

Swope Filter Tag Reader

Summary

The Swope Filter Tag Reader reads an NFC tag mounted on a filter frame and returns the first 32 bytes in the tag memory. It can also write the first 32 bytes of tag memory. This information can be used to update FITS headers.

This program reads and writes only unformatted (non-NDEF) MiFare Classic tags although the hardware can read and write other tags and formats.

Software

Communications is through an RS-422 serial port on the Galil DMC 4070 controller on the Swope IMB. The setup is 19,200 baud, 8-bits, no parity, one stop bit. There is no flow control; the Galil handshaking lines (RTS and CTS) are tied.

Commands sent to the tag reader are one-byte ASCII characters as described below. The command character is not echoed back to the sender although it is read and interpreted immediately on receipt.

Commands

Command	Name	Details
0 or 1	Select wheel	These are ASCII characters 0 (numeral zero, not oh) and 1 (numeral 1, not ell).
i	Card ID	Prints the 32-bit MiFare tag ID in hex. This probably isn't a useful number to know but if a reply is returned you know it's a MiFare Classic tag.
r	Read tag	If a tag is present a stream of 32 bytes is returned. A brand-new blank tag will likely contain zeros (0x00, not the ASCII numeral) so echoing the return to a terminal would be confusing for new tags. An end-of-string byte (\0) is <i>not</i> necessarily included in the 32 bytes.
R	Reset	Reboots the microcontroller and resets the tag readers. Total time should be less than one or two seconds and you'll see a "WD\r\n>" returned. The "WD" refers to a watchdog reset, which happens only with this command. The watchdog timer is disabled except for this reset function. The input stream to the tag reader is cleared on reboot so commands sent immediately following the "R" and before the WD response will be lost.

s	status	Returns system status information. In order, the returned items are the program name ("Filter_Reader"), this software version number (yyyymmdd), firmware version of the selected (see 0 or 1 above) card reader, and the contents of the program status byte. Bit 0 of the status byte shows the currently selected wheel number (0 or 1). The other seven bytes are program error codes. If they're not all zeros (that is if the status byte is not 0x00 or 0x01), you should probably do a reset.
w	Write tag	These commands write the succeeding 32 bytes to the tag. A carriage return (\r) terminates the data entry if fewer than 32 bytes are sent. The \r will not be included in the 32 bytes but the remainder of the 32 bytes will be padded with spaces (' ' or 0x20). Example: w H-Beta Filter\r will write "H-beta Filter" padded with spaces into the tag.
\r (CR)	No-op	Sending a carriage return will return the command prompt string "\r\n>"

Errors

The tag reader may reply with an error code. The error code format is "\r\nENN\r\n>" where NN are ASCII numerals as described below. Note that the "\r\n>" sequence right after ENN distinguishes this from a read-back of valid tag data which always returns 32 space-padded bytes.

Code	Meaning
E99	Not a command (byte sent but not recognized as a command)
E10	No tag or an unrecognized tag at wheel 0
E11	No tag or unrecognized tag at wheel 1
E20	Problem writing to tag on wheel 0
E21	Problem writing to tag on wheel 1
E80	Initialization error on card reader for wheel 0
E81	Initialization error on card reader for wheel 1

Reboot transmissions

The tag reader will send these messages without being commanded. All of these happen after a reboot and all are followed by a “\r\n>” sequence.

Message	Meaning
BO	Reboot after power brown-out (supply voltage under 2.7VDC)
PB	Pushbutton reset (the button on the MCU hardware was pushed)
PU	Power up start
WD	Watchdog reset (only happens with the R command)

Firmware

The firmware source code resides in a GitHub repository ([link](#)). It's written for an Atmel ATtiny4313, compiles under avr-gcc (Windows version) and uses a few avr-gcc specific macros.

Hardware

Tag reader

The NFC tag reader controller is the NXP PN532 chip. We chose an Adafruit-branded version of the [microBuilder board](#).

The PN532 can communicate via SPI, I2C, or serial interfaces. We chose I2C (TWI). In retrospect, it might've been easier to use SPI for this project since the PN532 has only one fixed I2C address option, which makes using two of them more difficult.

MCU

The MCU is an Atmel ATtiny4313. This chip has a hardware UART for serial communications to the Galil controller. We're using a 14.7456 MHz crystal that allows us to choose common serial baud rates.

Bus switch

Since the PN532 breakout boards have identical I2C addresses we will use the [PCA9543AD](#) I2C bus switch to select which of the two boards we want.

RS-422 conversion

To convert the RS-422 signals to TTL serial levels, we'll use XXX transceivers.

Glossary

Item	Description
I2C	Inter-Integrated Circuit bus
IMB	Instrument Mounting Base
MCU	MicroController Unit
NDEF	NFC Data Exchange Format
NFC	Near Field Communications
SPI	Serial Peripheral Interface
TTL	Transistor-Transistor Logic
TWI	Two-Wire Interface, similar to I2C, functionally equivalent
UART	Universal asynchronous receiver/transmitter