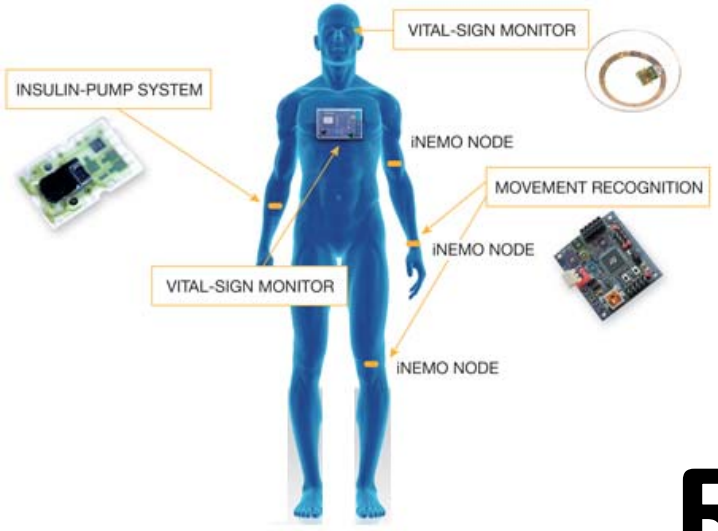

SPI
4-Wire
3-Wire



Real Time Embedded Systems

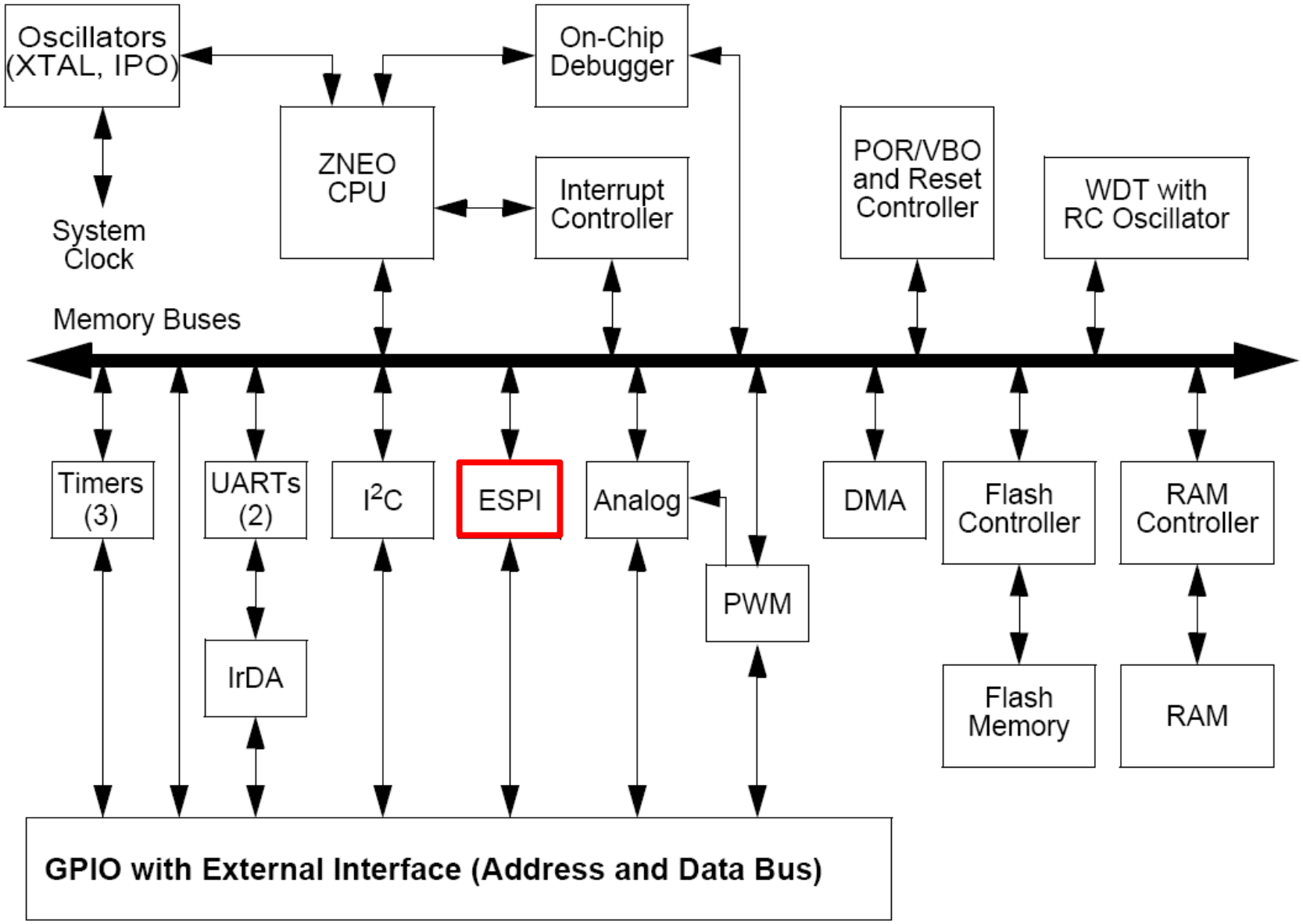
www.atomicrhubarb.com/embedded

Lecture 1 – January 17, 2012

Topic

Section Topic

- Where in the books
  - Catsoulis chapter/page
  - Simon chapter/page
  - Zilog UM197 (ZNEO Z16F Series Flash Microcontroller Contest Kit User Manual)
  - Zilog UM171 (ZiLOG Developer Studio II—ZNEO User Manual)
  - Zilog PS220 (ZNEO Z16F Series Product Specification)
  - Zilog UM188 (ZNEO CPU Core User Manual)
  - Assorted datasheets


Oscillators (XTAL, IPO)
On-Chip Debugger
ZNEO CPU
Interrupt Controller
POR/VBO and Reset Controller
WDT with RC Oscillator
System Clock
Memory Buses
Timers (3)
UARTs (2)
I2C
ESPI
Analog
DMA
Flash Controller
RAM Controller
PWM
IrDA
Flash Memory
RAM
GPIO with External Interface (Address and Data Bus)

Synchronous Serial Buses

- 1-wire
- 2-wire
- 3-wire
- 4-wire

... but not in this order

Serial Buses

- Communication between components
  - Typically between integrated circuit components
  - CPU to peripherals
- Typically short distances (at least that was the plan)
- Number of wires = number of SIGNAL wires
  - Power & ground not included in the counting
- Usually synchronous (data on one line and clock on another)
- Lots of variety, lots of similarity.

The nice thing about standards is that there are so many of them to choose from.

- Andrew Tannenbaum
  (or was it Grace Hopper)

Why so many?

- 1,2,3,4 wire?

Why so many?

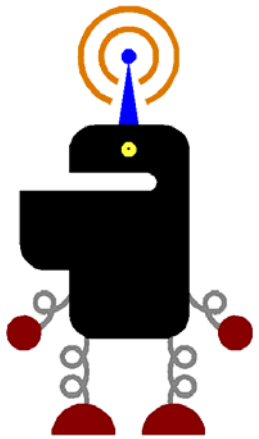
- 4-wire – SPI – Motorola
- 3-wire – Maxim IC
  Microwire – National Semiconductor
- 2-wire – I^2C – Phillips Semiconductor
- 1-wire – Dallas Semiconductor
  SensorPath – National Semiconductor
  UNI/O - Microchip

SPI (4 wire)

- Serial Peripheral Interface
- 4 wires (really 3 + 1 for each device)
- Low Cost
- Simple
- Intended for CPU to Peripheral communication and control.
- Defined by Motorola.

SPI Specification

- The Specification is hidden inside the HC08 microcontroller data sheets
- Motorola Semiconductor is now Freescale Semiconductor.
- Defines 4 signals



SPI Signals

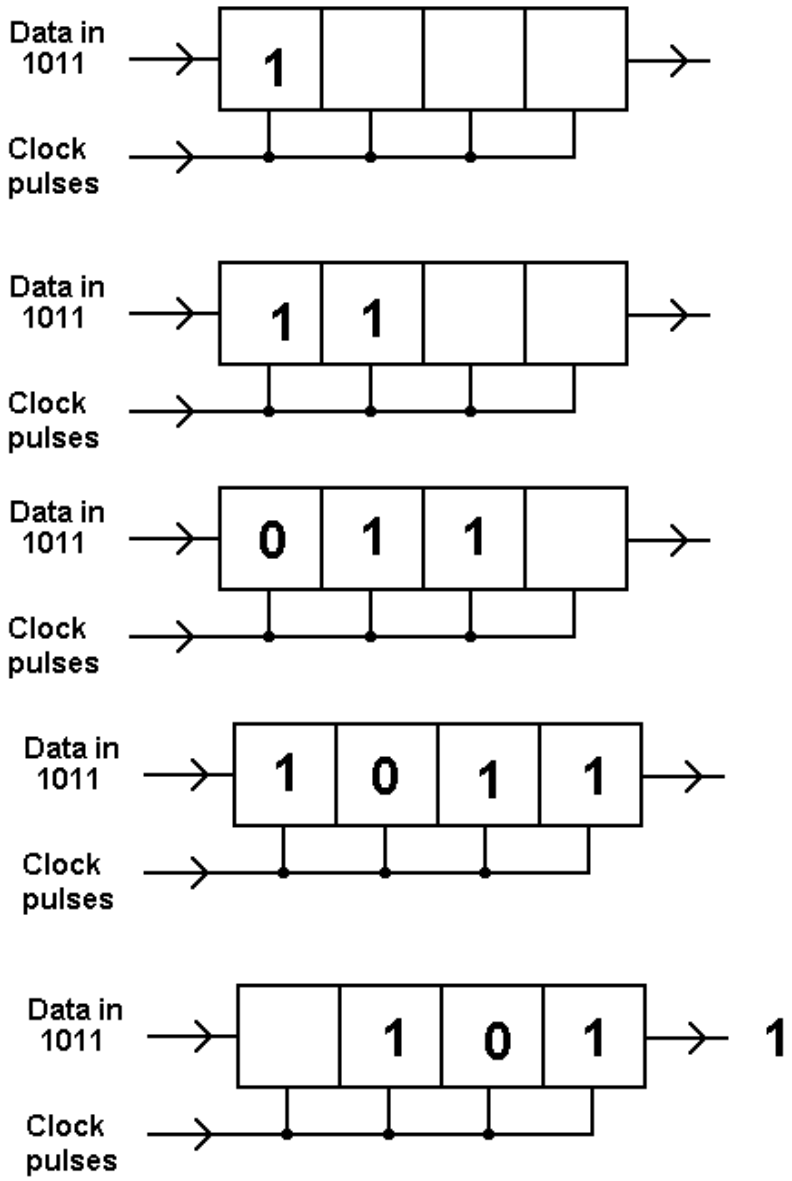
- SCLK – Serial Clock
- MISO – Master In, Slave Out
- MOSI – Master Out, Slave In
- SS – Slave Select

- Based on shifting data between 2 shift registers.

Whats a shift register?

Shift Register

- A cascade of flip flops sharing the same clock. The output of any one but the last flip-flop connected to the input of the next one in the chain. Resulting in a circuit that shifts by one position the one-dimensional "bit array" stored in it.



SPI Signals

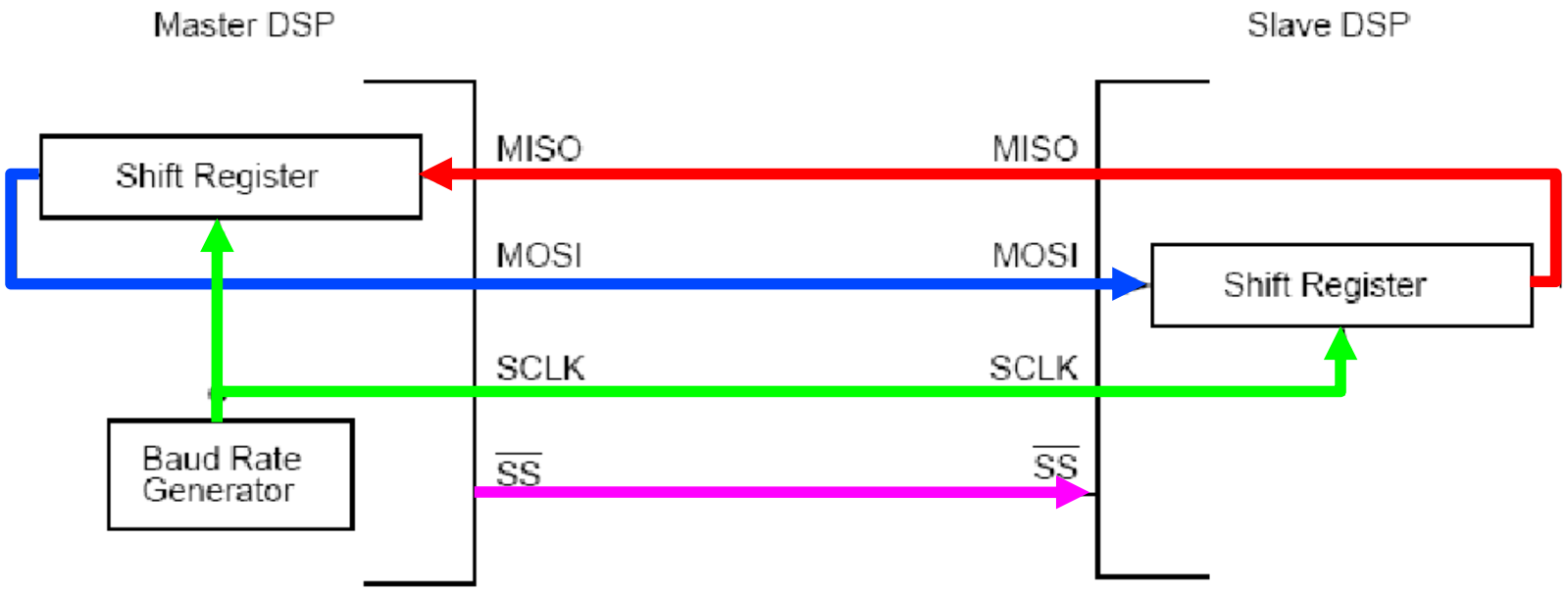
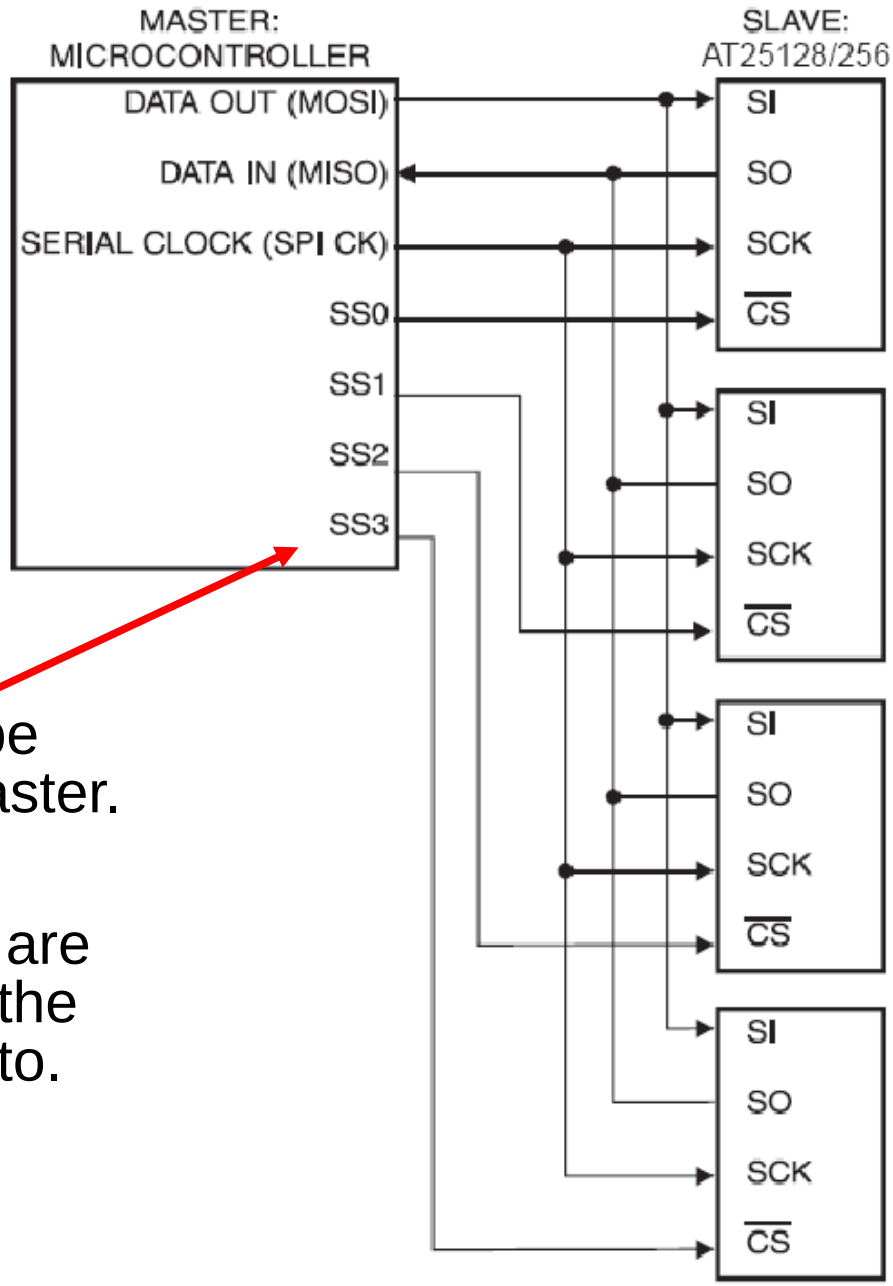


Figure 10-3. Full-Duplex Master/Slave Connections

- Swap 2 bytes, 1 bit at at time using the BRG as our serial clock.



- Multiple chips can be connected to the master.

- Individual SS wires are necessary to select the one we want to talk to.

SPI Modes

- 4 timing modes
- Based on clock polarity and clock

Table 94. ESPI Clock Phase (PHASE) and Clock Polarity (CLKPOL) Operation

PHASE	CLKPOL	SCK Transmit Edge	SCK Receive Edge	SCK Idle State
0	0	Falling	Rising	Low
0	1	Rising	Falling	High
1	0	Rising	Falling	Low
1	1	Falling	Rising	High

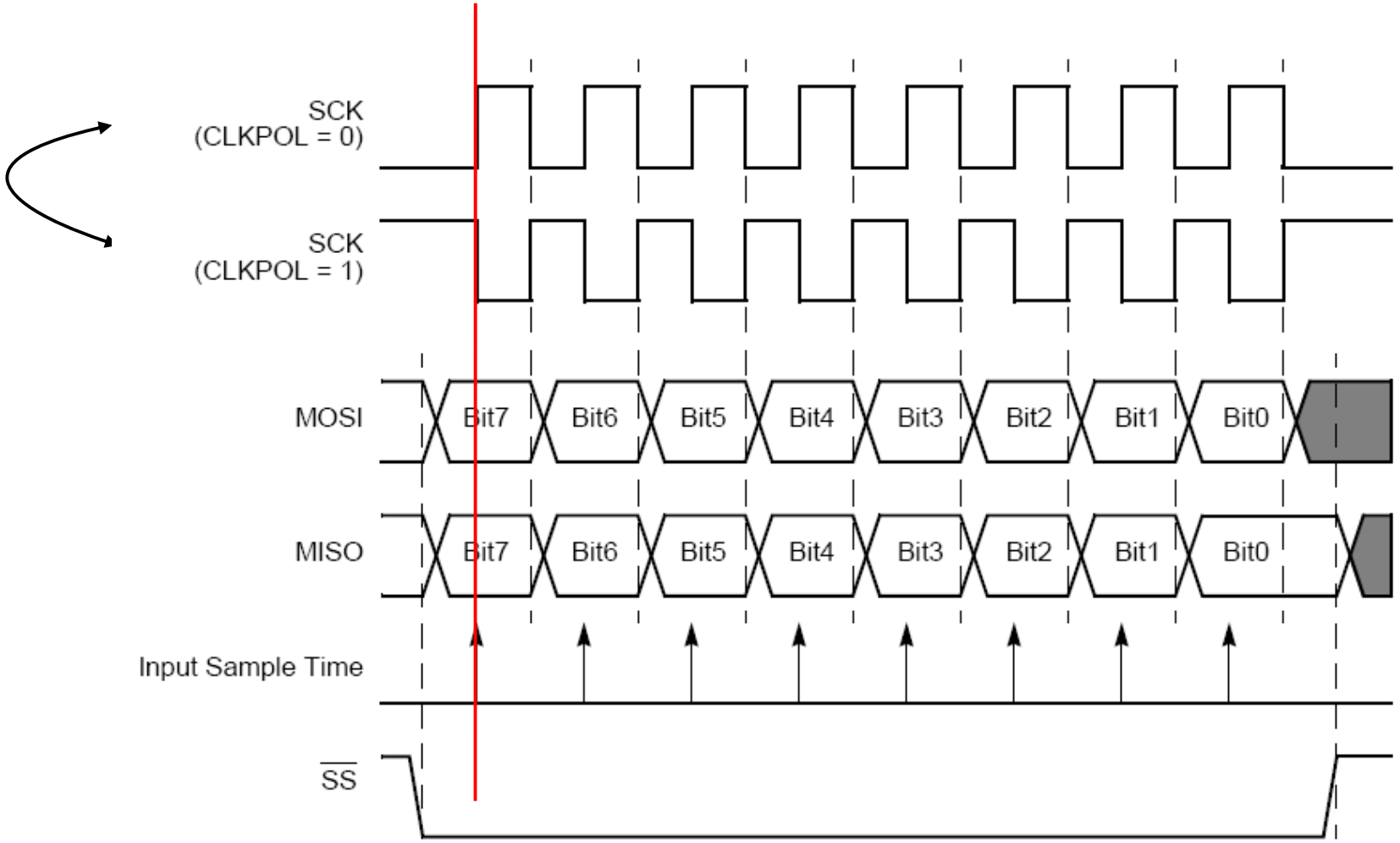


Figure 35. ESPI Timing When PHASE = 0

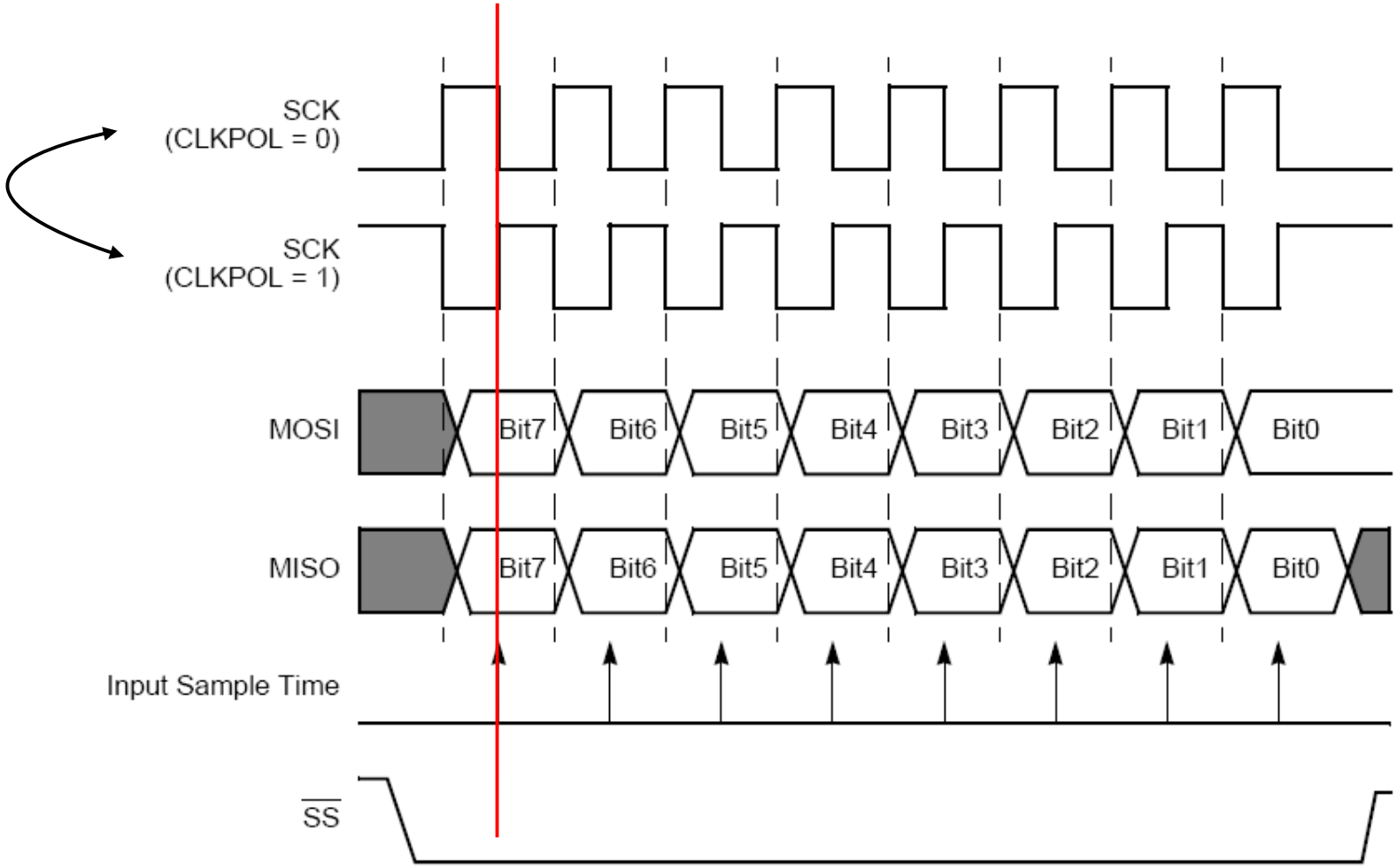
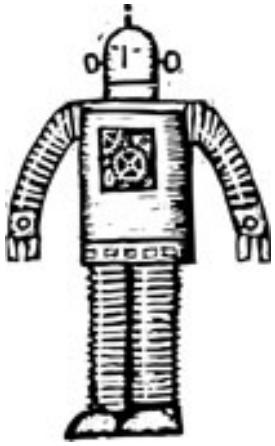


Figure 36. ESPI Timing when `PHASE` = 1

Timing

- It is common to see a reference to the SPI mode as CKPOL = 0, PHASE = 0
- This is by far the most common SPI transfer.



SPI data

- Is device specific
- Some need a single byte
- Some need multiple bytes
- Reads follow writes
- A dummy write is necessary to initiate a read.

What?

- A dummy write is necessary to initiate a read.

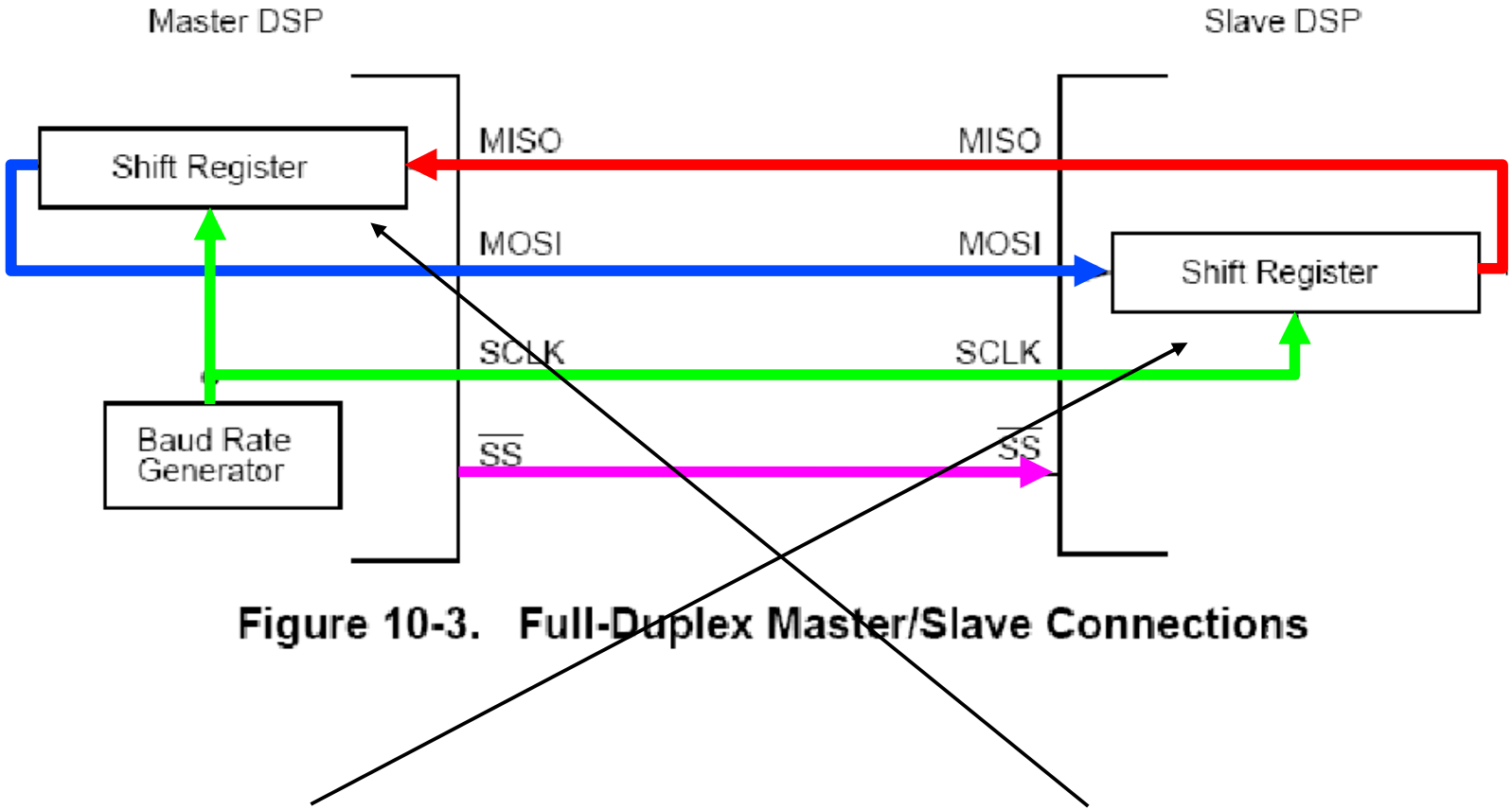


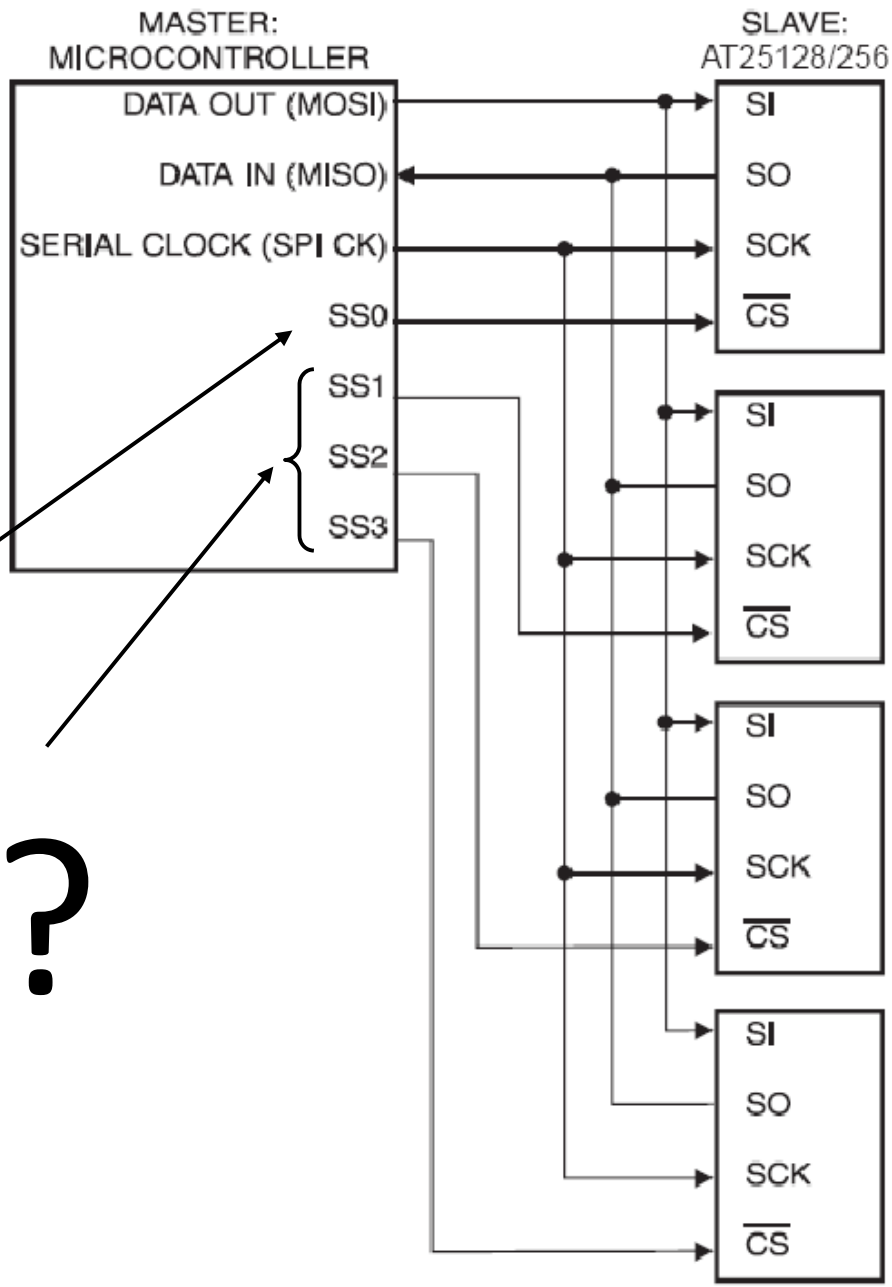
Figure 10-3. Full-Duplex Master/Slave Connections

To read THIS data we need to write something (anything) to shift the slave data to the master.

What if…

- … I only have
- one SS pin?

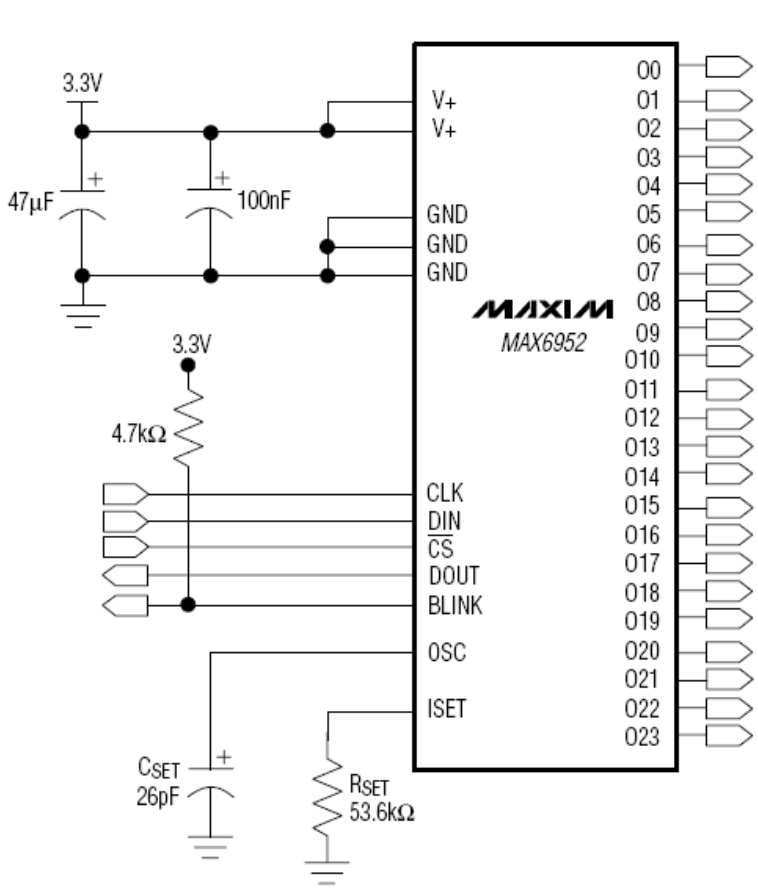
?

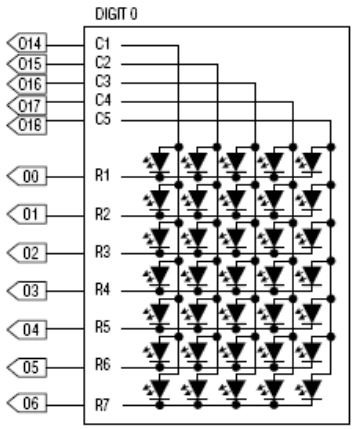


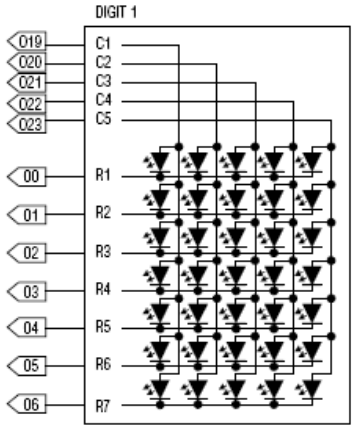
SPI Devices

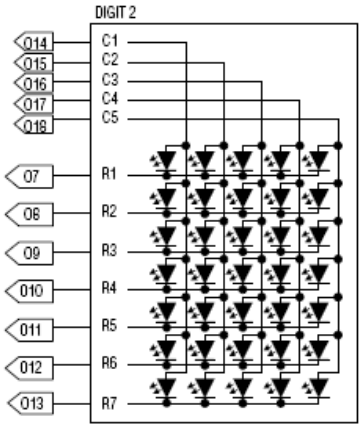
- IO Expanders
- Clocks
- Temperature Sensors
- Non-volatile memory
- … lots of things.

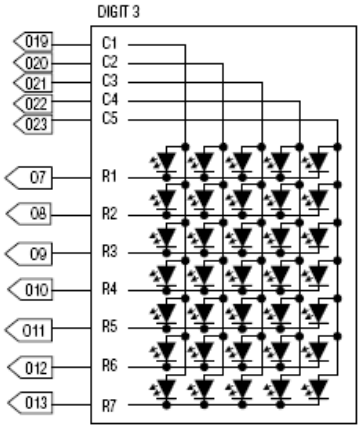
4-Wire Interfaced, 2.7V to 5.5V, 4-Digit 5 × 7 Matrix LED Display Driver











ESPI on the ZNEO

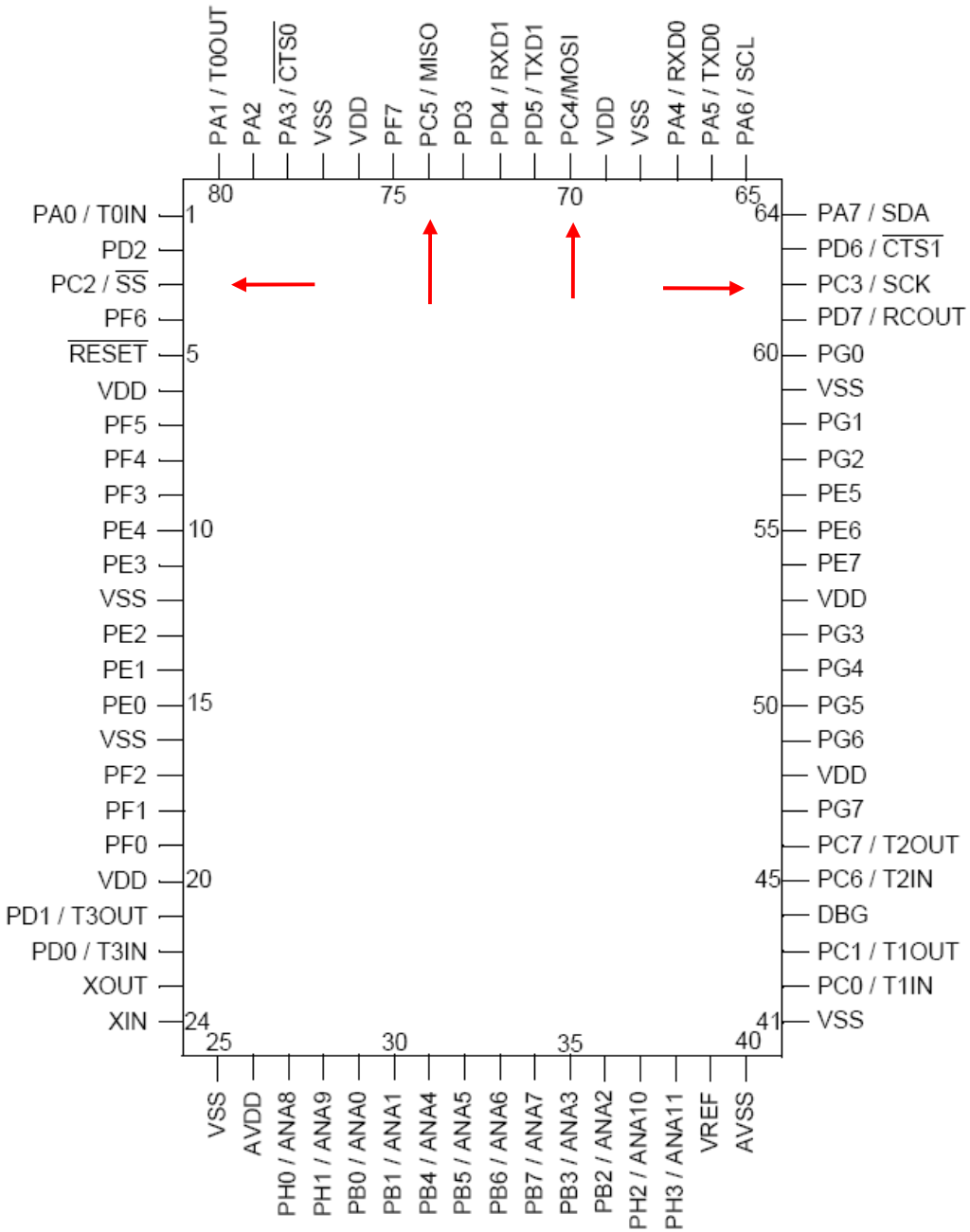
- **Enhanced** Serial Peripheral Interface

ESPI

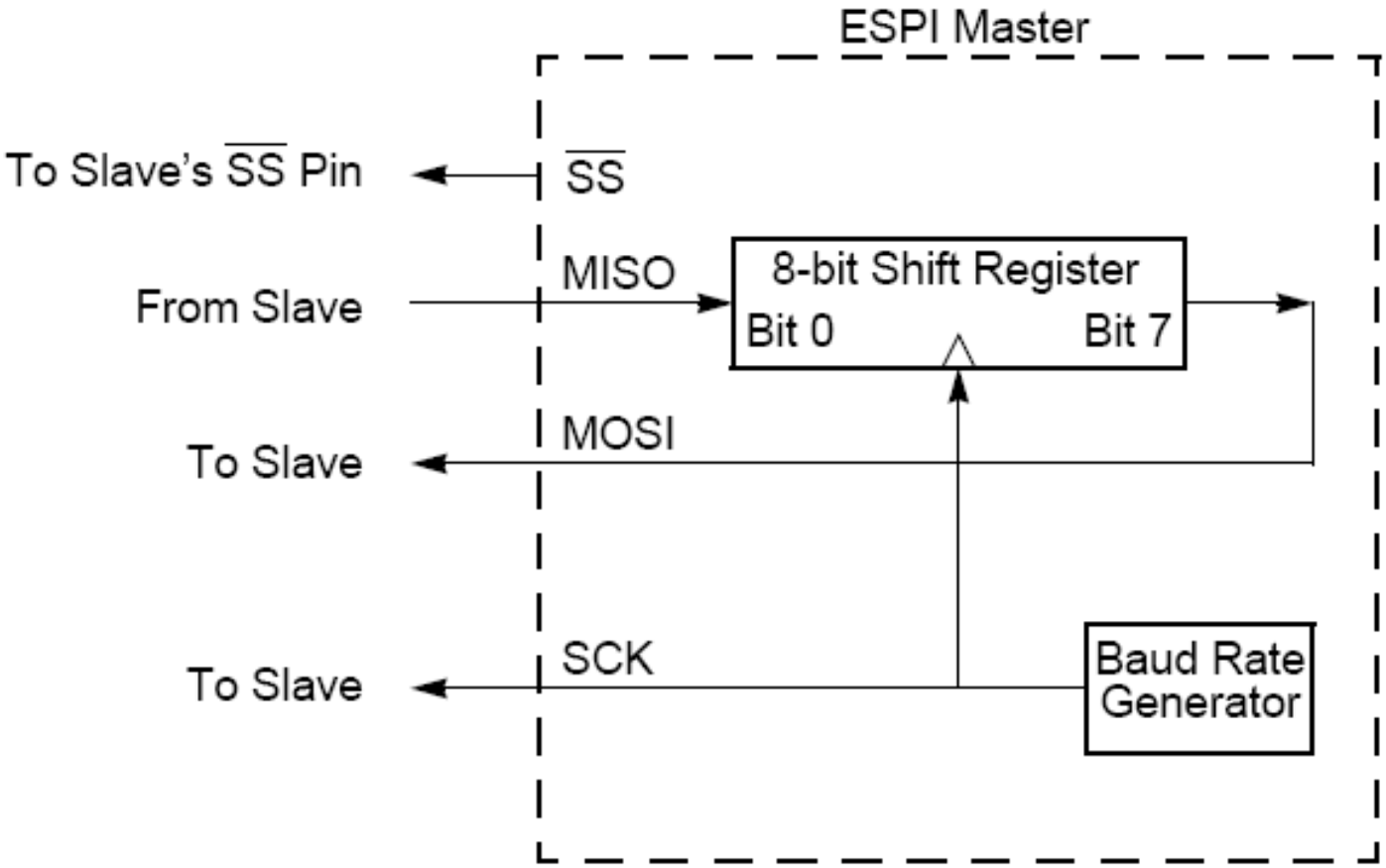
- The Enhanced Serial Peripheral Interface (SPI) is a synchronous interface allowing several SPI-type devices to be interconnected. SPI-compatible devices include EEPROMs, Analog-to-Digital Converters, and ISDN devices. Features of the SPI include:
- Full-duplex, synchronous, character-oriented communication
- Four-wire interface
- Data transfers rates up to a maximum of one-fourth the system clock frequency
- **Error detection**
- Write and mode **collision detection**
- Dedicated Baud Rate Generator

ESPI Pins

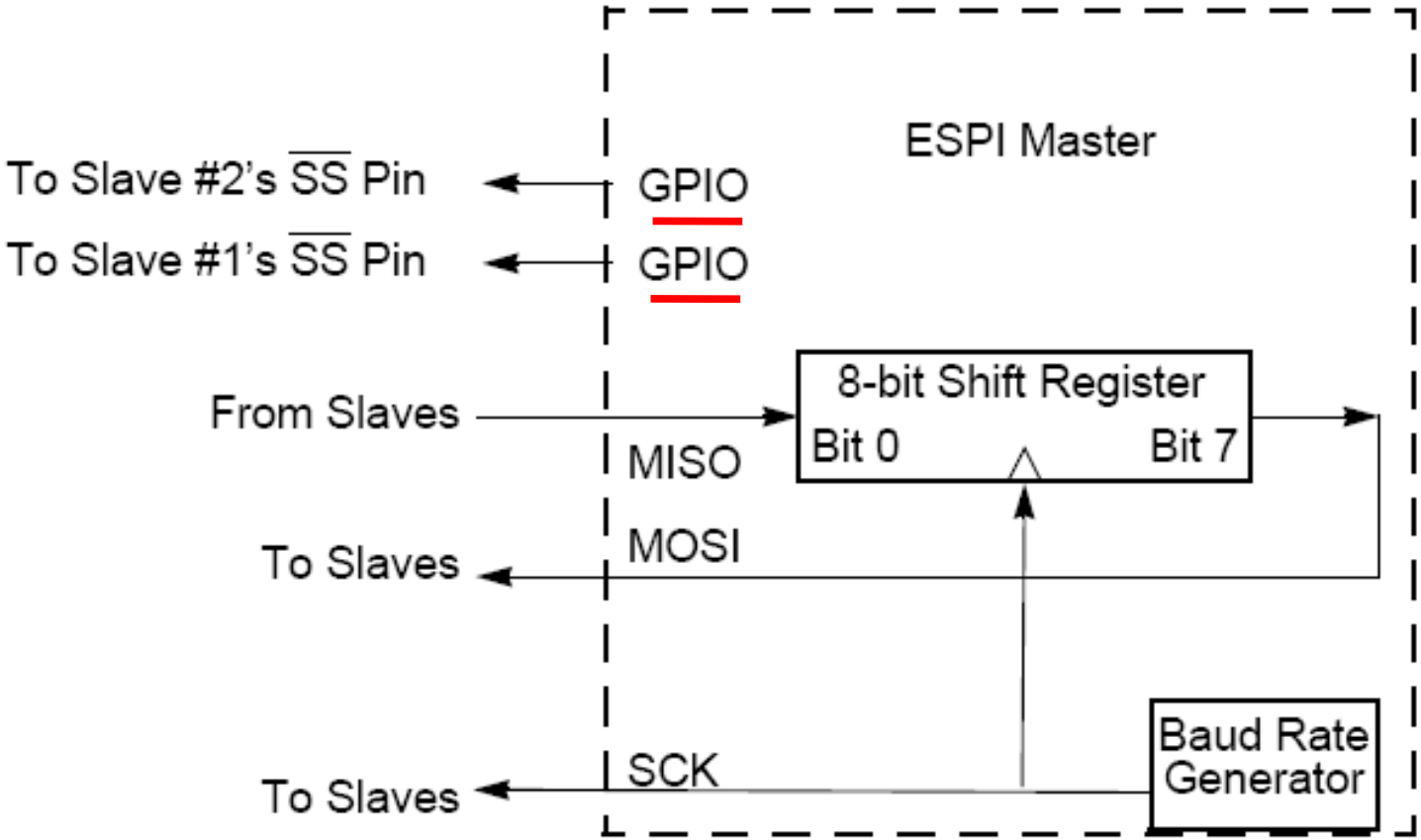
- PC2 - /SS
- PC3 - SCK
- PC4 - MOSI
- PC5 - MISO



ESPI Block Diagram



What was that thing we could do to access multiple SPI devices with only ONE /SS pin?


ESPI Master
To Slave #2's SS Pin
GPIO
To Slave #1's SS Pin
GPIO
From Slaves
8-bit Shift Register
Bit 0
Bit 7
MISO
MOSI
To Slaves
SCK
To Slaves
Baud Rate Generator

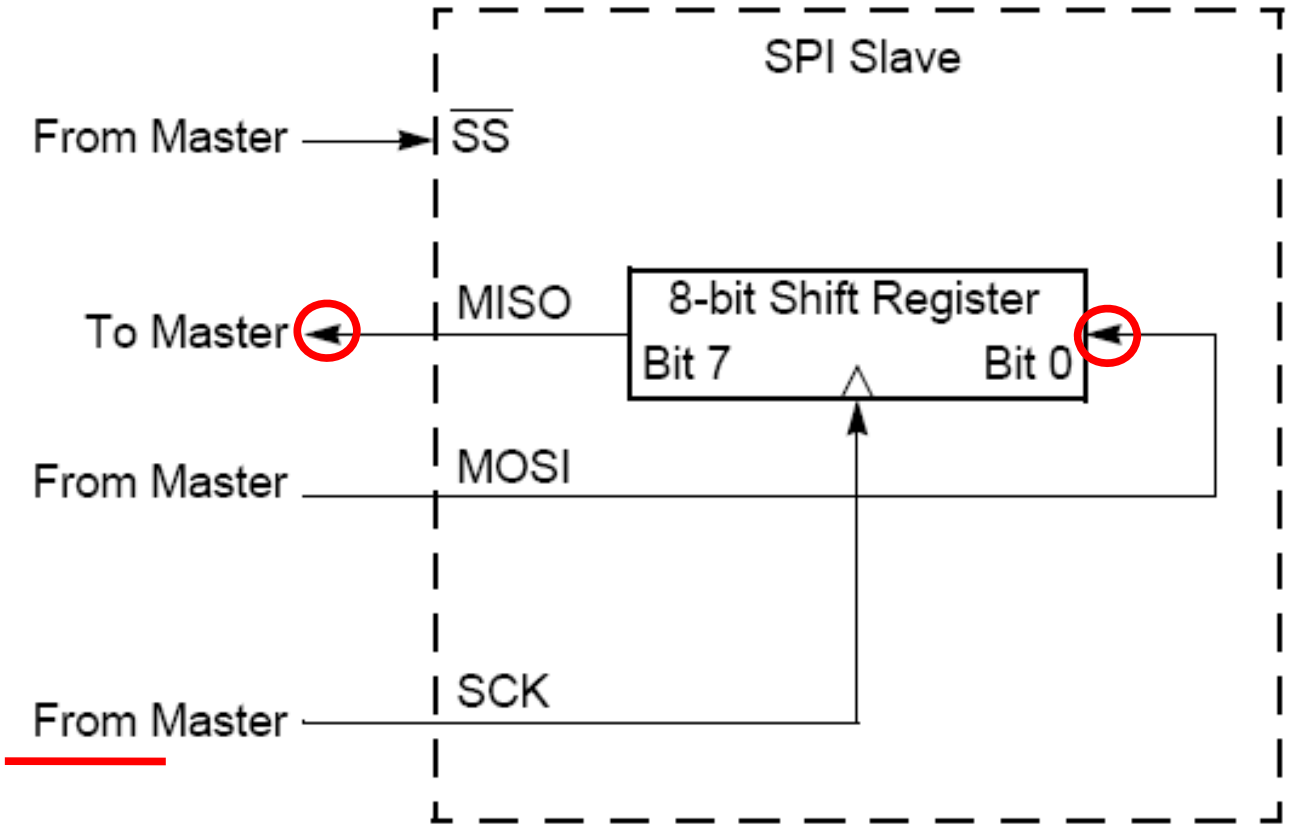


Figure 41. ESPI Configured as an SPI Slave

ESPI

- Several Modes
  - Master controller of a single device
  - Slave Device
  - Master controller of multiple devices

ESPI Registers

- ESPIDATA – Byte to be transmitted or byte received
- ESPITDCR – Transmit Data Command Register
- ESPICTL – Configuration register
- ESPISTAT – Status Register
- ESPISTATE – State register
- ESPIMODE – More configuration
- ESPIBRH/L – Baud Rate

ESPIDATA

Table 98. ESPI Data Register (ESPIDATA)

BITS	7	6	5	4	3	2	1	0
FIELD	DATA							
RESET	X	X	X	X	X	X	X	X
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FF_E260H							

DATA—Data

Transmit and/or receive data. Writes to the ESPIDATA register load the shift register. Reads from the ESPIDATA register return the value of the receive data register.

ESPICTL

Table 100. ESPI Control Register (ESPICTL)

BITS	7	6	5	4	3	2	1	0
FIELD	DIRQE	ESPIEN1	BRGCTL	PHASE	CLKPOL	WOR	MMEN	ESPIEN0
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FF_E262H							

DIRQE—Data Interrupt Request Enable

This bit is used to disable or enable data (TDRE and RDRF) interrupts. Disabling the data interrupts is needed when controlling data transfer by DMA or polling. Error interrupts are not disabled. To block all ESPI interrupt sources, clear the ESPI interrupt enable bit in the Interrupt Controller.

0 = TDRE and RDRF assertions do not cause an interrupt.
Use this setting if controlling data transfer through DMA or by software polling of TDRE and RDRF. The TUND, COL, ABT, and ROVR bits cause an interrupt.

1 = TDRE and RDRF assertions will cause an interrupt.
TUND, COL, ABT, and ROVR will also cause interrupts. Use this setting if controlling data transfer through interrupt handlers.

ESPIEN1, ESPIEN0—ESPI Enable and Direction Control

00 = ESPI block is disabled.
BRG is used as a general purpose timer by setting BRGCTL = 1.

01 = RECEIVE ONLY Mode.
Use this setting if the software application is receiving data but not sending. TDRE will assert, however the transmit interrupt and DMA requests will not assert. In SLAVE mode, the transmitted data will be all 1s.

In MASTER mode software must still write to the Transmit Data register to initiate the transfer.

10 = TRANSMIT ONLY Mode
Use this setting in MASTER or SLAVE mode when the software application is sending data but not receiving. RDRF will assert, but receive interrupt and DMA requests
not occur.

11 = TRANSMIT/RECEIVE Mode
Use this setting if the software application is both sending and receiving information. Both TDRE and RDRF will be active.

BRGCTL—Baud Rate Generator Control

The function of this bit depends upon ESPIEN1,0. When ESPIEN1,0 = 00, this bit allows enabling the BRG to provide periodic interrupts.

If the ESPI is disabled (`ESPIEN1, ESPIEN0 = 00`):

0 = The BRG timer function is disabled.
Reading the Baud Rate High and Low registers returns the BRG Reload value.

1 = The BRG timer function and time-out interrupt are enabled.
Reading the Baud Rate High and Low registers returns the BRG Counter value.

PHASE—Phase Select
Sets the phase relationship of the data to the clock. For more information on operation of the PHASE bit, see ESPI Clock Phase and Polarity Control on page 177.

CLKPOL—Clock Polarity
0 = SCK idles Low (0).
1 = SCK idles High (1).

WOR—Wire-OR (Open-Drain) Mode Enabled
0 = ESPI signal pins not configured for open-drain.
1 = All four ESPI signal pins (SCK, SS, MISO, MOSI) configured for open-drain function. This setting is used for Multi-Master and/or Multi-Slave configurations.

MMEN—ESPI Master Mode Enable
This bit controls the data I/O pin selection and SCK direction.
0 = Data-out on MISO, data-in on MOSI (used in SPI Slave mode), SCK is an input.
1 = Data-out on MOSI, data-in on MISO (used in SPI Master mode), SCK is an output.

SPISTAT

Table 102. ESPI Status Register (ESPISTAT)

BITS	7	6	5	4	3	2	1	0
FIELD	TDRE	TUND	COL	ABT	ROVR	RDRF	TFST	SLAS
RESET	0	0	0	0	0	0	0	1
R/W	R	R/W*	R/W*	R/W*	R/W*	R	R	R
ADDR	FF_E264H							
R/W* = Read access. Write a 1 to clear the bit to 0.								

TDRE—Transmit Data Register Empty

0 = Transmit data register is full or ESPI is disabled.

1 = Transmit data register is empty. A write to the ESPI (Transmit) Data register clears this bit.

TUND—Transmit Underrun

0 = A Transmit Underrun error has not occurred.

1 = A Transmit Underrun error has occurred.

COL—Collision

0 = A Multi-Master collision (mode fault) has not occurred.

1 = A Multi-Master collision (mode fault) has been detected.

ABT—Slave mode transaction abort

This bit is set if the ESPI is configured in Slave mode, a transaction is occurring and $\overline{SS}$ deasserts before all bits of a character have been transferred as defined by the NUMBITS field of the ESPIMODE register. This bit is also be set in Slave mode by an SCK monitor timeout (MMEN = 0, BRGCTL = 1).

0 = A Slave mode transaction abort has not occurred.

1 = A Slave mode transaction abort has been detected.

ROVR—Receive Overrun

0 = A Receive Overrun error has not occurred.

1 = A Receive Overrun error has occurred.

RDRF—Receive Data Register Full

0 = Receive Data register is empty.

1 = Receive Data register is full. A read from the ESPI (Receive) Data register clears this bit.

SPIMODE

Table 101. ESPI Mode Register (ESPIMODE)

BITS	7	6	5	4	3	2	1	0
FIELD	SSMD			NUMBITS[2:0]			SSIO	SSPO
RESET	000			000			0	0
R/W	R/W			R/W			R/W	R/W
ADDR	FF_E263H							

SSMD—SLAVE SELECT Mode
This field selects the behavior of $\overline{SS}$ as a framing signal. For a detailed description of these modes, see Slave Select on page 174.

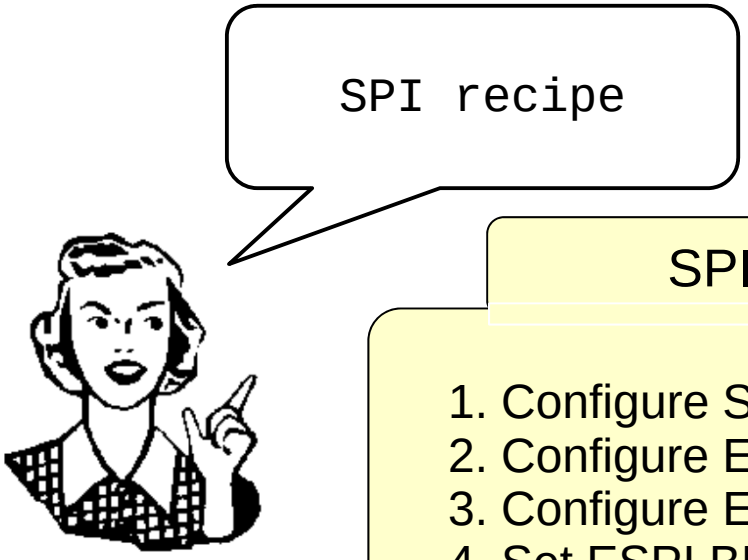
000 = SPI mode
When SSIO = 1, the $\overline{SS}$ pin is driven directly from the SSV bit in the Transmit Data Command register. The Master software or DMA must set SSV (or a GPIO output if the $\overline{SS}$ pin is not connected to the appropriate Slave) to the asserted state prior to or on the same clock cycle with which the transmit data register is written with the initial byte. At the end of a frame (after the last RDRF event), SSV is deasserted by software. Alternatively, SSV is automatically deasserted by hardware if the TEOF bit in the Transmit Data Command Register is set when the last transmit byte is loaded. In SPI mode, SCK is active only for data transfer (one clock cycle per bit transferred).

SPIBRH,L

ESPI Baud Rate High and Low Byte Registers

The ESPI Baud Rate High and Low Byte registers (see Table 105 and Table 106) combine to form a 16-bit reload value, BRG[15:0], for the ESPI Baud Rate Generator. The ESPI baud rate is calculated using the following equation:

$$\text{SPI Baud Rate (bps)} = \frac{\text{System Clock Frequency (Hz)}}{2 \times \text{BRG[15:0]}}$$



SPI

1. Configure SPI pins (alternate function)
2. Configure ESPISTL (phase, pol, etc)
3. Configure ESPIMODE (ssio, etc)
4. Set ESPI BRG
5. Write the spi_read functions **
6. Write the spi_write functions **
7. Write device specific functions
8. Configure interrupts if desired
9. Write Interrupt Service Routine, set vector
10. Configure SPI to generate interrupts
    (set DIRQE bit of ESPICTL)
10. Enable interrupts (EI)
11. Enable SPI (set bit 0 of SPICTL)

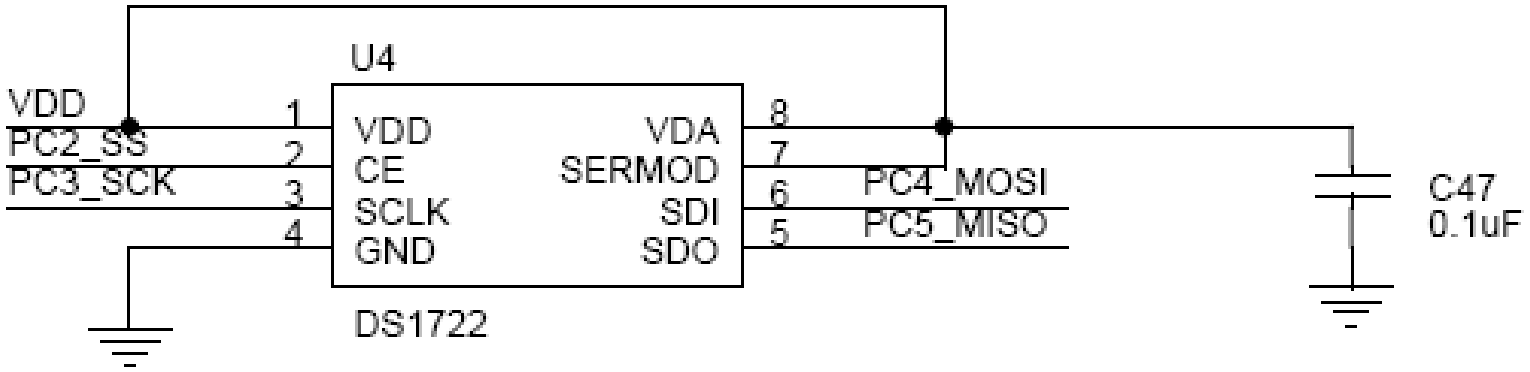
** non-trivial

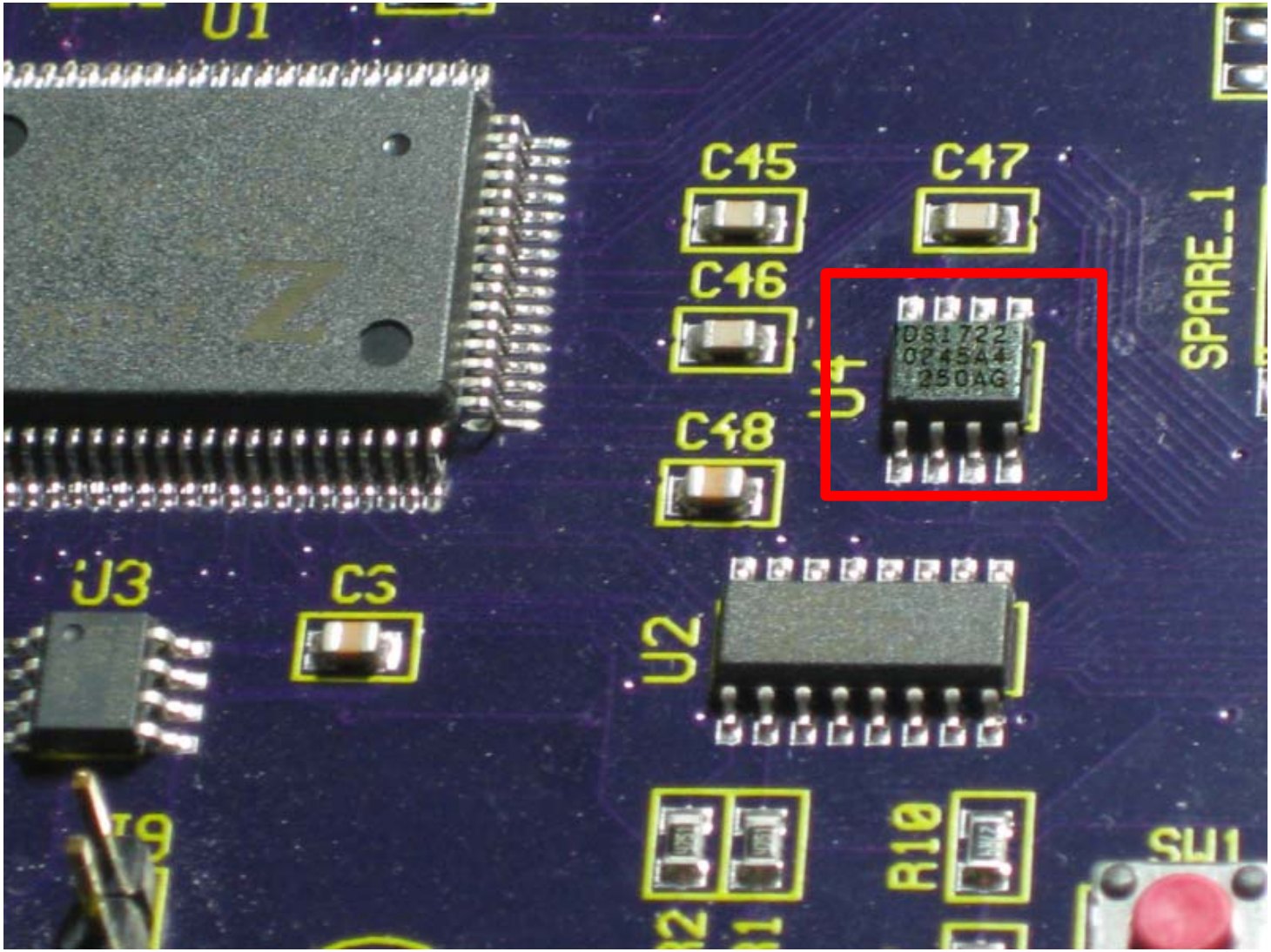
SPI Details

- Data read/written is device specific.

DS1722

- This is on the ZNEO developer board




U1
C45
C47
C46
U4
DS1722
0245A4
250AG
C48
SPARE_1
U3
C3
U2
R10
SW1

DS1722

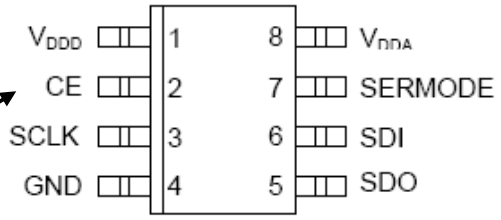
- Get the datasheet !

DALLAS SEMICONDUCTOR MAXIM

DS1722
Digital Thermometer with SPI/3-Wire Interface

www.maxim-ic.com

PIN ASSIGNMENT



DS1722S
8-Pin SO (150mil)

SOIC	SYMBOL	DESCRIPTION
PIN 1	V_{DDD}	**Digital Supply Voltage** 1.8V-5.5V. Defines the top rails for the digital inputs and outputs.
PIN 2	CE	**Chip Enable** Must be asserted high for communication to take place for either the SPI or 3-wire interface.
PIN 3	SCLK	**Serial Clock Input** Used to synchronize data movement on the serial interface for either the SPI or 3-wire interface.
PIN 4	GND	**Ground pin.**
PIN 5	SDO	**Serial Data Output** When SPI communication is selected, the SDO pin is the serial data output for the SPI bus. When 3-wire communication is selected, this pin must be tied to the SDI pin (the SDI and SDO pins function as a single I/O pin when tied together.)
PIN 6	SDI	**Serial Data Input** When SPI communication is selected, the SDI pin is the serial data input for the SPI bus. When 3-wire communication is selected, this pin must be tied to the SDO pin (the SDI and SDO pins function as a single I/O pin when tied together.)
PIN 7	SERMODE	**Serial Interface Mode Input** This pin selects which interface standard will be used: SPI when connected to V_{CC} ; standard 3-wire when connected to GND.
PIN 8	V_{DDA}	**Analog Supply Voltage** 2.65V – 5.5V input power pin.

Note

- CE = active HIGH
- /SS = active LOW

- Use the SSPO bit of the ESPIMODE register to control this.

Register Address Structure Table 4

Read Address	Write Address	Active Register
00h	80h	Configuration
01h	No access	Temperature LSB
02h	No access	Temperature MSB

Temperature/Data Relationships Table 3

								Address Location
S	2^6	2^5	2^4	2^3	2^2	2^1	2^0	02h
MSb				(unit = °C)			LSb	
2^{-1}	2^{-2}	2^{-3}	2^{-4}	0	0	0	0	01h

CONFIGURATION/STATUS REGISTER Figure 2

1	1	1	1SHOT	R2	R1	R0	SD
MSb							LSb

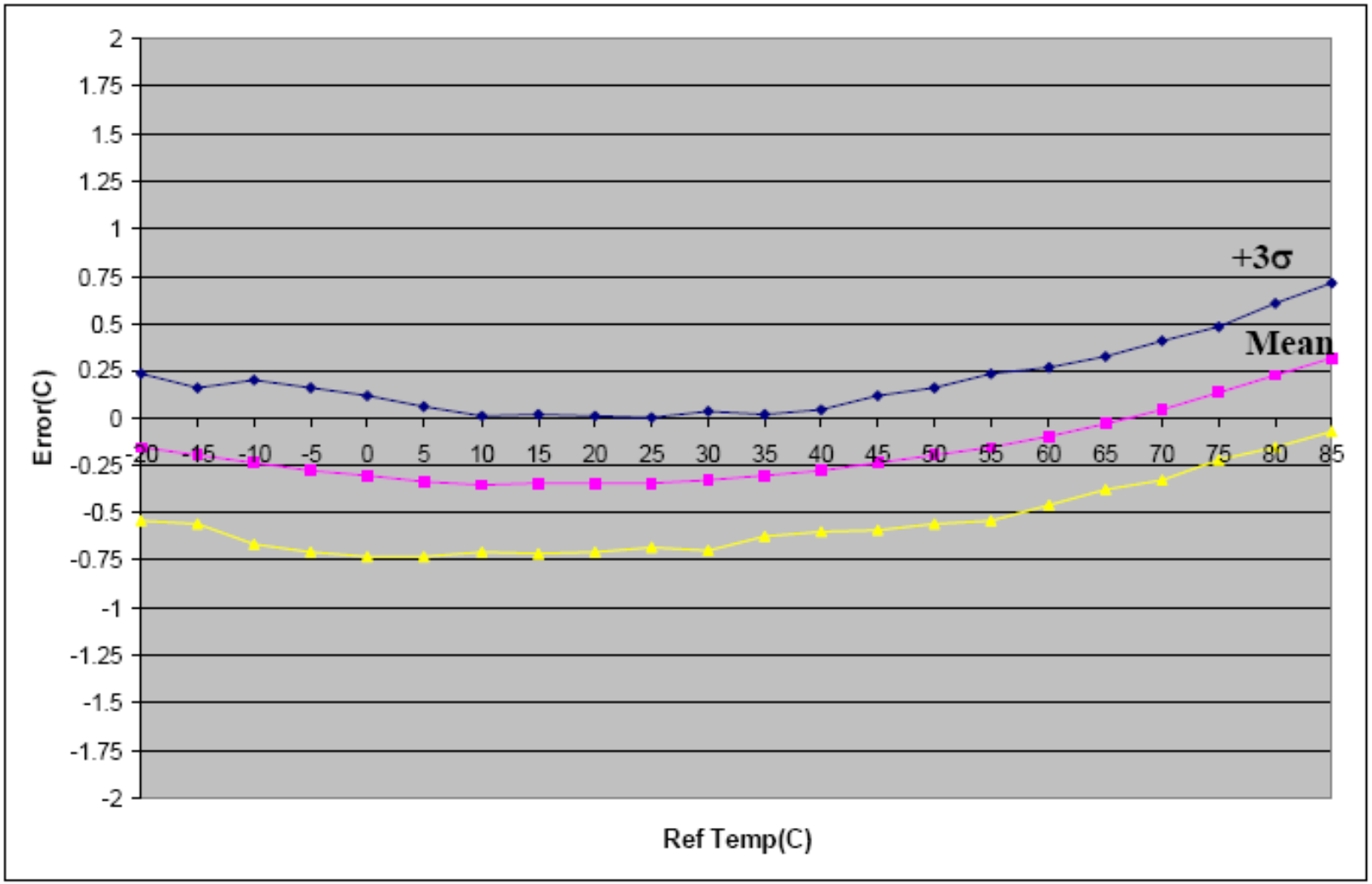
THERMOMETER RESOLUTION CONFIGURATION Table 5

R2	R1	R0	Thermometer Resolution	Max Conversion Time
0	0	0	8-bit	0.075s
0	0	1	9-bit	0.15s
0	1	0	10-bit	0.3s
0	1	1	11-bit	0.6s
1	x	x	12-bit	1.2s

x=Don't care.

TEMPERATURE	DIGITAL OUTPUT (BINARY)	DIGITAL OUTPUT (HEX)
+120°C	0111 1000 0000 0000	7800h
+25.0625°C	0001 1001 0001 0000	1910h
+10.125°C	0000 1010 0010 0000	0A20h
+0.5°C	0000 0000 1000 0000	0080h
0°C	0000 0000 0000 0000	0000h
-0.5°C	1111 1111 1000 0000	FF80h
-10.125°C	1111 0101 1110 0000	F5E0h
-25.0625°C	1110 0110 1111 0000	E6F0h
-55°C	1100 1001 0000 0000	C900h

DS1722 Error



3-wire

3-WIRE SERIAL DATA BUS

The 3-wire communication mode operates similar to the SPI mode. However, in 3-wire mode, there is one bi-directional I/O instead of separate data in and data out signals. The 3-wire consists of the I/O (SDI and SDO pins tied together), CE, and SCLK pins. In 3-wire mode, each byte is shifted in LSB first unlike SPI mode where each byte is shifted in MSB first. As is the case with the SPI mode, an address byte is written to the device followed by a single data byte or multiple data bytes. Figure 7 illustrates a read and write cycle. Figure 8 illustrates a multiple byte burst transfer. In 3-wire mode, data is input on the rising edge of SCLK and output on the falling edge of SCLK.

3-Wire/Microwire

- The MICROWIRE protocol is essentially a subset of the SPI interface, specifically CPOL = 0 and CPHA = 0.
- The Maxim 3-wire interface is found on some ICs from Maxim. The data flow to and from the device is multiplexed on one pin (DQ) while SPI needs two separate signals (MOSI, MISO).

ESPI Example

Example_
SpiDs1722

- Read from the DS1722 SPI Temperature sensor