

Supplement to article:

Experimental RFID Reader

Elektor Electronics September 2006, p. 34

Software and alignment

Martin Ossmann

After assembling the card, you should first check whether the 13.56-MHz oscillator is actually oscillating and supplying a clock signal to the microcontroller. After this, you can download the program to the microcontroller via the ISP interface.

Programming the Atmel microcontroller

The firmware is available for download as source code and a hex file. Don't overlook the fuses during the programming process. The following fuses must be enabled:

External Clock, 4 ms startup
 Boot Flash Size 1024
 Brown Out Level 2.7 V

Make certain that the JTAG interface is disabled.

Several pins of the microcontroller provide signals that are important for alignment. These diagnostic pins are listed in the following table.

Pin	Function
PortB.0	Modulation: 0 = off, 1 = on
PortB.1	Trigger: rising edge = Tx start; Trigger: falling edge = Rx start
PortB.2	Not used
PortB.3	Rx sampling pulse
PortB.4	Rx signal

Five jumpers are provided to select several program options, which take effect after a reset. They are listed in the following table.

Jumper fitted	Function
None	Operation via RS232 at 9600 baud, 8N1
JA	Fast REQA test for oscilloscope observation during initial setup
JB	REQA test with results shown on the LCD
JC	Fast WUPA test for oscilloscope observation
JD	Read 16 bytes of block 0 and display the first 8 bytes (UID etc.) on the LCD with the error code
JE	Read 16 bytes of block 4 and display the first 8 bytes on the LCD with the error code

Initial setup

You can begin the setup process after the microcontroller has been programmed. Start by fitting jumper JA to select the Fast REQA Test program option. This causes the program to repeatedly loop through the following sequence: (1) The transmitter is switched off for approximately 2 ms, which causes the Mifare card to perform a power-on reset (POR). (2) The transmitter is switched on for approximately 2.5 ms, after which a REQA command is sent to the card. The Trigger test pin is set to '1' when the command is sent, so it can be used to trigger an oscilloscope for making measurements. (3) The Trigger signal is reset to '0' when the Mifare card responds with a load modulation signal after the command has been transmitted. If you trigger the oscilloscope on the falling edge of the Trigger signal, you can use a sniffer probe to observe the load modulation.

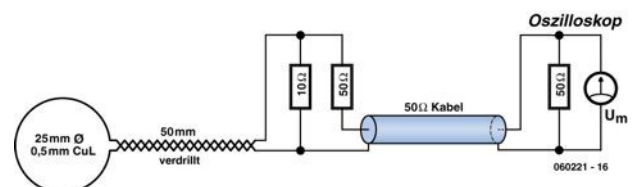


Figure 1. Construction of the sniffer probe.

Sniffer probe

Energy is essentially transmitted to the 13.56-MHz card by inductive coupling, so it is interesting to make an approximate measurement of the strength of the magnetic field generated by the experimental transmitter. This can be done using a sniffer probe made from 0.5-mm enamelled copper wire as illustrated in **Figure 1**. The two resistors produce a relatively well defined impedance, even at 13.56 MHz. The sniffer probe is connected to the oscilloscope by a length of 50-ohm cable, which is terminated in 50 ohms at the oscilloscope. The conversion factor of this probe is approximately 40 A/Vm.

We measured a voltage of 800 mV_{pp} in the middle of the transmit coil of a Philips Mifare reader. This corresponds to a field strength (H) of 32 A/m (peak-to-peak), which in turn corresponds to an effective value of 12 A/m.

All that's left to complete the receiver alignment is to adjust the comparator reference level with R13. Trimpot R13 must be adjusted so the comparator properly distinguishes '1' and '0' values. **Figure 5** shows the comparator output at the beginning of the response to the REQA command (note that the comparator output is inverted relative to the signal shown in Figure 3). To check that the adjustment is correct, it's a good idea to make another test with the reader operating in REQA Test mode (fit jumper JB and reset the reader). The received bytes are shown on the display in this mode, along with a code that indicates whether they were received without any errors (display e:0000).

High speed: assembly language

Here we would like to make a few remarks about the design of the software. Communication with the RFID card requires the utmost in speed, so it is programmed in assembly language. This consists of a single routine called TX-RX, which generates the transmit keying pulses at the right intervals. The intervals to be maintained are defined in a precalculated table to keep everything simple and very fast. After generating the transmit signal, the routine switches to receive mode and samples the comparator output at a rate of approximately 800 kS/s, which yields four samples for each half-bit. The samples are simply stored in a table, and the job of the assembly-language routine (which consists of around 50 instructions) is finished when the previously defined number of samples is reached. Analysis of the response using C routines can then start.

High-Intelligence: C

The transmit and receive functions both require some complex operations. For transmitting, checksums must be computed and added to the user data, bytes must be converted to bits with associated parity, and bits must be converted into transmit symbols and transmitted. All these calculations are most conveniently programmed in C, which fortunately does not involve any additional expense thanks to the freeware WinAVR (GNU) compiler. Things are also complicated at the receiver end. First, the software must identify the half-bit boundaries from the samples and determine the bit values. The bits then have to be converted into bytes, and the parity bits must be checked. Finally, a CRC check may be necessary. These tasks can also be programmed conveniently in C. At the end of all this, the user data is available so it can be evaluated and displayed.

If none of the jumpers are fitted, the reader can be operated via the RS232 interface. Several functions are available, as listed in the following table.

Command	Function
R	Read and display all data (4 sets of 16 bytes)
W	Write 4 bytes to an address
C	Read continuously until a key is pressed
T	Anticollision command test
S	Loop through the select sequence

Data protection

Several safeguards are used with ISO 14443 cards to enable the reader to detect errors that may occur in data transmissions of this sort. The reader and the associated software must be able to use all these methods properly. As already mentioned, each byte is protected by a parity bit. Groups of bytes in the UID are collectively protected at several places in the protocol by block checksums (bitwise XOR of the bytes), which are called block check characters. Relatively long commands and their responses are protected using a CRC (cyclic redundancy check) method. You can refer to the ISO standard for details, but in any case all these methods are implemented in the software.

If you want to incorporate new commands in the software, you must ensure that the right methods are used in each case. In the existing software, the reader records errors in the RXerror variables and then displays them so you can see which errors occurred.

Collision detection

Modern RFID devices, such as those that comply with ISO 14443, are designed to support collision detection and collision handling. This makes it possible for a single reader to recognise and address multiple cards 'in parallel'. How this works is briefly described below.

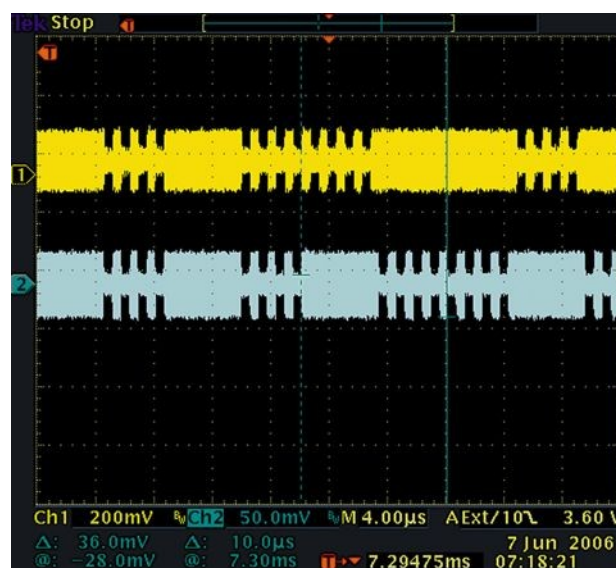


Figure 6. Signals from two cards with a bit collision.

The key factor for collision detection is the modulation and coding of the individual bits transmitted to the reader by the RFID devices. Only one half-bit is modulated in each bit interval. If the auxiliary carrier is detected in both halves, this means that one or more RFID devices transmitted a '1' at the same time that another RFID device (or devices) transmitted a '0'. This results in a collision. The corresponding signal waveforms are shown in **Figure 6**.

This figure was obtained by placing two Mifare cards next to the transmit coil (one on either side at a distance of 1 cm) and sensing their fields individually using

separate sniffer probes. A bit collision is clearly visible between the two cursor lines. Naturally, such a bit collision can only occur if all the cards transmit their bits synchronously.

RFID systems use several special commands to 'eliminate' collisions so the cards can be addressed individually. The key factor is unique identification of each card by its 64-bit UID. In somewhat simplified form, the process works in principle as follows.

The reader transmits the first k bits of the UIDs of all cards that should respond (in the beginning, $k = 0$). In response, each card transmits the remaining $(64 - k)$ bits of its UID. This means that all cards send their complete UID the first time. If no collision occurs, there is exactly one card and its UID is now fully known. On the other hand, if a collision occurs the reader can determine the position of the first bit where a collision occurs. It then knows that there are at least two cards with UIDs that are the same up to this position and different at this position. It can continue to process the cards with a '1' at this position and the cards with a '0' at this position separately by transmitting a UID sequence up to and including this position (with the value of k increased accordingly).

The reader repeats this procedure iteratively along the branches of a decision tree until all collisions have been resolved.

To speed up the process, the 64 bits are not all handled at the same time, but instead in groups so the bits that are already known do not have to be transmitted again each time, which would cost unnecessary time. Once the reader knows all the UIDs, it can activate the cards individually or place them in the quiescent (Halt) state.

Automatic collision detection and resolution is not yet implemented in the software for the experimental reader. It could be upgraded to include this function, but the current transmit power is not sufficient to energise more than one card. The simple amplitude demodulation scheme with a fixed reference level is also not especially well suited to processing responses from multiple cards.

(060221-1)

[1] <http://www.semiconductors.philips.com/products/identification/datasheets/index.html> - mifare

[2] www.semiconductors.philips.com/acrobat_download/other/identification/M028630.pdf