D
EYE PATTERN GENERATOR

D.1 INTRODUCTION

The MC145572 can provide the recovered eye pattern on a received baud–by–baud basis as a serial digital word once every 12.5 μs. Some applications may use this feature for monitoring performance. This appendix describes a circuit to receive the eye pattern word and convert it to an analog voltage for display on an oscilloscope.

This appendix includes the schematics and PALASM* 2 programmable logic equations to implement two versions of an eye pattern generator. One design requires manual scaling to the magnitude of the eye pattern data while the other design automatically scales to optimize the limited dynamic range of the digital–to–analog converter (DAC). The eye pattern generator takes the data available on the EYE DATA pin and the clocking from the SYSCLK pin and generates an analog eye pattern for display purposes. Note that BR14(b0) must be set to 1 to enable the EYE DATA and SYSCLK outputs.

D.2 DISCUSSION

The eye pattern data output from the MC145572 consists of the received 2B1Q quats after the echo canceling and DFE functions have been performed on the signal available at the RxP and RxN pins. The eye pattern data is output in digital form on the EYE DATA pin and is 19 bits long. It is in sign extended two's complement form. The eye pattern data generators described here use an 8–bit DAC (this is sufficient for acceptable display on an oscilloscope) to display the most significant 8 bits of data including the first sign bit, but not extended sign bits or less significant bits. The 8 bits are shifted into an 8–bit serial–to–parallel converter and are then latched into an Analog Device's AD557 digital–to–analog converter. When the resulting 8–bit window is correctly placed over the 19–bit eye data word, a full scale (approximately 1 or 2 V peak–to–peak) eye pattern signal is output from the AD557 clearly showing the 2B1Q quats. See Figure D–1 for an example of the 8–bit window positioned over a portion of the 19–bit eye data word. In this example, D16 happens to be the sign bit.

Two circuits are shown for decoding the eye pattern data available on the EYE DATA pin. The first method provides for manual positioning of the 8–bit DAC data window over the 19–bit eye data word. The manual positioning schematic is shown in Figure D–1. The second method provides for automatic or manual positioning of the 8–bit DAC data window over the 19–bit eye data word. The automatic positioning schematic is shown in Figure D–2. Whenever the manual method is used, the DIP switches can be changed to correctly position the 8–bit data window. Manual positioning capability is provided with the automatic positioning circuit since there may be some conditions in which the automatic positioner will not stabilize. DIP switches are shown but any convenient binary encoder may be used.

*PALASM is a registered trademark of Advanced Micro Devices.

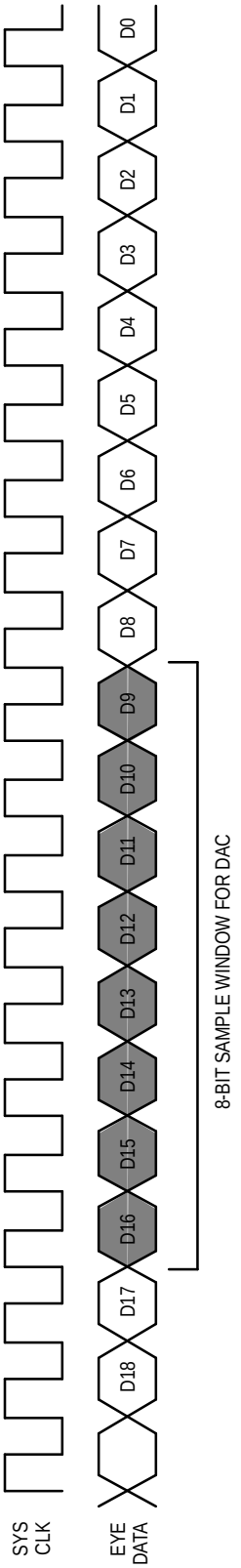

Figure D–1. 8–Bit Sample Window Positioned Over Bits D16:D9

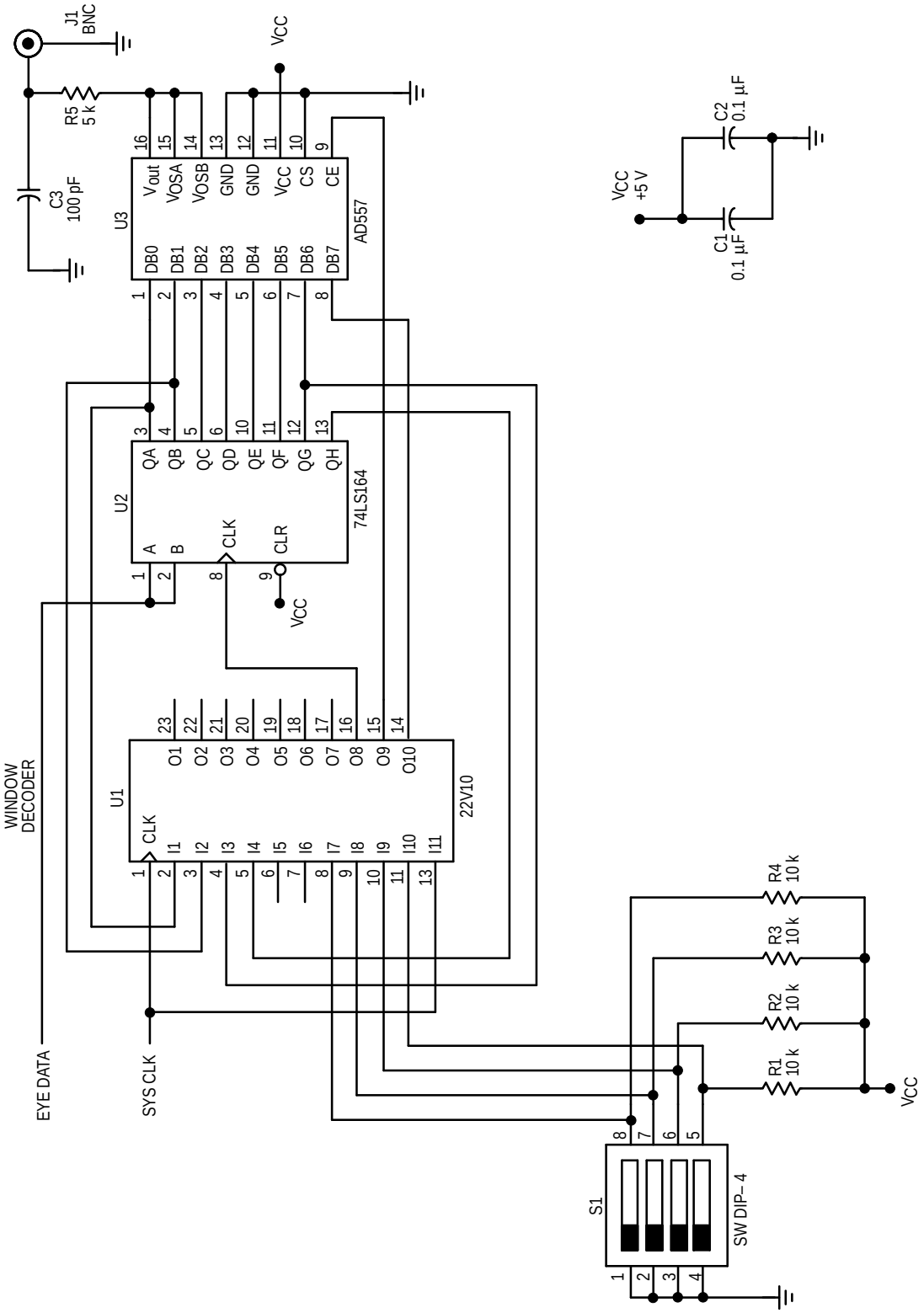

Figure D–2. Manual Eye Pattern Decoder

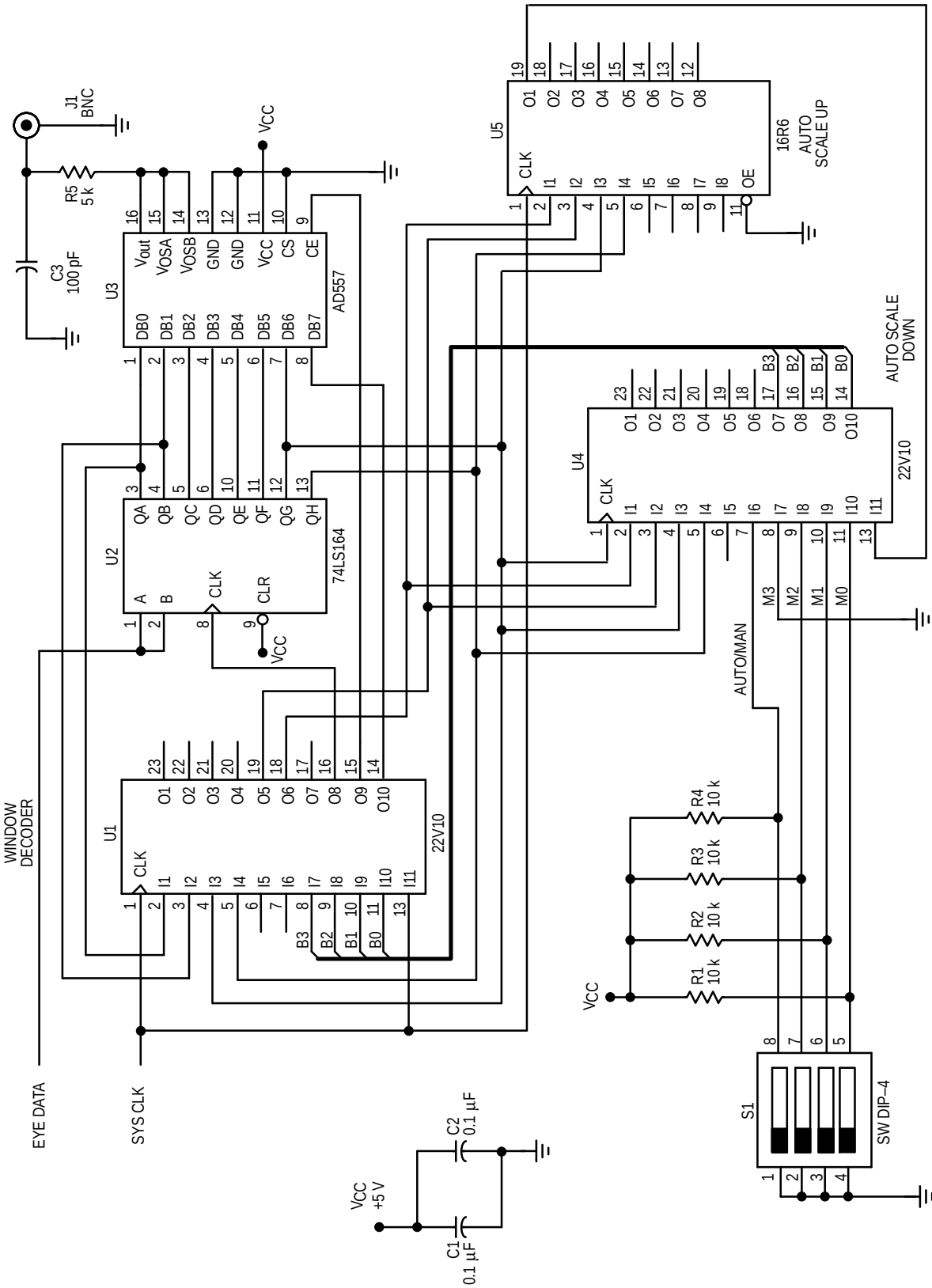

Figure D–3. Manual and Automatic Eye Pattern Decoder

D.3 WINDOW DECODER LOGIC EQUATIONS

The window decoder is used in both the manual and automatic circuits. It extracts the 8–bit data word to be displayed and provides control signals to the 74LS164 serial–to–parallel converter and to the AD557.

```
TITLE  WINDOW_DECODER      ; U1 IN THE SCHEMATIC
PATTERN      WINDEC.PL
REVISION     VER_1.0
COMPANY      MOTOROLA
CHIP   WIN_DECODER  PAL22V10
;PINS
;1     2      3      4      5      6      7      8      9      10     11     12
CLK    NC     SQ1    NC     SQ7    NC     NC     B3     B2     B1     B0     GND
;PINS
;13    14     15     16     17     18     19     20     21     22     23     24
ICLK   NSQ7   /ADC   SCLK   /ENA   LESS   Q4     Q3     Q2     Q1     Q0     VCC
EQUATIONS
       NSQ7   = /SQ7
       ADC    = /Q1 * Q2 * Q3 * Q4
                + /Q0 * Q2 * Q3 * Q4
       SCLK   = /ICLK * LESS
                + /ICLK * /Q4
                + /ICLK *  Q0 * Q1 * Q2 * Q3
       /ENA : = /SQ1  * /ENA
              +  Q0 *  Q1 *  Q2 *  Q3 * Q4
       LESS   =  Q4 * /Q3 *  B3
              +  Q4 * /Q3 * /B3 * /Q2 *  B2
              +  Q4 *  Q3 *  B3 * /Q2 *  B2
              +  Q4 * /Q3 * /B3 * /Q2 * /B2 * /Q1 *  B1
              +  Q4 * /Q3 * /B3 *  Q2 *  B2 * /Q1 *  B1
              +  Q4 *  Q3 *  B3 * /Q2 * /B2 * /Q1 *  B1
              +  Q4 *  Q3 *  B3 *  Q2 *  B2 * /Q1 *  B1
              +  Q4 * /Q3 * /B3 * /Q2 * /B2 * /Q1 * /B1 * /Q0 * B0
              +  Q4 * /Q3 * /B3 * /Q2 * /B2 *  Q1 *  B1 * /Q0 * B0
              +  Q4 * /Q3 * /B3 *  Q2 *  B2 * /Q1 * /B1 * /Q0 * B0
              +  Q4 * /Q3 * /B3 *  Q2 *  B2 *  Q1 *  B1 * /Q0 * B0
              +  Q4 *  Q3 *  B3 * /Q2 * /B2 * /Q1 * /B1 * /Q0 * B0
              +  Q4 *  Q3 *  B3 * /Q2 * /B2 *  Q1 *  B1 * /Q0 * B0
              +  Q4 *  Q3 *  B3 *  Q2 *  B2 * /Q1 * /B1 * /Q0 * B0
              +  Q4 *  Q3 *  B3 *  Q2 *  B2 *  Q1 *  B1 * /Q0 * B0
       Q0   : = ENA * /Q0
       Q1   : = ENA *  Q0 * /Q1
              + ENA * /Q0 *  Q1
       Q2   : = ENA *  Q0 *  Q1 * /Q2
              + ENA * /Q0 *  Q2
              + ENA * /Q1 *  Q2
       Q3   : = ENA *  Q0 *  Q1 * Q2 * /Q3
              + ENA * /Q0 *  Q3
              + ENA * /Q1 *  Q3
              + ENA * /Q2 *  Q3
       Q4   : = ENA *  Q0 *  Q1 * Q2 *  Q3 * /Q4
              + ENA * /Q0 *  Q4
              + ENA * /Q1 *  Q4
              + ENA * /Q2 *  Q4
              + ENA * /Q3 *  Q4
```

D.4 AUTOMATIC SCALE UP COUNTER LOGIC EQUATIONS

These equations shift the position of the 8–bit data window to the left when changes are detected in the most significant data bit.

```
TITLE         AUTO_SCALE_UP ; U5 IN THE SCHEMATIC
REVISION      VER_1.0
COMPANY       MOTOROLA
SCALE_UP      PAL16R6
;PINS
;1     2      3      4      5      6      7      8      9      10
CLK    LESS   WQ4    SQ6    SQ7    NC     NC     NC     NC     GND
;PINS
;11    12     13     14     15     16     17     18     19     20
NC     /UP    /QUP   /Q4    /Q3    /Q2    /Q1    /Q0    /CPU   VCC
STRING  RESET1  ' SQ6 * /SQ7 * WQ4 * /LESS * /QUP '
STRING  RESET2  ' /SQ6 * SQ7 * WQ4 * /LESS * /QUP '
EQUATIONS
       UP     = SQ6 *  SQ7 *  WQ4 * /LESS * /QUP
              + /SQ6 * /SQ7 *  WQ4 * /LESS * /QUP
       QUP  : =  WQ4 * /LESS
              +  WQ4 *  QUP
       CPU    =  UP * /Q0 * /Q1 * /Q2 * /Q3 * /Q4
       Q0   : = /Q0 *  UP
              +  Q0 * /UP
              +  RESET1 + RESET2
       Q1   : = /Q0 * /Q1 * UP
              +  Q0 *  Q1
              +  Q1 * /UP
              +  RESET1 + RESET2
       Q2   : = /Q0 * /Q1 * /Q2 * UP
              +  Q0 *  Q2
              +  Q1 *  Q2
              +  Q2 * /UP
              +  RESET1 + RESET2
       Q3   : = /Q0 * /Q1 * /Q2 * /Q3 * UP
              +  Q0 *  Q3
              +  Q1 *  Q3
              +  Q2 *  Q3
              +  Q3 * /UP
              +  RESET1 + RESET2
       Q4   : = /Q0 * /Q1 * /Q2 * /Q3 * /Q4 * UP
              +  Q0 *  Q4
              +  Q1 *  Q4
              +  Q2 *  Q4
              +  Q3 *  Q4
              +  Q4 * /UP
              +  RESET1 + RESET2
```

D.5 AUTOMATIC SCALE DOWN COUNTER LOGIC EQUATIONS

These equations shift the position of the 8–bit data window to the right when changes are not detected in the most significant data bits.

```
TITLE         AUTO_SCALE_DOWN     ; U4 IN THE SCHEMATIC
REVISION      VER_1.0
COMPANY       MOTOROLA
CHIP          SCALE_DOWN    PAL22V10
;PINS
;1     2      3      4      5      6      7      8      9      10     11     12
CLK    LESS   WQ4    SQ6    SQ7    NC     MAN    S3     S2     S1     S0     GND
;PINS
;13    14     15     16     17     18     19     20     21     22     23     24
/CPU   B0     B1     B2     B3     /Q2    /Q1    /Q0    /QDN   /DN    /CPD   VCC
STRING        RESET1  ' /SQ6 *  SQ7 * /LESS * WQ4 * /QDN '
              RESET2  '  SQ6 * /SQ7 * /LESS * WQ4 * /QDN '
EQUATIONS
       DN     = /SQ6 *  SQ7 * LESS * WQ4 * /QDN
              +  SQ6 * /SQ7 * LESS * WQ4 * /QDN
       QDN  : =  DN
              + /LESS * WQ4
              +  QDN  * WQ4
       CPD    =  DN * /Q0 * /Q1 * /Q2
       Q0   : = /Q0 *  DN
              +  Q0 * /DN
              +  RESET1 + RESET2
       Q1   : = /Q0 * /Q1 * DN
              +  Q0 *  Q1
              +  Q1 * /DN
              +  RESET1 + RESET2
       Q2   : = /Q0 * /Q1 * /Q2 * DN
              +  Q0 *  Q2
              +  Q1 *  Q2
              +  Q2 * /DN
              +  RESET1 + RESET2
       B0   : =  MAN * S0
              + /MAN *  CPU * /B0
              + /MAN *  CPD * /B0
              + /MAN * /CPU * /CPD * B0
       B1   : =  MAN *  S1
              + /MAN *  CPU *  B0  * /B1
              + /MAN *  CPD * /B0  * /B1
              + /MAN * /CPU * /CPD *  B1
              + /MAN *  CPU * /B0  *  B1
              + /MAN *  CPD *  B0  *  B1
       B2   : =  MAN *  S2
              + /MAN *  CPU *  B0  *  B1 * /B2
              + /MAN *  CPD * /B0  * /B1 * /B2
              + /MAN * /CPU * /CPD *  B2
              + /MAN *  CPU * /B0  *  B2
              + /MAN *  CPU * /B1  *  B2
              + /MAN *  CPD *  B0  *  B2
              + /MAN *  CPD *  B1  *  B2
       B3   :=  MAN *  S3
              + /MAN *  CPU *  B0  *  B1 *  B2 * /B3
              + /MAN *  CPD * /B0  * /B1 * /B2 * /B3
              + /MAN * /CPU * /CPD *  B3
              + /MAN *  CPU * /B0  *  B3
              + /MAN *  CPU * /B1  *  B3
              + /MAN *  CPU * /B2  *  B3
              + /MAN *  CPD *  B0  *  B3
              + /MAN *  CPD *  B1  *  B3
              + /MAN *  CPD *  B2  *  B3
```