Application Note: AZD040
IQ Switch® - ProxSense™ Series
Desgin Guidelines

This design guideline aids designers in using the 1-wire streaming protocol by making use of a UART peripheral.

1 Overview

1.1 What are the requirements for a UART peripheral to support the 1-wire streaming protocol?

The 1-wire streaming protocol operates in the same fashion as a UART protocol. Transmission starts with a start bit, followed by eight data bits and lastly a single stop bit. The following ICs use this protocol; refer to the relevant datasheet for exact byte contents.

- IQS127 S/D
- IQS128
- IQS132
- IQS133

After the first start condition (LOW) a synchronization byte (0xAA) is transmitted. This synchronization byte can be used to determine the baud rate of the transmission, or to verify the baud rate.

The baud rates of the ICs may vary due to changes in conditions such as temperature.

The UART required for this application should therefore allow for precise baud rate settings.

1.2 How to retrieve the data via the UART?

Upon capturing the data stream with most standard UARTs, the bit order of the data byte will be inverted. This can be adjusted manually, or by changing the Endian settings (if supported by the chosen UART).

To verify whether the UART is reading the data correctly just check the first byte of the data stream, which should be 0xAA. If 0x55 has been received the bit order needs to be flipped.

1.3 How to determine the start of a data transmission?

Each data stream of the 1-wire streaming protocol consists of a synchronization byte and 8 data bytes. There will at most be approx. 1ms between two of these data bytes. Between data transmissions will be approx. 9ms.

To ensure that the start of a data transmission is found, first ensure that the data transmission line is silent for at least 3-4ms. Once this has been done, the next level change of the data line will be the falling edge of the start condition of the synchronization byte.

2 Flow Diagrams

It is suggested to use the first data transmission to determine the baud rate. A timer can be used to determine the length of the synchronization byte, but it is important to disable all interrupts during this time.

Once the baud rate has been set, the data stream can be read using the UART, but always verify that the first byte has been read correctly. In the case that the first byte is not read correctly, a re-bauding routine should be executed to recalculate the baud rate used by the UART.

The following three flow diagrams illustrate such an implementation.

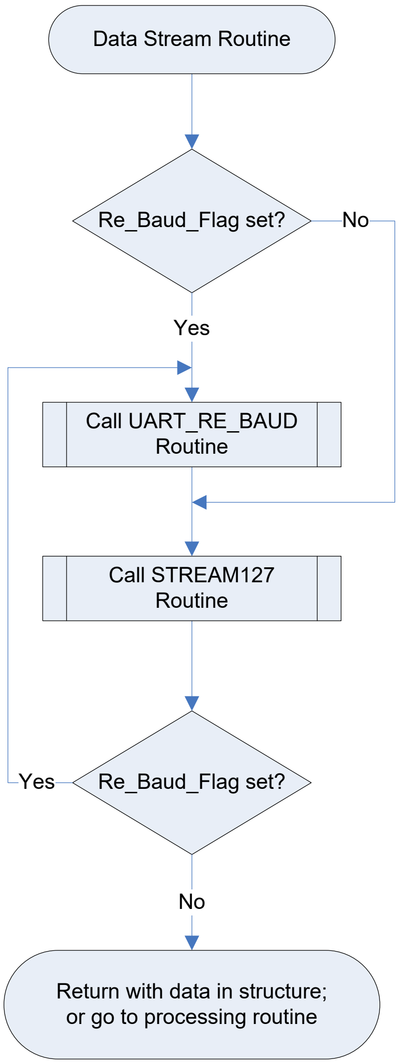

Figure 2.1 Data Stream Routine

As seen in this figure, the routine will check before every data packet whether it is necessary to re-baud or not. The RE_BAUD and STREAM127 routines are explained in the figures below.

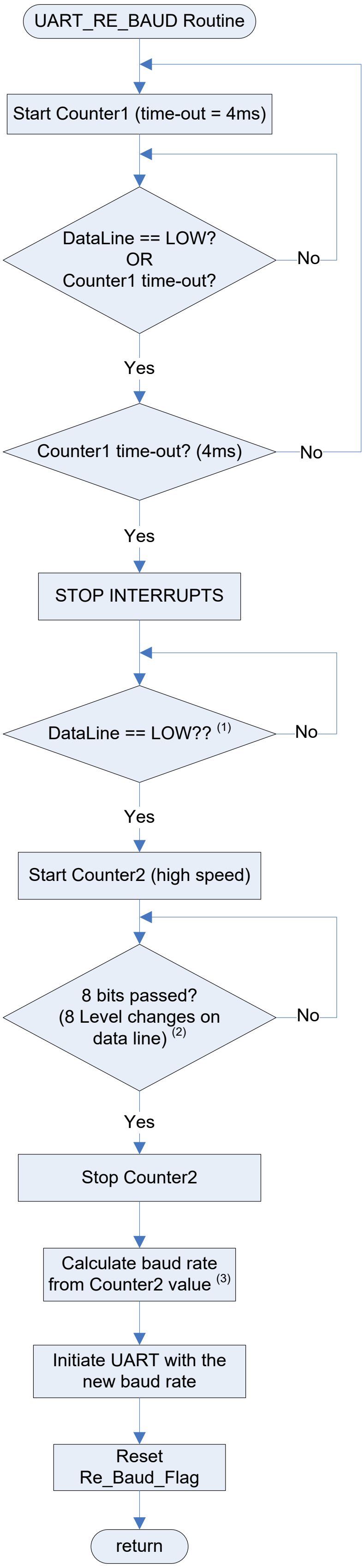

Figure 2.2 RE_BAUD routine

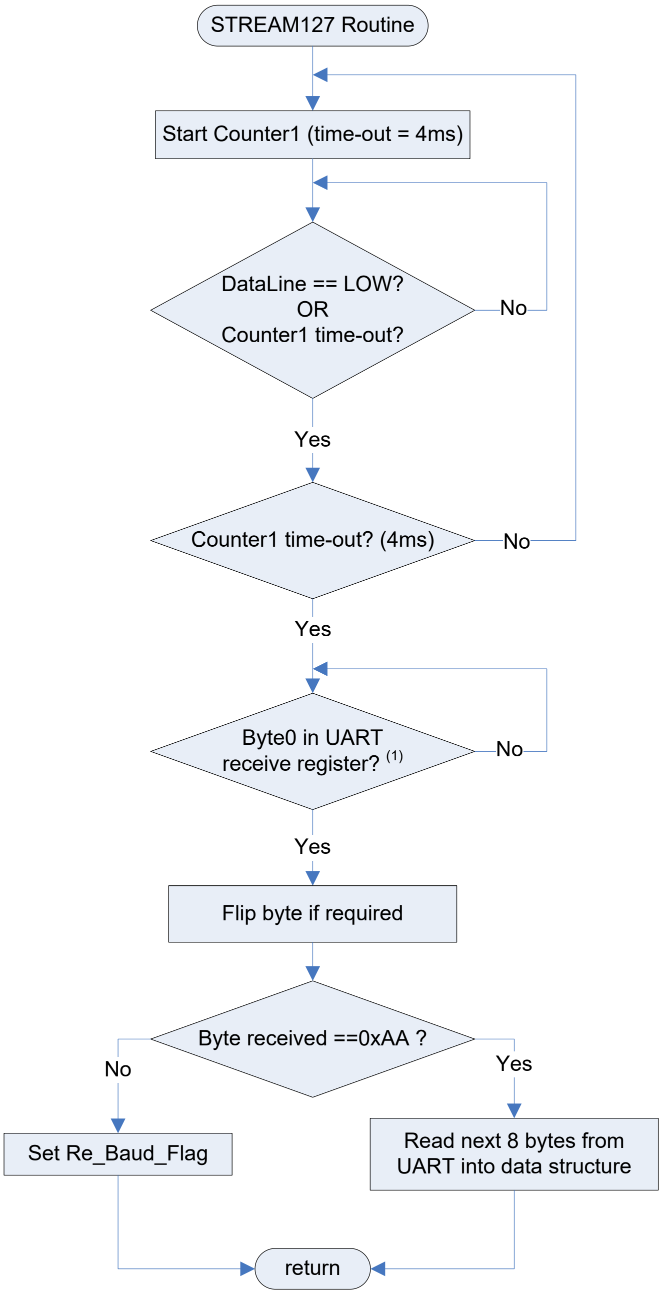

Figure 2.3 STREAM127 routine

(1)It is recommended to implement a watchdog timer at this point to ensure that the program will not freeze in the case of a malfunction.

(2)Note that the routine uses the start bit and the first 7 data bits to calculate the baud rate. This is done to use a total of 8 bits for easier calculations.

(3)Calculating the baud rate from the timer value will depend on how the microcontroller sets up the baud rate and on the speed of the timer. It is recommended to calculate a constant to be used for this calculation. I.e. calculate K such that- $baud\ rate = \frac{K}{timer\ \ result}$.

3 Example Code

This section gives sample code of a sample implementation of the 1-wire streaming protocol. It has been attempted to keep the code as general as possible, therefore it serves more as a logical guideline from which to develop code for a specific microcontroller.

- This section of code relates to the first flow diagram.

```
if (Re_Baud_Flag)
{
   while (RE_BAUD() == 1);
   Re_Baud_Flag = 0;
}
while (STREAM127(&OutBuffer) == 1);
```

- The RE_BAUD routine

```
char RE_BAUD(void)
{
 // Declare local variables

 // Setup timers 1 and 2.

 // Start Timer 1 (Timeout = 4ms)

   while (ReadInput(DataLine) && !Timer1Expired);
   if (!Timer1Expired)
   {
      return 1;      //return if not between transmissions
   }

 // disable interrupts

while (ReadInput(DataLine));      //wait for start condition

 // start counter2 (high speed counter)

 // wait for 8 edges

 // stop counter2 after last edge
```

```
 // get counter2 value

 // calculate baud rate from counter value

 // initialize UART with this baud rate

  return 0;
}
```

- The STREAM127 routine

```
char STREAM127(char *OutBuffer)
{
// Declare local variables

 // Setup timer 1.

 // Start Timer 1 (Timeout = 4ms)

  while (ReadInput(DataLine) && !Timer1Expired);
  if (!Timer1Expired)
  {
     return 1;          //return if not between transmissions
  }

  while (UART_Receive_buffer == empty);

  //flip UART_Received_Byte if needed

  If (UART_Received_Byte != 0xAA)
  {
     Re_Baud = 1;
  }

  for (i = 1; i <= 8; i++)
  {
     while (UART_Receive_buffer == empty);
```

```
    // flip UART_Received_Byte if needed
    OutBuffer[i] = UART_Received_Byte;
  }

  return 0;
}
```

Contact Information

	USA	Asia	South Africa
Physical Address	11940 Jollyville Suite 120-S Austin TX 78759 USA	Room 501A, Block A, T-Share International Centre, Taoyuan Road, Nanshan District, Shenzhen, Guangdong, PRC	1 Bergsig Avenue Paarl 7646 South Africa
Postal Address	11940 Jollyville Suite 120-S Austin TX 78750 USA	Room 501A, Block A, T-Share International Centre, Taoyuan Road, Nanshan District, Shenzhen, Guangdong, PRC	PO Box 3534 Paarl 7620 South Africa
Tel	+1 512 538 1995	+86 755 8303 5294 ext 808	+27 21 863 0033
Email	info@azoteq.com	info@azoteq.com	info@azoteq.com

Visit www.azoteq.com

for a list of distributors and worldwide representation.

Patents as listed on www.azoteq.com/patents-trademarks/ may relate to the device or usage of the device.