

Implementing the Auto-Offset Function on the AD9980

by Del Jones

INTRODUCTION

The AD9980 incorporates an auto-offset function. The auto-offset works by calculating the required offset setting to yield a given output code during clamp. When auto-offset is enabled (reg. 0x1B:5 = 1), the settings in Registers 0x0B to 0x10 are used by the auto-offset circuitry as desired clamp codes (or target codes) rather than as an offset value. The circuit compares the output code after clamp (but still during the *back porch*) to the target code and adjusts the offset up or down to compensate. In auto-offset mode, the target code is an 11-bit twos complement word, with Bit 7 of 0x0B being the sign bit for the red channel (Bit 7 of 0x0D for green; Bit 7 of 0x0F for blue).

CHANGES TO REGISTER DEFINITIONS

The definition of registers 0x0B to 0x10 changes depending on whether the auto-offset function is enabled or not. The following figures describe these differences.

Address	Name	Value	Control
0B	Red Offset	0 1 0 0 0 0 0 0 0 0 0 0	64
0C		0	ALL: 64
0D	Green Offset	0 1 0 0 0 0 0 0 0 0 0 0	64
0E		0	64
0F	Blue Offset	0 1 0 0 0 0 0 0 0 0 0 0	64
10		0	64

Figure 1. AD9980 with Auto-Offset Disabled

- Each set of registers consist of seven bits (binary notation) that are used to adjust the offset of the ADC.

Address	Name	Value	Control
0B	Red Target Code	0 0 0 0 0 0 0 1 0	2
0C		0 0 0 0 0 0 0 0 0	4
0D	Green Target Code	0 0 0 0 0 0 0 1 0	2
0E		0 0 0 0 0 0 0 0 0	4
0F	Blue Target Code	0 0 0 0 0 0 0 1 0	2
10		0 0 0 0 0 0 0 0 0	4

Figure 2. AD9980 with Auto-Offset Enabled

- Each set of registers consist of nine bits (twos complement notation) that are used to adjust the target code of the ADC's output after clamp.

Brightness Adjustment

If auto-offset is disabled, the lower nine bits of the offset registers control the absolute offset added to the channel. The offset control provides a +63/−64 LSB of adjustment range, with 1 LSB of offset corresponding to 1 LSB of output code.

With auto-offset enabled, registers 0x0B to 0x10 now contain target codes instead of offset values. These registers can still be used to adjust brightness. The difference is that when auto-offset is enabled, the exact black code output is now being adjusted so that the user knows exactly what code the black level is being set to. This is particularly useful for applications that desire to match NTSC or other video specifications that use less than full-scale code ranges. For example, if there is a requirement to only use 75% of the code range (code range of 32 to 224) for the Y input of a component video system, the Y (green) target code can be set to 128 and the gain adjusted to achieve the 224 maximum output code. When developing software to control brightness, this must be taken into consideration.

Target Code Value Restrictions

Even though there are nine bits in the target code registers (eight bits plus one sign bit), there are only seven bits of actual adjustment range. Also, a target code of 0 is invalid. Therefore, the range of valid target codes is as follows:

- For ground-clamped inputs
 - −63 (1 1100 0001) to −1 (1 1111 1111)
 - 1 (0 0000 0001) to +63 (0 0111 1111)
- For midscale-clamped inputs
 - 64 (1 0100 0000) to 191 (0 1011 1111)

This should also be taken into consideration when developing software for brightness control.

USING AUTO-OFFSET

To activate the auto-offset mode, set register 0x1B, Bit 5 to 1. Next, the target code registers (0x0B to 0x10) must be programmed. The values programmed into the target code registers should be the output code desired from the AD9980 during the back porch reference time. For example, for RGB signals all three registers would normally be programmed to a very small code (4 is recommended but any code between ± 63 , except 0, is valid), while for YPbPr signals the green (Y) channel would normally be programmed to a very small code (4), and the blue and red channels (Pb and Pr) would normally be set to 128. For midscale-clamped inputs, any target code value between 64 and 192 is valid, although the AD9980's offset range may not be able to reach every value. Example register settings for the auto-offset mode are shown in Table I.

The ability to program a target code for each channel gives users a large degree of freedom and flexibility. While, in most cases, all channels will be set either to 4 or 128, the flexibility to select other values allows for the possibility of inserting intentional skews between channels. It also allows for the ADC range to be skewed so that voltages outside of the normal range can be digitized. (For example, setting the target code to 40 would allow the sync tip, which is normally below black level, to be digitized and evaluated.)

Table I. Example Register Settings for Enabling Auto-Offset

Reg.	RGB Auto-Offset Clamping			YPbPr Auto-Offset Clamping		
	Value	Description		Value	Description	
Rx0B	0x02	Red Offset MSB	Red Target Code = +4	0x40	Red Offset MSB	Red Target Code = +128d
Rx0C	0x00	Red Offset LSB		0x00	Red Offset LSB	
Rx0D	0x02	Green Offset MSB	Green Target Code = +4	0x02	Green Offset MSB	Green Target Code = +4
Rx0E	0x00	Green Offset LSB		0x00	Green Offset LSB	
Rx0F	0x02	Blue Offset MSB	Blue Target Code = +4	0x40	Blue Offset MSB	Blue Target Code = +128d
Rx10	0x00	Blue Offset LSB		0x00	Blue Offset LSB	
Rx18	**** 000*	Bits 3:1 = '000'. This sets all three channels to ground clamp.		**** 101*	Bits 3:1 = '101'. This sets all Red and Blue channels to midscale clamp and the Green channel to ground clamp.	
Rx1B	**1* ****	Bits 5 = '1'. This enables the auto-offset clamping.		**1* ****	Bits 5 = '1'. This enables the auto-offset clamping.	
Rx1B	***x x***	Bits 4:3 = '10'. This sets the auto-offset circuit to operate each 64 clamps. '00' = every clamp; '01' = 16 clamps; '11' = every V_{SYNC} .		***x x***	Bits 4:3 = '10'. This sets the auto-offset circuit to operate each 64 clamps. '00' = every clamp; '01' = 16 clamps; '11' = every V_{SYNC} .	

Clamp Timing

The internal logic for the auto-offset circuit requires from 16 to 40 data clock cycles (programmable in 8-clock steps via Bits 4:3 of Register 0x2E) to perform its function. This operation is executed immediately after the clamping signal is deactivated. Therefore, it is important to deactivate the clamp signal at least 16 to 40 (depending on 0x2E – 4:3) data clock cycles before active video. This is true whether using the AD9980’s internal clamp circuit or an external clamp signal. See Figure 3. Therefore, given these requirements, it must be ensured that

$$\text{Clamp Placement} + \text{Clamp Duration} + \text{Auto-Offset Evaluation Time} < \text{Back Porch Time}$$

Or, referring to Figure 3

$$B + C + D < A$$

The definition of Bits 4:3 of Register 0x2E is shown in Table II. To provide the maximum amount of clamping time, it is recommended to program these bits to 00.

Table II. Auto-Offset Evaluation Time Register Bit Definition

Register	Bits	Description
0x2E	4:3	Auto-Offset Evaluation Time 00 = 16 clocks 01 = 24 clocks 10 = 32 clocks 11 = 40 clocks

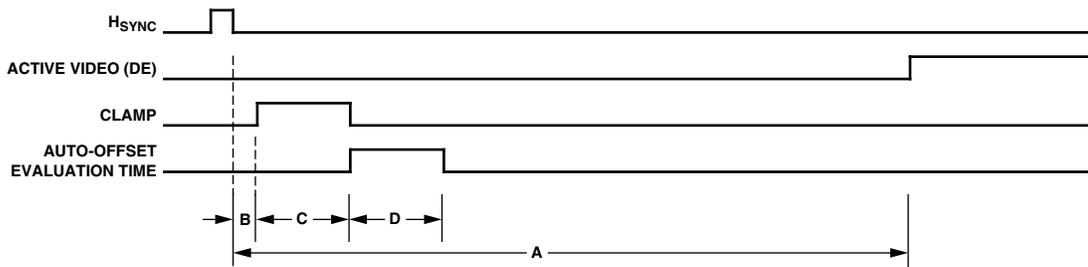
RECOMMENDED PROCEDURES

Auto-Offset in Continuous Update Mode

1. Set registers to enable auto-offset
 - a. 0x1B[5] = **11 1011—This enables AO with update every V_{SYNC}
 - b. 0x20[0] = 1
 - c. 0x29[1] = 0
 - d. 0x2E[4:3] = 00—Select shortest “Evaluation Time”
2. Set appropriate target code and clamp settings

Auto-Offset in Hold Mode

1. Set registers to enable auto-offset
 - a. 0x1B[5] = **11 1011—This enables AO with update every 64 clamps
 - b. 0x20[0] = 1
 - c. 0x29[1] = 0
 - d. 0x2E[4:3] = 00—Select shortest “Evaluation Time”
2. Set appropriate target code and clamp settings
3. Let run for at least 10 V_{SYNC} periods (~0.5 seconds)
4. Set the “Hold Auto-Offset” bit (0x2C[4]) to 1
5. Restart at mode change
 - a. May need updating under extreme environmental conditions in some applications



NOTES
 A. BACK PORCH TIME—VARIABLE, DEPENDING ON VIDEO MODE
 B. CLAMP START TIME (CLAMP PLACEMENT)—PROGRAMMABLE IN THE AD9980
 C. CLAMP TIME—PROGRAMMABLE IN THE AD9980
 D. TIME REQUIRED BY THE AUTO-OFFSET CIRCUIT—PROGRAMMABLE IN 8-CLOCK STEPS FROM 8 TO 40 PIXEL CLOCK PERIODS

Figure 3. Clamp Timing Requirements

