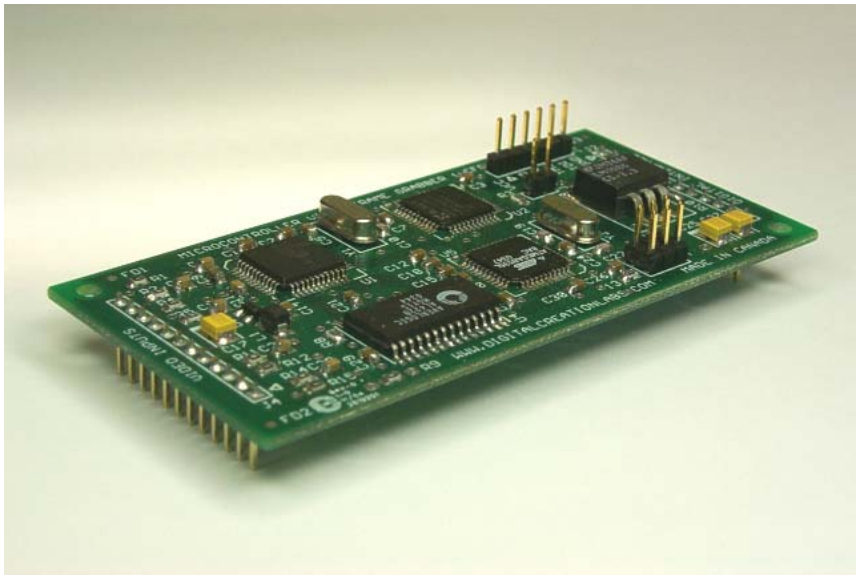
Microcontroller Frame Grabber™ (uCFG™)



Features

- 4 composite analog input video channels with built-in multiplexer
- 4 input channels can instead be used as 2 S-Video inputs for studio quality image capture
- 1 composite analog video output channel for live viewing
- Native support for both PAL & NTSC
- Acquire a full field (720x240 NTSC, 720x272 PAL) of color video data to an on-board image buffer
- Compact YCbCr 4:2:2 image data format (if needed, simple host RGB conversion)
- Selective acquisition of ODD or EVEN field
- Send commands/download image data over a simple LVTTL serial UART interface (or RS-232 with external level shifter) working at up to 230.4 kbps
- Supports 1:1, 2:1, 4:1 and 8:1 compressed image download modes
- Full software flow control allows downloading of image data to limited resources client
- Multiple download of image data stored in buffer with different decimation parameters. Allows quick thumbnail preview, then full resolution download
- Simple ASCII command set
- Frame grab triggered by software command or, if required, through 4 external hardware trigger inputs with programmable polarity
- Complete trigger latch logic allows acquisition of first triggered frame following latch clear, or automatic overwrite. Serial commands allow reading of trigger status, latch, as well as clearing of latch
- All board configuration settings are set through user software and stored in non-volatile EEPROM
- 256 bytes of general purpose nonvolatile EEPROM user storage space accessible through serial port commands
- I^2C bus available for expansion (i.e. Adding digital I/O's, A/D converter, etc...). I^2C bus can be controlled directly through serial port commands
- Compact plug-in daughtercard design can be inserted OEM into a product (two 0.1" single row headers on back side)

Table of Contents

DISCLAIMER: All information in this datasheet is subject to change without notice. Our best efforts have been made to ensure correctness of the data contained in this document, however some errors may still exist. It is therefore the responsibility of the user to verify all the critical parameters and determine suitability for use in their intended application.

Overview

The microcontroller frame grabber (uCFG) is an OEM plug-in module designed to allow the digitization and acquisition of a full color, full resolution (PAL 720 x 272 or NTSC 720 x 240) single video field into a local image buffer. The image data can then be downloaded in its entirety or as a decimated thumbnail over a very simple serial interface controlled by an ASCII command set. The original full color, full resolution, raw image data remains in the field buffer of the uCFG as long as a new image is not grabbed. This allows data to be read out multiple times with different compression ratios or decimation settings with full control over downloaded packet size. A high-speed LVTTL-level (+3.3V) serial UART interface working at up to 230.4 kbps is available for sending commands and downloading image data.

Block Diagram

The uCFG system block diagram is shown in Figure 1. The uCFG provides 4 separate analog video inputs. All inputs accept either NTSC or PAL composite color video signals depending on the board's configuration settings in non-volatile memory. Black & white (RS-170) video signals are also supported. The uCFG can be setup to operate in high quality S-Video mode where the luma and chroma components are separated out into two discrete signals (Y and C). In this mode, two channels of S-Video are available by using input pairs 1&3 and 2&4 respectively.

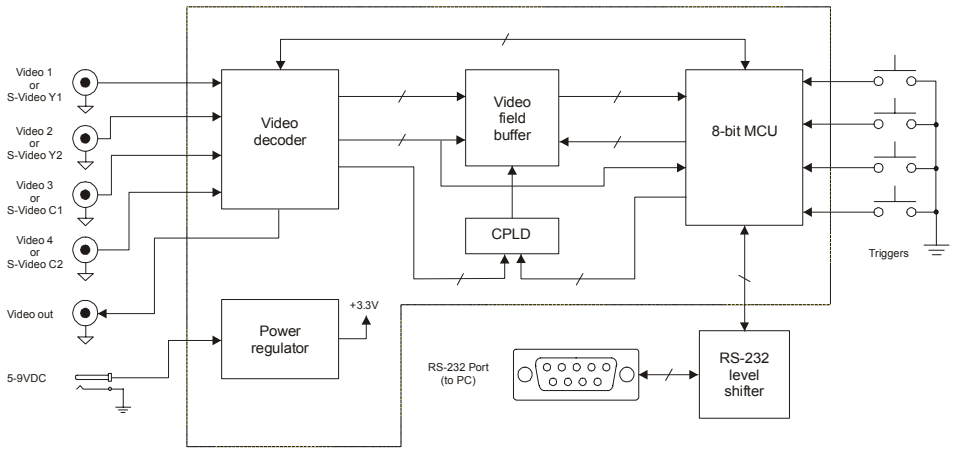


Figure 1: uCFG high level block diagram

The video inputs are fed to an internal video multiplexer used to select the active video channel for digitization. The output of the analog multiplexer is provided at the uCFG's video out terminal for live video preview. This signal is also fed to an internal video analog-to-digital converter (ADC) and is then converted to an industry standard ITU-BT656 4:2:2 digital video stream. The digital video stream is written to a video field buffer under the control of an 8-bit microcontroller (MCU) as well as a complex programmable logic device (CPLD). Once a single field of color video is stored into the buffer, the 8-bit MCU can read out the data and transmit it through its serial port (LVTTL, +3.3V level). These operations are all controlled by a complete ASCII command set.

4 external trigger inputs with programmable polarity are provided to trigger acquisition of a video field from the corresponding video channel. Software triggering is also possible through the ASCII commands.

For serial interface compatibility with full RS-232 levels (to connect directly to a PC), an external level shifter must be added (such as the Sipex SP3232 level shifter). The uCFGEVAL kit includes this level shifter on the motherboard for convenience.

Pin Descriptions

Figure 2 shows a top view of the uCFG board. J2 and J4 represent the main interface signal headers mounted on the back side of the board (for plug-in to motherboard). J3 is a two position header designed to receive a shunt (jumper) to force default board settings. J1 and J5 are factory programming headers for the CPLD and MCU respectively.

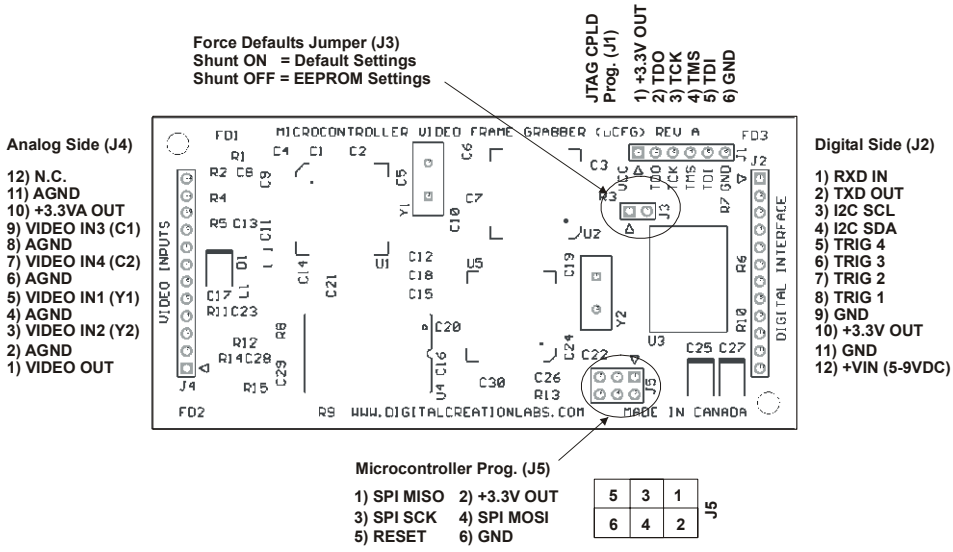


Figure 2: uCFG layout showing different connector pinouts (top view)

J1 Pinout (CPLD Programming Header)

The J1 header is used for factory programming of the on-board CPLD.

1	**+3.3V OUT**	+3.3V analog output (10 mA max)
2	**TDO**	JTAG TDO signal (LVTTL +3.3V)
3	**TCK**	JTAG TCK signal (LVTTL +3.3V)
4	**TMS**	JTAG TMS signal (LVTTL +3.3V)
5	**TDI**	JTAG TDI signal (LVTTL +3.3V)
6	**GND**	GND

J2 Pinout (Digital Side)

The J2 header is mounted at the back of the uCFG board and plugs in to the motherboard. It contains the power and digital interface signals to the uCFG.

1	**RXD IN**	UART RXD input line (LVTTL +3.3V)
2	**TXD OUT**	UART TXD output line (LVTTL +3.3V)
3	**I2C SCL**	I^2C bus SCL line (2.2K internal pull-up resistor to +3.3V)
4	**I2C SDA**	I^2C bus SDA line (2.2K internal pull-up resistor to +3.3V)
5	**TRIG 4**	Trigger input 4 (LVTTL +3.3V and 20K to 50K pull-up to +3.3V)
6	**TRIG 3**	Trigger input 3 (LVTTL +3.3V and 20K to 50K pull-up to +3.3V)
7	**TRIG 2**	Trigger input 2 (LVTTL +3.3V and 20K to 50K pull-up to +3.3V)
8	**TRIG 1**	Trigger input 1 (LVTTL +3.3V and 6K to 8K pull-up to +3.3V on TRIG 1 only)
9	**GND**	GND
10	**+3.3V OUT**	+3.3V digital output (10 mA max)
11	**GND**	GND (+VIN return)
12	**+VIN**	+5-9VDC main power input

J3 (Force Defaults Jumper)

If a shunt is loaded on J3, then the next time the power is cycled to the uCFG, it will assume a default configuration regardless of the EEPROM user settings. These defaults include:

Video Mode: NTSC
Video Mux Mode: Regular composite inputs (4 inputs)
UART Mode: Normal
Triggers: All enabled, active high
Trigger Mode: retriggerable
Baud Rate: 115.2 kbps

This is convenient when setting up a uCFG unit with unknown prior settings. If J3 is not shunted, then the power up settings will be read from EEPROM.

J4 Pinout (Analog Side)

The J4 header is mounted at the back of the uCFG board and plugs in to the motherboard. It contains the analog video signals to the uCFG.

1	**VIDEO OUT**	Live video output composite signal used to view live video channel currently selected (output of video multiplexer)
2	**AGND**	Analog GND (video out return)
3	**VIDEO IN2 (Y2)**	Input 2 composite video signal (Y2 for S-Video mode)
4	**AGND**	Analog GND (video in 2 return)
5	**VIDEO IN1 (Y1)**	Input 1 composite video signal (Y1 for S-Video mode)
6	**AGND**	Analog GND (video in 1 return)
7	**VIDEO IN4 (C2)**	Input 4 composite video signal (C2 for S-Video mode)
8	**AGND**	Analog GND (video in 4 return)
9	**VIDEO IN3 (C1)**	Input 3 composite video signal (C1 for S-Video mode)
10	**+3.3VA OUT**	+3.3V analog output (10 mA max)
11	**AGND**	Analog GND (video in 3 return)
12	**N.C.**	Not connected

J5 Pinout (Microcontroller Programming Header)

The J5 header is used for factory programming of the on-board microcontroller. It normally is not used otherwise. Please note, the trigger inputs TRIG 1-4 share the SPI port pins. Therefore, **when reprogramming the MCU, make sure that all TRIG signals are disconnected to avoid potential damage!**

1	**SPI MISO**	SPI signal (LVTTL +3.3V) *DISCONNECT TRIG 3 INPUT WHEN USED!
2	**+3.3V OUT**	+3.3V analog output (10 mA max)
3	**SPI SCK**	SPI signal (LVTTL +3.3V) *DISCONNECT TRIG 4 INPUT WHEN USED!
4	**SPI MOSI**	SPI signal (LVTTL +3.3V) *DISCONNECT TRIG 2 INPUT WHEN USED!

5	**RESET**	RESET input (LVTTL +3.3V)
6	**GND**	GND

Electrical Specifications

The following represents the electrical operating specifications of the uCFG. **Operating the uCFG outside of these specifications will result in damage and voids the warranty.** Please note that all digital I/O pins nominally operate at LVTTL (+3.3V) logic levels. **Important Note**: the uCFG board does note provide any additional Electro-Static Discharge (ESD) protection on any of its interface pins other then the basic protection provided by the IC's used. If the board is going to be used in a harsh ESD-prone environment, it is recommended to add external ESD protection on each of the main signal inputs/outputs. Contact us for more information.

Symbol	Parameter	Min	Typ	Max	Units
+Vin	Input supply voltage	5.0		9.0	V
VIL1	Input low voltage (SCL, SDA, TRIG n, MISO, RESET)	-0.5		0.99	V
VIH1	Input high voltage (SCL, SDA, TRIG n, MISO)	2.31		3.8	V
VIL2	Input low voltage (RXD, TCK, TMS, TDI)	0		0.8	V
VIH2	Input high voltage (RXD, TCK, TMS, TDI)	2.0		5.5	V
VIH3	Input high (RESET)	2.97		3.8	V
VOL1	Output low voltage (SCL, SDA, MOSI, SCK)	0		0.5	V
VOH1	Output high voltage (SCL, SDA, MOSI, SCK)	2.2		3.3	V
VOL2	Output low voltage (TXD, TDO)	0		0.4	V
VOH2	Output high voltage (TXD, TDO)	2.4		3.3	V
Vi(p-p)	Analog video signal input peak-peak	0.5	0.7	1.4	V
ICC	UCFG current consumption		175		mA
TAMB	Ambient operating temperature	0		70	°C

Mechanical Dimensions

Figure 3 shows the overall mechanical layout and dimensions of the uCFG board. Headers J2 and J4 are mounted on the back side of the PC board to allow plug-in to an external motherboard. Standard 12 pos 0.1" spacing through-hole headers are used. A side view also shows the worst case clearance height both on the top and bottom of the board.

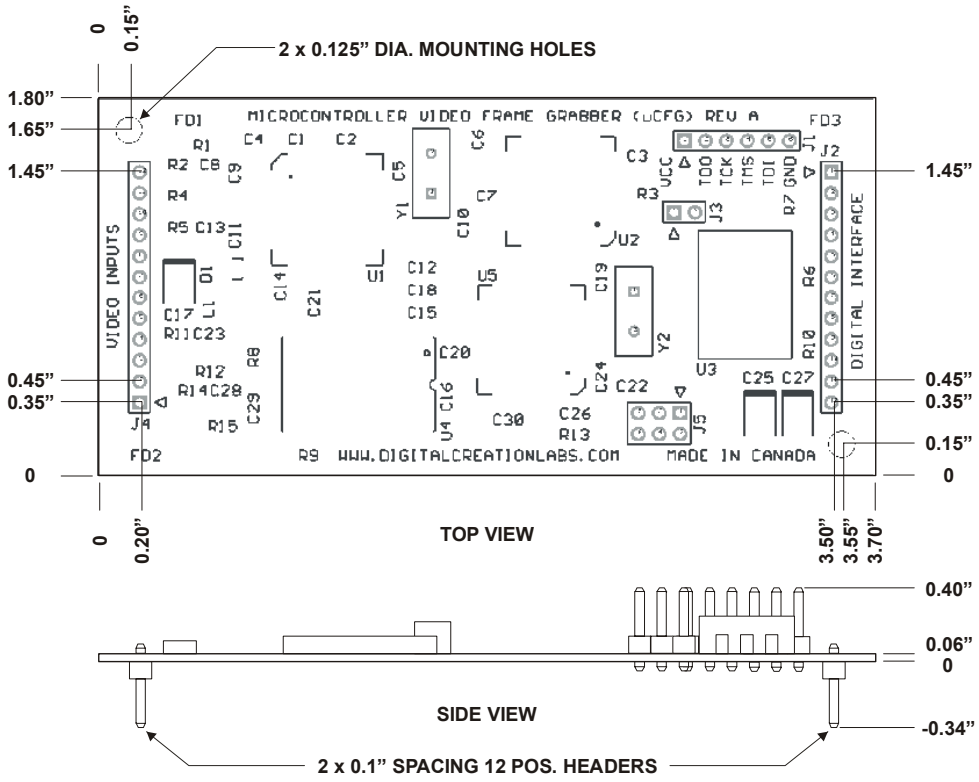


Figure 3: Mechanical dimensions of the uCFG board

Two 1/8" diameter mounting holes are provided to secure the uCFG daughtercard to the motherboard using a pair of standoffs. Please note that the overall clearance height of the board as measured from the motherboard surface will vary based on the selected standoff length. For example, using a pair of standard ½" length standoffs will result in a clearance height of 0.50" + 0.40" = 0.90" from the surface of the motherboard.

Digital Video Overview

The following provides a very brief overview of video signal fundamentals. For more detailed information, the reader is referred to the following books:

"*Video Demystified: a Handbook for the Digital Engineer*", by Keith Jack, Newnes Publishers, 2001.

"*Digital Video and HDTV: Algorithms and Interfaces*", by Charles Poynton, Morgan Kaufmann Publishers, 2003.

The NTSC/PAL Composite Signal

Figure 4 shows an illustration of the structure of the common National Television System Committee (NTSC) analog composite baseband video signal. PAL timings vary slightly (as well as number of lines, colorburst phase, etc..), but the overall video signal structure is very similar.

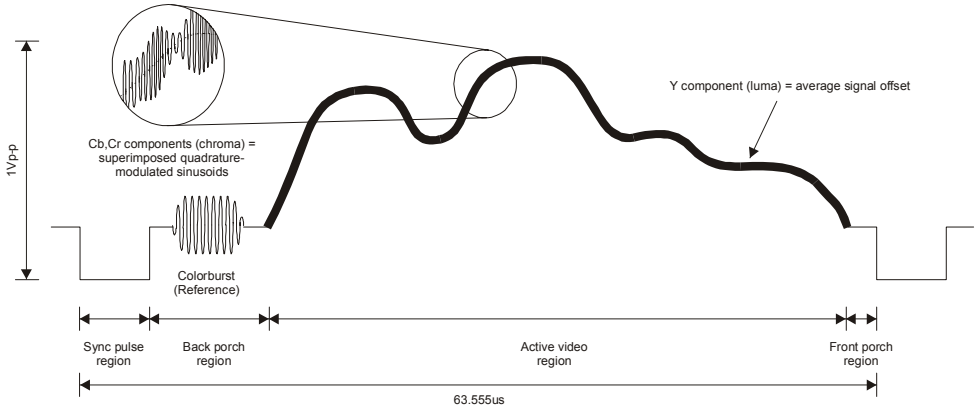


Figure 4: A typical single line composite NTSC video signal

There are four regions of interest: the sync pulse, the back porch, the active video region, and the front porch.

Sync pulse: this is the lowest level of the video signal used to synchronize a receiver to the incoming signal on a line-by-line basis.

Back porch: the back porch is set at the black level and contains a sinusoidal waveform called the colorburst. The colorburst serves as a color subcarrier reference with respect to which the color information of the active region is decoded.

Active video: the active video region contains the information for three different components, the luma (Y) as well as two chroma components (Cr, Cb). The luma represents the intensity information of the image at every point along the line. In the active region, the luma component is represented as the average value of the signal (this preserves compatibility with the old black & white TV standard). To add color information (while preserving b&w backwards compatibility), two color components are added to the luma signal using quadrature-modulation. The resulting superimposed color signal component is characterized by a sinusoidal wave whose phase with respect to the colorburst reference represents the hue of the color, and whose amplitude represents the saturation of that color.

Front porch: the front porch is set at the black level.

Image Encoding

An image is converted into an NTSC video signal by a video camera. The encoding process breaks down the information contained in the full image into a sequence of single-line video signals. The lines are encoded in a raster scan pattern as shown in Figure 5. The NTSC standard calls for scanning of 525 horizontal lines in a video frame at 30 frames per second. This just exceeds the threshold of human retinal persistence where image flicker is no longer perceived during playback.

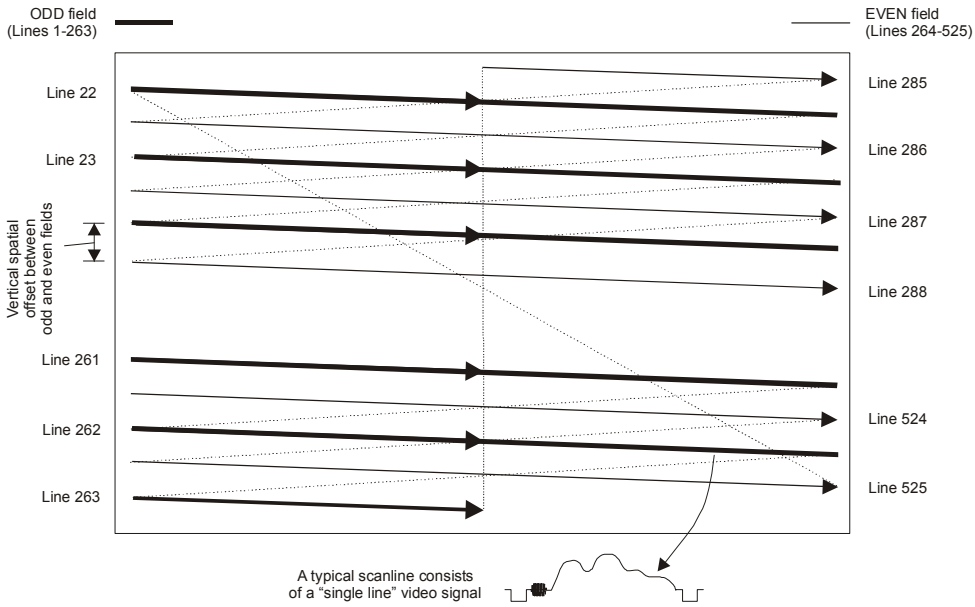


Figure 5: Video image raster scan pattern

Early TV pioneers decided that it would be possible to double the image refresh rate to 60 Hz by introducing the video interlacing technique. This trick again uses the retinal persistence of the human eye. Instead of scanning all the 525 lines of the image from top to bottom, the image is generated by scanning the 262.5 odd lines first, then followed by scanning the 262.5 even lines. Because these two half-images, or fields, are actually vertically offset in space (and displayed as such), the eye is again fooled into seeing a static 525 line picture in space.

Odd video field: is composed of the 262.5 odd lines of the video image.

Even video field: is composed of the 262.5 even lines of the video image.

Video frame: the full 525 image with odd and even field interlaced together.

For PAL, there are 625 lines per frame, and 312.5 lines per field, and 50 Hz field rate (25 Hz frame rate).

Freezing an Image

One of the side-effects of the video interlace technique is that the two fields *are actually acquired at different instants in time*. For the NTSC standard, the delay between acquiring the ODD and EVEN fields is 1/60 sec. What that means is that the ODD and EVEN half images that belong to the same video frame are slightly temporally mismatched. When displayed on a TV screen, our eyes are easily fooled by the rapidly changing image sequence. However, when freezing a full frame of video, the mismatch becomes obvious when an image of a fast moving object is taken. This is illustrated in Figure 6.

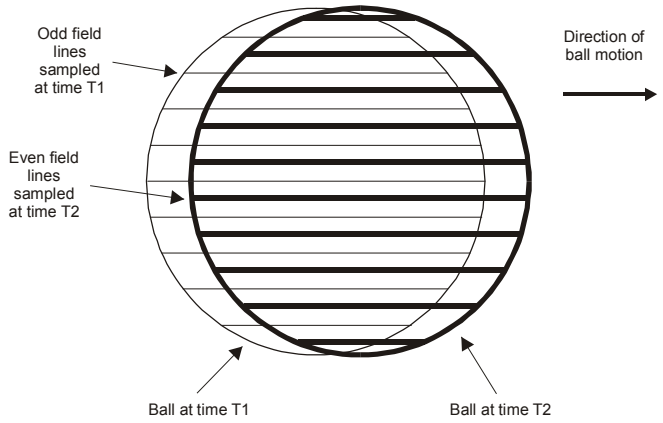


Figure 6: Spatio-temporal mismatch of the odd and even fields in a frozen video frame

There are some de-interlacing techniques available to try and compensate for the motion artifacts by identifying the regions of large motion and interpolating. This usually requires a fair bit of computational resources. The most common approach is to only freeze ONE of the two fields. The tradeoff is that only half of the vertical resolution is available, but at a fixed instant in time. If required, the other field can simply be interpolated from the acquired one through simple vertical interpolation techniques.

The uCFG allows the digitization of a single field of video with full control over which field is acquired (ODD, or EVEN). This provides the option to acquire a full-frame, high resolution image of static scenes. To do this, the ODD field is first captured and downloaded, followed by the EVEN field capture and download. The two are interlaced together and a full frame of 525 lines is obtained.

ITU-R BT656 Digital Video Stream Format

Digitizing a composite analog video signal requires and analog-to-digital converter (ADC). Some front-end analog signal preprocessing is also required to clamp the video signal and make sure it is positioned within bounds of the ADC. A video decoder is a specialized IC that performs all these steps and outputs video samples in a standard digital video format.

The ITU-R BT601 standard describes the fundamentals of the video digitization process. The ITU-R BT656 standard further specifies an 8 bit digital video parallel interface. Both standards apply to 4:2:2 sampled digital video. The 4:2:2 terminology simply means that during the video sampling, the luma is digitized at every pixel position (13.5 MHz sample rate) and the chroma components are sub-sampled by a factor 2 (6.75 MHz each). This sub-sampling is possible since the human eye is less sensitive to color spatial resolution then to intensity spatial resolution. The direct benefit is that a 4:2:2 data stream is 2/3 the overall data size as compared to a non-decimated image.

Figure 7 illustrates the sampling process. As shown, every one of the 720 pixel positions sampled has a Y, Cb, Cr value, except that every 2nd Cb and Cr components are dropped (decimated). The resulting BT656 4:2:2 data stream (gray box) is a simple concatenation of these 8-bit luma and chroma samples. The aggregate BT656 data stream rate is of 13.5 MHz + 2 x 6.75 MHz = 27 MHz.

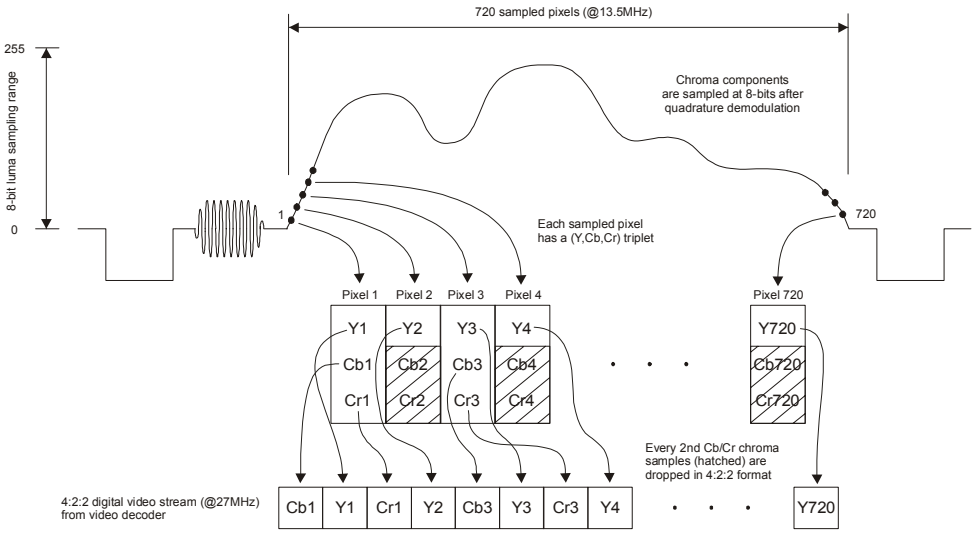


Figure 7: ITU-R BT656 4:2:2 digital video stream

Color Space Conversions

So far, we have explored the fundamentals of the analog composite video standard as well as digital video encoding techniques. As described previously, the color information contained in a composite video signal is decoded into the Y, Cb, Cr color space as per the ITU-R BT656 standard. This color space is very convenient for many types of image processing where operations are mainly performed on the black & white video component (in that case, the Y luma component can be used directly). To display color images on a PC, however, the more familiar RGB color space is used. In this color space, each pixel is described by an 8-bit value of its red, green and blue components. In order to perform a conversion from the Y, Cb, Cr color space to the RGB color space, a simple transformation can be applied:

$$R = 1.164 \times (Y-16) + 1.596 \times (Cr-128)$$
$$G = 1.164 \times (Y-16) - 0.392 \times (Cb-128) - 0.813 \times (Cr-128)$$
$$B = 1.164 \times (Y-16) + 2.017 \times (Cb-128)$$

(R, G, B are gamma-corrected and ranging between [0-255])

Equation 1: (Y, Cb, Cr) to (R, G, B) color space transformation equations

Operation of uCFG

The following section will explore in detail the operation of the uCFG board. First, let's begin by hooking up the basic signals to the uCFG board.

Basic Hookup

The basic wiring hookup to the uCFG board is illustrated in Figure 8. Please note the following restrictions: **the UART serial signals to/from the uCFG are LVTTL (+3.3V) levels, and NOT RS-232 +/-12V levels**. Therefore, **you cannot directly hookup the serial lines to a PC's serial port, otherwise you WILL damage the board and void the warranty**. Instead, you can directly connect the pins to an external microcontroller's UART lines directly. The RXD line is 5V tolerant, so you can connect to a MCU operating at 5V (or +3.3V). The TXD will swing from 0 to 3.3V, which should be also compatible with 5V MCU's (or +3.3V). **NONE of the other I/O signal lines are +5V tolerant (including the trigger lines)**. The polarity of the LVTTL TXD and RXD lines can both be configured in the uCFG's EEPROM as a user setting (if necessary).

In order to connect the board to a PC's serial port, you need to use an external level shifter such as the Sipex SP3232. For experimentation, it is more convenient to use the uCFGEVAL kit which has the level shifter built-in as well as all the various connectors brought out from the uCFG daughterboard.

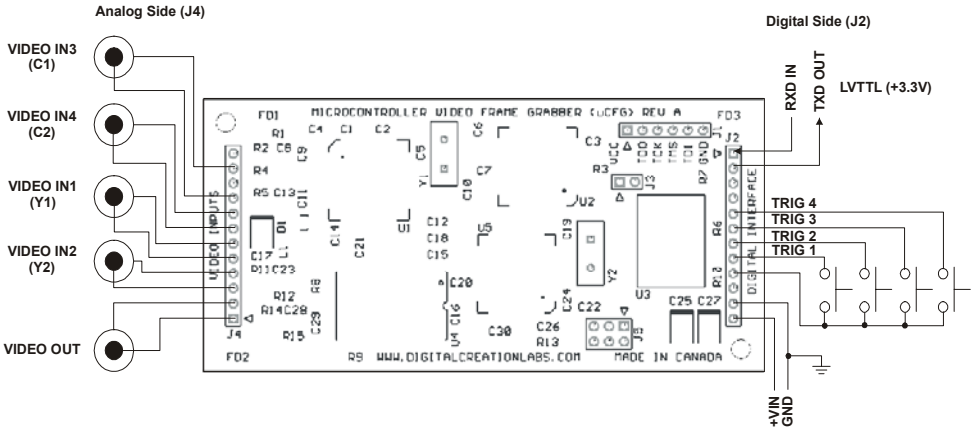


Figure 8: Basic uCFG hookup

To power the uCFG board, apply a voltage between +5VDC and +9VDC. It is recommended to use a regulated +5VDC supply to minimize power dissipation of the on-board +3.3V LDO regulator.

EEPROM Settings

The uCFG has some non-volatile EEPROM memory on-board. A portion of this memory (256 bytes) is reserved as general purpose user space. To read/write to this memory area, use the EEPR/EEPW ASCII commands. The rest of the memory is reserved for storing board configuration parameters. In order to setup the board, the uCFG PC Viewer software is needed. Also, the uCFGEVAL board should be used to connect the uCFG board to the PC's serial port. Using the software, the user can select the following options:

Board name: You can give the uCFG board a unique textual name description (25 chars max)

User string: General purpose user string (25 chars max)

Video mode: PAL or NTSC (setup according to your cameras)

Video mux mode: Composite (4 channel) or S-Video (2 channel) (setup according to your needs)

UART mode: Normal or Inverted (adjusts the polarity of the RXD/TXD lines)

Trigger enable: Check the trigger sources that are enabled (via hardware or software)

Trigger polarity: Adjust the polarity (active high = leading edge, active low = trailing edge)

Trigger mode: One-shot or Retriggerable (in one-shot mode, the unit inhibits future triggers until trigger latch is cleared. In retriggerable mode, triggers are always active regardless of the state of the trigger latch)

Startup baud rate: uCFG's startup baud rate (from 2400 baud to 230.4 kbps)

Once these parameter are changed, the power to the board should be cycled (or a BOOT command issued to force a reboot the board) for the changes to take effect.

PLEASE NOTE: If a shunt is loaded on J3, then the next time the uCFG powers up, it will assume the default configuration regardless of the EEPROM user settings. These defaults include:

Video mode: NTSC
Video mux mode: Regular composite inputs (4 inputs)
UART mode: Normal
Trigger enable: all enabled
Trigger polarity : all active high
Trigger mode: retriggerable
Startup baud rate: 115.2 kbps

This is convenient when setting up a uCFG unit with unknown prior settings. If J3 is not shunted, then the power up settings will be read from EEPROM.

FIFO Read Pointer Control

When a video field grab is commanded either through the software GRAB/TRIG or a hardware trigger, the uCFG board digitizes the very next video field (odd or even depending on request) in the ITU-R BT656 4:2:2 digital video format. This data stream is stored into a long First-In/First-Out (FIFO) memory field buffer. A FIFO is simply a memory storage element which is linearly accessible through a read pointer. This operation essentially freezes the digital data stream for the entire field as-is into the FIFO.

Once the capture is complete, the MCU can read out the contents of the FIFO one byte at a time starting from the beginning of the stored digital stream. This can be done at an arbitrary pace by using a basic ASCII command set. The linearly accessible FIFO allows the MCU to read out one byte of data at a time. The MCU then increments the read pointer to the next data byte in the stored data stream. It is also possible to directly increment the read pointer in the FIFO by an arbitrary amount without reading the data. This is useful to skip some data samples for horizontal decimation, or even skip an entire row's worth of data for vertical decimation. The read pointer can also be reset to point back to the beginning of the stream with the RRST command. The only restriction is that the pointer cannot be decremented directly.

Figure 9 shows the physical layout of the uCFG's internal FIFO buffer. As shown, the FIFO is a byte-wide uninterrupted memory buffer accessible with the help of a read pointer (used to index the memory location to read). For simplicity, the FIFO in Figure 9 has been split up into separate video lines even though all the data is actually continuous inside the FIFO. As can be seen, one line (720 pixels wide) of video data actually occupies 1440 bytes of physical FIFO space. Note that every second luma (Y) sample is surrounded by its chroma (Cb, and Cr) color components which are co-sited in space (i.e. belong to the same pixel). Every other luma sample is alone (due to the 4:2:2 chroma decimation).

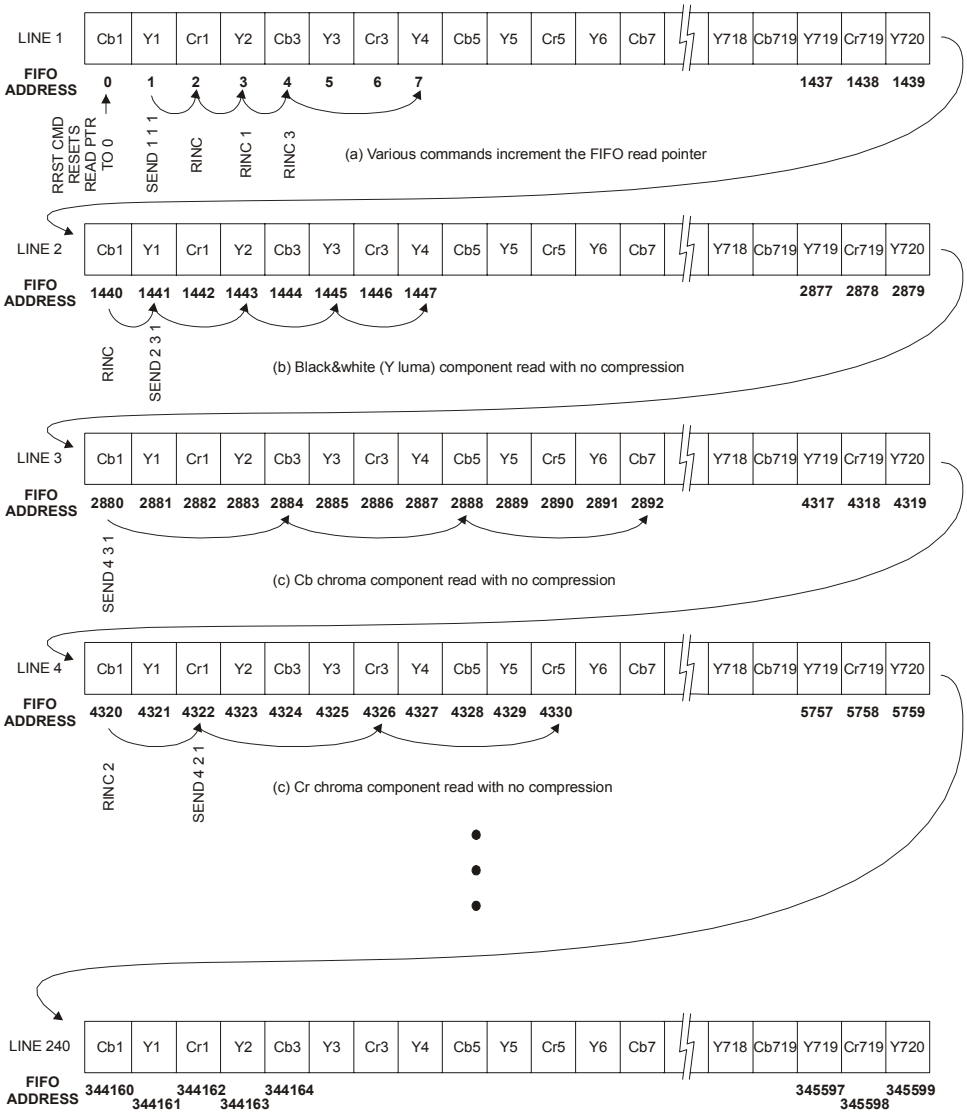


Figure 9: Internal layout of the uCFG's FIFO field buffer

The host controller is in full control of the FIFO access operations and readout. In fact, the host is responsible for issuing the appropriate commands to skip data samples when it requires image decimation. A number of example commands are illustrated in Figure 9 as well as their effect on the read pointer. (a) Different commands increment the read pointer directly. RRST resets the pointer to 0, the start of the FIFO. SEND 111 simply reads the current byte and increments the pointer by 1. RINC with no parameters assumes 1 by default and increments the pointer. RINC with parameters increments the pointer by the specified amount. (b) To download a black & white image only (the luma component of the color image) at the full 720 resolution, simply position the read pointer on the first luma sample of interest with RINC and issue a SEND 2 n 1 where n is the number of bytes to read. Here the command will read and return every 2nd sample until a total of n bytes have been returned on the serial port. To read a ½ decimated version, use the SEND 4 n 1 which skips every 2nd luma (i.e. return every 4th sample). The possibilities are quite numerous. (c) & (d) Illustrates color component download. Simply position the pointer to the appropriate starting position and issue a SEND 4 n 1 to read out every 4th sample.

To read a full color image, first read all the luma values (Y), then issue a read pointer reset command RRST. Then, read all the Cb components and issue another read pointer reset. Finally, read out all the Cr components.

Hopefully, it now becomes clear how powerful the different FIFO commands are. They give full control over the position of the read pointer, the skip factor, and the amount of data to be downloaded. As an example, reading out a sub image (region of interest) would be quite simple to do by directing the read pointer to the correct locations and selectively downloading the data. This feature could be used to perform a quick coarse thumbnail image preview, then a high resolution area of interest download. Another example could be downloading a black and white image only, but with color info only for a small region of interest within the master image…all these techniques reduce the downloaded image data size.

To use the compression feature, simply set the 3rd parameter of the SEND command (compression ratio) to 2, 4 or 8 and a compressed image data stream is returned instead of the raw sample values. For example, SEND 2 3 2 would return 3 compressed bytes at 2:1 compression. This means that 3 compressed bytes * 2 compressed pixels / byte = 6 image samples in total skipping every 2nd sample in the FIFO (i.e. Y samples only). The number of bytes returned from this call would be 1 integrator reset byte (for decompression) as header + 3 compressed data bytes = 4 bytes. Simply feed this compressed data vector of 4 bytes into the decompression routine, and the original vector of 6 raw image samples will be regenerated. Please note that the compression is done on-the-fly as data is returned on the serial port. The original raw image data in the FIFO remains unaltered regardless of the way in which the data is read out (i.e. decimated, compressed or not). The only time the data is altered is when the next field grab is commanded (or when the power is removed). It is therefore possible to read out the data multiple times with different settings. This can be useful to first obtain a thumbnail, followed by a full download.

Image Data Size and Download Time

The FIFO buffer on the uCFG can hold 393216 byte in total. This is large enough to hold a full field of full color 4:2:2 NTSC data (720x240) or PAL (720x272). The following table summarizes some of the various image download data stream sizes given the various parameter selections. To achieve the decimation, the skip factor is used in the SEND command. To achieve compression, the compression ratio parameter is used in the SEND command. To achieve B&W download, only the Y component data is downloaded.

B&W	720x240	360x120	180x60	90x30	45x15
1:1	172.80K	43.20K	10.80K	2.70K	675
2:1	86.40K	21.60K	5.40K	1.35K	338
4:1	43.20K	10.80K	2.70K	675	169
8:1	21.60K	5.40K	1.35K	338	84

Table 1: A few sample black & white compressed image data sizes (in bytes)

Color	720x240	360x120	180x60	90x30	45x15
1:1	345.60K	86.40K	21.60K	5.40K	1.35K
2:1	172.8K	43.20K	10.80K	2.70K	675
4:1	86.40K	21.60K	5.40K	1.35K	338
8:1	43.20K	10.80K	2.70K	675	169

Table 2: A few sample color compressed image data sizes (in bytes)

As far as the download times are concerned, the ASCII protocol adds some overhead to the download times. Therefore, to maximize download data throughput, it is recommended to request as much data per message as the host will accept. The uCFG PC Viewer software, for example, requests a whole line's worth of data samples at each SEND message.

Another consideration is that the 8 bit MCU on the uCFG requires some time to perform on-the-fly compression of the FIFO data (when requested). In fact, the larger the compression ratio, the higher the MCU overhead. For slow baud rates, this is irrelevant, however, at 115.2 kbps or higher baud rates, 8:1 compression download times incur some overhead. Here is a sample of download times obtained on a PC using the uCFGEVAL kit. The baud rate was set at 115.2 kbps and the results timed on a stopwatch.

Color	720x240	360x120	180x60	90x30	45x15
1:1	31 s	13 s	10 s	N/A	N/A
2:1	17 s	10 s	9 s	N/A	N/A
4:1	13 s	9 s	9 s	N/A	N/A
8:1	10 s	9 s	9 s	N/A	N/A

Table 3: Sample field download times at 115.2 kbps using a PC host running the uCFG viewer software (full color)

Download Video Compression

The uCFG board allows compressing video data on-the-fly just before the video samples are sent out the serial port. Of course, the original source data in the FIFO remains uncompressed, allowing multiple downloads with different compression ratios settings. The compression technique used in the uCFG allows compression ratios of 2:1, 4:1 or even 8:1 with varying image quality results. Please note, for resource-limited host applications, it is NOT necessary to use compression but does result in much smaller image data footprint (good for image storage).

To request download of a number of compressed samples, the SEND command is issued as usual, but the 3rd parameter (compression ratio, or cratio) is specified as being 1, 2, 4, or 8. A value of 1 (or omitted) results in no compression. Specifying a cratio of 2 will send two sample's worth of data in each byte. The top 4 and bottom 4 bits represent the compressed coefficients for the two samples. With a cratio of 4, each pair of bits represent the coefficients for 4 samples. Finally, with a cratio of 8, there is 1 bit for per sample for a total of 8 samples per byte. Also, the "number of bytes to download" parameter in the SEND command now represents the number of COMPRESSED bytes to send. In other words, specifying SEND s 2 8 (num bytes 2, cratio 8) will return 2 compressed bytes which actually represent 16 samples of compressed data.

To decompress a compressed vector of data as downloaded from the uCFG, you can use the following algorithm shown here coded in C. The routine works with any of the compression ratios (2, 4, or 8). Simply feed the vector as received from the uCFG. Of course, if no compression is used (i.e. cratio is 1) then there is no need to call this routine since the raw data samples will already be available. When the cratio is set to a value different then 1, please note that an additional byte is sent in the reply message of the SEND command ahead of the requested bytes (i.e. is the first value in the received compressed data vector). This value is the initial starting value of the integrator for the decompression process. If cratio is set to 1 or omitted, then that initial value byte is simply NOT sent, only the raw data values.

(C language listing of decompression algorithm available with the uCFGEVAL kit.)

Triggers

The uCFG board supports very flexible image capture triggering mechanisms, both in hardware and software. First, 4 external hardware triggers are provided for external hardware triggering (LVTTL +3.3V levels only). Each of the 4 trigger is associated with a respective camera channel. When in S-Video mode, triggers 1 or 3 and 2 or 4 trigger S-Video channels 1 and 2 respectively. The uCFG's EEPROM trigger settings allows disabling individual triggers, setting the trigger polarity, and selecting a triggering model. The polarity setting decides whether an image capture is triggered on a leading edge or a trailing edge of the respective trigger.

Every time a trigger is activated, the uCFG switches the analog video mux to the corresponding video channel and grabs an image into the FIFO buffer. A trigger latch variable in the MCU also logs the value of the last activated trigger input. The host system can poll (using TRLR) the value of the trigger latch to find out if there have been any activations since the last time the latch was cleared (TRLC). The host must then take action to download any image data after a detected activation (if desired). Of course, if any trigger activates during download, the image data is overwritten. That is why a command (TREN) is provided to temporarily disable/enable all triggers while image downloading is in effect. Alternatively, you may set the trigger mode to one-shot. In this mode, each time the trigger latch is cleared to 0, the board will function normally. The first trigger will grab an image and automatically inhibit any further triggers until the trigger latch is again cleared with the TRLC command. This is useful if the host is not able to poll the trigger latch very often and guaranties no overwrites.

In addition to hardware triggering, it is possible to simulate a hardware trigger through the software TRIG command. This will have the exact same effect as a hardware trigger.

I^2C Expansion Bus

The uCFG board provide a bridge function between the serial port and an I^2C expansion bus. A video decoder is already on the I^2C bus internally at decimal address 74, but otherwise, the I^2C bus is available for general use. The I2CR and I2CR commands allow reading and writing to a specified register of an I^2C device at a given address. Please note that the I^2C bus operates at +3.3V ONLY and is NOT compatible with +5V I^2C. Internal 2.2K pull-up resistors to +3.3V are provided on the SCL/SDA lines.

Command Set

Here is the complete ASCII command set supported by the uCFG. Please note that the only valid ASCII characters that will be recognized by the uCFG are a-z, A-Z, 0-9, <Spacebar>, <Return> and <Backspace>. Note that lowercase characters a-z will be converter to uppercase A-Z by the uCFG in the ECHO. The uCFG will also automatically append a \n (linefeed) character in its ECHO immediately following any \r (carriage return) received and echoed. At most 30 ASCII characters can be received, otherwise the uCFG will automatically generate a \r\n sequence.

All uCFG replies are preceded with the * character, and ended with \r\n. The only command that does not generate a \r\n terminated reply following the initial echo is the SEND command.

BOOT (Reboot Target)

Syntax:

BOOT\r

Parameters:

None

Possible Return Values:

*OK\r\n

Example:

Command: BOOT\r

Echo: BOOT\r\n

Reply: *OK\r\n

Command Description:

This command causes the uCFG to reset by tripping the internal watchdog. Even though the command returns immediately with an OK, you should wait up to 2 seconds for the watchdog to trip and reboot the hardware.

CFGR (Board Configuration Read)

Syntax:

CFGR a\r

Parameters:

a is the decimal EEPROM address valued between [0..255]. If larger number is used, only the lower 8 bits will be read.

Possible Return Values:

*nnn\r\n where *nnn* is the zero-padded decimal three digit EEPROM data value at the specified address

*ERROR\r\n

Example:

Command: CFGR 23\r

Echo: CFGR 23\r\n

Reply: *255\r\n

Command Description:

This command will read a single data byte from the configuration space of the on-board EEPROM. It is recommended to only let the PC Viewer software use this command for board setup purposes.

CFGW (Board Configuration Write)

Syntax:

CFGW a d\r

Parameters:

a is the decimal EEPROM address valued between [0..255]. If larger number is used, only the lower 8 bits will be read.

d is the decimal data byte value to be written at address *a*. *d* is valued between [0..255]. If a larger number is specified, only the lower 8 bits will be used.

Possible Return Values:

*OK\r\n

*ERROR\r\n

Example:

Command: CFGW 23 4\r

Echo: CFGW 23 4\r\n

Reply: *OK\r\n

Command Description:

This command allows a user to write a single data byte to a specified address in the configuration user space of the on-board EEPROM. **Be careful with this command, since the board settings will get corrupted if accidentally overwritten**. It is recommended to only let the PC Viewer software use this command for board setup purposes.

CGET (Get Currently Selected Video Channel)

Syntax:

CGET\r

Parameters:

None

Possible Return Values:

*nnn\r\n where *nnn* is a zero-padded three digit number in the range [001..004]

*ERROR\r\n

Example:

Command: CGET\r

Echo: CGET\r\n

Reply: *001\r\n

Command Description:

This command will read and return the currently selected analog video channel (1 to 4).

CSEL (Video Channel Select)

Syntax:

CSEL n\r

Parameters:

n is the channel to be selected. *n* is in the range [1..4]. A value of *n*=0 will be clamped to 1 and a value of *n*>4 will be clamped to 4. If in S-Video mode, *n* should be in the range [1..2]. If *n* is of value 3 or 4, then it will be aliased as 1 or 2 respectively.

Possible Return Values:

*OK\r\n

*ERROR\r\n

Example:

Command: CSEL 1\r

Echo: CSEL 1\r\n

Reply: *OK\r\n

Command Description:

This command will switch to analog video input multiplexer to the specified channel. The video output will immediately display the live feed of the selected channel.

EEPR (EEPROM Read)

Syntax:

EEPR a\r

Parameters:

a is the decimal EEPROM address in the range [0..255]. If larger number is used, will only read lower 8 bits.

Possible Return Values:

*nnn\r\n where *nnn* is a zero-padded decimal three digit EEPROM value at the specified address

*ERROR\r\n

Example:

Command: EEPR 23\r

Echo: EEPR 23\r\n

Reply: *255\r\n

Command Description:

This command allows a user to read a single byte from the general purpose user space of the on-board EEPROM.

EEPW (EEPROM Write)

Syntax:

EEPW a d\r

Parameters:

a is the decimal EEPROM address in the range [0..255]. If larger number is used, will only use lower 8 bits.

d is the decimal data byte value to be written at address a. d is in the range [0..255]. If larger number is used, will only use lower 8 bits.

Possible Return Values:

*OK\r\n

*ERROR\r\n

Example:

Command: EEPW 23 4\r

Echo: EEPW 23 4\r\n

Reply: *OK\r\n

Command Description:

This command allows a user to write a single byte to a specified address in the general purpose user space of the on-board EEPROM.

FD (Check if Forced Defaults)

Syntax:

FD\r

Parameters:

None

Possible Return Values:

*YES\r\n

*NO\r\n

Example:

Command: FD\r

Echo: FD\r\n

Reply: *NO\r\n

Command Description:

This command check whether shunt (jumper) J3 is loaded. When loaded, the board assumes the following default settings on power-up:

Video mode: NTSC
Video mux mode: Regular composite inputs (4 inputs)
UART mode: Normal
Trigger enable: all enabled
Trigger polarity : all active high
Trigger mode: retriggerable
Startup baud rate: 115.2 kbps

GRAB (Grab Video Field)

Syntax:

GRAB f\r

Parameters:

f is the optional field type, either O or E (Odd or Even). Default will be O if *f* is not specified or if invalid parameter.

Possible Return Values:

*OK\r\n

*ERROR\r\n

Example:

Command: GRAB E\r

Echo: GRAB E\r\n

Reply: *OK\r\n

Command Description:

This command will acquire a full field of raw color video data from the currently selected video channel into the FIFO. The command returns an error if either there is no valid video signal, or if a video signal of the wrong standard (as configured in EEPROM) is detected on the currently selected video channel. The timeout interval for detection of a valid signal is of 100 ms.

I2CR (I^2C Bus Read)

Syntax:

I2CR a r\r

Parameters:

a is the I^2C peripheral address of the device to be read. *a* is in the range [0..255]. If larger number is used, will only use lower 8 bits.

r is the I^2C device register address to be read. r is in the range [0..255]. If larger number is used, will only use lower 8 bits.

Possible Return Values:

*nnn\r\n where *nnn* is a zero-padded decimal three digit I^2C register value of the I^2C device at specified address

*ERROR\r\n

Example:

Command: I2CW 74 0\r

Echo: I2CW 74 0\r\n

Reply: *017\r\n

Command Description:

This command allows a user to read a single byte from a specified I^2C peripheral address and register.

I2CW (I^{2}C Bus Write)

Syntax:

I2CW a r d\r

Parameters:

a is the I^{2}C peripheral address of the device to be written. *a* is in the range [0..255]. If larger number is used, will only use lower 8 bits.

r is the I^{2}C device register address to be written. *r* is in the range [0..255]. If larger number is used, will only use lower 8 bits.

d is the decimal data byte value to be written at register *r* of I^{2}C device at address *a*. *d* is in the range [0..255]. If larger number is used, will only use lower 8 bits.

Possible Return Values:

*OK\r\n

*ERROR\r\n

Example:

Command: I2CW 74 2 192\r

Echo: I2CW 74 2 192\r\n

Reply: *OK\r\n

Command Description:

This command allows a user to write a single byte to a specified I^{2}C peripheral address and register.

PING (Check if Board Present)

Syntax:

PING\r

Parameters:

None

Possible Return Values:

*OK\r\n

Example:

Command: PING\r

Echo: PING\r\n

Reply: *OK\r\n

Command Description:

This command is used to check if board is connected and responding to messages.

RINC (Read Pointer Increment)

Syntax:

RINC n\r

Parameters:

n is the optional count (in bytes) by which to increment the FIFO read pointer. *n* is in the range [1..393216]. If *n* > 393216, then n is clamped to 393216. If *n* = 0, then call has no effect. Default value for *n* is 1 if not specified or if invalid parameter.

Possible Return Values:

*OK\r\n

Example:

Command: RINC 1440\r

Echo: RINC 1440\r\n

Reply: *OK\r\n

Command Description:

This command will increment the FIFO read pointer by the specified number of bytes. This function is useful for skipping data in the FIFO.

RRST (Read Pointer Reset)

Syntax:

RRST\r

Parameters:

None

Possible Return Values:

*OK\r\n

Example:

Command: RRST\r

Echo: RRST\r\n

Reply: *OK\r\n

Command Description:

This command will simply reset the FIFO read pointer to address 0, the beginning of the stored digital data stream.

SEND (Send Image Data)

Syntax:

SEND s n c\r

Parameters:

The parameters are optional, however, any parameters used will be parsed in order. That means that if *c* is specified, then both *s* and *n* must also be. If *n* is specified, then *s* must also be. If any of the parameters are omitted, then the last specified values (in the previous call) are used. This is useful to reduce protocol overhead in performing repeated calls. The power-on default parameters are *s*=1, *n*=16 and *c*=1.

s is the optional skip factor, representing which bytes to send (i.e. *s*=3 would send every 3rd byte).

n in the optional number of bytes to send (this will match the number of bytes in the reply (+1 byte if c is not 1)). *n* is in the range [1..393216]. If *n*>393216, then it is clamped to 393216.

c is the optional compression ratio. It is equal to one of the following values: [1,2,4,8]. If it isn't set to one of the mentioned values, then *c*=1 is assumed by default.

Possible Return Values:

Requested binary mode image data bytes

Example:

Command: SEND 2 360 2\r

Echo: SEND 2 360 2\r\n

Reply: requested binary mode image data (NOT terminated by \r\n). In this case, will return 361 bytes (because c is not 1) composed of integrator initialization byte for the decompression process, followed by the 360 "2:1 compressed bytes", since the compression ratio is set at 2 (which contain all the data for reconstructing 720 pixels, or a full row of data). The skip factor is set at 2, so in this case, will compress every 2nd byte of the possible 1440 (i.e. only the Y values).

Command Description:

This command will cause the uCFG to send the next *n* bytes from the FIFO out the serial port in binary format. The command will also skip the specified number of bytes internally and only send every *s*th byte from the FIFO. Note that the skip factor does not affect *n*. In other words, if *n* is 5 and *s* is 2, then 5 bytes will be sent, skipping every 2nd byte in the FIFO. If SEND command is sent with no parameters, the last used values for *n*, *s* and *c* will be assumed. The FIFO read pointer is automatically incremented during the execution of this command.

TEST (Test for Valid Video Signals)

Syntax:

TEST\r

Parameters:

None

Possible Return Values:

*CH1:s CH2:s CH3:s CH4:s\r\n where *s* is either 1 or 0 if valid signal detected or not

Example:

Command: TEST\r

Echo: TEST\r\n

Reply: *CH1:1 CH2:0 CH3:0 CH4:1\r\n

or in S-Video mode

Command: TEST\r

Echo: TEST\r\n

Reply: *CH1:1 CH2:0\r\n

Command Description:

This command is used to auto-detect the presence of valid video signals on all 4 input channels (2 in S-Video mode). If a valid video signal is detected (of the correct standard PAL or NTSC as set in the board configuration), a 1 is return next to the CH reply marker, otherwise a 0 is returned. Please note that the command can potentially delay up to 200 ms between each CHn: labels of the reply message (worst case scenario of 800 ms with none of the 4 video channels connected at all).

TREN (Trigger Enable/Disable)

Syntax:

TREN e\r

Parameters:

e is in the range [0,1] and represents the enabled (1) or disabled (0) state of the trigger inputs.

Possible Return Values:

*OK\r\n

*ERROR\r\n

Example:

Command: TREN 1\r

Echo: TREN 1\r\n

Reply: *OK\r\n

Command Description:

This command temporarily enables or disables ALL trigger inputs (both hardware and software triggering) until the command is called again to re-enable them or until the power is cycled.

TRIG (Issue Soft Trigger)

Syntax:

TRIG n\r

Parameters:

n is the trigger to be triggered. It is in the range [1..4]. If above 4, *n* is clamped to 4. If *n*=0, then it is clamped to 1. The Odd or Even field selection will be that of the last specified value with the GRAB command. Odd is the power up default.

Possible Return Values:

*OK\r\n

*ERROR\r\n

Example:

Command: TRIG 3\r

Echo: TRIG 3\r\n

Reply: *OK\r\n

Command Description:

This command has the same effect as activating a hardware trigger.

TRLC (Trigger Latch Clear)

Syntax:

TRLC\r

Parameters:

None

Possible Return Values:

*OK\r\n

Example:

Command: TRLC\r

Echo: TRLC\r\n

Reply: *OK\r\n

Command Description:

This command clears the trigger latch to 0.

TRLR (Trigger Latch Read)

Syntax:

TRLR\r

Parameters:

None

Possible Return Values:

*nnn\r\n where *nnn* is a zero-padded decimal three digit trigger latch register value. Note that the lower 4 bits of the value represents the state of the last trigger latch even. A high value in any of the bit positions means the associated latched trigger is active (regardless of the set polarity). No more then one bit position will be high at the same time since only one trigger can be active at any time (if not, the priority encoder will automatically arbitrate).

Example:

Command: TRLR\r

Echo: TRLR\r\n

Reply: *004\r\n (= 00000100 binary, 3rd bit position high, so trigger 3 was last latched as active)

Command Description:

This command reads the current trigger latch value.

TRRD (Current Trigger Read)

Syntax:

TRRD\r

Parameters:

None

Possible Return Values:

*nnn\r\n where *nnn* is a zero-padded decimal three digit instantaneous trigger register read. Note that the lower 4 bits of the value represents the state of trigger latches 1-4. A high value in any of the bit positions means the associated trigger is currently active (regardless of the set polarity). Since this is an instantaneous snapshot of current trigger inputs, one or all bits can be simultaneously high (values read before the priority encoding stage).

Example:

Command: TRRD\r

Echo: TRRD\r\n

Reply: *015\r\n (= 00001111 binary, all 4 bit positions high, so all 4 triggers are currently active)

Command Description:

This command simply reads the current state of the triggers regardless of whether they are enabled or not. The 4 lower bits of the return value represent the state of triggers 1-4. A binary 1 means that the trigger is active (active means either high or low depending on the configured trigger polarity being active high or active low in the EEPROM). This is useful for debugging trigger setup.

VER (Firmware Version)

Syntax:

VER\r

Parameters:

None

Possible Return Values:

ASCII string terminated with \r\n representing the product name and firmware version number.

Example:

Command: VER\r

Echo: VER\r\n

Reply: *UCFG 2.0\r\n

Command Description:

This command simply queries the device to determine the device type and version number.

Pseudo-code for Standard Operations

In this section, some pseudo-code examples and flowcharts are provided as a demonstration on how to control the uCFG with the various ASCII commands. Please note that due to the flexibility of the uCFG architecture and of its command set, there may be many ways of accomplishing a desired result. The examples shown are mere suggestions.

Grabbing and Downloading an Uncompressed Field

A very common operation to be performed with the uCFG is to grab a field of video and download it. Figure 10 demonstrates one possible sequence of commands to grab and downloading a full resolution field of NTSC color video (720x240).

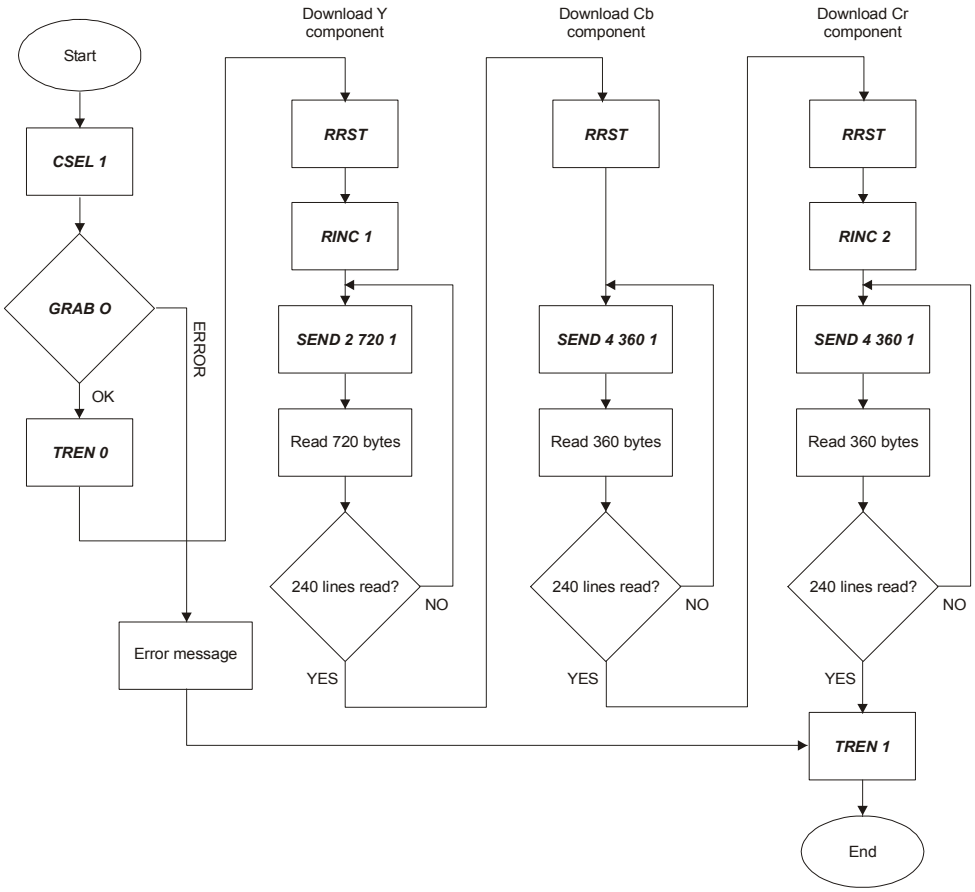


Figure 10: Flowchart showing possible command sequence to grab and download a full color, full resolution (720x240) NTSC field with no compression

First the video channel select command CSEL is issued (this is optional, otherwise the last selected channel will be used). The GRAB command with the odd (O) field parameter is then specified (use E for even field). After execution of the GRAB command, the return value is verified. If an error occurred, then either the selected channel has no valid video signal connected, or the signal is of the wrong standard (PAL/NTSC). Upon a successful field grab, the triggers are temporarily disabled with the TREN 0 command. This will prevent (in the event of a trigger) any new images from overwriting the FIFO data while downloading. Next, three passes are performed through the FIFO. The first pass will read out all the Y luma samples. The second pass will read out the Cb chroma samples, and the third, the Cr chroma samples. Of course, the samples could all be read in one pass, but it is simpler on the host side to split into three passes. As well, to read black & white data only, a single pass would be performed to collect the luma samples.

On the first pass, the FIFO read pointer is reset (RRST). As seen in Figure 9, FIFO address 0 actually points to the Cb1 sample (Cb value of first pixel of line 1). The read pointer therefore needs to be shifted by 1 to point to the first Y1 sample. This is done with the RINC 1 command. Next, for each line, a SEND 2 720 1 command is issued to send out every 2nd sample in the FIFO from the start position with no compression. This actually reads out all the Y samples of the first line. The process is then repeated for all the 240 lines of the image.

In this example, the host computer has enough buffer space to buffer 720 bytes of downloaded data. However, in the case of a resource-limited host (8-bit MCU), it may be desirable to instead download a few bytes at a time only. This is easy to do by changing the parameters of the SEND command. Of course, the increased protocol overhead cost will result in a longer download time.

The process is repeated for the 2nd and 3rd pass by first positioning the read pointer on the first Cb or Cr sample and then downloading every 4th data sample for a total of 360 per line (remembering that the chroma information is sub-sampled by a factor 2 in the 4:2:2 standard). Finally, once done the triggers are re-enabled with TREN 1.

Downloading a Compressed Field

Another common scenario is that of downloading a compressed image from the uCFG's FIFO. Figure 11 shows a sample flowchart. The initial steps are similar to the previous example. The main difference here is that the SEND command now requests 180 compressed data bytes per line at 4:1 compression. Every 2nd sample is skipped to obtain only the Y values. Because compression is enabled, remember that the return message as an extra byte at the beginning representing the initial value of the integrator for decompression purposes. As can be seen, each 720 pixel line is now represented by only 181 + 91 + 91 = 363 bytes of compressed data instead of 1440, or 4:1 compression.

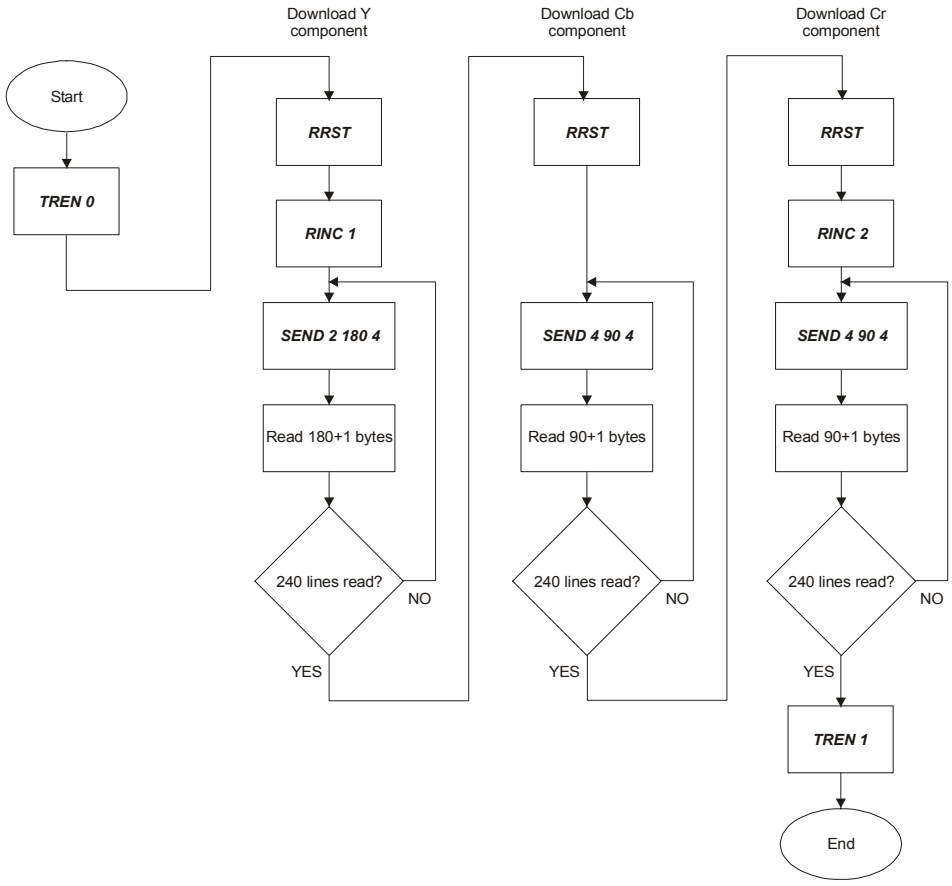


Figure 11: Flowchart showing possible command sequence to download a full color, full resolution (720x240) NTSC field with 4:1 compression

Of course, before the compressed data can be displayed to the user, each line's worth of data must be passed through the decompression routine to regenerate the raw 720 sample values of Y, and 360 of Cb and 360 of Cr.

Downloading a Decimated Field

Figure 12 shows a possible sequence of command to download an uncompressed color decimated field of half the full NTSC resolution (360x120). As before, most of the steps are the same. The main differences are that the SEND command now skips every 4 samples (i.e. every 2nd Y) only. This results in a horizontally decimated image by a factor 2.

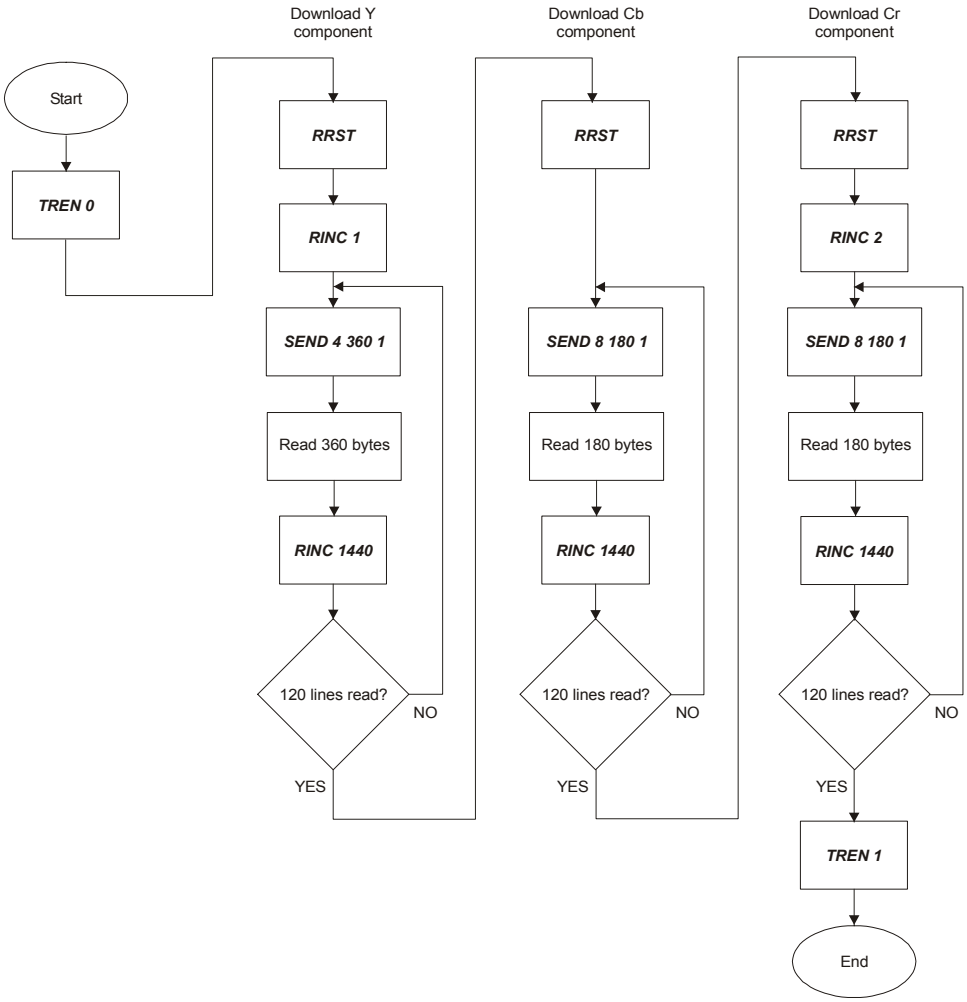


Figure 12: Flowchart showing possible command sequence to download a full color, half resolution (360x120) NTSC field with no compression

The second difference is that following the download of the decimated data for a full line, a RINC 1440 command is introduced. This command effectively skips an entire row (or line) of raw video data in the FIFO. This vertically decimates the number of lines in the downloaded field by a factor 2.

Converting 4:2:2 YCbCr to RGB for PC Display

The ultimate goal of the image downloading is to display the images to the user on a PC monitor. To perform this operation correctly, a few things must be explained. First off, PC monitors work in the RGB color space. This means that every pixel has an 8 bit red, green and a blue component. The downloaded data from the uCFG is in the YCbCr color space and, as well, the color components are decimated by a factor 2 (4:2:2). Before displaying an image on a PC, this data must therefore be upsampled and converted to the RGB domain.

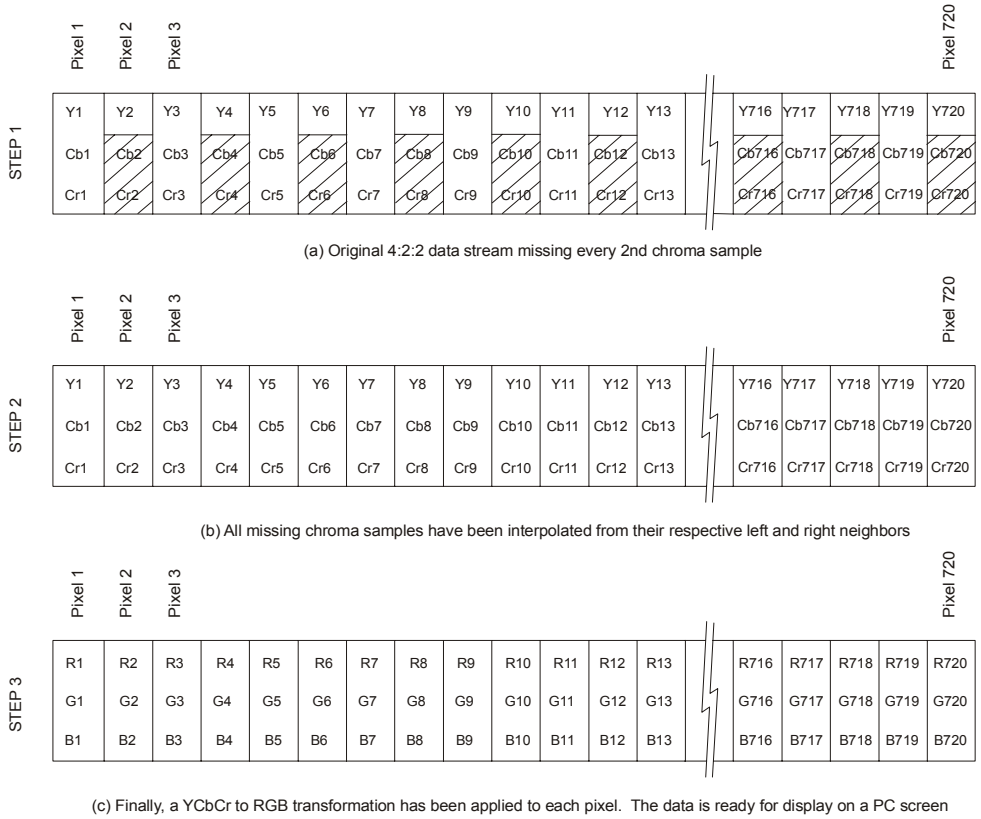


Figure 13: The different steps required to display downloaded data on a PC monitor in RGB format

Figure 13 illustrates the different steps required to convert a 4:2:2 YCbCr data stream into a non-decimated RGB stream suitable for display on the PC monitor. Step 1 shows a single line of downloaded 4:2:2 YCbCr color data samples shown overlaid over a row of the 720 pixels forming the image line. It can be seen that some of the decimated chroma samples are missing.

In Step 2, all the missing chroma values are filled in with a simple linear interpolation performed by taking the average of the previous and next respective samples. For example, Cr2 = (Cr1 + Cr3)/2. After this step, each pixel has a Y, a Cb, and a Cr value.

Step 3 performs a color space conversion from YCbCr to RGB. Each pixel's YCbCr values a fed through equation 1 and the resulting RGB output vector is obtained, ready for display on a PC monitor.

Vertical Field Interpolation

The uCFG was designed to freeze a single field of video data into its internal FIFO buffer. If this field is downloaded and displayed as-is on a PC screen, it appears squashed vertically. This is easy to understand since the full video image really requires two interlaced fields to replicate the original

image in its proper aspect ratio. One possibility is to acquire an ODD field, download it into every odd line of the PC's image array after RGB conversion, then do the same for the EVEN field. In this way, a full 720x480 resolution image is obtained. Of course, if there is any motion in the scene, there will be motion artifacts in the final image.

Another approach is the infer the missing field of video from the downloaded one. This is done by simply filling the destination PC image buffer's odd lines with the downloaded image data. The next step involves scanning through the missing lines in the image buffer and at every pixel location (and for every component), the following 2D filter kernel is applied:

```
[0.125 0.250 0.125]
[0.000   1     0.000]
[0.125 0.250 0.125]
```

Essentially, this kernel is a weighed average of the top, bottom, and diagonal neighbors of the missing samples. The resulting output image has full 720x480 resolution ready to be displayed on the PC screen with the correct aspect ratio.

Trigger Latch Operation

The uCFG provides flexible triggering techniques. Please note that using hardware triggers is NOT necessary to trigger the uCFG board. All the low level image acquisition functions can be triggered and controlled through simple ASCII serial commands such as CSEL and GRAB. In some applications, however, the operation of the uCFG is much simplified by using hardware or software based triggering. In response to trigger *n* (*n* in the range [1..4]), the uCFG board will issue a CSEL *n* to select the triggered channel of video, then a GRAB command. No replies are issued on the serial port. GRAB will acquire an odd or even field based on the last used setting. Odd is the power up default.

The behavior of the triggers depends on the trigger type setting. In retriggerable mode, a trigger will grab a field of video on the corresponding channel at any point in time. In one-shot mode, the triggers are disabled following the very first trigger activation and field grab. The ID of that trigger is latched in the trigger latch which can be read through the TRLR command. To re-enable the triggers, simply clear the trigger latch with the TRLC command. Please note the trigger latch always contains the ID of the last trigger that was activate, regardless of the mode (one-shot or retriggerable) and, as such, can be polled as often as necessary. In the unlikely event of simultaneous triggers, there is a priority structure in place. Trigger 1 has the highest priority followed by 2, 3, and 4. Triggers that were pre-empted or that occurred while the triggers were disabled are lost.

Example:

```
// The board has been setup in one-shot mode in EEPROM

TRLC      // Clears the trigger latch

TRLR      // Read the trigger latch
*000      // Trigger latch is clear as expected

TRIG 3    // Send the soft trigger 3 command which has
          // the same effect as activating trigger 3 in hardware

TRLR      // Read the trigger latch
*004      // Bit position 3 indicates that trigger 3 was last activated
          // and a fresh image from video input 3 now waits in the FIFO

TRIG 2    // Send the soft trigger 2 command which has
          // the same effect as activating trigger 2 in hardware

TRLR      // Read the trigger latch
*004      // As expected, because of the one-shot mode, the trigger 2 was
          // ignored and trigger 3 remains in the latch. FIFO contents remain
          // unchanged.

TRLC      // Now, clear the latch

TRLR      // Read the trigger latch
*000
```

```
TRIG 2    // Send the soft trigger 2 command which has
          // the same effect as activating trigger 2 in hardware

TRLR      // Read the trigger latch
*002      // Now, trigger 2 was latched in bit position 2
          // and a fresh image from video input 2 now waits in the FIFO
```

This example demonstrates the operation of the trigger latch logic. If the board had been setup in retriggerable mode, the activation of trigger 2 would have directly overwritten the original trigger 3 activation and the FIFO would hold a fresh video 2 image.

Schematics

(Full up-to-date circuit schematics are available with the uCFGEVAL kit.)

Liability Disclaimer

Digital Creation Labs Incorporated gives no warranty either expressed or implied and specifically disclaims all other warranties, including warranties for merchantability and fitness. In no event shall the liability of Digital Creation Labs Incorporated exceed the buyer's purchase price nor shall Digital Creation Labs Incorporated be liable for any indirect or consequential damages.

Life Support Policy

Digital Creation Labs Incorporated's products are not intended or authorized for use in any critical medical, life support, or life-saving devices or applications or any other system whose failure to perform correctly could result in life support failure.

FCC, CE, IC Certification

Digital Creation Labs Incorporated's products are provided as OEM unfinished products to be integrated into a customer's end product design. The customer is therefore responsible to test for, and obtain all the required EMI/EMC certifications of their final product wherever applicable.

Trademarks

Microcontroller Frame Grabber, uCFG and uCFGEVAL are all Trademarks of Digital Creation Labs Incorporated.