Serial Communications

Prof. Stephen A. Edwards

sedwards@cs.columbia.edu

Columbia University

Spring 2006
Early Serial Communication

<table>
<thead>
<tr>
<th>Letters</th>
<th>Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>⬞</td>
</tr>
<tr>
<td>B</td>
<td>⬞</td>
</tr>
<tr>
<td>C</td>
<td>⬞</td>
</tr>
<tr>
<td>D</td>
<td>⬞</td>
</tr>
<tr>
<td>E</td>
<td>⬞</td>
</tr>
<tr>
<td>F</td>
<td>⬞</td>
</tr>
<tr>
<td>G</td>
<td>⬞</td>
</tr>
<tr>
<td>H</td>
<td>⬞</td>
</tr>
<tr>
<td>I</td>
<td>⬞</td>
</tr>
<tr>
<td>J</td>
<td>⬞</td>
</tr>
<tr>
<td>K</td>
<td>⬞</td>
</tr>
<tr>
<td>L</td>
<td>⬞</td>
</tr>
<tr>
<td>M</td>
<td>⬞</td>
</tr>
<tr>
<td>N</td>
<td>⬞</td>
</tr>
<tr>
<td>O</td>
<td>⬞</td>
</tr>
<tr>
<td>P</td>
<td>⬞</td>
</tr>
<tr>
<td>Q</td>
<td>⬞</td>
</tr>
<tr>
<td>R</td>
<td>⬞</td>
</tr>
<tr>
<td>S</td>
<td>⬞</td>
</tr>
<tr>
<td>T</td>
<td>⬞</td>
</tr>
<tr>
<td>U</td>
<td>⬞</td>
</tr>
<tr>
<td>V</td>
<td>⬞</td>
</tr>
<tr>
<td>W</td>
<td>⬞</td>
</tr>
<tr>
<td>X</td>
<td>⬞</td>
</tr>
<tr>
<td>Y</td>
<td>⬞</td>
</tr>
<tr>
<td>Z</td>
<td>⬞</td>
</tr>
</tbody>
</table>

Morse code key

1  | ⬞       |
2  | ⬞       |
3  | ⬞       |
4  | ⬞       |
5  | ⬞       |
6  | ⬞       |
7  | ⬞       |
8  | ⬞       |
9  | ⬞       |
0  | ⬞       |
RS-232

Defined in early 1960s
Serial, Asynchronous, Full-duplex,
Voltage-based, point-to-point, 100 ft+ cables

+12V \{ SPACE = 0
+3V

−3V \{ MARK = 1
−12V

Idle  Start  LSB  B1  B2  B3  B4  B5  B6  MSB  Stop

Tx
<table>
<thead>
<tr>
<th>Signal</th>
<th>DB-9 DTE</th>
<th>DCE</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RxD</td>
<td>2</td>
<td>←</td>
<td>Data received by DTE</td>
</tr>
<tr>
<td>TxD</td>
<td>3</td>
<td>→</td>
<td>Data sent by DTE</td>
</tr>
<tr>
<td>SG</td>
<td>5</td>
<td></td>
<td>Ground</td>
</tr>
<tr>
<td>DSR</td>
<td>6</td>
<td>←</td>
<td>Data Set Ready (I’m alive)</td>
</tr>
<tr>
<td>DTR</td>
<td>4</td>
<td>→</td>
<td>Data Terminal Ready (me, too)</td>
</tr>
<tr>
<td>DCD</td>
<td>1</td>
<td>←</td>
<td>Carrier Detect (hear a carrier)</td>
</tr>
<tr>
<td>RTS</td>
<td>7</td>
<td>→</td>
<td>Request To Send (Yo?)</td>
</tr>
<tr>
<td>CTS</td>
<td>8</td>
<td>←</td>
<td>Clear To Send (Yo!)</td>
</tr>
<tr>
<td>RI</td>
<td>9</td>
<td>←</td>
<td>Ring Indicator</td>
</tr>
</tbody>
</table>
Most UARTs actually use 16× clocks
Variants

Parity bit: (Even = true when even number of 1s)

Two stop bits:
Baud Rate

Baud: bits per second

<table>
<thead>
<tr>
<th>Baud</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td>ASR-33 Teletype</td>
</tr>
<tr>
<td>300</td>
<td>Early acoustic modems</td>
</tr>
<tr>
<td>1200</td>
<td>Direct-coupled modems c. 1980</td>
</tr>
<tr>
<td>2400</td>
<td>Modems c. 1990</td>
</tr>
<tr>
<td>9600</td>
<td>Serial terminals</td>
</tr>
<tr>
<td>19200</td>
<td></td>
</tr>
<tr>
<td>38400</td>
<td>Typical maximum</td>
</tr>
</tbody>
</table>
Physical Variants

Connectors: DB-25, DB-9, Mini DIN-8
RS-422: Differential signaling  RS-485: Bus-like
OPB UART Lite

Serial port peripheral for the Microblaze
Full duplex operation
16-character transmit and receive FIFOs
Parameters that can be set at build time:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Address</td>
<td>0xFEFF0100</td>
</tr>
<tr>
<td>High Address</td>
<td>0xFEFF01FF</td>
</tr>
<tr>
<td>Baud Rate</td>
<td>9600</td>
</tr>
<tr>
<td>Bits per frame</td>
<td>8</td>
</tr>
<tr>
<td>Parity</td>
<td>None</td>
</tr>
<tr>
<td>Address</td>
<td>Role</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------</td>
</tr>
<tr>
<td>0xFEFF0100</td>
<td>Read characters from Receive FIFO</td>
</tr>
<tr>
<td>0xFEFF0104</td>
<td>Write characters to Receive FIFO</td>
</tr>
<tr>
<td>0xFEFF0108</td>
<td>Status register (read only)</td>
</tr>
<tr>
<td>0xFEFF010C</td>
<td>Control register (write only)</td>
</tr>
</tbody>
</table>
## Status and Control Registers

<table>
<thead>
<tr>
<th>Bit</th>
<th>Status</th>
<th>Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>Parity Error</td>
<td>-</td>
</tr>
<tr>
<td>25</td>
<td>Framing Error</td>
<td>-</td>
</tr>
<tr>
<td>26</td>
<td>Overrun Error</td>
<td>-</td>
</tr>
<tr>
<td>27</td>
<td>Interrupts Enabled</td>
<td>Enable Interrupts</td>
</tr>
<tr>
<td>28</td>
<td>Tx buffer full</td>
<td>-</td>
</tr>
<tr>
<td>29</td>
<td>Tx buffer empty</td>
<td>-</td>
</tr>
<tr>
<td>30</td>
<td>Rx buffer full</td>
<td>Clear Rx buffer</td>
</tr>
<tr>
<td>31</td>
<td>Rx buffer non-empty</td>
<td>Clear Tx buffer</td>
</tr>
</tbody>
</table>

Non-empty Rx buffer or emptying of Tx buffer generates an interrupt.
Philips invented the Inter-IC bus c. 1980 as a very cheap way to communicate slowly among chips
E.g., good for setting control registers
100, 400, and 3400 kHz bitrates

SCL: Clock, generated by a single master
SDA: Data, controlled by either master or slaves
I\(^2\)C Bus Transaction

Idle  Start  “0”  “1”  Ack  Stop

SCL

SDA

Write data

<table>
<thead>
<tr>
<th>S</th>
<th>slave address</th>
<th>W</th>
<th>A</th>
<th>data</th>
<th>A</th>
<th>data</th>
<th>A</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

< n data bytes >

Read data

<table>
<thead>
<tr>
<th>S</th>
<th>slave address</th>
<th>R</th>
<th>A</th>
<th>data</th>
<th>A</th>
<th>data</th>
<th>A</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

< n data bytes >  last data byte

Master transmitter  Slave receiver

Master transmitter  Slave receiver

S = Start condition  R/W = read / write not
A = Acknowledge  A = Not Acknowledge
P = Stop condition
USB: Universal Serial Bus

1.5 Mbps, 12 Mbps, and 480 Mbps (USB 2.0) Point-to-point, differential, twisted pair 3–5m maximum cable length
### USB Connectors

<table>
<thead>
<tr>
<th>Series &quot;A&quot; Connectors</th>
<th>Series &quot;B&quot; Connectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>♦ Series &quot;A&quot; plugs are always oriented <strong>upstream</strong> towards the <em>Host System</em></td>
<td>♦ Series &quot;B&quot; plugs are always oriented <strong>downstream</strong> towards the <em>USB Device</em></td>
</tr>
<tr>
<td><img src="image" alt="&quot;A&quot; Plugs" /> <em>(From the USB Device)</em></td>
<td><img src="image" alt="&quot;B&quot; Plugs" /> <em>(From the Host System)</em></td>
</tr>
<tr>
<td><img src="image" alt="&quot;A&quot; Receptacles" /> <em>(Downstream Output from the USB Host or Hub)</em></td>
<td><img src="image" alt="&quot;B&quot; Receptacles" /> <em>(Upstream Input to the USB Device or Hub)</em></td>
</tr>
</tbody>
</table>
USB signaling

NRZI: 0 = toggle, 1 = no change

Bit stuffing: 0 automatically inserted after six consecutive 1s

Each packet prefixed by a SYNC field: 3 0s followed by two 1s

Low- vs. full-speed devices identified by different pull-ups on D+/D- lines
USB Packets

Always start with SYNC
Then 4-bit type, 4-bit type complemented
2 bits distinguish Token, Data, Handshake, and Special, other two bits select sub-types
Then data, depending on packet type
Data checked using a CRC
Addresses (1-128) assigned by bus master, each with 16 possible endpoints
Polled bus: host initiates all transfers.
Most transactions involve three packets:

- “Token” packet from host requesting data
- Data packet from target
- Acknowledge from host

Supports both streams of bytes and structured messages (e.g., control changes).
USB Data Flow Types

- Control
  For configuration, etc.

- Bulk Data
  Arbitrary data stream: bursty

- Interrupt Data
  Timely, reliable delivery of data. Usually events.

- Isochronous Data
  For streaming real-time transfer: prenegotiated bandwidth and latency
Layered Architecture

- Host
  - Client SW
  - USB System SW
  - USB Host Controller

- Interconnect
  - Actual communications flow

- Physical Device
  - Function
    - USB Logical Device
    - USB Bus Interface

- Function Layer
- USB Device Layer
- USB Bus Interface Layer

Implementation Focus Area

Logical communications flow
USB: Flash Card Device

Bus 001 Device 002: ID 05e3:0760 Genesys Logic, Inc.
  bcdUSB 2.00
  bMaxPacketSize0 64
  idVendor 0x05e3 Genesys Logic, Inc.
  idProduct 0x0760
  bcdDevice 1.14
  iManufacturer 2 Genesys
  iProduct 3 Flash Reader
  iSerial 4 002364

Configuration Descriptor:
  bNumInterfaces 1
  MaxPower 300mA

Interface Descriptor:
  bNumEndpoints 2
  bInterfaceClass 8 Mass Storage
  bInterfaceSubClass 6 SCSI
  bInterfaceProtocol 80 Bulk (Zip)

Endpoint Descriptor:
  bEndpointAddress 0x81 EP 1 IN
  bmAttributes 2
    Transfer Type Bulk
    Synch Type none
  wMaxPacketSize 64

Endpoint Descriptor:
  bLength 7
  bDescriptorType 5
  bEndpointAddress 0x02 EP 2 OUT
  bmAttributes 2
    Transfer Type Bulk
    Synch Type none
  wMaxPacketSize 64

Language IDs: (length=4)
  0409 English(US)
USB: Mouse Device

Bus 002 Device 002: ID 04b4:0001 Cypress Semiconductor Mouse

Device Descriptor:
- bcdUSB: 1.00
- idVendor: 0x04b4 Cypress Semiconductor
- idProduct: 0x0001 Mouse
- bcdDevice: 4.90
- iManufacturer: 1 Adomax Sem.
- iProduct: 2 USB Mouse
- iSerial: 0

Configuration Descriptor:
- bNumInterfaces: 1
- bmAttributes: 0xa0
  - Remote Wakeup
- MaxPower: 100mA

Interface Descriptor:
- bNumEndpoints: 1
- bInterfaceClass: 3 Human Interface Devices
- bInterfaceSubClass: 1 Boot Interface Subclass
- bInterfaceProtocol: 2 Mouse
- iInterface: 5 EndPoint1 Interrupt Pipe

HID Device Descriptor:
- bDescriptorType: 34 Report
- wDescriptorLength: 52

Endpoint Descriptor:
- bEndpointAddress: 0x81 EP 1 IN
- bmAttributes: 3
  - Transfer Type: Interrupt
  - Synch Type: none
- wMaxPacketSize: 4
- bInterval: 10

Language IDs: (length=4)
- 0409 English(US)
The CY7C68001 USB interface

1.0 EZ-USB SX2™ Features

1.1 Introduction

The EZ-USB SX2™ USB interface device is designed to work with any external master, such as standard microprocessors, DSPs, ASICs, and FPGAs to enable USB 2.0 support for any peripheral design.

SX2 has a built-in USB transceiver and Serial Interface Engine (SIE), along with a command decoder for sending and receiving USB data. The controller has four endpoints that share a 4-KB FIFO space for maximum flexibility and throughput, as well as Control Endpoint 0.

SX2 has three address pins and a selectable 8- or 16-bit data bus for command and data input or output.

1.2 Features

- USB 2.0-certified compliant
- Operates at high (480 Mbps) or full (12 Mbps) speed
- Supports Control Endpoint 0:
  - Used for handling USB device requests
- Supports four configurable endpoints that share a 4-KB FIFO space:
  - Endpoints 2, 4, 6, 8 for application-specific control and data
- Standard 8- or 16-bit external master interface:
  - Glueless interface to most standard microprocessors, DSPs, ASICs, and FPGAs
  - Synchronous or Asynchronous interface
- Integrated phase-locked loop (PLL)
- 3.3V operation, 5V tolerant I/Os
- 56-pin SSOP and QFN package
- Complies with most device class specifications

1.3 Block Diagram

![Block Diagram of the CY7C68001 USB interface](image-url)
The CY7C68001 USB interface

Operates as a peripheral (i.e., not a host)
Operates at 12 or 480 Mbps speeds
Control endpoint 0
Four other user-configurable endpoints
4 kB FIFO buffer
500 bytes of descriptor RAM (Vendor, Product)
I²C bus interface for configuration from EEPROM
(Unused on the XSB board—processor must configure)
Five memory locations: one for each FIFO, one for control registers

Internal registers written by first applying address to control register, then reading or writing data to control register.

33 different configuration registers, including 500-byte descriptor “register”