Aarhus Universitet

ERTS - Group 2

Rapport

Assignment 2

Gruppemedlemmer:	**Studienumre:**
Daniel Tøttrup	201509520
Mathias Lønborg Friis	201505665
Stinus Lykke Skovgaard	201401682

1. oktober 2019

Kapitel 1

EXERCISE 3

Write a C-application that implements a command language interpreter, controlled via the USB-UART interface. The following commands must be implemented:

- **1 - Sets the binary value from 0-15 on the red led's by reading switch input (SW0-SW3)**
- **2 - Counts binary the red led's using a timer of 1 sec.**

1.1 Command 1

The essence of the first command is to set a binary value from 0-15 by using the four switches, which will be represented as a binary value by the four LED's on the board.

The code snippet below shows how we read input from the terminal, and execute the a command based on that input.

```
1  //read input from the terminal in byte
2  int userInput = inbyte();
3  xil_printf("Received: %d\r\n", userInput);
4
5  switch(userInput)
6  {
7  case (int)'1':
8  xil_printf("Received command 1\r\n");
9  execute_command_1();
10 break;
11 case (int)'2':
12 xil_printf("Received command 2\r\n");
13 execute_command_2();
14 break;
15 case (int)'3':
16 xil_printf("Received command 3\r\n");
17 execute_command_3();
18 break;
19 case (int)'4':
20 xil_printf("Received command 4\r\n");
21 execute_command_4();
```

```
22 break;
23 default:
24 xil_printf("Received unknown command.");
25 break;
26 }
```

Below is a code snippet of the implemented first command.
When the function **execute_command_1()** is called the switches on the board is read with the function **XGpio_DiscreteRead(&dip, 1)**. Where &dip is the InstancePtr that points to the XGpio instance that is being worked on, and the '1' is the channel.

The **writeValueToLEDs(int val)** takes the value returned from the **XGpio_DiscreteRead()** and writes the value to the LED registers.

```
1  void execute_command_1()
2  {
3  xil_printf("Executing command 1\r\n");
4  while(1)
5  {
6  dip_check = XGpio_DiscreteRead(&dip, 1);
7
8  writeValueToLEDs(dip_check);
9
10 }
11 }
12
13 void writeValueToLEDs(int val)
14 {
15 LED_IP_mWriteReg(XPAR_LED_IP_S_AXI_BASEADDR, 0, val);
16 }
```

Command 1 work as expected.

1.2 Command 2

Command 2 should, as states in the problem description, count the binary red LED's using the timer of 1 sec.

First the timer needs to be initialized. Below is a code snippet of how the timer is implemented. Comments in the code explains the implementation.

```
1  #define ONE_SECOND 325000000 // half of the CPU clock speed
2
3  // PS Timer related definitions
4  XScuTimer_Config *ConfigPtr;
```

```
5  XScuTimer *TimerInstancePtr = &Timer;
6
7  int main (void)
8  {
9  //initialize timer
10 ConfigPtr = XScuTimer_LookupConfig (XPAR_PS7_SCUTIMER_0_DEVICE_ID);
11 s32 Status = XScuTimer_CfgInitialize (TimerInstancePtr, ConfigPtr,
       ConfigPtr->BaseAddr);
12 if (Status != XST_SUCCESS){
13 xil_printf("Timer init() failed\r\n");
14 return XST_FAILURE;
15 }
16
17 // Load timer with delay in multiple of ONE_SECOND
18 XScuTimer_LoadTimer(TimerInstancePtr, ONE_SECOND);
19 // Set AutoLoad mode
20 XScuTimer_EnableAutoReload(TimerInstancePtr);
21 }
```

The function **execute_command_2()** sets op a counter and writes the value of counter to the LED's with the function **writeValueToLEDs(counter)** which was explained above. Then it starts the timer, and waits for the timer to expire, which will take one second. When the timer has expired the timer is cleared and the counter is incremented, before the loop starts again. Below is a code snippet of command 2.

```
1  void execute_command_2()
2  {
3  xil_printf("Executing command 2\r\n");
4  int counter = 0;
5  while(1)
6  {
7  xil_printf("Counter: %d\r\n", counter);
8  writeValueToLEDs(counter);
9
10 // Start the timer
11 XScuTimer_Start(TimerInstancePtr);
12
13 // Wait until timer expires
14 while(!XScuTimer_IsExpired(TimerInstancePtr));
15
16 // clear status bit
17 XScuTimer_ClearInterruptStatus(TimerInstancePtr);
18
19 counter++;
20
21 // Timer auto-enables
22
23 }
```

The function **execute_command_2()** works as expected.

Kapitel 2

EXERCISE 4

Kapitel 3

EXERCISE 5

Kapitel 4

EXERCISE 7