Hello world Intro\_kv\_1

**#include** "xil\_printf.h"

**#include** "xil\_io.h"

**#include** "xgpiops.h"

**#include** <stdbool.h>

**#include** "xparameters.h"

**#include** "xstatus.h"

**#include** <stdio.h>

**#define** PIN\_OFFSET 77

// Fonction pour effectuer l'action de lecture

**void** **effectuerLecture**(XGpioPs \*Gpio) {

uint8\_t val\_1 = XGpioPs\_ReadPin(Gpio, PIN\_OFFSET+1); // SW0

uint8\_t val\_2 = XGpioPs\_ReadPin(Gpio, PIN\_OFFSET+2); // SW1

xil\_printf("SW0: %d, SW1: %d\n\r", val\_1, val\_2);

}

**int** **main**() {

**int** Status;

XGpioPs Gpio; /\* The driver instance for GPIO Device. \*/

XGpioPs\_Config \*ConfigPtr;

**char** input;

bool continuer = true;

/\* Initialize the GPIO driver. \*/

ConfigPtr = XGpioPs\_LookupConfig(XPAR\_XGPIOPS\_0\_DEVICE\_ID);

Status = XGpioPs\_CfgInitialize(&Gpio, ConfigPtr, ConfigPtr->BaseAddr);

**if** (Status != XST\_SUCCESS) {

**return** XST\_FAILURE;

}

XGpioPs\_SetDirectionPin(&Gpio, PIN\_OFFSET+1, 0);

XGpioPs\_SetOutputEnablePin(&Gpio, PIN\_OFFSET+1, 0);

XGpioPs\_SetDirectionPin(&Gpio, PIN\_OFFSET+2, 0);

XGpioPs\_SetOutputEnablePin(&Gpio, PIN\_OFFSET+2, 0);

XGpioPs\_SetDirectionPin(&Gpio, PIN\_OFFSET+3, 1);

XGpioPs\_SetOutputEnablePin(&Gpio, PIN\_OFFSET+3, 1);

XGpioPs\_SetDirectionPin(&Gpio, PIN\_OFFSET+4, 1);

XGpioPs\_SetOutputEnablePin(&Gpio, PIN\_OFFSET+4, 1);

XGpioPs\_WritePin(&Gpio, PIN\_OFFSET+3, 1); // LD0

XGpioPs\_WritePin(&Gpio, PIN\_OFFSET+4, 1); // LD1

**do** {

**scanf**(" %c", &input);

**switch** (input) {

**case** 'q':

effectuerLecture(&Gpio);

**break**;

**case** 'p':

continuer = false;

**break**;

// Ajoutez d'autres cas si nécessaire

**default**:

// Autre logique selon les besoins

**break**;

}

} **while** (continuer);

**return** 0;

}

////////////////////////////////////////////////////////////////////////////////////

xgpio example  
  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\* @file xgpio\_example.c

\*

\* This file contains a design example using the AXI GPIO driver (XGpio) and

\* hardware device. It only uses channel 1 of a GPIO device and assumes that

\* the bit 0 of the GPIO is connected to the LED on the HW board.

\*

\*

\* <pre>

\* MODIFICATION HISTORY:

\*

\* Ver Who Date Changes

\* ----- ---- -------- -----------------------------------------------

\* 1.00a rmm 03/13/02 First release

\* 1.00a rpm 08/04/03 Removed second example and invalid macro calls

\* 2.00a jhl 12/15/03 Added support for dual channels

\* 2.00a sv 04/20/05 Minor changes to comply to Doxygen and coding guidelines

\* 3.00a ktn 11/20/09 Minor changes as per coding guidelines.

\* 4.1 lks 11/18/15 Updated to use canonical xparameters and

\* clean up of the comments and code for CR 900381

\* 4.3 sk 09/29/16 Modified the example to make it work when LED\_bits are

\* configured as an output. CR# 958644

\* ms 01/23/17 Added xil\_printf statement in main function to

\* ensure that "Successfully ran" and "Failed" strings

\* are available in all examples. This is a fix for

\* CR-965028.

\* 4.5 sne 06/12/19 Fixed IAR compiler warning.

\*

\* </pre>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Include Files \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**#include** "xparameters.h"

**#include** "xgpio.h"

**#include** "xil\_io.h"

**#include** "xil\_printf.h"

**#include** <stdbool.h>

**#include** "xstatus.h"

**#include** <stdio.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Constant Definitions \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**#define** LED 0x0F /\* Assumes bit 0 of GPIO is connected to an LED \*/

/\*

\* The following constants map to the XPAR parameters created in the

\* xparameters.h file. They are defined here such that a user can easily

\* change all the needed parameters in one place.

\*/

**#define** GPIO\_EXAMPLE\_DEVICE\_ID XPAR\_GPIO\_0\_DEVICE\_ID

/\*

\* The following constant is used to wait after an LED is turned on to make

\* sure that it is visible to the human eye. This constant might need to be

\* tuned for faster or slower processor speeds.

\*/

**#define** LED\_DELAY 10000000

/\*

\* The following constant is used to determine which channel of the GPIO is

\* used for the LED if there are 2 channels supported.

\*/

**#define** LED\_CHANNEL 1

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Type Definitions \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Macros (Inline Functions) Definitions \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**#ifdef** PRE\_2\_00A\_APPLICATION

/\*

\* The following macros are provided to allow an application to compile that

\* uses an older version of the driver (pre 2.00a) which did not have a channel

\* parameter. Note that the channel parameter is fixed as channel 1.

\*/

**#define** XGpio\_SetDataDirection(InstancePtr, DirectionMask) \

XGpio\_SetDataDirection(InstancePtr, LED\_CHANNEL, DirectionMask)

**#define** XGpio\_DiscreteRead(InstancePtr) \

XGpio\_DiscreteRead(InstancePtr, LED\_CHANNEL)

**#define** XGpio\_DiscreteWrite(InstancePtr, Mask) \

XGpio\_DiscreteWrite(InstancePtr, LED\_CHANNEL, Mask)

**#define** XGpio\_DiscreteSet(InstancePtr, Mask) \

XGpio\_DiscreteSet(InstancePtr, LED\_CHANNEL, Mask)

**#endif**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Function Prototypes \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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

/\*

\* The following are declared globally so they are zeroed and so they are

\* easily accessible from a debugger

\*/

XGpio Gpio; /\* The Instance of the GPIO Driver \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\*

\* The purpose of this function is to illustrate how to use the GPIO

\* driver to turn on and off an LED.

\*

\*

\* @return XST\_FAILURE to indicate that the GPIO Initialization had

\* failed.

\*

\* @note This function will not return if the test is running.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **printBinary**(u32 num) {

**for** (**int** i = 31; i >= 0; i--) {

**putchar**((num & (1u << i)) ? '1' : '0');

}

**putchar**('\n');

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**int** **main**(**void**)

{

**int** Status;

**volatile** **int** Delay;

bool continuer = true;

**char** input;

u32 gpioData;

/\* Initialize the GPIO driver \*/

Status = XGpio\_Initialize(&Gpio, GPIO\_EXAMPLE\_DEVICE\_ID);

xil\_printf("initiate begin");

**if** (Status != XST\_SUCCESS) {

xil\_printf("Gpio Initialization Failed\r\n");

**return** XST\_FAILURE;

}

xil\_printf("\n\r initiate ended");

/\* Set the direction for all signals as inputs except the LED output \*/

XGpio\_SetDataDirection(&Gpio, LED\_CHANNEL, 0xFFFFFFFF);

xil\_printf("\n\r setdirection ended\n");

/\* Loop forever blinking the LED \*/

**do** {

**scanf**("%c", &input);

**getchar**();

**switch** (input) {

**case** 'r':

xil\_printf("r \n");

gpioData = XGpio\_DiscreteRead(&Gpio, LED\_CHANNEL);

printBinary(gpioData);

**break**;

**case** 'w':

XGpio\_SetDataDirection(&Gpio, LED\_CHANNEL, 0x00000000);

/\* Set the LED to High \*/

XGpio\_DiscreteWrite(&Gpio, LED\_CHANNEL, LED);

/\* Wait a small amount of time so the LED is visible \*/

**for** (Delay = 0; Delay < LED\_DELAY; Delay++);

/\* Clear the LED bit \*/

XGpio\_DiscreteClear(&Gpio, LED\_CHANNEL, LED);

/\* Wait a small amount of time so the LED is visible \*/

**for** (Delay = 0; Delay < LED\_DELAY; Delay++);

**break**;

**case** 'p':

continuer = false;

**break**;

}

} **while** (continuer);

}