BiasDACusr.doc
MacArthur
7/19/01: first writing
10/02/01: corrected typos under DIPSWITCH SETTINGS

BiasDAC User/Configuration Guide

This document contain information on installing and configuring a BiasDAC. Other useful documentation:

- BiasDACspec.com contains performance specifications.
- CommsProtocol.doc describes the communications protocol for all fiber-ring devices, including BiasDAC.
- BiasDAC Schematic diagram.
- BiasDAC Bill of Materials.
- BiasDAC Mechanical drawings.
- Master/Slave documentation
- High-Frequency Filter documentation

DIPSWITCH SETTINGS

There is an 8-switch DIPswitch located in the upper left of the board. For the following discussion, "OFF" means the rightmost part of the switch (the side that says "open") is depressed, and "ON" means the leftmost part (the side that has the switch numbers) is depressed. The three uppermost settings determine the baud of the comms interface.

8	7	6	
OFF	OFF	OFF	baud define in EEPROM (see commsprotocol.doc)
OFF	OFF	ON	9600
OFF	ON	OFF	14.4K
OFF	ON	ON	19.2K
ON	OFF	OFF	38.4K
ON	OFF	ON	57.6K
ON	ON	OFF	undefined
ON	ON	ON	undefined

The default setting is ON-OFF-ON (57.6K)

Each device on a fiber loop must have a unique ID. Commsprotocol.doc allows up to 61 unique device IDs. These IDs are set either in EEPROM (with the Change Device ID command) or with dipswitches 1-5. If the dipswitches are all set to 0 (OFF), then the ID previously stored in EEPROM is used. Otherwise, the dipswitches determine the ID.

As there are only 5 dipswitches, that means that there are only 31 unique IDs which can be set with dipswitches. As a practical matter, it's probably best to rely on dipswitches rather than EEPROM settings, as are more user-friendly.

Switch 5 is the MSB of the ID number, and switch 0 is the LSB. The default value is ID # 1, meaning switch 1 is ON and switches 2-5 are OFF.

JUMPER SETTINGS

There are two sets of jumpers located in the upper right of the board. The lower set has the label "UART-RX" beneath it. It selects the serial source feeding the on-board microcontroller. When the jumper is in the leftmost setting, the source is the fiberoptic receiver. The middle setting is for the RS485 differential receiver. The rightmost setting is for the RS232 receiver.

The upper jumper set has the label "RS232-TX" above it. It determines what is transmitted out the RS232 transmitter. In the leftmost setting, the fiberoptic receiver is routed directly to the RS232 transmitter without going through the microcontroller. In the middle setting, the RS485 receiver is routed to the RS232 transmitter. In the rightmost setting, the microcontroller is routed to the RS232 transmitter.

Don't panic. This seems to be a difficult topic for folks to understand, so I'll give a few examples of how to set up these jumpers. It's important to understand that the communications protocol is based on a loop topology: data comes from the host PC, into the first device, then from the first device to the second, and so on, then from the last device back to the host PC.

Start with the simple case of a single device hooked up to a PC through RS232. Simply plug a 9-pin RS232 cable from the PC into the device, and set the lower jumper to the rightmost setting. This connects the device's RS232 receiver to the microcontroller. Setting the upper jumper to the rightmost setting connects the microcontroller to the RS232 transmitter. This completes the loop: PC to microcontroller to PC.

Now let's replace the RS232 cable with a fiber link (using an RS232-fiber interface box available from the Electronics Shop). This time, the lower jumper is set to the leftmost setting, which connects the fiberoptic receiver to the microcontroller. The upper jumper setting doesn't matter, as the fiberoptic transmitter is *always* connected to the microcontroller.

If you wanted to add another device to this configuration, you would run a fiber from the transmitter of the first device to the receiver of the second, and a fiber from the transmitter of the second device to the receiver of the RS232-fiber interface box. You can keep this up to a total of 61 devices sharing a fiber ring.

Now let's get tricky. Suppose you don't want to pony up the $200 for an RS232-fiber interface box. You can make the first device perform the interface. Connect the first device to the PC with an RS232 cable. Set the lower jumper to the rightmost setting (RS232 to microcontroller). Set the upper jumper to the *leftmost* setting (fiber receiver to RS232 transmitter). Put a fiberoptic cable between the transmitter and the receiver of the next device in the loop. Connect the last device in the loop back to the fiberoptic receiver of the first device. The upper jumper will close the loop by routing the fiberoptic receiver to the RS232 transmitter, and then to the PC.

One last case: you’ve got two boards sharing an enclosure, and you’ve wisely chosen to use the Master/Slave kit to connect and synchronize them. In addition to synchronizing the system clock, the kit connects the serial ports of the boards together over the RS485 interface. The correct jumper settings are:

For the Master: lower jumper is set to leftmost position for a fiberoptic input and rightmost position for RS232 input. If there are only two devices in the loop and the master is using the RS232 interface, then set the upper jumper to the middle position, which connects its RS232 transmitter to the RS485 receiver from the slave.

For the Slave: lower jumper is set to middle position, which connects the RS485 receiver to the microcontroller. This means that serial data is coming directly from the master without leaving the enclosure. Upper jumper is don’t-care.

To connect the Master/Slave into a fiber ring, run a fiberoptic cable into the master’s receiver, and another cable from the slave’s transmitter to the next device in the chain.

BIASDAC I/O FLAG SPECS

MECHANICAL SPECS

The Flags connector is located on the left side of the PCB, and labeled J6. Pin 1 is the pin closest to the rear of the board (the side with the heat sink). Pin 1 is identified by a very faint arrow on the left side of the connector.

The mate for J6 is a 6-pin Waldom plug, Digi-Key # WM2904-ND. You will also need 6 contacts (WM2555-ND) that are inserted into the plug.

The pinout is:
PIN 1 = Digital Ground
PIN 2 = INFLAG1
PIN 3 = INFLAG2
PIN 4 = OUTFLAG1
PIN 5 = OUTFLAG2
PIN 6 = Digital +5V

Pin 6 can be used as a source for SMALL AMOUNTS of +5V, e.g. for pull-up resistors.

INPUT FLAGS

Electrically, the input flags are LS gates, and expect LSTTL logic levels.

The input flags can be tested in program mode, as a trigger source for the Wait for Trigger command. Note that in the protocol spec, INFLAG1 is called PORTB.0 and INFLAG2 is called PORTB.1.

The flags can also be tested diagnostically with the Read From Memory command (refer to the protocol spec for details). INFLAG1 is mapped to bit 0 of register address 6, and INFLAG2 is mapped to bit 1 of address 6. To perform the read, send the following string of bytes:

```
00000010   (0x02)
00000000   (0x00)
00000110   (0x06)
00000000   (0x00)
00000000   (0x00)
```

BiasDAC will replace the last two bytes with the value of register 6. The status of INFLAG1 and INFLAG2 will appear in bits 0 and 1, respectively, of the last byte.

OUTPUT FLAGS

Electrically, the output flags are driven by open-collector drivers (DS75452). Because they are open-collector, you will need to add pull-up resistors (approx 2.2K) between the flags and +5V in order to generate TTL-compatible signals.

Each driver can sink up to 300 mA, and can tolerate up to 20V when off. Therefore, it is possible to directly drive small 12V relays (always remember to install snubber diodes across the relays).

The flags are activated with the Set/Clear Flags command, described in the protocol spec. Note that setting the flag turns on the driver, which brings the output LOW. The commands are:

01011100	(0x5C)	sets OUTFLAG1
01011000	(0x58)	clears OUTFLAG1
01011101	(0x5D)	sets OUTFLAG2
01011001	(0x59)	clears OUTFLAG2