Low Cost HF RFID Multiplexer Examples

ABSTRACT

The purpose of this report is to document low cost multiplexer examples for use with low power high frequency (HF) RFID transceivers (like the S6700 and the TRF7960) and their associated system or board microcontroller GPIO pins.

The scope of this document is limited to the applications, circuits and hardware discussed within.

References:

- Coto 2900 Series Relays
  http://www.cotorelay.com/html/reed_relay_2900_series.htm
- CD74HCT4066 Data Sheet
  http://focus.ti.com/lit/ds/symlink/cd74hct4066.pdf
- Selecting the Right Texas Instruments Signal Switch
  http://focus.ti.com/lit/an/szza030/szza030.pdf
- Mini-Circuits Silicon Switch Products
  http://www.minicircuits.com/products/switches_main.html
  http://www.minicircuits.com/pdfs/HSWA2-30DR+.pdf
  http://www.minicircuits.com/pdfs/RSW-2-25P.pdf
- Peregrine Semiconductor Silicon Switch Products
  http://www.peregrine-semi.com/
  http://www.psemi.com/pdf/datasheets/pe4257ds.pdf
  http://www.psemi.com/pdf/datasheets/pe42641ds.pdf
- Skyworks Silicon RF Switch Products
  http://www.skyworksinc.com

Contents

List of Figures

1 Multiplexer Concepts

In some RFID applications there are very valid reasons for using one reader and multiplexing (or switching) through multiple antennas. This is especially true with low power transceiver usage if the system design calls out for close-in absolute coverage in an area that if one larger antenna was used and medium to small form factor tags were being used, a magnetic “hole” would exist. Another example would be the need to read tags in multiple zones without the associated cost of multiple readers and multiple host controllers. The concepts and methods shown in this document with proven circuits is a genuine attempt to assist designers and users of embedded low power HF RFID systems achieve cost targets but not sacrifice system performance.

From a high level, a multiplexed RFID reader system will consist of one reader and multiple antennas. For practical purposes (and brevity) in this document, we will discuss two, four and eight channel versions.

The digital logic outputs from a microcontroller, also called GPIO, will control the silicon switches and relays to select a given channel. This sort of control must be written into the microcontroller firmware and can be done as a loop for automatic switching or done via GPIO control commands from a host controller. The decision of where the control lies must be considered by the designer carefully based on the tag protocol, the level of error recovery required and the timing requirements of the given application or system.

The silicon switches or relays used in the application must also be considered carefully, based on system cost targets, performance required, life expectancy and board space available. Isolation between channels and careful board layout are the key details that should not to be overlooked.

In this first example, a Coto 2911 (1 Form C) relay was used with one GPIO from a microcontroller. This is a good relay to use as it has a coaxial shield available and diodes inside to protect the microcontroller from the field collapse when the relay switches.

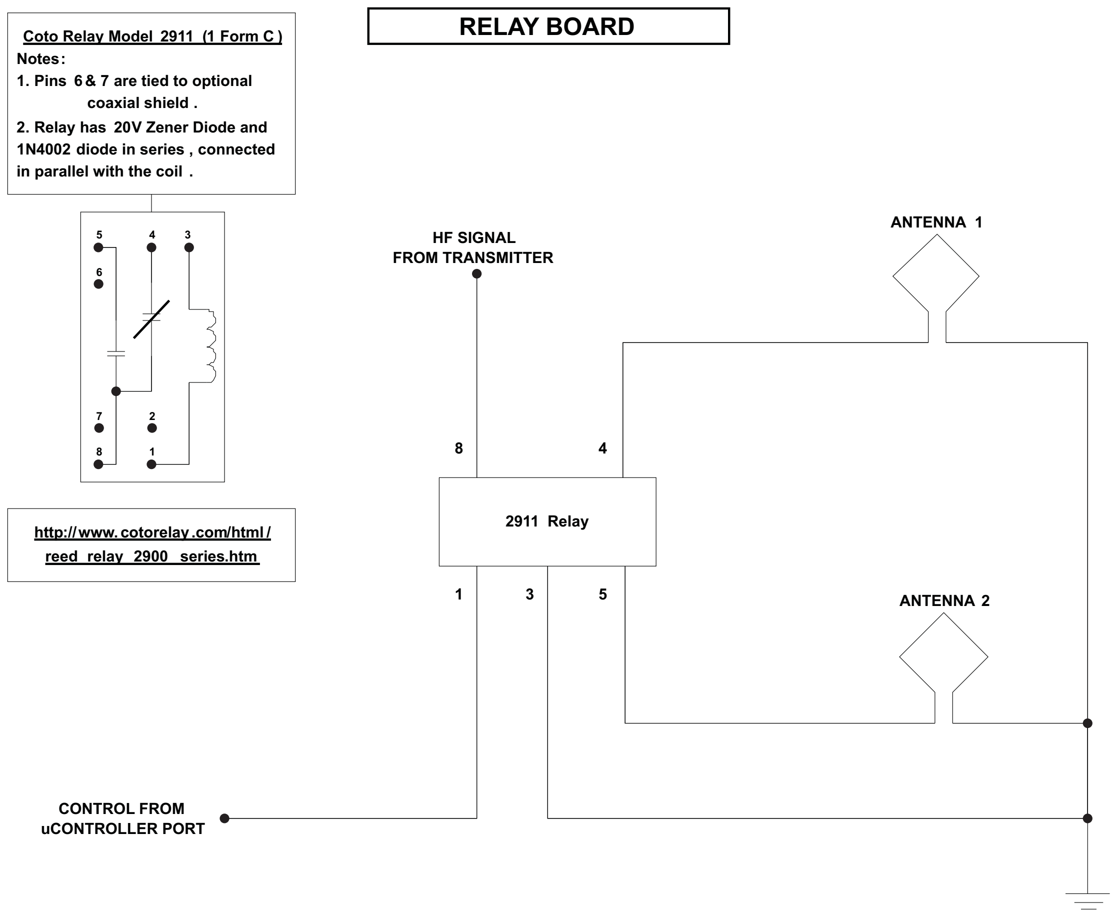

Coto Relay Model 2911 (1 Form C)
Notes:
1. Pins 6 & 7 are tied to optional coaxial shield.
2. Relay has 20V Zener Diode and 1N4002 diode in series, connected in parallel with the coil.
5
4
3
6
7
2
8
1
http://www.cotorelay.com/html/reed_relay_2900_series.htm
RELAY BOARD
HF SIGNAL FROM TRANSMITTER
ANTENNA 1
8
4
2911 Relay
1
3
5
ANTENNA 2
CONTROL FROM uCONTROLLER PORT

Figure 1. 2-Channel Multiplexer Using Two Relays and One GPIO

In this second example, two Coto 2904 relays were used with one GPIO from a microcontroller and an inverter.

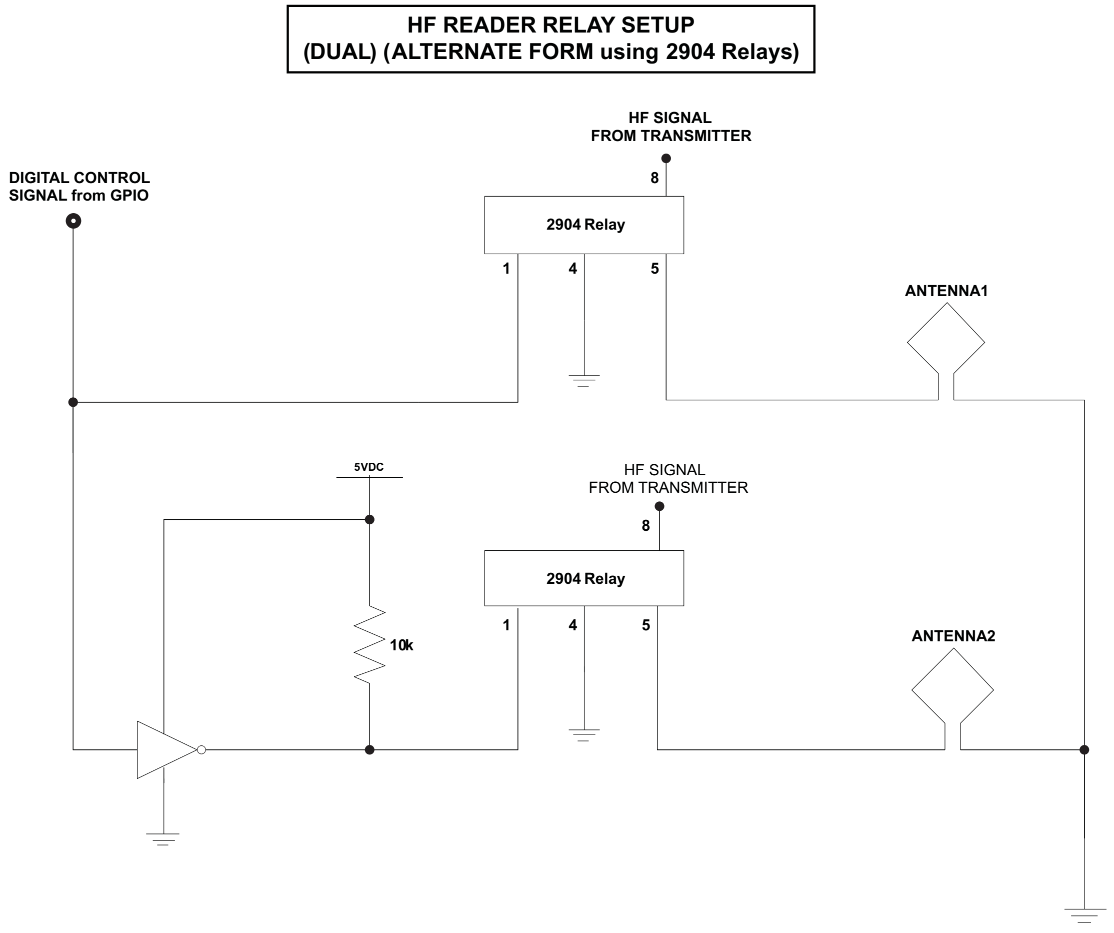


Figure 2. 2-Channel Multiplexer Using Two Relays and One GPIO

In this third example, a logic circuit is used in conjunction with one CD4066 silicon switch IC. The CD74HCT4066M switch is not a perfect switch to use, but it represents a very low cost example that does work, but there is some signal attenuation and a potential for crosstalk if the layout is not done correctly. This example could be improved by the use of higher quality silicon switches directly where the CD4066's are shown in Figure 3 (see the references at the beginning of this document for other switch manufacturers).

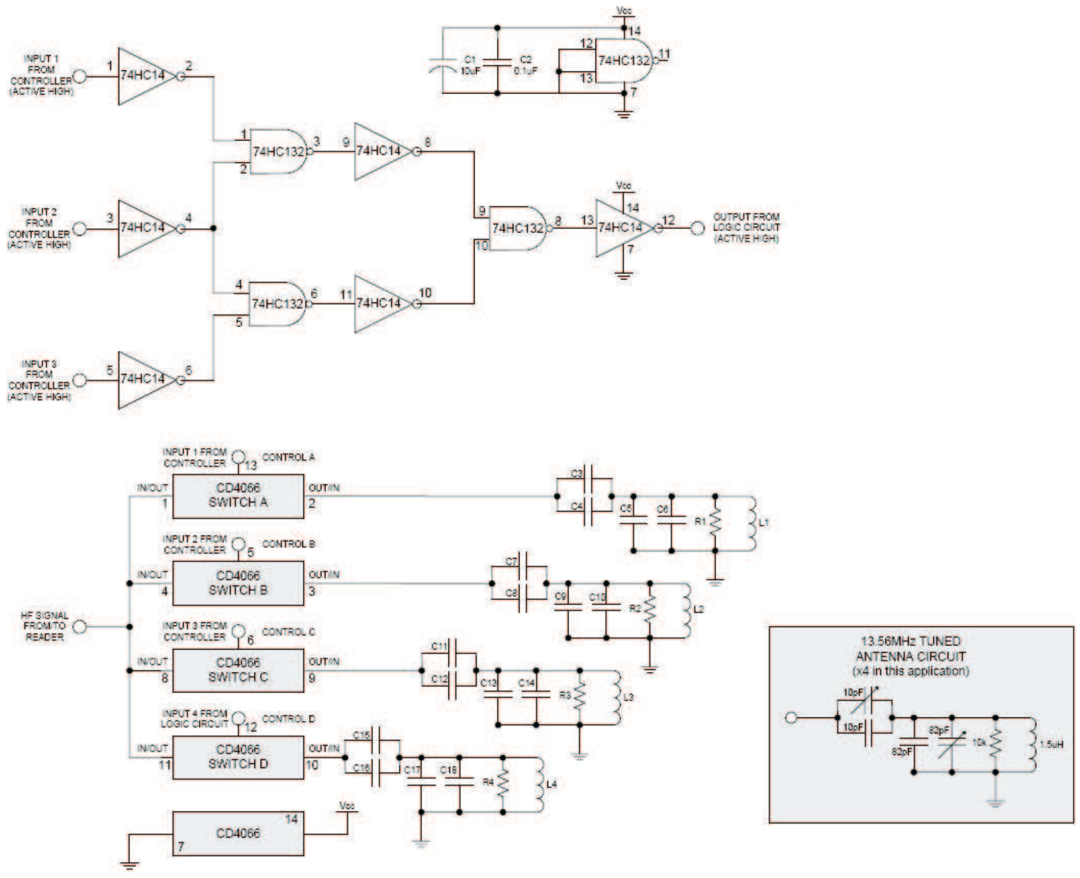
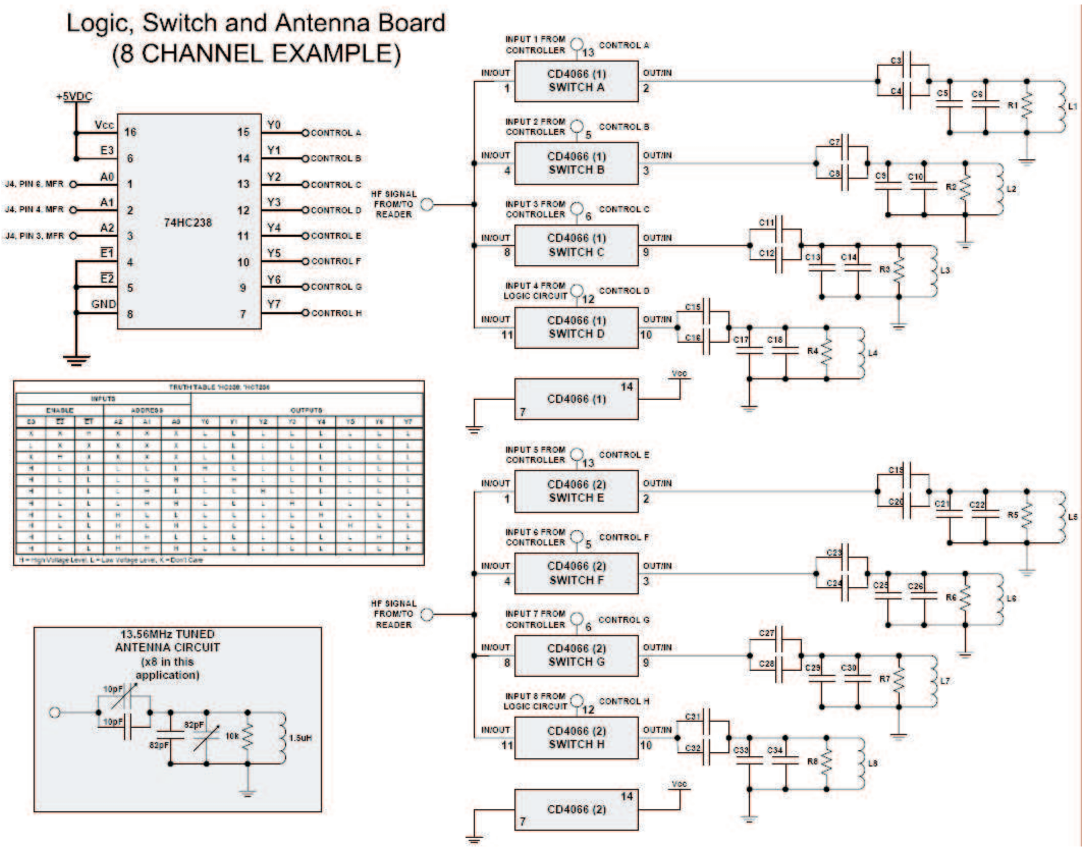

INPUT 1 FROM CONTROLLER (ACTIVE HIGH)
INPUT 2 FROM CONTROLLER (ACTIVE HIGH)
INPUT 3 FROM CONTROLLER (ACTIVE HIGH)
74HC14
74HC132
OUTPUT FROM LOGIC CIRCUIT (ACTIVE HIGH)
Vcc
C1 10uF
C2 0.1uF
HF SIGNAL FROM/TO READER
CD4066 SWITCH A
CD4066 SWITCH B
CD4066 SWITCH C
CD4066 SWITCH D
CD4066
IN/OUT
OUT/IN
CONTROL A
CONTROL B
CONTROL C
CONTROL D
INPUT 4 FROM LOGIC CIRCUIT
13.56MHz TUNED ANTENNA CIRCUIT (x4 in this application)
10pF
82pF
10k
1.5uH

Figure 3. 4-Channel HF RFID Multiplexer Using Silicon Switches and 3 GPIOs (Very Low Cost)

In this fourth example, a 3 to 8 mux is used in conjunction with two CD4066 silicon switch ICs. Again, the CD4066 might not be the right switch for every application, but it can be replaced with a switch of the designers' choice (see the references at the beginning of this document for other switch manufacturers).



TRUTH TABLE 'HC238, 'HCT238

INPUTS						OUTPUTS							
ENABLE			ADDRESS										
E3	$\overline{E2}$	$\overline{E1}$	A2	A1	A0	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	X	H	X	X	X	L	L	L	L	L	L	L	L
L	X	X	X	X	X	L	L	L	L	L	L	L	L
X	H	X	X	X	X	L	L	L	L	L	L	L	L
H	L	L	L	L	L	H	L	L	L	L	L	L	L
H	L	L	L	L	H	L	H	L	L	L	L	L	L
H	L	L	L	H	L	L	L	H	L	L	L	L	L
H	L	L	L	H	H	L	L	L	H	L	L	L	L
H	L	L	H	L	L	L	L	L	L	H	L	L	L
H	L	L	H	L	H	L	L	L	L	L	H	L	L
H	L	L	H	H	L	L	L	L	L	L	L	H	L
H	L	L	H	H	H	L	L	L	L	L	L	L	H

H = High Voltage Level, L = Low Voltage Level, X = Don't Care

Figure 4. 8-Channel HF RFID Multiplexer Using Silicon Switches and 3 GPIOs (Very Low Cost)

In the fifth example, shown in Figure 5, a full reader multiplexer schematic showing the TRF7960/61 being used with an MSP430F2370 is shown. The switches being used are SPDT from Peregrine Semiconductor (http://www.peregrine-semi.com/). This company makes a variety of packaged switches well suited for exactly this application. Please see the references at the beginning of this document for the links to the data sheets on these types of switches. The MSP430F2370 GPIO ports (controlling the switches) are already configured as outputs and the TRF7960/61 base application firmware available from Texas Instruments RFID Systems has been modified slightly (see below for the details) to perform the multiplexing operations. Please note the leftover GPIO ports on the MSP430 could also have switches on them to increase the number to the required number, based on a specific application.

Firmware considerations are very important to note here as the implementation of the hardware alone is not sufficient to realize a complete system as proposed and the current firmware on the TRF7960EVM does not have provisions for a host controlling the GP I/O. Using the available GP I/O of the embedded MSP430F2370 microcontroller running on TRF7960EVM as an example, a developer can reference the modifications below that are needed to gain control of the GP I/O from the host TRF7960EVM GUI and select/deselect the digital switches.In the firmware project, there are two files that need to be modified. They are hardware.h (where the definitions of the ports are) and host.c (where the statements for control are).

The part of hardware.h that should be modified is for the port definitions 1.3 through 1.7 on the MSP430 and starts at line 47.

It currently reads as follows:

```
#define LEDportSET          P1DIR |= 0xFC;
#define LEDallOFF           P1OUT &= ~0xFC;
#define LEDallON            P1OUT |= 0xFC;
#define LEDpowerON          P1OUT |= BIT7;
#define LEDpowerOFF         P1OUT &= ~BIT7;
#define LEDtypeAON          P1OUT |= BIT6;
#define LEDtypeAOFF         P1OUT &= ~BIT6;
#define LEDtypeBON          P1OUT |= BIT5;
#define LEDtypeBOFF         P1OUT &= ~BIT5;
#define LED15693ON          P1OUT |= BIT4;
#define LED15693OFF         P1OUT &= ~BIT4;
#define LEDtagitON          P1OUT |= BIT3;
#define LEDtagitOFF         P1OUT &= ~BIT3;
#define LEDopenON           P1OUT |= BIT2;
#define LEDopenOFF          P1OUT &= ~BIT2;
```

and should be changed to:

```
#define LEDportSET          P1DIR |= 0xFC;
#define LEDallOFF           P1OUT &= ~0xFC;
#define LEDallON            P1OUT |= 0xFC;
#define LEDpowerON          P1OUT |= BIT7;
#define LEDpowerOFF         P1OUT &= ~BIT7;
#define SWITCH_4ON          P1OUT |= BIT6;
#define SWITCH_4OFF         P1OUT &= ~BIT6;
#define SWITCH_3ON          P1OUT |= BIT5;
#define SWITCH_3OFF         P1OUT &= ~BIT5;
#define SWITCH_2ON          P1OUT |= BIT4;
#define SWITCH_2OFF         P1OUT &= ~BIT4;
#define SWITCH_1ON          P1OUT |= BIT3;
#define SWITCH_1OFF         P1OUT &= ~BIT3;
#define LEDopenON           P1OUT |= BIT2;
#define LEDopenOFF          P1OUT &= ~BIT2;
```

The part of host.c that should be modified is to allow for the host GUI to control the above newly defined ports 1.3 through 1.7 and can be added after the firmware version control (which starts at line 695).

```
 else if(*pbuf == 0xF5)
 {                                         /* SWITCH_1 on*/
          SWITCH_1ON;
 }
else if(*pbuf == 0xF6)
 {                                         /* SWITCH_1 off*/
          SWITCH_1OFF;
 }
else if(*pbuf == 0xF7)
 {                                         /* SWITCH_2 on*/
          SWITCH_2ON;
 }
else if(*pbuf == 0xF8)
 {                                         /* SWITCH_2 off*/
          SWITCH_2OFF;
 }
else if(*pbuf == 0xF9)
 {                                         /* SWITCH_3 on*/
          SWITCH_3ON;
 }
else if(*pbuf == 0xFA)
 {                                         /* SWITCH_3 off*/
          SWITCH_3OFF;
 }
else if(*pbuf==0xFB)
 {                                         /* SWITCH_4 on*/
          SWITCH_4ON;
 }
else if(*pbuf == 0xFC)
 {                                         /* SWITCH_4 off*/
          SWITCH_4OFF;
 }
```

These modifications, once added to the code and compiled with the rest of the project then loaded on a board similar to what is seen in Figure 5, allow for the TRF7960EVM Host GUI to now control these GP I/O lines and drive the digital switches as shown in the schematic. The commands to be sent from the GUI (using the Send Raw button in the Test Tab or by using a terminal program like HyperTerminal or Docklight) have the same structure as the Get Version command and would be: 0108000304**F5**0000, where the F5 part of the string would turn Switch 1 on and 0108000304**F6**0000 would turn Switch 1 off. These commands can also be simply issued directly with the GUI using the Send button in the Test Tab, as the GUI will assemble the string for the user. For example, the user would type F5 in the String to Send window and hit the Send button to turn on Switch 1, likewise type in F6 in same window, and hit Send to turn off Switch 1.

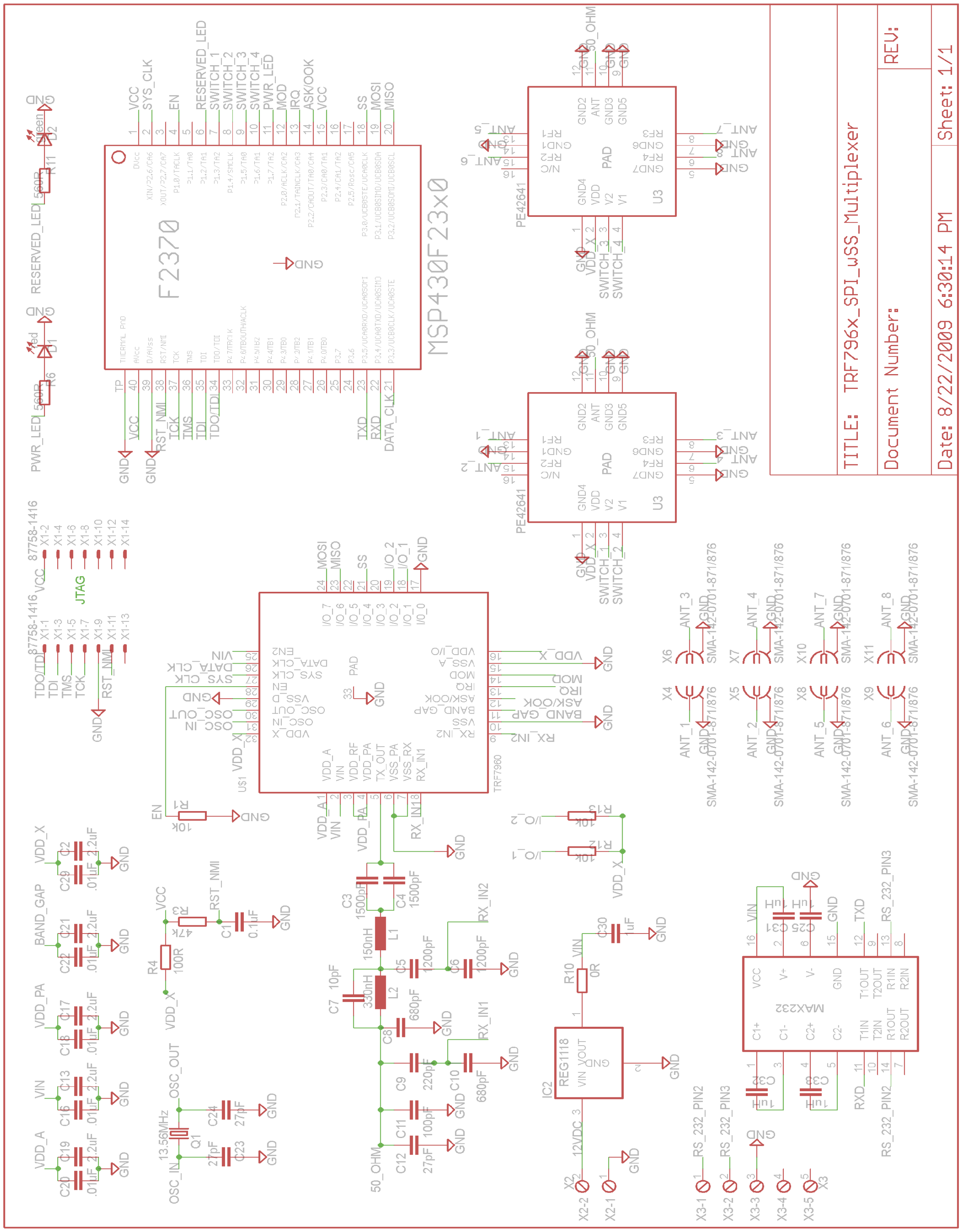


Figure 5. 8-Channel HF RFID Multiplexer Schematic Using Silicon Switches and 4 GP I/Os