

# Camera Control Tables (CTABs)

## Chapter 1

### 1.1 Introduction

The Camera Control Tables (CTABs) are one of the more flexible features of all of Bit-Flow's frame grabbers. At their most basic level, the CTABs are a set of programmable waveform generators. The number of different ways these waveforms can be programmed and used, however, is the heart the CTABs flexibility. It may also be the source of some confusion. This chapter, therefore, will attempt to explain the CTABs, how they can be used, and how they can be programmed.

#### 1.1.1 CTAB Concepts

The CTABs are simple a chunk of memory. At each memory address a value is programmed. Each bit of each memory address has a specific purpose. The memory is traversed by a counter, similar to how a CPU traverses program memory. The counter increments through memory, usually one address at a time. When the counter arrives at an address, the value of the bit, 0 or 1, causes some action to be take elsewhere on the board. See Figure 1-1 for a simple model of the CTAB.

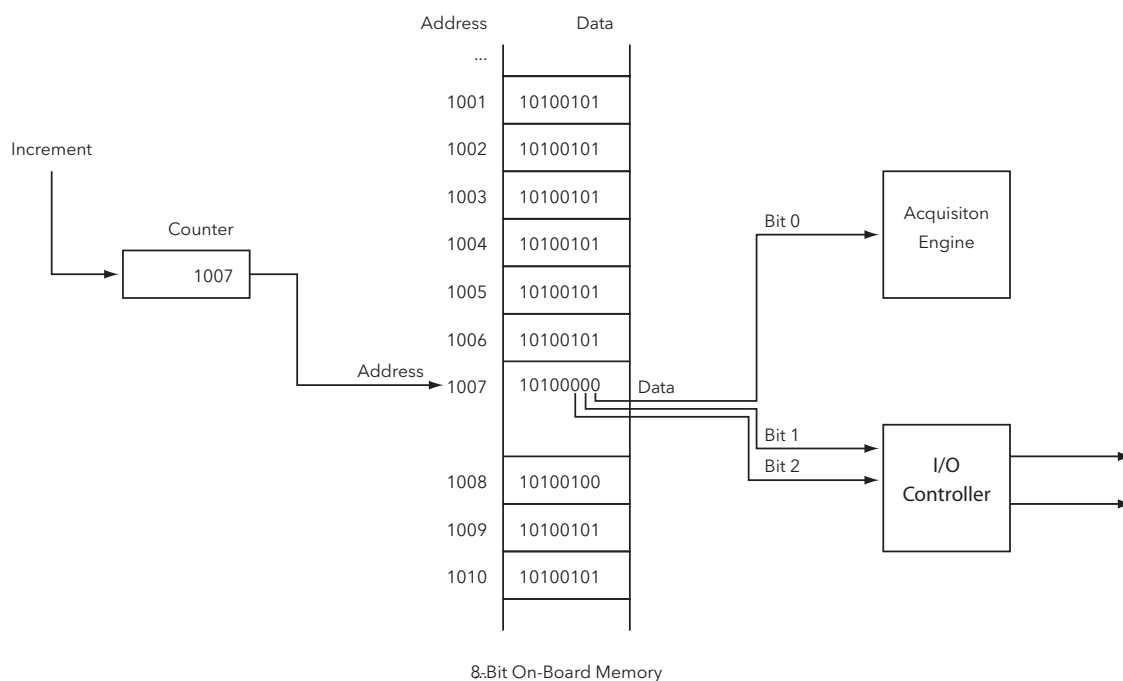


Figure 1-1 Simple CTAB model

In Figure 1-1 there is a counter which is used to indicate which entry in the CTAB memory is current. When the counter is incremented, the next entry becomes current and its value now controls other parts of the board. For example, a bit going from 0 to 1 might tell the acquisition engine to start acquiring pixels. Another bit might cause an I/O signal to go from a low state to a high state. Other bits can have control over the counter itself, for example one bit might cause the counter to jump to a specific location, or another bit might cause the counter to reset to zero.

### 1.1.2 CTAB structure

Every board has two CTABs, one for horizontal events and one for vertical events. Each CTAB has its own counter, there is a horizontal counter (HCOUNT) used to control horizontal events and a vertical counter (VCOUNT) used to control vertical events. These two CTABs are usually, but necessarily, tied to the connected camera's timing. This means that the horizontal CTAB (HCTAB) is used to control acquisition of the line and to create signals one a line by line basis. The vertical CTAB (VCTAB) is used to control acquisition of the frame and to create signals on a frame by frame basis. However, the CTABs are very flexible, and can be run independent of the camera's timing.

The purpose of each bit in the HCTAB differs from that of the bits in the VCTAB. There are similarities, but for various reasons, the horizontal events are not completely symmetrical with the vertical events. See Section 1.2 for details on the horizontal CTAB and Section 1.3 for details on the vertical CTAB.

### 1.1.3 When To Use the CTABs and When To Use the New Timing Generator (NTG)

One of the main uses of the CTABs is to control external devices like cameras and strobes. The New Timing Generator (NTG) is also capable of control these same devices. So the question is when does it make sense to use the CTABs and when does it make sense to use the NTG.

The NTG has been designed to be very easy to use. The NTG can be programmed in units of time (e.g. milliseconds), it is very simple to program, just a line/frame rate and exposure time parameters are all that are needed. The CTABs on the other hand, are somewhat more complicated to program, and their timing units are based on the camera's pixel clock and line rate. So the simple answer is: use the NTG if at all possible. The only time that you might need to use the CTABs is if you need to control more than one device, or if the waveform you are need is more complex than a simple pulse.

Also, as you will see in the coming sections, the CTABs can be used for more than just producing control signals. The CTABs can also be used for control capture of a sub-window from a camera. It can also produce interrupts, and has a few other esoteric functions. For all of these functions, the CTABs must be used.

Table 1-1 summarizes the capabilities of the NTG vs the CTABs.

**Table 1-1 NTG vs. CTABs**

<b>Feature</b>	<b>NTG</b>	<b>CTABs</b>
Produces pulses to drive external devices	Yes	Yes
Can be programmed in real world time units	Yes	No
Can support both line/frame rate and exposure time	Yes	Yes
Is simple to program	Yes	No
Can produce complex waveforms	No	Yes
Can be used to control the ROI	No	Yes
Can produce multiple independent signals	No	Yes
Can be controlled by a trigger/encoder	Yes	Yes
Can control both line scan cameras and area scan cameras	Yes	Yes
Can be programmed in CamVert	Yes	Yes
Can be programmed in CamEd	Yes	Yes
Can be programmed from software	Yes	Yes
Can be used to produce interrupts	No	Yes

## 1.2 The Horizontal CTAB (HCTAB)

The function of each bit in the HCTAB is illustrated in Table 1-2.

Table 1-2 The HCTAB Bits

Bit	Name	Function
0	HSTART	Start of Horizontal Acquisition Window (HAW)
1	HRESET	Reset HCOUNT to 0
2	ENHLOAD	LEN Mask, enable horizontal load
3	reserved	
4	GPH0	General purpose horizontal function 0
5	GPH1	General purpose horizontal function 1
6	GPH2	General purpose horizontal function 2
7	GPH3	General purpose horizontal function 3

### 1.2.1 HCTAB Bits

#### HSTART

This bit marks the start of the Horizontal Acquisition Window, HAW. Pixels will be acquired only while the HAW is active. A 1 in this bit tells the Acquisition Engine to start acquiring pixels. In other words, this bit is used to position the first pixel of the line acquired. Usually this is located with respect to the LEN signal.

#### HRESET

This bit will reset the HCOUNT to 0.

#### ENHLOAD

This bit controls how HCOUNT reacts to the assertion of the horizontal sync signal, LEN. If this bit is 1, then HCOUNT will jump (load) to 2000h when LEN asserts. If this bit is 0, then assertion of LEN will be ignored.

#### GPH0, GPH1, GPH2, GPH3

These bits are used to control I/O signals. They are combined with GPV0, GPV1, GPV2 and GPV3 respectively to create the CT0, CT1, CT2 and CT3 signals.

## 1.2.2 HCOUNT Flow Control

### **HCOUNT Increments**

HCOUNT is incremented every 8 pixel clocks (from the camera). This means that the granularity of the HCTAB is 8 pixel clocks and all events happen on 8-pixel clock boundaries.

There are only two instances when we HCTAB is not incremented. The first instance is when HCOUNT reaches 0000h, "Stop at Zero" case. The other instance is when HCOUNT reaches 1FF0h, the "Horizontal Stick" case. Both of these cases are described below.

### **HCOUNT Stops at 0**

Usually, HCOUNT will reach 0 because of a HRESET signal or because it has been reset automatically at the end of HAW. After HCOUNT is reset, there are two programmable options:

- HCOUNT keeps on counting.

- HCOUNT stays at zero until ENCODER is asserted.

The selection between the two options is done by the HCNT\_RLS\_ZERO bitfield. This bits controls how the HCOUNT can be released from zero.

### **HCOUNT Sticks at 1FF0h**

In some cases, it is desirable to hold the HCOUNT at a fixed point (in this case, the address 1FF0h) for an indeterminate amount of time, until the LEN signal is asserted. This is stick point is separate and independent from the stop a 0 control. The horizontal stick point is controlled by the HCNT\_RLS\_STK bitfield.

### **HCOUNT Jumps to 2000h**

HCOUNT will be loaded with the value 2000h by the assertion of LEN, if ENHLOAD is 1 when the assertion occurs. LEN usually marks the start of valid data in a line. The HSTART bit (controls the HAW) can be placed starting at address 2000h. By placing HSTART later then 2000h, an delay is introduced between LEN and the first pixel of the line. This is necessary with some cameras as the first pixels after LEN may not be valid. This delay can also be used to acquire a sub-window (ROI) out of the camera's total image output.

### **HCOUNT resets to 0**

This logic is used to reset HCOUNT to zero, and thus start the horizontal cycle over again. HCOUNT can be reset from several sources, according to HCNT\_RST bitfield. However, if HCOUNT reaches a location where the HRESET bit is set, HCOUNT always resets.

The horizontal counter will also reset to 0 if it reaches the end of the HCTAB, at address 8000h.

### 1.2.3 Simple HCTAB Example

Lets look at a simple example to clarify the concept of the HCTAB. Assume we want to program a free running horizontal window of 32 pixels active area. Just before the active area we want to fire a strobe using GPH0. HSTART defines the start of the HAW. Bit HRESET defines the reset of the HCOUNT. GPH0 is the strobe pulse. The size of the HAW is programmed in ACLP register.

Taking into account that the address counter is clocked by 1/8 the pixel clock, the HCTAB memory map will be as shown in Table 1-3.

Table 1-3 HCTAB Simple Example

HCTAB Address	HRESET	HSTART	GPH0	Comments
0	0	0	0	You got here from address 9
1	0	0	0	
2	0	0	0	
3	0	0	1	Fire the strobe
4	0	1	0	Start Horizontal Acquisition Window
5	0	0	0	Acquire
6	0	0	0	Acquire
7	0	0	0	Acquire
8	0	0	0	Acquire
9	1	0	0	Reset HCOUNT
10	0	0	0	

In this example, the HCOUNT starts at zero. The camera's pixel clock increments the HCOUNT every 8 clocks. For the first three HCTAB entries, nothing happens. Then when HCOUNT = 3, GPH0 goes 1, this bit controls one of the boards outputs, which in this case is used to fire a strobe. Next, HCOUNT increments to address 4, and HSTART is 1, so acquisition begins. The board continues to acquire the number of pixels specified by the register ACPL. Eventually, the HCOUNT reaches address 9, where HRESET is 1, and thus HCOUNT is reset to 0.

### 1.2.4 The HCTAB I/O Functions

There are four General Purpose Horizontal bits in the HCTAB: GPH0, GPH1, GPH2 and GPH3. These are combined with the four General Purpose Vertical bits to create four independent internal signal CT0, CT1, CT2 and CT3. These signals are combined as follows:

$$CT0 = GPV0 \text{ AND } GPH0$$

CT1 = GPV1 AND GPH1

CT2 = GPV2 AND GPH2

CT3 = GPV3 AND GPH3

The "AND" means that the bits are ANDed together using the logic of an AND gate. In other words, if both the horizontal and vertical bits are 1, then the corresponding CT will be 1 (high). Otherwise the CT will be 0 (low).

Any of the CT's (CT0 to CT3) can be steered to any of the Camera Controls signals (CC1 to CC4) (on the CL connectors) and to the GPOUTs (GPOUT0 to GPOUT6), on the I/O connector. Thus any of these general purpose CTAB bits GPH0 to GPH3 and GPV0 to GPV3 can control any of the I/O outputs on the board.

## 1.3 The Vertical CTAB (VCTAB)

The function of each bit in the VCTAB is illustrated in Table 1-4.

Table 1-4 The VCTAB Bits

Bit	Name	Function
0	VSTART	Start of Vertical Acquisition Window (VAW)
1	VRESET	Reset HCOUNT to 0
2	ENVLOAD	FEN Mask, enable vertical load
3	IRQ	CTAB Interrupt
4	GPV0	General purpose vertical function 0
5	GPV1	General purpose vertical function 1
6	GPV2	General purpose vertical function 2
7	GPV3	General purpose vertical function 3

### 1.3.1 VCTAB Bits

#### VSTART

This bit marks the start of the Vertical Acquisition Window, VAW. Lines will be acquired only while the VAW is active. A 1 in this bit tells the Acquisition Engine to start acquiring lines. In other words, this bit is used to position the first line of the frame acquired. Usually this is located with respect to the FEN signal.

#### VRESET

This bit will reset the VCOUNT to 0.

#### IRQ

When this bit is 1, and the CTAB interrupt is emitted from the board.

#### ENVLOAD

This bit controls how VCOUNT reacts to the assertion of the vertical sync signal, FEN. If this bit is 1, then VCOUNT will jump (load) to the 8000h when FEN asserts. If this bit is 0, then assertion of FEN will be ignored.

#### GPV0, GPV1, GPV2, GPV3

These bits are used to control I/O signals. They are combined with GPH0, GPH1, GPH2 and GPH3 respectively to create the CT0, CT1, CT2 and CT3 signals.



### 1.3.2 VCOUNT Flow Control

#### **VCOUNT Increments**

VCOUNT is incremented every line. To be precise, the end of every HAW increments the VCOUNT. The end of HAW is controlled by the horizontal acquisition engine. The granularity of the VCTAB is 1line time and all events happen on a line boundary.

There are only two instances when we VCTAB is not incremented. The first instance is when VCOUNT reaches 0000h, "Stop at Zero" case. The other instance is when VCOUNT reaches 7FF0h, the "Vertical Stick" case. Both of these cases are described below.

#### **VCOUNT Stops at 0**

Usually, VCOUNT will reach 0 because of a VRESET signal or because it has been automatically reset at the end of VAW. After VCOUNT is reset, there are two programmable options:

- VCOUNT keeps on counting.

- VCOUNT stays at zero until TRIGGER is asserted.

The selection between the two options is done by the VCNT\_RLS\_ZERO bitfield. This bits controls how the VCOUNT can be released from zero.

#### **VCOUNT Sticks at 7FF0h**

In some cases, it is desirable to hold the VCOUNT at a fixed point (in this case, the address 7FF0h) for an indeterminate amount of time, until the FEN signal is asserted. This is stick point is separate and independent from the stop a 0 control. The vertical stick point is controlled by the VCNT\_RLS\_STK bitfield.

#### **VCOUNT Jumps to 8000h**

VCOUNT will be loaded with the value 8000h by the assertion of FEN, if ENVLOAD is 1 when the assertion occurs. FEN usually marks the start of first valid line in the frame. The VSTART bit (controls the VAW) can be placed starting at address 8000h. By placing VSTART later then 8000h, an delay is introduced between FEN and the first line of the frame. This is necessary with some cameras as the first lines after FEN may not be valid. This delay can also be used to acquire a sub-window (ROI) out of the camera's total image output.

#### **VCOUNT resets to 0**

This logic is used to reset VCOUNT to zero, and thus start the vertical cycle over again. VCOUNT can be reset from several sources, according to VCNT\_RST bitfield. However, if VCOUNT reaches a location where the VRESET bit is set, VCOUNT always resets.

VCOUNT will also reset to 0 if it reaches the end of the VCTAB, at 20000h.

### 1.3.3 The VCTAB I/O Functions

There are four General Purpose Horizontal bits in the VCTAB: GPV0, GPV1, GPV2 and GPV3. These are combined with the four General Purpose Horizontal bits to create four independent internal signal CT0, CT1, CT2 and CT3. These signals are combined as follows:

CT0 = GPV0 AND GPH0

CT1 = GPV1 AND GPH1

CT2 = GPV2 AND GPH2

CT3 = GPV3 AND GPH3

The "AND" means that the bits are ANDed together using the logic of an AND gate. In other words, if both the horizontal and vertical bits are 1, then the corresponding CT will be 1 (high). Otherwise the CT will be 0 (low).

Any of the CT's (CT0 to CT3) can be steered to any of the Camera Controls signals (CC1 to CC4) (on the CL connectors) and to the GPOUTs (GPOUT0 to GPOUT6), on the I/O connector. Thus any of these general purpose CTAB bits GPH0 to GPH3 and GPV0 to GPV3 can control any of the I/O outputs on the board.

## 1.4 Understanding How the CTABs Relate to External Devices

The previous sections discuss how the CTABs work internally in the BitFlow frame grabbers. However, even more important is how the CTABs relate to the external devices such as cameras, strobes and other hardware devices. This section covers these topics in details.

### 1.4.1 Relating the CTABs to the Camera's Syncs

#### **CTABs and the Camera's Line and Frame Sync Signals (LEN and FEN)**

Recall that the LEN and FEN signals can modify how the HCOUNT and VCOUNT behave. Specifically, the assertion of LEN can force HCOUNT to jump to 2000h and the assertion of FEN can force VCOUNT to jump to 8000h. This behavior permits the CTABs to be synchronized to the camera. It allows the board to modify which is the first pixel in the line to be acquired, and which is the first line in the frame to be acquired. However, it is important to understand that these behaviors are optional, and need only be used when one of these special features is required. See also Figure 1-9 and Figure 1-10 for illustrations.

The bits HAW\_START and VAW\_START control whether the acquisition window is derived from the CTABs or directly from the LEN and FEN signals. Note that the horizontal and vertical behaviors can be independently controlled. Regardless of where the acquisition windows are derived, the CTABs can either jump or not jump based on the sync signals depending on the ENHLOAD/ENVLOAD CTAB bits and the HCNT\_LD/VCNT\_LD bits.

There are a few reasons why it might be desirable to modify which is the first pixel and/or first line acquired. One situation is when the camera output sync signals that are not actually synchronized to the active pixels. Some cameras have a few dead pixels at the start of every line and/or dead lines at the start of every frame, these can be discarded using these methods. Another situation is when a Region of Interest (ROI), which is a sub-window of the full frame is needed.

#### **Using the CTABs to Control the Location of First Active Pixel**

In order to modify which is the first pixel in the line capture, the following settings must be made:

- Set Bit HAW\_START = 0 - Cause Horizontal Acquisition Window (HAW) to be controlled from CTAB instead of LEN directly.

- Program CTAB bit ENHLOAD = 1 from 0 to 8000h - Causes HCOUNT to jump to 2000h upon assertion of LEN

- Program CTAB bit HSTART = 1 at location of first pixel - Locates first pixel acquired, at 2000h first pixel acquired is first pixel after LEN asserts, at 2001h, first pixel acquired is 8th pixel after LEN asserts, etc. Recall the HCOUNT increments every 8 pixel clocks. Note that the TRIM bits can be used if fewer than 8 pixel granularity is needed.

### Using the CTABs to Control the Location of First Active Line

In order to modify which is the first line in the frame capture, the following settings must be made:

- Set Bit VAW\_START = 0 - Cause Vertical Acquisition Window (VAW) to be controlled from CTAB instead of FEN directly.
- Program CTAB bit EVHLOAD = 1 from 0 to 20000h - Causes VCOUNT to jump to 8000h upon assertion of FEN
- Program CTAB bit VSTART = 1 at location of first line - Locates first line acquired, at 8000h first line acquired is first line after FEN asserts, at 8001h, first line acquired is 2nd line after FEN asserts, etc.

## 1.4.2 Driving the Camera with the CTAB waveforms

Many cameras can be driven with an external sync signal. The CTABs can be used to generate these signals. Since the CTABs are fully programmable, very complex waveforms can be generated if needed. The waveforms can be generated based on the system's horizontal timing, vertical timing or both. Up to four independent waveforms can be produced simultaneously.

Both the horizontal waveforms and the vertical waveforms can either be free-running or can be triggered by an external signal. The horizontal CTAB can be controlled by the encoder input signal, and the vertical CTAB can be controlled by the trigger input signal.

The waveform output can assert immediately after the encoder/trigger or a programmable delay can be added.

### Driving a Line Scan Camera from the HCTAB in Free-Running Mode

In order to drive a line scan camera with a pulse from the CTABs, the following settings must be made:

- Program the pulse into GPH0 - program the start and size of the pulse, see the camera requirements for min/max pulse width.
- Program GPV0 to 1 from 0 to 20000h - Since the output signal CT0 is create from GPH0 AND GPV0, we must program GPV0 to always be 1 regardless of the location of VCOUNT.
- Program CCx\_CON to steer CT0 to the correct CCx - Most cameras require the external driving pulse to be on CC1. In this case, program CC1\_CON to 0, which steers CT0 to CC1. However, if the camera requires the signal on CC2, then CC2\_CON must be programmed.

Assuming the above conditions are met, the waveform shown in Figure 1-2 will be output from the board.

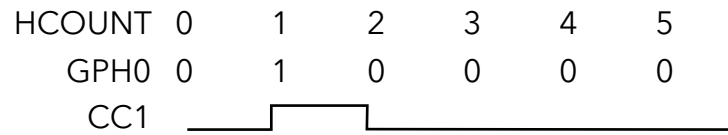


Figure 1-2 Driving a Line Scan Camera

In the case, a short pulse is emitted to the camera as soon as HCOUNT increments past 0. Usually this causes the camera to dump another line, denoted by asserting LEN, the board captures this line, then resets HCOUNT to zero, starting the cycle over again. In this case, the camera is run as fast as possible, the line rate is limited only by the maximum speed the camera is capable of. If slower line rate is needed, the HRESET CTAB bit can be used to reset the HCOUNT at a point in time later than the end of the of the line. In this case, the location of the HRESET bit can control the line rate. See the register HCNT\_RST for more information.

### Driving a Line Scan Camera from the HCTAB in Using an Encoder

In order to drive a line scan camera with a pulse from the CTABs, where the timing is based on an external encoder, the following settings must be made:

- Program the pulse into GPH0 - program the start and size of the pulse, see the camera requirements for min/max pulse width.
- Program GPV0 to 1 from 0 to 20000h - Since the output signal CT0 is create from GPH0 AND GPV0, we must program GPV0 to always be 1 regardless of the location of VCOUNT.
- Program CCx\_CON to steer CT0 to the correct CCx - Most cameras require the external driving pulse to be on CC1.
- Program the bit HCNT\_RLS\_ZERO to 1 - this makes HCOUNT stay at zero until an encoder signal is asserted.

Assuming the above conditions are met, the waveform shown in Figure 1-3 will be output from the board.

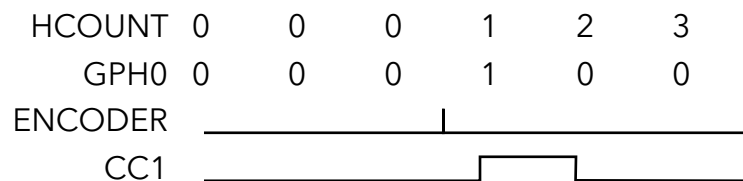


Figure 1-3 Driving a Line Scan Camera Using an Encoder

In the case, HCOUNT stays at zero until the encoder asserts. Once it does, a short pulse is emitted to the camera as soon as HCOUNT increments to 1. Again, this pulse tells the camera to output a line, once the line is capture, the boards resets HCOUNT to zero, where it waits the next encoder pulse. This means the line rate is controlled by the encoder frequency.

### Driving an Area Scan Camera from the VCTAB in Free-Running Mode

In order to drive an area scan camera with a pulse from the CTABs, the following settings must be made:

- Program the pulse into GPV0 - program the start and size of the pulse, see the camera requirements for min/max pulse width.
- Program GPH0 to 1 from 0 to 8000h - Since the output signal CT0 is create from GPH0 AND GPV0, we must program GPH0 to always be 1 regardless of the location of HCOUNT.
- Program CCx\_CON to steer CT0 to the correct CCx - Most cameras require the external driving pulse to be on CC1. In this case, program CC1\_CON to 0, which steers CT0 to CC1. However, if the camera requires the signal on CC2, then CC2\_CON must be programmed.

Assuming the above conditions are met, the waveform shown in Figure 1-4 will be output from the board.

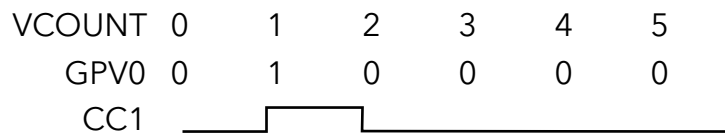


Figure 1-4 Driving an Area Scan Camera

In the case, a short pulse is emitted to the camera as soon as VCOUNT increments past 0. Normally the pulse tells the camera to dump a frame. Once the frame is capture fully by the board, VCOUNT is automatically reset, and the cycle starts over. This will run the camera as fast as is possible. In order so slow things, instead of automatically resetting VCOUNT, it can be reset based on the VRESET bit. See the register VCNT\_RST for more information.

### Driving an Area Scan Camera from the VCTAB in Using a Trigger

In order to drive an area scan camera with a pulse from the CTABs, where the timing is based on an external trigger, the following settings must be made:

- Program the pulse into GPV0 - program the start and size of the pulse, see the camera requirements for min/max pulse width.
- Program GPH0 to 1 from 0 to 8000h - Since the output signal CT0 is create from GPH0 AND GPV0, we must program GPH0 to always be 1 regardless of the location of HCOUNT.

Program CCx\_CON to steer CT0 to the correct CCx - Most cameras require the external driving pulse to be on CC1.

Program the bit VCNT\_RLS\_ZERO to 1 - this makes HCOUNT stay at zero until an encoder signal is asserted.

Assuming the above conditions are met, the waveform shown in Figure 1-5 will be output from the board.

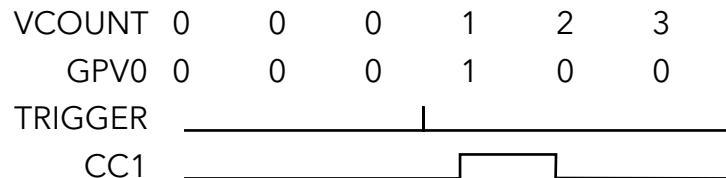


Figure 1-5 Driving a Area Scan Camera Using a Trigger

In the case, VCOUNT stays at zero until the trigger asserts. Once it does, a short pulse is emitted to the camera as soon as VCOUNT increments to 1. This normally results in the camera dumping a frame, by asserting FEN. After the frame is captured, VCOUNT is automatically reset to zero in order to wait for the next trigger. In this case, the frame rate is controlled by the trigger frequency.

### Controlling the Exposure Time of an Area Scan Camera from the VCTAB

Many camera have the ability to have their exposure time controlled by the duration of an the pulse. The CTABs can easily support these cameras because the waveforms they produce are fully programmable. This applies to both area scan and line scan cameras, though here we show an example for an area scan camera.

In order to drive an area scan camera with a pulse from the CTABs, where the width of the pulse controls the exposure time, the following settings must be made:

Program the pulse into GPV0 - program the start and size of the pulse to get the desired exposure. See the camera requirements for min/max pulse width.

Program GPH0 to 1 from 0 to 8000h - Since the output signal CT0 is create from GPH0 AND GPV0, we must program GPH0 to always be 1 regardless of the location of HCOUNT.

Program CCx\_CON to steer CT0 to the correct CCx - Most cameras require the external driving pulse to be on CC1.

Assuming the above conditions are met, the waveform shown in Figure 1-6 will cause the camera to exposure for a short period, then dump a frame.

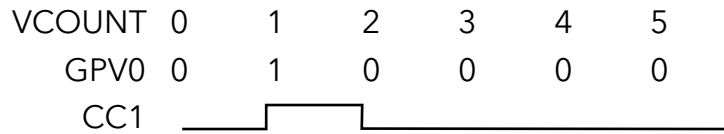


Figure 1-6 Driving an Area Scan In Pulse Width Mode With a Short Exposure

In Figure 1-7, a longer section is programmed into GPV0, causing the camera to expose for a longer period of time (in this case, 100 entries).

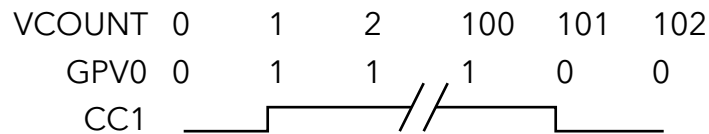


Figure 1-7 Driving an Area Scan In Pulse Width Mode With a Long Exposure

### 1.4.3 Driving More Than One Device From the CTABs

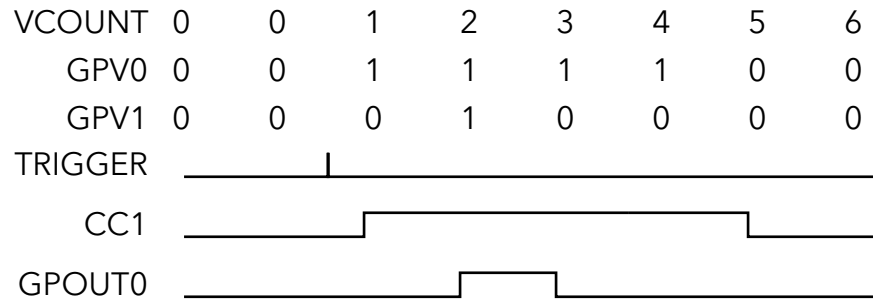
In many real world situations, more than one device needs to be controlled. In this example we set the board up to wait for a trigger. Once the trigger is asserted, the CTABs start the exposure on the camera, fire a strobe, then end the camera's exposure period.

In order to support this situation, the following steps must be taken:

- Program the exposure pulse into GPV0 - program the start and size of the pulse to get the desired exposure. See the camera requirements for min/max pulse width.
- Program the strobe pulse into GPV1 - program the start and size of the pulse depending on the needs of the strobe.
- Program GPH0 to 1 from 0 to 8000h - Since the output signal CT0 is create from GPH0 AND GPV0, we must program GPH0 to always be 1 regardless of the location of HCOUNT.
- Program GPH1 to 1 from 0 to 8000h - Since the output signal CT1 is create from GPH1 AND GPV1, we must program GPH1 to always be 1 regardless of the location of HCOUNT.
- Program CC0\_CON to steer CT0 to the correct CC1 - Most cameras require the external driving pulse to be on CC1.
- Program GPOUT0\_CON to steer CT1 to the correct GPOUT0 - The GPOUT0 output will be used to control the strobe.



Figure 1-8 shows the timing diagram for this setup.



**Figure 1-8 Controlling Both a Camera and a Strobe from the CTABs**

In this case, the VCTAB is waiting for the trigger with VCOUNT at 0. Then the trigger asserts, VCOUNT starts incrementing and GPV0 goes to 1, causing CC1 to go high, causing the camera to start exposing. When VCOUNT equals 2, GPV1 goes to 1, causing GPOUT0 to go high, causing the strobe to fire. At VCOUNT equal to 3, GPV1 goes low, ending the strobe pulse. At VCOUNT equal to 5, GPV0 goes to 0, ending the exposure time, and causing the camera to dump the newly acquired frame.

## 1.5 CTAB Complete Example 1

Figure 1-9 shows example of the timing of HCTAB.

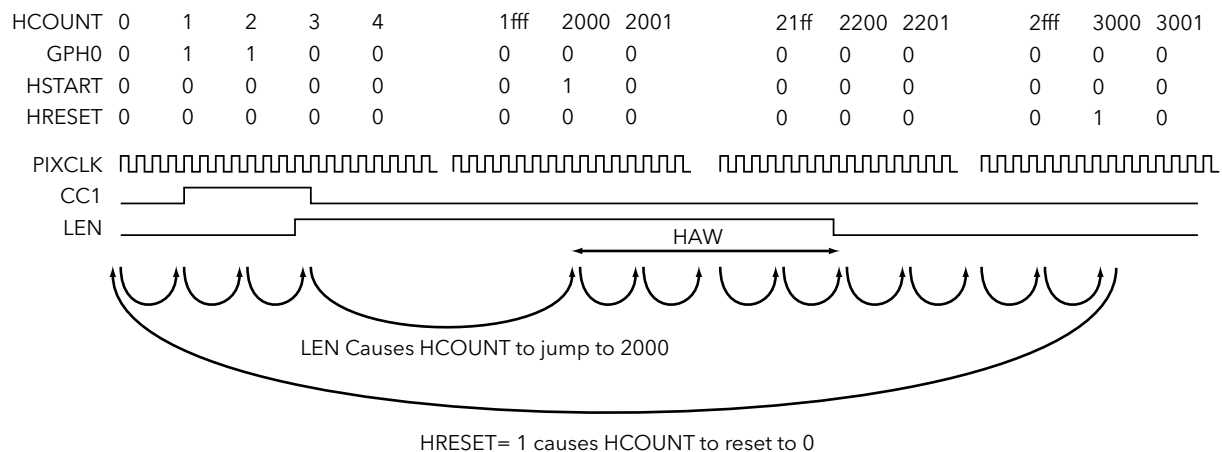


Figure 1-9 Complete HCTAB Example

In this illustration, designed more like a timing diagram, the HCTAB is shown horizontally. This view allows the table to be shown on the same axis as the other important signals.

In this example, HCOUNT starts at zero. The pixel clock (PIXCLK) is shown running, and incrementing HCOUNT every 8 clocks. The pixel clock waveform is representative. The first thing that happens at HCOUNT = 1 is that GPH0 goes to 1, this causes the camera control signal CC1 to go high. This signal stays high for two HCOUNT addresses. Next, the camera sends the board a LEN signal, tell the board that it is ready to read out a line of data. The LEN signal causes HCOUNT to jump to 2000h (ENHLOAD is assumed to be 1 everywhere). At 2000h, HSTART is 1, so the board immediately acquired pixels as defined by the HAW (ACPL register). Even the LEN deasserts, the board continues to increment HCOUNT, until HRESET goes to 1, at which point HCOUNT is reset to zero.

## 1.6 CTAB Complete Example 2

Figure 1-10 shows another example of the timing of HCTAB.

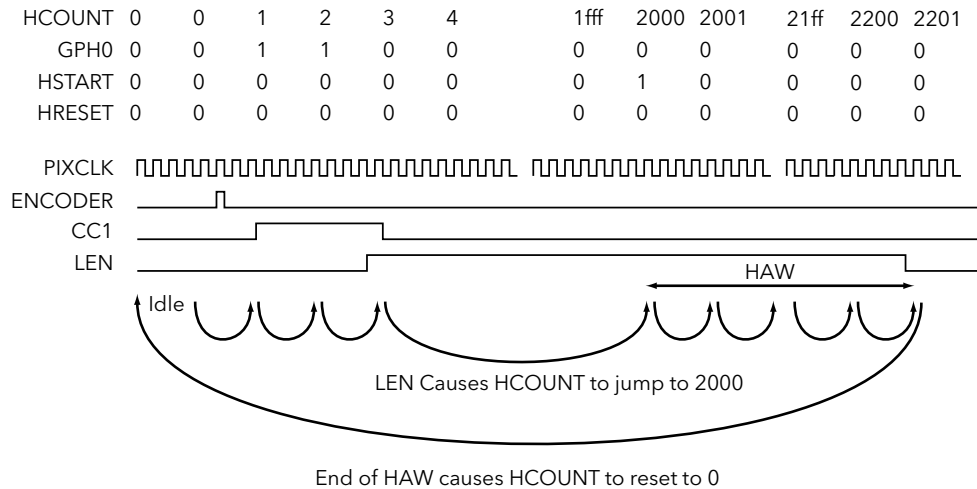


Figure 1-10 Complete HCTAB Example 2

This example differs from the previous in a few important ways. Primarily, this example illustrates the case when `HCNT_RLS_ZERO = 1`, that is, the case where the HCTAB waits for an encoder signal before counting up from zero. The board will wait indefinitely for this signal.

Another change from the previous example is that the HCOUNT here is shown resetting automatically after the end of the line. In the previous example, HCOUNT was not reset until HRESET was a 1, well after the line ended. The different modes are controlled by the `HCNT_RST` register.

The flow of this example goes as follows. Initially, HCOUNT is sitting a 0, it will wait here forever until an encoder signal comes into the board. Once this does, HCOUNT starts counting up, incrementing every 8 pixel clocks. At location 1, you can see the GPH0 is a 1, here it is tied to CC1, and tells the camera to dump a new line. The camera responds with a LEN signal, causing the HCOUNT to jump to 2000h. At this point, HSTART is a 1, telling the board to start acquiring pixels. The board acquires as many pixels as are programmed into the ACPL register, once this is done, HCOUNT is automatically reset to 0. The cycle then repeats next encoder pulse.

It is important to understand that these examples are just a few of the many ways the CTABs can be programmed. The CTABs are very flexible can accommodate almost any camera and almost any situations.

## 1.7 Programming the CTABs using CamVert

CamVert is BitFlow's low level camera configuration file editor. This editor exposes all of the controls in a camera file, but is perhaps not as easy to use the other camera editor CamEd. See Section 1.8 for information on how to CTABs using CamEd.

The CTABs encompass a very large amount of memory, and if the camera file contain a copy of all of this memory, the file would be very large. Fortunately, the information contained in the CTABs is very sparse, allowing for significant file size reduction if the CTABs are compressed. The camera configuration files contained a Run Length Encoded (RLE) version of the CTABs. The RLE algorithm only records changes in the CTAB memory as you walk from one end to the other. Because there usually are very few changes to the values, the RLE CTAB is very small.

In CamVert, the text editor displays the CTABs like this:

```
CTAB 0x00000000 0x00000414
CTAB 0x00000001 0x00001414
CTAB 0x00000005 0x00000414
CTAB 0x00002000 0x00000415
CTAB 0x00002001 0x00000414
CTAB 0x00008000 0x00000500
CTAB 0x00008001 0x00000400
CTAB 0x00020000 0x00000000
```

The above lists both the vertical and the horizontal CTABs in a single table. Only the addresses (first column) where a bit changes are shown. All other addresses are the same between one address and the next. The value at each address is shown in the second column. The vertical CTAB is 8 bits and the horizontal CTAB is 8 bits, but the data column is 32-bit to support future expansion.

This format above is quite difficult to understand, much less edit. For this reason, CamVert has a CTAB editor, which supports simple editing of the CTABs, see the next section to understand how this works.

### 1.7.1 The CamVert CTAB Editor

CamVert provides a simple dialog to make editing CTABs simple and fast. This CTAB editor is shown in Figure 1-11.

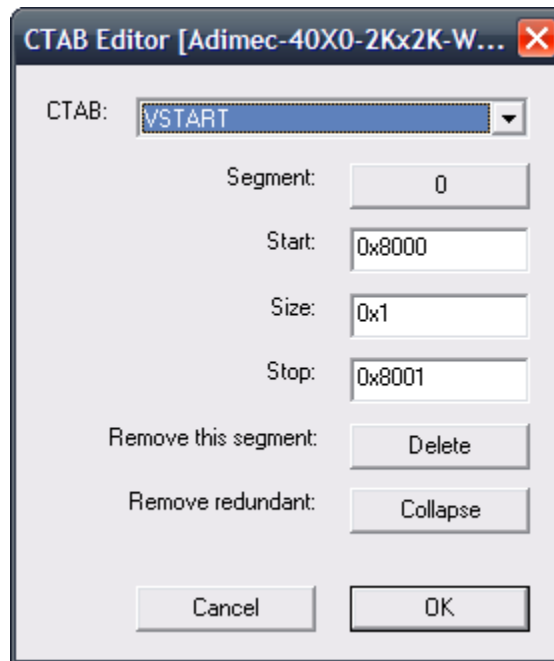


Figure 1-11 CamVert CTAB Editor

Table 1-5 shows all of the elements of this dialog and their purpose.

Table 1-5 CamVert CTAB Editor Controls

Item	Purpose
CTAB	Select the CTAB bit to edit. Only one bit at a time is show. Note that this dialog supports editing of both the HCTAB and the VCTAB.
Segment	This dialog edits "segments". A segment is a contiguous section of 1s in a of a single bit in CTAB memory. Note that any memory not in a segment will be set to 0s.
Start	The address of the start of this segment.

Table 1-5 CamVert CTAB Editor Controls

Item	Purpose
Size	The number of memory locations that this segment takes up.
Stop	The address of the end of this segment. Note the Start, Size and Stop are three variables that describe a segment which only has two parameters. So there are really only two degrees of freedom here. However, sometimes it is easier to use Start and Stop and some time it is easier to use Start and Size, so all three are provided.
Remove this segment	Deletes the current segment.
Remove redundant	Using this dialog, segments can be created that overlap. This control collapses overlapping segments into a single segment.

### 1.7.2 Programming a Simple Waveform

Lets look at how dialog works. Figure 1-12 shows an example waveform, including the HCOUNT address, the values for GPH0 at these addresses, and the resulting output on the CC1 pin.

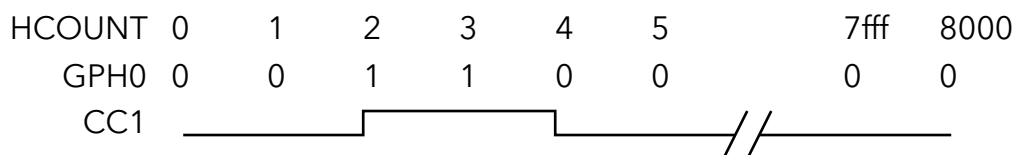


Figure 1-12 Example CTAB Waveform 1

This is a simple CTAB waveform, it has a single segment. The segment starts a address 2 and ends at address 4. The size of the segment is 2. In CamVert, this will be represent like this:

Segment: 0  
 Start: 2  
 Size: 2  
 Stop: 4

If the segment is now modified using the CamVert CTAB editor to the following values:

Segment: 0  
 Start: 1  
 Size: 4  
 Stop: 5

The resulting waveform will be as shown Figure 1-13.

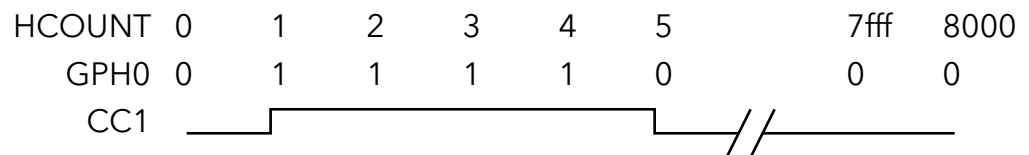


Figure 1-13 Example CTAB Waveform 2

### 1.7.3 Programming a More Complex Waveform

Figure 1-14 shows a slightly more complex example waveform. It is an inverted version of the waveform shown in Figure 1-13. Recall that the CamVert CTAB dialog displays the segments that make up a wave form, where a segment is defined as a contiguous section of 1s. The waveform below consists two such segments, make the representation in the CamVert editor slightly more complicated. The wave form must be broken into two segments. The first starts a zero and goes for 1 entry, the second starts at 5 and goes to the end of the table. As you can see, CamVert is not describing what most people would describe as a “asserted low pulse”. CamVert is describing the sections of memory where the signal is high.

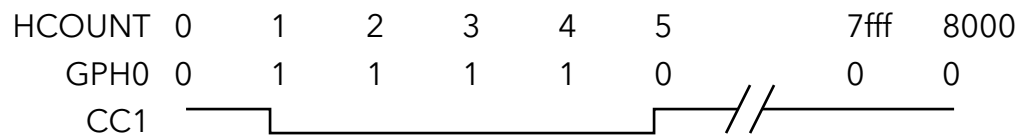


Figure 1-14 Example CTAB Waveform 3

Here is how this waveform would be represent:

Segment: 0  
 Start: 0  
 Size: 1  
 Stop: 1

Segment: 1  
 Start: 5  
 Size: 7ffb  
 Stop: 8000

If the segment is now modified using the CamVert CTAB editor to the following values:

Segment: 0  
 Start: 0  
 Size: 2  
 Stop: 2

Segment: 1  
 Start: 4  
 Size: 7ffc  
 Stop: 8000

The resulting waveform will be as shown Figure 1-15.



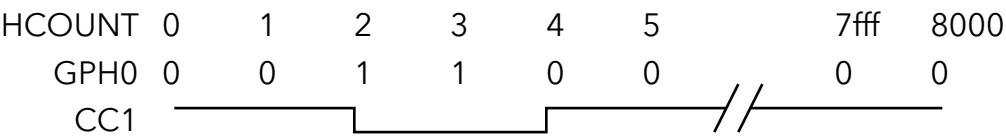
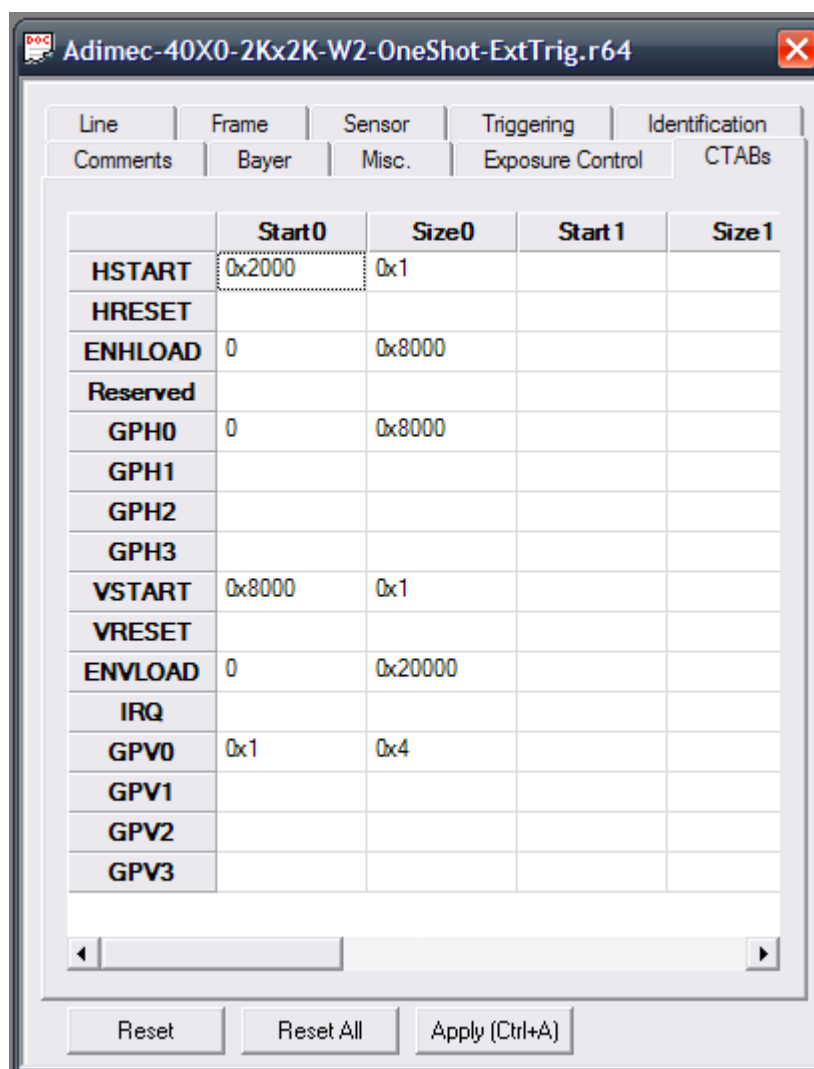


Figure 1-15 Example CTAB Waveform 4

## 1.8 Programming the CTABs using CamEd

CamEd is BitFlow's easy to use, high-level camera file editor. While not every parameter of a camera configuration file can be modified via CamEd, all of the most important parameters can. Furthermore, CamEd is camera-parameter-centric, this means the parameters are represented in a way similar to how they are in camera manual literature. This focus makes it very easy to modify camera configuration files.

CTABs are modified in CamEd using the CTAB page, shown in Figure 1-16.



	Start0	Size0	Start1	Size1
<b>HSTART</b>	0x2000	0x1		
<b>HRESET</b>				
<b>ENHLOAD</b>	0	0x8000		
<b>Reserved</b>				
<b>GPH0</b>	0	0x8000		
<b>GPH1</b>				
<b>GPH2</b>				
<b>GPH3</b>				
<b>VSTART</b>	0x8000	0x1		
<b>VRESET</b>				
<b>ENVLOAD</b>	0	0x20000		
<b>IRQ</b>				
<b>GPV0</b>	0x1	0x4		
<b>GPV1</b>				
<b>GPV2</b>				
<b>GPV3</b>				

Figure 1-16 The CamEd CTAB Editor

This editor is similar to the one in CamVert in that it shows a run length encoded version of each CTAB bit. However, all of the bits are shown simultaneously. Further, if a bit has multiple segments, all the segments are shown at the same time. Also, in order to keep things less cluttered, only the Start and Size of each segment is shown. All of these changes make editing the CTABs in CamEd quicker and simpler.

### 1.8.1 CamEd CTAB example 1

Understanding how CTAB segments are represented in CamEd is similar, though simpler, than CamEd. Here are a few examples (using the same waveforms as the Section 1.7).

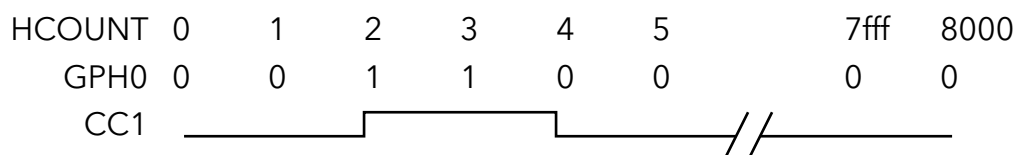


Figure 1-17 Example CTAB Waveform 5

In CamEd CTAB editor, this waveform will be represented as follows:

Table 1-6 CamEd Waveform 5

	Start0	Size0	Start1	Size1
GPH0	0x2	0x2		

Now, let's modify the waveform in CamEd CTAB editor by changing the Start0 and Size0 as follows.

Table 1-7 CamEd Waveform 6

	Start0	Size0	Start1	Size1
GPH0	0x1	0x4		

The resulting waveform is shown in Figure 1-18.

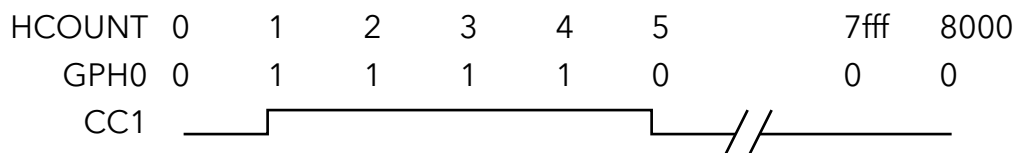


Figure 1-18 Example CTAB Waveform 6

## 1.8.2 CamEd CTAB Example 2

Next lets look at the “asserted low” wave form:

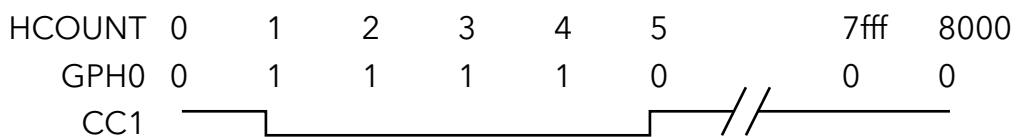


Figure 1-19 Example CTAB Waveform 7

In CamEd CTAB Editor

Table 1-8 CamEd Waveform 7

	Start0	Size0	Start1	Size1
GPH0	0x0	0x1	0x5	0x7ffb

Now lets say we want to make the asserted low pulse shorter and start later. In this case we have to modify Size0 (the size of the first asserted-high pulse) and Start1 (the start of the second asserted-high pulse). The changes in CamEd will look as follows:

Table 1-9 CamEd Waveform 8

	Start0	Size0	Start1	Size1
GPH0	0x0	0x2	0x4	0x7ffc

The resulting waveform is shown in Figure 1-20.

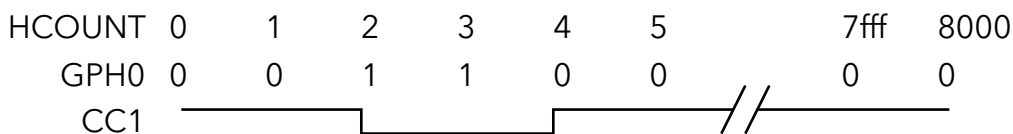


Figure 1-20 Example CTAB Waveform 8

## 1.9 Programming the CTABs from Software

The BitFlow SDK provides function to fully program and access all of the CTAB features. Programming the CTABs from software is quite easy, generally only one function is needed, CiCTabFill(). Please see the SDK Reference manual for complete documentation of this function. However, the function works similar to the CTAB editors seen in CamVert and CamEd.

Briefly, the function prototype is:

```
CiCTabFill(Bd Board, BFU32 Index, BFU32 NumEntries, BFU16
Mask, BFU16 Value)
```

Where:

Board - Handle to the board

Index - Start value for the section to be added

NumEntries - The size of the section to be added

Mask - The CTAB bit to add the section to, more than one bit can be written at the same time.

Value - The value of the section to be added. The Value is combined with the mask to the desired value 0 or 1 to the desired bit(s).

This function can write a contiguous section of 1s or 0s to any CTAB bit. More than one bit can be written with one call, but all bits written will have the same start and end address. Writing 1s when the memory is already a 1 causes no harm, similarly writing 0s to memory that is already 0 is also OK. Only the parts of memory from Index to Index + NumEntries are effected, no other memory locations are modified.

The the board is first opened and initialized, the CTAB memory is programmed according to the current camera configuration file. If you want to change a CTAB bit, it usually it is safest to clear out the memory (set to 0), then write the desired section of 1s.

### 1.9.1 Programming Example 1

First, lets program a simple waveform, shown in Figure 1-21.

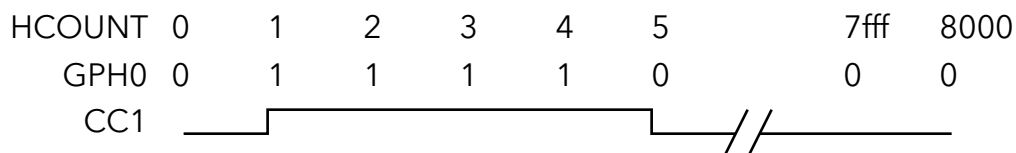


Figure 1-21 Example CTAB Waveform 9

To program this waveform, the code below is all that is required. Note that we are assuming that memory has already been cleared to 0.

```
CiCTabFill(hBoard, 0x0001, 0x0004, R64HCTabGPH0, 0xffff);
```

Where:

hBoard - The handle to the board  
 0x0001 - The start address  
 0x0002 - The size  
 R64HCTabGPH0 - The predefined mask that selects just the GPH0 bit  
 0xffff - Set all bits to 1, however, this is combined with the mask so only GPH0 is modified.

## 1.9.2 Programming Example 2

Next lets assume the memory is programmed as above (i.e. as shown in Figure 1-21), and that we want to modify the pulse to a shorter one as shown in Figure 1-22.

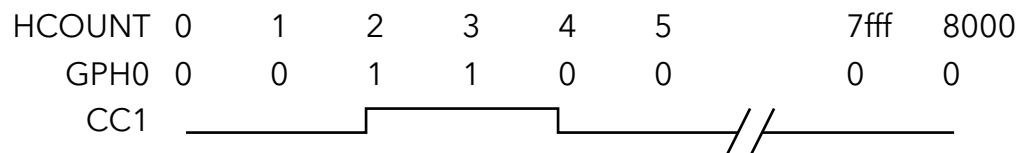


Figure 1-22 Example CTAB Waveform 10

This new pulse is shorter than the old pulse. There are a few different ways to create this shorter pulse. The first method, which is the simplest, is to erase the entire table, when write back the new section:

```
// Erase GPH0 for the entire CTAB
CiCTabFill(hBoard, 0x0000, 0x8000, R64HCTabGPH0, 0x0000);
// write new section
CiCTabFill(hBoard, 0x0001, 0x0002, R64HCTabGPH0, 0xffff);
```

The problem with this code it is quite inefficient as it is clearing a large section of memory that does not really have to be cleared. A more efficient solution would be to use variables that remember the last section written, and the only erase the section that has been written to 1s. In the following code snippet, two variable are used to store the section that was previously written.

```
// Erase the previous section of GPH0
CiCTabFill(hBoard, OldStart, OldSize, R64HCTabGPH0, 0x0000);
// write new section
CiCTabFill(hBoard, NewStart, NewSize, R64HCTabGPH0, 0xffff);
// Save the new section for next time in needs to be erased
```

```
OldStart = NewStart;
OldSize = NewSize;
```

An even more efficient method would be to erase or write back only the locations that change from the old section to the new section. It is left up to the reader to code this technique, however, it should be fairly straight forward.

### 1.9.3 Programming Example 3

In this example, let's program the waveforms used in Section 1.4.3. The waveforms are shown again in Figure 1-23.

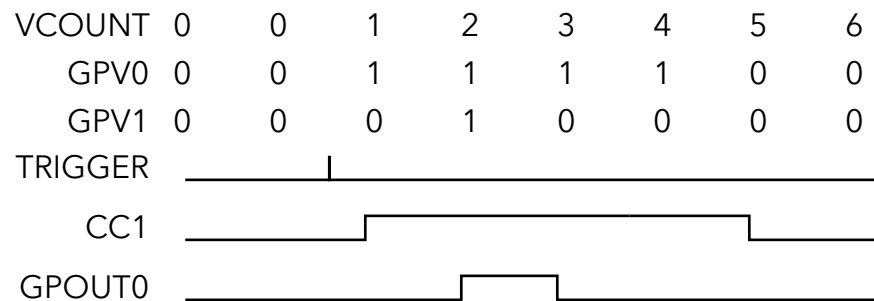


Figure 1-23 Example CTAB Waveform 11

In this case we have to program two pulses in two different CTAB bits. Here is the code:

```
// Program GPV0
CiCTabFill(hBoard, 0x0001, 0x0004, R64HCTabGPV0, 0xffff);
// Program GPV1
CiCTabFill(hBoard, 0x0002, 0x0001, R64HCTabGPV1, 0xffff);
```

### 1.9.4 Programming Example 4

Of course, programming two pulses in the same CTAB bit is as simple as calling `CiCTabFill()` twice, one call per pulse. An example waveform is shown again in Figure 1-24.

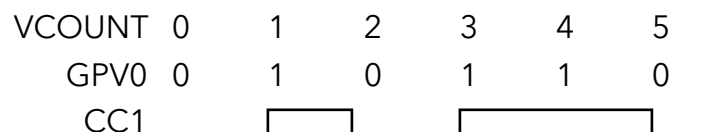


Figure 1-24 Example CTAB Waveform 12

In this case we have to program two pulses in the same different CTAB bit. Here is the code:

```
// Program GPV0 pulse 1
CiCTabFill(hBoard, 0x0001, 0x0001, R64HCTabGPV0, 0xffff);
// Program GPV1 pulse 2
CiCTabFill(hBoard, 0x0003, 0x0002, R64HCTabGPV0, 0xffff);
```



## 1.10 Understanding CTAB Time Intervals

The CTABs HCOUNT and VCOUNT are run by the connected camera's timing. Therefore in order to understand how long a certain event will take in the CTABs, you must understand the camera's timing parameters. The main two parameters that need to be understood are the pixel clock frequency, and the line rate. The pixel clock drives the horizontal CTAB and the line rate drives the vertical CTAB. While this seems complicated, once you understand the principles, it is very easy to calculate. Note the New Timing Generator (NTG) is run by a fixed frequency clock, and therefore does not suffer from this complication.

### 1.10.1 Calculating Horizontal CTAB intervals

Lets determine the period of the pulse being output to CC1, shown in Figure 1-25.

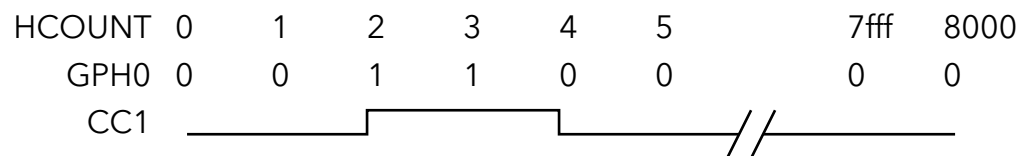


Figure 1-25 Simple HCTAB pulse

Here is the formula use to calculate the duration of this pulse:

$$\text{PulseDuration} = \frac{\text{NumHEntries} \cdot 8}{\text{PixelClockFrequency}}$$

Where:

PulseDuration - The length of the pulse in seconds

NumHEntries - Size of pulse in units of HCTAB entries

PixelClockFrequency - Pixel Clock Frequency in Hertz

The factor of 8 comes from the fact the HCOUNT increments every 8 pixel clocks. Lets assume that the pixel clock is 40 MHz. We see that the pulse takes up two HCTAB entries. The pulse duration there for is 0.4 microseconds.

### 1.10.2 Calculating Vertical CTAB intervals

For the vertical CTAB, lets determine the period of the pulse being output to CC1, shown in Figure 1-26.

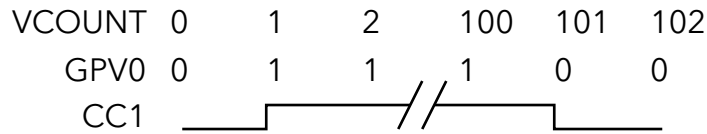


Figure 1-26 Simple VCTAB pulse

Here is the formula use to calculate the duration of this pulse:

$$\text{PulseDuration} = \frac{\text{NumVEntries}}{\text{LineRate}}$$

Where:

PulseDuration - The length of the pulse in seconds

NumVEntries - Size of pulse in units of VCTAB entries

LineRate - Line rate in Hertz

Lets assume that the line rate is 64 KHz. We see that the pulse takes up 100 VCTAB entries. The pulse duration there for is 1.56 milliseconds.

### 1.10.3 More Complicated Timing Calculations

Some timing calculations can be more complicated because HCOUNT and VCOUNT do not always increment by 1. Recall that they can be programmed to jump when LEN or FEN are asserted. This jump can complicate calculation of some timing intervals. The examples above will always be correct because the assumption is that HCOUNT/VCOUNT always increment normally during the duration of the example pulses. However, there are cases when this is not true, and the calculations are slightly different.

To understand these more complicated examples, refer to the timing diagram in Figure 1-9. A calculation of interest might be to figure out what the line rate of this setup is. Recall that in this case, the line rate is being controlled by the frame grabber and not by an encoder.

At first it might seem to make sense to take the range the HCOUNT covers in this diagram (3000h) and plug it into the formal shown in Section 1.10.1. However, this would be incorrect because HCOUNT does not traverse all 3000h HCTAB entries. If you look closely, HCOUNT jumps from location 0003h to location 2000h, skipping all the intermediate locations, when LEN asserts. This jump is essentially instantaneous. After the jump, the line is clocked out the camera, the a further delay occurs from the end of the line until the reset. This extra period is important as this is one way to control the line rate from the board.

Taking all these factors into account, the calculation then becomes:

$$\text{PulseDuration} = \frac{(\text{GPH0Pulse} + \text{HAW} + \text{ResetDelay}) \cdot 8}{\text{PixelClockFrequency}}$$

Where

PulseDuration - Then line period in seconds

GPH0Pulse - Is the length of the pulse in GPH0

HAW - The length of the line read out time

ResetDelay - The length of the time between the end of HAW and the reset

Note: The last three parameters are in units of HCTAB entries

For the case in Figure 1-9, assuming a 40 MHz Pixel clock, the formula becomes:

$$\text{PulseDuration} = \frac{(2 + 200h + 2e00h) \cdot 8}{40 \times 10^6}$$

Which works out to a line rate of 610.20 Hz.

