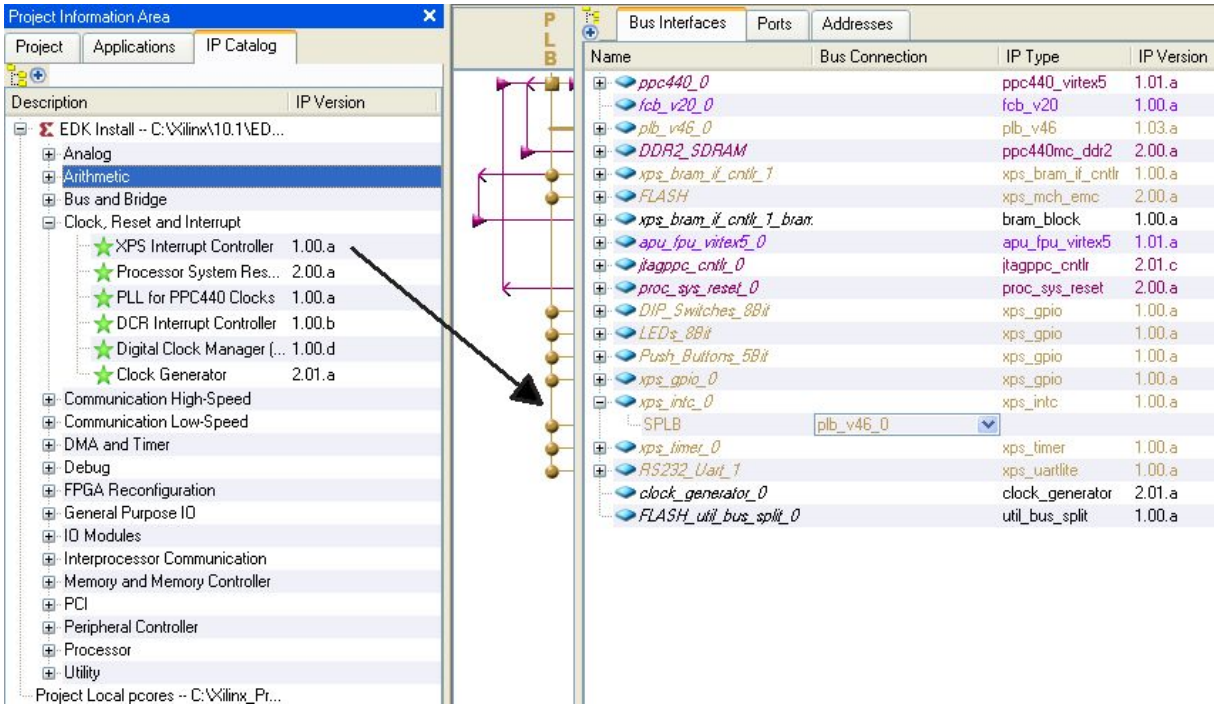
Anhang 1

Anleitung zur Benutzung des Interruptcontrollers am Beispiel der Pushbuttons

1. Interruptcontroller IP-Core zur Basisconfiguration hinzufügen und mit PLB Bus verbinden (falls nicht vorhanden uartlite IP-Core hinzufügen um Erfolg per Hyperterminal zu testen)



2. TAB System Assembly View und TAB Ports

- Push_Buttons_5Bit → Interrupt: New Connection
- xps_intc_0 → Intr: Push_Buttons_5Bit_IP2INTC_Irpt hinzufügen
- ppc440_0 → EICC440EXTIRQ: New Connection
- xps_intc_0 → Irq: EICC440EXTIRQ hinzufügen (Interrupteingang von ppc440_0)
- (xps_uartlite_0 → TX & RX : New Connextion)

3. TAB Project → Project Files → UCF File öffnen und falls nötig folgende Zeilen hinzufügen

Module RS232 Uart 1 constraints
Net fpga 0 RS232 Uart 1 RX pin LOC= AG15 ;
Net fpga 0 RS232 Uart 1 RX pin IOSTANDARD=LVCMOS33;
Net fpga 0 RS232 Uar t 1 TX pin LOC = AG20 ;
Net fpga 0 RS232 Uar t 1 TX pin IOSTANDARD=LVCMOS33;

Module Push But tons 5Bit constaints
Net fpga 0 Push But tons 5Bit GPIO IO pin<0> LOC = AJ6 ;
Net fpga 0 Push But tons 5Bit GPIO IO pin<0> IOSTANDARD=LVCMOS33;
Net fpga 0 Push But tons 5Bit GPIO IO pin<0> PULLDOWN;
Net fpga 0 Push But tons 5Bit GPIO IO pin<0> SLEW=SLOW;
Net fpga 0 Push But tons 5Bit GPIO IO pin<0> DRIVE=2;

4. Hardware → Generate Bitstream

5. Software → Launch SDK Studio

6. XilinX Tools → Generate Libraries and BSPs

7. C-Code einfügen und Project → Build All

8. Run → Run As → Run on Hardware

```
/* include header files*/

#include "xintc.h"
#include "xexception_l.h"
#include "xparameters.h"
#include "xbasic_types.h"
#include "xgpio.h"
#include "xstatus.h"
#include "xtmrctr.h"
#include "stdio.h"
#include "xintc_l.h"



/* parameter out of xparameter.h that is generated
 * throuh "XlilinX Tools --> Generate Libraries and BSPs" */

#define INTC_BASEADDR                   XPAR_INTC_0_BASEADDR
#define INTC_DEVICE_ID                  XPAR_INTC_0_DEVICE_ID
#define INTC_DEVICE_INTR_ID             XPAR_INTC_0_GPIO_1_VEC_ID


/* functions */

XStatus IntcLowLevelExample(Xuint32 IntcBaseAddress);

void SetupInterruptSystem();

void DeviceDriverHandler(void *CallbackRef);



/************************** Variable Definitions ****************************/

XGpio GpioInput;     /* The driver instance for GPIO Device configured as I/P */


static XIntc InterruptController; /* Instance of the Interrupt Controller */

/*
 * Create a shared variable to be used by the main thread of processing and
 * the interrupt processing */

volatile static Xboolean InterruptProcessed = XFALSE;


/* main function */

int main (void)
{

    XStatus Status;
```

```
53
    /* call interrupt example */
55  Status = IntcLowLevelExample(INTC_BASEADDR);
    if (Status != XST_SUCCESS)
57  {
        return XST_FAILURE;
59  }

61  return XST_SUCCESS;
}
63


65
/* function: interrupt controller example */
67
XStatus IntcLowLevelExample(Xuint32 IntcBaseAddress)
69 {
    XStatus Status;
71


73   /*
     * This step is processor specific, connect the handler for the interrupt
75   * controller to the interrupt source for the processor
     */
77   SetupInterruptSystem();

79
    /* Init Pushbuttons(Gpio) Datadirection: Inputs*/
81
    Status = XGpio_Initialize(&GpioInput, XPAR_PUSH_BUTTONS_5BIT_DEVICE_ID);
83
    XGpio_SetDataDirection(&GpioInput, 1, 0xF);
85


87  /* Global Enable Interrupts Gpio Pushbuttons (GIE Regsiter)*/
    XGpio_InterruptGlobalEnable(&GpioInput);
89
    /* Enable Interrupts for Channel 1  IER Regsiter*/
91  XGpio_InterruptEnable(&GpioInput, 0x00000001 );

93
     /*
95   *Connect a device driver handler that will be called when an interrupt
     *for the device occurs, the device driver handler performs the specific
97   *interrupt processing for the device
     */
99
     XIntc_RegisterHandler(IntcBaseAddress, INTC_DEVICE_INTR_ID,(XInterruptHandler
         )DeviceDriverHandler,(void *)0);
101



103



105
      /* Hardware Interrrupts enable (MER Register) */
107
      XIntc_Out32(IntcBaseAddress + XIN_MER_OFFSET, 0x00000003 );
109
     /* Enable Interrupt Pushbuttons in the Interruptcontroller (SIE Register)*/
111
     XIntc_Out32(IntcBaseAddress + XIN_SIE_OFFSET,
         XPAR_PUSH_BUTTONS_5BIT_IP2INTC_IRPT_MASK  );
113


115  /*
     * Wait for the interrupt to be processed, if the interrupt does not
```

```
117      * occur this loop will wait forever
         */
119
121     while (1)
        {
123
125         /*
            * If the interrupt occurred which is indicated by the global
127         * variable which is set in the device driver handler, then
            * stop waiting
129         */
            if (InterruptProcessed)
131         {
              printf("erfolg\n");
133           printf("----------\n");
              printf("----------\n");
135             InterruptProcessed=XFALSE;
            }
137     }

139
        return XST_SUCCESS;
141
    }
143
    /* function: Init PowerPC for Interrupthandling */
145
    void SetupInterruptSystem()
147 {

149     /* Initialize the PPC exception table */

151     XExc_Init();

153     /* Register the interrupt controller handler with the exception table */

155     XExc_RegisterHandler(XEXC_ID_NON_CRITICAL_INT,
                             (XExceptionHandler)XIntc_DeviceInterruptHandler,
157                          INTC_DEVICE_ID);

159     /* Enable non-critical exceptions */

161     XExc_mEnableExceptions(XEXC_NON_CRITICAL);

163 }

165
    /* Interrupt Handler: Pushbuttons */
167
    void DeviceDriverHandler(void *CallbackRef)
169 {
        /*
171      * Indic ate the interrupt has been processed using a shared variable
         */
173
        InterruptProcessed = XTRUE;
175
        /* delete Interrupt from the Pushbutton */
177     XGpio_InterruptClear(&GpioInput, 0x00000001);

179 }
```

Quelltext 1: *xintc.c*

nützliche Dokumente:

- http://www.xilinx.com/support/documentation/application_notes/xapp778.pdf
- Datasheet XPS Gpio
- Datasheet XPS Interrupt Controller