AN-1176 RFID Sensor Dog Door

This application note describes how to use a GreenPAK SLG46531V as the main controller for an RFID-based pet door.

We used an SM130 RFID module shown in Figure 1 to interface with the GreenPAK, and used the opportunity to apply serial communication principles.

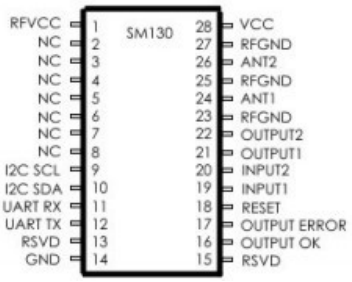

Figure 1. The SM130 RFID module

PINs 16 and 17 play a major role in this project, as these two pins let us know when the SM130 is looking for a tag, and when it has found a new one. PIN 17, also known as OUTPUT ERROR, goes to HIGH when communication with a tag is not successful. PIN 16, labelled OUTPUT OK, goes to HIGH when the communication is successful.

This means that OUTPUT ERROR can signal when the SM130 is being ordered to "Seek a Tag", and OUTPUT OK will signal with a HIGH when a tag has been found. These two signals are going to be the base of the process for controlling the electronic lock. But before we jump into that, we have to send the SM130 a "Seek Tag" from the GreenPAK.

The SM130 has the UART protocol enabled by default, so we will be using that for this project. In Figure 2 we can see the UART frames as shown in the SM130 datasheet.

Following is the UART frame for the commands sent by the host:

Header	Reserved	Length	Command	Data	CSUM
1 Byte	1 Byte	1 Byte	1 Byte	N Bytes	1 Byte

Figure 2. UART frames from SM130 datasheet

System function and circuit design

The SM130´s OUTPUT ERROR turns HIGH after the "Seek a Tag" command is received. OUTPUT OK turns HIGH when a tag is detected by the antenna, but this HIGH only lasts for a split second, so we have to plan accordingly.

The OUTPUT OK signal will be received by the GreenPAK as a trigger on PIN 17. Then, we'll use a delay to keep the signal HIGH for 5 seconds, during which time the lock will remain open.

The lock used for this application is at 12 V DC with a current of 0.6 A, but it can be used with 8 V as well. We have two choices of circuit to drive the lock, both of which will be explored in this app note.

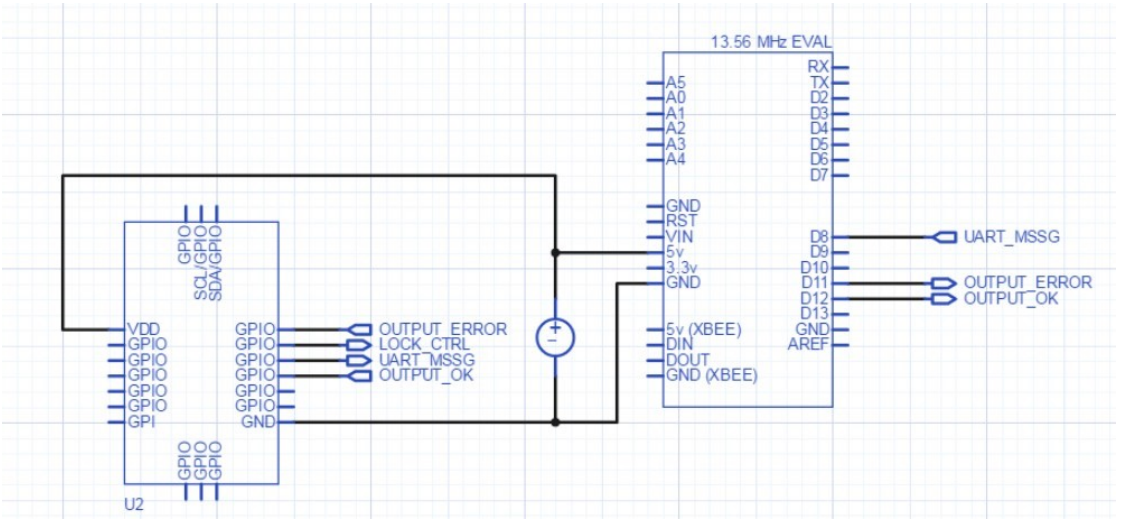

Figure 3. Schematic with SparkFun DEV-10406 RFID Evaluation Shield

To speed up development, you can buy the SparkFun DEV-10406 Spark fun RFID Evaluation Shield. You can simply connect the UART Rx (Pin D8) to any of the GPIO pins on the SLG46531V.

Implementing UART Communication

In order to output our UART frames, we set up the GreenPAK to output all of the bits stored in its Asychronous State Machine's RAM. We used the same technique discussed in AN-1137: Serial Output Tips and Techniques; please consult that app note for more information on how to assemble a serial output ASM.

At the hardware level, when the system is idling we need the data line to be HIGH, so we surrounded the frames with HIGH bits.

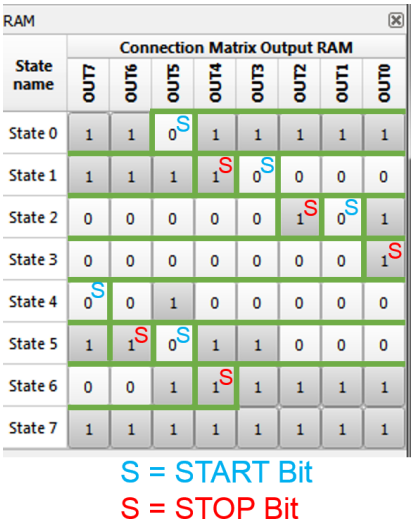

Figure 4. Configuration of the ASM's RAM table

The ASM is configured as shown in Figure 4. The Start and Stop bits are labeled with a blue 'S' and a red 'S'.

- 0xFF: Header Byte starts in State 0 and finishes in State 1
- 0x00: Reserved Byte starts in State 1 and finishes in State 2
- 0x01: Length Byte starts in State 2 and finishes in State 3
- 0x82: Command Byte starts in State 4 and finishes in State 5
- 0x83: Checksum starts in State 5 and finishes in State 6

This configuration will give us a waveform that, when analyzed by the receiver end, will output the command "Seek a Tag." Figure 5 shows the waveform analyzed by Saleae Logic Analyzer.

Figure 5. The "Seek a Tag" waveform in Saleae Logic Analyzer

The introduction of the Start and Stop bits make it a little more difficult to intuitively implement the code, so it's important to have a way to look at the waveform.

Next, we will be using the horizontal ASM serial output as shown in Figure 6.

As you can see in Figure 6, most of the resources of the GreenPAK are being used by the horizontal ASM serial output design, so we'll have to make the lock operation as simple as possible.

Before we do that, we need to figure out what clock we are going to use. The SM130 has a default baud rate of 19200 bps. We could generate that CLK speed inside the GreenPAK, but since we are low on resources it makes more sense to generate the CLK outside the GreenPAK.

Another important point is that the UART idle state is HIGH, so we need to send a continuous HIGH state after the SM130 has already started looking for a tag.

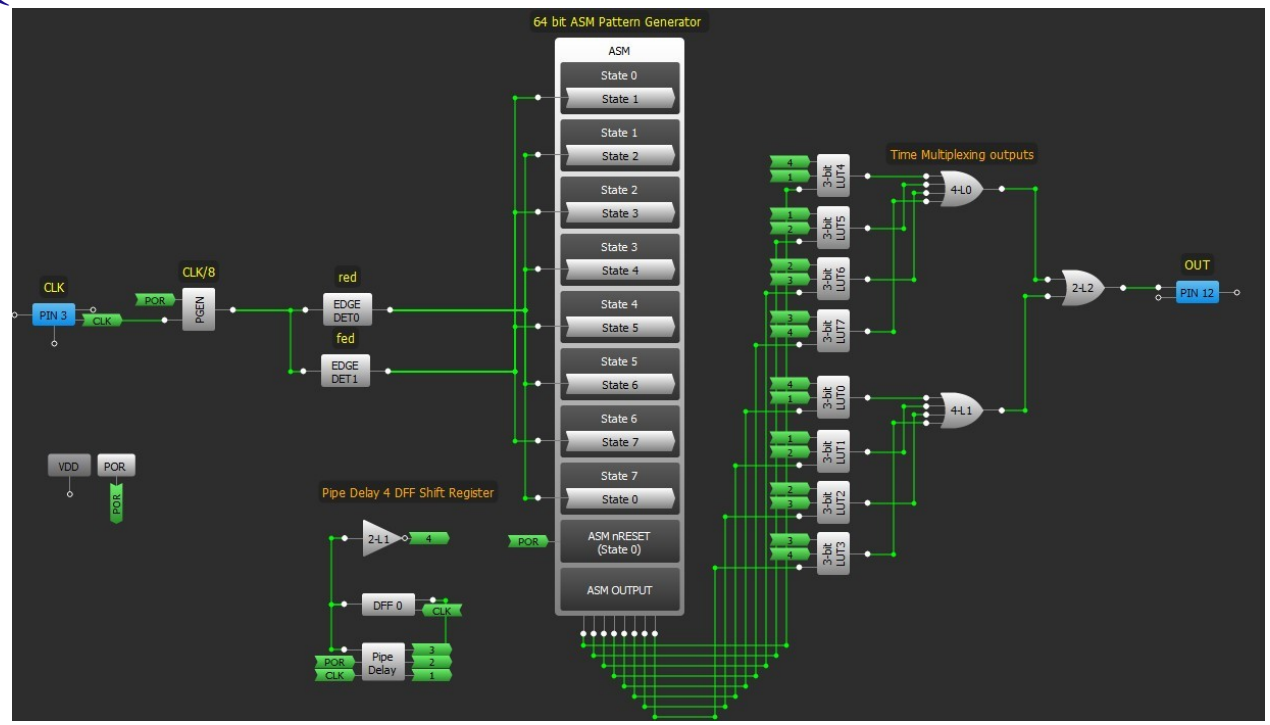

Figure 6. Using the ASM's serial output

Developing LOCK Control Using 2 LUTs

After the ASM is finished we only have two LUTs left to work with, both of which are 3-bit LUTs.

First, we have to find a way to extend the duration of the OUTPUT OK signal from a few fractions of a second to at least 5 seconds. Luckily, the GreenPAK has no problem dealing with this situation.

We'll reconfigure 3-bit LUT8 as a delay block. Using the falling edge of the OUTPUT OK signal, we'll trigger a falling edge delay of 5 seconds. This means we are going to delay the change from HIGH to LOW for 5 seconds.

We'll use OSC0/64, pre-divided by 8, to create a delay of 5 seconds, successfully using one LUT to drive the lock, as shown in Figure 7.

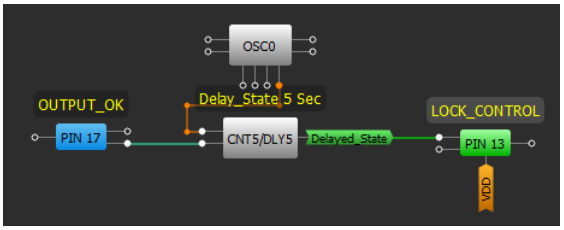

Figure 7. Creating a delay of 5 seconds to control the lock

Finally, to drive the ASM output HIGH, we will use the OUTPUT ERROR signal, which can only be HIGH when OUTPUT OK is LOW.

Here we find a problem: it is possible for the OUTPUT ERROR to be HIGH when the door is open, leading to some undesirable side effects. To prevent this, we will use the Delayed_State signal to drive the last 3-bit LUT.

The states are as follows:

- Delayed_State is HIGH, so the bus is not sending the UART message (it doesn't matter what the other signals are).
- Delayed_State its LOW, but OUTPUT_ERROR is HIGH, so the bus is not sending the UART message.
- Delayed_State is LOW and OUTPUT_ERROR is LOW, so the bus transmits the UART message.

In the LUT table we assign Delayed_State to IN2, OUTPUT_ERROR to IN1, and the ASM output to IN0. This turns the LUT into an OR gate, as the only time that a LOW state is sent through the bus is when the three signals are in a LOW state.

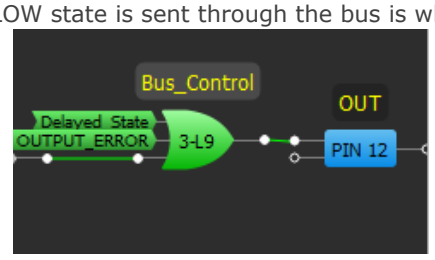

Figure 8. The Bus_Control LUT

Driving the Lock

The lock used for this application note is a 12v DC Cabinet Drawer Electric Solenoid Lock, and is pictured in Figure 9. When powered on, the shaft retracts and stays retracted as long as power is still supplied.

As stated before, this app note covers two ways to drive the system:

- Using the L298n H-Bridge module
- Using transistors

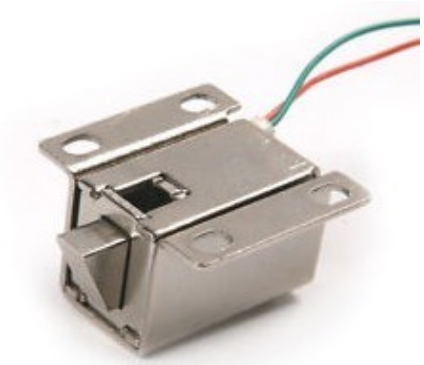

Figure 9. The lock used

It's true that driving the lock with the L298n its easier and more convenient, but if we want to make a custom PCB for this project, it would be a good idea to dive a little into driving the lock using transistors.

The lock is rated at 12 V DC and 0.6 A, which means the lock has a resistance of approximately 20 Ohms. The lock also works at 8 V with a current of 0.4 A. Usually transistors are rated with a maximum of 0.5 A, so knowing that we can drive the lock with less is a good thing.

If we supply the motor with the 12 V it requires, we'll need a resistor to limit the current. That resistor will go in series with the motor. A resistor value of 10 Ohms is enough, but we need to know how much power it's going to be dissipating. Since the resistor is going to be dissipating more than 1 W, we'll choose a value that withstands at least 2 W.

In Figure 10 we can see the transistors switching ON and OFF the lock.

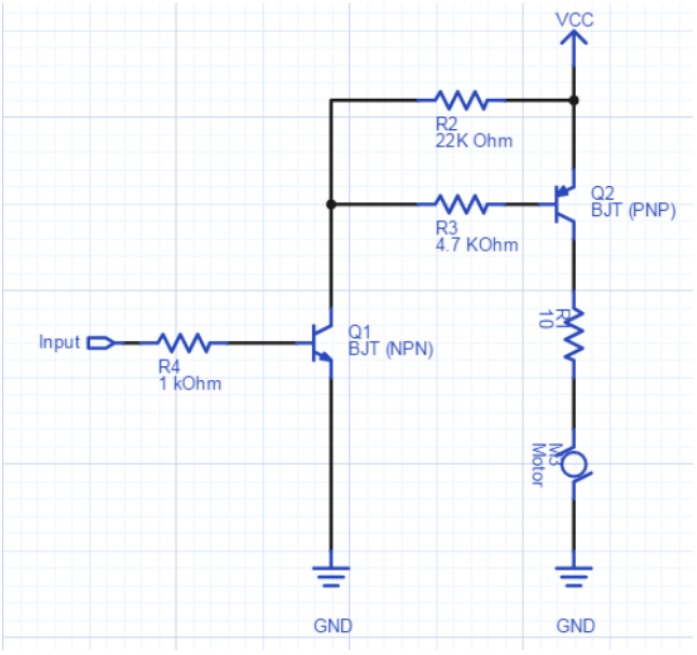

Figure 10. Using transistors to swithc the lock ON and OFF

In Figure 10, Q1 is an NPN BJT. When its base is LOW, Q1 behaves like an open switch, so R2 and R3 are in series and the voltage in Q2's base is similar to the voltage in its emitter, meaning Q2 also behaves like an open circuit.

When Q1's base is HIGH, it transmits some of the voltage between R2 and R3, making Q2's base voltage significantly smaller than the voltage in its emitter, allowing Q2 to turn ON the lock.

Results

Figure 11 shows the completed system. An Arduino Uno is used as an external clock, but the whole system is driven by the GreenPAK. As the command is already sent, the LED that signals searching turns on.

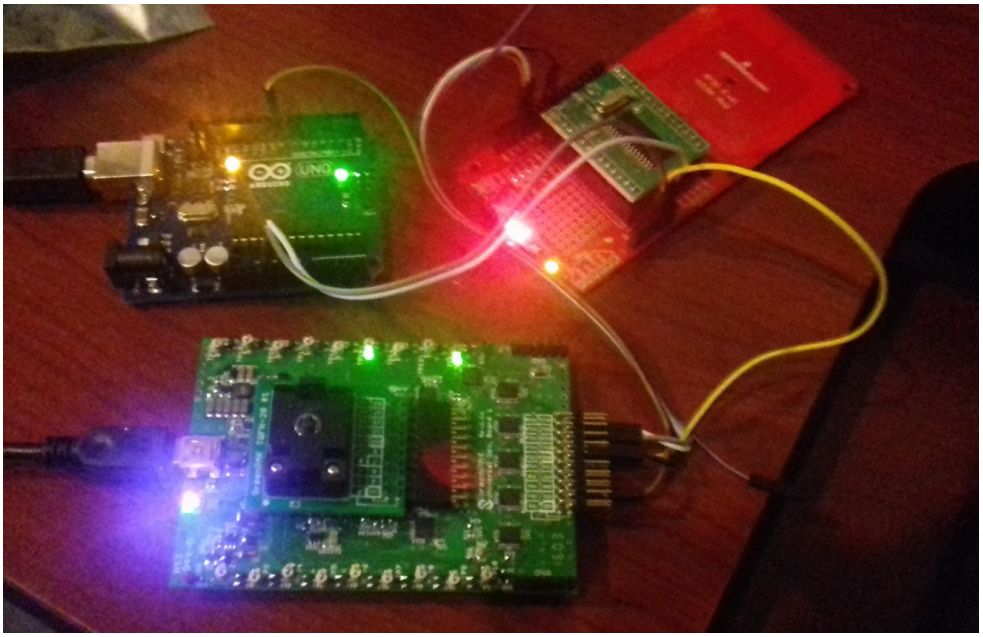

Figure 11. Prototype of system

Figure 12 shows a tag in the antenna prompting the GreenPAK to wait for the programmed time, after which the message is sent again.

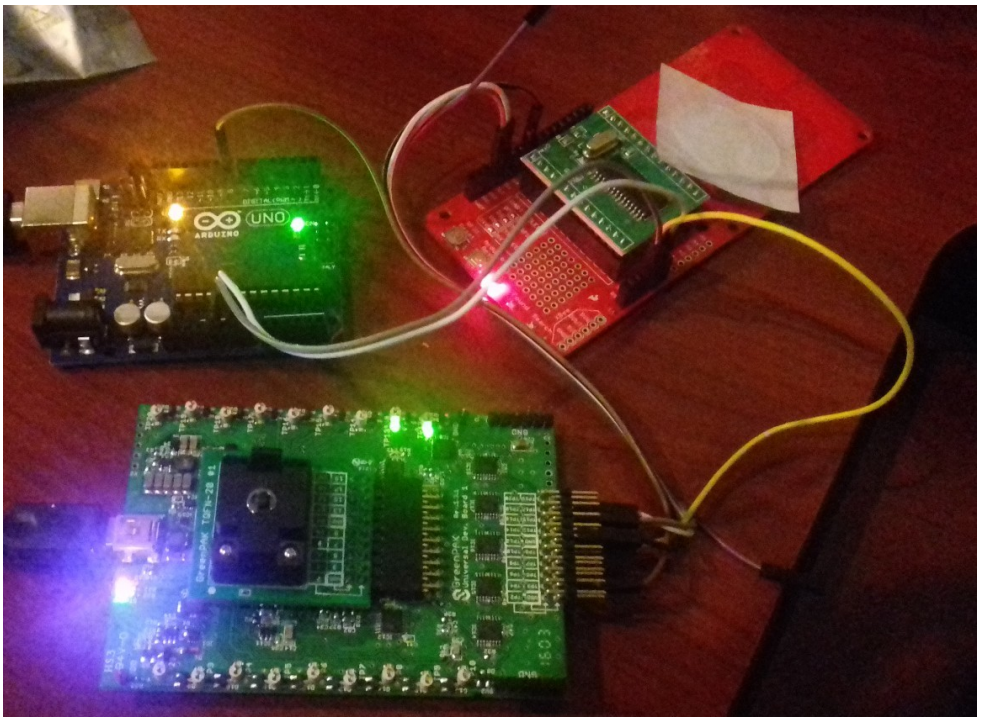

Figure 12. A tag propmts the GreenPAK to wait

This GreenPAK is using nearly all of its resources, so implementing the 19200 Baudrate becomes difficult. Further customizing the serial output could free enough resources to implement an internal clock.

Conclusion

This app note was created to lay out the basics of how to create a design for the GreenPAK SLG46531V to work as a master IC in some applications. It described a low-security system that is certainly more than enough to drive a doggy door, but that can be perfected by adding (for example) another GreenPAK to analyze the response of the SM130, which could be used to authenticate the tag used. Furthermore, the system could use another method to drive the lock, or we could even select another kind of lock such as an electromagnetic one.

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.