NEC

Preliminary User's Manual

*start*WARE-GHS-VR4131

Starter Kit VR4131

Document No. U16417EE1V0UM00
Date Published February 2003

Printed in Germany

- The information in this document is current as of 13.02, 2003. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets or data books, etc., for the most up-to-date specifications of NEC Electronics products. Not all products and/or types are available in every country. Please check with an NEC sales representative for availability and additional information.

- No part of this document may be copied or reproduced in any form or by any means without prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.

- NEC Electronics does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from the use of NEC Electronics products listed in this document or any other liability arising from the use of such NEC Electronics products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.

- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of customer's equipment shall be done under the full responsibility of customer. NEC Electronics no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.

- While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.

- NEC Electronics products are classified into the following three quality grades: “Standard”, “Special” and “Specific”.

The "Specific" quality grade applies only to NEC Electronics products developed based on a customer-designated “quality assurance program” for a specific application. The recommended applications of NEC Electronics product depend on its quality grade, as indicated below. Customers must check the quality grade of each NEC Electronics product before using it in a particular application.

"Standard": Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots.

"Special": Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, safety equipment and medical equipment (not specifically designed for life support).

"Specific": Aircraft, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems and medical equipment for life support, etc.

The quality grade of NEC Electronics products is “Standard” unless otherwise expressly specified in NEC Electronics data sheets or data books, etc. If customers wish to use NEC Electronics products in applications not intended by NEC Electronics, they must contact NEC Electronics sales representative in advance to determine NEC Electronics 's willingness to support a given application.

Notes: **1.** " NEC Electronics" as used in this statement means NEC Electronics Corporation and also includes its majority-owned subsidiaries.

2. " NEC Electronics products" means any product developed or manufactured by or for NEC Electronics (as defined above).

M8E 02.10

Regional Information

Some information contained in this document may vary from country to country. Before using any NEC product in your application, please contact the NEC office in your country to obtain a list of authorized representatives and distributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and other legal issues may also vary from country to country.

NEC Electronics Inc. (U.S.)
Santa Clara, California
Tel: 408-588-6000
800-366-9782
Fax: 408-588-6130
800-729-9288

NEC Electronics (Europe) GmbH
Duesseldorf, Germany
Tel: 0211-65 03 01
Fax: 0211-65 03 327

Sucursal en España
Madrid, Spain
Tel: 091- 504 27 87
Fax: 091- 504 28 60

Succursale Française
Vélizy-Villacoublay, France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

Filiale Italiana
Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

Branch The Netherlands
Eindhoven, The Netherlands
Tel: 040-244 58 45
Fax: 040-244 45 80

Branch Sweden
Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

United Kingdom Branch
Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

NEC Electronics Hong Kong Ltd.
Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

NEC Electronics Hong Kong Ltd.
Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

NEC Electronics Singapore Pte. Ltd.
Singapore
Tel: 65-253-8311
Fax: 65-250-3583

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-2719-2377
Fax: 02-2719-5951

NEC do Brasil S.A.
Electron Devices Division
Guarulhos, Brasil
Tel: 55-11-6465-6810
Fax: 55-11-6465-6829

Preface

Readers

This manual is intended for users who want to understand the functions of the *start*WARE-GHS-VR4131 Starter Kit.

Purpose

This manual presents the hardware manual of *start*WARE-GHS-VR4131 Starter Kit..

Organization

This system specification describes the following sections:

- Board Features
- Detailed Functional Description
- Board Operation

Legend

Symbols and notation are used as follows:

Weight in data notation	:	Left is high-order column, right is low order column
Active low notation	:	$\overline{xxx}$ (pin or signal name is over-scored) or /xxx (slash before signal name)
Memory map address:	:	High order at high stage and low order at low stage
Note	:	Explanation of (Note) in the text
Caution	:	Item deserving extra attention
Remark	:	Supplementary explanation to the text
Numeric notation	:	Binary . . . xxxx or xxxB Decimal . . . xxxx Hexadecimal . . . xxxxH or 0x xxxx

Prefixes representing powers of 2 (address space, memory capacity)

K (kilo): $2^{10} = 1024$

M (mega): $2^{20} = 1024^2 = 1,048,576$

G (giga): $2^{30} = 1024^3 = 1,073,741,824$

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 System Requirements

Host PC	A PC supporting Windows 9x, Windows 2000 or Windows NT is required for Green Hills MULTI® 2000 Embedded Development Kit. Pentium 133 MHz (at least), 32 MB of RAM, 256-color display (1024 × 768), mouse, CD-ROM drive and 60 Mbytes of free hard disk space are required to install the GHS compiler and debugger package.
Host interface	Serial (RS232C) interface capable to handle communication at 9600, 19200, 38400 or 115200 baud.

1.2 Package Contents

Please verify that you have received all parts listed in the package contents list attached to the *start*-WARE-GHS-VR4131 package. If any part is missing or seems to be damaged, please contact the dealer from whom you purchased your *start*WARE-GHS-VR4131.

Note: Updates to this User Manual, additional documentation and/or utilities for *start*WARE-GHS-VR4131, if available, may be downloaded from the NEC WEB page(s) *http://www.nec.de/support*.

1.3 Related Documents

VR4131 Preliminary User's Manual Hardware, NEC Doc. Number U15350EJ2V0UM00
VR4131 Preliminary Data Sheet, NEC Doc. Number U16219EJ1V0DS00
VR4100 Series User's Manual Architecture, NEC Doc. Number U15509EJ2V0UM00
Green Hills Documentation is provided with the installation of the MULTI® software.

1.4 Used Abbreviations

There are some abbreviations used in this document, which may require additional information to be understood correctly.

- RFU - reserved for future use
- NC - not connected
- GND - ground
- GHS - Green Hills
- GNU - S/W supplied under General Public License (GLP)
- MIPS - Microprocessor without interlocked pipeline stages

Chapter 2 Board Features

*Start*ware-GHS-VR4131 is a low-cost evaluation board for NEC's VR4131 64-bit high-performance microprocessor. It allows evaluation of the processor's performance as well as potential system performance, because the VR4131 can also be operated together with its typical system environment.

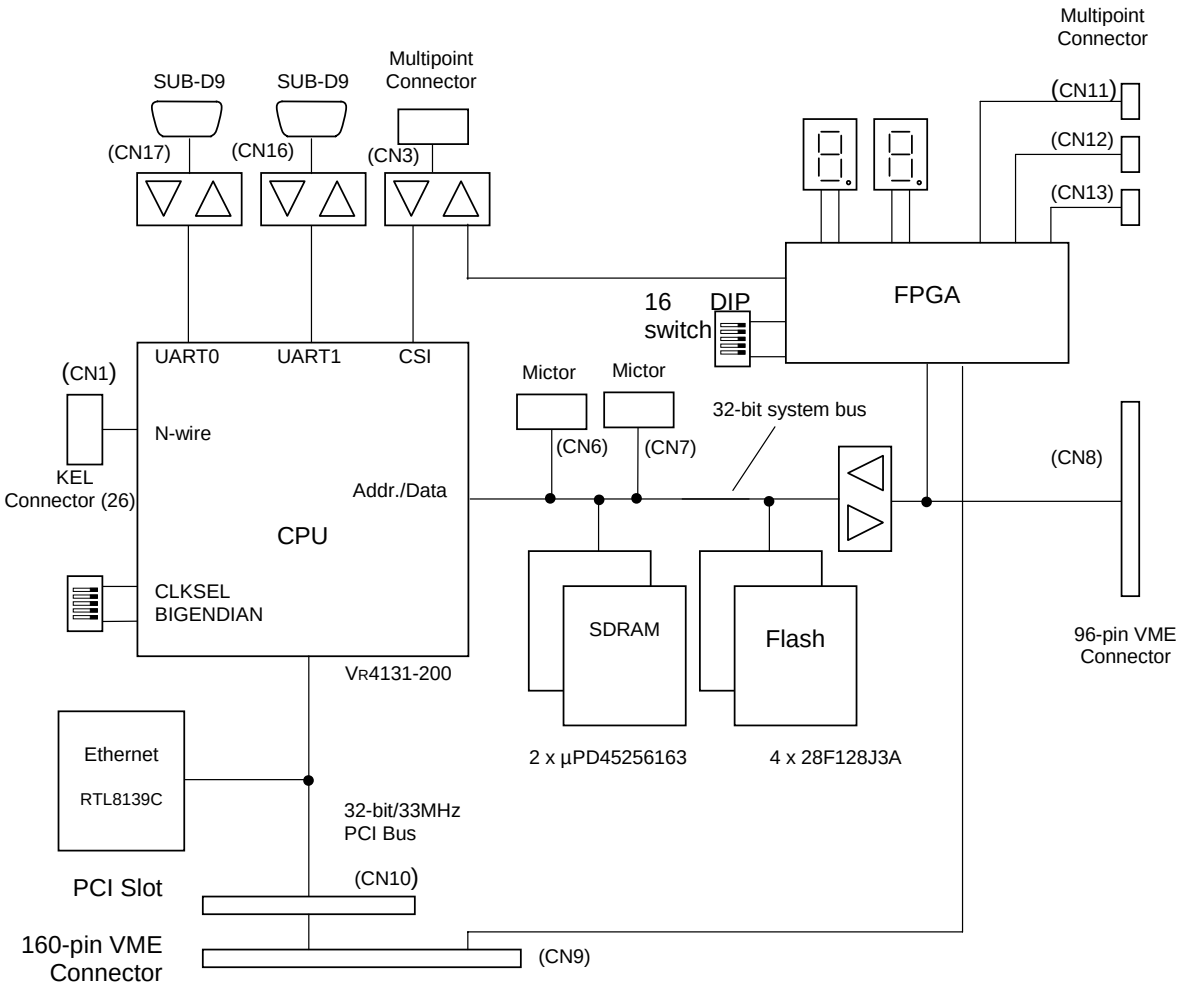
The *start*WARE-GHS-VR4131 board has a VR4131-200 processor soldered; processor speed is configurable via jumpers. The VR4131 is directly connected to 64MByte of SDRAM, built up with 2 µPD45256163 devices (total bus width 32-bit). The SDRAM supports full 100MHz SDRAM clock. 64MByte of flash are connected to the VR4131 memory bus directly. Mictor connectors are attached to the memory bus before the isolation buffer. A short summary of the board features is given below:

- VR4131-200 CPU
- 64 MByte Flash
- 64 MByte SDRAM
- N-wire debug interface (via KEL connector as used with the Midas ICE)
- 2 serial interfaces on conventional SUB-D connectors, buffered via RS-232 line drivers
- Ethernet Interface (Realtek RTL8139C controller
- Clock-synchronous interface extended with 2 special input and 2 special output signals on a simple multipoint connector
- IrDA module – TFDS6502
- PCI bus on a standard PCI slot connector (32-bit, 33MHz, R2.1)
- PCI bus with VR4131 specific extensions on a standard 160-pin VME connector
- System bus on a standard 96-pin VME connector
- Mictor Connectors for measurement purpose
- FPGA as an additional I/O device with several I/O ports on three Multipoint connectors
- 2 7-segment LED displays for debugging purpose
- 16 DIP switches for selection of SDRAM clock, PCI clock, Monitor and user defined functions
- Single Voltage Power Supply

The board software consists of three monitor programs:

- A simple download monitor program named "S-Boot". The S-Boot monitor downloads files to the processor's SDRAM and/or Flash memory and optionally starts the program. A standard terminal emulation program can be used on the host PC side for communication.
- The Green Hills target monitor communicates with Green Hills Multi Compiler/Debugger on the host PC. An evaluation license of Multi is included in the package.
- The E-Boot download monitor for high-speed downloads via Ethernet.

Figure 2-1: Simplified Block Diagram



Chapter 3 Functional Description

The VR4131 Evaluation board (*start*WARE-GHS-VR4131) is designed to evaluate the VR4131 MIPS RISC processor and uses the Realtek RTL8139C chip for ethernet downloading of software. It can also be used as development platform for operating systems like WinCE, VxWorks, Linux or Integrity. The VR4131 Evaluation board (*start*WARE-GHS-VR4131) is designed to be used either in a stand-alone mode using a 15V AC adapter or together with a Ravin-E Board as graphic interface and/or an I/O Board to increase the number of I/O interfaces. The VR4131 Evaluation board (*start*WARE-GHS-VR4131) provides 64 MB of SDRAM memory with two 256 Mbit chips and 64 MB of Flash memory with four 128 Mbit chips.
The I/O capabilities of the VR4131 Evaluation board (*start*WARE-GHS-VR4131) as stand alone board include 2 on-chip UARTs, an Ethernet Interface realized with RTL8139C, a PCI interface, a clock synchronous, serial interface, Real Time Clock (RTC) timers, and DMA channels. The FGPA increases the number of general purpose I/O pins. These boards are shipped with the NEC S-Boot monitor and two other monitors pre-installed in the Flash memory.

3.1 Flash Memory

The VR4131 Evaluation board (*start*WARE-GHS-VR4131) has 64 MByte on-board Flash memory where the user can store data or transfer code to. The system normally boots from the on-board Flash memory. The Flash memory is mapped at `0xBC00 0000` through `0xBFFF FFFF`. In order to boot from Flash memory, the code must start at the physical address `0xBFC0 0000`. Thus only the upper 4 MByte of Flash memory are available to the boot code. Flash memory is accessed via the VR4131 chip selects ROMCS0# and ROMCS1#.
Write operations to the Flash memory by Vr4131 program execution are possible and controlled by the BCUCNTREG1 register at address `0xAF00 0000` in the VR4131. Two additional “security levels” are provided to protect the Flash from unintended writing: Jumper JP7 and the VPEN bit at address `0xAA00 0010` in the FPGA. Typically the pre-installed monitor programs will be used for writing to Flash; however it is also possible to do this in a user application program.

3.2 Main Memory

The VR4131 Evaluation board (*start*WARE-GHS-VR4131) contains 64 MByte of main memory implemented in one physical bank and realized with two 256 Mbit SDRAM chips connected to CS0#. The physical address map for this memory is `0x0000 0000` to `0x03FF FFFF`. The CPU address is `0xA000 0000` to `0xA3FF FFFF` for uncached or `0x8000 0000` to `0x83FF FFFF` for cached accesses.

3.3 FPGA Functions

The FPGA controls the power up and power down sequences and realizes several ports for predefined and user definable data I/O. Furthermore the FPGA routs the PCI interrupts INTA# to INTD# to GPIOs of the VR4131.
Predefined ports are the LED display, the DIP switches, the Flash ROM control signals and the extension lines of the CSI interface. The other ports can be used as user definable GPIOs or to realize an additional connection between the FPGA and the VR4131.
The following block and state diagram show the functionality of the FPGA. The block diagram shows the part of the I/O lines except the power control logic. The functionality of this part is shown in the state diagram.

Figure 3-1: FPGA Block Diagram - I/O Circuits

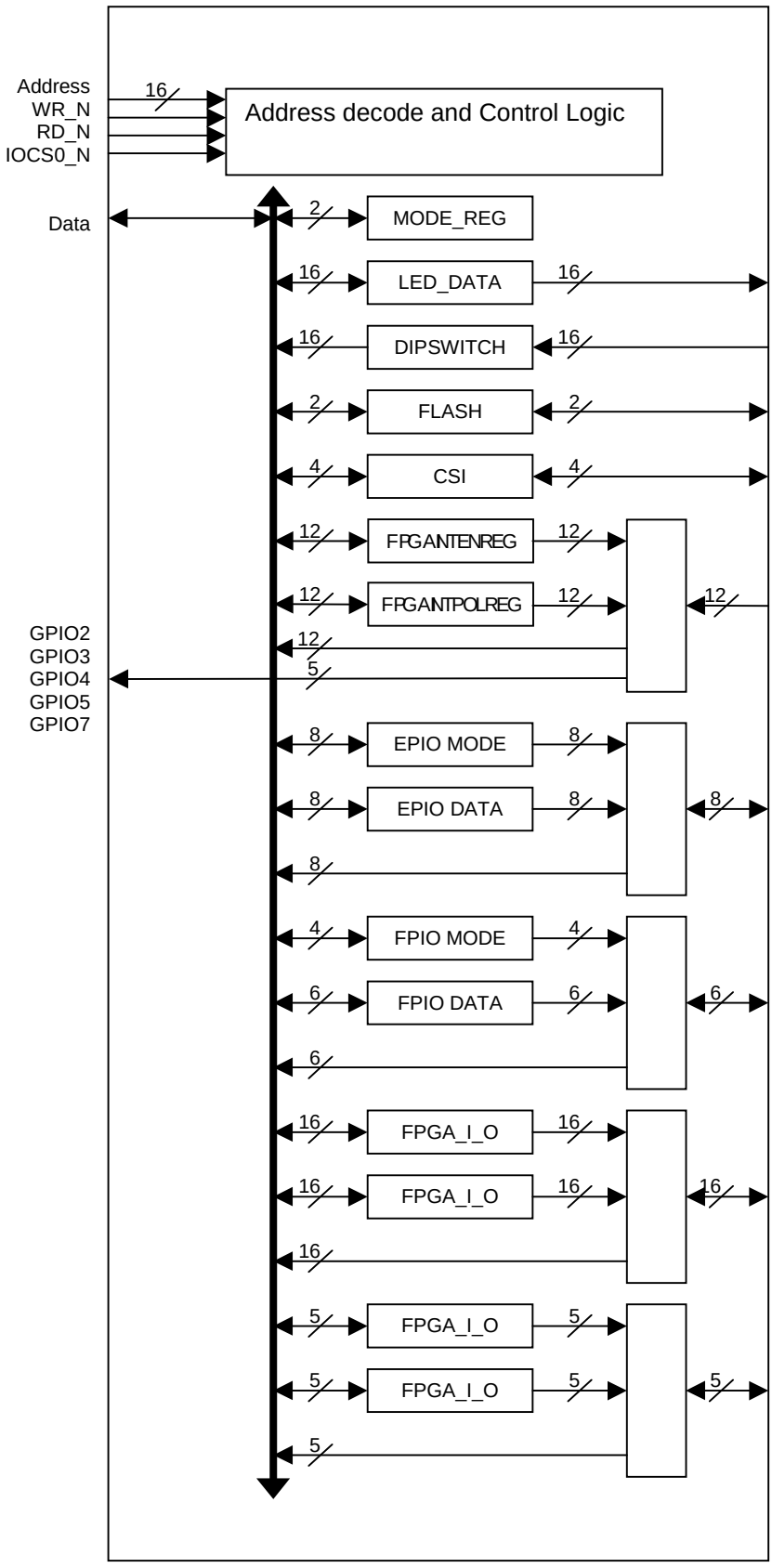
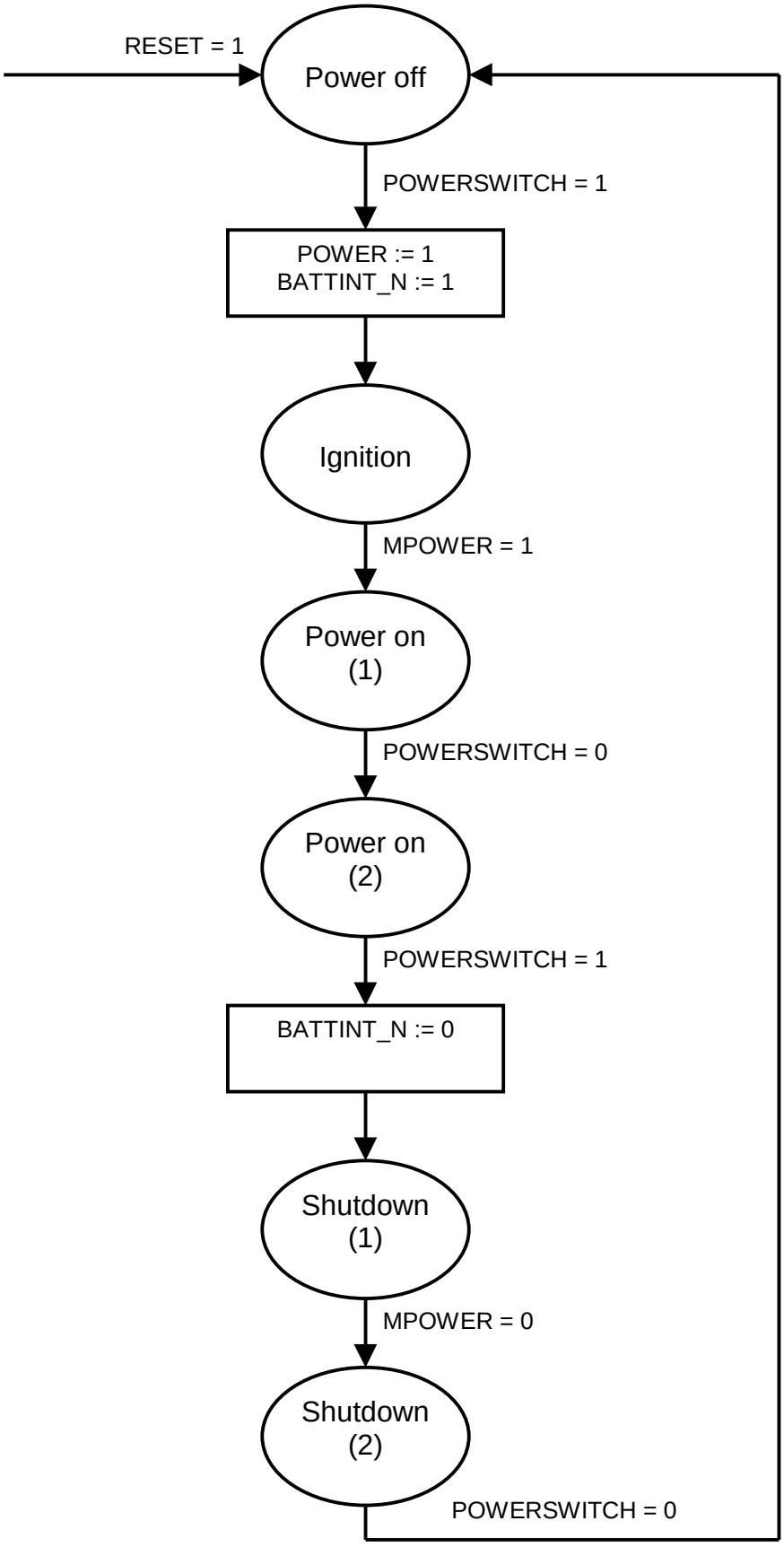


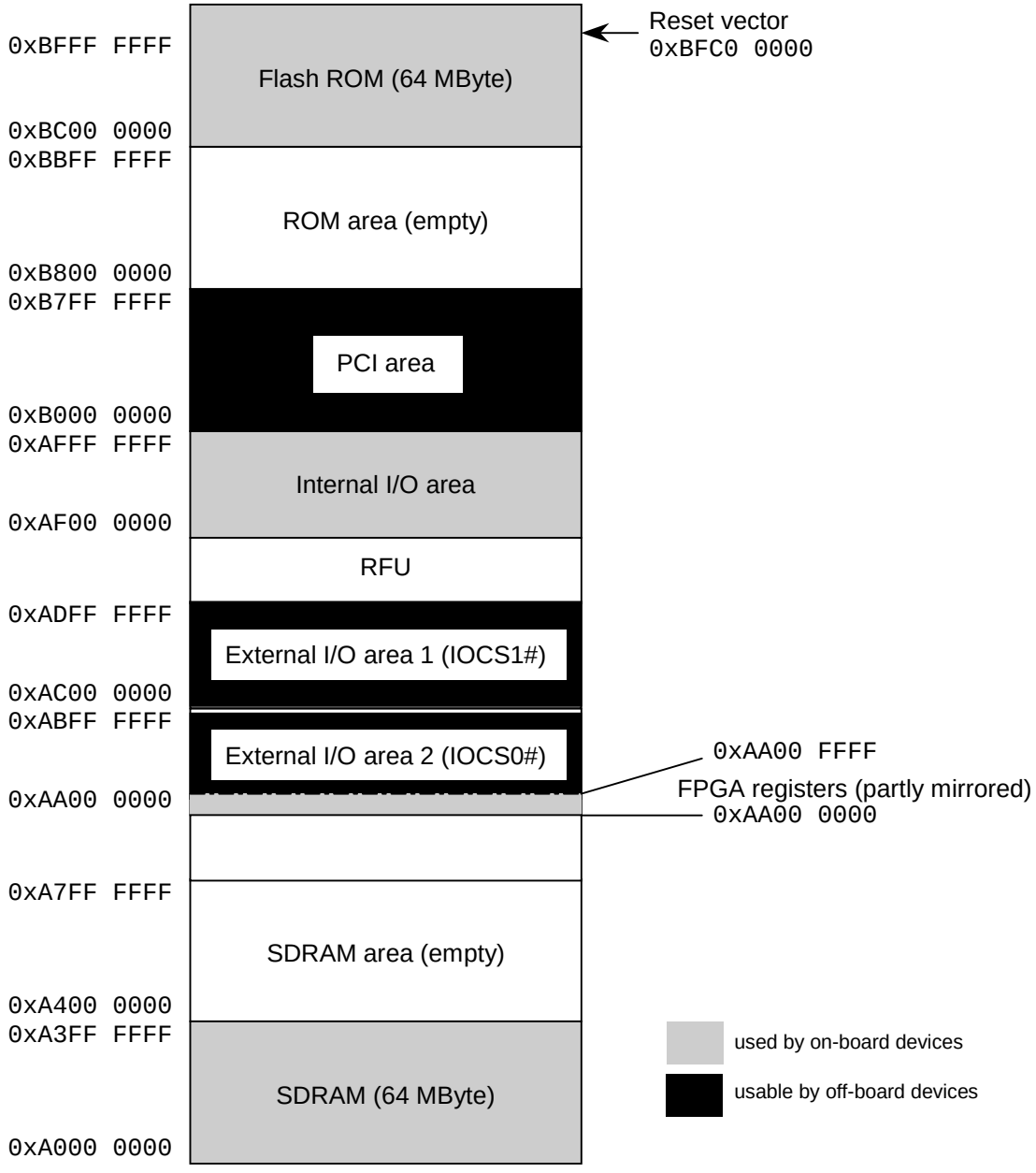
Figure 3-2: FPGA State Diagram – Power on Logic



3.4 Memory Mapping

The VR4131 processor regards external devices like the FPGA on the evaluation board as memory mapped I/O, which is controlled with one of the external chip select signals, IOCS0#. An access to an FPGA register is executed in the same way than a memory access. The 32-bit physical address space encompasses a total of 4 GByte; however these 4 GByte consist of 8 mirror images of a 512 MByte space subdivided in SDRAM, ROM and I/O spaces. The most significant address bits control, if the 512 MByte space is accessed cached or uncached respectively mapped or unmapped.
The following diagram shows the memory map for the *start*WARE-GHS-VR4131 board for uncached and unmapped addresses (kseg1) ranging from `0xA000 0000` to `0xBFFF FFFF`. Note that the same structure is seen for other address space segments like kseg0 (`0x8000 0000` to `0x9FFF FFFF`). Consult the processor documentation for details.

Figure 3-3: startWARE-GHS-VR4131 address map for kseg1 (unmapped, uncached)



Chapter 4 Detailed Functional Description

4.1 Usage of VR4131 GPIO Pins

Several GPIO pins of the VR4131 are directly used by the FPGA and other circuitry; therefore they cannot be used as user defined I/O pins. There is a total of 7 GPIOs freely usable; 4 GPIOs can additionally be used, if the debug serial I/O of the VR4131 is not used.
The following table lists the usage of all GPIO pins:

Table 4-1: VR4131 GPIO pin usage

Pin Name	Pin Function
GPIO0	is defined as a general interrupt input. The interrupt cause can be read in the FPGAINTREG
GPIO1	is connected to CN9 and CN11 and can be used as GPIO or interrupt pin
GPIO2	is used as interrupt input of the PCI Bus (INTA1_N, INTA2_N, FPIO4)
GPIO3	is used as interrupt input of the PCI Bus (INTB1_N, INTB2_N)
GPIO4	is used as interrupt input of the PCI Bus (INTC1_N, INTC2_N)
GPIO5	is used as interrupt input of the PCI Bus (INTD1_N, INTD2_N)
GPIO6	is configured as SYSDIR signal
GPIO7	is used as interrupt input caused on the I/O Board (INTE_N) or by the CSI interface expansion (RQC0, RQC1)
GPIO8	is connected to CN8 and CN11 and can be used as GPIO or interrupt pin
GPIO9	is connected to CN8 and CN11 and can be used as GPIO or interrupt pin
GPIO10	is connected to CN8 and CN11 and can be used as GPIO or interrupt pin
GPIO11	is connected to CN9 and CN11 and can be used as GPIO or interrupt pin
GPIO12	is connected to CN9 and CN11 and can be used as GPIO or interrupt pin
GPIO13	is connected to CN9 and CN11 and can be used as GPIO or interrupt pin
GPIO14	not available
GPIO15	in connected to CN11 for usage as GPIO, but default configured as DCD# signal
GPIO16 - GPIO31	is used as Data 16 –31 for the 32 Bit Data Bus
GPIO32 - GPIO35	is used as Debug Serial Interface; can be defined as GPIO, if the connection at CN29 is open

4.2 Jumper Settings

Before power-on, the board should be configured through the jumpers. The jumper settings are as shown in the following tables. The default setting is marked as a shaded area in the column entitled “Function”; an example for the default position 2-3 is given below:

Table 4-2: Example for default jumper setting marking

No.	Pos.	Function
1		Default
2		
3		
1		Not default
2		
3		

An overview about the placement of the jumpers on the evaluation board is shown in figure 4-1 at the end of chapter 4.2.

4.2.1 Pipeline Clock Setting

The VR4131 uses three pins (CLKSEL(2:0)), to configure the pipeline clock; these pins are sampled when the RTCRST# input is de-asserted. CN26,CN28 and CN24 are connected to these pins and select the pipeline clock as described in table 4-3 below:

Table 4-3: Pipeline clock setting for VR4131

CLKSEL2 (CN26)	CLKSEL1 (CN28)	CLKSEL0 (CN24)	PClock	VTClock, SCLK				
				Div2	Div3	Div4	Div5	Div6
1	1	1	RFU	RFU	RFU	RFU	RFU	RFU
1	1	0	199.1 MHz	99.5 MHz	66.4 MHz	49.8 MHz	39.8 MHz	33.2 MHz
1	0	1	181.0 MHz	90.5 MHz	60.3 MHz	45.2 MHz	36.2 MHz	30.2 MHz
1	0	0	165.9 MHz	82.9 MHz	55.3 MHz	41.5 MHz	33.2 MHz	27.6 MHz
0	1	1	153.1 MHz	76.6 MHz	51.0 MHz	38.3 MHz	30.6 MHz	RFU
0	1	0	132.7 MHz	66.4 MHz	44.2 MHz	33.2 MHz	26.5 MHz	RFU
0	0	1	99.5 MHz	49.8 MHz	33.2 MHz	RFU	RFU	RFU
0	0	0	RFU	RFU	RFU	RFU	RFU	RFU

The corresponding jumper positions for CN26, CN28 and CN24 look as shown in table 4-4.

Table 4-4: Jumper positions for pipeline clock setting

	CN24	CN28	CN26	PClock
1	■	■	■	RFU
2	■	■	■	
3				
1		■	■	199.1 MHz
2	■	■	■	
3	■			
1	■		■	181.0 MHz
2	■	■	■	
3		■		
1			■	165.9 MHz
2	■	■	■	
3	■	■		
1	■	■		153.1 MHz
2	■	■	■	
3			■	
1		■		132.7 MHz
2	■	■	■	
3	■		■	
1	■			99.5 MHz
2	■	■	■	
3		■	■	
1				RFU
2	■	■	■	
3	■	■	■	

4.2.2 Endianness Setting

Jumper CN25 selects the endianness as described below:

Table 4-5: Jumper positions for endianness setting

	CN25	Endianness
1		Little Endian
2		
3		
1		Big Endian
2		
3		

4.2.3 Data Bus Width Setting

The VR4131 can work with a 32-bit wide and with a 16-bit wide memory system. A 16-bit wide bus configuration reduces system performance but frees 16 data lines which can then be used as GPIOs. Selection is done via the DBUS32 pin which is sampled, when RTCRST# is de-asserted. Jumper CN2 selects the data bus width as described below:

Table 4-6: Jumper positions for data bus width setting

	CN2	Data bus width
1		16 bit data bus
2		
3		
1		32 bit data bus
2		
3		

4.2.4 MIPS16 Enable Setting

The VR4131 supports the MIPS16 instruction set extension as defined by MIPS. However before running MIPS16 code, the MIPS16EN pin must driven high during RTCRST#. Jumper CN27 defines whether the execution of MIPS16 instructions on the VR4131 is enabled or not as described below:

Table 4-7: Jumper positions for MIPS16 Enable setting

	CN27	MIPS16 Enable
1		MIPS16 disabled
2		
3		
1		MIPS16 enabled
2		
3		

Note that this setting is completely independent from the data bus width configuration.

4.2.5 Other Jumper Settings

JP7 defines whether write to Flash is allowed or not as described below. This jumper directly controls the VPEN inputs of the flash memories and guarantees that VPEN is permanently pulled down, when the jumper is not plugged in.

Table 4-8: Jumper positions for data bus width setting

	JP7	Write to Flash
1		Write to Flash disabled
2		
1		Write to Flash enabled
2		

4.2.6 Jumpers for Power Supply Control

Some jumpers on the evaluation board are just for current measurement purposes; they disconnect parts of the power supply completely when they are not present. JP2, JP3, JP4, JP9, JP10 and JP11 belong to this category. Please consult the circuit diagram for the influence of the different supply voltages. These jumpers control the supply voltages as follows:

Table 4-9: Jumper positions for power supply control (1/2)

	JP2	VDD3.3P
1		VDD3.3P not supplied
2		
1		VDD3.3P supplied
2		

	JP3	VDD2.5
1		VDD2.5 not supplied
2		
1		VDD2.5 supplied
2		

	JP4	VDD5.0
1		VDD5.0 not supplied
2		
1		VDD5.0 supplied
2		

	JP9	VDD3.3SDRAM
1		VDD3.3SDRAM not supplied
2		
1		VDD3.3SDRAM supplied
2		

	JP10	VDD3.3
1		VDD3.3 not supplied
2		
1		VDD3.3 supplied
2		

Table 4-9: Jumper positions for power supply control (2/2)

	JP11	VDD1.5
1		VDD1.5 not supplied
2		
1		VDD1.5 supplied
2		

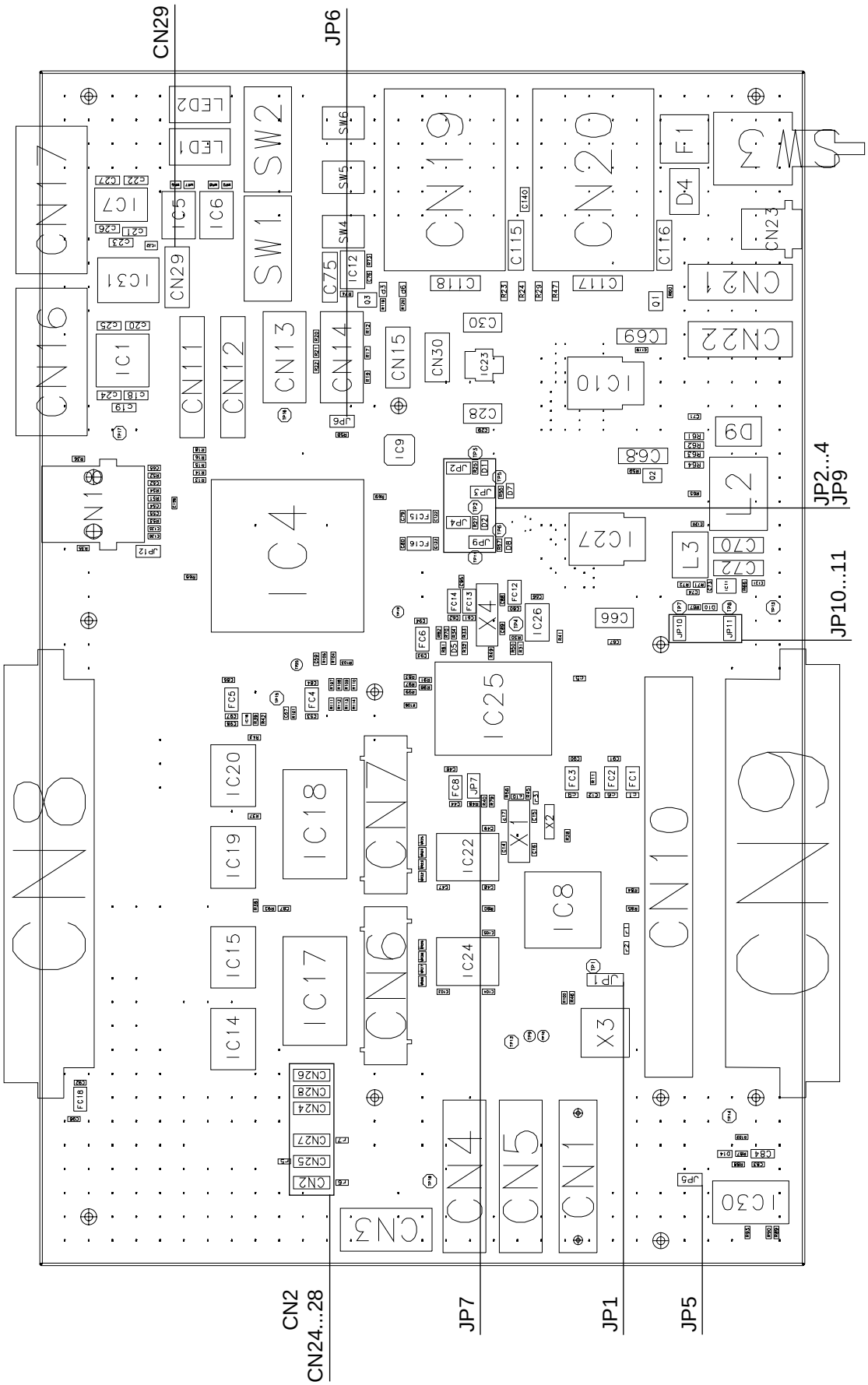
4.2.7 Jumpers for Debug SIU and GPIOs

The pins for the debug serial interface of the VR4131 are shared with general purpose I/O pins and can be configured to either usage with the jumpers at CN29. They are normally configured for usage are debug SIU, as the S-Boot monitor uses this interface.

Table 4-10: Debug serial interface usage

CN29										Debug serial interface usage
1		3		5		7		9		Debug Serial Interface activated
2		4		6		8		10		
1		3		5		7		9		Debug Serial Interface not activated
2		4		6		8		10		

Figure 4-1: Jumper positions

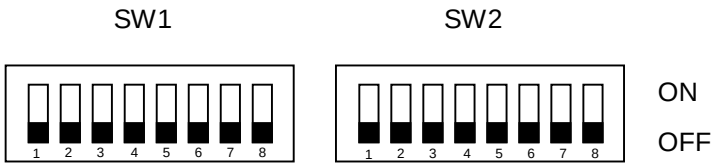


4.3 Switch Settings

4.3.1 DIP Switch Settings

The VR4131 Evaluation Board has two 8-bit DIP-switches to define several settings.

Figure 4-2: DIP switch setting for SW1 and SW2



SW1 is defined as follows:

Table 4-11: DIP switch setting for SW1

SW1-1	SW1-2	Boot Monitor
OFF	OFF	S-Boot Monitor
OFF	ON	Jump to Flash memory base address
ON	OFF	Green Hills Monitor
ON	ON	E-Boot Monitor

SW1-5	SW1-6	SW1-7	SDRAM Clock/VTClock
OFF	OFF	OFF	re-use setting from CLKSEL pins
OFF	OFF	ON	reserved
OFF	ON	OFF	PClock/2
OFF	ON	ON	PClock/3
ON	OFF	OFF	PClock/4
ON	OFF	ON	PClock/5
ON	ON	OFF	PClock/6
ON	ON	ON	Reserved

SW1-8	FPGA test mode
OFF	Test mode off
ON	Test mode on

Please note that the DIP switches are read by software running on the VR4131 and that the setting according to the switch setting will be done by software as well. Not all combinations between pipeline clock (PClock) and SDRAM clock are allowed; table Table 4-3, “Pipeline clock setting for VR4131,” on page 22 shows the details. The currently installed S-Boot Monitor takes care of the clock setting; other software does not necessarily do this.

4.3.2 Other Switch Settings

The board is equipped with 4 more switches related to powering up the board. The main power switch is SW3 on the front side of the board. It directly connects power to the switching regulators and activates LEDs D1, D2 and D7. Note that SW3 does not yet start the processor. LED D6 indicates that the automatically generated RTCRST is active. While the RTCRST is active, the processor must not be started. The push button SW6 generates a pulse at the VR4131 POWER input and starts up the processor. SW6 should not be pressed before the FPGA configuration is completely terminated, which is indicated by LED D3.
Push button SW5 generates the RSTSW# signal which is directly connected to the related pin of the processor. SW4 (“RTCRESET”) generates a reset signal that conforms to the VR4131 specification.
The position of the switches on the board can be found in Figure 4-1, “Jumper positions,” on page 28.

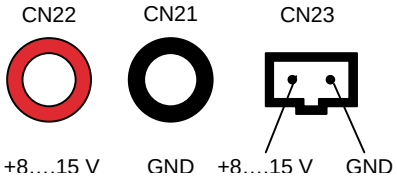
4.4 Connectors

The VR4131 evaluation board provides several connectors for testing purpose and for system extensions. This chapter will describe usage of the connectors one by one.

4.4.1 Power Supply Connectors (CN21, 22 and 23)

There are two connector types provided on the board to connect a power supply. You can either use the “classic” laboratory style connectors CN21 and CN22 or the texas style connector CN23. CN23 is used by the power supply that is delivered together with the board. The board consumes typically 150 mA at 15 V (does not include any externally connected boards). Feeding the board with lower voltages increases(!) the supply current due to the switched power supply circuits.

Figure 4-3: Power supply connectors (board front view)

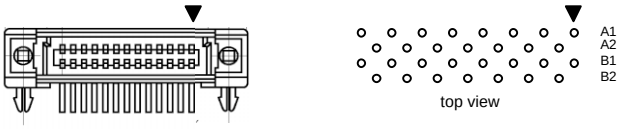


Caution: We urgently recommend to check polarity of the power supply that comes together with the board before connecting it to CN23.

4.4.2 Usage of the N-Wire ICE Connector (CN1)

For using the N-Wire ICE (RTE-1000-TP) connect the ICE to CN1.

Figure 4-4: N-Wire Connector



CN1 is directly connected to the VR4131 N-Wire-Interface.

GND	A1	B1	GND
GND	A2	B2	GND
GND	A3	B3	GND
GND	A4	B4	GND
GND	A5	B5	GND
GND	A6	B6	GND
RMODE_N/JTDI	A7	B7	GND
JTCK	A8	B8	GND
JTMS	A9	B9	GND
JTDO	A10	B10	GND
JTRST_N	A11	B11	NC.
BKTGIO_N	A12	B12	NC.
NC.	A13	B13	VDD3.3

JP1 enables/disables the N-Wire-Interface of the VR4131. JP5 connects the IRDOUT_N/JTDO pin to the FIR module. As this is a shared pin of the VR4131, this connection is not allowed when the N-Wire interface is used.

Table 4-12: Jumper setting to enable N-Wire

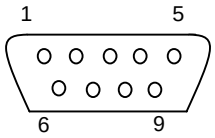
	JP1		JP5	N-Wire interface Enable
1		1		N-Wire interface disabled
2				
3		2		
1		1		N-Wire interface enabled
2				
3		2		

4.4.3 Using the Serial Interface (CN16)

The VR4131 has two on-chip serial interfaces, called the SIU (**s**erial **i**nterface **u**nit) and the DSIU (**d**ebug SIU). The SIU interface conforms to the RS232-C communication standard and supports up to 1.15Mbps. This unit is functionally compatible with the NS16550. The address map for the SIU is 0xAF00 0800 to 0xAF00 080A. For details refer to the VR4131 Users Manual. The processor's serial interface is connected to a RS232-C electrical interface via a level converter. It uses a standard male 9-pin SUB/D connector CN16 on the CPU PCB. The pin definitions are described in the following figure.

Figure 4-5: Serial interface connector CN16

1	DCD_N
2	RXD
3	TXD
4	DTR_N
5	GND
6	DSR_N
7	RTS_N
8	CTS_N
9	NC



front view

4.4.4 Usage of Debug Serial Interface (CN17)

The pins for the debug serial interface of the VR4131 are shared with GPIOs and are software configurable to either function. Both types of usage are supported by the VR4131 evaluation board; jumpers on CN29 need to be configured accordingly.
If the Debug Serial Interface is not activated, CN29 can be used as connector for connections to GPIO (35:32). These GPIOs are general purpose **outputs** – for details see also the VR4131 Users Manual.

Note: CN29 only affects the routing of the VR4131 signals on the board. The actual configuration is done by software on the VR4131.

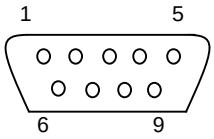
Table 4-13: Debug serial interface usage

CN29										Debug serial interface usage
1		3		5		7		9		Debug Serial Interface activated
2		4		6		8		10		
1		3		5		7		9		Debug Serial Interface not activated
2		4		6		8		10		

The address map for the DSIU is `0xAF00 0820` to `0xAF00 0827`. For details refer to VR4131 User's Manual. The processor's serial interface is connected to a RS232-C electrical interface via a level converter. It uses a standard male 9-pin SUB/D connector CN17 on the CPU PCB. The pin definitions are listed in the following table.

Figure 4-6: Serial interface connector CN17

1	NC.
2	DDIN
3	DDOUT
4	NC.
5	GND
6	NC.
7	DRTS_N
8	DCTS_N
9	NC.



front view

4.4.5 Using the FIR Interface

For using the FIR interface the N-Wire ICE (RTE-1000-TP-EE) must be disconnected from CN1. JP1 must disable the N-Wire-Interface of the VR4131. JP5 connects the IRDOUT_N/JTDO signal to the FIR module. This connection is not allowed when the N-Wire interface is used.

Table 4-14: FIR/N-Wire interface selection

	JP1		JP5	N-Wire interface Enable
1	■	1	■	FIR interface enabled
2	■		■	
3		2	■	
1		1		N-Wire interface enabled
2	■			
3	■	2		

4.4.6 Using the CSI Interface (CN3)

The CSI interface of the VR4131 is available on connector CN3. This interface is extended by four signals controlled by the FPGA. Additionally CN3 provides 3.3V and 5V power supplies. Signals on the connector are assigned as follows:

Table 4-15: CSI connector pin assignment (CN3)

3.3 V	1			2	SIN
SOUT	3			4	5 V
RQC0	5			6	RQC1
GNTC0	7			8	GNCT1
SECLK	9			10	GND

SIN, SOUT and SECLK are connected via permanently enabled, non-inverting buffers to the corresponding pins of the VR4131; the other signals are provided to support multi-master operation of the CSI. RQC0 and RQC1 are (input) signals to request the CPU's CSI interface. The CPU can confirm this request via the GNTC0 and GNTC1 (output) signals. These lines are fully software controlled by the CPU by reading and writing the CSICONTREG in the FPGA. The usage of these extra signals is not mandatory.

4.4.7 General Purpose IOs at CN11

There are several possibilities to connect user hardware to the board; one of them is using I/O ports. The VR4131 evaluation board provides two kinds of I/O ports: **G**PIO's are realized in the CPU (VR4131) and controlled using the respective GIU control registers in the VR4131. Additionally the **E**PIO signals can be controlled using the EPIO_MODEREG and EPIODATA registers in the FPGA.

At the multipoint connector CN11 the pin assignment is as follows:

Table 4-16: GPIO and EPIO pin assignment at CN11

EPIO1	1			2	GPIO1
EPIO8	3			4	GPIO8
EPIO9	5			6	GPIO9
EPIO10	7			8	GPIO10
EPIO11	9			10	GPIO11
EPIO12	11			12	GPIO12
EPIO13	13			14	GPIO13
EPIO15	15			16	GPIO15
	17			18	
GND	19			20	GND

Note: This connector can also be used as a jumper field to realize additional connections between the CPU and the FPGA e.g. for additional interrupts. In this case the used EPIO pins of the FPGA must be configured as outputs (using the EPIO_MODEREG) and the respective GPIOs of the VR4131 must be configured as inputs.

4.4.8 General Purpose IO's at CN12 and CN13

These FPGA_I_O's are realized in and connected to the FPGA. The CPU controls these I/O's and each I/O can be defined as input or output separately. Control is done via a set of four registers in the FPGA: FPGA_I_O_MODEREGL, FPGA_I_O_MODEREGH, FPGA_I_O_REGL, and FPGA_I_O_REGH.

Table 4-17: Pin assignment at CN12

VDD3.3	1			2	
FPGA_I_O(0)	3			4	FPGA_I_O(1)
FPGA_I_O(2)	5			6	FPGA_I_O(3)
FPGA_I_O(4)	7			8	FPGA_I_O(5)
FPGA_I_O(6)	9			10	FPGA_I_O(7)
FPGA_I_O(8)	11			12	FPGA_I_O(9)
FPGA_I_O(10)	13			14	FPGA_I_O(11)
FPGA_I_O(12)	15			16	FPGA_I_O(13)
FPGA_I_O(14)	17			18	FPGA_I_O(15)
GND	19			20	

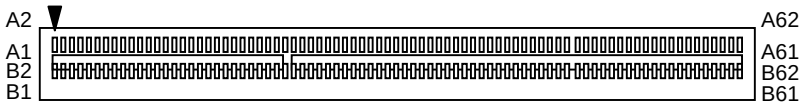
Table 4-18: Pin assignment at CN13

FPGA_I_O(16)	1			2	FPGA_I_O(17)
FPGA_I_O(18)	3			4	VDD3.3
FPGA_I_O(20)	5			6	FPGA_I_O(19)
reserved	7			8	NC
GND	9			10	GND

4.4.9 PCI Connector CN10

The PCI interface of the VR4131 processor is directly routed to a standard PCI connector CN10. The VR4131 supports up to three external PCI devices; a device connected to CN10 is using REQ1_N and GNT1_N signals of the CPU. The PCI interrupt pins INTA_N, … INTD_N are routed to and processed in the FPGA. Modification of the FPGA content will allow implementation of a simple PCI interrupt controller; however this function is currently not implemented.

Figure 4-7: PCI connector pin assignment



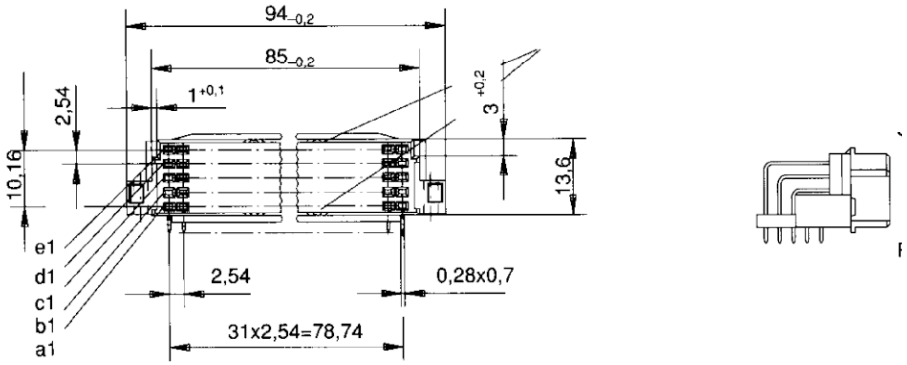
	A1	B1	
	A2	B2	
	A3	B3	GND
	A4	B4	
VDD5.0	A5	B5	VDD5.0
INTA1_N	A6	B6	VDD5.0
INTC1_N	A7	B7	INTB1_N
VDD5.0	A8	B8	INTD1_N
	A9	B9	
VDD3.3	A10	B10	
	A11	B11	
	A12	B12	
	A13	B13	
	A14	B14	
RST_N	A15	B15	GND
VDD3.3	A16	B16	PCLK1
GNT1_N	A17	B17	GND
GND	A18	B18	REQ1_N
	A19	B19	VDD3.3
AD30	A20	B20	AD31
VDD3.3	A21	B21	AD29
AD28	A22	B22	GND
AD26	A23	B23	AD27
GND	A24	B24	AD25
AD24	A25	B25	VDD3.3
IDSEL1	A26	B26	CBE3
VDD3.3	A27	B27	AD23
AD22	A28	B28	GND
AD20	A29	B29	AD21
GND	A30	B30	AD19
AD18	A31	B31	VDD3.3
AD16	A32	B32	AD17
VDD3.3	A33	B33	CBE2
FRAME_N	A34	B34	GND
GND	A35	B35	IRDY_N
TRDY_N	A36	B36	VDD3.3
GND	A37	B37	DEVSEL_N
STOP_N	A38	B38	GND
VDD3.3	A39	B39	LOCK_N
	A40	B40	PERR_N

	A41	B41	VDD3.3
GND	A42	B42	SERR_N
PAR	A43	B43	VDD3.3
AD15	A44	B44	CBE1
VDD3.3	A45	B45	AD14
AD13	A46	B46	GND
AD11	A47	B47	AD12
GND	A48	B48	AD10
AD9	A49	B49	
GND	A50	B50	GND
GND	A51	B51	GND
CBE0	A52	B52	AD8
VDD3.3	A53	B53	AD7
AD6	A54	B54	VDD3.3
AD4	A55	B55	AD5
GND	A56	B56	AD3
AD2	A57	B57	GND
AD0	A58	B58	AD1
VDD3.3	A59	B59	VDD3.3
	A60	B60	
VDD5.0	A61	B61	VDD5.0
VDD5.0	A62	B62	VDD5.0

4.4.10 I/O Board Connector CN9

The VR4131 evaluation board offers two “specialised” extension connectors; one is referred to as I/O Board connector. It is a 160-pin (5x32) VME-style connector that carries the complete PCI bus (with REQ2_N and GNT2_N) and several additional signals like GPIOs from the CPU, FPIOs from the FPGA and power control signals of the VR4131.

Figure 4-8: I/O board connector pin assignment

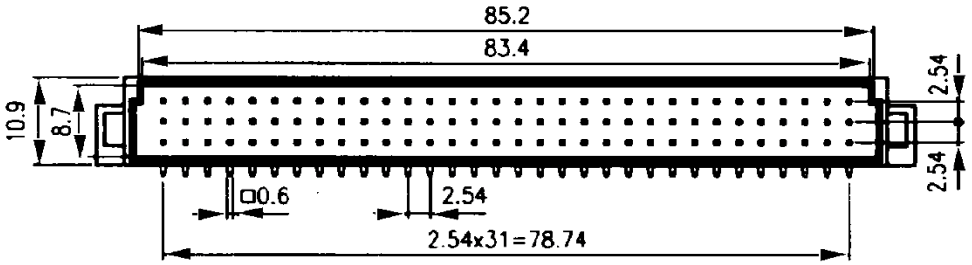


A1		B1		C1		D1		E1	
A2		B2		C2	GND	D2		E2	GND
A3	VDD5.0	B3	INTA2-N	C3	VDD5.0	D3	VDD5.0	E3	INTE-N
A4	INTC2-N	B4	VDD5.0	C4	INTB2-N	D4	INTD2-N	E4	GPIO1
A5		B5	VDD3.3	C5		D5		E5	GPIO11
A6		B6		C6		D6		E6	GPIO12
A7		B7		C7		D7		E7	GPIO13
A8	RST-N	B8	VDD3.3	C8	GND	D8	PCLK	E8	GND
A9	GNT2-N	B9	GND	C9	GND	D9	REQ2-N	E9	GND
A10		B10	AD30	C10	VDD3.3	D10	AD31	E10	FPIO0
A11	VDD3.3	B11	AD28	C11	AD29	D11	GND	E11	FPIO1
A12	AD26	B12	GND	C12	AD27	D12	AD25	E12	FPIO2
A13	AD24	B13	IDSEL2	C13	VDD3.3	D13	CBE3	E13	FPIO3
A14	VDD3.3	B14	AD22	C14	AD23	D14	GND	E14	GND
A15	AD20	B15	GND	C15	AD21	D15	AD19	E15	
A16	AD18	B16	AD16	C16	VDD3.3	D16	AD17	E16	
A17	VDD3.3	B17	FRAME-N	C17	CBE2	D17	GND	E17	
A18	GND	B18	TRDY-N	C18	IRDY-N	D18	VDD3.3	E18	
A19	GND	B19	STOP-N	C19	DEVSEL-N	D19	GND	E19	GND
A20	VDD3.3	B20		C20	LOCK-N	D20	PERR-N	E20	MPOWER
A21		B21	GND	C21	VDD3.3	D21	SERR-N	E21	SPOWER
A22	PAR	B22	AD15	C22	VDD3.3	D22	CBE1	E22	POWERON
A23	VDD3.3	B23	AD13	C23	AD14	D23	GND	E23	
A24	AD11	B24	GND	C24	AD12	D24	AD10	E24	
A25	AD9	B25	GND	C25		D25	GND	E25	
A26	GND	B26	CBE0	C26	GND	D26	AD8	E26	
A27	VDD3.3	B27	AD6	C27	AD7	D27	VDD3.3	E27	
A28	AD4	B28	GND	C28	AD5	D28	AD3	E28	
A29	AD2	B29	AD0	C29	GND	D29	AD1	E29	
A30	VDD3.3	B30		C30	VDD3.3	D30		E30	
A31	VDD5.0	B31	VDD5.0	C31	VDD5.0	D31	VDD5.0	E31	
A32		B32		C32		D32		E32	

4.4.11 Ravin Board Connector CN8

The other “specialised” connector CN8 uses a 96-pin (3x32) VME-style connector, that carries the CPU’s buffered address and data bus and I/O control signals. Three CPU GPIOs are routed to the connector as well. The intention of CN8 is to connect to a board with the RAVIN-E graphic display controller (µPD72255), that is offered as *start*WARE-GHS-RAVINE.

Figure 4-9: Ravin board connector pin assignment



A1	VDD5.0	B1	VDD5.0	C1	VDD5.0
A2	GND	B2	BD31	C2	BD30
A3	BD29	B3	BD28	C3	BD27
A4	BD26	B4	BD25	C4	BD24
A5	GND	B5	BD23	C5	BD22
A6	BD21	B6	BD20	C6	BD19
A7	BD18	B7	BD17	C7	BD16
A8	GND	B8	BD15	C8	BD14
A9	BD13	B9	BD12	C9	BD11
A10	BD10	B10	BD9	C10	BD8
A11	GND	B11	BD7	C11	BD6
A12	BD5	B12	BD4	C12	BD3
A13	BD2	B13	BD1	C13	BD0
A14	GND	B14	BEB0	C14	BEB1
A15	BEB2	B15	BEB3	C15	
A16	RESET	B16		C16	
A17	GND	B17	BA23	C17	BA22
A18	BA21	B18	BA20	C18	BA19
A19	BA18	B19	BA17	C19	BA16
A20	GND	B20	BA15	C20	BA14
A21	BA13	B21	BA12	C21	BA11
A22	BA10	B22	BA9	C22	BA8
A23	GND	B23	BA7	C23	BA6
A24	BA5	B24	BA4	C24	BA3
A25	BA2	B25	RDY	C25	
A26	GND	B26	BIOCS0_N	C26	BIOCS1_N
A27	GND	B27	BWR_N	C27	BRD_N
A28		B28		C28	
A29	GPIO8	B29	GPIO9	C29	GPIO10
A30		B30	VDD3.3	C30	
A31	GND	B31	GND	C31	GND
A32		B32	GND	C32	

4.5 FPGA Register Set

The FPGA has seventeen registers that allow to control extra I/O ports, a LED display and several other functionalities. This chapter explains the register programming functions register by register. The processor should access these registers with uncached load/store operations at the addresses shown in the table below:

Table 4-19: FPGA register set

Address	R/W	Register Symbol	Function
0x0A00 0000	R	REVREG	FPGA Revision
0x0A00 0004	R/W	MODE_REG	Modes for internal functions
0x0A00 0008	R/W	LEDREG	LED control register
0x0A00 000C	R	DIPSWITCHREG	DIP switch register
0x0A00 0010	R/W	FLASHACCREG	Flash access register
0x0A00 0014	R/W	CSICONTREG	CSI control register for additional control signals
0x0A00 0018	R/W	FPGAINTREG	Interrupt cause from FPGA
0x0A00 001C	R/W	FPGAINTENREG	Interrupt enable from FPGA
0x0A00 0020	R/W	FPGAINTPOLREG	Interrupt polarity from FPGA
0x0A00 0060	R/W	EPIO_MODEREG	EPIO input/output setting register
0x0A00 0064	R/W	EPIO_DATAREG	EPIO input/output data register
0x0A00 0068	R/W	FPIO_MODEREG	FPIO input/output setting register
0x0A00 006C	R/W	FPIO_DATAREG	FPIO input/output data register
0x0A00 0070	R/W	FPGA_I_O_MODEREGL	FPGA_I_O input/output setting lower register
0x0A00 0074	R/W	FPGA_I_O_DATAREGL	FPGA_I_O input/output data lower register
0x0A00 0078	R/W	FPGA_I_O_MODEREGH	FPGA_I_O input/output setting higher register
0x0A00 007C	R/W	FPGA_I_O_DATAREGH	FPGA_I_O input/output data higher register

4.5.1 REVREG (0x0A00 0000)

Bit	15	14	13	12	11	10	9	8
Name	FPGA_REV (7)	FPGA_REV (6)	FPGA_REV (5)	FPGA_REV (4)	FPGA_REV (3)	FPGA_REV (2)	FPGA_REV (1)	FPGA_REV (0)
R/W	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	1

Bit	7	6	5	4	3	2	1	0
Name	FPGA _SREV(3)	FPGA _SREV (2)	FPGA _SREV (1)	FPGA _SREV (0)	BOARD _REV (3)	BOARD _REV (2)	BOARD _REV (1)	BOARD _REV (0)
R/W	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:8	FPGA_REV (7:0)	Main Revision of the FPGA FPGA_REV (7:0) is decoded as following: FPGA_REV (7:0) = 0000 0001: Rev.: 1.x FPGA_REV (7:0) = 0000 0010: Rev.: 2.x others: reserved
7:4	FPGA_SREV (3:0)	Sub Revision of the FPGA FPGA_SREV (3:0) is decoded as following: FPGA_SREV (3:0) = 0000: Rev.: y.0 others: reserved
3:0	BOARD_REV (3:0)	Revision of the Board (startWARE-GHS-VR4131) BOARD_REV (3:0) is decoded as following: BOARD_REV (3:0) = 0000: Rev.: 1.0 BOARD_REV (3:0) = 0001: Rev.: 2.0 BOARD_REV (3:0) = 0010: Reserved BOARD_REV (3:0) = 0011: Reserved BOARD_REV (3:0) = 0100: Reserved BOARD_REV (3:0) = 0101: Reserved BOARD_REV (3:0) = 0110: Reserved BOARD_REV (3:0) = 0111: Reserved BOARD_REV (3:0) = 1000: Reserved BOARD_REV (3:0) = 1001: Reserved BOARD_REV (3:0) = 1010: Reserved BOARD_REV (3:0) = 1011: Reserved BOARD_REV (3:0) = 1100: Reserved BOARD_REV (3:0) = 1101: Reserved BOARD_REV (3:0) = 1110: Reserved BOARD_REV (3:0) = 1111: Reserved

4.5.2 MODE_REG (0x0A00 0004)

Bit	15	14	13	12	11	10	9	8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	Reserved	Reserved	Reserved	POWERMODE	Reserved	Reserved	Reserved	LEDMODE
R/W	R	R	R	R/W	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

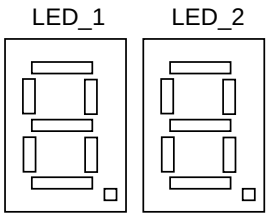
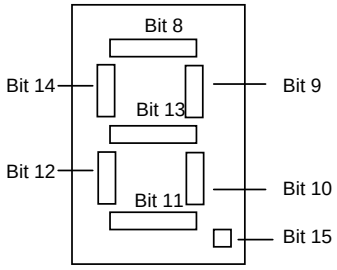
This register determines, how information in the LED register is used to control the 7-segment display on the board. The user can select between a BCD decoder and a bit-wise assignment of LED segments.
Furthermore the VR4131 can control by writing to this register, whether the SDRAM is switched off when the processor goes to hibernate mode, or not.

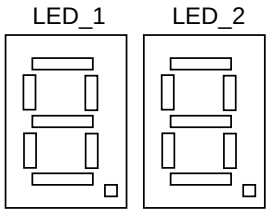
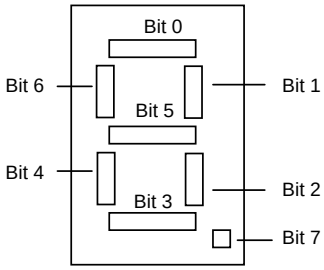
Bit	Name	Function	
15:5	Reserved		
4	POWERMODE	POWERMODE = 0:	MPOWER controls SDRAM power supply (SDRAM is switched off, when VR4131 goes to hibernate mode)
		POWERMODE = 1:	SPOWER controls SDRAM power supply (SDRAM is **not** switched off, when VR4131 goes to hibernate mode)
3:1	Reserved		
0	LEDMODE	LEDMODE = 0:	LED (7:0) will be decoded (binary to 7 segment)
		LEDMODE = 1:	LED (15:8) control Segments of LED – left side LED (7:0) control Segments of LED – right side

4.5.3 LEDREG (0x0A00 0008)

Bit	15	14	13	12	11	10	9	8
Name	LED15	LED 14	LED 13	LED 12	LED 11	LED 10	LED 9	LED 8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	LED 7	LED 6	LED 5	LED 4	LED 3	LED 2	LED 1	LED 0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	Name	Function
(15:8)	LED (15:8)	 If LEDMODE = 0, LED (15:8) are not used. The written data can be read, but will be ignored for LED control. If LEDMODE = 1, LED (15:8) controls LED1 directly 

Bit	Name	Function
(7:0)	LED (7:0)	 If LEDMODE = 0, LED (7:4) will be decoded as following: LED (7:4)= 0000: LED_1(7:0) = 1010_0000 0 LED (7:4)= 0001: LED_1(7:0) = 1111_1001 1 LED (7:4)= 0010: LED_1(7:0) = 1100_0100 2 LED (7:4)= 0011: LED_1(7:0) = 1101_0000 3 LED (7:4)= 0100: LED_1(7:0) = 1001_1001 4 LED (7:4)= 0101: LED_1(7:0) = 1001_0010 5 LED (7:4)= 0110: LED_1(7:0) = 1000_0010 6 LED (7:4)= 0111: LED_1(7:0) = 1111_1000 7 LED (7:4)= 1000: LED_1(7:0) = 1000_0000 8 LED (7:4)= 1001: LED_1(7:0) = 1001_0000 9 LED (7:4)= 1010: LED_1(7:0) = 1000_1000 A LED (7:4)= 1011: LED_1(7:0) = 1000_0011 b LED (7:4)= 1100: LED_1(7:0) = 1010_0110 C LED (7:4)= 1101: LED_1(7:0) = 1100_0001 d LED (7:4)= 1110: LED_1(7:0) = 1000_0110 E LED (7:4)= 1111: LED_1(7:0) = 1000_1110 F If LEDMODE = 0, LED (3:0) will be decoded as following: LED (3:0)= 0000: LED_2(7:0) = 1010_0000 0 LED (3:0)= 0001: LED_2(7:0) = 1111_1001 1 LED (3:0)= 0010: LED_2(7:0) = 1100_0100 2 LED (3:0)= 0011: LED_2(7:0) = 1101_0000 3 LED (3:0)= 0100: LED_2(7:0) = 1001_1001 4 LED (3:0)= 0101: LED_2(7:0) = 1001_0010 5 LED (3:0)= 0110: LED_2(7:0) = 1000_0010 6 LED (3:0)= 0111: LED_2(7:0) = 1111_1000 7 LED (3:0)= 1000: LED_2(7:0) = 1000_0000 8 LED (3:0)= 1001: LED_2(7:0) = 1001_0000 9 LED (3:0)= 1010: LED_2(7:0) = 1000_1000 A LED (3:0)= 1011: LED_2(7:0) = 1000_0011 b LED (3:0)= 1100: LED_2(7:0) = 1010_0110 C LED (3:0)= 1101: LED_2(7:0) = 1100_0001 d LED (3:0)= 1110: LED_2(7:0) = 1000_0110 E LED (3:0)= 1111: LED_2(7:0) = 1000_1110 F If LEDMODE = 1, LED (7:0) controls LED_2 directly. 

4.5.4 DIPSWITCHREG (0x0A00 000C)

Bit	15	14	13	12	11	10	9	8
Name	SW1-1	SW1-2	SW1-3	SW1-4	SW1-5	SW1-6	SW1-7	SW1-8
R/W	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	SW2-1	SW2-2	SW2-3	SW2-4	SW2-5	SW2-6	SW2-7	SW2-8
R/W	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

The DIPSWITCHREG register is a read only register that reflects the setting of the switches SW1 and SW2. Please see chapter 4.3 for the meaning of the different switch settings. The register is updated according to current switch positions any time, when it is read by the processor; this ensures that the processor “sees” the current switch position.

4.5.5 FLASHACCREG (0x0A00 0010)

Bit	15	14	13	12	11	10	9	8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	FRDY	VPEN
R/W	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

This register controls write access to the on-board flash memory. If jumper JP7 is set, the VR4131 can set the VPEN pins on the on-board flash memories to active level by writing to this register. Furthermore the processor can check the programming status with the FRDY bit.

Note: The FPGA hardware sets the FRDY bit to “0” after reset. However you will normally read a “1” from this bit, because it reflects the FRDY input of the FPGA which is externally pulled-up.

Bit	Name	Function	
15:2	Reserved		
1	FRDY	FRDY pin input data	1: High level 0: Low level
0	VPEN	VPEN pin output data	1: High level 0: Low level

4.5.6 CSICONTREG (0x0A00 0014)

Bit	15	14	13	12	11	10	9	8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	Reserved	Reserved	Reserved	Reserved	RQC1	RQC0	GNTC1	GNTC0
R/W	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

The CSICONTREG provides a software handshake for multi-master configurations on the CSI interface of the VR4131. This register reads the RQC(1:0) pins and writes the GNTC(1:0) pins. The correspondence is 1 bit per pin.
Writing a value to the RQC(1:0) bit does not affect the RQC(1:0) pin (the write data is ignored). When reading the CSICONTREG, the RQC(1:0) pin's state and the written GNTC(1:0) bits will be read.

Bit	Name	Function	
15:4	Reserved		
3	RQC1	RQC1 pin input data	1: High level 0: Low level
2	RQC0	RQC0 pin input data	1: High level 0: Low level
1	GNTC1	GNTC1 pin output data	1: High level 0: Low level
0	GNTC0	GNTC0 pin output data	1: High level 0: Low level

4.5.7 FPGAINTREG (0x0A00 0018)

Bit	15	14	13	12	11	10	9	8
Name	INTA1_N	INTB1_N	INTC1_N	INTD1_N	INTA2_N	INTB2_N	INTC2_N	INTD2_N
R/W	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	INTE_N	FPIO4	RQC0	RQC1	Reserved	Reserved	Reserved	LWAKE
R/W	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

This register reflects the current value of the interrupt signals of the PCI slot, the I/O Board and the on-board Ethernet controller. The interrupt-input signals are forwarded to the CPU via GPIO2, GPIO3, GPIO4, GPIO5 and GPIO7. The LWAKE signal (Board Rev. 1.0 only) is not forwarded to the CPU; if processing is required, the VR4131 must poll this register.

Bit	Name	Function
15	INTA1_N **Note 1**	Interrupt INTA1_N from PCI slot
14	INTB1_N **Note 1**	Interrupt INTB1_N from PCI slot
13	INTC1_N **Note 1**	Interrupt INTC1_N from PCI slot
12	INTD1_N **Note 1**	Interrupt INTD1_N from PCI slot
11	INTA2_N **Note 1**	Interrupt INTA2_N from I/O-Board
10	INTB2_N **Note 1**	Interrupt INTB2_N from I/O-Board
9	INTC2_N **Note 1**	Interrupt INTC2_N from I/O-Board
8	INTD2_N **Note 1**	Interrupt INTD2_N from I/O-Board
7	INTE_N **Note 1**	Interrupt INTE_N from I/O-Board
6	FPIO4 **Note 1**	Interrupt signal (FPIO4) from Ethernet Controller RTL8139C
5	RQC0 **Note 1**	Request RQC0 from CSI interface
4	RQC1 **Note 1**	Request RQC1 from CSI interface
3:1	Reserved	
0	LWAKE **Note 2**	Wake up signal from Ethernet Controller

Notes: 1. Implemented since FPGA Rev.2.0 and usable for Board Rev 2.0 and later.

2. Implemented since FPGA Rev.1.0 and usable for Board Rev 1.0 only.

4.5.8 FPGAINTENREG (0x0A00 001C)

Bit	15	14	13	12	11	10	9	8
Name	INTA1_EN	INTB1_EN	INTC1_EN	INTD1_EN	INTA2_EN	INTB2_EN	INTC2_EN	INTD2_EN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
Name	INTE_EN	FPIO4_EN	RQC0_EN	RQC1_EN	Reserved	Reserved	Reserved	Reserved
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	0	0	0	0	0	0

This register allows to enable/disable the interrupt signals coming from the PCI slot, the I/O Board and the on-board Ethernet controller. The interrupt-input signals are forwarded to the CPU via GPIO2, GPIO3, GPIO4, GPIO5 and GPIO7. Additional the RQC0 and RQC1 signal can be used as an interrupt request source. These signals are forwarded to the GPIO7 interrupt.

When this bit is set to 1, the corresponding pin is enabled. The default value is selected to secure the same functionality as Rev1.x. This register is implemented since FPGA Rev.2.0 and allows disabling unused interrupt signals.

Bit	Name	Function	
15	INTA1_EN **Note**	Interrupt enable for INTA1_N from PCI slot.	1: interrupt enabled 0: interrupt disabled
14	INTB1_EN **Note**	Interrupt enable for INTB1_N from PCI slot.	1: interrupt enabled 0: interrupt disabled
13	INTC1_EN **Note**	Interrupt enable for INTC1_N from PCI slot.	1: interrupt enabled 0: interrupt disabled
12	INTD1_EN **Note**	Interrupt enable for INTD1_N from PCI slot.	1: interrupt enabled 0: interrupt disabled
11	INTA2_EN **Note**	Interrupt enable for INTA2_N from I/O-Board.	1: interrupt enabled 0: interrupt disabled
10	INTB2_EN **Note**	Interrupt enable for INTB2_N from I/O-Board.	1: interrupt enabled 0: interrupt disabled
9	INTC2_EN **Note**	Interrupt enable for INTC2_N from I/O-Board.	1: interrupt enabled 0: interrupt disabled
8	INTD2_EN **Note**	Interrupt enable for INTD2_N from I/O-Board.	1: interrupt enabled 0: interrupt disabled
7	INTE_EN **Note**	Interrupt enable for INTE_N from I/O-Board.	1: interrupt enabled 0: interrupt disabled
6	FPIO4_EN **Note**	Interrupt enable for FPIO4 signal from Ethernet Controller RTL8139C	1: interrupt enabled 0: interrupt disabled
5	RQC0_EN **Note**	Enable interrupts on RQC0 signal from CSI interface.	1: interrupt enabled 0: interrupt disabled
4	RQC1_EN **Note**	Enable interrupts on RQC1 signal from CSI interface.	1: interrupt enabled 0: interrupt disabled
3:0	Reserved		

Note: Implemented since FPGA Rev.2.0 and usable for Board Rev 2.0 and later.

4.5.9 FPGAINTPOLREG (0x0A00 0020)

Bit	15	14	13	12	11	10	9	8
Name	INTA1_ POL	INTB1_ POL	INTC1_ POL	INTD1_ POL	INTA2_ POL	INTB2_ POL	INTC2_ POL	INTD2_ POL
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	INTE_ POL	FPIO4_ POL	RQC0_ POL	RQC1_ POL	Reserved	Reserved	Reserved	Reserved
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	1	1	0	0	0	0

This register allows selecting the polarity of the interrupt signals coming from the PCI slot, the I/O Board and the on-board Ethernet controller. The interrupt-input signals are forwarded to the CPU via GPIO2, GPIO3, GPIO4, GPIO5 and GPIO7. Additional the RQC0 and RQC1 signal can be used as an interrupt request source. These signals are forwarded to the GPIO7 interrupt.
When this bit is set to 0, the corresponding pin is active low. This register is implemented since FPGA Rev.2.0 and the default value is selected to secure the same functionality as Rev1.x.

Bit	Name	Function	
15	INTA1_POL **Note**	Interrupt polarity of INTA1_N from PCI slot.	1: active high 0: active low
14	INTB1_ POL **Note**	Interrupt polarity of INTB1_N from PCI slot.	1: active high 0: active low
13	INTC1_ POL **Note**	Interrupt polarity of INTC1_N from PCI slot.	1: active high 0: active low
12	INTD1_ POL **Note**	Interrupt polarity of INTD1_N from PCI slot.	1: active high 0: active low
11	INTA2_ POL **Note**	Interrupt polarity of INTA2_N from I/O-Board.	1: active high 0: active low
10	INTB2_ POL **Note**	Interrupt polarity of INTB2_N from I/O-Board	1: active high 0: active low
9	INTC2_ POL **Note**	Interrupt polarity of INTC2_N from I/O-Board.	1: active high 0: active low
8	INTD2_ POL **Note**	Interrupt polarity of INTD2_N from I/O-Board	1: active high 0: active low
7	INTE_ POL **Note**	Interrupt polarity of INTE_N from I/O-Board.	1: active high 0: active low
6	FPIO4_ POL **Note**	Interrupt polarity of FPIO4 signal from Ethernet Controller RTL8139C	1: active high 0: active low
5	RQC0_ POL **Note**	Interrupt polarity of RQC0 signal from CSI interface.	1: active high 0: active low
4	RQC1_ POL **Note**	Interrupt polarity of RQC1 signal from CSI interface	1: active high 0: active low
3:0	Reserved		

Note: Implemented since FPGA Rev.2.0 and usable for Board Rev 2.0 and later.

Caution: The interrupt output signals of the FPGA (GPIO2, GPIO3, GPIO4, GPIO5, and GPIO7) are always low active.

4.5.10 EPIO_MODEREG (0x0A00 0060)

Bit	15	14	13	12	11	10	9	8
Name	EPIOMOD (15)	Reserved	EPIOMOD (13)	EPIOMOD (12)	EPIOMOD (11)	EPIOMOD (10)	EPIOMOD (9)	EPIOMOD (8)
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	EPIOMOD (1)	Reserved
R/W	R	R	R	R	R	R	R/W	R
Reset	0	0	0	0	0	0	0	0

This register sets the I/O mode of the EPIO(n) pins of the FPGA. The correspondence is 1 bit per pin. When the EPIOMOD bit is set to 1, the corresponding EPIO pin is set to output and the value that has been written to the corresponding EPIO bit in the EPIO_REG register is output. When this bit is set to 0, the corresponding EPIO pin is set to input.
Note that the bit pattern of this register is adapted to available I/Os on the VR4131.

Bit	Name	Function	
15	EPIOMOD (15)	EPIOMOD (15) =	0: EPIO (15) is input 1: EPIO (15) is output
14	Reserved		
13:8	EPIOMOD (13:8)	EPIOMOD (13:8) =	0: EPIO (13:8) is input 1: EPIO (13:8) is output
7:2	Reserved		
1	EPIOMOD (1)	EPIOMOD (1) =	0: EPIO (1) is input 1: EPIO (1) is output
0	Reserved		

4.5.11 EPIO_REG (0x0A00 0064)

Bit	15	14	13	12	11	10	9	8
Name	EPIO (15)	Reserved	EPIO (13)	EPIO (12)	EPIO (11)	EPIO (10)	EPIO (9)	EPIO (8)
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	EPIO (1)	Reserved
R/W	R	R	R	R	R	R	R/W	R
Reset	0	0	0	0	0	0	0	0

This register reads and/or writes the EPIO(n) pins. The correspondence is 1 bit per pin. When 1 is set to the corresponding bit in the EPIO_MODEREG register, the data in EPIO_REG is written to the EPIO pin.
When the value of the corresponding bit in the EPIO_MODEREG register is 0, writing a value to the EPIO bit does not affect the EPIO pin (the write data is ignored).
When the value of the corresponding bit in the EPIO_MODEREG register is 0, reading the EPIO bit enables the corresponding EPIO pin's state to be read.

Bit	Name	Function
15	EPIO (15)	EPIO(15) pin input/output data. 1: High level 0: Low level
14	Reserved	
13:8	EPIO (13:8)	EPIO(13:8) pin input/output data. 1: High level 0: Low level
7:2	Reserved	
1	EPIO (1)	EPIO(1) pin input/output data. 1: High level 0: Low level
0	Reserved	

4.5.12 FPIO_MODEREG (0x0A00 0068)

Bit	15	14	13	12	11	10	9	8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	Reserved	Reserved	FPIOMOD (5)	FPIOMOD (4)	FPIOMOD (3)	FPIOMOD (2)	FPIOMOD (1)	FPIOMOD (0)
R/W	R	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register sets the I/O mode of the FPIO(3:0) pins. The correspondence is 1 bit per pin. When the FPIOMOD bit is set to 1, the corresponding FPIO pin is set to output and the value that has been written to the corresponding FPIO bit in the FPIO_REG register is output.
When this bit is set to 0, the corresponding FPIO pin is set to input.

Bit	Name	Function	
15:6	Reserved		
5	FPIOMOD (5)	always 0:	FPIO(5) is always input FPIO(5) shows the status of the LWAKE_CSTSCHG Pin of the RTL8139C – Ethernet Controller (Board Rev 1.0 only) or the status of the INTD2_N Pin of the I/O Board (Board Rev 2.0 and later)
4	FPIOMOD (4)	always 0:	FPIO(4) is always input FPIO(4) shows the status of the INTAB Pin of the RTL8139C – Ethernet Controller
3:0	FPIOMOD (3:0)	FPIOMOD (3:0) =	0: FPIO (3:0) is input 1: FPIO (3:0) is output

4.5.13 FPIO_REG (0x0A00 006C)

Bit	15	14	13	12	11	10	9	8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	Reserved	Reserved	FPIO (5)	FPIO (4)	FPIO (3)	FPIO (2)	FPIO (1)	FPIO (0)
R/W	R	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register reads and/or writes the FPIO(5:0) pins. The correspondence is 1 bit per pin. When 1 is set to the corresponding bit in the FPIO_MODEREG register, the data in FPIO_REG is written to the corresponding FPIO pin.
When the value of the corresponding bit in the FPIO_MODEREG register is 0, writing a value to the FPIO bit does not affect the FPIO pin (the write data is ignored).
When the value of the corresponding bit in the FPIO_MODEREG register is 0, reading the FPIO bit enables the corresponding FPIO pin's state to be read.

Note: FPIO(4) and FPIO(5) can only be used as input pins and they are already hard-wired on the board to pins of the Ethernet controller as shown in the table below. FPIO(3:0) pins are available on the I/O board connector CN9.

Bit	Name	Function	
15:6	Reserved		
5	FPIO (5)	FPIO(5) pin input data.	1: High level 0: Low level FPIO(5) shows the value of the LWAKE_CSTSCHG Pin of the RTL8139C – Ethernet Controller (Board Rev 1.0 only)
4	FPIO (4)	FPIO(4) pin input data.	1: High level 0: Low level FPIO(4) shows the value of the INTAB Pin of the RTL8139C – Ethernet Controller
3:0	FPIO (3:0)	FPIO(3:0) pin input/output data.	1: High level 0: Low level

4.5.14 FPGA_I_O_MODEREGL (0x0A00 0070)

Bit	15	14	13	12	11	10	9	8
Name	FPGA_I_O_MOD (15)	FPGA_I_O_MOD (14)	FPGA_I_O_MOD (13)	FPGA_I_O_MOD (12)	FPGA_I_O_MOD (11)	FPGA_I_O_MOD (10)	FPGA_I_O_MOD (9)	FPGA_I_O_MOD (8)
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	FPGA_I_O_MOD (7)	FPGA_I_O_MOD (6)	FPGA_I_O_MOD (5)	FPGA_I_O_MOD (4)	FPGA_I_O_MOD (3)	FPGA_I_O_MOD (2)	FPGA_I_O_MOD (1)	FPGA_I_O_MOD (0)
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register sets the I/O mode of the FPGA_I_O (15:0) pins. The correspondence is 1 bit per pin.
When the FPGA_I_O_MOD bit is set to 1, the corresponding FPGA_I_O pin is set to output and the value that has been written to the corresponding FPGA_I_O bit in the FPGA_I_O_REGL register is output.
When this bit is set to 0, the corresponding FPGA_I_O pin is set to input.

Bit	Name	Function
15:0	FPGA_I_O_MOD (15:0)	FPGA_I_O_MOD (15:0) = 0: FPGA_I_O (15:0) is input 1: FPGA_I_O (15:0) is output

4.5.15 FPGA_I_O_REGL (0x0A00 0074)

Bit	15	14	13	12	11	10	9	8
Name	FPGA_I_O (15)	FPGA_I_O (14)	FPGA_I_O (13)	FPGA_I_O (12)	FPGA_I_O (11)	FPGA_I_O (10)	FPGA_I_O (9)	FPGA_I_O (8)
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	FPGA_I_O (7)	FPGA_I_O (6)	FPGA_I_O (5)	FPGA_I_O (4)	FPGA_I_O (3)	FPGA_I_O (2)	FPGA_I_O (1)	FPGA_I_O (0)
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register reads and/or writes the FPGA_I_O (15:0) pins. The correspondence is 1 bit per pin. When 1 is set to the corresponding bit in the FPGA_I_O_MODEREGL register, the corresponding data is written to the FPGA_I_O pin.
When the value of the corresponding bit in the FPGA_I_O_MODEREGL register is 0, writing a value to the FPGA_I_O bit does not affect the FPGA_I_O pin (the write data is ignored).
When the value of the corresponding bit in the FPGA_I_O_MODEREGL register is 0, reading the FPGA_I_O bit enables the corresponding FPGA_I_O pin's state to be read.

Bit	Name	Function
15:0	FPGA_I_O (15:0)	FPGA_I_O (15:0) pin input/output data. 1: High level 0: Low level

4.5.16 FPGA_I_O_MODEREGH (0x0A00 0078)

Bit	15	14	13	12	11	10	9	8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	Reserved	Reserved	Reserved	FPGA_I_O_MOD (20)	FPGA_I_O_MOD (19)	FPGA_I_O_MOD (18)	FPGA_I_O_MOD (17)	FPGA_I_O_MOD (16)
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register sets the I/O mode of the FPGA_I_O (20:16) pins. The correspondence is 1 bit per pin. When the FPGA_I_O_MOD bit is set to 1, the corresponding FPGA_I_O pin is set to output and the value that has been written to the corresponding FPGA_I_O bit in the FPGA_I_O_REGH register is output.
When this bit is set to 0, the corresponding FPGA_I_O pin is set to input.

Bit	Name	Function
15:6	Reserved	
5:0	FPGA_I_O_MOD (20:16)	FPGA_I_O_MOD (20:16) = 0: FPGA_I_O (20:16) is input 1: FPGA_I_O (20:16) is output

4.5.17 FPGA_I_O_REGH (0x0A00 007C)

Bit	15	14	13	12	11	10	9	8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	Reserved	Reserved	Reserved	FPGA_I_O (20)	FPGA_I_O (19)	FPGA_I_O (18)	FPGA_I_O (17)	FPGA_I_O (16)
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register reads and/or writes the FPGA_I_O (20:16) pins. The correspondence is 1 bit per pin.
When 1 is set to the corresponding bit in the FPGA_I_O_MODEREGH register, the corresponding data is written to the FPGA_I_O pin.
When the value of the corresponding bit in the FPGA_I_O_MODEREGH register is 0, writing a value to the FPGA_I_O bit does not affect the FPGA_I_O pin (the write data is ignored).
When the value of the corresponding bit in the FPGA_I_O_MODEREGH register is 0, reading the FPGA_I_O bit enables the corresponding FPGA_I_O pin's state to be read.

Bit	Name	Function
15:6	Reserved	
5:0	FPGA_I_O (20:16)	FPGA_I_O (20:16) pin input/output data. 1: High level 0: Low level

Chapter 5 Board Operation

This chapter explains practical usage of the *start*WARE-GHS-VR4131 evaluation board.

5.1 Getting Started

This chapter gives a “step-by-step” description, how to start up the board.

- Check if all jumpers are in their default position
- Check if all DIP switches are in their default position
- Connect the *start*WARE-GHS-VR4131 Evaluation board with the serial cable to a host PC with a terminal emulation program (115200 baud, 8 bit, no parity, 1 stop bit, no handshake); use the VR4131 Debug Serial interface on CN17 for this purpose
- Connect the power supply to the board
- Start the terminal emulation program
- Operate power switch SW3 (LEDs D1, D2 and D7 should be activated)
- FPGA configuration will be done, which is indicated by LED D3
- LED D6 shows that the automatically generated RTCRST is active
- Push POWER button SW6 (LEDs D8 and D10 should be activated).

Now the board is ready to communicate with the host PC using the S-Boot monitor program that is described in the next chapter.

5.2 S-Boot Operation

5.2.1 Features

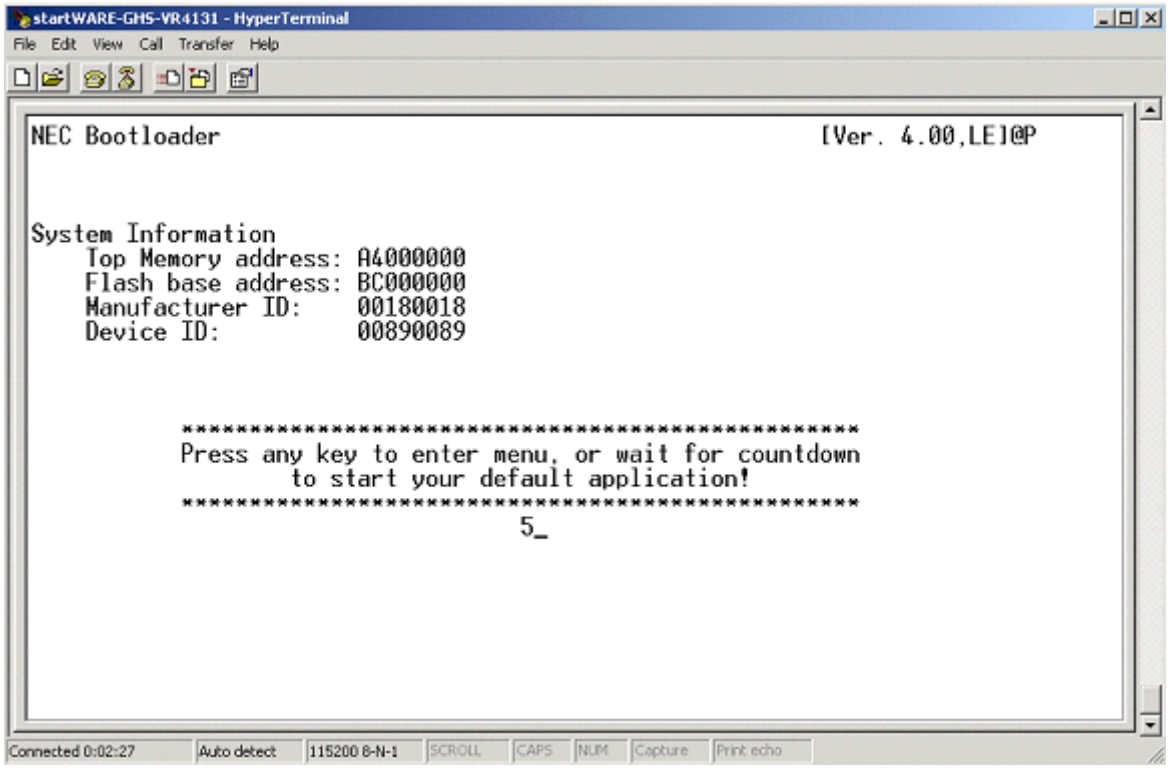
A simple boot monitor named SBOOT is implemented on the *start*WARE-GHS-VR4131 starter kit. This monitor offers the following features:

- A fast serial connection using the CPUs DSIU at 115200 Baud,
- Download capabilities to SDRAM,
- Download to FLASH,
- Automatic, programmable start of applications available in RAM or FLASH,
- A small Flash File System with simple directory structure.

5.2.2 S-Boot Startup Message

Once the starter kit is powered up turned on, you will find this screen on the host PC.

Figure 5-1: Startup Screen



The screen gives information about the current system configuration, please see Figure 5-1.

Remarks: **1.** Version information (Example - 4.0)

2. Endianess:
LE - Little Endian,
BE - Big Endian

3. Run mode:
@ - start from Flash
* - start from RAM

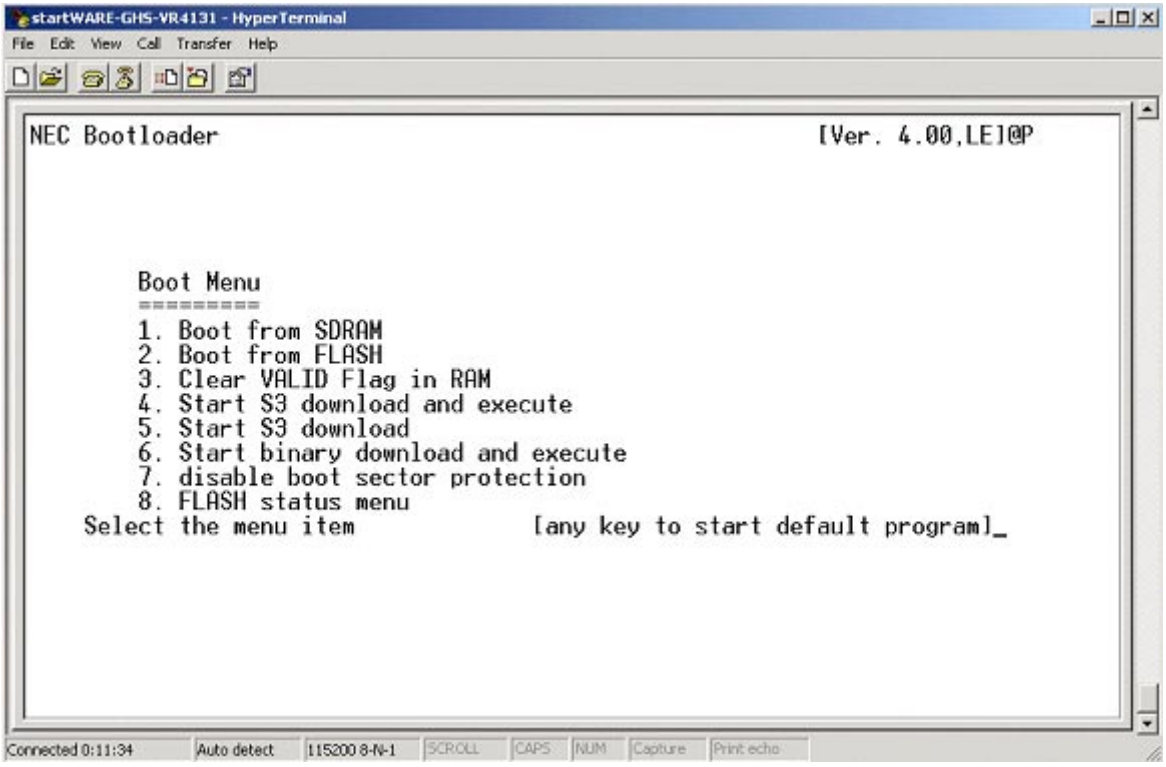
4. Protection settings:
P - Protected,
N - Not protected

The running countdown can be stopped by any input. Once the countdown reaches 0, the system will try to boot. In the first instance, it will try booting from SDRAM, if any information is available that there is something loaded. If none is found, it continues to check for information in the FLASH memory. If there is an application available in any of the FLASH locations, it starts this application. Please see item ‘Flowchart’ for a more detailed description of the procedure.

5.2.3 Flash Monitor of startWARE-GHS-VR4131

When the countdown is interrupted, the SBOOT stops boot procedure and an additional screen is shown:

Figure 5-2: startWARE-GHS-VR4131 Flash Monitor Boot Menu



This is the main menu of the system and provides access to various other options of the SBOOT. The user may manually boot from RAM or FLASH.

(1) Boot from SDRAM
SBOOT tries to boot from memory location last recently loaded with an application.

(2) Boot from FLASH
SBOOT tries to boot from a valid FLASH memory location available in the directory list and marked as ‘default’ start-up.

(3) Clear VALID Flag in RAM
makes an application located in main memory unavailable for next boot process.

(4) Start S3 download and execute
implies the option to download an S-record type file and start it directly after download.

(5) Start S3 download.
Behaves like item 4, but does not automatically start the application.

(6) Start binary download and execute.
Using this option starts a download of a raw binary file, recommended for larger images. Once this option is invoked, the user is asked for

a) a flash memory address, where the application is downloaded to,
b) a start address of the application.
This is necessary for images providing a different entry point than the first target address is.

(7) Disable boot sector protection
is not available for the Starter Kit.

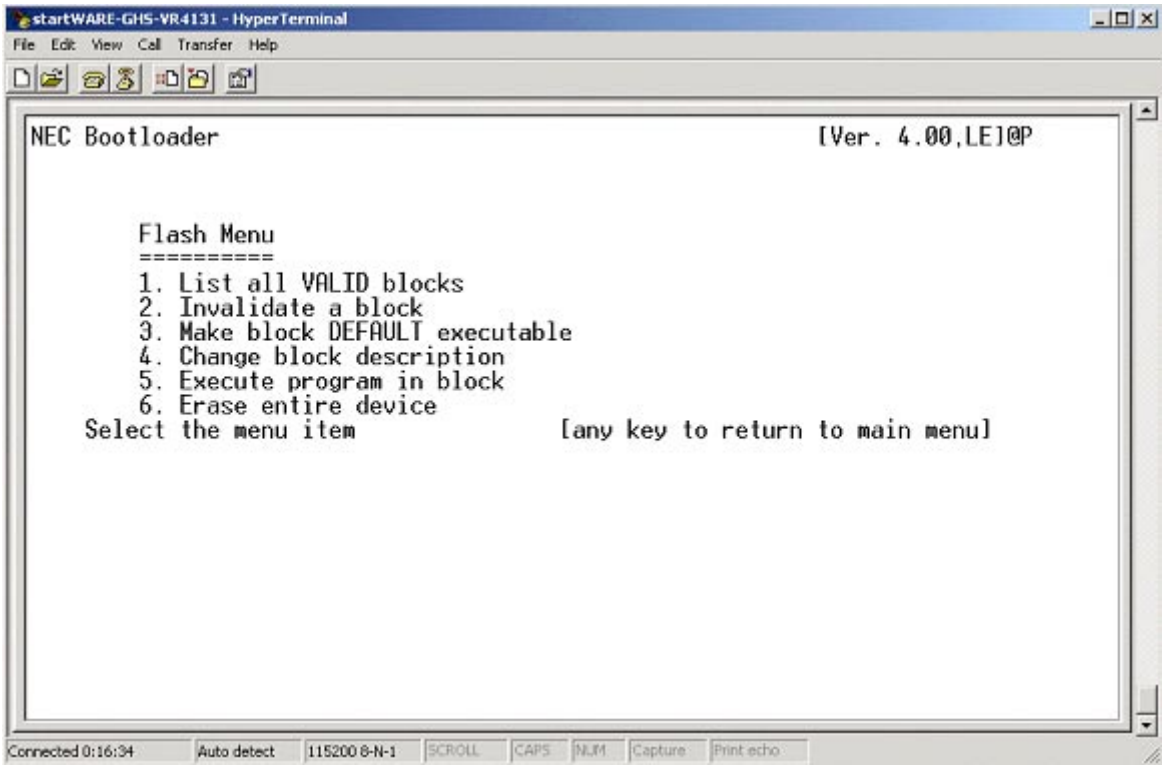
(8) FLASH status menu
opens a submenu handling the various FLASH options.

Note: The Flash memory is protected from accidental programming with jumper JP7. Before downloading a program to Flash, JP7 has to be inserted. We strongly recommend to remove it again after the flash memory has been programmed.

5.2.4 FLASH Options

There are several option available making the handling of the FLASH easier. Several target applications may reside in the FLASH memory and can be managed by SBOOT.

Figure 5-3: startWARE-GHS-VR4131 Flash Menu



(1) List all VALID blocks
displays a list of applications available in the FLASH memory. The default screen looks as shown in figure 5-4.

(2) Invalidate a block
disables the directory entry and makes the application unavailable for the next boot process.

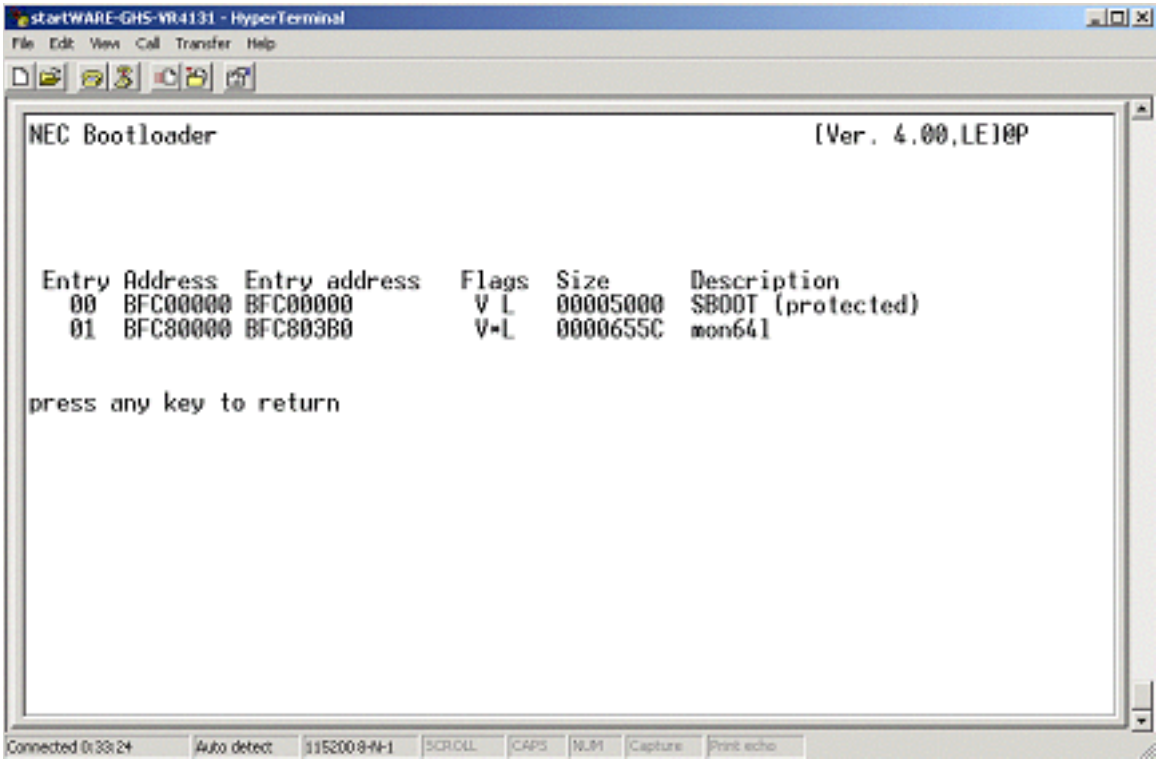
(3) Make block DEFAULT executable
gives the option to boot an application from a block specified by the User. The application must have a valid entry in the directory block structure, before it can be started by default boot process. In the Starter Kit, the GHS target monitor, here named mon64l is the default application to start with. Any last recently downloaded application to FLASH will automatically get the DEFAULT boot flag. This menu entry gives the User the option to change it manually.

(4) Change block description
of a downloaded application, if the automatically provided name is not sufficient. It is a simple text entry to name that directory entry into a more specific description. A maximum number of 32 characters is allowed here.

(5) Execute program in block
offers the option to start any of the available applications manually.

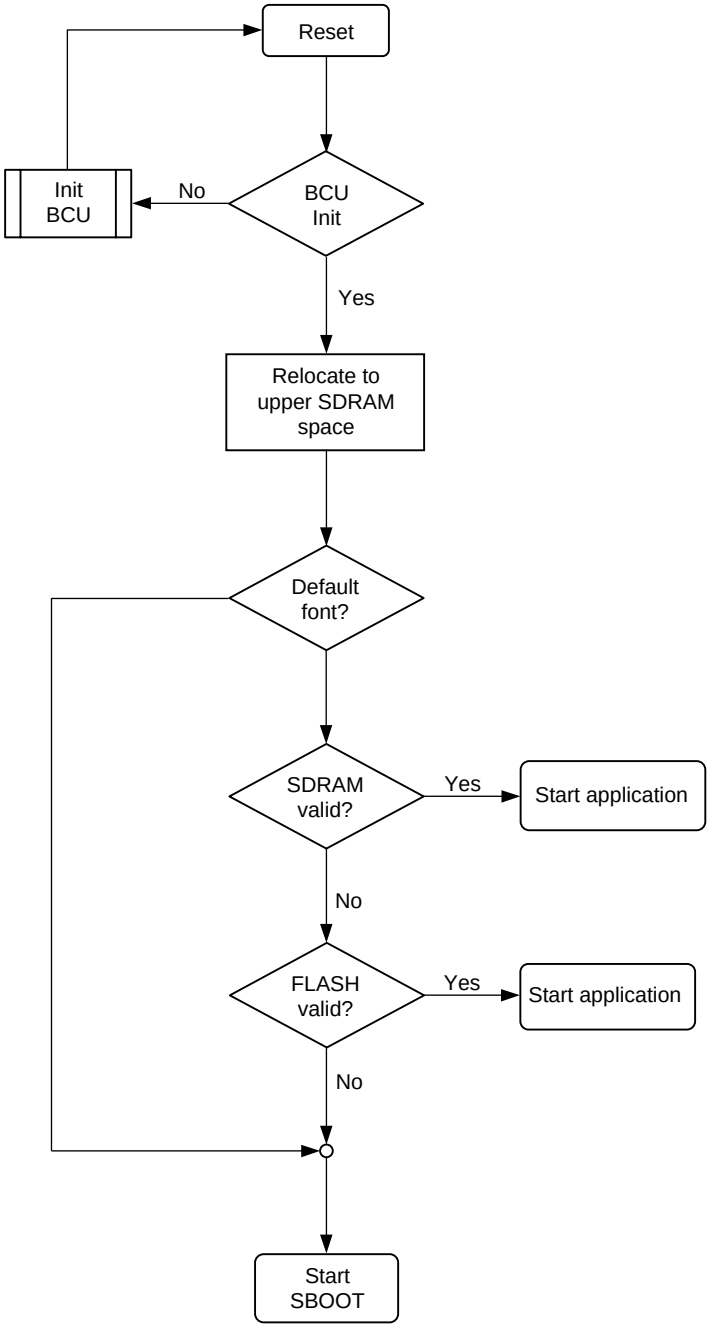
(6) The entire FLASH device is erased, except for the boot block. All applications are lost and the directory entry list is emptied as well.

Figure 5-4: startWARE-GHS-VR4131 Directory List



5.2.5 Reset Process Flow

Figure 5-5: startWARE-GHS-VR4131 Boot Flow Chart



5.2.6 SBOOT Specification

(1) Interrupts

SBOOT does not use any interrupt. The serial units are used in polling mode only. All interrupts are routed to addresses 0x80000000, 0x80000080, 0x80000100 and 0x80000180. It is recommended that all user applications are making usage of the BEV bit, so that own interrupt handling routines can be entered at the desired locations. The non-maskable interrupt (NMI) is treated as ordinary reset – no special handling implemented in here.

(2) Initialization

The caches are initialized prior to any download. A timer is not implemented. Only for the operation of SBOOT necessary peripheral units are supplied with a clock and are initialized.

(3) Memory Usage

The memory area used by SBOOT is from ranging from 0xA3FFB000 to 0xA3FFFFFF.

The SDRAM boot flags are stored in ‘Top memory location’ –0x8 and 0x4. Once an application is loaded into RAM, one address is containing the text ‘EXEC’; the other one contains the execution address of the last recently loaded application.

SBOOT allocates the reset vector location of the VR- CPU, i.e. the FLASH block located at address 0XBFC00000 cannot be used by any target applications. Usually, there is one more monitor delivered with the Starter Kit – the Green Hills target monitor located at 0xBFC80000. It is running from FLASH memory area, which prohibits a programming of the FLASH, unless the FLASH burning algorithm is located in SDRAM. In addition, SBOOT is protected by H/W bit, so that no accidental erase may happen.

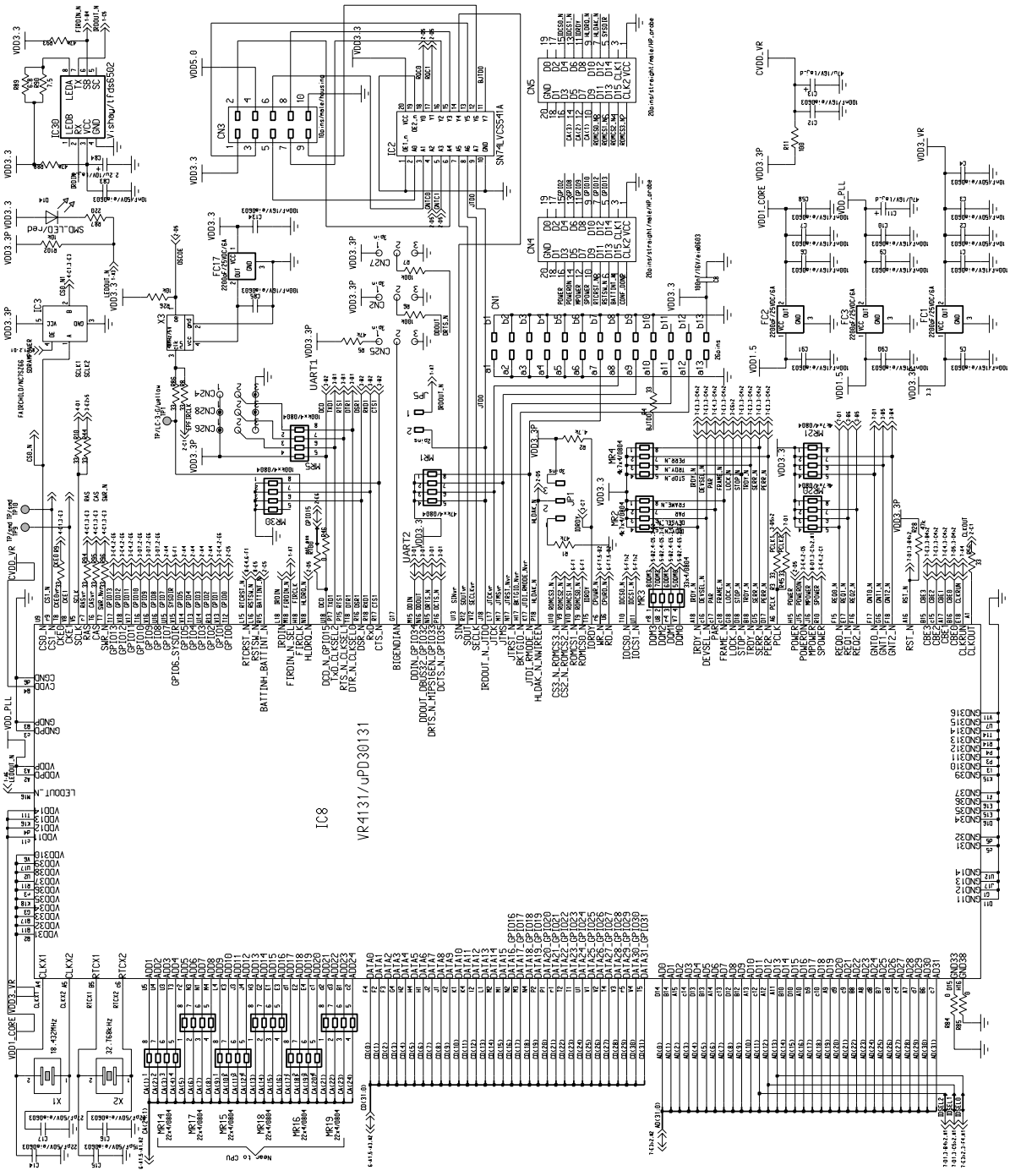
5.3 Reprogramming the FPGA

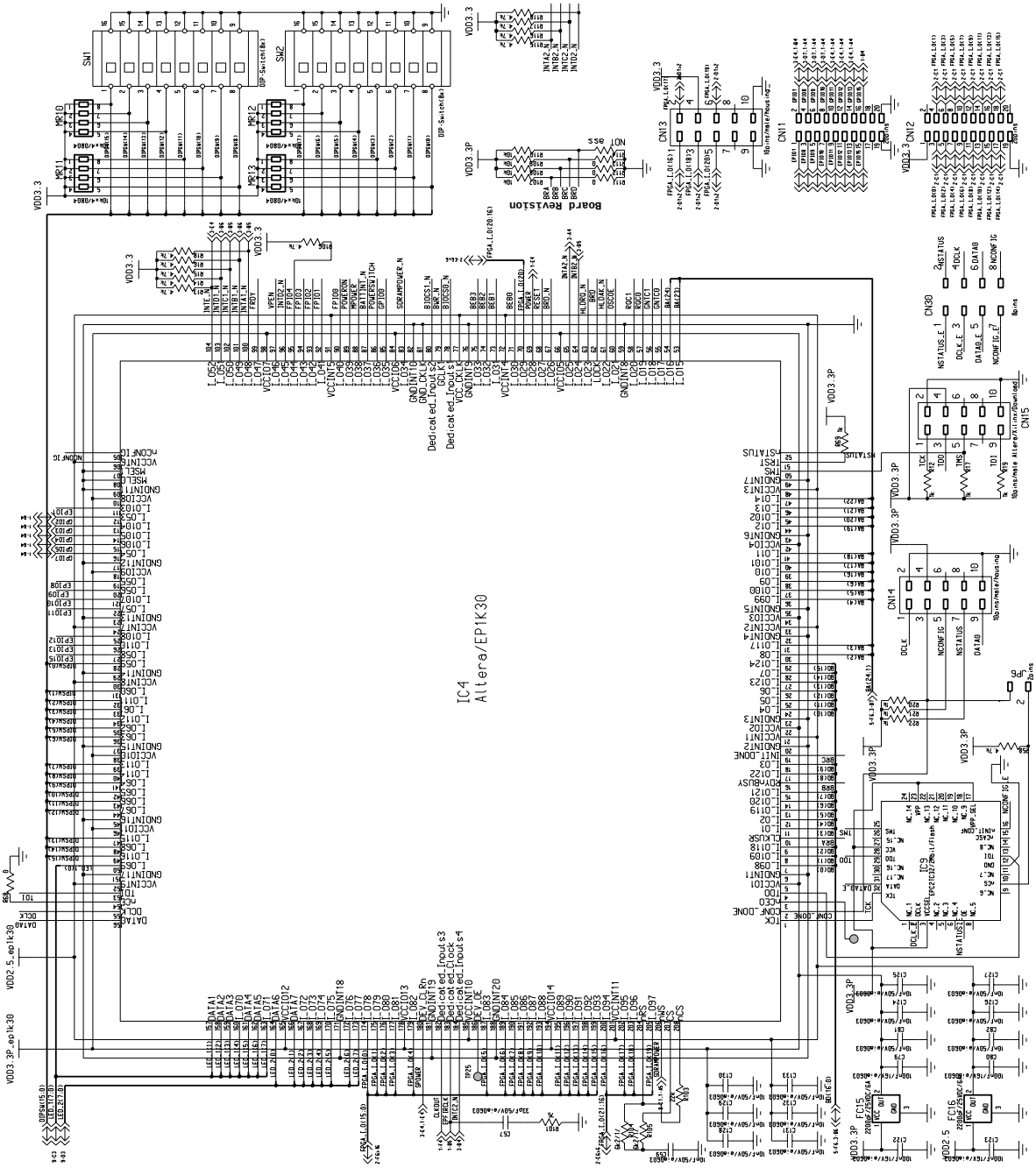
Caution: The functions described in this chapter should only be used by experienced customers. Reprogramming the FPGA in the wrong way, may damage the board. So watch your step!

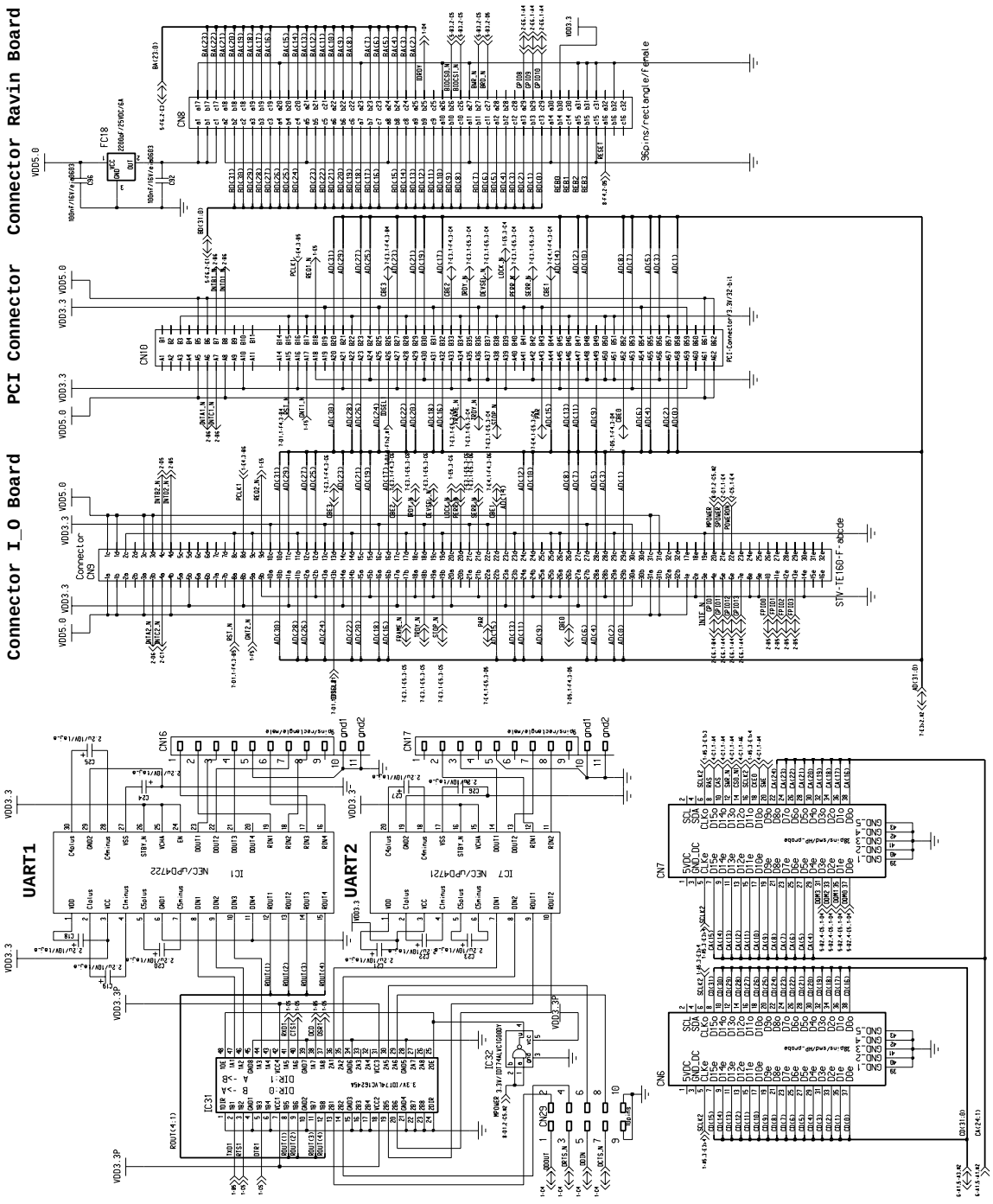
A short summary how the FPGA is reprogrammed is given below. The current FPGA content has been developed with the Quartus II (Rev. 2.0) tool which can be downloaded from the Altera website (www.altera.com). This tool is capable of downloading FPGA content via JTAG with special cables that are also available from Altera. (During development of the board the ByteBlasterMV™ cable was used.).

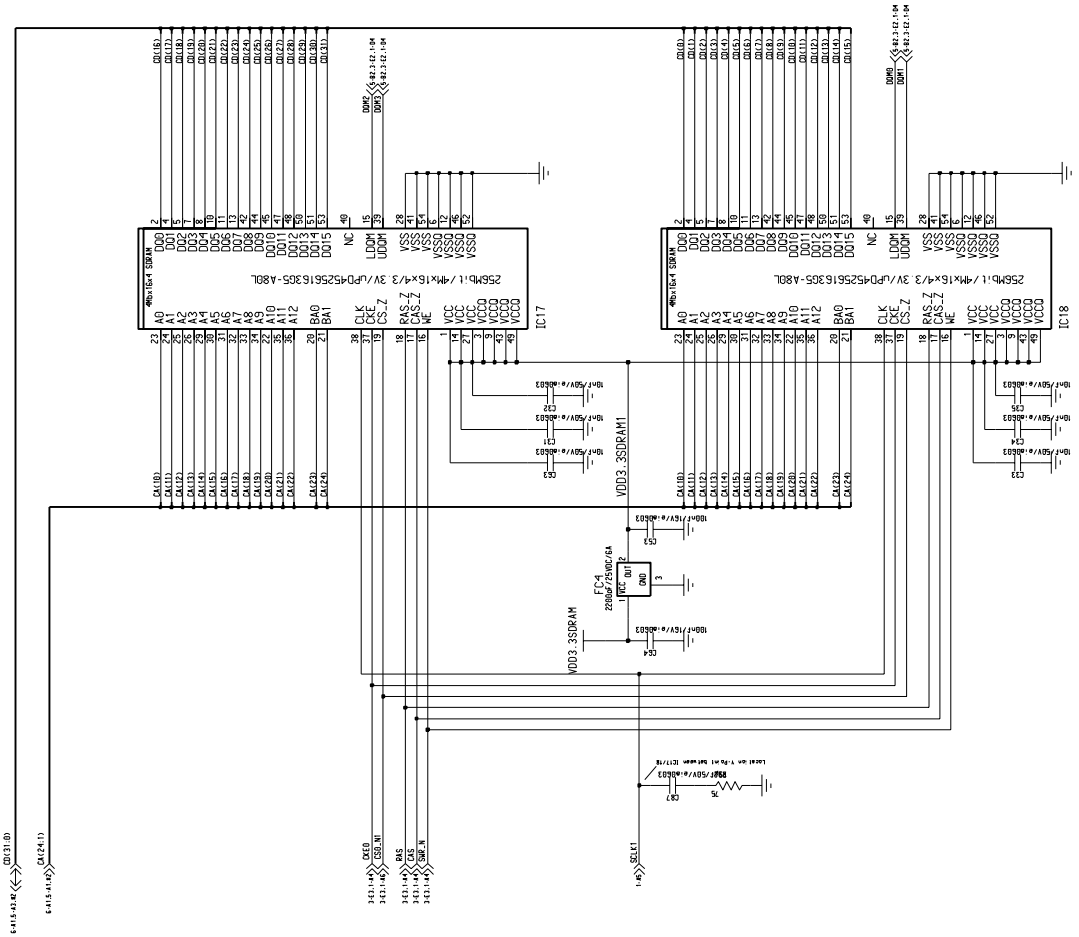
- Connect the *start*WARE-GHS-VR4131 evaluation board via the download cable to your host PC (CN15 on the evaluation board)
- Remove jumper JP6 on the evaluation board
- Power on the evaluation board with SW3
- Download the FPGA code
- Power off the evaluation board with SW3
- Remove the download cable and insert jumper JP6 again
- Power on the evaluation board with SW3 and the new FPGA content

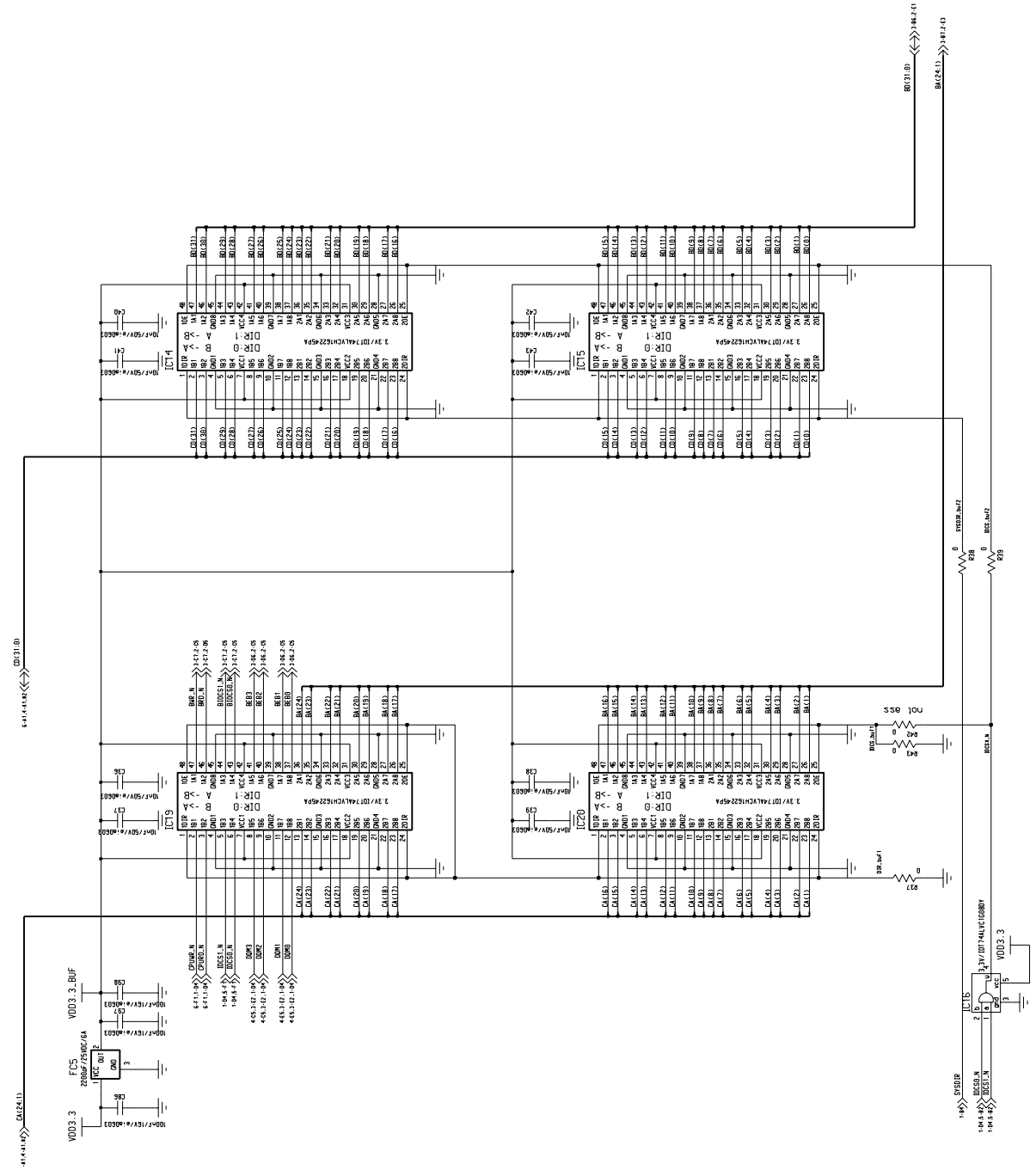
Appendix A Circuit Diagrams

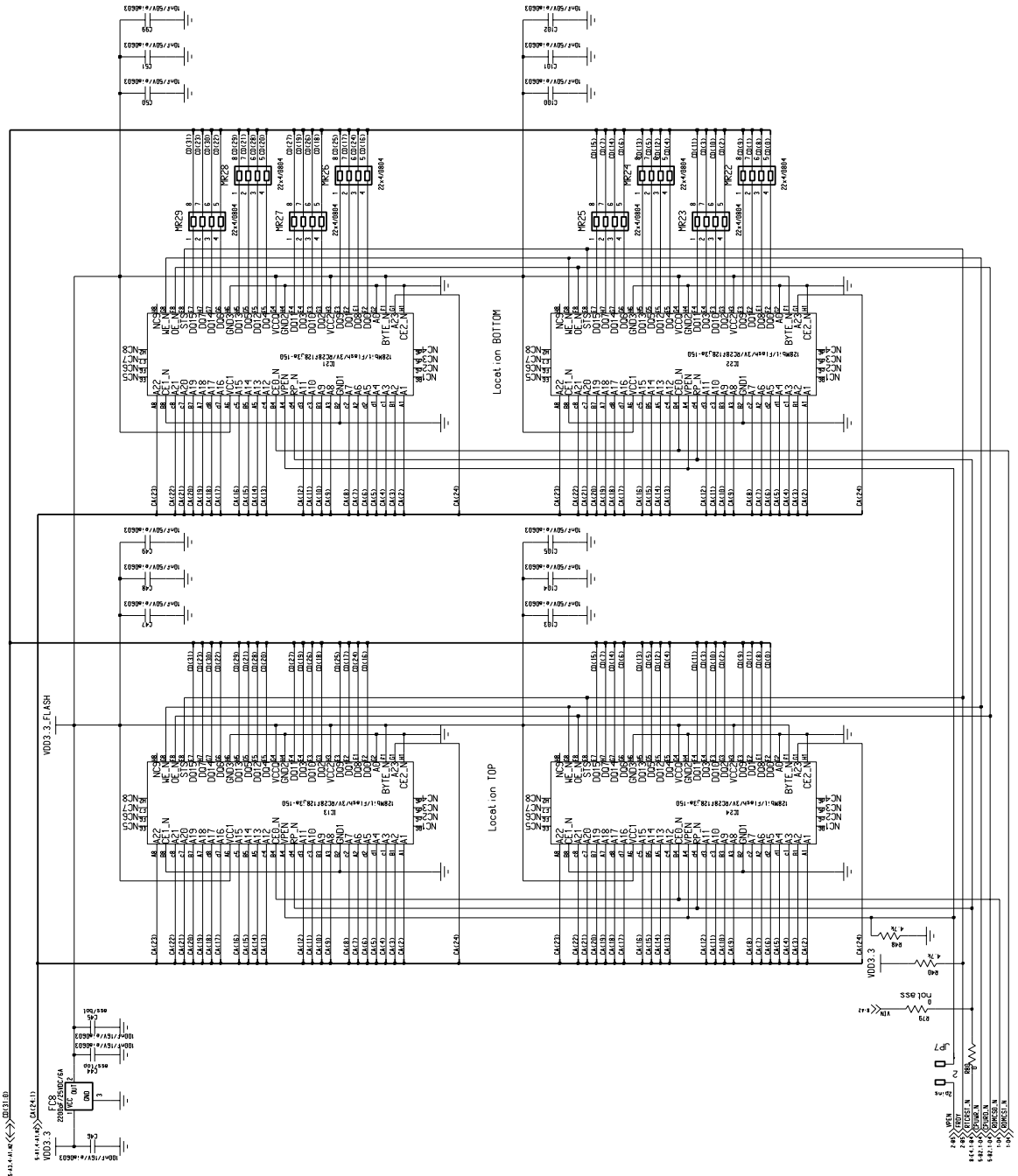


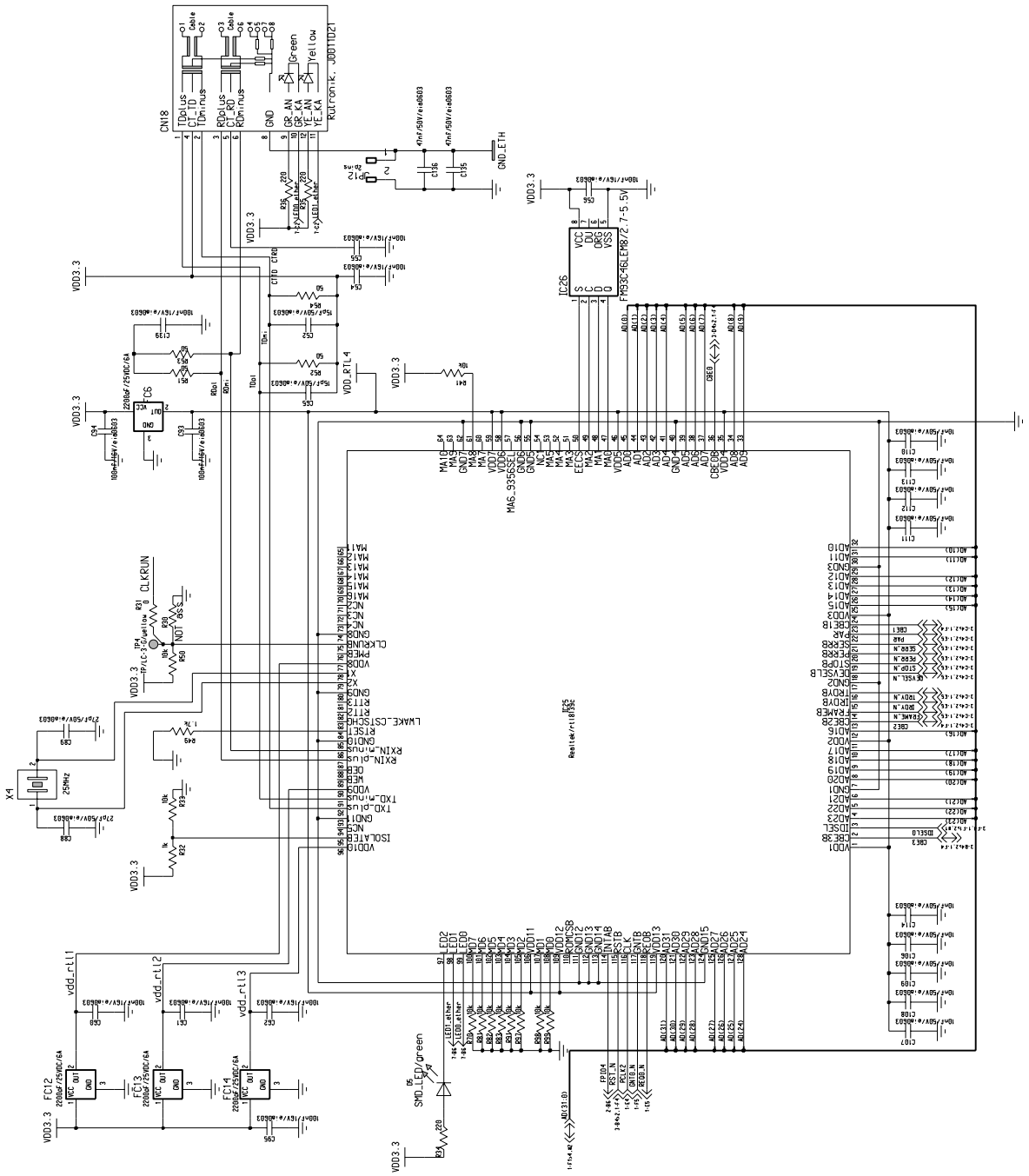

IC4
Altera/EP1K30
Board Revision
SW1
SW2
CN11
CN12
CN13
CN14
CN15
CN30
IC3
IC5
IC6
IC7
IC8
IC9
IC10
IC11
IC12
IC13
IC14
IC15
IC16
VDD3.3
VDD3.3P
VDD2.5
VDD2.5_eo1k30
VDD3.3P_eo1k30

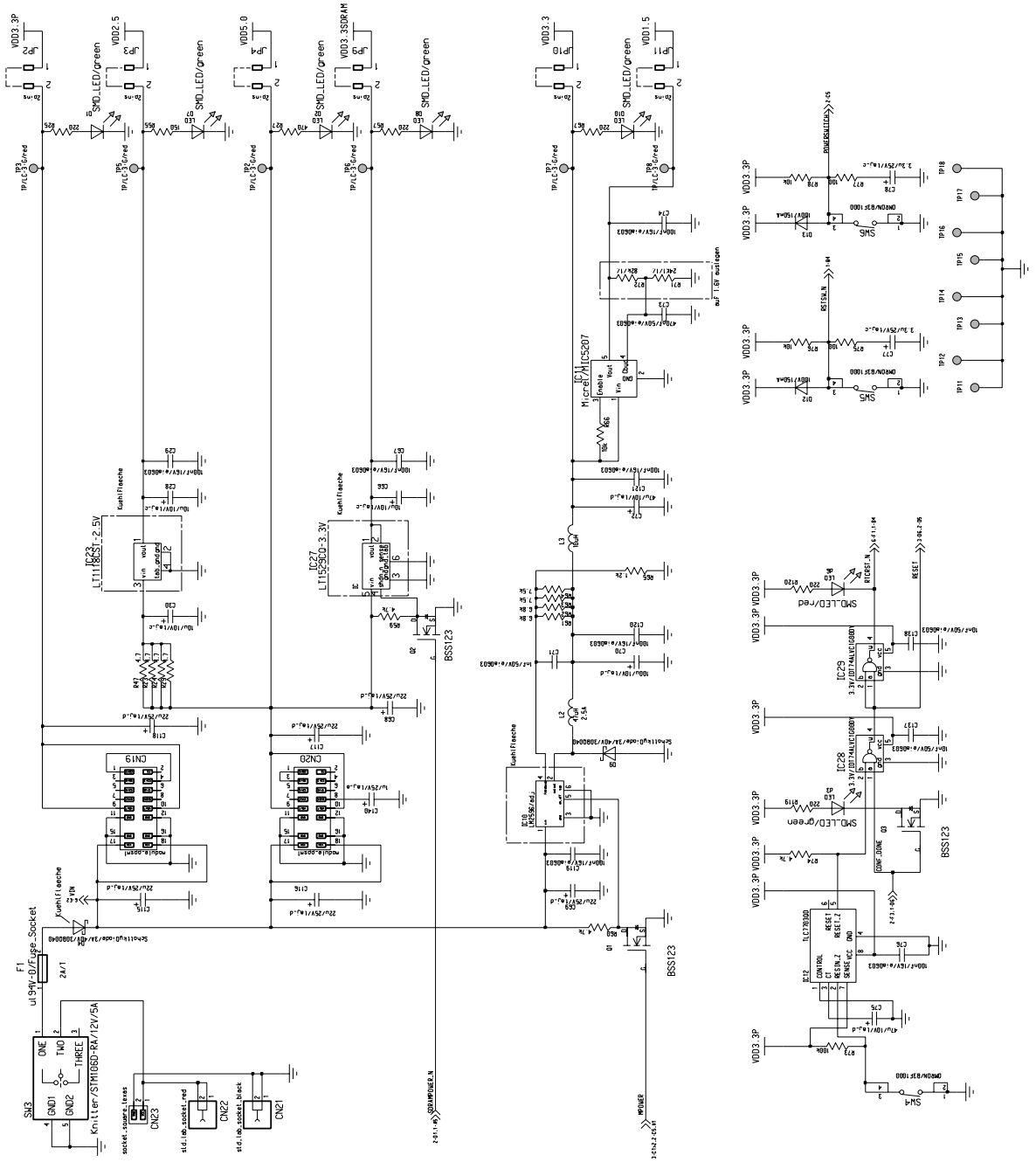

Connector I_O Board
PCI Connector
Connector Ravin Board
UART1
UART2
NEC/uPD4722
NEC/uPD4721
IC31
IC32
CN16
CN17
CN7
CN6
CN8
CN10
CN9
CN2
FC18
STV-TE160-F-abcde
PCI-Connector(3.3V/32-bit)
96pins/rectangle/female
VDD3.3
VDD5.0
VDD3.3P

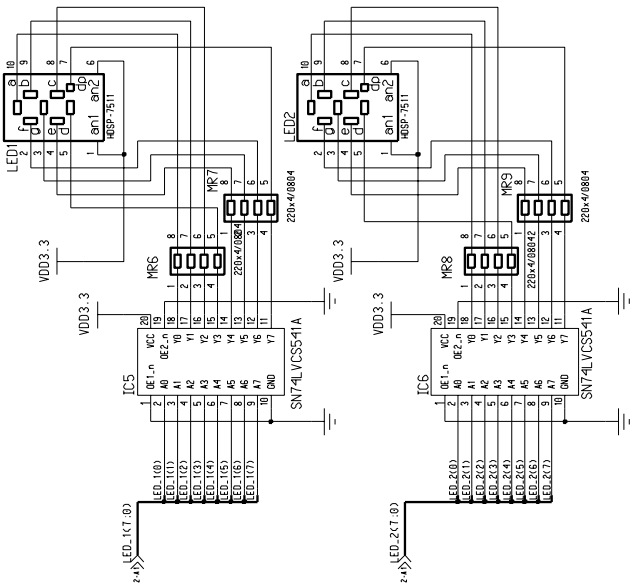





Location BOTTOM
Location TOP
VDD3.3_FLASH
VDD3.3
not ass
JP7






LED1
LED2
HDSP-7511
VDD3.3
MR6
MR7
MR8
MR9
220x4/0804
IC5
IC6
SN74LVCS541A
LED_1<7:0>
LED_2<7:0>

[MEMO]

Appendix B FPGA Code

The FPGA was designed using the Quartus II design tool available through www.altera.com, version V2.0. Below the source listing of the FPGA content is shown; however changing the FPGA content should only be done by experienced board users. The source files can as well be found on the CD.

B.1 File VR4131_TOP.V

```
//*********************************************
// DESIGNER    :Willi Nuesser
// OBJECT      :VR4131 Board
// DATE        :31. Oct. 2002
// REVISION    :2.0
//*********************************************

`include "VR4131DEF.h"

module VR4131_top(address_device, address_reg, data,
                            LEDout, DIPSWITCH, FPIO, CLKOUT, FPGA_I_O,
                            EPIO1, EPIO8, EPIO9, EPIO10, EPIO11, EPIO12, EPIO13,
                            EPIO15,BIOCS0_N, BIOCS1_N, BWR_N, BRD_N, RESET,
                            INTA1_N, INTB1_N, INTC1_N, INTD1_N, INTE_N,
                            GPIO0,GPIO2,GPIO3,GPIO4,GPIO5,GPIO7,BEB,
                            GNTC0, GNTC1, RQC0, RQC1, OSCOE, HLDAK_N, HLDRQ_N,
                            VPEN,FRDY,IORDY_INTA2_N,RDY_INTB2_N,
                            FPGA_IN_INTC2_N,INTD2_N, X1, X2, X3,
                            EPFIRCLK,POWER,MPOWER,SPOWER,POWERSWITCH,BATTINT_N,
                            POWERON,SDRAMPOWER, SDRAMPOWER_N, SPARE,
                            BOARD_REV);

        input  [24:16] address_device;
        input  [6:2] address_reg;
        inout  [15:0] data;
        inout  EPIO1, EPIO8, EPIO9, EPIO10, EPIO11, EPIO12, EPIO13, EPIO15;
        inout  [20:0] FPGA_I_O;
        inout  [4:0] FPIO;
        input  [3:0] BEB;
        input  BWR_N, BRD_N;
        input  BIOCS0_N, BIOCS1_N;
        input  RESET;
        input  [15:0] DIPSWITCH;
        output [15:0] LEDout;
        input  CLKOUT;
        input  INTA1_N, INTB1_N, INTC1_N, INTD1_N, INTE_N;
        input  GPIO0, POWERON, FPGA_IN_INTC2_N;
        output GPIO2,GPIO3,GPIO4,GPIO5,GPIO7;
        input  RQC0, RQC1, HLDAK_N, FRDY, RDY_INTB2_N,INTD2_N;
        output GNTC0, GNTC1, OSCOE, HLDRQ_N, VPEN;
        inout  IORDY_INTA2_N;
```

```
input  [3:0] BOARD_REV;

input  X1;
output        X2, X3;

input  EPFIRCLK, POWERSWITCH, MPOWER, SPOWER;
output POWER, BATTINT_N, SDRAMPOWER, SDRAMPOWER_N, SPARE;

reg    [19:0] clk_countreg_FIR;
reg [5:0] clk_countreg,clk_countreg1;
reg [11:0] led_test_reg;
reg clk10msreg, POWER_reg, BATTINT_reg;
reg [3:0] poweronseqreg;
reg pwrsw;
reg [3:0] pwrsw_cntr;
reg pwron_rst_done;
reg internal_reset;
reg BOARD_REV_0;
reg RDY_reg;

reg c, FPGA_READ_reg,IORDY_INTA2_N_reg;
reg [15:0] led_in_reg, MODE_REG, outreg, ledreg;
reg [15:0] int_en_reg, int_pol_reg;
reg [15:0] FPIO_reg, FPIO_modereg, EPIO_reg, EPIO_modereg,
FLASHACCreg, CSICONTreg;
reg [20:0] FPGA_I_O_reg, FPGA_I_O_modereg;

wire RC_CLK, FPGA_RESET;
wire SPARE = 0;
wire clk10ms = clk10msreg;
wire   INTA1_N_PE, INTB1_N_PE, INTC1_N_PE, INTD1_N_PE,
       IORDY_INTA2_N_PE,
              RDY_INTB2_N_PE, FPGA_IN_INTC2_N_PE,
              INTD2_N_PE, INTE_N_PE,
              FPIO4_PE, RQC0_PE, RQC1_PE,
              INTA1_N_P, INTB1_N_P, INTC1_N_P, INTD1_N_P,
              IORDY_INTA2_N_P,
              RDY_INTB2_N_P, FPGA_IN_INTC2_N_P,INTD2_N_P, INTE_N_P,
              FPIO4_P, RQC0_P, RQC1_P;
```

```
       integer i;

//***********************************************
//Tristated ports
//***********************************************

PORT16 data_port (.DIN(outreg[15:0]),.DOUT(data[15:0]),.EN(!BRD_N & FPGA_READ_reg & MPOWER));

PORT1 FPGA_I_O_0 (.DIN(FPGA_I_O_reg[0]),.DOUT(FPGA_I_O[0]),.EN(FPGA_I_O_modereg[0]));
PORT1 FPGA_I_O_1 (.DIN(FPGA_I_O_reg[1]),.DOUT(FPGA_I_O[1]),.EN(FPGA_I_O_modereg[1]));
PORT1 FPGA_I_O_2 (.DIN(FPGA_I_O_reg[2]),.DOUT(FPGA_I_O[2]),.EN(FPGA_I_O_modereg[2]));
PORT1 FPGA_I_O_3 (.DIN(FPGA_I_O_reg[3]),.DOUT(FPGA_I_O[3]),.EN(FPGA_I_O_modereg[3]));
PORT1 FPGA_I_O_4 (.DIN(FPGA_I_O_reg[4]),.DOUT(FPGA_I_O[4]),.EN(FPGA_I_O_modereg[4]));
PORT1 FPGA_I_O_5 (.DIN(FPGA_I_O_reg[5]),.DOUT(FPGA_I_O[5]),.EN(FPGA_I_O_modereg[5]));
PORT1 FPGA_I_O_6 (.DIN(FPGA_I_O_reg[6]),.DOUT(FPGA_I_O[6]),.EN(FPGA_I_O_modereg[6]));
PORT1 FPGA_I_O_7 (.DIN(FPGA_I_O_reg[7]),.DOUT(FPGA_I_O[7]),.EN(FPGA_I_O_modereg[7]));
PORT1 FPGA_I_O_8 (.DIN(FPGA_I_O_reg[8]),.DOUT(FPGA_I_O[8]),.EN(FPGA_I_O_modereg[8]));
PORT1 FPGA_I_O_9 (.DIN(FPGA_I_O_reg[9]),.DOUT(FPGA_I_O[9]),.EN(FPGA_I_O_modereg[9]));
PORT1 FPGA_I_O_10(.DIN(FPGA_I_O_reg[10]),.DOUT(FPGA_I_O[10]),.EN(FPGA_I_O_modereg[10]));
PORT1 FPGA_I_O_11 (.DIN(FPGA_I_O_reg[11]),.DOUT(FPGA_I_O[11]),.EN(FPGA_I_O_modereg[11]));
PORT1 FPGA_I_O_12 (.DIN(FPGA_I_O_reg[12]),.DOUT(FPGA_I_O[12]),.EN(FPGA_I_O_modereg[12]));
PORT1 FPGA_I_O_13 (.DIN(FPGA_I_O_reg[13]),.DOUT(FPGA_I_O[13]),.EN(FPGA_I_O_modereg[13]));
PORT1 FPGA_I_O_14 (.DIN(FPGA_I_O_reg[14]),.DOUT(FPGA_I_O[14]),.EN(FPGA_I_O_modereg[14]));
PORT1 FPGA_I_O_15 (.DIN(FPGA_I_O_reg[15]),.DOUT(FPGA_I_O[15]),.EN(FPGA_I_O_modereg[15]));
PORT1 FPGA_I_O_16 (.DIN(FPGA_I_O_reg[16]),.DOUT(FPGA_I_O[16]),.EN(FPGA_I_O_modereg[16]));
PORT1 FPGA_I_O_17 (.DIN(FPGA_I_O_reg[17]),.DOUT(FPGA_I_O[17]),.EN(FPGA_I_O_modereg[17]));
PORT1 FPGA_I_O_18 (.DIN(FPGA_I_O_reg[18]),.DOUT(FPGA_I_O[18]),.EN(FPGA_I_O_modereg[18]));
PORT1 FPGA_I_O_19 (.DIN(FPGA_I_O_reg[19]),.DOUT(FPGA_I_O[19]),.EN(FPGA_I_O_modereg[19]));
PORT1 FPGA_I_O_20 (.DIN(FPGA_I_O_reg[20]),.DOUT(FPGA_I_O[20]),.EN(FPGA_I_O_modereg[20]));

PORT1 FPIO_0 (.DIN(FPIO_reg[0]),.DOUT(FPIO[0]),.EN(FPIO_modereg[0]));
PORT1 FPIO_1 (.DIN(FPIO_reg[1]),.DOUT(FPIO[1]),.EN(FPIO_modereg[1]));
PORT1 FPIO_2 (.DIN(FPIO_reg[2]),.DOUT(FPIO[2]),.EN(FPIO_modereg[2]));
PORT1 FPIO_3 (.DIN(FPIO_reg[3]),.DOUT(FPIO[3]),.EN(FPIO_modereg[3]));
PORT1 FPIO_4 (.DIN(FPIO_reg[4]),.DOUT(FPIO[4]),.EN(FPIO_modereg[4]));

PORT1 EPIO_1 (.DIN(EPIO_reg[1]),.DOUT(EPIO1),.EN(EPIO_modereg[1]));
PORT1 EPIO_8 (.DIN(EPIO_reg[8]),.DOUT(EPIO8),.EN(EPIO_modereg[8]));
PORT1 EPIO_9 (.DIN(EPIO_reg[9]),.DOUT(EPIO9),.EN(EPIO_modereg[9]));
PORT1 EPIO_10 (.DIN(EPIO_reg[10]),.DOUT(EPIO10),.EN(EPIO_modereg[10]));
PORT1 EPIO_11 (.DIN(EPIO_reg[11]),.DOUT(EPIO11),.EN(EPIO_modereg[11]));
PORT1 EPIO_12 (.DIN(EPIO_reg[12]),.DOUT(EPIO12),.EN(EPIO_modereg[12]));
PORT1 EPIO_13 (.DIN(EPIO_reg[13]),.DOUT(EPIO13),.EN(EPIO_modereg[13]));
PORT1 EPIO_15 (.DIN(EPIO_reg[15]),.DOUT(EPIO15),.EN(EPIO_modereg[15]));

//PORT1 IORDY_INTA2 (.DIN(IORDY_INTA2_N_reg),.DOUT(IORDY_INTA2_N),.EN(BOARD_REV_0));
PORT1 IORDY_INTA2 (.DIN(RDY_INTB2_N),.DOUT(IORDY_INTA2_N),.EN(BOARD_REV_0));
```

```
//***********************************************
//some initializations
//***********************************************

        RCOSC osc (1, X1, X2, X3);

        assign RC_CLK = X1;
        assign FPGA_RESET = RESET | internal_reset;

//***********************************************
//Write to FPGA
//***********************************************

always @(posedge BWR_N or posedge FPGA_RESET)
        begin
        if (FPGA_RESET)
                begin
                BOARD_REV_0 <=(~BOARD_REV[0])&(~BOARD_REV[1])&(~BOARD_REV[2])&(~BOARD_REV[3]);
                MODE_REG <= 0;
                led_in_reg <=0;
                int_en_reg <= 16'b1111111111000000;
                int_pol_reg <= 16'b0000000000110000;
                FPGA_I_O_modereg <=0;
                FPGA_I_O_reg <= 0;
                EPIO_modereg <=0;
                FPIO_modereg <=0;
                FLASHACCreg <= 0;
                CSICONTreg <= 0;
                EPIO_reg <= 0;
                FPIO_reg <= 0;
                end
        else
                begin
                case ({BIOCS0_N, address_device[24:16], address_reg[6:2]})
                        {1'b0, `DEV_ADDR, `DEV_OFFS+ 1}: MODE_REG <= data;
                        {1'b0, `DEV_ADDR, `DEV_OFFS+ 2}: led_in_reg <= data;
                        {1'b0, `DEV_ADDR, `DEV_OFFS+ 4}: FLASHACCreg <= data;
                        {1'b0, `DEV_ADDR, `DEV_OFFS+ 5}: CSICONTreg <= data;
                        {1'b0, `DEV_ADDR, `DEV_OFFS+ 7}: int_en_reg <= data;
                        {1'b0, `DEV_ADDR, `DEV_OFFS+ 8}: int_pol_reg <= data;
                        {1'b0, `DEV_ADDR, `DEV_OFFS+24}: EPIO_modereg <= data;
                        {1'b0, `DEV_ADDR, `DEV_OFFS+25}: EPIO_reg <= data;
                        {1'b0, `DEV_ADDR, `DEV_OFFS+26}: FPIO_modereg[5:0] <= {2'b0,
                                data[3:0]};
                        {1'b0, `DEV_ADDR, `DEV_OFFS+27}:FPIO_reg[3:0] <= data[3:0];
                        {1'b0, `DEV_ADDR, `DEV_OFFS+28}: FPGA_I_O_modereg[15:0] <= data;
                        {1'b0, `DEV_ADDR, `DEV_OFFS+29}:  FPGA_I_O_reg[15:0] <= data;
                        {1'b0, `DEV_ADDR, `DEV_OFFS+30}:  FPGA_I_O_modereg[20:16] <= data[4:0];
                        {1'b0, `DEV_ADDR, `DEV_OFFS+31}: FPGA_I_O_reg[20:16] <= data[4:0];
                endcase
                end
        end
```

```
//***********************************************
//Read register from FPGA
//***********************************************

always @(negedge BRD_N)
       begin
       if (MPOWER)
              begin
              case ({BIOCS0_N, address_device[24:16], address_reg[6:2]})
                     {1'b0, `DEV_ADDR, `DEV_OFFS+ 0}: outreg = {`FPGA_REVISION, BOARD_REV};
                     {1'b0, `DEV_ADDR, `DEV_OFFS+ 1}: outreg = MODE_REG;
                     {1'b0, `DEV_ADDR, `DEV_OFFS+ 2}: outreg = led_in_reg;
                     {1'b0, `DEV_ADDR, `DEV_OFFS+ 3}: outreg = ~DIPSWITCH;
                     {1'b0, `DEV_ADDR, `DEV_OFFS+ 4}: outreg = {14'b0, FRDY, FLASHACCreg[0]};
                     {1'b0, `DEV_ADDR, `DEV_OFFS+ 5}: outreg = {12'b0, RQC1, RQC0,
                                                              CSICONTreg[1:0]};
                     {1'b0, `DEV_ADDR, `DEV_OFFS+ 6}: outreg = BOARD_REV_0?
                                                              {15'b0, ~INTD2_N}:
                            {INTA1_N_PE, INTB1_N_PE, INTC1_N_PE,
                            INTD1_N_PE, IORDY_INTA2_N_PE,
                            RDY_INTB2_N_PE,FPGA_IN_INTC2_N_PE,
                            INTD2_N_PE,INTE_N_PE,FPIO4_PE,
                            RQC0_PE,RQC1_PE,4'b0};
              {1'b0, `DEV_ADDR, `DEV_OFFS+ 7}:      outreg = int_en_reg[15:0];
              {1'b0, `DEV_ADDR, `DEV_OFFS+ 8}:      outreg = int_pol_reg[15:0];
              {1'b0, `DEV_ADDR, `DEV_OFFS+24}:      outreg = {EPIO_modereg[15],
               1'b0, EPIO_modereg[13:8],
              6'b0, EPIO_modereg[1], 1'b0};
              {1'b0, `DEV_ADDR, `DEV_OFFS+25}:      outreg = {EPIO15,1'b0, EPIO13, EPIO12,
              EPIO11, EPIO10, EPIO9, EPIO8,
              1'b0,   1'b0,   1'b0,   1'b0,
              1'b0,   1'b0, EPIO1,   1'b0};
              {1'b0, `DEV_ADDR, `DEV_OFFS+26}:     outreg = {12'b0, FPIO_modereg[3:0]};
              {1'b0, `DEV_ADDR, `DEV_OFFS+27}:     outreg = {10'b0,INTD2_N, FPIO[4:0]};
              {1'b0, `DEV_ADDR, `DEV_OFFS+28}:     outreg = FPGA_I_O_modereg[15:0];
              {1'b0, `DEV_ADDR, `DEV_OFFS+29}:     outreg = FPGA_I_O[15:0];
              {1'b0, `DEV_ADDR, `DEV_OFFS+30}:     outreg = {11'b0,FPGA_I_O_modereg[20:16]};
              {1'b0, `DEV_ADDR, `DEV_OFFS+31}:     outreg = {11'b0, FPGA_I_O[20:16]};
              default:                             outreg = 16'b 0000_0000_0000_0000;
              endcase
```

```
		case ({BIOCS0_N, address_device[24:16]})
			{1'b0, `DEV_ADDR}:		FPGA_READ_reg = 1;
			default:			FPGA_READ_reg = 0;
			endcase
		end
	end


//************************************************
//LED Control
//************************************************

always @(posedge clk10ms)
	begin
	if (!DIPSWITCH[8])			//Testmode
		begin
		case ({clk_countreg1[5:0]})
			6'd 0:		begin
				led_test_reg <= led_test_reg +1;
				case ({led_test_reg[3:0]})
					4'b 0000:	ledreg[7:0] <= `LEDPATTERN_0;
					4'b 0001:	ledreg[7:0] <= `LEDPATTERN_1;
					4'b 0010:	ledreg[7:0] <= `LEDPATTERN_2;
					4'b 0011:	ledreg[7:0] <= `LEDPATTERN_3;
					4'b 0100:	ledreg[7:0] <= `LEDPATTERN_4;
					4'b 0101:	ledreg[7:0] <= `LEDPATTERN_5;
					4'b 0110:	ledreg[7:0] <= `LEDPATTERN_6;
					4'b 0111:	ledreg[7:0] <= `LEDPATTERN_7;
					4'b 1000:	ledreg[7:0] <= `LEDPATTERN_8;
					4'b 1001:	ledreg[7:0] <= `LEDPATTERN_9;
					4'b 1010:	ledreg[7:0] <= `LEDPATTERN_A;
					4'b 1011:	ledreg[7:0] <= `LEDPATTERN_B;
					4'b 1100:	ledreg[7:0] <= `LEDPATTERN_C;
					4'b 1101:	ledreg[7:0] <= `LEDPATTERN_D;
					4'b 1110:	ledreg[7:0] <= `LEDPATTERN_E;
					4'b 1111:	ledreg[7:0] <= `LEDPATTERN_F;
				endcase
```

```
                        case ({led_test_reg[7:4]})
                               4'b 0000:      ledreg[15:8] <= `LEDPATTERN_0;
                               4'b 0001:      ledreg[15:8] <= `LEDPATTERN_1;
                               4'b 0010:      ledreg[15:8] <= `LEDPATTERN_2;
                               4'b 0011:      ledreg[15:8] <= `LEDPATTERN_3;
                               4'b 0100:      ledreg[15:8] <= `LEDPATTERN_4;
                               4'b 0101:      ledreg[15:8] <= `LEDPATTERN_5;
                               4'b 0110:      ledreg[15:8] <= `LEDPATTERN_6;
                               4'b 0111:      ledreg[15:8] <= `LEDPATTERN_7;
                               4'b 1000:      ledreg[15:8] <= `LEDPATTERN_8;
                               4'b 1001:      ledreg[15:8] <= `LEDPATTERN_9;
                               4'b 1010:      ledreg[15:8] <= `LEDPATTERN_A;
                               4'b 1011:      ledreg[15:8] <= `LEDPATTERN_B;
                               4'b 1100:      ledreg[15:8] <= `LEDPATTERN_C;
                               4'b 1101:      ledreg[15:8] <= `LEDPATTERN_D;
                               4'b 1110:      ledreg[15:8] <= `LEDPATTERN_E;
                               4'b 1111:      ledreg[15:8] <= `LEDPATTERN_F;
                        endcase
                        end
                default:;
        endcase
        end
else
begin
case ({MODE_REG[0],led_in_reg[3:0]})
                        5'b 0_0000:           ledreg[7:0] <= `LEDPATTERN_0;
                        5'b 0_0001:           ledreg[7:0] <= `LEDPATTERN_1;
                        5'b 0_0010:           ledreg[7:0] <= `LEDPATTERN_2;
                        5'b 0_0011:           ledreg[7:0] <= `LEDPATTERN_3;
                        5'b 0_0100:           ledreg[7:0] <= `LEDPATTERN_4;
                        5'b 0_0101:           ledreg[7:0] <= `LEDPATTERN_5;
                        5'b 0_0110:           ledreg[7:0] <= `LEDPATTERN_6;
                        5'b 0_0111:           ledreg[7:0] <= `LEDPATTERN_7;
                        5'b 0_1000:           ledreg[7:0] <= `LEDPATTERN_8;
                        5'b 0_1001:           ledreg[7:0] <= `LEDPATTERN_9;
                        5'b 0_1010:           ledreg[7:0] <= `LEDPATTERN_A;
                        5'b 0_1011:           ledreg[7:0] <= `LEDPATTERN_B;
                        5'b 0_1100:           ledreg[7:0] <= `LEDPATTERN_C;
                        5'b 0_1101:           ledreg[7:0] <= `LEDPATTERN_D;
                        5'b 0_1111:           ledreg[7:0] <= `LEDPATTERN_F;
endcase
```

```
        case ({MODE_REG[0],led_in_reg[7:4]})
                                5'b 0_0000:           ledreg[15:8] <= `LEDPATTERN_0;
                                5'b 0_0001:           ledreg[15:8] <= `LEDPATTERN_1;
                                5'b 0_0010:           ledreg[15:8] <= `LEDPATTERN_2;
                                5'b 0_0011:           ledreg[15:8] <= `LEDPATTERN_3;
                                5'b 0_0100:           ledreg[15:8] <= `LEDPATTERN_4;
                                5'b 0_0101:           ledreg[15:8] <= `LEDPATTERN_5;
                                5'b 0_0110:           ledreg[15:8] <= `LEDPATTERN_6;
                                5'b 0_0111:           ledreg[15:8] <= `LEDPATTERN_7;
                                5'b 0_1000:           ledreg[15:8] <= `LEDPATTERN_8;
                                5'b 0_1001:           ledreg[15:8] <= `LEDPATTERN_9;
                                5'b 0_1010:           ledreg[15:8] <= `LEDPATTERN_A;
                                5'b 0_1011:           ledreg[15:8] <= `LEDPATTERN_B;
                                5'b 0_1100:           ledreg[15:8] <= `LEDPATTERN_C;
                                5'b 0_1101:           ledreg[15:8] <= `LEDPATTERN_D;
                                5'b 0_1110:           ledreg[15:8] <= `LEDPATTERN_E;
                                5'b 0_1111:           ledreg[15:8] <= `LEDPATTERN_F;
        endcase
        case ({MODE_REG[0]})
                                     1'b 1:           ledreg <= ~led_in_reg;
        endcase
        end
        end

        assign LEDout = MPOWER? ledreg: 16'b 0;


//************************************************
//Flash access Control
//************************************************

        assign VPEN = FLASHACCreg[0];


//************************************************
//CSI control signals
//************************************************

        assign GNTC0 = CSICONTreg[0];
        assign GNTC1 = CSICONTreg[1];


//************************************************
//Ready Control
//************************************************

/*      always @(RDY_INTB2_N)
        if (BOARD_REV_0)
        begin
        IORDY_INTA2_N_reg = RDY_INTB2_N;
        end
*/
```

```
//************************************************
//Miscellaneous initialization
//************************************************

       assign HLDRQ_N = 1;

//************************************************
//Interrupt Control
//************************************************

       assign INTA1_N_P =                INTA1_N              ~^ int_pol_reg[15];
       assign INTB1_N_P =                INTB1_N              ~^ int_pol_reg[14];
       assign INTC1_N_P =                INTC1_N              ~^ int_pol_reg[13];
       assign INTD1_N_P =                INTD1_N              ~^ int_pol_reg[12];
       assign IORDY_INTA2_N_P =          IORDY_INTA2_N        ~^ int_pol_reg[11];
       assign RDY_INTB2_N_P =            RDY_INTB2_N          ~^ int_pol_reg[10];
       assign FPGA_IN_INTC2_N_P =        FPGA_IN_INTC2_N      ~^ int_pol_reg[9];
       assign INTD2_N_P =                INTD2_N              ~^ int_pol_reg[8];
       assign INTE_N_P =                 INTE_N               ~^ int_pol_reg[7];
       assign FPIO4_P =                  FPIO[4]              ~^ int_pol_reg[6];
       assign RQC0_P =                   RQC0                 ~^ int_pol_reg[5];
       assign RQC1_P =                   RQC1                 ~^ int_pol_reg[4];

       assign INTA1_N_PE =               INTA1_N_P            & int_en_reg[15];
       assign INTB1_N_PE =               INTB1_N_P            & int_en_reg[14];
       assign INTC1_N_PE =               INTC1_N_P            & int_en_reg[13];
       assign INTD1_N_PE =               INTD1_N_P            & int_en_reg[12];
       assign IORDY_INTA2_N_PE =         IORDY_INTA2_N_P      & int_en_reg[11];
       assign RDY_INTB2_N_PE =           RDY_INTB2_N_P        & int_en_reg[10];
       assign FPGA_IN_INTC2_N_PE =       FPGA_IN_INTC2_N_P    & int_en_reg[9];
       assign INTD2_N_PE =               INTD2_N_P            & int_en_reg[8];
       assign INTE_N_PE =                INTE_N_P             & int_en_reg[7];
       assign FPIO4_PE =                 FPIO4_P              & int_en_reg[6];
       assign RQC0_PE =                  RQC0_P               & int_en_reg[5];
       assign RQC1_PE =                  RQC1_P               & int_en_reg[4];

       assign GPIO2 = MPOWER? ~(INTA1_N_PE | FPIO4_PE |(IORDY_INTA2_N_PE & ~BOARD_REV_0)):1;
       assign GPIO3 = MPOWER? ~(INTB1_N_PE | (RDY_INTB2_N_PE & ~BOARD_REV_0)): 1;
       assign GPIO4 = MPOWER? ~(INTC1_N_PE | (FPGA_IN_INTC2_N_PE & ~BOARD_REV_0)): 1;
       assign GPIO5 = MPOWER? ~(INTD1_N_PE | (INTD2_N_PE & ~BOARD_REV_0)): 1;
       assign GPIO7 = MPOWER? ~(INTE_N_PE | RQC0_PE | RQC1_PE): 1;
```

```
//************************************************
//Clock generation
//************************************************

always @(posedge EPFIRCLK)
begin
       case ({clk_countreg_FIR[19:0]})
              20'd 240000:          begin
                                    clk_countreg_FIR[19:0] = 0;
                                    clk10msreg = clk10msreg ^ 1;
                                    end
              default:              clk_countreg_FIR[19:0] =clk_countreg_FIR[19:0]+1;
       endcase
end

always @(posedge clk10ms)
begin
       case ({clk_countreg1[5:0]})
              6'd 24:               clk_countreg1[5:0] = 0;
              default:              clk_countreg1[5:0] = clk_countreg1[5:0]+1;
       endcase
end

//************************************************
//Power On Control
//************************************************

always @(posedge RC_CLK)
begin
       case ({clk_countreg[5:0]})
              6'd 49:               begin
                                    clk_countreg[5:0] = 0;
                                    end
              default:              clk_countreg = clk_countreg+1;
       endcase
end

always @(posedge RC_CLK)
begin
   if ((clk_countreg[5:0] < 40) && (~pwron_rst_done))
   begin
        internal_reset <= 1;
   end
   else
   begin
        internal_reset <= 0;
        pwron_rst_done <= 1;
   end
end
```

```
always @(posedge RC_CLK)
begin
       if (FPGA_RESET)
              begin
              poweronseqreg <= 0;
              POWER_reg <= 0;
              BATTINT_reg <= 0;
              pwrsw_cntr <= 0;
              pwrsw <= ~POWERSWITCH;
              end
       else
              begin
              if (~POWERSWITCH!= pwrsw)
              begin
                 pwrsw_cntr <= pwrsw_cntr + 1;
                 if (pwrsw_cntr == 15) pwrsw <= ~POWERSWITCH;
              end
              else
              begin
                 pwrsw_cntr <= 0;
              end

              case ({poweronseqreg})
                     4'h 0: if (pwrsw)
                                           begin
                                           POWER_reg <= 1;
                                           BATTINT_reg <= 1;
                                           poweronseqreg <= 1;
                                           end
                     4'h 1: if (MPOWER)
                                           begin
                                           POWER_reg <= 0;
                                           poweronseqreg <= 2;
                                           end
                     4'h 2: if (!pwrsw)
                                           begin
                                           poweronseqreg <= 3;
                                           end
                      4'h 3: if (pwrsw)
                                           begin
                                           BATTINT_reg <= 0;

                                           poweronseqreg <= 4;
                                           end
                     4'h 4: if (!MPOWER)
                                           begin
                                           poweronseqreg <= 5;
                                           end
                     4'h 5: if (!pwrsw)
                                           begin
                                           poweronseqreg <= 0;
                                           end
```

```
                    default:       poweronseqreg <= 0;
                endcase
        end
end

assign POWER = POWER_reg;
assign BATTINT_N = BATTINT_reg;
assign SDRAMPOWER = (MPOWER &!MODE_REG[4]) | (SPOWER & MODE_REG[4]);
assign SDRAMPOWER_N =!SDRAMPOWER;
assign OSCOE = MPOWER;

endmodule
```

B.2 File PORT1.V

```
//************************************************
// DESIGNER:Willi Nuesser
// OBJECT      :VR4131 Board
// DATE        :31. Oct. 2002
// REVISION:2.0
//************************************************

module PORT1(DIN, DOUT, EN);
        input DIN;            //
        input EN;             //
        output DOUT;
        reg DOUT;

always@(EN or DIN)
        begin
        if(EN)
        DOUT=DIN;
        else
        DOUT=1'bz;
        End
endmodule
```

B.3 File PORT16.V

```
//***********************************************
// DESIGNER:Willi Nuesser
// OBJECT      :VR4131 Board
// DATE        :31. Oct. 2002
// REVISION:2.0
//***********************************************

module PORT16(DIN, DOUT, EN);
        input [15:0]DIN;              //
        input EN;              //
        output [15:0]DOUT;
        reg [15:0]DOUT;

always@(EN or DIN)
        begin
        if(EN)
        DOUT=DIN;
        else
        DOUT=16'bzzzzzzzzzzzzzzz;
        end
endmodule
```

B.4 File RCOSC.V

```
//************************************************
// DESIGNER    :Michael Kraemer
// OBJECT      :RC Oscillator
// DATE        :26. July 2002
// REVISION    :2.0
//************************************************

module rcosc(ena, x1, x2, x3);

   input ena, x1;
   output x2, x3;

   wire q;

   assign x2 = ena? q: 0;
   assign x3 = x1;

   not not1 (q, x1);

endmodule
```

B.5 File VR4131.DEF

```
//************************************************
// DESIGNER:Willi Nuesser
// OBJECT      :VR4131 Board
// DATE        :31. Oct. 2002
// REVISION:2.0
//************************************************

`define LEDPATTERN_0 8'b 1010_0000
`define LEDPATTERN_1 8'b 1111_1001
`define LEDPATTERN_2 8'b 1100_0100
`define LEDPATTERN_3 8'b 1101_0000
`define LEDPATTERN_4 8'b 1001_1001
`define LEDPATTERN_5 8'b 1001_0010
`define LEDPATTERN_6 8'b 1000_0010
`define LEDPATTERN_7 8'b 1111_1000
`define LEDPATTERN_8 8'b 1000_0000
`define LEDPATTERN_9 8'b 1001_0000
`define LEDPATTERN_A 8'b 1000_1000
`define LEDPATTERN_B 8'b 1000_0011
`define LEDPATTERN_C 8'b 1010_0110
`define LEDPATTERN_D 8'b 1100_0001
`define LEDPATTERN_E 8'b 1000_0110
`define LEDPATTERN_F 8'b 1000_1110

`define FPGA_REVISION 12'b 0000_0010_0000

`define DEV_ADDR 9'b 0000_0000_0
`define DEV_OFFS 5'b 0000_0
```

NEC

Facsimile Message

From:

Name

Company

Tel. FAX

Address

Although NEC has taken all possible steps to ensure that the documentation supplied to our customers is complete, bug free and up-to-date, we readily accept that errors may occur. Despite all the care and precautions we've taken, you may encounter problems in the documentation. Please complete this form whenever you'd like to report errors or suggest improvements to us.

Thank you for your kind support.

North America
NEC Electronics Inc.
Corporate Communications Dept.
Fax: +1-800-729-9288
+1-408-588-6130

Europe
NEC Electronics (Europe) GmbH
Market Communication Dept.
Fax: +49-211-6503-274

South America
NEC do Brasil S.A.
Fax: +55-11-6462-6829

Hong Kong, Philippines, Oceania
NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

Korea
NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: +82-2-528-4411

Taiwan
NEC Electronics Taiwan Ltd.
Fax: +886-2-2719-5951

Asian Nations except Philippines
NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

Japan
NEC Semiconductor Technical Hotline
Fax: +81- 44-435-9608

I would like to report the following error/make the following suggestion:

Document title: _____

Document number: _____ Page number: _____

If possible, please fax the referenced page or drawing.

Document Rating	Excellent	Good	Acceptable	Poor
Clarity	☐	☐	☐	☐
Technical Accuracy	☐	☐	☐	☐
Organization	☐	☐	☐	☐