Application Note

Current Loop Sensor using GreenPAK

AN-CM-336

Abstract

This application note describes the design procedure for interfacing current loop sensors with GreenPAK. A GreenPAK SLG88103 IC is used as a front-end operational amplifier to convert the current loop signal to analog voltages from 0 – 1V. Next, a GreenPAK SLG46620 is used to convert this analog signal to the RS232 protocol, which can easily be read by any third-party device. The design includes a dedicated pin which outputs a LOW signal whenever the sensor is disconnected from the design to ensure closed loop operation.

This application note comes complete with design files which can be found in the References section.

Contents

Figures

1 Terms and Definitions

PGA	Programmable Gain Amplifier
ADC	Analog to Digital Converter
DFF	D Flip Flop
LUT	Look Up Table
SPI	Serial to Parallel Interface
CNT/DLY	Counter/Delay
ACMP	Analog Comparator
OPAMP	Operational Amplifier
SPS	Samples per Second
OSC	Oscillator
LSB	Least Significant Bit
MSB	Most Significant Bit
P DLY	Programmable Delay

2 References

For related documents and software, please visit:

https://www.dialog-semiconductor.com/configurable-mixed-signal.

Download our free GreenPAK™ Designer software [1] to open the .gp files [2] and view the proposed circuit design. Use the GreenPAK development tools [3] to freeze the design into your own customized IC in a matter of minutes. Find out more in complete library of application notes [4] featuring design examples as well as explanations of features and blocks within the GreenPAK IC.

[1] GreenPAK Designer Software, Software Download and User Guide
[2] AN-CM-336 Current Loop Sensor using GreenPAK.gp, GreenPAK Design File
[3] GreenPAK Development Tools, GreenPAK Development Tools Webpage
[4] GreenPAK Application Notes, GreenPAK Application Notes Webpage
[5] SLG46620V, Datasheet
[6] SLG88103, Datasheet
[7] Arduino IDE Software Download
[8] Arduino Software Serial Parity Library

3 Introduction

Current loop output sensors are used to transmit data over long distances without dropping the sensor output value. Signal conditioning is performed at the receiving end to analyze and measure the output value. Generally, the output signal is in the range of 4 – 20mA before being conditioned and processed.

This application note describes how to use GreenPAK for signal conditioning of current loop output sensors. To confirm the sensors are working properly, the GreenPAK design outputs a digital LOW signal when the sensor is disconnected from the system.

This GreenPAK design has an input range of 4 – 20mA and outputs the serial data over the UART Tx line at a 19200 baud rate. The signal can then easily be read on any third-party device on the UART Rx pin.

The GreenPAK SLG88103 IC is used as a front-end operational amplifier to convert the current loop signal to analog voltages from 0 – 1V. Next, a GreenPAK SLG46620 is used to convert this analog signal to the RS232 protocol. The SLG46620V is a low-power, cost-effective, small device that can substitute for a system of discrete ICs and passive devices. This unique blend makes the SLG46620 an ideal candidate for portable, cost-sensitive consumer products.

4 Design Overview

Current loop sensors are usually powered through a 12 – 36V_{DC} power source. In this design, the sensor is supplied with a constant DC voltage source in series with a 50Ω resistor. The output current ranges from 4 – 20mA.

We know that by Ohm's Law, V=IR. At 4mA of current through the 50Ω resistor, the voltage is 0.2V. This is the minimum voltage of the sensor output. At 20mA of current through the 50Ω resistor, the voltage is 1V. This is the maximum voltage of the sensor output.

Whenever the sensor is disconnected from the system, the current through the resistor drops to 0mA and the voltage drops to 0V. We can sense this dropped voltage to detect if the sensor is connected to the system or not.

After conversion, the output from the op amp is fed directly to the GreenPAK SLG46620V IC at PIN8. Using PGA, ADC, DFFs, and LUTs, the signal through PIN8 is converted to the RS232 protocol, at a configuration of 8 data bits, with 1 even Parity bit at 19200bps. The IC then outputs the data serially on PIN12. To check if the sensor is connected to the system or not, the IC outputs a digital LOW signal at PIN13 whenever the sensor is disconnected from the system.

To make the circuit power efficient, an enable pin (PIN9) disables all the SLG46620V's macrocells when pulled HIGH, reducing the total power consumption of the IC to less than 10µA. This feature not only decreases the overall current consumption of the system, but also gives control to the third-party device to decide when to process data.

The internal working of the SLG46620V's macrocells is discussed in detail in the GreenPAK Design section.

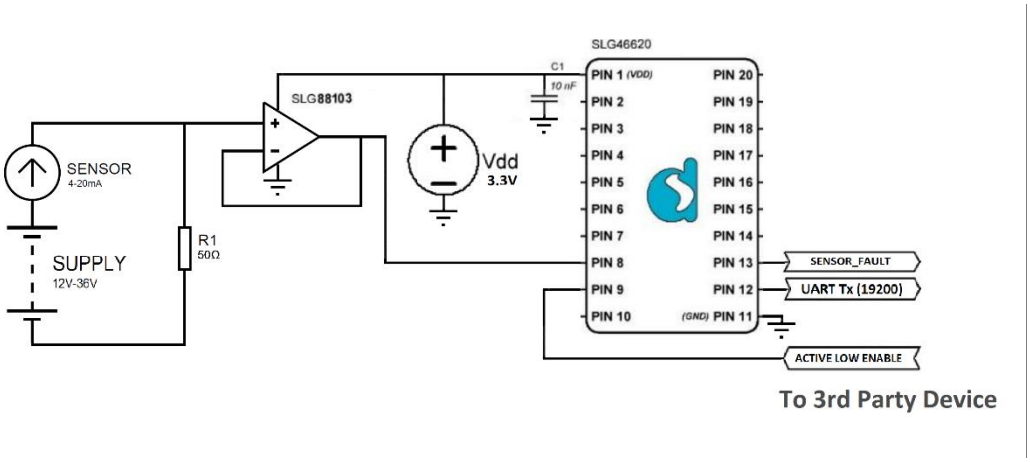

Figure 1: System Schematic

5 GreenPAK Design

The design consists of three parts:

1. Analog to parallel data conversion
2. Parallel to serial data conversion
3. Adding Start, Parity and Stop bits

5.1 Analog to Parallel Data Conversion

The IC receives the external analog signal through PIN8, which is configured to operate as an analog input/output. The signal then passes through a Programmable Gain Amplifier, or PGA, which sets the gain. Then, the signal enters the analog-to-digital converter (ADC) block. The operation mode of the ADC is single ended with x1 gain.

The ADC macrocell converts the analog signal in 8 bits of corresponding digital data. This 8-bit digital data from the ADC is fed directly to the SPI macrocell. The configuration of the PGA, ADC, and SPI macrocells can be seen in Figure 2. The SPI macrocell allows parallel data output. By enabling the block in Matrix 1, we can have the data on 8 parallel bits.

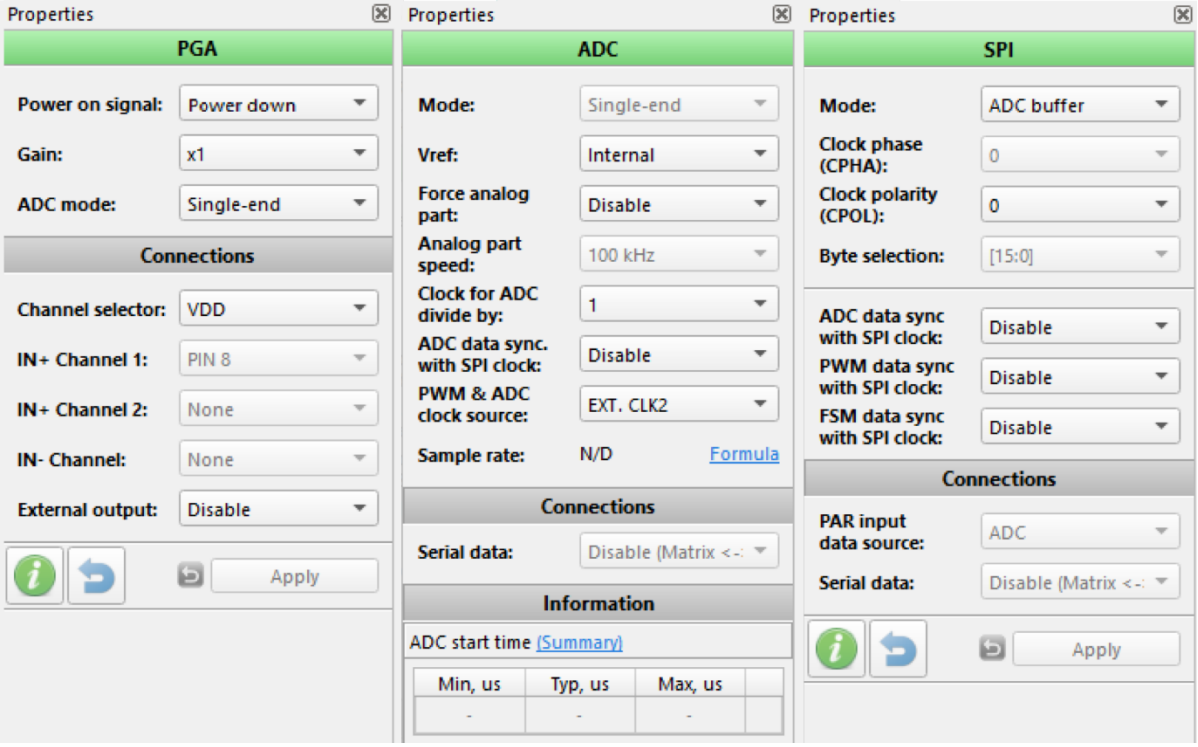

Figure 2: PGA, ADC, and SPI Configuration

The PWM and ADC clock source is defined as EXT. CLK2, which is the frequency at which our data will be outputted serially from the GreenPAK (19200 bits per second).

The configuration of the OSC macrocell is shown in Figure 3.

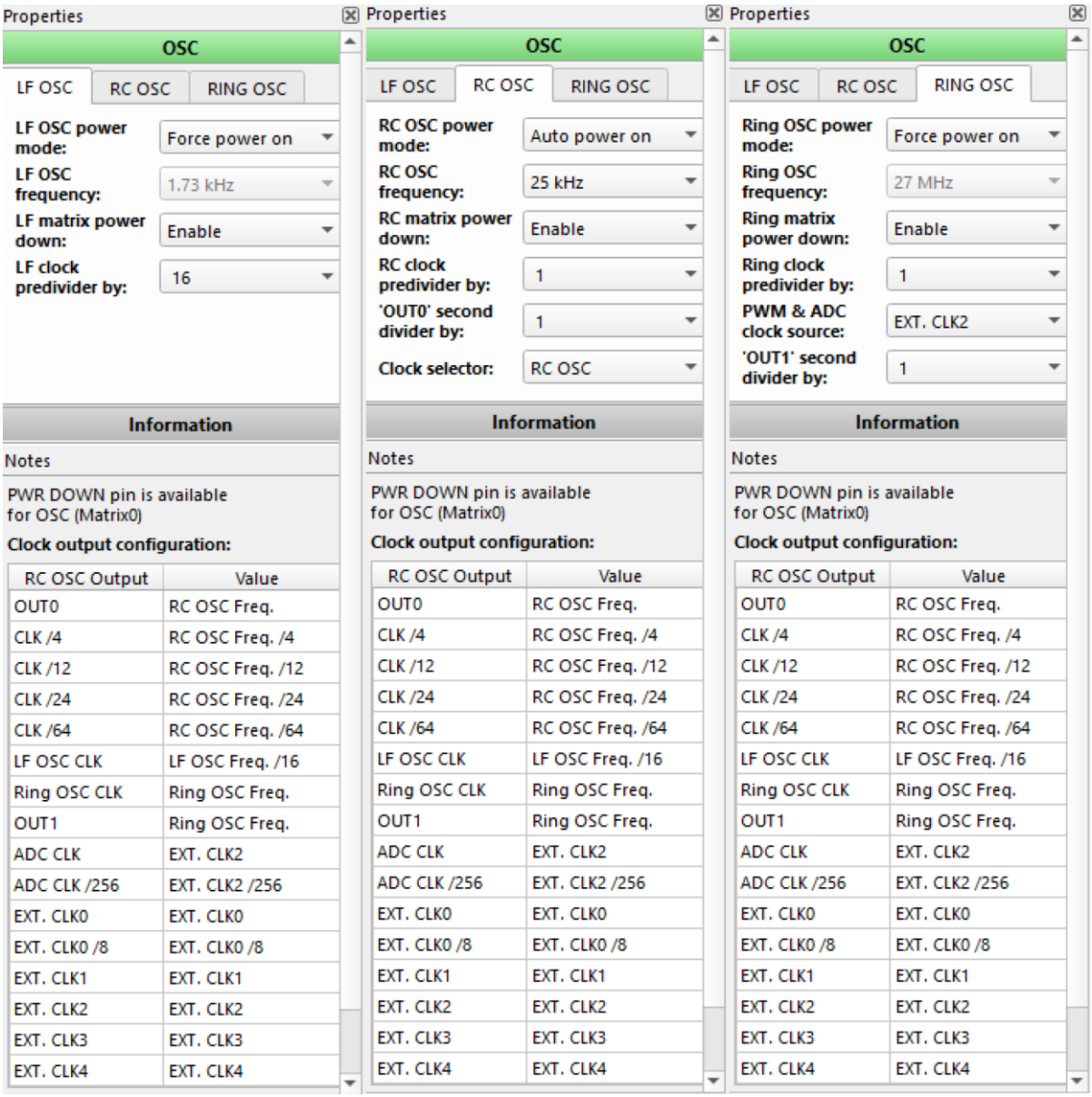

Figure 3: OSC Configuration

We have generated a clock frequency of 19.2kHz using the 14-bit CNT2/DLY2 and DFF0. This 19.2kHz frequency signal is not only used as the clock source of the ADC, but also to convert parallel data from the parallel data output block to serial data. This is explained further in the next section.

SPI Parallel Output configuration (in Matrix1) is shown in Figure 4.

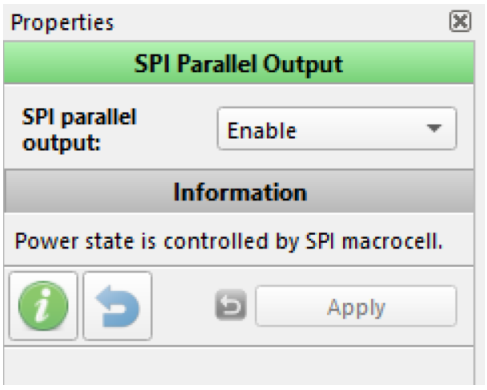

Figure 4: SPI Parallel Output Configuration

In the below figure, some wires from Matrix0 are connected to those in Matrix1. GreenPAK Designer has ports to carry signals between matrices. Out-ports from one matrix are available in the other matrix as In-ports, and vice versa.

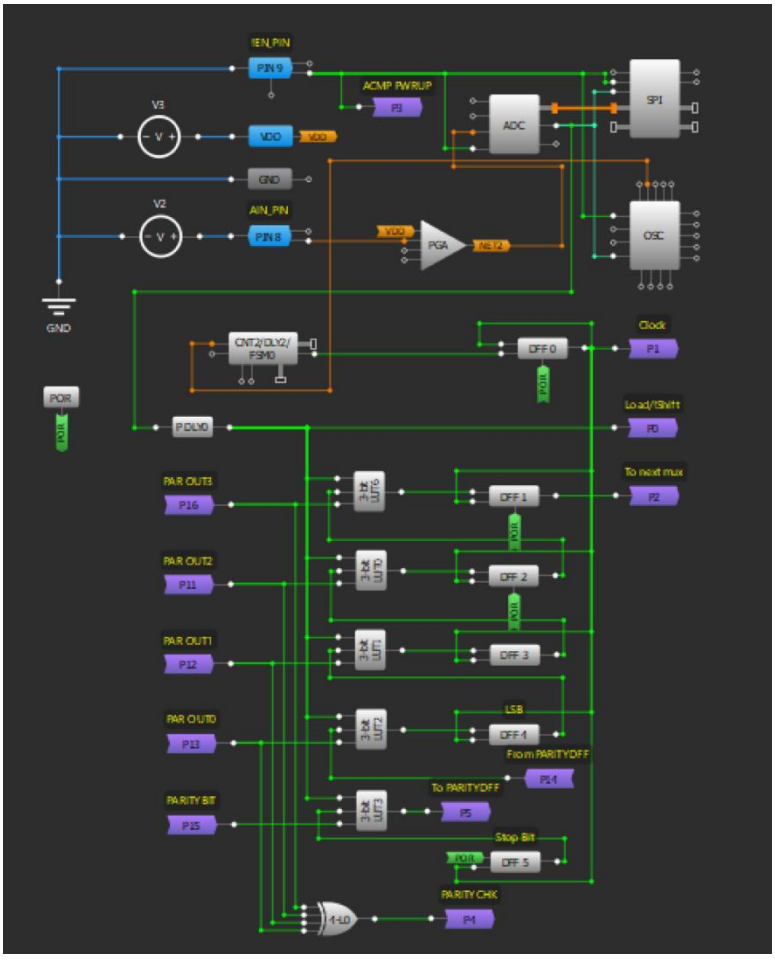

Figure 5: Matrix0

5.2 Parallel to Serial Conversion

The data from the Parallel Data Output macrocell in Matrix 1 is converted to serial data using a Parallel-In-Serial-Out (PISO) shift register. DFF1, DFF2, DFF3, DFF4 in Matrix0 and DFF6, DFF8, DFF9, DFF10 in Matrix1 are used as 8-bit PISO shift registers to store and transmit data from SPI parallel output serially to PIN12. DFF7, DFF11 and DFF5 are used to add Start, Parity and Stop bits respectively to the serial data. We will discuss these in the next section. Figure 6 shows the configuration of the 8-bit PISO DFFs.

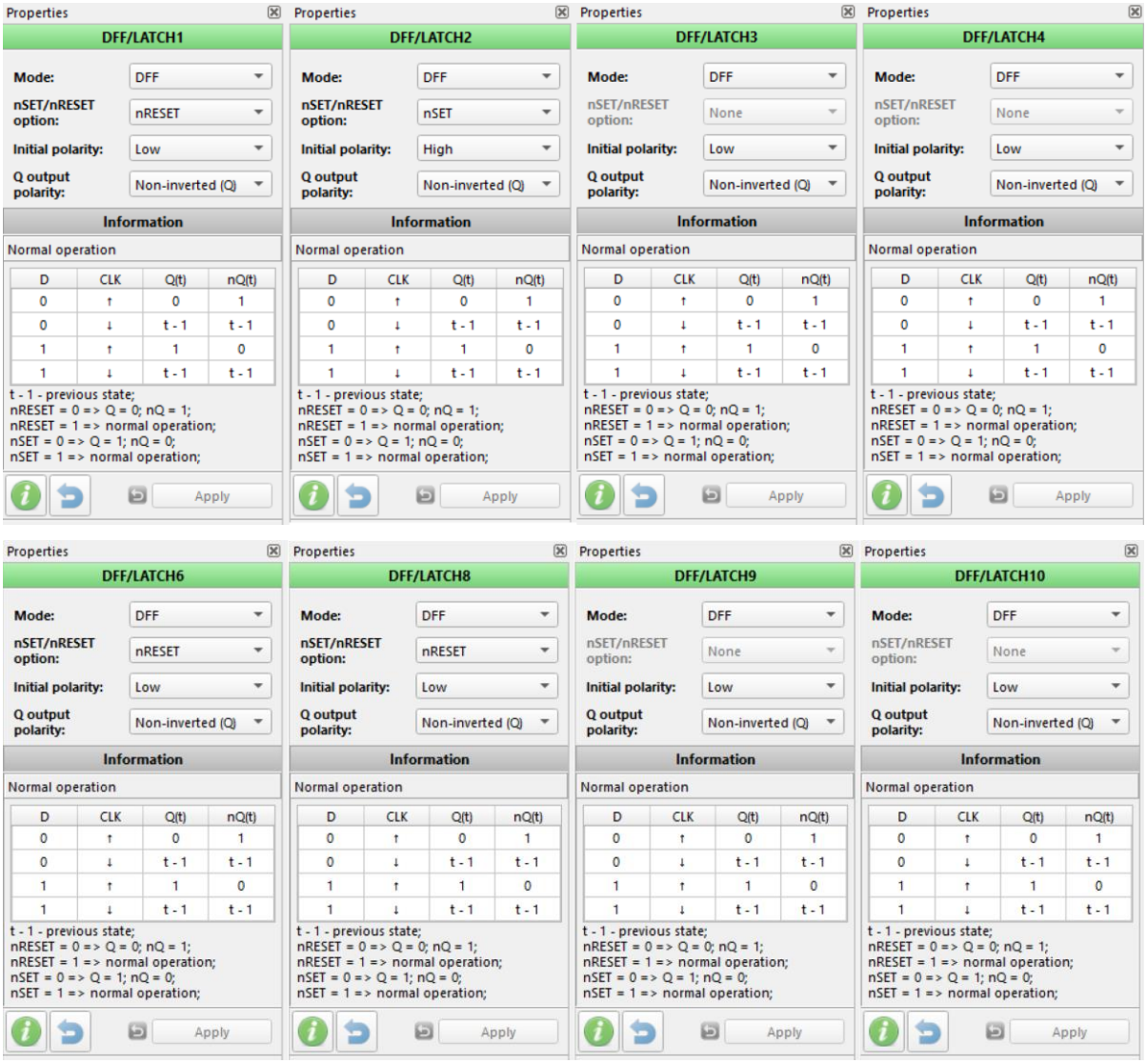

Figure 6: PISO DFF Configuration

The 8 parallel output bits from the SPI parallel output block are connected with the 8-bit PISO DFFs at the D pin of each DFF. A clock is applied at the CLK pin of the DFFs, which controls the shifting of the bits in the DFFs. Every time a rising edge is inputted at the CLK pin of a DFF, the value (0 or 1) stored in that DFF is shifted to the next DFF. In this way, after 8 consecutive high pulses at the CLK pins, the 8-bit data is outputted from the PISO shift register.

The clock signal at the input of the DFFs is generated using the 14-bit CNT2/DLY2 and DFF0. The frequency of this clock signal is 19200Hz. At this frequency, the GreenPAK communicates at a common baud rate of the RS232 protocol. The most common baud rates are 4800, 9600, 19200,

38400, 57600, and 115200. By selecting the 19200Hz frequency, the data transmits at 19200bps serially through the PISO shift register. The configuration of the 14-bit CNT2/DLY2 and DFF0 are shown in Figure 7.

Figure 7: 19200Hz Frequency Generator

After every successful transmission of the 8-bit data, the next step is to load the next 8 bits of data into the shift register. To do this, we use 3-bit LUTs to form a multiplexer (MUX) design within the SIPO. LUT0, LUT1, LUT2, LUT3, and LUT6 in Matrix0, and LUT8, LUT9, LUT10, LUT11, LUT12, and LUT15 in Matrix1 are used as MUXs.

All these LUTs have exactly the same configuration, except LUT12. Figure 8 shows the configuration of the LUTs.

Figure 8: LUT Configuration

To load the data within the DFFs, we'll first look at the timing diagram of the ADC and SPI parallel outputs with respect to the 19200Hz clock frequency as shown in Figure 9.

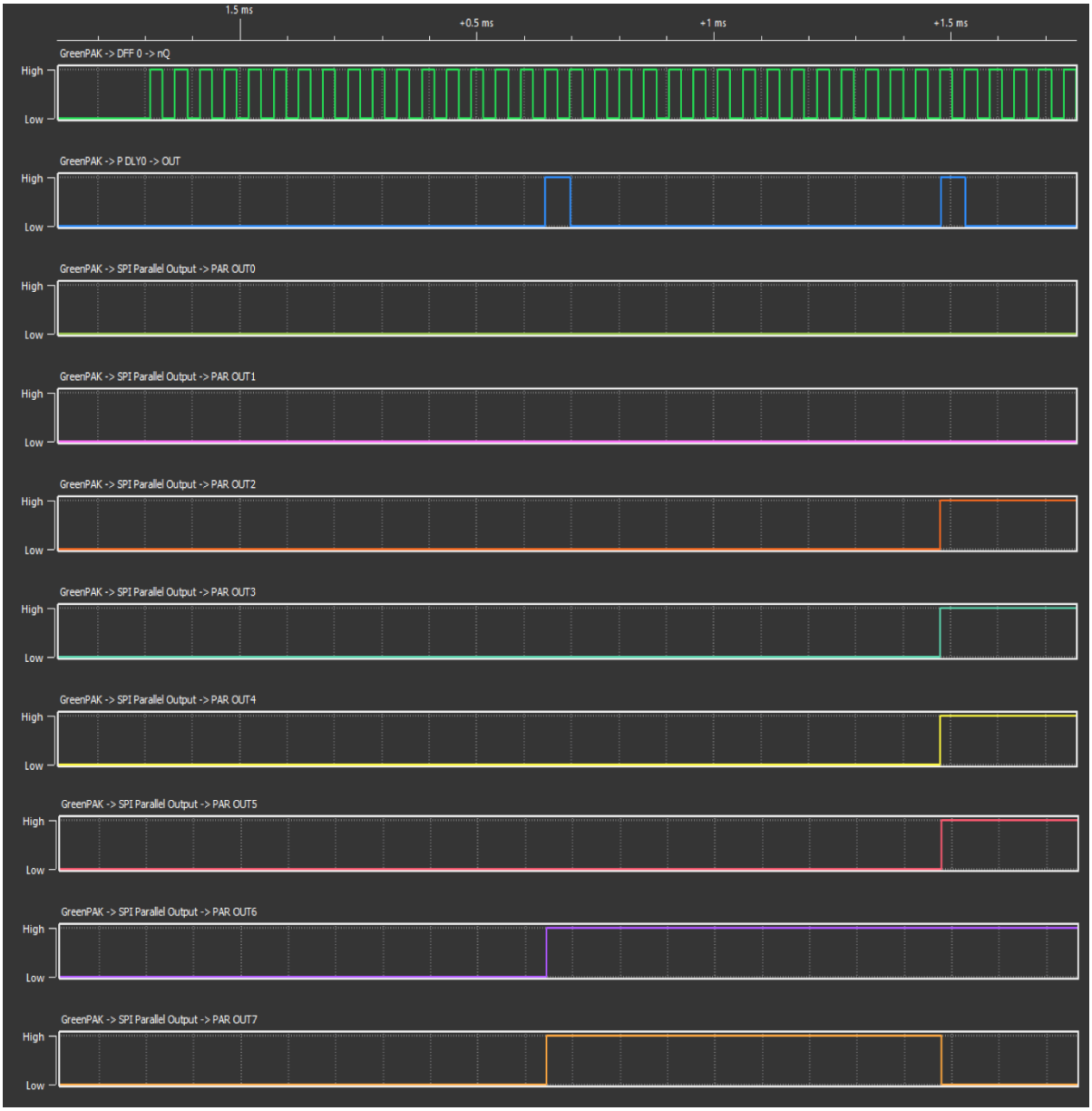

Figure 9: 8-bit Parallel Data Timing Diagram

The first waveform in green is the 19200Hz clock signal. The ADC is also supplied with this clock frequency through EXT. CLK2 of the OSC block. The next waveform is the ADC_INT signal of the ADC macrocell. According to GreenPAK Designer, “The Interrupt output is one clock period long and signifies the PAR data is valid.” See the timing diagram of the ADC output in Figure 10.

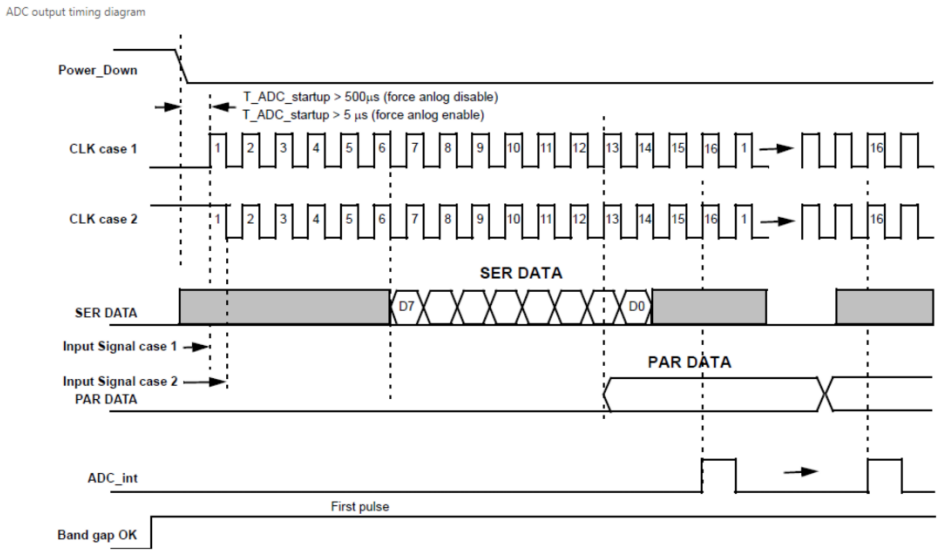

Figure 10: ADC Output Timing Diagram

Using the INT signal from the ADC_INT pin, we will be notified that the data at the parallel bits of the parallel data output is ready to be loaded in the DFFs.

Have a look at the next 8 waveforms carefully. Notice that the states of the outputs are changing with every rising edge of the INT signal. The first signal is PAR OUT0, and it is the least significant bit. The last signal is PAR OUT7, and it is the most significant bit.

By using the multiplexers, we can check for the INT rising edge and load the data into the 8-bit PISO DFFs. The time duration between two consecutive INT pulses is the time we will use to transmit the data serially from the GreenPAK.

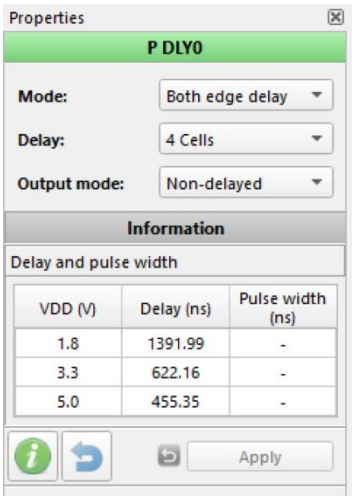

Figure 9: Programmable Delay

A Programmable Delay (P DLY) logic cell is used to generate a small delay in the signal. We have used this to generate an approximately 500ns delay at the interrupt signal of the ADC to make sure that the data at the parallel output block is available.

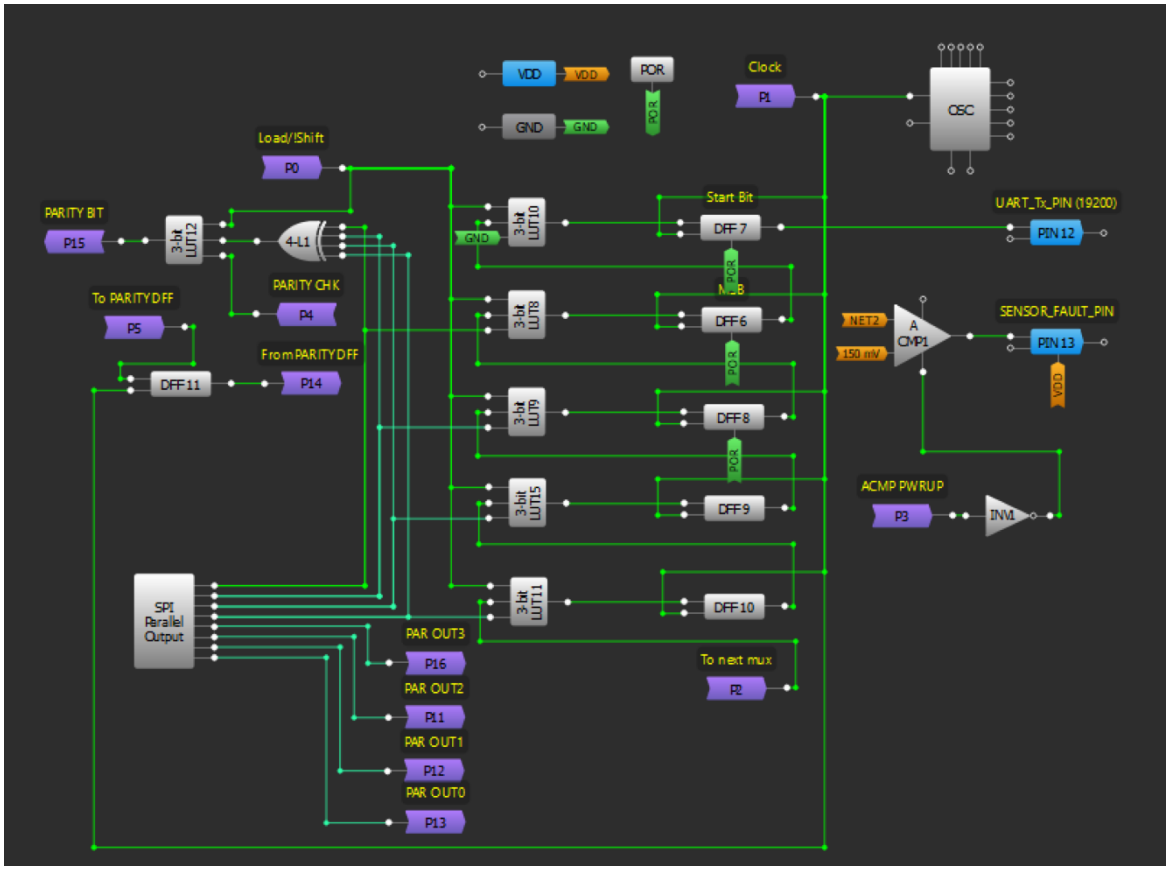

Figure 10: Matrix1

5.3 Adding Start, Parity, and Stop Bits

The RS232 protocol communicates 8 data bits in a single packet, with 1 Start bit, 1 Parity bit, and 1 or more Stop bits. After the conversion of our data from an analog signal to parallel data and then to serial data, the last step is to add these bits so that our data becomes standard RS232 data.

DFF7 is used to load the Start bit. The Start bit is one clock pulse long, a digital LOW signal which indicates the start of the data. The next 8 bits after the Start bit are the 8 bits of data.

After the transmission of the Start and 8 data bits, the next bit is a Parity bit. This is either HIGH or LOW depending on the number of 1s in the 8 bits of data. We have a LOW Parity bit if the number of 1s is even and a HIGH Parity bit if the number of 1s is odd. For example, if we have our 8 data bits as 01101000, we have an odd number of 1s and the Parity bit must be HIGH.

To generate a Parity bit, we have used the 4-bit LUT0 in Matrix0, and the 4-bit LUT1 in Matrix1 as XOR gates. An XOR gate has an output HIGH if the number of 1s on the input is odd, and an output LOW if the number of 1s is even. We have inputted these LUTs with the 8 parallel output bits from the SPI parallel output macrocell. At the rising edge of the interrupt signal from the ADC, we check the number of 1s and load the parity bit into DFF11.

A Stop bit is one or more HIGH bits sent after successfully transmitting 1 Start bit, 8 data bits, and 1 Parity bit serially. DFF5 is used to load the Stop bit.

Figure 13 shows the configuration of DFF7, DFF11, and DFF5 for the Start, Parity and Stop bits respectively.

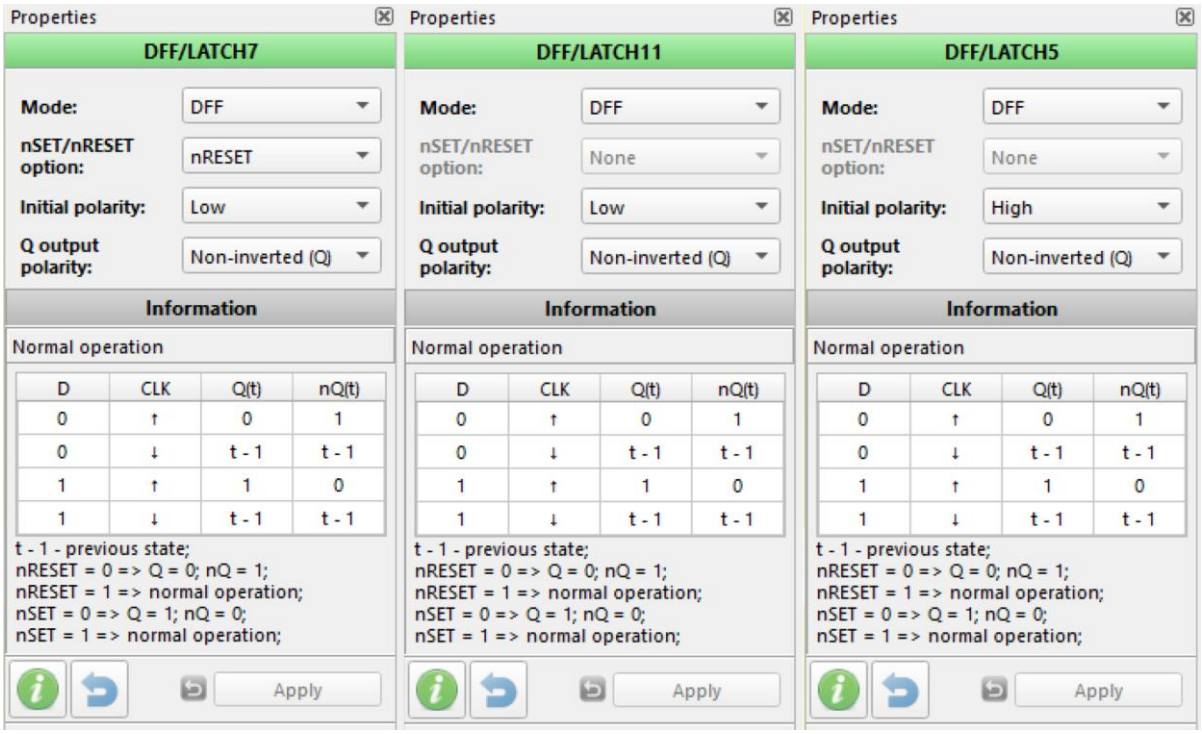

Figure 11: Start, Parity, and Stop bit Configuration

5.4 Sensor Fault Detection

Another interesting feature we can generate using GreenPAK is sensor fault indication. We have used an analog comparator for this purpose, which outputs a digital LOW signal at PIN13 of the GreenPAK SLG46620V whenever the voltage at the input drops below 150mV.

The current loop sensors have a minimum output value of 4mA and a maximum of 20mA. But whenever the sensor is disconnected from the loop, the current drops to 0mA. This results in a signal of 0mV at the AIN_PIN (PIN8 of SLG46620). This signal is fed directly to the third-party device (e.g., Arduino).

ACMP1 in Matrix1 is used as the sensor fault detector. IN+ is set to 150mV, while IN- is supplied with a NET connection from the PGA macrocell.

To enable this feature, we apply the active low enable signal into the inverter INV1 in Matrix1 and connect the output of INV1 to the PWR UP pin of the ACMP1. Figure 14 shows the configuration of ACMP1.

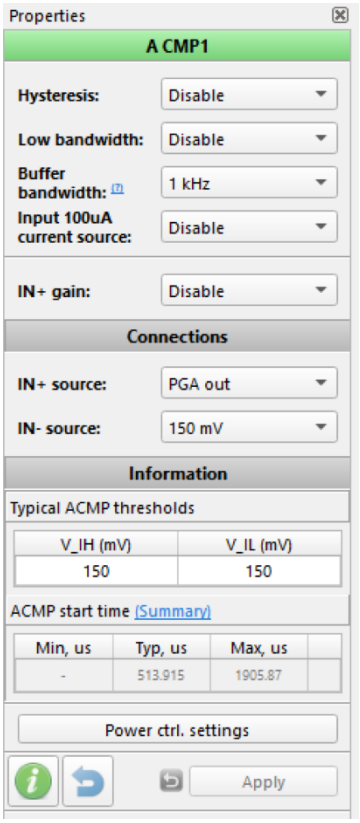

Figure 12: ACMP1 Configuration

6 Results

The Signal Wizard included in GreenPAK Designer lets us examine the design and ensure it works as expected.

Signal Wizard is very convenient for design inspection, where signals of different shapes can be generated without the need to use an external signal generator. The signal frequency and amplitude can be easily controlled. A custom signal can also be generated.

The test results are shown in Figure 15. The second waveform is the input signal applied to the AIN_PIN. Consider the signal at 2.2ms in the signal wizard. It is 300mV. At this position, with the rising edge of the clock signal (last waveform), the UART data is available at PIN12 (third waveform).

Every UART bit starts at the rising edge of the clock signal. The first bit is a LOW signal (0). This indicates the Start bit. The next 8 bits are decoded as 01000101. These are the 8 data bits which carry the actual data.

The bit after the 8 data bits is the Parity bit, which shows the number of 1s in the data. We have a HIGH signal (1) as the Parity bit because the number of 1s are odd in this case.

After the Parity bit, the next 1 or more bits are HIGH Stop bits.

The next data packet starts at the falling edge of UART.

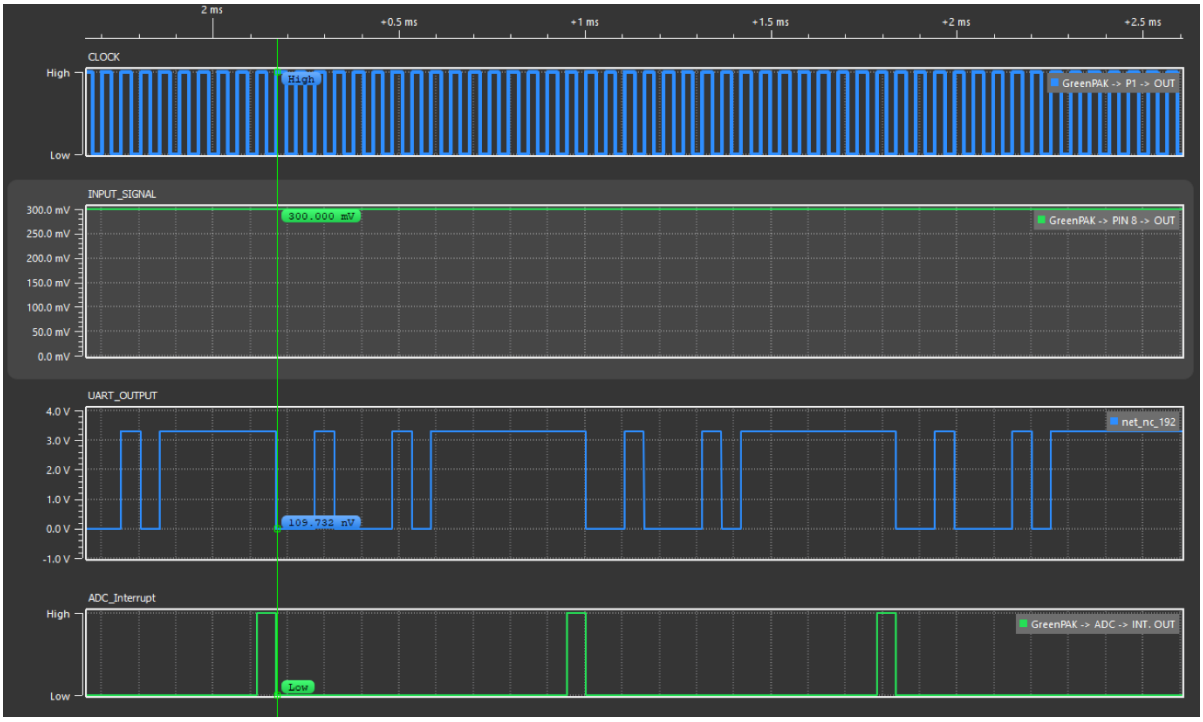

Figure 13: Results

Consider the next UART falling edge at approximately 2.2ms + 0.8ms. The first bit (0) is again a Start bit, followed by the same data bits 01000101, and a Parity bit of 1 since the number of 1s is odd.

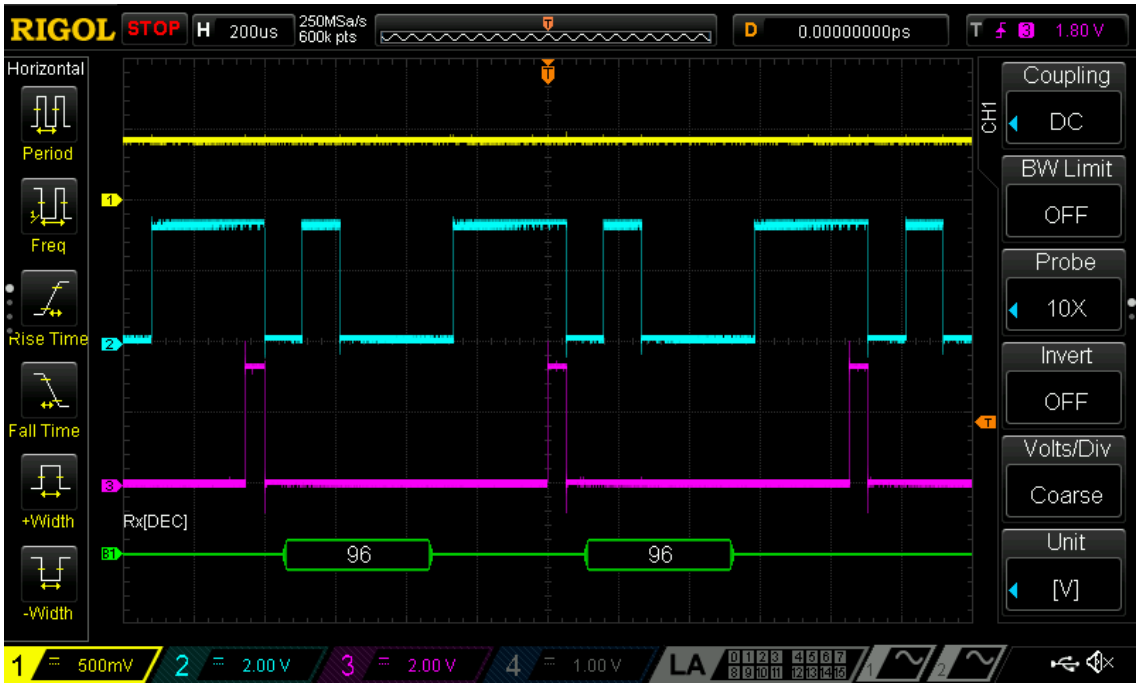

Figure 14: Waveform on Oscilloscope

7 Arduino Code

Following is the bare minimum code for reading values from GreenPAK on an Arduino development board.

```
#include <SoftwareSerialParity.h>

SoftwareSerialParity GreenPAK(2, 3); // RX, TX

int i;

void setup()

{

	GreenPAK.begin(19200,EVEN);

	Serial.begin(19200); // start serial to PC

}

void loop()

{

	if (GreenPAK.available() > 0)

	{

	i = GreenPAK.read();

	Serial.print(i, DEC);

	Serial.print(" ");

	}

}
```

The “Software Serial Parity” library is used to read data with even parity through Arduino. See the References section for more information.

8 Conclusion

This Application Note outlines how to build a signal conditioning circuit using a GreenPAK, which includes a current loop to RS232 converter circuit at a 19200 baud rate.

The GreenPAK IC demonstrates a high efficiency for integrating several functions in a low-cost and small-area IC solution, making it especially suitable for industrial devices.

Revision History

Revision	Date	Description
1.0	09-May-2022	Initial Version

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.