Stephen MacMahon
stephenm@xilinx.com

UART Interrupt Creation on Spartan 3A

This tutorial will demonstrate the UART Interrupt based application. To show this we will build a simple Interrupt application that will use the hyper-terminal to create an interrupt.

The second half of the tutorial will show how to convert from the standalone OS to the Xilkernel OS.

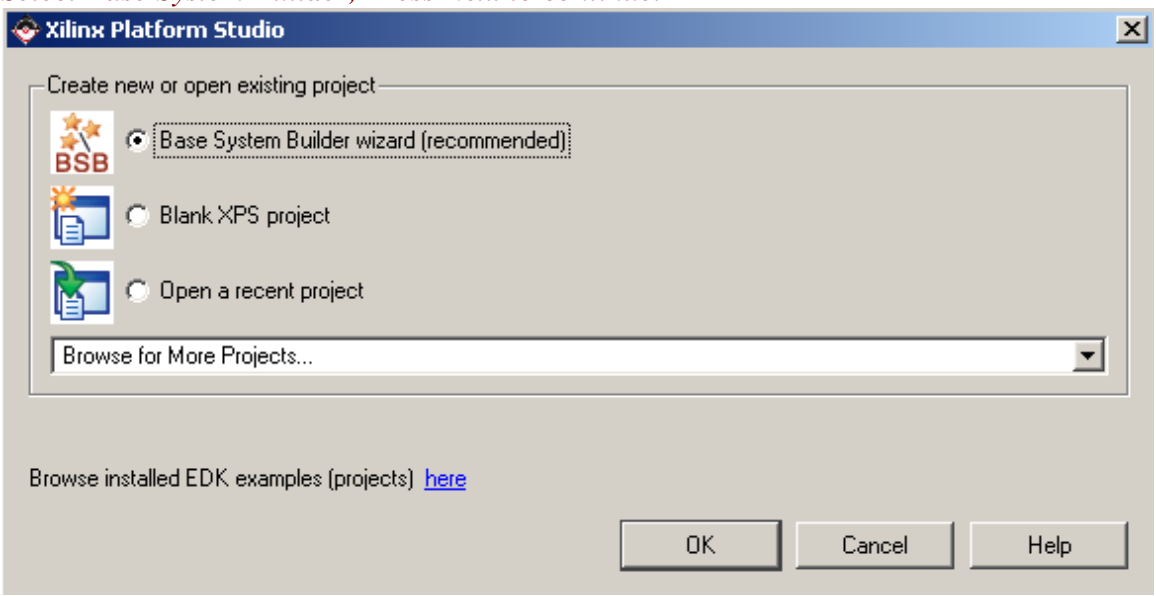
This application will use the Spartan 3A board. The design will be created using EDK 10.1.03i.

Step 1: Create the Interrupt Project.

To create the project, the Base System Builder (BSB) in XPS will be used. To launch XPS, go to Start -> Run and type "XPS", this will launch the XPS tool.

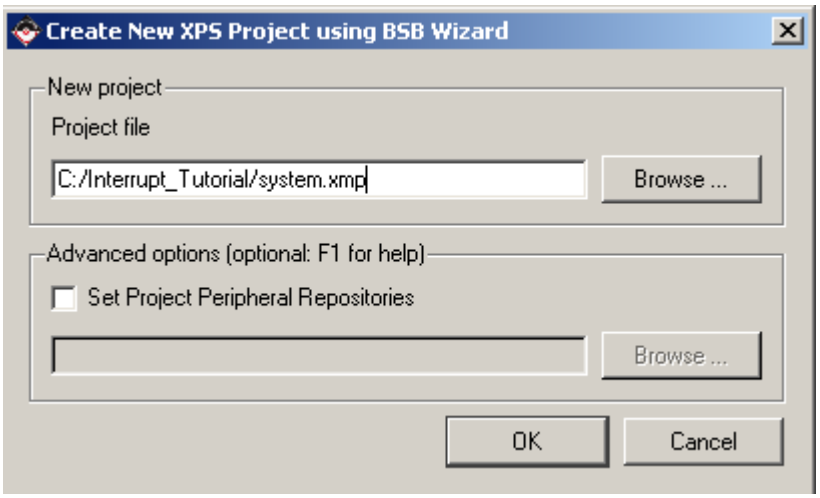
Follow the steps seen below to build the BSB design:

Select Base System Builder, Press Next to continue.



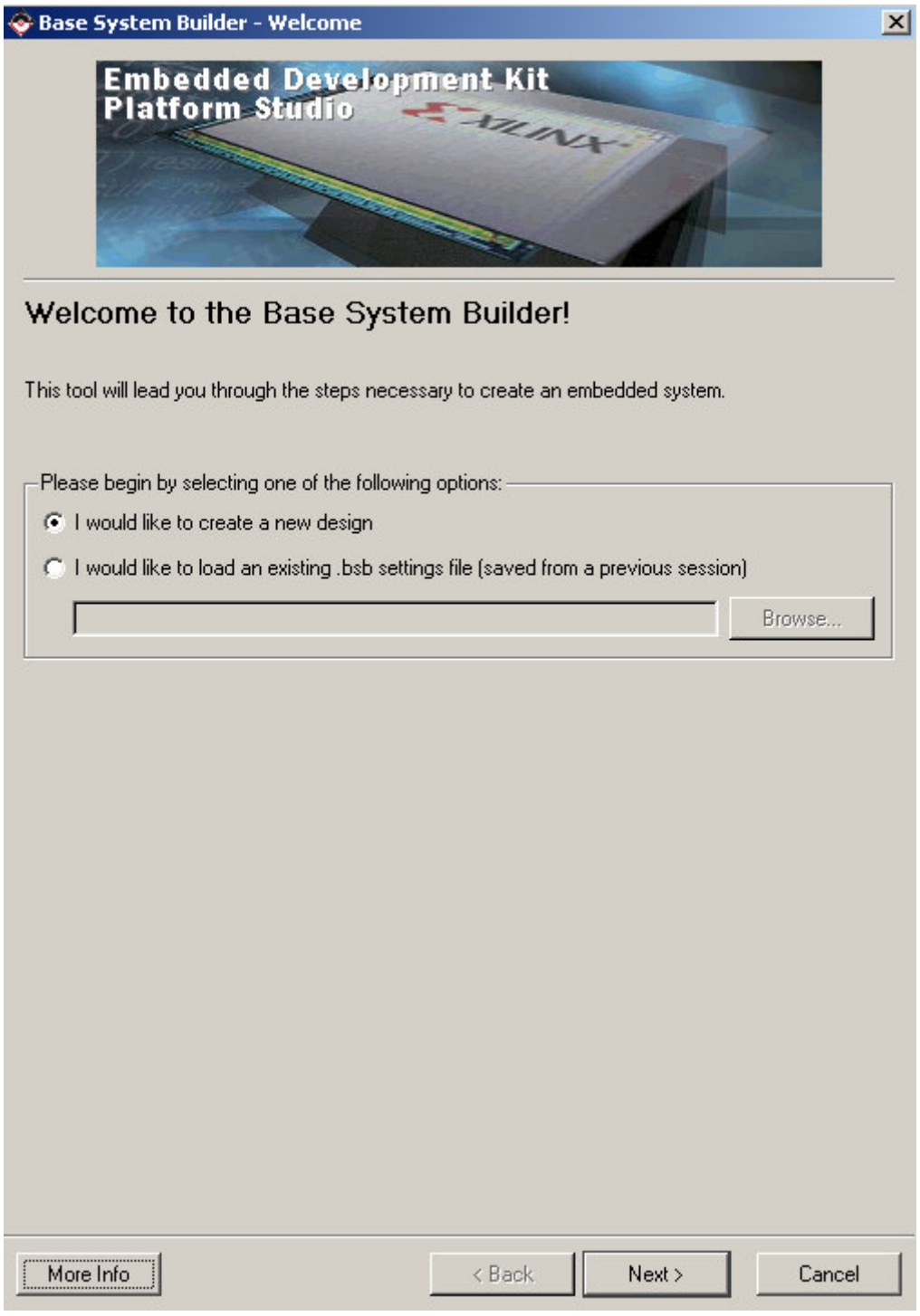
Stephen MacMahon
stephenm@xilinx.com

Browse to where you want to save your interrupt project, Press Next to continue.



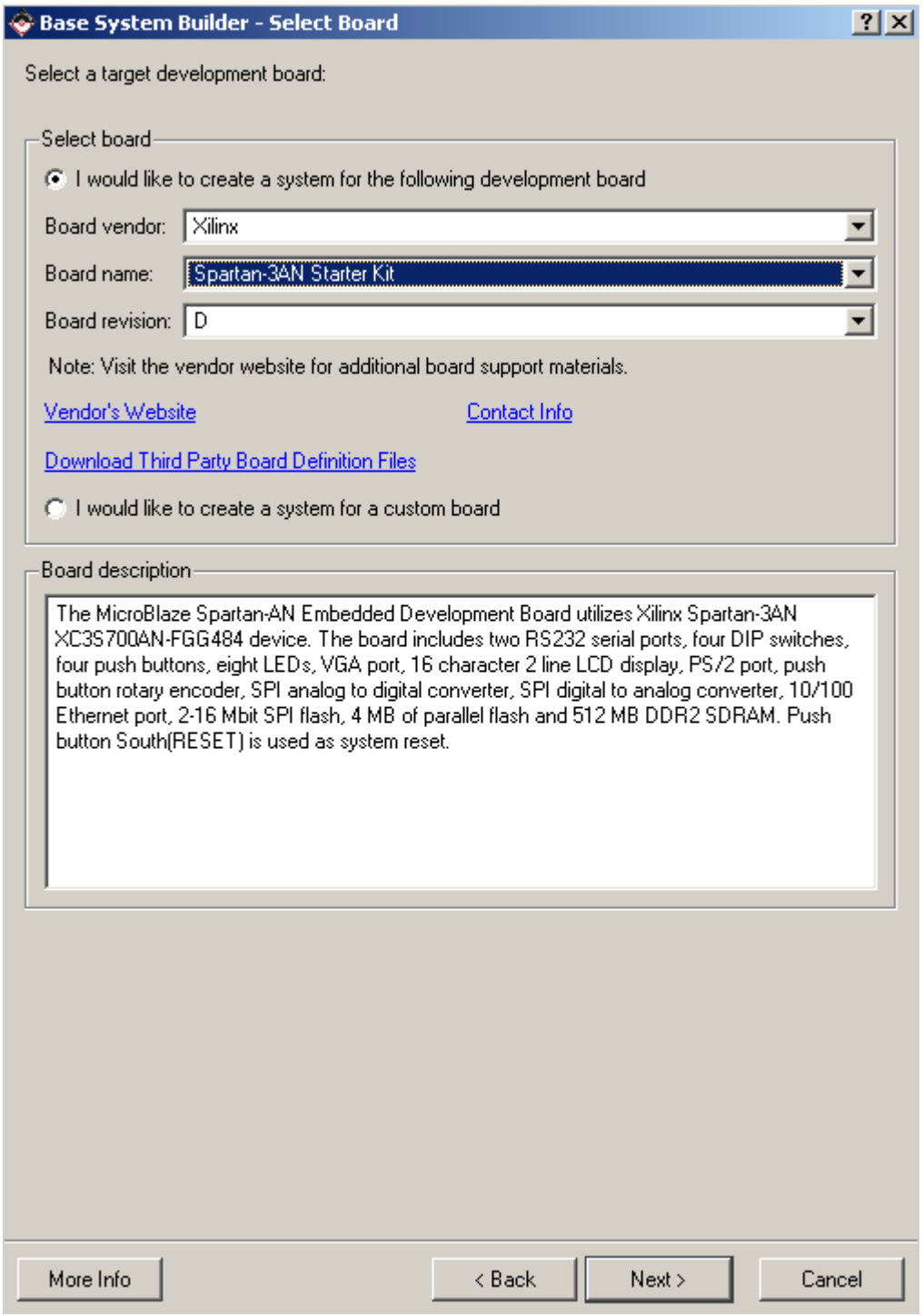
Stephen MacMahon
stephenm@xilinx.com

Select "I would like to create a new design", Click Next to continue.



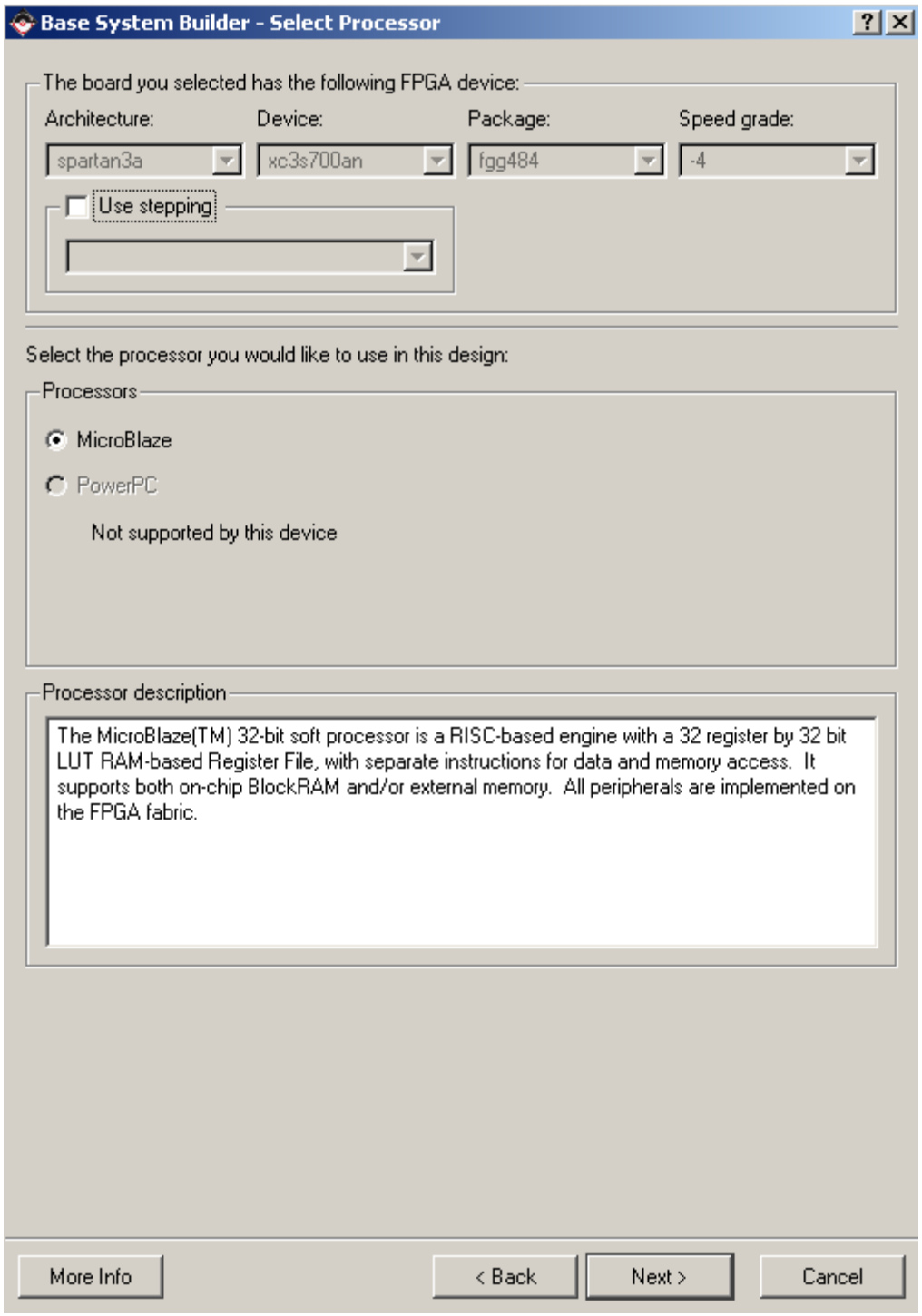
Stephen MacMahon
stephenm@xilinx.com

Select the Spartan 3A Board as shown below, Select Next to Continue



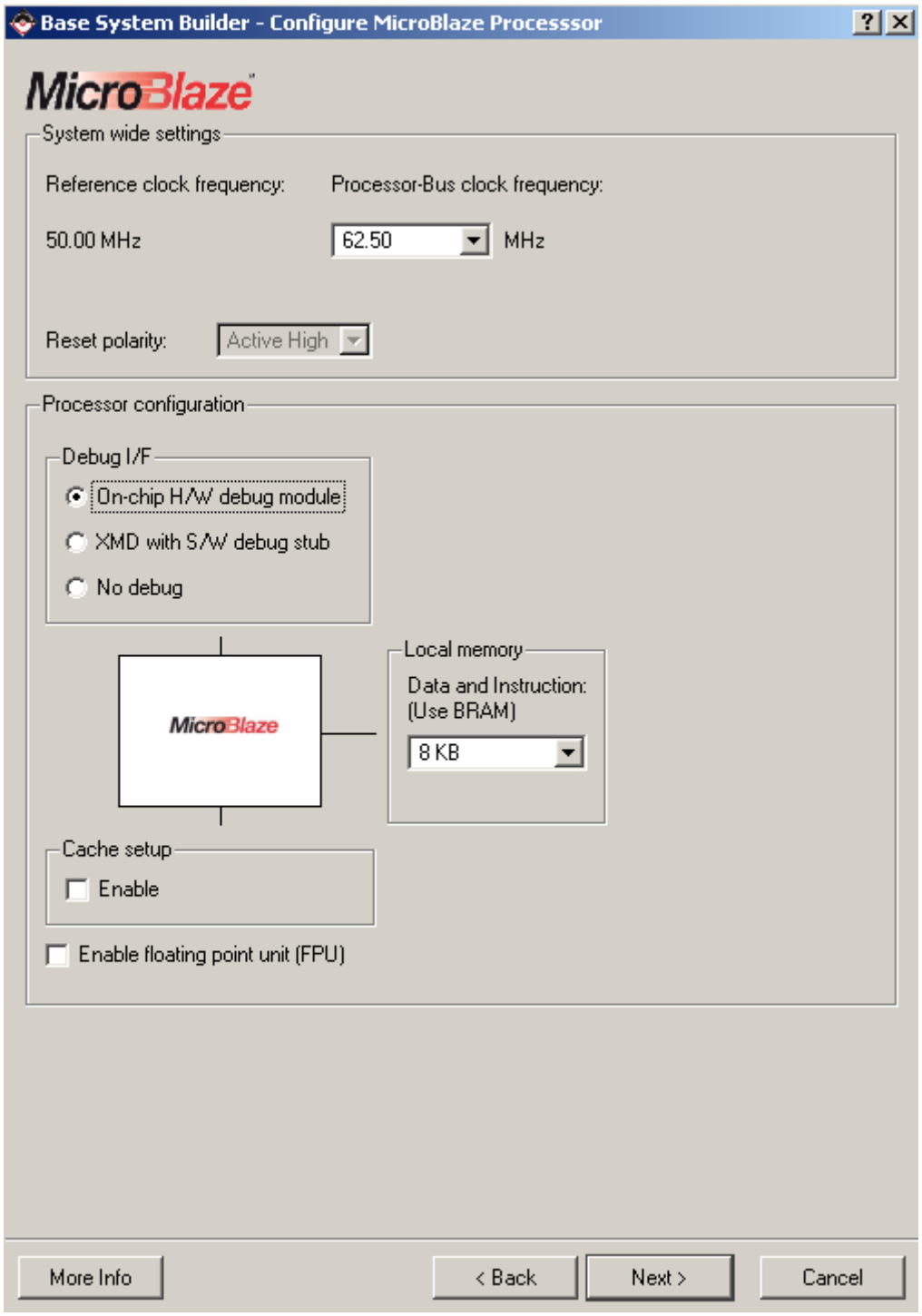
Stephen MacMahon
stephenm@xilinx.com

Select the Microprocessor, Press Next to Continue.



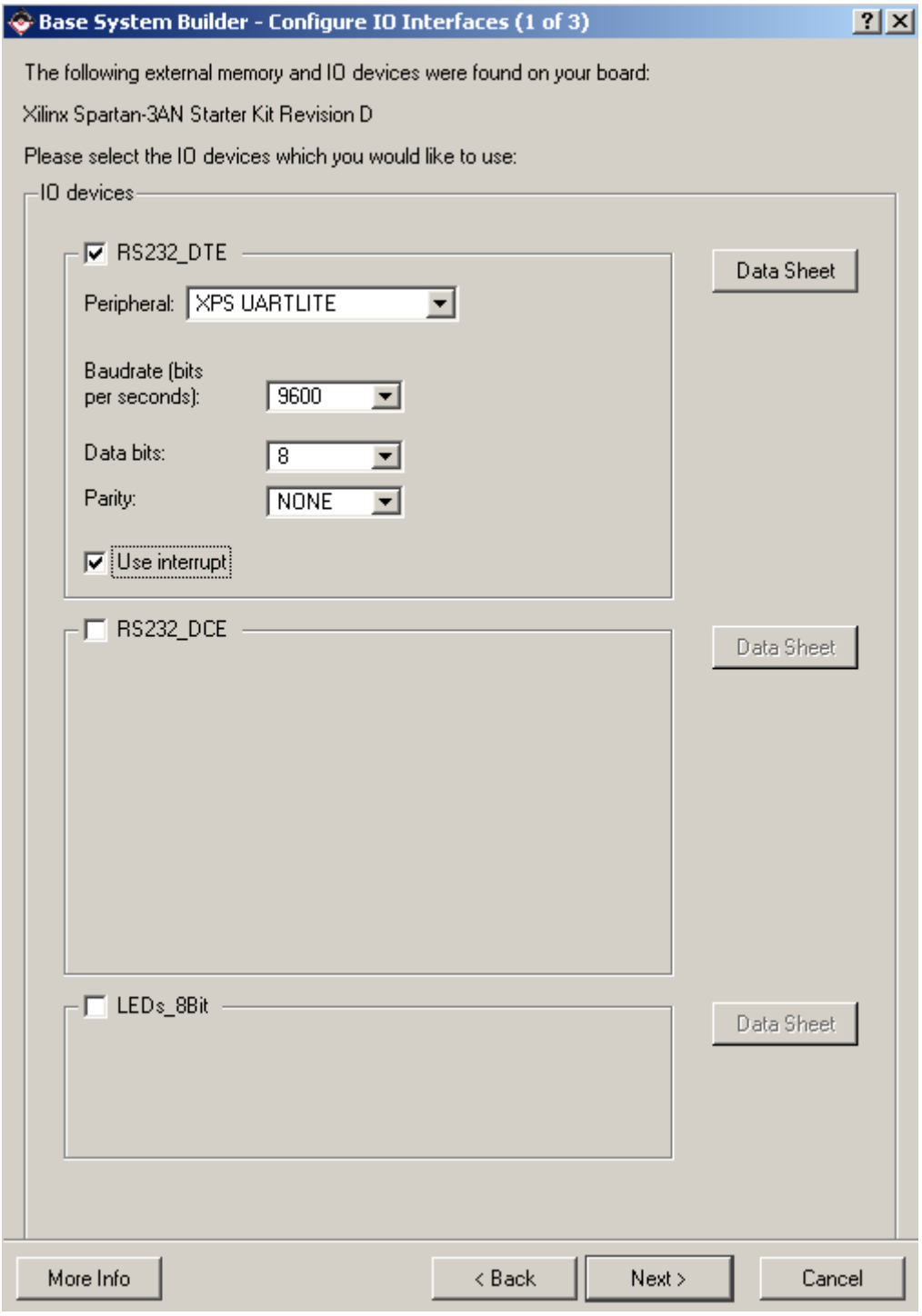
Stephen MacMahon
stephenm@xilinx.com

Select 8KB memory for the Local Memory, Press Next to continue.



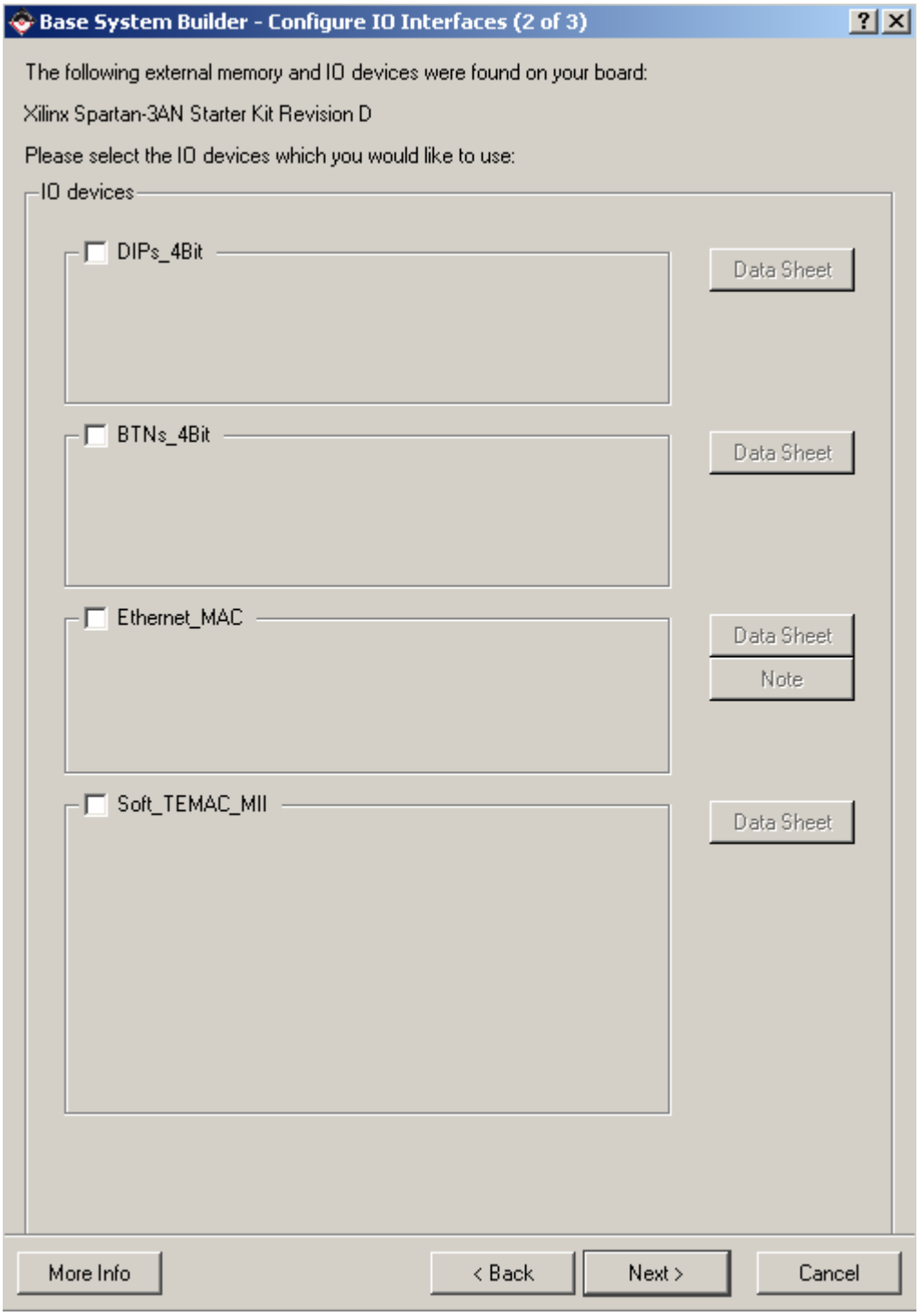
Stephen MacMahon
stephenm@xilinx.com

Select the peripherals as shown below, Press Next to Continue.



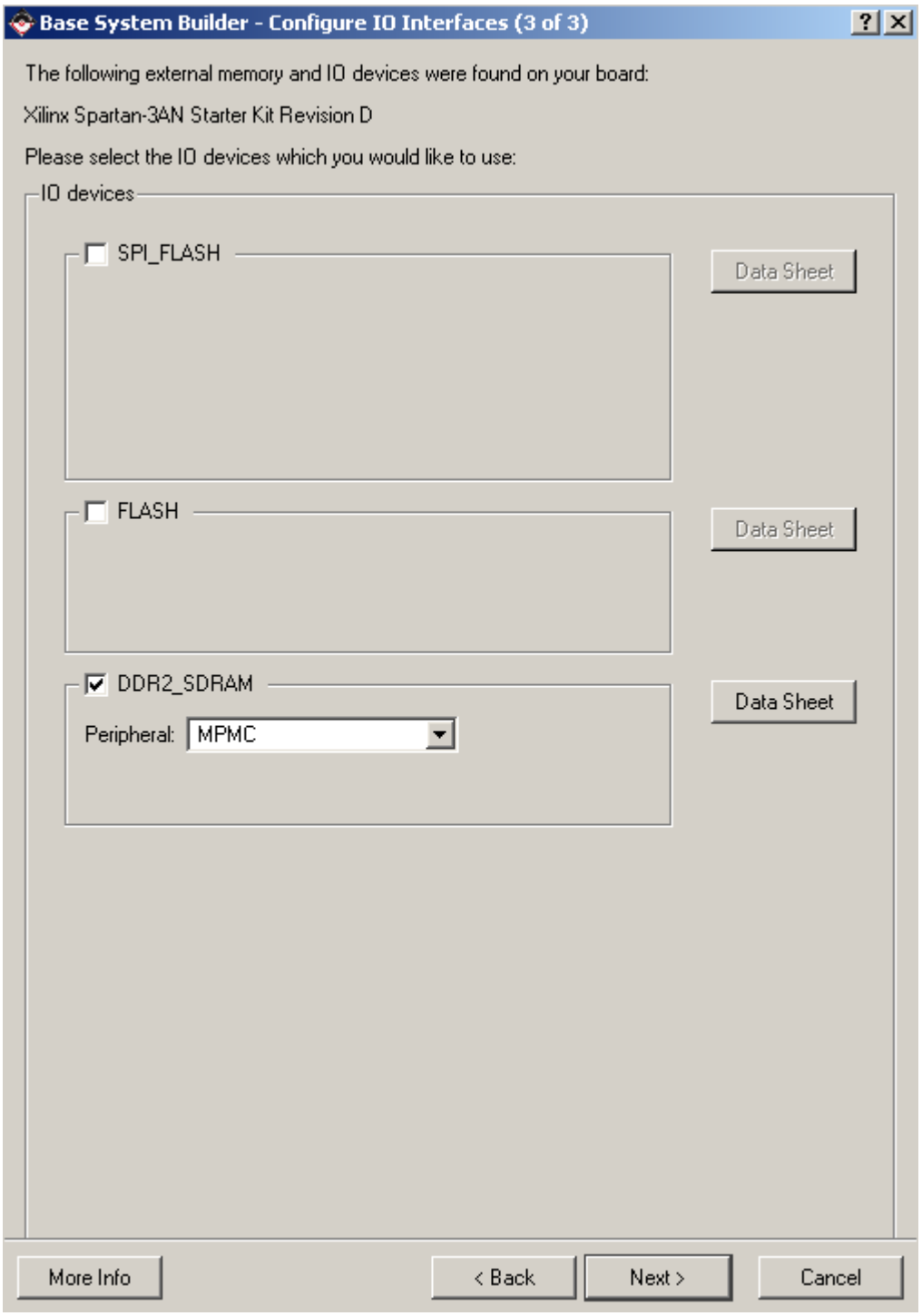
Stephen MacMahon
stephenm@xilinx.com

Select the peripherals as shown below, Press Next to Continue.



Stephen MacMahon
stephenm@xilinx.com

Select the peripherals as shown below, Press Next to Continue.

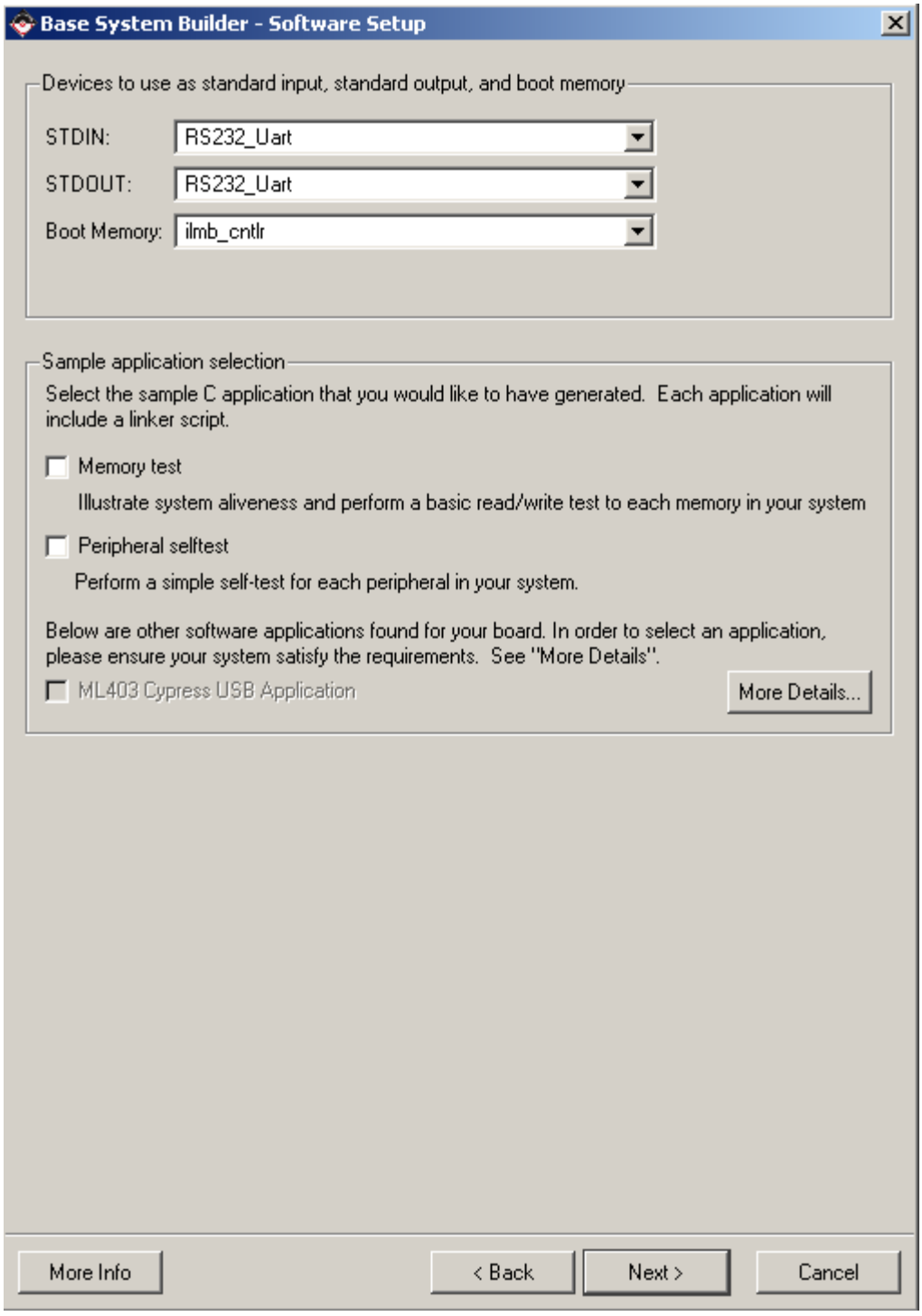


Stephen MacMahon
stephenm@xilinx.com

Select Next to continue

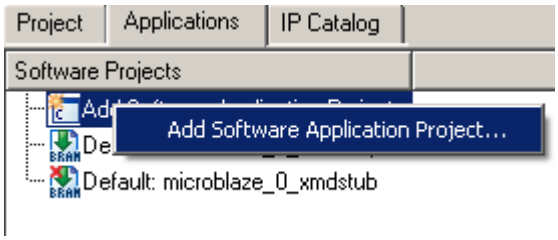
Stephen MacMahon
stephenm@xilinx.com

In software setup, de-select the button for the two Application, shown below.
Press Next -> Generate -> Finish to finish

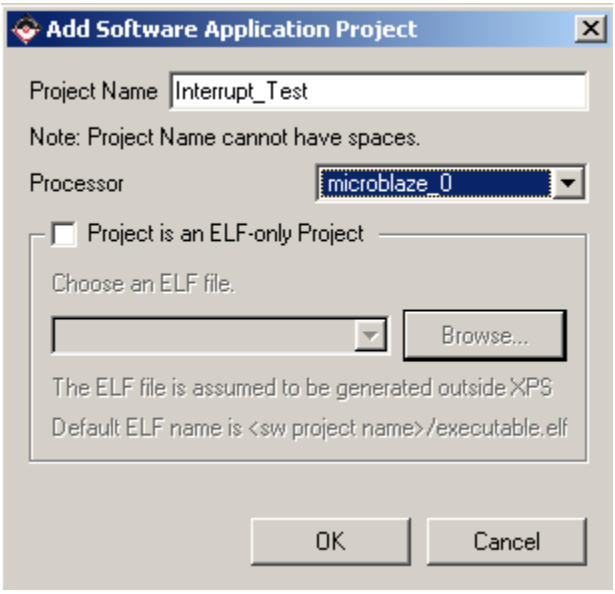


Stephen MacMahon
stephenm@xilinx.com

The next thing to do is to build the software application. This software application will be a simple application to show how to register the interrupt and to register the handler. To create a new application, Click on the Application tab on the left and right click on "Add Software Application Project" seen below:

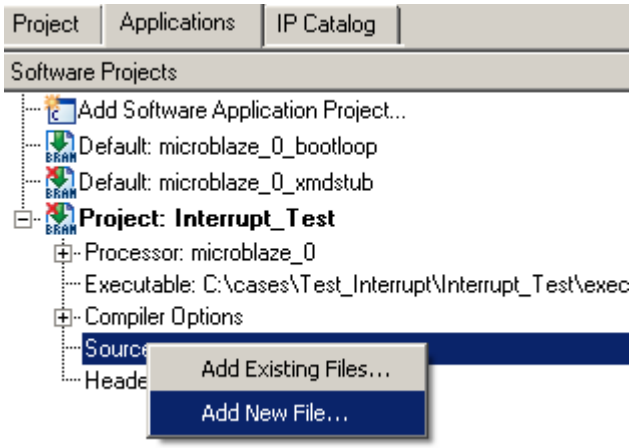


In the pop-up box, name your Application "Interrupt_Test" as shown below. Select OK to continue:

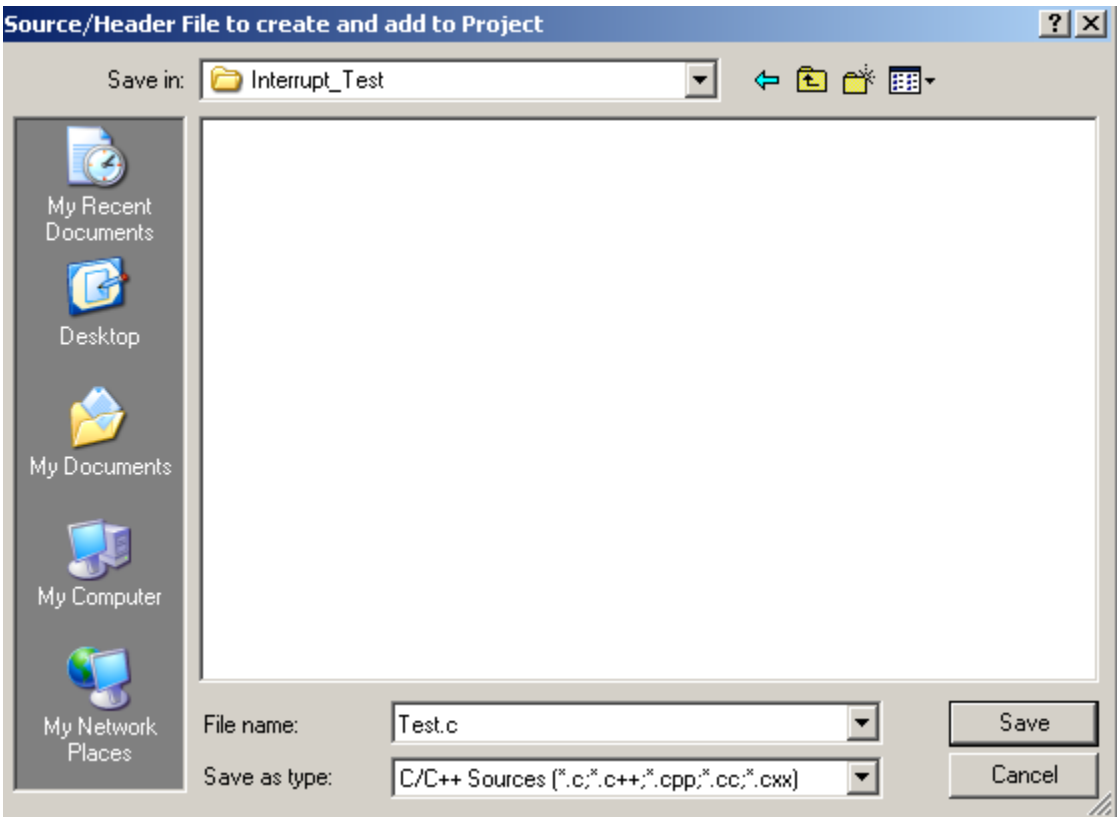


Stephen MacMahon
stephenm@xilinx.com

This will create an empty project for you. To create the source code for the Interrupt_Test application, right click on "source" in the application and select "Add New File…" This is seen below:



Browse to the Interrupt_Test folder, if it isn't there create one making sure you name it correctly. Name the source file "Test.c", making sure you use a lower case c for the file. This is seen below:



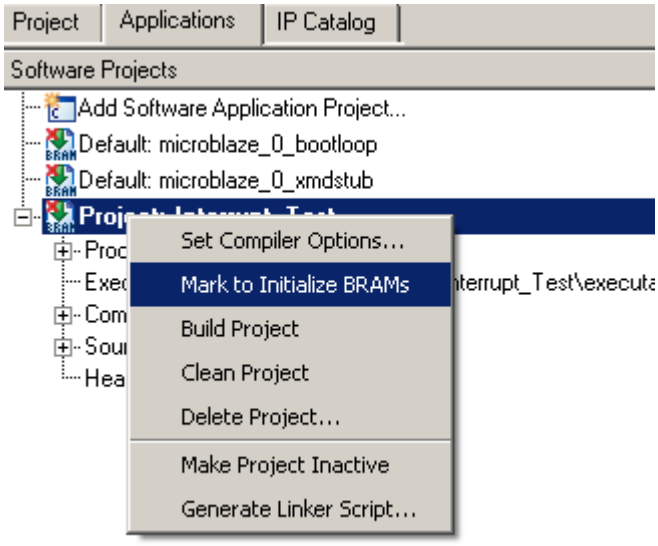
Stephen MacMahon
stephenm@xilinx.com

The next step is to copy the code seen below into the empty Test.c file. This code is shown below:

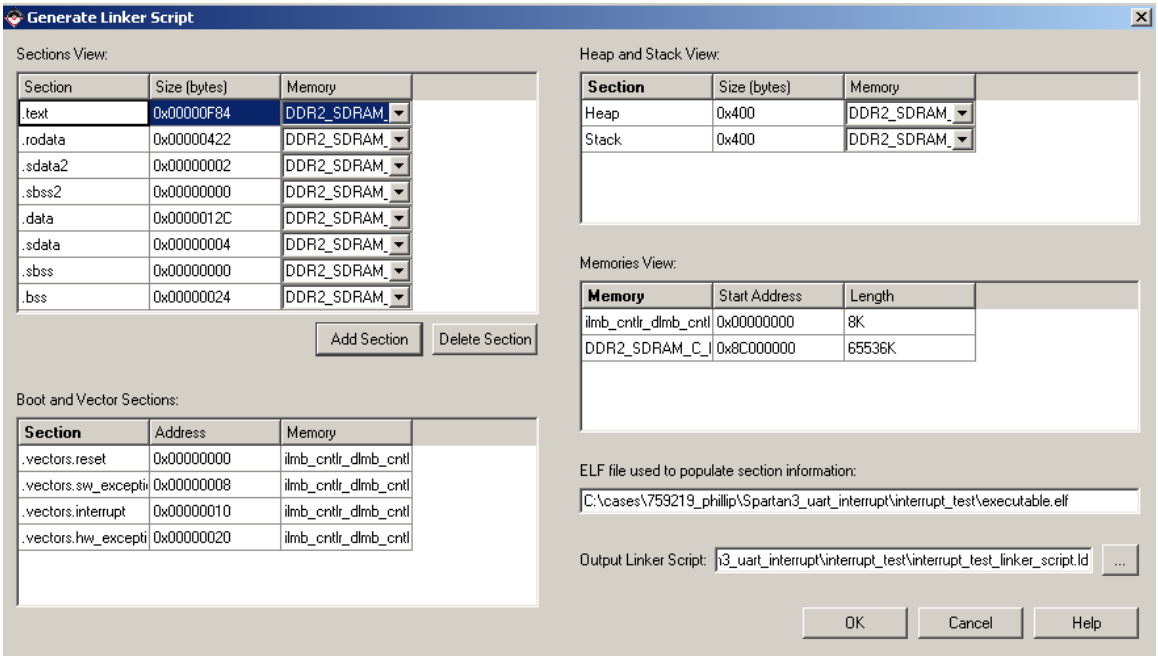
```
#include <xintc_l.h>
#include <xparameters.h>
#include <xuartlite_l.h>

#define Interrupt 0
#define Mask 0X000001


   //debug interrupt service routine
   void debug_int_handler(void *baseaddr_p)
   {
      char c;
      while (!XUartLite_mIsReceiveEmpty(XPAR_UARTLITE_0_BASEADDR))
      {
         c = XUartLite_RecvByte(XPAR_UARTLITE_0_BASEADDR);
         xil_printf("You Entered %c\r\n",c);
      }
   }


   int main()
   {

      microblaze_enable_interrupts();

      XIntc_RegisterHandler(XPAR_INTC_SINGLE_BASEADDR,Interrupt,
      (XInterruptHandler)debug_int_handler,(void *)XPAR_UARTLITE_0_BASEADDR);

      XIntc_mMasterEnable(XPAR_XPS_INTC_0_BASEADDR);

      XIntc_mEnableIntr(XPAR_XPS_INTC_0_BASEADDR, Mask);

      XUartLite_mEnableIntr(XPAR_UARTLITE_0_BASEADDR);

      xil_printf("Interrupt Setup Finished....\r\n");

      //Wait for interrupts to occur
      while (1)
      {
      }

      return 0;
   }
```

Stephen MacMahon
stephenm@xilinx.com

Now that the code has been written, the project can be intialized into the BRAMS. To do this right click on the Interrupt_Test application and select *"Mark to initialize brams…"* This is seen below:



Next Generate the Linker script for the Interrupt_Test application. To do this right click on the Interrupt_Test application and select *"Generate Linker Script"* Place all sections into the DDR2 SDRAM, this is seen below:



Stephen MacMahon
stephenm@xilinx.com

- The next step is to compile the Libraries and BSP's, to do this select Software -> Generate Libraries and BSP's.

- Then build the Interrupt_Test application, to do this go to Software -> Build all user applications…

- Now build the bitstream, to do this go to Hardware -> generate bitstream.

- Now update the bitstream with the software, to do this go to Device Configuration -> Update bitstream.

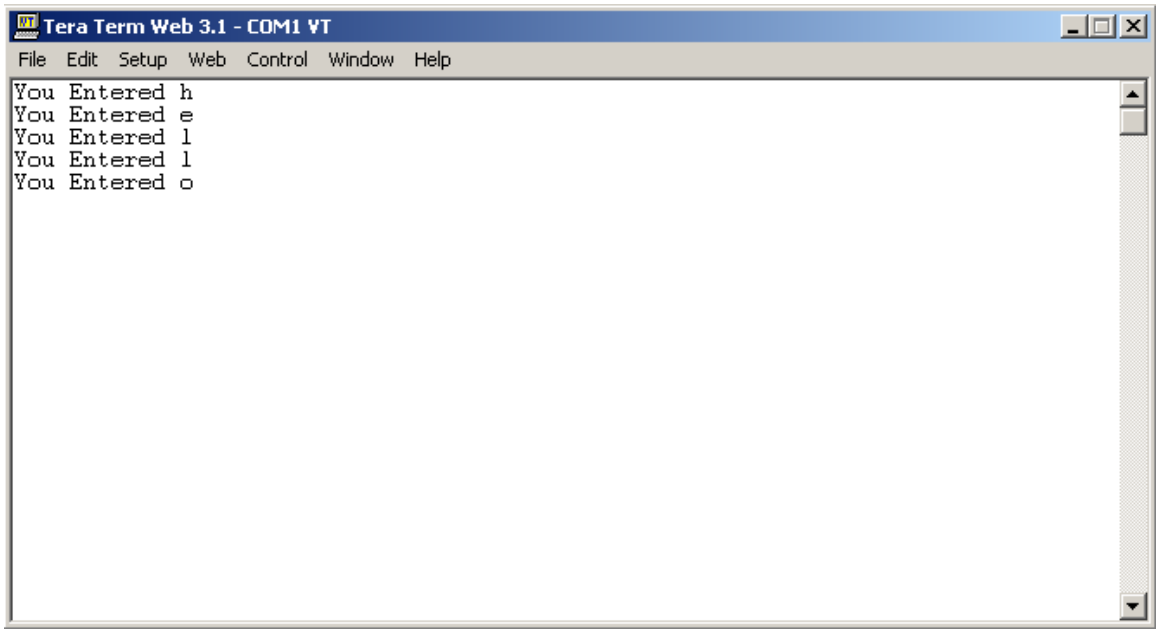
- Finally, download the bitstream to the board, to do this go to Device Configuration -> Download bitstream.

- Open the Hyper-Terminal if you haven't done so and set the baud rate to 9600

- Open the XMD, type the follow commands:

  dow interrupt_test/executable.elf
  run

- When you type something on the keyboard you should see it on the hyper-terminal, as seen below:



Stephen MacMahon
stephenm@xilinx.com

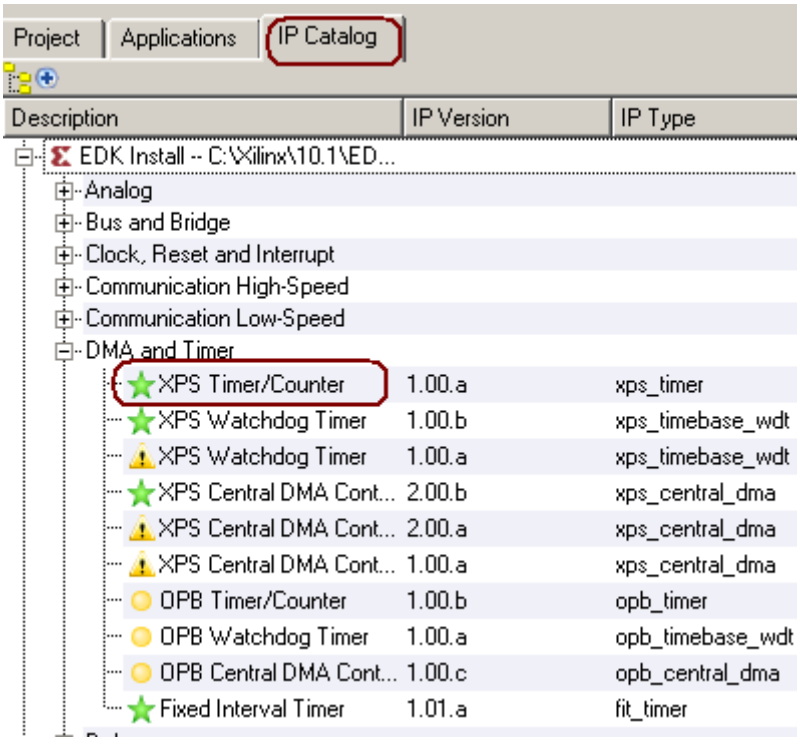
Updating design from Standalone OS to Xilikernel OS

Firstly, download the Mircoblaze application see on the page above. For this example the Spartan 3AN will be used:

http://xirweb/~stephenm/Projects/Spartan_3AN_Uart_interrupt.zip

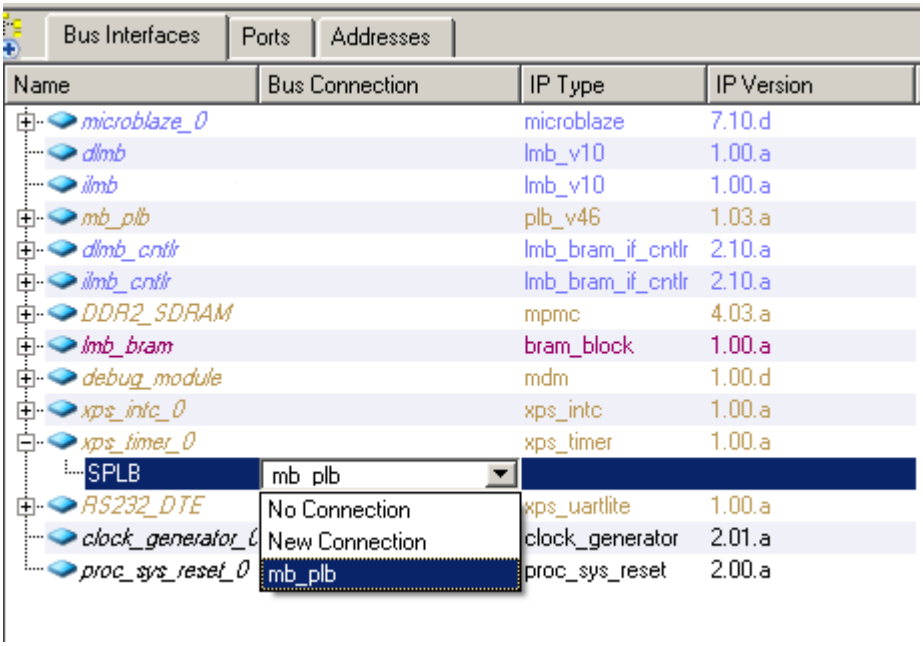
Step 1: Add timer, connect to plb and give it an address space.

A timer will need to be added, as this is needed by the Xilkernel OS when using Microblaze. To add the timer, go to the IP Catalog and drop the DMA and Timer list. Double click the XPS timer/Counter, seen below:

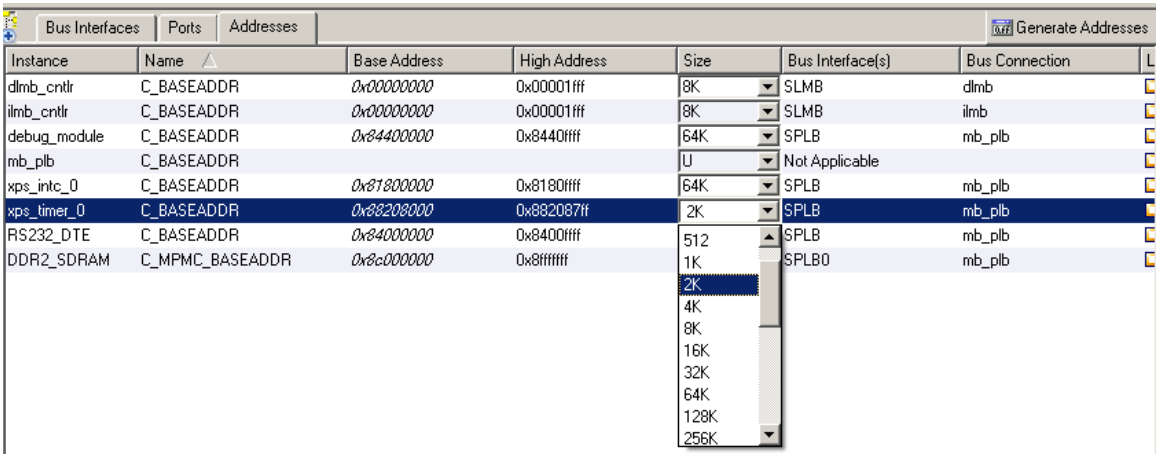


Stephen MacMahon
stephenm@xilinx.com

To connect the xps_timer_0 to the PLB, In system Assembly view select the Bus interface tab, drop down the xps_timer_0 and select mb_plb. Shown below:



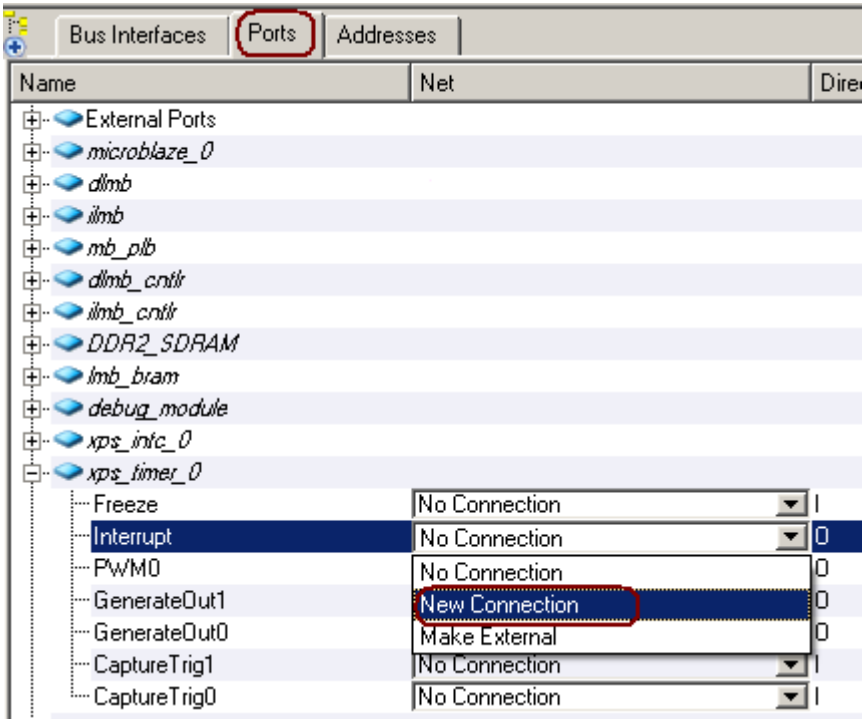
To give the xps_timer_0 an address space, In system Assembly view select the Addresses tab, drop down the xps_timer_0 and select 2K. Then click Generate Addresses. Shown below:



Stephen MacMahon
stephenm@xilinx.com

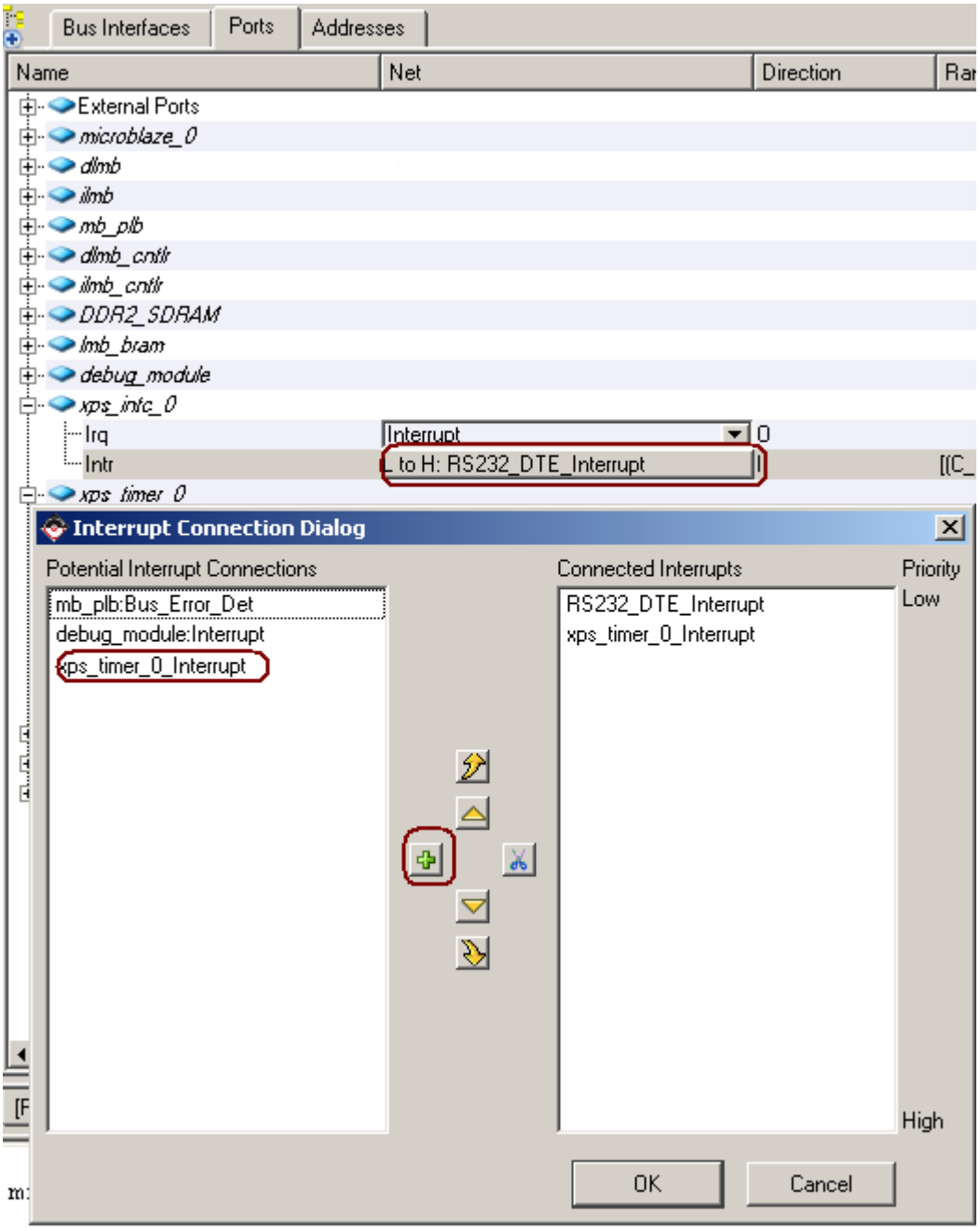
Step 2: Connect Timer to interrupt controller.

In the System Assembly view, click the Ports tab then drop down the timer ports. Make a new connection for the timer interrupt port, seen below:



Stephen MacMahon
stephenm@xilinx.com

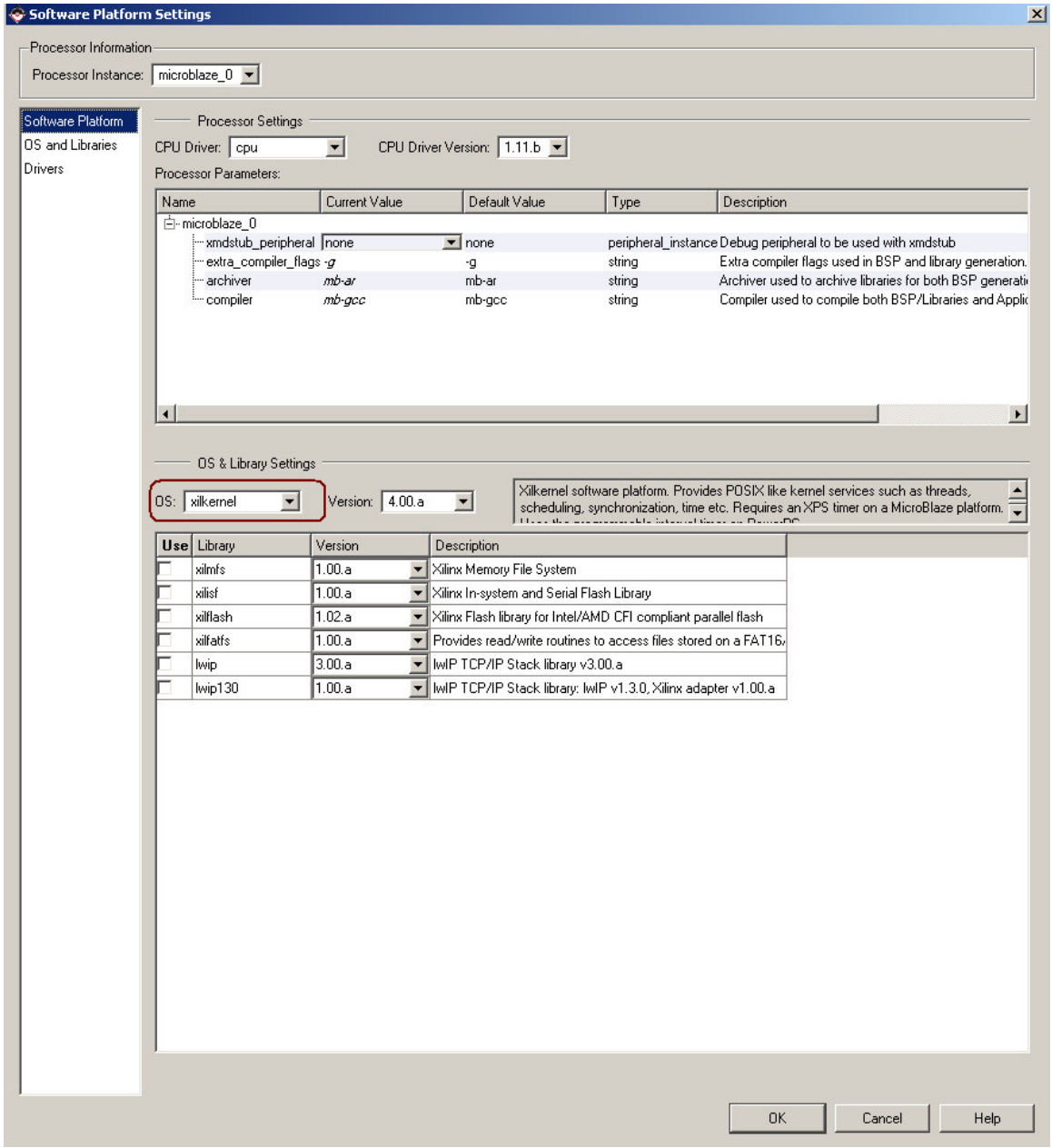
Then select the interrupt controller, Select the intr port. This should open the GUI. Select the xps_timer_0_interrupt signal and click on the green cross. Seen below:



Stephen MacMahon
stephenm@xilinx.com

Step 3: Change the OS from Standalone to Xilkernel

Go to Software -> Software Platform Settings. Change the OS to Xilkernel, seen below;

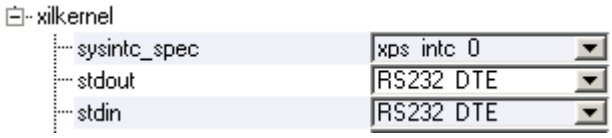


Stephen MacMahon
stephenm@xilinx.com

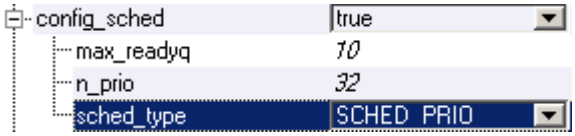
Step 4: Set-up the Xilkernel system

Go to Software -> Software Platform Settings. In OS an Libraries

For sysintc_spec, select xps_intc_0. For stdout and stdin, select RS232_DTE. Seen below:

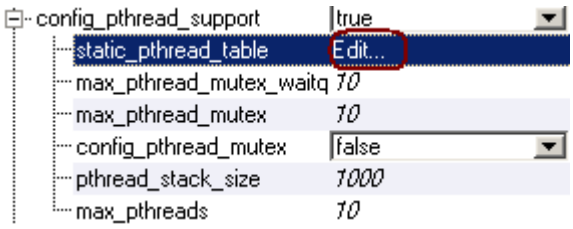


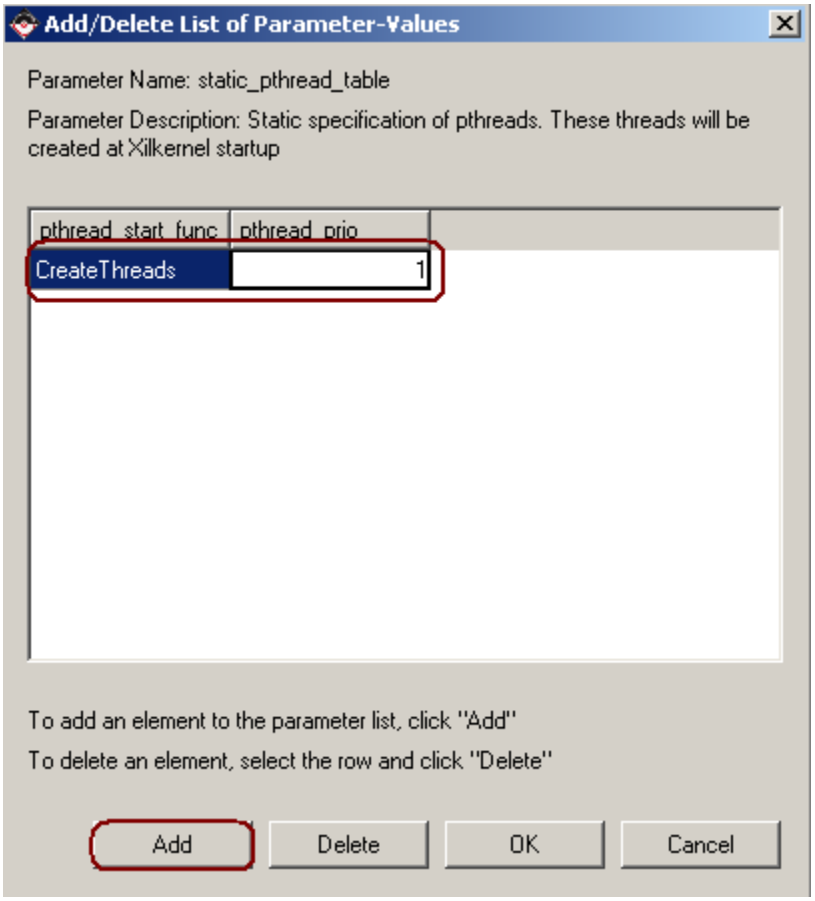
For sched_type, select SCHED_PRIO



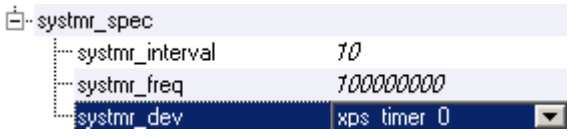
Stephen MacMahon
stephenm@xilinx.com

For config_pthread_support, double click on Edit (seen below). In the GUI, select Add And enter the thread name “CreateThreads” with a priority of 1 (seen below)





For the systmr_spec, in the systmr_dev, select the xps_timer_0



Stephen MacMahon
stephenm@xilinx.com

The MSS file should be same as below:

```
BEGIN OS
 PARAMETER OS_NAME = xilkernel
 PARAMETER OS_VER = 4.00.a
 PARAMETER PROC_INSTANCE = microblaze_0
 PARAMETER sysintc_spec = xps_intc_0
 PARAMETER stdout = RS232_DTE
 PARAMETER stdin = RS232_DTE
 PARAMETER sched_type = SCHED_PRIO
 PARAMETER systmr_dev = xps_timer_0
 PARAMETER static_pthread_table = ((CreateThreads,1))
END
```

Step 5: Run LibGen

To run LibGen, go to Software -> Generate Libraries and BSP's

Stephen MacMahon
stephenm@xilinx.com

Step 6: Build the Xilkernel Application

```
#include "xmk.h"
#include <os_config.h>
#include <sys/process.h>
#include <pthread.h>
#include <sys/intr.h>
#include <xparameters.h>
#include <xuartlite_l.h>
#include <xstatus.h>

//Variables
static pthread_t                tid;
static pthread_attr_t           attr;
static struct sched_param       spar;

//Funtion Prototype
void* uart_thread();

/* uartlite interrupt service routine */
void uart_int_handler(void *baseaddr_p) {
        char c;
        /* till uart FIFOs are empty */
        while (!XUartLite_mIsReceiveEmpty(XPAR_RS232_DTE_BASEADDR)) {
                /* read a character */
                c = XUartLite_RecvByte(XPAR_RS232_DTE_BASEADDR);
                /* print character on hyperterminal (STDOUT) */
                xil_printf ("Character: %c \r\n", c);
        }
}

void main()

{
        xilkernel_main();

}
```

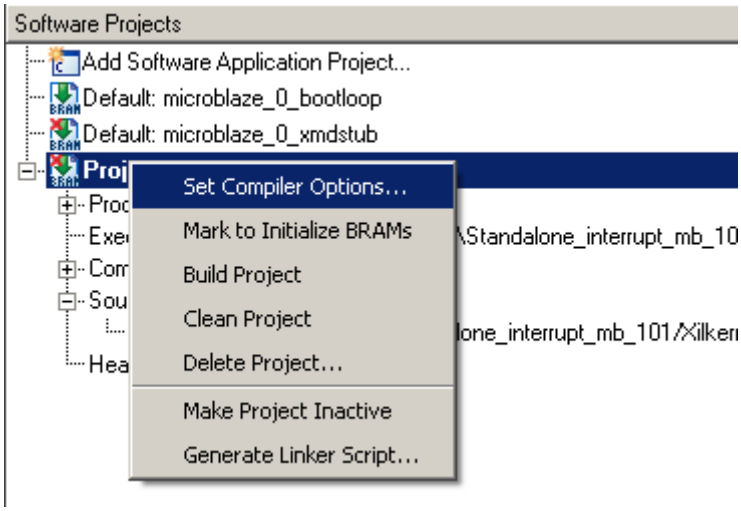
Stephen MacMahon
stephenm@xilinx.com

```
void* CreateThreads(void* dummy)
{
        pthread_attr_init (&attr);
        spar.sched_priority = 1;
        pthread_attr_setschedparam(&attr,&spar);
        pthread_create (&tid, &attr, (void*)uart_thread, NULL);

        return 0;
}

void* uart_thread()
{

        int_id_t UartIntrId = XPAR_INTC_0_UARTLITE_0_VEC_ID;

        register_int_handler(UartIntrId,uart_int_handler,NULL);

        enable_interrupt(UartIntrId);

        XUartLite_mEnableIntr(XPAR_RS232_DTE_BASEADDR);

        /* Wait for interrupts to occur */
        while (1);

}
```

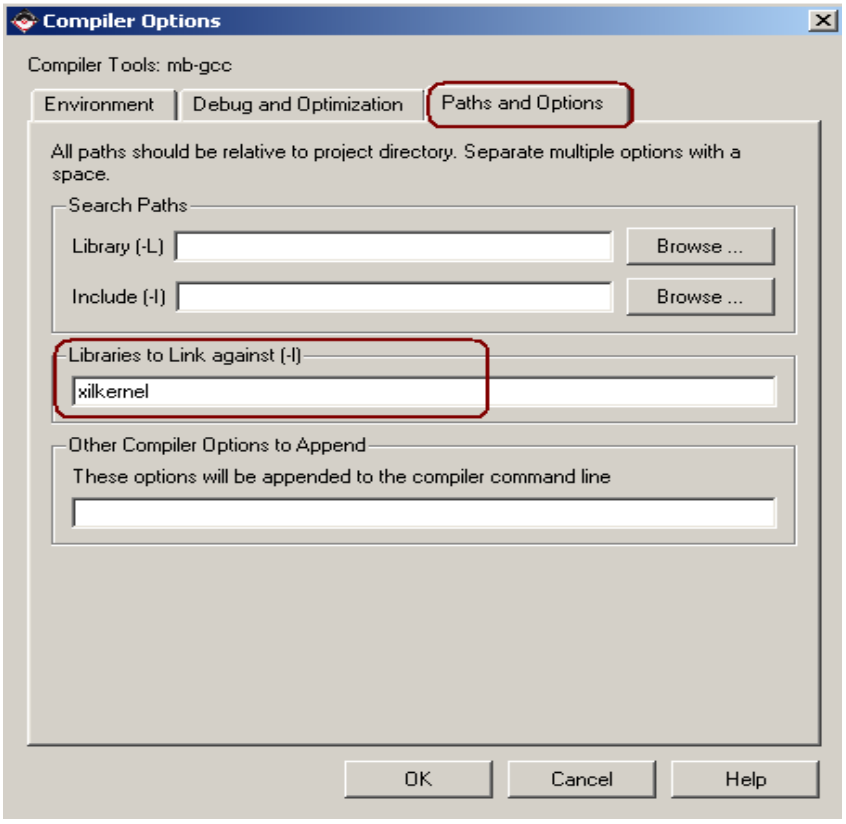
Stephen MacMahon
stephenm@xilinx.com

Step 7: Set compiler options to compile Xilkernel library files.

Right click on the Interrupt_Test application then select Set Compiler Options, seen below:



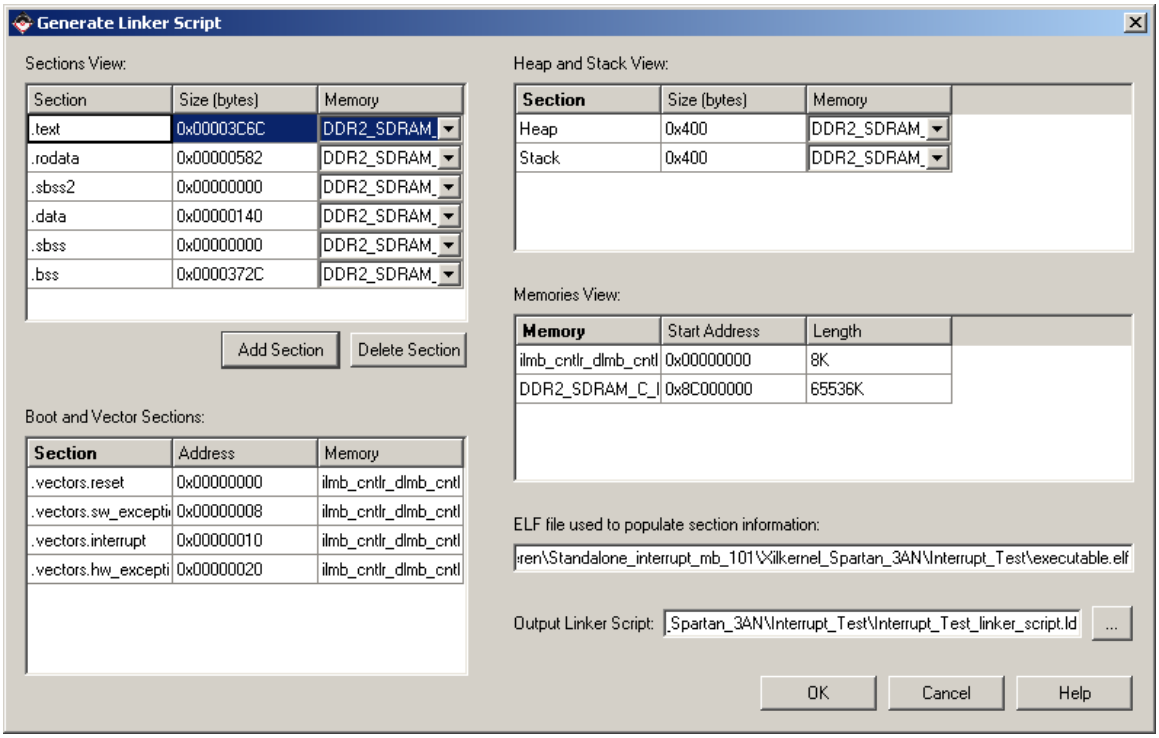
Under the Paths and Options tab, type Xilkernel into the libraires to link against field, shown below:



Stephen MacMahon
stephenm@xilinx.com

Step 8: Generate the linker script, and download application.

Place all section in the DDR2_SDRAM. Also, increase the heap to 0x400. Shown below:

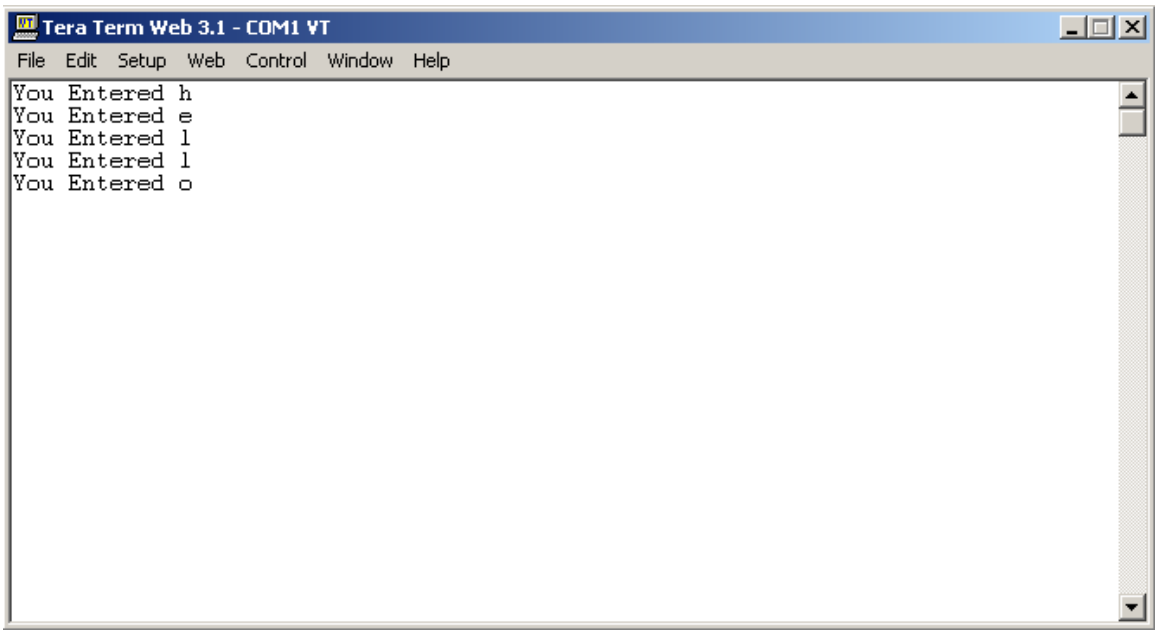


- Then build the Interrupt_Test application, to do this go to Software -> Build all user applications…
- Now build the bitstream, to do this go to Hardware -> generate bitstream.
- Now update the bitstream with the software, to do this go to Device Configuration -> Update bitstream.
- Finally, download the bitstream to the board, to do this go to Device Configuration -> Download bitstream.
- Open the Hyper-Terminal if you haven't done so and set the baud rate to 9600
- Open the XMD, type the follow commands:

  dow interrupt_test/executable.elf
  run

Stephen MacMahon
stephenm@xilinx.com

- When you type something on the keyboard you should see it on the hyper-terminal, as seen below:



The Xilkernel project can be downloaded from the link below:

http://xirweb/~stephenm/Projects/Xilkernel_Spartan_3AN.zip