EEC 417/517
Embedded Systems
Cleveland State University

Lab 7

SPI Mode, 7-Segment LED Displays, Shift Registers

Lab07 Setup, Keypad Scanning, Finite Arithmetic

Watchdog Timer

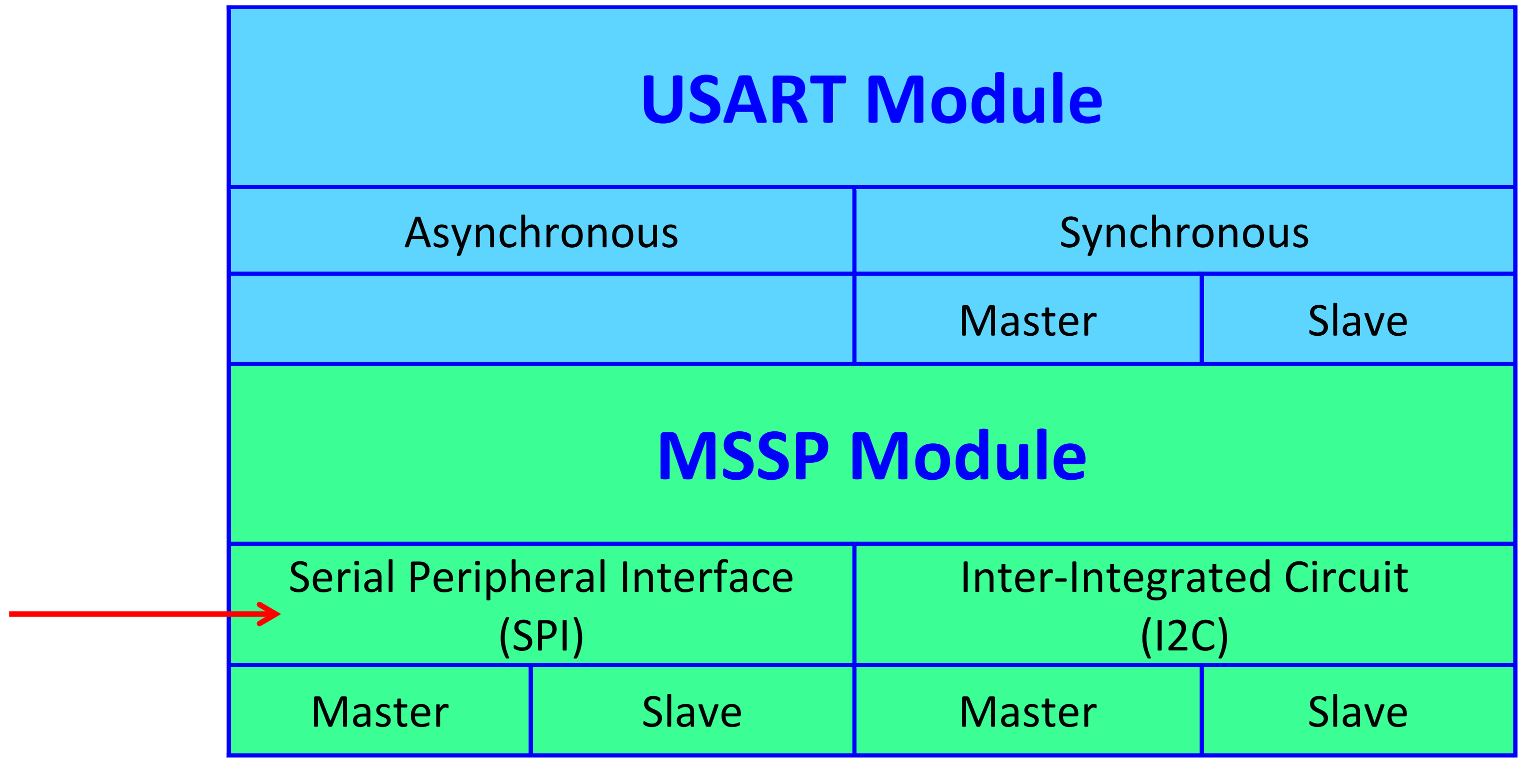
Dan Simon

Rick Rarick

Spring 2018

Lab 7 Outline

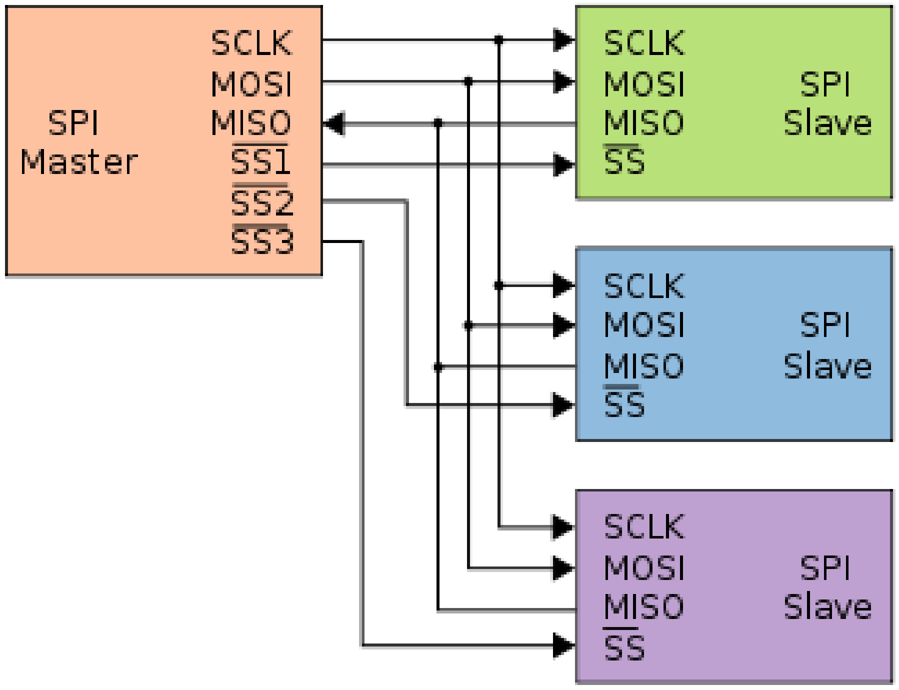
PIC Serial Communications
Master Synchronous Serial Port Module
Serial Peripheral Interface (SPI) Mode



SPI developed by Motorola

Serial Peripheral Interface (SPI)

Master-Slave Interconnection



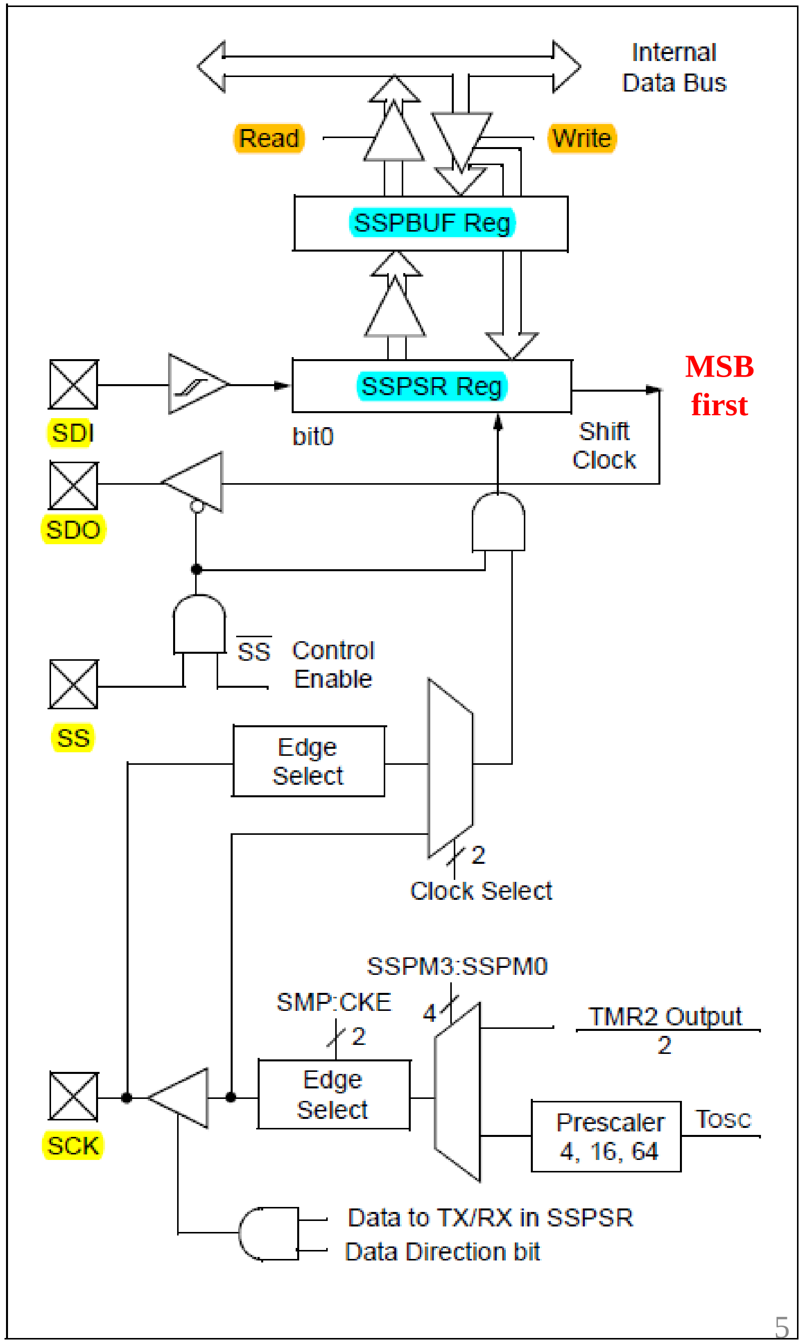
1. Master supplies clock signal
2. Master initiates data transfer
3. Full duplex

Master Synchronous Serial Port (MSSP)

SPI Mode

1. SDO = SPI Data Output
2. SDI = SPI Data Input
3. SCK = SPI Clock
4. SS = SPI Slave Select

- When a data byte is written to the SSPBUF register, the 8 data bits are automatically sent from SDO (RC5) pin.
- When a data byte is received at SDI (RC4) pin, the byte is automatically written to the SSPBUF register.



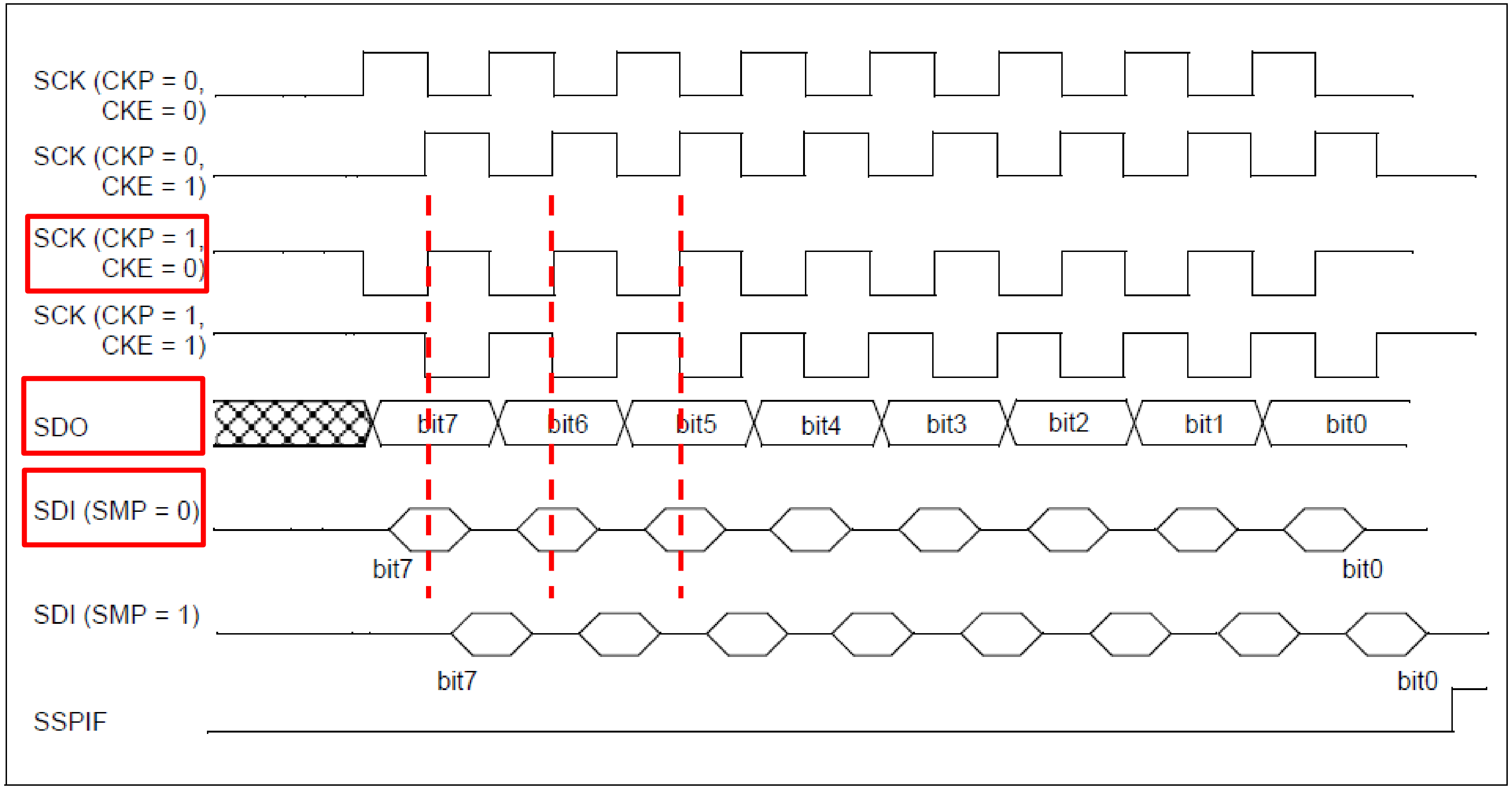
SSPCON<CKP> = 1: clock polarity - idle high

SSPSTAT<CKE> = 0: Transmit on rising edge of clock

SSPSTAT<SMP> = 0: Input data sampled at middle of data output time

Lab 7

FIGURE 9-2: SPI MODE TIMING, MASTER MODE



SDO = SPI Data Output	CKP = Clock Polarity } SSPCON
SDI = SPI Data Input	CKE = Clock Edge } SSPSTAT
SCK = SPI Clock	SMP = Sample Bit } SSPSTAT

REGISTER 9-2: SSPCON: SYNC SERIAL PORT CONTROL REGISTER (ADDRESS 14h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

$\underbrace{\text{SSPM3 SSPM2 SSPM1 SSPM0}}_{\text{SSP Mode}}$

bit 5 **SSPEN**: Synchronous Serial Port Enable bit

In SPI mode,
When enabled, these pins must be properly configured as input or output
1 = Enables serial port and configures SCK, SDO, SDI, and SS as the source of the serial port pins
0 = Disables serial port and configures these pins as I/O port pins

In I^{2}C mode,
When enabled, these pins must be properly configured as input or output
1 = Enables the serial port and configures the SDA and SCL pins as the source of the serial port pins
0 = Disables serial port and configures these pins as I/O port pins

bit 4 **CKP**: Clock Polarity Select bit

In SPI mode:
1 = Idle state for clock is a high level
0 = Idle state for clock is a low level

In I^{2}C Slave mode:
SCK release control
1 = Enable clock
0 = Holds clock low (clock stretch). (Used to ensure data setup time.)

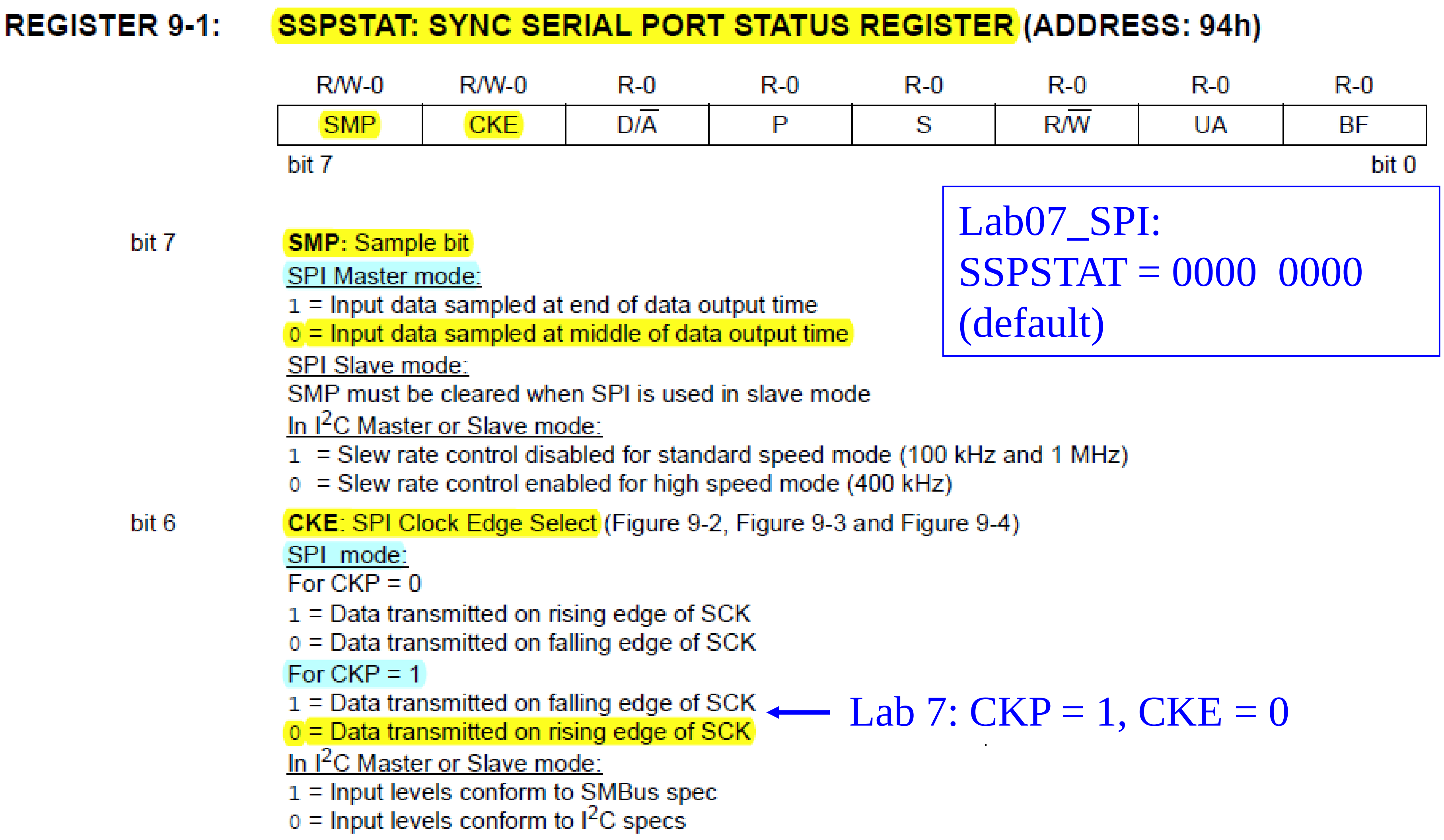
In I^{2}C Master mode:
Unused in this mode

Lab07_SPI:
SSPCON = 0011 0000

bit 3-0 **SSPM3:SSPM0**: Synchronous Serial Port Mode Select bits

0000 = SPI Master mode, clock = Fosc/4
0001 = SPI Master mode, clock = Fosc/16
0010 = SPI Master mode, clock = Fosc/64
0011 = SPI Master mode, clock = TMR2 output/2
0100 = SPI Slave mode, clock = SCK pin. $\overline{\text{SS}}$ pin control enabled.
0101 = SPI Slave mode, clock = SCK pin. $\overline{\text{SS}}$ pin control disabled. $\overline{\text{SS}}$ can be used as I/O pin.
0110 = I^{2}C Slave mode, 7-bit address
0111 = I^{2}C Slave mode, 10-bit address
1000 = I^{2}C Master mode, clock = Fosc / (4 * (SSPADD+1))
1011 = I^{2}C Firmware Controlled Master mode (slave idle)
1110 = I^{2}C Firmware Controlled Master mode, 7-bit address with START and STOP bit interrupts enabled
1111 = I^{2}C Firmware Controlled Master mode, 10-bit address with START and STOP bit interrupts enabled
1001, 1010, 1100, 1101 = Reserved

SPI Configuration



REGISTER 9-1: SSPSTAT: SYNC SERIAL PORT STATUS REGISTER (ADDRESS: 94h)

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
bit 7							bit 0

bit 7 **SMP:** Sample bit

SPI Master mode:
1 = Input data sampled at end of data output time
0 = Input data sampled at middle of data output time
SPI Slave mode:
SMP must be cleared when SPI is used in slave mode
In I^2C Master or Slave mode:
1 = Slew rate control disabled for standard speed mode (100 kHz and 1 MHz)
0 = Slew rate control enabled for high speed mode (400 kHz)

bit 6 **CKE**: SPI Clock Edge Select (Figure 9-2, Figure 9-3 and Figure 9-4)

SPI mode:
For CKP = 0
1 = Data transmitted on rising edge of SCK
0 = Data transmitted on falling edge of SCK
For CKP = 1
1 = Data transmitted on falling edge of SCK
0 = Data transmitted on rising edge of SCK ← Lab 7: CKP = 1, CKE = 0
In I^2C Master or Slave mode:
1 = Input levels conform to SMBus spec
0 = Input levels conform to I^2C specs

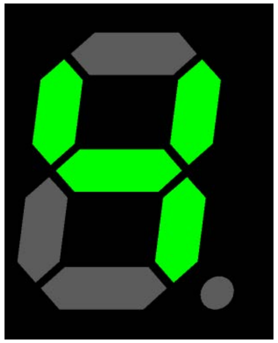
Lab07_SPI:
SSPSTAT = 0000 0000
(default)

Data sheet p. 67

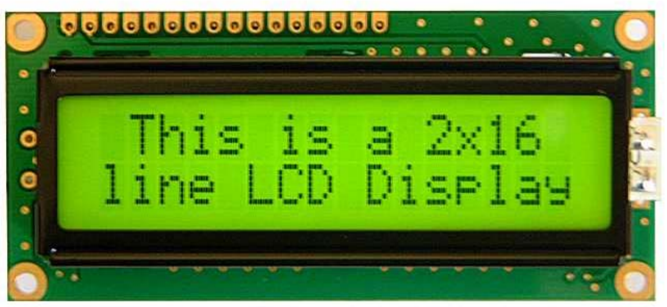
Lab 7 Outline

LED Displays and LCDs

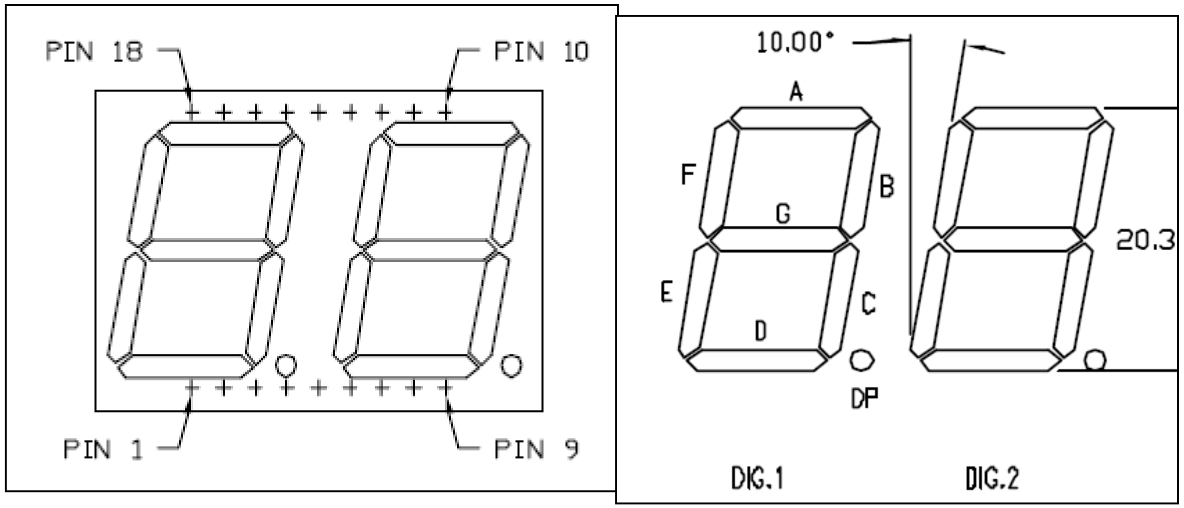
1. LED = light emitting diode
   a) Simple interface
   b) Inexpensive
   c)



2. LCD = liquid crystal display
   a) More complicated, extra cost.
   b) Built-in drivers convert ASCII code to display characters.
   c) Flexible

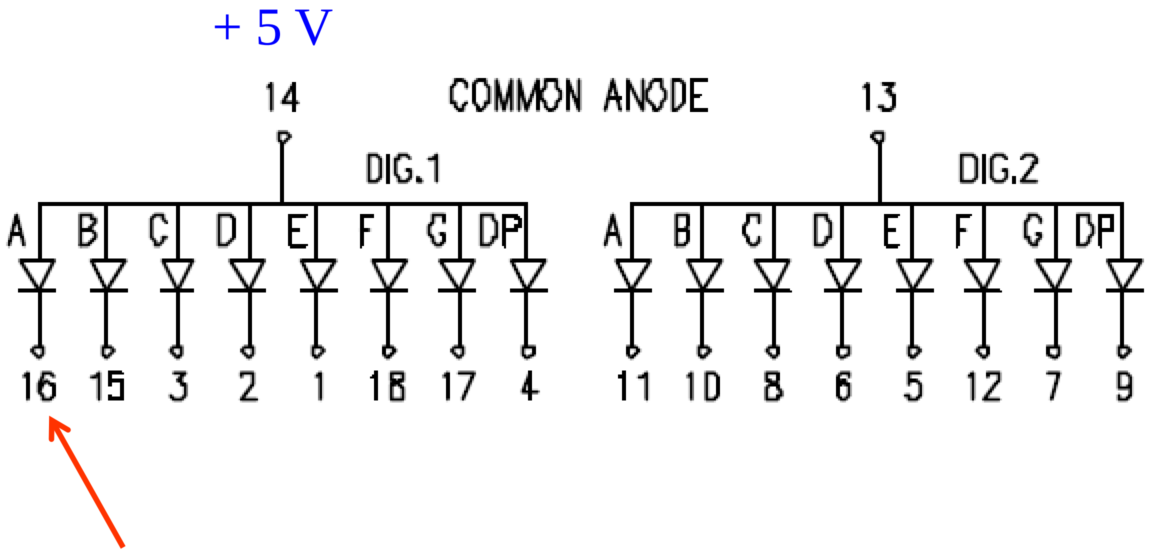


Your kit may have a Lumex or Avago 7-segment, 2-digit LED display



LDD-A814RI
LED display
Lumex Inc.

HDSP-521E
LED display
Avago Inc.

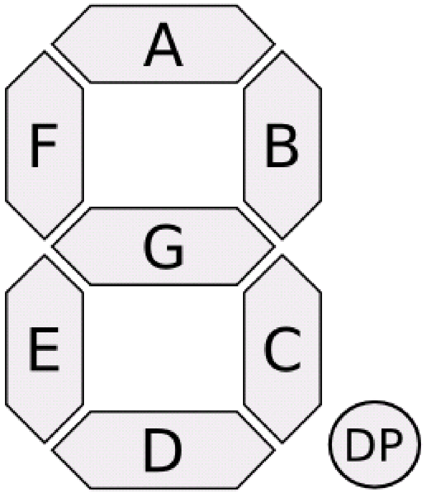


+ 5 V = off

0 V = on

Active low

7-Segment LEDs

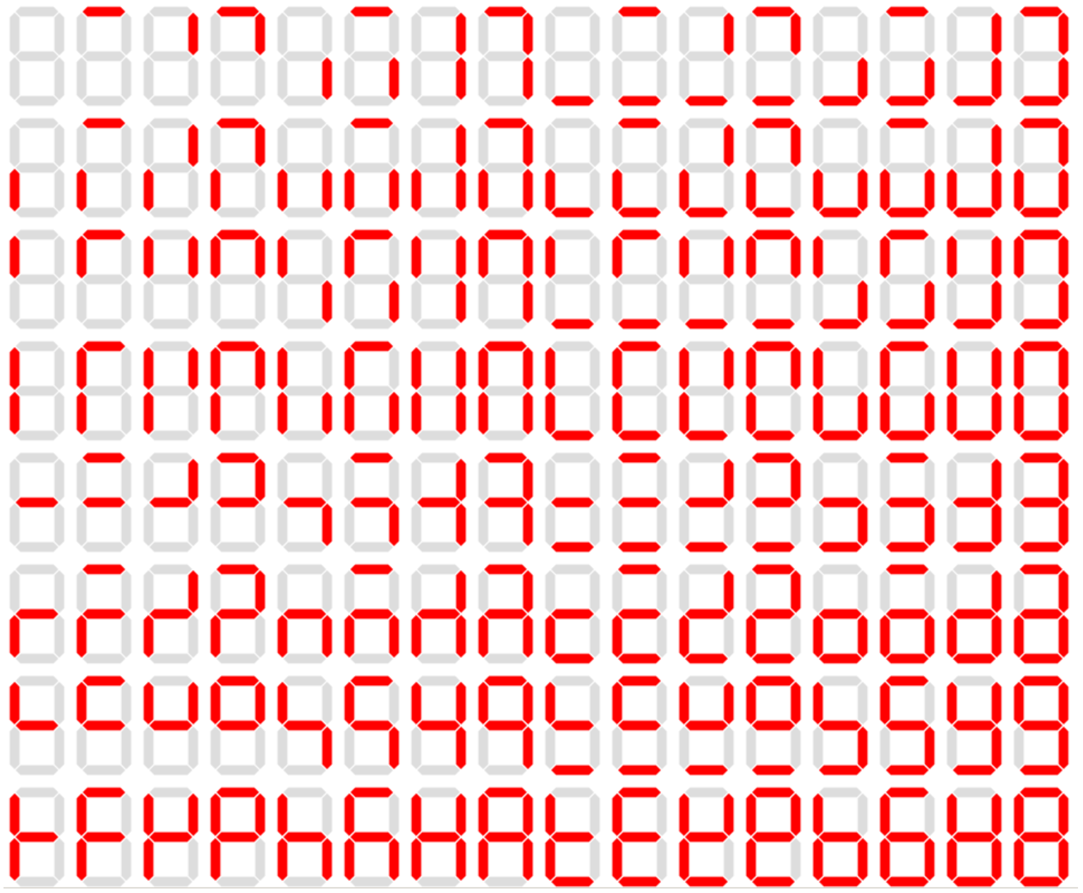


How many different symbols are possible?

$$\# \text{ symbols} = C(7,0) + C(7,1) + C(7,2) + \cdots + C(7,7)$$
$$= \sum_{r=0}^{7} \frac{7!}{r!(7-r)!} = 2^7 = 128$$

7-Segment LED Symbol Table

128
symbols

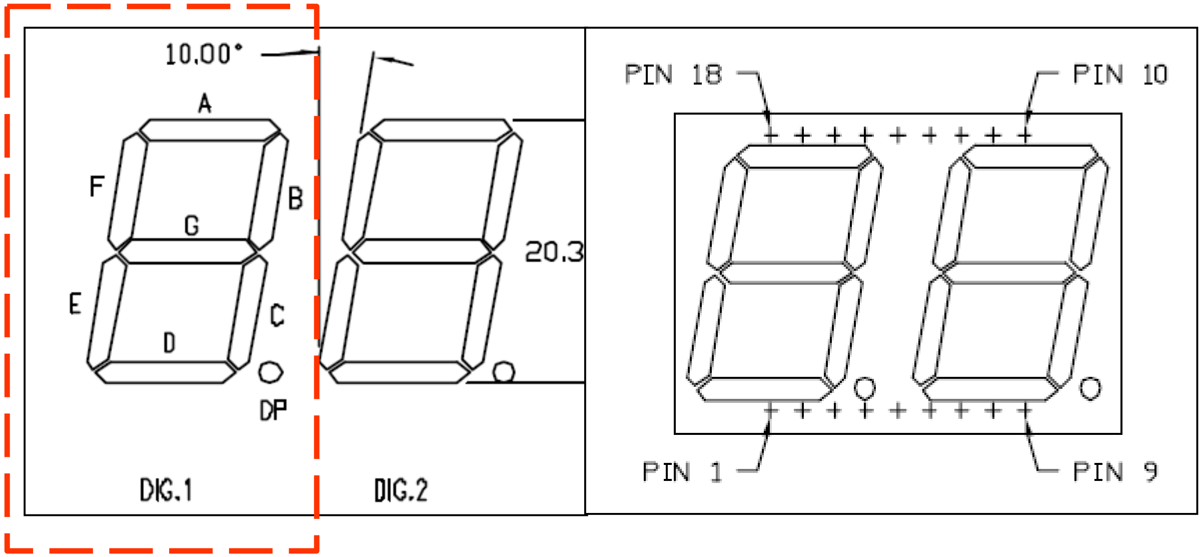


Encoding 7-Segment Decimal Digits

Binary Code Bit	7	6	5	4	3	2	1	0
Segment	DP	G	F	E	D	C	B	A
LED Pin (DIG.1)	4	17	18	1	2	3	15	16

Active low: set segment bit = 0 to turn on.

To turn on Segment A, set LED Pin-16 low.



Display Digit	**Binary Code**	**Hex Code**
0	1100 0000	0xC0
1	1111 1001	0xF9
2	1010 0100	0xA4
etc		

Lab 7 Outline

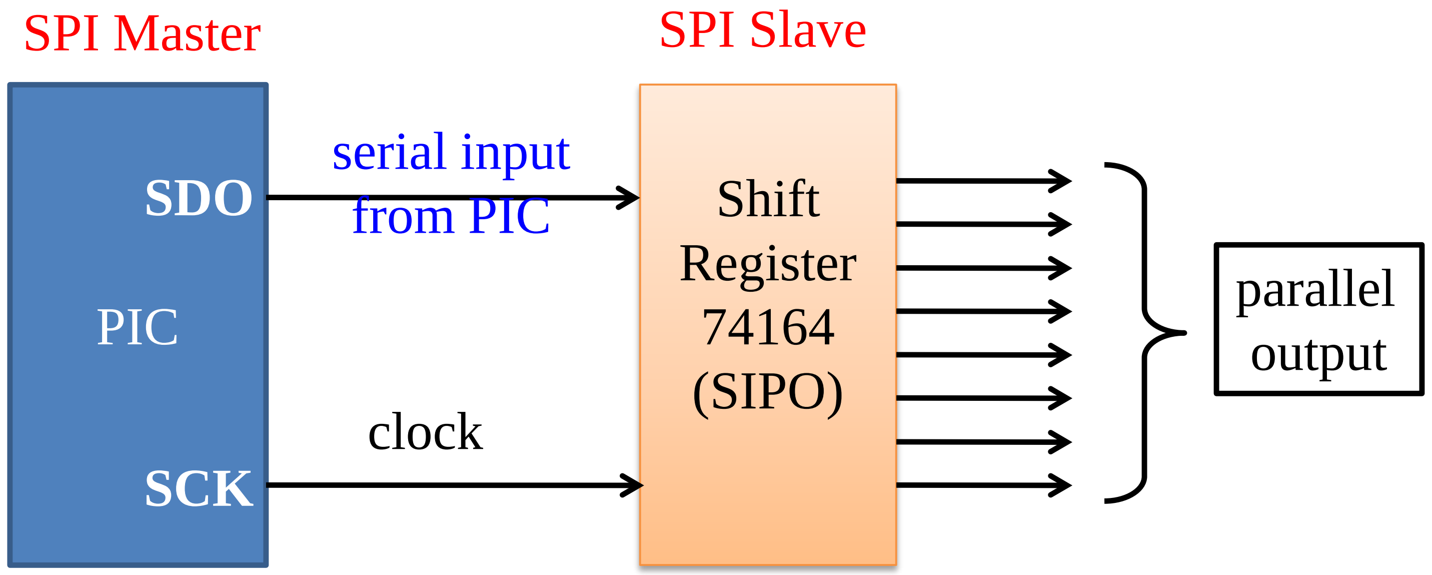
1. Master Synchronous Serial Port – SPI Mode
2. 7-Segment LED Display
3. **Shift Registers**
4. lab07_SPI Setup
5. Keypad Scanning
6. lab07_keypad Setup
7. Finite Arithmetic
8. Watchdog Timer

SPI Shift Register

1. The 74164 shift register converts serial input to parallel output (SIPO).

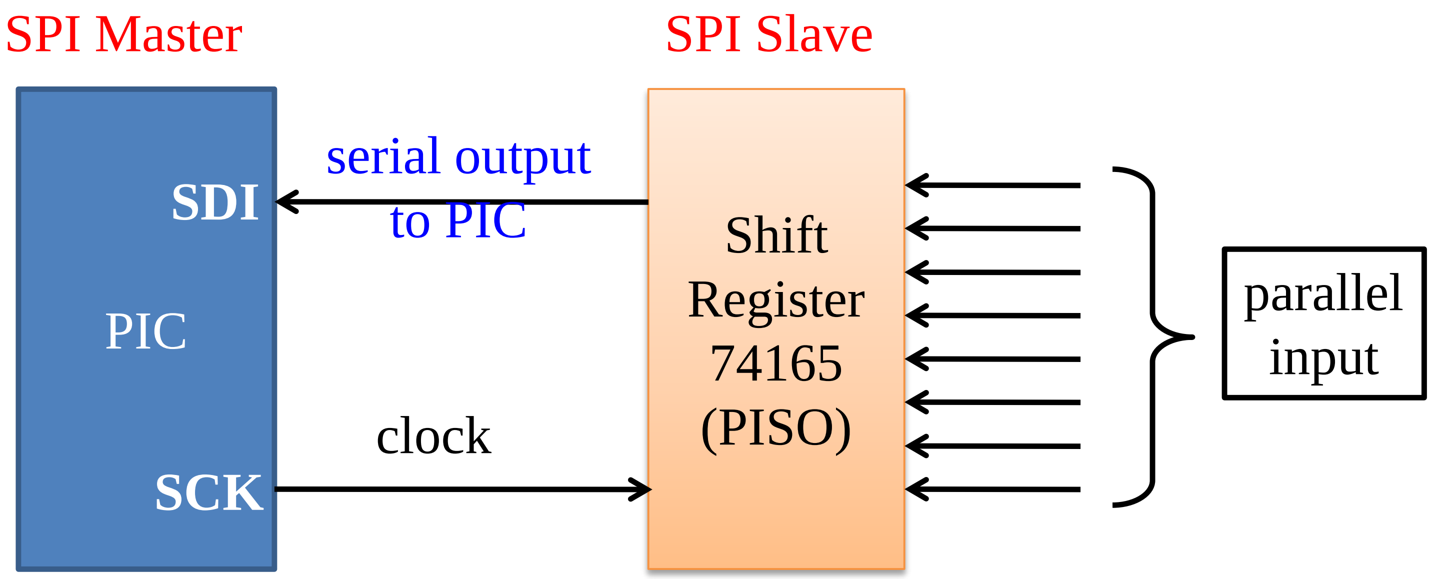
2. The PIC can be combined with a 74164 to convert a serial output to a parallel output.

3. This effectively expands the number of digital output pins on the PIC.



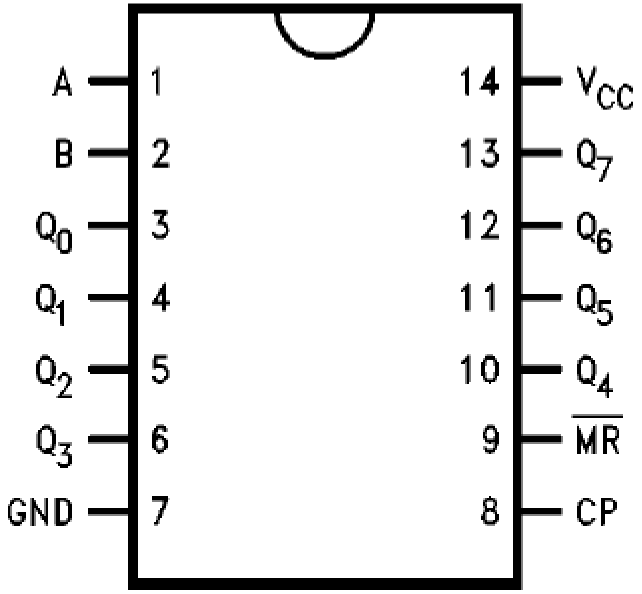
SPI Shift Register

1. The 74165 shift register converts parallel input to serial output (PISO).
2. The PIC can be combined with a 74165 to convert a parallel input to a serial input.
3. This effectively expands the number of digital input pins on the PIC.

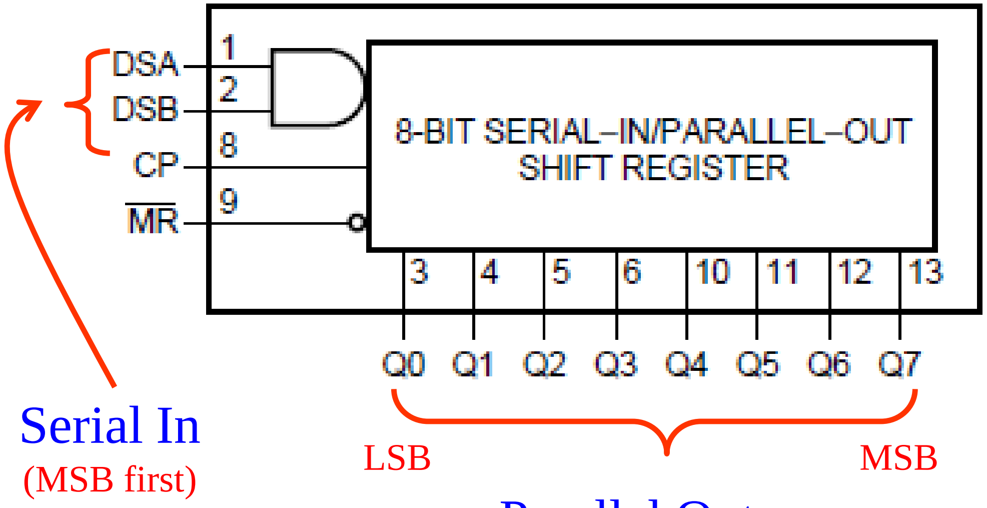


SIPO Shift Register (SPI Slave)

74164
PIN Diagram



74164
Logic Diagram



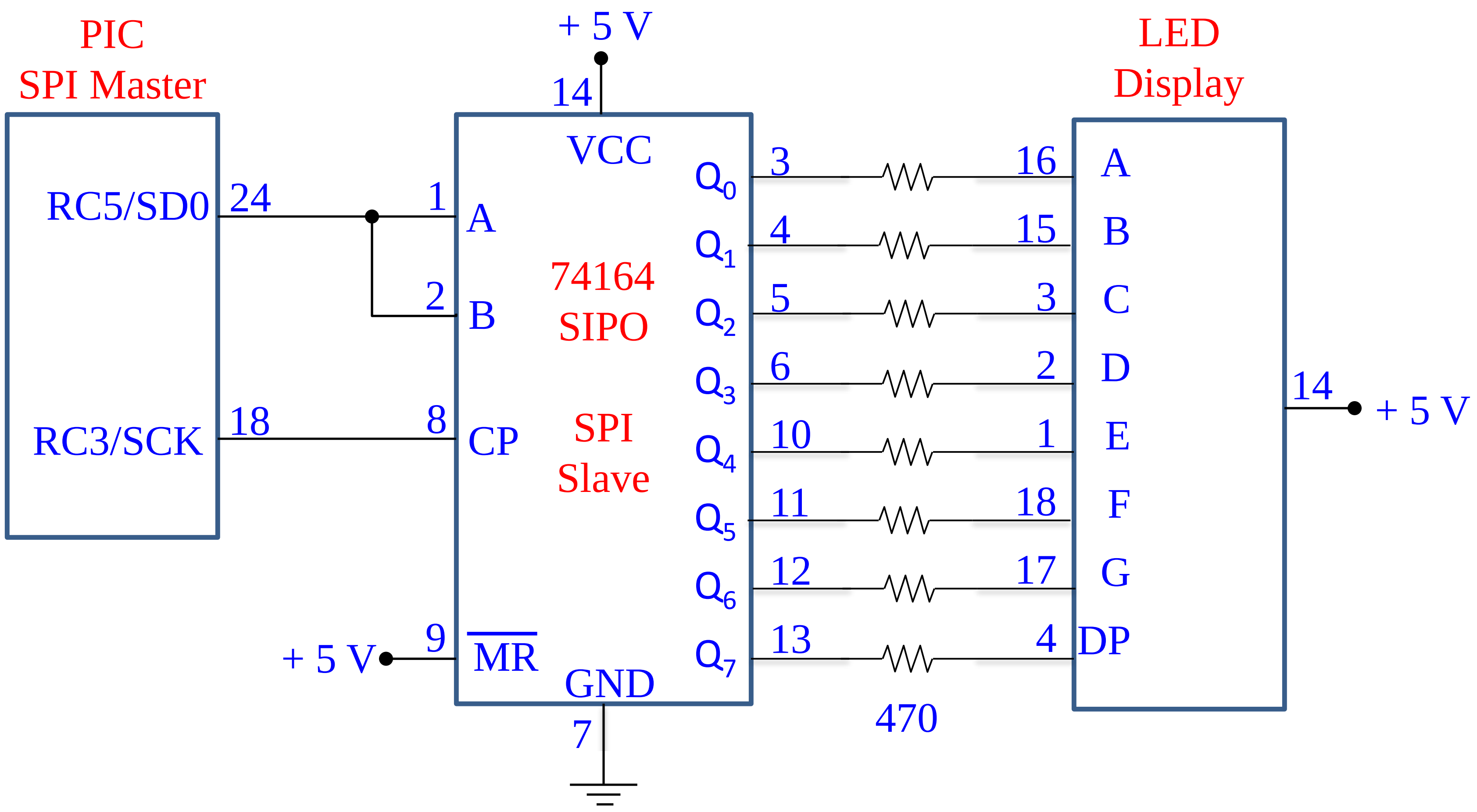
Parallel Out

Pin Names	Description
A, B	Data Inputs
CP	Clock Pulse Input (Active Rising Edge)
$\overline{MR}$	Master Reset Input (Active Low)
$Q_0–Q_7$	Outputs

Lab 7 Outline

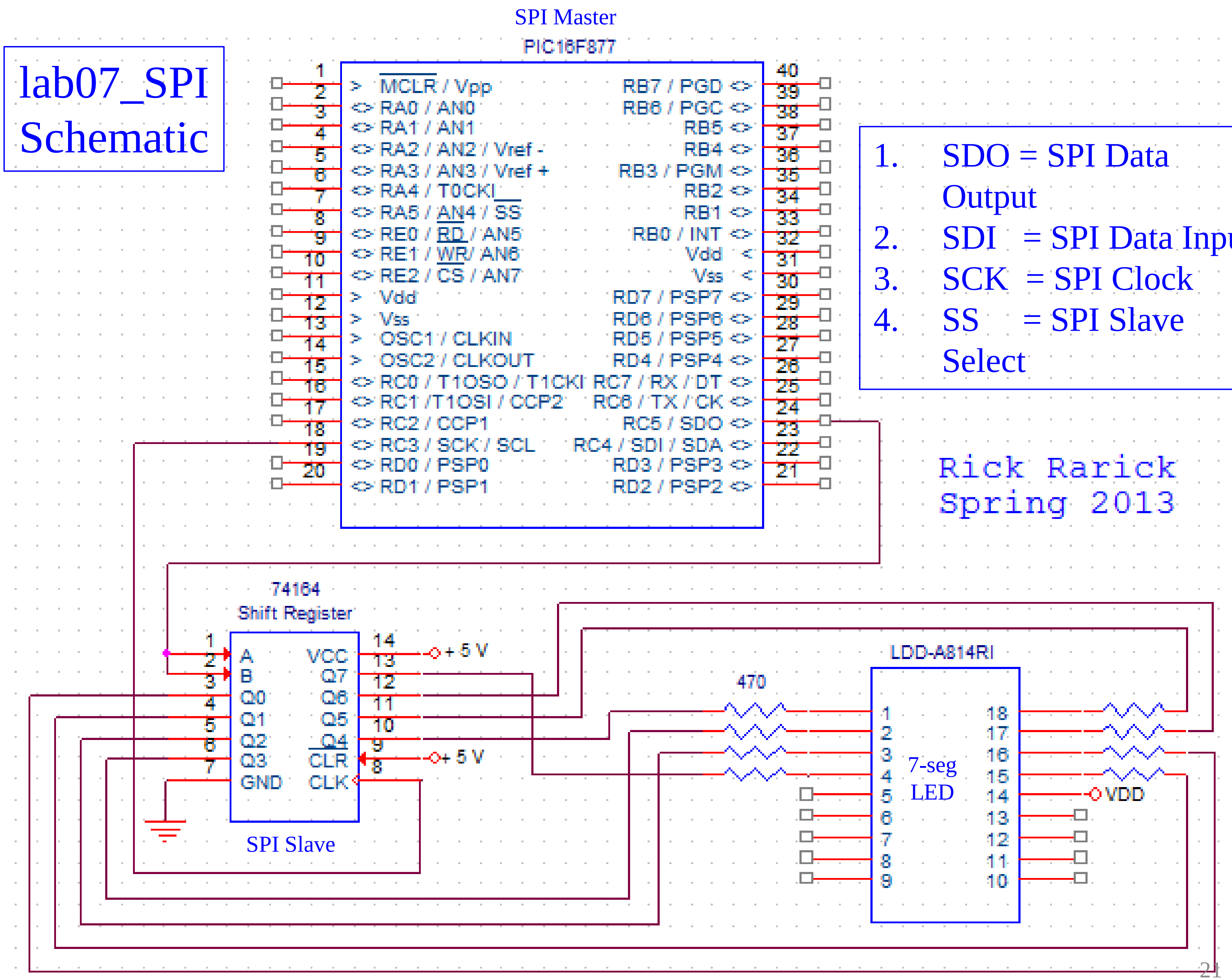
1. Master Synchronous Serial Port – SPI Mode
2. 7-Segment LED Display
3. Shift Registers
4. **lab07_SPI Setup**
5. Keypad Scanning
6. lab07_keypad Setup
7. Finite Arithmetic
8. Watchdog Timer

lab07_SPI Logical Diagram



SIPO – Serial In, Parallel Out

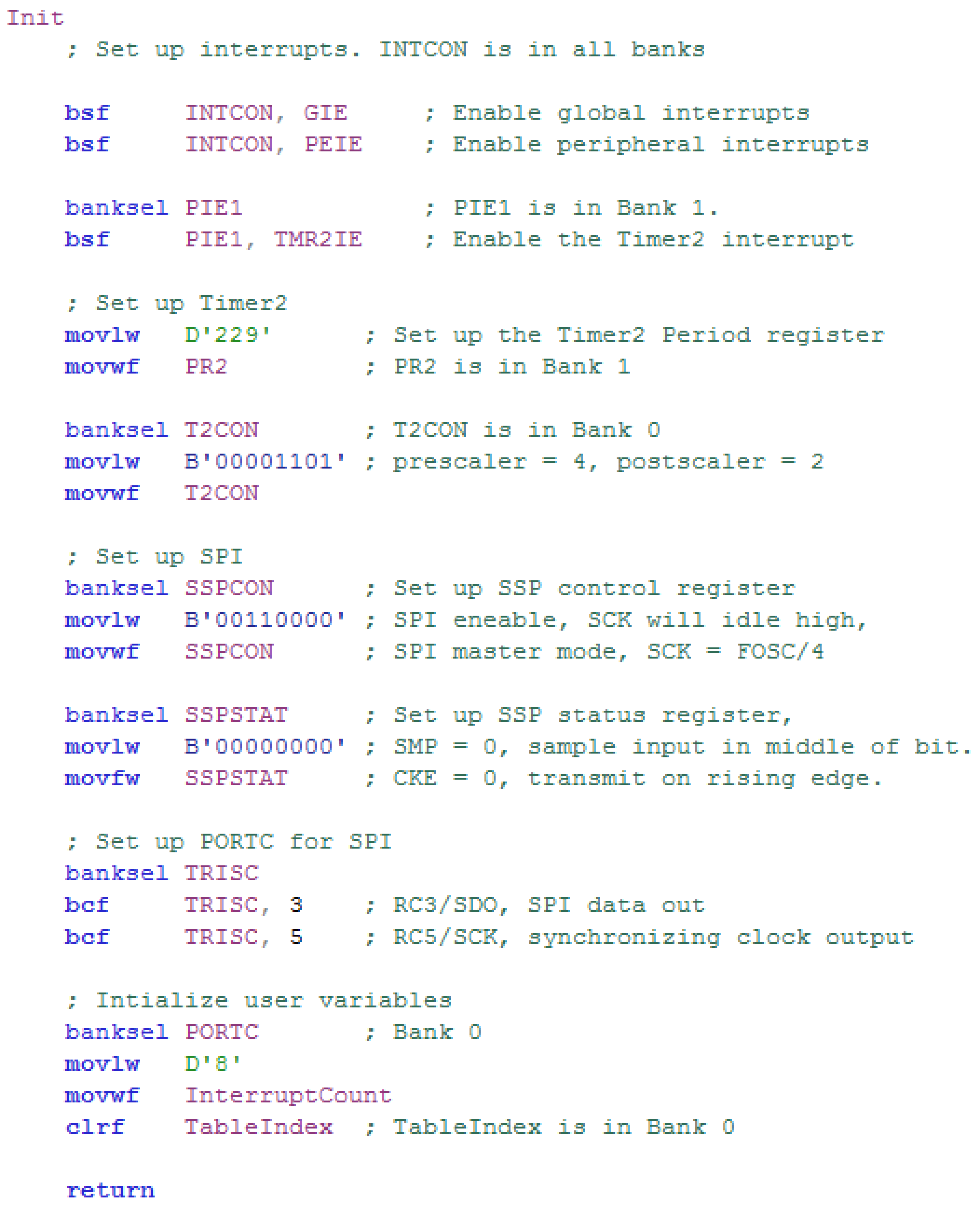
lab07_SPI Schematic



1. SDO = SPI Data Output
2. SDI = SPI Data Input
3. SCK = SPI Clock
4. SS = SPI Slave Select

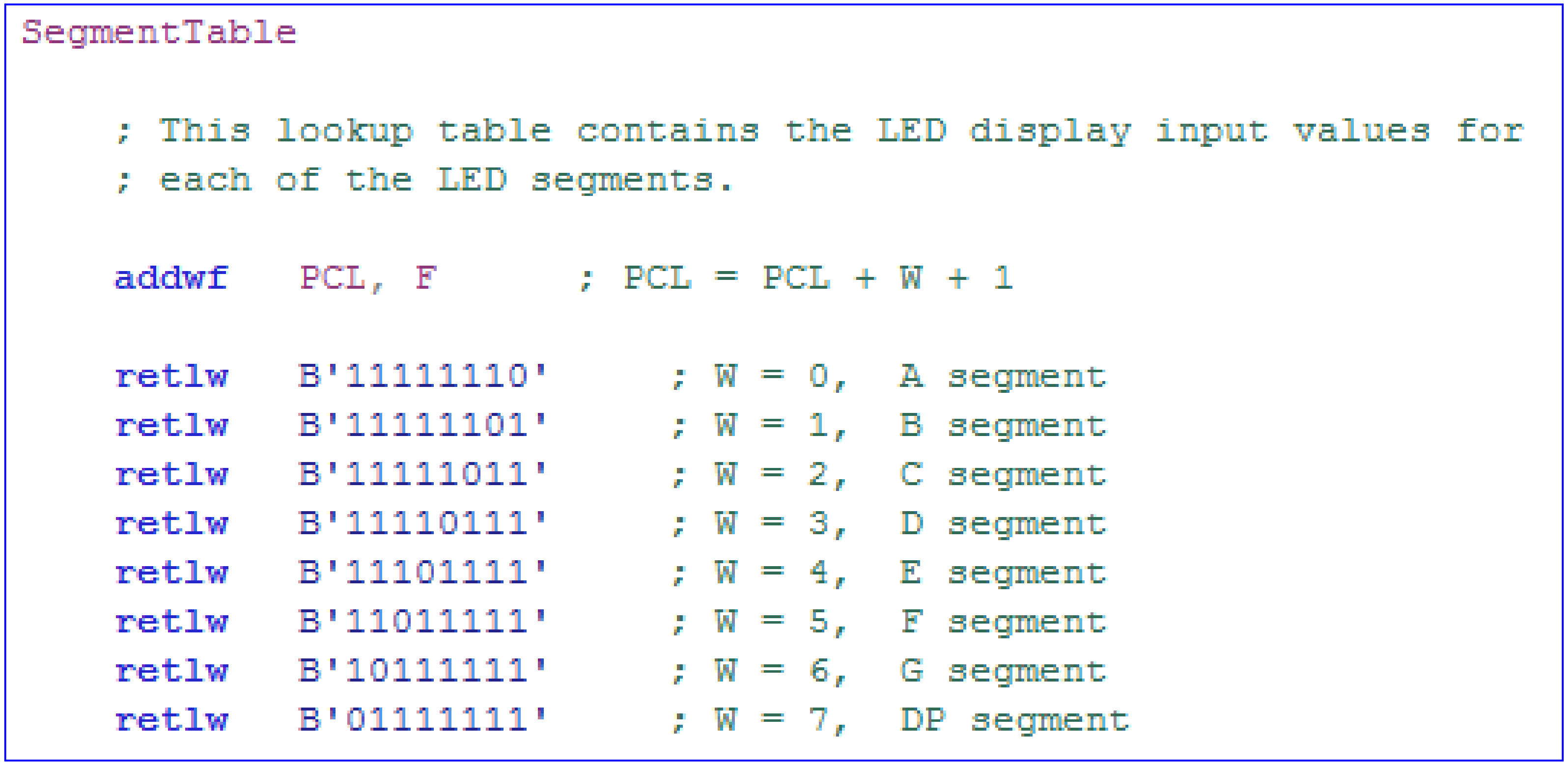
Rick Rarick
Spring 2013

lab07_SPI.asm



```
Init
    ; Set up interrupts. INTCON is in all banks

    bsf     INTCON, GIE      ; Enable global interrupts
    bsf     INTCON, PEIE     ; Enable peripheral interrupts

    banksel PIE1             ; PIE1 is in Bank 1.
    bsf     PIE1, TMR2IE     ; Enable the Timer2 interrupt

    ; Set up Timer2
    movlw   D'229'       ; Set up the Timer2 Period register
    movwf   PR2          ; PR2 is in Bank 1

    banksel T2CON        ; T2CON is in Bank 0
    movlw   B'00001101' ; prescaler = 4, postscaler = 2
    movwf   T2CON

    ; Set up SPI
    banksel SSPCON       ; Set up SSP control register
    movlw   B'00110000' ; SPI eneable, SCK will idle high,
    movwf   SSPCON       ; SPI master mode, SCK = FOSC/4

    banksel SSPSTAT      ; Set up SSP status register,
    movlw   B'00000000' ; SMP = 0, sample input in middle of bit.
    movfw   SSPSTAT      ; CKE = 0, transmit on rising edge.

    ; Set up PORTC for SPI
    banksel TRISC
    bcf     TRISC, 3     ; RC3/SDO, SPI data out
    bcf     TRISC, 5     ; RC5/SCK, synchronizing clock output

    ; Intialize user variables
    banksel PORTC        ; Bank 0
    movlw   D'8'
    movwf   InterruptCount
    clrf    TableIndex   ; TableIndex is in Bank 0

    return
```

lab07_SPI.asm



```
SegmentTable

    ; This lookup table contains the LED display input values for
    ; each of the LED segments.

    addwf   PCL, F       ; PCL = PCL + W + 1

    retlw   B'11111110'     ; W = 0,  A segment
    retlw   B'11111101'     ; W = 1,  B segment
    retlw   B'11111011'     ; W = 2,  C segment
    retlw   B'11110111'     ; W = 3,  D segment
    retlw   B'11101111'     ; W = 4,  E segment
    retlw   B'11011111'     ; W = 5,  F segment
    retlw   B'10111111'     ; W = 6,  G segment
    retlw   B'01111111'     ; W = 7,  DP segment
```

lab07_SPI.asm

```
Main
    call    Init                ; Initialize everything

MainLoop

    call    Delay_500ms

    movf    TableIndex, W       ; W = TableIndex

    call    SegmentTable        ; W now contains the index for the table.
                                ; Get the segment entry from the table and
                                ; and return it in W.

    movwf   SSPBUF              ; SSPBUF = W

    call    Delay_500ms         ; Wait for transmission

    incf    TableIndex, F       ; TableIndex = TableIndex + 1

    movf    TableIndex, W       ; W = TableIndex

    sublw   D'8'                ; W = 8 - W. If W = 0, Z = 1, otherwise,
                                ; Z = 0.

    btfss   STATUS, Z           ; Skip next if Z = 1 (TableIndex = 8)

    goto    MainLoop            ; If we reach this instruction, Z = 0
                                ; (TableIndex < 8), so get the
                                ; next table entry for the display.

    clrf    TableIndex          ; If Z = 1, reset TableIndex = 0

    goto    MainLoop
```

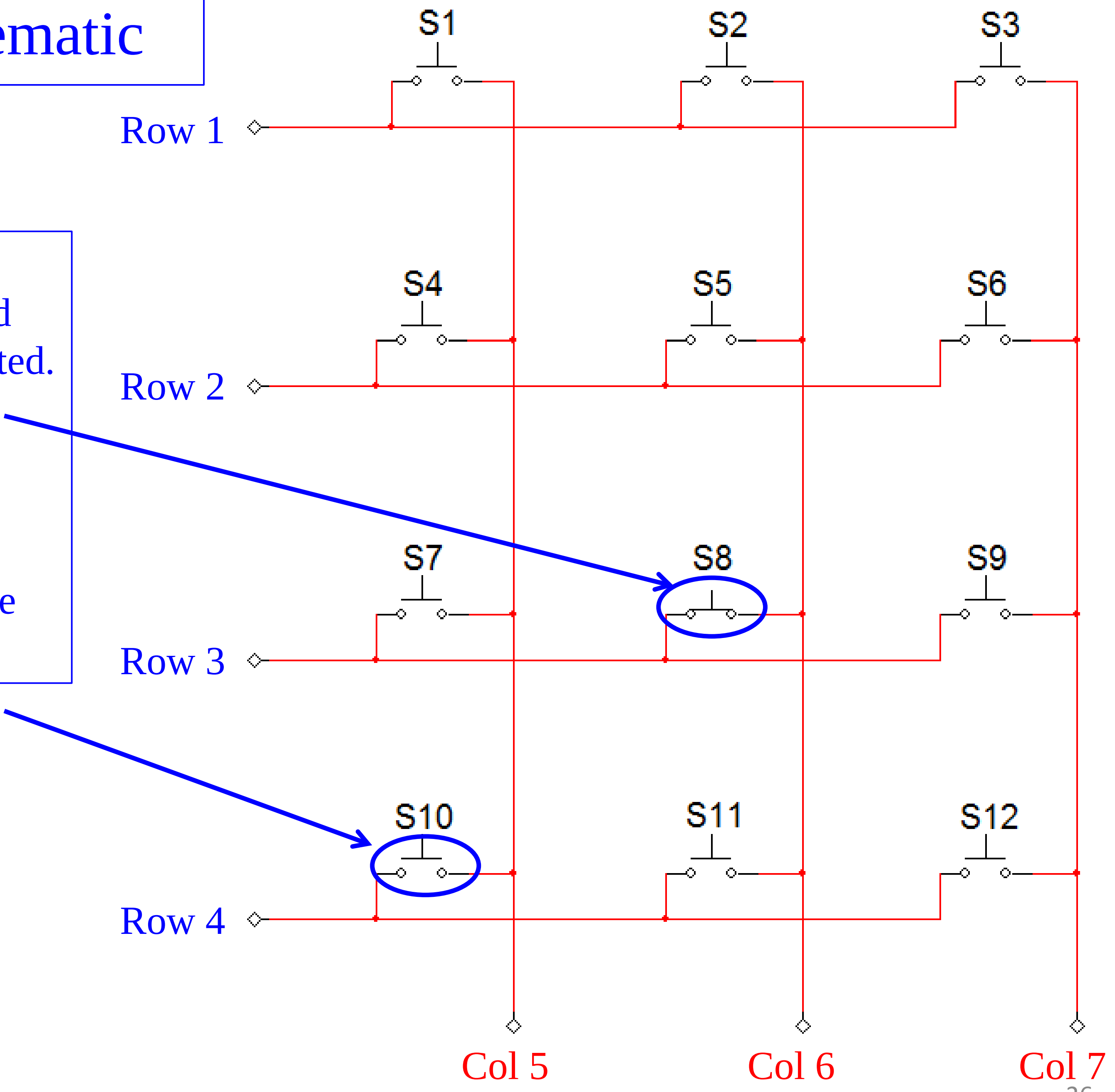
Lab 7 Outline

1. Master Synchronous Serial Port – SPI Mode
2. 7-Segment LED Display
3. Shift Registers
4. lab07_SPI Setup
5. **Keypad Scanning**
6. lab07_keypad Setup
7. Finite Arithmetic
8. Watchdog Timer

Keypad Schematic

1. When a key is pressed, its row and column are connected.

2. When no key is pressed, rows and columns are not connected. They are open circuits.

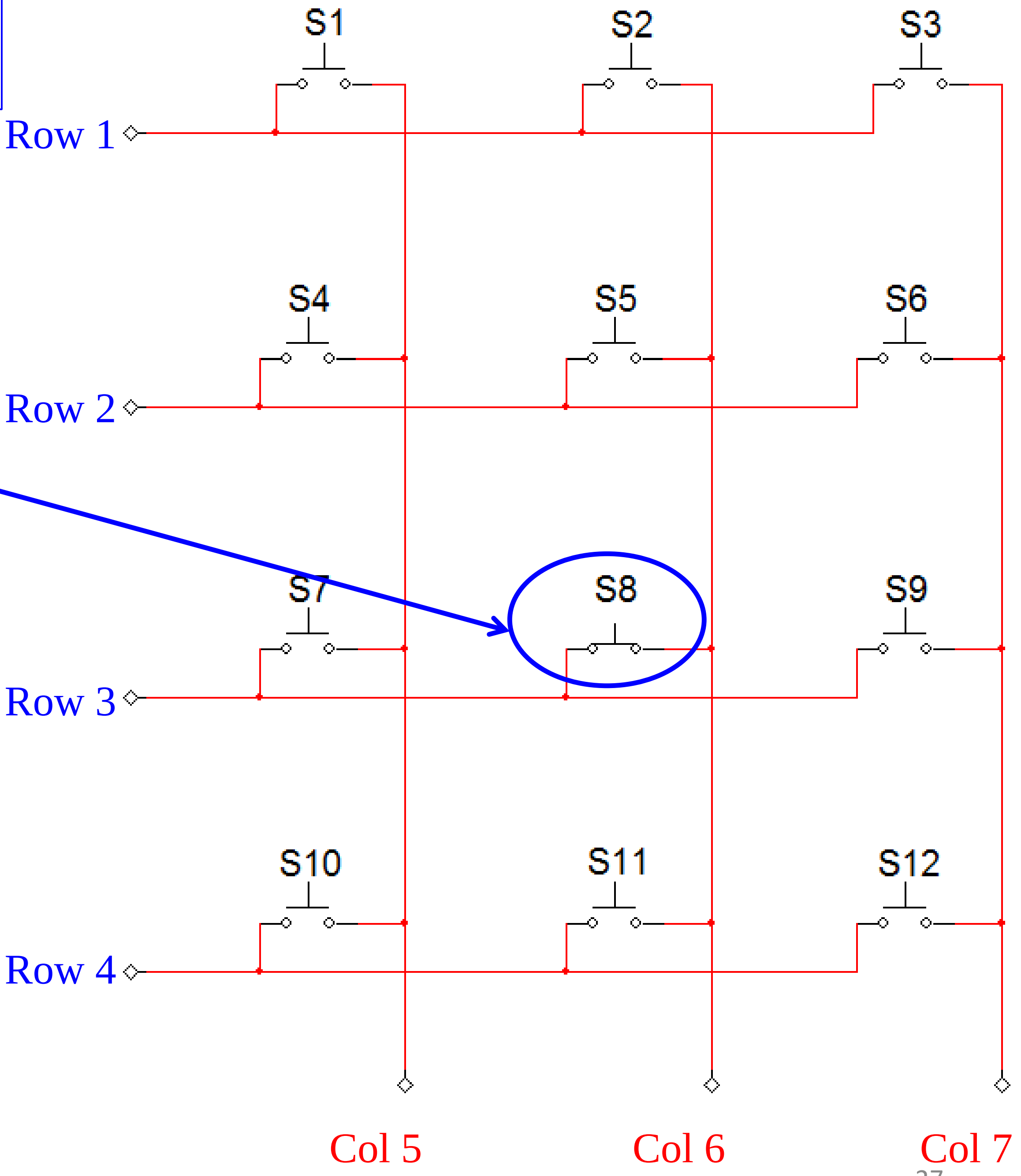


Key Formula

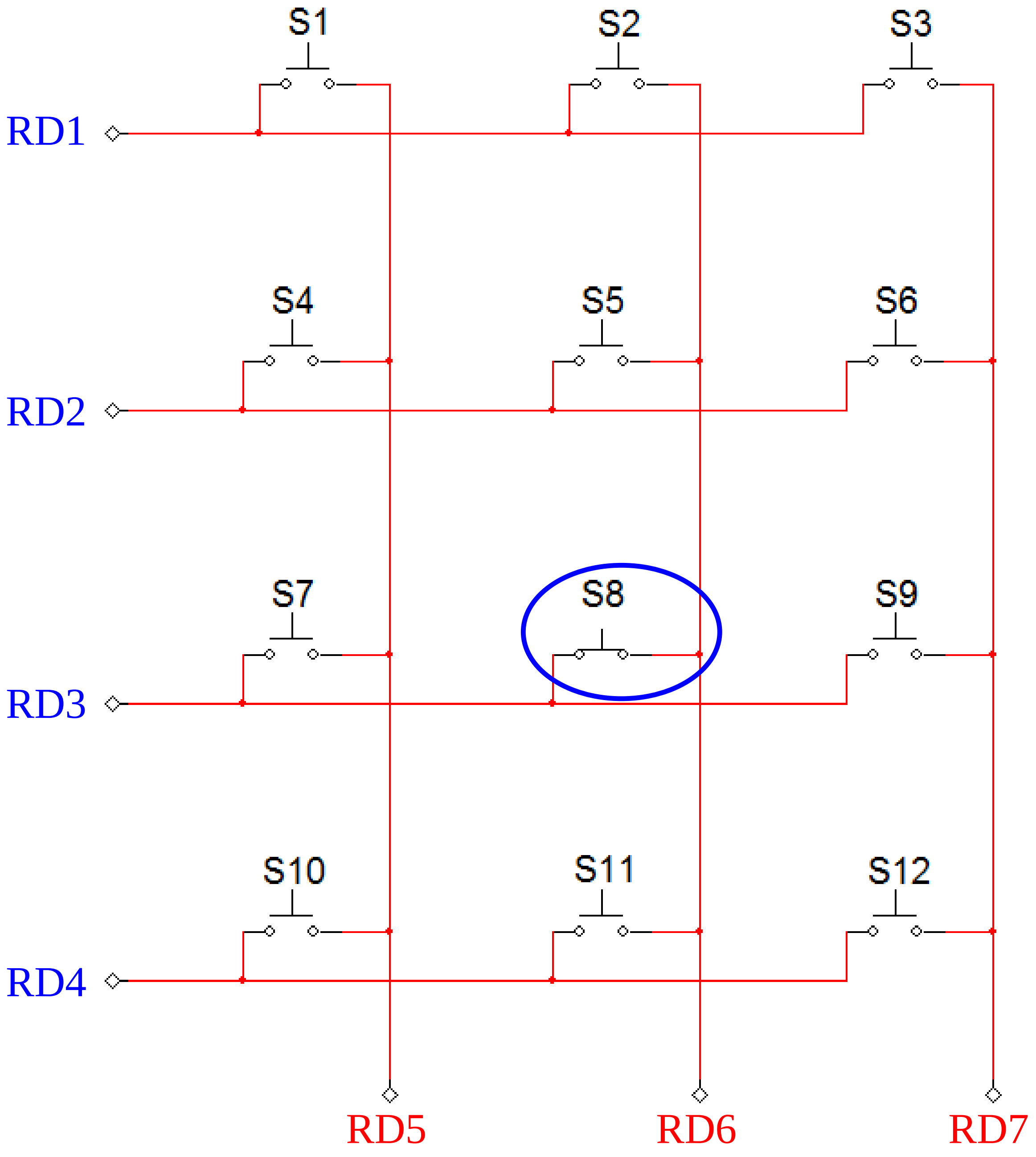
1. When Key 8 is pressed, **Row 3** and **Col 6** are connected.

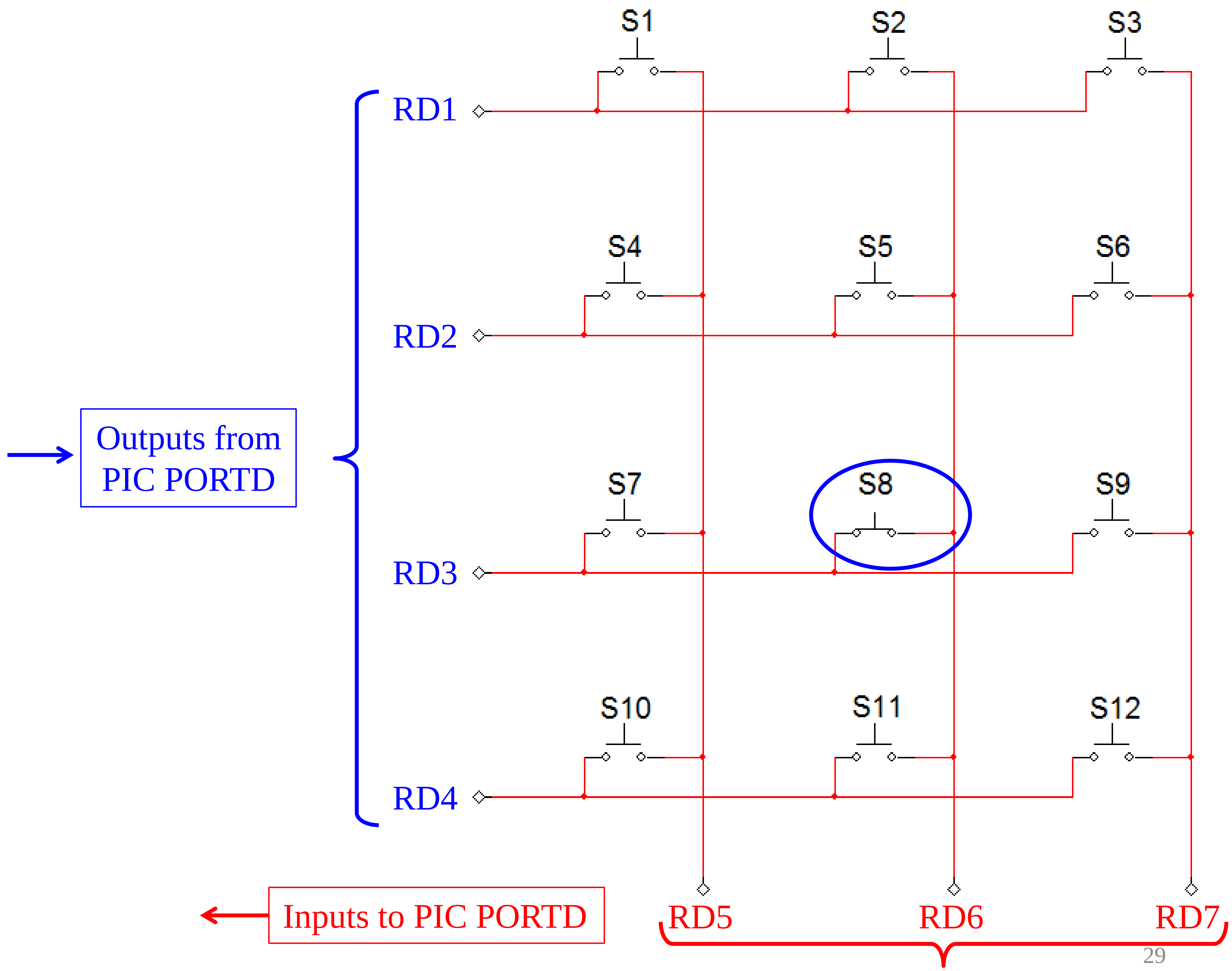
2. Key = 3 * (Row – 1) + (Col – 4)
   = 3 * (3 – 1) + (6 – 4)
   = 8

3. Key = 3 * Row + Col – 7



1. So, when a key is pressed, we need to determine the key's row and column.
2. How can we use the PIC to determine the row and column?
3. Connect rows and columns to PORTD.
4. Clear TRISD<RD4:RD1> for outputs.
5. Set TRISD<RD7:RD5> for inputs.

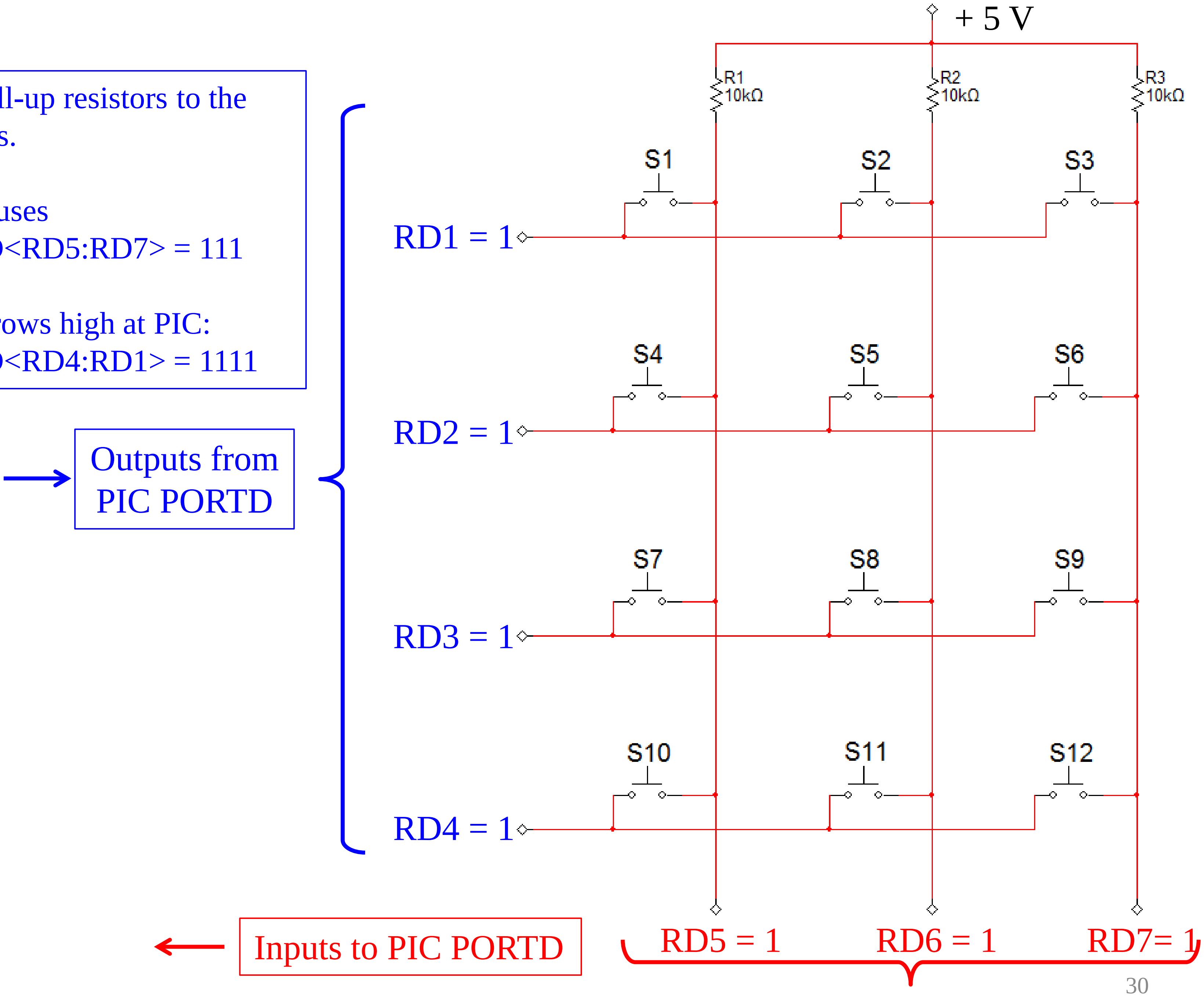



S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
S11
S12
RD1
RD2
RD3
RD4
Outputs from PIC PORTD
Inputs to PIC PORTD
RD5
RD6
RD7

1. Add pull-up resistors to the columns.

2. This causes PORTD<RD5:RD7> = 111

3. Set all rows high at PIC: PORTD<RD4:RD1> = 1111



1. Now cycle continuously through the rows, setting each one low, one at a time.

2. For example, set RD3 = 0.

3. Note that this has no effect on the column values since no key is pressed.

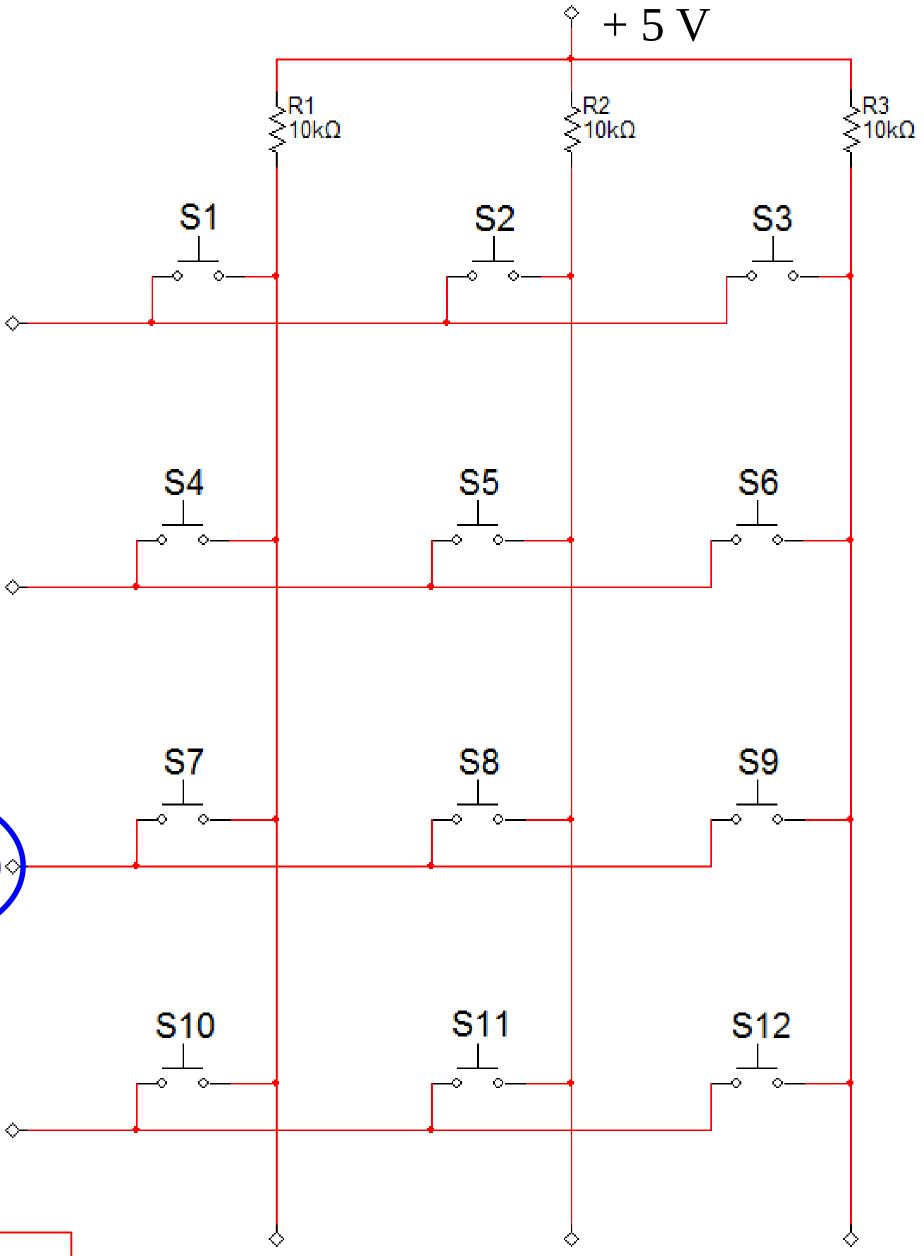
Outputs from PIC PORTD

RD1 = 1

RD2 = 1

RD3 = 0

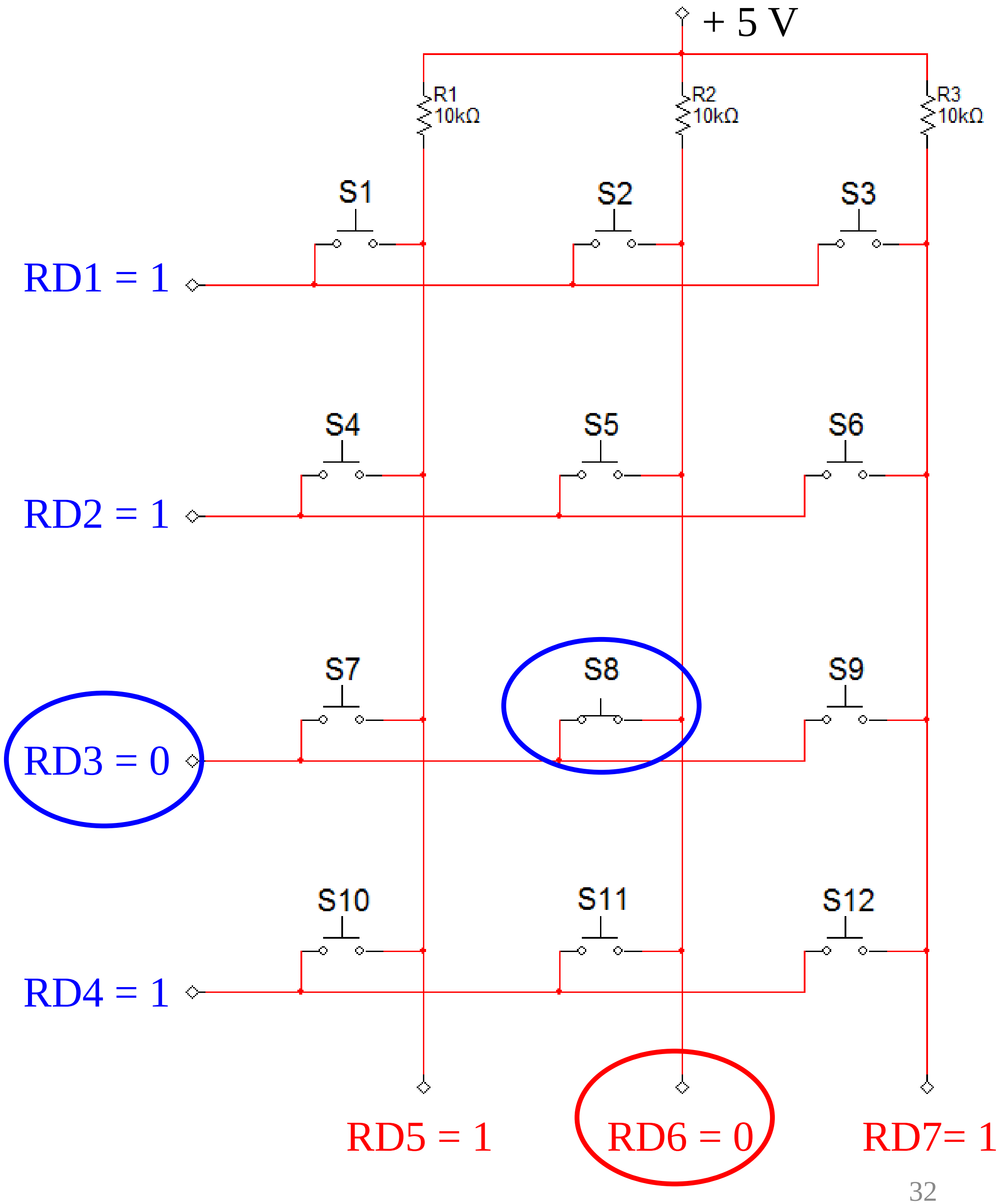
RD4 = 1



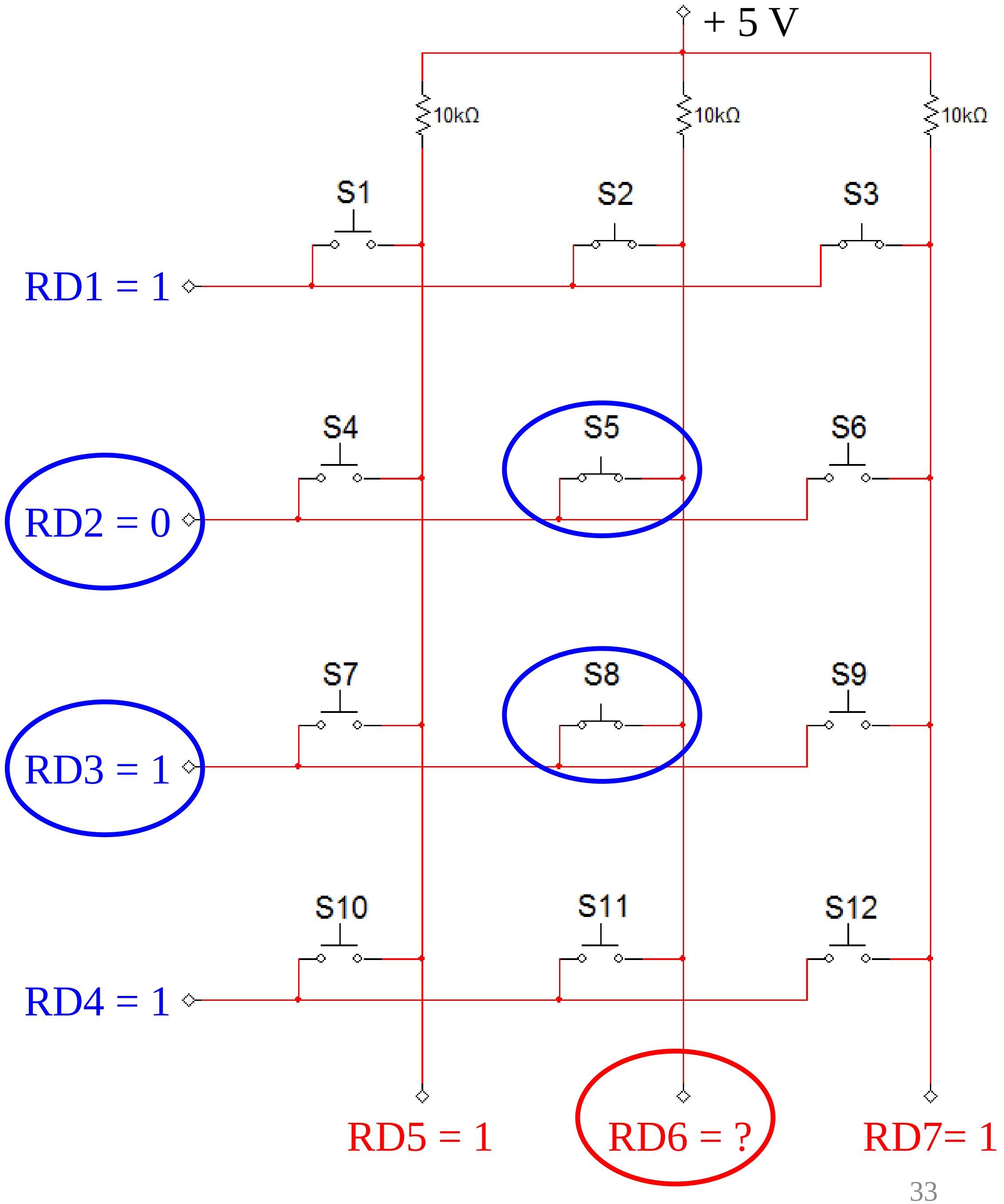
Inputs to PIC PORTD

RD5 = 1 RD6 = 1 RD7= 1

1. Each time a row is set to 0, the PIC reads the three columns.
2. If Key 8 is pressed, then, when the cycle reaches RD3 = 0, it will force RD6 = 0.
3. So the PIC will detect that a key was pressed on Row 3 and Col 6.



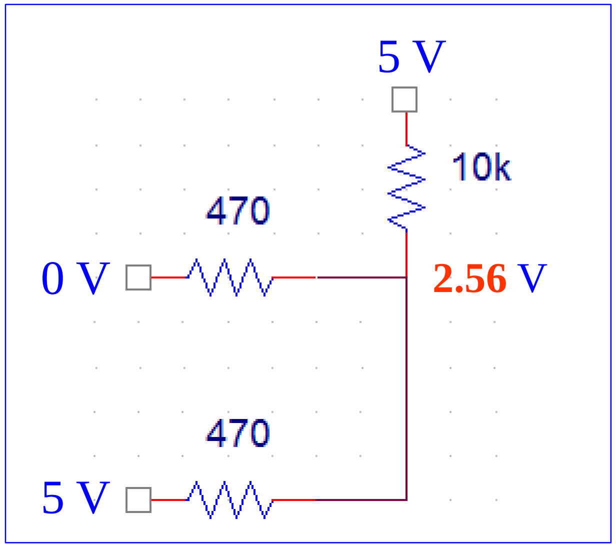
1. However, there is a problem with the this circuit.
2. If Key 5 and Key 8 are pressed at the same time, then, when the cycle reaches RD2 = 0, there will be a direct short between RD2 = 0 and RD3 = 1.
3. This may damage the PIC and/or keypad.

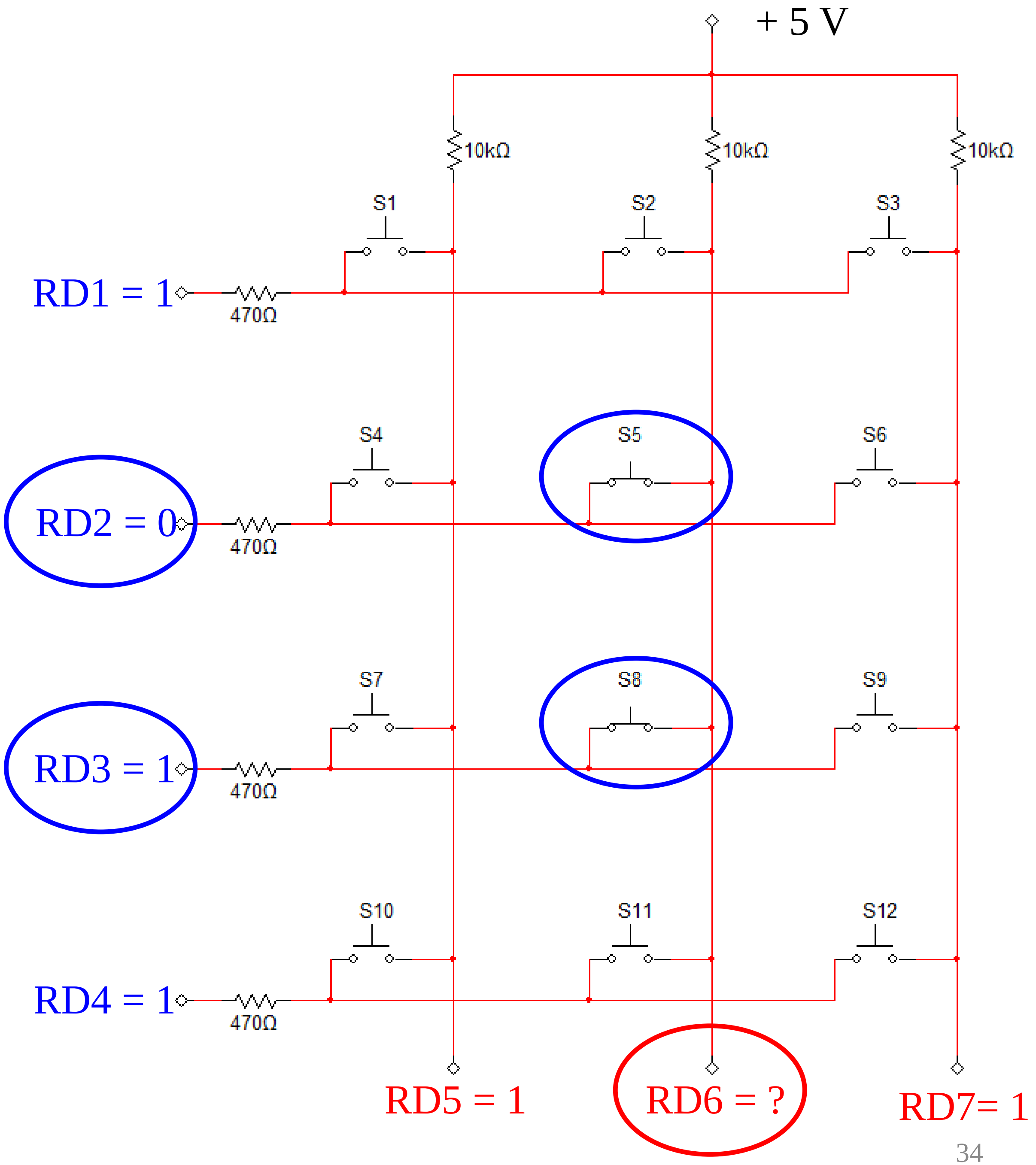


1. In order to protect the circuit, current-limiting resistors are added in to the rows of the keypad.

2. If Key 5 and Key 8 are pressed at the same time, then, when the cycle reaches RD2 = 0, the voltage at RD6 will be **2.56** volts.

3. So the current flowing into RD2 will be 2.56/470 = 5.4 mA.



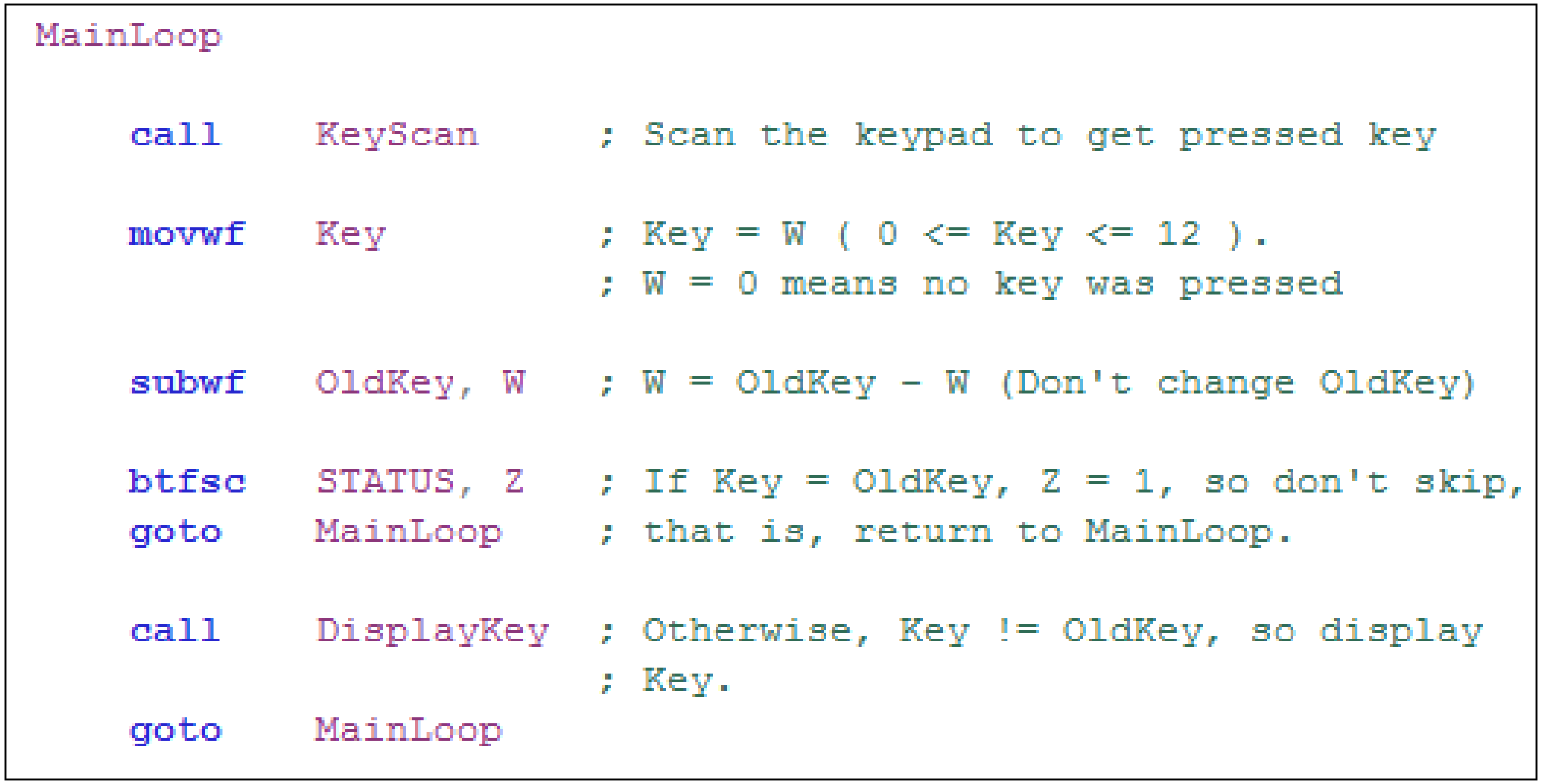


Keypad Scanning Pseudo-Code

```
Repeat indefinitely:
for i = 1 to 4

   set row(i) = 0, all other rows high.

   for j = 5 to 7

      if col(j) = 0, then we know the switch at
      row(i), col(j) is closed. So
      key = 3 * row(i) + col(j) - 7
      endif

   next j
next i
```

The assembly code for this nested for-loop is in lab07_keypad.asm

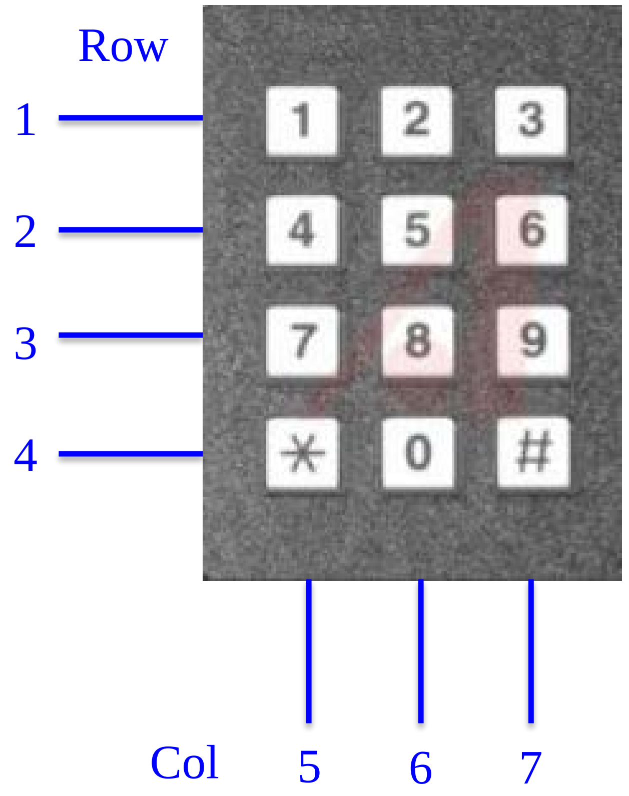
lab07_keypad.asm

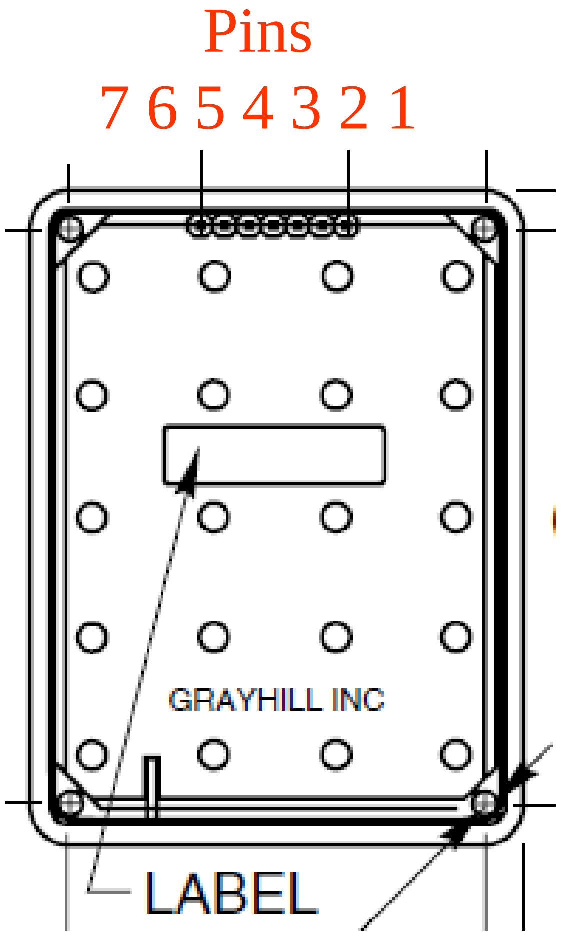


```
MainLoop

    call    KeyScan      ; Scan the keypad to get pressed key

    movwf   Key          ; Key = W ( 0 <= Key <= 12 ).
                         ; W = 0 means no key was pressed

    subwf   OldKey, W    ; W = OldKey - W (Don't change OldKey)

    btfsc   STATUS, Z    ; If Key = OldKey, Z = 1, so don't skip,
    goto    MainLoop     ; that is, return to MainLoop.

    call    DisplayKey   ; Otherwise, Key != OldKey, so display
                         ; Key.
    goto    MainLoop
```

Lab 7 Outline

Keypad Pin Numbers

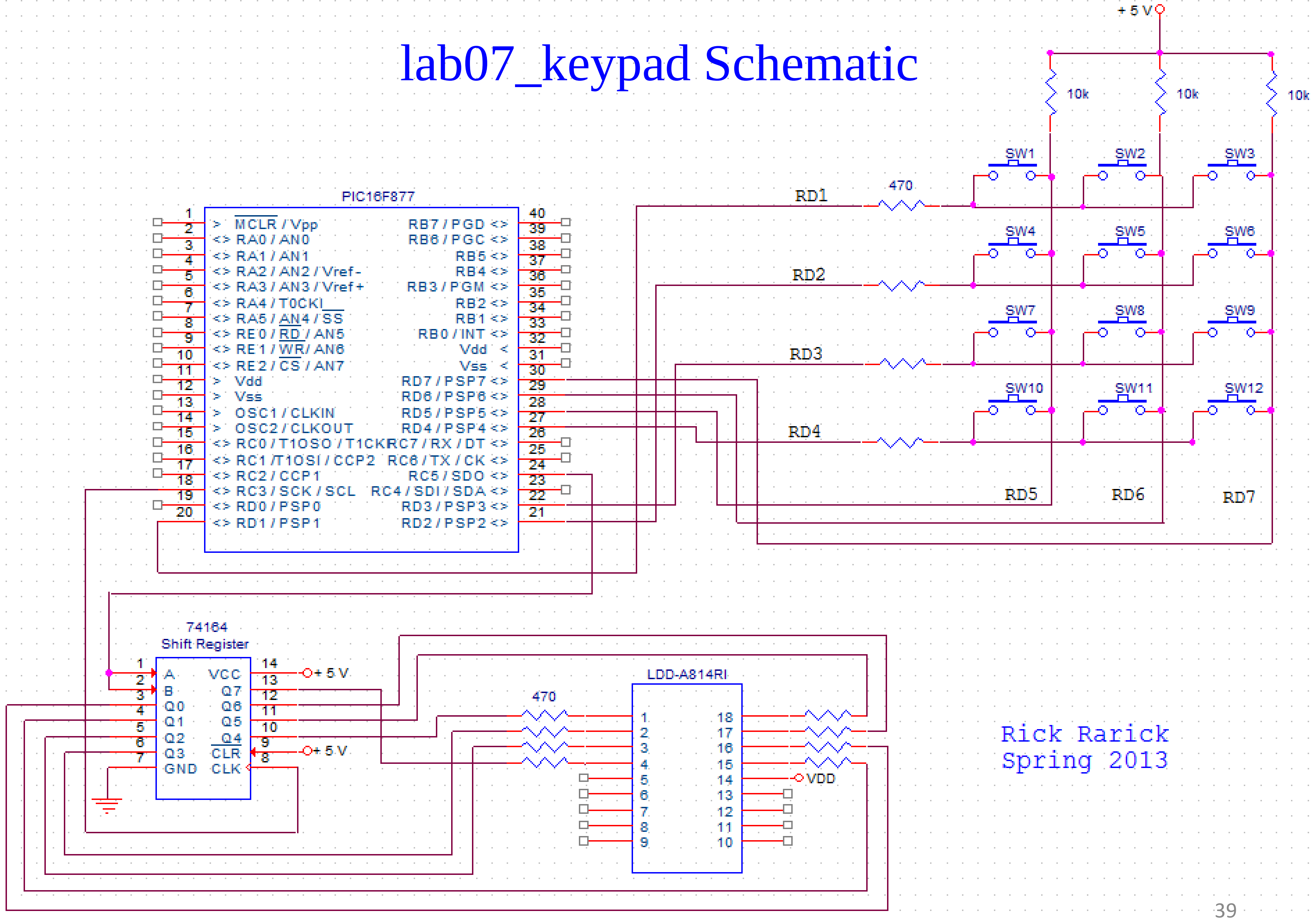




Grayhill 96AB2

Bottom View

lab07_keypad Schematic



Rick Rarick
Spring 2013

Lab 7 Outline

Finite Arithmetic

1. Computers and calculators use **finite** number systems.

2. Only a finite range of numbers can be stored in registers of finite size.

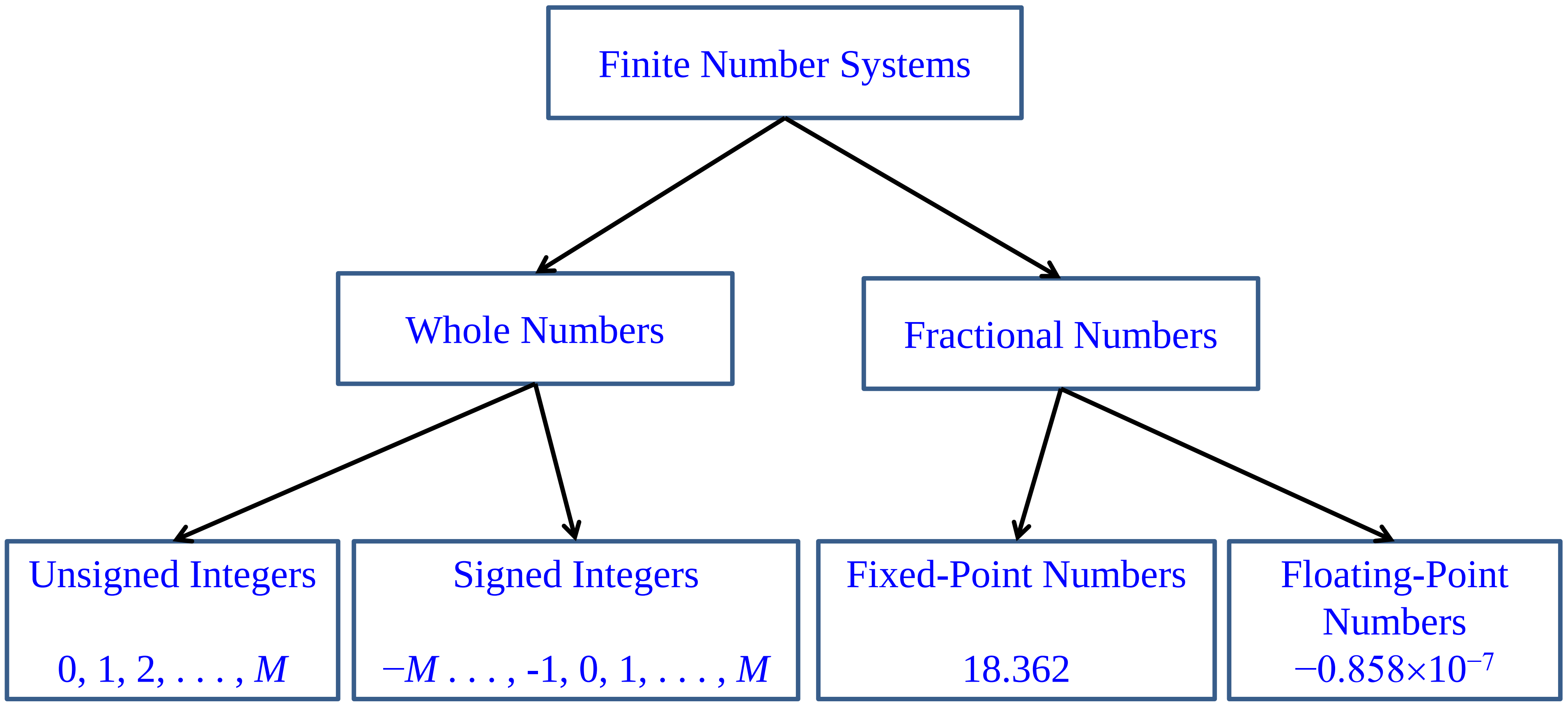
3. A special set of rules must be used for finite arithmetic .

4. For 8-bit registers, only eight binary digits are stored.

8-bit	1	0	0	1	0	1	1	1

5. **Various meanings or interpretations can be assigned to these bits.**

Finite Arithmetic



4-Bit Binary Representation of Integers

Interpretation: Unsigned Integers

Decimal Representation	Standard Binary Representation
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Interpretation: Signed Integers

Decimal Representation	Two's Complement Representation
7	0111
6	0110
5	0101
4	0100
3	0011
2	0010
1	0001
0	0000
– 1	1111
– 2	1110
– 3	1101
– 4	1100
– 5	1011
– 6	1010
– 7	1001
– 8	1000

Positive Integers MSB = 0

Negative Integers MSB = 1

4-Bit Two's Complement

1. The two's complement **representation** of a **positive** integer is the same as its unsigned representation.

2. The two's complement **representation** of a **negative** integer is determined by applying the two's complement **operation** to the absolute value or magnitude of the integer.

+ 5:	0101
Complement all bits:	1010
Add 1:	
– 5:	1011

two's complement operation

3. The two's complement **operation** is analogous to taking the negative of an integer.

4. Terminology: “Find (or take) the 2’s complement of 5” means “Apply the 2’s complement **operation** to 5”

4-Bit Two's Complement

Two's Complement Representation

7	0111
6	0110
5	0101
4	0100
3	0011
2	0010
1	0001
0	0000
– 1	1111
– 2	1110
– 3	1101
– 4	1100
– 5	1011
– 6	1010
– 7	1001
– 8	1000

Applying the two's complement **operation** to a positive integer gives the **two's complement representation** of the negative of the integer.

+ 5:	0101
Complement all bits:	1010
Add 1:	1011

2's comp operation

2's comp of + 5 = – 5

4-Bit Two's Complement

Two's Complement Representation

7	0111
6	0110
5	0101
4	0100
3	0011
2	0010
1	0001
0	0000
– 1	1111
– 2	1110
– 3	1101
– 4	1100
– 5	1011
– 6	1010
– 7	1001
– 8	1000

Applying the two's complement **operation** to a negative integer gives the two's complement **representation** of the **magnitude** of the negative integer.

– 5:	1011
Complement all bits:	0100
Add 1:	0101

2's comp of – 5 = + 5

So applying the two's complement operation twice is algebraically the same as applying the negative twice: – (– 5) = 5.

4-Bit Two's Complement

Two's Complement Representation

7	0111
6	0110
5	0101
4	0100
3	0011
2	0010
1	0001
0	0000
– 1	1111
– 2	1110
– 3	1101
– 4	1100
– 5	1011
– 6	1010
– 7	1001
– 8	1000

The 2’s complement of a number (“taking” the 2’s complement) is not the same as the 2’s complement representation of the number.

– 5:	1011
Complement all bits:	0100
Add 1:	0101

2's comp of – 5 = + 5

a) 2’s complement of – 5 = 0101
b) 2’s complement representation of – 5 = 1011

8-**b**it Two's Complement Representation

Sign bit

Most significant bit	7 bits								
0	1	1	1	1	1	1	1	=	127
0	1	1	1	1	1	1	0	=	126
0	0	0	0	0	0	1	0	=	2
0	0	0	0	0	0	0	1	=	1
0	0	0	0	0	0	0	0	=	0
1	1	1	1	1	1	1	1	=	−1
1	1	1	1	1	1	1	0	=	−2
1	0	0	0	0	0	0	1	=	−127
1	0	0	0	0	0	0	0	=	−128

[Video Tutorial: Understanding 2' Complement – Derek Molloy]

8-Bit Two's Complement

Example: In an 8-bit 2's complement system, what **decimal** integer does 0110 0011 represent?

1. It is a positive integer since the MSB = 0.

2. The remaining 7 bits give the magnitude: 0110 0011 = + 99

Example: In an 8-bit 2's complement system, what **decimal** integer does 1110 0011 represent?

1. It is a negative integer since the MSB = 1.
2. Calculate the 2's complement to get the magnitude:

   1110 0011 → complement bits → 0001 1100 → add 1

   → 0001 1101 → 29 = magnitude

3. 1110 0011 = – 29 **(when interpreted as a 2's complement integer)**

8-bit Addition

```
    145 =    1001  0001
+   146 =    1001  0010
-----------------------
    291 = 1  0010  0011
```

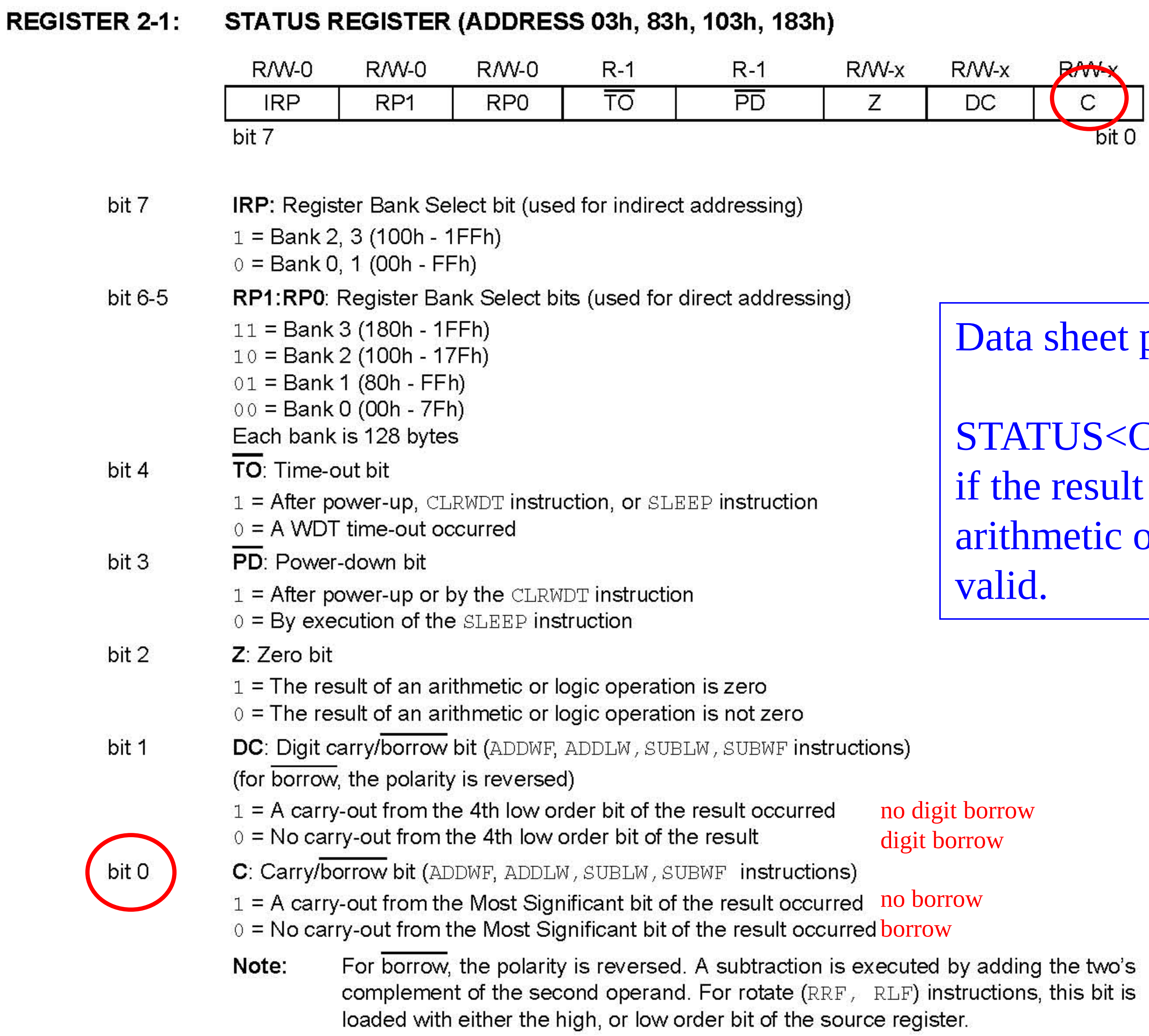
```
movlw    D'145'
addlw    D'146'
```

W = 0010 0011 = 35

STATUS<C> = 1

Carry bit = 1, so the 8-bit result **not valid**

1. The carry bit is STATUS<C> = STATUS<0>.
2. Determines whether the 8-bit result of an addition operation is valid.



REGISTER 2-1: STATUS REGISTER (ADDRESS 03h, 83h, 103h, 183h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{\text{TO}}$	$\overline{\text{PD}}$	Z	DC	C
bit 7							bit 0

bit 7 **IRP:** Register Bank Select bit (used for indirect addressing)
`1` = Bank 2, 3 (100h - 1FFh)
`0` = Bank 0, 1 (00h - FFh)

bit 6-5 **RP1:RP0**: Register Bank Select bits (used for direct addressing)
`11` = Bank 3 (180h - 1FFh)
`10` = Bank 2 (100h - 17Fh)
`01` = Bank 1 (80h - FFh)
`00` = Bank 0 (00h - 7Fh)
Each bank is 128 bytes

bit 4 **$\overline{\text{TO}}$**: Time-out bit
`1` = After power-up, `CLRWDT` instruction, or `SLEEP` instruction
`0` = A WDT time-out occurred

bit 3 **$\overline{\text{PD}}$**: Power-down bit
`1` = After power-up or by the `CLRWDT` instruction
`0` = By execution of the `SLEEP` instruction

bit 2 **Z**: Zero bit
`1` = The result of an arithmetic or logic operation is zero
`0` = The result of an arithmetic or logic operation is not zero

bit 1 **DC**: Digit carry/$\overline{\text{borrow}}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)
(for $\overline{\text{borrow}}$, the polarity is reversed)
`1` = A carry-out from the 4th low order bit of the result occurred no digit borrow
`0` = No carry-out from the 4th low order bit of the result digit borrow

bit 0 **C**: Carry/$\overline{\text{borrow}}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)
`1` = A carry-out from the Most Significant bit of the result occurred no borrow
`0` = No carry-out from the Most Significant bit of the result occurred borrow

Note: For $\overline{\text{borrow}}$, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high, or low order bit of the source register.

Data sheet p. 18

STATUS<C> tells you if the result of an arithmetic operation is valid.

STATUS<C>

1. Addition:

   C = 1 (carry occurred, 8-bit result **not valid**)

   C = 0 (carry did not occur, 8-bit result **valid**)

2. Subtraction:

   C = 1 (borrow did not occur, 8-bit result **valid**)

   C = 0 (borrow occurred, 8-bit result **not valid**)

3. Note the opposite interpretation of the carry bit for addition and subtraction.

Subtraction

- Subtraction is **defined** as adding the negative of a number to another number.

$$7 - 2 = 7 + (-2) = 5$$

- The PIC subtracts by adding the two's complement (the negative) of a number.

$$\begin{array}{rcl} 98 & = & 0110\ 0010 \\ \underline{-51} & \underline{=} & \underline{-0011\ 0011} \end{array}$$

(But there is no "–" sign in binary, so take the 2's comp to get the negative of 51 = 0011 0011)

$$0011\ 0011 \rightarrow 1100\ 1100 + 1 = 1100\ 1101 = \text{2's comp of } 51\ \ (= -51)$$

$$\begin{array}{rcrl} 98 & = & & 0110\ 0010 \\ \underline{-51} & \underline{=} & \underline{+} & \underline{1100\ 1101} \\ 47 & & \underset{\uparrow \atop C}{1} & \underbrace{0010\ 1111}_{47} \end{array}$$

(2's comp)

(STATUS< C > = 1, 8-bit result is **valid**)

Note that for 8-bit numbers, the largest number we can subtract is 128.

Subtraction

The PIC subtracts by adding the two’s complement:

$$\begin{array}{rcr} 51 & = & 0011\ \ 0011 \\ \underline{-\,98} & \underline{=} & \underline{-\,0110\ \ 0010} \end{array}$$

$$\begin{array}{rcrl} 51 & = & 0011\ \ 0011 & \\ \underline{-\,98} & \underline{=} & \underline{+\,1001\ 1110} & \text{(2's comp)} \\ -\,47 & & 0\ \ \underbrace{1101\ \ 0001}_{209} & (\text{STATUS<C>} = 0\text{, 8-bit result is } \textbf{not valid}) \end{array}$$

But the result is valid if it is **INTERPRETED** as an 8-bit two’s complement of the magnitude of − 47.

Subtraction

$$\begin{array}{rcll}
51 & = & 0011\ \ 0011 & \\
\underline{-\ 98} & \underline{=} & \underline{+\ 1001\ 1110} & \text{(2's comp)} \\
-\ 47 & & 0\ \ \underbrace{1101\ \ 0001}_{209} & \text{(STATUS<C> = 0, 8-bit result is } \textbf{not valid}\text{)}
\end{array}$$

Interpret 1101 0001 as a 2's comp integer:

1. Since the MSB = 1, it is negative.

2. Calculate its 2's comp to get the magnitude:

   1101 0001 + 1 = 1101 0010 = 47 = magnitude

3. So, 1101 0001 = – 47 **(when interpreted as a 2's complement integer)**

Subtraction

$$\begin{array}{rrl}
51 & = & 0011\ 0011 \\
\underline{-\,98} & \underline{=} & \underline{+\,1001\ 1110} \quad \text{(2's comp)} \\
-\,47 & & 0\ \underbrace{1101\ 0001}_{209} \quad \text{(STATUS<C> = 0, 8-bit result is } \textbf{not valid}\text{)}
\end{array}$$

```
movlw .98    ; W = 98
sublw .57    ; W = L – W = 57 – 98 = -47 = 1101 0001
```

- Who interprets the 1101 0001 = 209 as – 47 ?
- The programmer ! Code must be provided.

LCD

subtraction → W = 1101 0001 → code → $\boxed{-\,47}$

Lab 7 Outline

1. Master Synchronous Serial Port – SPI Mode
2. 7-Segment LED Display
3. Shift Registers
4. lab07_SPI Setup
5. Keypad Scanning
6. lab07_keypad Setup
7. Finite Arithmetic
8. **Watchdog Timer**

Watchdog Timer

1. Many embedded systems can’t be manually rebooted if something goes wrong (such as getting trapped in an infinite loop).

2. Examples: Vending machines, spacecraft.

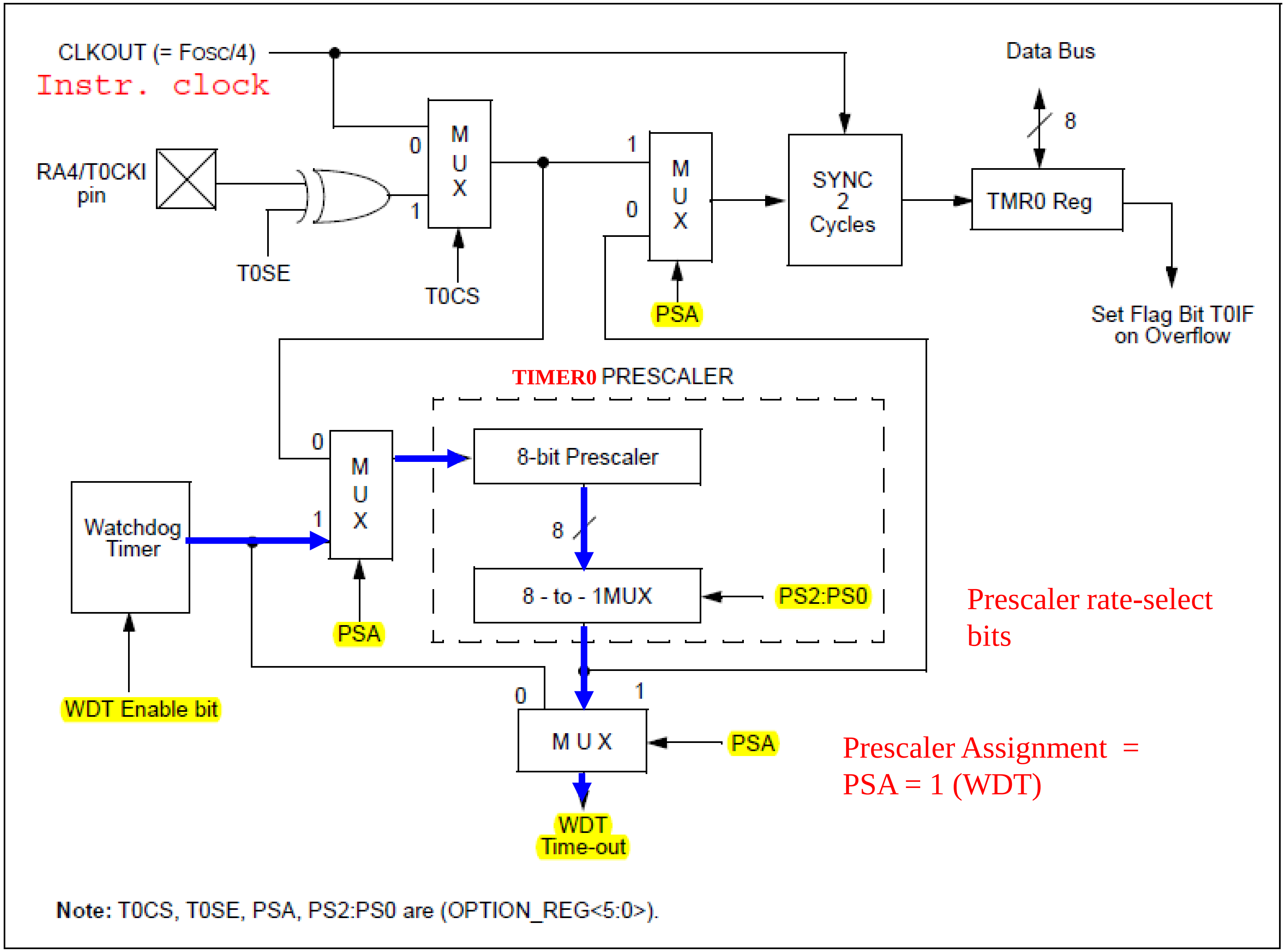
3. The Clementine spacecraft mission failed because of the lack of a watchdog timer. (See www.ganssle.com/watchdogs.pdf for an interesting article about the mission)

4. The WDT resets the PIC when it reaches some value (a few milliseconds).

5. The WDT reset is prevented with a CLRWDT instruction. The CLRWDT instruction should be placed in the main loop of the code. If the code gets stuck in an infinite loop, the WDT will reset the chip.

Watchdog Timer

FIGURE 5-1: BLOCK DIAGRAM OF THE TIMER0/WDT PRESCALER

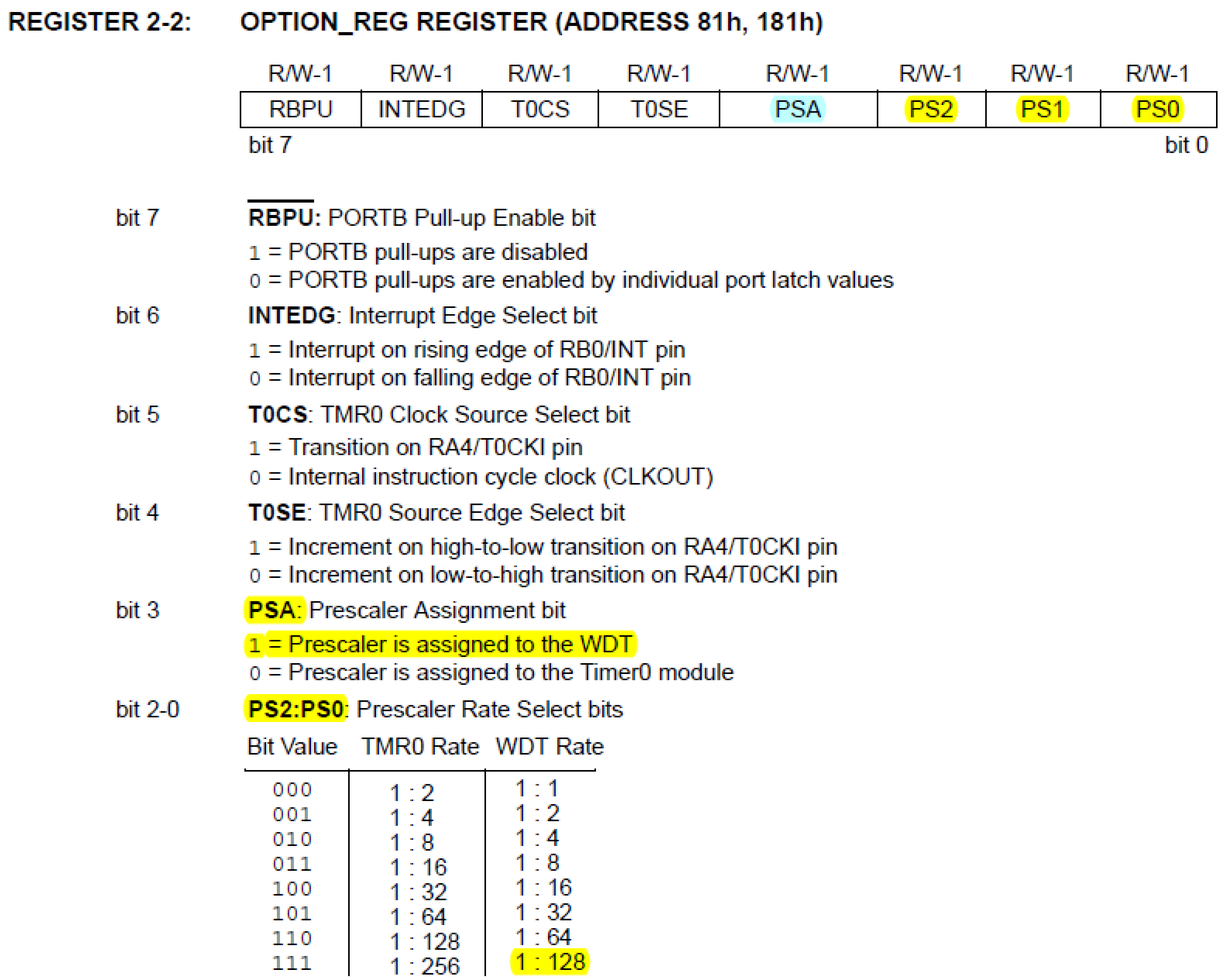


Prescaler rate-select bits

Prescaler Assignment = PSA = 1 (WDT)

Note: T0CS, T0SE, PSA, PS2:PS0 are (OPTION_REG<5:0>).

Watchdog Timer



REGISTER 2-2: OPTION_REG REGISTER (ADDRESS 81h, 181h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$:** PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Watchdog Timer

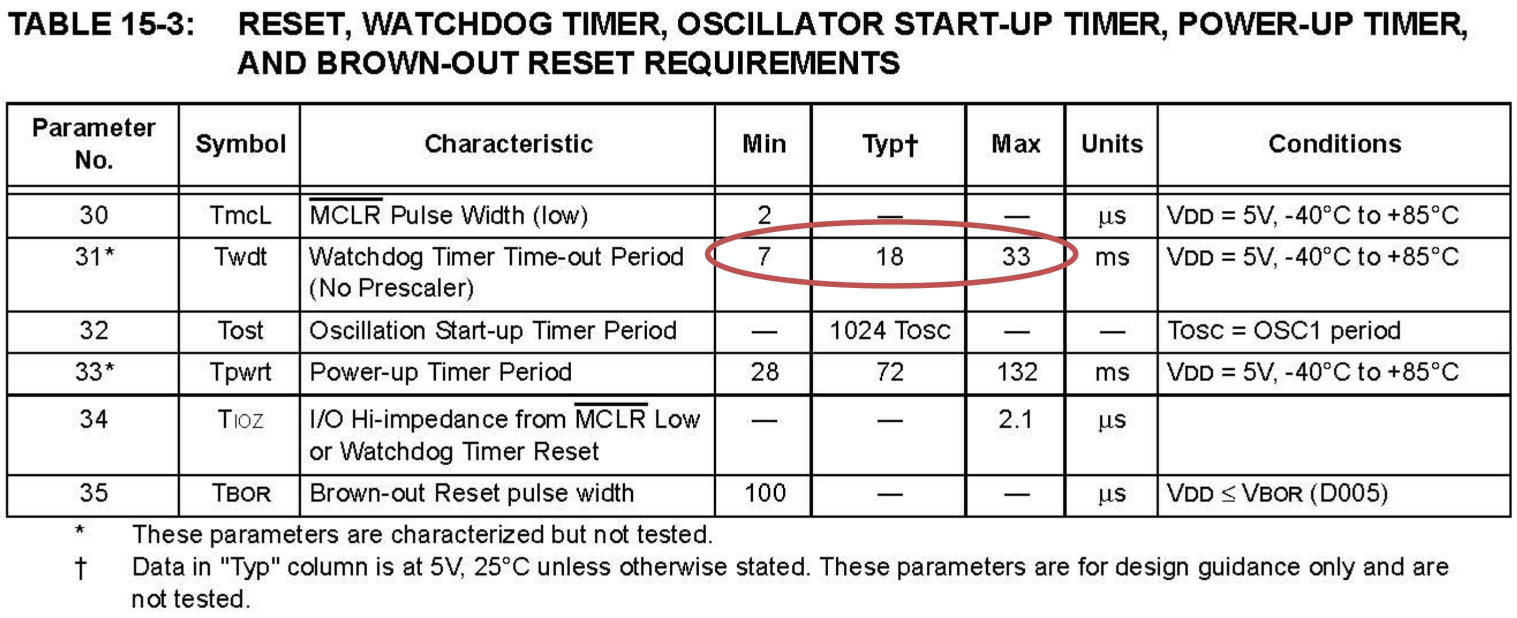


TABLE 15-3: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER, AND BROWN-OUT RESET REQUIREMENTS

Parameter No.	Symbol	Characteristic	Min	Typ†	Max	Units	Conditions
30	TmcL	$\overline{\text{MCLR}}$ Pulse Width (low)	2	—	—	µs	VDD = 5V, -40°C to +85°C
31*	Twdt	Watchdog Timer Time-out Period (No Prescaler)	7	18	33	ms	VDD = 5V, -40°C to +85°C
32	Tost	Oscillation Start-up Timer Period	—	1024 Tosc	—	—	Tosc = OSC1 period
33*	Tpwrt	Power-up Timer Period	28	72	132	ms	VDD = 5V, -40°C to +85°C
34	TIOZ	I/O Hi-impedance from $\overline{\text{MCLR}}$ Low or Watchdog Timer Reset	—	—	2.1	µs	
35	TBOR	Brown-out Reset pulse width	100	—	—	µs	VDD ≤ VBOR (D005)

* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Datasheet p. 164

The nominal watchdog period is between 7 and 33 ms, with a typical nominal value of 18 ms.

End of Lab 7