11
      4.2.3.8 Write and verify functions 11
    4.2.4 CPU-Tool menu 11
      4.2.4.1 Disassemble 12
      4.2.4.2 Assemble 12
      4.2.4.3 Call (addr) 12
      4.2.4.4 Display Regs 12
      4.2.4.5 Set Regs 12
      4.2.4.6 Dump Memory (start, len) 12
      4.2.4.7 Fill Memory (start,cnt,data) 12
    4.2.5 Devices 12
      4.2.5.1 Show available devices 13
      4.2.5.2 Show used interrupts 13
    4.2.6 Sci 13
      4.2.6.1 Select baudrate 13
      4.2.6.2 Open/Close sci-2 13
      4.2.6.3 Send text to serial line 13
    4.2.7 Internal Flash commands 13
      4.2.7.1 Show size 13
      4.2.7.2 Check empty (off, cnt) 13
      4.2.7.3 Show Sector locations 13
      4.2.7.4 Program data to internal flash 14
      4.2.7.5 Sector erase 14

4.2.7.6 Erase complete flash 14
4.2.6.18 Program mon to flash 14
4.2.8 External Flash commands 14
4.2.8.1 Read ID 14
4.2.8.2 Check empty 14
4.2.8.3 Show sector locations 15
4.2.8.4 Program data to external flash 15
4.2.8.5 Sector erase 15
4.2.8.6 Chip erase 15
4.2.6.19 Program mon to flash 15
4.2.8.7 Show flash-config 15
4.2.9 FlexCan Tools 15
4.2.9.1 Select can-controller 16
4.2.9.2 Init 16
4.2.9.3 Reset 16
4.2.9.4 Show state 16
4.2.9.5 Config Rx-mask 16
4.2.9.6 Prepare Rx (id) 16
4.2.9.7 Transmit (id,data) 16
4.2.9.8 Print Buffer (dev,buf-id) 16
4.2.9.9 Print buffer 16
4.2.10 EQAdc 16
4.2.10.1 Init eQAdc 17
4.2.10.2 Reset eQAdc 17
4.2.10.3 Read from pin 17
4.2.10.4 Show state of qAdc 17
4.2.10.5 Show AD-Register 17
4.2.11 Eth Tools 17
4.2.11.1 Prepare 17
4.2.11.2 Show version 17
4.2.11.3 Show state 17
4.2.11.4 Show phy-infos 18
4.2.11.5 Show phy-state 18
4.2.11.6 Send bootp 18
4.2.11.7 Show rx-data 18
4.2.12 Test commands 18
4.2.12.1 All Ram-Tests/Ram-Tests submenu 18
4.2.12.1.1 Data Bus Test 18
4.2.12.1.2 Address Bus Test 18
4.2.12.1.3 Short ram-tests 19
4.2.12.1.4 Long burst-ramtest 19
4.2.12.2 Write 0/0xffffffff pattern 19
4.2.12.3 All MPC5554 tests / MPC5554-Test-submenu 19
4.2.12.4 Test internal SRAM 19
4.2.12.5 External Flash Test 19
4.2.12.6 Internal Flash Test 20
4.2.12.7 Test SCI-2 20
4.2.13 Configuration Menu 20
4.2.13.1 Cache & Burst on 20
4.2.14 CPU-Cfg-Menu 20
4.2.14.1 Do Test-Loop 20

4.2.14.2 Get cache-state 20
4.2.14.3 Invalidate cache 20
4.2.14.4 Instruction-cache 21
4.2.14.5 data cache 21
4.2.15 Sci 21
4.2.16 MemCtrl Menu 21
4.2.16.1 Show config 21
4.2.16.2 Show flash-config 21
4.2.16.3 Switch bus-clock 21
4.2.16.4 Switch flash-burst 21
4.2.16.5 Switch sram-burst 21

1 . Overview

The EC555xMon is a small monitor which is used to show the basic operations for the MPC555x processor and the peripherals of the EC555x board. The monitor will first initialize the processor module then the memory interface and will support some functions for testing and configuration.

1.1 History

Vers.	Date	Change
1.00	2006-05-08	First final release

2 . System Configuration

The EC555xMon will initialize the MPC555x-board, this will include the MPC555x processor, the external devices and clock setting.

2.1 Clock configuration

The EC555x board uses a 8 MHz crystal as clock source, the monitor will configure the internal system pll (spll) to multiply this frequency to the final frequency (MPC555x: 8MHz / 2 * 33 = 132 MHz), the extern bus will operate with the half system-frequency (66 MHz).

The powerpc timebase/decrementer will work with the system-clock 132MHz, this will result in a resulution of ~7,6 ns.

2.2 External bus configuration

The monitor will configure the external bus to operate in 32-Bit mode, and will activate the following bus-pins:

1. All datalines (d0..31)
2. All addresslines (a8..a31)
3. Control-lines: RD/WR, BDIP, WE0/1/2/3, OE, TS, TA, TEA

2.3 Memory configuration

With the MPC555x it is possible to connect peripherals in three different types. First the internal peripheral (which is mapped by an internal memory-map-register) supported by Freescale. Then there is a memory controller which supports up to four chip-select lines. These are used for external devices like the SRAM (chip-select 1) and the FLASH (chip-select 0) on the EC555x board. Finally it is possible to connect additional peripherals over the MPC555x bus-interface. The chip-select 2 is currently not used by the monitor, the chip-select 3 is used by the monitor to check for an available external ethernet controller.

The location of the different peripherals are assigned by the monitor to fixed physical-addresses. The powerpc core will move this addresses to different logical addresses with the memory-management-unit (MMU).

2.3.1 Internal memory configuration

The internal peripherals of the MPC555x have four different regions (fixed assigned by Freescale):

1. The internal flash (Address: 0x00000000 - 0x001FFFFF)

2. Internal SRAM (Address: 0x40000000 - 0x4000FFFF)
3. Peripheral Bridge A (Address: 0xC3F80000 - 0xC3FFFFFF)
4. Peripheral Bridge B (Address: 0xFFF00000 - 0xFFFFFFFF)

The address locations could be reassigned to different logical areas using the MMU please see chapter 2.2.4 Address-location overview, where you can find the locations where the monitor mapps the resources.

2.3.2 Memory controller

The monitor use only chip-select 0 (external flash) and cs1 (external sram) of the MPC555x memory controller. Chip-select 2 and 3 are available for additional applications (on some evaluation boards an external ethernet controller is connected to CS3).

2.3.2.1 CS0 external flash

Title\Board	***EC5554***
Wait-states / Timing	4 (no burst)
Burst	6-1-1-1-1-1-1-1
Accesstime	6 clocks = 91 ns
Burst access (8 beats)	13 clocks = 197 ns
Buswidth	32

2.3.2.2 CS1 external sram

Title\Board	***EC5554***
Wait-states / Timing	0
Burst	2-1-1-1
Accesstime	2 clocks = 30 ns
Burst access (4 beats)	5 clocks = 76 ns
Buswidth	32

2.3.3 Address-location overview

The physical addresses of the internal peripherie is fixed assigned by freescale, but the core supports a remapping using the memory-management-unit (MMU).

2.3.3.1 EC555x Physical address-locations

	Size upto	***Physical configuration***	
Internal flash	2 MB	0x00000000	0x001FFFFF
CS0: external flash	16 MB	0x20000000	0x20FFFFFF
CS1: external sram	16 MB	0x24000000	0x24FFFFFF
CS2: not - used	16 MB	0x28000000	0x28FFFFFF
CS3: not - used	16 MB	0x2C000000	0x2CFFFFFF
Internal SRAM	64 kB	0x40000000	0x4000FFFF
Peripheral Bridge A	1 MB	0xC3F00000	0xC3FFFFFF
Peripheral Bridge B	1 MB	0xFFF00000	0xFFFFFFFF

The locations of the internal peripherals is fixed. The address of the chip-select could be assigned in the range 0x20000000 to 0x3fff8000.

2.3.3.2 EC555x Logical address-locations

Device	**Size**	**Location**	
		None-cachable	**Cachable**
Boot-device (int. flash, ext. flash, ext sram) *)	16 MB	-	0x00000000
CS0: External flash	16 MB	0x20000000	0x30000000
CS1: External sram	16 MB	0x24000000	0x34000000
CS2: unused	16 MB	0x28000000	0x38000000
CS3: unused (Evalboard ethernet-ctrl)	16 MB	0x2c000000	0x3c000000
Internal sram	64 kB	0x40000000	0x50000000
Internal flash	16 MB	0x60000000	0x70000000
Internal peripheral bridge A	1 MB	0xC3F00000	-
Internal peripheral bridge B	1 MB	0xFFF00000	-

*) Dependent on the boot-configuration, the corresponding device (ext/int flash, sram) will be additional mapped by the MMU to the boot-location (0).

2.3.4 System protection configuration

The MPC555x has some protection-features implemented, like a watchdog and the bus monitor timeout counter. The watchdog is disabled by the boot-configuration word.

The bus monitor is not changed, and will stay at the reset-default (enabled with 2042 external bus-clocks ~30 usec / 66MHz).

3 . Boot up sequence

The EC555xMon will first initialize the MPC555x processor (clock, interrupts, ...) and the external peripherals, like flash and sram. Each step will be printed out on the first serial line (eSCI_A), on the second SCI (eSCI_B) there will be a message that it is the wrong line).

Configuration for the serial line:

- Baudrate: 9600
- Parity: none
- Data-bits: 8
- Stop-bits: 1

The startup message looks like this:

```
 1: Welcome to the EC5554-Board
 2: ===========================
 3: (c) by Wuerz Elektronik
 4: e-mail: info@wuerz-elektronik.com
 5: http://www.wuerz-elektronik.com
 6: EC5554Mon Version 0.08
 7: Compile-date: May  1 2006, time: 01:24:48
 8: Sys-clock = 132 MHz
 9: Bus-clock = 66 MHz
10: Detected external ramsize: 0x00400000 bytes (4194304)
11: External flash0: Spansion S29CD-016G (1) (2 MBytes)
12: External flash1: Spansion S29CD-016G (1) (2 MBytes)
13: Booting from external flash
14: Enable caches and bursts
```

Line 1-7:	Startup-message (version, date of monitor)
Line 8-9:	Used clocks
Line 10-12:	Detected external memories
Line 13-..:	Booting messages

After this message the monitor will display the ec555x-menu. (see chapter 4.).

```
--- Main-Menu --    {help = ?}
===============================================================================
```

4 . Menu-Operation

The monitor uses an easy to handle menu-system to handle the monitor commands. The menu is able to handle new commands when the prompt character ':' is displayed.

4.1 Handling and display of a menu

In the menu-system you have two types of entries, first a command and second a sub-menu. You can recognize a sub-menu with the text „--> next menu " at the end of a line. To select a command or change to a sub-menu press only the key which is shown in the first column followed by a [CR]. It is possible to process multiple commands in one line (e.g. 'ABC [CR]' the commands ABC are executed), but its not possible to go into multiple sub-menus! It is not allowed to have spaces between the commands, because after the first space the rest of the line is used as parameters. You can start each command multiple times, if you add a decimal number before the command list. If you choose 0, the command-list is handled until you press the [ESC] key.

Here is a short list of possibilities to execute commands (not sub-menus):

Execute one command (A):	*A[CR]*	
Execute one command 3 times (A):	*3A[CR]*	
Execute one command endless (A):	*0A[CR]*	*-- press [ESC] to abort --*
Execute two commands 3 times (AC):	*3AC[CR]*	

Each command could have a parameter, this is signaled by two brackets after the command-name.

```
B) Addr Bus Test (addr,len)
```

In this case we have two parameters addr and len. If you enter no parameter, by only press B[CR] the command could select a default. In this special case the complete external ram will do an address-bus test, since a board could have different sizes of rams a constant default parameter is not that powerful.
A function could also have a default parameter:

```
D) Applications (erom,irom,ram)              110
```

In this case '110' means that the function will use data '110' if you do not enter an own parameter. All text which follows the first space-character is interpreted as parameters. So you could call this command „Applications" with a parameter of '100' by typing „A [SPACE] 100 [CR].
If you select multiple commands all commands will get the same parameter.
After each command you get a response (minimum OK or ERROR).

4.2 Menu-Entries

Now let us have a look at all commands in the EC555x Mon:

4.2.1 EC555x Main-Menu

This is the base menu of the EC555xMon.

```
-- Main-Menu --    {help = ?}
===============================================================================
A) Tools                                      --> sub menu
```

```
B) Tests                                              --> sub menu
C) Config                                             --> sub menu
.) Quit
```

Points A to C are all submenus which we will see later.
Attention, if you press '.' (== Quit) in this menu the monitor will do a reboot, because there is nothing else to do!

4.2.2 Tools List

In the tools-menu you find some additional submenus to access some support tools.

```
-- Tool-Menu --    {help = ?}
==============================================================================
A) Ram-Access                                         --> sub menu
B) CPU-Menu                                           --> sub menu
C) Devices                                            --> sub menu
D) Sci                                                --> sub menu
E) Internal Flash                                     --> sub menu
F) External Flash                                     --> sub menu
G) FlexCan                                            --> sub menu
H) eQAdc                                              --> sub menu
I) Ext. Ethernet                                      --> sub menu
```

– RamAcc:	dump, reading, writing, ... memory-locations
– CPU-Menu	a small lineassembler/disassembler
– Devices	Show currently assigned devices and used interrupts
– Sci	Configure baudrate, write text to 2. SCI
– Internal Flash:	programming, erasing, ...
– External Flash:	programming, erasing, ...
– FlexCan:	init, transmit, ...
– eQAdc	Read analog input signals
– Eth Tools:	init, show state send/receive bootp

For a detailed description please refer to the corresponding chapters.

4.2.3 Ram-tool

With the ram-tools you have the possibility to examine any valid location in the system. You can also modify all memory-mapped registers of the MPC555x (like ram or internal registers).

```
-- RamTool-Menu --    {help = ?}
=====================================================
:

A) Show external ramsize
B) Fill memory (addr,cnt,val)                         0 0 0
C) Find 1. diff (addr1,addr2)                         0 1
D) Dump memory (addr,cnt)
E) Summ memory (addr,cnt)
F) Read  8bit (addr)
G) Read 16bit (addr)
H) Read 32bit (addr)
I) Write 8bit (addr,val)
J) Write 16bit (addr,val)
K) Write 32bit (addr,val)
L) VWrite 8bit (addr,val)
M) VWrite 16bit (addr,val)
N) VWrite 32bit (addr,val)
.) Quit
```

4.2.3.1 Show external ramsize

Show the corrent amount of available external sram.

4.2.3.2 Fill memory

The fill-memory can be used to program HEX values (only hex-data) to the memory. This function has three parameters:

start = start-location to program (hex/dec)

cnt = count of repetition of the data (hex/dec)

data = is a byte-array (ONLY IN HEX - no 0x allowed)

```
Example program Data: 0x12, 0x34, 0x88 to location 0x20000 3 times:
E 0x20000 3 12 34 88 [ENTER]

if you make a dump you to 0x2000 you will get this result:
0x00020000 - 12 34 88 12 34 88 12 34 88 xx xx xx xx xx xx xx ????????????????
```

It is very important to have a blank between the command and the parameter, after the parameter there is no need for a blank (it's only used for better readability).

4.2.3.3 Find first differences

This function finds the first differences between source 1 and source 2. ATTENTION if the complete device is identical you will get an invalid access to an unassigned location --> RESET!

```
Example: find first difference between ram (0x80000000) and external flash (0x00000000)
G 0x80000000 0 [ENTER]
Example 2: find first not empty cell of the flash
G 0 1 [ENTER]               * the first cell should be 0xff!
```

It is very important to have a blank between the command and the parameter, after the parameter there is no need for a blank (it's only used for better readability).

4.2.3.4 Dump memory

Dump memory makes a hex-dump to the terminal of any valid address-location of the EC555x board. The default writes the address location 0 to 16 to the terminal. You can enter either decimal or hex address / count values.

```
Example: Dump address 0xc0000000 (256 Bytes)
D 0xc0000000 256 [ENTER]
```

It is very important to have a blank between the command and the parameter, after the parameter there is no need for a blank (it's only used for better readability).

4.2.3.5 Summ memory

Function to access a range of memory. The function has the main-task to access a larger range of memory to get a sequence of burst-accesses to memory ranges.

4.2.3.6 Read functions

The read-functions are used to read a 8-bit value (h), 16-bit (i) and 32-bit (j). You get the result in hex and decimal. You have to provide an address (default would be zero).

4.2.3.7 Write functions

The write-functions are used to write a 8-bit value (k), 16-bit (l) and 32-bit (m). You have to provide the address plus the data to write. There is no verification of the written data.

4.2.3.8 Write and verify functions

The write-functions are used to write a 8-bit value (n), 16-bit (o) and 32-bit (p). You have to provide the address plus the data to write. The written data is verified.

4.2.4 CPU-Tool menu

This is only a small tool to write short assembly sequences for test purposes.

```
A) Disassemble (start,cnt)
B) Assemble
C) call (addr)
D) Display Regs
E) Set Reg (reg, val)
F) Dump memory (start,len)
G) Fill Memory (start,cnt,data)
.) Quit
```

4.2.4.1 Disassemble

Here you can disassemble code, but it will show only the „normal“ opcodes, not the „simplified“! You have to give an address and a cnt (count of opcodes). Default would be one opcode at locaion 0.

```
Example: Disassemble Reset-Handler
A 0x100 0x40 [ENTER]
```

4.2.4.2 Assemble

Here you can try your powerpc knowledge. Attention this line-assembler only knows the primary-opcodes (not the simplified). Also register-Names are not known (lr, ...). You can quit the line-assembler by typing a dot '.'. If you enter an empty line, the lineassembler uses the previous opcode, and advance to the next location.

```
Example: Assemble at location 0x80000
B 0x80000 [ENTER]
Enter empty line to exit!

0x00080000->0x7cf202a6 = mfspr   r7,0x12    # 18 = dsisr
```

4.2.4.3 Call (addr)

Here you can try your code, before the code is started, the monitor will switch to the register-configuration, which could be shown with point D. One exception is, that the r1 could not be changed, since this is the system-stack-pointer, and that is required for the Interrupt handling. The called function could return to the monitor by a normal return instruction (bclr 20,0).

ATTENTION: if you overwrite the stack pointer (r1) then an interrupt-handler will write to this new location. If there is an invalid pointer in r1 the system will crash (reboot)!

4.2.4.4 Display Regs

Here the register-content which is used for the call, is displayed (only r0..r31, cr, xer, ctr).

4.2.4.5 Set Regs

The registers which are used in the call-operation could be configured here. You have to provide the register-name (r00) and the new content.

```
E r03 0x12345678                                 ; new value of r03 = 0x12345678
```

4.2.4.6 Dump Memory (start, len)

Please see chapter 4.2.3.4.

4.2.4.7 Fill Memory (start,cnt,data)

Please see chapter 4.2.3.5.

4.2.5 Devices

Show currently available io-device's for the monitor (only informational).

```
-- DevTool-Menu --    {help = ?}
==============================================================================
A) Show available devices
B) Show used interrupts
```

4.2.5.1 Show available devices

Display the currently available devices for the io-system.

4.2.5.2 Show used interrupts

Show the currently assigned interrupts.

4.2.6 Sci

Change baudrate and make some output to second serial line.

```
-- Sci-Menu --    {help = ?}
==============================================================================
A) Select baudrate (sci baudrate)             0 9600
B) Open/Close sci-2                           0
C) Send text to sci (sci text)                1 Hello World!
```

4.2.6.1 Select baudrate

Change baudrate from serial line. First parameter is the serial line (0,1) the second the baudrate.

```
Example: a 0 115200 [ENTER]
```

4.2.6.2 Open/Close sci-2

Use to open the second serial line which could be used by the commands below.

```
Example: b 1 [ENTER]                        OPEN
```

4.2.6.3 Send text to serial line

The first parameter is the index of the serial line (0 = sci1, 1=sci2), all additional parameters will be send to the serial line as text.

```
Example: c 1 This is a MPC5554[ENTER]
```

4.2.7 Internal Flash commands

With this library you can modify the internal mpc555x flash, like erase, program, ...)

```
-- IntFlashTool-Menu --    {help = ?}
==============================================================================
A) show size
B) Check empty (off,cnt)
C) Show sector lcoations
D) Program data (src off cnt)
E) Sector erase(off cnt)
F) Chip Erase (id)
G) Program mon to flash
H) Switch mon to ram
```

4.2.7.1 Show size

This will print the current size of the mounted internal flash.

4.2.7.2 Check empty (off, cnt)

With this feature you can check if the Flash is empty at the specified locations (start-address bytecount). It will run too, if you work currently with dual-mapping (internal monitor works from the external ram).

4.2.7.3 Show Sector locations

Show the offsets of all sectors from this flash.

4.2.7.4 Program data to internal flash

With this feature you can program any data, which is available for the cpu, to the internal flash, we have three parameters, first the location in the logical address-space, second the offset from the internal flash-start and last the count of bytes to write.

```
To program the 96k from location 0x24000000 to the internal flash offset 0x100000
C 0x24000000 0x100000 0x18000 [ENTER]
```

4.2.7.5 Sector erase

This function erases one or more blocks of the flash. This call needs two parameters, the first is the offset from flash-start and the second is the count of bytes to erase.

```
To erase 300 kByte from offset 0x80000:
E 0x80000 0x4b000 [ENTER]
```

ATTENTION: The erase operation will always be done on sector-borders. If you select only the one byte inside a sector, the complete sector is erased (no backup/restore will be done).

4.2.7.6 Erase complete flash

The complete internal flash will be erased.

4.2.6.18 Program mon to flash

When using a monitor running in a different device (like sram or external flash), the monitor could be programmed (copied) to the internal flash.

4.2.6.17 Switch monitor to ram

If you have to erase/program the internal flash you need to switch the monitor to ram first. The basic operation is that the monitor is copied to ram and the mapping of the MMU will be changed to execute the monitor from external sram.

ATTENTION: If your monitor runs in internal flash this version could be lost when you erase/program the area of the monitor.

4.2.8 External Flash commands

With this library you can modify a mounted external flash!

```
-- ExtFlashTool-Menu --    {help = ?}
==============================================================================
A) Read ID
B) Check empty (off,cnt)
C) Show sector lcoations
D) Program data (src off cnt)
E) Erase Sector (off cnt)
F) Chip Erase (id)
G) Reset (id)
H) Program mon to flash
I) Switch mon to ram
```

```
J) Show flash-config
```

4.2.8.1 *Read ID*

This command will read the manufacturer-ID and device-ID from the external flash and show the result.

4.2.8.2 *Check empty*

Check the external-flash area if it is empty.

4.2.8.3 *Show sector locations*

This function will print all currently available sectors, there locations and size

4.2.8.4 *Program data to external flash*

With this feature you can program any data, which is available for the cpu, to an external flash, we have three parameters, first the location in the logical address-space, second the offset from the internal flash-start and last the count of bytes to write.

```
To program the 96k from location 0x24000000 to the external flash offset 0x100000
C 0x24000000 0x100000 0x18000 [ENTER]
```

4.2.8.5 *Sector erase*

This function erases one or more blocks of the flash. This call needs two parameters, the first is the offset from flash-start and the second is the count of bytes to erase.

```
To erase 300 kByte from offset 0x80000:
E 0x80000 0x4b000 [ENTER]
```

ATTENTION: The erase operation will always be done on sector-borders. If you select only the one byte inside a sector, the complete sector is erased (no backup/restore will be done).

4.2.8.6 *Chip erase*

The selected external flash will be erased. Parameter: '0' only first flash, '1' only second flash, 'a' all mounted external flashes.

```
F 0 [ENTER]                          ; erase flash 0
F 1 [ENTER]                          ; erase flash 1
F a [ENTER]                          ; erase flash 0 and 1
```

4.2.6.19 *Program mon to flash*

When using a monitor running in a different device (like sram or internal flash), the monitor could be programmed (copied) to the external flash.

4.2.6.17 Switch monitor to ram

If you have to erase/program the external flash you need to switch the monitor to ram first. The basic operation is that the monitor is copied to ram and the mapping of the MMU will be changed to execute the monitor from external sram.

ATTENTION: If your monitor runs in external flash this version could be lost when you erase/program the area of the monitor.

4.2.8.7 *Show flash-config*

The configuration from the external device is displayed at the terminal.

4.2.9 FlexCan Tools

A small library to send/receive can-messages.

```
-- FlexCan-Menu --    {help = ?}
=============================================================================
A) Select can-controller (canIdx)
B) Init can-controller (baudrate)
C) Reset can-controller
D) Show can-State
E) Config rx-mask (0=gbl/14/15,mask)
F) Prepare Rx (ID,IDMask)
H) Tx packet (ID,Data...)
J) Show Rx-buffer
K) Show buffer ([start [end]])
```

4.2.9.1 *Select can-controller*

With this command one or multiple can-controller could be selected:

```
       A 0 [ENTER]                 ; select can-controller A
       A 0 1 2                     ; select all A,B,C
```

4.2.9.2 *Init*

Init all selected can-controller with the given baudrate.

1. parameter is the baudrate:

```
       B 100000 [ENTER]            ; use baudrate 100k
```

4.2.9.3 *Reset*

Reset (disable) can-controller

4.2.9.4 *Show state*

Shows some registers of all can-controller.

4.2.9.5 *Config Rx-mask*

Configure the rx-maks for the receive-buffer. Two parameters first the buffer-id (0=0..13,14,15).

```
       E 0 0xffff                  ; (config global mask (buffer 0..13) to 0xffff)
```

4.2.9.6 *Prepare Rx (id)*

Configure one of the 16-can-buffer (buf-idx 0..15) for receive-operation. This buffer will wait for a specific id.

```
       A 0 1 2 [ENTER]             ; select can-controller A,B,C
       B 1000000 [ENTER]           ; configures the CAN to 1MBit/s
       F 1234 [ENTER]              ; configures buffer to wait for a message with id 1234
```

4.2.9.7 *Transmit (id,data)*

Sends one Package to the can-bus, with your data:

```
A 0 [ENTER]                        ; select can-controller A
B 1000000 [ENTER]                  ; configures the CAN to 1MBit/s (if not already done)
H 4 8 9 a b c d e [ENTER]          ; send 8 bytes ('4', '8', ...)
```

4.2.9.8 *Print Buffer (dev,buf-id)*

Shows all configured rx-buffer.

4.2.9.9 *Print buffer*

Shows range of buffer from all selected can-controller.

Parameter: 1. Start-bufferindex, 2. last-bufferindex

```
k 2 4[ENTER]                       ; Show buffer 2..4 of selected can-controller
```

4.2.10 EQAdc

Small library to show assigned voltage at qadc-input pins.

```
-- EQAdc-Menu --    {help = ?}
==========================================================================
A) Init eQAdc
B) Reset eQAdc
C) Read from Pin (first last seconds)
D) Show eQAdc-State
E) Show AD-Regster
```

4.2.10.1 Init eQAdc

Activate the eqadc-device.

4.2.10.2 Reset eQAdc

Reset (deactivate) the eqadc-device.

4.2.10.3 Read from pin

Function to read current value from analog input-pin.

Parameters: 1 = First pin to read, 2 = Last pin to read, 3 = count of seconds to show values

Example:

```
A [ENTER]                          ; Init qadc-controller
C 5 8 20                           ; show value from pin 5 to 8 for 20 seconds
```

Reset (deactivate) the eqadc-device.

4.2.10.4 Show state of qAdc

Show the registers of the eQAdc device.

4.2.10.5 Show AD-Register

Show the internal ad-controller register

4.2.11 Eth Tools

The Eth-Tools are only usable if you have an Würz-Elektronik evaluation board with an smc91c111 ethernet controller mounted. The monitor supports only some very basic low-level operations like bootp, but no higher level support like tcp/ip. The two submenus are used to have access to additional low-level device operations (like chip-select configuration, access to the serial eeprom, ...).

```
-- EEth_Tool-Menu --    {help = ?}
==========================================================================
A) Expert-Menu                                   --> sub menu
B) EEProm-Menu                                   --> sub menu
C) Prepare
D) Show version
E) Show state
F) Show phy-infos
G) Show phy-state
H) Send bootp
I) Show rx-data
```

4.2.11.1 Prepare

The chipselect and receiver/transmitter will be initialized. If this step is done, the eth-chip will receive frames from the line. Since the device has only a common rx/tx buffer received frames could fill the limited on-chip memory of the device and no transmit is possible until the received frames are removed from the internal queue.

4.2.11.2 Show version

Show the vendor and version information read from the device.

4.2.11.3 Show state

Show current configuration of the smc91c111 device.

4.2.11.4 Show phy-infos

Show informations about the used ethernet-phy.

4.2.11.5 Show phy-state

Show current configuration of the used ethernet-phy.

4.2.11.6 Send bootp

Send bootp request for this device.

4.2.11.7 Show rx-data

Show all received frames.

No parameters show only size of received frames.

1 Parameter:	'1' .. '4' dump the first (16,32,64,128) bytes
	'0' dump the complete message

4.2.12 Test commands

The MPC555x-monitor has some tests implemented to verify the functionality of some peripherals.

```
-- Test-Menu --    {help = ?}
==============================================================================
A) Ram-Tests                                --> sub menu
B) Mpc5554 Tests                            --> sub menu
C) All Ram-Tests
D) All MPC-Tests
E) External SRAM
F) External Flash
G) Internal Flash
H) SCI-2
```

4.2.12.1 All Ram-Tests/Ram-Tests submenu

The all ram-tests will execute a set of tests, which could be found in the Ram-Tests submenu. At default all tests are done for the complete external sram. If not especially mentioned the non-cachable area will be used.

4.2.12.1.1 Data Bus Test

The data-bus test is used to test the data lines to a ram. The test writes different 32-bit patterns to this location (on a device which is smaller then 32-bit, with the data-bus-test also the lowest address lines will be tested: 16-bit --> addressline A30[PPC], 8-bit A30/31[PPC]).

This command requires one address-parameter (which location should be tested), if you do not support a parameter the monitor uses as a default the first SRAM location.

```
A 0x24000000 [ENTER]
```

It is very important to have a blank between the command and the parameter, after the parameter there is no need for a blank (it's only used for better readability).

4.2.12.1.2 Address Bus Test

The address-bus test writes to each location of the address-range the address-information (32-bit), after all locations are written, the MPC555x will verify all words. The command expects two parameters: starting-address and count of bytes (should be 32-bit aligned). If you do not support a parameter the monitor uses as a default the complete SRAM for testing). Pay attention if you request this test, and the monitor is in SRAM your program will crash!

```
B 0x24000000 0x00100000[ENTER]
```

It is very important to have a blank between the command and the parameter, after the parameter there is no need for a blank (it's only used for better readability).

4.2.12.1.3 Short ram-tests

The pattern-test will write to each location in the given address-range the address where it is located. In the second stage the information is verified.

```
A 0x24000000 0x00200000[ENTER]
```

4.2.12.1.4 Long burst-ramtest

This command will use the cachable area of the flash (0x34000000). To each ram-word (32-bit) a shifting 1 (0x1, 0x2, 0x4, 0x8.., 0x80000000) will be written, next a shifting '0' will be used (0xfffffffe, 0xfffffffd,...).

```
C 0x34000000 0x00100000[ENTER]     test 1 Mbytes in cachable area
```

4.2.12.2 Write 0/0xffffffff pattern

The complete address-range will be initialized with the pattern 0x00000000, 0xffffffff. After the write command all cells will be verifyied. In the next Step all cells will be written with the inverted data, and read again.

4.2.12.3 All MPC5554 tests / MPC5554-Test-submenu

The “All Mpc-Tests is a collection of the tests inside the MPC5554-submenu.

4.2.12.4 Test internal SRAM

Currently the first 32 kByte of the internal SRAM will be tested (the second 32 kBytes are used as stack/data section for the monitor.

4.2.12.5 External Flash Test

This test will test the complete external Flash. If the monitor is in the external flash, the code and data will be copied to the ram and the Flash could be tested.

******* ATTENTION *******

If your monitor is working in the external flash, the test should only be done if you have the possibilities to reprogram the monitor with an external tool (debugger, programmer, ...) or you do not need the monitor anymore. If you get an ERROR in this test it could be possible that the monitor is not restored at the test-end – the EC555x-board will not reboot after a reset.

The EC555xMon will do three steps for flash-testing:

1. Test chip-erase
2. Test data/addresslines (the monitor write a pattern to complete flash to test the address and data lines).
3. Test sector-erase

After the test the monitor will be copied to the external flash, so the flash is NOT empty after this test.

4.2.12.6 Internal Flash Test

This test is similar to the external flash-test.

******* ATTENTION *******

If your monitor is working in the external flash, the test should only be done if you have the possibilities to reprogram the monitor with an external tool (debugger, programmer, ...) or you do not need the monitor anymore. If you get an ERROR in this test it could be possible that the monitor is not restored at the test-end – the EC555x-board will not reboot after a reset.

4.2.12.7 Test SCI-2

This test will send some characters (~10) to the tx-side an expect exactly this data on the rx-side (done in polling mode). You have to connect rx-pin with the tx-pin to get this test running.

4.2.13 Configuration Menu

In the configuration-menu there are some sub menu's to configure the MPC555x.

```
-- Config-Menu --    {help = ?}
============================================================================
A) CPU-Menu                                 --> sub menu
B) Sci                                      --> sub menu
C) Memory-Ctrl                              --> sub menu
D) Cache & Burst on
```

4.2.13.1 Cache & Burst on

This entry will do 3 Steps:

1. Configure the external flash to the correct setting (burst-lengt, ...)
2. Activate bursts in the chip-select machine
3. Activate the data and instruction caches

4.2.14 CPU-Cfg-Menu

In the cpu-config-menu the caches of the core could be activated/deactivated.

```
-- CpuCfg-Menu --    {help = ?}
==============================================================================
A) Do Test-Loop
B) Get cache-state
C) Invalidate cache
D) Instruction-cache                                 1
E) Date-cache                                        2
```

4.2.14.1 Do Test-Loop

Short function to check the performance of the core (the count of sysclocks will be printed for a short test-loop).

4.2.14.2 Get cache-state

Show the current configuration of the caches of the MPC555x processor.

4.2.14.3 Invalidate cache

First the data-cache is flushed and invalidated, second the instruction-cache will be invalidated.

4.2.14.4 Instruction-cache

Enable (1) or disable (0) the instruction cache:

```
C [ENTER]                    ; enable instruction cache
```

The cache will be enabled only for the areas configured as cachable (see chapter 2.3.3)

4.2.14.5 data cache

Disable (0) or enable (1 or 2) the data cache. '1' is cache in write-through mode. '2' is in write-back mode.

```
d [ENTER]                    ; enable data-cache in write-back mode.
```

The cache will be enabled only for the areas configured as cachable (see chapter 2.3.3)

4.2.15 Sci

Same menu like in tools, for the description please see chapter 4.2.6.

4.2.16 MemCtrl Menu

The memory controller configuration could be checked or changed with this menu.

```
-- MemCtrlTool-Menu --    {help = ?}
==============================================================================
A) Show config
B) Show flash-config
C) Switch bus-clock (0=1/2,1=1/4)                    0
D) Switch flash-burst                                0
E) Switch sram-burst                                 0
```

4.2.16.1 Show config

The current configuration of the chip-select will be printed.

4.2.16.2 Show flash-config

Print the current configuration of the external flashes (flash-config-register).

4.2.16.3 Switch bus-clock

The default bus-clock divider is two (bus-clock is 66 MHz with 132 MHz system clock). This entry could be used to switch the bus-clock to sysclock/4 (33 MHz with 132 MHz system clock).

```
C 1[ENTER]                   ; switch to 33 MHz busclock
```

4.2.16.4 Switch flash-burst

The burst for the external flash could be activated (1) or deactivated (0) with this entry. The configuration will only affect the memory-controller configuration.

4.2.16.5 Switch sram-burst

The burst for the external sram could be activated (1) or deactivated (0) with this entry. The configuration will only affect the memory-controller configuration.