INTRODUCTION TO ALLEGRO SPI

By Kade Hudson, Field Applications Engineer
Allegro MicroSystems

INTRODUCTION

Serial peripheral interface (SPI) is a 4-wire, full duplex digital protocol. Full duplex means that data is transmitted and received simultaneously. This generally enables more information to be shared between devices compared to half-duplex communication protocols where data can only be transmitted or received.

SPI OVERVIEW

In its simplest form, SPI consists of one controller and one peripheral device (i.e., point to point). That is not to say that multiple peripherals cannot be used with one controller. There are ways of implementing what is called shared SPI (i.e., star configuration) or daisy chain SPI (i.e., ring configuration). For simplicity, this document will focus on one controller and one peripheral.

The SPI protocol implements four logic signals:

1. CS (chip select) output from the controller.
2. SCLK (serial clock) output from the controller.
3. MOSI (controller-out peripheral-in) output from the controller.
4. MISO (controller-in peripheral-out) output from the peripheral.

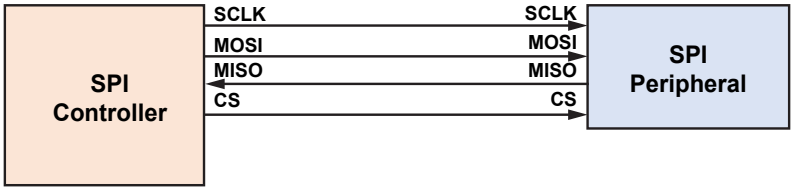

Figure 1: SPI Overview

DATA TRANSMISSION

Starting with the controller device, communication is enabled by pulling CS out of the idle state. This will either be a logic high or logic low signal, depending on what is specified by the peripheral device (detailed in the next section). When the CS line is idle, no communication occurs and, therefore, no data is transmitted or received from either device. Once pulled out of idle state, the controller begins issuing clock pulses and data over the MOSI line. Simultaneously, the peripheral device transmits data over the MISO line. When complete, the controller stops toggling the clock signal and returns the CS back to the idle state, thereby completing what is known as the message frame or payload.

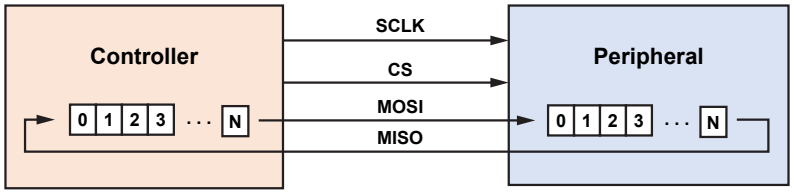

Figure 2: SPI Data Transfer

CLOCK FREQUENCY, POLARITY, PHASE, AND MORE

SPI is a flexible protocol—the controller configures the clock frequency (up to 10 MHz), polarity, and phase necessary for the peripheral device. By convention, polarity and phase are named CPOL and CPHA respectively. CPOL controls the polarity of the clock, and when CPOL = 0, the clock begins toggling from a logic low signal. When CPOL = 1, the clock begins toggling from a logic high signal.

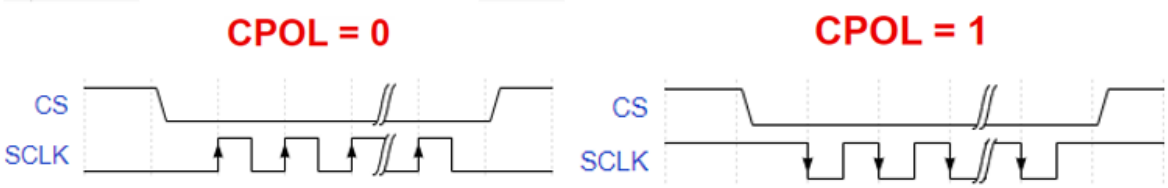

Figure 3: Clock Parity

CPHA controls the phase and is more complex as it is dependent on the value of CPOL. The combinations of polarity and phases are often referred to as modes, which are commonly numbered according to the convention shown in Table 1.

Table 1: SPI Modes

Mode	CPOL	CPHA	Description
0	0	0	Clock begins from a logic low state. Data is shifted out on the falling edge of the clock and will/should be sampled on the rising.
1	0	1	Clock begins from a logic low state. Data is shifted out on the rising edge of the clock and will/should be sampled on the falling.
2	1	0	Clock begins from a logic high state. Data is shifted out on the rising edge of the clock and will/should be sampled on the falling.
3	1	1	Clock begins from a logic high state. Data is shifted out on the falling edge of the clock and will/should be sampled on the rising.

SPI MODE 0

For SPI mode 0, CPOL and CPHA are 0. The clock begins toggling from a logic low state. Except for the MSB, data from the register is shifted out on the falling edge of the clock signal. The MSB is shifted out immediately when the CS line is brought low. Data is sampled by the receiving device on the rising edge of the clock. The blue line shows when the data is sampled, and the red line shows when the data is output.

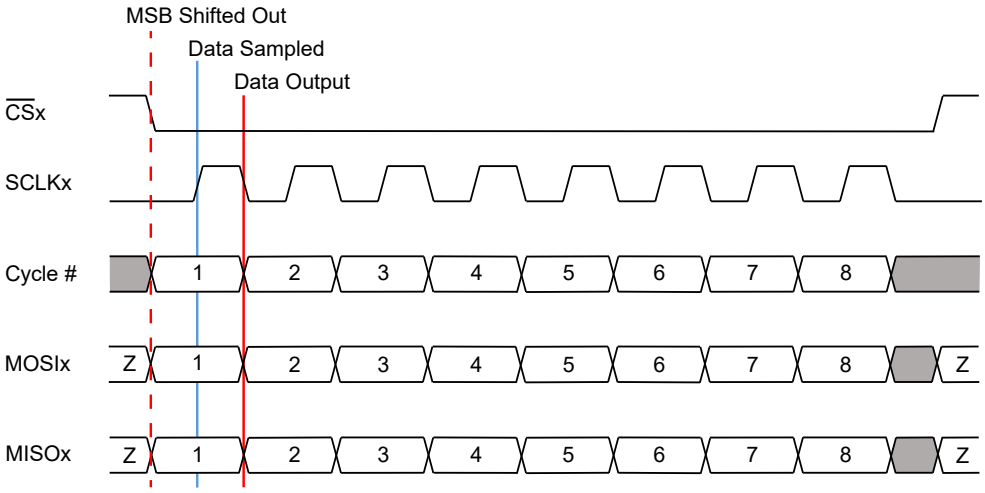

Figure 4: SPI Mode 0, CPOL = 0, CPHA = 0

SPI MODE 1

For SPI mode 1, CPOL is 0 and CPHA is 1. The clock begins toggling from a logic low state. Data from the register is shifted out on the rising edge of the clock signal and sampled by the receiving device on the falling edge. The blue line shows when the data is sampled, and the red line shows when the data is output.

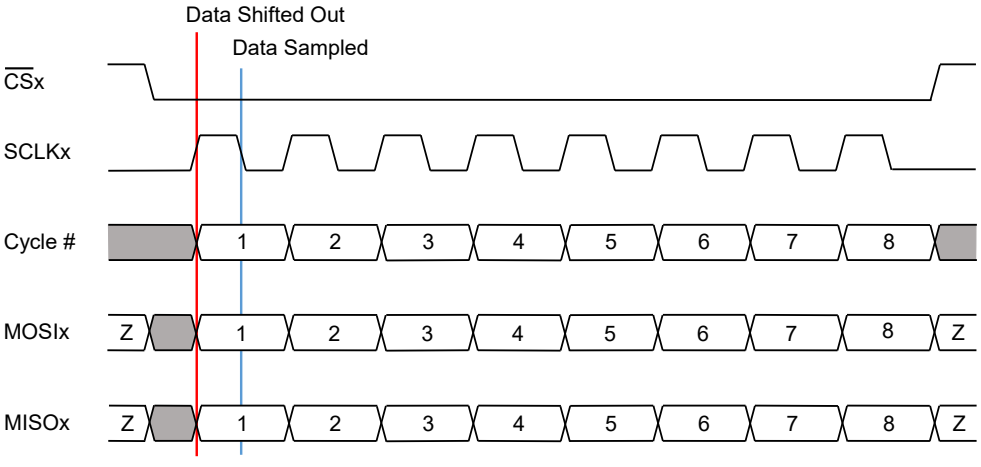

Figure 5: SPI Mode 1, CPOL = 0, CPHA = 1

SPI MODE 2

For SPI Mode 2, CPOL is 1 and CPHA is 0. The clock begins toggling from a logic high state. With the exception of the MSB, data from the register is shifted out on the rising edge of the clock signal. The MSB is shifted out immediately when the CS line is brought low. Data is sampled by the receiving device on the falling edge of the clock. The blue line shows when the data is sampled, and the red line shows when the data is output.

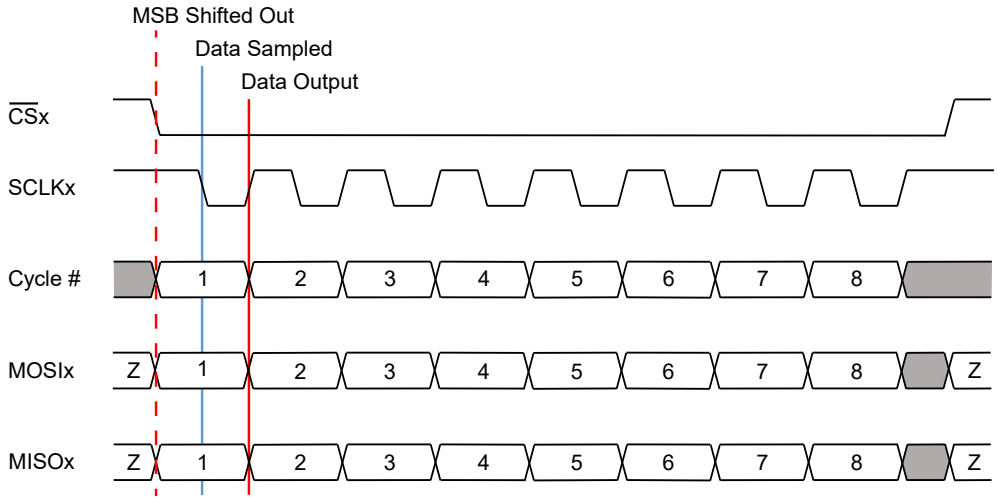

Figure 6: SPI Mode 2, CPOL = 1, CPHA = 0

SPI MODE 3

For SPI Mode 3, CPOL and CPHA are 1. The clock begins toggling from a logic low state. Data from the register is shifted out on the falling edge of the clock signal and sampled by the receiving device on the rising edge. The blue line shows when the data is sampled, and the red line shows when the data is output.

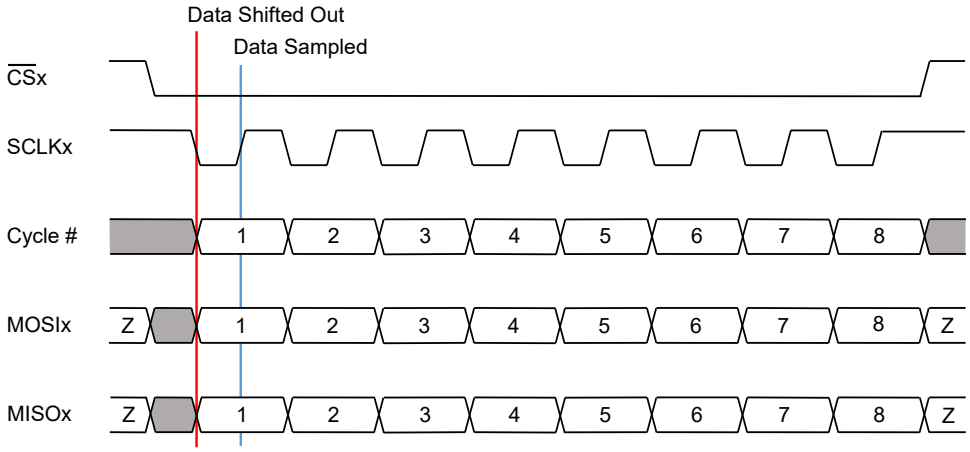

Figure 7: SPI Mode 3, CPOL = 1, CPHA = 1

Most, if not all, Allegro devices use SPI mode 3 (CPOL = 1 and CPHA = 1).

SPI IMPLEMENTATION

Apart from what has been described so far, there really is no standard for SPI, whether for the size of the payload, how it is structured, or when it is received after a request has been issued.

Typically, in a SPI frame structure, there are three major components: a header (e.g., bits containing an address), the data, and a footer containing some form of an acknowledgement (e.g., parity or cyclic redundancy check).

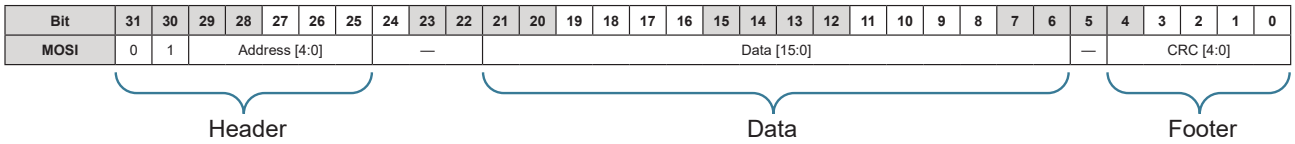

Figure 8: Typical SPI Frame Structure

Out-Of-Frame

The most straightforward (and likely the most common) type of data transfer is out-of-frame SPI communication, where data from the peripheral is received one frame after the controller requests it. For out-of-frame SPI communication, the controller requests information but does not receive a response until the next payload is fully received. The start of the frame from the peripheral for this type of SPI transmission can vary, but typically contains a register address and sometimes a frame counter. This is followed by data and, if a footer exists, either a parity bit or cyclic redundancy check (CRC).

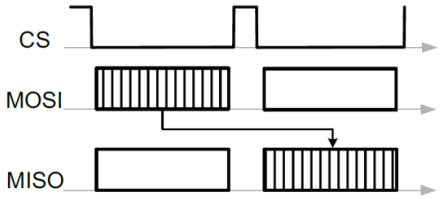

Figure 9: Out-of-Frame SPI Communication

In-Frame

With in-frame SPI, data received from the peripheral happens within the same controller data request payload. In Allegro's case, the start of the frame bits from the peripheral are typically a fixed type of diagnostic or status indicator, followed by the data requested and, in some cases, a footer containing a parity bit checker or CRC.

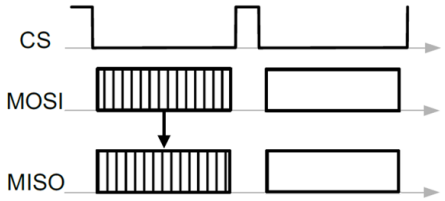

Figure 10: In-Frame SPI Communication

APPLICATION EXAMPLES

To give examples of how in-frame and out-of-frame SPI works, here is a brief overview from two Allegro devices: the A4412 and the A33115. Each is different in payload size, structure, and when/how data is received from the peripheral device.

A4412

The A4412 is a power management IC and implements in-frame SPI. The payload transmitted from the controller (MOSI) contains 5 address bits (A4:A0), a read/write bit (W/R), 8 data bits (D7:D0), and a parity bit. The P in the LSB position indicates parity for the ending frame. Allegro devices use an odd bit parity checker; that is, when all the 1's in the transaction are counted and the total is odd in number, then the parity is odd. Conversely, if all of the 1's are an even number, the parity is even, and P must be set to 1.

MOSI Read/Write Request Frame

MSB															LSB
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A4	A3	A2	A1	A0	W/R	NU	D7	D6	D5	D4	D3	D2	D1	D0	P
5-Bit Address							8-Bit Data								

Figure 11: A4412 SPI Implementation MOSI

The peripheral response (MISO) changes depending on whether the W/R bit was set in the MOSI frame. If the W/R bit is set to 1, the data bits within the MISO frame contain the contents from a high-level diagnostic register. If the W/R bit is set to 0, the data bits return the data contents from the address that was specified in the MOSI frame. Irrespective of the W/R bit being set, the first 7 bits of the payload (bits 15 through 9) always contain the same diagnostic bits.

MISO Response Following a Write Request

MSB															LSB
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FF	SE	ENBATS	WD_F	TSD_OK	VREG_OK	BUCK_OK	VCC_OK	VCP_OK	V5P_OK	V5B_OK	V5A_OK	V5CAN_ OK	3V3_OK	0	P
Diagnostics															

MISO Response Following a Read Request

MSB															LSB
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FF	SE	ENBATS	WD_F	TSD_OK	VREG_OK	BUCK_OK	D7	D6	D5	D4	D3	D2	D1	D0	P
Diagnostics							8-Bit Data								

Figure 12: A4412 SPI Implementation MISO

The table below reflects a read action to request data from the Configuration 0 register.

Register Map

HEX Address	Register Name	DEC Address	Type	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00	STATUS_0	0	RO	FF	POE_OK	VCC_OK	VDD_OK	V5P_OK	V5B_OK	V5A_OK	V5CAN_OK
0x01	STATUS_1	1	RO		NPOR_OK	WD_F	TSD_OK	VCP_OK	VREG_OK	3V3_OK	BUCK_OK
0x02	STATUS_2	2	RO	CLK_Hi	CLK_Lo	NPOR_S	POE_S	ENBATS	WD_STATE		
0x03	DIAG_0	3	RW1C	V5A_OV	V5A_UV	V5CAN_OV	V5CAN_UV	V5P_OV	V5P_UV	V5B_OV	V5B_UV
0x04	DIAG_1	4	RW1C	VDD_OV	VDD_UV	VREG_OV	VREG_UV	3V3_OV	3V3_UV	BUCK_OV	BUCK_UV
0x05	DIAG_2	5	RW1C		LX2_OK	LX1_OK	D1_OK		VCC_UV	VCP_OV	VCP_UV
0x06	OUTPUT_DISABLE	6	RW	V5P_DIS1	V5A_DIS1	V5B_DIS1	V5CAN_DIS1	V5P_DIS0	V5A_DIS0	V5B_DIS0	V5CAN_DIS0
0x07	WATCHDOG_MODE_KEY	7	WO	Keycode Entry (Write Only)							
			RO	0	0	0	0	0	0	0	Unlocked
0x08	CONFIG_0	8	RW			MAX_TIMER			MIN_TIMER		
0x09	CONFIG_1	9	RW				DITH_DIS	VALID_COUNT		EDGE_COUNT	
0x0A	VERIFY_RESULT_0	10	RW1C	V5A_OV_OK	V5A_UV_OK	V5CAN_OV_OK	V5CAN_UV_OK	V5P_OV_OK	V5P_UV_OK	V5B_OV_OK	V5B_UV_OK
0x0B	VERIFY_RESULT_1	11	RW1C	BIST_PASS	TSD_OK	VREG_OV_OK	VREG_UV_OK	3V3_OV_OK	3V3_UV_OK	BUCK_OV_OK	BUCK_UV_OK

Register Types:
RO = Read-Only
RW = Read or Write
RW1C = Read or Write 1 to clear
WO = Write-Only

Figure 13: A4412 Serial Registers

When the read request is transmitted, the MOSI outputs (in hexadecimal) 0x4000. This is requesting data from the Configuration 0 register. Within the same frame, the MISO transmits the diagnostic bits as well as the data bits from the Configuration 0 register. Referring to the A4412 datasheet and decomposing the message (0x2E49), the received data aligns perfectly with the default settings for this register (see Figure 15).

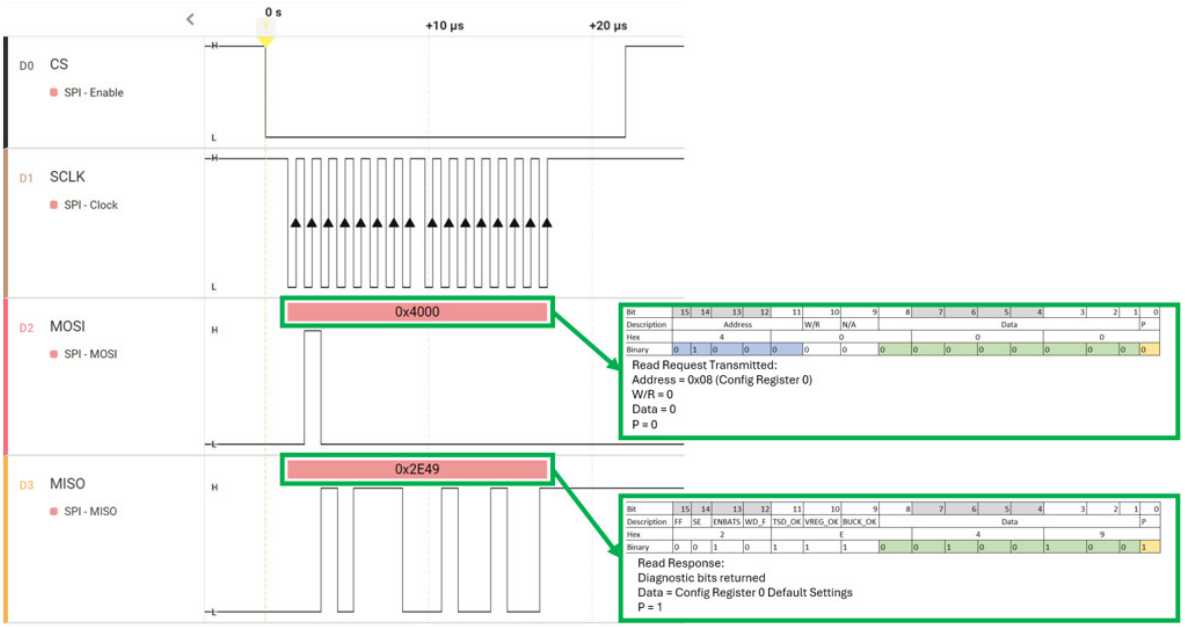

Figure 14: A4412 SPI Capture

0X08. CONFIGURATION REGISTER 0:

D7	D6	D5	D4	D3	D2	D1	D0
		WD_MAX_2	WD_MAX_1	WD_MAX_0	WD_MIN_2	WD_MIN_1	WD_MIN_0
0	0	1	0	0	1	0	0

Figure 15: A4412 Config 0 Register Default Settings

A33115

The A33115 is a position sensor and implements out-of-frame SPI. Furthermore, both read and write are differentiated in terms of transmitting and receiving. For MOSI, it has read and write requests; for MISO, it has read and write responses; and every transfer includes a 5-bit CRC. Each frame requires a CRC calculation.

MOSI Write Request Frame

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOSI	0	1	Address [4:0]					—			Data [15:0]																—	CRC [4:0]				

MOSI Read Request Frame

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOSI	0	0	Address [4:0]					—			Data [15:0] (Not Applicable)																—	CRC [4:0]				

MISO Response Following a Read Request

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MISO	1	Previous Address [4:0]					Frame Count [2:0]			S1	Data [15:0]																S0	CRC [4:0]				

MISO Response Following a Write Request

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MISO	1	1	0	0	0	0	Frame Count [2:0]			S1	ANGLE_OUT_P																S0	CRC [4:0]				

Figure 16: A33115 SPI Implementation

The screenshots below reflect two different read requests: one to request data from the primary angle and the other to request data from the null register (i.e., a register containing nothing but zeros).

PRIMARY SERIAL INTERFACE REGISTER REFERENCE

Direct Serial Interface Registers Bits Map

Address (0x00)	Register Symbol	Read/Write	Primary Addressed Byte (MSB)								Primary Addressed Byte (LSB)							
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	NULL_REG	RO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01	INDIRECT_WR_ADDRESS	RW	0	0	0	0	0	0	0	0	INDIRECT_WR_ADDR							
0x02	INDIRECT_WR_DATA_MSB	RW	INDIRECT_WR_DATA_3								INDIRECT_WR_DATA_2							
0x03	INDIRECT_WR_DATA_LSB	RW	INDIRECT_WR_DATA_1								INDIRECT_WR_DATA_0							
0x04	INDIRECT_WR_STATUS	WO/RO	EXW	0	0	0	0	0	0	WIP	0	0	0	0	0	0	0	WDN
0x05	INDIRECT_RD_ADDRESS	RW	0	0	0	0	0	0	0	0	INDIRECT_READ_ADDR							
0x06	INDIRECT_RD_STATUS	RO	EXR	0	0	0	0	0	0	RIP	0	0	0	0	0	0	0	RDN
0x07	INDIRECT_RD_DATA_MSB	RO	INDIRECT_RD_DATA_3								INDIRECT_RD_DATA_2							
0x08	INDIRECT_RD_DATA_LSB	RO	INDIRECT_RD_DATA_1								INDIRECT_RD_DATA_0							
0x09	SIN_S_REG	RO	SIN_S															
0x0A	COS_S_REG	RO	COS_S															
0x0B	TEMP12B_S	RW	0	0	0	0	TEMP_OUT_S											
0x0C	SIN_P_REG	RO	SIN_P															
0x0D	COS_P_REG	RO	COS_P															
0x0E	STATUS_REG	ROC	0	0	0	0	0	0	0	0	0	BACK_FROM_LPM	R	SLR	ABI	ACD	ROT_H	ANG_RDY
0x0F	CTRL	RW	0	0	0	0	0	0	0	0	0	0	0	0	FULL_RST	SOFT_RST	TC_WR_EN	TCR
0x10	PRIM_ANGLE	RO	ANGLE_OUT_P															
0x11	ERROR_0	ROC	IER	XEE	BSY	SME	EUE	ESE	POR	OVCC	UVCC	MSH	MSL	SMM_P	OFE_P	SAT_P	TSE_P	VCF_P
0x12	ERROR_1	ROC	IER_R	XEE_R	BYS_R	SME_R	EUE_R	ESE_R	POR_R	OVCC_R	UVCC_R	MSH_R	MSL_R	SMM_S	OFE_S	SAT_S	TSE_S	VCF_S
0x13	TEMP12B_P	RW	0	0	0	0	TEMP_OUT_P											
0x14	FIELD_REG	RO	FIELD															
0x15	TURNS_COUNTER_P	RW	0	0	TCW_ERROR_P	TCO_ERROR_P	0	TURNS_COUNT_P										
0x16	ANGLE_WITH_HYST	RO	ABI_UVW_ANGLE															
0x17	VELOCITY_REG	RO	VELOCITY															
0x18	ACCELERATION_REG	RO	ACCELERATION															

RO: Read only
WO: Write only
RW: Read and write
RC: Read and clear bit after reading

Figure 17: A33115 Serial Registers

The reason for polling the Null register in addition to the Angle register is to show the out-of-frame functionality. When the first read request is sent, the MOSI outputs (in hexadecimal) 0x20000018 (read request to the Angle register 0x10), and the MISO simultaneously transmits 0x80000011 (response from the Null register). As the datasheet for the A33115 shows, the read response cycle frame MSB starts with a 1 and the 5 bits for the address follow. Since the first two hexadecimal values are 0x80, the Null register is being transmitted. Therefore, the results are as expected as the previously requested frame was the register (not depicted in Figure 18).

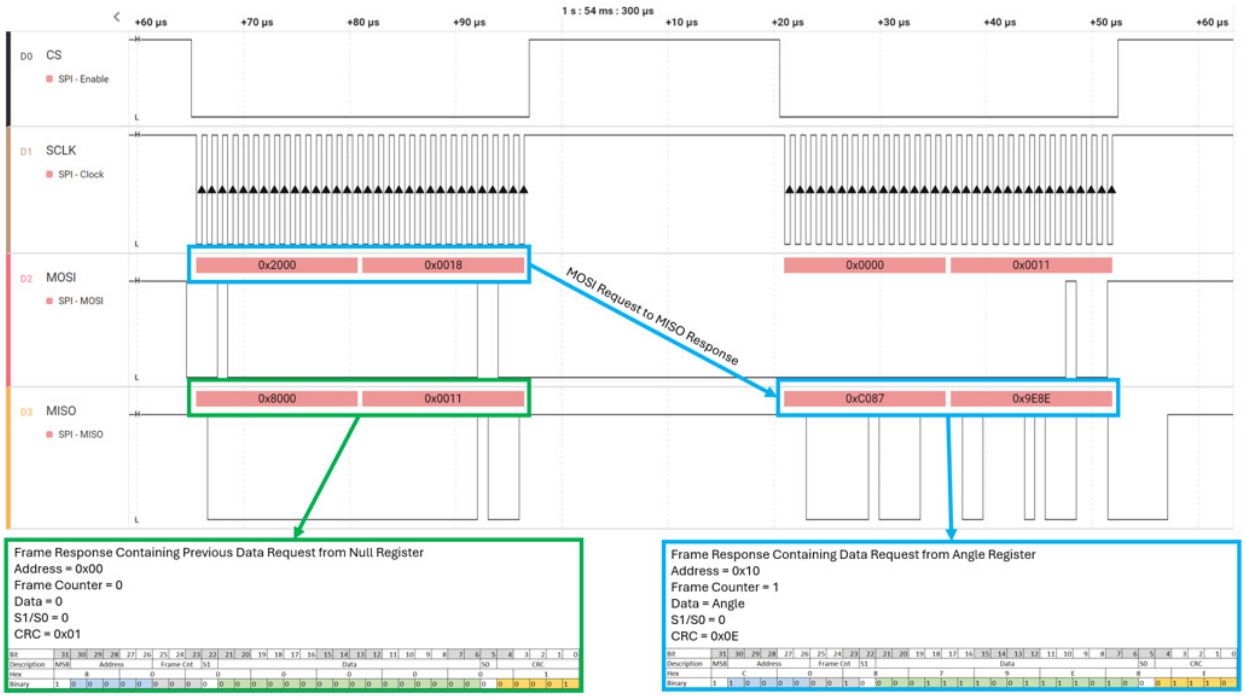

Figure 18: A33115 SPI Capture

When the second request is sent, the MOSI outputs (in hexadecimal) 0x00000011 (read request to the Null register), and the MISO simultaneously transmits 0xC0879E8E (response from the Angle register). Again, referring to the datasheet and decoding the data according to the read response cycle frame format, it can be seen that a valid response has been given, and the data is received from the previously requested address (0x10 or the primary Angle register).

CONCLUSION

When it comes to SPI, there are many ways in which it can be implemented. The important thing to remember is to refer to the peripheral device datasheet, application note, or programming manual to ensure robust and reliable communication.

Revision History

Number	Date	Description
–	March 12, 2025	Initial release

955 PERIMETER ROAD • MANCHESTER, NH 03103 • USA
+1-603-626-2300 • FAX: +1-603-641-5336 • ALLEGROMICRO.COM