Z80 Tube FIFO Specification

Summary

This note describes a simple FIFO structure for message passing between Amstrad CPC (host) and Raspberry Pi (client).

The Z80Tube CPLD will have two modes of operation

- A FIFO mode (described here)
- Acorn second processor/Z80 to 6502 translation mode (described elsewhere)

Only one mode can be active at a time.

The host will select between the two operating modes by writing an IO register bit.

Some of the FIFO pins will be overlaid on the same Raspberry PI GPIOs which would be used for the PiTubeDirect/Acorn Second Processor Application. This is essential to fit the two modes into an XC95108 CPLD.

On the host side the CPLD implements two IO space registers

- Flags and control
- FIFO Data

The general process for sending data from the host to the client is for the host to poll the flags register until the 'FIFO full' flag is deasserted and then write to the data register.

To receive data from the client the host will poll the flags until the 'FIFO data available' flag is asserted and then read from the data register.

A later version may interrupt the host when FIFO data is available.

On the Client side the CPLD provides two flags which are always available to RaspberryPi GPIOs plus an 8 bit databus with request/acknowledge handshaking.

To read from the host the Rpi will poll its own 'Data available' bit until it is asserted. Then, with the data bus tristated, it will assert the read request signal, wait until the acknowledge bit is asserted, read the databus and finally deassert the read request signal.

Writing from client to host is similar. The Rpi will poll its own 'FIFO data full' bit until it is deasserted. Then, it will drive data onto the data bus, assert the write request signal, wait until the acknowledge bit is asserted, tristate the databus and finally deassert the write request signal.

Initially we expect to provide a single byte buffer in both directions. Later versions may double or even quadruple the size of the FIFOs. This has no effect on the protocol used in software, but potentially will speed up some operations as host and/or client can push or pull multiple bytes to and from the FIFO without waiting for the other to respond each time.

FIFO Pin Outs

Amstrad CPC Side

Standard Z80 pins sufficient for all IO transactions

RaspberryPi Side

Pin Name	Bus Size	CPLD Pin Direction	Pi Pins	Function
DATA	7:0	Bidir	GPIO25,GPIO24,GPIO23,GPIO22, GPIO11,GPIO10,GPIO9,GPIO8	Data bus
WRRQ_B		input	GPIO21	Active low write request
RDRQ_B		Input	GPIO26	Active low read request
ACK_B		Output	GPIO20	Active low acknowledge
DATA_AVAIL		Output	GPIO17	(Read) FIFO Data available flag
FIFO_FULL		Output	GPIO18	(Write) FIFO Full flag

CPC IO Registers

The Z80Tube card reserves 16 IO addresses between &FC10 and &FC1F inclusive.

Registers &FC10 - &FC17 are used for the Acorn Second Processor application and mapped to the Acorn Tube FIFO registers.

Registers &FC18 and &FC19 are used for the FIFO application described here and for selection of the operating mode.

Registers &FC1A - &FC1F are reserved for other applications

Address	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Function
&FC18	Data Available	FIFO Full	X	X	Enable CPLD	Mode[2]	Mode[1]	Mode[0]	Status/Flag Register
&FC19	Data[7]	Data[6]	Data[5]	Data[4]	Data[3]	Data[2]	Data[1]	Data[0]	Data Register

On power up the CPLD is disabled. The host must write to the Enable CPLD and Mode bits to enable the correct mode:

- Enable CPLD=1 enables CPLD operation, otherwise all Rpi side pins are tristate
- <Mode[2:0]>=<001> selects FIFO operation
- <Mode[2:0]>=<000> selects Z806502/Acorn Second Processor/PiTubeDirect operation
- Other combinations of mode bits may be used for other applications.

Data available goes high when incoming data from the client is available to the host

FIFO Full goes high when the outgoing data buffer from host to client is full.

CPC Interactions

CPC Enable GPIOs and set FIFO Mode in CPLD (default is GPIOs disabled)

```
OUT &FC18, &09
```

CPC Enable GPIOs and set FIFO Mode in Acorn Second Processor Mode

```
OUT &FC18, &08
```

CPC FIFO Read

```
‘ Wait for Data Available Flag to go high
WHILE (INP(&FC18) AND &80)=0 : WEND
DATA = INP(&FC19)
```

CPC FIFO Write

```
' Wait for data full flag to go low
WHILE (INP(&FC18) AND &40)<>0: WEND
OUT &FC19, DATA
```

Pi FIFO Read Procedure

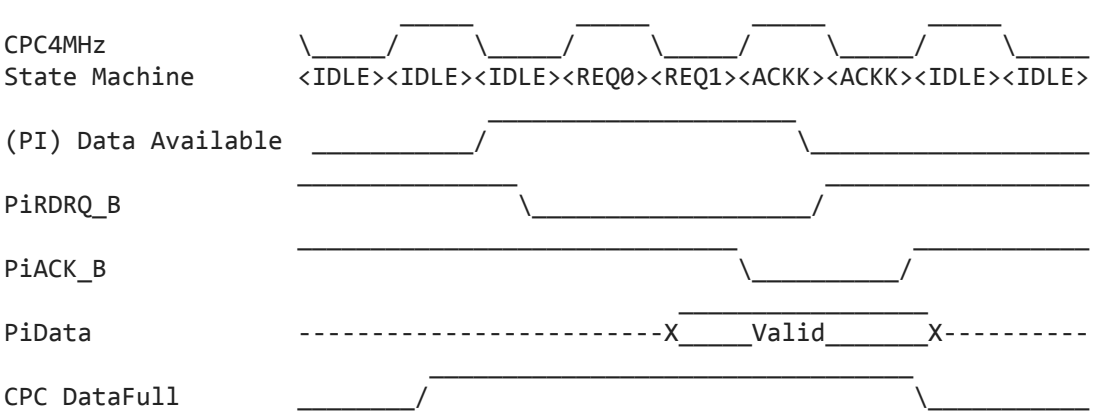
```
# Wait until DataAvailable goes High
while IN(DataAvailable) != 1:
pass

# Make Read Request (active low)
OUT(PIRDRQ_B,0)

# Wait until ACK_B goes low
while IN(PIACK_B) ==1:
    pass

# Read back the data
Data=IN(PIDATA)

# Deassert the Read Request
OUT(PIRDRQ_B,1)
```



* CPLD drives databus during REQ1 and ACK

Pi FIFO Write Procedure

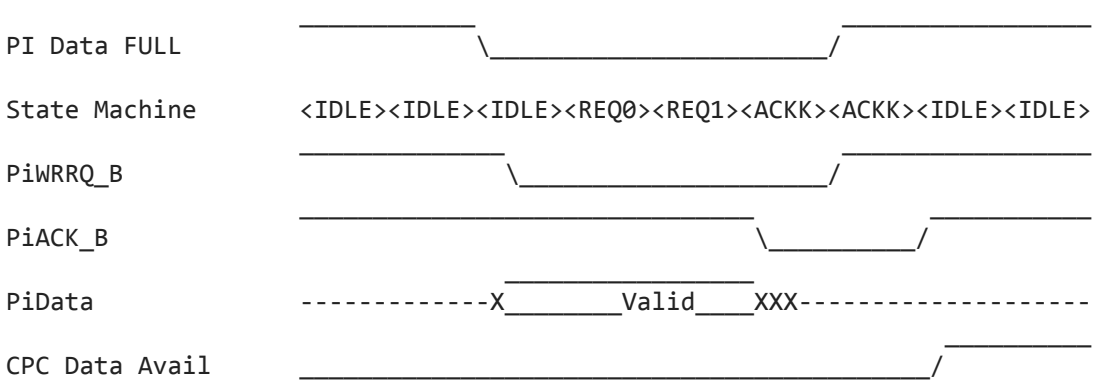
```
# Wait until DataFull goes Low
while IN(DataAvailable) == 1:
pass

# Enable PIDATA tristates and put Data on the bus
DIR(PIDATA, OUTPUT)
OUT(PIDATA, <data>)
# Make Write Request (active low)
OUT(PIWRRQ_B,0)

# Wait until ACK_B goes low
while IN(PIACK_B) ==1:
    pass

# Disable the databus
DIR(PIDATA,INPUT)

# Deassert the Read Request
OUT(PIRDRQ_B,1)
```



* CPLD reads databus and writes to FIFO in REQ1, PI can desassert data bus after acknowledge active edge