

DATA TRANSFER THROUGH USB 3.0 USING CYPRESS FX3

RAHUL YAMASANI

Submitted to: Prof. Dennis Akos

Introduction:

The first section contains a brief description of flow in the data transfer in the system. Next the discussion is carried to Interface Definition and state diagram using GPIF II designer. It is followed by explanation causing data transfer failure. Later, the discussion is made on how far Russian code was helpful and changes made to existing firmware. The last section explains things to try yet and finally followed by conclusion.

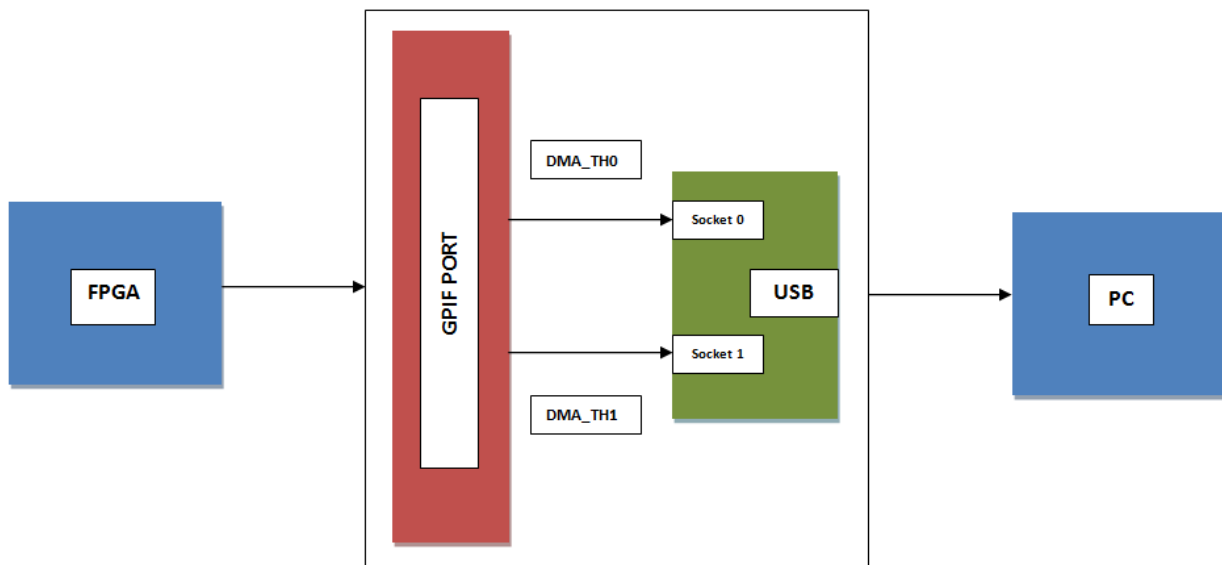
The Cypress has designed its own software and firmware for FX3. The software development kit designed for Cypress FX3 is based on eclipse environment. This SDK provides support on Windows, Linux and MacOS platforms. However, I have installed and tested on Linux and Windows platforms.

Block Diagram:

The entire transfer of Data stream from FPGA to PC can be divided into three phases

1. FPGA to GPIF on Cypress Fx3
2. GPIF to USB
3. USB to PC

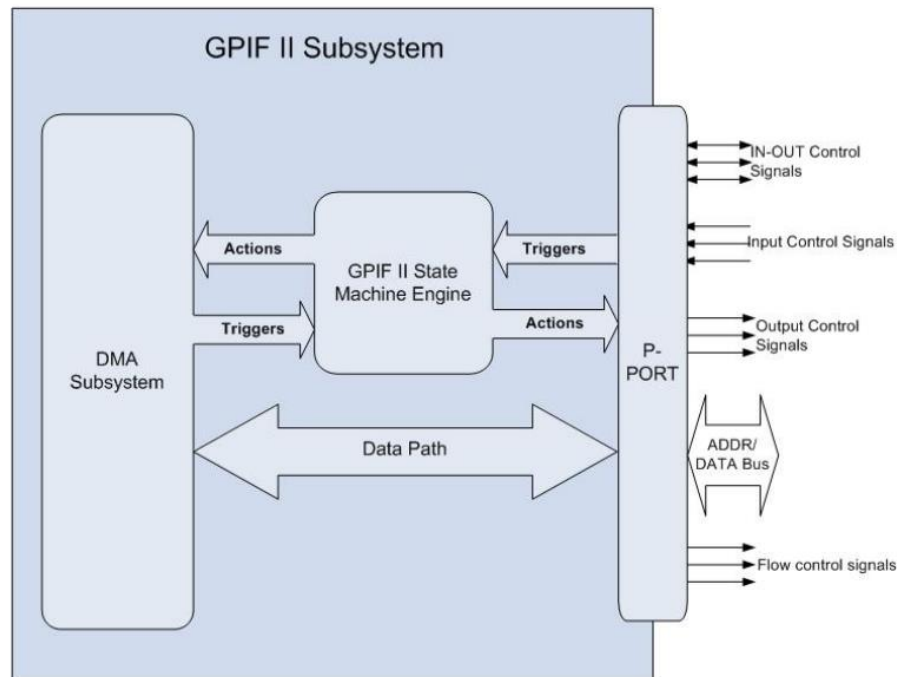
The following block diagram illustrates the data flow



1. FPGA to GPIF

FPGA generates a data stream of 8-bit each. It continuously feeds data onto GPIF Port which is monitored by GPIF II system. In our scenario, the functionality of GPIF II system is to encode the incoming data stream into a packet size of 1024 bytes (Super speed mode on Cypress FX3) and feed it the DMA.

The block diagram GPIF II subsystem is as follows.



GPIF II subsystem in Cypress FX3

<http://www.cypress.com/file/124206/download>

2. GPIF to USB

This section is crucial in the Data stream flow as the accuracy of the Data transfer is determined here. In this phase the drivers written using Cypress Fx3 SDK are used to transfer the Data from GPIF port received to USB. When Data is transferred from GPIF to USB, GPIF (Cypress FX3) becomes the producer and USB (PC) becomes the consumer. The incoming data arrives at high speed and it is important that it has to be transferred with minimal latency. This can be achieved by using DMA (Direct Memory Access). This is explained in detail in the upcoming sections.

In current implementation, two DMA blocks are used for transferring the data corresponding to Thread 0 and Thread 1 as shown in the above Block Diagram. DMA collects the data from GPIF and puts into USB that transfers the data stream to PC.

3. USB to PC

The incoming data from Cypress FX3 is fed into PC. On PC, we have a linux OS installed. Bulkread.c is used for collecting the incoming data stream into a file. Next the continuity is checked using checkdata.c

The code to be written to transfer DATA stream involves the following steps:

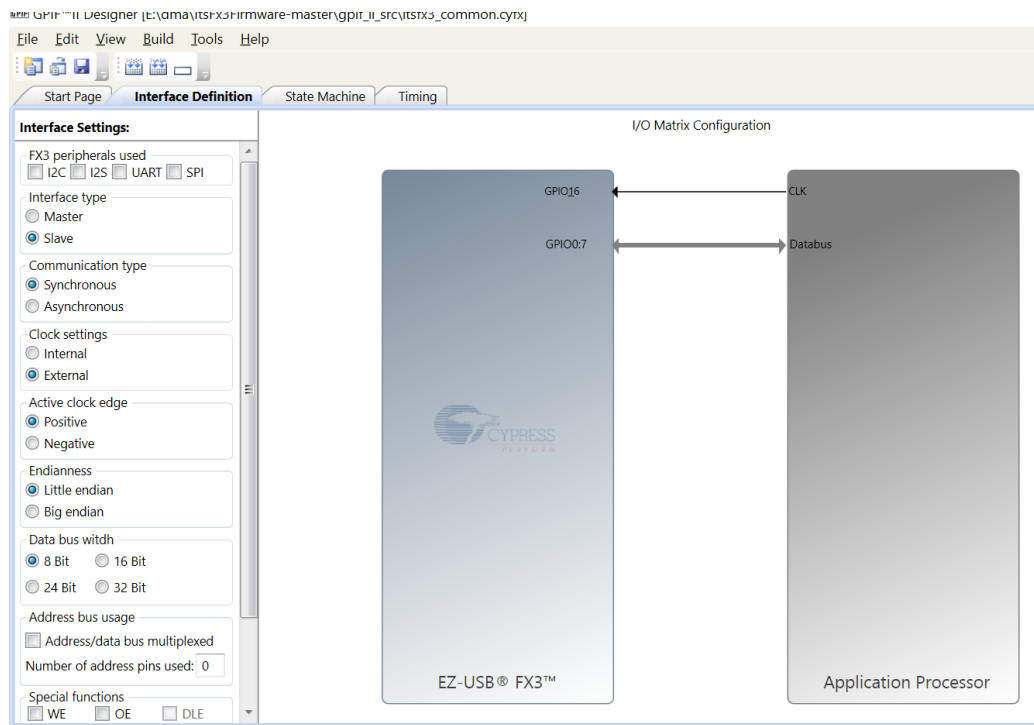
1. Writing Firmware

The firmware written for data transfer is the modified version of Synchronous Slave FIFO example provided by Cypress FX3 support. The actual code contains a single DMA to transfer the data. In our case as the data rate is high and intended to push further high, it is redesigned using two sockets that would serve the purpose of extra buffers. Accordingly, two DMA's have been configured to support the above idea. DMA buffer size and DMA buffer count becomes important parameters for successful transfer of data stream.

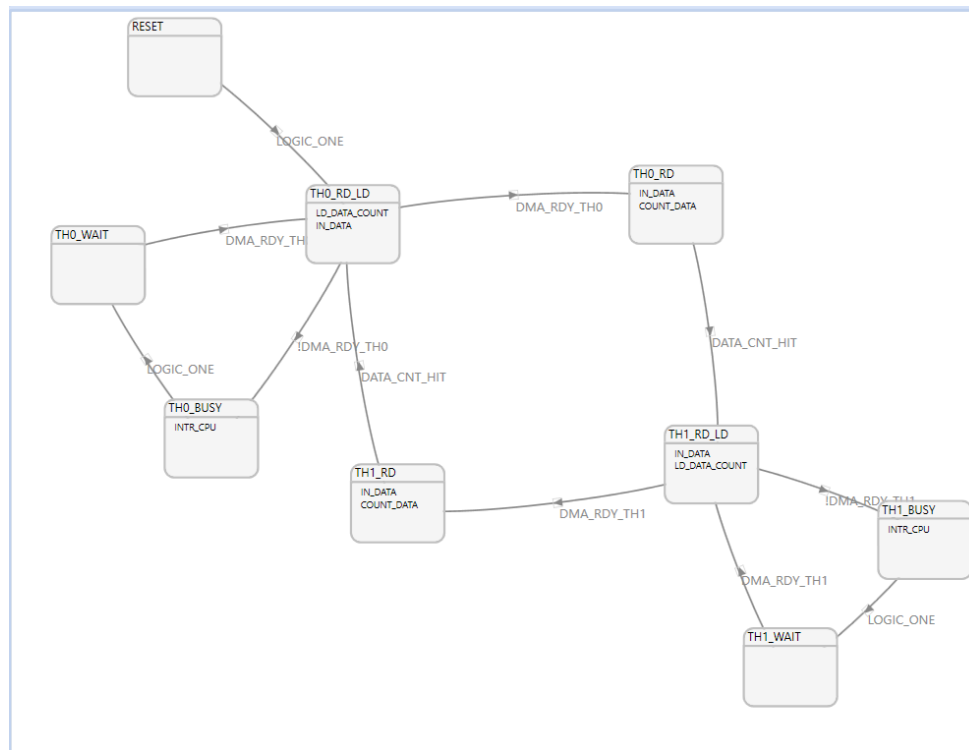
On completion of firmware, it is critical to design the interface and state diagram and is explained in the following section.

2. Designing Interface and State Diagram

Cypress FX3 provides GPIF II designer tool to design Interface Definition and state diagram. The system is configured in synchronous mode with data bus width of 8-bit which is shown in the diagram below.



The next step is to design state diagram



Explanation:

Contains two sockets with socket 0 corresponding to Thread 0 and socket 1 to Thread 1. As explained above, configured two DMA channels corresponding to each socket.

1. Loads data on Thread 0 (TH0_RD_LD). When DMA (DMA_RDY_TH0) is ready the data is read into buffers using the state TH_RD.
2. On Data count hit, it enters into the next thread where loading and reading are done as explained in step 1.
3. Additionally, each read state (TH0_RD_LD and TH1_RD_LD) is associated with INTR_CPU which enters into a wait state until DMA transfer gets ready. Probably this is why we are missing the Data stream.

On verifying the continuity of Data stream it has been observed that Data stream is discontinuous as many times it enters the interrupt. It is entering the interrupt indicating that DMA is not ready. The definite reason for this not understood.

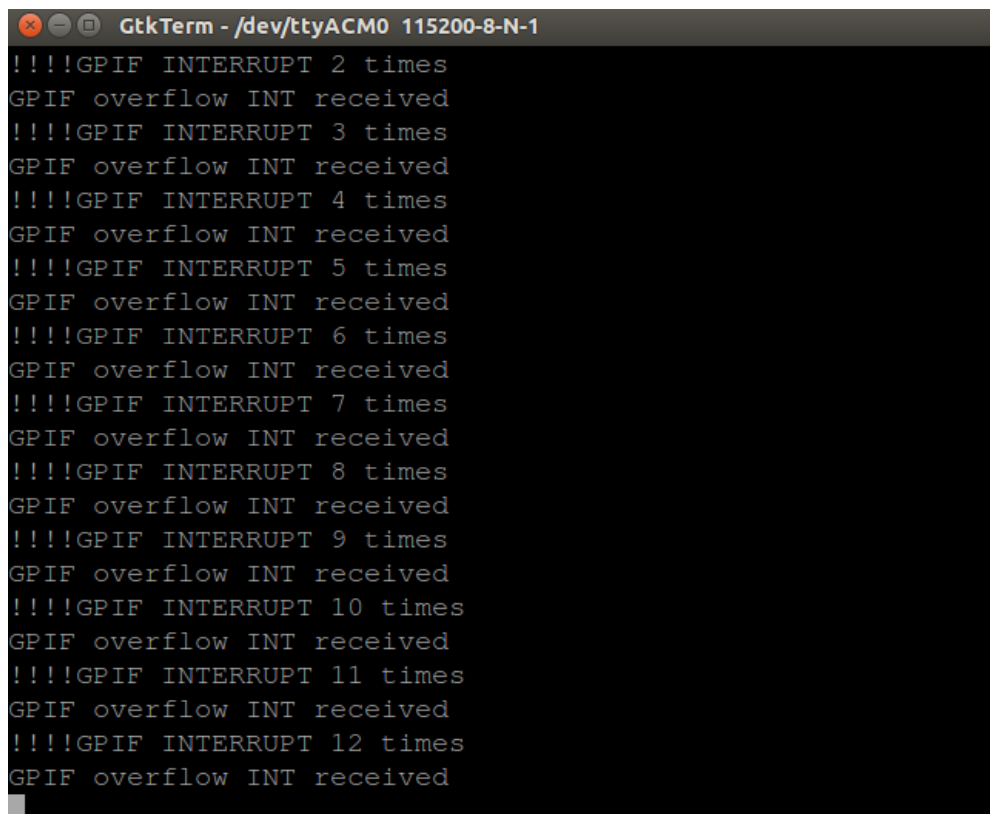
The DMA size and DMA count for Thread 0 are set to 16384 and 6.

The DMA size and DMA count for Thread 1 are set to 16384 and 16.

OBSERVATION:

Also, on changing the DMA count for thread 0 it is able to build the .img file but unable to transfer data stream and exiting out with data error = -16.

The incoming data stream on PC is read into a file and continuity is being checked after the entire transfer is completed. The error explained above is as follows: Ignore the interrupt for the first two times. It is due to change in USB events.

A screenshot of a terminal window titled "GtkTerm - /dev/ttyACM0 115200-8-N-1". The terminal displays a series of messages indicating GPIF interrupts and overflow events. The messages are as follows:

```
!!!!GPIF INTERRUPT 2 times  
GPIF overflow INT received  
!!!!GPIF INTERRUPT 3 times  
GPIF overflow INT received  
!!!!GPIF INTERRUPT 4 times  
GPIF overflow INT received  
!!!!GPIF INTERRUPT 5 times  
GPIF overflow INT received  
!!!!GPIF INTERRUPT 6 times  
GPIF overflow INT received  
!!!!GPIF INTERRUPT 7 times  
GPIF overflow INT received  
!!!!GPIF INTERRUPT 8 times  
GPIF overflow INT received  
!!!!GPIF INTERRUPT 9 times  
GPIF overflow INT received  
!!!!GPIF INTERRUPT 10 times  
GPIF overflow INT received  
!!!!GPIF INTERRUPT 11 times  
GPIF overflow INT received  
!!!!GPIF INTERRUPT 12 times  
GPIF overflow INT received
```

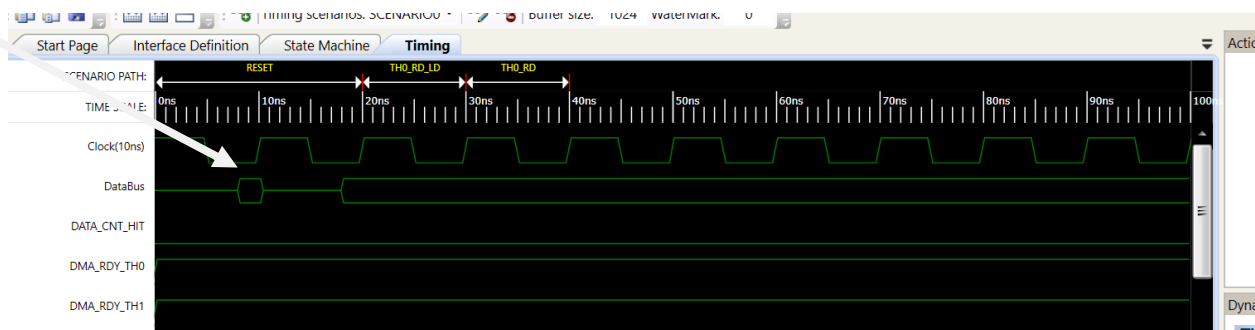
After entering into GPIF interrupt 10 times (12-2) it has been verified that the Data stream is discontinuous for 10 times as shown below

```

dma@dma-U47A ~/Documents/cyusb_linux_1.0.4/src
File Edit View Search Terminal Help
Counter off! Index: 0 Actual Value: 2b Supposed Value: a Location: 12165480448
Counter off! Index: 0 Actual Value: 4 Supposed Value: 2a Location: 12176916480
Counter off! Index: 0 Actual Value: 7c Supposed Value: 3 Location: 13269925888
Counter off! Index: 0 Actual Value: 24 Supposed Value: 7b Location: 13456605184
Counter off! Index: 0 Actual Value: bb Supposed Value: 23 Location: 13710393344
Counter off! Index: 0 Actual Value: c0 Supposed Value: ba Location: 13712834560
Counter off! Index: 0 Actual Value: 7f Supposed Value: bf Location: 13723828224
Counter off! Index: 0 Actual Value: fa Supposed Value: 7e Location: 13729759232
Counter off! Index: 0 Actual Value: 6b Supposed Value: f9 Location: 13741211648
Counter off! Index: 0 Actual Value: 7 Supposed Value: 6a Location: 13741228032

```

Also, the reason for data miss at same location ever time is because the DMA initiation has certain fixed latency as shown in diagram below:



How helpful was Russian Code:

Russian Code helped me lot in understanding the flow of the Data transfer. Their state diagram was clearer and handles the exception occurring during the data flow. Because of this, I was able to debug and conclude that Data loss occurs, whenever a GPIF software interrupt that indicates DMA being busy occurs. From this discussion it is evident that Data loss can be prevented by avoiding the condition for DMA busy

Based on Cypress Fx3 documentation and I made several modifications to the Russian diagram explained as follows:

1. Re designed state diagram by adding additional socket and thread.
2. Re designed the firmware code based on various Cypress Fx3 examples and Forums
3. Used COMMIT Flag to sync the data in between the threads.
4. Modified various values for DMA buffer size and DMA and buffer Count.

Things to try:

1. It has been studied that DMA polls DMA_TH0 flag to indicate the data transfer complete. DMA_TH0 is used has a latency of 3 cycles. This can be avoided by using Thread_watermark flag as it has zero latency as described in [1].
2. I have posted the Problem on Cypress FX3 forums and expect some valuable suggestions to try and I have seen people solving their issues after posting.

Conclusion:

I have completely understood the firmware code and have an in depth knowledge of what parameters effects data transfer. I am able to compile code on both Linux and Windows environment. Make JTAG Debugger available for debugging. Re design state diagram and Interface design to meet our needs.

It is deduced that data miss is occurring only because of generation of software interrupt. Modifying flags in the state diagram would provide some good results. Quitting the data transfer on discontinuity is incorporated in to the code but has been commented. The reason for data being at the same start location every time has been found.

References:

1. <http://www.cypress.com/file/136056/download>
2. http://www.cypress.com/system/files/document/files/FX3_SDK_Linux_Support_0.pdf
3. <http://www.cypress.com/file/124206/download>
4. <http://www.cypress.com/documentation/application-notes/an65974-designing-ez-usb-fx3-slave-fifo-interface>
5. https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0ahUKEwjM17bMrtTOAhXBkx4KHbc5D5gQFggI4MAE&url=http%3A%2F%2Fwww.cypress.com%2Ffile%2F136056%2Fdownload&usg=AFQjCNEl-9HapHFGigYw5XRSfz_g-kP6cQ&bvm=bv.129759880,d.cGc&cad=rja

Appendix:

Installation:

EZ-USB FX3 SDK Installer:

This is the master installer file that installs the USB suite with OS supported host driver and applications, Eclipse and GCC Tool chain that supports ARM Cross compiler.

Installation on Windows operating systems:

Installation on windows is straight forward and can be downloaded from the following link.

<http://www.cypress.com/documentation/software-and-drivers/ez-usb-fx3-software-development-kit>

I have installed the SDK Windows 10 and documentation confirms supports Windows 8,7 and vista:

Installation on Linux Environment:

Installing EZ-USB FX3 Software Development Kit:

Download Link : <http://www.cypress.com/documentation/software-and-drivers/ez-usb-fx3-software-development-kit>

After installation, update the path environment variables as in the following document:

http://www.cypress.com/system/files/document/files/FX3_SDK_Linux_Support_0.pdf

The default version of Arm Cross Compiler (arm-none-eabi-gcc) is 4.8.0.1.

```
U can check the version of arm using arm-none-eabi-gcc --version
```

It might be possible to get some weird compile time errors. First, update the cross compiler to latest version. The following link would help you to do so:

<https://wiki.paparazziuav.org/wiki/Installation/Linux>

The next step is to Install CyUSB suite for Linux platforms