Still More Lab 6 Considerations; Embedded System Power Issues; Project Information

55:036, Embedded Systems and Systems Software

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:

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

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);
```

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

Lab 6—RS-232 Communication

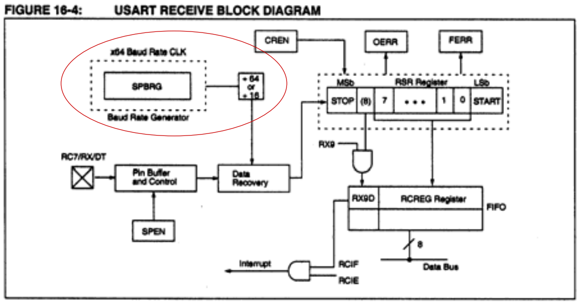
- Configuring the USART:

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

Must specify the setting for BRGH and the value for SPRRG to obtain the desired baud rate

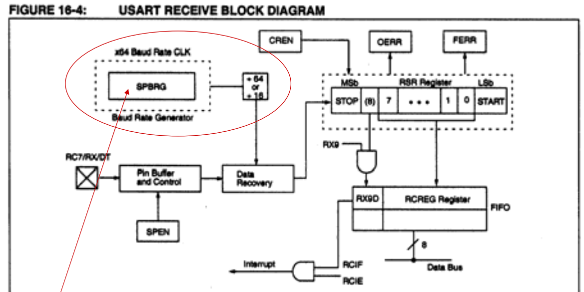
The USART Clock Generator

FIGURE 16-4: USART RECEIVE BLOCK DIAGRAM



The UART Clock Generator

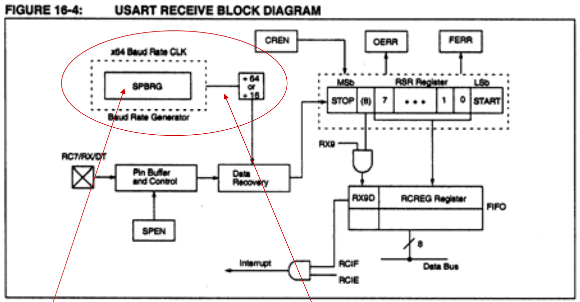
FIGURE 16-4: USART RECEIVE BLOCK DIAGRAM



Eight bit counter, clocked at f_{osc}

The UART Clock Generator

FIGURE 16-4: USART RECEIVE BLOCK DIAGRAM



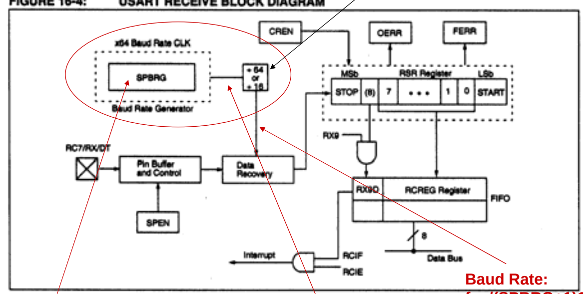
Eight bit counter, clocked at f_{osc}

f_{osc}/(SPBRG+1)

The UART Clock Generator

Determined by BRGH

FIGURE 16-4: USART RECEIVE BLOCK DIAGRAM



Eight bit counter, clocked at f_{osc}

f_{osc}/(SPBRG+1)

Baud Rate:
f_{osc}/(SPBRG+1)16
or
f_{osc}/(SPBRG+1)64

Setting the Baud Rate

	BRGH=1 (High rate)	BRGH=0 (Low Rate)
Baud Rate =	$\frac{F_{OSC}}{16(SPBRG+1)}$	$\frac{F_{OSC}}{64(SPBRG+1)}$

Relationship between BRGH, F_{osc}, SPBRG, and baud rate

Achievable Baud rates (BRGH=1)

HIGH	2500	-	0	2062.50	-	0	1562.50	-	0	1250	-	0
LOW	9.77	-	255	8.06	-	255	6.10	-	255	4.88	-	255

BAUD RATE (Kbps)	Fosc = 16 MHz KBAUD	% ERROR	SPBRG value (decimal)	10 MHz KBAUD	% ERROR	SPBRG value (decimal)	7.15909 MHz KBAUD	% ERROR	SPBRG value (decimal)	5.0688 MHz KBAUD	% ERROR	SPBRG value (decimal)
0.3	NA	-	-	NA	-	-	NA	-	-	NA	-	-
1.2	NA	-	-	NA	-	-	NA	-	-	NA	-	-
2.4	NA	-	-	NA	-	-	2.41	+0.23	185	2.40	0	131
9.6	9.62	+0.16	103	9.62	+0.16	64	9.52	-0.83	46	9.60	0	32
19.2	19.23	+0.16	51	18.94	-1.36	32	19.45	+1.32	22	18.64	-2.94	16
76.8	76.92	+0.16	12	78.13	+1.73	7	74.57	-2.90	5	79.20	+3.13	3
96	100	+4.17	9	89.29	-6.99	6	89.49	-6.78	4	105.60	+10.00	2
300	333.33	+11.11	2	312.50	+4.17	1	447.44	+49.15	0	316.80	+5.60	0
500	500	0	1	625	+25.00	0	447.44	-10.51	0	NA	-	-
HIGH	1000	-	0	625	-	0	447.44	-	0	316.80	-	0
LOW	3.91	-	255	2.44	-	255	1.75	-	255	1.24	-	255

BAUD RATE (Kbps)	Fosc = 4 MHz KBAUD	% ERROR	SPBRG value (decimal)	3.579545 MHz KBAUD	% ERROR	SPBRG value (decimal)	1 MHz KBAUD	% ERROR	SPBRG value (decimal)	32.768 kHz KBAUD	% ERROR	SPBRG value (decimal)
0.3	NA	-	-	NA	-	-	0.30	+0.16	207	0.29	-2.48	6

Achievable Baud rates (BRGH=1)

HIGH	2500	-	0	2062.50	-	0	1562.50	-	0	1250	-	0
LOW	9.77	-	255	8.06	-	255	6.10	-	255	4.88	-	255

BAUD RATE (Kbps)	Fosc = 16 MHz KBAUD	% ERROR	SPBRG value (decimal)	10 MHz KBAUD	% ERROR	SPBRG value (decimal)	7.15909 MHz KBAUD	% ERROR	SPBRG value (decimal)	5.0688 MHz KBAUD	% ERROR	SPBRG value (decimal)
0.3	NA	-	-	NA	-	-	NA	-	-	NA	-	-
1.2	NA	-	-	NA	-	-	NA	-	-	NA	-	-
2.4	NA	-	-	NA	-	-	2.41	+0.23	185	2.40	0	131
9.6	9.62	+0.16	103	9.62	+0.16	64	9.52	-0.83	46	9.60	0	32
19.2	19.23	+0.16	51	18.94	-1.36	32	19.45	+1.32	22	18.64	-2.94	16
76.8	76.92	+0.16	12	78.13	+1.73	7	74.57	-2.90	5	79.20	+3.13	3
96	100	+4.17	9	89.29	-6.99	6	89.49	-6.78	4	105.60	+10.00	2
300	333.33	+11.11	2	312.50	+4.17	1	447.44	+49.15	0	316.80	+5.60	0
500	500	0	1	625	+25.00	0	447.44	-10.51	0	NA	-	-
HIGH	1000	-	0	625	-	0	447.44	-	0	316.80	-	0
LOW	3.91	-	255	2.44	-	255	1.75	-	255	1.24	-	255

BAUD RATE (Kbps)	Fosc = 4 MHz KBAUD	% ERROR	SPBRG value (decimal)	3.579545 MHz KBAUD	% ERROR	SPBRG value (decimal)	1 MHz KBAUD	% ERROR	SPBRG value (decimal)	32.768 kHz KBAUD	% ERROR	SPBRG value (decimal)
0.3	NA	-	-	NA	-	-	0.30	+0.16	207	0.29	-2.48	6

UART--Reading and Writing

char ReadUSART(void);

```
firstCh = ReadUSART();
nextCh = ReadUSART();
```

UART--Reading and Writing

char ReadUSART(void);

~~firstCh = ReadUSART();~~
~~nextCh = ReadUSART();~~

ReadUSART just reads the USART 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

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);

```
WriteUSART('H');
WriteUSART('e');
WriteUSART('l');
WriteUSART('l');
WriteUSART('o');
WriteUSART(' ');
putrsUSART("World");
```

UART--Reading and Writing

void WriteUSART(char data);

~~WriteUSART('H');~~
~~WriteUSART('e');~~
~~WriteUSART('l');~~
~~WriteUSART('l');~~
~~WriteUSART('o');~~
~~WriteUSART(' ');~~
~~putrsUSART("World");~~

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

UART--Reading and Writing

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

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 Vdd can dramatically decrease power:
  - e.g. for DS1305
    - Vcc timekeeping supply current (Osc on):
      - = 81 uA at 5V
      - =25.3 uA at 2V
  - 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, I^2C)
      - 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)

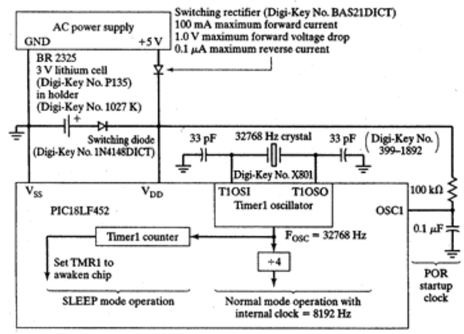
Factors Contributing to IC Device Power Consumption--Continued

- 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

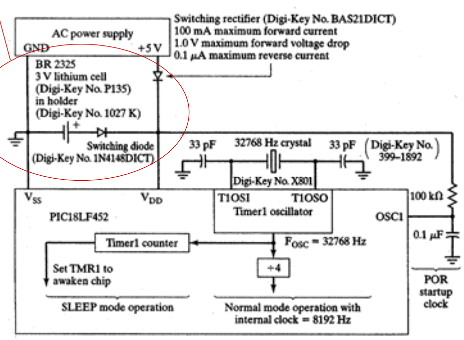
An Example



(a) Circuit

Battery Backup

An Example

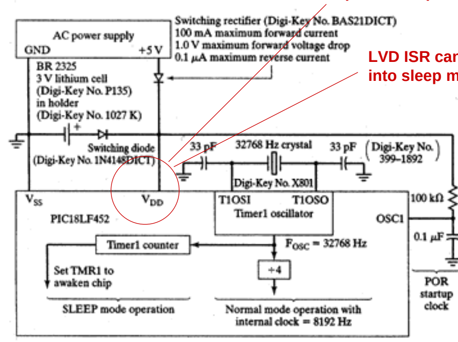


(a) Circuit

An Example

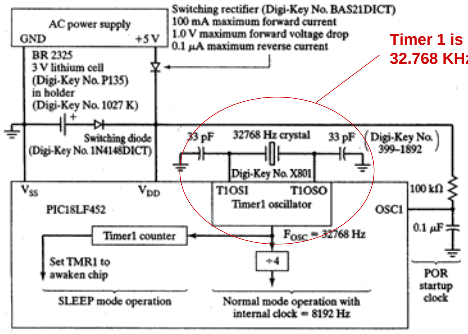
PIC Low-voltage Detect (LVD) Can generate LVD when Vdd drops below specified threshold

LVD ISR can put the PIC into sleep mode



(a) Circuit

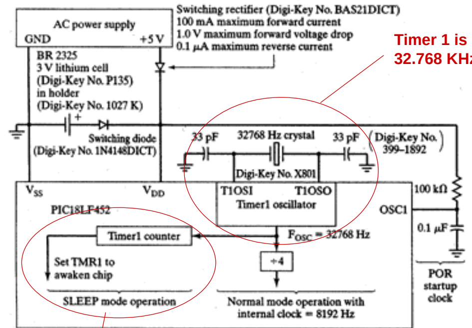
An Example



(a) Circuit

Timer 1 is driven by a 32.768 KHz Osc.

An Example

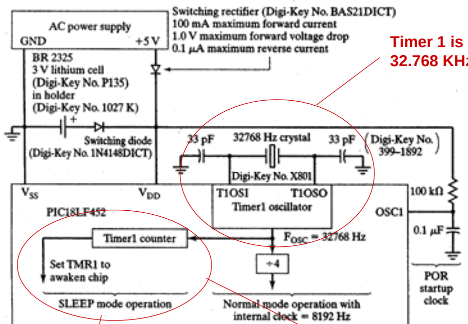


(a) Circuit

Timer 1 is driven by a 32.768 KHz Osc.

Timer1 roll-over will occur every 2 seconds.
Configure PIC to wakeup on Timer1 roll-over

An Example



(a) Circuit

Timer 1 is driven by a 32.768 KHz Osc.

Timer1 interrupt will occur every 2 seconds
Configure PIC to wakeup on Timer1 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. I^{2}C
  - 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.

Final Project

Yep, that's right—
One week from today

- 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 12
  - Proposals may be submitted any time prior to the deadline to expedite approval and ordering of parts.

Final Project

- 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