AN1273: Using the UART Interface on the Si34071 IEEE 802.3bt Compliant PD

This document provides information about the UART interface of the IEEE 802.3bt compliant Power over Ethernet (PoE) Si34071 device as well as some code examples on how to use it effectively with your host controller.

The Si34071 device with the surrounding BOM provides the necessary detection, up to 5-event classification and mark, and operating current levels compliant with IEEE 802.3bt Type 3 and Type 4 PoE standards.

The Si34071 provides various diagnostic messages through the UART Interface and offers control messages to turn on Si34071's Maintain Power Signature (MPS) feature. The UART Interface uses a command based protocol and utilizes only two pins which makes it easy to connect to the host controller.

KEY FEATURES

- Fully compliant IEEE 802.3bt PoE interface
- Various diagnostic messages through UART
- Supports standard and short Maintain Power Signature (MPS)

Table of Contents

1. Si34071 UART Interface and Commands

The Si34071 has a UART interface which offers asynchronous, full duplex communication. The UART baud rate is 9600 and it uses 8-N-1 protocol which means that every 10-bit frame has 1 start bit, 8 data bits, no parity bit and 1 stop bit. The UART port of the host controller has to follow the same configuration. The bit order on the RX and TX pins is little-endian which means LSB (Least Significant Bit) comes first. Byte-order with 2-byte responses is MSB, LSB which means that Most Significant Byte comes first.

The Si34071 uses a simple command interface without any termination character. The UART port on the Si34071 is enabled once the dc/dc is up and running and no communication is possible in the Detection, Classification and UVLO states. Once the command has been sent out by the host controller, the Si34071 receives it, executes it and, if required, responds with the corresponding 1 or 2 byte data. A summary of UART commands supported by the Si34071 is summarized in the table below. For a detailed description of the commands, see the Si34071 Data Sheet.

Table 1.1. Si34071 UART Commands

Command	Name	Type	Response	Description
0x01	CLASS	R	1 byte	Class requested and class assigned.
0x02	STATUS	R	1 byte	Class pulses detected and PSE MPS support.
0x03	VPWR	R	2 bytes	Input voltage after the rectifier (vpos-vneg).
0x04	MPS_ON	W	No Response	Enable MPS pulse generation.
0x05	MPS_OFF	W	No Response	Disable MPS pulse generation.

2. Si34071 UART Examples and Considerations

2.1 UART Test Configuration

The Si34071 UART port was connected to an 8-bit microcontroller. For that purpose, the SLSTK2030A LaserBee Starter Kit was used. The connection diagram of the system is based on the Si34071AC5V8KIT EVB and is presented in the following figure.

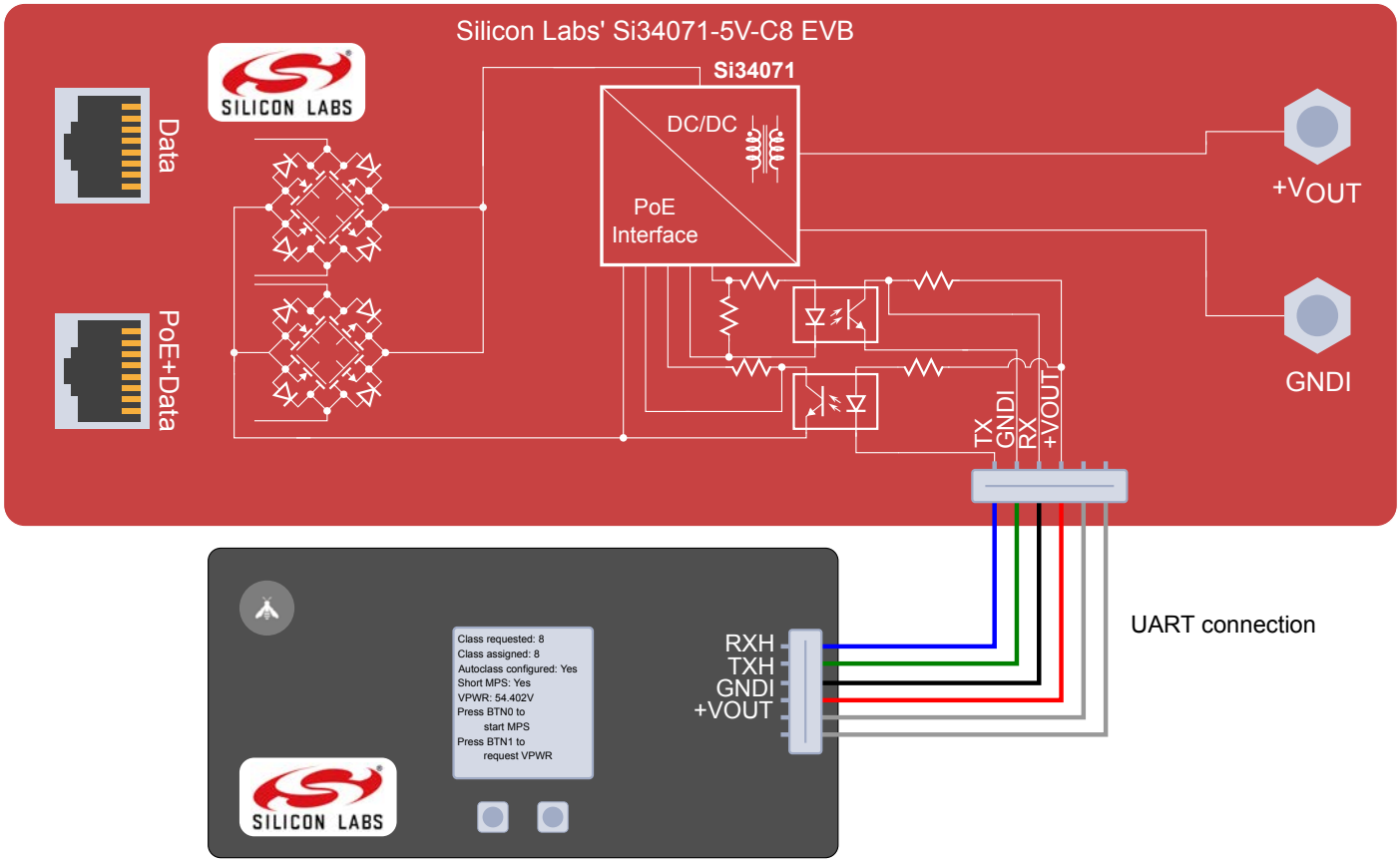


Figure 2.1. Connection Diagram for UART Test Configuration with the Si34071-5V-C8 EVB

2.2 UART Communication with Blocking Statements

The following code examples written in C are for demonstration purposes only and cannot be considered as a substitute for the application program. Please take into consideration that, based on the application MCU (host controller), the instruction set might differ from the one that is presented below, and the code should be adapted to the design accordingly.

2.2.1 Requesting 1-Byte Data from the Si34071 – Class Requested and Class Assigned

To use the UART interface with blocking statements, the simple steps below should be followed:

1. Put the command in the UART buffer.
2. Wait until the command is transmitted.
3. Clear the transmit flag manually if set by the hardware during transmit.
4. Wait until the response is received.
5. Get data from the UART buffer and process it.

```
// Put command 0x01 into UART buffer
SBUF1 = 0x01;
// Wait till it is transmitted
while (!SCON1_TI)
  ;
// Clear UART transmit flag
SCON1_TI = 0;
// Wait till data is received from Si34071
while (!SCON1_RI)
  ;
// Get data from UART buffer then process it
Byte = SBUF1;
```

2.2.2 Requesting 2-Byte Data from the Si34071 – Voltage between Vpos-Vneg Pins

To request a 2-byte data from the Si34071, a temporary variable shall be used to store the intermediate data during receive:

```
// Put command 0x03 into UART buffer
SBUF1 = 0x03;
// Wait till it is transmitted
while (!SCON1_TI)
  ;
// Clear UART transmit flag
SCON1_TI = 0;
// Wait till first byte is received from Si34071
while (!SCON1_RI)
  ;
// Get data from UART buffer and put it into a temporary variable
Byte1 = SBUF1;
// Wait till second byte is received from Si34071
while (!SCON1_RI)
  ;
// Get data from UART buffer then process it
Byte2 = SBUF1;
// Clear UART receive flag
SCON1_RI = 0;
// Process 2-byte response
vpwr = ((Byte1 * 256 + Byte2) * 148.2f + 11830);
```

2.2.3 Controlling MPS with the Si34071

The Si34071 automatically adjusts the MPS pulse length to the IEEE 802.3bt short MPS based on the length of the first classification pulse produced by the PSE. It is important to disable UART receive in the host controller before turning on the MPS functionality, as the transmit line in the Si34071 is used for MPS pulse generation which could result in invalid data reception in the host MCU.

```
// Disable UART receive before turning on MPS
SCON1 &= ~(SCON1_REN__RECEIVE_ENABLED);
// Put MPS ON command into UART buffer
SBUF1 = 0x04;
// Wait till data is transmitted
while (!SCON1_TI)
  ;
// Clear UART transmit flag
SCON1_TI = 0;
```

After sending the turn-off MPS command to the Si34071 it is necessary to re-enable the UART receive in the host controller. If MPS was turned off during a low pulse, which results in a low-level of the transmit line of the Si34071, it might be indicated as a start bit by the host controller. Therefore before executing any other command by the host MCU, it is recommended to get and discard any data that might be present in the UART receive buffer.

```
// Put MPS OFF command into UART buffer
SBUF1 = 0x05;
// Wait till data is transmitted
while (!SCON1_TI)
  ;
// Clear UART transmit flag
SCON1_TI = 0;
// Enable UART receive after turning off MPS
SCON1 |= (SCON1_REN__RECEIVE_ENABLED);
// IMPORTANT! Discard data from receive buffer
Byte = SBUF1;
// Clear UART receive flag
SCON1_RI = 0;
```

2.3 Interrupt-Driven UART Communication

Although the response time for UART communications of the Si34071 is only a few milliseconds, an interrupt-driven approach is a more elegant way to request and process data while the main cycle in the application MCU is allowed to process data and execute various tasks. For the following code snippets, a 4-byte deep, circular UART in and out buffer was implemented, but it might not be necessary as many controllers support hardware buffering and can be configured to use predefined buffers for communication.

2.3.1 Requesting 1-Byte Data from the Si34071

In an interrupt-driven environment UART output buffers are processed in the controller's interrupt routine. Similarly, the received data is put into the circular UART input buffer in the same route. The implemented UART input and output buffers are denoted as *ubi* and *ubo* respectively.

```
// UART Interrupt routine
SI_INTERRUPT (UART1_ISR, UART1_IRQn) {
  if (SCON1_RI == 1)
  {
    // Clear interrupt flag
    SCON1_RI = 0;
    // Read a character from UART
    Byte = SBUF1;
    // Put into the UART buffer input (ubi)
    Ring_Buf_Push(&ubi, Byte);
  }
  if (SCON1_TI == 1)
  {
    // Clear interrupt flag
    SCON1_TI = 0;
    // Pop a byte from the UART buffer output
    if (Ring_Buf_Pop(&ubo, &Byte))
    {
      // Transmit to UART
      SBUF1 = Byte;
    }
    else
    {
      // Set Tx_Ready to 1 if buffer is empty
      TX_Ready = 1;
    }
  }
}
```

A flag denoted as *TX_Ready* is used indicate whether UART transmission is in progress. If it is not the case and there is some unprocessed data in the output buffer then the transmit interrupt flag is set to 1 which triggers the interrupt routine to be called.

```
// Put data into UART buffer out (ubo)
// it will be processed in the interrupt
Ring_Buf_Push(&ubo, 0x01);
MSG = 0x01;
if (ubo.head != ubo.tail && TX_Ready)
{
  // Set the flag to zero indicating transmit
  // is in progress
  TX_Ready = 0;
  // Set transmit flag to 1
  SCON1_TI = 1;
}
Interrupts_Disabled();
// Pop data from UART input buffer (ubi), when received
// Buffer is filled up in the interrupt routine
Pop_Result = Ring_Buf_Pop(&ubi, &Byte1);
Interrupts_Enabled();
if (Pop_Result)
{
  if (MSG == 0x01)
  {
    //Process data here
  }
}
```

2.3.2 Requesting 2-Byte Data from the Si34071

Handling a 2-byte response needs a flag to indicate that the first data is processed and stored. Putting and getting data to and from the buffers are handled by the routines *Push(buffer_t*, uint8_t)* and *Pop(buffer_t*, uint8_t)* as presented above.

```
if (Pop_Result)
{
  if (MSG == 0x03)
  {
    // If first byte is not yet received, store it
    // in a temporary variable
    if (FirstReceived == false)
    {
      FirstReceived = true;
      Byte1 = Byte;
    }
    else
    {
    //Process data and set flag to false
      Byte2 = Byte;
      vpwr = ((Byte1 * 256 + Byte2) * 148.2f + 11830);
      FirstReceived = false;
    }
  }
}
```

2.4 Utilizing VPWR Measurements

The host controller can utilize the voltage measurement between the VPOS-VNEG pins by sending the VPWR command to estimate the voltage drop on the cable, which is connected to the PSE. If the measurement results show that a relatively short Ethernet cable is attached to the PD, the host controller might switch into a higher power state to utilize the full range of available power provided by the PSE.

2.5 Testing UART Communications

To check the UART transmission of the Si34071 any free Terminal is suitable which is capable of presenting UART messages in decimal or hexadecimal form. For testing purpose, a serial terminal program called RealTerm was used. The options for UART communication is shown in the following figure.

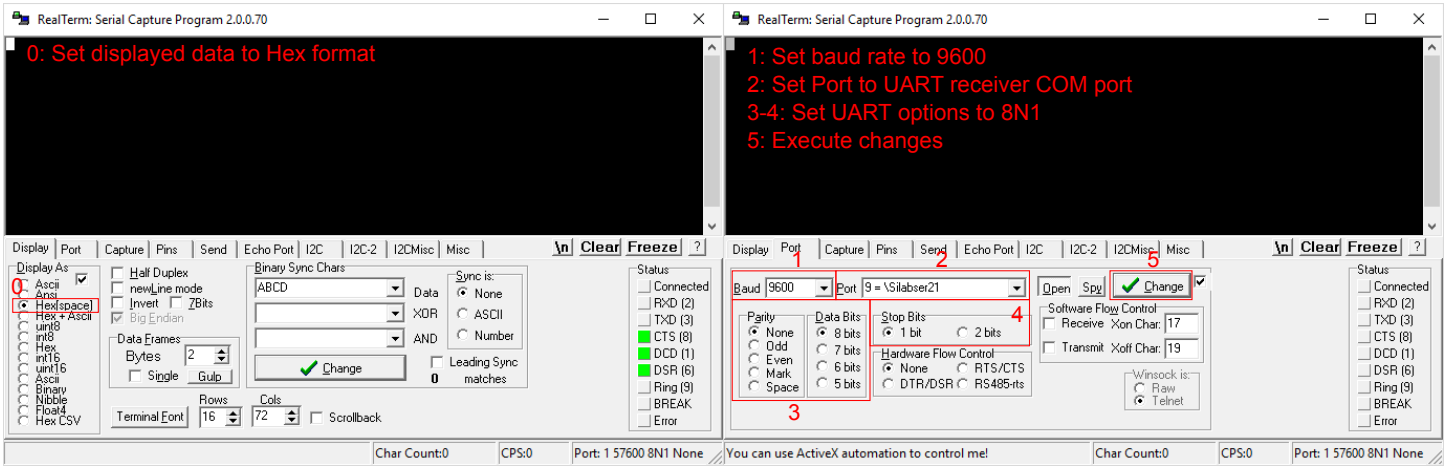


Figure 2.2. RealTerm Options for UART Communications

A sequence of UART commands and the corresponding responses from the Si34071 is shown in the figure below. For the meaning of the individual bits in the response, please refer to the Si34071 Data Sheet. The RealTerm terminal shows invalid data reception because the UART receive had not been disabled before sending out the turn-on MPS command (0x04).

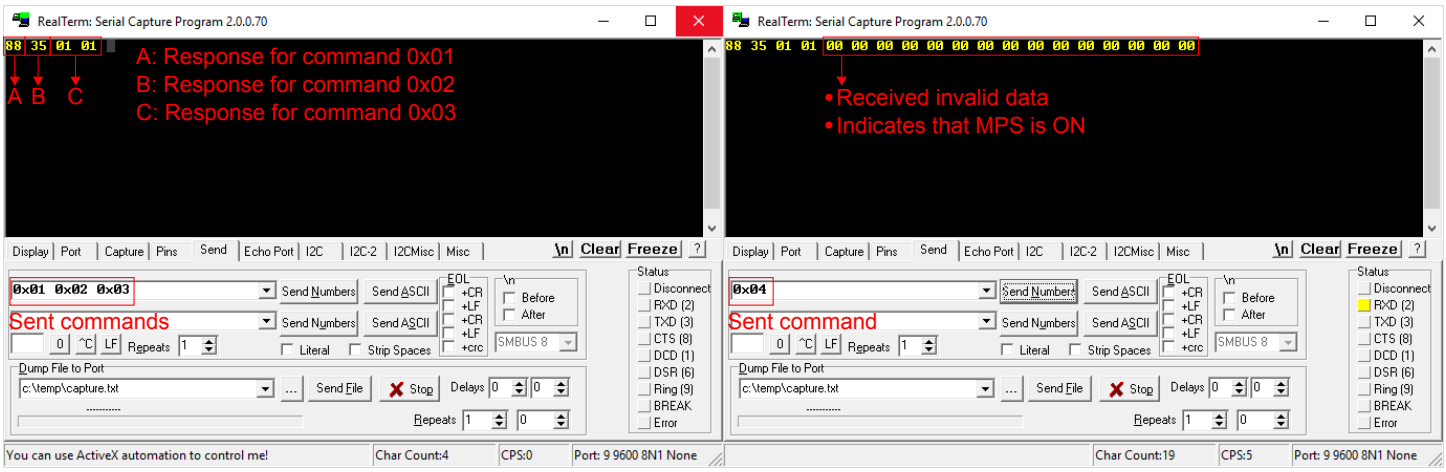
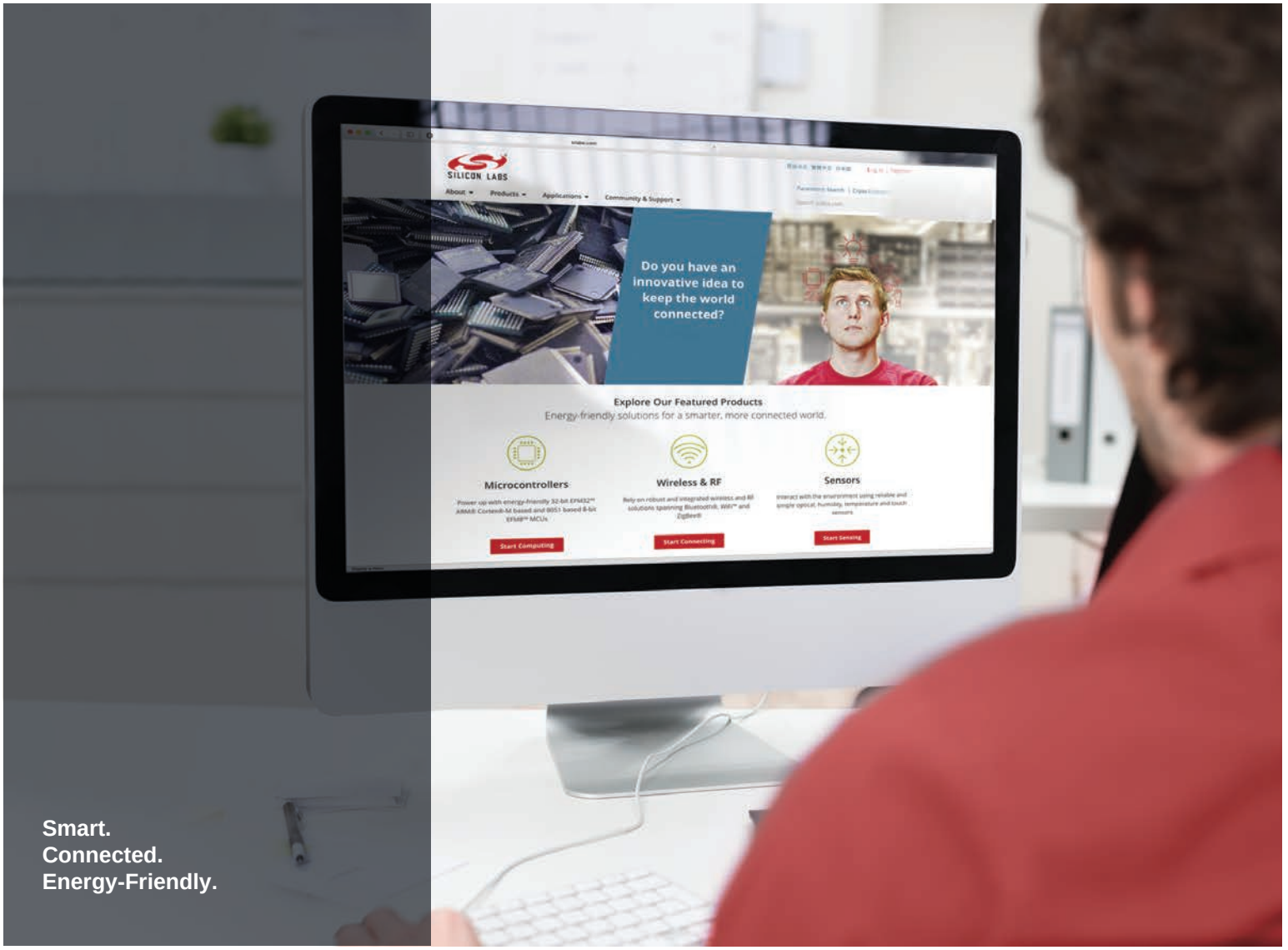


Figure 2.3. UART Responses of Si34071 in RealTerm



Products
www.silabs.com/products

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com