# Gowin USB 2.0 Device Controller IP

# User Guide

**IPUG927-1.5E, 04/27/2023**

**Disclaimer**

GOWINSEMI assumes no liability and provides no warranty (either expressed or implied) and is not responsible for any damage incurred to your hardware, software, data, or property resulting from usage of the materials or intellectual property except as outlined in the GOWINSEMI Terms and Conditions of Sale. GOWINSEMI may make changes to this document at any time without prior notice. Anyone relying on this documentation should contact GOWINSEMI for the current documentation and errata.

## Revision History

| Date | Version | Description |
|---|---|---|
| 03/05/2021 | 1.0E | Initial version published. |
| 01/04/2022 | 1.1E | ● PID configuration interface in synchronized mode added.<br>● Interface for interface configuration added. |
| 03/01/2022 | 1.2E | ● Data width of descriptors extended.<br>● HID function added. |
| 03/24/2022 | 1.3E | The description of resource utilization added. |
| 07/21/2022 | 1.4E | The description of USB Device Descriptor interface updated. |
| 04/27/2023 | 1.5E | ● ULPI supported.<br>● I/O list updated. |

# Contents

# List of Figures

# List of Tables

# 1 About This Guide

## 1.1 Purpose

The purpose of Gowin USB 2.0 Device Controller IP User Guide is to help you learn the features and usage of Gowin USB 2.0 Device Controller IP by providing the descriptions of functional description, signal description and interface configuration, etc.

## 1.2 Related Documents

The latest user guides are available on the GOWINSEMI Website. You can find the related documents at: http://www.gowinsemi.com.

- DS100, GW1N series of FPGA Products Data Sheet
- DS117, GW1NR series of FPGA Products Data Sheet
- DS891, GW1NRF series of FPGA Products Data Sheet
- DS821, GW1NS series of FPGA Products Data Sheet
- DS871, GW1NSE series of FPGA Products Data Sheet
- DS881, GW1NER series of FPGA Products Data Sheet
- DS861, GW1NSR series of FPGA Products Data Sheet
- DS102, GW2A series of FPGA Products Data Sheet
- DS226, GW2AR series of FPGA Products Data Sheet
- SUG100, Gowin Software User Guide

# 1.3 Terminology and Abbreviations

The terminology and abbreviations used in this manual are as shown in Table 1-1.

**Table 1-1 Terminology and Abbreviations**

| Terminology and Abbreviations | Meaning |
|---|---|
| FS | High Speed |
| HID | Human Interface Device |
| HS | High Speed |
| IP | Intellectual Property |
| LS | Low Speed |
| PID | Packet ID |
| ULPI | UTMI+ Low Pin Interface |
| USB | General Serial Bus |
| UTMI | USB 2.0 Transceiver Macrocell Interface |

# 1.4 Support and Feedback

Gowin Semiconductor provides customers with comprehensive technical support. If you have any questions, comments, or suggestions, please feel free to contact us directly by the following ways.

Website: www.gowinsemi.com

E-mail: support@gowinsemi.com

# 2 Function Overview

## 2.1 Overview

Universal Serial Bus, commonly known as USB, is a kind of external bus to standardize the connection and communication between computers and external devices.

Based on USB 2.0 Protocol, Gowin USB 2.0 Device Controller IP supports three modes, including High Speed (HS), Full Speed(FS) and Low Speed(LS). It also supports Control Transaction, Bulk Transaction, Isochronous Transaction, Interrupt Transaction, and UTMI. Gowin USB 2.0 Device Controller IP can realize up to 15 IN/OUT data transfer endpoints, where each endpoint supports Bulk Transaction, Isochronous Transaction and Interrupt Transaction. The IP has two ends, including UTMI and parallel data interface. It connects to USB PHY through UTMI and user design through parallel data interface.

**Table 2-1 Gowin USB 2.0 Device Controller IP Overview**

| Gowin USB 2.0 Device Controller IP | |
|---|---|
| Logic Resource | Please refer to Table 2-2 and Table 2-3. |
| Delivered Doc. | |
| Design Files | Verilog (encrypted) |
| Reference Design | Verilog |
| TestBench | Verilog |
| Test and Design Flow | |
| Synthesis Software | GowinSynthesis |
| Application Software | Gowin Software (V1.9.9 Beta and above)） |

**Note!**

For the devices supported, you can click **here** to get the information.

# 2.2 Features

The features of USB 2.0 Device Controller IP include:

- Supports HS (480 Mbps), FS (12 Mbps), LS (1.5 Mbps)
- Supports 1 control IN/OUT endpoint and 15 data IN/OUT endpoints
- Supports multiple USB devices
- Supports USB 2.0 UTMI and ULPI

# 2.3 Resource Utilization

Gowin USB 2.0 Device Controller IP can be implemented by Verilog. Its resource utilization may vary when the design is employed in different devices, or at different densities, speeds, or grades. Taking Gowin GW2AR-18 and GW1NSR-4 series of FPGA ptoducts as an instance, the resource utilization is as shown in Table 2-2 and Table 2-3.

**Table 2-2 Resource Utilization (I)**

| Device | Speed Grade | Resource | Utilization | Notes |
|---|---|---|---|---|
| GW2AR-18 | -6 | LUT | 1513 | - |
| | | REG | 478 | |
| | | ALU | 113 | |
| | | BSRAM | 0 | |

**Table 2-3 Resource Utilization (Ⅱ)**

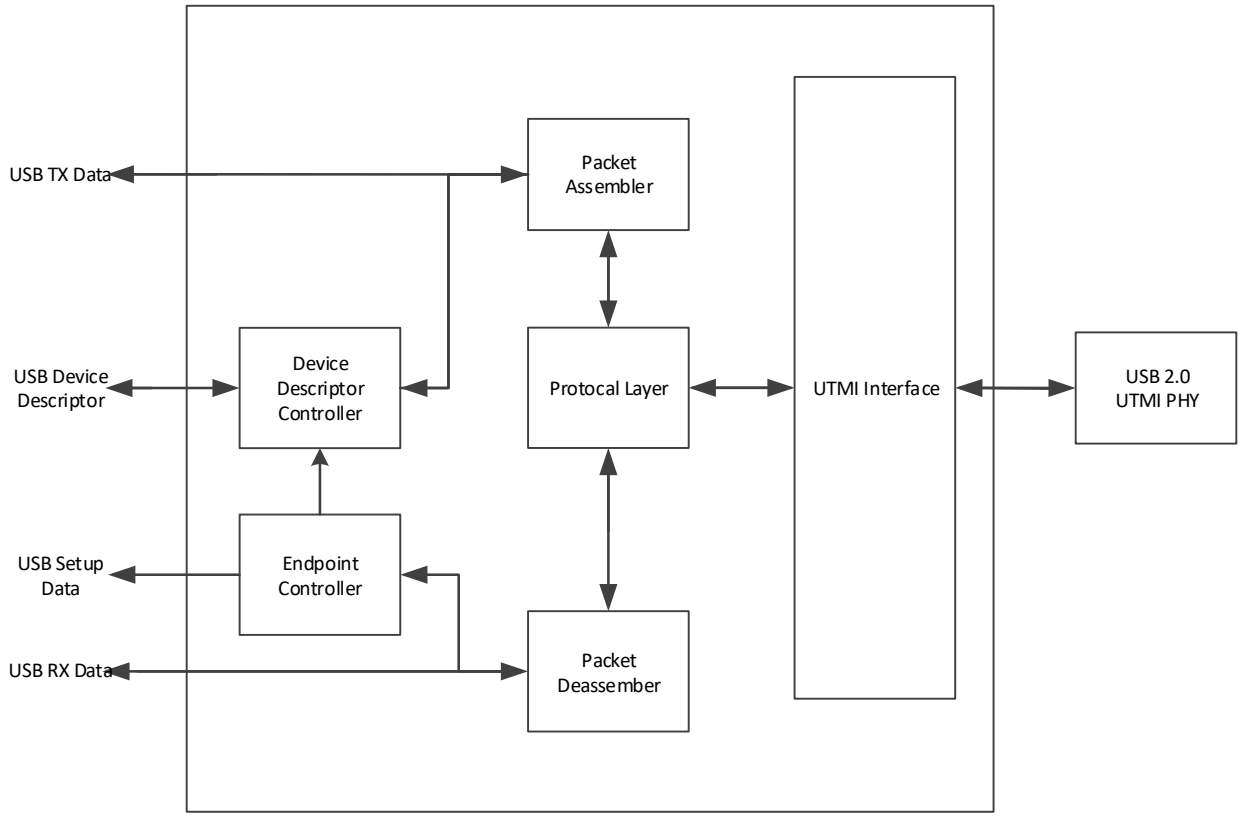
| Device | Speed Grade | Resource | Utilization | Notes |
|---|---|---|---|---|
| GW1NSR-4 | -6 | LUT | 1416 | - |
| | | REG | 489 | |
| | | ALU | 106 | |
| | | BSRAM | 0 | |

# 3 Functional Description

## 3.1 USB Device Controller

USB Device Controller is located between User Design and PHY. USB Controller connects user design with PHY in series, receives commands from USB and realizes the data transaction between user design and USB. The functional block diagram of USB device controller is as follow.

**Figure 3-1 USB Device Controller Block Diagram**



# 3.2 USB Device Controller TX User Interface

**Table 3-1 Device Controller TX User Interface**

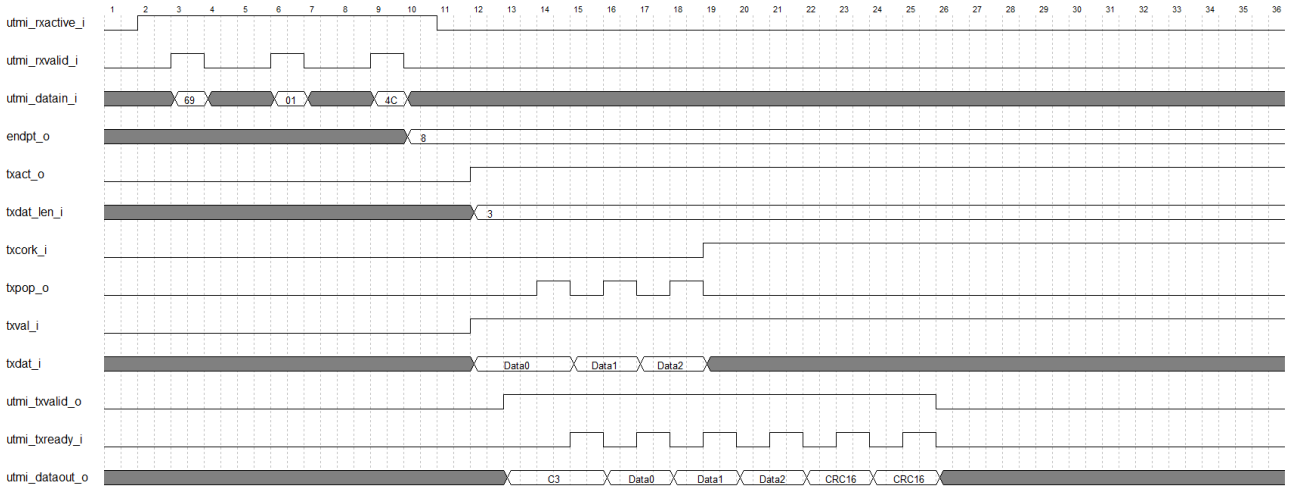
| Name | I/O | Bit Width | Description |
|---|---|---|---|
| txact_o | Output | 1 | Transmitting Work Signal<br>High, indicates that the device is in data transmit state. |
| txdat_i | Input | 8 | Transmitting Data<br>IP transmits this data through the USB interface. |
| txdat_len_i | Input | 12 | The Number of Bytes of Transmitting Data<br>It can be used to control the number of bytes of TX data. |
| txiso_pid_i | Input | 4 | Valid only for synchronous transfer; can be configured as 4'b0011 (DATA0), 4'b1011 (DATA1), 4'b0111 (DATA2) or 4'b1111 (MDATA). |
| txcork_i | Input | 1 | Transmitting Valid Data Indicator Signal<br>Low, indicates that TXDAT is insufficient. |
| txpop_o | Output | 1 | Transmitting Read Signal<br>High, indicates that read next data. |
| endpt_o | Output | 4 | Endpoint Selection Indicator Signal<br>It indicates the current communication endpoint of USB. |

When Device receives IN command transmitted by Host, txact_o signal is set to 1 and endpt_o outputs the endpoint address where Host is going to read data. After txact_o is set, you should check whether the transmitting data of corresponding endpoint is prepared. And if it is not ready, set txcork_i to 1 and do not return the data. If the data is ready, set txcork_i to 0 and set txdat_len_i to the number of data bytes (≤ Max data packet size) that current endpoint can transmit. Txdat_len_i should be maintained during the the process of read data.

Device Controller determines whether to return packets to Host according to the status of txcork_i. If the status of txcork_i is high, it means that the transmitting data is not ready and cannot be transmitted. So Device replies data of length 0 to Host. If txcork_i is low, it indicates that transmitting data is ready and can be transmitted. Controller read the corresponding transmitting data according to the size of txdat_len_i. If

txpop_o is high, it indicates that the data of txdat_i is read and request to read next data. Device data transmit timing Diagram is as follow.

**Figure 3-2 USB Device Controller Data Transmit Timing Diagram**




# 3.3 USB Device Controller RX User Interface

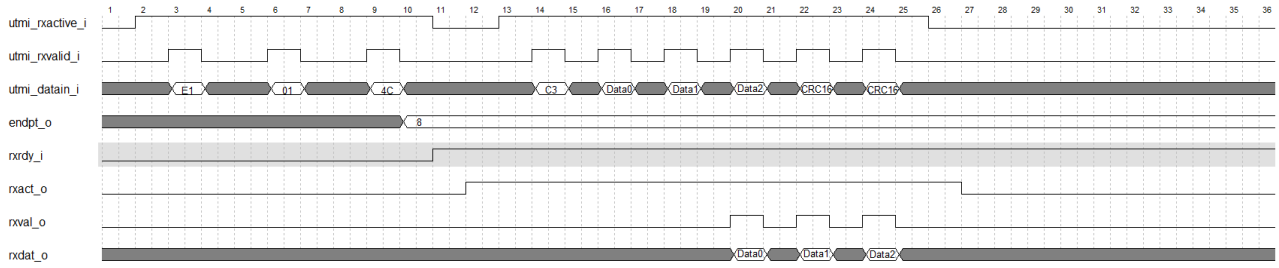
**Table 3-2 Device Controller RX User Interface**

| Name | I/O | Bit Width | Description |
|---|---|---|---|
| rxact_o | Output | 1 | Receiving Work Signal<br>High, indicates that the device is in data receive state. |
| endpt_o | Output | 4 | Endpoint Selection Indicator Signal<br>It indicates the current communication endpoint of USB. |
| rxdat_o | Output | 8 | Receiving Data<br>IP will output the data received from the USB interface |
| rxval_o | Output | 1 | Receive Valid Data Signal<br>High, indicates that RXDAT is valid. |
| rxrdy_i | Input | 1 | Receive Ready Signal<br>High, indicates that it can receive RXDAT. |

When Device receives OUT command transmitted by Host, Device Controller set rxact_o to 1. It indicates the device is in receive state and endpt_o outputs the endpoint address. After rxact_o is set, you should check the data receive ability of corresponding endpoint. If the endpoint can receive new data, set rxrdy_i to 1. Device Controller transmits data to users through rxdat_o and rxval_o. If the endpoint cannot receive new data,

set rxrdy_i to 0. Device Controller will not output data. Device data receive timing Diagram is as follow.

**Figure 3-3 USB Device Controller Data Receive Timing Diagram**




# 3.4 USB Device Descriptor Interface

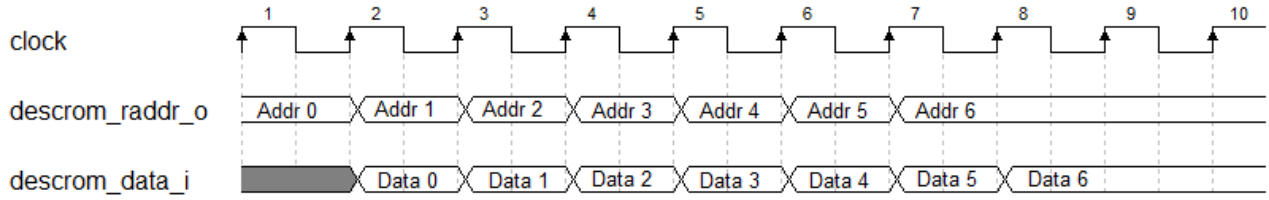
**Table 3-3 Device Descriptor Interface**

| Name | I/O | Bit Width | Description |
|---|---|---|---|
| descrom_rdata_i | I | 8 | USB Device Descriptor Data |
| descrom_raddr_o | O | 16 | USB Device Descriptor Address |
| desc_index_o | O | 8 | USB Device Descriptor Index |
| desc_type_o | O | 8 | USB Device Descriptor Type |
| desc_dev_addr_i | I | 16 | Device Descriptor Start Address |
| desc_dev_len_i | I | 8 | Device Descriptor Byte Length |
| desc_qual_addr_i | I | 16 | Device Qualifier Start Address |
| desc_qual_len_i | I | 8 | Device Qualifier Byte Length |
| desc_fscfg_addr_i | I | 16 | Device Full Speed Configuration Start Address |
| desc_fscfg_len_i | I | 8 | Device Full Speed Configuration Byte Length |
| desc_hscfg_addr_i | I | 16 | Device High Speed Configuration Start Address |
| desc_hscfg_len_i | I | 8 | Device High Speed Configuration Byte Length |
| desc_oscfg_addr_i | I | 16 | Device Other Speed Configuration Start Address |
| desc_strlang_addr_i | I | 16 | Device String Descriptor Start Address |
| desc_strvendor_addr_i | I | 16 | Device Vendor String Start Address |
| desc_strvendor_len_i | I | 8 | Device Vendor String Byte Length |

| Name | I/O | Bit Width | Description |
|---|---|---|---|
| desc_strproduct_addr_i | I | 16 | Device Product String Start Address |
| desc_strproduct_len_i | I | 8 | Device Product String Byte Length |
| desc_strserial_addr_i | I | 16 | Device Serial String Start Address |
| desc_strserial_len_i | I | 8 | Device Serial String Byte Length |
| desc_hidrpt_addr_i_ | I | 16 | USB Device DescriptorHID Start Address |
| desc_hidrpt_len_i | I | 16 | USB Device DescriptorHID Byte Length |
| desc_have_strings_i | I | 1 | High, indicates that string descriptors exist in the device descriptor. |

USB Device Controller IP provides device descriptor data input interface, which mainly includes Device Descriptor, Device Qualifier, Device Full Speed Configuration, Device High Speed Configuration, Device Other Speed Configuration, Device String Descriptor, Device Vendor String, Device Product String, Device Serial String; desc_index_o and desc_type_o are used to help distinguish between control descriptors and string descriptors that need to be read. You need to provide the start address and byte length of each part according to storage location of its own device descriptor data, and the data of each part should be saved on consecutive addresses. Based on the address and length provided by the user, the Controller outputs the descriptor data read on descrom_raddr_o after receiving the device descriptor read request transmitted by Host. And the user returns the corresponding descriptor data on descrom_rdata_i. The USB Device Controller IP parameters, Self Power and Descriptor Max Packet Size, should be in line with the settings within the user device descriptor. Device descriptor data read timing diagram is as follow.

**Figure 3-4 USB Device Controller Descriptor Read Timing Diagram**



# 3.5 USB Device Control Interface

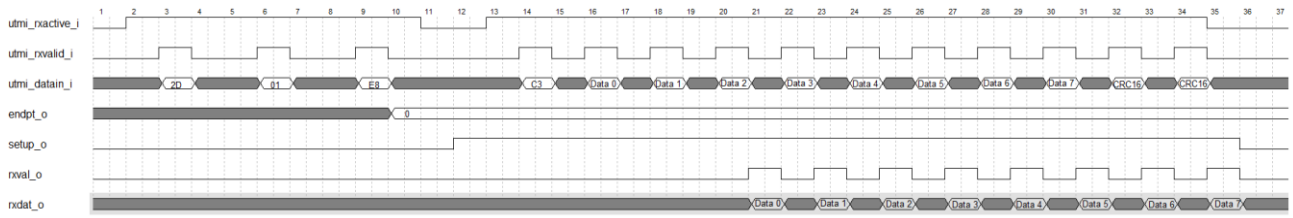
## 3.5.1 Control Data Interface

**Table 3-4 Control Interface**

| Name | I/O | Bit Width | Description |
|---|---|---|---|
| setup_o | Output | 1 | Active Data Indicator Signal<br>High, indicates that USB configuration data is in active state. |
| rxval_o | Output | 1 | Valid Data Indicator Signal<br>High, indicates thatUSB configuration data is valid. |
| rxdat_o | Output | 8 | Receiving Data<br>IP will output the data received from the USB interface |

Device Controller IP has implemented most of the USB control endpoint functions internally, but for different users in different scenarios, it is difficult to cover all the configuration functions, so the user configuration function is reserved. When Device receives the configuration data packet transmitted by Host, Controller will output the configuration data to users through setup_o, rxval_o, rxdat_o. Device configuration data receive timing diagram is as follow.

**Figure 3-5 USB Device Controller Control Data Receive Timing Diagram**



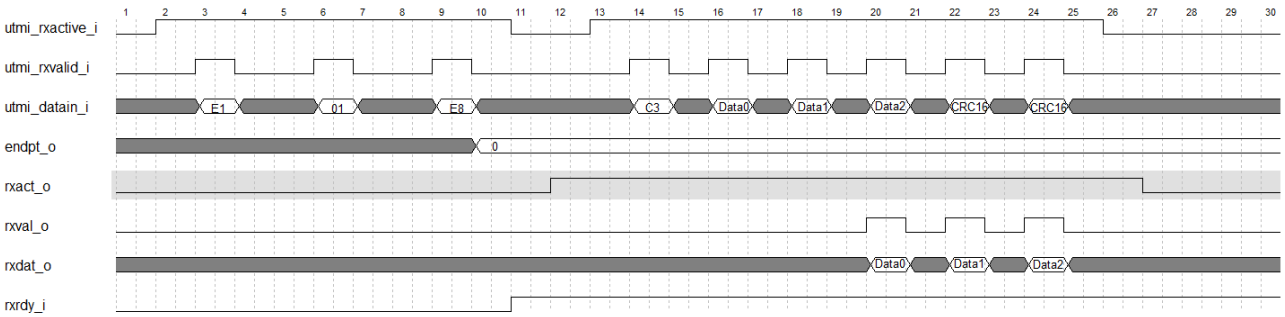
## 3.5.2 Configuration Data RX Interface

**Table 3-5 Configuration Data RX Interface**

| Name | I/O | Bit Width | Description |
|---|---|---|---|
| rxact_o | Output | 1 | Receiving Work Signal<br>High, indicates that the device is in data receive state. |
| rxval_o | Output | 1 | Valid Data Indicator Signal<br>High, indicates thatUSB configuration data is valid. |
| rxdat_o | Output | 8 | Receiving Data<br>IP will output the data received from the USB interface |
| rxrdy_i | Input | 1 | Receive Ready Signal<br>High, indicates that it can receive RXDAT. |
| endpt_o | Output | 4 | Endpoint Selection Indicator Signal<br>It indicates the current communication endpoint of USB. |

After Device receives the configuration data packet transmitted by Host, if the configuration data packet indicates that Host will write configuration data to Device, and the configuration data length is not 0, then in the subsequent Host OUT command, you can receive the configuration data through the RX interface. The device configuration data write timing diagram is as follow.

**Figure 3-6 USB Device Controller Control Endpoint Data Receive Timing Diagram**



### 3.5.3 Configuration Data TX Interface

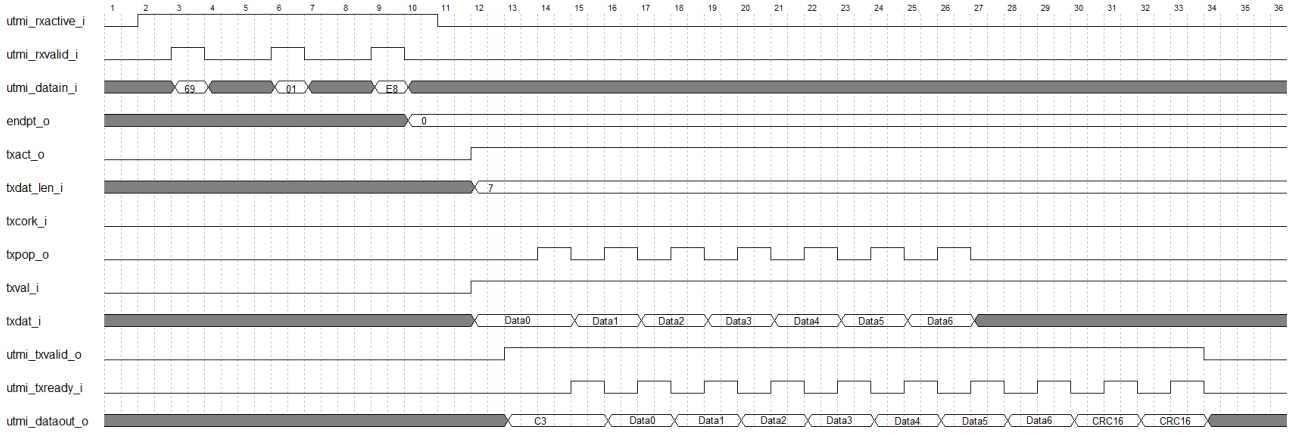
**Table 3-6 Configuration Data TX Interface**

| Name | I/O | Bit Width | Description |
|---|---|---|---|
| txact_o | Output | 1 | Transmitting Work Signal<br>High, indicates that the device is in data transmit state. |
| txdat_i | Input | 8 | Transmitting Data<br>IP transmits this data through the USB interface. |
| txval_i | Input | 1 | Transmitting Valid Data Indicator Signal<br>High, indicates the data users input is valid. |
| txdat_len_i | Input | 12 | The Number of Bytes of Transmitting Data<br>It can be used to control the number of bytes of TX data. |
| txcork_i | Input | 1 | Transmitting Valid Data Indicator Signal<br>Low, indicates that TXDAT is insufficient. |
| txpop_o | Output | 1 | Transmitting Read Signal<br>High, indicates that read next data. |
| endpt_o | Output | 4 | Endpoint Selection Indicator Signal<br>It indicates the current communication endpoint of USB. |

After Device receives the configuration data packet transmitted by Host, if the configuration data packet indicates that the configuration data can be read from Device, you need to make configuration data ready. Then in the subsequent Host IN command, you can transmit configuration data through TX interface.

If txval_i is set to 1, transmit configuration data through TX; If it is set to 0, return data to Host through IP internal logic. Configuration data transmit timing diagram is as follow.

**Figure 3-7 USB Device Controller Control Endpoint Data Transmit Timing Diagram**




## 3.5.4 Interface for Interface Configuration

**Table 3-7 Inferface for Interface Configuration**

| Name | I/O | Bit Width | Description |
|---|---|---|---|
| inf_alter_i | Input | 8 | USB interface configuration data input |
| inf_alter_o | Output | 8 | USB interface configuration data output |
| inf_sel_o | Output | 8 | USB interface configuration selection signal |
| inf_set_o | Output | 1 | USB interface configuration update indicator, active-high. |

After the Device receives the Set Interface instruction from the Host, inf_sel_o outputs the interface signal to be configured; inf_alter_o outputs the replacement of the interface, and set inf_set_o to a high level.

After the Device receives the Get Interface instruction from the Host, inf_sel_o outputs the interface signal to be read, and then reads the configuration information on inf_alter_i and returns it to the Host.

# 4 Signal Description

## 4.1 Signal Description

The descriptions of Gowin USB 2.0 Device Controller IP signals are as shown in Table 4-1.

**Table 4-1 Signal Description**

| No. | Signal Name | I/O | Data Width | Description | Note |
|---|---|---|---|---|---|
| 1 | clk_i | I | 1 | Input the clock signal transmitting by PHY, and the frequency must be 60MHz. | |
| 2 | reset_i | I | 1 | Asynchronous reset signal | |
| 3 | usbrst_o | O | 1 | USB Reset Indicator Signal<br>High indicates the detection of USB bus reset. | |
| 4 | highspeed_o | O | 1 | HS Indicator Signal<br>High indicates that the device is in HS mode. | |
| 5 | suspend_o | O | 1 | Suspended Indicator Signal<br>High indicates that the device is suspended. | |
| 6 | online_o | O | 1 | Configuration Indicator Signal<br>High indicates that the device is in configuration state. | |
| 7 | txact_o | O | 1 | Transmitting Work Signal<br>High indicates that the device is in data transmit state. | |
| 8 | txdat_i | I | 8 | Transmitting Data<br>IP transmits this data through the | |

| No. | Signal Name | I/O | Data Width | Description | Note |
|---|---|---|---|---|---|
| | | | | USB interface. | |
| 9 | txval_i | I | 1 | Transmitting Valid Data Indicator Signal<br>High indicates that the data users input is valid (Only valid when control endpoint transmits data.) | |
| 10 | txdat_len_i | I | 12 | The Number of bytes of Transmitting Data<br>It can be used to control the number of bytes of TX data. | |
| 11 | txiso_pid_i | I | 4 | USB Pakcet ID input in synchronous transfer mode. | |
| 12 | txcork_i | I | 1 | Transmitting Valid Data Indicator Signal<br>Low indicates that TXDAT is insufficient. | |
| 13 | txpop_o | O | 1 | Transmitting Read Signal<br>High indicates that read next data. | |
| 14 | rxact_o | O | 1 | Receiving Work Signal<br>High indicates that the device is in data receive state. | |
| 15 | rxdat_o | O | 8 | Receiving Data<br>IP will output the data received from the USB interface. | |
| 16 | rxval_o | O | 1 | Receiving Valid Data Signal<br>High indicates that RXDAT is valid. | |
| 17 | rxrdy_i | I | 1 | Receive Ready Signal<br>High indicates that it can receive RXDAT. | |
| 18 | rxpktval_o | O | 1 | Valid Data Package Signal<br>High indicates that RX data package is valid. | |
| 19 | setup_o | O | 1 | Configuration Data Active Indicator Signal<br>High indicates that USB configuration data is in active state. | |
| 20 | endpt_o | O | 4 | Endpoint Selection Indicator Signal | |

| No. | Signal Name | I/O | Data Width | Description | Note |
|---|---|---|---|---|---|
| | | | | It indicates the communication endpoint that USB selects. | |
| 21 | sof_o | O | 1 | USB Frame Synchronization Signal | |
| 22 | inf_alter_i | I | 8 | USB interface configuration data output | |
| 23 | inf_alter_o | O | 8 | USB interface configuration data input | |
| 24 | inf_sel_o | O | 8 | USB interface selection | |
| 25 | inf_set_o | O | 1 | The USB interface configuration update, active-high. | |
| 26 | utmi_dataout_o | O | 8 | Transmits data to USB PHY. | |
| 27 | utmi_txvalid_o | O | 1 | Valid PHY_DATAOUT Data Indicator Signal | |
| 28 | utmi_opmode_o | O | 2 | Operation mode selection signal<br>2'b00: Normal<br>2'b01: No driver<br>2'b10: Disable bit stuffing and NRZI encoding<br>2'b11: Operation of not automatically generating start and end signals | |
| 29 | utmi_xcvrselect_o | O | 2 | Transmitting Mode Selection Signal:<br>2'b00: HS Transfer<br>2'b01: FS Transfer<br>2'b10: LS Transfer<br>2'B11: Reserved | |
| 30 | utmi_termselect_o | O | 1 | Termination Selection:<br>1' b0: HS termination enable<br>1' b1: FS termination enable | |
| 31 | utmi_reset_o | O | 1 | Output reset signal, active-high. | |
| 32 | utmi_datain_i | I | 8 | Receive data from USB PHY | |
| 33 | utmi_txready_i | I | 1 | Transmitting Data Ready Signal<br>It indicates that PHY can receive transmitting data. | |
| 34 | utmi_rxactive_i | I | 1 | Receive Active Signal | |
| 35 | utmi_rxvalid_i | I | 1 | Receive Enable Signal | |

| No. | Signal Name | I/O | Data Width | Description | Note |
|---|---|---|---|---|---|
| | | | | Valid PHY_DATAIN Indicator Signal | |
| 36 | utmi_rxerror_i | I | 1 | Receive Error Signal<br>High indicates that an error occurs on receive. | |
| 37 | utmi_linestate_i | I | 2 | Line Status of Receiver<br>2'b00：SE0<br>2'b01："J"<br>2'b10："K"<br>2'b11：SE1 | |
| 38 | descrom_rdata_i | I | 8 | USB Device Descriptor Data | USB Device Descriptor |
| 39 | descrom_raddr_o | O | 16 | USB Device Descriptor Address | |
| 40 | desc_index_o | O | 8 | USB Device Descriptor Index | |
| 41 | desc_type_o | O | 8 | USB Device Descriptor Type | |
| 42 | desc_dev_addr_i | I | 16 | Device Descriptor Start Address | |
| 43 | desc_dev_len_i | I | 16 | Device Descriptor Byte Length | |
| 44 | desc_qual_addr_i | I | 16 | Device Qualifier Start Address | |
| 45 | desc_qual_len_i | I | 16 | Device Qualifier Byte Length | |
| 46 | desc_fscfg_addr_i | I | 16 | Device Full Speed Configuration Start Address | |
| 47 | desc_fscfg_len_i | I | 16 | Device Full Speed Configuration Byte Length | |
| 48 | desc_hscfg_addr_i | I | 16 | Device High Speed Configuration Start Address | |
| 49 | desc_hscfg_len_i | I | 16 | Device High Speed Configuration Byte Length | |
| 50 | desc_oscfg_addr_i | I | 16 | Device Other Speed Configuration Start Address | |
| 51 | desc_strlang_addr_i | I | 16 | Device String Descriptor Start Address | |
| 52 | desc_strvendor_addr_i | I | 16 | Device Verdor String Start Address | |
| 53 | desc_strvendor_len_i | I | 16 | Device Verdor String Byte Length | |
| 54 | desc_strproduct_addr_i | I | 16 | Device Product String Start Address | |
| 55 | desc_strproduct_len_i | I | 16 | Device Product String Byte Length | |
| 56 | desc_strserial_addr_i | I | 16 | Device Serial String Start Address | |

| No. | Signal Name | I/O | Data Width | Description | Note |
|---|---|---|---|---|---|
| 57 | desc_strserial_len_i | I | 16 | Device Serial String Byte Length | |
| 58 | desc_hidrpt_addr_i_ | I | 16 | USB Device Descriptor HID Start Address | |
| 59 | desc_hidrpt_len_i | I | 16 | USB Device Descriptor HID Byte Length | |
| 60 | desc_bos_addr_i | I | 16 | USB Device Descriptor BOS Start Address | |
| 61 | desc_bos_addr_i | I | 16 | USB Device Descriptor BOS Byte Length | |
| 62 | desc_have_strings_i | I | 1 | High indicates that there are string descriptors in the device descriptor. | |
| 63 | ulpi_nxt_i | I | 1 | ULPI indication signal; high indicates that data has been successfully transmitted or received. | ULPI |
| 64 | ulpi_dir_i | I | 1 | ULPI direction signal; high: RX, low: TX | |
| 65 | ulpi_data_io | IO | 8 | ULPI bi-directional port | |
| 66 | ulpi_stp_o | O | 1 | ULPI stop signal; data TX and RX interrupt flag, active-high. | |

# 4.2 Parameter Configuration

The parameter configuration of Gowin USB 2.0 Device Controller IP is shown in Table 4-2.

**Table 4-2 Configuration Options**

| Options | Description |
|---|---|
| Interface | USB controller interface selection, including UTMI and ULPI; UTMI logic resource utilization is less than ULPI; ULPI IO is less than UTMI. |
| Speed Mode | USB controller speed modes: High Speed, Full Speed, Low Speed. |
| Self Powered | USB self powered function control |
| Descriptor Max Packet Size | USB Descriptor Max Packet Size: 8 bytes, 16 bytes, 32 bytes, 64 bytes. Low Speed mode only supports max packet size of 8 bytes. |
| Endpoint 1 Input Transfer Type | Endpoint transfer types: Bulk Transaction, |

| Options | Description |
| --- | --- |
| | Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 1 Output Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 2 Input Transfer Type | Endpoint transfer type: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 2 Output Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 3 Input Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 3 Output Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 4 Input Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 4 Output Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 5 Input Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 5 Output Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports |

| Options | Description |
| --- | --- |
| | Interrupt Transaction. |
| Endpoint 6 Input Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 6 Output Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 7 Input Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 7 Output Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 8 Input Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 8 Output Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 9 Input Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 9 Output Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 10 Input Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 10 Output Transfer | Endpoint transfer types: Bulk Transaction, |

| Options | Description |
| --- | --- |
| Type | Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 11 Input Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 11 Output Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 12 Input Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 12 Output Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 13 Input Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 13 Output Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 14 Input Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 14 Output Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |
| Endpoint 15 Input Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports |

| Options | Description |
| --- | --- |
| | Interrupt Transaction. |
| Endpoint 15 Output Transfer Type | Endpoint transfer types: Bulk Transaction, Isochronous Transaction, and Interrupt Transaction. Low Speed mode only supports Interrupt Transaction. |

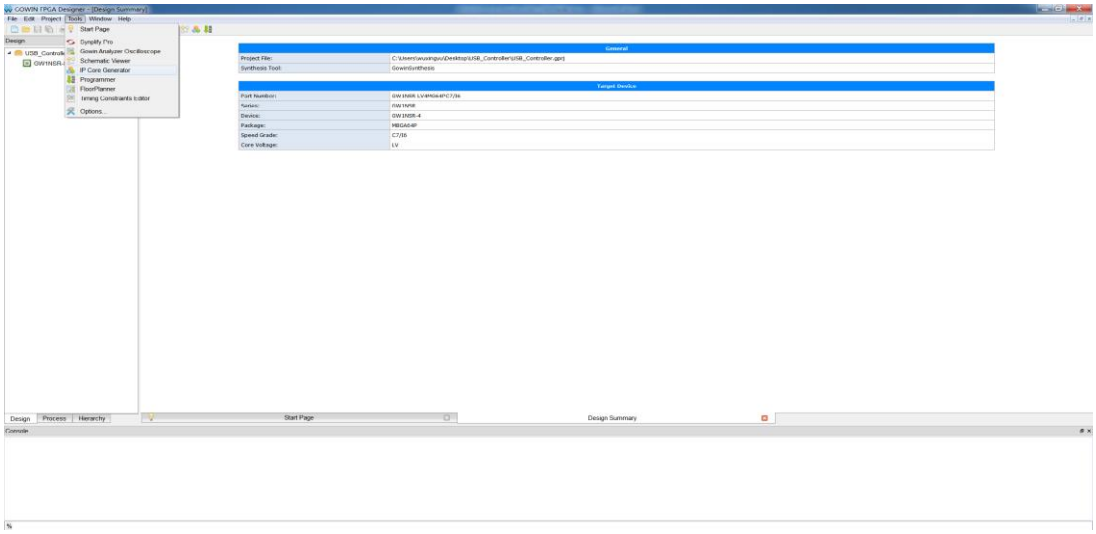
# 5 Interface Configuration

You can start the IP Core Generator via Tools on the menu bar, invoke and configure USB 2.0 Device Controller.

1. Open IP Core Generator

After creating the project, you can click the "Tools" tab in the upper left, select and open the IP Core Generator via the drop-down list, as shown in Figure 5-1.

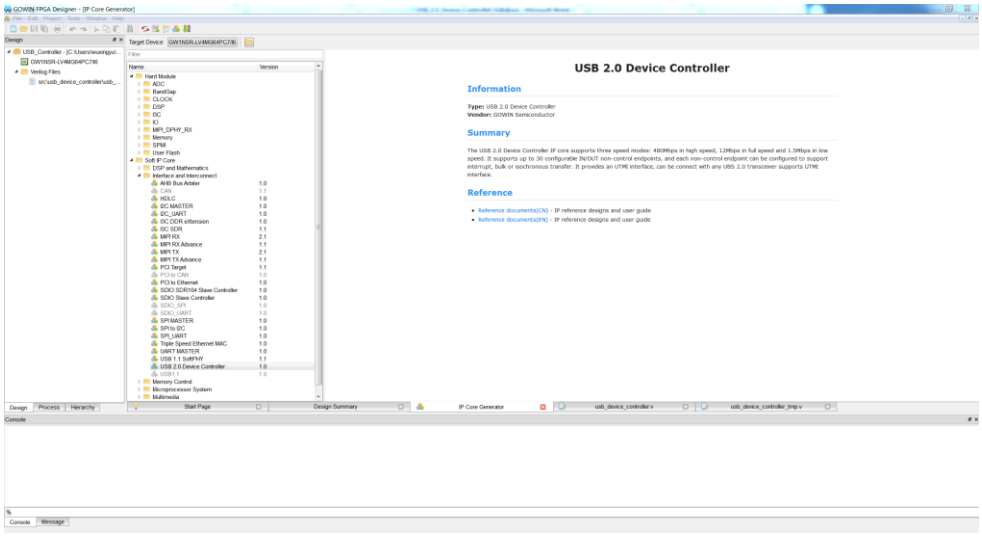
**Figure 5-1 IP Core Generator**



2. Open the USB 2.0 Device Controller IP core

Select "Soft IP Core > Interface and Interconnect > USB 2.0 IP", as shown in Figure 5-2. Double click to open the configuration interface.

**Figure 5-2 USB 2.0 Device Controller IP Core**

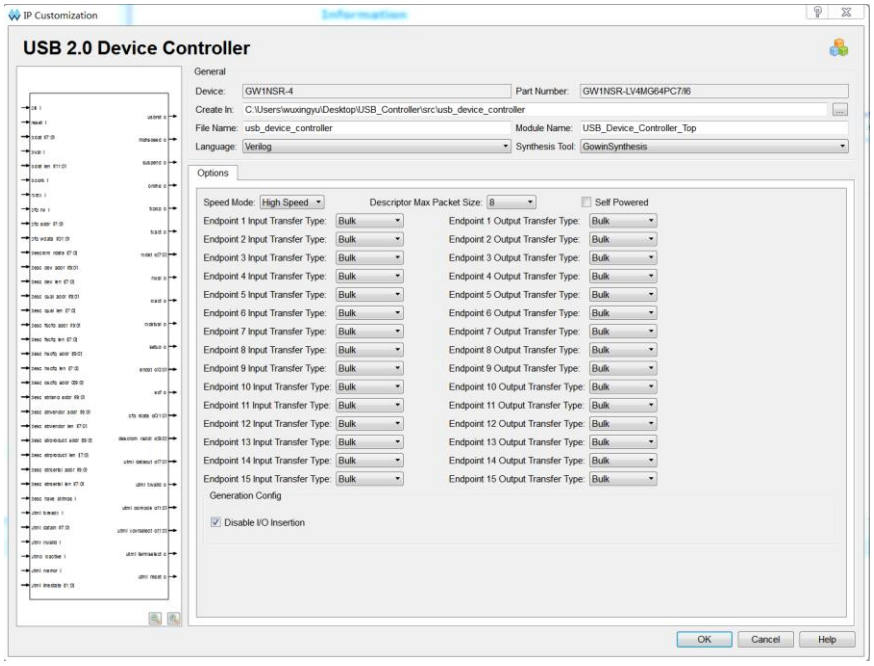



3. USB 2.0 Device Controller IP Core Configuration Interface

The USB 2.0 Device Controller IP core configuration interface is shown in Figure 5-3. The interface diagram is on the left. The options are on the right.

- You can configure the file name in File Name;
- You can configure the top module name in Module Name;
- You can configure the speed mode, power mode, and endpoint transfer type, etc. in Options.

**Figure 5-3 USB 2.0 Device Controller Configuration Interface**

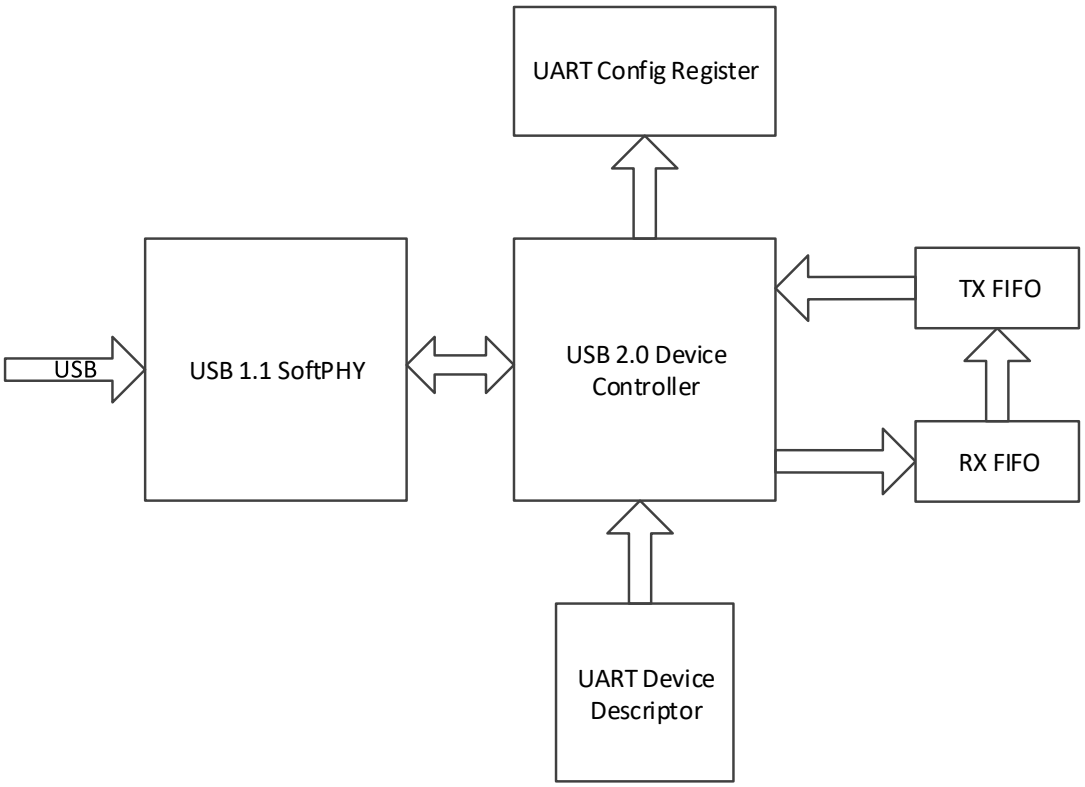


# 6 Reference Design

## 6.1 USB Reference Design

This chapter introduces the structure and usage of USB 2.0 Device Controller IP reference design instance. This reference design is a USB-to-UART data loopback design instance. Its basic structure is shown in Figure 6-1.

**Figure 6-1 USB 2.0 Device Controller Reference Design Block Diagram**




In this design instance, USB is converted into parallel UTMI signal through Gowin USB 1.1 SoftPHY IP module, and USB data packet parsing is completed through USB 2.0 Device Controller, UART Device Descriptor module contains USB device descriptor information internally, UART Config

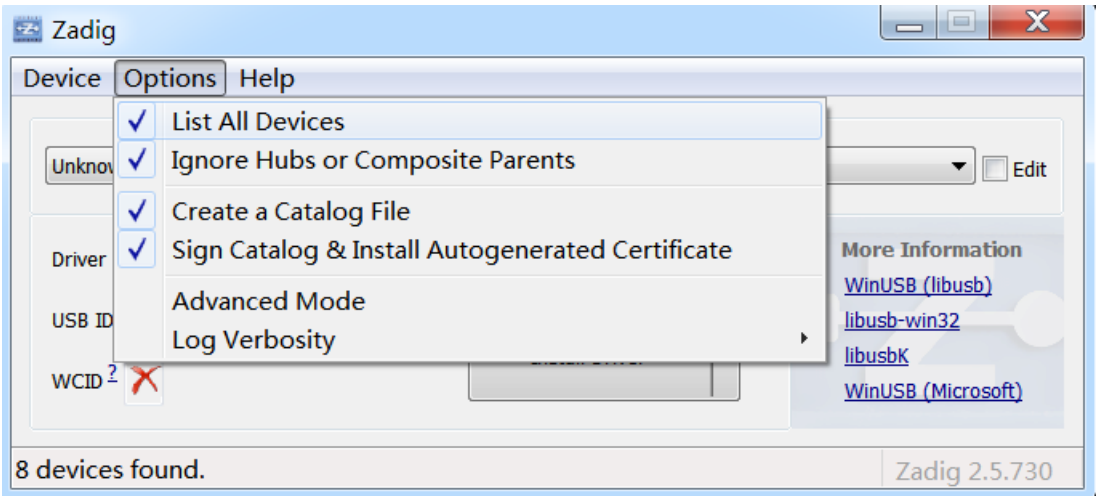
Register module receives configuration commands from USB Host to configure UART module parameters, TX/RX FIFO provides data buffer and loopback function.

# 6.2 USB CDC Driver Installation

You can install the CDC driver on the device through the zadig software as follows.

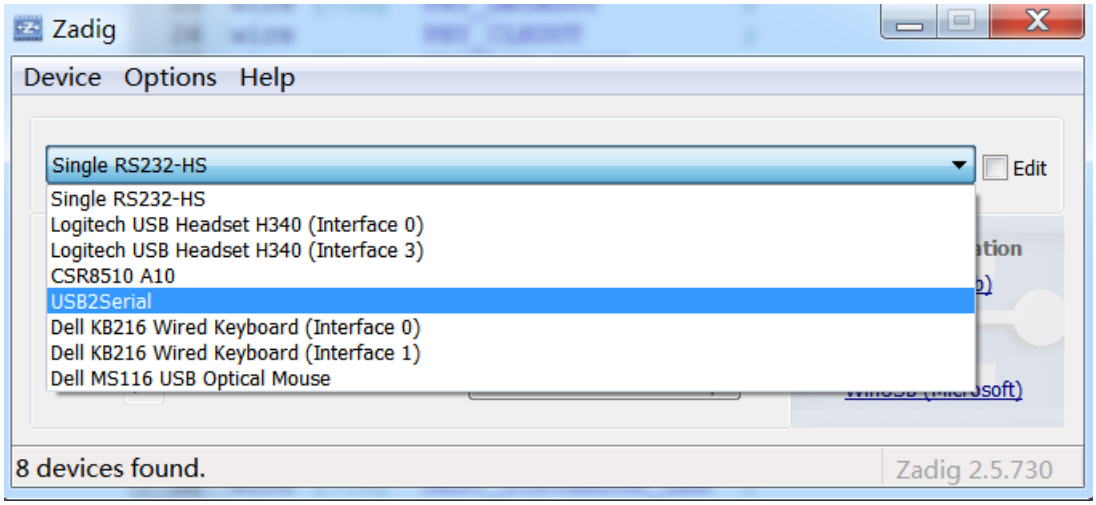
Open zadig ( Administrator permission) and select "List All Devices" in the Options drop-down menu.

**Figure 6-2 Search USB Device**




Select "USB2Serial" in the device drop-down list box.

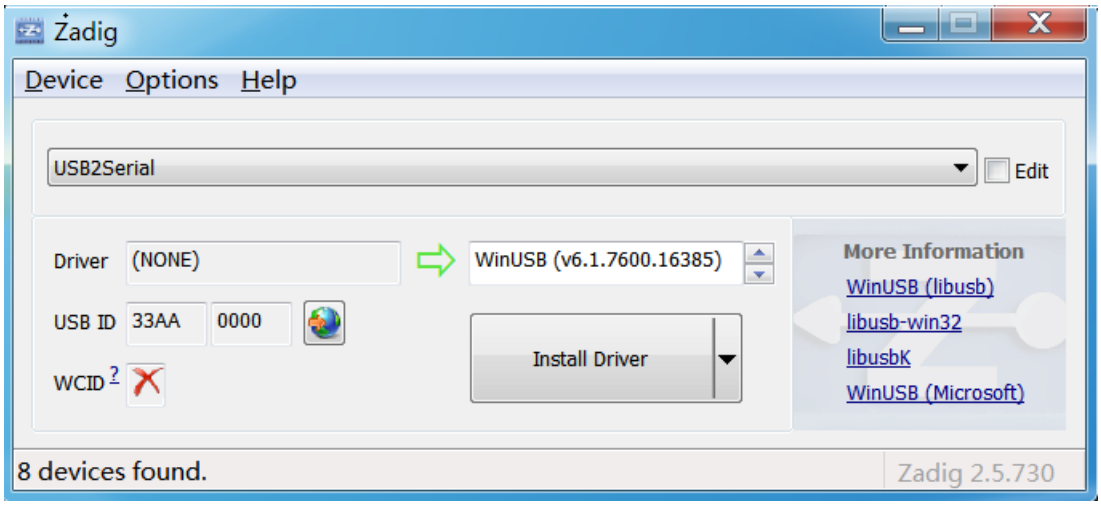
**Figure 6-3 Select USB Device**




After selecting "USB2Serial" device, install WinUSB driver first, then

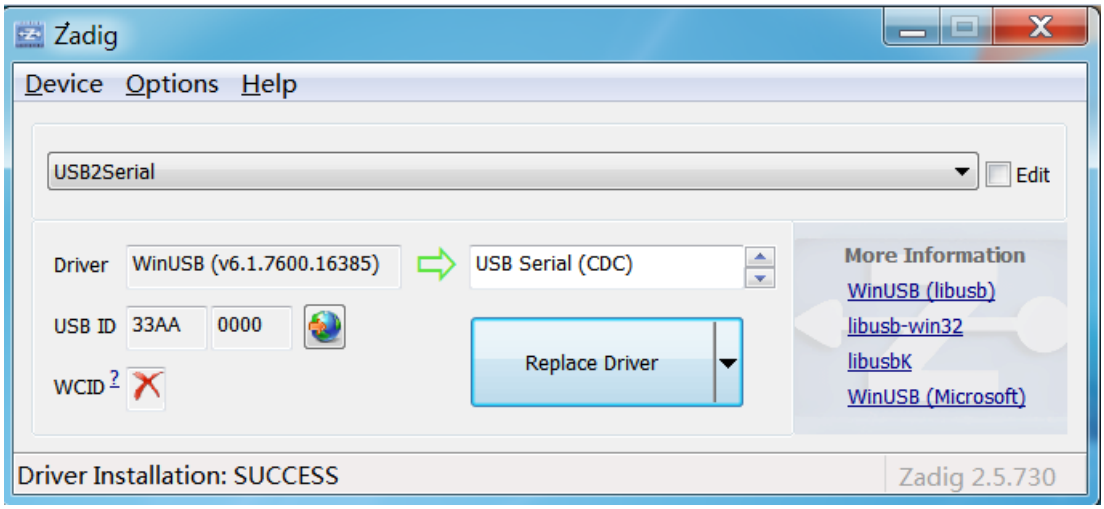
click Install Driver. Wait a few moments to complete the installation of the driver for this device interface.

**Figure 6-4 Zadig WinUSB Driver Installation**




After installing WinUSB driver, then install CDC driver and click "Replace Driver". After the CDC driver is successfully installed, you can control the device with software such as serial assistant.

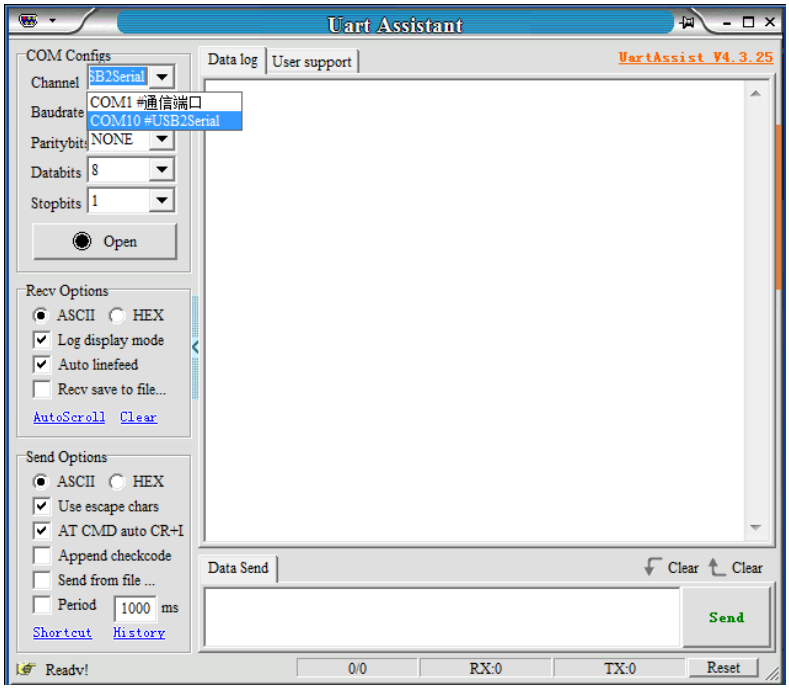
**Figure 6-5 Zadig CDC Driver Installation**



# 6.3 Data Transfer Test

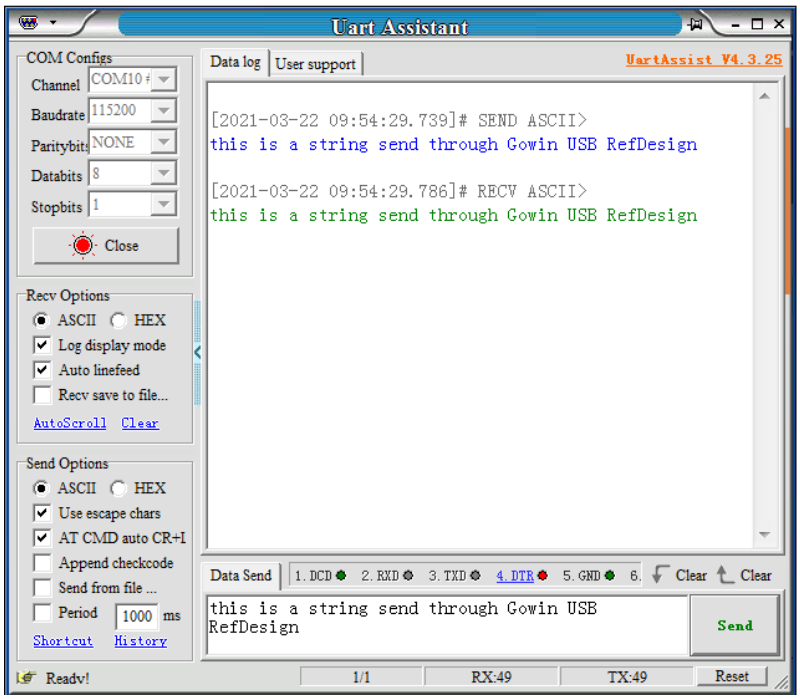
Open the serial debugging assistant software and select the serial port USB2Serial, as shown in the figure below.

**Figure 6-6 Select Serial Device in Serial Debugging Assistant**




Connect serial port TXD to RXD for serial data loopback test.

**Figure 6-7 Serial Data Loopback Test**




GOWIN
PROGRAMMING FOR THE FUTURE