Please read the legal disclaimer at the end of this document.

Revision 1.0
# Table of Contents

Table of Figures ........................................................................................................... 4

Chapter 1 - CYC1000 IoT / Maker Board ................................................................. 5
  1.1 About Arrow CYC1000 Board ........................................................................... 5
  1.2 Useful Links ....................................................................................................... 5
  1.3 Getting Help ....................................................................................................... 6

Chapter 2 - Introduction to the CYC1000 Board ..................................................... 7
  2.1 Layout and Components ................................................................................... 7
  2.2 Block Diagram .................................................................................................. 8

Chapter 3 - Connections and Peripherals of the CYC1000 Board ......................... 10
  3.1 Board Status Elements .................................................................................... 10
  3.2 Clock Circuitry .................................................................................................. 10
  3.3 Peripherals Connected to the FPGA ................................................................. 11
    3.3.1 LEDs ........................................................................................................... 11
    3.3.2 Push Buttons ............................................................................................... 12
    3.3.3 Accelerometer ............................................................................................. 12
    3.3.4 SDRAM Memory ........................................................................................ 13
    3.3.5 Serial Configuration Flash Memory ............................................................. 14
    3.3.6 Arduino MKR Connectors ........................................................................... 15
    3.3.7 PMOD Connector ........................................................................................ 16
    3.3.8 User I/O ....................................................................................................... 17
    3.3.9 Communication and Configuration ............................................................. 17
    3.3.10 Power Tree ................................................................................................ 19

Chapter 4 - Software and Driver Installation ............................................................. 20
  4.1 Installing Quartus Prime Software .................................................................... 20
  4.2 Installing Arrow USB Programmer .................................................................... 21

Chapter 5 - New Project with CYC1000 ................................................................. 24
  5.1 Creating a new Blinky Project with CYC1000 .................................................. 24
  5.2 Building a Blinky Project with CYC1000 .......................................................... 28
    5.2.1 Block Diagram ............................................................................................ 29
    5.2.2 Components of the Design ......................................................................... 29
    5.2.3 Catalog IP .................................................................................................... 30
    5.2.4 Create a PLL ................................................................................................. 30
    5.2.5 Create and Configure the PLL .................................................................... 31
5.2.6 Create and Configure the Counter ................................................................. 34
5.2.7 Create and Configure the Multiplexer .......................................................... 36
5.2.8 Adding the Components to the Schematic .................................................... 38
5.2.9 Connecting the Components ....................................................................... 40
5.2.10 Add inputs, outputs to the schematic .......................................................... 43
5.2.11 Analysis and Synthesis ............................................................................... 46
5.2.12 Adding Timing Constraints ......................................................................... 46
5.2.13 Pinning Assignments .................................................................................. 48
5.2.14 Compiling the Design ................................................................................. 51
5.2.15 Reading the Compilation Report .................................................................. 52

Chapter 6 - Configuring the CYC1000 .................................................................... 54
  6.1 Configure the FPGA in JTAG mode ................................................................. 54
  6.2 Serial configuration flash memory programming ............................................ 57
      6.2.1 Programming File generation ................................................................. 58
      6.2.2 Device Programming ............................................................................. 60
  6.3 Testing the Design ......................................................................................... 62

Chapter 7 - Common Issues and Fixes .................................................................... 63

Chapter 8 - Appendix ............................................................................................. 64
  8.1 Revision History ............................................................................................. 64
  8.2 Legal Disclaimer ............................................................................................. 65
Table of Figures

Figure 1 - CYC1000 Board (top view) ................................................................. 7
Figure 2 - CYC1000 Block Diagram .................................................................. 8
Figure 3 – Position of Indication LEDs .............................................................. 10
Figure 4 – CYC1000 Clock Tree ........................................................................ 10
Figure 5 – LED Connections ............................................................................ 11
Figure 6 – Button Connections ........................................................................ 12
Figure 7 – Accelerometer Connections .............................................................. 12
Figure 8 – SDRAM Connections ...................................................................... 13
Figure 9 – Flash Connections .......................................................................... 14
Figure 10 - Arduino MKR Header Connections ............................................... 15
Figure 11 – PMOD Header Connections ............................................................. 16
Figure 12 - User I/O Connections ..................................................................... 17
Figure 13 – UART Connections ......................................................................... 17
Figure 14 – JTAG Connections ......................................................................... 18
Figure 15 – Power Tree Connections ................................................................ 19
Chapter 1 - CYC1000 IoT / Maker Board

1.1 About Arrow CYC1000 Board

The CYC1000 is a customizable IoT / Maker Board ready for evaluation, development and/or use in a product. It is built around the Intel Cyclone 10 LP FPGA, which is optimized for low-cost and low-power, making them ideal for high-volume and cost-sensitive applications. High density sea of programmable gates and on-board resources allow implementation of Nios II 32-bit microcontroller IP, which provides the ideal solution for I/O expansion, chip-to-chip interfacing, industrial, automotive and consumer applications.

The CYC1000 is equipped with an Arrow USB Programmer2, SDRAM, flash memory, accelerometer sensor and PMOD/ADRUINO MKR connectors making it a fully featured plug and play solution without any additional costs.

The CYC1000 board contains all the tools needed to use the board in conjunction with a computer that runs a 64-bit Linux / Microsoft Windows 7 operating system or later.

1.2 Useful Links

A set of useful links that can be used to get relevant information about the CYC1000 or the Cyclone 10 LP FPGA.

- CYC1000 at Arrow Shop
- CYC1000 at Trenz Electronic Shop
- Intel Cyclone 10 LP Webpage
- CYC1000 Wiki Page
1.3 Getting Help

Here are the addresses where you can get help if you encounter any problems:

- **Arrow Electronics**
  
  **In Person**
  Arrow EMEA
  + 49 (0) 6102 5030 0

  **Online**
  [https://arrow.com](https://arrow.com)

- **Trenz Electronic GmbH**

Chapter 2 - Introduction to the CYC1000 Board

2.1 Layout and Components

Figure 1 shows a top view of the board. It depicts the layout of the board and indicates the location of the various connectors and key components.

The following are available on the CYC1000 board:

- Intel Cyclone 10 LP 10CL025YU256C8G device
- Arrow USB Programmer2 on-board for programming; JTAG Mode
- 64MBit SDRAM 166MHz
- 16Mbit serial configuration flash memory
- 12MHz MEMS Oscillator
- One optional MEMS Oscillator of preferred frequency
- 8x red user LEDs
- 2x board indication LEDs
- 2x user push buttons
- 3-axis accelerometer
- 12-pin PMOD header
- Arduino MKR header
- User JTAG header
- User I/O header
2.2 Block Diagram

Figure 2 represents the block diagram of the board. All the connections are established through the Cyclone 10 LP FPGA device to provide maximum flexibility for users. Users can configure the FPGA to implement any system design.

![Figure 2 - CYC1000 Block Diagram](image)

**FPGA Device**
- Available Cyclone 10 LP Devices for the CYC1000

<table>
<thead>
<tr>
<th>Resources</th>
<th>10CL006</th>
<th>10CL010</th>
<th>10CL016</th>
<th>10CL025</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic Elements (LE)</td>
<td>6,272</td>
<td>10,320</td>
<td>15,408</td>
<td>24,624</td>
</tr>
<tr>
<td>M9K Memory (Kb)</td>
<td>270</td>
<td>414</td>
<td>504</td>
<td>594</td>
</tr>
<tr>
<td>18 x 18 Multiplier</td>
<td>15</td>
<td>23</td>
<td>56</td>
<td>66</td>
</tr>
<tr>
<td>PLLs</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

**Configuration and Debug**
- On-board Arrow USB Programmer2 (mini-USB type B connector)

**Memory Devices**
- 4MBit to 32MBit external flash memory
- 64MBit to 256MBit external SDRAM memory
Headers
- PMOD Header
- Arduino MKR Header
- User JTAG Header
- User I/O Header

Buttons and Indicators
- 2x side-buttons
- 8x red user LEDs
- 2x board indication LEDs

Sensors
- One 3-axis accelerometer

Power
- Recommended external supply voltage range: +5.0 V (nominal)
- Recommended I/O signal voltage range: 0 to +3.3 V
Chapter 3 - Connections and Peripherals of the CYC1000 Board

3.1 Board Status Elements

In addition to the 8 LEDs that the FPGA can control, there are 2 additional LEDs which can indicate the status of the board.

![Power Indicator LED and CONF DONE LED](image)

<table>
<thead>
<tr>
<th>Board Reference</th>
<th>LED Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>3.3V</td>
<td>On when 3.3V power is active</td>
</tr>
<tr>
<td>D10</td>
<td>CONF_DONE</td>
<td>On when configuration data was loaded to Cyclone 10 LP device</td>
</tr>
<tr>
<td></td>
<td></td>
<td>without error</td>
</tr>
</tbody>
</table>

3.2 Clock Circuitry

All the external clocks of the system can be seen in Figure 4. The default clock (CLK12M) is at 12MHz and is connected and driving the FPGA’s user logic and the Arrow USB Programmer. There is an optional slot of another clock (CLK_X) to add another preferred clock source to the FPGA. Both clocks are driving the internal PLLs.

For more information on clocks and PLLs of the Cyclone 10 LP, please refer to this [document](#).
3.3  Peripherals Connected to the FPGA

3.3.1  LEDs

There are eight red user-controllable LEDs connected to the FPGA. Each LED is driven directly and individually by the Cyclone 10 LP FPGA; driving its associated pin to a high logic level for on or low logic level for off.

<table>
<thead>
<tr>
<th>Board Reference</th>
<th>FPGA Pin No.</th>
<th>Description</th>
<th>I/O Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK12M</td>
<td>PIN_M2</td>
<td>12MHz clock input</td>
<td>3.3 V</td>
</tr>
<tr>
<td>CLK_X</td>
<td>PIN_E15</td>
<td>Optional clock input</td>
<td>3.3 V</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Board Reference</th>
<th>FPGA Pin No.</th>
<th>I/O Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED1</td>
<td>PIN_M6</td>
<td>3.3 V</td>
</tr>
<tr>
<td>LED2</td>
<td>PIN_T4</td>
<td>3.3 V</td>
</tr>
<tr>
<td>LED3</td>
<td>PIN_T3</td>
<td>3.3 V</td>
</tr>
<tr>
<td>LED4</td>
<td>PIN_R3</td>
<td>3.3 V</td>
</tr>
<tr>
<td>LED5</td>
<td>PIN_T2</td>
<td>3.3 V</td>
</tr>
<tr>
<td>LED6</td>
<td>PIN_R4</td>
<td>3.3 V</td>
</tr>
<tr>
<td>LED7</td>
<td>PIN_N5</td>
<td>3.3 V</td>
</tr>
<tr>
<td>LED8</td>
<td>PIN_N3</td>
<td>3.3 V</td>
</tr>
</tbody>
</table>
3.3.2 Push Buttons

The board has two push buttons connected to the FPGA. Push buttons drive their associated pins low logic level when pressed and high logic level when released.

![Figure 6 – Button Connections](image)

<table>
<thead>
<tr>
<th>Board Reference</th>
<th>FPGA Pin No.</th>
<th>Description</th>
<th>I/O Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESET</td>
<td>PIN_H5</td>
<td>nCONFIG</td>
<td>3.3 V</td>
</tr>
<tr>
<td>USER_BTN</td>
<td>PIN_N6</td>
<td>User button</td>
<td>3.3 V</td>
</tr>
</tbody>
</table>

3.3.3 Accelerometer

The board comes with a digital accelerometer (LIS3DH), commonly known as the G-Sensor. This G-Sensor is a small, thin, ultra-low power consumption, 3-axis accelerometer with digital I2C/SPI serial interface, standard output. The LIS3DH has user-selectable full scales of +/-2g, +/-4g, +/-8g, +/-16g and it is capable of measuring accelerations with output data rates from 1 Hz to 5 kHz. The supplied power to the board (coming either from micro-USB connection or user VIN) can be monitored through the ADC channel 3 of the accelerometer.

![Figure 7 – Accelerometer Connections](image)
<table>
<thead>
<tr>
<th>Board Reference</th>
<th>FPGA Pin No.</th>
<th>Description</th>
<th>I/O Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEN_INT1</td>
<td>PIN_B1</td>
<td>Interrupt 1</td>
<td>3.3 V</td>
</tr>
<tr>
<td>SEN_INT2</td>
<td>PIN_C2</td>
<td>Interrupt 2</td>
<td>3.3 V</td>
</tr>
<tr>
<td>SEN_SDI</td>
<td>PIN_G2</td>
<td>Data In (MOSI)*</td>
<td>3.3 V</td>
</tr>
<tr>
<td>SEN_SDO</td>
<td>PIN_G1</td>
<td>Data Out (MISO)*</td>
<td>3.3 V</td>
</tr>
<tr>
<td>SEN_SPC</td>
<td>PIN_F3</td>
<td>Clock*</td>
<td>3.3 V</td>
</tr>
<tr>
<td>SEN_CS</td>
<td>PIN_D1</td>
<td>Chip Select*</td>
<td>3.3 V</td>
</tr>
</tbody>
</table>

*For SPI connection

3.3.4 SDRAM Memory

The CYC1000 board supports 64MBit (default version) or up to 256MBit (customized version) SDRAM which can operate up to 166 MHz clock frequency. Below are the connections and pinning of the SDRAM used in the CYC1000.

![Figure 8 – SDRAM Connections](image)

<table>
<thead>
<tr>
<th>Board Reference</th>
<th>FPGA Pin No.</th>
<th>Description</th>
<th>I/O Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>PIN_A3</td>
<td>SDRAM Address [0]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>A1</td>
<td>PIN_B5</td>
<td>SDRAM Address [1]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>A2</td>
<td>PIN_B4</td>
<td>SDRAM Address [2]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>A3</td>
<td>PIN_B3</td>
<td>SDRAM Address [3]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>A4</td>
<td>PIN_C3</td>
<td>SDRAM Address [4]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>A5</td>
<td>PIN_D3</td>
<td>SDRAM Address [5]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>A6</td>
<td>PIN_E6</td>
<td>SDRAM Address [6]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>A7</td>
<td>PIN_E7</td>
<td>SDRAM Address [7]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>A8</td>
<td>PIN_D6</td>
<td>SDRAM Address [8]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>A9</td>
<td>PIN_D8</td>
<td>SDRAM Address [9]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>A10</td>
<td>PIN_A5</td>
<td>SDRAM Address [10]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>A11</td>
<td>PIN_E8</td>
<td>SDRAM Address [11]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>A12</td>
<td>PIN_A2</td>
<td>SDRAM Address [12]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>A13</td>
<td>PIN_C6</td>
<td>SDRAM Address [13]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>BA0</td>
<td>PIN_A4</td>
<td>SDRAM Bank Address [0]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>BA1</td>
<td>PIN_B6</td>
<td>SDRAM Bank Address [1]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>RAS</td>
<td>PIN_B7</td>
<td>SDRAM Row Address Strobe</td>
<td>3.3 V</td>
</tr>
<tr>
<td>CAS</td>
<td>PIN_C8</td>
<td>SDRAM Column Address Strobe</td>
<td>3.3 V</td>
</tr>
<tr>
<td>WE</td>
<td>PIN_A7</td>
<td>SDRAM Write Enable</td>
<td>3.3 V</td>
</tr>
<tr>
<td>CS</td>
<td>PIN_A6</td>
<td>SDRAM Chip Select</td>
<td>3.3 V</td>
</tr>
</tbody>
</table>
## 3.3.5 Serial Configuration Flash Memory

The CYC1000 board supports up to 32MBit of serial flash memory that can be used for user data and programming non-volatile storage. The configuration bit stream is downloaded into the serial configuration device which automatically loads the configuration data into the Cyclone 10 LP when the board is powered on. Device memory capacity not consumed storing configuration data can be used as general-purpose non-volatile memory, which is perfect for program and data storage. Several interface peripherals available with Nios II embedded processors allow you to access the serial configuration device as a memory module connected to your embedded system.

<table>
<thead>
<tr>
<th>Board Reference</th>
<th>FPGA Pin No.</th>
<th>Description</th>
<th>I/O Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>PIN_B14</td>
<td>SDRAM Input Clock</td>
<td>3.3 V</td>
</tr>
<tr>
<td>CKE</td>
<td>PIN_F8</td>
<td>SDRAM Clock Enable</td>
<td>3.3 V</td>
</tr>
<tr>
<td>DQ0</td>
<td>PIN_B10</td>
<td>SDRAM Data[0]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>DQ1</td>
<td>PIN_A10</td>
<td>SDRAM Data[1]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>DQ2</td>
<td>PIN_B11</td>
<td>SDRAM Data[2]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>DQ3</td>
<td>PIN_A11</td>
<td>SDRAM Data[3]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>DQ4</td>
<td>PIN_A12</td>
<td>SDRAM Data[4]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>DQ5</td>
<td>PIN_D9</td>
<td>SDRAM Data[5]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>DQ6</td>
<td>PIN_B12</td>
<td>SDRAM Data[6]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>DQ7</td>
<td>PIN_C9</td>
<td>SDRAM Data[7]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>DQ8</td>
<td>PIN_D11</td>
<td>SDRAM Data[8]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>DQ9</td>
<td>PIN_E11</td>
<td>SDRAM Data[9]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>DQ10</td>
<td>PIN_A15</td>
<td>SDRAM Data[10]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>DQ11</td>
<td>PIN_E9</td>
<td>SDRAM Data[11]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>DQ12</td>
<td>PIN_D14</td>
<td>SDRAM Data[12]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>DQ13</td>
<td>PIN_F9</td>
<td>SDRAM Data[13]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>DQ14</td>
<td>PIN_C14</td>
<td>SDRAM Data[14]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>DQ15</td>
<td>PIN_A14</td>
<td>SDRAM Data[15]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>DQM0</td>
<td>PIN_B13</td>
<td>SDRAM Lower Data Mask</td>
<td>3.3 V</td>
</tr>
<tr>
<td>DQM1</td>
<td>PIN_D12</td>
<td>SDRAM Upper Data Mask</td>
<td>3.3 V</td>
</tr>
</tbody>
</table>

---

**Figure 9 – Flash Connections**
### Arduino MKR Connectors

The CYC1000 board offers connectivity to Arduino MKR compatible shields that could also alternatively be used as GPIOs. The MKR connectors offer up to 23 digital I/Os.

<table>
<thead>
<tr>
<th>Board Reference</th>
<th>FPGA Pin No.</th>
<th>Description</th>
<th>I/O Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>AREF</td>
<td>PIN_P11</td>
<td>J1/1</td>
<td>Input reference voltage or GPIO</td>
</tr>
<tr>
<td>AIN0</td>
<td>PIN_R12</td>
<td>J1/2</td>
<td>GPIO [0]</td>
</tr>
<tr>
<td>AIN1</td>
<td>PIN_T13</td>
<td>J1/3</td>
<td>GPIO [1]</td>
</tr>
<tr>
<td>AIN2</td>
<td>PIN_R13</td>
<td>J1/4</td>
<td>GPIO [2]</td>
</tr>
<tr>
<td>AIN3</td>
<td>PIN_T14</td>
<td>J1/5</td>
<td>GPIO [3]</td>
</tr>
<tr>
<td>AIN4</td>
<td>PIN_P14</td>
<td>J1/6</td>
<td>GPIO [4]</td>
</tr>
<tr>
<td>AIN5</td>
<td>PIN_R14</td>
<td>J1/7</td>
<td>GPIO [5]</td>
</tr>
<tr>
<td>AIN6</td>
<td>PIN_T15</td>
<td>J1/8</td>
<td>GPIO [6]</td>
</tr>
<tr>
<td>D0</td>
<td>PIN_N16</td>
<td>J1/9</td>
<td>Digital In [0]</td>
</tr>
<tr>
<td>D1</td>
<td>PIN_L15</td>
<td>J1/10</td>
<td>Digital In [1]</td>
</tr>
<tr>
<td>D2</td>
<td>PIN_L16</td>
<td>J1/11</td>
<td>Digital In [2]</td>
</tr>
<tr>
<td>D3</td>
<td>PIN_K15</td>
<td>J1/12</td>
<td>Digital In [3]</td>
</tr>
<tr>
<td>D4</td>
<td>PIN_K16</td>
<td>J1/13</td>
<td>Digital In [4]</td>
</tr>
<tr>
<td>D5</td>
<td>PIN_J14</td>
<td>J1/14</td>
<td>Digital In [5]</td>
</tr>
<tr>
<td>D6</td>
<td>PIN_N2</td>
<td>J2/1</td>
<td>Digital In [6]</td>
</tr>
<tr>
<td>D7</td>
<td>PIN_N1</td>
<td>J2/2</td>
<td>Digital In [7]</td>
</tr>
</tbody>
</table>

**Figure 10 - Arduino MKR Header Connections**
<table>
<thead>
<tr>
<th>Board Reference</th>
<th>FPGA Pin No.</th>
<th>MKR Header</th>
<th>Description</th>
<th>I/O Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>D8</td>
<td>PIN_P2</td>
<td>J2/3</td>
<td>Digital In [8]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>D9</td>
<td>PIN_J1</td>
<td>J2/4</td>
<td>Digital In [9]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>D10</td>
<td>PIN_J2</td>
<td>J2/5</td>
<td>Digital In [10]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>D11</td>
<td>PIN_K2</td>
<td>J2/6</td>
<td>Digital In [11]*</td>
<td>3.3 V</td>
</tr>
<tr>
<td>D12</td>
<td>PIN_L2</td>
<td>J2/7</td>
<td>Digital In [12]*</td>
<td>3.3 V</td>
</tr>
<tr>
<td>D13</td>
<td>PIN_P1</td>
<td>J2/8</td>
<td>Digital In [13]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>D14</td>
<td>PIN_R1</td>
<td>J2/9</td>
<td>Digital In [14]</td>
<td>3.3 V</td>
</tr>
<tr>
<td>D11_R</td>
<td>PIN_K1</td>
<td>J2/6</td>
<td>Digital In [11] with resistor*</td>
<td>3.3 V</td>
</tr>
<tr>
<td>D12_R</td>
<td>PIN_L1</td>
<td>J2/7</td>
<td>Digital In [12] with resistor*</td>
<td>3.3 V</td>
</tr>
<tr>
<td>RESET</td>
<td>PIN_H5</td>
<td>J2/10</td>
<td>System reset of the board</td>
<td>3.3 V</td>
</tr>
<tr>
<td>GND</td>
<td>N/A</td>
<td>J2/11</td>
<td>Ground output to the connector</td>
<td>N/A</td>
</tr>
<tr>
<td>3.3V</td>
<td>N/A</td>
<td>J2/12</td>
<td>3.3V power to the connector</td>
<td>N/A</td>
</tr>
<tr>
<td>VIN</td>
<td>N/A</td>
<td>J2/13</td>
<td>User power into to the CYC1000</td>
<td>N/A</td>
</tr>
<tr>
<td>5V</td>
<td>N/A</td>
<td>J2/14</td>
<td>5V power to the connector</td>
<td>N/A</td>
</tr>
</tbody>
</table>

*Can only choose one, hence same name pinning

3.3.7 PMOD Connector

The CYC1000 board offers connectivity to PMOD compatible connectors (2x6-pin or 1x12-pin), making it possible to add a big variety of sensors or ICs to the system. Below is the connection schematic and pinning information.

![PMOD Header Connections](image)

**Figure 11 – PMOD Header Connections**

<table>
<thead>
<tr>
<th>Board Reference</th>
<th>FPGA Pin No.</th>
<th>Description</th>
<th>I/O Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIO_01</td>
<td>PIN_F13</td>
<td>PMOD Pin 1</td>
<td>3.3 V</td>
</tr>
<tr>
<td>PIO_02</td>
<td>PIN_F15</td>
<td>PMOD Pin 2</td>
<td>3.3 V</td>
</tr>
<tr>
<td>PIO_03</td>
<td>PIN_F16</td>
<td>PMOD Pin 3</td>
<td>3.3 V</td>
</tr>
<tr>
<td>PIO_04</td>
<td>PIN_D16</td>
<td>PMOD Pin 4</td>
<td>3.3 V</td>
</tr>
<tr>
<td>PIO_05</td>
<td>PIN_D15</td>
<td>PMOD Pin 5</td>
<td>3.3 V</td>
</tr>
<tr>
<td>PIO_06</td>
<td>PIN_C15</td>
<td>PMOD Pin 6</td>
<td>3.3 V</td>
</tr>
<tr>
<td>PIO_07</td>
<td>PIN_B16</td>
<td>PMOD Pin 7</td>
<td>3.3 V</td>
</tr>
<tr>
<td>PIO_08</td>
<td>PIN_C16</td>
<td>PMOD Pin 8</td>
<td>3.3 V</td>
</tr>
<tr>
<td>GND</td>
<td>N/A</td>
<td>Ground</td>
<td>N/A</td>
</tr>
<tr>
<td>3.3V</td>
<td>N/A</td>
<td>3.3 V Power to PMOD</td>
<td>N/A</td>
</tr>
</tbody>
</table>
3.3.8 User I/O

The CYC1000 board has two additional pins that can be connected to the board and take GPIOs.

![User I/O Connections](image)

Figure 12 - User I/O Connections

<table>
<thead>
<tr>
<th>Board Reference</th>
<th>FPGA Pin No.</th>
<th>Description</th>
<th>I/O Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIN</td>
<td>PIN_T12</td>
<td>GPIO</td>
<td>3.3 V</td>
</tr>
<tr>
<td>AIN7</td>
<td>PIN_R11</td>
<td>GPIO</td>
<td>3.3 V</td>
</tr>
<tr>
<td>GND</td>
<td>N/A</td>
<td>Ground</td>
<td>N/A</td>
</tr>
</tbody>
</table>

3.3.9 Communication and Configuration

The CYC1000 board uses a single chip to perform configuration of the device and USB to UART communications, having each described below.

3.3.9.1 UART Communication

UART to USB communication supports USB 2.0 High Speed (up to 480 Mb/s) independently of other protocols used in the chip like JTAG. Below is the connection schematic and pinning information.

![UART Connections](image)

Figure 13 – UART Connections

<table>
<thead>
<tr>
<th>Board Reference</th>
<th>FPGA Pin No.</th>
<th>Description</th>
<th>I/O Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDBUS0</td>
<td>PIN_R7</td>
<td>Transmitter output of FT2232H (Tx)</td>
<td>3.3 V</td>
</tr>
<tr>
<td>BDBUS1</td>
<td>PIN_T7</td>
<td>Receiver input of FT2232H (Rx)</td>
<td>3.3 V</td>
</tr>
<tr>
<td>BDBUS2</td>
<td>PIN_R6</td>
<td>Ready To Send handshake output (RTS)</td>
<td>3.3 V</td>
</tr>
<tr>
<td>BDBUS3</td>
<td>PIN_T6</td>
<td>Clear To Send handshake input (CTS)</td>
<td>3.3 V</td>
</tr>
<tr>
<td>BDBUS4</td>
<td>PIN_R5</td>
<td>Data Transmit Ready (DTR)</td>
<td>3.3 V</td>
</tr>
<tr>
<td>BDBUS5</td>
<td>PIN_T5</td>
<td>Data Set Ready (DSR)</td>
<td>3.3 V</td>
</tr>
</tbody>
</table>
3.3.9.2 JTAG Chain Configuration

There are two types of configuration methods supported by CYC1000:

1. **JTAG Configuration**: configuration using JTAG ports. JTAG configuration scheme allows you to directly configure the device core through JTAG pins (TDI, TDO, TMS and TCK pins). The Quartus Prime software automatically generates a .sof that can be downloaded to the Cyclone 10 LP with a download cable through the Quartus Prime Programmer.

2. **Configuration from EPCQ-A flash**: configuration using external flash. Before configuration, you need to program the configuration data .jic into the configuration flash memory (EPCQ-A) which provides non-volatile storage for the bit stream. The information is retained within EPCQ-A even if the CYC1000 is turned off. When the board is powered on, the configuration data in the EPCQ-A is automatically loaded into the Cyclone 10 LP FPGA.

The FPGA device can be configured through JTAG interface on CYC1000, but the JTAG chain must form a closed loop, which allows Quartus Prime programmer to detect the FPGA device.

CYC1000 offers two ways of configuring your board.

1) Through the on-board Arrow USB Programmer
2) Pins for connecting user’s preferred JTAG interface

![Figure 14 – JTAG Connections](image)

<table>
<thead>
<tr>
<th>Board Reference</th>
<th>FPGA Pin No.</th>
<th>Description</th>
<th>I/O Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCK</td>
<td>PIN_H3</td>
<td>Test Interface Clock</td>
<td>3.3 V</td>
</tr>
<tr>
<td>TDO</td>
<td>PIN_J4</td>
<td>Test Data Out</td>
<td>3.3 V</td>
</tr>
<tr>
<td>TDI</td>
<td>PIN_H4</td>
<td>Test Data In</td>
<td>3.3 V</td>
</tr>
<tr>
<td>TMS</td>
<td>PIN_J5</td>
<td>Test Mode Select</td>
<td>3.3 V</td>
</tr>
</tbody>
</table>

For detailed information about how to configure the Cyclone 10 LP, please refer to [Chapter 6](#).
3.3.10 Power Tree

The CYC1000 is powered by Enpirion’s buck regulator, which provides high efficiency up to 1A with integrated magnetics, switches, control, and compensation. As seen from the diagram below, the board can be powered either by a micro-USB connection, or by user input voltage from the Arduino MKR header (takes precedence over the USB bus). All devices are powered by 3.3V voltage line and the 5V and 3.3V lines are fed back to the Arduino header to power that connection if needed. The Cyclone 10 LP FPGA is powered by 2 Enpirion devices, while a Microchip LDO provides auxiliary voltage.

![Power Tree Connections Diagram](image-url)
Chapter 4 - Software and Driver Installation

Firstly it is required to create your Basic Intel Account if you don’t own one already. It is required to download the software. Below are guides for installing the software and drivers for Windows operating systems.

4.1 Installing Quartus Prime Software

4.1.1 Go to the Intel Download Center: [Link].
4.1.2 Select Windows as the operating system (highlighted in red).
4.1.3 Select Release 18.1, or your preferred version (highlighted in red).
4.1.4 Download the following files from the “Individual Files” tab (highlighted in yellow):
   - Quartus Prime Lite Edition (Free)
   - ModelSim-Intel FPGA Edition (includes Starter Edition)
   - Cyclone 10 LP device support

4.1.5 Click on button to begin the download and save them in the same folder.
4.1.6 After the download is finished, run the Quartus Prime installer.

4.1.7 When prompted to select the components, the installer will detect automatically the Cyclone 10 LP device support and ModelSim packages when they are in the same folder. Make sure these components are selected:

![Select Components](image)

4.1.8 Finish the installation of the Quartus Lite and proceeded to the next section to install Arrow USB Programmer2 to be able to connect to the CYC1000 board.

4.2 Installing Arrow USB Programmer2

The CYC1000 board uses version 2 of the Arrow USB Programmer2 programming solution, that is an FTDI FT2232H Hi-Speed USB controller plus a programmer DLL. Since this FTDI USB controller is a very common standard device, usually no specific drivers are needed to make the CYC1000 work.
4.2.1 Download the appropriate version of Arrow USB Programmer2 for CYC1000 from Trenz Electronic Wiki page or alternatively this direct [link](#).

4.2.2 After downloading the file, run the installer to install the Arrow USB Programmer2. The setup executable installs the programmer DLL and adds some keys to the registry of the PC.

4.2.3 After connecting the CYC1000 board to the PC, two unknown devices might appear in the “Other devices” section of device manager of the PC.

Windows usually automatically finds the appropriate drivers for these devices. After some time, the “Other devices” section should be empty. Instead, two USB Serial Converters should be listed in the section “USB Serial Bus controllers”:
Furthermore, a USB Serial Port should be listed in the “Ports (COM & LPT)” section.

![Network adapters]
- Ports (COM & LPT)
  - intel(R) Active Management Technology - SOL (COM3)
  - USB Serial Port (COM9)
- Print queues

*Note: The number of the port will most probably be different from the one shown here.*

In case Windows does not automatically find the appropriate drivers go to [http://www.ftdichip.com/Drivers/D2XX.htm](http://www.ftdichip.com/Drivers/D2XX.htm) to download the setup executable to install the required drivers.
Chapter 5 - New Project with CYC1000

5.1 Creating a new Blinky Project with CYC1000

5.1.1 Launch Quartus Prime Lite Edition from the Start Menu.

5.1.2 In the Quartus Prime tool, create a new project: File -> New Project Wizard.

The New Project Wizard walks you through the project settings, such as the name, directories, files, directories, device family and other settings. These settings can be changed later if needed.

5.1.3 Click “Next”.

5.1.4 Browse in the project directory and choose a preferred location for the new project. Then create new folder named CYC1000_blinky. This will be the folder containing all the project files.

5.1.5 Enter the project name: “top”.

5.1.6 Click “Next”.
5.1.7 Project Type

In this page you choose the Project Type. In this tutorial, a new project will be created, and thus the default settings of empty project should be selected.

![Project Type screenshot]

5.1.8 Click “Next”.

5.1.9 Add Project Files

The Add File window will appear. For this tutorial, new design files will be created so no files will be added. For other designs, files could be added here.

![Add Files screenshot]

5.1.10 Click “Next”.

5.1.11 Select the Device Part Number of the CYC1000 Board
In the Family and Device Settings, use the pull-down menu to select the family as Cyclone 10 LP. Then in the Name Filter enter **10CL025YU256C8G**.

Rather than entering the exact part number, the pull-down menus can be used to select the correct family, package, pin count, and speed grade. Quartus Prime will use these settings to compile the design, and also provide the programming file that you will use later to program the device.

5.1.12 Click “Next”.

5.1.13 EDA Tool Settings

In the EDA tool Settings window, disable any EDA tools, if there are any present. EDA tools are third party tools that work with Quartus Prime for design entry, simulation, verification and board-level timing. For this tutorial, no EDA software will be used, as only Quartus Prime will be used.
5.1.14 Click “Next”.

5.1.15 Project Summary Page

This is the Summary Page that showing the settings Quartus Prime will use for this Project. Those settings can be changed if required at a later time.

5.1.16 Click “Finish”.

5.2 Building a Blinky Project with CYC1000

Overview: In this section you will create the components to a design, make connections, set the pins and compile a project. The goal is to go through the design process of a simple blinky project, where the toggle speed of the LEDs could be controlled by one of the pushbuttons of the CYC1000.
5.2.1  Block Diagram

The final system that will be built with the following steps will look as follows when complete:

5.2.2  Components of the Design

There are three components in the system: a PLL, a counter and a mux. The components, in the following steps, will be built separately and then connected together. A user push button on the board controls the mux. The mux in turn control which of the counter outputs (slow counting or fast counting) will be shown on the LEDs. There are different ways to create components, such as RTL or schematic. In this lab, schematics will be used. There are also different ways for entering schematics such as Qsys and IP Catalog. This lab will focus on the IP Catalog.
5.2.3 Catalog IP

The IP Catalog allows you to create and modify design files with custom variations. The IP Catalog window is open by default when you open Quartus Prime. If it’s not present, you can open it by going to the tab Tool → IP Catalog.

![IP Catalog window](image)

5.2.4 Create a PLL

In the IP Catalog, browse for ALTPLL, via: Basic Functions → Clocks; PLLs and Resets → PLL or type in the search field for “PLL”.

5.2.4.1 In the Search bar of the IP Catalog, type “pll” and select ALTPLL which stands for Altera Phase Locked Loop.

![Search for ALTPLL](image)
5.2.4.2 Click “Add”. When the Save IP Variation window appears, enter the file name variation as PLL and select VHDL (Verilog can be used as well). Both Verilog and VHDL schematics will be created.

![Save IP Variation window]

5.2.4.3 Click “OK”.

5.2.5 Create and Configure the PLL

The next step is to configure the PLL component that we just named.

5.2.5.1 Enter the PLL reference clock frequency to match the clock input on the CYC1000 Board. Since we have a 12 MHz coming into the FPGA, the inclk0 input will be 12MHz.

The setting should look like this:

![PLL configuration window]

5.2.5.2 Click “Next”.
5.2.5.3 Simplify the PLL, by disabling ‘areset’ and ‘locked outputs’.

The setting should look like this:

5.2.5.4 Click “Next”.

5.2.5.5 Continue to select Next to go through the various options (e.g. Pages 3 to Pages 5) but leaving the default options as they are. The page numbers can be seen on the top of the window.
5.2.5.6 On page 6, (c0-Core/External Output Clock) select “Enter output clock frequency” and set the requested setting to 20 MHz, leave the rest as default. For simplification, there is one input to the PLL (12 MHz), and one output of the PLL (20 MHz).

5.2.5.7 Click “Next” until reaching page 12.

5.2.5.8 On page 12 there is a list of output files that will be generated. Since the design will be done in a schematic, you will need to select PLL.bsf checkbox. The .bsf file provides a symbol that can be used in the schematic design we will be creating later.
5.2.5.9  Click “Finish”. The PLL (1st component) will now be created.

5.2.5.10 If this is the first time that you are using this version of Quartus Prime, you might see a pop-up Window for Quartus Prime IP Files, that asks if the tool should add IP files automatically after generating them.

5.2.5.11 Select “Automatically add Quartus Prime IP Files to all projects”.

5.2.5.12 Click “Yes” to allow all of the IP to automatically be added to the project, and so that this message will not be seen for other designs.

5.2.6  Create and Configure the Counter

The next step is to create the counter which will drive the LEDs on the CYC1000 board.

5.2.6.1  To create this counter, select the IP Catalog and expand the Basics → Arithmetic and select the LPM_COUNTER or type “counter” in the search field.

![Image of IP Catalog]

Note: LPM stands for Library of Parameterized Modules

5.2.6.2 Click “Add”.

5.2.6.3 When the Save IP Variation pop up appears, enter “simple_counter” and select VHDL as below:

![Save IP Variation](image)

5.2.6.4 Click “OK”.

5.2.6.5 The next step is to increase the size of the counter to a number of bits large enough to divide down the clock so we can see the LEDs toggling.

5.2.6.6 Change this number to 32.

5.2.6.7 Let the counter to be Up only, so the LEDs will show the counters counting up.

![LPM_Counter](image)

5.2.6.8 Select “Next” until reaching Page 5.
5.2.6.9 Select simple_counter.bsf checkbox to generate a symbol for our schematic design.

![Image of LPM_COUNTER]

5.2.6.10 Click “Finish”.

The counter is now created.

5.2.7 Create and Configure the Multiplexer

The next step is to create a mux component. This mux will be used along with a push button on the CYC1000 board to control the speed of the counter, where the counter outputs will be seen on the LEDs.

5.2.7.1 To create this mux, select IP Catalog and expand Basic Functions → Miscellaneous and select LPM_MUX or type mux in the search field.

![Image of IP Catalog]

5.2.7.2 Click “Add”.

www.arrow.com
5.2.7.3 In the Save IP Variation, enter the name of the counter_mux and the file type to be VHDL.

5.2.7.4 Click “OK”.

5.2.7.5 Select 2 data inputs and the width of the input and output buses to be 8 bits. The reason for 8 bits is that there are 8 LEDs to be toggled (showing count values).

The screen should look like this now:

5.2.7.6 Click “Next” until Page 3.
5.2.7.7 Select counter_mux.bsf checkbox to generate a symbol for our schematic design.

5.2.7.8 Click “Finish”.

5.2.8 Adding the Components to the Schematic

The next step would be to connect all three components together.

5.2.8.1 To do so, select File menu, then select New and select Block Diagram/Schematic File.

5.2.8.2 Click “OK”.

A new schematic will be created, where the components can be added.
5.2.8.3 Right click on the schematic page and select **Insert → Symbol...** as seen below.

5.2.8.4 In the new window, expand “Project” and the three components that were created can now be seen.

5.2.8.5 Select “PLL”.

5.2.8.6 Click “OK”.

5.2.8.7 The PLL component can be added now by left clicking on the schematic page.

5.2.8.8 Just like in the steps from 5.2.8.3 to 5.2.8.6, do the same for counter_mux and simple_counter to add them to the schematic page. The order of adding the components does not matter, as the connections between them will happen in the following steps.
5.2.8.9 After adding three components, your schematic should look similar to the following. To place them similarly, simply drag the components to the appropriate locations.

![Schematic diagram]

5.2.9 Connecting the Components

Next step is to make the proper connections between the three components we just added to the schematic.

5.2.9.1 Select the “Node Tool”.

![Node Tool selection]

![Schematic diagram with connections]
5.2.9.2 Connect the c0 of the PLL to the simple_counter as shown below:

![](image1.png)

This will mean that a single signal (c0) is connected to the simple_counter (clock).

5.2.9.3 Select the “Bus Tool”.

![](image2.png)
5.2.9.4 Using the bus tool create a connection coming out of the simple_counter and one connection for each of the inputs of the counter_mux as show below.

5.2.9.5 Right click on the output bus of the simple counter that you just created and select “Properties”.

Set the name of the bus to: \texttt{counter[31..0]}

The view of the “Bus Properties” should look like this:

5.2.9.6 Click “OK”.

5.2.9.7 Do the same for input buses of the mux:

Name the top bus input: \texttt{data1x[7..0] \rightarrow counter[24..31]}

Name the bottom bus input: \texttt{data0x[7..0] \rightarrow counter[19..26]}
5.2.10 Add inputs, outputs to the schematic

5.2.10.1 Click on the “Pin Tool” as show below and select “Input”.

5.2.10.2 Add one input pin for inclk0 of the PLL and add other one input pin for sel of counter_mux.
Your schematic should look like this:

5.2.10.3 Rename the pin_name1 to **CLK12M** by double clicking its current name. This is going to be the clock signal coming into the FPGA.

5.2.10.4 Rename the pin_name2 to **USER_BTN** by double clicking its current name. This is going to be the user button of the CYC1000 board to select the mux.

5.2.10.5 Using the **"Node Tool"** connect:
- **CLK12M** → **inclk0** (of the PLL component)
- **USER_BTN** → **sel** (of the counter_mux component)

Your schematic should look like this now:
5.2.10.6 Click on the “**Pin Tool**” as before, but this time select “**Output**”.

![Pin Tool Image]

5.2.10.7 Add one output pin for the LEDs.

5.2.10.8 Rename the pin to **LED[7..0]**.

5.2.10.9 Using the “**Bus Tool**”, make the connection between counter_mux component and output pin:

\[
\text{result}[7..0] \rightarrow \text{LED}[7..0]
\]

The final schematic should look like the following:

![Schematic Image]

Looking at the schematic, even though the buses are not connected together by wires, the names of counter tell Quartus Prime to connect the signals together. Overall, the user button will toggle between displaying higher 8 bits of the counter and 8 lower bits of the counter. The signals of the counter that are not connected will not be used by Quartus Prime.
5.2.10.10 Save your design. Open the File Menu and select “Save”. Save it as top.bdf

5.2.11 Analysis and Synthesis

The next step is to run Analysis and Synthesis to ensure that there are no errors in the design. To run Analysis and synthesis open Processing → Start → Analysis and Synthesis or from clicking button on the top toolbar.

There should be no errors. If there are errors, they should be fixed before continuing and Analysis and Synthesis run again.

5.2.12 Adding Timing Constraints

Timing Constraints tell the Quartus what are the timing requirements for this design. Timing Constraints are required in every CPLD/FPGA design.

5.2.12.1 To add the timing constraints, select File → New and under the “Other File” section, select “Synopsys Design Constraints File” and select “OK”.

![Timing Constraints Menu](image-unavailable)
5.2.12.2 Type or copy the following lines into this new file:

```
# create input clock which is 12MHz
create_clock -name CLK12M -period 83.333 [get_ports {CLK12M}]
#
# derive PLL clocks
derive_pll_clocks
#
# derive clock uncertainty
derive_clock_uncertainty
#
# set false path
set_false_path -from [get_ports {USER_BTN}]
set_false_path -from * -to [get_ports {LED*}]
```

The first line “create_clock” tells Quartus Prime that the clock, CLK12M is 83.333 ns (12 MHz). It also assigns the CLK12M to a pin (port) in the .sdc format.

The second line “derive_pll_clocks” tells the software to look if there are any PLLs, and if so, automatically derive the clock multiplication/division of the outputs of the PLL even if they are used internally within the CPLD/FPGA.

The third line “derive_clock_uncertainty” tells the software to automatically determine the internal clock uncertainty. No clock is ideal, and thus there will be some internal jitter within the FPGA associated with it.

The fourth and fifth line “set_false_path” tells the software to not do any timing optimization to the stated paths/pins. The I/Os of this design are trivial, so they can be ignored in the Timing Analysis.

5.2.12.3 Use File → Save to save it as top.sdc.
5.2.12.4 Ensure that the file is added to the Project: Assignments → Settings and select “Timing Analyzer”. The top.sdc should have been already added by default. If it is not, it will need to be added manually.

5.2.13 Pinning Assignments

Before the design can be downloaded to the FPGA, pin assignments that match the hardware on the board are needed. There are different ways to do this such as the Pin Planner, Assignment Editor, and text files.

The following steps will show one of these ways, the Pin Planner. Since there are only 10 pins that need to be assigned, the Pin Planner can be used. If many pins are needed, other ways can be used such as the Quartus Assignment Editor, or by importing constraints from a text file or spreadsheet.
5.2.13.1 Open the Pin Planner: **Assignments → Pin Planner.**

A new window will open as seen below:

5.2.13.2 To make pin assignments, select the CLK12M (node name) on the bottom portion and drag and drop it to pin M2 of the Top View of the FPGA or alternatively set the Location field of the CLK12M to **PIN_M2.**

Note that the **Location of the CLK12M is now set to Location PIN_M2 (as seen in blue colour in the top view of the FPGA).**
5.2.13.3 The other pins need to be assigned as well. Just like previously set all the pins to their appropriate locations using the table below, by either drag and drop or writing manually the location.

<table>
<thead>
<tr>
<th>Node Name</th>
<th>Pin Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED[7]</td>
<td>PIN_N3</td>
</tr>
<tr>
<td>LED[5]</td>
<td>PIN_R4</td>
</tr>
<tr>
<td>LED[4]</td>
<td>PIN_T2</td>
</tr>
<tr>
<td>LED[3]</td>
<td>PIN_R3</td>
</tr>
<tr>
<td>LED[2]</td>
<td>PIN_T3</td>
</tr>
<tr>
<td>LED[1]</td>
<td>PIN_T4</td>
</tr>
<tr>
<td>LED[0]</td>
<td>PIN_M6</td>
</tr>
<tr>
<td>USER_BTN</td>
<td>PIN_N6</td>
</tr>
</tbody>
</table>

5.2.13.4 Now the Pin Planner should look like this after assigning all the pin locations.

5.2.13.5 The specific pins are now selected, but the I/O standards now need to be set as well. The button, LEDs, and clock pins are the same I/O standard for CYC1000 since all banks and peripherals are powered by 3.3V. The USER_BTN, the LEDs and clock pins are 3.3-V LVTTL. These I/O standards can be set in the Pin Planner, by selecting the I/O Standard. Select the I/O standard either from the “All Pins” tab or the “Groups” tab and change the 2.5V (default) to the specific I/O standard.
The Pin Planner should now look like this:

5.2.13.6 Close the Pin Planner. The settings are automatically saved.

5.2.14 Compiling the Design

5.2.14.1 You can set the default I/O Standard which can eliminate some design warning and save you time from setting the standard for some pins manually.

Open Assignments → Device → Device and Pin Options → Voltage and set Default I/O Standard to “3.3-V LVTTL” and press “OK” to all the windows.
The next step is to compile and complete the design. This step will verify that there are no errors, create internal databases, and create programming files that will be used in the next steps.

5.2.14.2 To compile the design, select **Processing ➔ Start Compilation** or push the button on the toolbar.

If there are errors, they will need to be resolved and re-compiled before the design can be programmed to the board. When Compiling finishes and there are no errors, there will be a message at the bottom of the window that states: Full Compilation was successful and a 100% indication along with the compile time in the right bottom corner.

5.2.15 Reading the Compilation Report

After successfully compiling the design, a Compilation Report should appear as shown above:
This report is very useful with a lot of information about the design. Last message state that the design was fully constrained, Timing Analysis and compilation successful, but there is more to it:

- In the Flow Summary, it can be seen how many logic elements the whole design took, along with total PLLs, registers, pins, etc.

- In Analysis and Synthesis, more detailed information about the resources used can be seen in Resource Usage Summary, as well how many LEs were used for each component in Resource Utilization by Entity.

- In the Fitter, more detailed information about the pins and their banks can be seen.

- Timing Analyzer shows various timing information concerning the design, as well as if the design has met the timing requirements. In this case timing requirements were met, but in other cases that requirements might not be met, could be solved by going over the information provided in the reports inside this folder. Most notable reports in this folder are the maximum frequency the design can achieve, setup and hold slack, unconstrained paths in case they were missed, etc.
Chapter 6 - Configuring the CYC1000

After successfully compiling your project, there should new files generated. In case of Cyclone 10 LP devices, only the .sof file is generated automatically.

6.1 Configure the FPGA in JTAG mode

6.1.1 Connect your CYC1000 board to your PC using an USB cable. Since the Arrow USB Blaster should be already installed, the Window’s Device Manager should display the following entries are highlighted in red (port number may differ depending on your PC). If the Arrow USB Blaster is not installed, please refer to Chapter 4.2 for installing the drivers.

6.1.2 Open the Quartus Prime Programmer from Tools → Programmer or double-click on Program Device (Open Programmer) from the Tasks pane.
6.1.3 The programmer should add the programming file automatically. After opening the program this should be the view of the new window:

![New window view]

6.1.4 Click Hardware Setup... and double click Arrow-USB-Blaster entry in the Hardware Setup tab. The Currently selected hardware should now show Arrow-USB-Blaster [USB0] (depending on your PC, the USB port number may variant).

![Hardware Setup dialog]

6.1.5 Click “Close”.

6.1.6 Make sure the hardware setup is Arrow-USB-Blaster [USB0] and the mode is JTAG. If the Mode is not set to JTAG, click on it and select JTAG from the drop-down menu.

![Hardware Setup dialog with JTAG selected]
6.1.7 If the configuration has been added by default, you can skip the following steps and continue with the 6.1.12 point.

6.1.8 Click “Auto Detect” on the left side of the Programmer.

6.1.9 Select **10CL025Y** device and click “OK” on the Select Device window.

6.1.10 Double click <none> to choose programming file.

6.1.11 Navigate to `<project_directory>/output_files/` in your compilation directory. Select and open the **top.sof** file.
6.1.12 Make sure the Programmer shows the correct file and correct part in the JTAG chain and check the Program/Configure checkbox.

6.1.13 Click Start to program the CYC1000. When the configuration is complete, the Progress bar should reach 100% (Successful).

The design is now programmed to the FPGA.

**Note:** that turning off and then on the FPGA will result into losing its configuration.

6.2 Serial configuration flash memory programming

The configuration data to be written to EPCQ-A will be part of the JTAG indirect configuration file (.jic). This configuration data is automatically loaded from the serial configuration flash into the Cyclone 10 LP device when the board is powered up.
6.2.1 Programming File generation

6.2.1.1 In Quartus Prime, go to **File → Convert Programming Files...**

![Convert Programming File dialog box](image)

6.2.1.2 Set the programming file type to **JTAG Indirect Configuration File (.jic)**.

6.2.1.3 Select **EPCQ16A** from the drop-down menu for configuration device and make sure that the **Active Serial** is set to mode. The output programming file settings should look like this:

![Output programming file settings](image)
6.2.1.4 Select **Flash Loader** under Input files to covert settings and click on **Add Device...** button.

6.2.1.5 On the new window select **Cyclone 10 LP** as Device family and **10CL025Y** as Device name.

6.2.1.6 Click **OK** to add device to Flash Loader.

6.2.1.7 Select **SOF Data** under Input files to convert and click on **Add File...** button.

6.2.1.8 Go to `<project_directory>/output_files/` and open `top.sof`. 
6.2.1.9 Make sure that your settings are same as the picture below and if everything is correct, click **Generate**.

6.2.1.10 Click **OK** on the successful file generation notification and close Convert Programming File window.

6.2.2 Device Programming

6.2.2.1 Open Programmer.
6.2.2.2 Select **output_files/top.sof** and click **Change File...** button.

6.2.2.3 Go to `<project_directory>/output_files/` and open **output_file.jic**. When you add the .jic file, the Programmer will automatically update the JTAG chain and put EPCQ-A flash memory.

6.2.2.4 Make sure the Programmer shows the correct file and correct parts in the JTAG chain and check the Program/Configure checkbox.

6.2.2.5 Click **Start** to configure EPCQ-A. The programming could take a while.

6.2.2.6 When the programming is finished, the CYC000 should be able to keep its configuration data even after powered off.
At this point our program is stored in the EPCQ-A flash memory, but the Cyclone 10 LP current configuration is the Serial Flash Loader which is responsible for programming configuration flash memory. We can simply reconfigure the FPGA with our program by pushing RESET button which will reset the FPGA and automatically loads the configuration from EPCQ-A.

6.3 Testing the Design

Does not matter which way the CYC1000 was configured, the results should be the same for both methods, with the only difference being if configuration is retained after power off.

On the board by default, the LEDS should now toggle in a slow counting sequence.

Push and hold the USER_BTN to see that the LEDs will now toggle in a very fast counting sequence. USER_BTN is on the side of the LEDs.

Releasing the USER_BTN, will make the LEDs toggle at a slower rate as before.
Chapter 7 - Common Issues and Fixes

1) **Issue:** In some rare cases when using Windows 10 operating system, the programmer DLL is not properly loaded/unloaded, causing the Quartus Programmer to not detect the Arrow USB Programmer2.

**Solution:** Restart the Altera JTAG Server using the Services application of Windows.
Chapter 8 - Appendix

8.1 Revision History

<table>
<thead>
<tr>
<th>Version</th>
<th>Change Log</th>
<th>Date of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1.0</td>
<td>Initial Version</td>
<td>03/02/2020</td>
</tr>
</tbody>
</table>
8.2 Legal Disclaimer

ARROW ELECTRONICS

EVALUATION BOARD LICENSE AGREEMENT
By using this evaluation board or kit (together with all related software, firmware, components, and
documentation provided by Arrow, "Evaluation Board"), You ("You") are agreeing to be bound by the terms and
conditions of this Evaluation Board License Agreement ("Agreement"). Do not use the Evaluation Board until
You have read and agreed to this Agreement. Your use of the Evaluation Board constitutes Your acceptance of
this Agreement.

PURPOSE
The purpose of this evaluation board is solely intended for evaluation purposes. Any use of the Board beyond
these purposes is on your own risk. Furthermore, according the applicable law, the offering Arrow entity
explicitly does not warrant, guarantee or provide any remedies to you with regard to the board.

LICENSE
Arrow grants You a non-exclusive, limited right to use the enclosed Evaluation Board offering limited features
only for Your evaluation and testing purposes in a research and development setting. Usage in a live environment
is prohibited. The Evaluation Board shall not be, in any case, directly or indirectly assembled as a part in any
production of Yours as it is solely developed to serve evaluation purposes and has no direct function and is not
a finished product.

EVALUATION BOARD STATUS
The Evaluation Board offers limited features allowing You only to evaluate and test purposes. The Evaluation
Board is not intended for consumer or household use. You are not authorized to use the Evaluation Board in any
production system, and it may not be offered for sale or lease, or sold, leased or otherwise distributed for
commercial purposes.

OWNERSHIP AND COPYRIGHT
Title to the Evaluation Board remains with Arrow and/or its licensors. This Agreement does not involve any
transfer of intellectual property rights ("IPR") for evaluation board. You may not remove any copyright or other
proprietary rights notices without prior written authorization from Arrow or it licensors.

RESTRICTIONS AND WARNINGS
Before You handle or use the Evaluation Board, You shall comply with all such warnings and other instructions
and employ reasonable safety precautions in using the Evaluation Board. Failure to do so may result in death,
personal injury, or property damage.

You shall not use the Evaluation Board in any safety critical or functional safety testing, including but not limited
to testing of life supporting, military or nuclear applications. Arrow expressly disclaims any responsibility for
such usage which shall be made at Your sole risk.

WARRANTY
Arrow warrants that it has the right to provide the evaluation board to you. This warranty is provided by Arrow
in lieu of all other warranties, written or oral, statutory, express or implied, including any warranty as to
merchantability, non-infringement, fitness for any particular purpose, or uninterrupted or error-free operation,
all of which are expressly disclaimed. The evaluation board is provided “as is” without any other rights or
warranties, directly or indirectly.

You warrant to Arrow that the evaluation board is used only by electronics experts who understand the dangers
of handling and using such items, you assume all responsibility and liability for any improper or unsafe handling
or use of the evaluation board by you, your employees, affiliates, contractors, and designees.
LIMITATION OF LIABILITIES
In no event shall Arrow be liable to you, whether in contract, tort (including negligence), strict liability, or any other legal theory, for any direct, indirect, special, consequential, incidental, punitive, or exemplary damages with respect to any matters relating to this agreement. In no event shall arrow’s liability arising out of this agreement in the aggregate exceed the amount paid by you under this agreement for the purchase of the evaluation board.

IDENTIFICATION
You shall, at Your expense, defend Arrow and its Affiliates and Licensors against a claim or action brought by a third party for infringement or misappropriation of any patent, copyright, trade secret or other intellectual property right of a third party to the extent resulting from (1) Your combination of the Evaluation Board with any other component, system, software, or firmware, (2) Your modification of the Evaluation Board, or (3) Your use of the Evaluation Board in a manner not permitted under this Agreement. You shall indemnify Arrow and its Affiliates and Licensors against and pay any resulting costs and damages finally awarded against Arrow and its Affiliates and Licensors or agreed to in any settlement, provided that You have sole control of the defense and settlement of the claim or action, and Arrow cooperates in the defense and furnishes all related evidence under its control at Your expense. Arrow will be entitled to participate in the defense of such claim or action and to employ counsel at its own expense.

RECYCLING
The Evaluation Board is not to be disposed as an urban waste. At the end of its life cycle, differentiated waste collection must be followed, as stated in the directive 2002/96/EC. In all the countries belonging to the European Union (EU Dir. 2002/96/EC) and those following differentiated recycling, the Evaluation Board is subject to differentiated recycling at the end of its life cycle, therefore: It is forbidden to dispose the Evaluation Board as an undifferentiated waste or with other domestic wastes. Consult the local authorities for more information on the proper disposal channels. An incorrect Evaluation Board disposal may cause damage to the environment and is punishable by the law.