Cyclic Redundancy Checking Ensures Correct Data Communications

By Ken Kavanagh

Electronic systems operating in industrial environments must often endure temperature extremes, electrically noisy environments, or other harsh conditions—nevertheless, it is critical that they work correctly. For example, if the data sent to a DAC controlling the position of a robotic arm were corrupted, the arm could move in an unintended direction. This could not only be dangerous but costly: imagine the arm smashing into the side of a new car on a production line—or, worse yet, into a production worker.

Several methods are available to ensure that the correct data has been received before action is taken. The simplest is for the controller to read back the data that was sent. If the received data doesn't match the sent data, one of them has been corrupted—and new data must be sent and verified. This method is reliable, but it also comes with a large overhead: each piece of data must be verified, doubling the amount of data transferred.

An alternative, *cyclic redundancy checking* (CRC), is to send a checksum with each packet of data. The receiving device will indicate if a problem has occurred, so the controller does not need to verify reception. Checksums are commonly generated by applying a polynomial equation to the data. CRC-8 produces an 8-bit checksum when applied to a 24-bit word. Combining the checksum with the data, transmitting all 32 bits to a device that can analyze the combination, and indicating errors that occur—though not a totally perfect solution—is more efficient than the write-and-read method.

Many Analog Devices DACs implement CRC in the form of a *packet error check* (PEC). 24-bit data is written when the PEC function is not required. To add the PEC function, the 24-bit data is augmented with a corresponding 8-bit checksum. If the received checksum does not agree with the data, an output pin is brought low to indicate the error. The controller clears the error, returns the pin high, and resends the data. Figure 1 shows an example of how the data is applied using an SPI interface. Table 1 lists a sample of Analog Devices parts that can use packet error checking.

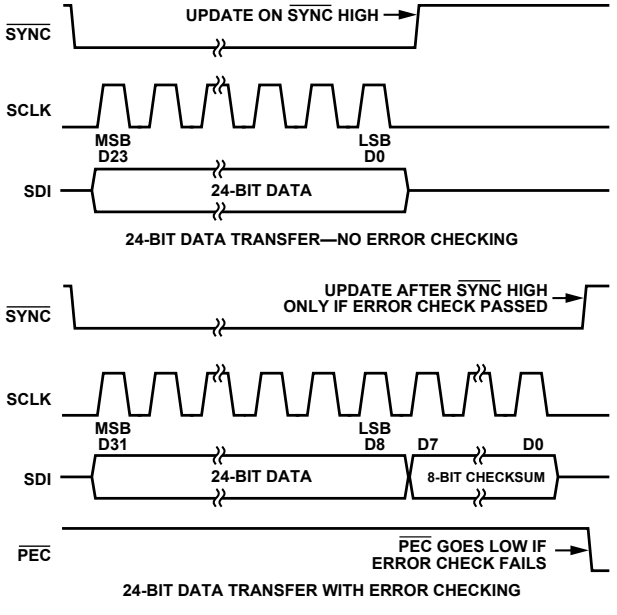

Figure 1. SPI write with and without packet error checking.

Table 1. Examples of Analog Devices Parts That Use Packet Error Checking

Part Number	Description
AD5360/AD5361	16-channel, 16-/14-bit, ±10-V DACs
AD5362/AD5363	8-channel, 16-/14-bit, ±10-V DACs
AD5748	Industrial current/voltage output driver
AD5749	Industrial current output driver
AD5750/ AD5750-1	Industrial current/voltage output drivers with programmable ranges
AD5751	Industrial current/voltage output driver
AD5755/AD5735	4-channel, 16-bit, 4-mA to 20-mA current- and voltage-output DACs
AD5757/AD5737	4-channel, 16-bit, 4-mA to 20-mA current-output DACs
ADT7470	Temperature sensor hub and fan controller

Generating the Packet Error Checksum

The CRC-8 algorithm uses the polynomial $C(x) = x^8 + x^2 + x^1 + 1$. For $x = 2$, this is equivalent to the binary value 100000111. To generate the checksum, the 24-bit data is left-shifted by eight bits to create a 32-bit number ending in eight Logic 0s. The CRC polynomial is aligned so that its MSB is adjacent to the leftmost Logic 1 of the 32-bit data. An exclusive-or (XOR) function is applied to the data to produce a new (shorter) number. (Numbers that match give Logic 0, nonmatches give Logic 1.) The CRC polynomial is again aligned so that its MSB is adjacent to the leftmost Logic 1 of the first result, and the procedure is repeated. Eventually, the original data will be reduced to a value that is less than the CRC polynomial. This is the 8-bit checksum. Figure 2 demonstrates how the checksum is developed.

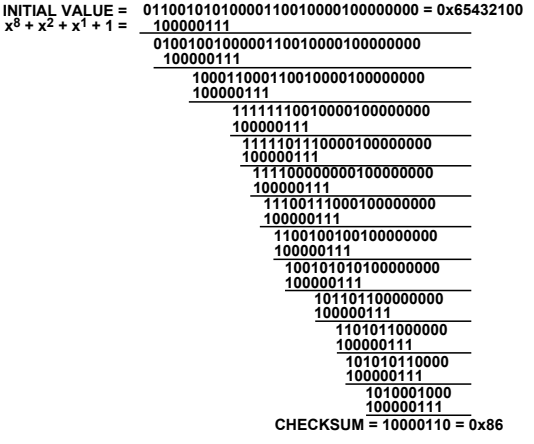

Figure 2. Generating a checksum for a 24-bit number (0x654321).

Conclusion

The example shown in Figure 2 uses the (hex) value of 0x654321 as a sample 24-bit data word. Applying the CRC-8 polynomial to the data generates a checksum of 0x86. When the data and checksum are sent to a compatible Analog Devices product, the data will be accepted only if both pieces of data arrive correctly. This method increases the reliability of data transfers and ensures that corrupted data is almost never accepted.

Author

Ken Kavanagh [ken.kavanagh@analog.com] is an applications engineer in ADI's Precision DAC Group. Currently responsible for providing application support for the *nano*DAC® and *dense*DAC™ portfolios, Ken has worked in applications since 1994. He earned a BEng from the University of Limerick in 1999.