

Gowin SGDMA Windows Driver Installation & Sample Applications Guide

Dependencies

- Target machine running Windows 10 or Windows 11

Directory Structure

```
gowin_sgdma_windows_vx.x.x/  
|__ driver/           - Driver  
|__ app/              - Sample applications  
|__ README.md         - This document  
|__ README.pdf        - PDF version of this document
```

Driver Installation

Install the driver through Windows *Device Manager*.

1. Open *Device Manager*
2. Initially, the device appears as a **PCI Memory Controller** with a yellow exclamation mark
3. Right-click the device and select **Update driver** → **Browse my computer for drivers**, then choose the SGDMA driver directory
4. If prompted about an unverified driver publisher, select **Always install this driver software**
5. You should now see **Gowin SGDMA Device** → **Gowin SGDMA** in *Device Manager*

Note: This driver is not signed with a certified signature; it uses a test signature instead. You must enable “test-signed drivers” in the Windows boot configuration to install this driver. Enable it by running the following command in `cmd`. For more information, see the [MSDN website](#).

```
bcdedit /set testsigning on
```

Loading Mode (MSI Interrupt / Hardware Polling)

The driver supports two loading modes, selected by the `POLL_MODE` registry value installed by `GDMA.inf`:

- `POLL_MODE = 1` (default) — polling mode
- `POLL_MODE = 0` — interrupt mode (MSI)

To change the mode, edit the `POLL_MODE` value in the `[GDMA_Inst.NT.Services.AddReg]` section of `GDMA.inf` before installing the driver:

```
[GDMA_Inst.NT.Services.AddReg]  
HKR,Parameters,"POLL_MODE",0x00010001,1 ; set to 0 for interrupts, default is 1  
(hardware polling)
```

Sample Applications

gdma_rw

This program can open any device node and perform read/write operations. It is typically used to access the memory space of the *control* or *user* PCIe BARs, and it can also perform single DMA operations via the *h2c** and *c2h** nodes, where the asterisk (*) indicates the channel index (0-3).

Usage

```
gdma_rw.exe <DEVNODE> <read|write> <ADDR> [OPTIONS] [DATA]
  - DEVNODE : control | user | h2c_* | c2h_*,
              where * is a numeric wildcard (0-3).
  - ADDR :    Target offset address of the read/write operation.
              Can be in decimal or hexadecimal.
  - OPTIONS :
              -a ALIGN buffer alignment (default: PAGE_SIZE)
              -b open file as binary. only use with the -f option below.
              -e C2H: open with FILE_ATTRIBUTE_ARCHIVE for EOP flush
(streaming),
              each read returns on first EOP boundary
              -f PATH use contents of file at PATH as input or write output
into file
              -l LEN length of data to read/write (default: 4, or file size for
write)
              -v more verbose output
  - DATA :   Space-separated byte data in decimal or hex (big endian).
              e.g. for the 4-byte value 0x44332211 (decimal 1144201745),
              DATA can be either: 0x11 0x22 0x33 0x44
              or: 17 34 51 68
```

Examples

Read a 4-byte control register at offset 0x1000:

```
gdma_rw.exe control read 0x1000 -l 4
```

Write 2 bytes (0x1234) to a control register at offset 0x2004:

```
gdma_rw.exe control write 0x2004 0x34 0x12
```

Read 4KB from C2H_0 starting at offset 0 into a binary file (my_data.bin):

```
gdma_rw.exe c2h_0 read 0 -l 0x1000 -b -f my_data.bin
```

Read 4KB of data from C2H channel 0 at offset 0x100:

```
gdma_rw.exe c2h_0 read 0x100 -l 0x1000
```

Write the contents of my_data.bin to H2C channel 0 at offset 0x0:

```
gdma_rw.exe h2c_0 write 0x0 -b -f my_data.bin
```

perf

This program measures H2C and C2H channel performance. You can specify the packet size and test duration.

Usage

```
perf.exe -d <h2c_X|c2h_X> [-l <bytes|K|M|G>] [-t <seconds>] [-c <card>] [-a <align>] [-e] [-v]
```

ARGS:

- d h2c_* or c2h_*, where * is a numeric wildcard (0-3).
- l Packet size per transfer. Supports K/M/G suffixes and hexadecimal.
- t Test duration in seconds
- c Device card index (default 0)
- a Buffer alignment in bytes (default 4096)
- e C2H: open with FILE_ATTRIBUTE_ARCHIVE for EOP flush (streaming), each read returns on first EOP boundary
- v More verbose output
- h|-? Show help

Examples

Test H2C_0 performance with 1MB packets for 5 seconds:

```
perf.exe -d h2c_0 -l 1M -t 5
```

Test C2H_0 performance with 128KB packets for 4 seconds:

```
perf.exe -d c2h_0 -l 128K -t 4
```

Reference Performance

H2C reference:

```
Transferred : 1875902464 bytes in 5.002 s
Packet size : 1048576 bytes
Throughput   : 3.000 Gbps (375.0 MB/s)
Efficiency   : 75.0 % (vs PCIe Gen2 x1 = 4.0 Gbps)
```

C2H reference:

```
Transferred : 1783234560 bytes in 4.000 s
Packet size : 131072 bytes
Throughput   : 3.566 Gbps (445.8 MB/s)
Efficiency   : 89.2 % (vs PCIe Gen2 x1 = 4.0 Gbps)
```

user_event

This program opens all user event device files (event_0 to event_3) and waits for events to be triggered by the FPGA. When an event is triggered, the corresponding message is printed to the terminal. The program loops indefinitely; press CTRL-C to exit.

```
user_event.exe
```

References

- [MSDN: Testsigning Boot Configuration](#)