

# Nexys2 EDK Board Support Files Release Notes



Revision: October 26, 2009

This document was produced by  
Digilent Romania. For questions,  
contact [support@digilent.ro](mailto:support@digilent.ro)

## Supporting Both RAM and Flash on Shared Buses

Because the Micron RAM and Intel Flash memories share the same address and data buses, as well as the same OE and WE lines, they are exclusive to each other. That means that with Base System Builder (BSB), only one of the RAM or flash memories can be selected.

However, it is possible to use both the RAM and flash memories on shared buses. See [www.digilentinc.com](http://www.digilentinc.com) for these design examples:

- Nexys2\_500\_Edk\_11\_Ram\_Flash.zip is an EDK 11.3 reference design for the Nexys2 (500K gate) board.
- Nexys2\_1200\_Edk\_11\_Ram\_Flash.zip is an EDK 11.3 reference design for the Nexys2 (1200K gate) board.
- Nexys\_200\_Edk\_11\_Ram\_Flash.zip is an EDK 11.3 reference design for the Nexys (200K gate) board. For the Nexys (400K gate) and (1000K gate) boards change in the project properties the FPGA device accordingly.

These reference designs use three `util_io_mux` I/O multiplexer components to multiplex the Address, Data, and WEN lines. The shared OEN line is connected through an AND glue logic from the two memory controllers.

If you want to quickly build your own system that includes both memories, use the following steps:

1. Create a BSB design with the RAM memory only.
2. Add an `xps_mch_emc` component from the IP catalog, rename it as `Intel_Flash`, and connect it to the PLB bus.
3. Open the `system.mhs` file of the design and locate the following lines at the end of the file:

```
BEGIN xps_mch_emc
...
END
```

4. Use a text editor to open the `system.mhs` file from one of the above reference designs, copy lines 191 to 216 (the content of the `Intel_Flash` parameter and port declarations), and paste them in the local `system.mhs` file between the `BEGIN` and `END` lines.
5. Copy lines 259 to 300 (i.e., the `io_mux` and the `AND` port declarations) from the reference design and paste them into the local `system.mhs` file the end of the file.
6. In the local `system.mhs` file, find the following lines at the `Micron_RAM` declaration (lines 180 and 181):

```
PORT Mem_OEN = fpga_0_Micron_RAM_Mem_OEN_pin
PORT Mem_WEN = fpga_0_Micron_RAM_Mem_WEN_pin
```

Change the lines as follows:

```
PORT Mem_OEN = Micron_RAM_Mem_OEN
PORT Mem_WEN = Micron_RAM_Mem_WEN
```

Insert, before the END statement (that ends the Micron\_RAM declaration), the following lines:

```
PORT Mem_DQ_I = Micron_RAM_Mem_DQ_IPORT Mem_DQ_O = Micron_RAM_Mem_DQ_O
PORT Mem_DQ_T = Micron_RAM_Mem_DQ_T
```

Remove from the Micron\_RAM declaration the following line:

```
PORT Mem_DQ = fpga_0_Micron_RAM_Mem_DQ_pin
```

Now the Data bus is no longer an output pin and is connected through the DQ\_I, DQ\_O, and DQ\_T lines to the data I/O multiplexer.

7. At the beginning of the system.mhs file, where the external port declarations are located, delete the following lines (lines 24, 25, 26 and 28):

```
PORT fpga_0_Micron_RAM_Mem_A_pin =
fpga_0_Micron_RAM_Mem_A_pin_vslice_9_31_concat, DIR = 0, VEC = [9:31]

PORT fpga_0_Micron_RAM_Mem_OEN_pin = fpga_0_Micron_RAM_Mem_OEN_pin, DIR
= 0

PORT fpga_0_Micron_RAM_Mem_WEN_pin = fpga_0_Micron_RAM_Mem_WEN_pin, DIR
= 0

PORT fpga_0_Micron_RAM_Mem_DQ_pin = fpga_0_Micron_RAM_Mem_DQ_pin, DIR =
IO, VEC = [0:15]
```

Add the following port declarations here, after the PORT fpga\_0\_rst\_1\_sys\_rst\_pin line:

```
PORT Flash_Ram_Address_Mux_IO_O_pin = Flash_Ram_Address_Mux_IO_O, DIR =
O, VEC = [0:22]

PORT Flash_Ram_Data_Mux_IO = Flash_Ram_Data_Mux_IO, DIR = IO, VEC =
[0:15]

PORT Flash_Ram_OEN_AND_Res_pin = Flash_Ram_OEN_AND_Res, DIR = O, VEC =
[0:0]

PORT Flash_Ram_WEN_Mux_IO_O_pin = Flash_Ram_WEN_Mux_IO_O, DIR = O, VEC =
[0:0]

PORT fpga_0_INTEL_FLASH_Mem_RPN_pin = fpga_0_INTEL_FLASH_Mem_RPN_pin,
DIR = O

PORT fpga_0_INTEL_FLASH_Mem_CEN_pin = fpga_0_INTEL_FLASH_Mem_CEN, DIR =
O
```

8. Open the project's system.ucf file and replace its contents with the contents of the target board's system.ucf file (found in the reference design's /data subfolder). For example, if the target board is the Nexys2 (500K gate) board, then the system.ucf file can be found in the Nexys2\_500\_Edk\_11\_Ram\_Flash.zip archive in the /data subfolder.

9. Now you can start generating the bitstream.

## Revision History

### Version 2.1

- The board support files are compatible with BSB under EDK 10, EDK 11, and Linux.
- Only EDK standard I/O interfaces are used, custom IP cores were removed.
- Micron\_Ram and Intel\_Flash are exclusive to each other under BSB.
- For the Nexys 200K, 400K, and 1000K gate boards, only the PMOD\_RS232\_Conn\_JA serial port is selectable, i.e., from BSB, a PMOD-RS232 port can be connected only to the JA port.
- If a PMOD-RS232 port needs to be connected to another port, update the system.ucf file accordingly. Refer to the Nexys documentation to find the correct ucf constraints.

### Version 1.20

- The board support package is compatible with EDK version 9.2 and above.
- In order to access the Nexys onboard memories, the Nexys\_EMG OPB bus compliant memory controller was replaced by a modified version of the PLBv4.6 bus compliant XPS\_MCH\_EMG memory controller, namely XPS\_MCH\_EMG\_DIGILENT.
- Created the *Nexys Base System Builder Guide for EDK* document.

### Version 1.01

- Added support for Nexys 2 board, rev. A.

### Version 1.00

- This is the release version.