Praktikum

Entwurf digitaler Systeme für FPGAs

Versuch 5

SPI mit Beschleunigungssensor

Betreuer:
Tobias Kleinfeld, Raum BB 811, 0203-379-1090, tobias.kleinfeld@uni-due.de
Pascal Raffelberg, Raum BB 818, 0203-379-2819, pascal.raffelberg@uni-due.de

Teilnehmer:

_____

Gruppen-Nr.

_____

Matr. Nr.:

_____

Versuchsdatum:
Versuchsbeginn/ende:

Bescheinigung der Teilnahme
Testat erteilt / nicht erteilt

Unterschrift: Datum:

Duisburg, den 05. Juli 2017 T. Kleinfeld

1 Einleitung

In diesem Versuch soll ein Beschleunigungssensormodul angesteuert und ausgelesen werden. Die Kommunikation erfolgt hierbei über das weit verbreitete SPI-Interface, welches hier als SPI-Controller implementiert werden soll. Das Ziel ist es somit, den Beschleunigungssensor auszulesen und die Daten dem Nutzer auf eine sinnvolle Weise anzuzeigen.

1.1 SPI

SPI steht für Serial Peripheral Interface und beschreibt ein weit verbreitetes Master-Slave-Bus-System, welches von Motorola in den 1980er Jahren entwickelt wurde. Die Aufgabe dieses Interfaces ist es, eine serielle Datenverbindung zwischen einem Hauptbaustein und einer Anzahl N peripheren Komponenten aufzubauen. Das zugehörige Aufbauschema ist in Abbildung 1 gezeigt. Zusammen mit dem I2C-Bus stellt es die gängigsten Schnittstellen für periphere Chip-Hardware (ADCs, DACs, Sensorik, etc.) dar.

Der Hauptbaustein wird **Master** genannt und darf nur einmal existieren. Die peripheren Komponenten heißen **Slaves**. Ihre Anzahl wird in der Regel nur durch die verfügbaren Chip-Select-Leitungen (CS) des Masters begrenzt. Zu beachten ist, dass der Master zu jedem Zeitpunkt die Datenhoheit über den Bus besitzt. Ein Slave kann nicht von sich aus mit seiner Umgebung interagieren.

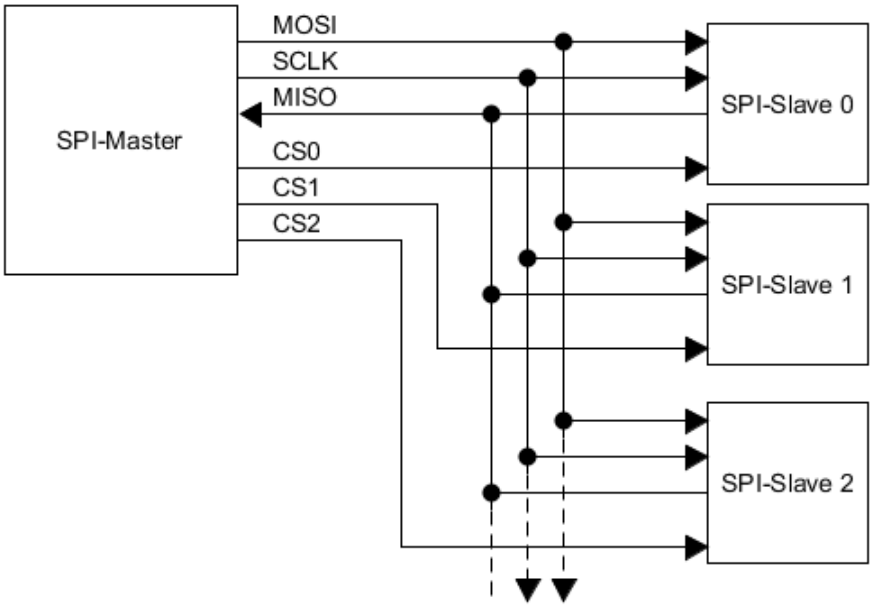

Abbildung 1: SPI im Master-Slave Aufbau

1.1.1 Steuerungs- und Datenleitungen im SPI-Bus-System

Für die normale SPI-Kommunikation werden mindestens 4-Datenleitungen gebraucht.

- SCLK: Serieller Clock vom Master zu den Slaves. Er synchronisiert das Bus-System definiert somit auch die Zeitpunkte zur Datenbereitstellung und Datenaufnahme auf dem Bus.
- MOSI: **M**aster-**O**utput-**S**lave-**I**nput => Eine serielle Datenleitung vom Master zu allen Slaves.
- MISO: **M**aster-**I**nput-**S**lave-**O**utput => Eine serielle Datenleitung vom den Slaves zum Master. Der Ausgang der Slaves ist hochohmig, wenn der Chip nicht durch das CS aktiviert ist.
- CS: Chip-Select => Jeder Slave hat seine eigene CS-Leitung und dient zur Aktivierung des Slaves. Das Signal ist oft **low-active** ausgelegt.

1.1.2 Interner SPI-Aufbau

Um den genauen Ablauf einer Übertragung zu verstehen, muss man den internen Aufbau einer SPI-Schnittstelle genauer kennen. Abbildung 2 zeigt den vereinfachten internen Aufbau von zwei miteinander in Verbindung stehenden SPI-Interfaces. Es ist zu erkennen, dass eine SPI-Instanz im Grunde aus 3 Hauptbereichen besteht.

Der erste Bereich kann als Datenbereich angesehen werden. Es besteht aus einem **8-Bit-Shiftregister**. Seine Ein- und Ausgänge bilden dabei die MISO und MOSI Anschlüsse des SPI. Je nach Ausführungsart werden für die Daten-Ein-/Ausgabe noch Speicher als Datenpuffer eingesetzt.

Als zweiter Bereich ist der Takt-Block zu nennen. Der Master stellt den SPI-Takt mittels

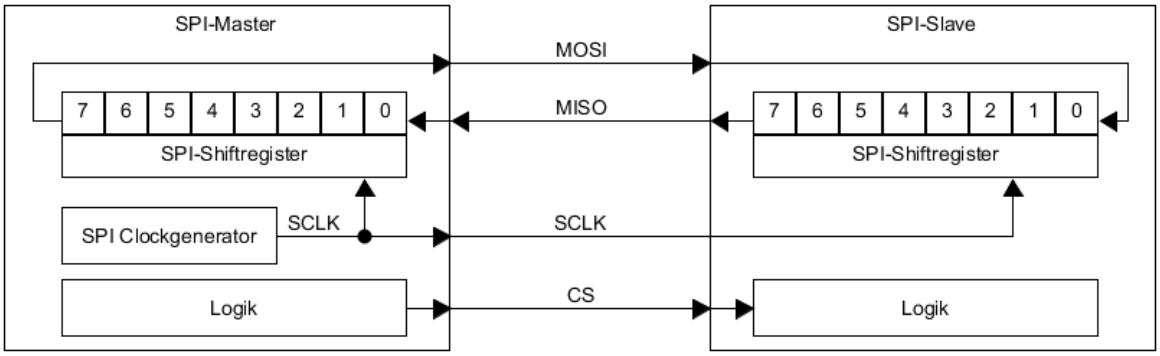

Abbildung 2: Interner SPI-Aufbau.

Generator am SCLK-Anschluss für die Slaves bereit. Durch diesen wird der Datenfluss durch dem Master kontrolliert und die Synchronität im Bus-System erreicht. Über die **Phasen-** und **Polaritäts-Parameter** des SCLK können darüber hinaus der Logikpegel während des Ruhezustands und der Zeitpunkt (steigende oder fallende Flanke) der Datenübernahme im Master konfiguriert werden. Diese Konfiguration wird in der Regel durch den jeweiligen Slave-Baustein bestimmt.
Als Besonderheit des SCLK ist weiter zu beachten, dass dieser keinen kontinuierlichen Taktgenerator im Sinne eines Systemtakts darstellt. Der Takt muss vom Modul **aktiv** generiert werden und darf **nicht** durch **„Clock-Gating“** erzeugt werden. Durch diese Maßnahme können Rippel auf der SCLK-Leitung deutlich vermieden werden.

Den dritten Bereich bildet die Aktivierungslogik für die CS-Leitungen. Im Ruhezustand werden alle Aktionen auf dem Bus vom Slave ignoriert. Nur wenn ein Slave aktiviert ist, tritt er in eine Interaktion mit dem Master. Im SPI-Master kann grundsätzlich jeder physische GPIO-Pin als CS-Leitungen dienen. Er unterliegt somit der vollen Nutzer-Kontrolle. Im SPI-Slave hingegen ist die CS-Logik als fester Bestandteil im Modul eingebettet. Dies ist notwendig, da sie die übrige Hardware direkt kontrolliert.

1.1.3 SPI-Übertragungsprotokoll

Eine Datenübertragung unterliegt immer der vollen Kontrolle des Masters. Vor dem Beginn einer Übertragung wählt der Master einen Slave aus, indem er deren Chip-Select-Leitung aktiviert. Anschließend können Daten-Blöcke über MISO und MOSI übertragen werden. Ein Block beinhaltet dabei immer genau 8-Bit und stellt somit den Inhalt im Shiftregister dar. Sobald eine Übertragung gestartet wurde, wird der aktuelle Inhalt beider Shiftregister, Bit für Bit, herausgeshiftet. Nach genau 8 Takten befindet sich der Inhalt des Master-Registers im Slave-Register und der Inhalt des Slave-Registers im Master-Register. Somit stellt jede Übertragung immer ein gleichzeitiges Senden und Empfangen dar!
Nach einer vollen Übertragung von 8-Bit kann der Master nun entweder weitere Daten sende oder aber die aktive Verbindung durch das zurücksetzten der CS-Leitung beenden. Eine **Pause** in der Übertragung kann dabei jederzeit durch das Anhalten des SCLK erreicht werden.

1.2 Der Beschleunigungssensor ADXL345

Beim ADXL345 handelt es sich um einen 3-Achsen Beschleunigungssensor der Firma Analog Devices. Der Chip besitzt einen einstellbaren Messbereich mit Werten von ±2g mit 10-Bit-Auflösung bis hin zu ±16g mit 13-Bit-Auflösung. Die Konfiguration und das anschließende Auslesen erfolgt über SPI. Die maximale Taktfrequenz für das SPI-Interface beträgt 5 MHz.

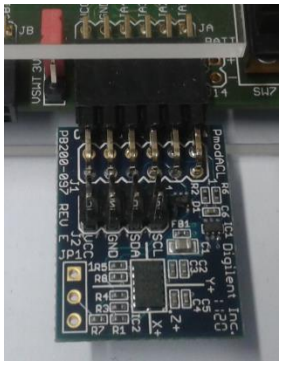

Abbildung 3: Beschleunigungssensor-Modul

Der Chip selber ist bereits als fertig als Modul auf einem PCB aufgebaut. Dieser kann direkt an einem der PMOD-Stecker des FPGA angeschlossen werden. Abbildung 3 zeigt das Modul im angeschlossenen Zustand.

Das Lesen und Schreiben auf den Chip erfolgt über ein einfaches Protokoll und ist Adressenbasiert. Auf diese Weise können Konfigurationswerte nach Auswahl in die jeweiligen Register geschrieben oder aus diesen gelesen werden. Messwerte setzten sich hierbei aus 2x8-Bit Registerwerten zusammen und liegen als **2er-Komplement** vor. Eine genaue Auflistung der Registeradressierung finden Sie im ADXL345-Datenblatt auf Seite 23. Aus diesem Aufbau (Adressierung und Registerwerte) ergibt sich, dass eine vollständige Übertragung somit immer aus mindestens 16-Bit besteht. Die Timings des Protokolls sind für den Lesezugriff in Abbildung 5 und für den Schreibzugriff in Abbildung 4 angegeben.

Der ADXL345 stellt dem Nutzer eine Reihe an Betriebsmodi bereit. Der einfachste Modus ist hierbei das kontinuierliche Messen ohne FIFO (fifo bypassed). In diesem nimmt der Chip automatisch in regelmäßigen Abständen die Werte für die 3 Achsen auf und aktualisiert die zugehörigen Registerwerte. Ein Pufferung findet nicht statt. Dies führt dazu, dass alle alten

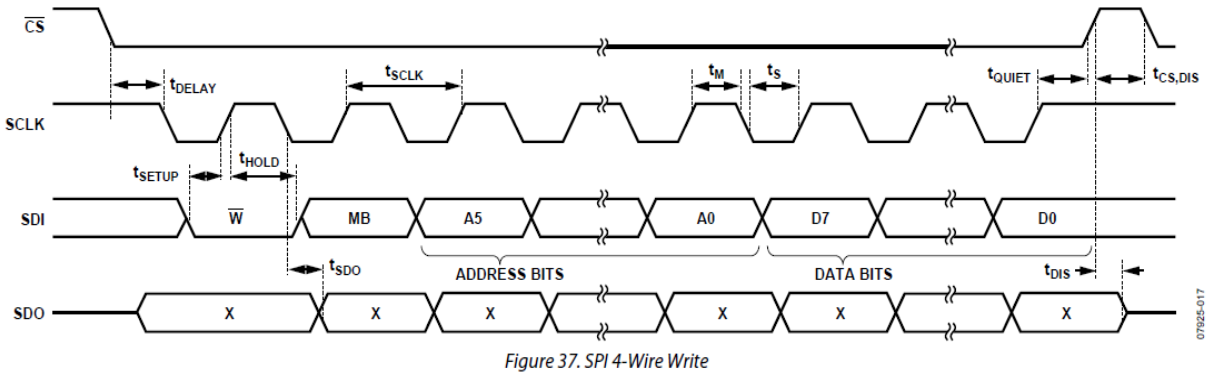

Figure 37. SPI 4-Wire Write

Abbildung 4: SPI Schreibzugriff

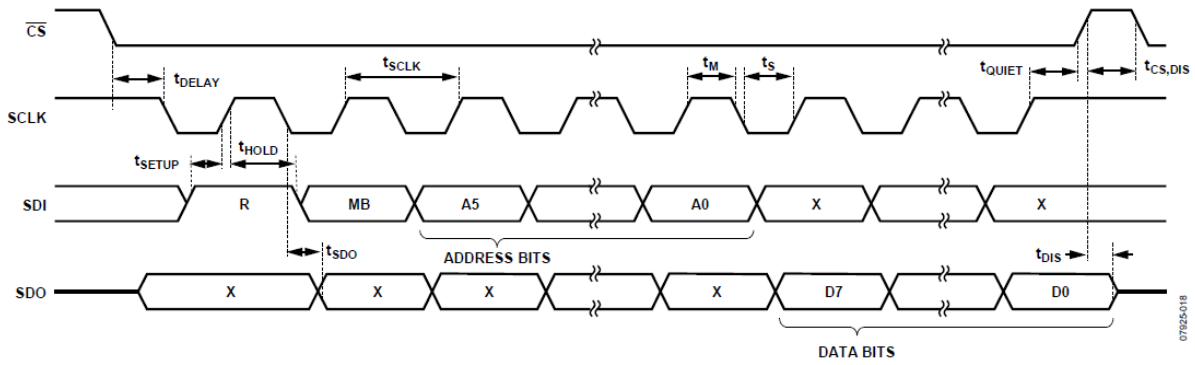

Figure 38. SPI 4-Wire Read

Abbildung 5: SPI Lesezugriff

Werte direkt überschrieben werden.

2 Modulbeschreibung

Das Verilog ADXL345-Modul soll ein einfaches Interface zum Auslesen des Beschleunigungssensors implementieren. Die Konfiguration für den ADXL345 und die anschließende wiederkehrende Auslese des Sensors soll dabei fest in die Hardware implementiert werden. Das Topmodul einer möglichen Modulimplementierung ist als Blockschaltbild in Abbildung 6 angegeben. Es beinhaltet die zwei Hauptblöcke SPI_16BitInterface und den ADXL345Controller.

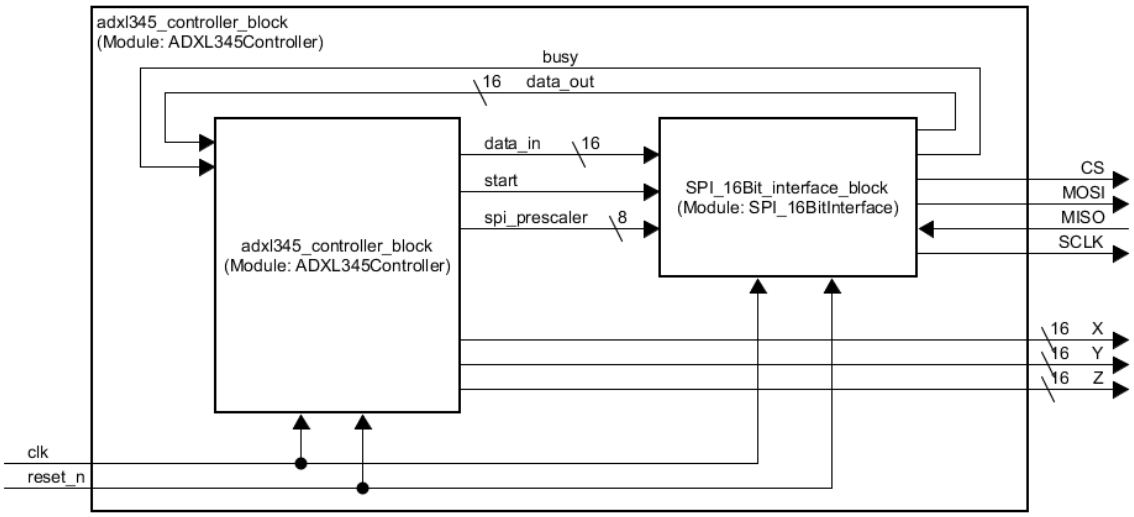

Abbildung 6: Blockdiagramm des ADXL345-Hauptmodul

2.1 SPI_16BitInterface

Aus der SPI-Protokollbeschreibung in Abschnitt 1.2 ist bekannt, dass eine Übertragung zum Chip immer aus mindestens 16-Bit bestehen muss. Das SPI_16BitInterface implementiert dieses Verhalten. Es bildet dabei einen Mapper um eine herkömmliche SPI-Schnittstelle und sorgt dafür, dass immer genau 16-Bit an Daten in einem Chip-Select-Fenster übertragen und empfangen werden. Dieses beschriebene Verhalten ist ausreichend, um sämtliche Funktionen des ADXL345 nutzen zu können. Der Aufbau des 16-Bit-Mappers ist in Abbildung 7 gezeigt.

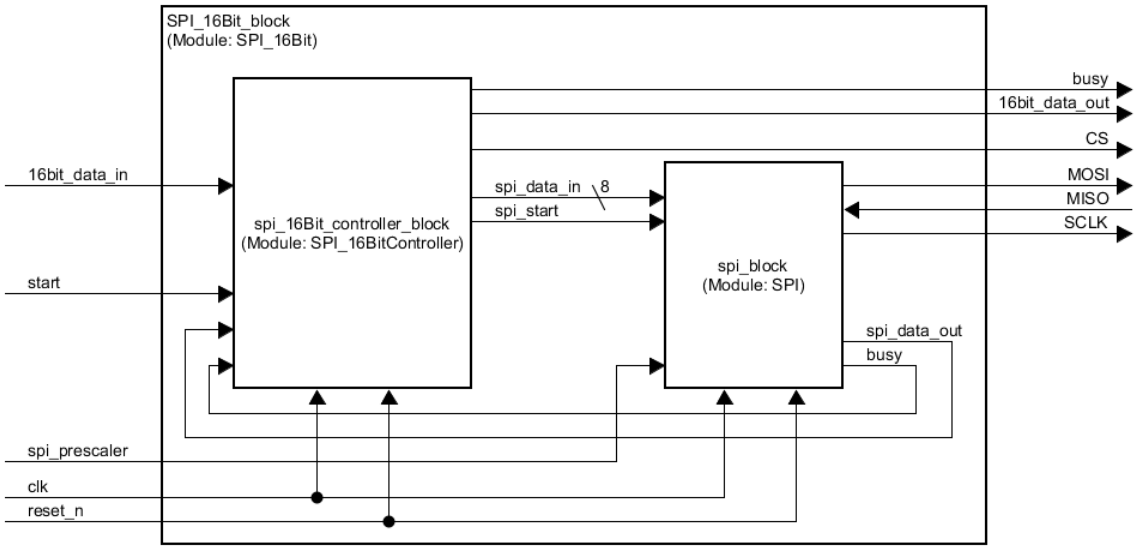

Abbildung 7: Blockschaltbild des SPI_16BitInterfaces

2.2 ADXL345Controller

Der ADXL2345Controller ist für Befehlsansteuerung des Beschleunigungssensors zuständig. Dieses beinhaltet die Initialisierung des Sensors und das anschließende wiederholende Auslesen der 3 Achsen. Für diese Abfolge müssen zusammengesetzte 16-Bit-Befehle an den Sensor gesendet werden. Dies kann dadurch erreicht werden, dass der ADXL345Controller das SPI_16BitInterface ansteuert und die fertigen Befehle über diesen sendet. Das abzubildende Verhalten kann dabei wieder gut als State-Maschine implementiert werden. Es umfasst dabei die Bereiche: einmalige Chip-Initialisierung, Starten der Sensormessung im Chip, Auslesen der Messwerte und Messwertspeicherung der 3 Achsen.

3 Vorbereitungsaufgaben

Bitte beantworten Sie die nachfolgenden Fragen. Sie dienen Ihnen als Grundlage für die Durchführung des Versuchs.

3.1 Allgemeine Fragen zur SPI und zum Sensor

1. Was versteht man unter der Polarität und der Phase bzgl. des SPI-Taktes? Welcher Teilnehmer (Master oder Slave) bestimmt die Polarität und Phase? Wie sind diese beim ADXL345 definiert?

2. Wie wird dem SPI-Slave signalisiert, ob es sich um einen Schreib- oder Lesezugriff handelt?

3. Welche Funktion hat das „MB“-Bit aus dem Timing-Diagramm?

4. Im SPI-Modul wird ein Shiftregister eingesetzt. Geben Sie den Verilog-Code einer Always-Struktur an, um ein 8Bit-Shiftregister zu implementieren.

3.2 Implementierung des ADXL345Controllers

Der ADXL345 soll im einfachen, kontinuierlichen Modus für den Messbereich ±2g betrieben werden. Ein FIFO ist nicht erforderlich.

5. Finden Sie heraus, welche Register des ADXL345 mit Inhalten beschrieben werden müssen, um ihn im gewünschten Modus betreiben zu können. Die zugehörigen Werte entnehmen Sie bitte dem Datenblatt und den Quick-Start Guide.

   Hinweis: Das „Justify-Bit“ im „Data-Format-Register“ muss zu „Null“ gesetzt werden, sodass die Daten wie in Figure 49 (ADXL-Datenblatt) angezeigt rechtsbündig anliegen.[1]

Der ADXL345Controller soll als Zustandsautomat umgesetzt werden. Dieser Zustandsautomat initialisiert den angeschlossenen Slaves einmalig in einem Initialisierungsteil und liest anschließen die Achsen kontinuierlich aus. Die Ergebnisse sollen anschließend in einem Speicher abgelegt werden.

[1] Die „Left-Justified“-Funktion liegt aufgrund eines internen Chipfehlers nicht vor.

6. Geben Sie die Abfolge des Initialisierungs- und Hauptteils in Form eines Flussdiagramms an.

7. Erstellen Sie aus dem Flussdiagramm ein State-Diagramm.

3.3 Messwertverarbeitung und Anzeige

Die erfassten Messwerte sollen dem Nutzer nun sinnvoll angezeigt werden. Hierfür stehen dem Entwickler mittlerweile verschiede Systeme zur Verfügung: LED-Bargraph, LCD-Display, UART-Verbindung.

8. Entwickeln Sie für sich ein Konzept zur Anzeige der Messwerte. Nehmen Sie im Zweifelsfall die LED-Bargraph-Anzeige.

9. Die Daten liegen als 2er Komplement vor. Geben Sie den Verilog-Code an, mit welchem die Umrechnung der Daten in eine einfache vorzeichenbehaftetet Zahl erfolgen kann.

4 Versuchsdurchführung

1. Erzeugen Sie ein neues Projekt ADXL345.

4.1 SPI-Implementierung

2. Implementieren Sie ein allgemeines SPI-Modul. Testen Sie dieses anschließend. Beachten Sie bitte auch die Polarität und die Phase des SCLK.

3. Implementieren Sie das SPI-16BitInterface. Testen Sie das Senden und den Empfang. Achten Sie auch auf die CS-Leitung.

4.2 ADXL345Controller

4. Implementieren Sie die State-Maschine des ADXl345Controllers. Testen Sie die Initialisierungssequenz und das Auslesen der Messwerte. Achten Sie vor allem auf eine gute Interaktion zwischen dem SPI-Modul und dem ADXL345Controller.

4.3 Anzeige der Messwerte

5. Implementieren Sie Ihr gewähltes Modul zur Messwertanzeige. Testen Sie dieses anschließend.