Linux Development Kit For The Nios Embedded Processor

Application Notes VERSION 1.0

Issued: August 3, 2001

LDK – Hardware Components

- Quartus Project
- Hardware Add-on Boards

Introduction

This document is intended to familiarize the reader with the different components that make up the FPGA core included with the Linux Development Kit (LDK). The first section of this document explains the functionality of the three hardware boards included with the kit. The second section explains the steps for installing the Quartus project onto a PC and other procedures that must be followed to be able to rebuild the project. The next section explains the project itself. This includes a description of the memory map, a description of the third party components included with the project, and a description of the circuit surrounding the Nios core within this project. The final component of this section will outline the steps that must be taken to update the custom GERMS monitor included with the project if changes are made to the memory map.

Readers will also notice an appendix included with this document, Appendix A. This Appendix provides the reader with the pin-out of the current project. This should help any reader who is building his or her own custom board for the Excalibur kit and needs to see the current pin-out of the project.

Since the project can be placed anywhere on the user's hard drive, this document, when referring to directory locations, will reference the location of the project as *project_directory*. All subdirectories underneath this main directory will also be referenced in italics. This document also assumes the reader has installed their Nios HDK in the default location *c:\Altera\Excalibur\…*.

Linux Development Kit Add-on Boards

Memory Expansion Board

The first of three boards included with the Linux Development Kit is the Memory Expansion Board. This board is to be connected into the SODIMM socket J2. The board comes with flash memory and SDRAM. The flash memory is composed of two AM29LC323DB (32 Mb) chips manufactured by AMD. The SDRAM is composed of two MT48LC2M32B2 (64 Mb) chips manufactured by Micron.

OS Support Board

The next board included with the Linux Development Kit is the Operating System (OS) Support Board. This board consists of a Real Time Clock and Temperature Sensor on an SPI bus, as well as, a header for an IDE interface and a CompactFlash interface. The Real Time Clock is Dallas's DS1306 and the Temperature Sensor is Dallas's DS1722. The CompactFlash header is configured into True IDE Mode and connected to the IDE header. The IDE interface is not supported by the uCLinux Kernel being shipped with the LDK. This board mounts on the 5 volt prototype headers JP11, JP12 and JP13.

Ethernet Board

The final board to ship with the LDK is the Ethernet Connectivity Board. This 10-BaseT board consists of Cirrus's CS8900 ISA Ethernet Controller and a RJ-45 connector. This board mounts on the 3.3 volt prototype headers JP8, JP9 and JP10.

Installing the Quartus Project

On the Linux Development Kit Software Support CD there is a directory called "Quartus Project", which contains three component files.

1. Fsm.pm
2. Nios111-patch-build6.exe
3. hardwarev2_1.zip

The following three sections explain the contents of the files and the procedures for installing them on a PC.

Fsm.pm

This file comes with the SDRAM controller component. It is to be copied into the directory *c:\Altera\Excalibur\sopc_builder\bin*. Failure to do so will result in errors while trying to generate new cores with the SDRAM control included.

Nios111-patch-build6.exe

This patch file allows components to link to the ifetch signal coming from the Nios. Before this patch became available controls were not allowed to connect to the ifetch signal. To install the patch, copy it into a temporary directory and run it. This patch is needed for the SDRAM control since it is using the ifetch pin in some of its logic.

hardwarev2_1.zip

This compressed file contains the full Quartus project included with the Linux Development Kit. It contains the definition of the Nios core, the SDRAM controller (located in the sub directory *project_directory\altera_avalon_sdram_controller*), the Ethernet control (located in the sub directory *project_directory\altera_avalon_CS8900*), and the definition for the SODIMM flash controller (located in the sub directory *project_directory\mtx_sodimm_flash_controller*).

All of these components are considered to be local components to the current project. However, if the user wishes to make them into global components, which could be then used in other projects, each directory should be moved to the directory c:*\Altera\Excalibur\sopc_builder\components*.

Besides the components for the Nios core, this project also includes an s-record for the custom germs monitor. The file is located in the sub directory *project_directory\cpu32_sdk\Custom_Germs*. If the user wishes to modifying the GERMS monitor software then the reference for the on-board ROM will need to be updated.

Quartus Project

The current project was designed under version 1.0 of the Quartus II development software included with the Excalibur kit. The Nios core is version 1.1.1. This section assumes you have installed the Quartus project onto your PC already.

Memory Map

Table 1 below outlines the memory map of the current project. The labels used to reference each component should be preserved to link up with the software developed for this project. The memory maps can change, but the software will have to be recompiled if this is done.

Table 1: Nios Memory Map for LDK

Name	Label	Starting Address	End Address	Size (bytes)
Mon	na_mon	0x0000 0000	0x0000 03FF	1 Kbytes
uart0	na_uart	0x0000 0400	0x0000 041F	32 bytes
timer0	na_timer0	0x0000 0420	0x0000 043F	32 bytes
led_pio	na_led_pio	0x0000 0460	0x0000 046F	16 bytes
button_pio	na_button_pio	0x0000 0470	0x0000 047F	16 bytes
Spi	na_spi	0x0000 0480	0x0000 049F	32 bytes
uart1	na_uart1	0x0000 04A0	0x0000 04BF	32 bytes
ide_interface	na_ide_interface	0x0000 0500	0x0000 057F	128 bytes
ide_ctl_in	na_ide_ctl_in	0x0000 0580	0x0000 058F	16 bytes
Enet	na_enet	0x0000 4000	0x0000 001F	32 bytes
Enet_Reset	na_enet_reset	0x0000 4020	0x0000 402F	16 bytes
Sram	na_sram	0x0004 0000	0x0007 FFFF	256 Kbytes
Flash	na_flash	0x0010 0000	0x001F FFFF	1 Mbyte
flash_kernel	na_flash_kernel	0x0080 0000	0x00FF FFFF	8 Mbytes
Sdram	na_sdram	0x0100 0000	0x01FF FFFF	16 Mbytes

The next table, **Error! Reference source not found.**, shows the memory map of labels created and used by the Altera System Builder Wizard, and the compiler/linker tools. These links were created automatically based of selections in the Nios System Builder Wizard.

Table 2: System Variable Map

Name	Label	Starting Address	End Address	Size (bytes)
Printf Uart	nasys_printf_uart	0x0000 0400	0x0000 041F	32 bytes
GDB Uart	nasys_gdb_uart	0x0000 04A0	0x0000 04BF	32 bytes
Main Flash	nasys_main_flash	0x0010 0000	0x001F FFFF	1 Mbyte
Program Memory	nasys_program_mem	0x0100 0000	0x01FF FFFF	16 Mbytes
Data Memory	nasys_data_mem	0x0100 0000	0x01FF FFFF	16 Mbytes
Stack Top	nasys_stack_top	0x0200 0000		
Vector Table	nasys_vector_table	0x0004 0000	0x0004 0100	

The next table, **Error! Reference source not found.**, shows the external interrupts that have been created and their corresponding number.

Table 3: System Interrupt Assignment

Name	Label	IRQ Number
Uart0	na_uart0_irq	17
Timer0	na_timer0_irq	16
Uart1	na_uart1_irq	18
Enet	na_enet_irq	35
SPI	na_spi_irq	19
Printf Uart	nasys_printf_uart_irq	17
GDB Uart	nasys_gdb_uart_irq	18

Components

On-Chip ROM (MON)

This component is defined as read-only memory located at address 0x0000 and contained within the Nios processor. The memory stores a custom germs monitor, which not only allows the user to erase and program the on-board flash memory, but to also erase and program the memory on the Memory Expansion Card. This file has been placed in the directory *project_directory\cpu32_sdk\Custom_Germs*. If you decide to change the location of either flash devices, or rename either of the devices. Please see the section, Changing the GERMS, for information about changing the germs program. Otherwise, this program will be included in the core every time you generate a new one.

In the Nios System Builder the monitor ROM has been configured as the boot device. Therefore, every time the processor is powered on or reset its program will be started.

UARTs and Timers (UART0, UART1, TIMER0)

Two UART components and one timer have been included with the Quartus project. See **Error! Reference source not found.** for more information on the interrupt number associated with each device. The first UART, UART0, is setup through the Nios System Builder Wizard to function as the Host Communication port. This means germs monitor and other system libraries, such as printf, will use this port by default to output characters. The second UART, UART1, is setup as the Debugging Communication port. The target GDB stub uses this port to communicate with the host-debugging system. The timer device is used by the uCLinux kernel and must be present for the system to function.

On-Board Memory (SRAM, FLASH)

These two components provide an interface with the SRAM and flash memories located on the board. The SRAM on the Excalibur board is composed of two IDT71V016SA12 chips. For more information on these chips please refer to [1] and [2]. Each chip is 64 K by 16-bits, and the two chips are placed in parallel to create a 32-bit bus. The class.ptf file for this component uses zero wait states for both a read and a write operation, and has a hold time of half a clock cycle. The first 256 words of memory has been reserved to hold the Interrupt Vector Table; this was setup up in the Nios System Builder.

The on-board flash memory is AMD's 29LV800B; please refer to [3] for more information. This memory has been configured as a 512 K x 16-bit memory. The first 512 Kbytes is free for storing user programs and nonvolatile data. The next 256 bytes are allocated to store a user-defined core. This is where the core shipped with the LDK is to be stored.

The final 256 bytes store a factory default core. For more information on the on-board flash setup and any other aspects of the Excalibur board please refer to [4].

On-Board I/O Peripherals (LED_PIO, SEVEN_SEG_PIO, BUTTON_PIO)

These components have been included with the LDK's Quartus project to allow an interface with the various button and LED components included with the Excalibur board. The SEVEN_SEG_PIO component interfaces with the seven-segment display, the LED_PIO interfaces with LED1 and LED2, and the BUTTON_PIO interfaces with the four push button switches and the 14-pin DIP switch. For the LED_PIO component, bit 0 is connected to LED1 and bit 1 is connected to LED2. For the BUTTON_PIO component, the push buttons labeled SW4-7 are connected to bits 0-3, and the 8-pin DIP switch is connected to bits 4-11.

SPI Bus (SPI)

The SPI component has been placed in the project to interface with the Real Time Clock and the Temperature Sensor included with the OS Support Board. The SPI component has been configured to transmit and receive 16-bits, the first eight being a register address for the SPI device, and the last eight being the data byte to be read/written. The Real Time Clock has been configured as device 0 on the bus, and the Temperature Sensor has been configured as device 1. For information on the internal registers for the Real Time Clock and the Temperature Sensor please refer to their respective data sheets, [5] and [6]. The following table, shows the different options selected in the SPI Wizard.

Table 4: SPI Options

SPI Wizard Option	Selection
Nios is Master or Slave?	Master
SPI shift register width	16
MSB/LSB first?	MSB
SPI clock rate	250.0 KHz
Number of slave devices	2
SS_n delay	5.0 us
Polarity of SPI clock	low (SCK is low when idle)
SPI clock phase	1 (data sampled on falling edge)

The final thing to note about the SPI component of this project is the polarity of the slave select signals. The default polarity of the slave select signals is active low, and there are no options to allow changing this. Unfortunately the chip select signals for both devices are active high, thus inverters were inserted into the Quartus project to reverse the polarity of the slave select signals coming out of the Nios core.

Ethernet Board (ENET, ENET_RESET)

The Ethernet control placed in the Quartus project is Altera's altera_avalon_CS8900 ethernet control version 1.0. This control is currently referenced as a local control, but moving it to the directory *c:\altera\Excalibur\sopc_builder\components* will result in a global control. This control provides an interface with the ethernet chip CS8900. The interface only supports the cs8900's I/O mode. The base address is hardwired to 0x300,and the memory map of the control directly interfaces with the eight registers of the chip. For more information on the control please contact the Altera Corporation, and for more information on the ethernet chip, CS8900, please refer to [7].

The second component for the Ethernet interface is a general input-output pin called enet_reset. Writing a one to this location will place the CS8900 into reset mode, and writing a zero to this location will activate the chip.

Memory Expansion Board - SDRAM (SDRAM)

The Memory Expansion Board included with the LDK contains SDRAM and Flash memory. The SDRAM used on the board is two MT48LC2M32B2 chips manufactured by Micron[8]. These two 32-bit chips are placed in series to create an overall memory space of 4 M x 32-bits. The Nios SDRAM Controller for SODIMM SDRAM, included with this kit, controls these SDRAM chips. Part of the functionality of this control is to address decode the single address space of the SDRAM component into separate chip selects for the two devices.

Wizard options in the class.ptf file for this control have been modified to meet the specifications of the SDRAM on the Memory Expansion Board. These options have been provided below in **Table 5**. If the user wishes to use the control with another SDRAM device the options in the class.ptf file should be examined and modified to meet the different requirements of the new SDRAM. The control is currently placed as a local component only visible by the LDK Quartus project. If a user wishes to use the control with multiple projects, the directory *altera_avalon_sdram_controller* should be moved to the components directory *c:\altera\Excalibur\sopc_builder\components\* to allow for global access of the control.

Table 5: SDRAM Options

Wizard Option	Value
sdram_data_width	32
sdram_addr_width	11
sdram_bank_width	2
sdram_row_width	11
sdram_col_width	8
sdram_num_chipselects	2
refresh_period	15.625 us
powerup_delay	100 us
cas_latency	1
precharge_control_bit	10
Auto-regresh period t_rfc	70 ns
PRECHARGE command period t_rp	20 ns
LOAD MODE REGISTER command to ACTIVE or REFRESH command t_mrd	2 clocks
ACTIVE to READ or WRITE delay t_rcd	20 ns
Access time from clock edge	17 ns
wr_auto_precharge	1 clock + 7 ns
t_wr_precharge	14 ns
init_refresh_commands	2
init_nop_delay	0
shared_data	1
enable_ifetch	1
highperf	1

Memory Expansion Board – Flash Memory (FLASH_KERNEL)

The flash memory on the Memory Expansion Board is composed of two 29LV323DB devices. These devices are manufactured by AMD. Please consult the data sheet [9] for more information of the operation of each device. The flash memory has been configured as a 2M x 16-bit device. For the LDK Quartus project the two device have been placed one after the other to create an overall memory space of 4M x 16-bit.

Included with the Quartus project is a file defining a custom class to interface the Nios with these chips. The class.ptf file is located in the *mtx_sodimm_flash_controller* directory. This class defines the memory as having a 16-bit bi-directional data bus, needing 4 wait states on a read or write, requiring a hold time of half a clock cycle, and having an address alignment of dynamic. This last option means each 32-bit option performed to the flash memory will be translated by the Avalon bus into two separate 16-bit operations.

Since the flash memory appears to the Nios as one signal memory unit, address decoding logic has been added to the Quartus project to create separate chip selects for the two devices. If a user wishes to create two separate flash memory spaces user defined interfaces could be placed into the Nios core but the user would have to remember to specify the correct number of wait states and the correct hold time. On top of this, the read select, the write select, the data bus and the address bus would have to be shared between both devices.

IDE/CompactFlash Interface (IDE_INTERFACE, IDE_CTL_IN)

An interface between the Nios core and the IDE header on the OS Support Board has been included with the Quartus project shipping in the LDK. This interface is based entirely on the reference project included with the Excalibur kit. The first component of this project is the IDE_INTERFACE. This component is a user-defined interface set to have 5 address pins and 16 data pins. As well the device is configured as memory mapped registers, with 2 clock cycles of setup time, 2 clock cycles of hold time, and 7 wait states. Some decoding logic has been included with the project to create a read and write signal from the outgoing Nios signals.

The second component for the IDE is 5 parallel input pins called IDE_CTL_IN. These inputs have their interrupts disabled and are connected as specified in **Table 6**. For more information on the IDE interface please refer to the test project included with the Excalibur kit. The test software included with this project will run with the LDK Quartus project.

Table 6: IDE Input Signals

Pin Number	Signal
ide_ctl_in[0]	ground
ide_ctl_in[1]	ide_intrq
ide_ctl_in[2]	ide_iordy
ide_ctl_in[3]	ide_iocs16_n (passes through an inverter)
ide_ctl_in[4]	ide_dasp_n (passes through an inverter)

It should be noted that the Linux Kernel shipped with the LDK does not support the IDE interface. It has been included to allow users who may wish to develop their own interface drivers to do so without having to modify the Quartus project.

The CompactFlash interface included with the OS Support Board has been hard wired to function in True IDE mode only. All of the signals going to the IDE header go to the corresponding location on the CompactFlash connector. As well, the software routines included with the Excalibur kit to interface with an IDE device will also work with a CompactFlash device.

PLL

The other component used in the LDK Quartus project is a Phase Locked Loop control. The Excalibur board comes with a 33 MHz oscillator. This clock signal is buffered into four separate clock signals one of which is tied into the CPLD. The Excalibur board also has the functionality to take an outputted clock signal from the CPLD, buffer it, and deliver it to individual devices on the board.

For the Linux Development Kit, the SDRAM on the Memory Expansion Card uses one of the buffered clock signals from the outputted CPLD clock signal. The PLL component placed in the design is used to generate a 33 MHz clock signal with zero delay from the incoming clock signal. For more information on the phase locked loop components please refer to the datasheet for an APEX 20K device[10].

Changing the GERMS

The LDK core included with the kit is running a custom version of the GERMS monitor. It has been modified to allow erasing and programming of the flash memory included with the Memory Expansion Card. If a user wishes to change the location of the flash memory, or rename its reference the custom GERMS code will have to be recompiled and re-included with the core.

The first step in this process is to regenerate the Nios core. This action will update the nios_map.h header file. The next step is to re-build the custom germs monitor. This code has been included in the directory *project_directory/cpu32_sdk/Custom_Germs*. In this directory there exists a makefile, the file nios_germs_monitor.s, and the file nios_germs_monitor.srec. The makefile is used to re-build the nios_germs_monitor.s file. The s-record file is the result of this compile and is programmed into the on-board memory.

Once the core has been re-generated the next step is to run the makefile, thus compiling a new GERMS monitor. Finally the resultant s-record should be re-included with the on-board ROM and the core re-generated. At this point the new germs monitor has been updated and placed into the Nios core.

References

1. IDT71V016SA data sheet. Integrated Device Technology, Inc. August 30, 2000. 71V0016SA_DS_61290.pdf. Available at http://www.idt.com/products/pages/Asynchronous_SRAMs-71V016SA.html.
2. Nios Development Board Schematic. Altera Corporation. February 6, 2001. Nios Dev Board.DSN.
3. Am29LV800B data sheet. Advanced Micro Devices. August 14, 2000. 21490.pdf. Available at http://www.amd.com/products/nvd/techdocs/techdocs.html#3volt.
4. Nios Development Board Guide. Altera Corporation. March 14, 2001. Nios_Development_Board_Guide.pdf. Included with Excalibur Kit from Altera.
5. DS1306 data sheet, Serial Alarm Real Time Clock. Dallas Semiconductor/Maxim. January 11, 2001. Available at http://dbserv.maxim-ic.com/quick_view2.cfm?qv_pk=2687.
6. DS1722 data sheet, Digital Thermometer with SPI/3-wire Interface. Dallas Semiconductor/Maxim. February 15 2000. Available at http://dbserv.maxim-ic.com/quick_view2.cfm?qv_pk=2766
7. CS8900A data sheet, Crystal LAN ISA Ethernet Controller. Cirrus Logic. April 2001. Avaliable at http://www.cirrus.com/design/products/overview/index.cfm?ProductID=46.
8. MT48LC2M32B2 data sheet. Micron Technology, Inc. September 2000. Available at http://www.micron.com/products/datasheet.jsp?path=/DRAM/SDRAM&fileID=15.
9. Am29LV322D/323D/324D data sheet. Advanced Micro Devices. 21534.pdf May 8, 2001. Available at http://www.amd.com/products/nvd/techdocs/techdocs.html#3volt
10. APEX 20K data sheet. Altera Corporation. May 2001. Available at http://www.altera.com/literature/ds/apex.pdf.

Appendix A Pin-out of CPLD

This list contains the pin-out generate by Quartus for the LDK project. If a new project is created this list can be copied into the new directory to preserve the pin-out.

```
MISO : LOCATION = Pin_W20;
MOSI : LOCATION = Pin_N15;
apex_reload_n : LOCATION = Pin_C19;
clk : LOCATION = Pin_L6;
clk_out : LOCATION = Pin_P5;
exp33v_enet_a[0] : LOCATION = Pin_R5;
exp33v_enet_a[10] : LOCATION = Pin_P20;
exp33v_enet_a[11] : LOCATION = Pin_K15;
exp33v_enet_a[1] : LOCATION = Pin_N20;
exp33v_enet_a[2] : LOCATION = Pin_K20;
exp33v_enet_a[3] : LOCATION = Pin_K22;
exp33v_enet_a[4] : LOCATION = Pin_K19;
exp33v_enet_a[5] : LOCATION = Pin_P22;
exp33v_enet_a[6] : LOCATION = Pin_N22;
exp33v_enet_a[7] : LOCATION = Pin_R22;
exp33v_enet_a[8] : LOCATION = Pin_P21;
exp33v_enet_a[9] : LOCATION = Pin_K16;
exp33v_enet_d[0] : LOCATION = Pin_L20;
exp33v_enet_d[10] : LOCATION = Pin_K5;
exp33v_enet_d[11] : LOCATION = Pin_R4;
exp33v_enet_d[12] : LOCATION = Pin_J7;
exp33v_enet_d[13] : LOCATION = Pin_J5;
exp33v_enet_d[14] : LOCATION = Pin_K3;
exp33v_enet_d[15] : LOCATION = Pin_N2;
exp33v_enet_d[1] : LOCATION = Pin_J18;
exp33v_enet_d[2] : LOCATION = Pin_K18;
exp33v_enet_d[3] : LOCATION = Pin_M17;
exp33v_enet_d[4] : LOCATION = Pin_N18;
exp33v_enet_d[5] : LOCATION = Pin_M20;
exp33v_enet_d[6] : LOCATION = Pin_L16;
exp33v_enet_d[7] : LOCATION = Pin_R21;
exp33v_enet_d[8] : LOCATION = Pin_N6;
exp33v_enet_d[9] : LOCATION = Pin_J3;
ext_addr[0] : LOCATION = Pin_G17;
ext_addr[10] : LOCATION = Pin_A3;
ext_addr[11] : LOCATION = Pin_A4;
ext_addr[12] : LOCATION = Pin_C3;
ext_addr[13] : LOCATION = Pin_C1;
ext_addr[14] : LOCATION = Pin_D3;
ext_addr[15] : LOCATION = Pin_D2;
ext_addr[16] : LOCATION = Pin_C2;
ext_addr[17] : LOCATION = Pin_F3;
ext_addr[18] : LOCATION = Pin_B3;
ext_addr[19] : LOCATION = Pin_E3;
ext_addr[1] : LOCATION = Pin_A8;
ext_addr[2] : LOCATION = Pin_B8;
ext_addr[3] : LOCATION = Pin_A7;
ext_addr[4] : LOCATION = Pin_B7;
ext_addr[5] : LOCATION = Pin_B6;
ext_addr[6] : LOCATION = Pin_A6;
```

```
ext_addr[7] : LOCATION = Pin_A5;
ext_addr[8] : LOCATION = Pin_B5;
ext_addr[9] : LOCATION = Pin_B4;
ext_be_n[0] : LOCATION = Pin_F5;
ext_be_n[1] : LOCATION = Pin_F2;
ext_be_n[2] : LOCATION = Pin_F4;
ext_be_n[3] : LOCATION = Pin_H5;
ext_data[0] : LOCATION = Pin_C4;
ext_data[10] : LOCATION = Pin_C5;
ext_data[11] : LOCATION = Pin_D6;
ext_data[12] : LOCATION = Pin_C6;
ext_data[13] : LOCATION = Pin_F9;
ext_data[14] : LOCATION = Pin_H10;
ext_data[15] : LOCATION = Pin_D7;
ext_data[16] : LOCATION = Pin_C7;
ext_data[17] : LOCATION = Pin_E9;
ext_data[18] : LOCATION = Pin_E10;
ext_data[19] : LOCATION = Pin_D9;
ext_data[1] : LOCATION = Pin_H11;
ext_data[20] : LOCATION = Pin_C8;
ext_data[21] : LOCATION = Pin_F10;
ext_data[22] : LOCATION = Pin_G11;
ext_data[23] : LOCATION = Pin_C9;
ext_data[24] : LOCATION = Pin_C10;
ext_data[25] : LOCATION = Pin_H12;
ext_data[26] : LOCATION = Pin_D10;
ext_data[27] : LOCATION = Pin_G12;
ext_data[28] : LOCATION = Pin_G13;
ext_data[29] : LOCATION = Pin_F11;
ext_data[2] : LOCATION = Pin_G10;
ext_data[30] : LOCATION = Pin_B11;
ext_data[31] : LOCATION = Pin_B10;
ext_data[3] : LOCATION = Pin_D8;
ext_data[4] : LOCATION = Pin_E7;
ext_data[5] : LOCATION = Pin_D4;
ext_data[6] : LOCATION = Pin_D5;
ext_data[7] : LOCATION = Pin_G9;
ext_data[8] : LOCATION = Pin_F8;
ext_data[9] : LOCATION = Pin_E8;
ext_flash_cs_n : LOCATION = Pin_E1;
ext_flash_we_n : LOCATION = Pin_H13;
ext_oe_n : LOCATION = Pin_A2;
ext_sram_addr17 : LOCATION = Pin_D1;
ext_we_n : LOCATION = Pin_E6;
flash_byte_n : LOCATION = Pin_V5;
ide_addr[0] : LOCATION = Pin_T1;
ide_addr[1] : LOCATION = Pin_U3;
ide_addr[2] : LOCATION = Pin_R1;
ide_cs0_n : LOCATION = Pin_R2;
ide_cs1_n : LOCATION = Pin_U4;
ide_csel : LOCATION = Pin_M15;
ide_dasp_n : LOCATION = Pin_P2;
ide_data[0] : LOCATION = Pin_R3;
ide_data[10] : LOCATION = Pin_N1;
ide_data[11] : LOCATION = Pin_M2;
ide_data[12] : LOCATION = Pin_T22;
ide_data[13] : LOCATION = Pin_T20;
```

```
ide_data[14] : LOCATION = Pin_T21;
ide_data[15] : LOCATION = Pin_P1;
ide_data[1] : LOCATION = Pin_N17;
ide_data[2] : LOCATION = Pin_L14;
ide_data[3] : LOCATION = Pin_K1;
ide_data[4] : LOCATION = Pin_U5;
ide_data[5] : LOCATION = Pin_M3;
ide_data[6] : LOCATION = Pin_N19;
ide_data[7] : LOCATION = Pin_L15;
ide_data[8] : LOCATION = Pin_P19;
ide_data[9] : LOCATION = Pin_R20;
ide_intrq : LOCATION = Pin_U22;
ide_iocs16_n : LOCATION = Pin_P18;
ide_iordy : LOCATION = Pin_R19;
ide_rd_n : LOCATION = Pin_K2;
ide_wr_n : LOCATION = Pin_N3;
j1_j9_p11_irq_l : LOCATION = Pin_J1;
j2_j9_p12_irq_u : LOCATION = Pin_J2;
k4_j8_p6_nmemw : LOCATION = Pin_K4;
l17_j9_p8_ncs_l : LOCATION = Pin_L17;
l7_j8_p15_niow_l : LOCATION = Pin_L7;
led[0] : LOCATION = Pin_T18;
led[1] : LOCATION = Pin_T19;
m16_j9_p14_reset : LOCATION = Pin_M16;
m6_j9_p13_nsbhe : LOCATION = Pin_M6;
n21_j8_p16_ncs_u : LOCATION = Pin_N21;
n5_j8_p14_nior_l : LOCATION = Pin_N5;
p4_j8_p5_nmemr : LOCATION = Pin_P4;
pb[0] : LOCATION = Pin_Y9;
pb[1] : LOCATION = Pin_T9;
pb[2] : LOCATION = Pin_Y8;
pb[3] : LOCATION = Pin_W9;
proto5_sel_n : LOCATION = Pin_V7;
reset_n : LOCATION = Pin_F12;
rtc_ce : LOCATION = Pin_U2;
rxd : LOCATION = Pin_W8;
rxd1 : LOCATION = Pin_F14;
sclk : LOCATION = Pin_P17;
seven_seg[0] : LOCATION = Pin_W17;
seven_seg[10] : LOCATION = Pin_Y17;
seven_seg[11] : LOCATION = Pin_V8;
seven_seg[12] : LOCATION = Pin_Y7;
seven_seg[13] : LOCATION = Pin_U11;
seven_seg[14] : LOCATION = Pin_R11;
seven_seg[15] : LOCATION = Pin_D18;
seven_seg[1] : LOCATION = Pin_U18;
seven_seg[2] : LOCATION = Pin_Y18;
seven_seg[3] : LOCATION = Pin_W18;
seven_seg[4] : LOCATION = Pin_U8;
seven_seg[5] : LOCATION = Pin_T11;
seven_seg[6] : LOCATION = Pin_R10;
seven_seg[7] : LOCATION = Pin_C18;
seven_seg[8] : LOCATION = Pin_V17;
seven_seg[9] : LOCATION = Pin_V18;
sodimm_flash_a[10] : LOCATION = Pin_AB15;
sodimm_flash_a[11] : LOCATION = Pin_AB16;
sodimm_flash_a[12] : LOCATION = Pin_AA13;
```

```
sodimm_flash_a[13] : LOCATION = Pin_Y5;
sodimm_flash_a[14] : LOCATION = Pin_Y6;
sodimm_flash_a[15] : LOCATION = Pin_T6;
sodimm_flash_a[16] : LOCATION = Pin_P7;
sodimm_flash_a[17] : LOCATION = Pin_V13;
sodimm_flash_a[18] : LOCATION = Pin_H16;
sodimm_flash_a[19] : LOCATION = Pin_F16;
sodimm_flash_a[1] : LOCATION = Pin_AA14;
sodimm_flash_a[20] : LOCATION = Pin_D17;
sodimm_flash_a[21] : LOCATION = Pin_C17;
sodimm_flash_a[2] : LOCATION = Pin_AA15;
sodimm_flash_a[3] : LOCATION = Pin_AB18;
sodimm_flash_a[4] : LOCATION = Pin_AA16;
sodimm_flash_a[5] : LOCATION = Pin_AA9;
sodimm_flash_a[6] : LOCATION = Pin_AB8;
sodimm_flash_a[7] : LOCATION = Pin_AA10;
sodimm_flash_a[8] : LOCATION = Pin_AA11;
sodimm_flash_a[9] : LOCATION = Pin_AA12;
sodimm_flash_ce0_n : LOCATION = Pin_W14;
sodimm_flash_ce1_n : LOCATION = Pin_P12;
sodimm_flash_dq[0] : LOCATION = Pin_AB5;
sodimm_flash_dq[10] : LOCATION = Pin_AA8;
sodimm_flash_dq[11] : LOCATION = Pin_AB7;
sodimm_flash_dq[12] : LOCATION = Pin_AA7;
sodimm_flash_dq[13] : LOCATION = Pin_AB6;
sodimm_flash_dq[14] : LOCATION = Pin_AA6;
sodimm_flash_dq[15] : LOCATION = Pin_AB17;
sodimm_flash_dq[1] : LOCATION = Pin_AA5;
sodimm_flash_dq[2] : LOCATION = Pin_AA4;
sodimm_flash_dq[3] : LOCATION = Pin_AB4;
sodimm_flash_dq[4] : LOCATION = Pin_AB3;
sodimm_flash_dq[5] : LOCATION = Pin_AB19;
sodimm_flash_dq[6] : LOCATION = Pin_AB20;
sodimm_flash_dq[7] : LOCATION = Pin_AA17;
sodimm_flash_dq[8] : LOCATION = Pin_AA18;
sodimm_flash_dq[9] : LOCATION = Pin_AB21;
sodimm_flash_oe_n : LOCATION = Pin_W7;
sodimm_flash_reset_n : LOCATION = Pin_Y13;
sodimm_flash_we_n : LOCATION = Pin_W6;
sodimm_sdram_a[0] : LOCATION = Pin_U16;
sodimm_sdram_a[10] : LOCATION = Pin_U14;
sodimm_sdram_a[1] : LOCATION = Pin_V16;
sodimm_sdram_a[2] : LOCATION = Pin_U15;
sodimm_sdram_a[3] : LOCATION = Pin_W16;
sodimm_sdram_a[4] : LOCATION = Pin_V15;
sodimm_sdram_a[5] : LOCATION = Pin_Y16;
sodimm_sdram_a[6] : LOCATION = Pin_W15;
sodimm_sdram_a[7] : LOCATION = Pin_T14;
sodimm_sdram_a[8] : LOCATION = Pin_Y15;
sodimm_sdram_a[9] : LOCATION = Pin_R13;
sodimm_sdram_ba[0] : LOCATION = Pin_Y14;
sodimm_sdram_ba[1] : LOCATION = Pin_U13;
sodimm_sdram_cas_n : LOCATION = Pin_R12;
sodimm_sdram_cke : LOCATION = Pin_T13;
sodimm_sdram_cs_n[0] : LOCATION = Pin_C16;
sodimm_sdram_cs_n[1] : LOCATION = Pin_D16;
sodimm_sdram_dq[0] : LOCATION = Pin_V19;
```

```
sodimm_sdram_dq[10] : LOCATION = Pin_T17;
sodimm_sdram_dq[11] : LOCATION = Pin_P16;
sodimm_sdram_dq[12] : LOCATION = Pin_AA3;
sodimm_sdram_dq[13] : LOCATION = Pin_W2;
sodimm_sdram_dq[14] : LOCATION = Pin_Y2;
sodimm_sdram_dq[15] : LOCATION = Pin_Y4;
sodimm_sdram_dq[16] : LOCATION = Pin_W5;
sodimm_sdram_dq[17] : LOCATION = Pin_W21;
sodimm_sdram_dq[18] : LOCATION = Pin_W22;
sodimm_sdram_dq[19] : LOCATION = Pin_Y21;
sodimm_sdram_dq[1] : LOCATION = Pin_U20;
sodimm_sdram_dq[20] : LOCATION = Pin_W19;
sodimm_sdram_dq[21] : LOCATION = Pin_V20;
sodimm_sdram_dq[22] : LOCATION = Pin_W1;
sodimm_sdram_dq[23] : LOCATION = Pin_AB2;
sodimm_sdram_dq[24] : LOCATION = Pin_V1;
sodimm_sdram_dq[25] : LOCATION = Pin_Y1;
sodimm_sdram_dq[26] : LOCATION = Pin_V2;
sodimm_sdram_dq[27] : LOCATION = Pin_Y22;
sodimm_sdram_dq[28] : LOCATION = Pin_AA20;
sodimm_sdram_dq[29] : LOCATION = Pin_AA19;
sodimm_sdram_dq[2] : LOCATION = Pin_W4;
sodimm_sdram_dq[30] : LOCATION = Pin_V21;
sodimm_sdram_dq[31] : LOCATION = Pin_V22;
sodimm_sdram_dq[3] : LOCATION = Pin_V4;
sodimm_sdram_dq[4] : LOCATION = Pin_W3;
sodimm_sdram_dq[5] : LOCATION = Pin_Y3;
sodimm_sdram_dq[6] : LOCATION = Pin_V3;
sodimm_sdram_dq[7] : LOCATION = Pin_Y19;
sodimm_sdram_dq[8] : LOCATION = Pin_R17;
sodimm_sdram_dq[9] : LOCATION = Pin_Y20;
sodimm_sdram_dqm[0] : LOCATION = Pin_Y12;
sodimm_sdram_dqm[1] : LOCATION = Pin_T12;
sodimm_sdram_dqm[2] : LOCATION = Pin_Y11;
sodimm_sdram_dqm[3] : LOCATION = Pin_E17;
sodimm_sdram_ras_n : LOCATION = Pin_E16;
sodimm_sdram_we_n : LOCATION = Pin_C15;
sram0_cs_n : LOCATION = Pin_E4;
sram1_cs_n : LOCATION = Pin_E2;
sw[0] : LOCATION = Pin_V9;
sw[1] : LOCATION = Pin_U9;
sw[2] : LOCATION = Pin_T10;
sw[3] : LOCATION = Pin_U10;
sw[4] : LOCATION = Pin_V10;
sw[5] : LOCATION = Pin_P11;
sw[6] : LOCATION = Pin_U12;
sw[7] : LOCATION = Pin_Y10;
temp_ce : LOCATION = Pin_T3;
txd : LOCATION = Pin_D15;
txd1 : LOCATION = Pin_F13;
v11_j8_p7_nior_u : LOCATION = Pin_V11;
v12_j8_p13_niow_u : LOCATION = Pin_V12;
v6_j8_p38x : LOCATION = Pin_V6;
```