Microcontroller – IR, USB, PDI Interface

Presented by Arpan Jati

February 15, 2012

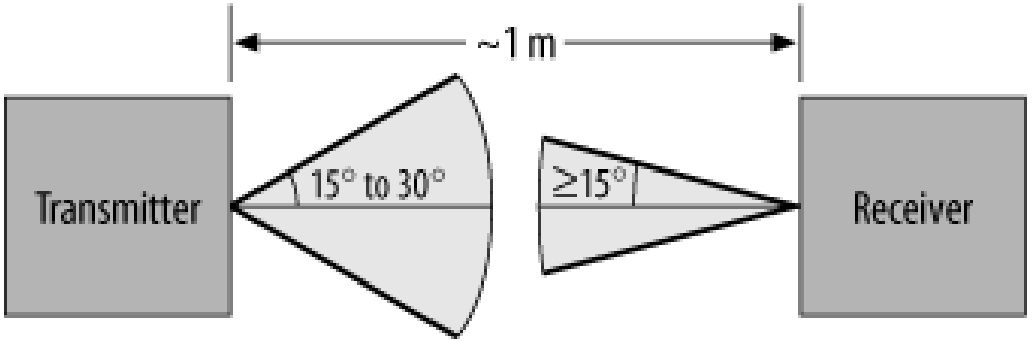
INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

Infrared Data Association (IrDA)

- Point to point protocol using asynchronous serial transfer over short distances
  - Serial interface using pulses of infrared light to transmit data
- Common in laptops, PDAs and cellphones
- Also used for remote control of electronics
- IrDA is a consortium of over 150 companies that maintain and develop the standard
- Assume that only two devices will be communicating and their proximity will exclude interference from other IrDA devices
  - Does not deal with collision and error detection issues
- Overall guiding principle is that it should be cheap to implement
- IrDA specification 1.0 supported data rates between 2400 bps and 115.2 kbps over distances of 1 meter
  - Devices negotiate the data rate up or down depending on their capabilities – How is it different from RS-232?

IrDA Transmission

- Transmitter beams out its transmission at an angle of 15 to 30 degrees on either side of line of sight
- Receiver has a viewing angle of 15 degrees on either side of the line of sight



IrDA Bit Encoding

- Uses Return to Zero (RZ) Encoding
  - Frame consists of transmission intervals divided into subintervals representing individual bits
  - Logic 0 represented by pulse for 3/16th width of bit subinterval
  - Logic 1 represented by absence of pulse

- At data rates of 4 Mbps, 4PPM (Pulse Position Modulation) is used to differentiate bits
  - Location of pulse within the subinterval determines transmitted bit pattern

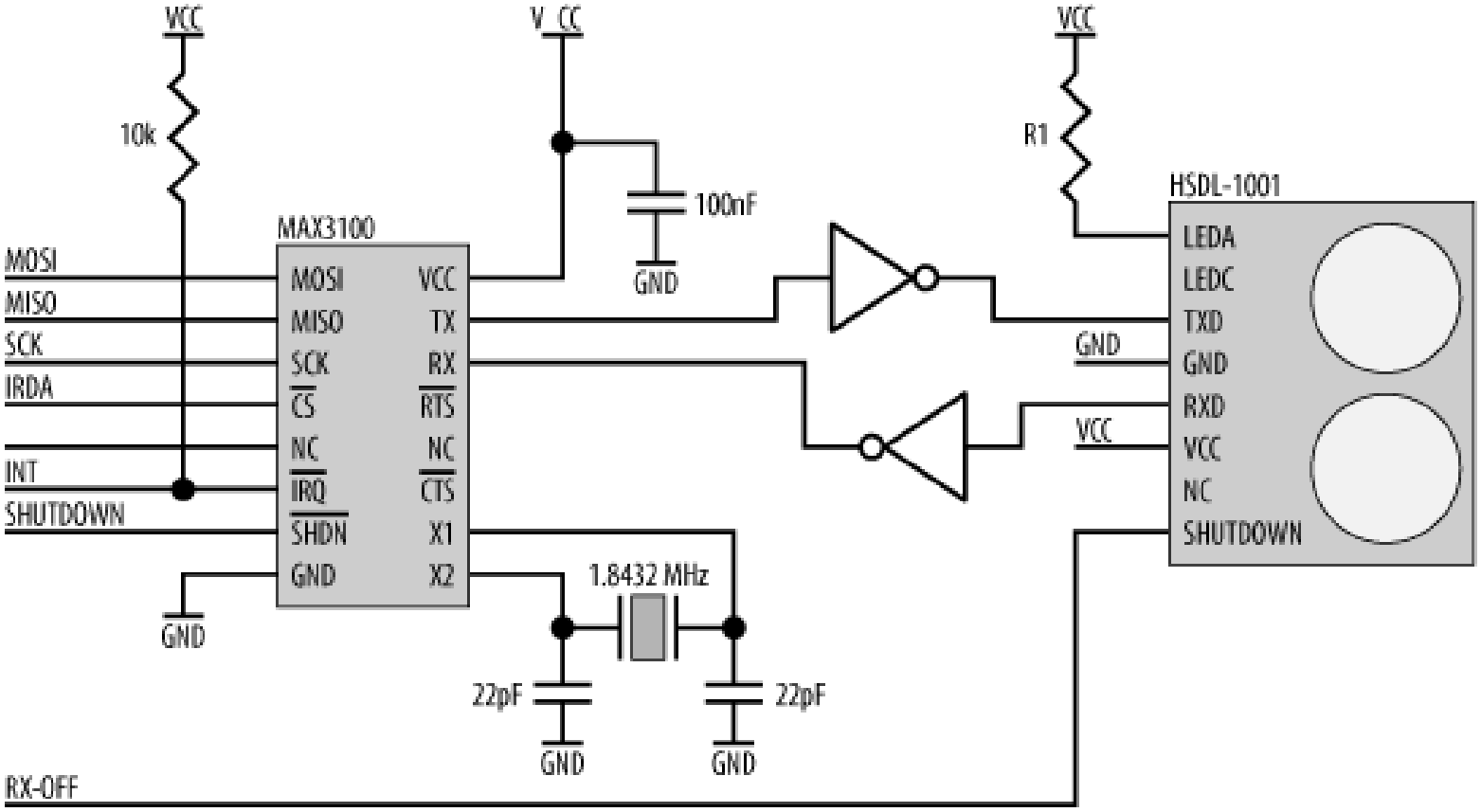
IrDA Sample Data Packet

- Sample data packet at 4 Mbps

- Most UARTs are not capable of encoding in RZ or PPM encoding
  - EnDec (Encoder-Decoder) converts standard UART output to RZ or vice versa
  - E.g. HSDL-7001 from Agilent, MCP2120 from Microchip
  - Some UARTs (MAX3100) incorporate on-chip EnDec – Directly interfaces with Infrared transceiver

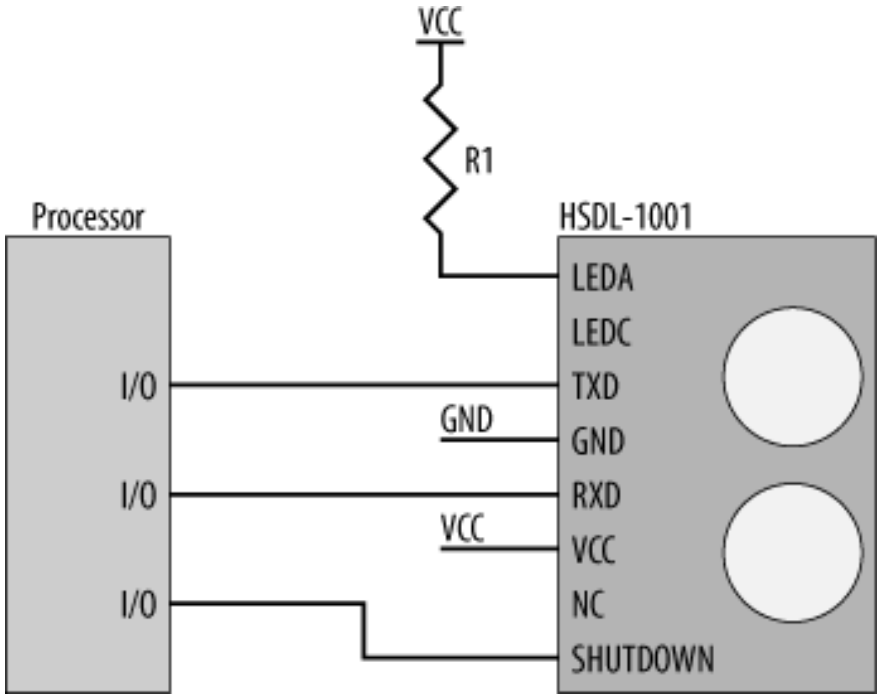
IrDA Interface to Microcontroller

- MAX3100 for converting UART (RS232 in this case) signal into SPI
  - Also compliant to IrDA: Just add any IrDA transceiver



Can you make your own RZ bit stream in software?

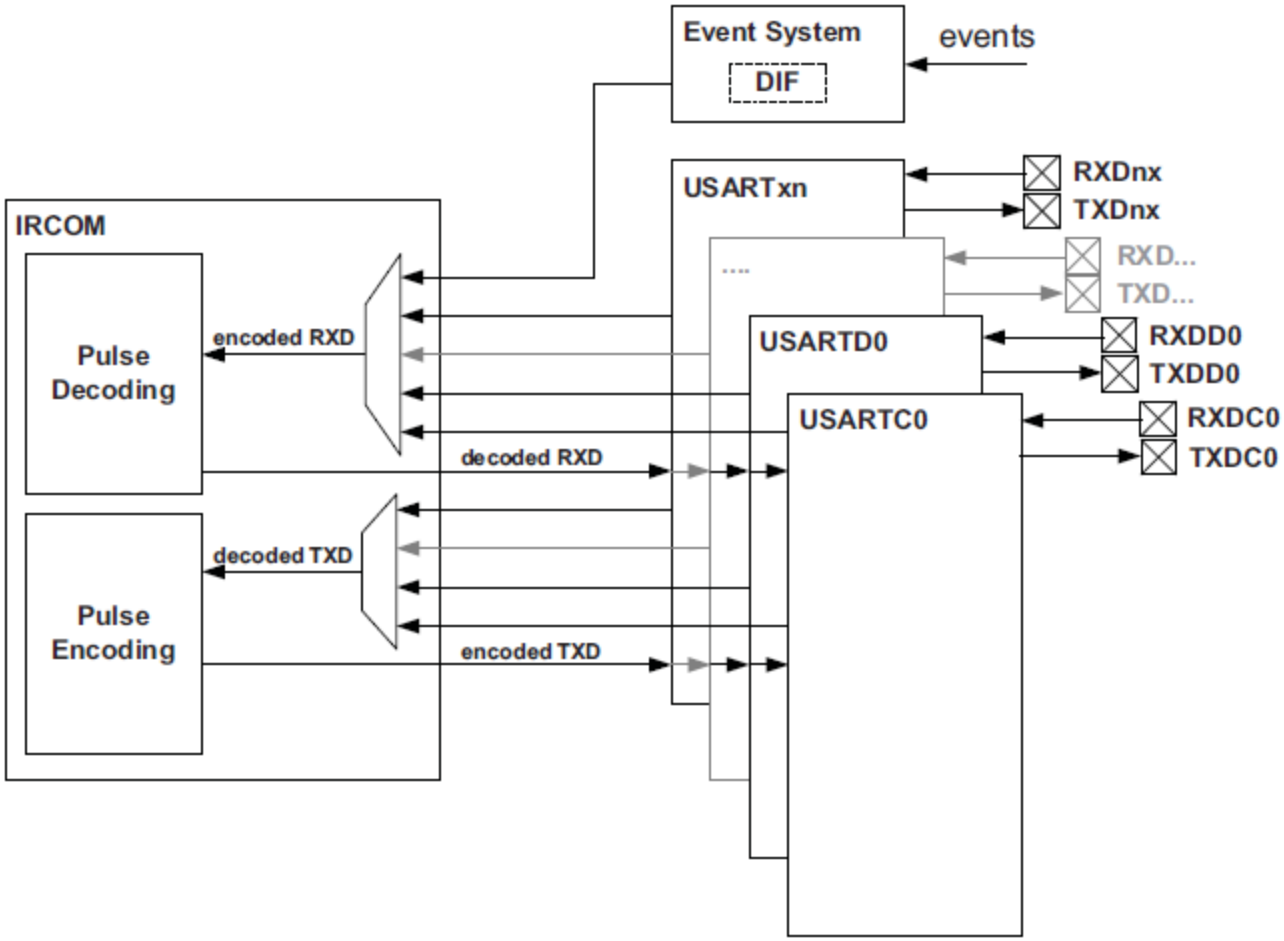
Crude IrDA Interface



- Manually toggle transmitted I/O line as appropriate
- Sample receiver using a GPIO
- Ensure correct timing in software

xMega: IR Communication Module

- xMega contains IRCOM module that is IrDA compatible
- Can be connected to any USART to enable infrared pulse encoding/ decoding



Universal Serial Bus (USB)

- Advantages:
  - Uniform connector
  - Device communicates directly with the host OS - No need for manual configuration
  - Can provide power through the same cable as data

- Communication Architecture
  - Tiered star
  - Up to 127 devices can be connected
  - Single host

- USB knows about and supports multiple device classes: Each class represents functionality provided by the device to the host (e.g. audio, infrared, mass storage, monitor, printer …)
  - A single USB peripheral can span across multiple classes

USB Packets

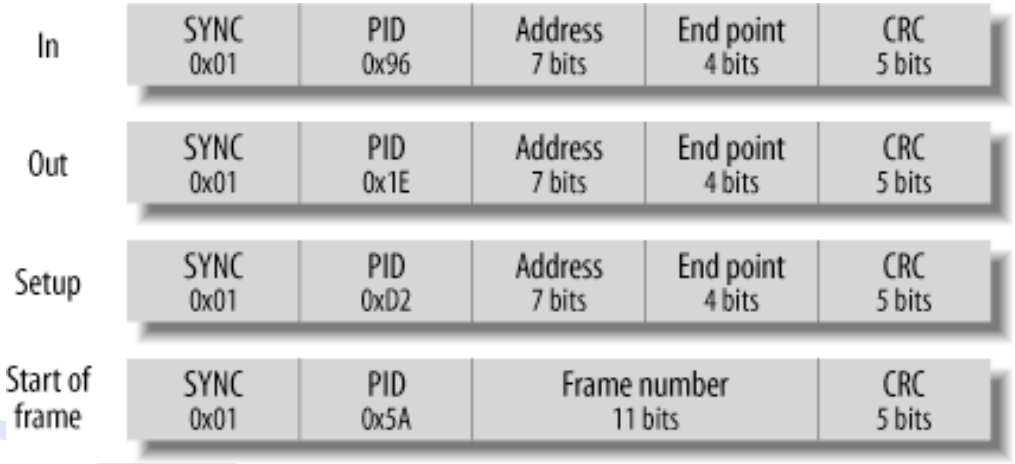
- Data transfer happens using packets (a transfer may have multiple packets)

- Packet consists of:
  - Synchronization (SYNC) byte: Phase locks the receiver clock (equivalent to start bit of RS-232)
  - Packet ID (PID): Indicates function of packet (in-built error detection)
  - Content (Data, Address)
  - Cyclic Redundancy Check (CRC)

- Four types of packets:
  - Token Packet
  - Data Packet
  - Handshake Packet
  - Start of Frame Packet

USB Packets: Token Packet

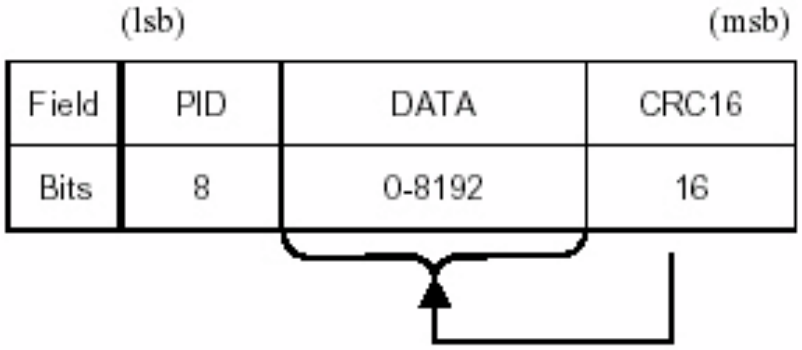
- Token packet: Indicate the type of transaction to follow
- Consists of:
  - SYNC byte
  - PID indicating packet type
  - Address of device being accessed
  - End-point address (internal destination within the device)
  - 5-bit CRC
- IN Packet: Data transfer from endpoint to host
- OUT Packet: Data transfer from host to endpoint
- Setup: Used to begin control transfers



Do you see a pattern in the PID?

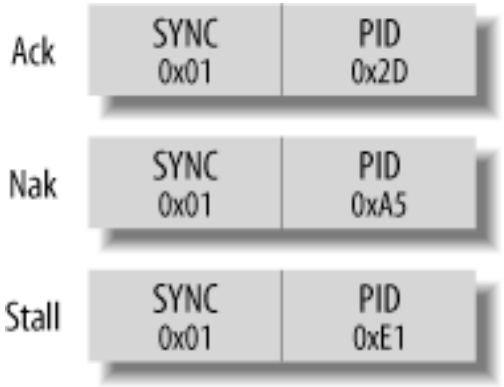
USB Packets: Data Packet

- Two types of Data packets:
  - Data0
  - Data1
  - High speed mode defines two more types - Data2 and mData
- Transmission of data packets alternate between two types
- Single data packet can transfer between 0 and 1024 bytes (in multiple of bytes)



USB Packets: Handshake Packet

- Three types of handshaking packets:
  - Ack: Successful data reception is acknowledge with Ack Packet
  - Nak: Indicates that the device is unable to accept data from host (OUT) or it has no data to transmit to the host (IN)
  - Stall: Indicate that the device is unable to receive/transmit data



USB Packets: Start of Frame Packet

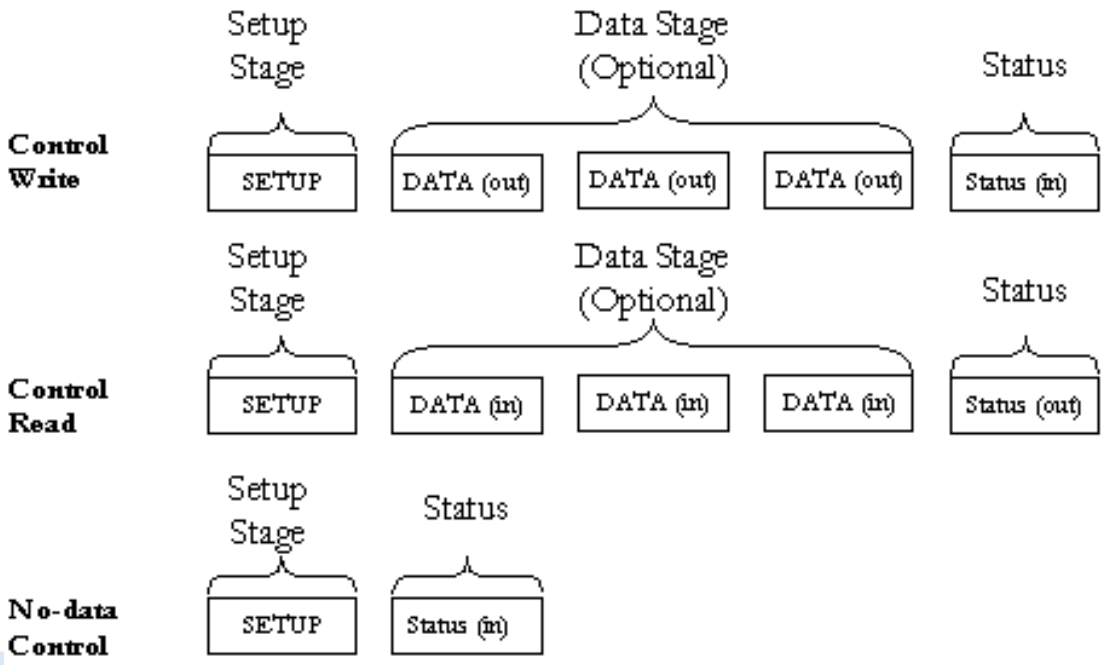
- Issued by host at nominal rate of once every 1 ms (full speed) or once every 125 uS (high speed)
  - Indicate start of a new frame
  - Contains 11 bit frame number
  - Does not cause receiving function to generate a return packet - No guarantee of delivery

USB Transfers

- Four types of transfers take place over USB:
  - Control transfer: Used to configure the bus and devices
  - Interrupt transfer: Used to retrieve data at regular intervals (ranging from 1 to 255 milliseconds)
  - Isochronous transfer: Used for moving time critical data (e.g. audio); can only be unidirectional and does not include CRC field
  - Bulk transfer: Moves the data asynchronously; can be bidirectional

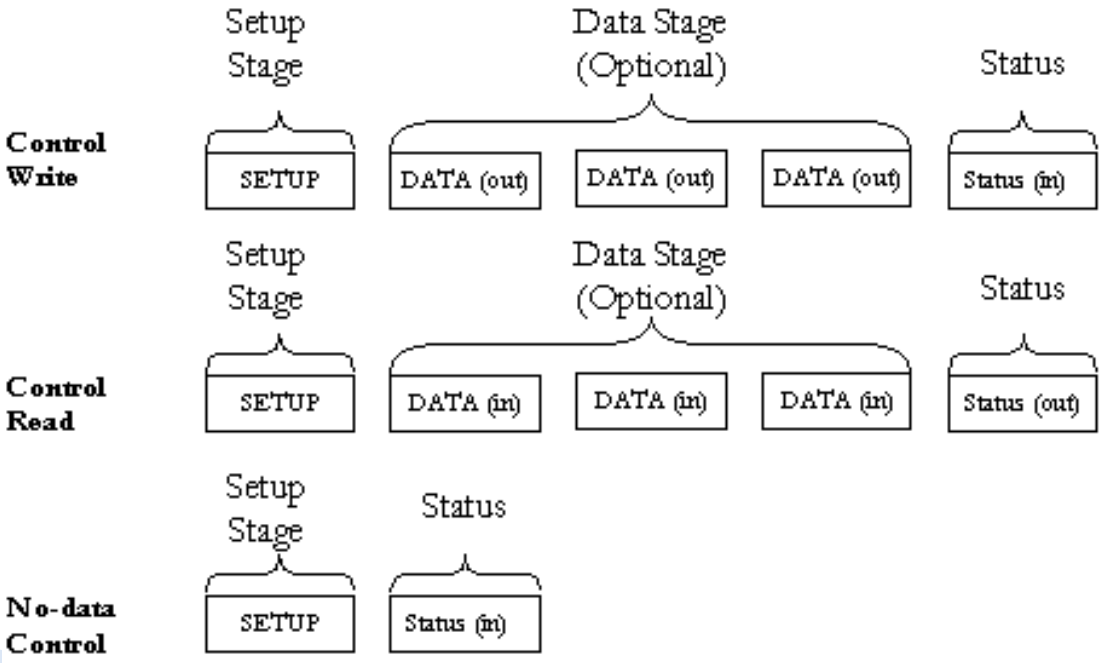
Control Transfer

- Can have up to three stages:
  - Setup stage: Sends a SETUP Token Packet specifying the device and endpoint address; a Data Packet is also sent specifying the details of type of request; endpoint responds with ACK packet if it successfully receives the setup data



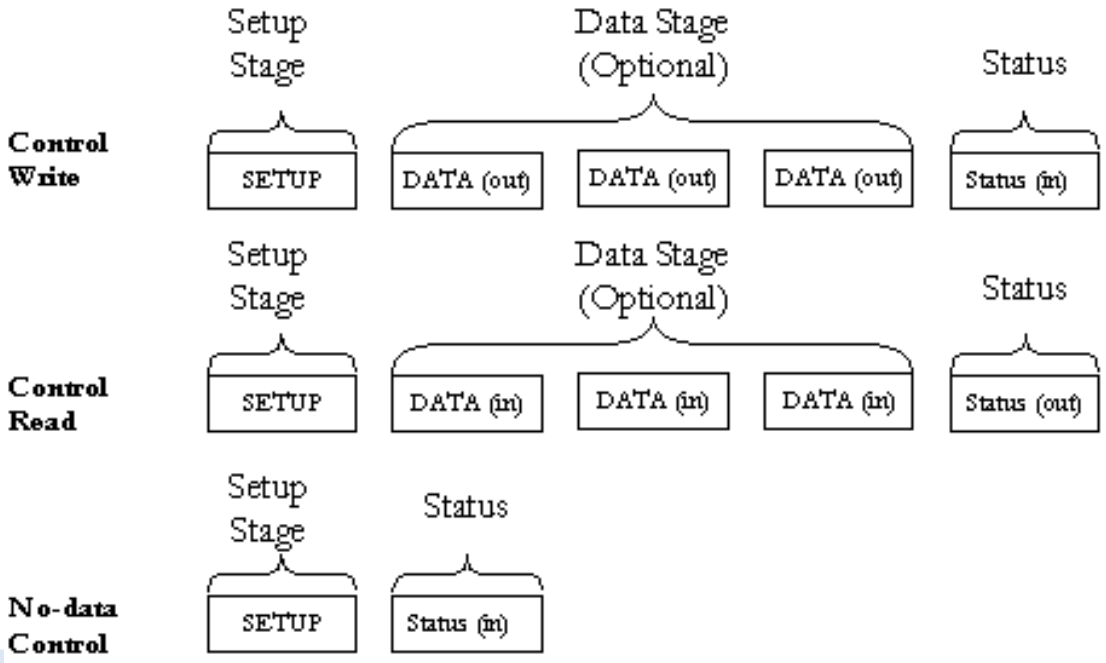
Control Transfer

- Can have up to three stages:
  - Data stage: Consists of one or more IN/OUT transfers (setup request indicates the amount of data to be transmitted in this stage)
    - IN Transfer: IN Packet -> Data/Stall/Nak Packet -> Ack Packet
    - OUT Transfer: OUT Packet -> Data Packet ->Ack/Nak/Stall Packet



Control Transfer

- Can have up to three stages:
  - Status stage:Reports the status of overall request
    - If the host sent IN tokens during the data stage to receive data then it should acknowledge the successful receipt of this data
    - If the host sent OUT tokens during data stage to send data, the device acknowledges the successful receipt of data



Control Transfer Summary

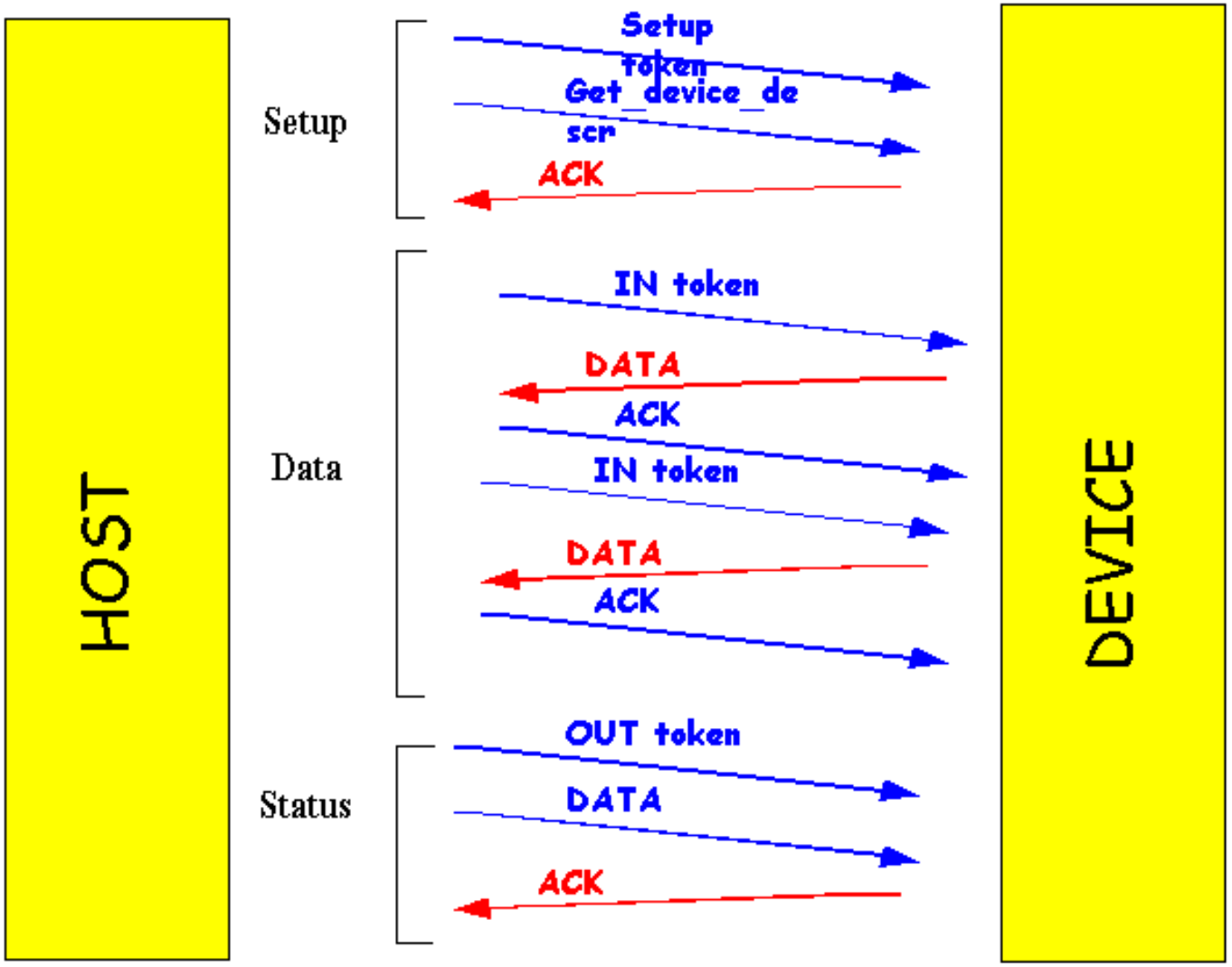
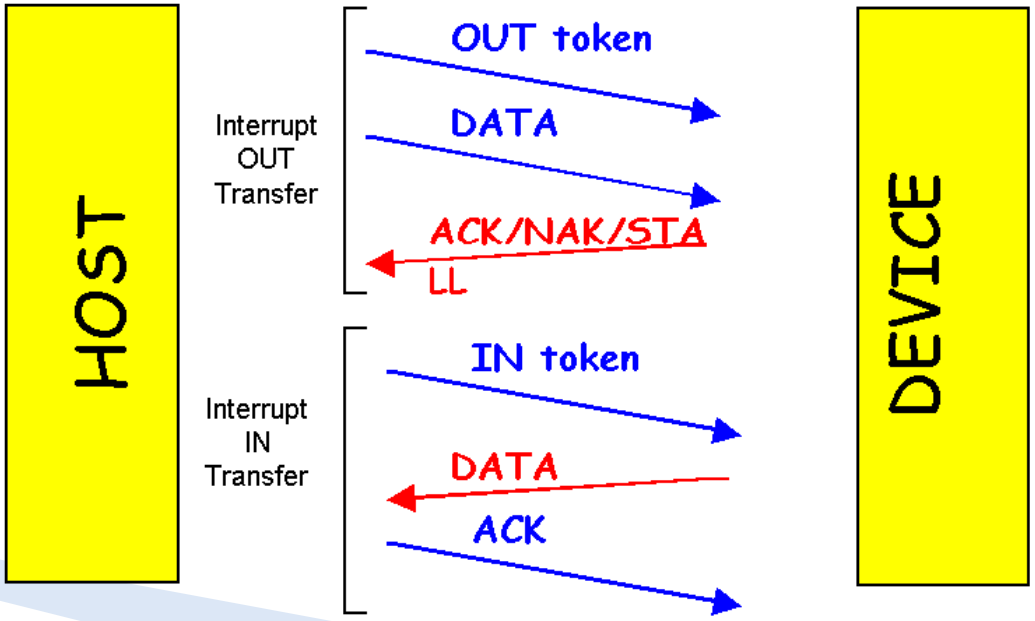


Image Source: http://www.pulsewan.com/data101/usb_basics.htm

Interrupt Transfer

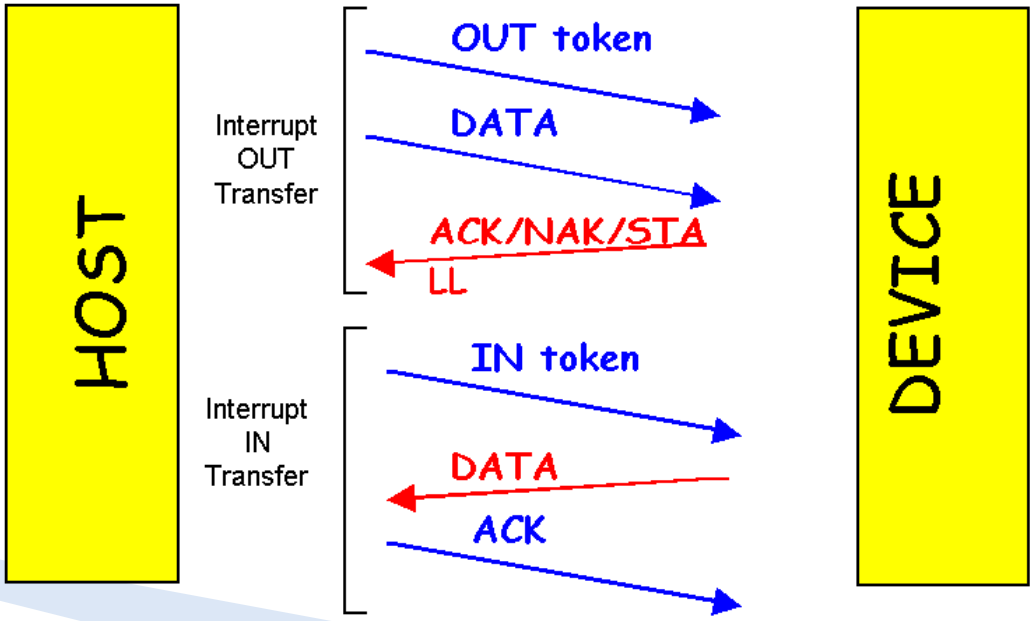
- Unlike the normal definition of interrupt - Interrupt request is queued by the device until the host polls it asking for data

- Interrupt IN transaction:
  - Host periodically polls the device sending an IN Token
  - If the device has queued interrupt, it sends details of interrupt as data packet following IN Token; Upon successful receive of data, host returns ACK Token
  - If the device does not have any queued interrupt, it returns a NAK Token



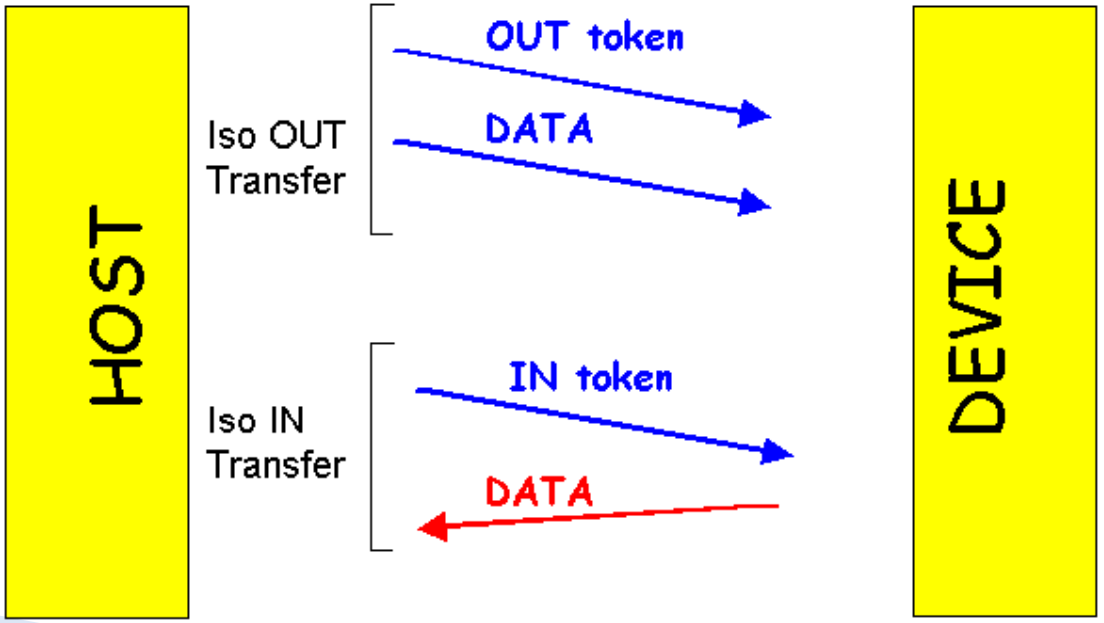
Interrupt Transfer

- Unlike the normal definition of interrupt - Interrupt request is queued by the device until the host polls it asking for data
- Interrupt OUT transaction:
  - Host issues an OUT Token if it wishes to transmit the data (e.g. to serve the interrupt)
  - Data packet is sent following the OUT Token
  - Device responds with appropriate handshake token (ACK/NAK/STALL)



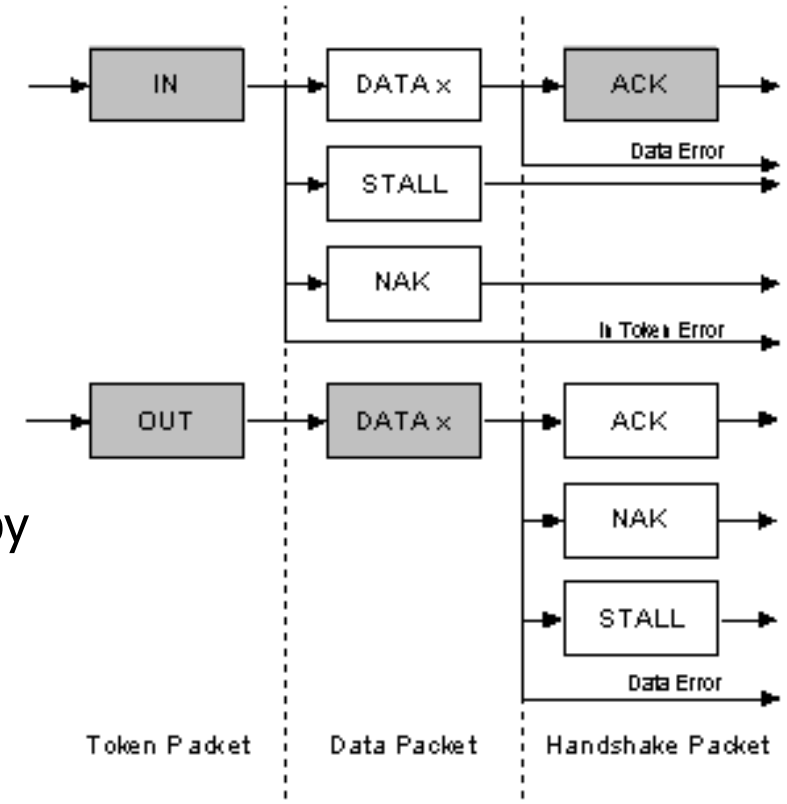
Isochronous Transfer

- Isochronous transfer occur continuously and periodically
  - Typically contain time sensitive information such as audio/video stream
- What can be cut to increase the data throughput (since the data is time critical)?
- Host issues an IN/OUT packet to read/write the data from/to device
  - In the next stage, data is transmitted in the direction specified



Bulk Transfer

- Used for large bursty data
  - Can you give some examples?
- Bulk IN transaction:
  - Host issues an IN Token
  - Device responds with Data/Stall/NAK Packet
- Bulk OUT transaction:
  - Host issues an OUT Token followed by Data packet
  - Device responds with Ack/Stall/NAK Packet



Bandwidth Management

- Host is responsible for managing the bandwidth of the bus
- Done at enumeration when configuring Isochronous and interrupt based devices as well as throughout the operation of the bus
- Specification places the upper limit of 90% of bandwidth to be allocated for periodic transfers (Isochronous/Interrupt) on a full speed bus (80% for high speed bus)
- Out of the remaining bandwidth, control transfers get preference
- Bulk transfer get its slice of what is left - Use un-allocated bandwidth on the bus after all transactions have been allocated

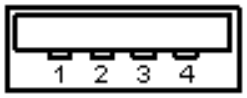
USB: Physical Interface

- Uses shielded 4-wire interface
- Data transmission over differential twisted pair (which other serial standard does it this way?) - D+, D-
- Other two wires - V_{BUS} that provides power to the device and GND
- Wires are color coded

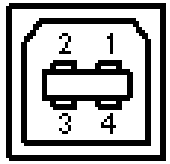
Connector Pin	Signal	Purpose	Wire Color
1	V_{BUS}	USB Device Power (+5V)	Red
2	D+	Differential data line	Green
3	D-	Differential data line	White
4	GND	Power and signal ground	Black

USB Connectors

- USB uses two types of plugs (jacks) and two types of receptacles (sockets)
  - Series A: Used for connections from a device to the host (upstream); Series A receptacle is found on the host and plug on the cable that attaches to the host
  - Series B: Receptacle is on the device and plug is at the end of cable coming downstream from the host

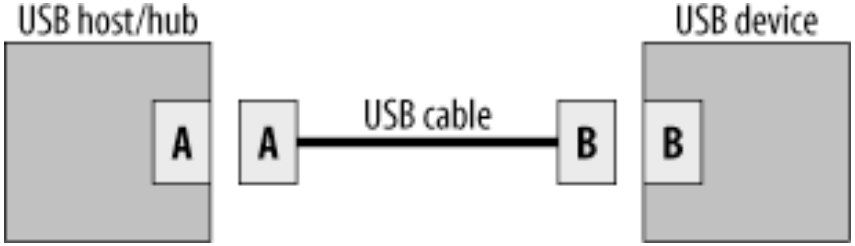


Series A: Plug and Receptacle



Series B: Plug and Receptacle

What kind of plugs and receptacles will be present on a USB hub?



USB Connectors

- Cables/connections not specified by the standard:
  - TypeA-TypeA straight cable
  - Gender changers
  - Extension cables (TypeA plug on one end and TypeA socket on other)

- Mini-USB B connectors: Included in the standard as replacement for TypeB connectors due to form factor

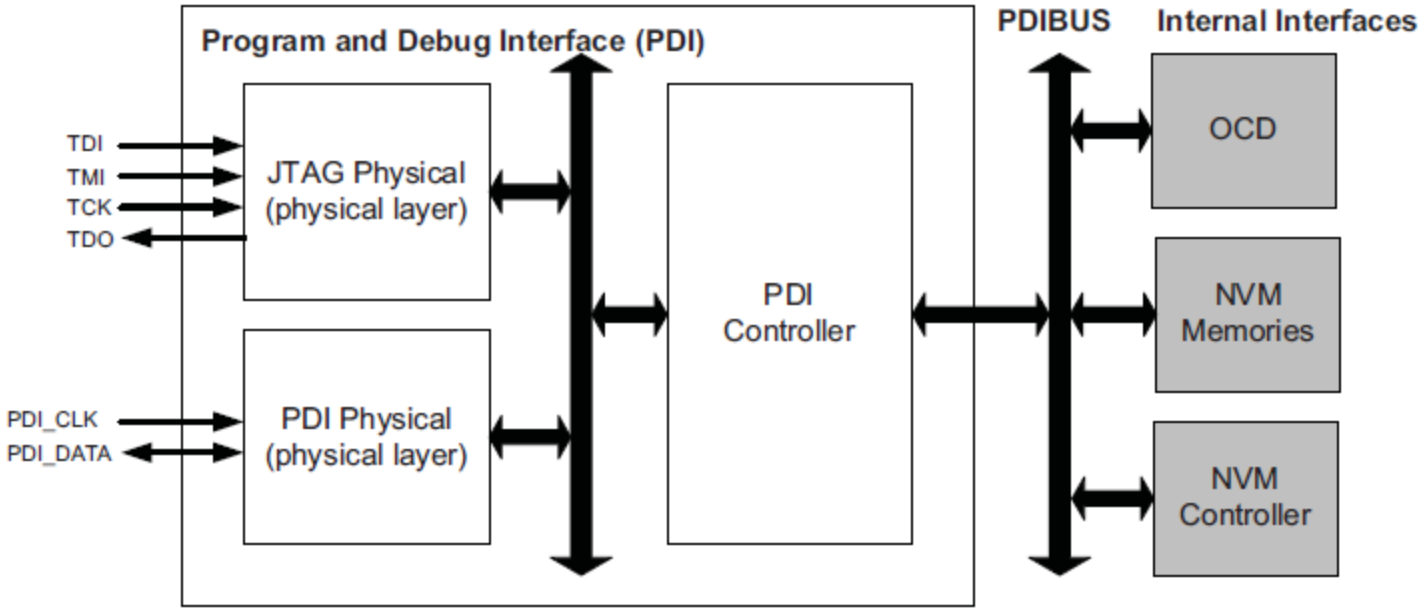
- On-the-go specification: added peer-to-peer functionality to USB
  - Introduced USB hosts into devices such as mobile phones
  - Included a specification for miniA plugs, miniA receptacles and miniAB receptacles

USB Electricals

- On the tranmitter side:
  - "1" is transmitted by pulling D+ over 2.8V and D- under 0.3 V
  - "0" is transmitted by pulling D- over 2.8V and D+ less than 0.3V
- On the receiving side:
  - "1" is defined as D+ 200mV greater than D-
  - "0" is defined as D- 200 mV greater than D+
- Typically low power devices can take up to 100 mA and high power devices can draw up to 500 mA
- Suspend mode: USB device enters suspend mode when there is no activity on the bus for more than 3.0 ms
  - After further 7.0 ms the device should enter the shutdown mode and should no longer draw more than rated suspend current
  - How do you prevent the device from entering the suspend mode?
  - Mandatory on all devices
  - Maximum suspend current is proportional to active current (for low power devices, the limit is 500 uA, for high power the limit is 2.5 mA)

PDI Interface

- Program and Debug Interface (PDI): Atmel proprietary interface for external programming and on-chip debugging
- Supports programming of Non-volatile memory (NVM): Flash, EEPROM, Fuses, Lock bits …
- Programming and Debugging possible through 2 physical interfaces:
  - 2-pin PDI Interface (PDI_CLK, PDI_DATA)
  - JTAG Interface



PDI Physical

- Uses bi-direction, synchronous, half duplex, serial receiver and transmitter
  - Similar to?

- Physical layer includes start of frame detection, frame error detection, parity generation, parity error detection and collision detection

JTAG Interface

- Can be used for two purposes:
  - Boundary scan capability to test the PCB
  - Programming the device through JTAG capability of PDI interface

- Boundary scan has the capability of driving and observing logic levels on I/O pins
  - All microcontroller (or board) components having JTAG capabilities are connected serially by TDI/TDO signals
  - External controller sets up the devices to drive values at their output pins and observes the input values received from other devices
  - Received data is compared with the expected result

JTAG Physical

- Low level serial communication over 4 lines – TDI, TDO, TCK, TMS

- JTAGEN fuse must be programmed and JTAG disable bit in MCU control register must be cleared to enable JTAG interface
  - Done by default

- JTAG PDICOM instruction sets JTAG to access PDI for external programming and on-chip debugging