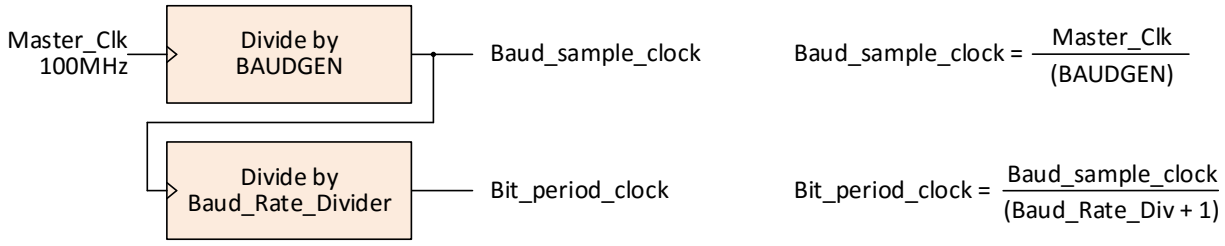
Sample Programming Problems for Test 2

1. Write a C function to setup the UART for ~38.4Kbaud. Recall there are two registers that must be programmed (Baudgen, which sets the sample clock, and Baud_Rate_Div, which sets the Baud rate). Set up the clocks so each RXD bit is sampled 16 times.



$$\text{Baud\_sample\_clock} = \frac{\text{Master\_Clk}}{(\text{BAUDGEN})}$$

$$\text{Bit\_period\_clock} = \frac{\text{Baud\_sample\_clock}}{(\text{Baud\_Rate\_Div} + 1)}$$

```
#define                UART BAUDGEN     0xE000 0018
#define                Baud_Rate_Div    0xE000 0034
void UART_config{




return; }
```

2. The bits of a 32-bit register at address 0x4B000020 are connected to 32 proximity detectors – each bit will be a '1' if it's corresponding detector is near a solid object. The bits of a 32-bit register at 0x4B001020 are connected to 32 LEDs, and writing a '1' to any bit will turn on a corresponding LED. Write a C function that can be called to illuminate LEDs 7-0 whenever the corresponding proximity detectors 7-0 are detecting an object (that is, only illuminate and LED if a corresponding detector is active). Do not change any other bits in the LED register.

3. Write a C function to poll the SPI port, and if the RX FIFO has any data in it, then read the data into "mychar".

```
#define SPI_STATUS0xE000 6004
#define SPI_RXD      0xE000 6020
void mySPI_function () {
uint32_t status = 0x0;
char mychar;
```

```
return; }
```

SPI Configuration and Status Registers			
Name	**Function**	**Address**	**Bits**
CR (Configuration Register)	SPI Configuration (enables, setups)	0x 0000 0000	18
SR (Status Register)	Interrupt Status (FIFO underflow, full, not empty, etc.)	0x 0000 0004	7
IER (Interrupt Enable Reg.)	Enables possible interrupt sources	0x 0000 0008	7
IDR (Interrupt Disable Reg.)	Disables possible interrupt sources	0x 0000 000C	7
IMR (Interrupt Mask Reg.)	Mask bits for possible interrupt sources	0x 0000 0010	7
ER (Enable Register)	Enable SPI controller	0x 0000 0014	1
DR (Delay Register)	Set various intra-frame delays	0x 0000 0018	32
TXD (Transmit Data)	SPI write data port (128 byte FIFO)	0x 0000 001C	8
RXD (Receive Data)	SPI read data port (128 byte FIFO)	0x 0000 0020	8
SICR (Slave Idle Count)	Set time in quiescent state before start detected	0x 0000 0024	8
TXWR (Transmit FIFO level)	Set transmit FIFO not full level	0x 0000 0028	7
RX_thres_req0 (Receive level)	Set receive FIFO not empty level	0x 0000 002C	7
Mod_id_reg0 (Module ID)	Read-only module ID number	0x 0000 00FC	6

Bits in SPI Status Register		
Name	**Function (definitions for '1' bit)**	**Bit#**
TXUF	Tx FIFO Underflow detected	6
RXFULL	Rx FIFO Full	5
RXNEMPTY	Rx FIFO Not Empty	4
TXFULL	Tx FIFO Full	3
TXOW	Tx FIFO Not Full	2
MODF	Incorrect mode detected	1
RXOVR	Receiver Overflow interrupt clear	0

4. Write a C function to poll the UART port on the Blackboard. When a character is received, move it into “mychar”. If more than 100 baud_sample_clocks have passed with no new character received, write a ‘1’ to “recerror”.

5. Write a function to export a string using UART1. Your function should send characters until the maximum string length of 64 is reached, or until a null (a ‘\0’ or ascii value 0) marks the end of the string.

Additional Topics to Review for Test 2

Events, faults, interrupts

Stack and context switching

Processor modes, CPSR, APSR

Assembler functions, .elf file

Interrupt handling
- Vector table
- Pending interrupts
- Taken interrupts
- Interrupt handler
  - Initialization/Setup general steps
  - Mode switch on IRQ
  - Shadow registers and context switch
  - Clear flags and RTE

UART signals, timings, protocol, and bandwidth

I2C signals, timings, protocols, and bandwidth

SPI signals, timings, protocol, and bandwidth

Serial bus comparisons: relative advantages and disadvantages; when/why to use what bus; etc.