Engineer-to-Engineer Note EE-354

ANALOG DEVICES **Technical notes on using Analog Devices DSPs, processors and development tools**
Visit our Web resources http://www.analog.com/ee-notes and http://www.analog.com/processors or e-mail processor.support@analog.com or processor.tools.support@analog.com for technical support.

UART Enhancements on ADSP-BF60x Blackfin® Processors

Contributed by Andreas Pellkofer

Rev 1 – August 1, 2012

Introduction

Compared to earlier ADSP-BF5xx Blackfin® processors, the ADSP-BF60x series introduces new features to the UART module. This EE-Note summarizes the changes, describes their benefits, and assists in porting code.

The new features are grouped into the following categories:

- Automatic RTS/CTS hardware flow control (introduced with ADSP-BF54x processors)
- Increased receive FIFO size
- Finer bit rate granularity (introduced with ADSP-BF54x processors)
- More efficient programming model (e.g., 32-bit registers)
- Improved interrupt processing
- New Multi-Drop Bus (`MDB`) Mode
- New LIN Break / Inter-frame gap support

This EE-Note briefly covers the above topics. The document assumes that the reader is familiar with the ADSP-BF54x UART module. For a complete description of the ADSP-BF60x UART module, see the *ADSP-BF60x Blackfin Processor Hardware Reference*[1]. Furthermore, for details on the changes from the ADSP-BF53x UART module to the ADSP-BF54x UART module, refer to *UART Enhancements on ADSP-BF54x Blackfin Processors (EE-331)*[2].

Feature	ADSP-BF60x	ADSP-BF54x	ADSP-BF53x
MDB / ICP v2.0	Y	N	N
SIR IrDA	Y	Y	Y
Automatic Flow Control	Y	Y	N
False Start Bit Detection	Y	Y	N
32 Bit MMR's	Y	N	N
Data width	5-8 bit	5-8 bit	5-8 bit
LIN Break / Inter-frame gap	Y	N	N
Receive FIFO depth (words)	8	4	0
Improved bit granularity	Y	Y	N
Stop Bits	1; 1.5; 2; 2.5	1; 1.5 (5-bit); 2 (non-5-bit)	1; 1.5 (5-bit); 2 (non-5-bit)

Table 1. Feature summary

Programming Model

The UART on ADSP-BF53x processors was closely aligned to the industry-standard, 16450-compatible programming model. Besides the obvious advantage of code compatibility, this also had a down-side: the aged programming model did not adequately support the pipelined, throughput-optimized Blackfin architecture.

The UART module already available on the ADSP-BF50x and ADSP-BF54x processors removed disadvantages of this aged standard. Especially, the sharing of MMR addresses has been removed. However, this module does not exactly match the known programming model from other Blackfin processor peripherals.

Consolidated Bit Grouping

Due to 16450 history, the first three generations of Blackfin UART modules are known for a series of 8-bit control and status registers. This legacy support results in suboptimal programming model. Many operations require multiple MMR accesses, each only utilizing eight bits on the 32-bit wide access bus. Therefore, the grouping of control and status bits has been consolidated. Now most operations can be performed with a single MMR access.

The former 8-bit registers `UART_LCR`, `UART_MCR`, and `UART_GCTL` have been replaced by a single 32-bit wide control register, simply called `UART_CTL`. The status bits formerly found in `UART_LSR`, `UART_MSR`, and `UART_IIR` (ADSP-BF53x processors only) have been combined into the single status register `UART_STAT` register, which is 32-bit wide.

Actually, all registers of the ADSP-BF60x UART module have been implemented as 32-bit registers to be consistent across the UART interface. But data registers like `UART_THR`, `UART_RBR`, `UART_SCR`, and `UART_TAIP` utilize only the lower 8-bit with the other bits being reserved.

The low byte and the high byte from the Clock Divisor Latch Registers `UART_DLL` and `UART_DLH` are now found in the UART clock rate (`UART_CLK`) register. They are now combined to the divisor field (`UART_CLK.DIV`). This applies also to the `EDBO` bit, which previously was present in the `UART_GCTL` register.

ADSP-BF60x Bits UART_	**ADSP-BF54x Bits UART_**	**ADSP-BF53x Bits UART_**
CTL.EN	GCTL.UCEN	GCTL.UCEN
CTL.LOOP_EN	MCR.LOOP_ENA	MCR.LOOP_ENA
CTL.UMOD[]	GCTL.IREN	GCTL.IREN
CTL.WLS[]	LCR_WLS[]	LCR_WLS[]
CTL.STB	LCR.STB	LCR.STB
CTL.STBH	---	---
CTL.PEN	LCR.PEN	LCR.PEN
CTL.EPS	LCR.EPS	LCR.EPS
CTL.STP	LCR.STP	LCR.STP
CTL.FPE	GCTL.FPE	GCTL.FPE
CTL.FFE	GCTL.FFE	GCTL.FFE
CTL.SB	LCR.SB	LCR.SB
CTL.FCPOL	MCR.FCPOL	---
CTL.RPOLC	GCTL.RPOLC	GCTL.RPOLC
CTL.TPOLC	GCTL.TPOLC	GCTL.TPOLC
CTL.MRTS	MCR.MRTS	---
CTL.XOFF	MCR.XOFF	---
CTL.ARTS	MCR.ARTS	---
CTL.ACTS	MCR.ACTS	---
CTL.RFIT	MCR.RFIT	---
CTL.RFRT	MCR.RFRT	---
---	---	LCR.DLAB
CLOCK.EDBO	GCTL.EDBO	---

Table 2. UART_CTL vs. UART_LCR, UART_MCR, and UART_GCTL

ADSP-BF60x Bits UART_	ADSP-BF54x Bits UART_	ADSP-BF53x Bits UART_
STAT.DR	LSR.DR	LSR.DR
STAT.OE	LSR.OE	LSR.OE
STAT.PE	LSR.PE	LSR.PE
STAT.FE	LSR.FE	LSR.FE
STAT.BI	LSR.BI	LSR.BI
STAT.THRE	LSR.THRE	LSR.THRE
STAT.TEMT	LSR.TEMT	LSR.TEMT
STAT.TFI	LSR.TFI	---
STAT.ASTKY	---	---
STAT.ADDR	---	---
STAT.RO	---	---
STAT.SCTS	MSR.SCTS	---
STAT.CTS	MSR.CTS	---
STAT.RFCS	MSR.RFCS	---
---	---	IIR.NINT
---	---	IIR.STATUS[01]
---	---	IIR.STATUS[10]
---	---	IIR.STATUS[11]

Table 3. UART_STAT vs. UART_LSR and UART_MSR

ADSP-BF60x Bits UART_	ADSP-BF54x Bits UART_	ADSP-BF53x Bits UART_
`IMSK.ERBFI`	`IER.ERBFI`	`IER/IIR.ERBFI`
`IMSK.ETBEI`	`IER.ETBEI`	`IER/IIR.ETBEI`
`IMSK.ELSI`	`IER.ELSI`	`IER/IIR.ELSI`
`IMSK.EDSSI`	`IER.EDSSI`	`---`
`IMSK.EDTPTI`	`IER.EDTPTI`	`---`
`IMSK.ETFI`	`IER.ETFI`	`---`
`IMSK.ERFCI`	`IER.ERFCI`	`---`
`IMSK.EAWI`	`---`	`---`
`IMSK.ERXS`	`IER.EGLSI`	`---`
`IMSK.ETXS`	`IER.EGLSI`	`---`

Table 4. UART_IMSK vs. UART_IER and UART_IER/IIR

Atomic Register Operations

If transmit, receive and status events are handled by different interrupt service routines, atomic register accesses are preferred. Read-modify-write operation, however, are candidates of failure if interrupted by other service routines accessing the same MMR register.

Typically, UART mode as in `UART_CTL` are global and permanent settings and are not subject of ever being altered at run time. The interrupt enable bits however do need to be toggled at run time for efficient interrupt processing. As the ADSP_BF54x UART module, also the ADSP-BF60x UART module features a pair of Set and Clear ports to the interrupt enable register. Interrupt service routines can toggle them by atomic write-1-to-set (`W1S`) or write-1-to-clear (`W1C`) operations, avoiding any need for explicit temporary disabling of global interrupts. They are now renamed to `UART_IMSK_SET` and `UART_IMSK_CLR` for consistency with other peripherals.

Unlike on the ADSP-BF54x UART module, on ADSP-BF60x processors the interrupt enable register also features a regular read/write (`RW`) port, called the `UART_IMSK` register. This is similar to the ADSP-BF53x generation, but is now enhanced with several new interrupt features.

Interrupt Enhancements

Minor changes compared to [2].

The new UART module basically features the same status and data interrupts like the UART module on the ADSP-BF50x and ADSP-BF54x processors. Besides the regular transmit and receive interrupt request channels, each UART also features a status interrupt output. On ADSP-BF53x processors, the status interrupt was used only to signal UART line errors like overrun, framing, parity, and line break. On ADSP-BF54x processors, it is used for the following purposes:

- Line errors
- Alternate RX and TX signaling (enhanced)
- Transmit finished
- Modem status
- FIFO threshold
- CTS signal change

On ADSP-BF60x processors, the following new (additional) purposes are available:

- Address word in MDB-Mode present in the receive data buffer (`UART_RBR`)

One more status bit has been added, that does not trigger an interrupt:

- Reception On-Going

Moreover, status interrupts can trigger system fault events. See [1] and the System Event Controller chapter for more details on the System Fault Interface.

Alternate RX and TX Signalling

Transmit and receive interrupts are routed through the DMA controller. Unlike legacy Blackfin products, DMA channels are now dedicated to a specific peripheral. DMA channel assignment is now obsolete.

Similar to the last generation UART module, there are three separate interrupt paths for data transmission, data reception, and status. In the BF54x UART module, the `EGLSI` bit in the `UART_GCTL` register allows routing of the TX and RX interrupts to normal interrupt outputs; but only both at the same time.

The ADSP-BF60x UART module can do this for TX and RX individually through the `UART_IMSK.ETXS` and `UARTx_IMSK.ERXS` bits, respectively.

Transmit Finish Interrupts

Compared to the earlier UART module, there are no changes of interrupt signaling on the transmit path.

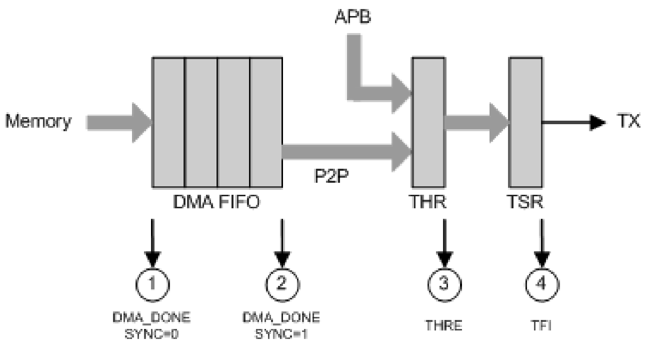

Figure 1. Transmit interrupt options

Figure 1 illustrates the resulting four interrupt options for UART transmit operation.

1. Regular TX DMA interrupt. By this time a new work unit can be initiated, but the DMA must not be disabled yet as data is still in the DMA FIFO.
2. TX DMA interrupt with the `SYNC` bit set. The DMA interrupt is delayed until all data has left the DMA FIFO. By this time, it is safe to disable and to re-assign the DMA channel.
3. Regular TX UART interrupt. In DMA mode, this request functions as DMA request and cannot generate interrupts. In non-DMA mode, the DMA channels forward this transmit request to the SIC controller. With the `EGLSI` bit set, this request goes to the status interrupt channel instead.
4. Transmission finished interrupt. At this time, all data has left the UART and it is safe to disable the UART module. If `ETFI` is enabled, a status interrupt is generated. If DMA is enabled and `EDTPTI` is set, the DMA controller requests an interrupt over the TX DMA channel. The user may choose to service the TFI event by the transmit or the status interrupt routine.

Receive FIFO

The ADSP-BF53x UART module features a two-stage receive buffer that consists of the receive buffer (UART_RBR) register and the serial sampling (UART_RSR) register. The latter is not accessible by software.

If enabled, a receive interrupt is issued when a data word has been copied from RSR to UART_RBR. This is performed at the time the first stop bit has been sampled. If software (or DMA) does not read the UART_RBR register before the receipt of stop bit of the second data word, the second word will overwrite the first word in UART_RBR and an overrun condition is reported. This means that software must respond within one data frame to avoid overrun errors.

On ADSP-BF54x processors, an additional 4-deep FIFO has been inserted between the RSR and UART_RBR registers. This is shown in Figure 2. Compared to the ADSP-BF53x implementation, the interrupt response time requirement has been relaxed by a factor of five. This feature does not require any software changes.

On the ADSP-BF60x processors, the FIFO has been increased to a depth of 8 stages.

The RFCS status bit, indicating the receive FIFO's count status, has been moved from the UART_MSR register to the UART_STAT register. The behavior of the bit depends on the receive FIFO threshold (RFIT) bit in the UART_CTL register formally: UART_MCR register).

If RFIT=0, the RFCS bit indicates whether four or more data words are available in the receive FIFO. If RFIT=1, the RFCS bit indicates whether seven or more data words are available in the receive FIFO. The RFCS bit clears automatically when the FIFO has drained below the watermark due to core or DMA reads from the UART_RBR register.

The enable receive FIFO count interrupt (ERFCI) bit enables interrupt signaling to the UART Status Interrupt channel based on the RFCS bit. In non-DMA mode, the interrupt service routine may rely on the status interrupt instead of the regular receive interrupt and reduce the interrupt load to one-half or one-fourth. Four (RFIT=0) or seven (RFIT=1) bytes can be read from the receive buffer. When the RFCS bit clears, the interrupt request is also de-asserted.

The new UART module offers one completely new feature monitoring ongoing reception. The reception on-going bit (RO) in the UART_STAT register remains high from the detection of a start transition until the sampling of the 1st stop bit.

Figure 2 illustrates the resulting five interrupt options for UART receive operation.

1. Regular RX DMA interrupt. By this time, all receive operation has been completed. Unless further data is in the receive pipe, it is safe to disable the DMA channel and/or the UART controller.
2. Normal interrupt in non-DMA mode when the ERBFI bit is set. All the receive status flags (DR, PE, FE, and BI) consistently report at this point of time. Consequently, the ELSI interrupt is also partly aligned here.
3. FIFO threshold interrupt option 1. When RFIT=0, the ERFCI interrupt signals that four bytes are ready in the UART receive buffer.
4. FIFO threshold interrupt option 2. When RFIT=1, the ERFCI interrupt signals that eight bytes are ready in the UART receive buffer.
5. Overrun error. By the time the stop bit of the 9th word is received before the receive buffer has drained; an overrun error occurs and is reported to the ELSI interrupt.

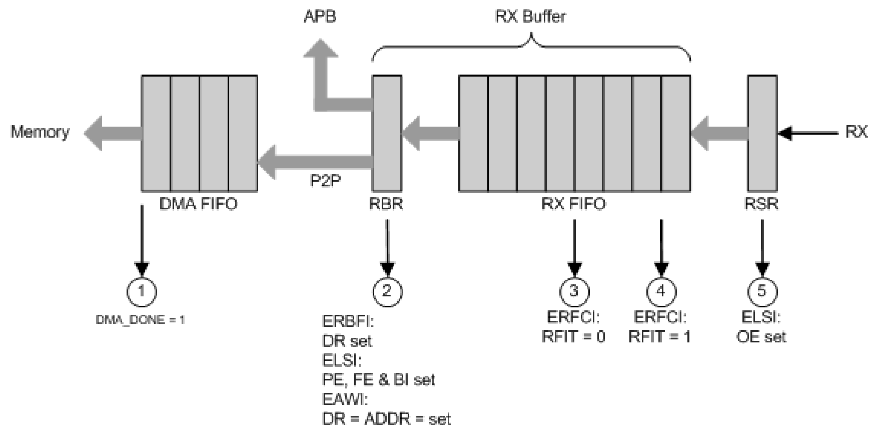

Figure 2. Receive interrupt options

UART Multi-Drop Bus (MDB) Mode

Beside the traditional UART mode (`UARTx_CTL.MOD[5:4]`=0) and IrDA SIR mode (`UARTx_CTL.MOD[5:4]`=2), the new UART module features the Multi-Drop Bus (`MDB`) mode (`UARTx_CTL.MOD[5:4]`=1).

For transmitting an address word, the Transmit Address/Insert Pulse register (`UARTx_TAIP`) is used instead of the `UART_THR`. They are physically the same registers but with different effects.

Writes to the `UARTx_TAIP` register enhance the UART frame by a so-called address bit. The address bit is inserted between the data bits and the optional parity bit.

On the receiver side, the `UARTx_STAT.ADDR` bit indicates if the received data is either a data word or an address word. It is updated by hardware every time a new frame has been received. With the `UART_IMSK.EAWI` interrupt enable bit set, reception of an address byte triggers a special status interrupt.

The `UART_STAT.ASTKY` is the sticky version of the `UART_STAT.ADDR`. It is set by hardware whenever `UART_STAT.ADDR` is set. `UART_STAT.ASTKY` can only be cleared by software with a `W1C` operation.

Remark, that DMA only writes into `UARTx_THR`.

UART LIN Break Command

For supporting the Local Interconnect Network (`LIN`) protocol with smaller software overhead, the UART Mode Break Command and Inter-Frame Gap Transmit Operation has been introduced. Through the `UARTx_TAIP` register, active low (`UARTx_TAIP[7]`=0) or active high pulses (`UARTx_TAIP[7]`=1) of length `UARTx_TAIP[6:0]` / bit rate can be generated.

The generated inter-frame pulse is always terminated by regular stop bits as defined by the `UART_CTL.STB` and `UART_CTL.STH` bits.

The UART receiver can detect break commands through the Break Indicator (`UART_STAT.BI`) flag. This flag reports that an entire UART frame has been received in low state. It does however not report whether the duration of the received low pulse was exact or at least 13 bit times as LIN masters transmit. Typically, the Break Indicator suffices LIN requirements. If however the pulse with needs to be determined more precisely, functionality of the GP Timers can be used.

Hardware Flow Control

Minor changes compared to [2].

Hardware flow control is a common handshake that enables a UART receiver to prevent the counterpart's transmitter from sending further data when the receive buffer may overflow.

The receiver generates an output signal, called $\overline{\text{RTS}}$ (request to send). It de-asserts the signal when the receive buffer is filled above a certain threshold. The transmitter senses to an input signal called $\overline{\text{CTS}}$ (clear to send). It stops transmission when $\overline{\text{CTS}}$ is de-asserted.

However, it still cleanly finishes transmission of the currently processed word.

For bi-directional data flow, the hardware connection would look like Figure 3.

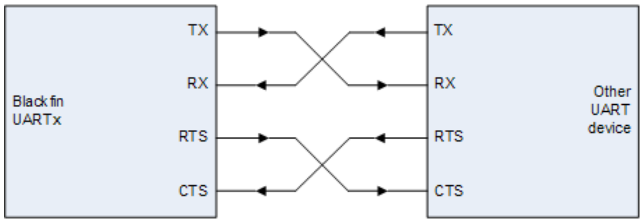

Figure 3. Hardware flow control connection

Typically, $\overline{\text{RTS}}$ and $\overline{\text{CTS}}$ signals are active-low signals (send when low). Setting the FCPOL bit in the UART_MCR register inverts the signal polarity, and $\overline{\text{RTS}}$ and $\overline{\text{CTS}}$ become active-high signals (send when high).

RTS Generation of the Receiver

The ARTS bit in the UART_MCR register enables automatic generation of the $\overline{\text{RTS}}$ signal. If ARTS=0, the state of the $\overline{\text{RTS}}$ output signal is controlled by the MRTS bit manually. If, however, ARTS=1, the $\overline{\text{RTS}}$ output tells about the status of the receive buffer.

Similar to the RFIT bit, which controls the interrupt timing, the receive FIFO $\overline{\text{RTS}}$ threshold (RFRT) bit controls the assertion of the $\overline{\text{RTS}}$ signal. Unlike the interrupt, $\overline{\text{RTS}}$ assertion and de-assertion is subject to hysteresis.

$\overline{\text{RTS}}$ de-asserts if there are two (RFRT=0) or four (RFTR=1) bytes in the receive buffer and a third or fifth start bit is detected. $\overline{\text{RTS}}$ asserts again, when the receive buffer drains below the watermark of two (RFRT=0) or four (RFRT=1) due to UART_RBR reads again.

Since the $\overline{\text{RTS}}$ signals must also be enabled at port muxing level, it might be advantageous to have a pull-down resistor (pulling up if FCPOL =0) on this signal to prevent it from floating during the processor's reset cycling.

Also, a pull-up resistor on the TX output avoids floating signals while the processor is in reset.

CTS Sensing on the Transmitter

There is a new bit in the UART_MCR register called ACTS. If set, it enables automatic $\overline{\text{CTS}}$ sensing. Once the transmitter detects de-asserted level on the $\overline{\text{CTS}}$ input pin, it finishes transmission of the word currently in the TSR shift register. However, it prevents the word currently in the UART_THR register from being transferred to the TSR register until $\overline{\text{CTS}}$ is asserted again.

In addition, there is a new bit called CTS in the UART_MSR register that indicates assertion of the $\overline{\text{CTS}}$ input signal. There is also a sticky version of the CTS bit called SCTS, that can be cleared by W1C software operation.

If enabled by the EDDSI bit, a so-called modem status interrupt is signaled to the UART status interrupt, whenever SCTS=1.

Similar to the $\overline{\text{CTS}}$ input pin, software can also pause transmission. This is done by setting the XOFF bit in the UART_MCR register temporarily.

In loopback mode (LOOP_EN=1), the receiver's $\overline{\text{RTS}}$ output is internally looped back to the transmitter's $\overline{\text{CTS}}$ input.

Bit Rate Generation

The UART module's base clock is derived from the system clock (`S0CLK`) by dividing it by the 16-bit integer as represented by the `UART_CLK` 32-bit register.

For safety, the received UART data is oversampled 16 times. Thus, the effective bit rate is 16 times slower than the base clock.

If the UART is running at high bit rates, the granularity of the supported bit rates becomes very limited. For example, assume an `S0CLK` frequency of 120 MHz and a nominal bit rate of 1 Mbit per second. Then, a divisor value of 7 would result in a bit rate of 1071 Mbit per second, which is off by 7%. A divisor value of 8 would result in 937500 bits per second, which is not much better.

For this reason, the ADSP-BF60x UART features the enable-divide-by-one (`EDBO`) bit in the global control register. This was introduced with the ADSP-BF54x and ADSP-BF50x processors. With this bit set, the divide-by-16 circuit is bypassed. Consequently, the effective bit rate is:

$$\text{bitrate} = \frac{\text{S0CLK}}{16^{1-\text{EDBO}} \cdot \text{UART\_CLK.DIV}}$$

With the `EDBO` bit set, the receiver still oversamples and makes one-out-of-three-samples majority decisions. However, the sample points in the middle of a bit are no longer generated by traditional synchronous logic.

In the above example, a divisor value of 120 would nicely generate the desired bit rate, when the `EDBO` bit is set.

Autobaud Detection

The principle of autobaud detection has not changed compared to the ADSP-BF53x or ADSP-BF54x UART modules.

On ADSP-BF60x processors, each of the two UARTs has a general purpose (GP) timer associated. Its alternate capture input (`TM_AUX_IN`) senses the UART receive input.

- `TACI0` senses `UART0 RX`
- `TACI1` senses `UART1 RX`

If the recommended autobaud pattern `0x40` is used and the timer is configured to capture periods between two falling edges, the `TIMER_TMRn_PER` register will contain 8 bit times.

The UART clock divisor value is still calculated by shifting the resulting value in the `TIMER_TMRn_PER` register seven bit positions to the right. If, however, the `EDBO` bit is set, the `TIMER_TMRn_PER` value must only be shifted by three bit positions.

New System Features

DMA configuration

The new designed Direct Memory Access (DMA) engine features two new bit fields in the control register, named `PSIZE` (Peripheral Transfer Word Size) and `MSIZE` (Memory Transfer Word Size). For the UART, `PSIZE` will always be set to 1 (bus width of the UART to DMA connection is 8 bit).

A proper use of the `MSIZE` value helps to reduce the amount of memory accesses performed by the DMA. The higher the `MSIZE`, the lower the time a DMA utilizes the System Crossbar (SCB). The configuration requires the knowledge about memory address alignment and number of bytes to be transferred. A setting of `MSIZE` = 5 requires an alignment of 32 bytes and a count that is a multiple of 32 bytes. E.g. a transfer of n times 32 bytes would result in n number of memory accesses only.

Timeout Detection

Beside the Autobaud Detection functionality, the General Purpose Timers offer a new feature called Windowed Watchdog Mode. This can help to detect timeouts on the UART interface as well as to determine the correct pulse width when using the UART interface in a Local Interconnect Network (LIN).

Trigger Routing Unit

The Trigger Routing Unit (TRU) can provide system-level sequence control without core intervention. DMA transfers can be synchronized (e.g. transmit after receive completed) or periodically triggered by a timer.

Code Porting Checklist

Porting code from ADSP-BF50x / ADSP-BF54x projects or even ADSP-BF53x processors requires multiple changes. It's not just a couple of code lines that must be updated; especially control and status registers need to be changed extensively.

Required Changes

- See [2] for ADSP-BF53x code.
- Beside the new IDDE, header files are auto-generated to meet a standardized code style. All definitions need to be changed.
- All control registers are now combined into a single register.
- All status registers are now combined into a single register.
- All divisor registers are now combined into a single register.
- Refer to the DMA chapter in [1] for the new Direct Memory Access engine.
- Refer to the SEC chapter in [1] for the new System Event Controller.

Recommended Changes

- See [2] for ADSP-BF53x code.

References

[1] *ADSP-BF60x Blackfin Processor Hardware Reference.* Rev 0.41, June 2012. Analog Devices, Inc.

[2] *UART Enhancements on ADSP-BF54x Blackfin Processors (EE-331).* Rev 1, Nov. 2007. Analog Devices, Inc.

[3] *Multi-Drop Bus 4.2 Specifications, Feb. 2011.* Vending.org (http://www.vending.org/technology)

[4] *LIN specification 2.2.* Rev A. LIN Administration (http://www.lin-subbus.org)

Document History

Revision	**Description**
Rev 1 – August 1, 2012 by A. Pellkofer	Initial Release