INTRODUCTION

This application note will explain how to use the Enhanced Universal Asynchronous Receiver Transmitter (EUSART) in Asynchronous (full-duplex) mode, capable of auto-wake-up on character reception and auto-baud calibration. A 9-bit transmission is used to implement odd parity checking. This example will be implemented on the PICkit™ 1 Flash Starter Kit.

The EUSART is new to the PIC16F6XX Family of PIC® microcontrollers and includes the ability to wake-up from Sleep when there is activity on the receive pin, automatically calculate an incoming baud rate, detect when the receive operation is Idle, and transmit/receive a 12-bit Break character. Also, it is possible to specify the clock polarity in Synchronous mode and transmit polarity in Asynchronous mode. These features make it a much more versatile and easier to use communication peripheral, capable of use in a variety of applications.

The EUSART peripheral uses two registers, SPBRG and SPBRGH, to implement a 16-bit Baud Rate Generator (BRG) which supports both the Asynchronous and Synchronous modes. This allows more baud rates while decreasing bit error rates. The 8-bit mode is still possible through the BRG16 control bit in the BAUDCTL register.

To access the new features, the Baud Rate Control Register (BAUDCTL) was added to the EUSART. The BAUDCTL register is described in the “PIC16F688 14-Pin Flash-Based, 8-Bit CMOS Microcontroller with nanoWatt Technology Data Sheet” (DS41203), as shown in Table 1.

### TABLE 1: BAUD RATE CONTROL REGISTER BIT DESCRIPTION

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>ABDOVF</th>
<th>The auto-baud feature is implemented by using the SPBRG and the SPBRGH registers as a 16-bit counter for measuring the incoming baud rate. ABDOVF is a flag bit used to indicate when the BRG registers have overflowed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 6</td>
<td>RCIDL</td>
<td>This Status bit will indicate when the receive operation is Idle. Check this bit before setting the WUE bit or going to Sleep, as either would cut off any information reception.</td>
</tr>
<tr>
<td>Bit 5</td>
<td>—</td>
<td>Unimplemented.</td>
</tr>
<tr>
<td>Bit 4</td>
<td>SCKP</td>
<td>This control bit is used in Synchronous mode to indicate the Idle state is used for the data Clock (CK) and in Asynchronous mode to indicate the transmit polarity.</td>
</tr>
<tr>
<td>Bit 3</td>
<td>BRG16</td>
<td>Setting this bit will enable the 16-bit Baud Rate Generator instead of the legacy 8-bit BRG.</td>
</tr>
<tr>
<td>Bit 2</td>
<td>—</td>
<td>Unimplemented.</td>
</tr>
<tr>
<td>Bit 1</td>
<td>WUE</td>
<td>Setting this bit will activate the auto-wake-up feature and cause an RCIF interrupt on the first falling edge on the receive pin.</td>
</tr>
<tr>
<td>Bit 0</td>
<td>ABDEN</td>
<td>Setting this bit will activate the auto-baud calculation on the next character, which should be a 55h. It is cleared in hardware upon completion of the auto-baud sequence. It is left set when the BRG register overflows at which point it needs to be cleared in software.</td>
</tr>
</tbody>
</table>

**Note 1:** Refer to “PIC16F688 Rev. A Silicon Errata” (DS80181), Module 2 for earlier parts.
THE SYSTEM

Hardware
The hardware setup makes use of the PICkit 1 Flash Starter Kit. Refer to the hardware schematics in the “PICkit™ 1 Flash Starter Kit User’s Guide” (DS40051). The transceiver portion of the prototyping area needs to be populated with the following components. This includes the MAX232CPE, five 1 µF capacitors C11-C15, two 1k resistors R20-R21, the serial port connector, and the pins necessary to connect +5, GND, Tx and Rx to the PICkit 1 J3 connector. Connect GND and +5 to pins 13 and 14 respectively, on the main board J3 connector. Rx is connected to J3, pin 4 (RC5 on the PIC device), while Tx is connected to J3, pin 5 (RC4 on the PIC device). The PIC16F688 is placed in the programming socket of the PICkit 1. Both USB and serial connectors are connected to the PC.

Setup
The PICkit 1 software is used to program the PIC16F688. The PICmicro® microcontroller can not be programmed while the transceiver portion of the board is powered through the J3 connector. A solution would be to disconnect the +5 wire while the chip is being programmed, or replace R19 on the PICkit with a 1k resistor. A terminal such as HyperTerminal, can be used to communicate with the PIC device. Simply set the communications parameters to 8-bit data, odd parity, one Stop bit and no flow control. To send a Break character in HyperTerminal, press Ctrl-Break.

Firmware
The firmware implements a simple terminal program that communicates with the PC. It utilizes the EUSART in Asynchronous 9-bit Data mode using the 9th data bit to simulate odd parity. The parity must be odd so that the Sync character of “0x55” provides the auto-baud with the properly timed 5 rising edges on the Rx pin.

The main loop of the program uses a state machine to indicate the communication status by blinking one of 8 LEDs. Figure 1 shows the program flow for the main loop. Refer to Table 2 for the purpose of each state.
FIGURE 1: MAIN LOOP PROGRAM FLOW

Note 1: The State can be changed in the Receive (RC) interrupt.
The Receive (RC) interrupt routine does all the serial communication and changes the State variable when necessary. Figure 2 shows the program flow for the RC interrupt.

<table>
<thead>
<tr>
<th>TABLE 2: MAIN LOOP STATE MACHINE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>States:</strong> S0-S7, Blink LEDs D0-D7</td>
<td></td>
</tr>
<tr>
<td><strong>State</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>S0</td>
<td>Original state</td>
</tr>
<tr>
<td>S1</td>
<td>Awake, after break byte received</td>
</tr>
<tr>
<td>S2</td>
<td>Auto, baud rate calculated</td>
</tr>
<tr>
<td>S3</td>
<td>Bad character received (parity did not match up)</td>
</tr>
<tr>
<td>S4</td>
<td>“4” command received</td>
</tr>
<tr>
<td>S5</td>
<td>“5” command received</td>
</tr>
<tr>
<td>S6</td>
<td>ABOVF bit is set, the BRG has overflowed during auto-baud</td>
</tr>
<tr>
<td>S7</td>
<td>Not implemented</td>
</tr>
<tr>
<td>S8cmds</td>
<td>Command: set WUE and Sleep, requires break and “U” to re-sync</td>
</tr>
</tbody>
</table>

**FIGURE 2: RC INTERRUPT PROGRAM FLOW**
The program starts in the main loop at state zero with the WUE bit enabled. The first two characters received must be a Break followed by ASCII “U”. A small delay is required by the user in between the Break and Sync characters. LEDs D1 and D2 indicate when each character has been received.

Upon auto-calibration, a command line is presented at the terminal:

- Pushing the Enter key will display the command line again.
- A “4” or “5” will blink LED D4 or D5.
- A “?” command will list the commands recognized.
- An “S” will implement the Sleep command using State S8. The WUE bit is enabled and the chip is put to Sleep. A Break and Sync character is then required to wake-up and recalculate the baud rate. After putting the device into Sleep mode, the terminal’s baud rate can be changed before waking up the chip again.

LED D3 will indicate when a character’s parity was incorrect. D6 indicates when the BRG register has overflowed during an auto-baud calibration. This occurs when a character other than “U” is used to calibrate. A Break and Sync character is then required to recalculate the baud rate. A second Break-Sync sequence is sometimes necessary.

### Program specifics

#### BAUD RATES

A few of the higher standard baud rates might not work with the internal 8 MHz oscillator due to bit error rates. When in 16-bit mode, a BRG value of more than or equal to 4 is necessary. A BRG value of more than or equal to 25 will significantly decrease the bit error rate. Refer to the “PIC16F688 14-Pin Flash-Based, 8-Bit CMOS Data Sheet”, (DS41203) or the “PICmicro® Mid-Range MCU Family Reference Manual” (DS33023) for more information.

#### RECOVERING FROM BRG OVERFLOW

The BRG overflow occurs when the BRG counter overflows or rolls over. This will happen when the measured Rx signal remains in one state for too long. The ABDOVF bit is set and must be cleared in software. Auto-baud will remain active with the ABDEN bit set.

Figure 3 shows S6, the code used for dealing with a BRG overflow.

#### FIGURE 3: BRG OVERFLOW

```
S6
D6
CALL delay_50ms ;blink LED D6 once
OFF
CALL delay_50ms
BANKSEL PIE1
BCF PIE1,RCIE ;disable RC
;interrupt to clear
;ABDEN
;bit without
;causing an
;interrupt
BANKSEL BAUDCTL
BCF BAUDCTL,ABDEN
BCF BAUDCTL,ABDOVF
MOVF RCREG,W ;clear RCIF
BANKSEL PIE1
BSF PIE1,RCIE ;re-enable RC
;interrupt
GOTO WakeReset ;re-set Wake-up ;Enable
```

The ABDOVF bit is polled at the beginning of the main loop. Once detected, State 6 is entered. Both ABDEN and ABDOVF must be cleared in software with ABDEN being cleared first.
WAKE-UP AUTO-BAUD SEQUENCE

Figure 4 shows the RC Interrupt Service Routine.

**FIGURE 4: USART RX**

<table>
<thead>
<tr>
<th>RC_ISR</th>
<th>USART RX</th>
</tr>
</thead>
<tbody>
<tr>
<td>BTFS C flg,wue</td>
<td>;after BREAK, awake</td>
</tr>
<tr>
<td>GOTO Awake</td>
<td></td>
</tr>
<tr>
<td>BTFS C flg,auto</td>
<td>;after auto</td>
</tr>
<tr>
<td>GOTO Auto</td>
<td></td>
</tr>
</tbody>
</table>

RCvChar

| MOVF RCREG,W   | ;store data          |
| MOVWF RsSTA,W  |                      |
| MOVWF ParityBit| ;store parity        |
| CALL CalcParity| ;count character     |
| MOVF ParityBit,w| ;parity            |
| XORWF ParityByte,f| ;test for error  |
| BTFS ParityByte| f                    |
| GOTO goodChar  |                      |

badChar

| MOV LW .3      | ;bad character       |
| MOVWF STATE    | ;blink D3            |
| MOV LW .32     | ;post "Error Error" |
| MOVWF char_pos | ;on terminal         |
| CALL SendString|                      |

goodChar

| MOVF TxData,w  | ;for a command       |
| CALL CmdTest   | ;entered             |
| BTFS C flg,cmd | ;was a cmd executed?|
| GOTO ISR_DONE  | ;yes, finish        |
| CALL tx_char   | ;no, return original |

Awake

| BTFS C BAUDCTL,wUe| ;wait till done    |
| GOTO $-1          |                      |
| BSF BAUDCTL,ABDEN | ;set auto-baud      |
| BSF flg,auto      | ;calculation        |
| MOVF RCREG,W      | ;clear RCIF         |
| BCF flg,Fwue      | ;blink D1           |
| MOV LW .1          |                      |
| MOVWF STATE       |                      |

Auto

| MOVF RCREG,W  | ;clear RCIF          |
| BCF flg,auto  | ;blink D2            |
| MOV LW .2     |                      |
| MOVWF STATE   |                      |
| MOV LW .48    | ;post "type? for help" |
| MOVWF char_pos|                      |

Flags are used in the RC interrupt to determine which task needs to be done; wake-up, auto-baud, or receiving a character. A flag is also used to determine when a command was detected.

Once the WUE bit is set, any high-to-low transition at the RX pin will activate an interrupt. That means the Start bit of any character will wake-up the chip. But if it is not a Break character with all '0's, then a false end-of-character may occur and cause data or framing errors as well as a second interrupt at the real end-of-character. Since this application activates auto-baud after the first interrupt, it would distort any attempt by the auto-baud to measure the baud rate. More information can be found in Section 8.3.4 of the "PIC16F688 14-Pin Flash-Based, 8-Bit CMOS Data Sheet" (DS41203).

IMPLEMENTING PARITY

The parity implementation used is a modified version of Application Note AN774, “Asynchronous Communications with the PICMicro® USART”, (DS00774). The parity calculation routine is changed so that it produces an odd parity bit instead of an even.

CONCLUSION

The EUSART further improves the versatility and ease of use of the PICmicro® MCU in many communications applications. The added features support:

- A wider range of available baud rates using the 16-bit Baud Rate Generator (BRG)
- A power saving wake-up on character receive capability
- Automatically calculating the incoming baud rate
- The ability to implement the Local Interconnect Network (LIN) protocols as well as the J1708 automotive protocol using the 12-bit Break character transmit and receive capability.

With this application note, the user has a guide to implementing code that utilizes the added features of the EUSART for asynchronous communications.
REFERENCES

- “PIC16F688 14-Pin Flash-Based, 8-Bit CMOS Microcontrollers with nanoWatt Technology Data Sheet” (DS41203)
- “Asynchronous Communications with the PICmicro® USART”, AN774 (DS00774)
- “PICkit™ 1 Flash Starter Kit User’s Guide” (DS40051)
Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip’s Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable.”

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip’s code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.
## AMERICAS

**Corporate Office**
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277

**Technical Support:** 480-792-7627

Web Address: www.microchip.com

### Americas Branches

**Americas Corporate Office**
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277

**Technical Support:** 480-792-7627

Web Address: www.microchip.com

### Americas Branches

- **Atlanta**
  - 3780 Mansell Road, Suite 130
  - Alpharetta, GA 30022
  - Tel: 770-640-0034
  - Fax: 770-640-0307
- **Boston**
  - 2 Lan Drive, Suite 120
  - Westford, MA 01886
  - Tel: 978-692-3848
  - Fax: 978-692-3821
- **Chicago**
  - 333 Pierce Road, Suite 180
  - Itasca, IL 60143
  - Tel: 630-285-0071
  - Fax: 630-285-0075
- **Dallas**
  - 16200 Addison Road, Suite 255
  - Addison Plaza
  - Addison, TX 75001
  - Tel: 972-818-7243
  - Fax: 972-818-9294
- **Detroit**
  - Tri-Atria Office Building
  - 32255 Northwestern Highway, Suite 190
  - Farmington Hills, MI 48334
  - Tel: 248-538-2250
  - Fax: 248-538-2260
- **Kokomo**
  - 2767 S. Albright Road
  - Kokomo, IN 46902
  - Tel: 765-864-8360
  - Fax: 765-864-8387
- **Los Angeles**
  - 1300 Terra Bella Avenue
  - Mountain View, CA 94043
  - Tel: 650-215-1444
  - Fax: 650-962-9608
- **San Jose**
  - 1300 Terra Bella Avenue
  - Mountain View, CA 94043
  - Tel: 650-215-1444
  - Fax: 650-961-0286
- **Toronto**
  - 6285 Northam Drive, Suite 108
  - Mississauga, Ontario L4V 1X5, Canada
  - Tel: 905-673-0699
  - Fax: 905-673-6509

### Technical Support:
- **Americas**
  - 480-792-7627

### Technical Support:
- **China**
  - 86-10-85282100
  - Fax: 86-10-85282104

### Technical Support:
- **Europe**
  - 07/12/04