Lab 6—RS-232 Communication

The following routines are provided for devices with a single USART peripheral:

- **BusyUSART**: Is the USART transmitting?
- **CloseUSART**: Disable the USART.
- **DataRdyUSART**: Is data available in the USART read buffer?
- **getsUSART**: Read a string from the USART.
- **OpenUSART**: Configure the USART.
- **putsUSART**: Write a string from data memory to USART.
- **putrsUSART**: Write a string from program memory to USART.
- **ReadUSART** (or **getcUSART**): Read a byte from the USART.
- **WriteUSART** (or **putcUSART**): Write a byte to the USART.

Lab 6—RS-232 Communication

- Configuring the USART:
  
  ```c
  OpenUSART(USART_TX_INT_OFF &
              USART_RX_INT_OFF &
              USART_ASYNCH_MODE &
              USART_NINE_BIT &
              USART_CONT_RX &
              USART_BRGH_HIGH,
              32);
  ```

Both Tx and Rx interrupts disabled
Lab 6—RS-232 Communication

• Configuring the USART:

OpenUSART(USART_TX_INT_OFF &
USART_RX_INT_OFF &
USART_ASYNCH_MODE &
USART_NINE_BIT &
USART_CONT_RX &
USART_BRGH_HIGH, 32);

Configure USART
for Asynchronous I/O

The USART Clock Generator

The UART Clock Generator

Eight bit counter, clocked at fosc
The UART Clock Generator

Eight bit counter, clocked at $f_{osc}$ / $(SPBRG+1)$

Baud Rate: $f_{osc} / (SPBRG+1)$

Setting the Baud Rate

Baud Rate = $f_{osc}$ / $(SPBRG+1)$

Achievable Baud rates (BRGH=1)

Baud Rate = $f_{osc}$ / $(SPBRG+1)$
Achievable Baud rates (BRGH=1)

<table>
<thead>
<tr>
<th>BRGH</th>
<th>UART--Reading and Writing</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>char ReadUSART(void);</td>
</tr>
<tr>
<td>1</td>
<td>firstCh = ReadUSART();</td>
</tr>
<tr>
<td>1</td>
<td>nextCh = ReadUSART();</td>
</tr>
</tbody>
</table>

UART--Reading and Writing

char ReadUSART(void);

while (!DataRdyUSART()) /* busy wait*/;
firstCh = ReadUSART();
while (!DataRdyUSART()) /* busy wait*/;
nextCh = ReadUSART();

Wait for new data to arrive before reading UART

ReadUSART just reads the UART ReceiveBuffer. Doesn't wait for a new character to arrive.
UART--Reading and Writing

char ReadUSART(void);

while (!DataRdyUSART()) /* busy wait*/;
firstCh = ReadUSART();
while (!DataRdyUSART()) /* busy wait*/;
nextCh = ReadUSART();

wait for new data to arrive before reading USART

Caution: In Lab 6, do not busy wait while waiting for RS-232 input, as discussed last time in lecture

UART--Reading and Writing

void WriteUSART(char data);

while(BusyUSART()) ;
WriteUSART('H');
while(BusyUSART()) ;
WriteUSART('e');
while(BusyUSART()) ;
WriteUSART('l');
while(BusyUSART()) ;
WriteUSART('l');
while(BusyUSART()) ;
WriteUSART('o');
while(BusyUSART()) ;
WriteUSART(' ');
putsUSART("World");

WriteUSART() doesn’t check for Tx Buffer empty before writing to the USART.

Wait for USART Tx Buffer to become empty before writing another byte to it.
Minimizing Embedded System Power Consumption

• Low power consumption is especially important for:
  – battery-powered applications
  – heat-sensitive applications
• Some applications require battery-backup to remain operational though power failures
  – A “sleep mode” may be used to permit the system to retain critical state information and data
• These days, power consumption is an issue for all most all electronic devices
  – e.g. Energy Star

Factors Contributing to IC Device Power Consumption

• Supply voltage \( (V_{dd}) \)
  – Lowering \( V_{dd} \) can dramatically decrease power:
    – e.g. for DS1305
      • \( V_{cc} \) timekeeping supply current (Osc on):
        \[ \begin{align*}
        &= 81 \text{ uA at 5V} \\
        &= 25.3 \text{ uA at 2V}
        \end{align*} \]
  – Many devices have low-power versions available that can operate with low \( V_{dd} \)
    • e.g. PIC18LF452 can operate down to 2.0 V

Factors Contributing to IC Device Power Consumption--Continued

• Clock Frequency
  – Essentially a linear relationship between clock frequency and power consumption
  – Should use the lowest clock frequency suitable for the application
    • Considerations in selecting a clock frequency
      – task execution time—e.g. interrupt service time
      – timer resolution (tick rate)
      – I/O speeds (RS-232, SPI, I2C)
      – Others?
    – A good low frequency clock source for a microcontroller is a 32.768 KHz watch crystal (like the one we are using for the DS1305 in Lab 6)

Factors Contributing to IC Device Power Consumption--Continued

• I/O pins
  – floating input pins can consume power
  – unused I/O pins should be configured as outputs or pulled high or low
• Device Features
  – Generally speaking, the more features a device has, the more power it consumes
  – Should select microcontrollers and other devices with the minimum feature set needed by your application
  – Also, turn off features (modules) when they are not needed
    • Most PIC modules can be switched completely off—e.g. ADC, MSSP, USART, …
Saving Power

- **Sleep mode**
  - Many devices have an inactive (sleep) mode in which the device consumes little power.
  - Eg. PIC Microcontroller sleep mode
    - Entered by executing a `sleep` instruction
    - Puts the device into quiescent state
      - turns off oscillator
      - stops instruction execution
    - Processor can be woken up by:
      - reset operation
      - watchdog timer
      - certain interrupts

An Example

**Battery Backup**

**An Example**

**PIC Low-voltage Detect (LVD)**

Can generate LVD when Vdd drops below specified threshold

LVD ISR can put the PIC into sleep mode
An Example

Timer 1 is driven by a 32.768 KHz Osc.

Timer interrupt will occur every 2 seconds
Configure PIC to wakeup on Timer interrupt
PIC can update time-of-day clock & check if LVD is still present. If so, go back to sleep

Final Project

• Conducted during the last three weeks of class
• Assignment: Design and implement an embedded application of your choosing
• Constraint: Your system must include at least one of the following:
  – Use of a PIC feature not used in previous labs—e.g. CCP unit
  – Use of a protocol not used in labs—e.g. I2C
  – Use of a peripheral chip not used in lab
• Scope/complexity of your application must be at least comparable to that of lab 5 and lab 6.
• Stretch yourselves—more points will be awarded to more ambitious projects
Final Project

• Important Dates:
  – Project proposals due on Tues, April 10
    • Short (<1 page)
    • Provide enough detail to allow me to assess the scope and feasibility of your proposed design
  – Project proposal must be approved, before you can proceed with your project
    • Proposals will be approved/declined by Thursday, April 13
  – Proposals may be submitted any time prior to the deadline to expedite approval and ordering of parts.

• Important Dates (Continued):
  – Project Report due date is Friday, May 4 by 5:00 p.m.
    • Report format essentially same as for lab reports
    • Make sure that you provide sufficient detail regarding project specification and design.
  – Last project demonstration/ sign-off date is Thursday, May 3
  – In-class presentations: T, May 1, Th, May 3

Note: There is no Pre-Lab requirement for the Final Project