

Auto Graphics Mode for Non Supported Video Formats

1. Introduction

The purpose of this engineering note is to assist the user of the ADV7844/ADV7842 to configure the Component Processor (CP) core to process HD, ED, and graphics standards that are not supported by the primary mode and video standard controls.

For example, it is possible to program the CP to support other component and graphics standards not directly supported using the video standard, such as MAC 13 graphics formats, and other formats that the CP can support if configured correctly.

In standard operation, the primary mode and video standard controls configure the CP core to process the most common HD, ED, SD, and RGB graphics formats (to check which modes are supported, refer to the primary mode and video standard selection table in the Hardware Manual).

This engineering note provides a detailed description and worked examples of how to configure the CP core to process non standard video formats i.e. those formats not supported via PRIM_MODE and VID_STD.

Pr.0

Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use, nor for any infringements of patents or other rights of third parties that may result from its use. Specifications subject to change without notice. No license is granted by implication or otherwise under any patent or patent rights of Analog Devices. Trademarks and registered trademarks are the property of their respective owners.

2. Auto Graphics Mode Configuration

The following steps are required to correctly configure the CP to process non standard video formats:

1. Enter auto graphics mode.
2. Generate the pixel clock.
3. Configure the expected free run line length.
4. Set the expected number of free run lines per field.
5. Configure the interlaced or progressive parameter.
6. Manually adjust the output vertical and horizontal alignment if required.
7. Insert time codes if required.
8. Set input color space¹
9. Adjust digital clamping position²

¹ Required for component inputs only

² Required for HD component inputs only, due to their tri-level syncs.

2.1. Entering Auto Graphics Mode

To process a signal not supported automatically by the primary mode and video standard controls:

1. Program the primary mode to graphics: `PRIM_MODE[5:0] = 0b0010`
2. Program the video standard to auto graphics: `VID_STD[3:0] = 0b00111`

2.2. Generating the Pixel Clock

A PLL is used to synthesize a pixel clock (TLLC) from the incoming horizontal syncs. For a non standard video format, the PLL can be configured manually to derive a pixel clock of any frequency. This is achieved by programming the PLL feedback divider block.

Firstly, `PLL_DIV_MAN_EN` is set to 1 to enable manual programming of the PLL block.

One of the two following methods is used to calculate the value of `PLL_DIV_RATIO[12:0]` to give the required pixel clock. Method one or method two is chosen, depending on the information available about the non standard format.

Method one uses the rule that the `PLL_DIV_RATIO[12:0]` is always equal to the number of total luma samples per line.

Equation 1 describes method two, where the pixel clock frequency is divided by the incoming Hsync frequency. This equation describes the multiplying process of the PLL to generate a pixel clock from the incoming Hsyncs.

$$PLL_DIV_RATIO[12:0] = \left[\frac{F_{pixelclock}}{F_{Hsync}} \right]$$

Equation 1. Calculating `PLL_DIV_RATIO[12:0]`

Engineering Note

Note that because the PLL divide ratio is located in multiple registers, it cannot be updated in one I2C register write. To avoid an instantaneous incorrect value, the circuit is designed so that the full 13 bits must be written to in sequence, one register after the other, with no interruption between the two writes.

2.3. Configuring the Free Run Mode Line Length

The expected line length for non standard formats must be programmed to the CP core. [FR_LL\[10:0\]](#) (free run line length) is the number of system clock cycles in the ideal line length of the video format.

Equation 2 shows that to calculate the [FR_LL \[10:0\]](#) manual parameter, the line period is divided by the system clock period. The numerator in this equation can be calculated either directly from the Hsync period or by using the total number of luma pixel periods per line multiplied by the pixel clock period.

Note that because the free run line length is located in multiple registers, it cannot be updated in one I2C register write. To avoid an instantaneous incorrect value, the circuit is designed so that the full 11 bits must be written to in sequence, one register after the other, with no interruption between the two writes.

$$FR_LL[10:0] \left[\frac{T_{line}}{T_{28.6363MHz}} \right]$$

Equation 2. Calculating FR_LL[10:0]

2.4. Setting the Free Run Mode Number of Lines per Field

The expected number of lines per field for non standard formats must be programmed to the CP core. The control [LCOUNT_MAX\[10:0\]](#) is used to manually set the number of lines per field and is programmed with the ideal number of lines per field for the required standard.

Note that because the number of lines per field is located in multiple registers, it cannot be updated in one I2C register write. To avoid an instantaneous incorrect value, the circuit is designed so that the full 11 bits must be written to in sequence, one register after the other, with no interruption between the two writes

2.5. Configuring the Interlaced or Progressive Parameter

The expected interlaced or progressive configuration must be programmed to the CP core for non standard formats.

For progressive modes, the [INTERLACED](#) bit (CP Map, Address 0x91, [6]) is set to 0.

For interlaced modes, the [INTERLACED](#) bit is set to 1.

2.6. Adjusting the Vertical or Horizontal Alignment

Engineering Note

When programming the CP core for a non supported standard, it is possible that the horizontal and vertical alignment of the output video is not aligned as required. If AV codes are required, it is possible to adjust manually the output vertical and horizontal alignment as follows:

- Adjust the horizontal position and width by varying the values in [DE_H_START\[9:0\]](#) and [DE_H_END\[9:0\]](#)
- Adjust the vertical position and height by varying the values in [DE_V_START\[3:0\]](#) and [DE_V_END\[3:0\]](#)
- Adjust the VBI start and end position by varying the values in [CP_START_VBI\[11:0\]](#) and [CP_END_VBI\[11:0\]](#) (and [CP_START_VBI_EVEN\[11:0\]](#) and [CP_END_VBI_EVEN\[11:0\]](#) for interlaced standards)
- Adjust the EAV and SAV code position by varying the values in [CP_START_EAV\[11:0\]](#) and [CP_START_SAV\[11:0\]](#)

2.7. Time Code Insertion

In the event that it is required to insert time codes and/or blank pixel data, the ADV7844/ADV7842 cannot determine the start and end of active video on each horizontal line nor the start and end of the VBI region. The user must set these manually using the following controls

- [CP_START_SAV](#)
- [CP_START_EAV](#)
- [CP_START_VBI](#) and ([CP_START_VBI_EVEN](#))
- [CP_END_VBI](#) and ([CP_END_VBI_EVEN](#))

2.7.1. [CP_START_SAV](#) and [CP_START_EAV](#)

[CP_START_SAV](#) and [CP_START_EAV](#) influence the time code insertion and vertical blanking as follows:

[CP_START_SAV](#) sets the position of the start of the active video (SAV) code. This control varies the total number of pixels between the leading edge of output HS and the start of active video within a horizontal line minus 4 pixels (the length of the SAV code). [CP_START_EAV](#) sets the position of the end of active video (EAV) code. This control varies the total number of pixels between the leading edge of output HS and the end of active video within a horizontal line of video.

The [CP_START_SAV](#) and [CP_START_EAV](#) controls operate by increasing or decreasing the number of blanking samples between EAV and SAV.

[CP_START_SAV](#) and [CP_START_EAV](#) (in pixels) can be calculated as follows:

$CP_START_SAV = HSYNC_WIDTH + HSYNC_BACK_PORCH - 4$
 $CP_START_EAV = TOTAL_LINE_WIDTH - HSYNC_FRONT_PORCH$
where $TOTAL_LINE_WIDTH = PLL_DIV_RATIO$ (in pixels)

Manipulating [CP_START_SAV](#) and [CP_START_EAV](#) does not change the position of the H_Sync. The ADV7844/ADV7842 will adjust the output pixel data to ensure the correct sample is sent first following manipulating [CP_START_SAV](#) and [CP_START_EAV](#). However, when moving EAV and SAV, it must be ensured that no pixel corruption is introduced by partially blanking a pixel i.e. if a pixel contains 4 samples, blanking three of them will cause a corrupted pixel.

The affect of [CP_START_SAV](#) and [CP_START_EAV](#) is illustrated in Figure 1.

Engineering Note

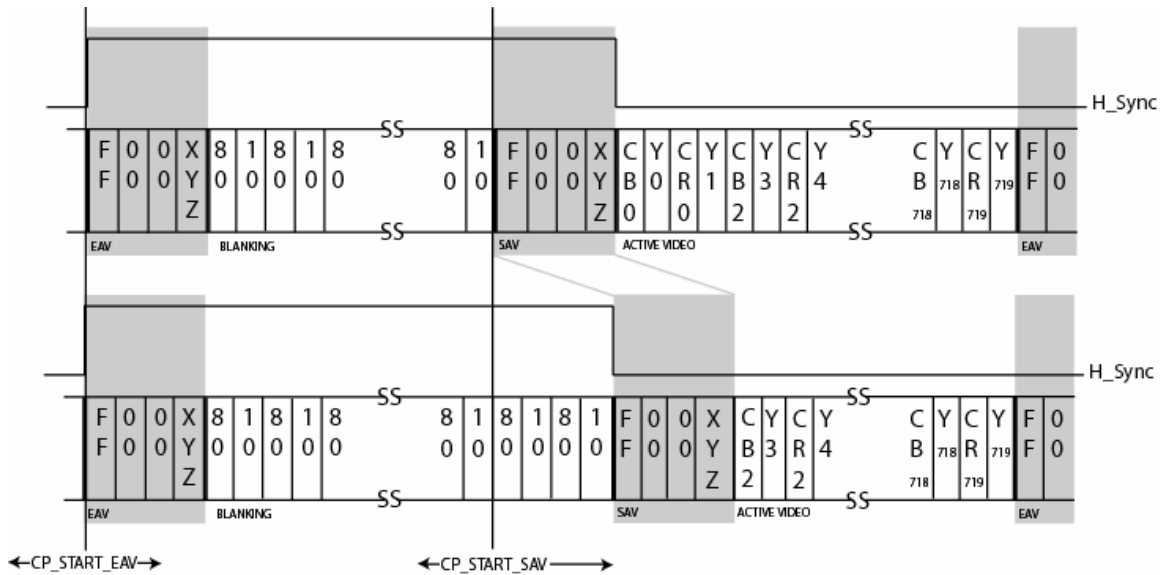


Figure 1. CP_START_SAV and CP_START_EAV

2.7.2. CP_START_VBI and CP_END_VBI

CP_START_VBI sets the position of the start of the vertical blanking interval (VBI). This control varies the number of VBI lines at the end of a frame of a non interlaced standard or the number of VBI lines at the end of a frame of an interlaced standard. **CP_END_VBI** sets the position of the end of the vertical blanking interval (VBI). This control varies the number of VBI lines at the start of a frame of a non interlaced standard or the number of VBI lines at the start of a frame of an interlaced standard.

CP_START_VBI and **CP_END_VBI** (in lines) can be calculated as follows:

$CP_START_VBI = TOTAL_NUMBER_LINES - VSYNC_FRONT_PORCH + 1$ (in a field or in field 1 for interlaced inputs)

$CP_END_VBI = VSYNC_WIDTH + VSYNC_BACK_PORCH + 1$ (in a field or in field 1 for interlaced inputs)

$CP_START_VBI_EVEN = FIELD1_TOTAL_LINES - rounded_down_VSYNC_FRONT_PORCH_FIELD2 + 1$

$CP_END_VBI_EVEN = FIELD1_TOTAL_LINES + VSYNC_WIDTH_FIELD2 + rounded_up_VSYNC_BACK_PORCH_FIELD2 + 1$

where $TOTAL_NUMBER_LINES = LCOUNT_MAX$ (in a field or in field 1 for interlaced inputs)

and where $CH2_FR_FIELD_LENGTH = TOTAL_NUMBER_LINES$ (in field 2 for interlaced inputs)

and where field 1 is the odd field and field 2 is the even field

Manipulating **CP_START_VBI** and **CP_END_VBI** will impact the DE signal output from the CP core – DE will not become elongated (it will remain high for the same active pixel duration) but will toggle on more/fewer lines depending on whether VBI is increased/decreased.

If the incoming video format is interlaced, **CP_START_VBI** and **CP_END_VBI** indicate the values for the odd field - **CP_START_VBI_EVEN** and **CP_END_VBI_EVEN** specify the values for the even field.

The affect of **CP_START_VBI** and **CP_END_VBI** is illustrated in Figure 2.

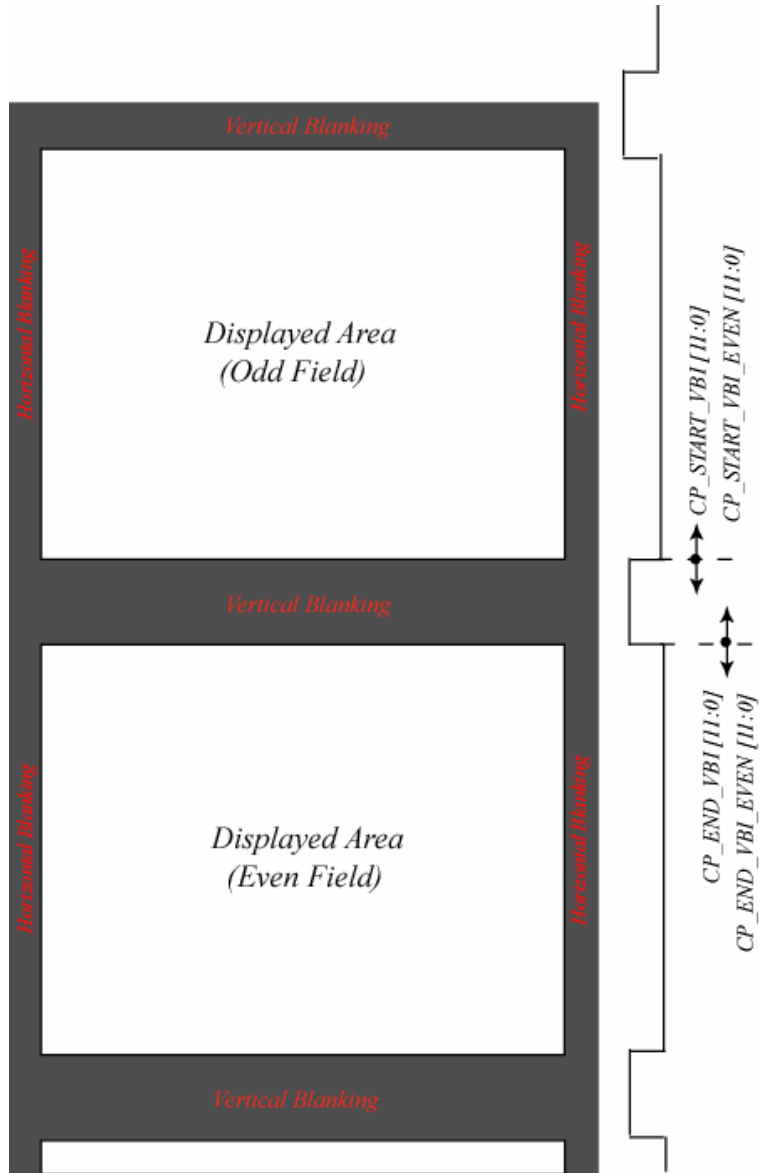



Figure 2. CP_START_VBI and CP_END_VBI

Engineering Note

2.8. Component Clamping

When using autographics on an analog component input, the user must set [INP_COLOUR_SPACE](#) to adjust the voltage clamp levels for the incoming signal.

INP_COLOR_SPACE[3:0]	Description
0000	Forces RGB (16-235) input color space
0001	Forces RGB (0-255) input color space
0010	Forces YCrCb 601 (16-235) input color space
0011	Forces YCrCb 709 (16-235) input color space
0100	Forces xvYCC 601 input color space (the CSC is effectively bypassed for this input color space configuration)
0101	Forces xvYCC 709 input color space (the CSC is effectively bypassed for this input color space configuration)
0110	Forces YCrCb 601 (0-255) input color space
0111	Forces YCrCb 709 (0-255) input color space
1111 	Automatic input color space selection depending on VID_STD[5:0] and PRIM_MODE[3:0]
1xxx	Undefined

2.9. HD Tri-Level Syncs

When using autographics on a HD (720p/1080i/1080p) analog component input, the user must set the voltage clamp start position for the incoming signal. The start position of the digital fine clamp must be adjusted by setting: `dfc_pos_start` (0xC8 [7:0], CP Map) = 64 dec.

Engineering Note

3. Worked Examples

Example 1.

4.4 1920x1080i @59.94/60 Hz (Format 5)

This format is available only in a 16:9 aspect ratio. This timing is based on CEA-770.3-C [21], but there are two differences: First, CEA-770.3-C uses tri-level sync, while CEA-861-D uses bi-level. Bi-level sync timing is accomplished using the second half of the CEA-770.3-C tri-level sync, defining the actual sync time to be the rising edge of that pulse. See Figure 3 .

Second, CEA-770.3-C uses a composite sync while CEA-861-D uses separate sync signals, thus eliminating the need for serrations during vertical sync.

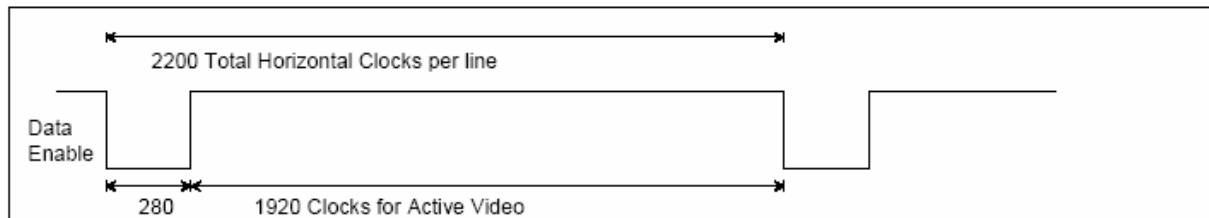


Figure 3 Excerpt from 1080i50/60Hz CEA Specification

Follow these steps:

- Program the primary mode to graphics: PRIM_MODE[5:0] = 0b0010
- Program the video standard to auto graphics: VID_STD[3:0] = 0b00111
- Enable and program manual PLL divider ratio = $F_{pixelclk}/F_{hsync} = 74.25 \text{ MHz}/33.75 \text{ kHz} = 2200$ (898 Hex)
- Free run line length = $F_{system_clock}/F_{hsync} = 28.636363 \text{ MHz}/33.75 \text{ kHz} = 848$ (350 Hex)
- CP_LCOUNT_MAX = Number of lines per frame = 1250
- Interlaced = 1
- INP_COLOR_SPACE = 0111 = YPbPr 709 Full Range
- CP_DFC_POS_START = 0x40

HSYNC_WIDTH = 44 pixels

HSYNC_BACK_PORCH = 148 pixels

TOTAL_LINE_WIDTH = 2200

HSYNC_FRONT_PORCH = 88

- CP_START_SAV = $44 + 148 - 4 = 188$ (0BC Hex)
- CP_START_EAV = $2200 - 88 = 2112$ (840 Hex)

FIELD1_TOTAL_LINES = 562

FIELD2_TOTAL_LINES = 563

LCOUNT_MAX = TOTAL_NUMBER_OF_LINES = FIELD1_TOTAL_LINES + FIELD2_TOTAL_LINES = 562 + 563 = 1125

VSYNC_FRONT_PORCH_FIELD1 = 2

VSYNC_FRONT_PORCH_FIELD2 = 2.5

VSYNC_FIELD_WIDTH1 = 5

VSYNC_FIELD_WIDTH2 = 5

VSYNC_BACK_PORCH_FIELD1 = 15

VSYNC_BACK_PORCH_FIELD2 = 15.5

- CP_START_VBI = $1125 - 2 + 1 = 1124$ (field 1) (464 Hex)

Engineering Note

- $CP_END_VBI = 5 + 15 + 1 = 21$ (field 1) (16 Hex)
- $CP_START_VBI_EVEN = 562 - 2 + 1 = 561$ (field 2) (231 Hex)
- $CP_END_VBI_EVEN = 562 + 5 + 16 + 1 = 584$ (field 2) (248 Hex)

Example 2.

UXGA Graphics. 1600 X 1200 Active Pixels. 2160 X 1250 Total Pixel. Pixel clock = 162MHz. $F_{hsync} = 75\text{kHz}$

Follow these steps:

- Program the primary mode to graphics: $PRIM_MODE[5:0] = 0b0010$
- Program the video standard to auto graphics: $VID_STD[3:0] = 0b00111$
- Enable and program manual PLL divider ratio = Total Samples Per Line = 2160 (870Hex)
- Free run line length = $F_{system_clock}/F_{hsync} = 28.636363\text{ MHz}/75\text{ kHz} = 382$ (17E Hex)
- CP_LCOUNT_MAX = Number of lines per frame = 1250 (4E2 Hex)
- Interlaced = 0

Example 3.

1080p Component. 1920 X 1080 Active Pixels. 2200 X 1125 Total Pixel. Pixel clock = 148.5MHz. $F_{hsync} = 67.5\text{kHz}$

Follow these steps:

- Program the primary mode to graphics: $PRIM_MODE[5:0] = 0b0010$
- Program the video standard to auto graphics: $VID_STD[3:0] = 0b00111$
- Enable and program manual PLL divider ratio = Total Samples Per Line = 2200 (898Hex)
- Free run line length = $F_{system_clock}/F_{hsync} = 28.636363\text{ MHz}/67.5\text{ kHz} = 424$ (1A8 Hex)
- CP_LCOUNT_MAX = Number of lines per frame = 1125(465 Hex)
- Interlaced = 0
- $INP_COLOR_SPACE = 0111 = YPbPr$ (709 Full Range)
- $CP_DFC_POS_START = 0x40$

Engineering Note

4. Glossary

HD = High Definition
ED = Enhanced Definition
SD = Standard Definition
RGB = Analog Graphics

HS = Horizontal Sync
VS = Vertical Sync
DE = Data Enable

HSYNC_FRONT_PORCH = Portion of each line of video between the end of active video and the start of HS
HSYNC_BACK_PORCH = Portion of each line of video between the end of HS and start of active video
VSYNC_FRONT_PORCH = Portion of each frame of video between the end of active video and the start of VS
VSYNC_BACK_PORCH = Portion of each frame of video between the end of VS and start of active video

EAV = End of active video
SAV = Start of active video
VBI = Vertical Blanking Interval

TOTAL_LINE_WIDTH = Total number of pixels in one line of video
TOTAL_NUMBER_LINES = Total number of lines in one frame of video

PRIM_MODE = Primary Mode, IO map, 0x01[3:0]
VID_STD = Video Standard, IO map, 0x00[5:0]

PLL_DIV_MAN_EN = PLL divide ratio manual enable, IO map, 0x16[7]
PLL_DIV_RATIO = Manual PLL divide ratio setting, IO map, 0x16[4:0], 0x17[7:0]
FR_LL = Free run line length, CP map, 0x8F[2:0], 0x90[7:0]
CP_LCOUNT_MAX = Maximum total number of lines in a frame register, CP map, 0xAB [7:0], 0xAC [7:4]
INTERLACED = Interlaced or Progressive bit, CP map, 0x91[6]

DE_H_START = Vary the leading edge position of DE control register, CP map, 0x8B [3:2], 0x8D [7:0]
DE_H_END = Vary the trailing edge position of DE control register, CP map, 0x8B [1:0], 0x8C [7:0]
DE_V_START = Vary the start position of the VBI region control register, CP map, 0x8E [7:4]
DE_V_END = Vary the end position of the VBI region control register, CP map, 0x8E [3:0]

CP_START_VBI = Manual value for start of VBI position control register, CP map, 0xA5 [7:0], 0xA6 [7:4]
CP_END_VBI = Manual value for end of VBI position control register, CP map, 0xA6 [7:4], 0xA7 [7:0]
CP_START_VBI_EVEN = Manual value for start of VBI position in even fields control register, CP map, 0xA8 [7:0], 0xA9 [7:4]
CP_END_VBI_EVEN = Manual value for end of VBI position in even fields control register, CP map, 0xA9 [4:0], 0xAA [7:0]

CP_START_SAV = Manual value for SAV position control register, CP map, 0x26[4:0], 0x27[7:0]
CP_START_EAV = Manual value for EAV position control register, CP map, 0x28[4:0], 0x29[7:0]

INP_COLOR_SPACE = Control to select the color space conversion, IO map, 0x02[7:4]