ECE385

DIGITAL SYSTEMS LABORATORY

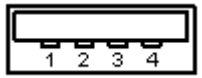
Experiment 8

USB Keyboard and VGA Interface

Joe Meng
Zuofu Cheng

Experiment 8: USB Protocol

- The Universal Serial Bus (USB) standard defines the connection and communication protocols between computers and electronic devices
- Also provides power supply
- Compatible with a wide variety of devices
- USB Port has 4 pins

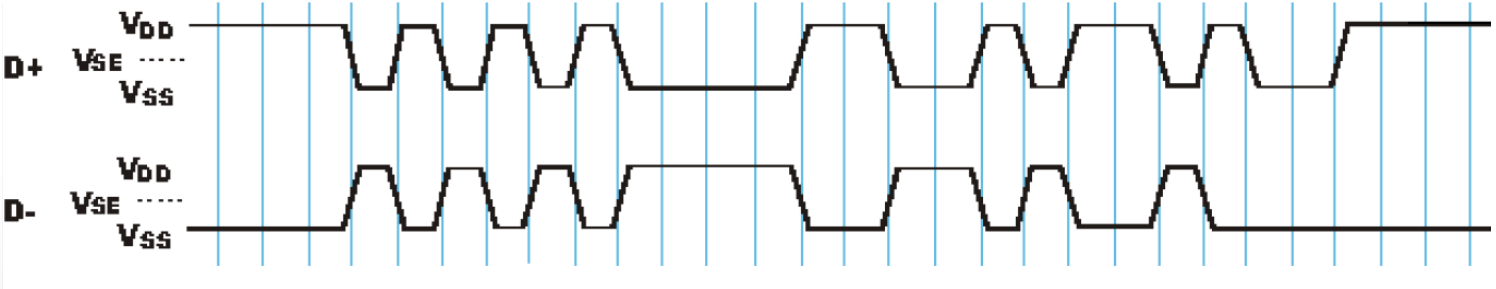


Type A

Type B

Pin	Name
1	VDD (5V)
2	D-
3	D+
4	GND

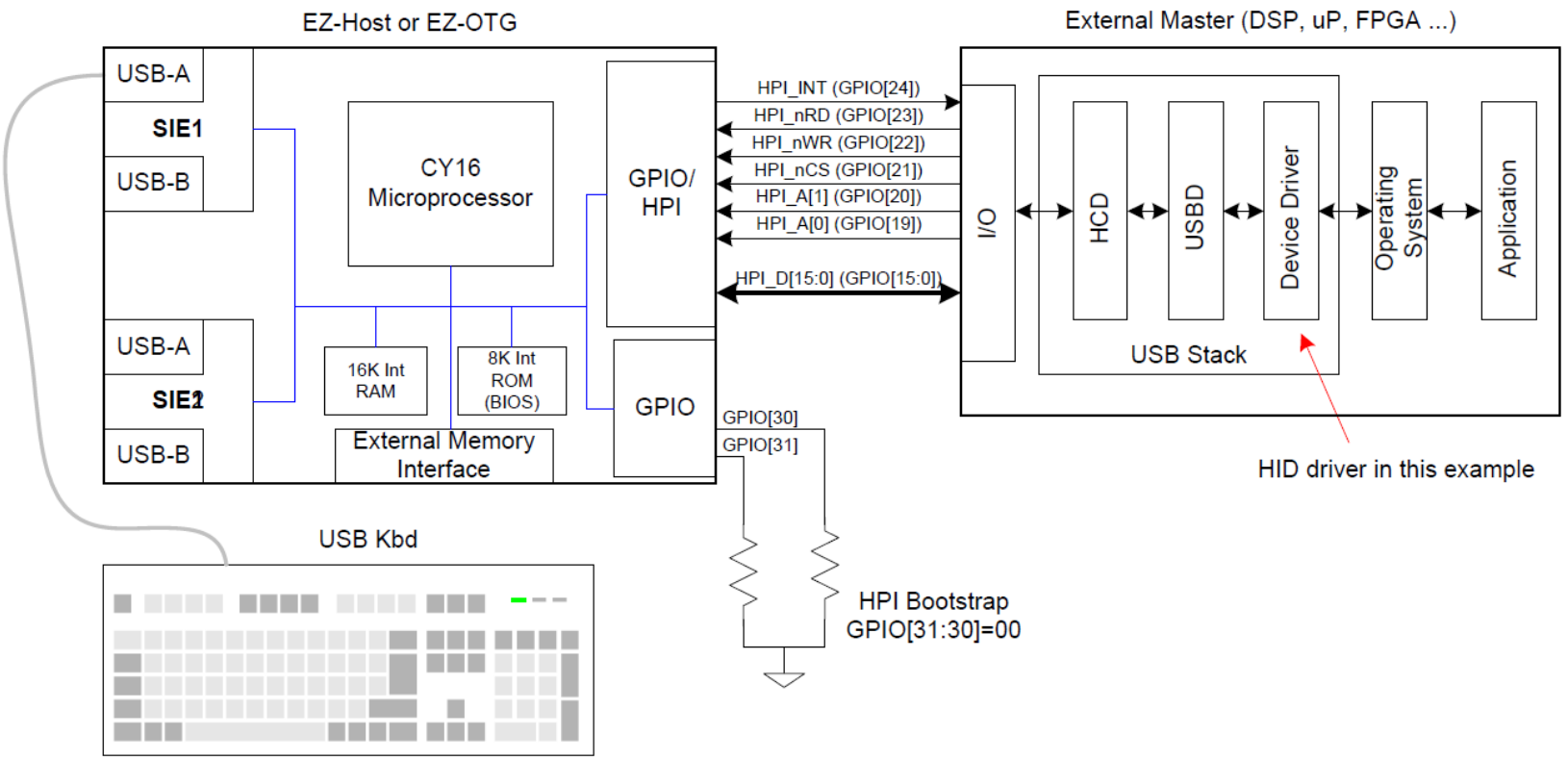
- Data transmit via the D+ and D- lines in a differential pattern
  - Physical layer



- Data are always polled by the host. Device cannot initiate transmission.

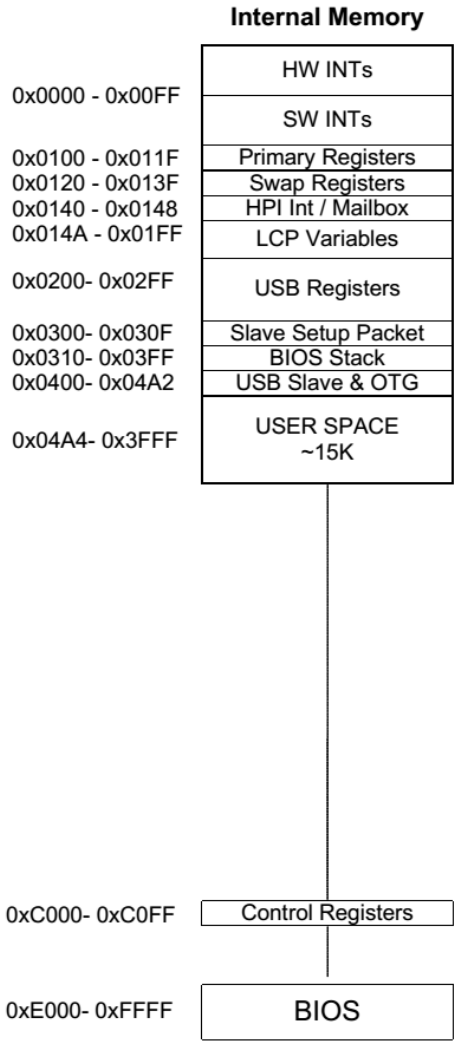
USB Controller: Cypress EZ-OTG

- DE2-115 board comes with the Cypress EZ-OTG (CY7C67200) USB controller
  - Can make DE2-115 act as a host or a USB device



USB Controller: Cypress EZ-OTG

- Rule #1 of USB: All transfers are initiated by the host (in this case, this is the EZ-OTG)
- Rule #2 Each device can have multiple endpoints, all transactions are done from host to endpoint
- Different kinds of transactions are possible, but all transactions must follow above rules
- We program USB by programming memory of the EZ-OTG, which acts as a bridge between a parallel memory interface and the serial USB interface and handles the low level USB operations (assembling packets, etc)
- Memory space of EZ-OTG is shown on the right
- We'll read/write into "USER SPACE" and interact with software built into the EZ-OTG



USB Controller: Cypress EZ-OTG

- EZ-OTG has 4 HPI registers for commands and configurations
  - Read and write on the registers to control EZ-OTG by specifying the address
  - Functionality of each bit field in each register is defined in the data sheet

Port Registers	HPI A [1]	HPI A [0]	Access
HPI DATA	0	0	RW
HPI MAILBOX	0	1	RW
HPI ADDRESS	1	0	W
HPI STATUS	1	1	R

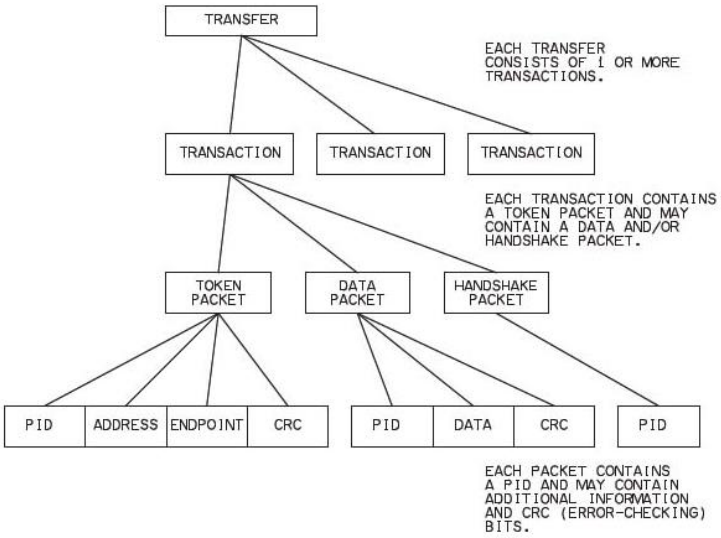
HPI Status ←

Bit16	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
VBUS Flag	ID Flag	Reserved	SOF/EOP2 Flag	Reserved	SOF/EOP1 Flag	Reset2 Flag	Mailbox IN Flag
Bit7	**Bit6**	**Bit5**	**Bit4**	**Bit3**	**Bit2**	**Bit1**	**Bit0**
Resume2 Flag	Resume1 Flag	SIE2msg	SIE1msg	Done2 Flag	Done1 Flag	Reset1 Flag	Mailbox OUT Flag

- EZ-OTG also has a RAM, a ROM (BIOS), and a 16-bit RISC processor (CY16) which can be programmed (we'll use the CY16's built in software for host)
  - Many commands can be made by writing the command into the RAM and then activate through the HPI registers
- Note 4 register "window" into bigger EZ-OTG memory space

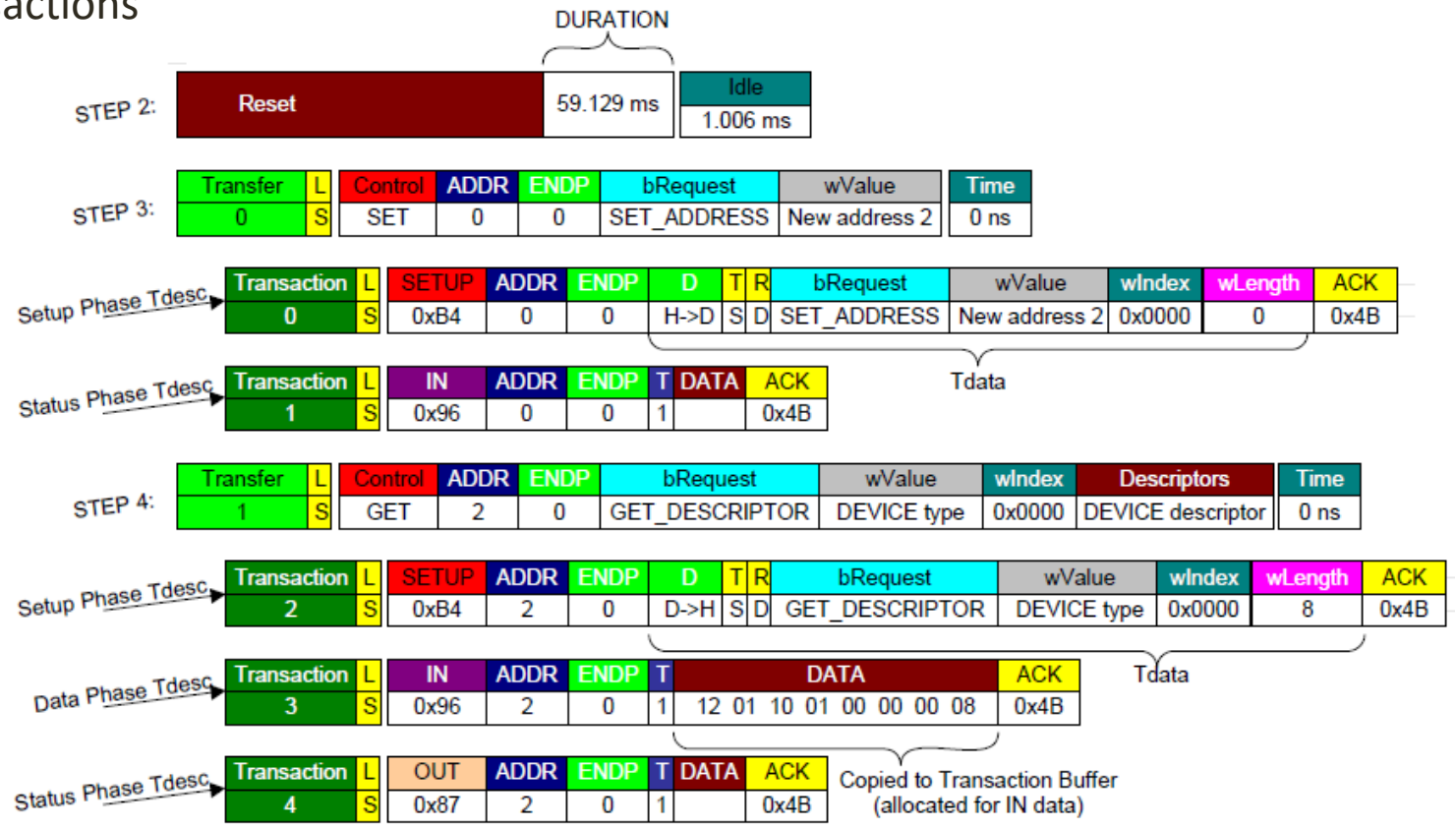
USB Transfer

- Each **USB transfer** is made up of one or more **USB transactions**
  - Four types: control, bulk, isochronous, and interrupt
- Each transaction is then made up of 2-3 **USB packets**
  - First packet is the direction token sent by the Host Controller, which can be a setup, IN, or, OUT
  - Second packet is the data packet, sent by either the Host or the Device, depending on direction
  - Third packet is the ACK
- EZ-OTG communicates with the USB device and stores the packet data in the RAM



EZ-OTG HPI Transactions

- Example: First few transfers for USB keyboard enumeration made through HPI transactions



- Tdesc (or TD, Transaction Descriptor) is a data structure that stores information in bit fields specified by the USB specification and the EZ-OTG data sheet
- In the lab we will work to complete Step 3 (see USB tutorial)

Example: Interface Descriptor

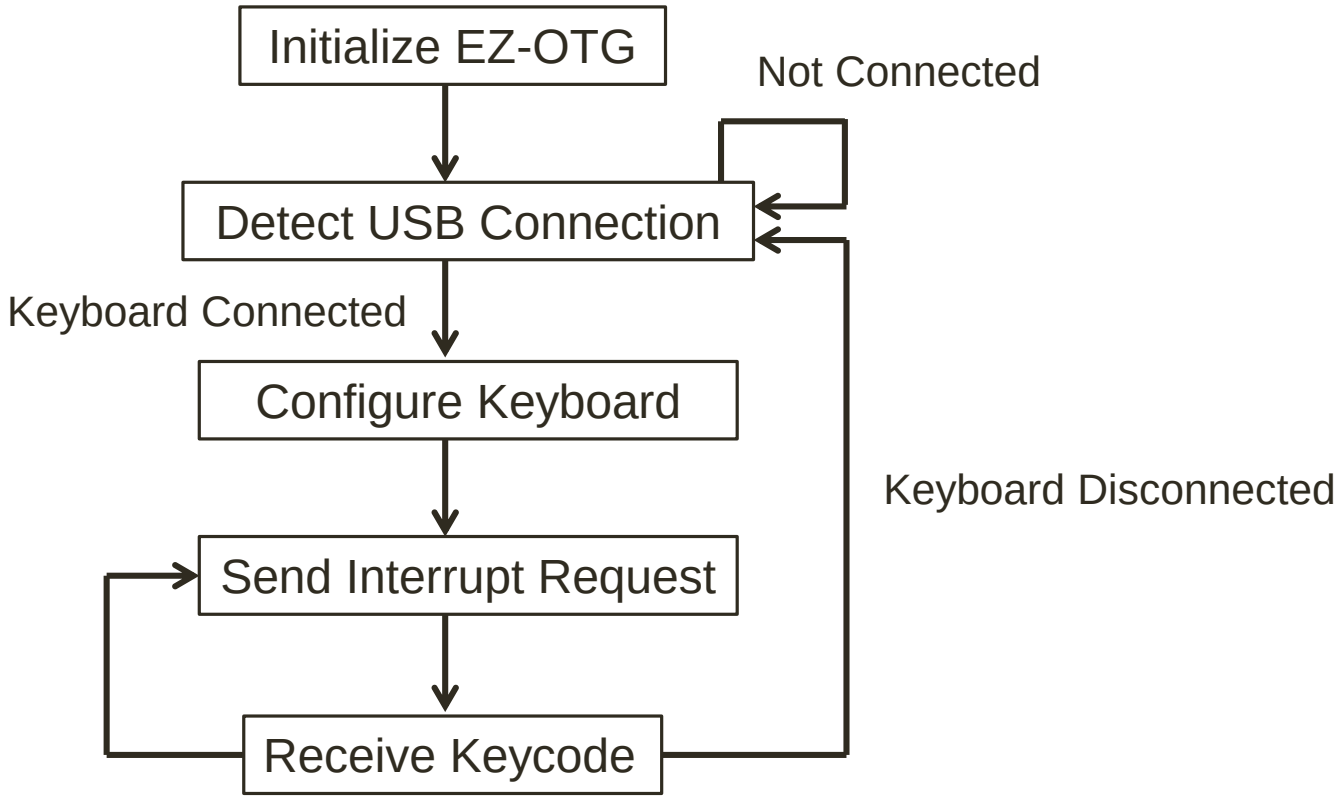
- Keyboard/mouse interface descriptor packets as in the HID specification

Part	Offset/Size (Bytes)	Description	Sample Value
bLength	0/1	Size of this descriptor in bytes.	0x09
bDescriptorType	1/1	Interface descriptor type (assigned by USB).	0x04
bInterfaceNumber	2/1	Number of interface. Zero-based value identifying the index in the array of concurrent interfaces supported by this configuration.	0x00
bAlternateSetting	3/1	Value used to select alternate setting for the interface identified in the prior field.	0x00
bNumEndpoints	4/1	Number of endpoints used by this interface (excluding endpoint zero). If this value is zero, this interface only uses endpoint zero.	0x01
bInterfaceClass	5/1	Class code (HID code assigned by USB).	0x03
bInterfaceSubClass	6/1	Subclass code. 0 No subclass 1 Boot Interface subclass	0x01
bInterfaceProtocol	7/1	Protocol code. 0 None 1 Keyboard 2 Mouse	0x01
iInterface	8/1	Index of string descriptor describing this interface.	0x00

- Sample values are just for reference; actual values may vary from device to device

USB Keyboard with Nios

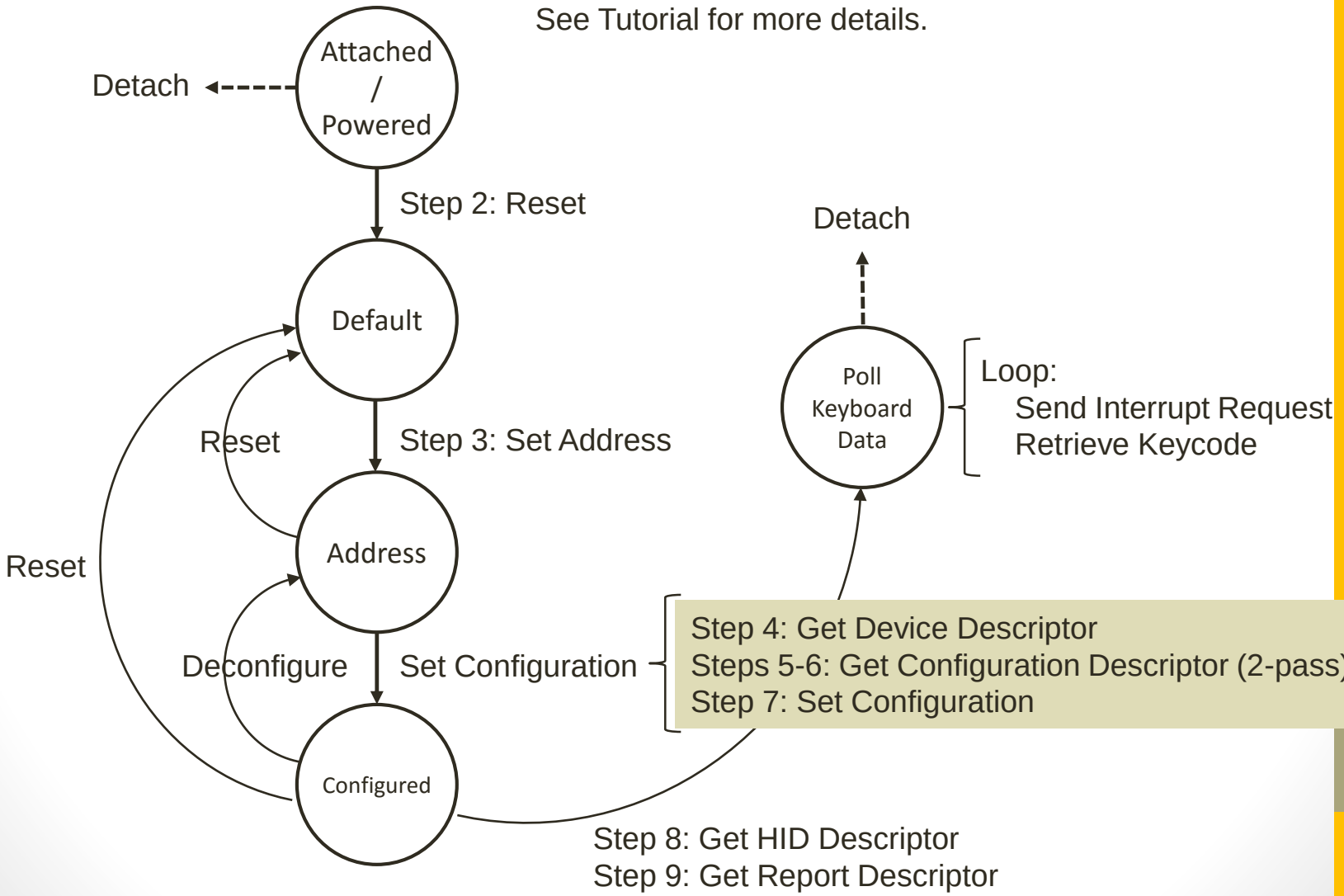
All done in software functions. Summarized as 9 major steps in the tutorial.



The retrieved keycode is sent to the hardware for further use.

USB Keyboard Configuration

The steps correspond to different stages in the program.
See Tutorial for more details.



Basic Read and Write

- IORD and IOWR are Altera's built-in functions for the Nios program to interact with memory-mapped peripherals
- IORD(base, offset)
  - Read and return data from the memory location specified by (base address + offset). Offset is word-wise, that is, a offset of 1 is equivalent to a 32-bit or 4-byte offset in the address
- IOWR(base, offset, data)
  - Write data to the memory location specified by (base address + offset)
- Example:
  - IOWR(CY7C67200_BASE,HPI_MAILBOX,COMM_RESET);
  - Writes COMM_RESET (0xFA50) into register HPI_MAILBOX, which is HPI_MAILBOX=1 word offset from the EZ-OTG base address (CY7C67200_BASE)
  - By writing this value we make EZ-OTG do a software reset

Read/Write with EZ-OTG RAM

- Standard Read procedure (Write is similar)
  - Write the address to access to HPI_ADDR
  - EZ-OTG will fetch the data from the specified address and make it ready to be transferred via HPI_DATA
  - Read from HPI_DATA
  - (Optional: continuous read) EZ-OTG will load the data at the next available address to HPI_DATA
    - If more data are to be read from the next address, we can simply read again from HPI_DATA without giving the next address

Example:

```
alt_u16 UsbRead(alt_u16 Address)
{
        IOWR(CY7C67200_BASE,HPI_ADDR,Address);
        return IORD(CY7C67200_BASE,HPI_DATA);
}
```

Working with Transaction Descriptors

- TDs are the actual packets to be transferred

- Choose a start address (e.g. 0x0500)
- Write the start address into HPI_ADDR
  - IOWR(CY7C67200_BASE,HPI_ADDR,0x0500)
- Write the TDs in 16-bit words to HPI_DATA
  - IOWR(CY7C67200_BASE,HPI_DATA,0x050C)
  - IOWR(CY7C67200_BASE,HPI_DATA,0x0008)
  - …
- After all TDs are written, write the start address to HUSB_SIE1_pCurrentTDPtr so EZ-OTG knows where to start reading the TDs
  - UsbWrite(HUSB_SIE1_pCurrentTDPtr,0x0500)
- Then, EZ-OTG sends these packets to the USB device

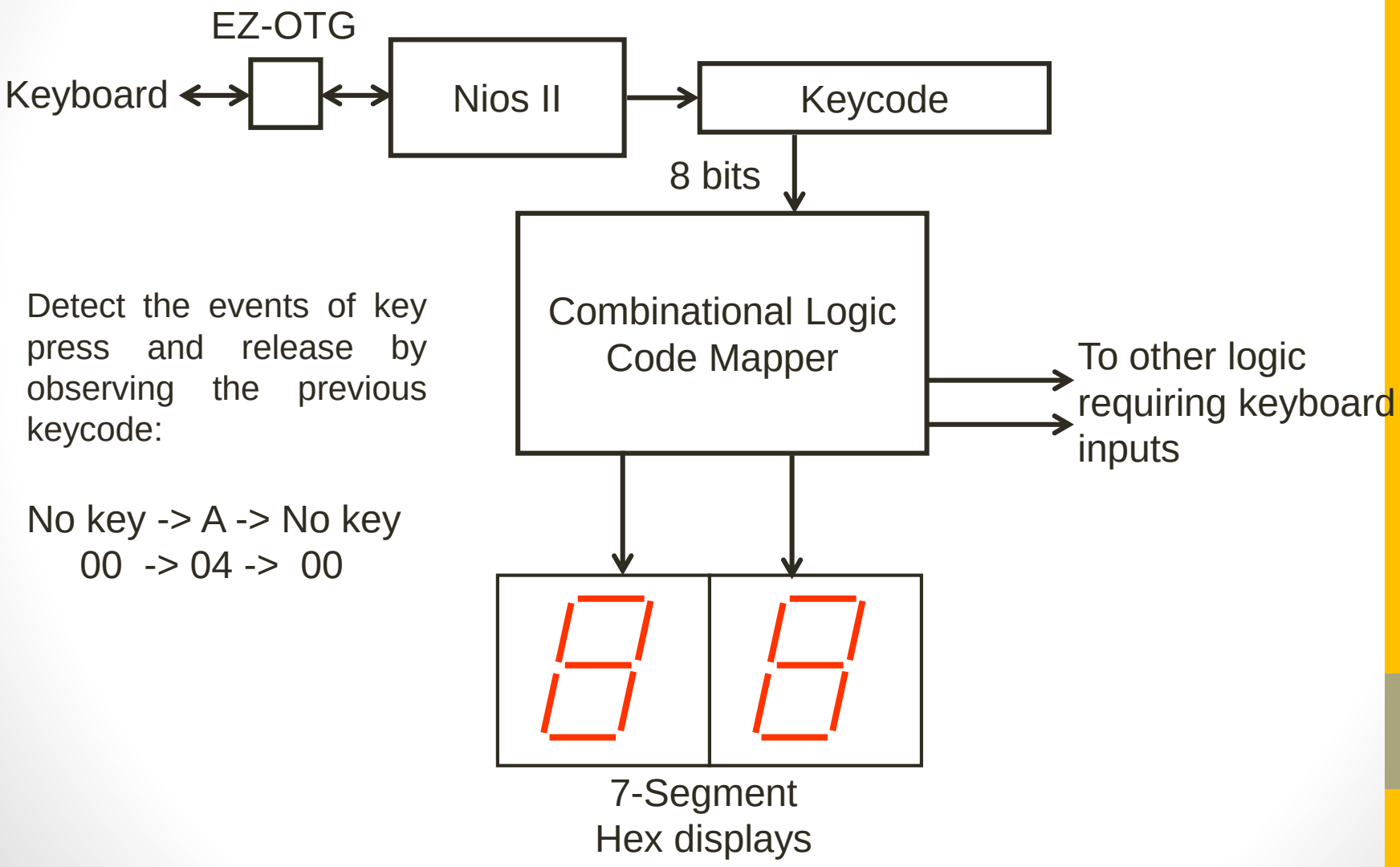
Keycode

- We continuously request a report descriptor from the keyboard, which contains key press information in the following format
- Byte 0 indicates the modifiers (e.g. Ctrl, Alt, Shift, etc.)
- Byte 1 is reserved and should always be 0
- Bytes 2-7 contains keycodes of up to 6 pressed keys
  - ‘A’ is 04, ‘B’ is 05, ‘C’ is 06, and so on
  - Release is 00

Byte	Description
0	Modifiers
1	Reserved (0)
2	Keycode 1
3	Keycode 2
4	Keycode 3
5	Keycode 4
6	Keycode 5
7	Keycode 6

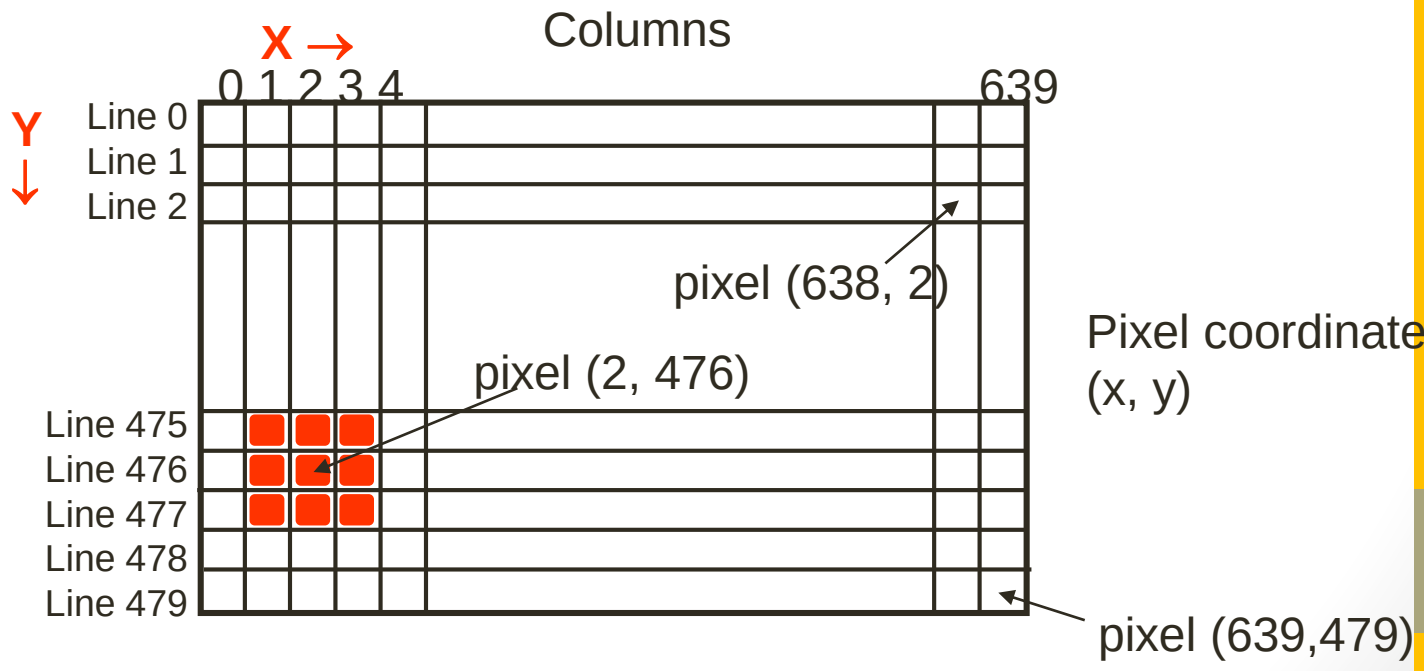
Report Descriptor

Hardware Block Diagram



VGA Monitor Operation

- VGA (Video Graphics Array) Standard
  - The screen is organized as a matrix of pixels
    - 640 horizontal pixels x 480 vertical lines
  - An Electron Beam “paints” each pixel from left to right in each row, and each row from top to bottom



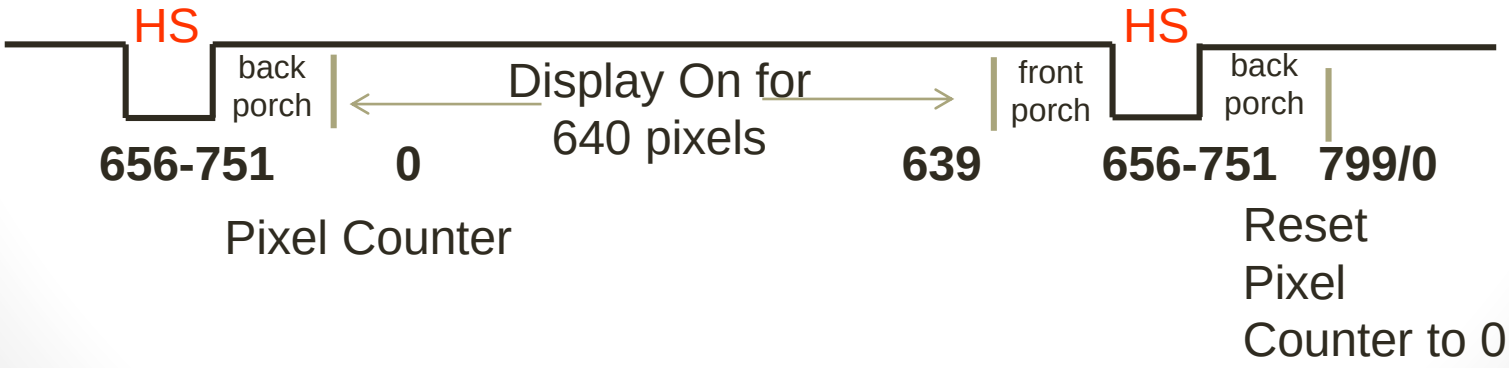
Drawing a Shape

A Shape can be defined by specifying a boundary around a center. In the previous example, the center is (2, 476) and the box is defined *Center ± Size*. For Size=1 all pixels in the box satisfy:

(X≥2-1) AND (X≤2+1) AND
(Y≥476-1) AND (Y≤476+1)

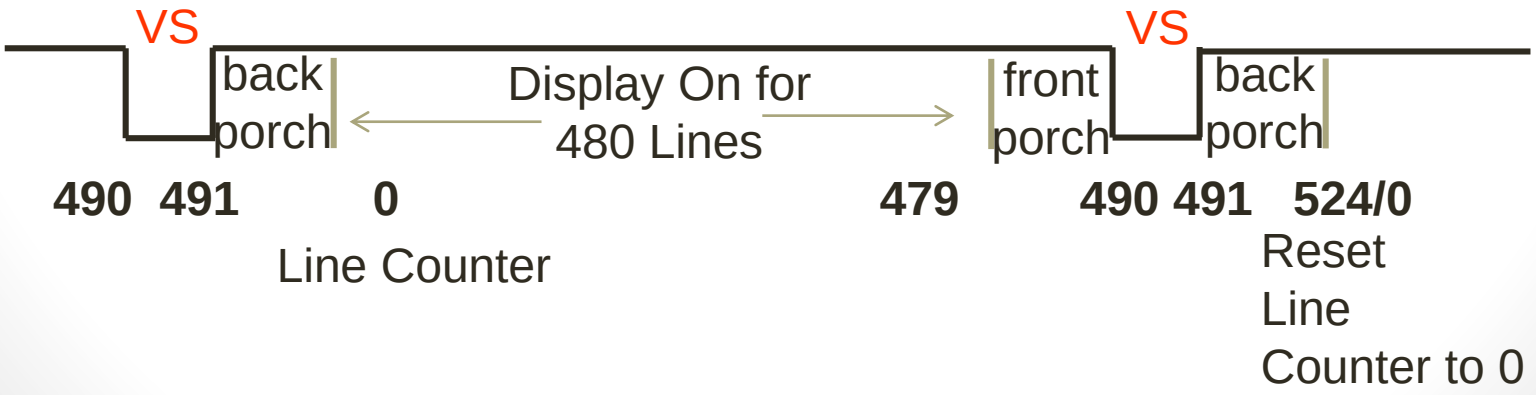
VGA Horizontal Timing

- To generate a Horizontal Sync Pulse, use a 10-bit pixel counter modulo-800
  - Counter increments with a 25MHz clock (pixel clock)
  - Pixel Counts <0 thru 639>: Display On
  - Pixel Counts <640 thru 799>: Display Off
  - Pixel Count <656 thru 751>: HS Pulse Active for 96 pixels
  - Pixel Count 799: Reset pixel counter
  - HS Pulse is Active Low for most monitors

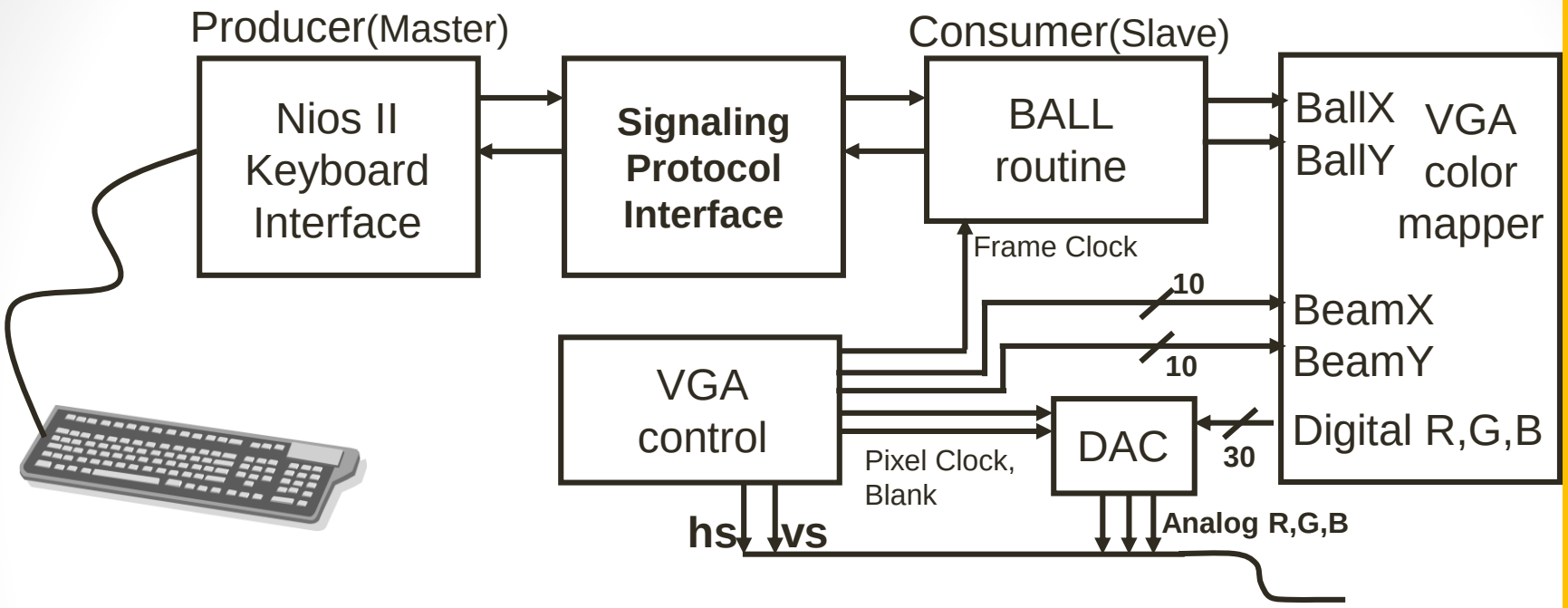


VGA Vertical Timing

- To generate Vertical Sync Pulse (VS), start a 10-bit Line Counter modulo-525
  - Counter increments every 800 pixels
  - Line Counts <0 thru 479> : Display On
  - Line Counts <480 thru 524>: Display Off
  - Line Counts 490 and 491: VS Pulse Active
  - Line Count 524: Reset Line Counter
  - VS Pulse is Active Low for most monitors



Producer-Consumer Signaling Protocol



Ball routine: partially given
Color mapper: given
VGA controller: given