The BitBox

Hardware Reference Manual

BitFlow, Inc.
400 West Cummings Park, Suite 5050
Woburn, MA 01801
USA
Tel: 781-932-2900
Fax: 781-933-9965
Email: support@bitflow.com
Web: www.bitflow.com
Revision 1.0

Revision History:

Revision	Date	Comments
A.0	2016-09-27	First release
A.1	2016-10-01	Minor Corrections

Table of Contents

Preface

Chapter P

P.1 Purpose

This Hardware Reference Manual is intended for anyone using the BitBox general purpose I/O module. The purpose of this manual is two-fold. First, this manual completely describes how the module works. Second, it is a reference manual describing in detail the functionality of all of the registers on the associated frame grabber.

P.1.1 Support Services

BitFlow, Inc. provides both sales and technical support for the BitBox family of products.

P.1.2 Technical Support

Our web site is www.bitflow.com.

Technical support is available at 781-932-2900 from 9:00 AM to 6:00 PM Eastern Standard Time, Monday through Friday.

For technical support by email (support@bitflow.com) or by FAX (781-933-9965), please include the following:

Product name
Camera type and mode being used
Software revision number
Computer CPU type, PCI chipset, bus speed
Operating system
Example code (if applicable)

P.1.3 Sales Support

Contact your local BitFlow Sales Representative, Dealer, or Distributor for information about how BitFlow can help you solve your most demanding camera interfacing problems. Refer to the BitFlow, Inc. web site (www.bitflow.com) for a list of North American representatives and worldwide distributors.

P.1.4 Conventions

Table P-1 shows the conventions that are used for numerical notation in this manual.

Table P-1 Base Abbreviations

Base	Designator	Example
Binary	b	1010b
Decimal	None	4223
Hexidecimal	h	12fah

Table P-2 shows the numerical abbreviations that are used in this manual.

Table P-2 Numeric Abbreviations

Abbreviation	Value	Example
K	1024	256K
M	1048576	1M

P.2 Bitfield definitions

P.2.1 Example Bitfield Definition

Here is what each bitfield definition looks like:

BITFIELD R/W, CON0[7..0], Cyton-CXP, Axion-CL

Bitfield discussion.

P.2.2 Bitfield Definition Explanation.

The definitions is broken into three sections (see Table P-3).

Table P-3 Bitfield Sections.

Section	Meaning
Bitfield name	This is the name of the bitfield. This name is used to program this bitfield from software or from a camera configuration file. When programming bitfields from software using a BFRegPeek() or BFRegPoke() functions, the bitfield is preceded with “REG_”. For example the bitfield CFREQ is referred to in software as REG_CFREQ.
Bitfield details	This section describes how the bitfield is accessed. The first part describes how the bits can be accessed. For example R/W means the register can be both read and written. See theTable P-4 for details.The second part is the wide register that the bitfield is located in. In the example above this bitfield is in CON0. Following the wide register name is a bitfield location description, in hardware engineering format. For example, [7..0], means the bitfield has 8 bits, location in positions 0 to 7. Finally this section also indicates if the register is specific to only one product family.
Bitfield discussion	This section explains the purpose of the bitfield in detail. Usually the meaning of every possible value of the bitfield is listed.

Table P-4 explains the abbreviations used in the bitfield definitions.

Table P-4 Abbreviations

Access	**Meaning**
R/W	Bitfield can be read and written.
RO	Bitfield can only be read. Writing to this bit has no effect.
WO	Bitfield can only be written. Reading from this bit will return meaningless values.
Karbon-CL	This bitfield is functional only on the Karbon-CL.
Karbon-CXP	This bitfield is functional only on the Karbon-CXP.
Neon	This bitfield is functional only on the Neon
R64	This bitfield is functional only on the R64 family.
Alta	This bitfield is functional only on the Alta family.
Cyton-CXP	This bitfield is functional only on the Cyton-CXP family
Axion-CL	This bitfield is functional only on the Axion-CL family

General Description and Architecture

Chapter 1

1.1 The BitBox

The purpose of this chapter is to explain, at a block diagram level, how the BitBox works. Currently there is just one model in the BitBox family:

IOB-ISO35-C144 - I/O module, ISO DIN 35 rail mount, 144 pins

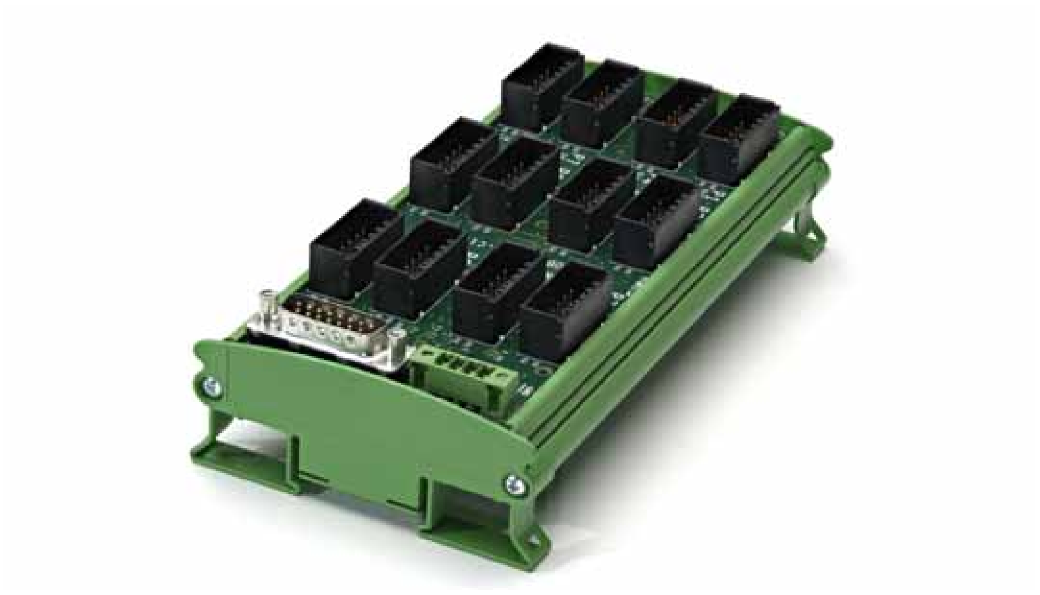

Figure 1-1 The BitBox

The BitBox is a general purpose I/O module that extends the I/O capabilities of the frame grabber and hence the PC. The target application space is complex machine vision system that require a large number of controls signals going to an from the host application. The usual solution is to buy general purpose I/O board, and run a large cable from the “rail” and other devices to the huge connector on the back of the PC. The BitBox solves a number of problems with this approach. It breaks out the signals right on the rail. It uses a small low cost cable to bring the signals to the PC. Finally, it uses the frame grabber that is already in the system as a host controller, no need to buy another board just for I/O!

The BitBox offers 36 fully programmable inputs and outputs in various electrical formats (e.g. TTL, Differential, etc). The BitBox inputs can be routed to various destinations on the frame grabber. They can cause interrupts, trigger cameras or other device. The level of every input can be read at any time by the host software (via registers on the frame grabber). The outputs can be driver by a number of different sources. They can be driven by programmable wave form generators, they can be driven from other inputs, they can be driven by static register that can be toggled by host software at any time.

The BitBox inputs and outputs are controlled by a small standard cable the goes from the frame grabber to the BitBox. Even though BitBox can simultaneously manage 72 signals, the cable between the BitBox and the frame grabber has only 15 wires.

1.1.1 Input and Output Signal Types

The BitBox has four types of inputs:

TTL
Differential (LVDS)
Optocoupler
24 V

The BitBox has four types of outputs:

TTL
Differential (RS-422)
Optocoupler
Open Collector

1.1.2 Connection Signals

All of the inputs and all of the outputs are simultaneously available. The user can mix and match I/O signal types as needed by their applications.

The I/O connectors on the BitBox are grouped in blocks of 12 signals. Each block of 12 takes a 12-pin block connector. This makes wire harness fabrication much easier as the harness can be built away from the machine chassis, and connected to the BitBox at final system assembly. The 12-pin block connectors use a screw-less friction system that makes inserting and removing wires quick and simple.

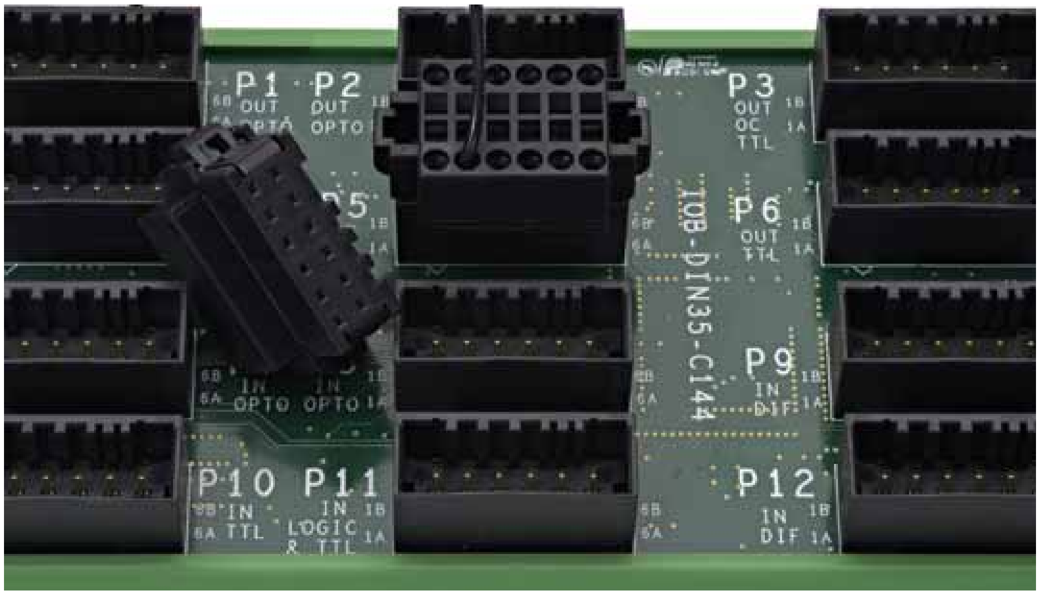

Figure 1-2 12-Pin Block Connectors

Power for the BitBox can come from one of two sources. It can come from a local power source that provides between 5 to 24 volts. Alternatively, it can come from the PC via the control cable. The control cable, in addition to carrying the control signals, can also carry power. Power in this case is provided by the PC's internal power supply.

1.1.3 Minimum Requirements

The minimum components required to use the BitBox are:

A cable between the BitBox and the PC
At least one 12-pin block connector
A BitFlow Cyton or Axion frame grabber.
A power source (direct or via the frame grabber)

These components are explained in further detail in later sections of this manual

1.2 Wiring the BitBox

Wiring the BitBox is quite simple. The following subsections describe all that is needed:

1.2.1 Wiring The 12-Pin Block Connectors

First you must determine to which BitBox input or output each of your signals will be connected. Then determine which 12-signal block each belongs to. You can wire each of these groups to one 12-pin block connector.

Note: These support wire sizes from 20 to 26.

Adding a wire to one of these block connectors is easy:

Push the tip of the insertion tool into the square hole
Push the stripped end the wire into the nearest round hole
Remove insertion tool

Figure 1-3 illustrates the locations of these holes

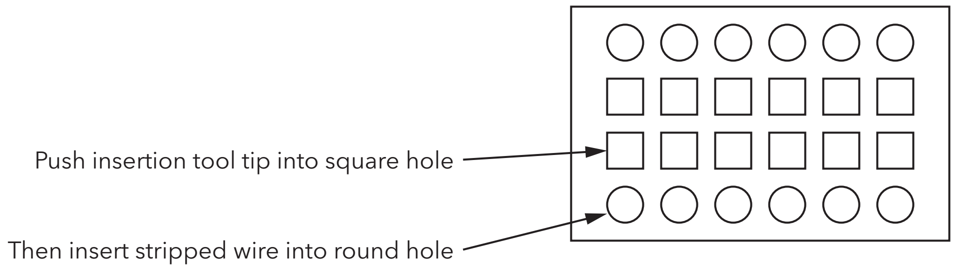
Push insertion tool tip into square hole
Then insert stripped wire into round hole

Figure 1-3 12-Pin Block Connector Wire Insertion

To remove a wire, simply re-insert the tip of the insertion tool, and pull the associate wire out.

Note: The insertion tool part number is ACC-BOX-2.5-50 and is available from BitFlow.

Note: The 12-pin block connectors are part number CONN-BOX-IO12 and are available from BitFlow.

1.2.2 Inserting/Removing 12-Pin Block Connectors

To insert the 12-pin block connectors into the BitBox, simply push the connectors into the appropriate mating socket. The connectors are key so it is not possible to put the connectors in the wrong way.

To remove the 12-pint block connectors from the BitBox, squeeze the two sides of the connector (see Figure 1-4) and pull firmware away from the BitBox.

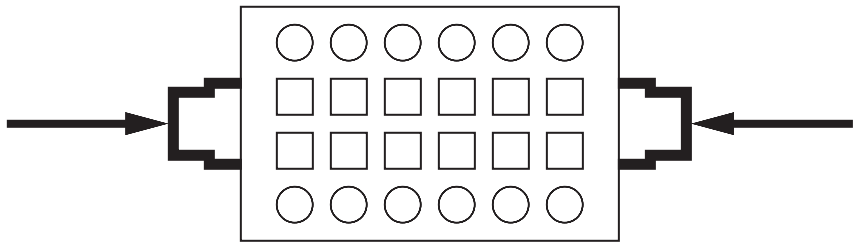

Figure 1-4 Removing 12-Pin Block Connectors

1.3 BitBox Cables

There are a number of cabling option available to connect the BitBox to the frame grabber in your PC. Low cost ribbon cables can be used for development environments or when the distance between the PC and the BitBox is small and there is relatively little risk of electrical interference. For longer distances and in electrically noisy environments, shielded cable that runs all the signals on twisted pair conductors is recommended. In addition, a power cable is needed, either directly to the box or internally from the PC. Figure 1-5 illustrates all of the cable options and how they are interconnected.

Note: All of the cables shown below are availale from BitFlow. In adition, wiring diagrams are available for customers who wish to fabricate their own cables.

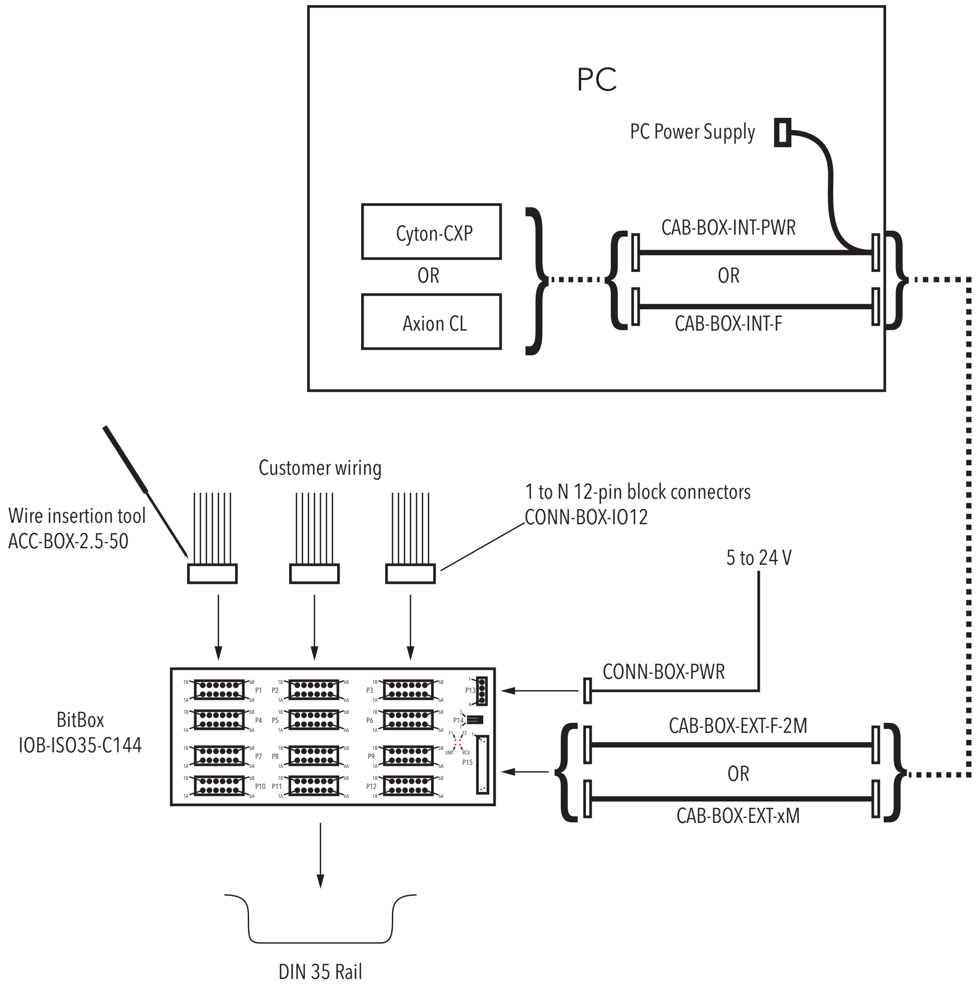

Figure 1-5 BitBox Cable Configurations

1.4 BitBox Signal Routing

1.4.1 Input Signal Routing

The BitBox Inputs can be routed to a number of different destinations on the frame grabber. For more details please see Section 4.3. Each input's level can be read at anytime by software peeking the corresponding bitfield. See Section 5.1 for bitfield names that correspond to each input. In addition, every Bitbox input can be routed to any of the internal signals on any VFG. The list of internal signals is as follows:

VFGx_TRIG_SEL
VFGx_ENCA_SEL
VFGx_ENCB_SEL
VFGx_ENCDIV_SEL
VFGx_ENCQ_SEL

Each of these signals can be used in a variety of ways. They can trigger acquisition, they can trigger the timing sequencer, they can be sent to the camera or they can be sent back off the frame grabber to an output inside the PC (on the frame grabbers own I/O connector) or an output on the BitBox. The routing is controlled by the frame grabbers registers. Please see Section 5.1 for more information on the individual control registers.

1.4.2 Output Signal Routing

Each BitBox output can be driven by a number of different sources. There is a simple register for each output that can force the output to a high or low state. This same register can also be set to drive the output from a dynamic source. For each output there are a few choices for the dynamic source:

VFG0_CCx
VFG1_CCx
VFG2_CCx
VFG3_CCx
VFG0_ATS_CTx

Please see the Table 5-1 for a complete list.

Each VFG has four dynamic CCx signals: CC1, CC2, CC3 and CC4. Please see Section 4.5 on more information on these signals. In addition, VFG0 has four more dynamic signals VFG0_ATS_CT0, VFG0_ATS_CT1, VFG0_ATS_CT2 and VFG0_ATS_CT3. See Section 3.1.1 for more information on these signals. Not all models of Cytons and Axions have four VFGs, therefore on some models, not all of the dynamic sources will be available. However, all models of frame grabbers will have at least VFG0_CCx and VFG0_BCTx.

The CCx signals can, in turn, be driven by a number of different sources. For example, they can be driven by a Timing Sequencer (TS) which can create programmable wave forms, they can be driver by just about any of the inputs (which can also be coming from the BitBox). Please see Section Figure 4-7 for more information on how these signals can be routed.

The routing of these output signals is controlled by a set of registers on the frame grabber, please see Section 5.1 for more information on how to route these registers.

BitBox Electrical Interfacing

Chapter 2

2.1 The BitBox Inputs and Outputs

The BitBox has four types of inputs :

TTL
Differential (LVDS)
Optocoupler
24 V

The BitBox has four types of outputs:

TTL
Differential (RS-422)
Optocoupler
Open Collector

This chapter provides details on the recommend ways to electrically interface to these inputs and output.

2.2 The TTL inputs

The TTL input uses a 74LVTH241 (see Figure 2-1). The input has a 1K pullup to 3.3V, see figure 1. If the input is unconnected, it will read1. Figure 2-2 shows how to connect a TTL driver to the BitBox TTL input. The BitBox and the user's grounds (GND) must be connected.

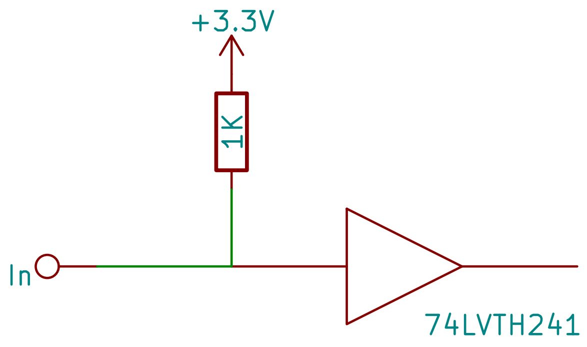

Figure 2-1 The TTL Input

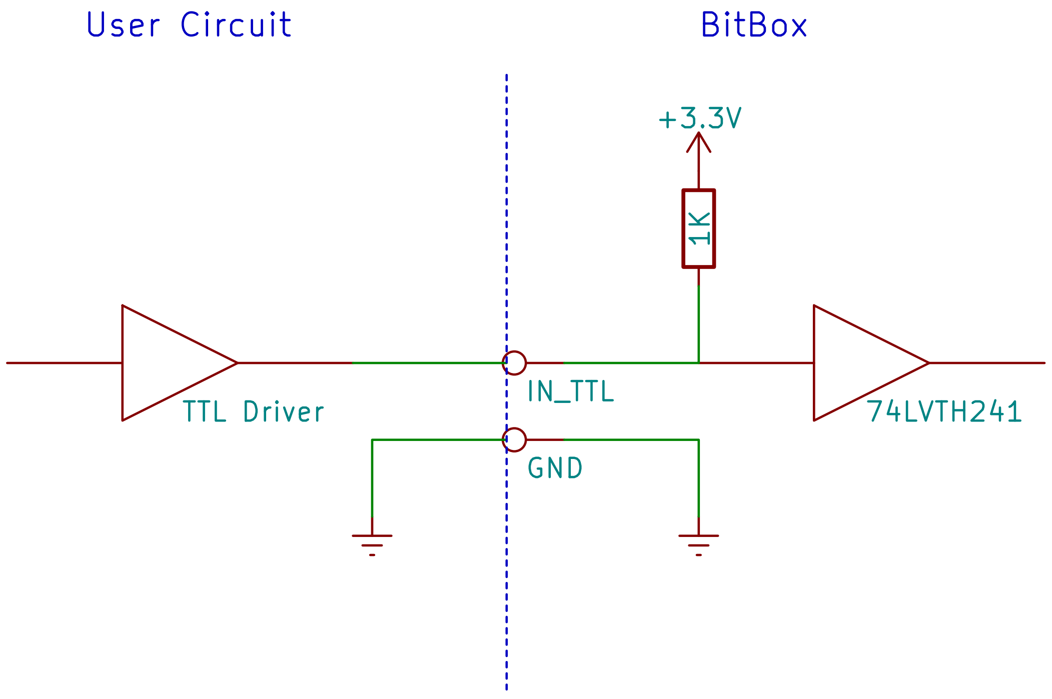

Figure 2-2 Connecting the TTL input

The pullup will allow an easy connection of a mechanical closure circuit, see Figure 2-3. It is recommended that the user applies a filter on that IO to clean up the mechanical chatter. See the registers TRIG_FILTER, ENCA_FILTER and ENCB_FILTER in Section 5.1.

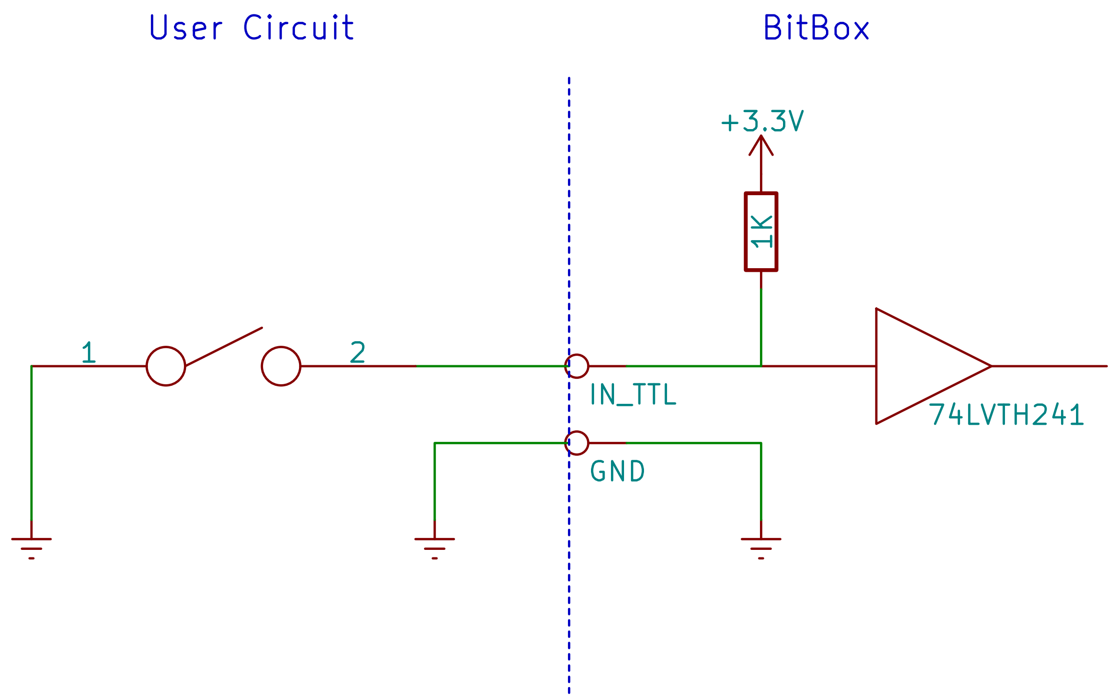

Figure 2-3 Connecting a closure element to the TTL input

2.3 The TTL Outputs

The TTL output is implemented with a 74LVTH241 (see Figure 2-4). Figure 2-5 shows how to connect the TTL output. The user's and the BitBox's grounds (GND) must be connected.

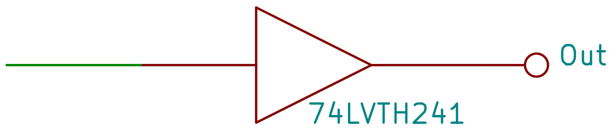

Figure 2-4 The TTL Output

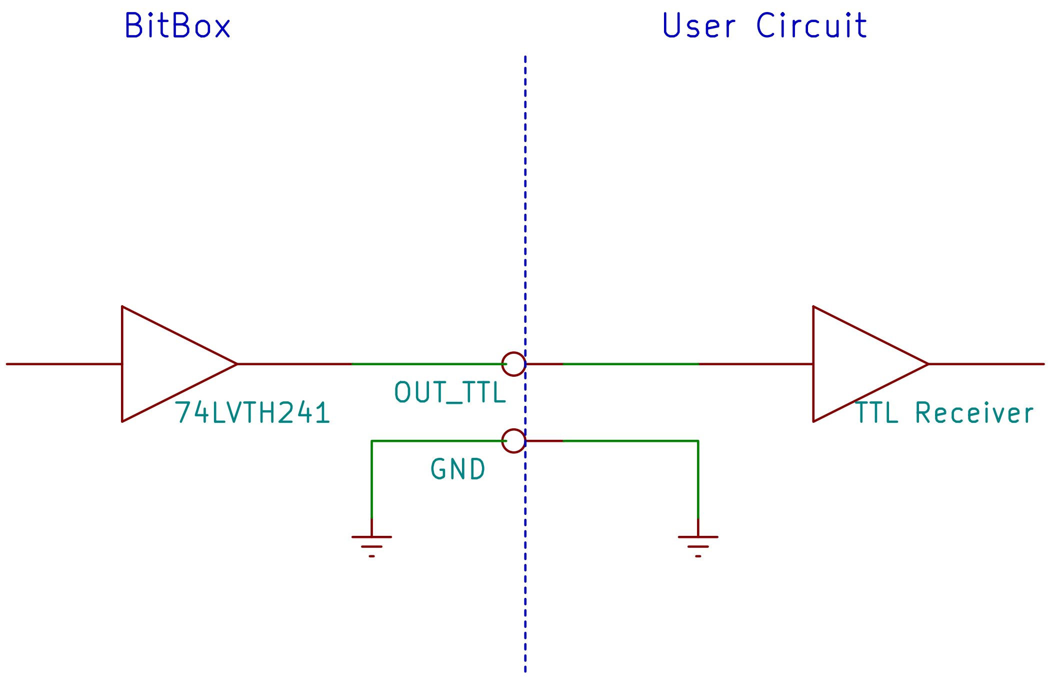

Figure 2-5 Connecting the TTL Output

2.4 The Differential Inputs

The differential inputs are implemented with a SN65LVDS3486, which are RS644. They have a 100 Ohm termination, see Figure 2-6. Figure 2-7 shows the connection of the differential receiver. The user's and the BitBox GNDs must be connected.

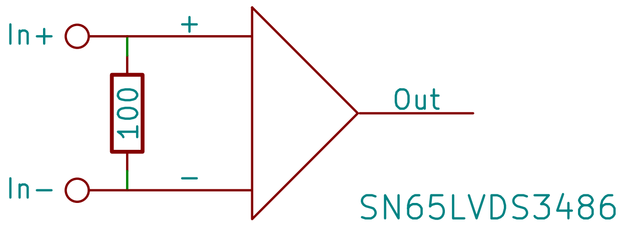

Figure 2-6 The Differential Receivers

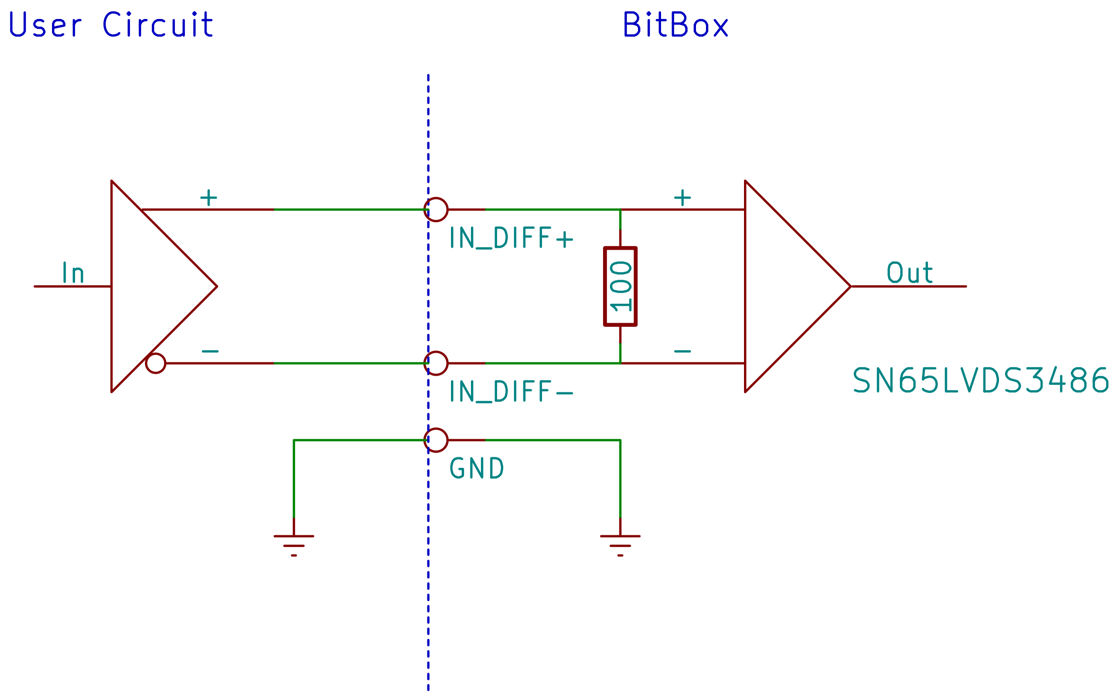

Figure 2-7 Connecting the Differential Receivers

2.5 The Differential Outputs

The differential outputs are implemented with a DS34LV87T, which are RS422, see Figure 2-8. Figure 2-9 shows how to connect the differential outputs. The user's and the BitBox GNDs must be connected. The user should have a 100 Ohm termination.

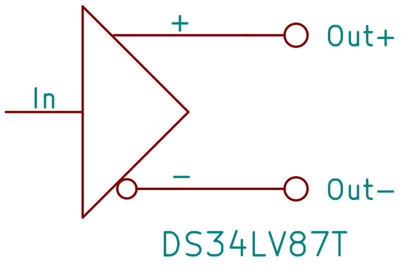

Figure 2-8 The Differential Transmitters

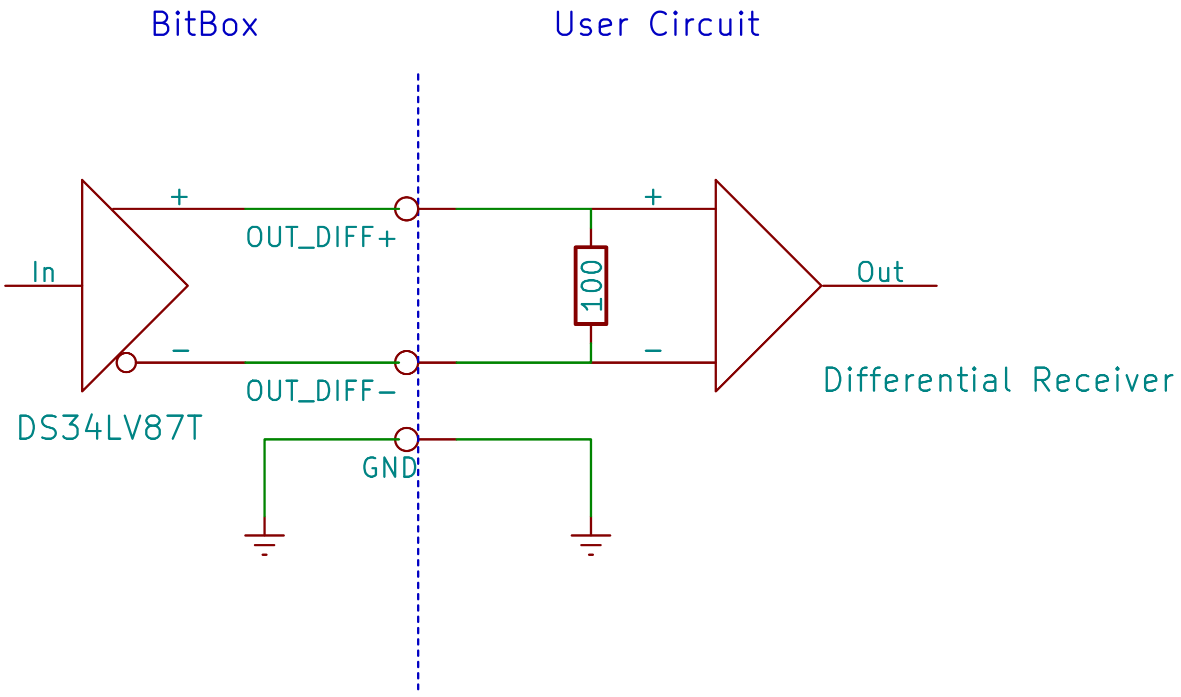

Figure 2-9 Connecting The Differential Transmitters

2.6 The Optocoupler Inputs

The opto input has been implemented with a Vishay SFH6325, see Figure 2-10. The user has access to the anode and the cathode of the input circuit. There is a 200 Ohm resistor in series with the anode. This resistor will limit the current through the input photodiode Warning: care must be exercised when driving the photodiode. The maximum current allowable through the photodiode is 25mA. That means that the maximum allowable voltage on the anode is 5V. Figure 2-11 shows a simple TTL circuit that drives the optocoupler input.The 'HI' of the 74LVTH241 will be ~3.3V. That will result in a current of ~16mA through the photodiode.

The user's TTL driver is connected to the anode input. The GND of the user's TTL driver is connected to the cathode. The user's TTL driver will drive a current through the photodiode. This same current returns to the user's GND through the cathode. The current that flows through the photodiode will generate light inside that photocoupler. This light will induce a current through the output transistor. The key to this circuit is that there is no galvanic connection between the user's circuit and the BitBox. This situation is desirable in many situations, especially in medical applications. No current flows between the user's circuit and the BitBox. Information is passed along as light.

Note: the user's GND and the BitBox GND are not connected.

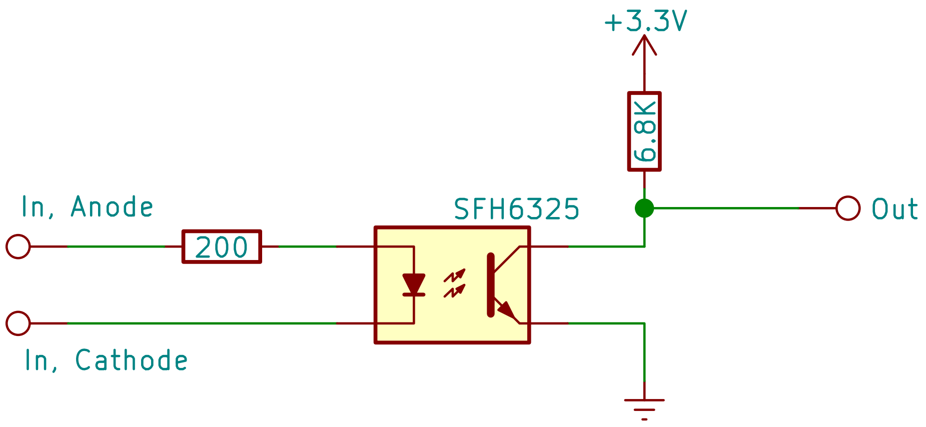

Figure 2-10 The Optocoupler Input

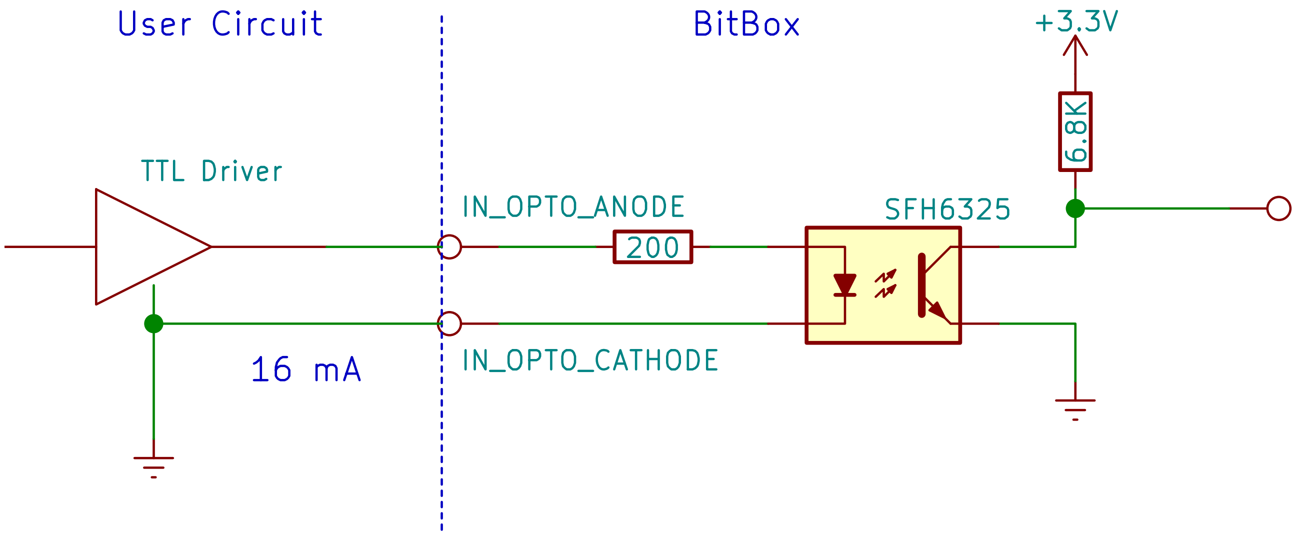

Figure 2-11 Driving the Optocoupler Input with a TTL signal

2.7 The Optocoupler Outputs

The optocoupler used to output signals off the BitBox is a SFH6325, see Figure 2-12. It has three signals that need to connect to the user's circuit:

Opto_Out, the collector of the optocoupler's output transistor.

Opto_GND, the emitter of the optocoupler's output transistor.

Opto_VCC, power to the optocoupler's output stage; the Opto_Out is also connected to the Opto_VCC through a 6.8K pullup resistor.

Figure 2-13 shows a simple connection of the optocoupler output to a user's TTL receiver. The user will supply its own VCC and GND for the optocoupler's output stage. The Opto_Out will drive the TTL receiver. The key feature of this circuit is that there is no galvanic connection between the BitBox and the user's circuit. The BitBox circuitry will drive the input photodiode which will generate a light inside the optocoupler. This light will induce a current in the output transistor.

Note: the user's GND and the BitBox GND are not connected.

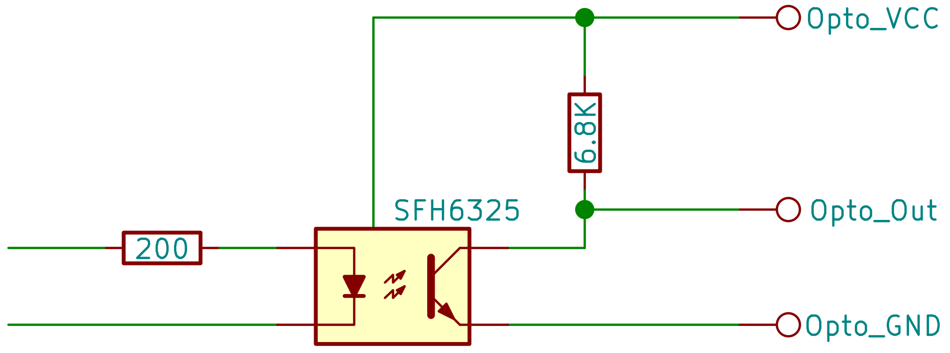

Figure 2-12 The Optocoupler Output

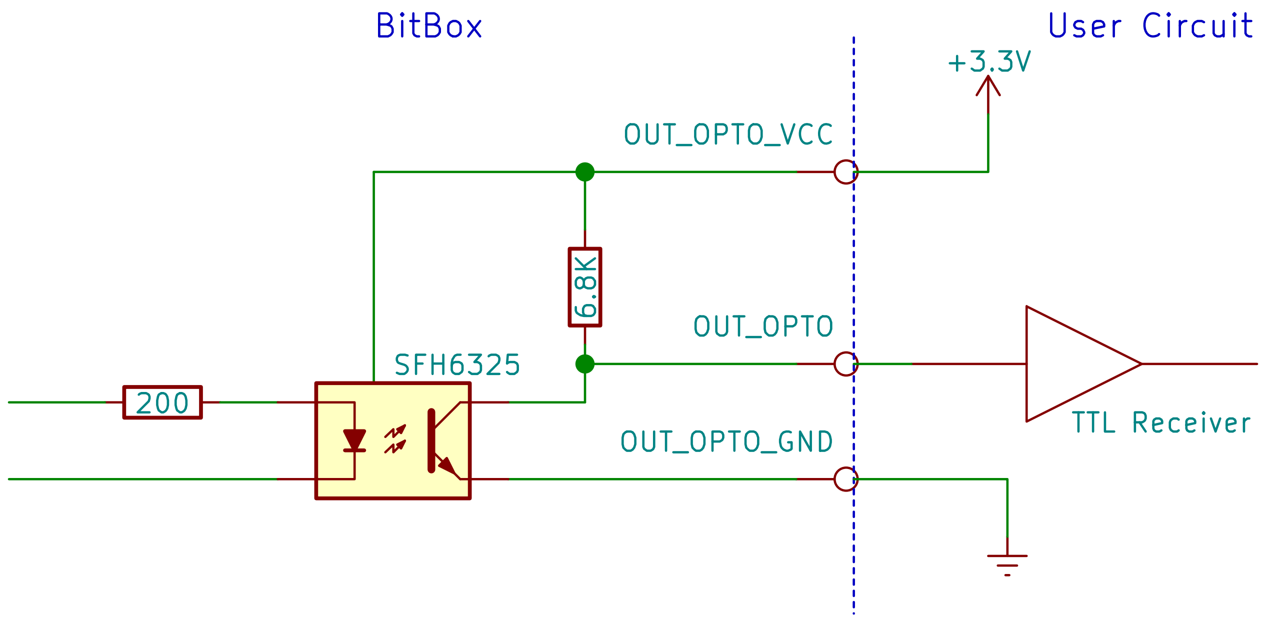

Figure 2-13 Connecting the Optocoupler Output to a TTL Receiver

2.8 The 24V Inputs

The 24V input to the board has been implemented with a TTL receiver and a voltage divider, see Figure 2-14. Figure 2-15 shows how to connect a 24V logic input source. The user's and the BitBox's GNDs must be connected. Any voltage between 12V and 24V on the In_24V input will be recognized as a '1' by the TTL receiver. Maximum allowed voltage on In_24V is 26V.

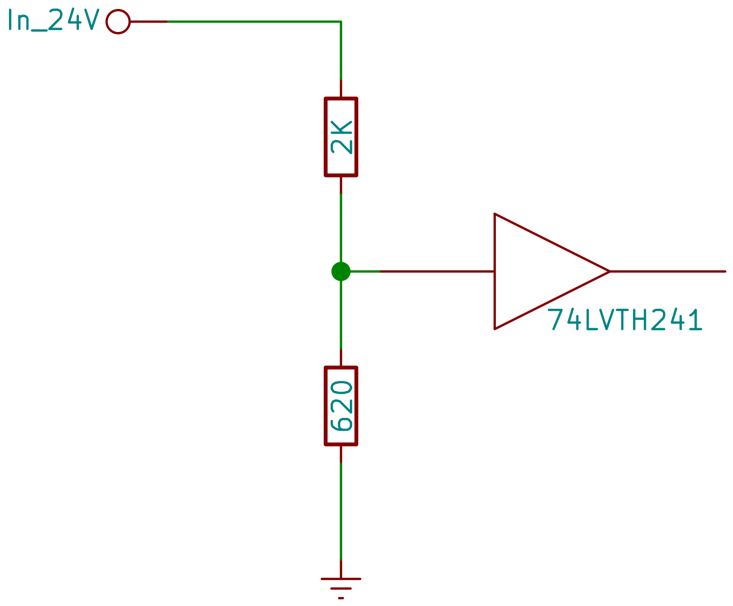

Figure 2-14 Implementation of the 24V Input

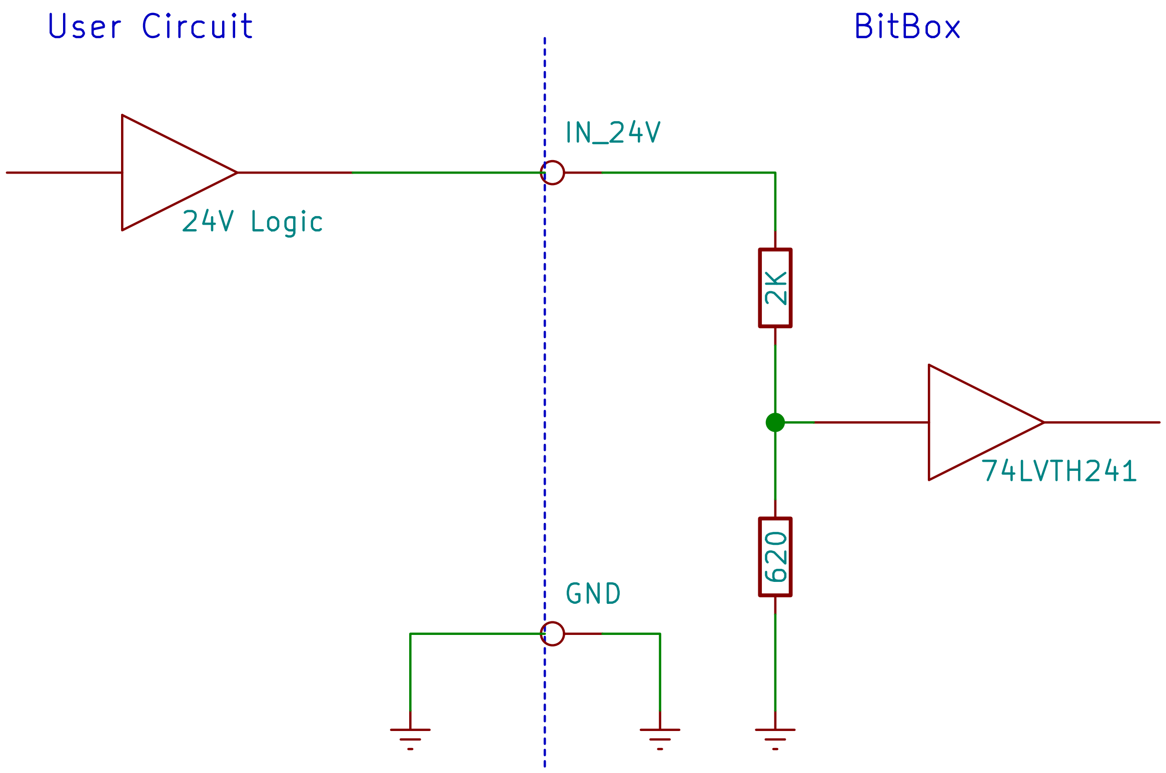

Figure 2-15 Connecting a 24V Input Logic Signal

2.9 The Open Collector Outputs

The Open Collector (OC) output has been implemented with a MMBT2222 transistor, see Figure 2-16. The user has access to the collector of the transistor, OC, and to a 2K pullup resistor, OC_24V. Figure 2-17 shows a simple circuit that supplies a logic output level of up to 24V. The user connects to the OC_24V output the voltage that he wants the output logic to have in the HI state. For a HI of 24V, connect to a 24V power supply. The maximum voltage is 26V. The minimum voltage is 3V. The user's and the BitBox GNDs must be connected. Figure 2-18 shows a simple circuit that is intended to sink current (as opposed to supplying logic levels). The circuit in Figure 2-18 can sink up to 12 mA. The BitBox's on-board 2K pullup resistor limits the current to 12mA for a 24V power supply. The user's and the BitBox GNDs must be connected. Figure 2-19 shows how to connect a circuit that needs to sink 100mA from 24V. The user needs to supply his own limiting resistor, which should be 240 Ohm for 100mA from 24V. The user's and the BitBox GNDs must be connected.

The maximum current allowed to sink through the transistor is 200mA.

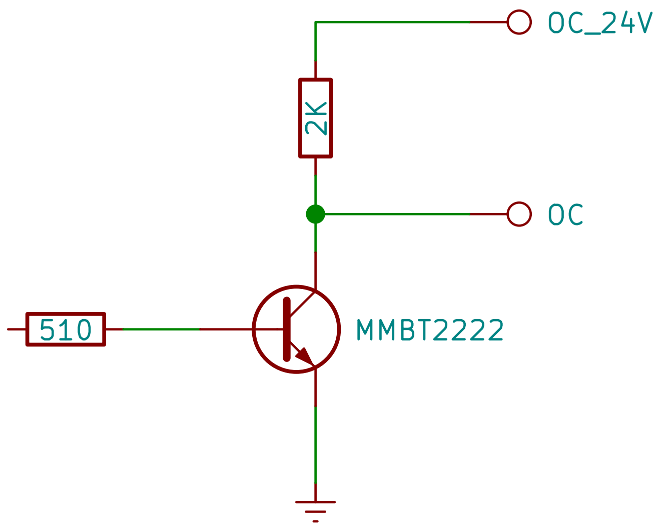

Figure 2-16 Implementation of the Open Collector Output

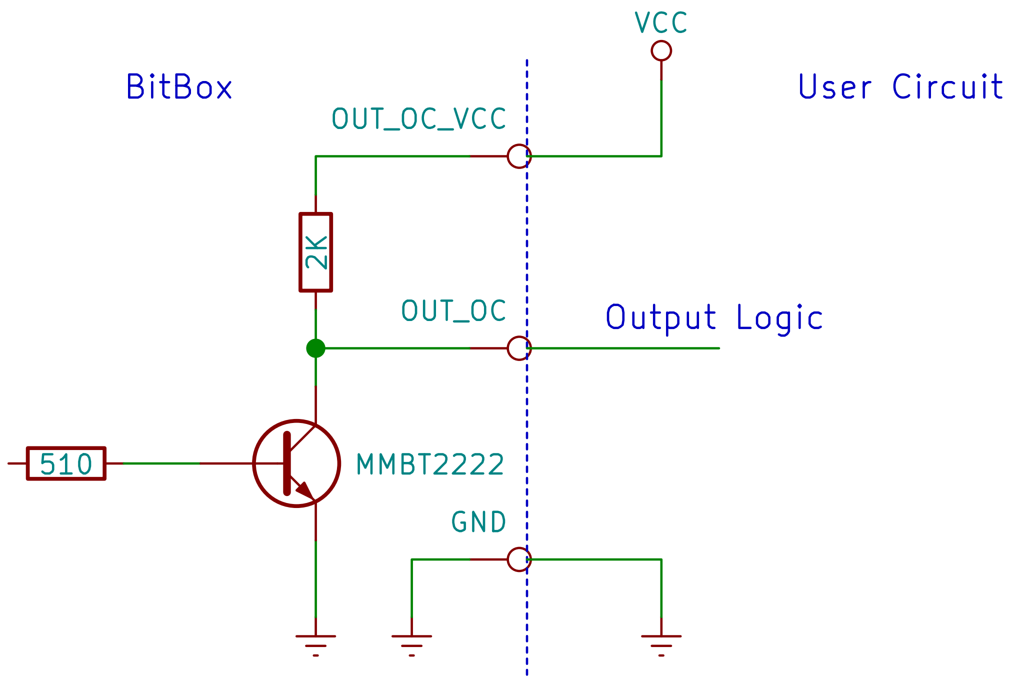

Figure 2-17 Simple 24 V Output Logic

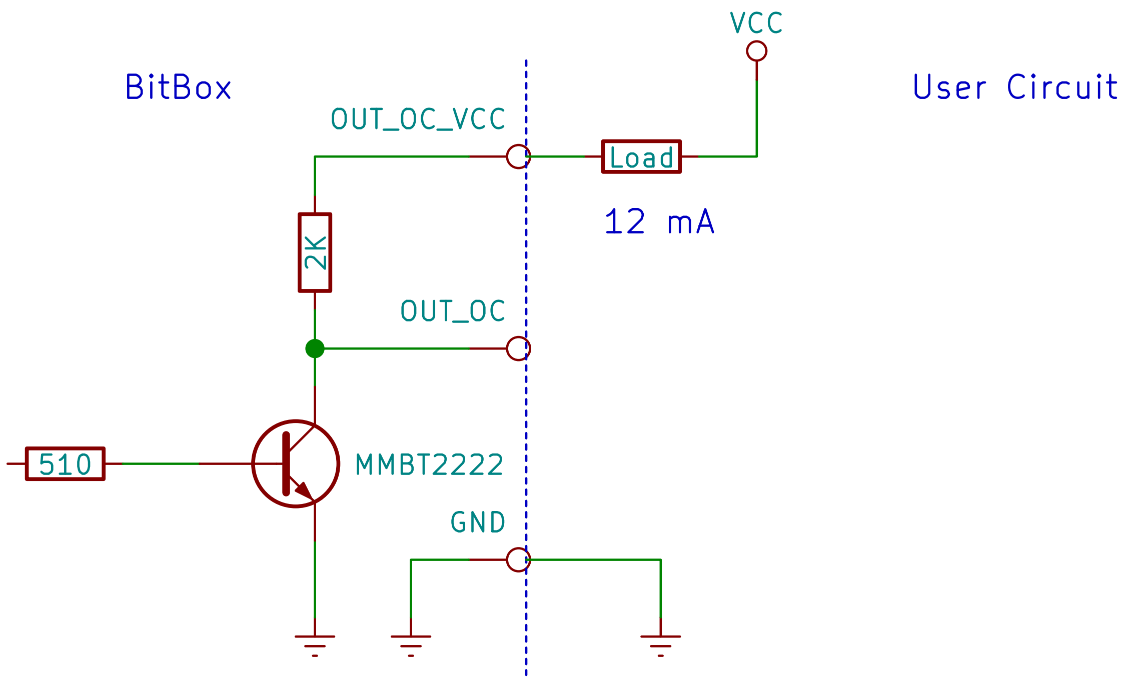

Figure 2-18 Circuit for sinking current, 12mA

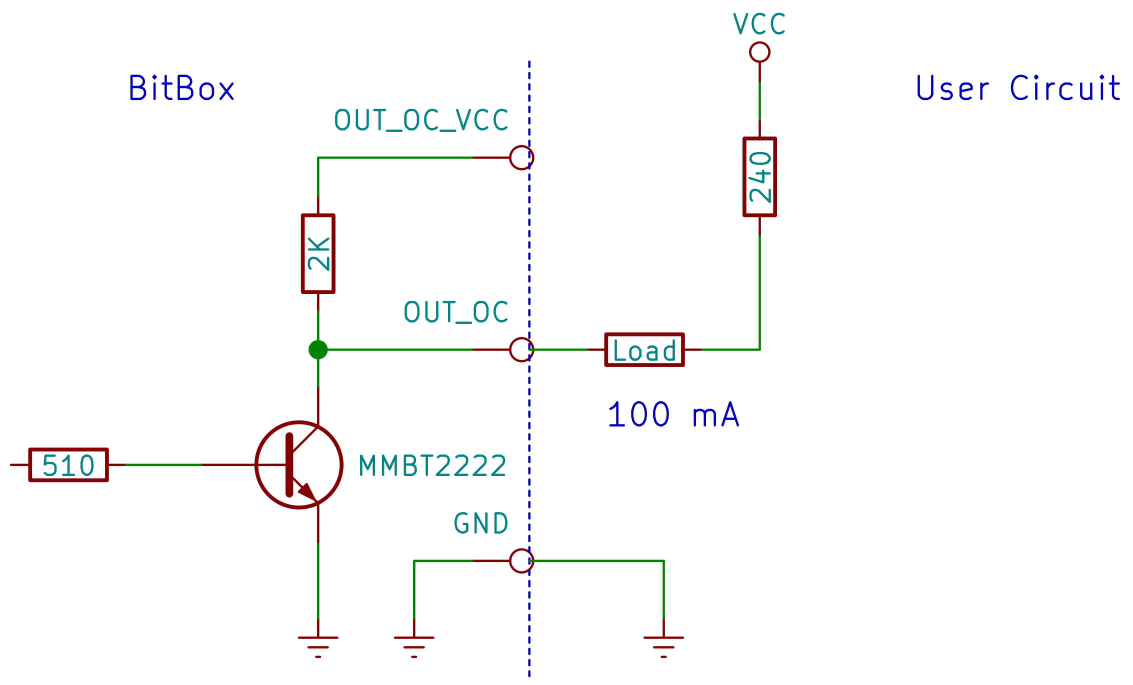

Figure 2-19 Circuit for sinking current, 100 mA

Timing Sequencer

Chapter 3

3.1 Introduction

This section covers the Timing Sequencer (TS) which is available on the Cyton-CXP and the Axion-CL. The TS is a sophisticated programmable pulse generator. The TS takes the place of the NTG on previous models of BitFlow frame grabbers.

The TS improves on the NTG in the following ways:

Driven by a “nice” clock frequency clock so that “normal” pulse sizes and periods can easily be reproduce (for example 1 micro second pulse every 10 milliseconds)
Higher accuracy signals, granularity down to 100 nanoseconds
Supports complex pulse trains of different lengths
Provides synchronized method to switch from one pulse sequence to another
Can be reprogrammed while being used (with some restrictions)
Supports triggering at arbitrary points in a pulse train

3.1.1 The Auxiliary Timing Sequencer

The first Virtual Frame Grabber (VFG0) has two independent Timing Sequencers: TS and ATS. They work exactly the same, and have the same control registers. The Auxiliary Timing Sequencer registers are listed here, even though they work in the exact same way as the main timing sequencer. All of the description that follows applies to both the TS and the ATS.

Note: The Auxiliary Timing Sequencer (ATS) is only available on VFG0.

3.1.2 Description

TS Table

The TS is programmed through the TS registers. The sequence of pulses that the TS will output is programmed by building up “instructions” in the TS table. The table can hold up to 256 instructions, which can create extremely complex signals. The TS Table is programmed indirectly via address/data type registers. Once the table is programmed it can be run at any time via the TS control registers.

Building Pulses

The TS was design to support a wide range of pulse lengths. At the same time, the TS was design to be able to create pulses of very accurate duration. The solution to these two opposing problems is to build up a pulse of a desired length via multiple sub-pulses, each sub-pulse programmed with a different granularity. The following granularities are available:

100 seconds
100 milliseconds
100 microseconds
100 nanoseconds

Each sub-pulse can have a length of 1 to 1023 units, where the units are selected from the list above.

For example, let's say you want a pulse that is, for example, 1.2345678 seconds long. This is done by programing the TS table with three sub pulse as shown below:

Entry 1: 12 * 100 milliseconds = 1.2 seconds
Entry 2: 345 * 100 microseconds = 0.0345 seconds
Entry 3: 678 * 100 nanoseconds = 0.0000678 seconds

When these three entries are "run", they are output sequentially and seamlessly, creating a single pulse of the desired length:

1.2 + 0.0345 + 0.0000678 = 1.2345678

As you can see, this system provides both a wide range of durations as well as very accurate durations.

Pulses can be either high or low (0 or 1). By programming both high pulses and low pulses any pulse train can be created.

Chaining Pulses

Pulses are chained together by a linked list. Each pulse entry has a "next" field which tells the system where in the table the next pulse should come from. This facility is used to build up complex sequences as well as looping sequences.

Triggering

Each entry in the TS table can produce one pulse. Each entry has a "condition" under which it will get executed. The conditions can be immediate. In other words, as soon as the TS gets to this entry, it immediately produces the programmed pules. Or it can be programmed to wait for a trigger. Various trigger conditions are supported.

By adding this condition, the pulse train produced can be run in "one-shot" mode, when one trigger produces one pulse.

3.2 TS_CONTROL

Bit	Name
0	TS_RUN_LEVEL
1	TS_RUN_LEVEL
2	TS_RUN_LEVEL
3	Reserved
4	TS_CT0_DEFAULT_STATE
5	TS_CT1_DEFAULT_STATE
6	TS_CT2_DEFAULT_STATE
7	TS_CT3_DEFAULT_STATE
8	Reserved
9	Reserved
10	Reserved
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15	Reserved
16	TS_IDX_JUMP
17	TS_IDX_JUMP
18	TS_IDX_JUMP
19	TS_IDX_JUMP
20	TS_IDX_JUMP
21	TS_IDX_JUMP
22	TS_IDX_JUMP
23	TS_IDX_JUMP
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	TS_TRIG_SEL
29	TS_TRIG_SEL
30	TS_TRIG_SEL
31	TS_TRIG_SEL

TS_RUN_LEVEL R/W, TS_CONTROL[2..0], Cyton-CXP, Axion-CL, Axion-CL

These bits control the operation of the TS. These bits are used to start and stop the sequencer. They can also be used to program the table to jump to a new section. Jumps are always synchronous (i.e. not immediate). Jumps will only occur from an index in the sequence that has the TS_END_OF_SEQUENCE bit set.

This bit can be read at any time in order get the current status.

The following table shows the command available for this register.

TS_RUN_LEVEL	Meaning
0 (000b)	Idle - TS is not running
1 (001b)	Run - Start running immediately from index in the TS_IDX_ JUMP register
2 (010b)	Jump - Jump to index set in the TS_IDX_JUMP register next time the current index has the TS_END_OF_SEQUENCE bit set to 1
3 (011b)	Stop - Stop running the next time the current index has the TS_END_OF_SEQUENCE bit set to 1
4 (100b)	Abort - Stop running immediately

TS_CT0_ DEFAULT_STATE R/W, TS_CONTROL[4], Cyton-CXP, Axion-CL

This is the output state of CT0 when the TS is Idle.

TS_CT1_ DEFAULT_STATE R/W, TS_CONTROL[5], Cyton-CXP, Axion-CL

This is the output state of CT1 when the TS is Idle.

TS_CT2_ DEFAULT_STATE R/W, TS_CONTROL[6], Cyton-CXP, Axion-CL

This is the output state of CT2 when the TS is Idle.

TS_CT3_ DEFAULT_STATE R/W, TS_CONTROL[7], Cyton-CXP, Axion-CL

This is the output state of CT3 when the TS is Idle.

TS_IDX_JUMP R/W, TS_CONTROL[23..16], Cyton-CXP, Axion-CL

This is the entry that the table will start from when the TS_RUN_LEVEL register is set to Run.

This is the entry that the table will jump to (synchronously) when the TS_RUN_LEVEL register is set to Jump.

TS_TRIG_SEL R/W, TS_CONTROL[28..31], Cyton-CXP, Axion-CL

These bits select the source of the TS trigger.

TS_TRIG_SEL	Meaning
0 (000b)	Selected trigger (VGFx_TRIG_SEL)
1 (001b)	Selected encoder A (VFGx_ENCA_SEL)
2 (010b)	Selected encoder B (VFGx_ENCB_SEL)
3 (011b)	Selected quad encoder output (VFGx_ENCQ_SEL)
4 (100b)	Gated trigger (VGFx_TRIG_SEL gated by VFGx_ENCB_SEL)
5 (101b)	Selected encoder divider output (VFGx_ENCDIV_SEL)

3.3 TS_TABLE_CONTROL

Bit	Name
0	TS_IDX_ACCESS
1	TS_IDX_ACCESS
2	TS_IDX_ACCESS
3	TS_IDX_ACCESS
4	TS_IDX_ACCESS
5	TS_IDX_ACCESS
6	TS_IDX_ACCESS
7	TS_IDX_ACCESS
8	Reserved
9	Reserved
10	Reserved
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15	Reserved
16	Reserved
17	Reserved
18	Reserved
19	Reserved
20	Reserved
21	Reserved
22	Reserved
23	Reserved
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	Reserved
30	Reserved
31	Reserved

TS_IDX_ACCESS R/W, TS_TABLE_CONTROL[7..0], Cyton-CXP, Axion-CL

Indirect access to the TS table. Set this bitfield to the index value that you wish to modify. Access is done through via TS_TABLE_ENTRY register.

3.4 TS_TABLE_ENTRY

Bit	Name
0	TS_NEXT
1	TS_NEXT
2	TS_NEXT
3	TS_NEXT
4	TS_NEXT
5	TS_NEXT
6	TS_NEXT
7	TS_NEXT
8	Reserved
9	Reserved
10	TS_RESOLUTION
11	TS_RESOLUTION
12	TS_STATE_CT0
13	TS_STATE_CT1
14	TS_STATE_CT2
15	TS_STATE_CT3
16	Reserved
17	TS_COUNT
18	TS_COUNT
19	TS_COUNT
20	TS_COUNT
21	TS_COUNT
22	TS_COUNT
23	TS_COUNT
24	TS_COUNT
25	TS_COUNT
26	TS_COUNT
27	TS_CONDITION
28	TS_CONDITION
29	TS_CONDITION
30	TS_TERMINATE
31	TS_END_OF_SEQUENCE

TS_NEXT R/W, TS_TABLE_ENTRY[7..0], Cyton-CXP, Axion-CL

Index of next pulse. Only relevant if TS_TERMINATE = 0.

TS_RESOLUTION R/W, TS_TABLE_ENTRY[11..10], Cyton-CXP, Axion-CL

The time units of this pulse. The length of this pulse is set in the register TS_COUNT. The following table shows the available resolutions.

TS_RESOLUTION	Meaning
0 (000b)	100 nanoseconds
1 (001b)	100 microseconds
2 (010b)	100 milliseconds
3 (011b)	100 seconds

TS_STATE_CT0 R/W, TS_TABLE_ENTRY[12], Cyton-CXP, Axion-CL

The level of the CT0 signal for this pulse.

TS_STATE_CT1 R/W, TS_TABLE_ENTRY[13], Cyton-CXP, Axion-CL

The level of the CT1 signal for this pulse.

TS_STATE_CT2 R/W, TS_TABLE_ENTRY[14], Cyton-CXP, Axion-CL

The level of the CT2 signal for this pulse.

TS_STATE_CT3 R/W, TS_TABLE_ENTRY[15], Cyton-CXP, Axion-CL

The level of the CT3 signal for this pulse.

TS_COUNT R/W, TS_TABLE_ENTRY[26..17], Cyton-CXP, Axion-CL

The length of this pulse. The units for the length are set in the TS_RESOLUTION register.

TS_CONDITION R/W, TS_TABLE_ENTRY[29..27], Cyton-CXP, Axion-CL

This register is used to control the conditions under which this pulse will be output. The following table shows the options for this bitfield.

TS_CONDITION	Condition when pulse is output
0 (000b)	Immediate
1 (001b)	Rising edge of trigger
2 (010b)	Falling edge of trigger
3 (011b)	Trigger high
4 (100b)	Trigger low
5 (101b)	Both rising and falling edge of trigger

TS_TERMINATE R/W, TS_TABLE_ENTRY[30], Cyton-CXP, Axion-CL

When this bit is set to 0, the table will stop running after the current pulse is output. The TS_RUN_LEVEL bitfield will then read back idle.

When this bit is set to 1, the table will jump to the index set in the TS_NEXT bitfield after the current pulse is finished.

TS_END_OF_ SEQUENCE R/W, TS_TABLE_ENTRY[31], Cyton-CXP, Axion-CL

If this bit is set to 1 and TS_RUN_LEVEL is set to Jump, the TS will jump to the index set in the TS_INDX_JUMP bitfield after the current pulse is output. This bit allows for synchronous switching between one section of the table and another section.

If the TS_RUN_LEVEL bitfield is not set to Jump, then this bitfield will have no effect.

If this bit is set to 0, the TS will not jump from this index.

3.5 ATS_CONTROL

Bit	Name
0	ATS_RUN_LEVEL
1	ATS_RUN_LEVEL
2	ATS_RUN_LEVEL
3	Reserved
4	ATS_CT0_DEFAULT_STATE
5	ATS_CT1_DEFAULT_STATE
6	ATS_CT2_DEFAULT_STATE
7	ATS_CT3_DEFAULT_STATE
8	Reserved
9	Reserved
10	Reserved
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15	Reserved
16	ATS_IDX_JUMP
17	ATS_IDX_JUMP
18	ATS_IDX_JUMP
19	ATS_IDX_JUMP
20	ATS_IDX_JUMP
21	ATS_IDX_JUMP
22	ATS_IDX_JUMP
23	ATS_IDX_JUMP
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	ATS_TRIG_SEL
29	ATS_TRIG_SEL
30	ATS_TRIG_SEL
31	ATS_TRIG_SEL

ATS_RUN_LEVEL R/W, ATS_CONTROL[2..0], Cyton-CXP, Axion-CL, Axion-CL

These bits control the operation of the ATS. These bits are used to start and stop the sequencer. They can also be used to program the table to jump to a new section. Jumps are always synchronous (i.e. not immediate). Jumps will only occur from an index in the sequence that has the ATS_END_OF_SEQUENCE bit set.

This bit can be read at any time in order get the current status.

The following table shows the command available for this register.

ATS_RUN_LEVEL	Meaning
0 (000b)	Idle - ATS is not running
1 (001b)	Run - Start running immediately from index in the ATS_IDX_ JUMP register
2 (010b)	Jump - Jump to index set in the ATS_IDX_JUMP register next time the current index has the ATS_END_OF_SEQUENCE bit set to 1
3 (011b)	Stop - Stop running the next time the current index has the ATS_END_OF_SEQUENCE bit set to 1
4 (100b)	Abort - Stop running immediately

ATS_CT0_ DEFAULT_STATE R/W, ATS_CONTROL[4], Cyton-CXP, Axion-CL

This is the output state of CT0 when the ATS is Idle.

ATS_CT1_ DEFAULT_STATE R/W, ATS_CONTROL[5], Cyton-CXP, Axion-CL

This is the output state of CT1 when the ATS is Idle.

ATS_CT2_ DEFAULT_STATE R/W, ATS_CONTROL[6], Cyton-CXP, Axion-CL

This is the output state of CT2 when the ATS is Idle.

ATS_CT3_ DEFAULT_STATE R/W, ATS_CONTROL[7], Cyton-CXP, Axion-CL

This is the output state of CT3 when the ATS is Idle.

ATS_IDX_JUMP R/W, ATS_CONTROL[23..16], Cyton-CXP, Axion-CL

This is the entry that the table will start from when the ATS_RUN_LEVEL register is set to Run.

This is also the entry that the table will jump to (synchronously) when the ATS_RUN_LEVEL register is set to Jump.

ATS_TRIG_SEL R/W, ATS_CONTROL[28..31], Cyton-CXP, Axion-CL

These bits select the source of the ATS trigger.

ATS_TRIG_SEL	Meaning
0 (000b)	Selected trigger (VGFx_TRIG_SEL)
1 (001b)	Selected encoder A (VFGx_ENCA_SEL)
2 (010b)	Selected encoder B (VFGx_ENCB_SEL)
3 (011b)	Selected quad encoder output (VFGx_ENCQ_SEL)
4 (100b)	Gated trigger (VGFx_TRIG_SEL gated by VFGx_ENCB_SEL)
5 (101b)	Selected encoder divider output (VFGx_ENCDIV_SEL)

3.6 ATS_TABLE_CONTROL

Bit	**Name**
0	ATS_IDX_ACCESS
1	ATS_IDX_ACCESS
2	ATS_IDX_ACCESS
3	ATS_IDX_ACCESS
4	ATS_IDX_ACCESS
5	ATS_IDX_ACCESS
6	ATS_IDX_ACCESS
7	ATS_IDX_ACCESS
8	Reserved
9	Reserved
10	Reserved
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15	Reserved
16	Reserved
17	Reserved
18	Reserved
19	Reserved
20	Reserved
21	Reserved
22	Reserved
23	Reserved
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	Reserved
30	Reserved
31	Reserved

ATS_IDX_ ACCESS

R/W, ATS_TABLE_CONTROL[7..0], Cyton-CXP, Axion-CL

Indirect access to the TS table. Set this bitfield to the index value that you wish to modify. Access is done through via TS_TABLE_ENTRY register.

3.7 ATS_TABLE_ENTRY

Bit	Name
0	ATS_NEXT
1	ATS_NEXT
2	ATS_NEXT
3	ATS_NEXT
4	ATS_NEXT
5	ATS_NEXT
6	ATS_NEXT
7	ATS_NEXT
8	Reserved
9	Reserved
10	ATS_RESOLUTION
11	ATS_RESOLUTION
12	ATS_STATE_CT0
13	ATS_STATE_CT1
14	ATS_STATE_CT2
15	ATS_STATE_CT3
16	Reserved
17	ATS_COUNT
18	ATS_COUNT
19	ATS_COUNT
20	ATS_COUNT
21	ATS_COUNT
22	ATS_COUNT
23	ATS_COUNT
24	ATS_COUNT
25	ATS_COUNT
26	ATS_COUNT
27	ATS_CONDITION
28	ATS_CONDITION
29	ATS_CONDITION
30	ATS_TERMINATE
31	ATS_END_OF_SEQUENCE

ATS_NEXT R/W, ATS_TABLE_ENTRY[7..0], Cyton-CXP, Axion-CL

Index of next pulse. Only follow if ATS_TERMINATE = 0.

ATS_ RESOLUTION R/W, ATS_TABLE_ENTRY[11..10], Cyton-CXP, Axion-CL

The time units of this pulse. The length of this pulse is set in the register ATS_COUNT. The following table shows the available resolutions.

ATS_RESOLUTION	Meaning
0 (000b)	100 nanoseconds
1 (001b)	100 microseconds
2 (010b)	100 milliseconds
3 (011b)	100 seconds

ATS_STATE_CT0 R/W, ATS_TABLE_ENTRY[12], Cyton-CXP, Axion-CL

The level of the CT0 signal for this pulse.

ATS_STATE_CT1 R/W, ATS_TABLE_ENTRY[13], Cyton-CXP, Axion-CL

The level of the CT1 signal for this pulse.

ATS_STATE_CT2 R/W, ATS_TABLE_ENTRY[14], Cyton-CXP, Axion-CL

The level of the CT2 signal for this pulse.

ATS_STATE_CT3 R/W, ATS_TABLE_ENTRY[15], Cyton-CXP, Axion-CL

The level of the CT3 signal for this pulse.

ATS_COUNT R/W, ATS_TABLE_ENTRY[26..17], Cyton-CXP, Axion-CL

The length of this pulse. The units for the length are set in the ATS_RESOLUTION register.

ATS_ CONDITION

R/W, ATS_TABLE_ENTRY[29..27], Cyton-CXP, Axion-CL

This register is used to control the conditions under which this pulse will be output. The following table shows the options for this bitfield.

ATS_CONDITION	Condition when pulse is output
0 (000b)	Immediate
1 (001b)	Rising edge of trigger
2 (010b)	Falling edge of trigger
3 (011b)	Trigger high
4 (100b)	Trigger low
5 (101b)	Both rising and falling edge of trigger

ATS_ TERMINATE

R/W, ATS_TABLE_ENTRY[30], Cyton-CXP, Axion-CL

When this bit is set to 0, the table will stop running after the current pulse is output. The ATS_RUN_LEVEL bitfield will then read back idle.

When this bit is set to 1, the table will jump to the index set in the ATS_NEXT bitfield after the current pulse is finished.

ATS_END_OF_ SEQUENCE

R/W, ATS_TABLE_ENTRY[31], Cyton-CXP, Axion-CL

If this bit is set to 1 and ATS_RUN_LEVEL is set to Jump, the ATS will jump to the index set in the ATS_INDX_JUMP bitfield after the current pulse is output. This bit allows for synchronous switching between one section of the table and another section.

If the ATS_RUN_LEVEL bitfield is not set to Jump, then this bitfield will have no effect.

If this bit is set to 0, the ATS will not jump from this index.

The Cyton And Axion I/O System

Chapter 4

4.1 Introduction

The I/O system on the Cyton and Axion family of frame grabbers are based on the Karbon-CXP with some minor changes. This system provides unprecedented flexibility. The goal of this system is to handle all the I/O needs of any machine vision application connected to the real world in a wide variety of ways. The goal is flexibility and observability.

4.1.1 Concepts

The basic concept is that the outside world of a machine vision system can have a wide variety of signals, possibly using different electrical standards. The Cyton and/or Axion user can choose whichever ones best suit their needs. These inputs can then be routed to a wide range of internal destinations. Also, the board can generate its own signals of use in this system.

Once the input sources are chosen, they are routed to one of a number of internal signals. These internal signals can then be used to control a wide variety of functions. For example, a function might be to cause the board to acquire a frame.

Finally the internal signals can be routed off the board to a wide number of destinations. These can be used to control something in the outside world. For example, a signal might be routed such that it fires a strobe light or initiates the start of exposure in a camera.

The state of all of the possible inputs can be observed by software at any time by peeking the associated RD_XXX bit.

4.1.2 I/O Between Virtual Frame Grabbers

Because BitFlow's frame grabber can acquire from more than one camera, it has always been a contention between the desire to let each camera be independent, for example, each with its own trigger, and for them to be synchronized, i.e. all cameras using one trigger. The Cyton and Axion I/O system provides for the best of both worlds by fully supporting independent and synchronized triggers using the same flexibility as each individual VFG gets.

What this means is that one of the sources, the master VFG's trigger can be the trigger for all of the VFGs. Thus whatever signal is triggering VFG0, can also trigger all the other VFGs on the board. Of course, each VFG can choose to use the master VFG's trigger, or choose amongst its own sources. Further, the triggers can be synchronized while the encoders are independent.

4.2 Overview of the Cyton and Axion I/O System Routing

Figure 4-1 below shows a generic version of the I/O system routing. For each internal signal (A and B in this case) a source must be chosen. Each internal signal can be used to control one or more internal circuits or can be routed straight to an output signal. For each output signal (X & Y in this case) a source must be chosen. In addition there are internal signal generators that can be chosen as a source for an internal signal or an external signal. Even the internal signal generators can be triggered by an internal signal.

Note: Figure 4-1 is a schematic version of the actual circuit, it is greatly simplified to make it easier to understand.

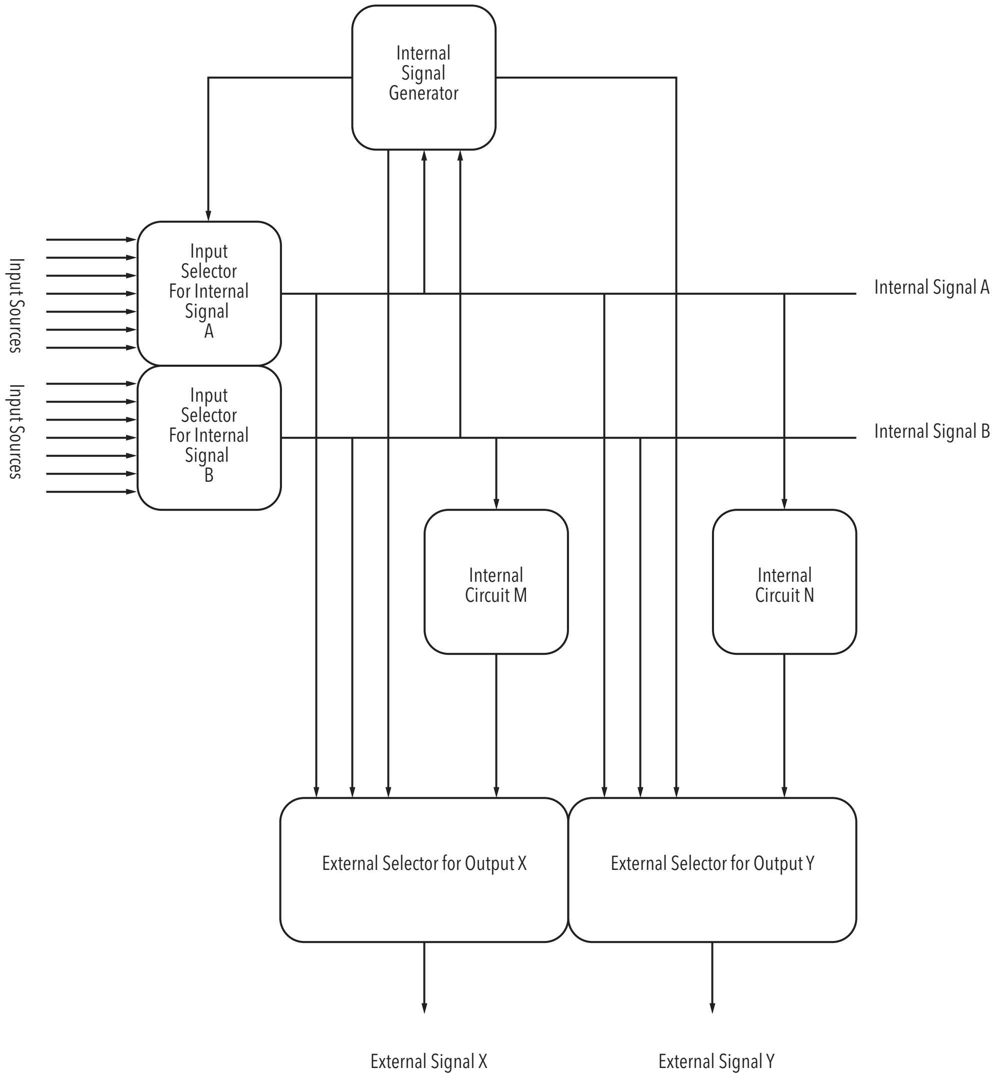
Internal Signal Generator
Input Sources
Input Sources
Input Selector For Internal Signal A
Input Selector For Internal Signal B
Internal Signal A
Internal Signal B
Internal Circuit M
Internal Circuit N
External Selector for Output X
External Selector for Output Y
External Signal X
External Signal Y

Figure 4-1 Conceptual I/O System Routing

4.3 Input Selection

Figure 4-2 illustrates the I/O System input selection circuitry. As discussed above, each internal signal has its own selector. For each internal signal there are 64 possible sources.

Note: The signals BOX_IN_XXX are available via an external I/O Box, the BitFlow BitBox, which can be mounted on an external rail system. Contact BitFlow for more information on the BitBox.

Note: Each VFG has a copy of the circuit shown in Figure 4-2. This is why the outputs do not specify the VFG number (e.g. "VFGx_TRIG_SEL). However, some inputs do specify the VFG number (e.g. VFG0_TRIG_SEL), which means this input comes explicitly from VFG0.

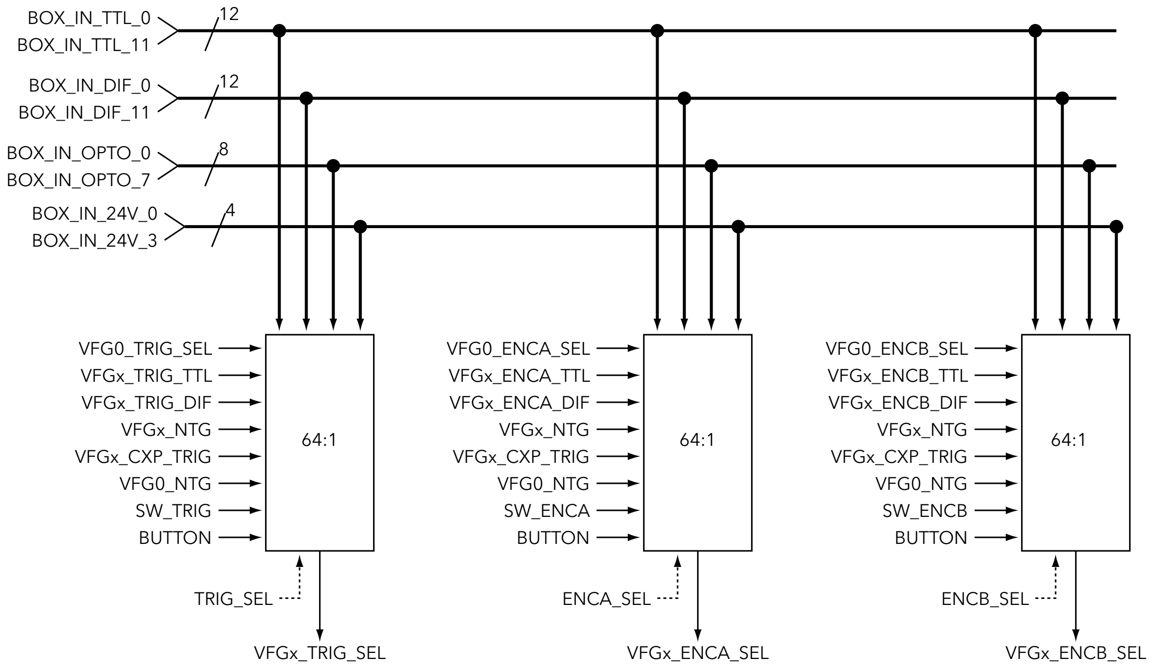

Figure 4-2 I/O System Input Selection

4.4 Internal Signals

There are five internal I/O signals. The "x" in the signal name refers to VFG number of the current VFG being used. In general, the "x" is always used as all VFGs are symmetrical. The one exceptions is VFG0, which can route many of its signals to other VFGs on the same physical board. For example VFG2 can use the trigger selected on VFG 0 (i.e. VFG0_TRIG_SEL) as its trigger source.

VFGx_TRIG_SEL - normally used as a frame trigger, can also be used to initiate acquisition of N frames, or can be use to control the start and end of a frame when used with a line scan camera.

VFGx_ENCA_SEL - Normally used as a line trigger, to initiate the capture of one line. When using a single phase encoder, this is the signal to use.

VFGx_ENCB_SEL - Normally used as a line trigger when using a quadrature encoder. The encoder B should be the second phase of the quadrature encoder with Encoder A being the first phase.

VFGx_ENCDIV_SEL - This signal is the output of the encoder divider (multiplier) circuit. This circuit can be driven by more than one source. The output signal is related to the input signal, but the frequency is either divided down or multiplied up.

VFGx_ENCQ_SEL - This signal is the output of the quadrature encoder circuit. This circuit takes to inputs, the selected encoder A and the selected encoder B. The output signal follows the rules as programmed in to the quadrature encoder circuit, see Section 8.1 for more information.

Note: The internal signals have names such as "Trigger" and "Encoder" because they are hardwired to certain internal circuits. However, if you are not using them for this functionality, they can be used for any purpose that the routing supports.

Each of the internal signals is hardwired to a number of destinations. Even though a signal might be connected to a circuit, the circuit may not be "listening" to the signal. Each circuit has its own control registers which tells it to use a given signal or not. Figure 4-3 shows all the internal circuits that each internal signal is connected to.

Figure 4-4 shows the details of the encoder handler block that is part of Figure 4-3.

The Filter is used to remove unwanted noise on the incoming signal. The filter is programmable and it will "swallow" a pulse shorter than the programmed size.

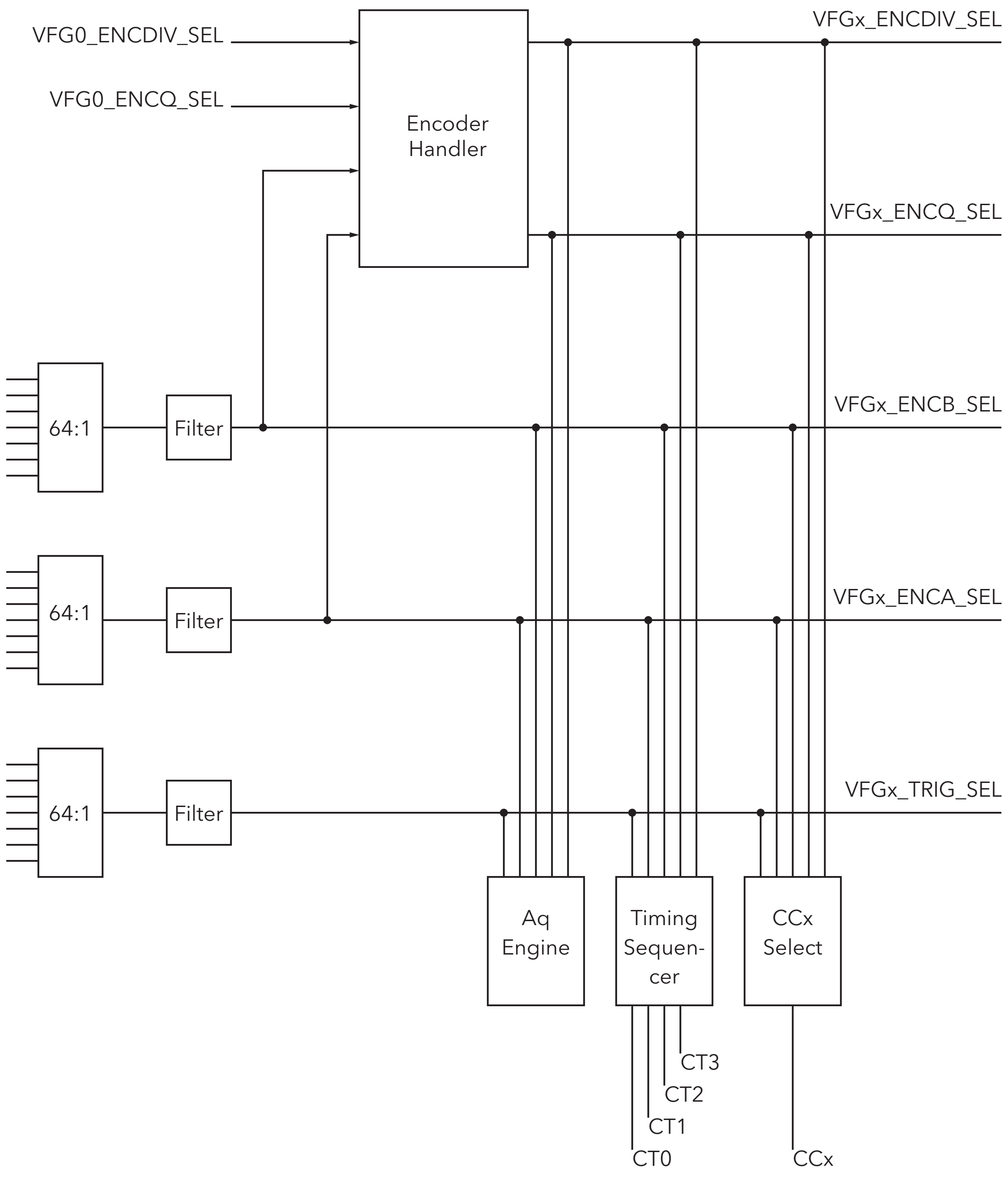

Figure 4-3 Internal Signal Routing

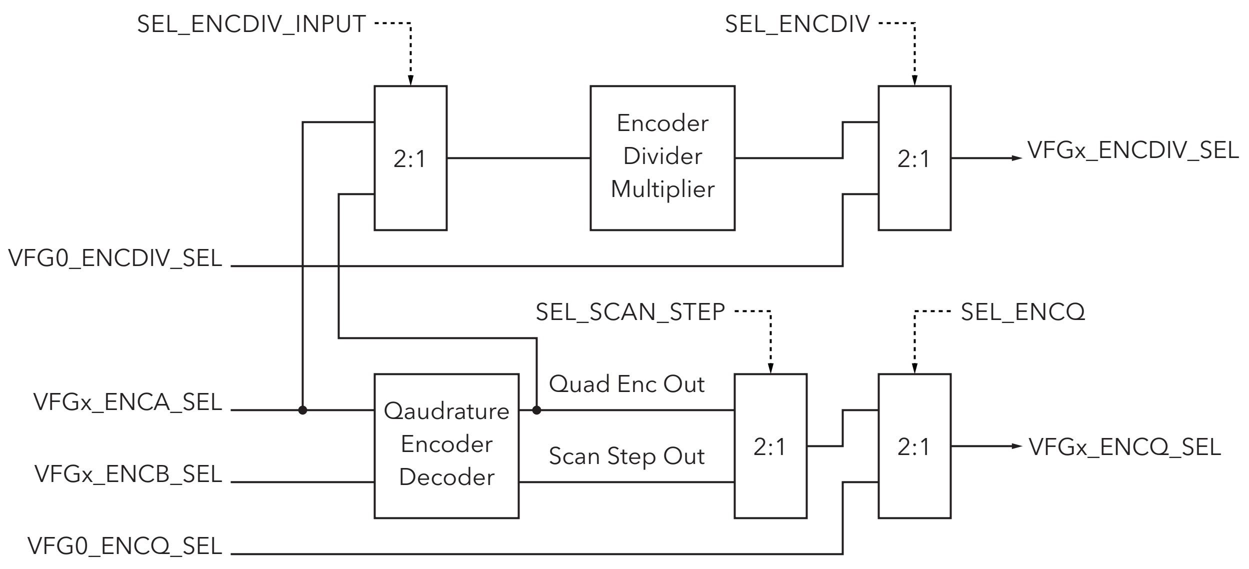

Figure 4-4 Encoder Handler

4.5 Output Signal Selection

There are four dynamic output signals for each VFG, CC1 to CC4, and thirty six static output signals,VFG0_GPOUTx_Xxx, which are only on VFG0. The dynamic signals can be driven by a variety of sources as shown in Figure 4-5. The static signals are controlled by the values in the registers with the same names (e.g. VFG0_GPOUT0_TTL is controlled by the register BOX_OUT_GPOUT0_TTL) in VFG0.

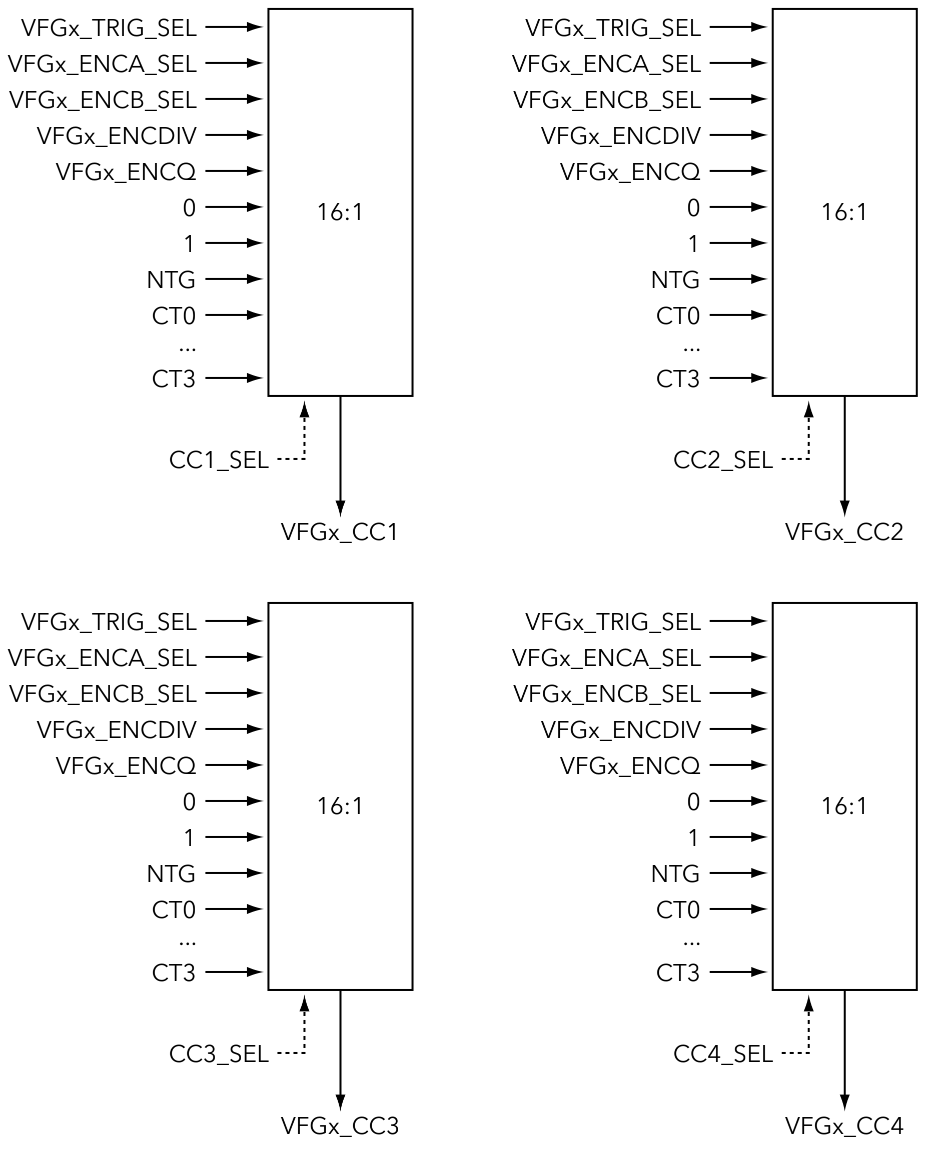

Figure 4-5 Output Signal Source Selection

4.6 I/O Connector Output Signal Routing

The CCx output signals are routed to the CXP link or the board's main I/O connector Figure 4-6 illustrates how they are routed.

Note: Each VFG has an instance of the circuit shown below. This means each VFG has its own CC2, CC3 and CC4 signals present on the board's main I/O connector.

Note: The signals VFGx_CC1 to VFGx_CC3 can be sourced from many different signals. Please see Section 4.5 for more information on selecting the source.

Note: The signals VFGx_CC1 to VFGx_CC3 can also be routed to the BitBox, which is an externally mountable I/O Box.

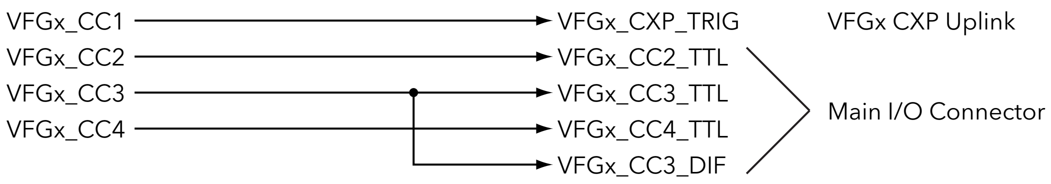

Figure 4-6 Output Signal Routing

4.7 BitBox Output Signal Routing

The BitFlow BitBox has 3 banks of 12 outputs. One bank is TTL, one bank is differential and one bank is a mix of Optocoupled and Open Collector outputs. Each of the 36 outputs can be set to a static output level, or controlled by a dynamic source (waveform). Figure 4-7 shows how the outputs are controlled.

Note: Figure 4-7 shows the choices and routing for just 3 outputs. A total of 12 of these circuits (each with their own control registers) are required on the board in order to control 36 outputs.

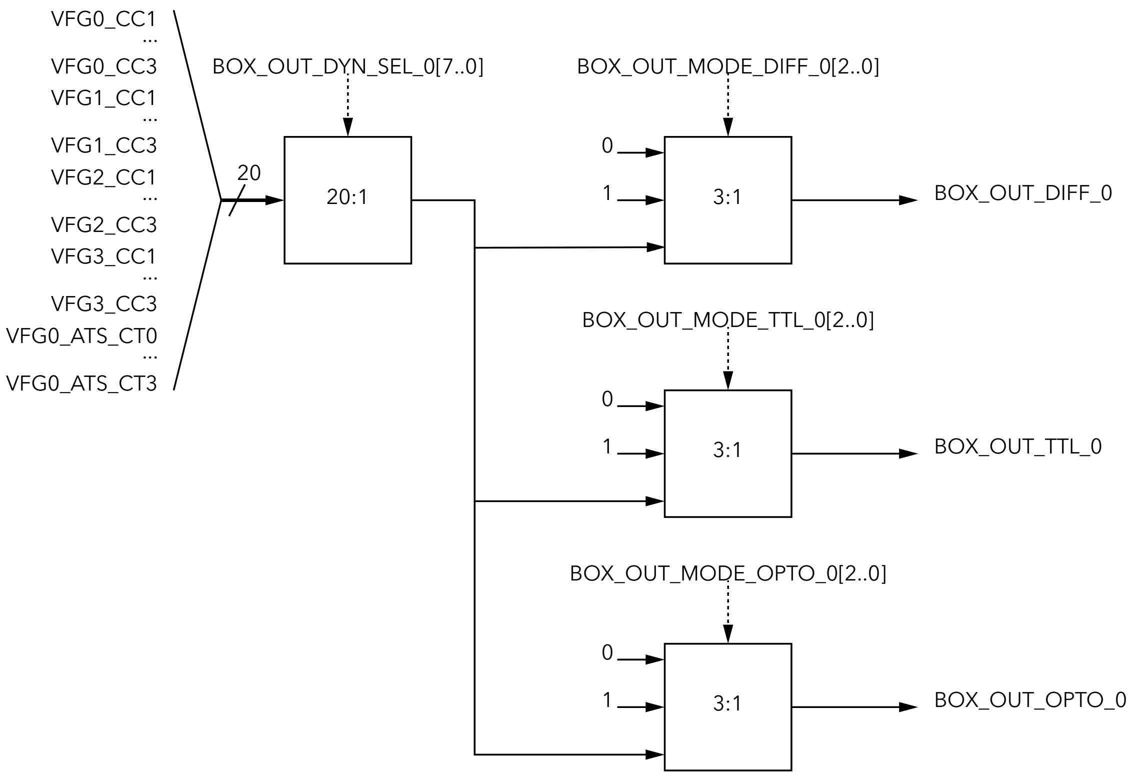

Figure 4-7 BitBox Output Signal Routing

Note: The BitFlow BitBox is an externally rail mounted box which can take a wide variaty of inputs and outputs. It is connected to the Cyton/Axion though a small cable. Contact BitFlow for more information on the BitBox.

The Cyton and Axion I/O System Registers

Chapter 5

5.1 Introduction

The registers documented in this section are used to control the I/O system on the Cyton-CXP and the Axion-CL

5.2 CON60

Bit	**Name**
0	RD_BOX_IN_TTL_0
1	RD_BOX_IN_TTL_1
2	RD_BOX_IN_TTL_2
3	RD_BOX_IN_TTL_3
4	RD_BOX_IN_TTL_4
5	RD_BOX_IN_TTL_5
6	RD_BOX_IN_TTL_6
7	RD_BOX_IN_TTL_7
8	RD_BOX_IN_TTL_8
9	RD_BOX_IN_TTL_9
10	RD_BOX_IN_TTL_10
11	RD_BOX_IN_TTL_11
12	RD_BOX_IN_DIF_0
13	RD_BOX_IN_DIF_1
14	RD_BOX_IN_DIF_2
15	RD_BOX_IN_DIF_3
16	RD_BOX_IN_DIF_4
17	RD_BOX_IN_DIF_5
18	RD_BOX_IN_DIF_6
19	RD_BOX_IN_DIF_7
20	RD_BOX_IN_DIF_8
21	RD_BOX_IN_DIF_9
22	RD_BOX_IN_DIF_10
23	RD_BOX_IN_DIF_11
24	ENINT_CXP
25	INT_CXP
26	Reserved
27	Reserved
28	Reserved
29	SW_TRIG
30	SW_ENCA
31	SW_ENCB

RD_BOX_IN_TTL_X

RO, CON60[11..0], Cyton-CXP, Axion-CL

These bits reflect the real-time state of the 12 TTL inputs on the BitBox.

RD_BOX_IN_DIF_X

RO, CON60[23..12], Cyton-CXP, Axion-CL

These bits reflect the real-time state of the 12 differential inputs on the BitBox.

ENINT_CXP

R/W, CON60[24], Cyton-CXP, Axion-CL

This bit enables interrupts from the CXP subsystem.

INT_CXP

RO, CON60[25], Cyton-CXP, Axion-CL

This bit indicates the existence of an interrupt from the CXP subsystem. The individual interrupt must be cleared in the CXP subsystem in order for this bit to reset.

SW_TRIG

R/W, CON60[29], Cyton-CXP, Axion-CL

Writing the bit to 1 causes the internal software trigger signal to be asserted. Writing it to a 0 will de-assert the internal software trigger signal.

SW_ENCA

R/W, CON60[30], Cyton-CXP, Axion-CL

Writing the bit to 1 causes the internal encoder A signal to be asserted. Writing it to a 0 will de-assert the internal encoder A signal.

SW_ENCB

R/W, CON60[31], Cyton-CXP, Axion-CL

Writing the bit to 1 causes the internal encoder B signal to be asserted. Writing it to a 0 will de-assert the internal encoder B signal.

5.3 CON61

Bit	Name
0	RD_BOX_IN_OPTO_0
1	RD_BOX_IN_OPTO_1
2	RD_BOX_IN_OPTO_2
3	RD_BOX_IN_OPTO_3
4	RD_BOX_IN_OPTO_4
5	RD_BOX_IN_OPTO_5
6	RD_BOX_IN_OPTO_6
7	RD_BOX_IN_OPTO_7
8	RD_BOX_IN_24V_0
9	RD_BOX_IN_24V_1
10	RD_BOX_IN_24V_2
11	RD_BOX_IN_24V_3
12	Reserved
13	Reserved
14	Reserved
15	Reserved
16	Reserved
17	Reserved
18	Reserved
19	Reserved
20	Reserved
21	Reserved
22	Reserved
23	RD_CXP_TRIG_OUT
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	Reserved
30	Reserved
31	Reserved

RD_BOX_IN_ OPTO_X

RO, CON61[7..0], Cyton-CXP, Axion-CL

These bits reflect the real-time state of the eight Opto-Isolated inputs on the BitBox.

RD_BOX_IN_ 24V_X

RO, CON61[11..8], Cyton-CXP, Axion-CL

These bits reflect the real-time state of the four 24V inputs on the BitBox.

RD_CXP_TRIG_ OUT

RO, CON61[23], Cyton-CXP, Axion-CL

This bit reflects the real-time state of the CXP trigger signal going to the camera.

5.4 CON62

Bit	Name
0	RD_TRIG_TTL
1	RD_TRIG_DIF
2	RD_TRIG_VFG0
3	RD_SCAN_STEP
4	RD_SW_TRIG
5	RD_ENCA_TTL
6	RD_ENCA_DIF
7	RD_ENCA_VFG0
8	RD_ENCA_SW
9	RD_ENCB_TTL
10	RD_ENCB_DIF
11	RD_ENCB_VFG0
12	RD_ENCB_SW
13	RD_BUTTON
14	Reserved
15	Reserved
16	Reserved
17	Reserved
18	Reserved
19	Reserved
20	Reserved
21	Reserved
22	Reserved
23	Reserved
24	RD_CXP_TRIG_IN
25	EN_TRIG
26	EN_ENCA
27	EN_ENCB
28	Reserved
29	RD_ENCB_SELECTED
30	RD_ENCA_SELECTED
31	RD_TRIG_SELECTED

RD_TRIG_TTL	RO, CON62[0], Cyton-CXP, Axion-CL
	This bit reflects the real-time state of the VFG's TTL trigger input.
RD_TRIG_DIF	RO, CON62[1], Cyton-CXP, Axion-CL
	This bit reflects the real-time state of the VFG's differential trigger input.
RD_TRIG_VFG0	RO, CON62[2], Cyton-CXP, Axion-CL
	This bit reflects the real-time state of VFG0's selected trigger signal.
RD_SCAN_STEP	RO, CON62[3], Cyton-CXP, Axion-CL
	This bit reflects the real-time state of the VFG's scan step circuitry output (from the quadrature encoder circuit).
RD_SW_TRIG	RO, CON62[4], Cyton-CXP, Axion-CL
	This bit reflects the real-time state of the VFG's software trigger.
RD_ENCA_TTL	RO, CON62[5], Cyton-CXP, Axion-CL
	This bit reflects the real-time state of the VFG's TTL encoder A input.
RD_ENCA_DIF	RO, CON62[6], Cyton-CXP, Axion-CL
	This bit reflects the real-time state of the VFG's differential encoder A input.
RD_ENCA_VFG0	RO, CON62[7], Cyton-CXP, Axion-CL
	This bit reflects the real-time state of VFG0's selected encoder A signal.
RD_ENCA_SW	RO, CON62[8], Cyton-CXP, Axion-CL
	This bit reflects the real-time state of the VFG's software encoder A.
RD_ENCB_TTL	RO, CON62[9], Cyton-CXP, Axion-CL
	This bit reflects the real-time state of the VFG's TTL encoder B input.

RD_ENCB_DIF RO, CON62[10], Cyton-CXP, Axion-CL

This bit reflects the real-time state of the VFG's differential encoder B input.

RD_ENCB_VFG0 RO, CON62[11], Cyton-CXP, Axion-CL

This bit reflects the real-time state of VFG0's selected encoder B signal.

RD_ENCB_SW RO, CON62[12], Cyton-CXP, Axion-CL

This bit reflects the real-time state of the VFG's software encoder B.

RD_BUTTON RO, CON62[13], Cyton-CXP, Axion-CL

This bit reflects the real-time state of the VFG's button input.

RD_CXP_TRIG_IN RO, CON62[24], Cyton-CXP, Axion-CL

This bit reflects the real-time state of the CXP trigger signal coming from the camera.

EN_TRIG R/W, CON62[25], Cyton-CXP, Axion-CL

This bit is used to enable the selected trigger.

EN_ENCA R/W, CON62[26], Cyton-CXP, Axion-CL

This bit is used to enable the selected encoder A.

EN_ENCB R/W, CON62[27], Cyton-CXP, Axion-CL

This bit is used to enable the selected encoder B.

RD_ENCB_SELECTED RO, CON62[29], Cyton-CXP, Axion-CL

The bit reflects the real-time status of the VFG's select encoder B input.

RD_ENCA_SELECTED RO, CON62[30], Cyton-CXP, Axion-CL

The bit reflects the real-time status of the VFG's selected encoder A input.

RD_TRIG_ SELECTED

RO, CON62[31], Cyton-CXP, Axion-CL

The bit reflects the real-time status of the VFG's selected trigger input.

5.5 CON63

Bit	Name
0	SEL_TRIG
1	SEL_TRIG
2	SEL_TRIG
3	SEL_TRIG
4	SEL_TRIG
5	SEL_TRIG
6	SEL_ENCA
7	SEL_ENCA
8	SEL_ENCA
9	SEL_ENCA
10	SEL_ENCA
11	SEL_ENCA
12	SEL_ENCB
13	SEL_ENCB
14	SEL_ENCB
15	SEL_ENCB
16	SEL_ENCB
17	SEL_ENCB
18	SEL_CC1
19	SEL_CC1
20	SEL_CC1
21	SEL_CC1
22	SEL_CC2
23	SEL_CC2
24	SEL_CC2
25	SEL_CC2
26	Reserved
27	Reserved
28	SEL_LED
29	SEL_LED
30	SEL_LED
31	SEL_LED

SEL_TRIG R/W, CON63[5..0], Cyton-CXP, Axion-CL

Selects the VFG's trigger source.

SEL_TRIG	Source
0 (000000b)	Forced low
1 (000001b)	Forced high
2 (000010b)	This VFG's differential trigger VFGx_TRIGGER+/-
3 (000011b)	This VFG's TTL trigger VFGx_TRIGGER_TTL
4 (000100b)	Selected trigger from VFG0
5 (000101b)	This VFG's Timing Sequencer (TS)
6 (000110b)	Button
7 (000111b)	The camera's CXP trigger
8 (001000b)	This VFG's software trigger, SW_TRIG
9 (001001b)	This VFG's scan step circuit
10 (001010b)	VFG0's Timing Sequencer (TS)
11-27	Reserved
28 to 39	BOX_IN_TTL_0 to BOX_IN_TTL_11
40 to 51	BOX_IN_DIF_0 to BOX_IN_DIF_11
52 to 59	BOX_IN_OPTO_0 to BOX_IN_OPTO_7
61 to 63	BOX_IN_24V_0 to BOX_IN_24V_3

SEL_ENCA R/W, CON63[11..6], Cyton-CXP, Axion-CL

Selects this VFG's encoder A source.

SEL_ENCA	Source
0 (000000b)	Forced low
1 (000001b)	Forced high
2 (000010b)	This VFG's differential encoder A, VFGx_ENCA+/-
3 (000011b)	This VFG's TTL encoder A, VFGx_ENCA_TTL
4 (000100b)	Selected encoder A from VFG0
5 (000101b)	This VFG's Timing Sequencer (TS)
6 (000110b)	Button
7 (000111b)	The camera's CXP trigger

SEL_ENCA	Source
8 (001000b)	This VFG's software encoder A, SW_ENCA
9 (001001b)	VFG0's Timing Sequencer (TSS
10-27	Reserved
28 to 39	BOX_IN_TTL_0 to BOX_IN_TTL_11
40 to 51	BOX_IN_DIF_0 to BOX_IN_DIF_11
52 to 59	BOX_IN_OPTO_0 to BOX_IN_OPTO_7
61 to 63	BOX_IN_24V_0 to BOX_IN_24V_3

SEL_ENCB R/W, CON63[17..12], Cyton-CXP, Axion-CL

Selects the source of encoder B.

SEL_ENCB	Source
0 (000000b)	Forced low
1 (000001b)	Forced high
2 (000010b)	This VFG's differential encoder B VFGx_ENCB+/-
3 (000011b)	This VFG's TTL encoder B VFGx_ENCB_TTL
4 (000100b)	Selected encoder B from VFG0
5 (000101b)	This VFG's Timing Sequencer (TS)
6 (000110b)	Button
7 (000111b)	The camera's CXP trigger
8 (001000b)	This VFG's software encoder B, SW_ENCB
9 (001001b)	VFG0's Timing Sequencer (TS)
10-27	Reserved
28 to 39	BOX_IN_TTL_0 to BOX_IN_TTL_11
40 to 51	BOX_IN_DIF_0 to BOX_IN_DIF_11
52 to 59	BOX_IN_OPTO_0 to BOX_IN_OPTO_7
61 to 63	BOX_IN_24V_0 to BOX_IN_24V_3

SEL_CC1 R/W, CON63[21..18], Cyton-CXP, Axion-CL

Selects the source of CC1.

SEL_CC1	Source
0 (0000b)	Forced low
1 (0001b)	Forced high
2 (0010b)	CT0 (from TS)
3 (0011b)	CT1 (from TS)
4 (0100b)	CT2 (from TS)
5 (0101b)	CT3 (from TS)
6 (0110b)	VFGx_TRIG_SEL
7 (0111b)	VFGx_ENCA_SEL
8 (1000b)	VFGx_ENCB_SEL
9 (1001b)	VFG0_CT0
10 (1010b)	VFG0_CT1
11 (1011b)	VFG0_CT2
12 (1100b)	VFG0_CT3
13 (1101b)	VFGx_ENCDIV_SEL
14 (1110b)	VFGx_ENCQ_SEL

SEL_CC2 R/W, CON63[25..22], Cyton-CXP, Axion-CL

Selects the source of CC2.

SEL_CC1	Source
0 (0000b)	Forced low
1 (0001b)	Forced high
2 (0010b)	CT0 (from TS)
3 (0011b)	CT1 (from TS)
4 (0100b)	CT2 (from S)
5 (0101b)	CT3 (from TS)
6 (0110b)	VFGx_TRIG_SEL
7 (0111b)	VFGx_ENCA_SEL
8 (1000b)	VFGx_ENCB_SEL

SEL_CC1	Source
9 (1001b)	VFG0_CT0
10 (1010b)	VFG0_CT1
11 (1011b)	VFG0_CT2
12 (0101b)	VFG0_CT3
14 (1110b)	VFGx_ENCDIV_SEL
15 (1111b)	VFGx_ENCQ_SEL

SEL_LED R/W, CON63[31..28], Cyton-CXP, Axion-CL

Selects the source of the VFG's green LED. The LED receives a 1/2 second pulse every time the selected event occurs.

SEL_CC1	Source
0 (0000b)	Board emits an interrupt to the host
1 (0001b)	VFGx_TRIG_SEL
2 (0010b)	VFG0_TRIG_SEL
3 (0011b)	Button
4 (0100b)	FVAL from camera
5 (0101b)	VAW
6 (0110b)	VWIN
7 (0111b)	CC1
8 (1000b)	CC2
9 (1001b)	CC3
10 (1010b)	CC4
11 (1011b)	VFGx_NTG or VFGx_TS
12 (1100b)	VFG0_NTG or VFG0_TS
13 (1101b)	AQSTAT[1]
14 (1110b)	Overstep, OVS
15 (1111b)	Reserved

5.6 CON64

Bit	Name
0	SEL_CC3
1	SEL_CC3
2	SEL_CC3
3	SEL_CC3
4	SEL_CC4
5	SEL_CC4
6	SEL_CC4
7	SEL_CC4
8	Reserved
9	Reserved
10	Reserved
11	Reserved
12	Reserved
13	TRIGPOL
14	ENCA_POL
15	ENCB_POL
16	Reserved
17	Reserved
18	Reserved
19	Reserved
20	Reserved
21	Reserved
22	Reserved
23	Reserved
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	LED_RED
29	LED_ORANGE
30	LED_GREEN
31	LED_BLUE

SEL_CC3 R/W, CON64[3..0], Cyton-CXP, Axion-CL

Selects the source of CC3.

SEL_CC1	Source
0 (0000b)	Forced low
1 (0001b)	Forced high
2 (0010b)	CT0 (from TS)
3 (0011b)	CT1 (from TS)
4 (0100b)	CT2 (from TS)
5 (0101b)	CT3 (from TS)
6 (0110b)	VFGx_TRIG_SEL
7 (0111b)	VFGx_ENCA_SEL
8 (1000b)	VFGx_ENCB_SEL
9 (1001b)	VFG0_CT0
10 (1010b)	VFG0_CT1
11 (1011b)	VFG0_CT2
12 (0101b)	VFG0_CT3
14 (1110b)	VFGx_ENCDIV_SEL
15 (1111b)	VFGx_ENCQ_SEL

SEL_CC4 R/W, CON64[7..4], Cyton-CXP, Axion-CL

Selects the source of CC4.

SEL_CC1	Source
0 (0000b)	Forced low
1 (0001b)	Forced high
2 (0010b)	CT0 (from TS)
3 (0011b)	CT1 (from TS)
4 (0100b)	CT2 (from TS)
5 (0101b)	CT3 (from TS)
6 (0110b)	VFGx_TRIG_SEL
7 (0111b)	VFGx_ENCA_SEL
8 (1000b)	VFGx_ENCB_SEL

SEL_CC1	Source
9 (1001b)	VFG0_CT0
10 (1010b)	VFG0_CT1
11 (1011b)	VFG0_CT2
12 (0101b)	VFG0_CT3
14 (1110b)	VFGx_ENCDIV_SEL
15 (1111b)	VFGx_ENCQ_SEL

TRIGPOL R/W, CON64[13], Cyton-CXP, Axion-CL

Selects the edge of the trigger signal that corresponds to its assertion.

TRIGPOL	Meaning
0	Trigger asserted on rising edge
1	Trigger asserted on falling edge

ENCA_POL R/W, CON64[14], Cyton-CXP, Axion-CL

Selects the edge of encoder A signal that corresponds to its assertion.

ENCA_POL	Meaning
0	Encoder A asserted on rising edge
1	Encoder A asserted on falling edge

ENCB_POL R/W, CON64[15], Cyton-CXP, Axion-CL

Selects the edge of encoder B signal that corresponds to its assertion.

ENCB_POL	Meaning
0	Encoder B asserted on rising edge
1	Encoder B asserted on falling edge

LED_RED R/W, CON64[28], Cyton-CXP, Axion-CL

Setting this bit to 1 turns the red LED on. Setting this bit to 1 on any VFG turns the LED on. This bit must be set to 0 on all VFGs in order to turn this LED off.

LED_ORANGE R/W, CON64[29], Cyton-CXP, Axion-CL

Setting this bit to 1 turns the orange LED on. Setting this bit to 1 on any VFG turns the LED on. This bit must be set to 0 on all VFGs in order to turn this LED off.

LED_GREEN R/W, CON64[30], Cyton-CXP, Axion-CL

Setting this bit to 1 turns the green LED on. Setting this bit to 1 on any VFG turns the LED on. This bit must be set to 0 on all VFGs in order to turn this LED off.

5.7 ADDR_TRIG_FILTER

Bit	Name
0	TRIG_FILTER
1	TRIG_FILTER
2	TRIG_FILTER
3	TRIG_FILTER
4	TRIG_FILTER
5	TRIG_FILTER
6	TRIG_FILTER
7	TRIG_FILTER
8	TRIG_FILTER
9	TRIG_FILTER
10	TRIG_FILTER
11	TRIG_FILTER
12	TRIG_FILTER
13	TRIG_FILTER
14	TRIG_FILTER
15	TRIG_FILTER
16	TRIG_FILTER
17	TRIG_FILTER
18	Reserved
19	Reserved
20	Reserved
21	Reserved
22	Reserved
23	Reserved
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	Reserved
30	Reserved
31	Reserved

TRIG_FILTER RO, ADDR_TRIG_FILTER[17..0], Cyton-CXP, Axion-CL

The trigger circuit includes a programmable noise filter. The value of this register controls the size of the noise pulse that will be considered noised and will be filtered out. Any pulses over this size will be consider signal. The units of this register are 4 nanoseconds. If this register is programmed to 0, nothing will be filtered out. If this register is programmed to 0x3ffff, pulses of up to 1 millisecond will be removed.

5.8 ADDR_ENCA_FILTER

Bit	Name
0	ENCA_FILTER
1	ENCA_FILTER
2	ENCA_FILTER
3	ENCA_FILTER
4	ENCA_FILTER
5	ENCA_FILTER
6	ENCA_FILTER
7	ENCA_FILTER
8	ENCA_FILTER
9	ENCA_FILTER
10	ENCA_FILTER
11	ENCA_FILTER
12	ENCA_FILTER
13	ENCA_FILTER
14	ENCA_FILTER
15	ENCA_FILTER
16	ENCA_FILTER
17	ENCA_FILTER
18	Reserved
19	Reserved
20	Reserved
21	Reserved
22	Reserved
23	Reserved
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	Reserved
30	Reserved
31	Reserved

ENCA_FILTER RO, ADDR_ENCA_FILTER[17..0], Cyton-CXP, Axion-CL

The encoder A circuit includes a programmable noise filter. The value of this register controls the size of the noise pulse that will be considered noised and will be filtered out. Any pulses over this size will be consider signal. The units of this register are 4 nanoseconds. If this register is programmed to 0, nothing will be filtered out. If this register is programmed to 0x3ffff, pulses of up to 1 millisecond will be removed.

5.9 ADDR_ENCB_FILTER

Bit	Name
0	ENCB_FILTER
1	ENCB_FILTER
2	ENCB_FILTER
3	ENCB_FILTER
4	ENCB_FILTER
5	ENCB_FILTER
6	ENCB_FILTER
7	ENCB_FILTER
8	ENCB_FILTER
9	ENCB_FILTER
10	ENCB_FILTER
11	ENCB_FILTER
12	ENCB_FILTER
13	ENCB_FILTER
14	ENCB_FILTER
15	ENCB_FILTER
16	ENCB_FILTER
17	ENCB_FILTER
18	Reserved
19	Reserved
20	Reserved
21	Reserved
22	Reserved
23	Reserved
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	Reserved
30	Reserved
31	Reserved

ENCB_FILTER RO, ADDR_ENCB_FILTER[17..0], Cyton-CXP, Axion-CL

The encoder B circuit includes a programmable noise filter. The value of this register controls the size of the noise pulse that will be considered noised and will be filtered out. Any pulses over this size will be consider signal. The units of this register are 4 nanoseconds. If this register is programmed to 0, nothing will be filtered out. If this register is programmed to 0x3ffff, pulses of up to 1 millisecond will be removed.

5.10 BOX_OUT_DYN_SEL_SET_A

Bit	Name
0	BOX_OUT_DYN_SEL_0
1	BOX_OUT_DYN_SEL_0
2	BOX_OUT_DYN_SEL_0
3	BOX_OUT_DYN_SEL_0
4	BOX_OUT_DYN_SEL_0
5	BOX_OUT_DYN_SEL_0
6	BOX_OUT_DYN_SEL_0
7	BOX_OUT_DYN_SEL_0
8	BOX_OUT_DYN_SEL_1
9	BOX_OUT_DYN_SEL_1
10	BOX_OUT_DYN_SEL_1
11	BOX_OUT_DYN_SEL_1
12	BOX_OUT_DYN_SEL_1
13	BOX_OUT_DYN_SEL_1
14	BOX_OUT_DYN_SEL_1
15	BOX_OUT_DYN_SEL_1
16	BOX_OUT_DYN_SEL_2
17	BOX_OUT_DYN_SEL_2
18	BOX_OUT_DYN_SEL_2
19	BOX_OUT_DYN_SEL_2
20	BOX_OUT_DYN_SEL_2
21	BOX_OUT_DYN_SEL_2
22	BOX_OUT_DYN_SEL_2
23	BOX_OUT_DYN_SEL_2
24	BOX_OUT_DYN_SEL_3
25	BOX_OUT_DYN_SEL_3
26	BOX_OUT_DYN_SEL_3
27	BOX_OUT_DYN_SEL_3
28	BOX_OUT_DYN_SEL_3
29	BOX_OUT_DYN_SEL_3
30	BOX_OUT_DYN_SEL_3
31	BOX_OUT_DYN_SEL_3

BOX_OUT_DYN_SEL_0 R/W, BOX_OUT_DYN_SEL_SET_A[7..0], Cyton-CXP, Axion-CL

This register controls the dynamic source for the three BitBox output signals BOX_OUT_DIFF_0, BOX_OUT_TTL_0 and BOX_OUT_OPTO_0. These three outputs always have the same dynamic source (they can individually be set to static high or low value). The following table lists all the possible sources.

Table 5-1 BitBox Dynamic Output Source Selections

BOX_OUT_DYN_SEL_0	Source
0 (00000b)	VFG0_CC1
1 (00001b)	VFG0_CC2
2 (00010b)	VFG0_CC3
3 (00011b)	VFG0_CC4
4 (00100b)	VFG1_CC1
5 (00101b)	VFG1_CC2
6 (00110b)	VFG1_CC3
7 (00111b)	VFG1_CC4
8 (01000b)	VFG2_CC1
9 (01001b)	VFG2_CC2
10 (01010b)	VFG2_CC3
11 (01011b)	VFG2_CC4
12 (01100b)	VFG3_CC1
13 (01101b)	VFG3_CC2
14 (01110b)	VFG3_CC3
15 (01111b)	VFG3_CC4
16 (10000b)	VFG0_ATS_CT0
17 (10001b)	VFG0_ATS_CT1
18 (10010b)	VFG0_ATS_CT2
19 (10011b)	VFG0_ATS_CT3
20 to 256	Reserved

Note: Sources that start with VFG1, VFG2 and VFG3 are not available on the Axion-1xE. Sources that start with VFG3 are not available on the Cyton-CXP2 or the Axion-2xE.

BOX_OUT_DYN_SEL_1

R/W, BOX_OUT_DYN_SEL_SET_A[15..8], Cyton-CXP, Axion-CL

This register controls the dynamic source for the three BitBox output signals BOX_OUT_DIFF_1, BOX_OUT_TTL_1 and BOX_OUT_OPTO_1. These three outputs always have the same dynamic source (they can individually be set to static high or low value). Table 5-1 lists all the possible sources.

BOX_OUT_DYN_SEL_2

R/W, BOX_OUT_DYN_SEL_SET_A[23..16], Cyton-CXP, Axion-CL

This register controls the dynamic source for the three BitBox output signals BOX_OUT_DIFF_2, BOX_OUT_TTL_2 and BOX_OUT_OPTO_2. These three outputs always have the same dynamic source (they can individually be set to static high or low value). Table 5-1 lists all the possible sources.

BOX_OUT_DYN_SEL_3

R/W, BOX_OUT_DYN_SEL_SET_A[31..24], Cyton-CXP, Axion-CL

This register controls the dynamic source for the three BitBox output signals BOX_OUT_DIFF_3, BOX_OUT_TTL_3 and BOX_OUT_OPTO_3. These three outputs always have the same dynamic source (they can individually be set to static high or low value). Table 5-1 lists all the possible sources.

5.11 BOX_OUT_DYN_SEL_SET_B

Bit	Name
0	BOX_OUT_DYN_SEL_4
1	BOX_OUT_DYN_SEL_4
2	BOX_OUT_DYN_SEL_4
3	BOX_OUT_DYN_SEL_4
4	BOX_OUT_DYN_SEL_4
5	BOX_OUT_DYN_SEL_4
6	BOX_OUT_DYN_SEL_4
7	BOX_OUT_DYN_SEL_4
8	BOX_OUT_DYN_SEL_5
9	BOX_OUT_DYN_SEL_5
10	BOX_OUT_DYN_SEL_5
11	BOX_OUT_DYN_SEL_5
12	BOX_OUT_DYN_SEL_5
13	BOX_OUT_DYN_SEL_5
14	BOX_OUT_DYN_SEL_5
15	BOX_OUT_DYN_SEL_5
16	BOX_OUT_DYN_SEL_6
17	BOX_OUT_DYN_SEL_6
18	BOX_OUT_DYN_SEL_6
19	BOX_OUT_DYN_SEL_6
20	BOX_OUT_DYN_SEL_6
21	BOX_OUT_DYN_SEL_6
22	BOX_OUT_DYN_SEL_6
23	BOX_OUT_DYN_SEL_6
24	BOX_OUT_DYN_SEL_7
25	BOX_OUT_DYN_SEL_7
26	BOX_OUT_DYN_SEL_7
27	BOX_OUT_DYN_SEL_7
28	BOX_OUT_DYN_SEL_7
29	BOX_OUT_DYN_SEL_7
30	BOX_OUT_DYN_SEL_7
31	BOX_OUT_DYN_SEL_7

BOX_OUT_DYN_ SEL_4

R/W, BOX_OUT_DYN_SEL_SET_B[7..0], Cyton-CXP, Axion-CL

This register controls the dynamic source for the three BitBox output signals BOX_ OUT_DIFF_4, BOX_OUT_TTL_4 and BOX_OUT_OPTO_4. These three outputs always have the same dynamic source (they can individually be set to static high or low value). Table 5-1 lists all the possible sources.

BOX_OUT_DYN_ SEL_5

R/W, BOX_OUT_DYN_SEL_SET_B[15..8], Cyton-CXP, Axion-CL

This register controls the dynamic source for the three BitBox output signals BOX_ OUT_DIFF_5, BOX_OUT_TTL_5 and BOX_OUT_OPTO_5. These three outputs always have the same dynamic source (they can individually be set to static high or low value). Table 5-1 lists all the possible sources.

BOX_OUT_DYN_ SEL_6

R/W, BOX_OUT_DYN_SEL_SET_B[23..16], Cyton-CXP, Axion-CL

This register controls the dynamic source for the three BitBox output signals BOX_ OUT_DIFF_6, BOX_OUT_TTL_6 and BOX_OUT_OPTO_6. These three outputs always have the same dynamic source (they can individually be set to static high or low value). Table 5-1 lists all the possible sources.

BOX_OUT_DYN_ SEL_7

R/W, BOX_OUT_DYN_SEL_SET_B[31..24], Cyton-CXP, Axion-CL

This register controls the dynamic source for the three BitBox output signals BOX_ OUT_DIFF_7, BOX_OUT_TTL_7 and BOX_OUT_OPTO_7. These three outputs always have the same dynamic source (they can individually be set to static high or low value). Table 5-1 lists all the possible sources.

5.12 BOX_OUT_DYN_SEL_SET_C

Bit	Name
0	BOX_OUT_DYN_SEL_8
1	BOX_OUT_DYN_SEL_8
2	BOX_OUT_DYN_SEL_8
3	BOX_OUT_DYN_SEL_8
4	BOX_OUT_DYN_SEL_8
5	BOX_OUT_DYN_SEL_8
6	BOX_OUT_DYN_SEL_8
7	BOX_OUT_DYN_SEL_8
8	BOX_OUT_DYN_SEL_9
9	BOX_OUT_DYN_SEL_9
10	BOX_OUT_DYN_SEL_9
11	BOX_OUT_DYN_SEL_9
12	BOX_OUT_DYN_SEL_9
13	BOX_OUT_DYN_SEL_9
14	BOX_OUT_DYN_SEL_9
15	BOX_OUT_DYN_SEL_9
16	BOX_OUT_DYN_SEL_10
17	BOX_OUT_DYN_SEL_10
18	BOX_OUT_DYN_SEL_10
19	BOX_OUT_DYN_SEL_10
20	BOX_OUT_DYN_SEL_10
21	BOX_OUT_DYN_SEL_10
22	BOX_OUT_DYN_SEL_10
23	BOX_OUT_DYN_SEL_10
24	BOX_OUT_DYN_SEL_11
25	BOX_OUT_DYN_SEL_11
26	BOX_OUT_DYN_SEL_11
27	BOX_OUT_DYN_SEL_11
28	BOX_OUT_DYN_SEL_11
29	BOX_OUT_DYN_SEL_11
30	BOX_OUT_DYN_SEL_11
31	BOX_OUT_DYN_SEL_11

BOX_OUT_DYN_ SEL_8

R/W, BOX_OUT_DYN_SEL_SET_C[7..0], Cyton-CXP, Axion-CL

This register controls the dynamic source for the three BitBox output signals BOX_ OUT_DIFF_8, BOX_OUT_TTL_8 and BOX_OUT_OC_0. These three outputs always have the same dynamic source (they can individually be set to static high or low value). Table 5-1 lists all the possible sources.

BOX_OUT_DYN_ SEL_9

R/W, BOX_OUT_DYN_SEL_SET_C[15..8], Cyton-CXP, Axion-CL

This register controls the dynamic source for the three BitBox output signals BOX_ OUT_DIFF_9, BOX_OUT_TTL_9 and BOX_OUT_OC_1. These three outputs always have the same dynamic source (they can individually be set to static high or low value). Table 5-1 lists all the possible sources.

BOX_OUT_DYN_ SEL_10

R/W, BOX_OUT_DYN_SEL_SET_C[23..16], Cyton-CXP, Axion-CL

This register controls the dynamic source for the three BitBox output signals BOX_ OUT_DIFF_10, BOX_OUT_TTL_10 and BOX_OC_2. These three outputs always have the same dynamic source (they can individually be set to static high or low value). Table 5-1 lists all the possible sources.

BOX_OUT_DYN_ SEL_11

R/W, BOX_OUT_DYN_SEL_SET_C[31..24], Cyton-CXP, Axion-CL

This register controls the dynamic source for the three BitBox output signals BOX_ OUT_DIFF_11, BOX_OUT_TTL_11 and BOX_OUT_OC_3. These three outputs always have the same dynamic source (they can individually be set to static high or low value). Table 5-1 lists all the possible sources.

5.13 BOX_OUT_MODE_SET_A

Bit	Name
0	BOX_OUT_MODE_TTL_0
1	BOX_OUT_MODE_TTL_0
2	BOX_OUT_MODE_TTL_1
3	BOX_OUT_MODE_TTL_1
4	BOX_OUT_MODE_TTL_2
5	BOX_OUT_MODE_TTL_2
6	BOX_OUT_MODE_TTL_3
7	BOX_OUT_MODE_TTL_3
8	BOX_OUT_MODE_TTL_4
9	BOX_OUT_MODE_TTL_4
10	BOX_OUT_MODE_TTL_5
11	BOX_OUT_MODE_TTL_5
12	BOX_OUT_MODE_TTL_6
13	BOX_OUT_MODE_TTL_6
14	BOX_OUT_MODE_TTL_7
15	BOX_OUT_MODE_TTL_7
16	BOX_OUT_MODE_TTL_8
17	BOX_OUT_MODE_TTL_8
18	BOX_OUT_MODE_TTL_9
19	BOX_OUT_MODE_TTL_9
20	BOX_OUT_MODE_TTL_10
21	BOX_OUT_MODE_TTL_10
22	BOX_OUT_MODE_TTL_11
23	BOX_OUT_MODE_TTL_11
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	Reserved
30	Reserved
31	Reserved

BOX_OUT_ MODE_TTL_0

R/W, BOX_OUT_MODE_SET_A[1..0], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_TTL_0. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_0 bitfield.

BOX_OUT_MODE_TTL_0	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_TTL_1

R/W, BOX_OUT_MODE_SET_A[3..2], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_TTL_1. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_1 bitfield.

BOX_OUT_MODE_TTL_1	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_TTL_2

R/W, BOX_OUT_MODE_SET_A[5..4], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_TTL_2. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_2 bitfield.

BOX_OUT_MODE_TTL_2	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_TTL_3

R/W, BOX_OUT_MODE_SET_A[7..6], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_TTL_3. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_3 bitfield.

BOX_OUT_MODE_TTL_3	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_TTL_4

R/W, BOX_OUT_MODE_SET_A[9..8], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_TTL_4. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_4 bitfield.

BOX_OUT_MODE_TTL_$	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_TTL_5

R/W, BOX_OUT_MODE_SET_A[11..10], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_TTL_5. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_5 bitfield.

BOX_OUT_MODE_TTL_5	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_TTL_6

R/W, BOX_OUT_MODE_SET_A[13..12], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_TTL_6. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_6 bitfield.

BOX_OUT_MODE_TTL_6	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_TTL_7

R/W, BOX_OUT_MODE_SET_A[15..14], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_TTL_7. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_7 bitfield.

BOX_OUT_MODE_TTL_7	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_TTL_8

R/W, BOX_OUT_MODE_SET_A[17..16], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_TTL_8. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_8 bitfield.

BOX_OUT_MODE_TTL_8	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_TTL_9

R/W, BOX_OUT_MODE_SET_A[19..18], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_TTL_9. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_9 bitfield.

BOX_OUT_MODE_TTL_9	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_TTL_10

R/W, BOX_OUT_MODE_SET_A[21..20], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_TTL_10. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_10 bitfield.

BOX_OUT_MODE_TTL_10	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_TTL_11

R/W, BOX_OUT_MODE_SET_A[23..22], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_TTL_11. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_11 bitfield.

BOX_OUT_MODE_TTL_11	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

5.14 BOX_OUT_MODE_SET_B

Bit	Name
0	BOX_OUT_MODE_DIFF_0
1	BOX_OUT_MODE_DIFF_0
2	BOX_OUT_MODE_DIFF_1
3	BOX_OUT_MODE_DIFF_1
4	BOX_OUT_MODE_DIFF_2
5	BOX_OUT_MODE_DIFF_2
6	BOX_OUT_MODE_DIFF_3
7	BOX_OUT_MODE_DIFF_3
8	BOX_OUT_MODE_DIFF_4
9	BOX_OUT_MODE_DIFF_4
10	BOX_OUT_MODE_DIFF_5
11	BOX_OUT_MODE_DIFF_5
12	BOX_OUT_MODE_DIFF_6
13	BOX_OUT_MODE_DIFF_6
14	BOX_OUT_MODE_DIFF_7
15	BOX_OUT_MODE_DIFF_7
16	BOX_OUT_MODE_DIFF_8
17	BOX_OUT_MODE_DIFF_8
18	BOX_OUT_MODE_DIFF_9
19	BOX_OUT_MODE_DIFF_9
20	BOX_OUT_MODE_DIFF_10
21	BOX_OUT_MODE_DIFF_10
22	BOX_OUT_MODE_DIFF_11
23	BOX_OUT_MODE_DIFF_11
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	Reserved
30	Reserved
31	Reserved

BOX_OUT_ MODE_DIFF_0

R/W, BOX_OUT_MODE_SET_B[1..0], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_DIFF_0. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_0 bitfield.

BOX_OUT_MODE_DIFF_0	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_DIFF_1

R/W, BOX_OUT_MODE_SET_B[3..2], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_DIFF_1. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_1 bitfield.

BOX_OUT_MODE_DIFF_1	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_DIFF_2

R/W, BOX_OUT_MODE_SET_B[5..4], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_DIFF_2. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_2 bitfield.

BOX_OUT_MODE_DIFF_2	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_DIFF_3

R/W, BOX_OUT_MODE_SET_B[7..6], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_DIFF_3 . This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_d bitfield.

BOX_OUT_MODE_DIFF_3	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_DIFF_4

R/W, BOX_OUT_MODE_SET_B[9..8], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_DIFF_4. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_4 bitfield.

BOX_OUT_MODE_DIFF_4	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_DIFF_5

R/W, BOX_OUT_MODE_SET_B[11..10], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_DIFF_5. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_5 bitfield.

BOX_OUT_MODE_DIFF_5	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_DIFF_6

R/W, BOX_OUT_MODE_SET_B[13..12], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_DIFF_6. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_6 bitfield.

BOX_OUT_MODE_DIFF_6	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_DIFF_7

R/W, BOX_OUT_MODE_SET_B[15..14], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_DIFF_7. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_7 bitfield.

BOX_OUT_MODE_DIFF_7	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_DIFF_8

R/W, BOX_OUT_MODE_SET_B[17..16], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_DIFF_8. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_8 bitfield.

BOX_OUT_MODE_DIFF_8	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_DIFF_9

R/W, BOX_OUT_MODE_SET_B[19..18], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_DIFF_9. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_9bitfield.

BOX_OUT_MODE_DIFF_9	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_DIFF_10

R/W, BOX_OUT_MODE_SET_B[21..20], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_DIFF_10. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_10 bitfield.

BOX_OUT_MODE_DIFF_10	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_DIFF_11

R/W, BOX_OUT_MODE_SET_B[23..22], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_DIFF_11. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_11 bitfield.

BOX_OUT_MODE_DIFF_11	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

5.15 BOX_OUT_MODE_SET_C

Bit	Name
0	BOX_OUT_MODE_OPTO_0
1	BOX_OUT_MODE_OPTO_0
2	BOX_OUT_MODE_OPTO_1
3	BOX_OUT_MODE_OPTO_1
4	BOX_OUT_MODE_OPTO_2
5	BOX_OUT_MODE_OPTO_2
6	BOX_OUT_MODE_OPTO_3
7	BOX_OUT_MODE_OPTO_3
8	BOX_OUT_MODE_OPTO_4
9	BOX_OUT_MODE_OPTO_4
10	BOX_OUT_MODE_OPTO_5
11	BOX_OUT_MODE_OPTO_5
12	BOX_OUT_MODE_OPTO_6
13	BOX_OUT_MODE_OPTO_6
14	BOX_OUT_MODE_OPTO_7
15	BOX_OUT_MODE_OPTO_7
16	BOX_OUT_MODE_OC_0
17	BOX_OUT_MODE_OC_0
18	BOX_OUT_MODE_OC_1
19	BOX_OUT_MODE_OC_1
20	BOX_OUT_MODE_OC_2
21	BOX_OUT_MODE_OC_2
22	BOX_OUT_MODE_OC_3
23	BOX_OUT_MODE_OC_3
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	Reserved
30	Reserved
31	Reserved

BOX_OUT_ MODE_OPTO_0

R/W, BOX_OUT_MODE_SET_C[1..0], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_OPTO_0. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_0 bitfield.

BOX_OUT_MODE_OPTO_0	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_OPTO_1

R/W, BOX_OUT_MODE_SET_C[3..2], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_OPTO_1 This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_1 bitfield.

BOX_OUT_MODE_OPTO_1	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_OPTO_2

R/W, BOX_OUT_MODE_SET_C[5..4], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_OPTO_2 This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_2 bitfield.

BOX_OUT_MODE_OPTO_2	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_OPTO_3

R/W, BOX_OUT_MODE_SET_C[7..6], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_OPTO_3. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_3 bitfield.

BOX_OUT_MODE_XXX_3	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_OPTO_4

R/W, BOX_OUT_MODE_SET_C[9..8], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_OPTO_4. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_4 bitfield.

BOX_OUT_MODE_OPTO_4	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_OPTO_5

R/W, BOX_OUT_MODE_SET_C[11..10], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_OPTO_5. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_5 bitfield.

BOX_OUT_MODE_OPTO_5	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_OPTO_6

R/W, BOX_OUT_MODE_SET_C[13..12], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_OPTO_6. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_6 bitfield.

BOX_OUT_MODE_OPTO_6	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_OPTO_7

R/W, BOX_OUT_MODE_SET_C[15..14], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_OPTO_7. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_7 bitfield.

BOX_OUT_MODE_OPTO_7	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_OC_0

R/W, BOX_OUT_MODE_SET_C[17..16], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_OC_0 This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_8 bitfield.

BOX_OUT_MODE_OC_0	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_OC_1

R/W, BOX_OUT_MODE_SET_C[19..18], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_OC_1. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_9 bitfield.

BOX_OUT_MODE_OC_1	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_OC_2

R/W, BOX_OUT_MODE_SET_C[21..20], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_OC_2. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_10 bitfield.

BOX_OUT_MODE_OC_2	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BOX_OUT_ MODE_OC_3

R/W, BOX_OUT_MODE_SET_C[23..22], Cyton-CXP, Axion-CL

This bit controls the BitBox output BOX_OUT_MODE_OC_3. This bit controls whether this output is static (high or low) or dynamic. If the bit is dynamic, the source is controlled by the associated BOX_OUT_DYN_SEL_11 bitfield.

BOX_OUT_MODE_OC_3	Meaning
0 (00b)	Static output low
1 (01b)	Static output high
2 (10b)	Dynamic output
3 (11b)	Reserved

BitBox Programming

Chapter 6

6.1 Programming the BitBox

The BitBox is not really programming or configured. It powers up in a fixed state and presents a number of inputs and outputs to the frame grabber. The programming really takes place on the frame grabber and how the BitBox signals are routed into the board and off the board.

There are two main ways to control how the BitBox is used: from a camera configuration file (*.BFML file) or from software. The simplest method is set up your I/O via a camera configuration file. The nice thing about using the method is that it will configure the I/O of the board when using any application. Even if you plan on modifying the I/O configuration from your software, it usually makes sense to put the board in some kind of default I/O setup via the camera configuration file. This mode can always be modified later from software.

6.2 BitBox Access from the Camera Configuration File

Cyton and Axion camera configuration files (*.BFML) can be edited using BitFlow's CamML BFML file editor, an XML file editor or any text editor. By far the easiest method is to use CamML. All of the examples in this and the following sections can be done via any of these methods. However, it is simplest to view the underlying XML structure in order to understand how I/O is being configured.

The BFML files contain a number of modes for each camera. Each mode contains an I/O section. The I/O section can also contain a BitBox section which is used to route the inputs and control the outputs.

Note: These sections will be easier to understand if you refer to the BitBox input routing diagram in Figure 4-2 and the output routing in Figure 4-7.

6.2.1 Example: Selecting a BitBox Input as a Source For A VFG function

In the example below, the box's TTL input BOX_IN_TTL_0 has been selected as the source for the VFGs trigger.

Here is an example of what this I/O section might look like:

```
<io>
  <trigger_src source="BoxInTTL0"/>
</io>
```

Note: The <io> section is just one sub-node of a <mode> node. We have removed all the other nodes in this and the followin examples in order to keep the presentation clear and easy to understand.

To select the input BOX_IN_DIFF_5 as the trigger source, you would do this:

```
<io>
  <trigger_src source="BoxInDIFF5"/>
</io>
```

To select BOX_IN_DIFF_5 as the trigger, BOX_IN_TTL_0 as encoder A, and BOX_IN_TTL_1 as encoder B, you would do this:

```
<io>
  <trigger_src source="BoxInDIFF5"/>
  <enca_src source="BoxInTTL0"/>
  <encb_src source="BoxInTTL1"/>
</io>
```

Inputs are pretty simple, just select the Box source for the function that you want to use.

6.2.2 Example: Routing a BitBox Input to the Camera

In this example, we will take the input from the BitBox,BOX_IN_TTL_0, and send it to the internal trigger signal, then send the trigger signal to the camera via the CC1 signal.

```
<io>
  <trigger_src source="BoxInTTL0"/>
  <cc1_src source="TrigSel" />
</io>
```

You can send the same BitBox signal out to CC3, which might, for example power a strobe light by doing the following:

```
<io>
  <trigger_src source="BoxInTTL0"/>
  <cc1_src source="TrigSel" />
  <cc3_src source="TrigSel" />
</io>
```

6.2.3 Example: Setting BitBox Outputs To Static Values

The BitBox outputs have a subsection of the <io> block which is labeled <bitbox>. This subsection contains one or more output routing instructions. Since the output routing is quite flexible, these sections can get fairly large, but they are quite simple to understand.

In this example, we just set BOX_OUT_TTL_0 low:

```
<io>
  <bitbox>
    <bbx_out_route source="Low" destination="BoxOutTTL0"/>
  </bitbox>
</io>
```

As many of these routes as needed can be added to the <bitbox> section, here we set two outputs high and one low:

```
<io>
  <bitbox>
    <bbx_out_route source="High" destination="BoxOutTTL0"/>
    <bbx_out_route source="High" destination="BoxOutTTL1"/>
    <bbx_out_route source="Low" destination="BoxOutTTL2"/>
  </bitbox>
</io>
```

6.2.4 Example: Routing BitBox Inputs to BitBox Outputs

Routing a BitBox input to a BitBox output is fairly easy. In this example we select BOX_IN_TTL_0 as the trigger source. We then select the trigger as the source for CC1, and then we route CC1 to the output BOX_OUT_TTL_0:

```
<io>
  <trigger_src source="BoxInTTL0"/>
  <cc1_src source="CT0" />
  <bitbox>
    <bbx_out_route source="VFG0CC1" destination="BoxOutTTL0"/>
  </bitbox>
</io>
```

This doesn't really do much, as you could have just wired your signal going into the box directly to the signal leaving the box, since they are both TTL. However the following configuration would convert your TTL signal going into the BitBox to an opto-isolated signal:

```
<io>
  <trigger_src source="BoxInTTL0"/>
  <cc1_src source="CT0" />
  <bitbox>
    <bbx_out_route source="VFG0CC1" destination="BoxOutOpto0"/>
  </bitbox>
</io>
```

You could also broadcast your TTL signal to eight opto-isolated signals that all go to different destinations:

```
<io>
  <trigger_src source="BoxInTTL0"/>
  <cc1_src source="CT0" />
  <bitbox>
    <bbx_out_route source="VFG0CC1" destination="BoxOutOpto0"/>
    <bbx_out_route source="VFG0CC1" destination="BoxOutOpto1"/>
    <bbx_out_route source="VFG0CC1" destination="BoxOutOpto2"/>
    <bbx_out_route source="VFG0CC1" destination="BoxOutOpto3"/>
    <bbx_out_route source="VFG0CC1" destination="BoxOutOpto4"/>
    <bbx_out_route source="VFG0CC1" destination="BoxOutOpto5"/>
    <bbx_out_route source="VFG0CC1" destination="BoxOutOpto6"/>
    <bbx_out_route source="VFG0CC1" destination="BoxOutOpto7"/>
  </bitbox>
</io>
```

6.2.5 Example: Setting BitBox Outputs To Dynamic Signal

The <bitbox> section can also be used to route dynamic signals from the VFG to the BitBox outputs.

In this example, we just set an BOX_OUT_TTL_0 to the CC1 signals, which can be coming from a number of different sources:

```
<io>
  <bitbox>
    <bbx_out_route source="VFG0CC1" destination="BoxOutTTL0"/>
  </bitbox>
</io>
```

Next lets create a source for CC1 like this:

```
<io>
  <cc1_src source="CT0" />
  <bitbox>
    <bbx_out_route source="VFG0CC1" destination="BoxOutTTL0"/>
  </bitbox>
</io>
```

Now the CC1 signals is being driven by the CT0 signal, which comes out of the Timing Sequencer (TS). To complete this example, we should also program the TS. Here we program the TS to a 50 Hz square wave. The square wave is on the signal CT0, which is selected as the source for CC1. CC1 is then selected as the source for BOX_OUT_TTL_0:

```
<io>
  <cc1_src source="CT0" />
  <bitbox>
    <bbx_out_route source="VFG0CC1" destination="BoxOutTTL0"/>
  </bitbox>
</io>
<timing_sequencer>
  <ts_segment trigger="NoWait">
    <ts_period>10.0</ts_period>
    <ts_ct0_state>1</ts_ct0_state>
  </ts_segment>
  <ts_segment trigger="NoWait">
    <ts_period>10.0</ts_period>
    <ts_ct0_state>0</ts_ct0_state>
  </ts_segment>
<timing_sequencer>
```

The same signal can be sent to multiple outputs on the BitBox:

```
<io>
  <cc1_src source="CT0" />
  <bitbox>
    <bbx_out_route source="VFG0CC1" destination="BoxOutTTL0"/>
    <bbx_out_route source="VFG0CC1" destination="BoxOutDiff0"/>
    <bbx_out_route source="VFG0CC1" destination="BoxOutOpto0"/>
  </bitbox>
</io>
```

```
<timing_sequencer>
  <ts_segment trigger="NoWait">
    <ts_period>10.0</ts_period>
    <ts_ct0_state>1</ts_ct0_state>
  </ts_segment>
  <ts_segment trigger="NoWait">
    <ts_period>10.0</ts_period>
    <ts_ct0_state>0</ts_ct0_state>
  </ts_segment>
<timing_sequencer>
```

The Cyton and the Axion have two timing sequencers, each of which can be programmed separately. In this following example, we program the two Sequencers, and then send each TS's output to a different BitBox output.

```
<io>
  <cc1_src source="CT0" />
  <bitbox>
    <bbx_out_route source="VFG0CC1" destination="BoxOutOC0"/>
    <bbx_out_route source="VFG0ATSCT0" destination="BoxOutOC1"/
   >
  </bitbox>
</io>
<timing_sequencer>
  <ts_segment trigger="NoWait">
    <ts_period>10.0</ts_period>
    <ts_ct0_state>1</ts_ct0_state>
  </ts_segment>
  <ts_segment trigger="NoWait">
<timing_sequencer_aux>
  <ts_segment trigger="NoWait">
    <ts_period>33.33</ts_period>
    <ts_ct0_state>1</ts_ct0_state>
  </ts_segment>
  <ts_segment trigger="NoWait">
    <ts_period>33.33</ts_period>
    <ts_ct0_state>0</ts_ct0_state>
  </ts_segment>
<timing_sequencer_aux>
```

6.3 Controlling BitBox Routing From Software

All of the routing of the inputs and outputs can be fully controlled from software. As mentioned previously in this chapter, it is often simpler and faster to set up the BitBox routing using the camera configuration file (*.BFML). However, there are many application where you may want to change on-the-fly the BitBox routing from your application. This section illustrates some common example.

Note: The registers that control BitBox routing are all described in Section 5.1.

Note: Code in this section will only work on the Gen 2 family of frame grabbers (Axion, Cyton).

6.3.1 Setting the BitBox Outputs To Static Values

Setting the BitBox output to static high or low values is easy from software. Just poke the appropriate mode register for the output you want to control. In this case we set BOX_OUT_TTL_0 high:

```
BFRegPoke(hBoard, REG_BOX_OUT_MODE_TTL_0, 1);
```

Set it low:

```
BFRegPoke(hBoard, REG_BOX_OUT_MODE_TTL_0, 0);
```

6.3.2 Setting the BitBox Outputs to a Dynamic Source

Setting the BitBox output to a dynamic source, such a the output of the TS, is also pretty simple, but it does require more register accesses. In this example we program the mode bit first, for dynamic access, then we program the dynamic selection bits to drive our output from CC1:

```
// Mode is dynamic
BFRegPoke(hBoard, REG_BOX_OUT_MODE_TTL_0, 2);
// Source is CC1
BFRegPoke(hBoard, REG_BOX_OUT_DYN_SEL_0, 0);
```

To change the source to the Auxiliary Timing Sequencer CT0

```
// Source is ATS CT0
BFRegPoke(hBoard, REG_BOX_OUT_DYN_SEL_0, 17);
```

Set the output back to static (low):

```
BFRegPoke(hBoard, REG_BOX_OUT_MODE_TTL_0, 0);
```

6.3.3 Switching Trigger Sources From Software (Register Access)

In this example, we will set the board up to use BOX_IN_TTL_0 as a trigger. This is done with the following snippet of the BFML file:

```
<io>
  <trigger_src source="BoxInTTL0"/>
</io>
```

Now let's look at the code that would be required to support changing the trigger source from BOX_IN_TTL_0 to BOX_IN_TTL_1, BOX_IN_TTL_2, etc. based on a variable (coming from elsewhere in the code) called *TrigSource*. Let's assume *TrigSource* will be in the range of 0 to 11 to request BOX_IN_TTL_0 to BOX_IN_TTL_11. Here is the simple code:

```
BFRegPoke(hBoard, REG_SEL_TRIG, TrigSource + 28);
```

If you look up the definition of the register REG_SEL_TRIG, you will see that programming it to 28 will tell the board to use BOX_IN_TTL_0, setting it to 29 will use BOX_IN_TTL_1, and so on.

6.3.4 Switching Trigger Sources From Software (API)

In the example above, the trigger input source is selected by poking the trigger selection register directly. In addition, there are API functions which can be used to accomplish the same thing. The advantage here is that API functions are easier to understand, they also support setting the trigger source to the Axion/Cyton's connectors (i.e. switching the source between the BitBox and the main I/O connector).

Note: The API functions used here are documented in the SDK Software Reference Manual, please see this manual for more details on these functions.

To set the VFG's trigger input to BOX_IN_TTL_0:

```
CiConTriggerInputSet(hBoard, BFTrigBoxInTTL0,
   BFTrigAssertedHigh);
```

To set the VFG's trigger input to BOX_IN_24V_0:

```
CiConTriggerInputSet(hBoard, BFTrigBoxIn24V3,
   BFTrigAssertedHigh);
```

To set the VFG's trigger input to TTL trigger on its main I/O connector (VFG0_TRIGGER_TTL):

```
CiConTriggerInputSet(hBoard, BFTrigTTL, BFTrigAssertedHigh);
```

Note: The constants used above, BFTrigBoxInTTL0, BFTrigBoxIn24V3, etc. are all defined in the SDK header file "BFBoard.h".

6.3.5 Switching Encoder Sources From Software (API)

As in the examples above, the encoder sources can also be set via API functions. There are two encoders, A and B. The encoder functions takes a parameter that specifies which encoder you are using.

To set the VFG's encoder A input to BOX_IN_TTL_0:

```
CiConTriggerEncoderSet(hBoard, BFTypeEncA, BFEncBoxInTTL0,
   BFEncAssertedHigh);
```

To set the VFG's encoder B input to BOX_IN_TTL_1:

```
CiConTriggerEncoderSet(hBoard, BFTypeEncB, BFEncBoxInTTL1,
   BFEncAssertedHigh);
```

Specifications

Chapter 7

7.1 Introduction

This chapter describes the general specifications of the BitBox. The numerical values for the specifications are listed in Table 7-1. If more information is available for a given specification there will be an entry in the column marked "Details".

Table 7-1 BitBox Specifications

Specifications	**Value**	**Units**	**Details**
Input Voltage	5 to 24	Volts	
Current	0.18	Amps @ 24 Volts	Section 7.2
Temperature range	0 to 50	Degrees Celsius	
Humidity	25% to 80%		
Mechanical dimensions	17.4 x 8.87 x 4.6	Centimeter	with no connectors plugged in
Mechanical dimensions	6.85 x 3.5 x 1.8	Inches	with no connectors plugged in
Mounting	DIN 35 Rail		
Differential Drivers	DS34LV87T		Section 2.5
Differential Receivers	SNLVDS3486		Section 2.4
TTL Drivers	SN74LVTH241		Section 2.3
TTL Receivers	SN74LVTH241		Section 2.2
Maximum cable length PC to BitBox	10	Meters	

7.2 BitBox Power Requirements

The BitBox will work with an input anywhere in the range from 5 to 24 volts. The amount of current that BitBox will draw depends on the input voltage. Table 7-2 shows the current draw for a few different input voltages.

Table 7-2 BitBox Current Draw

Input Volltage	Current
5 Volts	0.6 Amps
12 Volts	0.26 Amps
24 Volts	0.18 Amps

Mechanical

Chapter 8

8.1 Introduction

This chapter describes the mechanical characteristics of the BitBox. This includes description of all of the connectors on the board and pin-outs for these connectors.

The mechanical layout of the BitBox is shown in Figure 8-1.

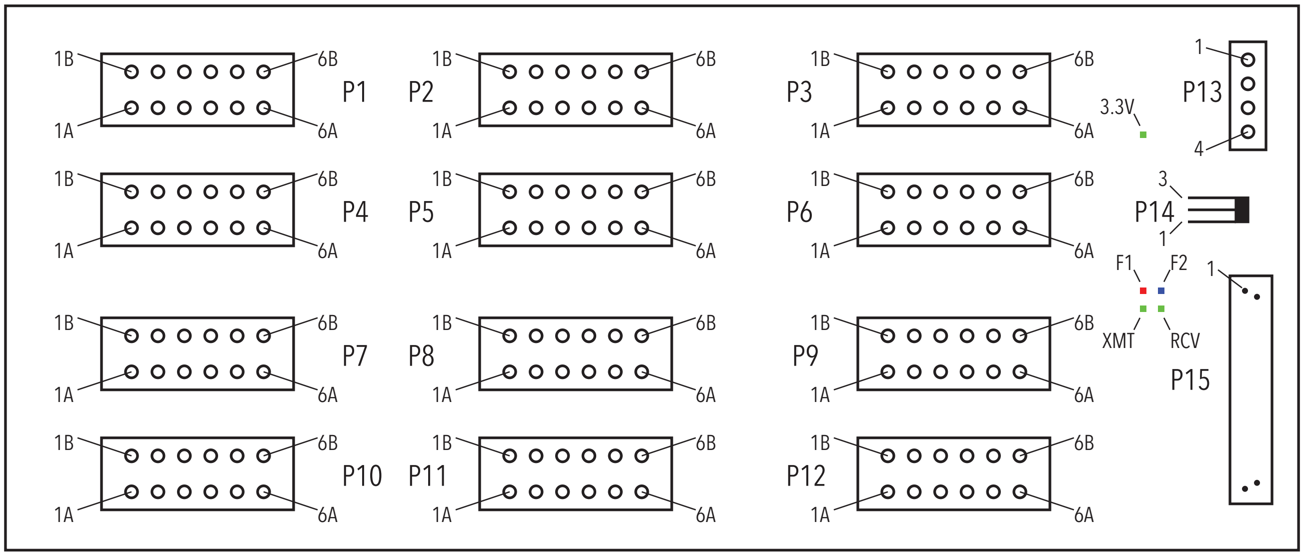

Figure 8-1 BitBox Board Layout

8.2 The BitBox Connectors

There are fourteen connectors on the BitBox:

P1 through P12 - Input/Output sockets for 12-pin block connectors
P13 - Power connector
P15 - Control cable connector, goes to frame grabber

Figure 8-1 shows the locations of these connectors. The following sections show the details of each of these connectors.

8.3 Jumpers

There is one user configurable jumper on the BitBox, it controls the source of the power that is provided to the BitBox control logic..

Table 8-1 Jumper P14

Jumper Position	Meaning
1 & 2	Power is taken from the frame grabber via the control cable (connected to P15)
2 & 3	Power is taken from the power connector (P13)

8.4 LEDs

The BitBox has 4 LEDs. Table 8-2 Describes the function of these LEDs.

Table 8-2 LEDs

LED Number	Color	Function
F1	Red	Illuminated when the output BOX_OUT_TTL_10 is high
F2	Blue	Illuminated when the output BOX_OUT_TTL_11 is high
XMT	Green	Illuminated for a short period of time when one of the input signals has changed.
RCV	Green	Illuminated for a short period of time when one of the output signals has changed.
3.3V	Green	When illuminated the BitBox is powered

Note: The LEDs F1 and F2 will illuminate for a short period of time (~500 milliseconds). if the output they are associated with recieves a pulse of any duration. For example, if BOX_OUT_TTL_10 *is sent a 10 microsecond pulse from the frame grabber, F1 will still illuminate for about 500 milliseconds. The reason for this is to provide visibility of short duration state changes that ordinarily would be too short to be seen via an LED.*

8.5 The Power Connector (P13)

There are two ways to power the BitBox. Option 1: power can come from the PC, via the control cable. Option2: power can be locally provided by connecting it to P13. The jumper P15 controls the source of the power (see Section 8.3).

Note: if power is coming from the control cable, the internal PC connector CAM-BOX_ INT_PWR must be used. Further, this cable has a power connector which must be connected to one of the PC's power supply power taps.

Note: Jumper P13 must be in set to the correct position to take power from this connector.

Note: The BitBox will work with any input voltage from 5 to 24 Volts.

Table 8-3 Power Connector

Pin	Voltage
1	GND
2	GND
3	GND
4	5 to 24 Volts

8.6 The Control Connector (P15)

This connector is used to connect the BitBox to a BitFlow frame grabber. This cable provides all communications between the BitBox's inputs and output and the I/O circuitry on the frame grabber. This connector accepts standard D-Sub 15 pin female connectors.

Note: The BitBox will not function unless it is connected to a frame grabber.

Note: All control are LVDS differential, and should be wired to the frame grabber via twisted pair conductors. Please contact BitFlow for further information on cables.

Table 8-4 P15 Control Connector

Pin	Function	Comment
1	GND	
2	SYNC_OUT-	Differential pair
3	CLK_OUT-	Differential pair
4	DATA_OUT-	Differential pair
5	SYNC_IN-	Differential pair
6	CLK_IN-	Differential pair
7	DATA_IN-	Differential pair
8	12V	Only needed if power comes from the PC. See Jumper P13.
9	GND	
10	SYNC_OUT+	Differential pair
11	CLK_OUT+	Differential pair
12	DATA_OUT+	Differential pair
13	SYNC_IN+	Differential pair
14	CLK_IN+	Differential pair
15	DATA_IN+	Differential pair

8.7 The Output Connectors P1 Through P6

The pin-out for the output connector blocks are in the following tables. Each connector block has 12 pins (1A through 6A and 1B through 6B).

Note: Each block of 12 signals accepts one 12-pin block connector. Contact BitFlow for these connectors.

Note: PWR in the "I/O" column means that the user must supply power for this circuit to work, please see Section 2.1 for more information.

Table 8-5 Output Connector P1

Pin	I/O	Signal	Comment
1A	PWR	BOX_OUT_OPTO_VCC_0	Optocoupler
2A	Out	BOX_OUT_OPTO_0	Optocoupler
3A	GND	BOX_OUT_OPTO_GND_0	Optocoupler
4A	PWR	BOX_OUT_OPTO_VCC_1	Optocoupler
5A	Out	BOX_OUT_OPTO_1	Optocoupler
6A	GND	BOX_OUT_OPTO_GND_1	Optocoupler
1B	PWR	BOX_OUT_OPTO_VCC_2	Optocoupler
2B	Out	BOX_OUT_OPTO_2	Optocoupler
3B	GND	BOX_OUT_OPTO_GND_2	Optocoupler
4B	PWR	BOX_OUT_OPTO_VCC_3	Optocoupler
5B	Out	BOX_OUT_OPTO_3	Optocoupler
6B	GND	BOX_OUT_OPTO_GND_3	Optocoupler

Table 8-6 Output Connector P2

Pin	I/O	Signal	Comment
1A	PWR	BOX_OUT_OPTO_VCC_4	Optocoupler
2A	Out	BOX_OUT_OPTO_4	Optocoupler
3A	GND	BOX_OUT_OPTO_GND_4	Optocoupler
4A	PWR	BOX_OUT_OPTO_VCC_5	Optocoupler
5A	Out	BOX_OUT_OPTO_5	Optocoupler
6A	GND	BOX_OUT_OPTO_GND_5	Optocoupler
1B	PWR	BOX_OUT_OPTO_VCC_6	Optocoupler
2B	Out	BOX_OUT_OPTO_6	Optocoupler
3B	GND	BOX_OUT_OPTO_GND_6	Optocoupler
4B	PWR	BOX_OUT_OPTO_VCC_7	Optocoupler
5B	Out	BOX_OUT_OPTO_7	Optocoupler
6B	GND	BOX_OUT_OPTO_GND_7	Optocoupler

Table 8-7 Output Connector P3

Pin	I/O	Signal	Comment
1A	PWR	BOX_OUT_OC_VCC_0	Open Collector
2A	PWR	BOX_OUT_OC_VCC_1	Open Collector
3A	PWR	BOX_OUT_OC_VCC_2	Open Collector
4A	PWR	BOX_OUT_OC_VCC_3	Open Collector
5A	Out	BOX_OUT_TTL_10	TTL
6A	GND	GND	
1B	Out	BOX_OUT_OC_0	Open Collector
2B	Out	BOX_OUT_OC_1	Open Collector
3B	Out	BOX_OUT_OC_2	Open Collector
4B	Out	BOX_OUT_OC_3	Open Collector
5B	Out	BOX_OUT_TTL_11	TTL
6B	GND	GND	

Table 8-8 Output Connector P4

Pin	I/O	Signal	Comment
1A	Out	BOX_OUT_DIFF_0+	RS422
2A	Out	BOX_OUT_DIFF_1+	RS422
3A	Out	BOX_OUT_DIFF_2+	RS422
4A	Out	BOX_OUT_DIFF_3+	RS422
5A	Out	BOX_OUT_DIFF_4+	RS422
6A	Out	BOX_OUT_DIFF_5+	RS422
1B	Out	BOX_OUT_DIFF_0-	RS422
2B	Out	BOX_OUT_DIFF_1-	RS422
3B	Out	BOX_OUT_DIFF_2-	RS422
4B	Out	BOX_OUT_DIFF_3-	RS422
5B	Out	BOX_OUT_DIFF_4-	RS422
6B	Out	BOX_OUT_DIFF_5-	RS422

Table 8-9 Output Connector P5

Pin	I/O	Signal	Comment
1A	Out	BOX_OUT_DIFF_6+	RS422
2A	Out	BOX_OUT_DIFF_7+	RS422
3A	Out	BOX_OUT_DIFF_8+	RS422
4A	Out	BOX_OUT_DIFF_9+	RS422
5A	Out	BOX_OUT_DIFF_10+	RS422
6A	Out	BOX_OUT_DIFF_11+	RS422
1B	Out	BOX_OUT_DIFF_6-	RS422
2B	Out	BOX_OUT_DIFF_7-	RS422

Table 8-9 Output Connector P5

Pin	I/O	Signal	Comment
3B	Out	BOX_OUT_DIFF_8-	RS422
4B	Out	BOX_OUT_DIFF_9-	RS422
5B	Out	BOX_OUT_DIFF_10-	RS422
6B	Out	BOX_OUT_DIFF_11-	RS422

Table 8-10 Output Connector P6

Pin	I/O	Signal	Comment
1A	Out	BOX_OUT_TTL_0	TTL
2A	Out	BOX_OUT_TTL_2	TTL
3A	Out	BOX_OUT_TTL_4	TTL
4A	Out	BOX_OUT_TTL_6	TTL
5A	Out	BOX_OUT_TTL_8	TTL
6A	GND	GND	
1B	Out	BOX_OUT_TTL_1	TTL
2B	Out	BOX_OUT_TTL_3	TTL
3B	Out	BOX_OUT_TTL_5	TTL
4B	Out	BOX_OUT_TTL_7	TTL
5B	Out	BOX_OUT_TTL_9	TTL
6B	Out	GND	

8.8 The Input Connectors P7 Through P12

The pin-out for the input connector blocks are in the following tables. Each connector block has 12 pins (1A through 6A and 1B through 6B).

Note: Each block of 12 signals accepts one 12-pin block connector. Contact BitFlow for part numbers and how to order these connectors.

Table 8-11 Output Connector P7

Pin	I/O	Signal	Comment
1A	In	BOX_IN_OPTO_ANODE_6	Optocoupler
2A	In	BOX_IN_OPTO_ANODE_7	Optocoupler
3A			
4A		Test1	See Section 8.9
5A		3.3V	See Section 8.9
6A		GND	See Section 8.9
1B	In	BOX_IN_OPTO_CATHODE_6	Optocoupler
2B	In	BOX_IN_OPTO_CATHODE_7	Optocoupler
3B			
4B		Test2	See Section 8.9
5B		3.3V	See Section 8.9
6B		GND	See Section 8.9

Table 8-12 Output Connector P8

Pin	I/O	Signal	Comment
1A	In	BOX_IN_OPTO_ANODE_0	Optocoupler
2A	In	BOX_IN_OPTO_ANODE_1	Optocoupler
3A	In	BOX_IN_OPTO_ANODE_2	Optocoupler
4A	In	BOX_IN_OPTO_ANODE_3	Optocoupler
5A	In	BOX_IN_OPTO_ANODE_4	Optocoupler
6A	In	BOX_IN_OPTO_ANODE_5	Optocoupler
1B	In	BOX_IN_OPTO_CATHODE_0	Optocoupler
2B	In	BOX_IN_OPTO_CATHODE_1	Optocoupler
3B	In	BOX_IN_OPTO_CATHODE_2	Optocoupler
4B	In	BOX_IN_OPTO_CATHODE_3	Optocoupler
5B	In	BOX_IN_OPTO_CATHODE_4	Optocoupler
6B	In	BOX_IN_OPTO_CATHODE_5	Optocoupler

Table 8-13 Output Connector P9

Pin	I/O	Signal	Comment
1A	In	BOX_IN_DIFF_0+	LVDS
2A	In	BOX_IN_DIFF_1+	LVDS
3A	In	BOX_IN_DIFF_2+	LVDS
4A	In	BOX_IN_DIFF_3+	LVDS
5A	In	BOX_IN_DIFF_4+	LVDS
6A	In	BOX_IN_DIFF_5+	LVDS
1B	In	BOX_IN_DIFF_0-	LVDS
2B	In	BOX_IN_DIFF_1-	LVDS
3B	In	BOX_IN_DIFF_2-	LVDS
4B	In	BOX_IN_DIFF_3-	LVDS
5B	In	BOX_IN_DIFF_4-	LVDS
6B	In	BOX_IN_DIFF_5-	LVDS

Table 8-14 Output Connector P10

Pin	I/O	Signal	Comment
1A	In	BOX_TTL_TTL_0	TTL
2A	In	BOX_TTL_TTL_2	TTL
3A	In	BOX_TTL_TTL_4	TTL
4A	In	BOX_TTL_TTL_6	TTL
5A	In	BOX_TTL_TTL_8	TTL
6A	GND	GND	
1B	In	BOX_TTL_TTL_1	TTL
2B	In	BOX_TTL_TTL_3	TTL
3B	In	BOX_TTL_TTL_5	TTL
4B	In	BOX_TTL_TTL_7	TTL
5B	In	BOX_TTL_TTL_9	TTL
6B	GND	GND	

Table 8-15 Output Connector P11

Pin	I/O	Signal	Comment
1A	In	BOX_IN_24V_0	24V
2A	In	BOX_IN_24V_2	24V
3A	GND	GND	
4A	GND	GND	
5A	In	BOX_IN_TTL_10	TTL
6A	GND	GND	
1B	In	BOX_IN_24V_1	24V
2B	In	BOX_IN_24V_3	24V

Table 8-15 Output Connector P11

Pin	I/O	Signal	Comment
3B	GND	GND	
4B	GND	GND	
5B	In	BOX_IN_TTL_11	TTL
6B	GND	GND	

Table 8-16 Output Connector P12

Pin	I/O	Signal	Comment
1A	In	BOX_IN_DIFF_6+	LVDS
2A	In	BOX_IN_DIFF_7+	LVDS
3A	In	BOX_IN_DIFF_8+	LVDS
4A	In	BOX_IN_DIFF_9+	LVDS
5A	In	BOX_IN_DIFF_10+	LVDS
6A	In	BOX_IN_DIFF_11+	LVDS
1B	In	BOX_IN_DIFF_6-	LVDS
2B	In	BOX_IN_DIFF_7-	LVDS
3B	In	BOX_IN_DIFF_8-	LVDS
4B	In	BOX_IN_DIFF_9-	LVDS
5B	In	BOX_IN_DIFF_10-	LVDS
6B	In	BOX_IN_DIFF_11-	LVDS

8.9 P7 Test Pins

The test pins on the BitBox can be used for system integrity checking. Table 8-17 shows how the test pins can be configured for testing.

Note: The 3.3 Volts on pins P7-5A and P7-5B should only be used to control logic levels (i.e. pull up power). They should not be used to power external devices.

Table 8-17 Test Pin Configurations

Pin	Connect To	Commend
Test1	Open	Normal BitBox operation
Test1	GND	Loop back mode - all 36 output signals from the frame grabber are returned to frame grabber on the 36 input signals
Test2		Reserved

Index