



Venus Serial Port Mode Selection and DIO Linux Demo Applications User Manual

Venus SBC SBC with “Kaby Lake U” Processor

For Version 1.0.0 and later

Revision A.0

July 2025

Revision	Date	Comment
A.0	16/07/2025	Initial release

**FOR TECHNICAL SUPPORT
PLEASE CONTACT:**

support@diamondsystems.com

©Copyright 2025
Diamond Systems Corporation

www.diamondsystems.com

CONTENTS

1.	INTRODUCTION	3
2.	HARDWARE OVERVIEW	4
2.1	Description	4
2.2	Specifications	4
3.	GENERAL PROGRAMMING GUIDELINES	5
3.1	Initialization and exit function calls	5
3.2	Error handling	6
4.	VENUS SERIAL PORT MODE SELECTION APPLICATION API DESCRIPTION	7
4.1	VenusGPIOInitBoard	7
4.2	VenusGPIOUARTMode	8
4.3	VenusGPIORelease	9
5.	VENUS DIO APPLICATION API DESCRIPTION	10
5.1	VenusDIOInitBoard	10
5.2	SetLastError	11
5.3	GetLastError	12
5.4	GetErrorString	13
5.5	VenusDIOConfig	14
5.6	VenusDIOOutputByte	15
5.7	VenusDIOInputByte	16
5.8	VenusDIOOutputBit	17
5.9	VenusDIOInputBit	18
6.	COMMON TASK REFERENCE	19
6.1	Serial Port I/O Feature Overview	19
6.2	Digital I/O Feature Overview	19
6.3	Performing Serial port mode selection Operations	20
6.4	Performing Digital IO Operations	20
7.	INTERFACE CONNECTOR DETAILS	23
7.1	Venus Digital GPIO Connector – J15	23
8.	Build the DIO AND SERIALMODE APPLICATION	23
9.	STEPS TO RUN DIO APPLICATION	24
10.	Steps to Run Serial Mode Application	28

1. INTRODUCTION

This user manual contains all essential information about the Venus SBC serial port mode selection application and DIO Demo application programming guidelines and usage instructions. This manual also includes the Venus serial port mode selection application API descriptions and DIO Demo application API description with usage examples.

2. HARDWARE OVERVIEW

2.1 Description

Based on the "KabyLake" 7th generation Core i7/U series processor, **Venus** offers the highest available CPU performance in a small form factor rugged SBC with modest power consumption. It incorporates a full suite of rugged features such as soldered memory, latching connectors, a thicker PCB, and true -40/+85°C operating temperature, making it suitable for the most demanding vehicle applications.

High I/O density, multiple expansion sockets, rugged design, modest power consumption of 14W, and wide temperature operation combined to make Venus an extremely attractive option for applications requiring high CPU performance or ruggedness.

2.2 Specifications

- Intel "Kaby Lake" 7th Gen Core i7-7660U 2.8GHz
- 4GB DDR4 RAM soldered on board
- Expansion socket for up to 16GB additional / 20GB total RAM
- 2x Gigabit Ethernet
- 2x SATA 3.0 ports + mSATA socket
- VGA, HDMI & dual channel LVDS with 3 simultaneous independent display support
- 4x RS-232/422/485 ports
- 6x USB 2.0 ports
- 4x USB 3.0 ports
- HD audio
- 16 GPIO lines with 3.3/ 5V logic levels
- TPM module
- CSI camera serial interface
- 2x PCIe MiniCard sockets; one socket supports mSATA
- OneBank-Plus PCIe/104 + PCI-104 expansion socket
- +9-18VDC input voltage
- 14W power consumption typical
- 3.5 inch form factor: 5.75" x 4.0" (146mm x 102mm)
- -40°C to +85°C operating temperature
- Bottom-mounted heat spreader cooling

3. GENERAL PROGRAMMING GUIDELINES

3.1 Initialization and exit function calls

Serial port mode selection applications begin with the following functions and should be called in a sequence to initialize the GPIO's of Venus SBC. These functions should be called prior to Venus Serial mode selection functions.

- `VenusGPIOInitBoard ()`: This function initializes the Venus SBC GPIO pins to user space.

At the termination of the serial mode selection application the user should call `VenusGPIOFreeBoard ()` function to un-export the GPIO pins which is exported in `VenusGPIOInitBoard ()` function.

These function calls are important in initializing and to free the resources used by the software. Following is an example of the framework for an application.

```
#include "VenusSerialMode.h"

ERRPARAMS errorParams; // structure for returning error code and error string
int main(void) {

    Mode = GetMode();
    port = 0;
    if(VenusGPIOInitBoard(port) != NONE)
    {
        printf("Failed to get GPIO line\n");
        return -1;
    }
    if(VenusGPIOUARTMode(port, Mode) != NONE)
    {
        printf("Failed to select the UART Mode\n");
        return -1;
    }
}
```

In the example above, `DE_NONE` is macro defined in the included header file, "`VenusSerialMode.h`" and `VenusDIO.h`. In `VenusGPIOInitBoard ()` function argument will be sent based upon the Port selection. . In `VenusDIOInitBoard ()` function argument `DIO_I2C_SLAVE` is macro defined in `VenusDIO.c` which is I2C slave device ID

```
#include "VenusDIO.h"
#define DIO_I2C_SLAVE          0x22

if ( ( VenusDIOInitBoard (DIO_I2C_SLAVE) != DE_NONE) )
{
    GetLastError ( &errorParams );
    printf ( "Init error: %s %s\n",GetErrorString ( &errorParams ),
(errorParams.errstring) );
    return 0;
}
```

3.2 Error handling

A serial port mode selection application functions and DIO application functions provide a basic error handling mechanism that stores the last reported error in the software. If the application is not behaving properly, last error can be checked by calling the function `GetLastError_GPIO()`. This function takes an `ERRPARAMS` structure pointer as its argument.

Nearly all of the available functions in the Venus Serial Port mode selection application API and Venus DIO application API return a `BYTE` value upon completion. This value represents an error code that will inform the user as to whether or not the function call was successful. User should always check if the result returns a `DE_NONE` value (signifying that no errors were reported), as the code below illustrates:

```
ERRPARAMS errorParams;

if ( ( VenusDIOInitBoard (DIO_I2C_SLAVE) != DE_NONE) )
{
    GetLastError ( &errorParams );
    printf ( "Init error: %s %s\n",GetErrorString ( &errorParams ),
(errorParams.errstring) );
    return 0;
}
```

In this code snippet, the result of executing `VenusGPIOInitBoard ()` and `VenusDIOInitBoard ()` is stored and checked against the expected return value (`DE_NONE`). At any point of time, if a function does not complete successfully, an error code other than `DE_NONE` will be generated, and the current API function will be terminated.

4. VENUS SERIAL PORT MODE SELECTION APPLICATION API DESCRIPTION

4.1 VenusGPIOInitBoard

Function Definition

```
mDSCInt VenusGPIOInitBoard(mDSCInt port);
```

Function Description

Initializes the GPIO chip interface by opening the GPIO chip using the CHIP_NAME.

Function Parameters

Name	Description
port	port mDSCInt Reserved for future use (currently unused).

Return Value

0 on success.

-1 on failure (with error printed to stderr).

Usage Example

```
Port = 0 or 1
```

```
mDSCInt VenusGPIOInitBoard(mDSCInt port);
```

4.2 VenusGPIOUARTMode

Function Definition

```
mDSCInt VenusGPIOUARTMode(mDSCInt port, mDSCInt mode);
```

Function Description

Configures GPIO lines to set the UART mode (RS232, RS422, RS485) for the specified port by toggling the relevant GPIO pins.

Function Parameters

Name	Description		
Port	Port	mDSCInt	UART port index (0 or 1).
mode	Mode	mDSCInt	UART mode: 0 – RS232 1 – RS422 2 – RS485

Return Value

0 on success.

-1 on failure (with error details printed).

Usage Example

Port - 0 or 1

Mode - 0 - RS232; 1 - RS422; 2 - RS485

```
mDSCInt VenusGPIOUARTMode(mDSCInt port, mDSCInt mode);
```

4.3 VenusGPIORelease

Function Definition

```
void VenusGPIORelease(int port);
```

Function Description

Releases all GPIO lines associated with the specified UART port.

Function Parameters

Name	Description
Port	Port void UART port index (0 or 1).

Return Value

void – No return value; logs errors if GPIO line access fails.

Usage Example

```
Port =1;
```

```
void VenusGPIORelease(int port);
```

5. VENUS DIO APPLICATION API DESCRIPTION

5.1 VenusDIOInitBoard

Function Definition

BYTE VenusDIOInitBoard (int addr);

Function Description

Initializes the Venus board and configures I2C communication.

Function Parameters

Name	Description
addr	addr int Pass the slave address.

Return Value

DE_NONE on success, or error code (DE_INVALID_PARM, HW_FAILURE, SW_FAILURE) on failure.

Usage Example

```
addr= I2C Slave address;  
BYTE VenusDIOInitBoard (int addr);
```

5.2 SetLastError

Function Definition

BYTE SetLastError(BYTE error_code, const char* error_str)

Function Description

Sets the last known error code and message.

Function Parameters

Name	Description
error_code	error_code BYTE Error code to set.
error_str	error_str const char* Corresponding error string

Return Value

The same error_code.

Usage Example

error_code - the error code to set
error_str - the error string to set

```
BYTE SetLastError(BYTE error_code, const char* error_str);
```

5.3 GetLastError

Function Definition

BYTE GetLastError(ERRPARAMS* errparams)

Function Description

Retrieves the last error code and string.

Function Parameters

Name	Description
errparams	errparams ERRPARAMS* Pointer to <i>ERRPARAMS</i> structure.

Return Value

DE_NONE

Usage Example

errparams - structure containing error code and string info

```
BYTE GetLastError(ERRPARAMS* errparams);
```

5.4 GetErrorString

Function Definition

Char* GetErrorString(ERRPARAMS* errparams)

Function Description

Returns a string for the given error code.

Function Parameters

Name	Description
errparams	errparams ERRPARAMS* Pointer to <i>ERRPARAMS</i> with ErrCode set.

Return Value

Error string for a given error code

Usage Example

errparams = structure containing error code and string info

```
Char* GetErrorString(ERRPARAMS* errparams);
```

5.5 VenusDIOConfig

Function Definition

BYTE VenusDIOConfig (UINT port, UINT dir)

Function Description

Sets DIO port directions.

Function Parameters

Name	Description
port	port UINT Port (0 or 1)
dir	dir UINT Direction byte (0–255)

Return Value

DE_NONE, or error code on failure.

Usage Example

```
port = 0 or 1  
dir = 0x00 - Input mode; 0xFF - Output mode;
```

```
BYTE VenusDIOConfig (UINT port, UINT dir);
```

5.6 VenusDIOOutputByte

Function Definition

BYTE VenusDIOOutputByte (UINT port, UINT data)

Function Description

Writes a byte to the specified DIO port.

Function Parameters

Name	Description		
port	port	UINT	0 or 1
Data	data	UINT	Byte value to write

Return Value

DE_NONE, or error code.

Usage Example

port - 0 or 1

data - 0x00 - Low; 0xFF - High;

```
BYTE VenusDIOOutputByte (UINT port, UINT data);
```

5.7 VenusDIOInputByte

Function Definition

BYTE VenusDIOInputByte (UINT port, UINT* data)

Function Description

This function frees the I2C bus device and it so no longer available for communication.

Function Parameters

Name	Description		
port	port	UINT	0 or 1
data	data	UINT	Pointer to store the read value

Return Value

DE_NONE, or error code.

Usage Example

```
port = 0 or 1
data = 0x00 - Low; 0xFF - High;

BYTE VenusDIOInputByte (UINT port, UINT* data);
```


5.8 VenusDIOOutputBit

Function Definition

BYTE VenusDIOOutputBit (UINT port, UINT bit, UINT value);

Function Description

Sets a single bit on a DIO port.

Function Parameters

Name	Description		
port	port	UINT	0 or 1
bit	bit	UINT	Bit number (0–7)
value	Value	UINT	0 or 1

Return Value

DE_NONE, or error code.

Usage Example

port - 0 or 1

value = 0 - Low; 1 - High

bit = 0 - 7 bit number

```
BYTE VenusDIOOutputBit (UINT port, UINT bit, UINT value);
```

5.9 VenusDIOInputBit

Function Definition

BYTE VenusDIOInputBit (UINT port, UINT bit, UINT* value)

Function Description

Reads a single bit from a DIO port.

Function Parameters

Name	Description		
port	port	UINT	0 or 1
bit	bit	UINT	Bit number (0–7)
value	value	UINT	Pointer to store result (0 or 1)

Return Value

DE_NONE, or error code.

Usage Example

```
port - 0 or 1
value = 0 - Low; 1 - High;
bit    = 0 - 7 bit number
```

```
BYTE VenusDIOInputBit (UINT port, UINT bit, UINT* value)
```

6. COMMON TASK REFERENCE

6.1 Serial Port I/O Feature Overview

I/O Connectors

The Venus SBC provides 1 I/O connectors for the attachment of serial ports. There are 4 ports available. Diamond's cable numbers 6980500 for serial port be used to connect loopback. These cables comes as part of the Venus cable kit, part number CK-VNS-01/02.

6.2 Digital I/O Feature Overview

I/O Connectors

The Venus SBC provides 1 I/O connectors for the attachment of digital I/O signals. Diamond's cable numbers 6980517 for digital I/O may be used to connect the user's signals to these connectors. These cables comes as part of the Venus cable kit, part number CK-VNS-01/02.

Digital I/O signals

There are 16 digital I/O signals, divided into two groups as DIOA0-DIOA7, DIOB0-DIOB7. Ports A-B are on I/O connector J15 pins 2-17.

Digital I/O signals use 3.3 V signaling only. Each signal's direction is independently programmable. On system startup or reset, all signals are automatically set to input mode.

Digital I/O Operations

The DIO application provides various operations on DIO channel i.e. input byte, output byte, input bit, output bit and DIO loopback. This section describes about input byte and output byte DIO operation. The DIO port must be configured in either input or output mode based on DIO operation need to be performed.

Output Byte:

- Select Write a Byte to port option from main menu □ Enter port number (0-1):
- Enter value 0-255 or q to quit

To set Port 0 all the pins to high except pin 3 and pin 7, run the DIO application and provide input as follows:

- Select Write a Byte to port option from main menu: 3
- Enter port number (0-1): 0
- Enter value 0-255 or q to quit: 119 The Byte value 119 is sent to port 0.

Measure the voltage on Port 0 all the pins using a multi-meter. It should show 3.3/5V on all the pins except pin 3 and pin 7, the application generated expected voltage.

Input Byte:

- Select Read a Byte from port option from main menu:
- Enter port number (0-1):
- Press ENTER key to stop reading ...

To provide 3.3V to Port 0 pin 0 from VCC and it should read and display 0x01. To see the output, Run DIO application and provide input as follows

- Select Read a Byte from port option from main menu:1
- Enter port number (0-1):0

The application should show 0x01 on the screen.

6.3 Performing Serial port mode selection Operations

The serial port mode selection application supports many combination. The three Venus serial port mode selection application using the below function, **Input parameters:**

The following input parameters can be used in the application

Port 1 – UART 1& 2 , 2- UART 3 & 4

Mode 0 – RS232, 1 - RS485, 2 – RS422

6.4 Performing Digital IO Operations

The DIO application supports four types of direct digital I/O operations: input bit, input byte, output bit, and output byte. Digital I/O is fairly straightforward - to perform digital input, provide a pointer to the storage variable and indicate the port number and bit number if relevant. To perform digital output, provide the output value and the output port and bit number, if relevant.

Step-By-Step Instructions

If digital input is being performed, create and initialize a pointer to hold the returned value of type byte*.

Some boards have digital I/O ports with fixed directions, while others have ports with programmable directions. Make sure the port is set to the required direction, input or output. Use function VenusDIOConfig () to set the direction.

Call the selected digital I/O function. Pass it an int port value, and either a pointer to or a constant byte digital value. If you are performing bit operations, then you will also need to pass in an int value telling the software which particular bit (0-7) of the DIO port you wish to operate on.

Example Usage for Digital I/O Operations

```
/* 1. Configure Port 0 in output mode */

UINT config =0x00;
    UINT port = 0;
    BYTE output_byte = 0;

if( (VenusDIOConfig (port, config) != DE_NONE) )
{
    GetLastError ( &errorParams );
    printf ( "VenusDIOConfig() error: %s %s\n",GetErrorString ( &errorParams ),
errorParams.errstring );
    return ;
}

if ( (VenusDIOOutputByte (port, output_byte) != DE_NONE) )
{
    GetLastError ( &errorParams );
    printf ( "VenusDIOOutputByte() error: %s %s\n",GetErrorString (
&errorParams ), errorParams.errstring );
    return ;
}
```

```
/* 2. input bit - read bit 6 (port 0 is in input mode) */

UINT config = 0xFF; //Configure port as Input direction
UINT port = 0;
UINT input_byte = 0;

printf ("Enter port number (0-1):");
fgets ( input_buffer, 20, stdin );
sscanf ( input_buffer, "%d", &port );

if( (VenusDIOConfig (port, config) != DE_NONE) )
{
    GetLastError ( &errorParams );
    printf ( "VenusDIOConfig() error: %s %s\n",GetErrorString ( &errorParams ),
errorParams.errstring );
    return ;
}

if( (VenusDIOInputByte (port, &input_byte) != DE_NONE) )
{
    GetLastError ( &errorParams );
    printf ( "VenusDIOInputByte() error: %s %s\n",GetErrorString (
&errorParams ),errorParams.errstring );
    return ;
}

/* 3. input byte - read all 8 bits of port 0 (port 0 is in input mode) */
UINT config = 0xFF; //Configure port as read direction
UINT port = 0;
UINT digital_value = 0; // digital representation of the bit
int bit = 0;
if( (VenusDIOConfig (port, config) != DE_NONE) )
{
    GetLastError ( &errorParams );
    printf ( "VenusDIOConfig() error: %s %s\n",GetErrorString ( &errorParams ),
errorParams.errstring );
    return ;
}
for(bit=7 ;bit>=0; bit--)
{
    if ( (VenusDIOInputBit (port, bit, &digital_value) != DE_NONE) )
    {
        GetLastError ( &errorParams );
        printf ( "VenusDIOInputBit() error: %s %s\n",GetErrorString (
&errorParams ), errorParams.errstring );
        return ;
    }

    printf("\t%x",digital_value);
}
```

```
    }

/* 4. output bit - set the bit 6 of port 1 (assumes port 1 is in output mode) */
UINT config = 0x00; //Configure port as output direction 1- Input, 0- Output
UINT port = 1;
UINT digital_value = 1;
UINT bit = 6;

VenusDIOConfig (port, config);

If ((VenusDIOOutputBit (port, bit, digital_value)!= DE_NONE) )
{
    GetLastError(&errorParams); printf ( "VenusDIOOutputBit() error: %s
    %s\n",GetErrorString( & errorParams), errorParams.errstring ); return;
}

/* 5. Output byte - set the port 1 to "0xFF" (assumes port 1 is in output mode) */
UINT config = 0x00; //Configure port as output direction 1- Input, 0- Output
UINT port = 1;
UINT output_byte = 0xFF;

VenusDIOConfig (port, config);

if (VenusDIOOutputByte (port, output_byte)!= DE_NONE)
{
    GetLastError(&errorParams); printf ( "VenusDIOOutputByte() error: %s
    %s\n",GetErrorString(&errorParams), errorParams.errstring ); return;
}
```

7. INTERFACE CONNECTOR DETAILS

7.1 Venus Digital GPIO Connector – J15

Venus provides three digital I/O ports with 8 lines on port A and B. Port A and B are provided on connector J15

VENUS Digital GPIO Connector Pin out

J15			
VCC (3.3 V)	1	2	DIO A0
DIO A1	3	4	DIO A2
DIO A3	5	6	DIO A4
DIO A5	7	8	DIO A6
DIO A7	9	10	DIO B0
DIO B1	11	12	DIO B2
DIO B3	13	14	DIO B4
DIO B5	15	16	DIO B6
DIO B7	17	18	
GND	19	20	GND

8. Build the DIO AND SERIALMODE APPLICATION

```
dscgquest@dscgquest-Default-string:~/VENUS_DEMOS/VenusDemo$ ls
BuildAll  Common  Executables  VENUSDIO  VenusSerialportConfig
dscgquest@dscgquest-Default-string:~/VENUS_DEMOS/VenusDemo/BuildAll$ sudo make clean
[sudo] password for dscgquest:
rm -f ../VenusSerialportConfig/VenusSerialMode
rm -f ../VENUSDIO/VENUSDIO
rm ../Executables/* -rf
dscgquest@dscgquest-Default-string:~/VENUS_DEMOS/VenusDemo/BuildAll$ sudo make
gcc -o ../VenusSerialportConfig/VenusSerialMode ../VenusSerialportConfig/VenusSerialMode.c -L ../Common/ -lvenusgpio -lm -lpthread -lgpio -I ../Common/
cp ../VenusSerialportConfig/VenusSerialMode ../Executables
gcc -o ../VENUSDIO/VENUSDIO ../VENUSDIO/VENUSDIO.c -L ../Common/ -lvenusdio_01.01.000 -lm -lpthread -I ../Common/
cp ../VENUSDIO/VENUSDIO ../Executables
```

9. STEPS TO RUN DIO APPLICATION

Step 1:

The **cd** (change directory) command in Linux is used to navigate between directories in the filesystem.

```
root@dscguest-Default-string:/home/dscguest/Downloads# cd VENUSDIO/
root@dscguest-Default-string:/home/dscguest/Downloads/VENUSDIO# ls
libvenusdio_01.01.000.a Makefile VENUSDIO VENUSDIO.c VenusDIO.h
root@dscguest-Default-string:/home/dscguest/Downloads/VENUSDIO#
```

Step 2: Run the Application

./VENUSDIO is a command used to execute the compiled binary named VENUSDIO from the current directory in a Linux environment.

```
root@dscguest-Default-string:/home/dscguest/Downloads/VENUSDIO# ./VENUSDIO

VENUS DIO FUNCTIONS DEMO

DIO INPUT AND OUTPUT: Main Menu
#####
1) Read Byte from port
2) Read all bits from port
3) Write Byte to port
4) Write Bit to a port
q) Quit Program
#####
```

Explanation of the command:

./- Specifies the current directory, ensuring the shell looks for the executable in the current location
VENUSDIO — The compiled application, built from VENUSDIO.c

Step 3: Read a Byte from the port

Allows the user to read a byte value from a specified digital input/output port. This would be used to get the current state of 8 digital lines.

```
root@dscguest-Default-string:/home/dscguest/Downloads/VENUSDIO# ./VENUSDIO

VENUS DIO FUNCTIONS DEMO

DIO INPUT AND OUTPUT: Main Menu
#####
1) Read Byte from port
2) Read all bits from port
3) Write Byte to port
4) Write Bit to a port
q) Quit Program
#####
1
Enter port number (0-1):0
Byte value received from port 0 = 0x20
#####
```

Step 4: Read all bits from the port

Allow the user to read the individual state of each digital bit (0 or 1) from a port. This would be used to get the current state of 8 digital lines each.

```
root@dscguest-Default-string:/home/dscguest/Downloads/VENUSDIO# ./VENUSDIO

VENUS DIO FUNCTIONS DEMO

DIO INPUT AND OUTPUT: Main Menu
#####
1) Read Byte from port
2) Read all bits from port
3) Write Byte to port
4) Write Bit to a port
q) Quit Program
#####
2
Enter port number (0-1):0
The port 0 each bit values are
-----
BitNo:  7      6      5      4      3      2      1      0
-----
Value:  0      0      1      0      0      0      0      0
-----
#####
```

Step 5: Write a Byte to the port.

Allow the user to write a byte value to a digital output port. This effectively sets the state of 8 digital lines simultaneously.

```
root@dscguest-Default-string:/home/dscguest/Downloads/VENUSDIO# ./VENUSDIO

VENUS DIO FUNCTIONS DEMO

DIO INPUT AND OUTPUT: Main Menu
#####
1) Read Byte from port
2) Read all bits from port
3) Write Byte to port
4) Write Bit to a port
q) Quit Program
#####
3
Enter port number (0-1):0
Enter value 0-255 or q to quit:255
The Byte value 255 is sent to port 0
Enter value 0-255 or q to quit:q
#####
```

Step 6: Write a Bit to a port

Allowing the user to set the state (high/low or 0/1) of a single specific digital bit on a port.

```
root@dscguest-Default-string:/home/dscguest/Downloads/VENUSDIO# ./VENUSDIO

VENUS DIO FUNCTIONS DEMO

DIO INPUT AND OUTPUT: Main Menu
#####
1) Read Byte from port
2) Read all bits from port
3) Write Byte to port
4) Write Bit to a port
q) Quit Program
#####
4
Enter port number (0-1):0
Enter Bit (0-7):or q to quit :0
Enter Bit value (0-1):1
Enter Bit (0-7):or q to quit :q
#####
```

Step 7: Exit the Application

Allow the user to exit the application.

```
root@dscguest-Default-string:/home/dscguest/Downloads/VENUSDIO# ./VENUSDIO

VENUS DIO FUNCTIONS DEMO

DIO INPUT AND OUTPUT: Main Menu
#####
1) Read Byte from port
2) Read all bits from port
3) Write Byte to port
4) Write Bit to a port
q) Quit Program
#####
q
root@dscguest-Default-string:/home/dscguest/Downloads/VENUSDIO#
```

10.Steps to Run Serial Mode Application

Step 1:

"To install the gpiod library for accessing GPIO pins using the gpiod library."

```
dscguest@dscguest-Default-string:~$ sudo apt-get install gpiod libgpiod-dev
```

Step 2:

The **cd** (change directory) command in Linux is used to navigate between directories in the filesystem.

```
dscguest@dscguest-Default-string:~/VENUS_DEMOS$ ls
VenusDemo
dscguest@dscguest-Default-string:~/VENUS_DEMOS$ cd VenusDemo/
dscguest@dscguest-Default-string:~/VENUS_DEMOS/VenusDemo$ ls
BuildAll  Common  Executables  VENUSDIO  VenusSerialportConfig
dscguest@dscguest-Default-string:~/VENUS_DEMOS/VenusDemo$ cd Executables/
dscguest@dscguest-Default-string:~/VENUS_DEMOS/VenusDemo/Executables$ ls
VENUSDIO  VenusSerialMode
dscguest@dscguest-Default-string:~/VENUS_DEMOS/VenusDemo/Executables$
```

Step 3:

./VENUSSERIALMODE is a command used to execute the compiled binary named VENUSSERIALMODE from the current directory in a Linux environment.

Change the UART 1, UART 2, UART 3 and UART 4 for RS232 mode

```
dscguest@dscguest-Default-string:~/VENUS_DEMOS/VenusDemo/Executables$ sudo ./VenusSerialMode
[sudo] password for dscguest:
    GPIO MODE SELECTION: Main Menu
#####
1. Select mode for UART 1 and UART 2
2. Select mode for UART 3 and UART 4
#####
Your option...
1
Select the Mode:
0 - RS232
1 - RS485
2 - RS422
0
UART1 and UART2 modes changed successfully.
Do you want to configure again : 'y' -yes 'q'-quit
y
    GPIO MODE SELECTION: Main Menu
#####
1. Select mode for UART 1 and UART 2
2. Select mode for UART 3 and UART 4
#####
Your option...
2
Select the Mode:
0 - RS232
1 - RS485
2 - RS422
0
UART3 and UART4 modes changed successfully.
Do you want to configure again : 'y' -yes 'q'-quit
```

Change the UART 1, UART 2, UART 3 and UART 4 for RS485 mode

```
GPIO MODE SELECTION: Main Menu
#####
1. Select mode for UART 1 and UART 2
2. Select mode for UART 3 and UART 4
#####
Your option...
1
Select the Mode:
0 - RS232
1 - RS485
2 - RS422
1
UART1 and UART2 modes changed successfully.
Do you want to configure again : 'y' -yes 'q'-quit
y
GPIO MODE SELECTION: Main Menu
#####
1. Select mode for UART 1 and UART 2
2. Select mode for UART 3 and UART 4
#####
Your option...
2
Select the Mode:
0 - RS232
1 - RS485
2 - RS422
1
UART3 and UART4 modes changed successfully.
Do you want to configure again : 'y' -yes 'q'-quit
```

Change the UART 1, UART 2, UART 3 and UART 4 for RS422 mode

```
GPIO MODE SELECTION: Main Menu
#####
1. Select mode for UART 1 and UART 2
2. Select mode for UART 3 and UART 4
#####
Your option...
1
Select the Mode:
0 - RS232
1 - RS485
2 - RS422
2
UART1 and UART2 modes changed successfully.
Do you want to configure again : 'y' -yes 'q'-quit

y
GPIO MODE SELECTION: Main Menu
#####
1. Select mode for UART 1 and UART 2
2. Select mode for UART 3 and UART 4
#####
Your option...
2
Select the Mode:
0 - RS232
1 - RS485
2 - RS422
2
UART3 and UART4 modes changed successfully.
Do you want to configure again : 'y' -yes 'q'-quit
```