Background	Customizing the OpenVex ISR		Sending Data to the Netbook
00000	000	000	

Interrupts and Serial Communication on the PIC18F8520

Kyle Persohn

COEN 4720 Fall 2011 Marquette University

6 October 2011



Background 00000	Customizing the OpenVex ISR	Common Pitfalls 000	Sending Data to the Netbook 0
.			
Outling			

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



Background ●0000 Customizing the OpenVex ISR 000

Common Pitfalls

 $\begin{array}{c} \text{Sending Data to the Netbook} \\ \text{o} \end{array}$

Serial Communication

Synchronous vs Asynchronous

Hardware

Synchronous

- Master/Slave using clock timing (half-duplex)
- Ex. MSSP (SPI, I²C)

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()



Background 0●000	Customizing the OpenVex ISR	Common Pitfalls	Sending Data to the Netbook 0
Serial Communication			
Vex Contro	oller		

- 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!



nd	Customiz 000	ing the Ope	nVex ISR				tfalls		Sending Data to the Netbook O
nmunication									
t doe	s this	look	like?						
Agilent '	lechnologies					THU OC	T 06 01:49:	54 2011	
2	3	4	*	502.0불	100.0봉/	Stop	urt 🔓	TTL	
Time	Rx	Tx	Errors						
								↑	
		-							
519.2us	N								
605.7us	4								
778.8us	2							+	
	Agilent 1 Agilent 1 2 Time 86.46us 173.0us 259.5us 346.1us 432.6us 519.2us	Agilent Technologies 2 2 Time Rx 86.460us 1 173.00us SP 259.5us C 346.1us 0 432.66us E 519.2us N 605.7us 4 692.3us 7	Agilent Technologies 2 3 Time Rx Tx 86,460us 1 1 173.0us SP 2 259.5us C 3 346,11us 0 4 432.6us E 5 519.2us N 6 605.7us 4 6 692.3us 7	Agilent Technologies B B C Time Rx Tx Errors 86.46us 1 1 1 173.0us SP 2 2 3 259.5us C 3 346.1us 0 4 432.6us E 5 5 5 5 5 4 6 92.3us 7 4 6 92.3us 7 4 6 4 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 7 6 6 7 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 7 6 6 7 6 6 7 6 6 7 6 7 6 7 7 7	OOO OOO Inmunication Image: Constraint of the second se	Agilent Technologies Social Stress Image: Stress Str	OOO OOO Inmunication t does this look like? Agilent Technologies THU OC Image: Second	OOC OOC Immunication Coco Agilent Technologies THU OCT 06 01:49 Imme Rx Tx S6.460us 1 173.0us SP 259.5us C 346.1us O 432.6us E 519.2us N 605.7us 4 692.3us 7	OOO OOO nmunication t does this look like? Agilent Technologies THU OCT 06 01:49:54 2011 Image: State Sta

D,	RX					ທ່ານ		וויבוו			
D ₆	тх										
s _c	RX HX	iX	SP	cX	0)	EX	NX	4	7)	2)	0)
	ŢΧ										
L	ister Menu	l									
	Lister Dis	play 👌	Scroll Lister		m to ction		Undo Zoom		Options		♠



Background ○○○●○	Customizing the OpenVex ISR	Common Pitfalls 000	Sending Data to the Netbook 0		
PIC18 Interrupt Syster	n				
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)



Background	Customizing the OpenVex ISR	Common Pitfalls	Sending Data to the Netbook
PIC18 Interrupt System		000	
			10

What happens when an interrupt is triggered?

- 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



Background 00000	Customizing the OpenVex ISR ●○○	Common Pitfalls 000	Sending Data to the Netbook 0			
Enabling Interrupt Sou	Enabling Interrupt Sources					
Changes to	b Lib/vex_usart.	с				

```
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;
}
```



Background 00000	Customizing the OpenVex ISR ○●○	Common Pitfalls 000	Sending Data to the Netbook O
Interrupt Flag	s		
Chang	es to Lib/interrupt	cs.c	
void {	d InterruptHandlerLow		
	<pre>/* Timer 3 overflow into if (PIR2bits.TMR3IF) { PIR2bits.TMR3IF = 0 ++Timer3_overflows; }</pre>	-	
}	<pre>if (PIR1bits.RCIF) { rxbyte = RCREG; /* your code to do s }</pre>	something with a R	X char */

Background 00000	Customizing the OpenVex ISR ○○●	Common Pitfalls 000	Sending Data to the Netbook 0
Interrupt Flags			
Clearing t	he 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



Background 00000	Customizing the OpenVex ISR	Common Pitfalls ●0○	Sending Data to the Netbook 0
Concurrency Issues			
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.



Background 00000	Customizing the OpenVex ISR	Common Pitfalls o●○	Sending Data to the Netbook 0
Concurrency Issues			
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.



Background 00000	Customizing the OpenVex ISR	Common Pitfalls ○○●	Sending Data to the Netbook 0
Troubleshooting			
Ask Your	self		

- 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.



Background	Customizing the OpenVex ISR	Common Pitfalls	Sending Data to the Netbook
00000		000	●
Transmit	ting 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';



Background	Customizing the OpenVex ISR	Common Pitfalls	Sending Data to the Netbook
00000		000	0
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/

