Application Report
SLAA522A–February 2012–Revised September 2012

Migrating from the USCI Module to the eUSCI Module

Priya Thanigai *MSP430 Applications*

ABSTRACT

The purpose of this application report is to enable easy migration for designs based on the USCI_A and USCI_B modules to the eUSCI_A and the eUSCI_B modules. The document highlights the new features in the eUSCI module and the main differences between the USCI and the eUSCI modules.

1 Overview

The architecture of the eUSCI module is similar to the USCI module in the F2xx, F4xx, and F5xx families. However, there are a host of new features added in the eUSCI, as well as changes made to the existing features. While most of the code written for the USCI is compatible with an eUSCI-based device, it is recommended to review the register names and make changes to the firmware accordingly.

One significant change that must be accounted for in the firmware is the servicing of the interrupt flags. The interrupt flags associated with the USCI module vectored to two different interrupt service routines, depending on whether they were data or status flags. With the eUSCI module, all the interrupt flags are prioritized and combined to source a single interrupt vector the UCAxIV or the UCBxIV. For more details on how to configure the interrupt service routine, see the code examples on the MSP430 code example page: http://www.ti.com/lit/zip/slac491.

Table 1 highlights the significant differences between the modules, and the following sections give details of the new features.

Table 1. Overview Comparison Between the USCI and eUSCI Modules

Parameter	MSP430F2xx	MSP430FR57xx
UART		
Enhanced baud rate generation	No	Yes
TXEPT interrupt (from USART)	No	Yes
Start edge interrupt	No	Yes
Selectable glitch filter	No	Yes
Interrupt vector generator	No	Yes
SPI		
Enhanced baud rate generation	No	Yes
Enhanced bit rate specs	4 to 6 MHz	10 MHz
Interrupt vector generator	No	Yes
I2C		
Preload of transmit buffer	No	Yes
Clock low timeout	No	Yes
Byte counter	No	Yes
Multiple slave addresses	No	Yes
Address bit mask	No	Yes
Hardware clear of interrupt flags	Yes	No
Interrupt vector generator	No	Yes

2 eUSCI_A: UART and SPI Modules

2.1 UART

The UART module in the eUSCI_A closely matches the USCI_A in terms of functionality and overall usage. The following features have been added for enhanced functionality:

- Additional Interrupts: UCSTTIE and UCTXCPTIE

  The UCTXCPTIE is designed to interrupt on transmit completion. This is different from TXIFG, which indicates that TX buffer is empty and can be rewritten as the byte has been moved to the shift register. TXCPTIE interrupts when the bits have been moved out from the shift register onto the bus.

  UCSTTIE is designed to interrupt at the start of a transaction; that is, reception of a start bit.

- Enhanced baud rate generation

  The modulation bits in the USCI are based on the UCBRSx setting as shown in the *BITCLK Modulation Pattern* table in the *MSP430x2xx Family User's Guide* (SLAU144). These bit patterns are fixed and cannot be modified if the user requires.

  The eUSCI module uses a free modulation scheme that allows increased flexibility when configuring the eUSCI registers for a specific baud rate.

  Also, the recommended UCBRSx settings for commonly used baud rate fractions have been included in the *USCI, UART Mode* section in the *MSP430FR57xx Family User's Guide* (SLAU272) for ease of use.

- Selectable deglitch filter lengths

  Glitch suppression is used on the receiver side to reduce the probability of 'false' starts. In the USCI module, this was implemented by a fixed deglitch filter with a deglitch time: t_{tau} = 150 ns. In the eUSCI this deglitch time can be selected to four different values. For specific values, see the device-specific data sheet.

Some eUSCI registers, while retaining the same name used in USCI, have been changed from 8 bit to 16 bit. It is recommended to access these registers using word length instructions. Key register changes are highlighted below:

- eUSCI_Ax Control 0 Register (UCAxCTL0) and eUSCI_Ax Control 1 Register (UCAxCTL1) have been combined into a single word-length eUSCI_Ax Control Word 0 (UCAxCTLW0).
- Selectable deglitch timing settings in UCAxCTL1
- eUSCI_Ax Baud Rate Control 0 Register (UCAxBR0) and eUSCI_Ax Baud Rate Control 1 Register (UCAxBR1) have been combined into a single word-length eUSCI_Ax Baud Rate Control Word Register (UCAxBRW).
- eUSCI_Ax IrDA Receive Control Register (UCAxIRRCTL) and eUSCI_Ax IrDA Transmit Control Register (UCAxIRTCTL) have been combined into a single word-length eUSCI_Ax IrDA Control Register (UCAxIRCTL).
- Interrupt Enable Register (IE2) and Interrupt Flag Register (IFG2) have been replaced by the eUSCI_Bx I2C Interrupt Enable Register (UCBxIE) and eUSCI_Bx I2C Interrupt Flag Register (UCBxIFG).
- All module interrupts are now mapped to UCAxIV.

2.2 SPI

The SPI module in the eUSCI_A closely matches the USCI_A in terms of functionality and overall usage. The following features have been added for enhanced functionality:

- Increased bit rate

  The key difference is the maximum bit rate specification. For the USCI, per the *MSP430F22x2, MSP430F22x4 Mixed Signal Microcontroller* data sheet (SLAS504), the maximum allowable bit rate was approximately 4 Mbits per second.

  The eUSCI is designed to support up to 10 Mbits per second.

- 4-pin SPI

  The 4-pin SPI mode is selected using the MODEx bits as in the USCI. In the USCI module, this pin

does not serve a true slave chip select functionality and is used only to prevent conflict with other masters. In the eUSCI module, this mode serves two purposes – it can either prevent conflict with other masters or be used to generate the enable signal for a corresponding 4-wire slave. The functionality is determined based on the setting of the UCSTEM bit in the UCAxCTLW0 register.

Some registers, while retaining the names, have been changed from 8 bit to 16 bit. It is recommended to access these registers using word length instructions. Key register changes are highlighted below:

- UCAxCTL0 and UCAxCTL1 have been combined into a single word register UCAxCTLW0.
- UCAxBR0 and UCAxBR1 have been combined into a single word register UCAxBRW.
- UCAxSTAT is now a word length eUSCI_Ax Status Register (UCAxSTATW)
- Interrupt changes are similar to the UART module

3 eUSCI_B: SPI and I2C Modules

The SPI module on the eUCSI_B is identical to the eUSCI_A . Therefore, the discussion in this section is focused on the eUSCI I2C module.

3.1 I2C

3.1.1 Functional Changes to the I2C State Machine

This section deals with the key modifications in the hardware that require software changes when porting code from the USCI I2C to the eUSCI I2C module.

- **Hardware Clear of Interrupt Flags**

  One of the main considerations when migrating to the eUSCI is the clearing of interrupt flags in I2C mode. In the USCI module, certain interrupt flags were cleared automatically in hardware triggered by events on the I2C bus. Examples of such interrupt flags and their corresponding bus clearing events are listed below:

  – Master and slave mode: UCTXIFG cleared by NACK from receiver
  – Master mode only: UCNACKIFG cleared by a start condition on the bus
  – Slave mode only: Start interrupt flag UCSTTIFG cleared by a stop condition on the bus
  – Slave mode only: Stop interrupt flag UCSTPIFG cleared be a start condition on the bus

  In the eUSCI module, all of the above mentioned flags are required to be cleared in software by the user either in response to servicing an interrupt or based on a related bus event.

  For example consider the case where the eUSCI is configured as a master transmitter and the slave 'NACKs' on the Nth byte. UCTXIFG is set once more on the (N+1)th instance. In the TX ISR, the application code is expected to check for UCNACKIFG = 1 and clear any pending TXIFGs in software.

  Another example is when the eUSCI is configured as a slave transmitter, a NACK does not clear pending TXIFGs and the application code is expected to clear the pending UCTXIFG in the UCSTPIFG interrupt service routine, for instance, after the I2C stop condition has occurred. However, this is application dependent, as the TXIFG may already have been serviced based on when the STOP occurs on the bus. In this case, the application code may use the byte counter feature to determine if an extra byte has been loaded to UCTXBUF and then update the transmit pointers accordingly.

- **Clock Stretching on Address Acknowledge Cycle**

  In the USCI module, the slave transmitter was designed such that on receiving the start condition from the master, the slave would clock stretch on the acknowledge cycle of the slave address until the transmit buffer was loaded with the first byte. In applications where the master does not support clock stretching, the MSP430 slave needs to ensure the highest priority is given to the I2C TX Interrupts (by disabling other interrupts), to prevent clock stretching.

  In the eUSCI module, clock stretching on the address acknowledge cycle can be prevented by preloading the transmit buffer. This is achieved by setting the UCPRELOAD bit in the UCBxSTAT register. This allows the transmit buffer to be loaded before a start condition is received, thereby, preventing the need to clock stretch on the first acknowledge.

3.1.2 Feature Additions and Improvements

- **Multiple-Slave Addresses**

  This is one of the most significant differences with the eUSCI I2C module. The eUSCI module is capable of allowing the MSP430 to act as multiple I2C slaves with unique slave addresses that are user programmable. It supports up to four slave addresses in hardware with dedicated transmit and receive interrupt flag for each slave address.

  Each slave address is enabled by setting the corresponding enable bit. The priority of the slave addresses is based on the register index; that is, UCBxI2COA0 has the least priority.

  As each slave address has an associated transmit and receive interrupt flag, the ISRs can be configured to react according to the slave address that is matched on the bus.

  An example application would be using two slave addresses on a single MSP430: one for a sensor application and the other as an EEPROM as shown in the following example.

NOTE: Status flags are shared between multiple slave addresses.

```
UCB0I2COA0 = 0x48 // This is the EEPROM address
UCB0I2COA1 = 0x40 // This is the sensor address
//......USCI Init......
#pragma vector = USCI_B0_VECTOR
__interrupt void USCI_B0_ISR(void)
{
  switch(__even_in_range(UCB0IV,30))
  {
  case  0: break;                           // Vector  0: No interrupts
  case  2: break;                           // Vector  2: ALIFG
  case  4: break;                           // Vector  4: NACKIFG
  case  6: break;                           // Vector  6: STTIFG
  case  8:                                  // Vector  8: STPIFG
    UCB0IFG &= ~UCSTPIFG;                   // Check RX byte counter
      __bic_SR_register_on_exit(LPM0_bits);
    break;
  case  10: break;                          // Vector  10: SA3 RX
  case  12: break;                          // Vector  12: SA3 TX
  case  14: break;                          // Vector  14: SA2 RX
  case  16: break;                          // Vector  16: SA2 TX
  case  18:                                 // Vector  18: SA1 RX
     // Place Sensor RX related code here
     Break;
  case  20: break;                          // Vector  20: SA1 TX
    // Place Sensor TX related code here
     Break;
  case  22:                                 // Vector  22: SA1 RX
     // Place EEPROM RX related code here
     Break;
  case  24: break;                          // Vector  24: SA1 TX
    // Place EEPROM TX related code here
     Break;

  case 26: break;                           // Vector 26:byte counter
  case 28: break;                           // Vector 28:clock timeout
  case 30: break;                           // Vector 30: UCBIT9IFG
  default: break;
  }
}
```

- **Address Bit Masking**

  The address bit masking in combination with the software selection of address acknowledge allows the eUSCI to respond to up to 2^{10} slave addresses in software, making it a powerful and flexible feature.

  In the USCI module, for a slave device, the response to the address byte was controlled by the hardware. In the eUSCI module, it is now selectable by using the UCSWACK bit in the UCBxCTL2 register. This gives control to the application which can either:

  – Transmit an ACK by setting the UCTXACK bit, or

  – Transmit a NACK by setting the UCTXNACK bit

  Note that when using the software ACK or NACK feature, the user is expected to set the corresponding ACK or NACK bit in time; otherwise, the module stretches the clock until the bit is set.

  This feature can also be leveraged in multi-master mode. For example, if UCB0I2COA = 100 0000 and UCB0ADDMASK = 0x01 (last position is bit masked), then the eUSCI module responds to both 100 0000 and 100 0001 with an ACK, and the module automatically transitions to a slave receiver if any one of these addresses are seen on the bus.

- **Clock Low Timeout**

  The clock low timeout feature is implemented using the internal MODOSC. No additional resources, such as an external crystal or timers, are required to use this feature. There are three selectable intervals for the timeout. For more details regarding delay times, see the *MSP430FR57xx Family User's Guide* (SLAU272).

  This feature allows the eUSCI module to be compatible with the system management bus (SMBus) protocol. Per the protocol specification, the device that is stretching the clock is required to timeout after a maximum timeout interval of 10 ms. The eUSCI internal timer maintains this timeout interval and supplies feedback via a flag, UCCLOTIFG, to indicate that a timeout has occurred.

  As the timeout flag is interrupt driven, the interrupt service routine associated with it can be used to perform further processing on the application side. The master or slave may choose to release the clock line and reset the hardware I2C module on detection of a timeout.

- **Byte Counter and Auto Stop Insertion**

  The byte counter is useful in both master and slave modes, as it eliminates the need for software counters. When using the eUSCI in master mode, the byte counter is especially useful when used in conjunction with the automatic stop generation. An I2C stop is automatically generated once the byte counter counts down to zero. Note that the byte counter is a direct reflection of the number of bytes seen on the bus. Regardless of the whether the byte is acknowledged or not, if it is on the bus, the byte counter increments.

  In Slave mode the byte counter holds the count of the number of bytes between a start and a stop (or restart).

- **Selectable Deglitch Intervals**

  In certain applications sudden unplugging of master/slave devices on the I2C bus can be detected falsely as an I2C start or stop condition. The deglitch filter protects against this and is software configurable for different intervals.

  Most of the register names and the functionality of the bits have been changed in the eUSCI module. For more information on porting firmware, see the *MSP430FR57xx Family User's Guide* (SLAU272).

4 References

1. *MSP430FR57xx Family User's Guide* (SLAU272)
2. *MSP430x2xx Family User's Guide* (SLAU144)
3. *MSP430F22x2, MSP430F22x4 Mixed Signal Microcontroller* data sheet (SLAS504)
4. *MSP430FR573x, MSP430FR572x Mixed Signal Microcontroller* data sheet (SLAS639)
5. SMBus Protocol Specification: http://smbus.org/

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components which meet ISO/TS16949 requirements, mainly for automotive use. Components which have not been so designated are neither designed nor intended for automotive use; and TI will not be responsible for any failure of such components to meet such requirements.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265