Magstripe Interfacing - A Lost Art

By Acidus (acidus@yak.net www.yak.net/acidus)

-- Intro

Just like Sun Microsystems, people have been forecasting the death of magstripes for years. Yet they are still the most common form of physical authentication in the world. Their wide-spread deployment makes components for them cheap, and home brewed applications limitless. While there is a great wealth of knowledge on the Internet about magstripes, most are over 6 years old, mostly for very specific micro controllers, or have out of date source code with no comments. Straight answers about how Magstripes work and how to interface to a modern PC simply don't exist. I plan to correct that.

-- Brief History

Count Zero wrote the definitive work on magstripes in Nov of 1992 for Phrack 37, entitled "Card-O-Rama: Magnetic Stripe Technology and Beyond" [1]. While an excellent work, discussing the physical characteristics of magstripes, as well as how the data in encoded on them, it contains no information about interfacing magstripe readers. While several people have published works on readers and copiers [2], the definitive guide on interfacing readers to computers was written by Patrick Gueulle in June of 1998 entitled "Interfacing a TTL Magcard Reader to the PC Game port"[3]. This work is extremely short, with no explanation of its Pascal source code.

It has be over 6 years since someone wrote something of substance about magstripe interfacing. The uncommented source code that you can find out there, in [3] for example, is so horribly dated that it will not run on any modern Windows OS (2K, XP). This article will explain in detail interfacing a Magstripe to a computer, explain how to control it, and present easily ported source code that people can use.

-- Magstripe Basics

See [1] for much more information about this subject. Magstripes consist of several magnetic particles held to a PVC card with a glue, and the orientation of these particles (and their magnetic fields) is how the data is stored. Magstripes can contain several Tracks of information, each .110 inches wide. These tracks are defined by several standards; we are most interested in Track 2. This is the most widely used track, having been standardized by the American Banker's Association. This track contains up to 40 characters from a 16 character set.

So how is the magnetic representation understood by computers? Well the reader contains a head which outputs an analog signal of the magnetic fluxes on the card. A specialized chip, called an F2F decoder, converts these signals into digital outputs. Interfacing directly to the analog signals would be insane, and F2F chips are critical for easy interfacing (see [2] for interfacing without a F2F chip). Each F2F chip needs 5 volts (5V) and a ground (GND) as inputs, and for output has a Card Present (CP) line, as well as 1-3 pairs of Clock (CLK) and Data (DATA) lines, one pair for each track the reader supports. These F2F chips decode the magstripe data of each track as a bit stream, using the CLK and DATA line. When the CLK line goes high, DATA line is the value of that bit (low=0, high=1). The CP line goes high when the reader detects that a card is being swiped through it. We will not use the CP line in our implementation.

-- Our Approach.

Using an F2F chip, we can read the bit stream of the data on the card. From the ABA standard, we know how those bits represent numbers and characters (shown later in Figure 3). We simply need a way for a computer to read in the bit stream, and write some software to convert it to the characters defined in the ABA standard. The good news is readers with built in F2F chips are easy to find and pretty cheap. They can be purchased from Digikey, Jameco, etc under the name TTL readers. You don't want to buy the expensive readers that connect directly to a serial or parallel port, as these reader will require special software to read from them.

We are going to adapt an approach shown in [3], and interface through the game port. This has several advantages. The game port provides 5V and GND to run the reader without an external power supply; its has 4 easy to read inputs; game ports are usually free whereas serial and parallel ports are not; and even legacy free PCs without parallel ports, serial ports or ISA slots still have game ports.

-- Parts

Getting a TTL reader is pretty easy. Digikey has a large section on them. Simply search for "mag card." Other online stores carry them as well. You want the simplest and cheapest one you can get. We are only interested in Track 2 readers, we don't care about cabling since we will make our own, and we don't want motorized readers. We want the readers where you manually swipe the card (these are a lot cheaper). I am a big fan of the Omron V3A family of readers, specifically the V3A-4, since it offers exactly what we need. Expect to spend ~$15-30 dollars.

In addition, you will need a DB15 male connector to plug into the game port. Make sure you don't buy a DB15 HD for VGA connections; Jameco part #15034 is what you want. You'll also need soldering tools, some wire, a hot glue gun, and some electrical tape. I used few feet of speaker wire to connect the reader to the game port, so the reader could sit in front of the computer.

-- How to Interface

Make sure you can get the data sheet for your TTL reader, and that it supports Track 2. Check the manufacturers site. Using the pinout from the data sheet, solder wires to the 5V, GND, DATA, and CLK pins, making sure you are using the CLK/DATA pair for Track 2 if you reader supports multiple tracks. The contacts you have to solder to could be quite small; after soldering the wires, I covered the contacts on the reader with hot glue to make sure they wouldn't shift, break, or short each other out. Take your time and solder carefully.

Next, solder the ends of the 5V, GND, DATA and CLK to the DB15 connector as shown in Figure 1.

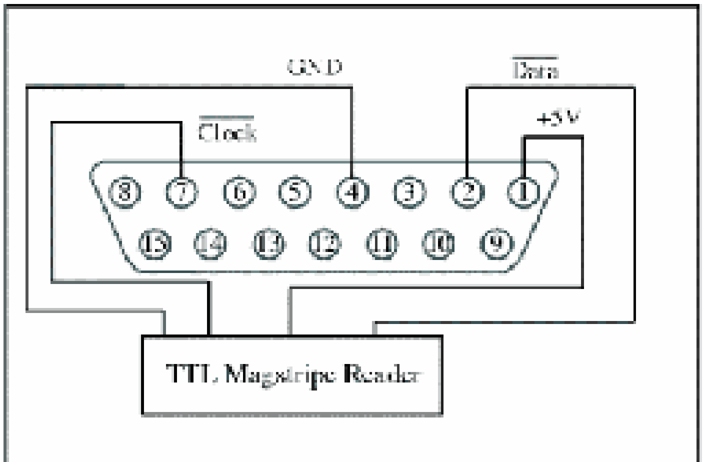
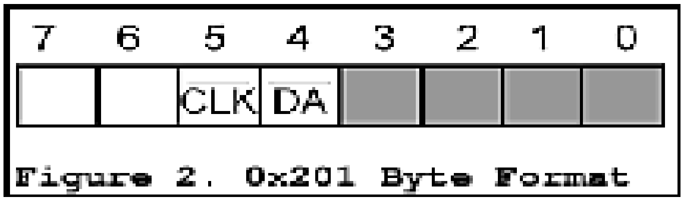


Figure 1. Wiring the Magstripe

A word of warning: not all the grounds on a game port will really be grounds. Check using a LED to make sure the 5V and GND going to your reader are really active.

What we have done is soldered the reader outputs to the input pins on the game port that correspond to joystick buttons. We can now access the bit stream from the F2F chip as if we are checking the status of joystick buttons! We read from the game port by reading from I/O port 0x201. If we wired the reader to a game port as shown in Figure 1, when we read from I/O port 0x201, we will receive a byte whose format is described in Figure 2.



7	6	5	4	3	2	1	0
		$\overline{\text{CLK}}$	$\overline{\text{DA}}$	▨	▨	▨	▨

Figure 2. 0x201 Byte Format

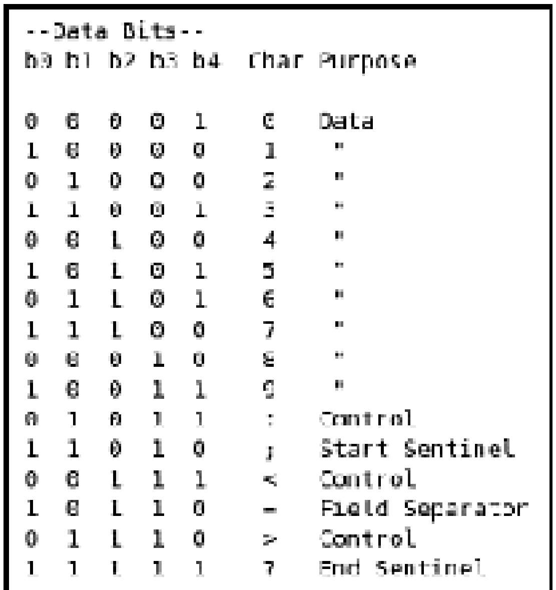
Notice that the inputs are inverted. Thus for each corresponding bit, 0 means a 1 from the card and a 1 means a 0 from the card. How do we read a byte from port 0x201? It varies from language to language, but is normally of the form "inputByte = INP(address)."We then use "AND 16" to extract the DATA bit from the 5th bit of the read byte . This gives us the bit stream.

-- Bit Stream Explained

The bit stream of a track 2 magstripe card looks like this:

[leading zeros...][start][Data...][end][LRC][trailing zeros...]

The data on the card is a 16 character set, represented by 5 bits, 4 for the character, 1 as odd parity. The character set for Track 2 is shown in Figure 3.



--Data Bits--

b0	b1	b2	b3	b4	Char	Purpose
0	0	0	0	1	0	Data
1	0	0	0	0	1	"
0	1	0	0	0	2	"
1	1	0	0	1	3	"
0	0	1	0	0	4	"
1	0	1	0	1	5	"
0	1	1	0	1	6	"
1	1	1	0	0	7	"
0	0	0	1	0	8	"
1	0	0	1	1	9	"
0	1	0	1	1	:	Control
1	1	0	1	0	;	Start Sentinel
0	0	1	1	1	<	Control
1	0	1	1	0	=	Field Separator
0	1	1	1	0	>	Control
1	1	1	1	1	?	End Sentinel

Figure 3. Track 2 Set

We are only interested in 0-9, and the start, stop and field characters. They show us where in the bit stream we have valid data, and how that data is divided into fields. The number of leading zeros and trailing zeros vary, and are there to Sync the clock inside the F2F chip. The trailing zeros are there so you can run your mag card backwards through a reader. Please note the F2F chip doesn't look for the start or stop characters, or anything like that. It simply reads the fluxes and outputs the CLK and DATA lines. Our program must scan the stream and find the start character. Once you find it, you know where the 5 bit boundaries are for each character, and can read the data on the card. We are interested in all Data from the Start character to the Stop Character. The LRC is a checksum uses to make sure the data on the card is correct. The source code doesn't check the LRC. Rarely is it necessary, and for the most part any problems you have will have to deal with the timing loop, as described in the next section.

-- Problems

Remember all those advantages for interfacing to a game port? There is one big downside. The game port doesn't generate an interrupt when a joystick is moved or a button is pressed. This means in our software we have to use lots of loops when reading the bit stream so we can trap the changes of the CLK line. To read a single data bit from the DATA line, we have to do the following:

```
Step 1: Loop, checking for when the CLK goes high
Step 2: Save the value of the DATA line
Step 3: Loop, waiting for the CLK to go low
Step 4: If we still have more bits to read, go to Step 1
```

This is a time critical loop. The program has to catch each and every bit, in real time, since the bits are not saved or cached in anyway. If you have several programs running and your computer is off doing something else and misses a bit, the data will be wrong. How time critical it is can vary with language and hardware. On a Pentium 150, the PASCAL code from [2] compiled and worked in fine in DOS, as did a version in C, but an implementation in Qbasic, even compiled, failed. A Pentium II 266 under Win2K ran the VB version just fine with no load, but failed in I tried while defragging a drive. The 3.4+ GHz machines of today should have no problem.

Lastly, a note on IO port access. If you want to use my VB code, and are using Win2K or XP, you will need to grab the Inpout32 from [4]. This is a DLL that allows you to directly access IO ports under 2K and XP, which don't allow direct access like Win 9x and ME do.

U**pdate(7/2/2004):** Please note the all development has moved from VB to C, and currently runs on Windows 9x, ME, NT, 2K, and XP, and Linux. I will explain the VB source code in the following section, though the concepts in C remain the same.

-- Source Code Explained

VB is used because it's easy to understand and port, and I don't want the language to interfere with the explanation. The code is limited in that it will only deal with cards slid in the proper direction. It is heavily commented, so here is a quick overview. We read the DATA from the card just as described above, using a set of time critical loops. The array is sized to 240 since we will never have more than 240 bits on Track 2. We don't need to use the CP line, because the CLK line will not go high until a card is in the reader.

After the first stage, our array contains entire bytes from 0x201, when we only care about the Data bit. The next stage uses AND 16 to mask off the DATA from the 5th bit. The array now has only 1's and 0s, the raw bit stream. Next we scan the array looking for 11010. This marks the start of the data. Once found, and we than read 5 bits at a time, looking for the end character, 11111. When we find it, we read through the bit stream from the start character to the stop character at 5 bit intervals (since each character in the stream is 5 bits), and decode the characters using the chart in Figure 3. We append these decoded characters to a string until we have read all the data between the start and stop.

Here is a sample of the decoded bit stream of a Visa

```
Account Number:        4313 0123 4567 8901
Expires:               5/06
Output:                ;4313012345678901=0506101xxxxxxxxxxxxx?
```

The 101 after the expiration data is common to all Visa cards. See [1] and [2] for many more examples of card formats.

-- Improvements

The code give here is very basic, so people can understand whats going on. More advanced code and application are available from [5]. One of the first improvements would be allowing the card to be swiped in both directions. You capture the bit stream the same way. You then look for the start character, then attempt to find the end character and then the LRC. You then calculate the LRC to make sure the data is correct. If any of those steps fail, simply try again going backwards through the stream. Interrupt driven programming would also be a plus. We didn't use the CP line, because our polling method doesn't need it, and the game port doesn't have it. Using the CP line and the CLK line, you could wire something to say the strobe line on a parallel port and trigger an interrupt so the computer doesn't have to keep polling until a card is really there. Further, a backend of card prefixes could be constructed, so a program could attempt to guess what type of card was swiped (similar to how Ethereal IDs packet types).

Update (7/2/2004): The more advanced code is called Stripe Snoop, and is available for download at http://stripesnoop.sf.net. It has all the improvements described above, and more.

-- Closing and Thanks

"If I have seen farther than others, it is only because I have stood on the shoulders of giants." Those giants, most notably Count Zero, made this article possible. Thanks to all the hackers who learned so much and documented their discoveries. Please take this code and improve on it as much as you'd like. Just remember to give credit as I have: hackers have been working on Magstripes for nearly 15 years. Swipe all the cards in your wallet, you would be amazed at the stuff you will find encoded on them; I've found SSNs, PIN numbers, birth dates,and more.

Check out [5] for more information and some cool applications of the code. All referenced material is also available from [5], in case the original sites go down.

There is no group, there is only code.

--References

Copies of most of the info from these links can be found at www.yak.net/acidus

[1] Card-O-Rama: Magnetic Stripe Technology and Beyond, Count Zero, http://www.phrack.org/phrack/37/P37-06 -The Definitive guide on magstripes: Formats, encoding, and reading.

[2] Magnetic Stripe reader/writer, Luis Padilla Visdomine, http://www.gae.ucm.es/~padilla/extrawork/stripe.html -An excellent web page, Luis builds a mag reader and writer from scratch. Lots of examples of card formats, rather advanced.

[3] Interface a TTL Magcard Read to the PC Games Port, Patrick Gueulle http://www.blackmarket-press.net/info/plastic/magstripe/card_tech/3IFD.pdf -A very short paper on PC interfacing with Source Code.

[4] Logix 4 U Homepage, www.logix4u.cjb.net - Contains the inpout32.dll needed to directly access IO ports using INP and OUT on Win2k and XP machines.

[5] Most Significant Bit Homepage, Acidus, www.yak.net/acidus - My homepage, lots of info on a variety of subjects.