Read Sections 8.2, 9.1, 9.2.1-4, 12.3, 12.4 (Fifo and SCI)

Recap

- Serial communication; what does the frame look like
- Baud rate vs bandwidth, latency vs real time
- SCI shift register versus SCI data register
- How RDRF is set; how RDRF is cleared
- How TDRE is set; how TDRE is cleared

Overview

- Synchronization: hardware/software, between threads
- SCI interrupts
- Fifo queue: what why how

What is a Fifo; It is a structured way to pass data

- **Fifo_Put** stores data
- **Fifo_Get** retrieves data
  - First in first out means the data remains in order
Blind Cycle Counting Synchronization
Blind cycle counting is appropriate when the I/O delay is fixed and known. This type of synchronization is blind because it provides no feedback from the I/O back to the computer.

Gadfly or Busy Waiting Synchronization
Check busy/ready flag over and over until it is ready

Interrupt Synchronization
Request interrupt when busy/ready flag is ready

Synchronizing with an input device
Synchronizing with an output device

Blind Cycle
- Write data
- Wait a fixed time

Busy-Wait
- Busy Status
- Write data
- Ready

Interrupt
- Full Fifo
- Put data into Fifo
- Empty Fifo
- Get data from Fifo
- Write data
- Return from interrupt

Run Lab 8 Transmitter on TExaS and show context switch

Finish instruction
Push registers on stack (with I=0)
Disable (I=1)
Vector fetch into PC

When does the interrupt occur?
- Armed: C0I = 1 in ritual
- Enabled: I = 0
- Triggered: C0F set when TCNT equals TC0

What happens in ISR?
- Ack: Software movb #$01,TFLG1
  Makes C0F become 0
- Read ADC data, encode, send two frames
- rti

Run Lab 8 Receiver on TExaS and show context switch

Finish instruction
Push registers on stack (with I=0)
Disable (I=1)
Vector fetch into PC

When does the interrupt occur?

- **Armed**: RIE=1 in ritual
- **Enabled**: I=0
- **Triggered**: New frame arrives, setting RDRF

What happens in ISR?

- **Ack**: Software RDRF become 0
- read status, read data
- Pass data to foreground via global memory
- Put into FIFO
- rti

How do we set the ISR vector?

Why FIFO?

- Simple unstructured globals are hard to manage
- Is there data in there?
- Are you writing new data overtop old data?
- Are you reading garbage?

Fifo is a structured way to pass data

- **Fifo_Put** stores data
- **Fifo_Get** reterives data

First in first out means the data remains in order

Real way to implement thread synchronization

- The producer needs to stall if Fifo is full
- The consumer needs to stall if Fifo is empty

Lab8 way to implement thread synchronization

- The producer throws data away if Fifo is full
The consumer waits if Fifo is empty

How is the FIFO used?

**RDRF ISR**
0) Incoming frame sets RDRf
1) clear RDRF (acknowledge interrupt)
2) every other time put 10-bit value into FIFO: **Fifo_Put**
3) rti

**Main program**
1) Initialize Timer, LCD, **Fifo_Init**, SCI
2) **Fifo_Get** (wait here until data is available)
3) Convert from 10-bit sample to decimal fixed-point
4) Display on LCD
5) repeat 2,3,4 over and over

---

**Interface Latency** is the time from trigger flag being set and the time the ISR is executed

List the steps required to implement
Fifo_Init
Fifo_Put
Fifo_Get

Jonathan W. Valvano
The following software is in Chapter 12

```
FIFO_SIZE equ 10
PutPt rmb 2
GetPt rmb 2
Fifo rmb FIFO_SIZE
Fifo_Init movw #Fifo,PutPt
        movw #Fifo,GetPt
        rts

; Input  RegA data to put
; Output RegB 1=OK, 0=full
Fifo_Put
    pshx
    ldx PutPt   ;Temporary
    staa 1,x+   ;Try to put
    cpx #Fifo+FIFO_SIZE
```

Figure 12.10. Flowcharts of the put and get operations.
bne  skip
ldx  #Fifo   ;Wrap
skip clrb
  cpx  GetPt   ;Full if same
  beq  ok
  incb       ;1 means OK
  stx  PutPt
ok   pulx
  rts
; Input  none
; Output RegA data from Get
; RegB  1=ok, 0=empty
Fifo_Get
  pshx
  clrb
  ldx  GetPt
  cpx  PutPt   ;Empty?
  beq  done
  incb       ;1=OK
  ldaa 1,x+    ; Data
  cpy  #Fifo+FIFO_SIZE
  bne  no      ;wrap?
  ldx  #Fifo  ;yes
no   stx  GetPt
done pulx
  rts

Program 12.4. Implementation of a two-pointer FIFO.

; baud rate=9600
SCI_Init jsr RxFifo_Init ;FIFO is empty
    jsr  TxFifo_Init  ;FIFO is empty
movb #$2C,SCI0CR2 ;arm just RDRF
movw #52,SCI0BD
cli
rts

* Inputs: none Outputs: RegA is ASCII
SCI_InChar
  pshb
iloop  jsr  RxFifo_Get  ;B=0 if empty
tbeq B,iloop
pulb
rts  ;A=character

* Inputs: RegA is ASCII Outputs: none
SCI_OutCh
  pshb  ;A=character
oloop  jsr  TxFifo_Put   ;save in FIFO
tbeq B,oloop  ;B=0 if full
movb #$AC,SCI0CR2 ;arm TDRE
pulb
rts

SCIhandler
  ldaa SCI0SR1
  bita #$20
  beq  CkTDRE   ;Not RDRF set
  ldaa SCI0DRL  ;ASCII character
  bsr  RxFifo_Put
CkTDRE  ldaa SCI0SR1
  bpl  sdone    ;Not TDRE set
  ldaa SCI0CR2  ;bit 7 is TIE
bpl  sdone     ;disarmed?
bsr  TxFifo_Get
tbeq B,nomore
staa SCI0DRL ;start output
bra  sdone
nomore movb #$2C,SCI0CR2 ;disarm TDRE
sdone  rti
org  $FFD6
fdb  SCIhandler

Program 12.5. An interrupting SCI (note: Lab7 uses SCI1).

The bottom line
Interrupts means software runs only when it needs to
Fifo queue is used to pass data between threads

Exam questions
Can you draw a SCI frame (voltage versus time)?
Can you implement a Fifo with 2-byte storage?
Can you tell what sets/clears RDRF / TDRE?
Given a SCI example, determine bandwidth