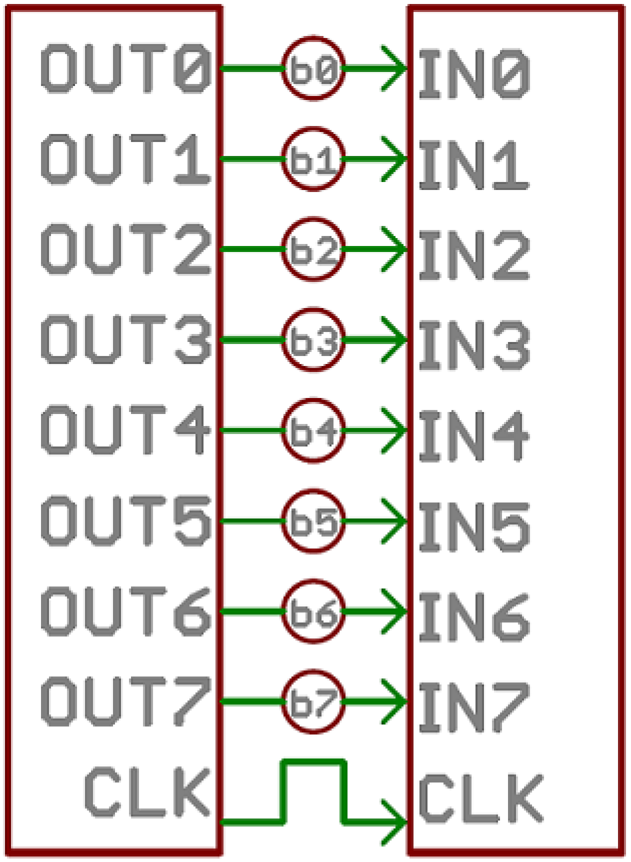
M04: Serial Communication

Prof. Rosa Zheng

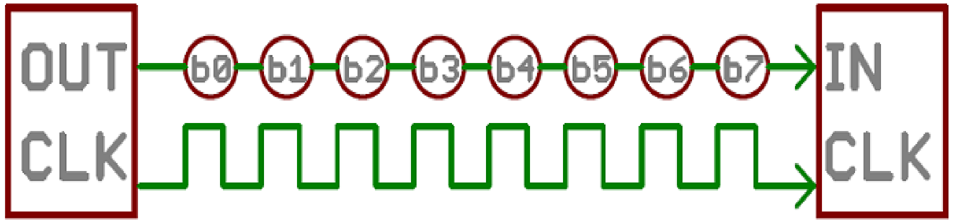
References:

1. TIVA C Workshop student workbook.
2. TM4C123G datasheet: spms376e.pdf
3. Understanding the I2C bus: http://www.ti.com/lit/an/slva704/slva704.pdf
4. SparkFun Tutorials on Serial Communications https://learn.sparkfun.com/tutorials/serial-communication/all

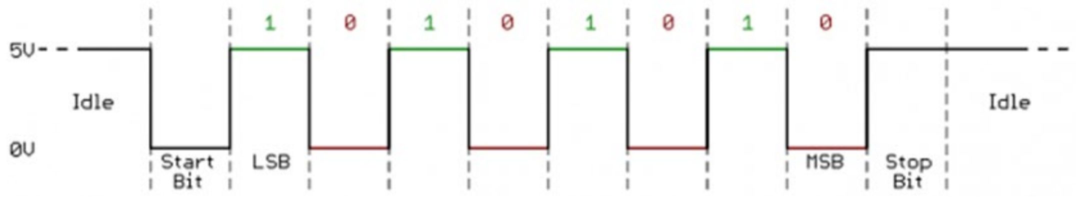
Parallel or Serial Communication



Serial



UART data frame



Reference to PuTTY or TeraTerm settings

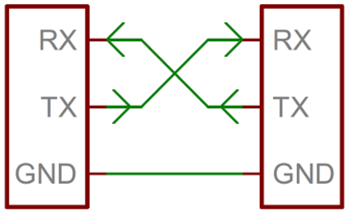
Parallel

Serial Communication Buses

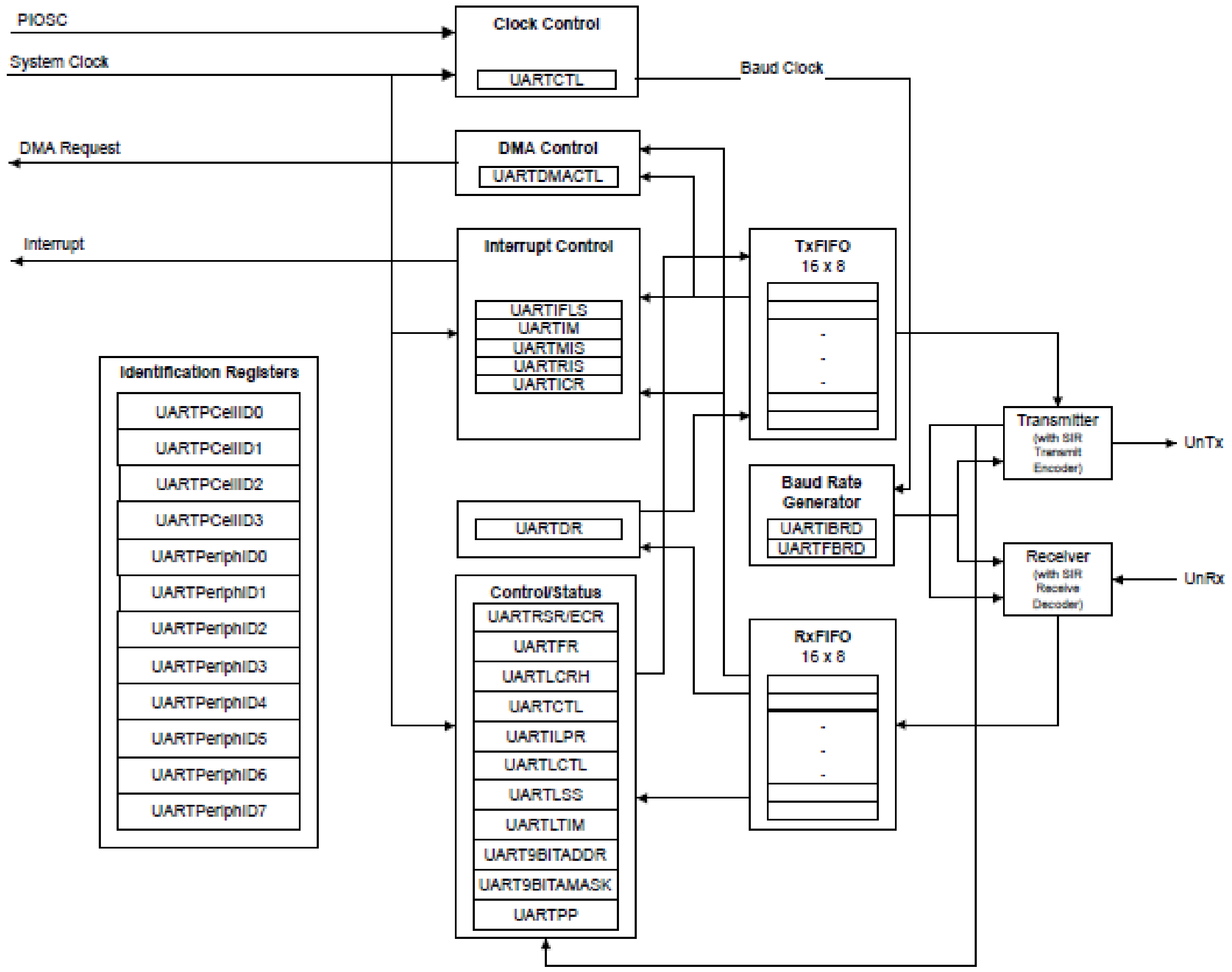
- **Universal Asynchronous Receiver/Transmitter (UART) (also USART if support synchronous)**
  - Circuitry protocol and connectors (RS232), bridging parallel bus and serial port
- **Universal Serial Bus (USB)**
  - Protocols: USB1.1, USB2.0, USB 3.0
- **Synchronous Serial Interface (SSI)**
  - Based on RS-422, differential, simplex, non-multiplexed, relies on a time-out to frame the data.
- **Serial Peripheral Interface (SPI)**
  - Single-ended, duplex, multiplex and uses a select-line to frame the data
- **Inter-Integrated Circuit (I2C or I-square-C)**
  - Synchronous, multi-master, multi-slave, packet switched, single-ended, with only two wires for physical connection.

UART

- Separate 16x8 bit transmit and receive FIFOs
- Programmable baud rate generator
  - 2400, 4800, 9600, ….. And 115,200
- Programmable serial interface
  - 5, 6, 7, or 8 data bits
  - even, odd, stick, or no parity bits
  - 1 or 2 stop bits
  - baud rate generation, from DC to processor clock/16
- Modem flow control on UART1 (RTS/CTS)
- IrDA and EIA-495 9-bit protocols
- Physical connections;



Block Diagram



Basic Operation

- Initialize the UART
  - Enable the UART peripheral, e.g.

```
SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
```

  - Set the Rx/Tx pins as UART pins

```
GPIOPinConfigure(GPIO_PA0_U0RX);
GPIOPinConfigure(GPIO_PA1_U0TX);
GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);
```

  - Configure the UART baud rate, data configuration

```
ROM_UARTConfigSetExpClk(UART0_BASE, ROM_SysCtlClockGet(), 115200,
                        UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE |
                        UART_CONFIG_PAR_NONE));
```

  - Configure other UART features (e.g. interrupts, FIFO)
- Send/receive a character
  - Single register used for transmit/receive
  - Blocking/non-blocking functions in driverlib:

```
UARTCharPut(UART0_BASE, ‘a’);
newchar = UARTCharGet(UART0_BASE);
UARTCharPutNonBlocking(UART0_BASE, ‘a’);
newchar = UARTCharGetNonBlocking(UART0_BASE);
```

UART Interrupts

Single interrupt per module, cleared automatically

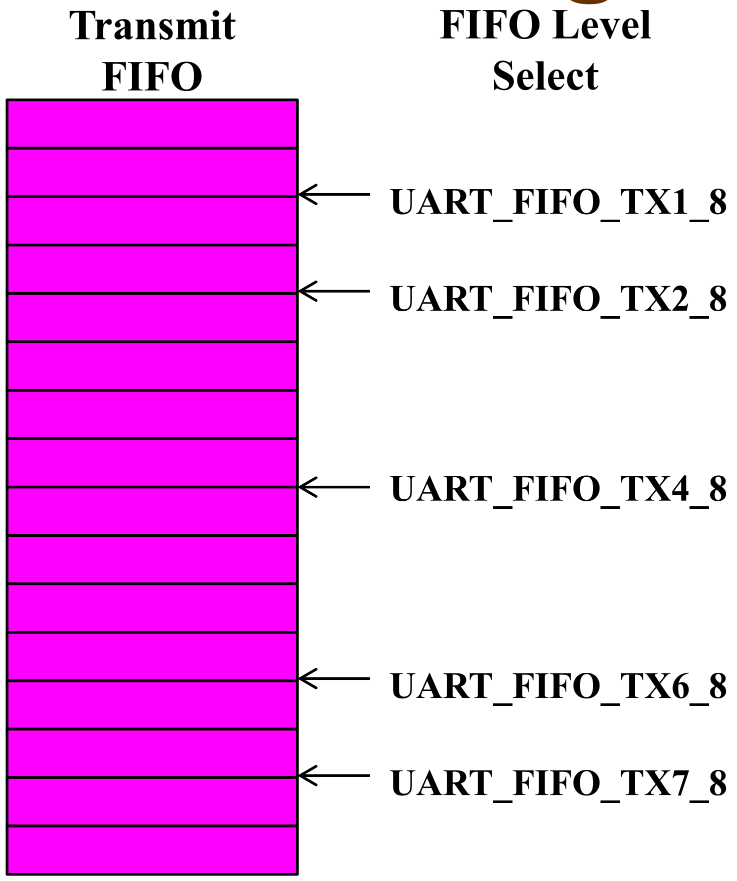
Interrupt conditions:

- Overrun error
- Break error
- Parity error
- Framing error
- Receive timeout – when FIFO is not empty and no further data is received over a 32-bit period
- Transmit – generated when no data present (if FIFO enabled, see next slide)
- Receive – generated when character is received (if FIFO enabled, see next slide)

Interrupts on these conditions can be enabled individually

Your handler code must check to determine the source of the UART interrupt and clear the flag(s)

Using the UART FIFOs



- **Both FIFOs are accessed via the UART Data register (UARTDR)**
- **After reset, the FIFOs are enabled*, you can disable by resetting the FEN bit in UARTLCRH, e.g.**

```
UARTFIFODisable(UART0_BASE);
```

- **Trigger points for FIFO interrupts can be set at 1/8, 1/4, 1/2,3/4, 7/8 full, e.g.**

```
UARTFIFOLevelSet(UART0_BASE,
  UART_FIFO_TX4_8,
  UART_FIFO_RX4_8);
```

* Note: the datasheet says FIFOs are disabled at reset

UART “stdio” Functions

- TivaWare “utils” folder contains functions for C stdio console functions:

```
c:\TivaWare\utils\uartstdio.h
c:\TivaWare\utils\uartstdio.c
```

- Usage example:

```
UARTStdioInit(0); //use UART0, 115200
UARTprintf(“Enter text: “);
```

- See `uartstdio.h` for other functions
- Notes:
  - Use the provided interrupt handler `UARTStdioIntHandler()` code in `uartstdio.c`
  - Buffering is provided if you define UART_BUFFERED symbol
    - Receive buffer is 128 bytes
    - Transmit buffer is 1024 bytes

Other UART Features

- **Modem flow control on UART1 (RTS/CTS)**
- **IrDA serial IR (SIR) encoder/decoder**
  - External infrared transceiver required
  - Supports half-duplex serial SIR interface
  - Minimum of 10-ms delay required between transmit/receive, provided by software
- **ISA 7816 smartcard support**
  - UnTX signal used as a bit clock
  - UnRx signal is half-duplex communication line
  - GPIO pin used for smartcard reset, other signals provided by your system design
- **LIN (Local Interconnect Network) support: master or slave**
- **µDMA support**
  - Single or burst transfers support
  - UART interrupt handler handles DMA completion interrupt
- **EIA-495 9-bit operation**
  - Multi-drop configuration: one master, multiple slaves
  - Provides “address” bit (in place of parity bit)
  - Slaves only respond to their address