µProcSys STaTS Device

1 Overview

- SPI Terminal Driver
- Terminal UART Repeater
- Temperature Sensor
- 3-5 V Compatible
- STM32F3 Based

2 Description

The Microprocessor Systems **S**PI **T**erminal **a**nd **T**emperature **S**ensor (STaTS) Devices is a self contained, state-of-the-art SPI Sensor platform; combining the usefulness of an inaccurate temperature sensor with the unnecessary bulk of a highly sophisticated evaluation board. The STaTS device additionally allows for an SPI-to-UART and UART-to-UART passthrough of ASCII characters, useful for embedded systems requiring multiple terminals implemented in an inexplicably complicated fashion.

[*Less Sarcasm*] This fake "datasheet" and accompanying Nucleo-F303RE program have been written to provide an instructional opportunity to implement a **S**erial **P**eripheral **I**nterface (SPI) communication scheme for an arbitrary SPI sensor design as well as provide a test UART interface platform. The device has been constructed to roughly model a possible SPI communication implementation with the understanding that many SPI peripheral devices exist with drastically different interface schemes. The scheme presented here is similar to I^2C in the use of registers to differentiate data bytes while still leveraging the full-duplex communication nature of the SPI protocol. The internal temperature sensor of the STM32F303RE is used to measure temperature. Further, terminal interfacing is included in order to blend topics already learned with the tools currently being instructed. If the student groups desire, this example SPI device may be omitted and substituted with any other SPI device of their choosing for the associated laboratory task if given the explicit approval from the course instructor.

3 UART Interface

The UART interface provided through the STaTS module consists of a terminal connection to a computer (port 1) and a wired UART port (port 2) for connection to a second device. The terminal connection is provided via a virtual USB com port and has a baud rate of 115200 Hz. The wired UART port has a baud rate of 38400 Hz. When a character is received from port 1 it is directly written to port 2. Likewise, characters received on port 2 will be re-transmitted on port 1. Port 1 contains several display functionalities to format the terminal such that the characters received on port 1 are printed scrolling on the top half of the display and the characters from port 2 are scrolled on the bottom.

4 Serial Interface

The STaTS has a standard 4-wire serial interface that is compatible with SPI (and possibly other) interface standards. This interface requires 8-bit word transfers. Communication to and from the STaTS is controlled through serial registers.

When the STaTS is selected as the communication target (the CS line is pulled LOW), the number of the desired register to interact with is transferred first from the Master. During this initial transfer, the STaTS transmits the current value of STS_REG. This transfer is followed by the data word to write into the indicated register (on SDI) and the data word read from the register (on SDO).

The transaction diagram for this serial interface is given in Figure 1 below. In this diagram, the R# bits are the target register, the S# bits are the STS_REG register, DI# bits are the data written to the STaTS and DO# bits are the data transmitted from the STaTS. A "DEBUG" line is also included in the device for troubleshooting communication issues. This line toggles when the CS line is toggled as well as when a single SPI transaction is complete.

Note that there is a required delay between the assertion of the CS line and the SCLK starting, a delay between transmitted bytes, and a delay at the end of the transmission. This value, t_{delay}, should be approximately 10 µs[1]. This delay should also be enforced between CS assertions.

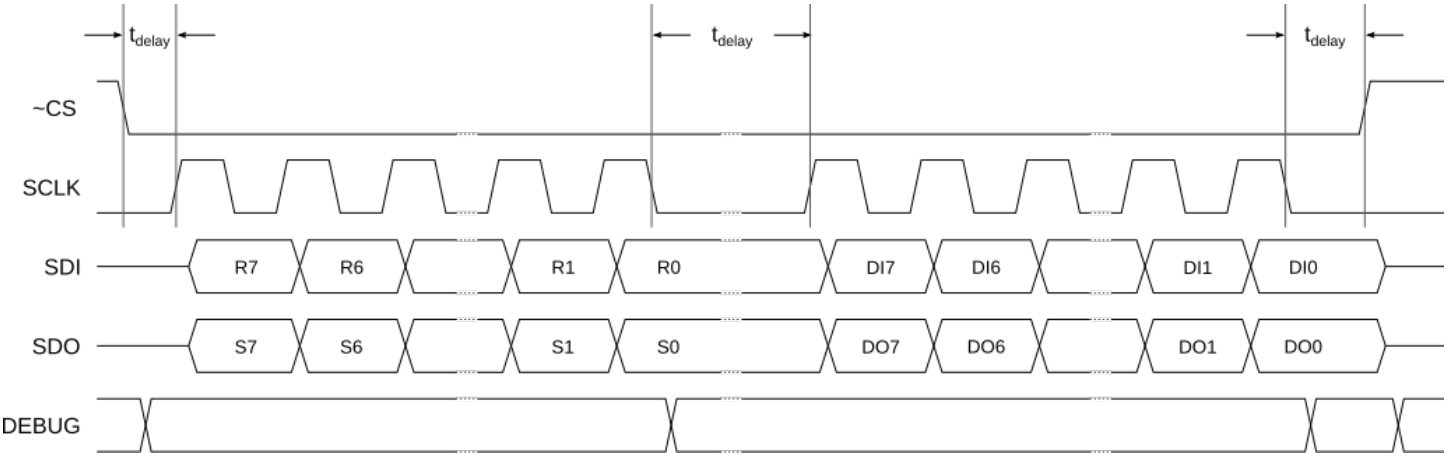

Figure 1: Transaction diagram for the STaTS serial interface.**CORRECT VERSION**

[1]This number is likely much larger than it needs to be; however, it is given as such to ensure enough time has passed.

5 Register Map

The registers contained within the STaTS device are given in Table 1

Table 1: STaTS Register Map

Register Number	Register Name	Description
0	CTL_REG	Control register
1	STS_REG	Status register
2	TMP_AVG	Number of averages per each temperature reading
3	TMP_LO	Low byte for temperature measurement
4	TMP_HI	High byte for temperature measurement
5	CH_BUF	Terminal character buffer
6	TXT_ATTR	Terminal text attribute buffer
7	V_MAJ	Major version number
8	V_MIN	Minor version number
9	DEVID	Device identification number

6 Register Descriptions

6.1 CTL_REG (Reg#: 0): Write Only, Reads 0x00

This register controls the current operation of the STaTS. A map of the bits is given below.

7	6	5	4	3	2	1	0
ULKDID	resv.	resv.	CHBCLR	TRMRST	TRMCLR	RDTMP	RST

Bit 0: RST: Reset the SPI device. Setting this bit to one will force a software reset of the STaTS device.

Bit 1: RDTMP: Request a new temperature measurement to be taken.

Bit 2: TRMCLR: Clear the terminal driven by the STaTS device. This does not reset the text attributes.

Bit 3: TRMRST: Clear the terminal driven by the STaTS device and reset the text attributes.

Bit 4: CHBCLR: Clear the character buffer to be sent from the STaTS device to the Master.

Bit 5: resv.: Unused/Reserved

Bit 6: resv.: Unused/Reserved

Bit 7: ULKDID: Unlock the Device ID for writing. Device ID register will re-lock after the next full SPI register+data word transfer.

6.2 STS_REG (Reg#: 1): Read Only

This register indicates the current operation and faults of the STaTS. A map of the bits is given below.

7	6	5	4	3	2	1	0
CHBOV	NCHBF1	NCHBF0	TOVRN	TRDY	TBUSY	FLT	RDY

Bit 0: RDY: 1 if STaTS is ready for operation.

Bit 1: FLT: 1 if a fault has occurred during temperature measurement.

Bit 2: TBUSY: 1 if a temperature measurement is currently underway.

Bit 3: TRDY: 1 if a temperature measurement is ready for reading. This bit is automatically cleared when TMP_HI is read.

Bit 4: TOVRN: 1 if previous temperature measurement was overwritten by a newer temperature measurement prior to reading. This bit is cleared when the STS_REG is read through an explicit register request.

Bit 5: NCHBF0: See below.

Bit 6: NCHBF1: Together with NCHBF0 indicate how many characters are in the terminal receiving buffer (CH_BUF) awaiting transmit (2-bits: maximum 3 characters). These bits are cleared when CHBCLR is asserted.

Bit 7: CHBOV: 1 if a terminal receiving overrun has occurred. This bit is cleared when the STS_REG is read through an explicit register request.

6.3 TMP_AVG (Reg#: 2): Read/Write, Reset Value: 0x01

The register TMP_AVG controls the number of temperature readings averaged to produce one output temperature measurement. If a value of 0x00 is written to the register, its current value is unchanged. Any other value will overwrite the current value of TMP_AVG. This allows for the value to be read without modification. The value of TMP_AVG returned during a read/write cycle is always the value prior to writing, if applicable.

6.4 TMP_LO (Reg#: 3) and TMP_HI (Reg#: 4): Read Only

TMP_LO and TMP_HI store the low and high byte of the temperature measurement, respectively. The value of the temperature is a 12-bit number, right justified. Upon reading of TMP_HI, TDRY is automatically cleared. The full 12-bit number is the averaged values of the raw ADC measurements of the temperature sensor. The equation to convert this value to temperature is[2]:

$$\text{Temperature in Celcius} = 357.6 - 0.187\text{Value}$$

[2]Beware that the temperature sensor is internal to the chip and should always read a higher temperature than ambient.

6.5 CH_BUF (Reg#: 5): Read/Write

CH_BUF stores characters received by the STaTS through the terminal interface connected to it. This is a FIFO buffer which holds up to three characters. Upon read of a character from the buffer, the number of characters of the buffer is reduced by one, as indicated by NCHBF1 and NCHBF0. When the character buffer is empty, reading of the buffer will return 0x00. If three characters are in the buffer and a fourth character is received form the terminal, the oldest character is removed from the buffer and an overrun is indicated by CHBOV.

Writing a value to CH_BUF will result in the ASCII value written being transmitted to the connected terminal. The FIFO buffer is unaffected by this. To avoid writing to the terminal when reading from the buffer, write a value of 0x00.

6.6 TXT_ATTR (Reg#: 6): Write Only

This register is used to modify the terminal text attribute properties of characters written to CH_BUF. Any valid VT100 attribute is accepted. Writing to the register is cumulative, allowing for multiple attributes to be active at once. Resetting of the text attributes may be done by writing 0x00 to this register or by resetting the terminal through TRMRST.

6.7 V_MAJ (Reg#: 7) and V_MIN (Reg#: 8): Read Only

V_MAJ and V_MIN together form the full version number of the firmware on installed on the STaTS device; where the firmware version is [V_MAJ].[V_MIN]

6.8 DEVID (Reg# 9): Read/Write, Reset Value: 0xB6

This register returns the STaTS device ID. This value may be changed to differentiate multiple STaTS devices. The DEVID register is write protected. To change the device ID, the unlock bit in the control register (ULKDID) needs to be written to 1. This unlocks the DEVID register to change *for the next SPI transaction only*. Changes to the DEVID register do not survive power cycles or system resets.

7 Technical Specification

For a real device, there would be a bunch of information here about voltage ratings, current consumption, static and dynamic behavior, etc.

8 Pinout

SPI Signal Connections

Arduino Pin	Aux. Header Pin	Signal	Description
A2	CN7.32	CS	Chip Select Line
D11	CN10.15	SDI	Slave Data In
D12	CN10.13	SDO	Slave Data Out
D13	CN10.11	SCLK	Serial Clock
D7	CN10.23	DEBUG	Debug Line

UART Signal Connections

Arduino Pin	Aux. Header Pin	Signal	Description
D6	CN10.25	TX	UART Transmit Line
N/A	CN10.18	RX	UART Receive Line

9 Firmware Installation

In order to make a "STaTS" device, an Nucleo-F303RE development board is needed. Firmware is provided on the Microprocessor Systems website (under Lab 3).

To flash the firmware on the device on Windows:

1. Download and install the STSW-LINK004: ST-LINK Utility.
2. Run the STM32 ST-LINK Utility and connect to the Nucleo-F303RE board. This should be done without the DISCO board connected to avoid connection to the wrong target.
3. Flash the firmware by going to **Target →Program...**, selecting the firmware (.bin file) and clicking **Open**.
4. The default values on the next screen should be OK, click **Start** to complete the process.

To flash the firmware on the device on Linux:

1. Install the stlink-tools package. For Ubuntu based systems, run the command:

```
$ sudo apt install stlink-tools
```

2. Flash the firmware through the command:

```
$ st-flash write LAB-03_STaTS_Firmware.bin 0x8000000
```

To flash on MacOS...to be determined.