DT0044
Design tip

An analog diagnostic output system For the STLUX

By Dennis Nolan

Main components	
STLUX385	Digital controller for lighting and power supply applications

Purpose and benefits

When working with complex digital control systems such as microcontroller based motor control, lighting or power conversion, a real time analog output that represents one or more of the program variables is a very useful diagnostic tool. This design tip describes how to implement two diagnostic analog outputs in an STLUX system. The system needs only one pin from the target system to send data over a standard UART interface. A second microcontroller board receives the data and outputs two analog diagnostic signals that can be observed in real time on an oscilloscope along with other analog signals from the system.

Implementation in the STLUX

Modern scopes, with their digital storage, deep record lengths, extensive triggering options, and ability to analyze and export captured data, are ideal instruments for development and troubleshooting. When working with power electronics, having the ability to look at internal variables or "signals" alongside real world signals such as circuit voltages or currents is extremely useful.

Some of the more sophisticated microcontrollers have several built in digital to analog converters, and plenty of pins to utilize them. The STLUX, however, is a more streamlined, highly optimized and low-pin count device where most of the internal functionalities, such as the DACs, are kept internal." In order to provide the diagnostic capability for the STLUX firmware developer to have up to two analog diagnostic channels available, the system proposed here utilizes the readily available STM32F4DISCOVERY board to connect to the STLUX and provide the two DAC outputs. In order to use this functionality, you only need to add a few lines of code to the STLUX program to "hijack" the standard STLUX UART peripheral. Best of all, with pins being a scarce resource on the STLUX, you only have to claim one pin, the TX pin, to provide the communication link to the discovery board. RX is not used or needed since data flows only one way.

Of course, conventional use of the UART channel for functions such as receiving commands and displaying status with a terminal emulator must be given up, but this can be

done on a temporary basis, with the functionality easily restored in code as needed. Many of the available STLUX based evaluation boards, including the LED lighting drivers and power supply adapters, already include a 3.3V logic level optically isolated UART port. Connecting one of these ports to the discovery board DAC output adapter is very easy. Only three wires are required: 3.3V (to power the output side of the opto), GND (actually, isolated GND), and TX. The discovery board is powered via its USB connector, which can be plugged into a USB power supply. If no optical isolation is available on the STLUX target, the ground of the discovery must be connected directly to the target board ground. In this case, only the addition of the TX connection is required.

WHEN OPERATING WITHOUT OPTICAL ISOLATION, BE VERY CAREFUL THAT THE STLUX BOARD IS OPERATED FROM AN ISOLATED SUPPLY OR AN ISOLATION TRANSFORMER.

All of the STLUX demo boards operate with a standard UART communication setting of 115200N81, meaning 115.2 Kilobaud, no parity, 8 data bits, and 1 stop bit. In order to allow for the possibility of two signals, this default will be modified to use 9 data bits. In most UART communication schemes, the ninth bit is used as the parity check bit. In this case, we will use it to indicate the DAC channel number. The baud rate also places the limitation that DAC updates cannot be sent at intervals shorter than 100 microseconds. At 115.2 Kb, each bit time is 8.68 microseconds. Since each frame has 11 bits (1 start + 9 data + 1 stop), the full frame transmission time is 95.49 microseconds, just clearing the 100 µS window. This works out nicely for the streetlight demo board, whose firmware is based on a 100 µS "heartbeat" interrupt. We can easily add code to send out one byte of data with each run of the interrupt.

Let's look at a specific example. In this case, code has been added to send the calculated on-time and off-time for the PFC stage of the 120W "streetlight" demo board (STEVAL-LL066V1) to the DAC output adapter. The first code block is:

```
#define dacoutput  // comment out to turn DAC output off and return
uart to normal

vu16    NEAR    word0;          // 16 bit scratchpad register

vu8     NEAR    dacchannel;     // 8 bit indicating current DAC
                                // channel
```

This sets up the required ram variables and the enabling #define.

In this example, the following block of code has been placed in the 100µS "heartbeat" interrupt of the program. The high and low registers that make up the 16-bit SMED peripheral time thresholds for the PFC off-time and on-time are loaded into 16 bit working register word0 and then the word0 value is clamped at 255 so as not to exceed the range of the 8-bit UART transmission channel. Without the clamping, the value might have caused a "rollover" of the DAC output which can be confusing. Another alternative would be to "right shift" the value of word0 in order to scale it down appropriately, but this was not found to be necessary for these particular variables as they rarely reach the limit. If you notice that the DAC signal is clipping excessively, the scaling can be applied.

```
#ifdef dacoutput                // conditional compilation so this
functionality                   // can be turned off
if(dacchannel)                  // if DAC channel 1 is selected
{
// PFC off time for DAC channel 1
word0 = (SMED4->TMR_T0H)<<8;        // high byte
word0 = word0 + (SMED4->TMR_T0L);   // low byte
if(word0>255) word0=255;            // clamp at 255
UART->CR1 |= 64 ;                   // set bit8 for channel 1
UART->DR = word0;                   // send data
dacchannel=0;                       // "flip" to channel 0 for next
                                    // interrupt run
}
else  // if DAC channel zero selected
{
// PFC on time for DAC channel 0
word0 = (SMED5->TMR_T3H)<<8;
word0 = word0 + (SMED5->TMR_T3L);
if(word0>255) word0=255;
UART->CR1 &= ~64 ;                  // clear bit8 for channel 0
UART->DR = word0;
dacchannel=255;                     // "flip" to channel 1 for next
                                    // interrupt run
}
#endif
```

The following code block must be placed at the end of the code which configures the UART. In the streetlight program, it was added immediately after the line:

```
Uart_Init();
```

In Main.c

```
#ifdef dacoutput
UART->CR1 |= 16;                // change to 9 data bits
#endif
```

The addition of these three small code blocks is all that is required to add the diagnostic output functionality to an STLUX application. Of course, the code of the main block can be modified so that any two program variables can be monitored.

This scheme sends out two signals that utilize both channels of the STM32F4 discovery board. A compromise required here is that you only get an update of a given signal once every other interrupt, or every 200 µS. Most signals associated with the microcontroller firmware are slow moving enough so that this update rate is sufficient to track what is going on. If you would like to have a new update of a signal every 100 µS, or just want to monitor one channel, the following main code block should be substituted (for the earlier presented two channel version):

```
#ifdef dacoutput  // conditional compilation so this functionality
                  // can be turned off
// PFC on time
word0 = (SMED5->TMR_T3H)<<8;
word0 = word0 + (SMED5->TMR_T3L);
if(word0>255) word0=255;
UART->DR = word0;
#endif
```

In this case, just the PFC on time is transmitted, once every 100 µS. Additionally, the last code block, which changes the data length from 8 to 9 bits, should be omitted.

Discovery board implementation

Figure 1 shows all the required connections to the STM32F4 discovery board, except for the USB power supply which plugs in at the left. On the top of the picture, the black, pink, and blue wires are, respectively, GND, 5V, and TX (from the STLUX target). For use with the STLUX streetlight demo, these connect to connector J2 pins 1, 3, and 4 respectively. The blue wire connection is at PC7, which is the RX connection for USART6 of the STM32F4. On the lower side of the board, the yellow and green wires connect to pins PA4 and PA5 which are, respectively, the DAC0 and DAC1 diagnostic signals. These would generally be connected to an oscilloscope probe. The two pushbuttons are CPU reset (black) and user (blue).

The firmware which runs on the discovery board operates in one of three modes:

1. Single Channel
2. Dual Channel
3. test/calibration

By default, the program will power up in mode 2, dual channel mode. This is indicated by the two LEDs (red and green) which will illuminate near the pushbuttons. If you wish to

operate in another mode, it is necessary to use the pushbuttons. To change the operating mode:

1. Press both buttons simultaneously.

2. Release the reset (black) button.

3. Continue to press the user (blue) button and the LEDs will slowly cycle through:

red and green (two channel)

red, green, and blue (test/calibration)

red (single channel)

4. When the desired operating mode is displayed, release the user button. If your desired operating mode "goes past", just wait for it to cycle around again.

The program will continue in the selected operating mode, as indicated by the LEDs until the CPU is reset.

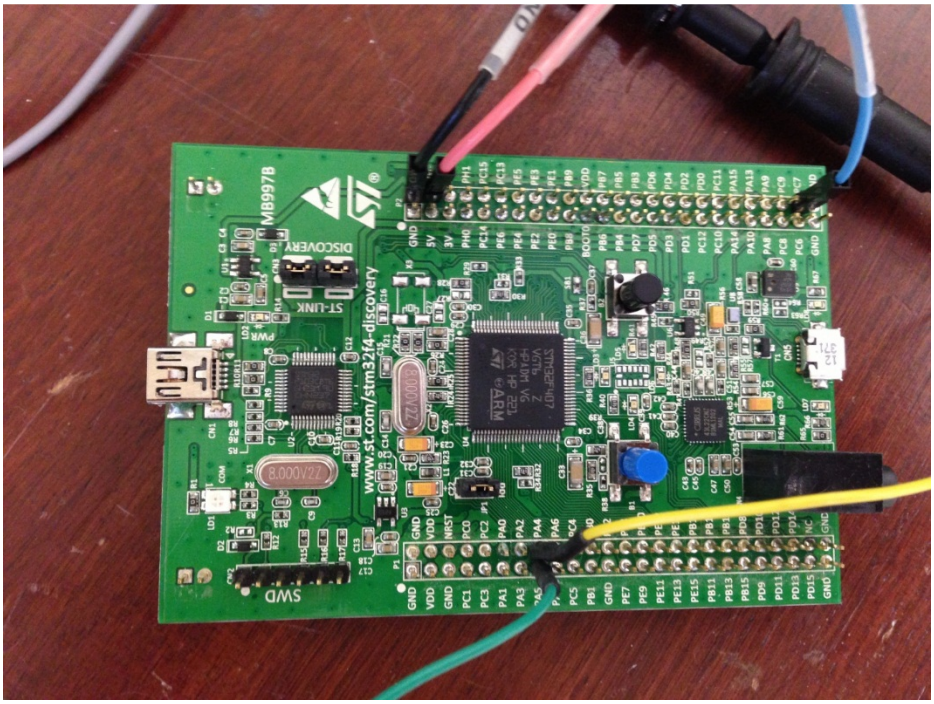
A word of description about the operating Modes 1 and 2 which have been previously described: Be sure to use the proper mode to match the code you have decided to run in your STLUX application (1 channel or 2). Outputs will be quite difficult to interpret if you have a mismatch. When operating in single channel mode, the same signal is output from both DACs. Since the DACs of the STM32F4 are 12-bit DACs, the 8 bit signal is automatically left shifted by 4 places so that the DAC output will swing full scale.

DAC calibration and resolution

When operated in mode 3 (test/calibration), the discovery board will output a positive going sawtooth waveform at DAC1 (PA4) and a negative going sawtooth at DAC2 (PA5). Each waveform has a period of about eight milliseconds and represents the DAC ranging over the entire effective 8 bits of resolution from 0 to 255. If your scope has a fine adjustment on the vertical scaling, a useful calibration may be to adjust the scaling and offset so that the range just reaches over five divisions on your scope. In this way, each division can be read as 51 "counts" out of the total 255.

A word about DAC resolution: Although it may seem a shame to "waste" the 12 bit native resolution of the STM32 DACs, attempting to send more than 9 bits (including DAC channel) with each transmission would change the timing and reduce the update rate from one update every 100 µS. It also makes no sense since virtually all modern digital oscilloscopes have only 8 bits of resolution in their high speed analog to digital converters. This may be surprising, but have a look at the specs of your instrument.

Figure 1. F4 DISCOVERY BOARD



Support material

List any related support material such as evaluation boards, Gerber files etc. and any documents which might be useful to the customer, for example the datasheets, the evaluation board user manual etc.

Related design support material
STM32F4DISCOVERY, Discovery kit for STM32F407/417 lines
Documentation
Datasheet, STM32F4DISCOVERY, Discovery kit for STM32F407/417 lines
Datasheet, STLUX385A, Digital controller for lighting and power supply applications

Revision history

Date	Version	Changes
27-Feb-2015	1	Initial release

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.