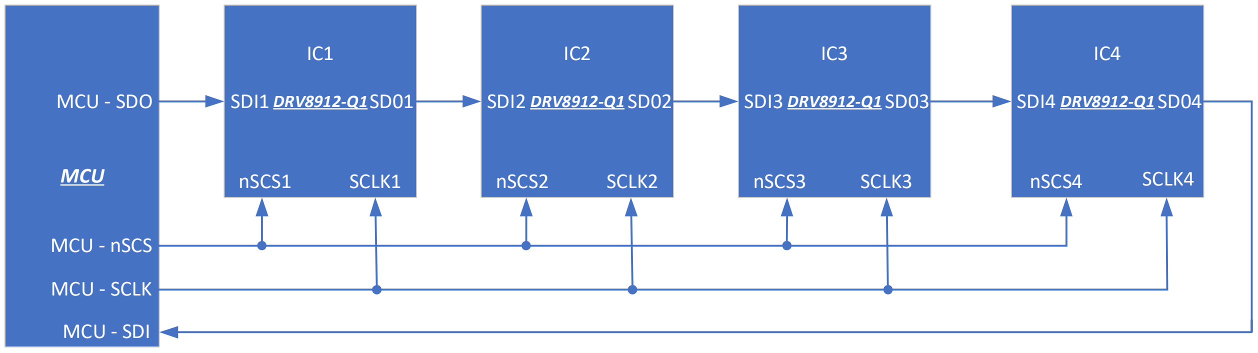
07/31/2024

Daisy Chain SPI with Motor Drivers

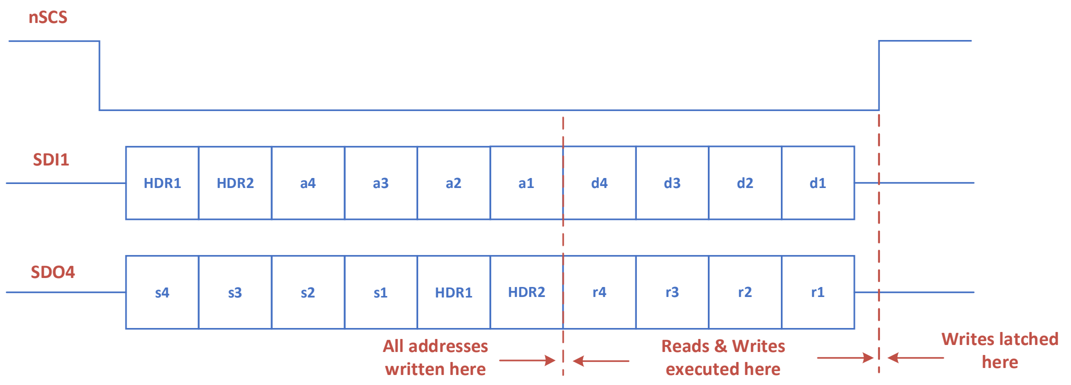
Example hardware implementation N = 4

- Four DRV8912-Q1 devices connected in daisy chain mode for SPI communication with MCU.
- Reuse this example for all motor driver devices supporting this specific SPI daisy chain protocol.
- Reduces the number of GPIO control required by N-1 pins. Only one nSCS control is needed.
- Register read and writes from and to all the devices in the daisy chain supported.
- All N devices are latched with written data to registers simultaneously at nSCS rising edge.

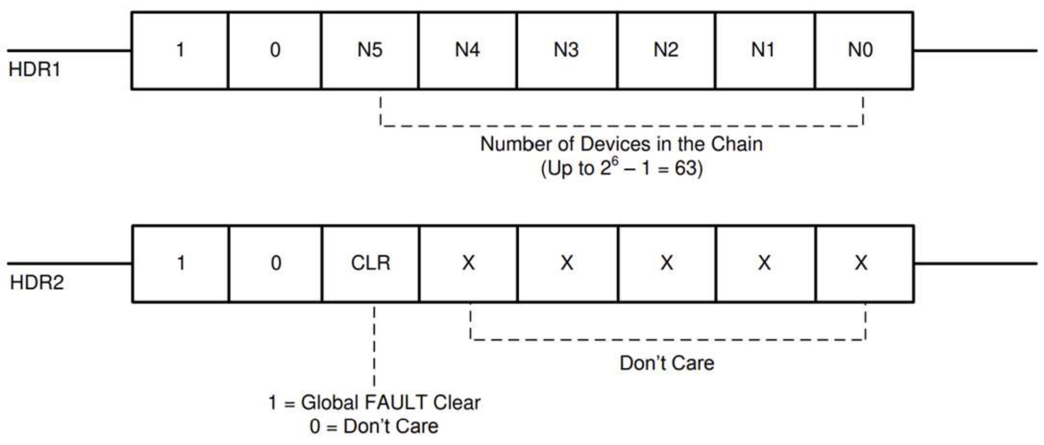
Block Diagram



SPI Data Bytes Model

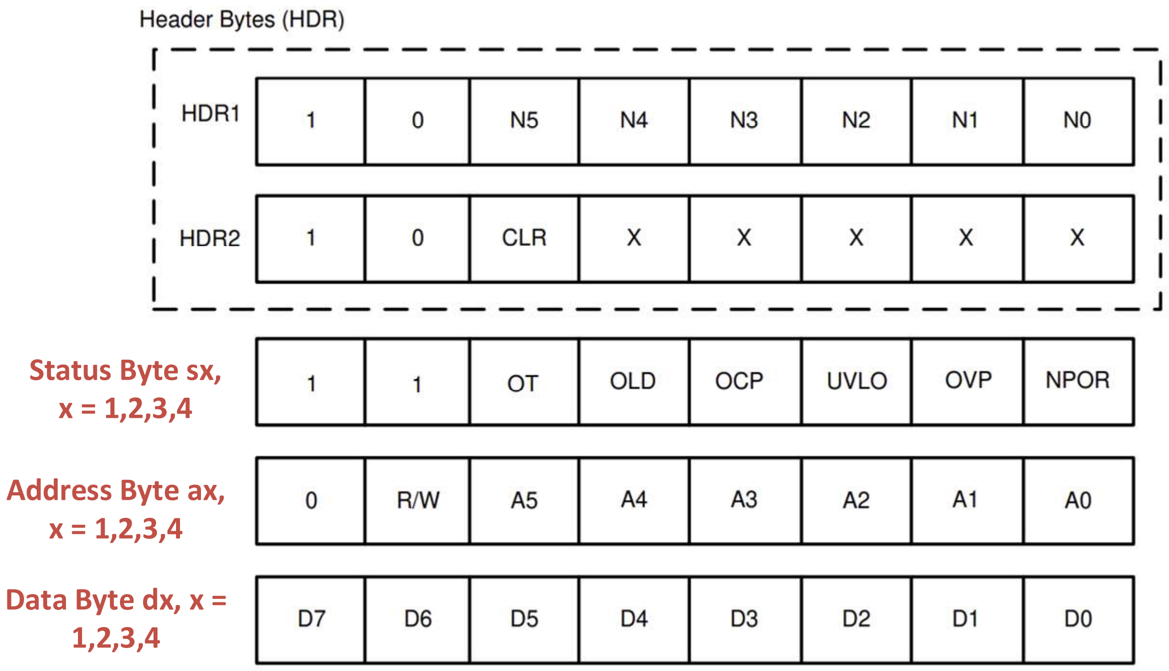

nSCS
SDI1
HDR1 HDR2 a4 a3 a2 a1 d4 d3 d2 d1
SDO4
s4 s3 s2 s1 HDR1 HDR2 r4 r3 r2 r1
All addresses written here
Reads & Writes executed here
Writes latched here

Setting up Header Bytes



HDR1 = 0b10000100 = 0x84, for four devices
HDR2 = 0b10000000 = 0x80, assuming global fault clear is 0

Read / Write Bytes Organization in the Daisy Chain



Daisy Chain Read/Write Registers

SPI Controller Transactions Summary

The SDI1 of the first device in the chain receives data from the SPI controller in the following format for all the four devices in the daisy chain.

- 2 bytes of Header; HDR1 and HDR2
- 4 bytes of Address; a4, a3, a2 and a1
- 4 bytes of Data; d4, d3, d2 and d1

While the data is being transmitted through the chain of four devices, the SPI controller receives data from SDO3 in the following format.

- 4 bytes of Status; s4, s3, s2 and s1
- 2 bytes of Header; this will be identical to the Header HDR1 and HDR2 sent by the controller
- 4 bytes of Report; r4, r3, r2 and r1, these are the returned read value bytes from the device

Reading a register from all the four devices in the daisy chain

To initiate a read the R/W bit in the address field must be a logic level 1. This can be realized by logical -OR- with the address of the register that will be read from. Let's consider an example transaction that will read the OCP_STAT_1 registers from all the four devices in the daisy chain.

Pseudo code

```
/* SPI initialized in the controller per "Programming" section in the DRV8912-Q1 datasheet */
#define HDR1 0x84                               // define HDR1 value
#define HDR2 0x80                               // define HDR2 value
#define Dummy_value 0x00

nSCS = 1;                                       // nSCS = logic HIGH at device initialization

/* variables declaration */
volatile uint8     SPI_status_s4;               // status byte s4
volatile uint8     SPI_status_s3;               // status byte s3
volatile uint8     SPI_status_s2;               // status byte s2
volatile uint8     SPI_status_s1;               // status byte s1

volatile uint8     SPI_address_a4;              // address byte a4
volatile uint8     SPI_address_a3;              // address byte a3
volatile uint8     SPI_address_a2;              // address byte a2
volatile uint8     SPI_address_a1;              // address byte a1

volatile uint8     SPI_data_d4;                 // data byte d4
volatile uint8     SPI_data_d3;                 // data byte d4
volatile uint8     SPI_data_d2;                 // data byte d4
volatile uint8     SPI_data_d1;                 // data byte d4
volatile uint8      SPI_HDR1;                          // returned HDR1 byte
volatile uint8      SPI_HDR2;                          // returned HDR2 byte

SPI_address_a4 = 0x4000 || OCP_STAT_1;// set R/W bit to 1 by logical OR with register address
SPI_address_a3 = 0x4000 || OCP_STAT_1;// set R/W bit to 1 by logical OR with register address
SPI_address_a2 = 0x4000 || OCP_STAT_1;// set R/W bit to 1 by logical OR with register address
SPI_address_a1 = 0x4000 || OCP_STAT_1;// set R/W bit to 1 by logical OR with register address

/* SPI transactions start here */
nSCS = 0;                                               // this enables the DRV8912-Q1 devices for SPI transactions

SPI transmit ( HDR1);                           // transmit HDR1 0x84
SPI_status_s4 = SPI receive;                    // receive status byte s4

SPI transmit (HDR2);                            // transmit HDR2 0x80
SPI_status_s3 = SPI receive;                    // receive status byte s3

SPI transmit (SPI_address_a4);                  // transmit  address a4
SPI_status_s2 = SPI receive;                    // receive status byte s2

SPI transmit (SPI_address_a3);                  // transmit  address a3
SPI_status_s1 = SPI receive;                    // receive status byte s1

SPI transmit (SPI_address_a2);                  // transmit  address a2
SPI_HDR1 = SPI receive;                         // receive HDR1 byte
/* optional */
If (SPI_HDR1 != HDR1)
     Return (error);

SPI transmit (SPI_address_a1);                  // transmit  address a1
SPI_HDR2 = SPI receive;                         // receive HDR2 byte
/* optional */
If (SPI_HDR2 != HDR2)
     Return (error);

SPI transmit (Dummy_value);                     // transmit  a dummy value for d4 to receive register content r4
SPI_data_d4 = SPI receive;                      // receive r4 value

SPI transmit (Dummy_value);                     // transmit  a dummy value for d4 to receive register content r3
SPI_data_d3 = SPI receive;                      // receive r3 value

SPI transmit (Dummy_value);                     // transmit  a dummy value for d4 to receive register content r2
SPI_data_d2 = SPI receive;                      // receive r2 value

SPI transmit (Dummy_value);                     // transmit  a dummy value for d4 to receive register content r1
SPI_data_d1 = SPI receive;                      // receive r1 value

nSCS = 1;                                               // this completes the daisy chain SPI register read transactions
```

Writing a register to all the four devices in the daisy chain

To initiate a write the R/W bit in the address field must be a logic level 0. This can be realized by transmitting just the address of the register that will be written to. Let's consider an example transaction that will write to the CONFIG_CTRL registers in all the four devices in the daisy chain.

Pseudo code

```
/* SPI initialized in the controller per "Programming" section in the DRV8912-Q1 datasheet */
#define HDR1 0x84                              // define HDR1 value
#define HDR2 0x80                              // define HDR2 value
#define Register_value 0x0F                    // define OCP_REP = 1, OTW_REP = 1, EXT_OVP = 1, and
                                               // CLR_FLT = 1, Reserved = 0, IC_ID = read only

nSCS = 1;                                      // nSCS = logic HIGH at device initialization

/* variables declaration */
volatile uint8     SPI_status_s4;              // status byte s4
volatile uint8     SPI_status_s3;              // status byte s3
volatile uint8     SPI_status_s2;              // status byte s2
volatile uint8     SPI_status_s1;              // status byte s1

volatile uint8     SPI_address_a4;             // address byte a4
volatile uint8     SPI_address_a3;             // address byte a3
volatile uint8     SPI_address_a2;             // address byte a2
volatile uint8     SPI_address_a1;             // address byte a1

volatile uint8     SPI_data_d4;                // data byte d4
volatile uint8     SPI_data_d3;                // data byte d4
volatile uint8     SPI_data_d2;                // data byte d4
volatile uint8     SPI_data_d1;                // data byte d4

volatile uint8     SPI_HDR1;                   // returned HDR1 byte
volatile uint8     SPI_HDR2;                   // returned HDR2 byte

SPI_address_a4 = CONFIG_CTRL;                  //  R/W bit is 0
SPI_address_a3 = CONFIG_CTRL;                  // R/W bit is 0
SPI_address_a2 = CONFIG_CTRL;                  // R/W bit is 0
SPI_address_a1 = CONFIG_CTRL;                  // R/W bit is 0

/* SPI transactions start here */
nSCS = 0;                                      // this enables the DRV8912-Q1 devices for SPI transactions

SPI transmit ( HDR1);                          // transmit HDR1 0x84
SPI_status_s4 = SPI receive;                   // receive status byte s4

SPI transmit (HDR2);                           // transmit HDR2 0x80
SPI_status_s3 = SPI receive;                   // receive status byte s3
```

```
SPI transmit (SPI_address_a4);            // transmit  address a4
SPI_status_s2 = SPI receive;              // receive status byte s2

SPI transmit (SPI_address_a3);            // transmit  address a3
SPI_status_s1 = SPI receive;              // receive status byte s1

SPI transmit (SPI_address_a2);            // transmit  address a2
SPI_HDR1 = SPI receive;                   // receive HDR1 byte
/* optional */
If (SPI_HDR1 != HDR1)
    Return (error);

SPI transmit (SPI_address_a1);            // transmit  address a1
SPI_HDR2 = SPI receive;                   // receive HDR2 byte
/* optional */
If (SPI_HDR2 != HDR2)
    Return (error);

SPI transmit (Register_value);            // transmit  desired register value for register r4
SPI_data_d4 = SPI receive;                // receive r4, value of register prior to this write operation

SPI transmit (Register_value);            // transmit  desired register value for register r3
SPI_data_d3 = SPI receive;                // receive r3, value of register prior to this write operation

SPI transmit (Register_value);            // transmit  desired register value for register r2
SPI_data_d2 = SPI receive;                // receive r2, value of register prior to this write operation

SPI transmit (Register_value);            // transmit  desired register value for register r1
SPI_data_d1 = SPI receive;                // receive r1, value of register prior to this write operation

nSCS = 1;                                 // this completes the daisy chain SPI register write transactions
```

Summary

The above examples show a single daisy chain transaction each of a read and write operation. It is possible to mix read and write operations within a single daisy chain transaction. To achieve this the R/W bit in the address fields of the desired register of a target device in the daisy chain must be configured accordingly. R/W = 1 will be a read operation and R/W = 0 will be a write operation for that specific register, in the specific location of the daisy chain.

Please review the application note, https://www.ti.com/lit/an/slvae25a/slvae25a.pdf ***Daisy Chain Implementation for Serial Peripheral Interface***.