Interrupts and Serial Communication on the PIC18F8520

Kyle Persohn

COEN 4720
Fall 2011
Marquette University

6 October 2011

MARQUETTE
UNIVERSITY

Outline

1. Background
   - Serial Communication
   - PIC18 Interrupt System
2. Customizing the OpenVex ISR
   - Enabling Interrupt Sources
   - Interrupt Flags
3. Common Pitfalls
   - Concurrency Issues
   - Troubleshooting
4. Sending Data to the Netbook

Synchronous vs Asynchronous

Hardware

Synchronous

- Master/Slave using clock timing (half-duplex)
- Ex. MSSP (SPI, I^{2}C)

Asynchronous

- Framing bits identify data payloads
- Ex. UART/RS-232

Software

Synchronous

- Polling (blocking) read/write
- Ex. Xinu `kprintf()`

Asynchronous

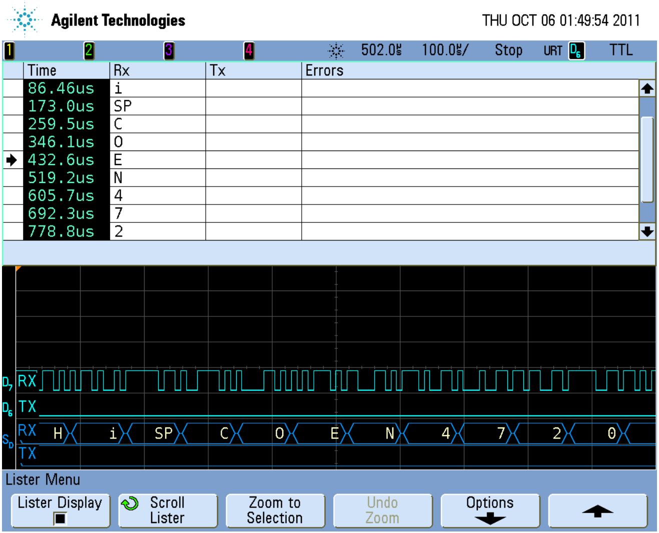
- Interrupt/message driven
- Ex. Xinu `printf()`

MARQUETTE UNIVERSITY

Vex Controller

- Asynchronous (Hardware)
- Full-Duplex
- Baud Rate: 115200 bps
- 8 data bits
- No parity
- 1 stop bit
- Abbreviated 115200/8N1
- Make sure your netbook software (picocom, etc) agrees!

What does this look like?



Agilent Technologies THU OCT 06 01:49:54 2011

	Time	Rx	Tx	Errors
	86.46us	i		
	173.0us	SP		
	259.5us	C		
	346.1us	O		
➡	432.6us	E		
	519.2us	N		
	605.7us	4		
	692.3us	7		
	778.8us	2		

Interrupt Sources

- UART RX/TX
- Timer Overflow
- External Rising/Falling Edge
- Digital I/O Port Changes State
- Analog to Digital Converter (ADC) Completes Conversion
- Capture/Compare Timer Match
- Many more (see datasheet)

What happens when an interrupt is triggered?

1. Processor *immediately* jumps to interrupt vector (high or low priority, based on source)
2. Interrupt Service Routine (ISR) determines source of interrupt (see datasheet for flag registers)
    - You (the programmer) can use pre-mapped constants in the SDCC platform header file(s)
3. Do something useful (ex. update motor speed)
4. Acknowledge interrupt (clear flag)
5. Return from interrupt

Changes to Lib/vex_usart.c

```
void usart_init(void)
{
  usart_open(USART_TX_INT_OFF & USART_RX_INT_ON &
    USART_BRGH_HIGH & USART_ASYNCH_MODE &
    USART_EIGHT_BIT, BAUD_115200);
  delay1ktcy(50);
  stdout = STREAM_USART;
}
```

Changes to Lib/interrupts.c

```
void    InterruptHandlerLow(void) INTERRUPT
{
    . . .blah blah blah. . .

    /* Timer 3 overflow interrupt */
    if ( PIR2bits.TMR3IF )
    {
        PIR2bits.TMR3IF = 0;
        ++Timer3_overflows;
    }

    if ( PIR1bits.RCIF )
    {
        rxbyte = RCREG;
        /* your code to do something with a RX char */
    }
}
```

Clearing the Interrupt Flag

Check the datasheet to see how the flag gets cleared...

- Hardware logic might clear the flag for you.
  Example: Reading from the USART receive buffer
- The user might need to manually clear the flag in software.
  Example: Timer overflow

Sharing Data

- Global variables might be necessary for sharing information between `main()` and an ISR.
- Remember: There's no test and set registers, semaphores, etc.
- Declare shared memory with the keyword `volatile`.

Atomicity

Once you turn on interrupts, your `main()` code may be preempted!

Definition

An **atomic operation** is a sequence of one or more machine instructions that are executed sequentially, without interruption.

Example

Incrementing a variable (`i+=2`) compiles to many instructions:

`load, add, store`

Disable interrupts around code blocks that might leave the system in an inconsistent state if it were to be interrupted.

Ask Yourself...

- Did you enable the peripheral? Check configuration register.
- Did you enable the peripheral interrupt? Check PIRXbits.
- Are global interrupts enabled? Check INTCON register.
- Did you recompile the OpenVex library after making changes to the interrupt vector? The library code has a separate makefile than your user-space program.

Transmitting Characters

- Check out `debug.h` for handy macros.
- OpenVex maps `stdout` to the serial port. Use `printf()`
- Manaully assign a character to the memory-mapped USART transmit buffer:

```
TXREG = ‘a’;
```

Resources

- PIC18F8520 Datasheet. Microchip Technology. 2004.
- POSIX Serial Programming Guide. `http://www.easysw.com/~mike/serial/serial.html`.
- SDCC User Guide. `http://sdcc.sourceforge.net/doc/sdccman.html/`