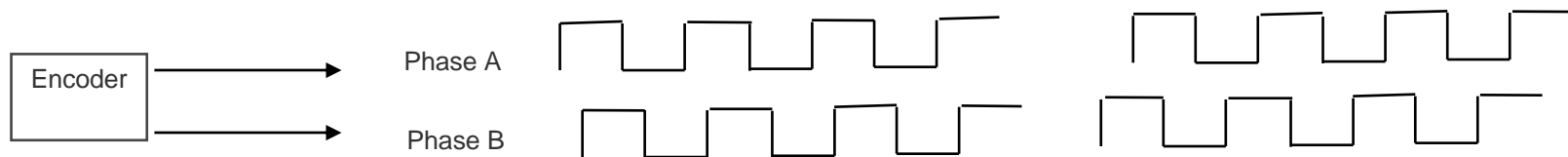# 11015 MS2

MPLAB® Simulators
Advanced Stimulus

Additional Examples

Class

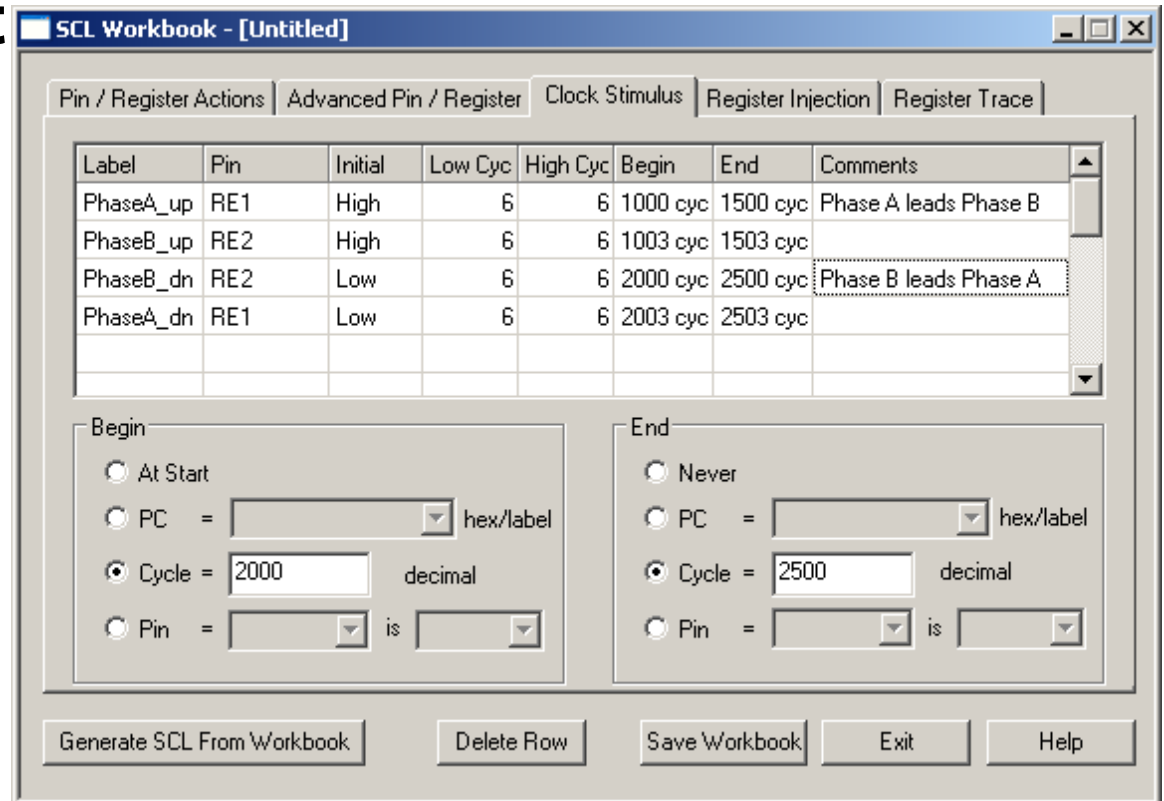# Using Clock tab for Grey Code Encoder stimulus

- **Grey code encoder has 2 pulse trains, one out of phase with the other. When turning in one direction the phase A leads phase B. In the opposite direction phase B will lead phase A.**

- **A goes high when B is low   = Clockwise
A goes high when B is high = Anti-Clockwise**

Class

Using Clock tab for Grey Code Encoder stimulus

- **Encoder 64 counts/rev. At 300 RPM (5 RPS) we get 320 pulses per second. This is a 3125 us pulse so we will use 3us.**

- **At 20 MHZ we get 0.2us Instruction cycle**

- **Pulse width of 6 cycles = 1.2us period for each half of a full wave. (2.4us pulse)**

SCL Workbook - [Untitled]

Pin / Register Actions | Advanced Pin / Register | Clock Stimulus | Register Injection | Register Trace

Label	Pin	Initial	Low Cyc	High Cyc	Begin	End	Comments
PhaseA_up	RE1	High	6	6	1000 cyc	1500 cyc	Phase A leads Phase B
PhaseB_up	RE2	High	6	6	1003 cyc	1503 cyc	
PhaseB_dn	RE2	Low	6	6	2000 cyc	2500 cyc	Phase B leads Phase A
PhaseA_dn	RE1	Low	6	6	2003 cyc	2503 cyc	

Begin
- ○ At Start
- ○ PC = [ ] hex/label
- ● Cycle = [2000] decimal
- ○ Pin = [ ] is [ ]

End
- ○ Never
- ○ PC = [ ] hex/label
- ● Cycle = [2500] decimal
- ○ Pin = [ ] is [ ]
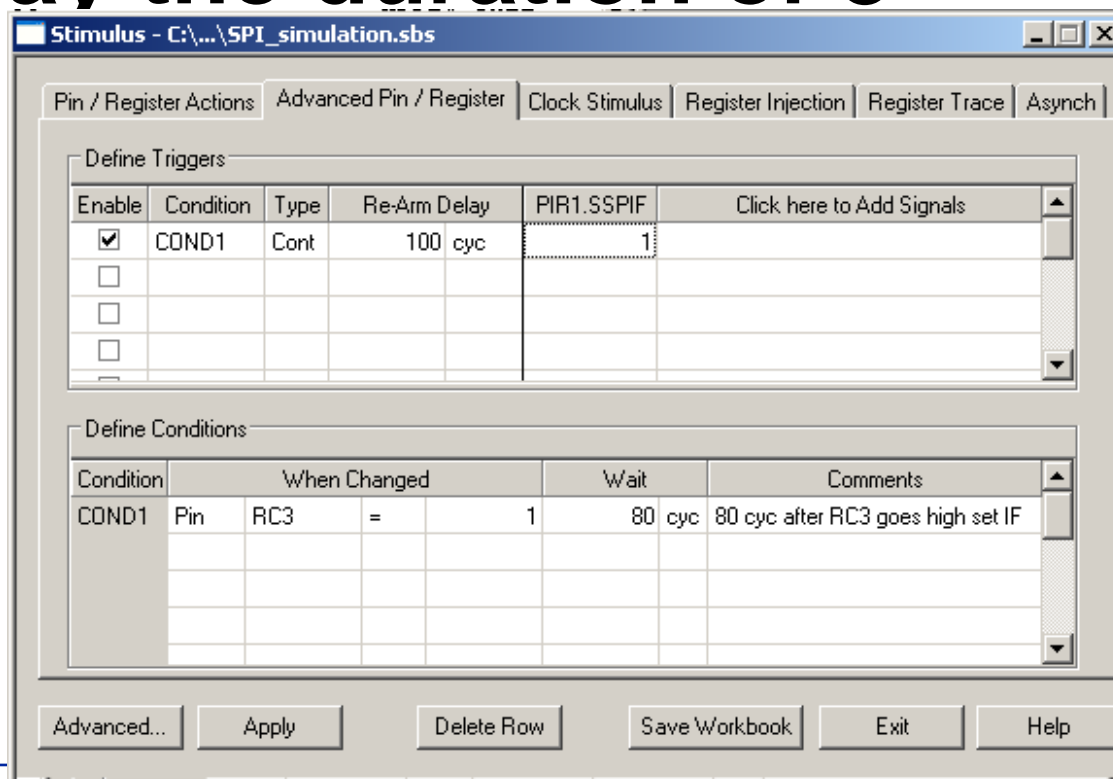
Generate SCL From Workbook | Delete Row | Save Workbook | Exit | Help

Class

Low power wake up

- **On startup under the simulator all input pins are at a zero level state. To initialize the low power wakeup pin we need to set this early to a high state.**
**We will do this by using simple Pin/Register stimulus to set RA0 at 50 ic.**

Low power wake up

- **Now to set the Low power wake up. We need to make RA0 go low as if the level to the pin dropped. Use Clock stimulus to set a clock on RA0. Begin at cycle = 100,000. Initial high, low for 2 and high for 9,999,998. End set at default Never. Under simulation this will wake the process about every 4 seconds.**
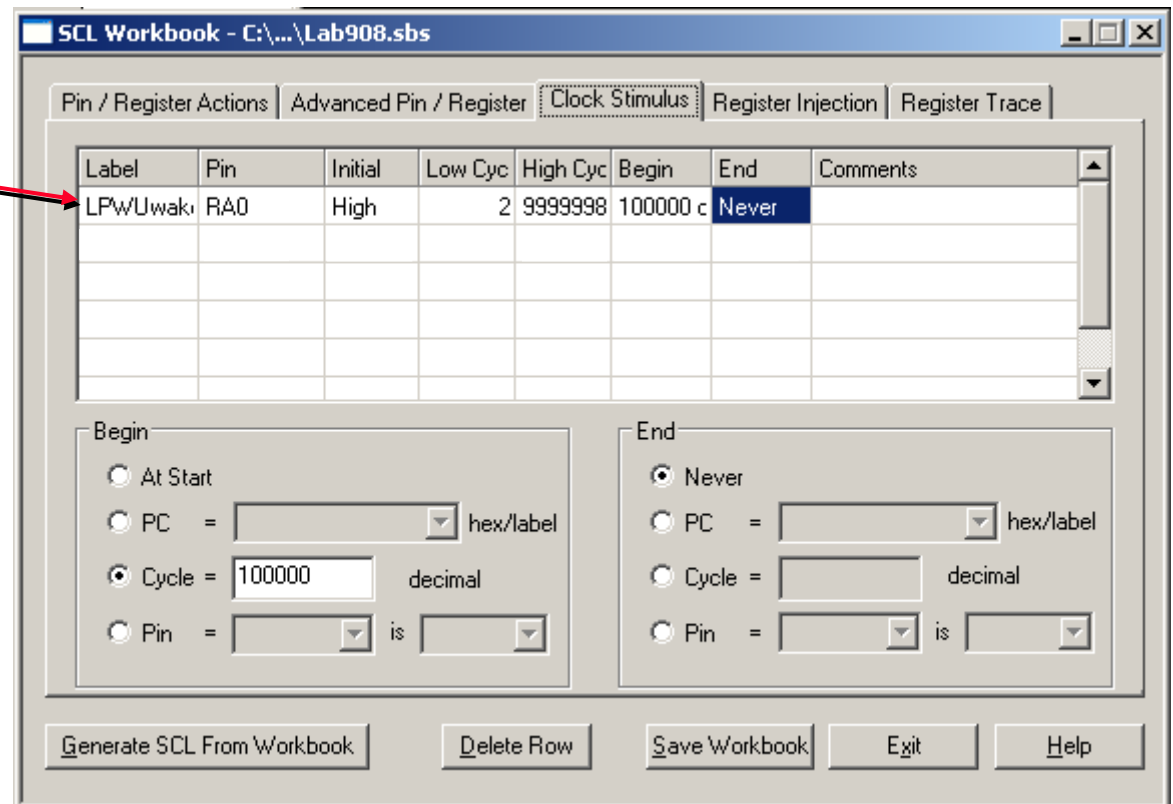
SPI input

- **You can either clock the data in using the firmware (your own code) and driving an output pin. Here we use 1 clock per 8 or 16 bit data**

- **Or provide a clock as in this example as if it was an external clock. To use your code to clock data eliminate the SPI clock step.**

- **The data is injected using a file which is attached to the SSPBUF, SFR with trigger on Demand and RAW format.**

- **RC3 is pulsed 20 cycles low 5 cycles high only when RC2 is high.**

Class

SPI input

- **RC2 is set high using an Asynchronous pulse for 143 cycles which allows 6 low to high transitions of RC3.**

- **Each low to high transition of RC3 sets the SSPIF flag. (not quite true SPI which has 1 clock per bit)**

- **Your program should wait until the IF bit is set then read the SSPBUF (into Wreg)**

- **Clear the IF flag and loop for the next read.**

- **When you read the SSPBUF the data from the file will be injected as you read.**

Class

SPI input

- **Asynch Pulse to initiate a message**

SPI input

- ## SPI Clock

SPI input

- **SPI IRQ Flag set by using a conditional statement**

SPI input

- **If you wanted to clock bits and have the IF set after 8 bit clocks. You could set the wait time on COND1 to delay the duration of 8 clock cycles.**

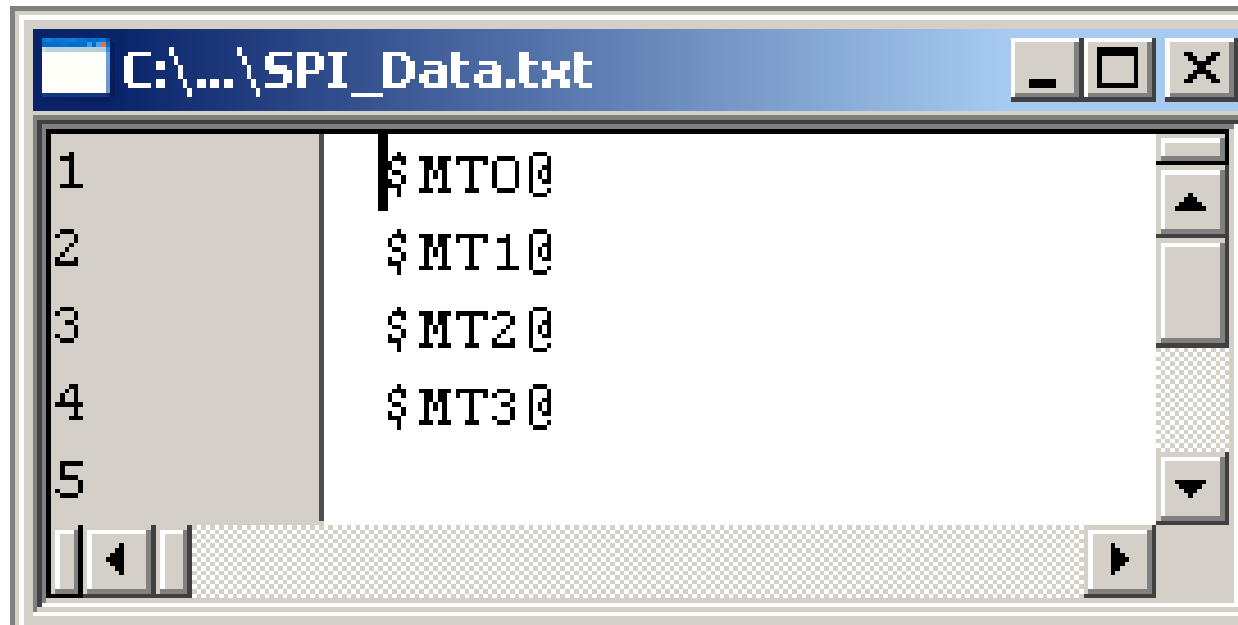- **You can then set the re-arm delay to avoid a retrigger.**

SPI input

- ## SPI Data file injection
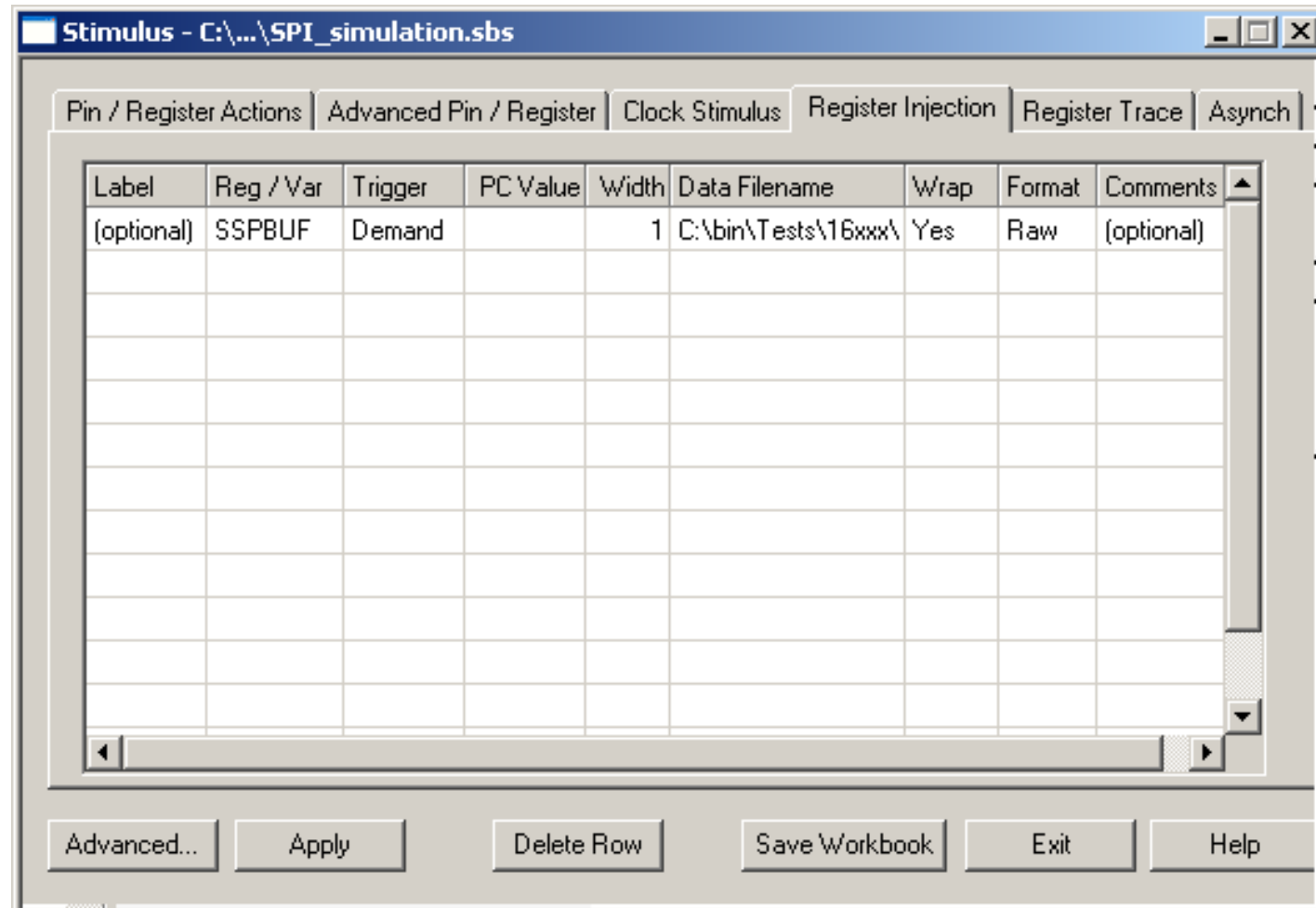
SPI input

- ## SSPBUF Data
  ## 1 character is read on each IRQIF
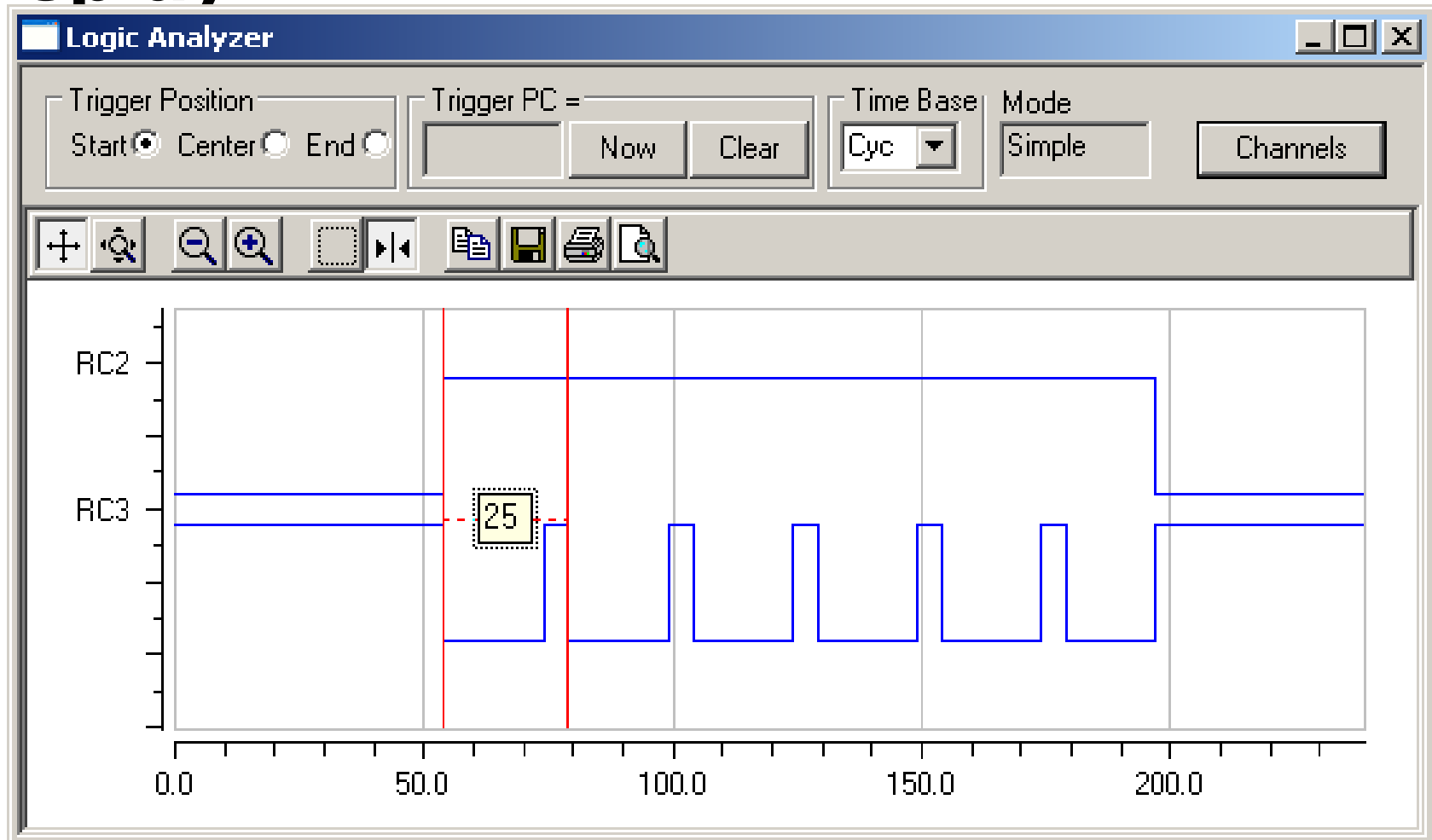 ## The string is 5 characters plus CRLF

C:\...\SPI_Data.txt

```
1    $MTO@
2    $MT1@
3    $MT2@
4    $MT3@
5
```

Class

SPI input

- ## SPI Clock as seen within the Logic Display

SPI input

- ## SPI Clock as seen within the Logic Display when using 8 clock pulses per IRQ

- ## To show the delay we set RC1 at the same time as the SSPIF flag