Scalar Operation Manual v3.0

FieldLine Inc

Contact: Tyler Maydew, tm@fieldlineinc.com, 303-242-6048

Magnetometer Description:

The FieldLine Scalar magnetometer is an optically pumped rubidium magnetometer. The device is in a small form factor package (Figure 1) and is powered by +12-18VDC.

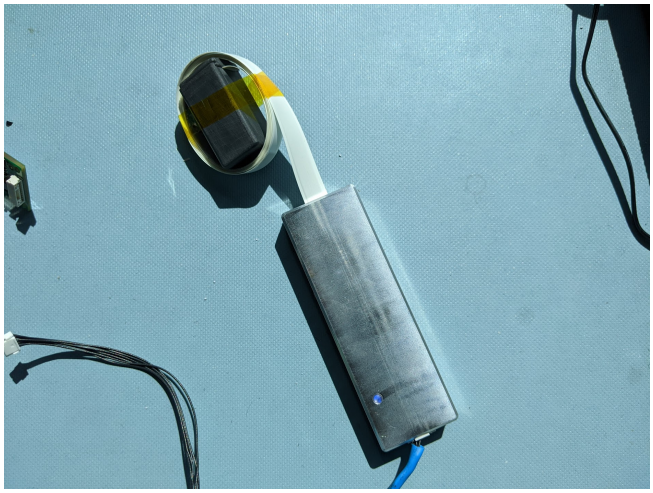

Figure 1 - FieldLine Scalar Magnetometer

The system primarily communicates over UART. The following manual will describe two different methods of working with and communicating with the device. The pin out for the connector is shown below in Figure 2.

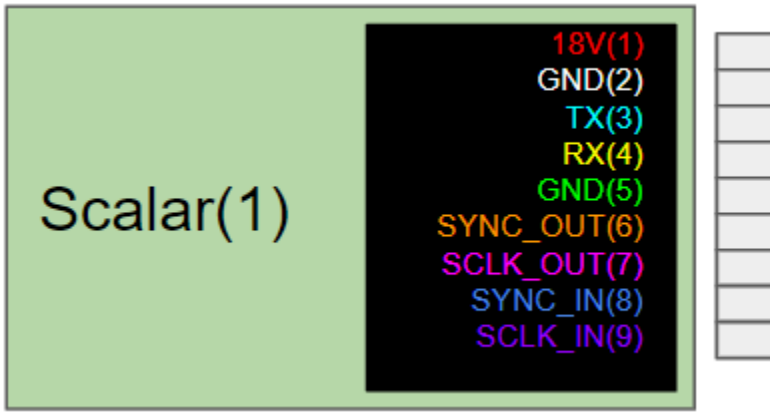

Figure 2 - Scalar User Pin Out

There are a few features to note on the scalar. One is the presence of an LED that will blink during some operations and be solid during others. The following Table 1 describes the action each color indicates.

Table 1: Light Description

LED Description	**System Action**	Corresponding System State
Blinking Orange	System Startup	3
Blinking Green	Heat Stabilizing	4
Blinking Blue	Scanning for Magnetic Resonance	5
Solid Blue	Locked on Magnetic Resonance	6
Blinking Purple	Sensor is flashing a build	N/A

Another feature of note is the sensor is remotely updatable. FieldLine will provide you with details if a new update becomes available.

Magnetometer Conversion Formula:

The sensor provides the magnetic field data in a Frequency Coded Format. To convert data form the magnetometer into T use the following formula (Equation 1):

value*4e6/(2^32-1)/(Rb87 Gyromagnetic Ratio) -- Eq 1

Rb87 Gyromagnetic Ratio = 6.99583 Hz/nT

For a quick reference you can substitute 7Hz/nT as the Gyromagnetic Ratio.

NOTE: The magnetometer will automatically attempt to regain field lock by continually scanning if lock is lossed. The magnetometer does have a dead zone of operation when it is perpendicular to Earth's Field.

Sensor Operation (GUI):

1. Attach the connector side of the provided power cable assembly to the scalar.
2. The cable assembly splits out power from the digital io. Connect a power supply (+12-+18V) to the power section of the cable and connect the UART-to-USB connector to the other side.
3. If this is the first time using the system make sure to go to https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers and download the proper driver to operate the USB-UART converter.
4. Open the FieldLine UART Recorder v0.0.27.
5. Once the recorder is open go to Settings -> Configure Chassis. A pop up (figure 1) will appear:

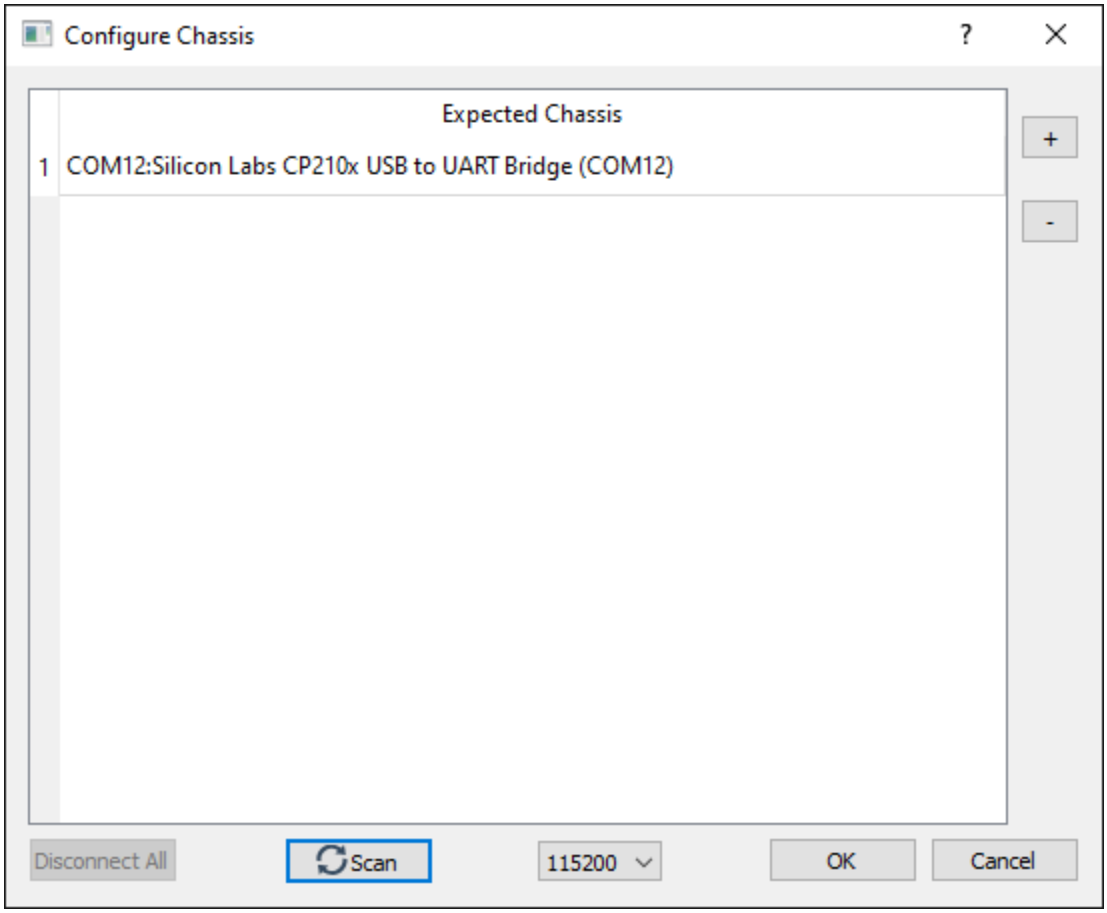

Figure 1 - Configure Chassis

6. The COM port displaying your sensor will appear. Click on line 1, and then hit OK. If you ever want to close the COM port, simply open the configure chassis window and hit disconnect all. If the COM port is not showing up, check if the COM port is appearing on your windows PC.
7. After hitting okay, the connected GUI will show as seen in Figure 2.

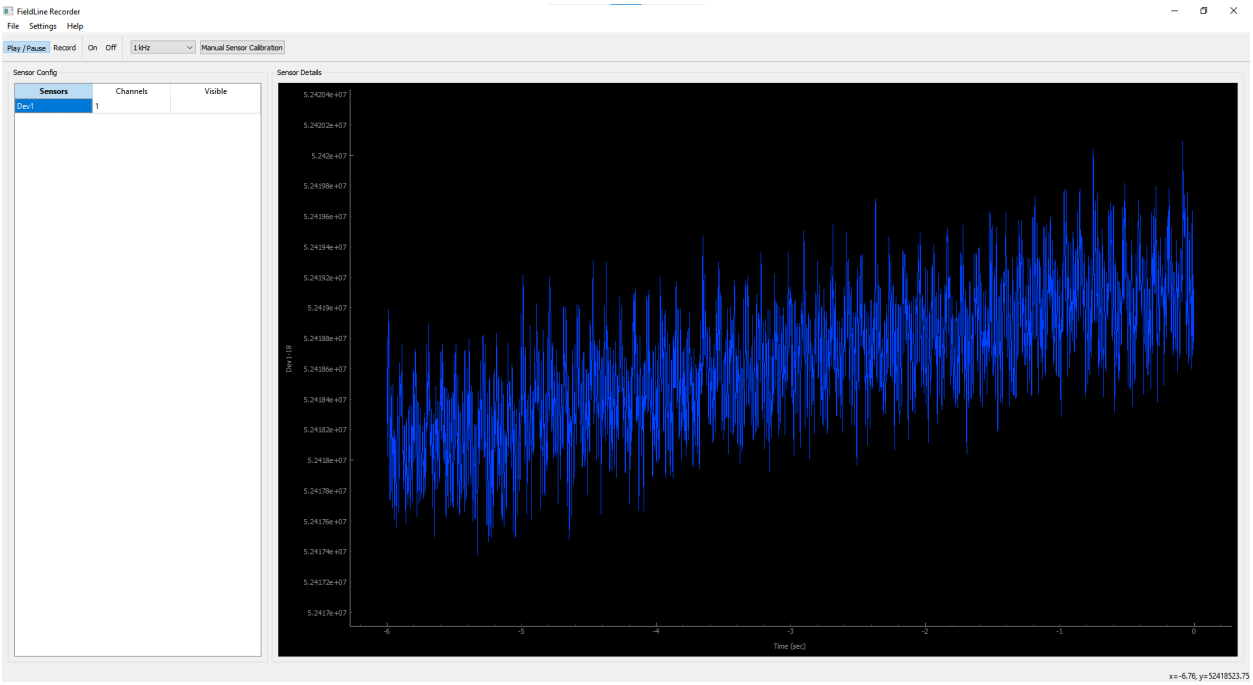

Figure 2 - Magnetometer Signal Present

8. Some notes on the GUI:
    a. To start a scalar please select the Device and then hit the “On” button. “Off” will disable the scalar back to its idle state.
    b. The “Play/Pause” button will enable or disable the streams being requested.
    c. There is a pull down menu where you can select between 1kHz, 500 Hz, 250 Hz and 125 Hz data rates. If you are using 115200 Baud you will only be able to stream one stream at a time at 1kHz. Higher baud rates can handle more streams.

9. Once the sensor is started it will take about ~2minutes to get the magnetometer locked and ready. The sensor will go through the following stages, with the LED colors and states as identified on channel 35:
    a. Warm Up - LED: Blinking ORANGE (state 3)
    b. Laser Lock Scan - LED : Blinking GREEN (state 4)
    c. Scan for Magnetic Resonance: - LED: Blinking BLUE (state 5)
    d. Magnetic Lock: - LED: Solid BLUE (state 6)
10. At this point magnetometer data will appear on channel 18. To turn on a channel click on the “Dev0” box under Sensor Config and the following pop up (Figure 5) will show. Type the channel number (found in the channel list section) and hit “OK”:

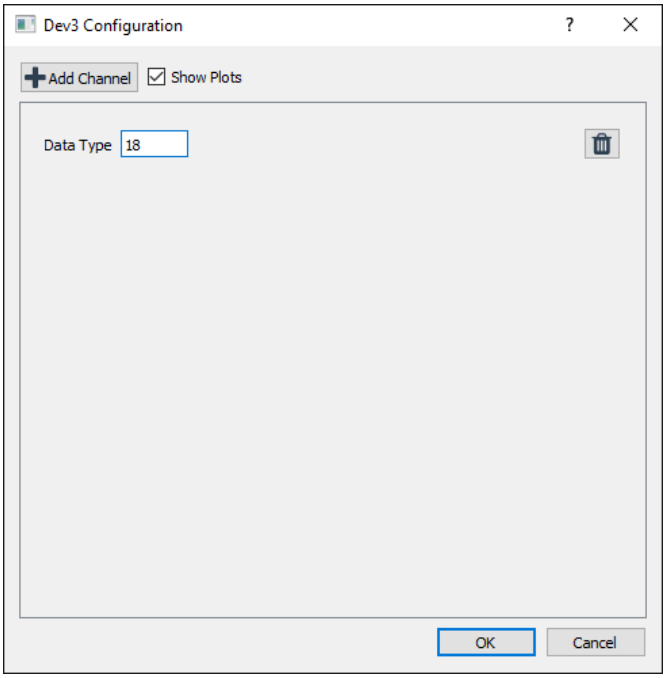

Figure 5 (Sensor Pop Up)

11. To convert the data to nanoTesla (nT) using the following Eq. 1:

    value*4e6/(2^32-1)/(6.99583 Hz/nT) -- Eq 1

12. If you want to record data using the GUI simply click the “record” button at the top and the prompt in Figure 6 will appear:

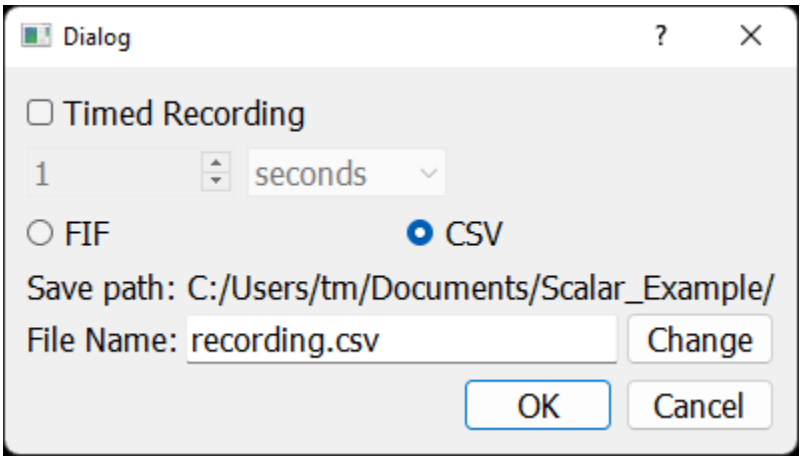

Figure 6 - Recording Prompt

The recordings will be either a .fif or .csv format. To record data a channel must be on and playing. You can do a timed or untimed recording. To stop an untimed recording simply hit the “Record” button again.

13. To turn off the sensor either disconnect power or send the off command.

SENSOR OPERATION (COMMAND LINE):

The sensor communicates via UART and therefore can be easily communicated with using a simple script of your preferred programming language. The following information covers the communication format. The command input stream is the format that data is sent to the magnetometer, and the data stream packet format is how the output data is packetized.

NOTE: The system does utilize an “escape byte” of 0x1b. An escape byte will precede a “special” (0x0A [START], 0x0D [STOP], or 0x1B [ESCAPE]) character.

Command Input Stream Format:

The input stream format is in ASCII. All commands **must** start with an address identifier byte either “@” or “#” depending on what is desired. To address an internal system register the command must be preceded with the “@” symbol. To enable an external data stream the command must be preceded with a “#”. All commands must be sent in a hexadecimal ASCII format.

Below is the input data format:

[address_identifier_byte][address_byte][data_B1.1][data_B1.2][data_B0.1][data_B0.2]

[address_identifier_byte]:
“@” : Will allow user to write to an available user register. See Appendix B Table 1 for user registers.
“#” : Will enable the corresponding data stream. See Appendix B Table 2 for user streams.

[address_byte]:
“XX” two ASCII characters to represent the 8 bit address. Ex. “13” to address register 19.

[data]:
“XXXX” four ASCII characters to present the 16 bit data being written to the sensor. Ex. “FFFF” to send the decimal value “65535”.

NOTE: It is critical that when typing the address for 2, the user types 02. Identically with the data bytes the user should type 0201 to represent 0x201.

Enabling Outbound Data:

To enable outbound data the user will send the '#' followed by the stream address "XX" and then should provide one of the following three values:

1. "FFFF" - This will indicate to the system to output data from this schedule exactly 1 time.
2. "0000" - This will indicate to the system to cease outputting data from this stream
3. "XXXX" - Any other combination of bytes will enable the stream at the specified sampling frequency.

Data stream packet format:

Below is the outbound data streaming format.
NOTE: The following three bytes are treated as special and are escaped:

start_byte = 0x0A, stop_byte = 0x0D, escape byte= 0x1B

CHECKSUM Disabled:

[start_byte][time_stamp1][time_stamp2][data_type1][data_B1][data_B2][data_B3][data_B4]....[data_typeN][data_B1][data_B2][data_B3][data_B4][stop_byte]

CHECKSUM Enabled

[start_byte][time_stamp1][time_stamp2][data_type1][data_B1][data_B2][data_B3][data_B4]....[data_typeN][data_B1][data_B2][data_B3][data_B4][stop_byte][Fletcher-16 C1 Byte][Fletcher 16 C0 Byte]

NOTE: When the checksum is enabled the user should treat any two bytes following a valid stop bytes as the checksum. There will be NO escape bytes in the checksum, i.e. a 0x0A is a valid checksum and not the indication of a new packet.

Maximum Bandwidth:

Due to the UART Protocol each byte of data requires 10 bits. Each packet requires at minimum 9 bytes of data (without checksum). We will disregard escape bytes for this calculation and treat it as a best case maximum data rate.We can calculate the amount of time we need to send a single data type with our default 115200 BAUD to be:

9 [bytes] * 10 [bits/byte] / (115200 bits/sec) = 781.25 us

Knowing that our maximum sampling frequency is 25 kHz (40us) we can see our maximum supported sampling rate for the scalar is 1.28 kHz. However due to system constraints this will have to be rounded to 1 kHz. To send an additional data type will cost another 5 bytes. Thus at 115200 BAUD we can only stream one data type. A user can request one time reads for slow speed data.

Examples

Below are some examples on operating the system.

One Time Read:

The following three commands will (1) write test data to the scratch register, (2) set the read register to the scratch and (3) then stream the data in the scratch.

```
@044f6b //writes 0x4f6b to the scratch
@030004 //sets the read register to 4
#03ffff //schedules the one time read register for on transaction
```

Expected response:
{0x0A, 0x00, 0x00, 0x03, {0x00, 0x04, 0x4f, 0x6b}, 0x0d}

Typical Operation Flow (with Sync):

1. Reset sample count. @000001
2. Enable Status stream. See Appendix A for details. - #230001
3. Start Magnetometer. - @4D001F
4. Monitor status stream until it is in state 6. (Light is solid Blue)
5. Disable Status. #230000
6. Enable magnetometer data - #120001
7. When finished either remove power or disable magnetometer - @4D0000

Status Polling

1. Start Magnetometer. - @4D001F
2. Enable magnetometer data - #120001
3. Poll status by sending #23FFFF at a slow rate. (Too quickly will exceed maximum bandwidth)
4. When finished either remove power or disable magnetometer - @4D0000

Useful Streams

The following lists useful magnetometer streams:

6: Version Number
7: Serial Number (Least Significant Word)
8: Serial Number (Most Significant Word)
18: Magnetometer Output Data
23: Magnetometer Output Data (in pT)
35: Current System State

Appendix A. Sample Communication Python Code

```
#Below is some sample python code that will set the magnetometer at 921600 Baud rate and start the magnetometer
#while streaming channels 18 and 35 at 1 kHz

import glob
import os
import serial
import string
import time

version = '#06ffff'+'\n'
speed_up = '@440003'+'\n'
start_sensor = '@4D001F'+'\n'
start_mag_output = '#120001' +'\n'
start_state_status = '#230001' + '\n'

COM_PORT = "COM12" #make sure to set your own COM PORT

serialPort = serial.Serial(port =
COM_PORT,baudrate=115200,bytesize=8,timeout=2,stopbits=serial.STOPBITS_ONE)
serialPort.write(speed_up.encode())
serialPort.close()
#The above section of code opens the serial port at the default speed and tell it to change the speed to 921600 by writing 3 to the indicated register
#we have to then close the port and reopen at the faster speed

serialPort = serial.Serial(port =
COM_PORT,baudrate=921600,bytesize=8,timeout=2,stopbits=serial.STOPBITS_ONE)
serialPort.write(version.encode())
received_data = None
while(received_data != b'\r'):
    received_data = serialPort.read()
    print(received_data)
#by setting the version register to "FFFF" we requested a one time read of the version number then we print it out, not very nicely :)
#this verifies we are connected to the system, so then we can start the sensor

serialPort.write(start_sensor.encode())
serialPort.write(start_mag_output.encode())
serialPort.write(start_state_status.encode())
```

```
received_data = None
while(received_data != b'\r'):
    received_data = serialPort.read()
    print(received_data)
```

At this point I would probably write a piece of code that polls the output channel 35 (state status) until you see the state as 5. Then proceed with whatever you want to do!

```
serialPort.close() #make sure you close the serial port when you're done!
```

Appendix B. Register Map

Table 2. User Register Map

Address (Hex)	Name	Bits 15:02	Bit 1	Bit 0
0x00	Sync	0	CLRSTR	RST_COUNT
0x03	Read	ADDR[15:0]		
0x04	Scratch	WDATA[15:0]		
0x17	Frequency	FREQ[15:0]		
0x43	Checksum	0		CHECK_EN
0x44	Uart Rate	0	SET_BAUD[1:0]	
0x4D	Enable Scalar	ENABLE[15:0]		

Sync Register

Bit	Field	Type	Reset	Description
15:2	0	R/W	0x0	Reserved
1	CLRSTR	W	0x0	Clear Stream 1: Clears all enabled streams
0	RST_COUNT	W	0x0	Reset Sample Count 1: Resets sample count to 1

Read Register

Bit	Field	Type	Reset	Description
15:0	ADDR[15:0]	R/W	0x0	Read Register Address. Write the address of the register you want to read out on the read register stream.

Scratch Register

Bit	Field	Type	Reset	Description
15:0	WDATA	R/W	0x0	Write any 16 bit value to the scratch register

Sample Frequency

Bit	Field	Type	Reset	Description
15:0	FREQ[15:0]	R/W	0x19	Sampling Frequency register. The base frequency is 25kHz. The sampling frequency will be set to 25kHz/FREQ[15:0]

Checksum Register

Bit	Field	Type	Reset	Description
15:1	0	R/W	0x0	Reserved
0	CHECK_EN	R/W	0x0	Enable Fletcher 16 Checksum. 1: Two bytes {C1, C0} will be appended after the stop byte. 0: Normal operation

UART Rate

Bit	Field	Type	Reset	Description
15:2	0	R/W	0x0	Reserved
1:0	SET_BAUD	R/W	0x0	Change system BAUD rate. 0b00: 115200 0b01: 230400 0b10: 460800 0b11: 912600

Enable Scalar

Bit	Field	Type	Reset	Description
15:0	ENABLE[15:0]	R/W	0x0	Enable Magnetometer: 0x1F: Starts full system operation 0x00: Disables Magnetometer