

Module: R5: RV-fpga**Section: C Programming Task: Binary Down Counter****Assessment 2.2****Binary Down Counter**

➤ Binary Down Counter:

- **Description:** This program is designed to display a binary down counter on LEDs, starting from the value provided by the switches. The program reads the initial value from the switches, initializes the counter, and then decrements the counter value, displaying it on the LEDs until the counter reaches zero.
- **Code Explanation:**
 1. **Main Function:**

The main function is the entry point of the program. **en** is initialized to **0xFFFF**, enabling all **GPIO** input/output. **inp_val** and **counter** are declared for later use. The **write(GPIO_inout, en)** function enables GPIO input/output.
 2. **UART Initialization:**

The UART is initialized for serial communication using the **uartInit()** function.
 3. **Infinite Loop:**

The program runs an infinite loop to continuously read the switch values and update the counter. The **inp_val** variable reads the value from the switches using the **read_switch(GPIO_sw)** function. The value is shifted 16 bits to the right to get the initial counter value. The counter is decremented in each iteration. The current counter value is written to the LEDs using the **write(GPIO_led, counter)** function.

The counter value is printed using **printfNexys**.

Also a delay is introduced using the **DELAY(10000)** function to control the speed of the counter.

The program continuously reads the switch value in a nested loop. If a non-zero value is detected, the loop breaks, and the process starts again.

4. Conclusion:

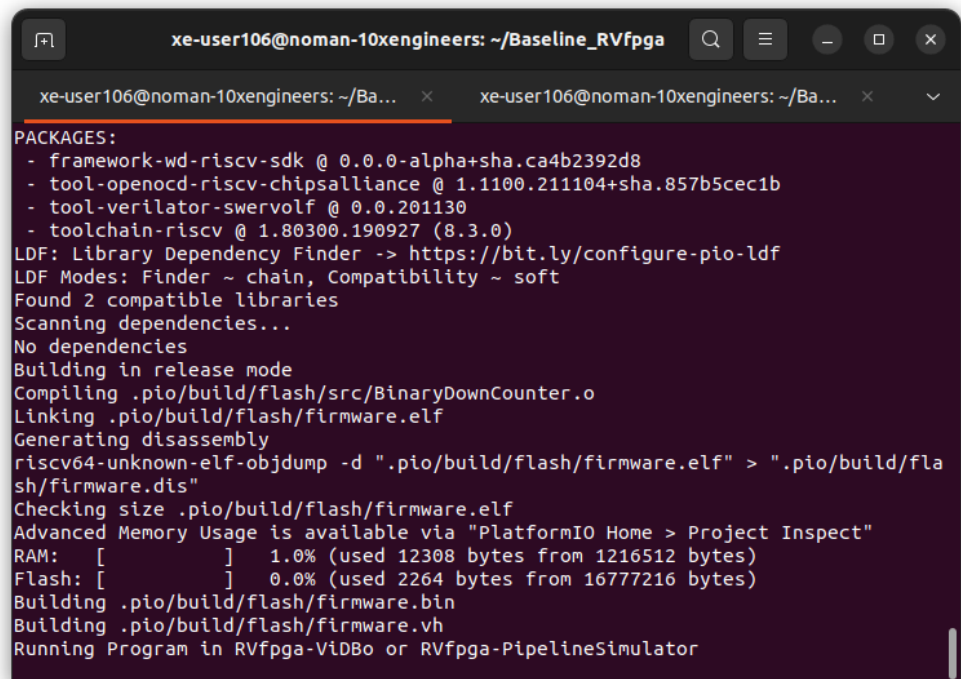
This program effectively demonstrates a binary down counter using switches and LEDs. It reads the initial value from the switches, initializes the counter, and decrements the counter value, displaying it on the LEDs until the counter reaches zero. The use of UART for printing the counter value and the delay function helps in controlling the counter speed and providing real-time responses.

5. Terminal Output:

Use the following command in source folder.

```
make flash
```

```
make watch
```



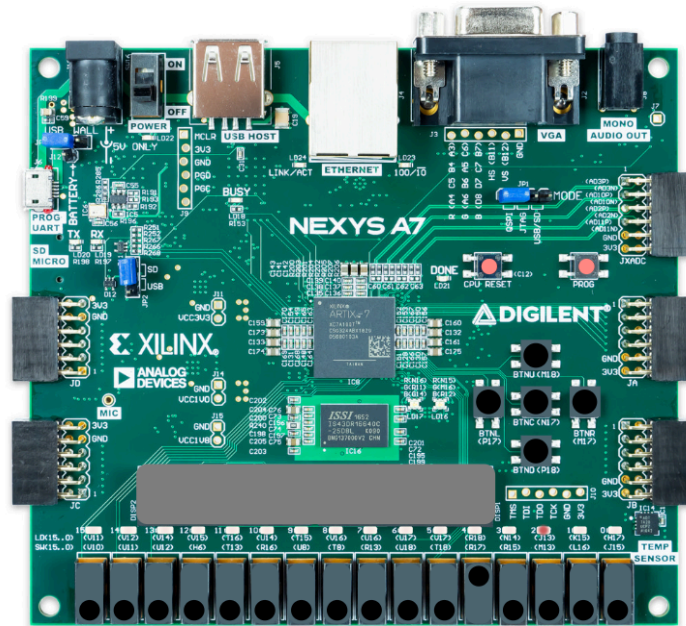
```
xe-user106@noman-10xengineers: ~/Baseline_RVfpga
xe-user106@noman-10xengineers: ~/Ba... x xe-user106@noman-10xengineers: ~/Ba... x
PACKAGES:
- framework-wd-riscv-sdk @ 0.0.0-alpha+sha.ca4b2392d8
- tool-openocd-riscv-chipsalliance @ 1.1100.211104+sha.857b5cec1b
- tool-verilator-swervolf @ 0.0.201130
- toolchain-riscv @ 1.80300.190927 (8.3.0)
LDF: Library Dependency Finder -> https://bit.ly/configure-pio-ldf
LDF Modes: Finder ~ chain, Compatibility ~ soft
Found 2 compatible libraries
Scanning dependencies...
No dependencies
Building in release mode
Compiling .pio/build/flash/src/BinaryDownCounter.o
Linking .pio/build/flash/firmware.elf
Generating disassembly
riscv64-unknown-elf-objdump -d ".pio/build/flash/firmware.elf" > ".pio/build/flash/firmware.dis"
Checking size .pio/build/flash/firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM: [          ] 1.0% (used 12308 bytes from 1216512 bytes)
Flash: [          ] 0.0% (used 2264 bytes from 16777216 bytes)
Building .pio/build/flash/firmware.bin
Building .pio/build/flash/firmware.vh
Running Program in RVfpga-ViDBo or RVfpga-PipelineSimulator
```

Visit this link to visualize Virtual Board:

<http://localhost:8000/nexys-a7.html>

7 SEGMENT DISPLAYS:

0 0 0 0 0 0 0 0



6. UART Output:

```

0
Counter=
15
Counter=
14
Counter=
13
Counter=
12
Counter=
11
Counter=
10
Counter=
9
Counter=
8
Counter=
7
Counter=
6
Counter=
5
Counter=
4
Counter=
3
Counter=
2
Counter=

```