

# **Magnetic Card Analyzer**

**CSCI-190 Project Report**

**F. Thomas Hogans**

## Project Abstract

This project aims to read a specific set of data from the magnetic stripe of a standard magnetic card. Utilizing a microcontroller and card reader, the project accepts the card into the reader and outputs the data via a serial connection to any terminal platform. The data output is the content of the card's second track, which according to ISO 7811 standards, must contain the personal/account number for which the card was issued.

Due to the varied nature of the data, dependent upon the type of card being analyzed, there is no uniform way to output the data. Therefore the track data will be dumped in its "raw" form; i.e. in ASCII form, but including control characters and separators. This will provide the discerning analyst with more information about the actual content of the magnetic stripe and avoid issues with a predefined output pattern not being compatible with the data. An example of how this could be a problem is the difference on track two between driver's licenses and credit cards. The credit card uses the "expiration date" field for simply a month and a year; whereas the license fills this area with birth date and expiration, with no control character separating the two. In order to make the program compatible with all types of these situations, an unpractical amount of ISO standard handling would have to be included. As this aims to serve as a general analyzer of the magnetic stripe data, and not specifically geared towards any particular type of card, the raw output method makes the most sense.

## Project Specification

This project reads the data from Track 2 of the magnetic stripe of any ISO-compliant magnetic stripe card. There are several requirements that must be met for this process to complete in a reliable and accurate fashion:

- The connections between the card reading mechanism and microcontroller must be secured and free from interference.
- The microcontroller must be able to output the final data in a convenient, human-readable fashion.
- Integrity checks must be performed on received data to verify accuracy.

The following hardware components were utilized for this project:

- ZiLOG Z8 Encore! Z8F6403 Evaluation Board (99C0868-001)
- Panasonic ZUM2121S451 Insertion-Type Magnetic Stripe Reader

The **Z8F6403** is part of the ZiLOG Z8 Encore! Family of 8-bit multipurpose microcontrollers. It features a 20Mhz microprocessor, 64k Flash program memory, four 16-bit timers, and 24 vectored interrupts in an 80-pin QFP package, among other features. Its high-level development options make it a prime candidate for embedded development, especially for a simple demonstrational project such as this.

The Panasonic **ZUM2121S451** magnetic stripe reader is a low-cost, effective solution for reading magnetic striped cards. It takes a 5 volt power input and has a card reading speed of 10~120cm/sec. It reads track 2 of the 3 tracks on the stripe. This track is written with a 5-bit scheme (4 data bits + 1 parity), which allows for sixteen possible characters, which are the numbers 0-9, plus the six characters : ; < = > ? . The selection of six punctuation symbols and the 16 codes that represent them map to the [ASCII](#) range 0x30 through 0x3f, which defines 10 digit characters plus those 6 symbols. The data format is as follows:

- **Start sentinel** — one character (generally ';')
- **Primary account number** — up to 19 chars
- **Separator** — one char (generally '=')
- **Expiration date** — four characters (MMYY)
- **Service code** — three characters
- **Discretionary data** — as in track one
- **End sentinel** — one character (generally '?')
- **LRC** — one character

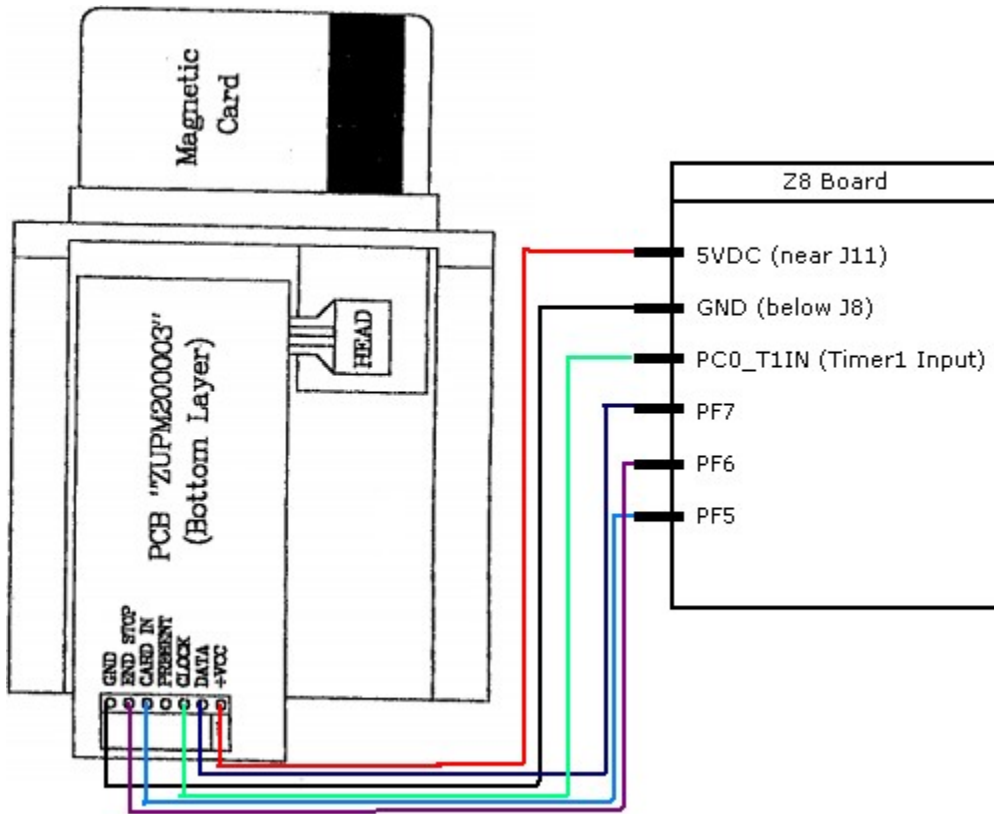
The software that manages the serial output and connection with the card reader is a ZiLOG Developer Studio II project file named `magreader.zdproj`. The following files are also included:

- **magreader.c**: the main program code
- **uart.c**: routines for handling UART serial connection
- **timer.c**: routines for initializing and handling timer events

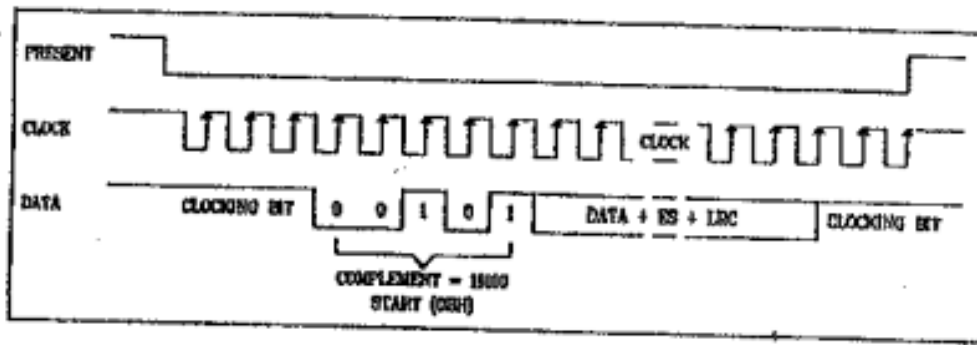
## Implementation & Construction

The different phases of the reading process can be broken down into three distinct parts. First, the physical data must be extracted from the magnetic strip and converted to digital data. This is accomplished entirely by the card reader, and therefore does not need to be addressed by the software.

The second step, which must be implemented in the software, is to read the signals from the card reader and recreate the bit stream. This process is achieved by connecting the card reader to the microcontroller, lining the pins of the card reader to the following general-purpose input/output pins:



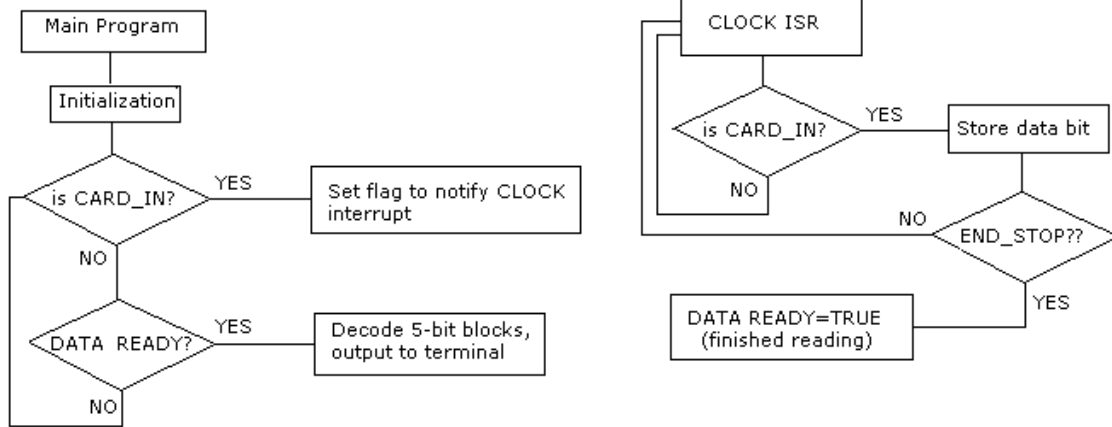
The operation of the microcontroller must be able to relate correctly to the signals being applied on the pins via the reader. As shown above, a number of status indicators are available to determine the action and position of the card. By utilizing the `CARD_IN` and `END_STOP` signals, it is possible to determine when the card has first entered the reader and when it has been fully inserted. In between these two events, the `CLOCK` and `DATA` signals rapidly rise and fall to provide an output of the bit stream that has been processed from the magnetic stripe. The `DATA` signal represents the actual bits of data that are being read; the `CLOCK` signal is an indicator of when a new bit is being represented on the `DATA` signal. When the `CLOCK` moves from low to high, the reader is instructing to interpret the state of the `DATA` signal as the next bit in the bit stream. As the bits are received least-significant bit first, it is simply a matter of adding each new bit onto the left of the previously received ones. See the diagram below:



After the bit stream has been obtained from the card reader, the software must make sense of it. The data is represented in groups of 5 bits, each group representing a value from 0000b to 1111b (0 to 16) plus 1 parity bit. The codes for the characters are shown below:

Parity	D3	D2	D1	D0	Character	Function
1	0	0	0	0	0 (0H)	Data
0	0	0	0	1	1 (1H)	Data
0	0	0	1	0	2 (2H)	Data
1	0	0	1	1	3 (3H)	Data
0	0	1	0	0	4 (4H)	Data
1	0	1	0	1	5 (5H)	Data
1	0	1	1	0	6 (6H)	Data
0	0	1	1	1	7 (7H)	Data
0	1	0	0	0	8 (8H)	Data
1	1	0	0	1	9 (9H)	Data
1	1	0	1	0	: (AH)	Control
0	1	0	1	1	; (BH)	Start Sentinel
1	1	1	0	0	< (CH)	Control
0	1	1	0	1	= (DH)	Field Separator
0	1	1	1	0	> (EH)	Control
1	1	1	1	1	? (FH)	End Sentinel

The program operation is to essentially wait until the magnetic card is inserted and read the data until an END\_STOP signal tells us that the card is pushed all the way in. In order to accomplish this, the main program loop waits for the CARD\_IN signal and sets a global flag for a gated input timer that interrupts when the CLOCK moves from low to high. The interrupt, if the CARD\_IN flag is set, knows to record the DATA bits until it receives an END\_STOP signal. The following flowchart illustrates the operation of the software:



## Retrospective

This project has provided a great opportunity for exploring the underlying fundamentals of embedded systems and their interaction with external components. In addition, the exposure it allows to such a ubiquitous technology as the magnetic stripe complements the learning experience in an unparalleled way.

On a more technical level, this also provided an opportunity to use certain elements of the Z8's functionality that had not been addressed in labs. Previously we used timers mostly for output, and except for PWM sensor input, never used the input modes. This project's use of a gated timer to monitor the CLOCK signal, along with the overall CLOCK/DATA scheme, lays the foundation for a very common low-level data transfer procedure that would likely have to be utilized in future embedded ventures.