Models LR-F28335DAQA & LR-28335DAQA8x8

Technical Reference Manual

June 25, 2013

Document Number: 4-903-68

TABLE OF CONTENTS

I. Introduction

The Model LR-F28335DAQA is a high precision data acquisition daughtercard designed to work with the Spectrum Digital, Inc. F28335 eZdsp™ development system. The daughtercard contains the following features:

Hardware

- 4 A/D input channels, single ended, ±10V range, simultaneous sampling
- 8 D/A output channels, single ended, ±10V range, simultaneous output update
- An RS232 asynchronous serial communications interface
- A software controllable output signal with a +15V to ground swing
- Eight general purpose digital logic inputs
- Two general use LEDs
- Optional support for up to 8 A/D channels
- Optional support for a fiber optic asynchronous communications interface
- Optional support for a 10/100 802.3 Ethernet interface
- Support for up to 16 A/D and 16 D/A channels by using the optional model LR-28335DAQA8x8 daughtercard.

Software

Three sample Code Composer Studio projects are included with the daughtercard. The first demonstrates a 3 phase, 60 Hz sinewave generator. This project also includes the necessary high-level functions needed to configure and access the serial port as well as all of the other hardware present on the board. The second CCS project is identical to the first, except that it is built using the TI DSP/BIOS real time kernel. This project runs out of internal RAM. The third project is similar to the second, except that it has been configured to run out of the onboard FLASH.

II. Hardware Description

1. Preparing the F28335 eZdsp board for use with the model LR-F28335DAQA

The daughtercard requires several voltages for operation. The following chart lists these voltages and shows their source:

Voltage	Source
+5V	eZdsp™ board, P2-1 and P2-2
+3.3V	eZdsp™ board, P8-1
±15V	On-board switching power inverter

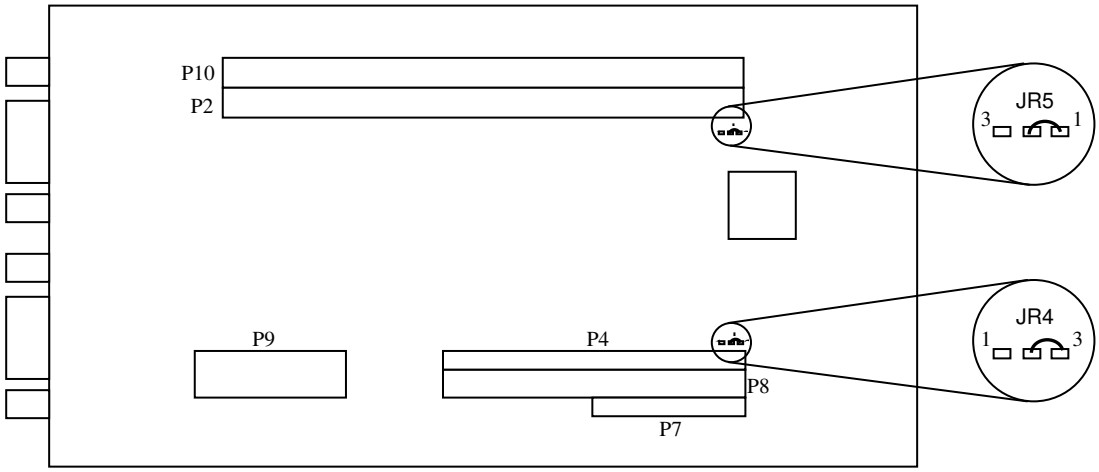


Figure 1

Back side of the F28335 eZdsp board

To bring power (+5V and +3.3V) to the daughtercard, two small jumper wires must be installed on the **back side** of the eZdsp board as shown in Figure 1. The JR5 jumper connects the +5V signal to the P2 connector and the JR4 jumper connects the +3.3V signal to the P4 and P8 connectors.

Additionally, the two dip switches (SW1 and SW2) on the eZdsp board must be configured properly in order for the daughtercard to function. Figure 2 shows the proper dip switch settings for debugging a program from RAM, and Figure 3 shows the switch settings for operating from the DSP's on-board FLASH.

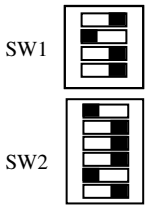


Figure 2

Figure 3

2. External Interrupt

One external interrupt, XINT1, is used for daughterboard operation. It is configured to trigger on a high-to-low transition. The BUSY signal from the A/D chip is tied to this interrupt, and its high-to low transition indicates that the A/D has completed conversion on all 4 channels.

3. A/D Circuit

The A/D IC used is the Analog Devices AD7865BSZ-1. This IC has 4 single-ended input channels, each with a range of ±10V. With the falling edge of CONVST, all 4 channels are sampled simultaneously (with 4 internal sample and hold circuits), and conversion begins. When conversion on all 4 channels is complete, the BUSY signal is lowered which triggers an interrupt as described in the above section.

The A/D IC is memory mapped to the DSP's zone 6 address space. This allows reading the A/D conversion results by making 4 consecutive read operations from memory address 0x00108000.

As an option, the daughtercard can be purchased with an additional 4 A/D input channels, allowing a total of 8 simultaneously sampled channels. The second A/D IC is accessed at memory address 0x00108004.

The A/D input connector is J1, located at the top left side of the board. The pinout of J1 is as follows:

15	13	11	9	7	5	3	1
A/D #8	A/D #7	A/D #6	A/D #5	A/D #4	A/D #3	A/D #2	A/D #1
GND	GND	GND	GND	GND	GND	GND	GND
16	14	12	10	8	6	4	2

4. D/A Circuit

The D/A IC used is the Analog Devices AD7841BSZ. This IC has 8 single-ended output channels with a range of ±10V. When power is applied to the data acquisition board, the D/A device is momentarily held in a cleared state (all outputs at 0.0 volts). This allows time for the firmware to write initial values to the D/A registers.

Data can be written to the device at any time, however, the analog outputs are not updated until the LDAC# signal is brought low. This feature allows all 8 channels to be updated simultaneously. This D/A IC does not have an internal reference, therefore an external bipolar 5V reference was included. This reference consists a REF02

device, a precision +5V reference. The negative 5V reference is created with an OPA177, a precision op-amp configured as an inverting circuit with a gain of –1. The two precision 10.0K resistors forming the divider used to achieve the –1 gain have a 0.1% tolerance.

The D/A IC is memory mapped to 8 independent addresses in the DSP's zone 6 address space. This results in each of the 8 channels having its own memory mapped address. The address mapping is as follows:

D/A Channel	Memory Address
1	0x108008
2	0x108028
3	0x108048
4	0x108068
5	0x108088
6	0x1080A8
7	0x1080C8
8	0x1080E8

The D/A output connector is J2, located at the lower center of the board. The pinout of J2 is as follows:

2	4	6	8	10	12	14	16
GND	GND	GND	GND	GND	GND	GND	GND
D/A #1	D/A #2	D/A #3	D/A #4	D/A #5	D/A #6	D/A #7	D/A #8
1	3	5	7	9	11	13	15

5. Signal Protection

All A/D inputs and D/A outputs are protected from unwanted and unintentional transient voltages by the use of bipolar transient voltage suppressors (TVSs). These devices essentially clamp the input and output signals at the ±15 volt power supply rails.

6. Software controllable output

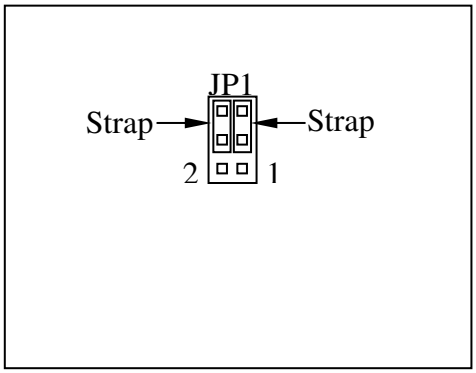
The Software controllable output circuit consists of a 2N3904 transistor and associated resistors. The output signal swings from 15V to ground and appears at J2-17 and J2-19 (J2-18 and J2-20 are GND). The signal is controlled directly by the DSP using a GPIO output line. This signal is useful for controlling external power circuits.

7. Serial Communications

The Model LR-F28335DAQA contains a serial port for general communications use. This port can be configured as a standard RS232 port, or, optionally, as a fiber optic serial port. Both configurations share the DSP's SCI-A port, and as such, cannot be used simultaneously.

RS-232 Serial Port

The RS232 configuration is selected when the dual jumper JP1 has straps in positions 3-5 and 4-6. Figure 1 shows the jumper settings for RS232 operation.

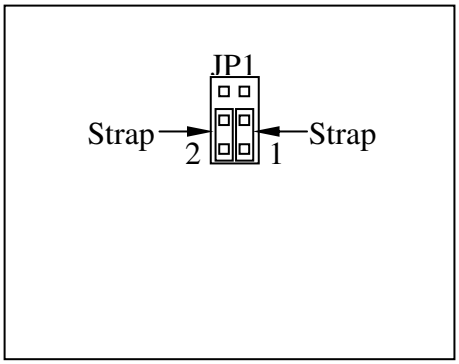


Strap settings for RS232 operation
Figure 1

The RS232 serial port is configured as a DCE device, which allows the daughtercard to connect to a PC using a standard, "straight-thru" cable. The serial port is capable of operating at speeds of up to 921.6 kbps.

Fiber Optic Serial Port (optional)

The fiber configuration is selected when the dual jumper JP1 has straps in positions 1-3 and 2-4. Figure 2 shows the jumper settings for fiber operation.



Strap settings for fiber operation
Figure 2

The fiber optic devices used on the daughtercard accept standard 1mm, multimode, coated plastic fiber. Both the LED transmitting device and the photo detector device have integral plastic housings, which accept the fiber and have a screw locking mechanism. The fiber optic serial port is capable of operating up to 921.6 kbps.

8. Switch Inputs

Provision is made for 8 general-purpose input pins***. These 8 inputs are connected directly to DSP GPIO pins, and as such, can only accept signals in the range of 0V to +3.3 V That is, these inputs ARE NOT 5V tolerant***. The 8 lines are pulled up with 10K resistors, and are intended to be connected to passive switches. In this situation, one side of a switch would be grounded, and the other side connected to one of the 8 lines. The 8 inputs appear at connector J4.

The pin-out for connector J4 is as follows:

15	13	11	9	7	5	3	1
S8	S7	S6	S5	S4	S3	S2	S1
GND	GND	GND	GND	GND	GND	GND	GND
16	14	12	10	8	6	4	2

9. LEDs

Two LEDs are provided on the daughtercard. These LEDs are of the low current variety (2mA). The LEDs are controlled with two GPIO DSP pins. See the software

section below for a description of the high level functions available to control the LEDs.

10. Ethernet Interface (optional)

As an option, an Ethernet interface can be included on the daughtercard. Access to the Ethernet LAN is provided by a Lantronix, Inc. Xport™ Device Server. The Xport™ Device Server interfaces to the eZdsp™ through SCI-C. Web pages, including HTML and JAVA applets, can be loaded into the Xport™ allowing DSP information to be displayed in a web browser window. No software needs to be installed on the PC end, other than the JAVA virtual machine, which is freely available from the Oracle, Inc. website.

For more technical information on the Ethernet interface option, please refer to document number 4-903-40, available from the Link Research website.

11. Using the model LR-28335DAQA8x8 daughtercard (optional)

If 8 A/D and 8 D/A channels are insufficient for a particular application, it is possible to achieve a total of 16 A/D and 16 D/A channels by adding the Link Research model LR-28335DAQA8x8 daughtercard. This daughtercard contains 8 additional A/D, and 8 additional D/A channels. None of the other features (such as an RS-232 port, LEDs, etc.) are included with this daughtercard.

The LR-28335DAQA8x8 daughtercard plugs directly into the LR-F28335DAQA daughtercard, resulting in a 3 board stackup when used with the F28335 eZdsp board. The specifications of the A/D and D/A channels on this second daughtercard are identical to those of the primary daughtercard. In addition, all 16 A/D channels can be simultaneously sampled, and likewise, all 16 D/A channels can be simultaneously updated.

III. Software Description

1. General Information

Three Code Composer Studio (CCS) projects are included on the CD. Two utilize DSP/BIOS and the third does not. All three projects perform the same functions, and demonstrate how the included software API can be used to control all aspects of the hardware.

The source files written by Link Research for the non-DSP/BIOS version of the software begin with "LR", for example "`LR_f28335_main.c`". An include file named "`LR_f28335.h`" contains all function prototypes, hardware address defines, constant definitions, and other related code. For the DSP/BIOS version, the source files include the word "bios", for example, "LR_F28335_bios_main.c"

The following sections describe the various portions of the software.

2. Burning the Sinewave into FLASH

In order for the sample projects to function properly, one cycle of a sinewave must be stored in the F28335's FLASH memory starting at address 0x00300000. The sinewave is comprised of 60000 samples, is scaled to 14-bits, and is in two's complement format. To burn this data into the F28335 eZdsp's FLASH, perform the following steps:

1. Open Code Composer Studio (CCS)
2. Click on "Tools -> F28xx on-chip Flash Programmer"
3. Select ONLY flash sections G and H as shown in figure 4
4. Click on "Browse" in the Operation section.
5. Locate the file: "sine60000_f28335.out" in the project directory
6. Click the "Erase, Program,Verify" button, and then click "Execute Operation"
7. When this operation has completed, the sinewave data should reside permanently in FLASH memory.

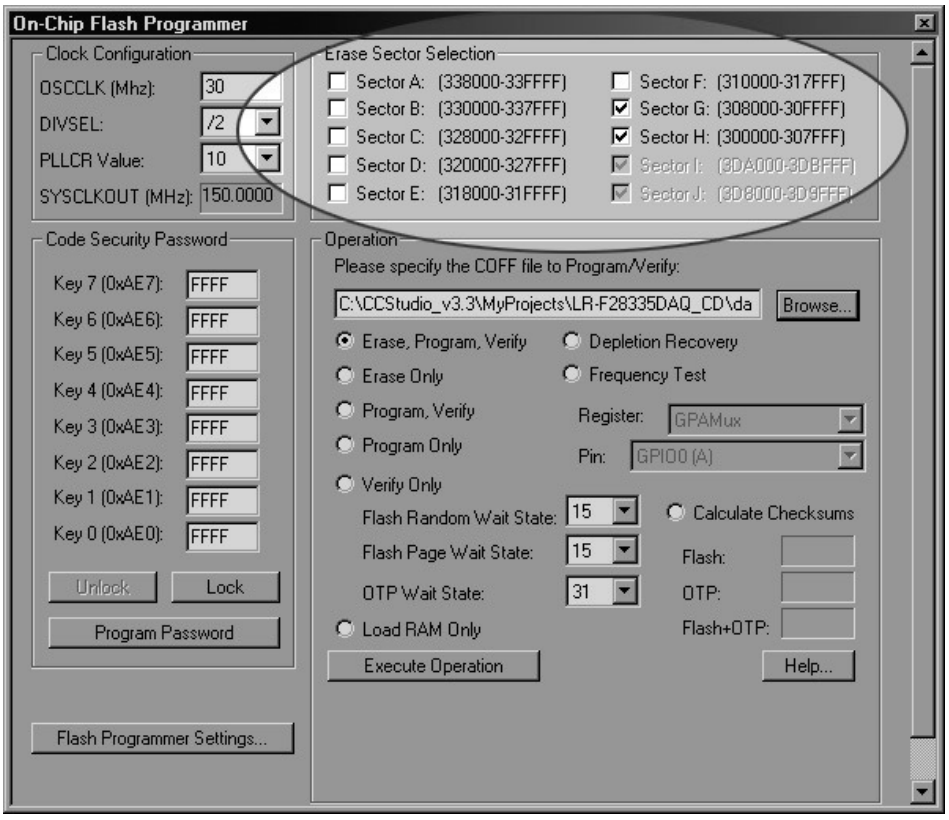


Figure 4

The sinewave frequency is controlled by the variable: `sine_table_step`. Since the sampling rate is set at 40,000Hz and there 60000 samples per cycle, this variable

must be set to 1.5 times the desired output frequency. For example, for a 60 Hz output, set this variable to 90. For 2 Hz, set it to 3, for 400, set it to 600, etc

Memory Map – Device Addressing

The two 4-channel A/D converters and the single 8-channel D/A converter are accessed in a memory mapped manner. Each A/D IC has a single hardware address used to access the 4 channels of converted data. When the A/D signals the processor that it is done converting, the processor can then make 4 consecutive reads of the A/D chip to obtain samples from channels 1, 2, 3 and 4, in that order.

The D/A chip has 8 unique hardware addresses used to access each of its 8 channels. The following table shows the hardware address for the both A/Ds and the D/A:

Device/Channel	Hardware Address
A/D #1 (channels 1-4)	0x0108000
A/D #2 (channels 5-8)	0x0108004
D/A channel 1	0x0108008
D/A channel 2	0x0108028
D/A channel 3	0x0108048
D/A channel 4	0x0108068
D/A channel 5	0x0108088
D/A channel 6	0x01080A8
D/A channel 7	0x01080C8
D/A channel 8	0x01080E8

3. Demo Programs and User Programs using the sinewave

Any program using the sinewave previously burned into sections G and H of the flash must use only sections A thru F for their firmware code. Also, when burning firmware into flash, care must be taken not to erase sections G and H. If this is done, then the sinewave must be re-programmed into sections G and H. A typical user program would be burned into flash using the settings shown in figure 5.

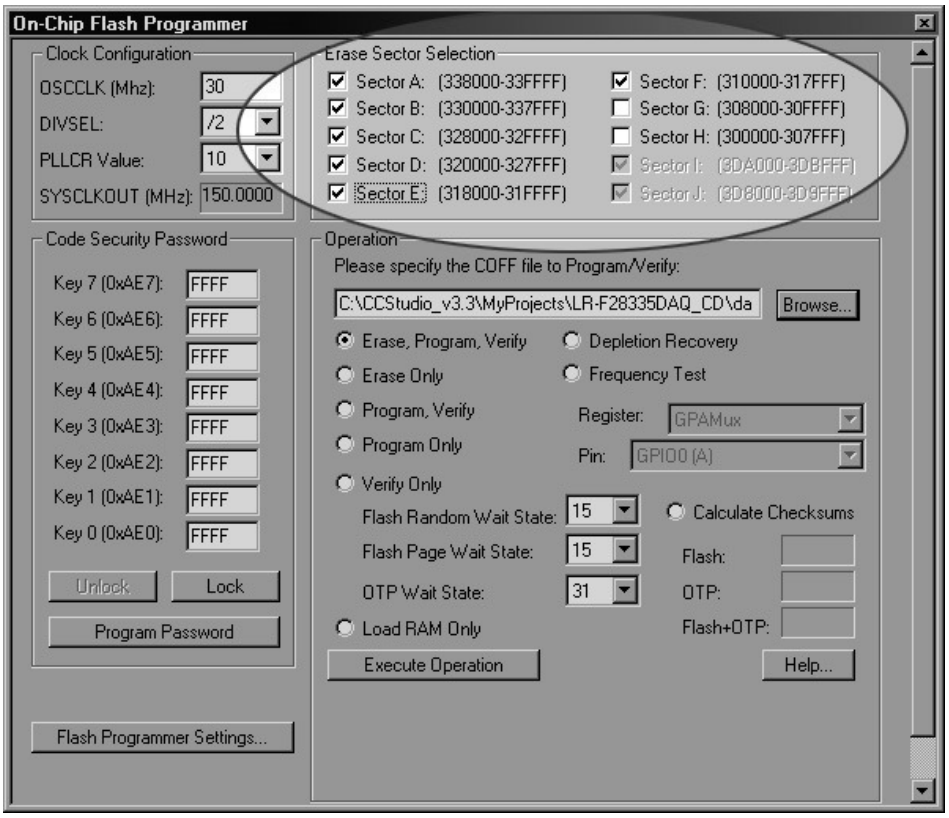


Figure 5

4. A/D and D/A Operation

The AD7865BSZ-1 A/D converter IC requires a periodic "start of conversion" (SOC) signal to produce equally spaced, simultaneously sampled data. The F28335 DSP has a provision where it can provide a start of conversion signal to external A/D converters, and this is what is used. This feature makes use of ePwm3. The ePwm3 output is configured to have a period of exactly 1/40000 seconds and a duty cycle of 50%. This results in a sampling frequency of 40,000 kHz.

On the falling edge of the SOC signal, the A/D chip changes from its sampling mode to its hold mode. At this point all 4 analog input signals have been sampled. Next, the A/D performs a 14-bit conversion on the 4 sampled analog signals. Finally, it lowers its BUSY line to signal the DSP that all 4 conversions are complete and ready to be read. The falling edge of BUSY triggers an interrupt. The interrupt service routine can then read the 4 data values from the A/D chip.

The D/A converter can accept data destined for any of its 8 channels at any time. To achieve simultaneous updates of all outputs, it is necessary to make use of the LDAC# input pin. When this pin is high, any data written to any of the D/As 8 input registers will not affect any of the outputs. Once all data is written to all channels, taking LDAC# low will simultaneously cause all outputs to be updated.

5. A/D and D/A Number Format

All data processed by the DSP should be in Q13 format. This means that the lower 13 bits of a 16-bit word are the significant bits, and the 14th bit is the sign bit. Data read from the A/D within the interrupt service routine (ISR) will be sign-extended before being presented to the user. Data processed by the DSP and destined to be written to the D/A chip, need not be sign-extended, since bits 14 and 15 do not go anywhere. The binary representation of data read from the A/D or written to the D/A is shown in the following table:

Voltage input	Binary value
+10V	0x1FFF
0V	0x0000
-10V	0x2000

It should be noted that the particular D/A used on the daughtercard, the AD7841BSZ, has uni-polar input values. That is, writing 0x0000 produces –10V output, writing 0x2000 produces 0V, and writing 0x3fff produces +10V output. This minor incompatibility is easily handled by software in the ISR.

6. Software Controllable Output Functions

Two high level functions control the state of the Software Controllable Output (SCO)line.

```
void set_SCO_low(void);
```

This function is used to pull the SCO line to ground potential. The function takes no arguments and returns no value.

```
void set_SCO_high(void);
```

This function is used to turn the SCO transistor off, thus allowing the pull-up resistor to pull the SCO signal to the +15V level. The function takes no arguments and returns no value.

For these function to operate, the function **`init_GPIO()`** must have previously been called to insure proper initialization of the GPIO pins.

7. Interrupt Service Routine

There is one interrupt service routine (ISR) for this project. It is triggered by a falling edge transition on external interrupt #1 (XINT1). The interrupt is triggered every 25uS by an external periodic signal as described in section II-2. The event causing the interrupt is the “data ready” signal from the A/D converter. The functions performed in the ISR are the following:

- Read 4 new A/D samples, one for each channel, sign-extend if necessary, and store in global variables ad_value1, ad_value2, ad_value3, and ad_value4.
- Send these 4 data samples to D/A channels 5 thru 8 respectively. (Whatever signal is injected into A/D channel #1, will be seen on D/A channel #5, the signal on A/D channel #2, will be seen on D/A channel #6, etc.)
- Get the next sample for each phase of the 3-phase sinewave from the sine table, and send them to the D/A converter channels 1, 2, and 3. Store the 3 D/A values in 3 global variables: da_value1, da_value2, and da_value3.
- Pulse the D/A converter to update all outputs.
- Advance the 3 sine table pointers for the next time through.
- Return from interrupt.

8. Serial Port Functions

Software to handle the serial port consists of several high level functions. These functions manage 2 buffers, one for transmit data and one for receive data. The length of these buffers is 256 bytes each, but can be easily modified by changing 2 **`#define`** statements in `LR_f28335.h`.

Before the serial port functions can be used, the function **`Initialize_Serial_Port(int data_rate)`** must be called. This function does several things – it enables the SCI function of the DSP, it sets the baud rate, parity setting, number of data bits, and number of stop bits to be used. It also initializes the transmit and receive buffer pointers. As written, the software sets up the UART for 8 data bits, no parity, and 2 stop bits. The data rate is set to the value passed as the parameter to the function and must be one of the pre-defined values listed in `LR_f28335.h`.

The high level functions that can be used to access the serial port are as follows:

```
int  send_char(unsigned char);
```

This function places a single character in the transmit buffer. The one and only argument is the character to be transmitted. The function returns 0 if successful, -1 otherwise.

```
int  send_block(unsigned char*, unsigned int);
```

This function sends a block of characters to the transmit buffer. The first argument is a pointer to the block, and the second argument is the length of the block to send. The function returns 0 if successful, -1 otherwise.

```
unsigned int   get_recv_count(void);
```

This functions returns the number of characters in the receive buffer.

```
int  recv_char(unsigned char*);
```

This function returns one character from the receive buffer. The function returns 0 if successful, -1 otherwise.

```
int  recv_block(unsigned char*, unsigned int);
```

This function returns a block of characters from the receive buffer. The first argument is a pointer to the area in memory that will receive the data, and the second argument is the length of the block to receive. The function returns 0 if successful, -1 otherwise.

```
void service_serial_port(void);
```

This is the function that handles all the low-level operations of the serial port. It polls the UART for available receive data, and transfers this data to the receive buffer. It also polls the UART to see if the transmitter is ready to accept another character. If so, it sends the next character from the transmit buffer. This function should be called periodically, at a rate fast enough to prevent receiver overruns. This rate is dependent on the data rate the UART is programmed for. In the DSP/BIOS version, this function is incorporated into a "TASK", and as such, the user does not have to worry about calling this function periodically.

Note that in their current form, none of these functions should be called from inside the ISR. Should it prove necessary that these functions be called from inside the ISR, they can be easily modified for such operation. Please contact Link Research for assistance.

9. LED Functions

There are two LEDS on the daughtercard, and two functions are provided for each LED. They are:

```
void led1_on(void);
void led1_off(void);
void led2_on(void);
void led2_off(void);
```

These functions should be self-explanatory. Note that the function **`init_GPIO(void)`** must have been previously called to properly initialize the LED GPIO pins. Also, the file `LR_f28335.h` should be included in the source file calling any of these functions.

10. Differences between the DSP/BIOS and the non-DSP/BIOS sample CCS projects

The DSP/BIOS project differs from the non-DSP/BIOS project in the following areas:

- Memory SECTION definitions
- Interrupt vector configuration
- How the Serial Port polling is performed

(a) Memory SECTION definitions

In the non-DSP/BIOS version, all the MEMORY and SECTION specifications are in the linker command file (.CMD). In the DSP/BIOS version, the bios configuration tool is capable of setting up the MEMORY areas. This can be seen by opening the .tcf file and going to the System->MEM section. All the compiler-defined sections and the DSP/BIOS sections can be placed in appropriate memory sections here. However, this project has some user-defined sections that require a separate .cmd file. This can be seen in the file user_bios_RAM.cmd.

(b) Interrupt vector configuration

Whereas in the non-DSP/BIOS version the interrupt vector table is handled by the **DSP28_PieVect.c** source file, in the DSP/BIOS version, the bios configuration tool can handle this task. There is only one interrupt used in this project. It is the "End of Conversion" interrupt which appears on XINT1 (external interrupt #1). XINT1 maps to the PIE (Peripheral Interrupt Expansion) interrupt #1.4. Therefore, to see how XINT1 is configured in the DSP/BIOS configuration tool, go to the "Scheduling -> HWI-> PIE_INTERRUPTS-> PIE_INT1_4" section.

(c) How the Serial Port polling is performed

In the non-DSP/BIOS version, the function "service_serial_port()" had to be periodically called inside the program's main loop. In the DSP/BIOS version, this function has been incorporated into a DSP/BIOS "Task" having priority 1. To see this configuration, go to "Scheduling -> TSK" and examine the "service_serial_port" task.

11. Differences between the FLASH and RAM versions of the DSP/BIOS sample CCS projects

Because the BIOS configuration (.tcf) file is substantially different for the FLASH compared to the RAM version, two separate projects were created. One is designed to operate in RAM, and is the debug version, while the other is designed to be burned into the F28335's FLASH and run stand-alone (i.e. without Code Composer Studio).

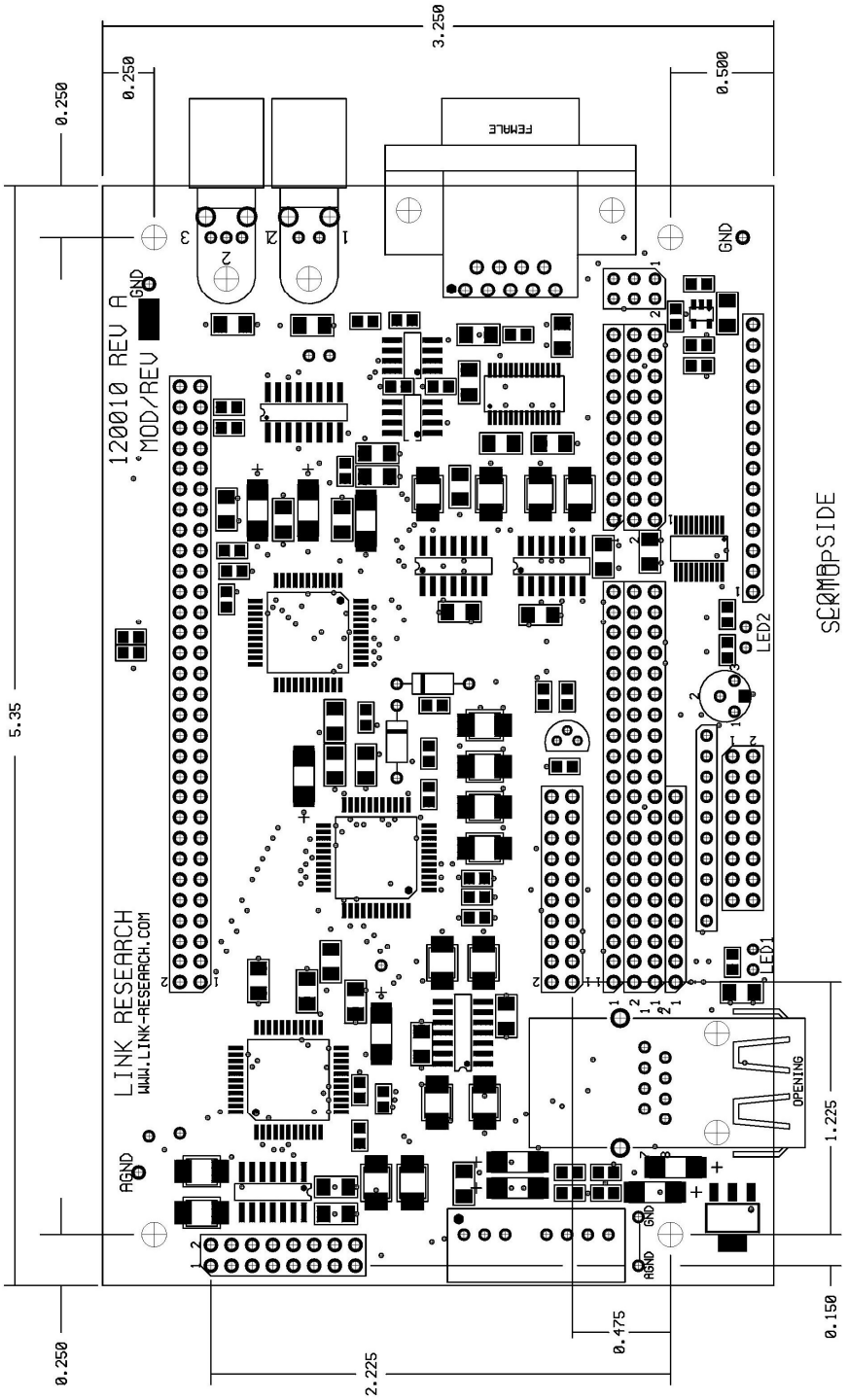
Notice, however, that the source files for the FLASH project are taken directly from the RAM project directory, since no changes had to be made to these files. Doing it this way eliminates errors caused by having two copies of the same file. The files that are different between the FLASH and RAM versions are the TCF file and the user CMD file.

12. What if the CCS projects won't compile/link on my system?

By far, the most common reason why Code Composer Studio projects do not compile on the user's system, is because of different versions of DSP/BIOS. The sample projects included with the model LR-F28335DAQA were built under CCS version 3.3, and DSP/BIOS version 5.32. If the user's system is anything other than this, there is a good chance that build errors will result when a project is built.

If DSP/BIOS version issues arise resulting in build errors on the user's system, the user should create a new TCF file from scratch. This is not difficult. Link Research provides an application note, AN-01, which gives a step by step procedure for creating a TCF file from scratch. This application note is available from the Link Research web site, www.link-research.com.

Daughtercard Dimensional Drawing



Version History

Date	Version	Description
25-June-2013	1.0	Original