11016 MS3

Debugging Techniques and Using Stimulus within the MPLAB® Simulators

Class Objective

- When you finish this class you will…
- Understand how to create stimulus to debug code within the MPLAB® Simulator
  - Pin Initialization
  - Waveform generation / Measurement
  - Load testing interrupts
  - Peripheral data injection
  - Algorithm verification
  - Simpler Pin stimulus file format
- Understand Complex Breakpoints
- Use Instruction Trace Effectively
  - Triggers
  - Filters
- Know how to output data without a UART

Agenda

- Using Stimulus within the simulator combined with the Logic Analyzer to verify Stimulus generation
- Peripheral data injection
- Complex Breakpoints
- Using Instruction Trace effectively
- Logging data
- Find out more

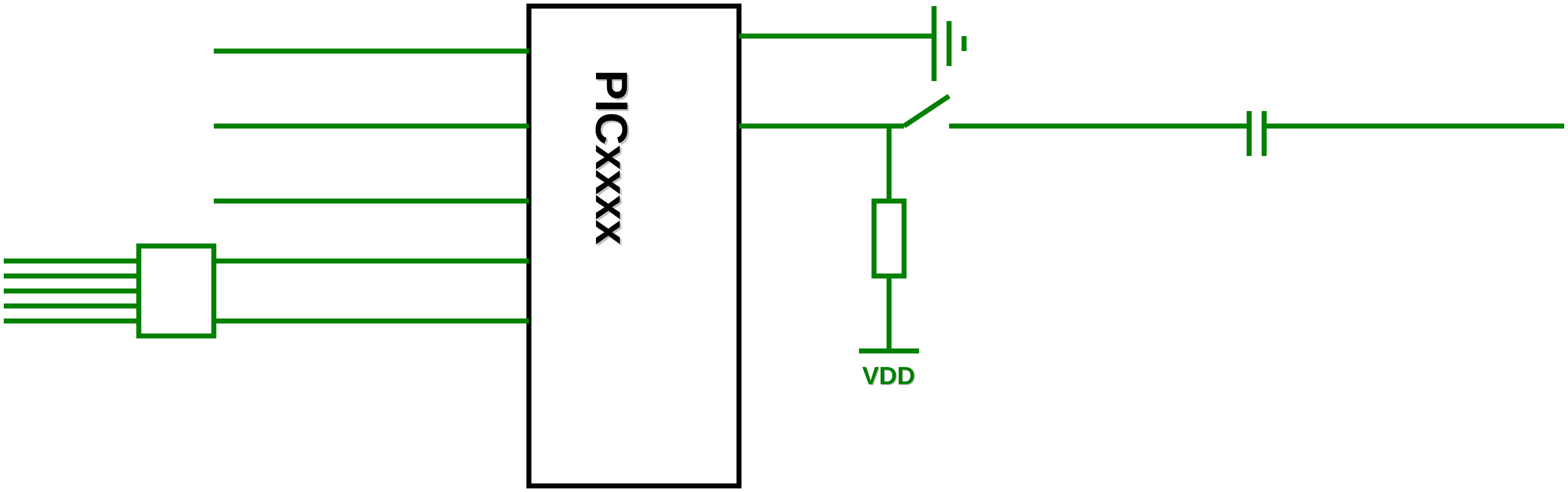
Stimulus Overview

- Stimulus is tied very tightly to and considered an integral part of the Simulator
- Used mostly to simulate target hardware I/O to/from the PIC® Microcontroller
- Asynch stimulus is user triggered
- Synch stimulus is system triggered
- Most cases stimulus is injection of signals and data
- Also used to export data

Stimulus Pin Initialization

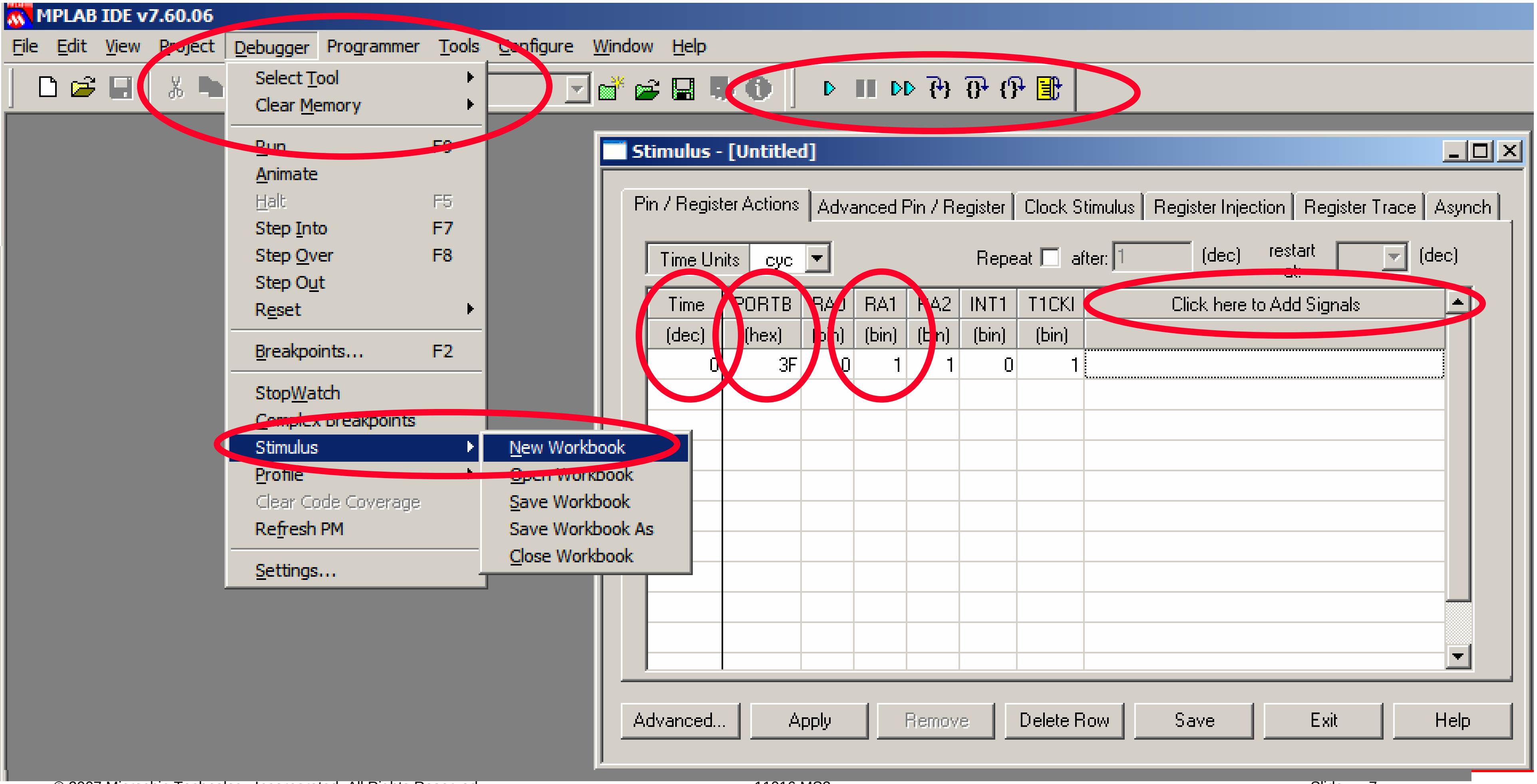
Stimulus Pin Initialization

- Code is complete so Firmware ready
- With Real Silicon on the Board all IO pins will have a known State
- Simulator starts at zero, may not change after POR



Stimulus Pin Initialization

- Use Pin/Register stimulus at time zero



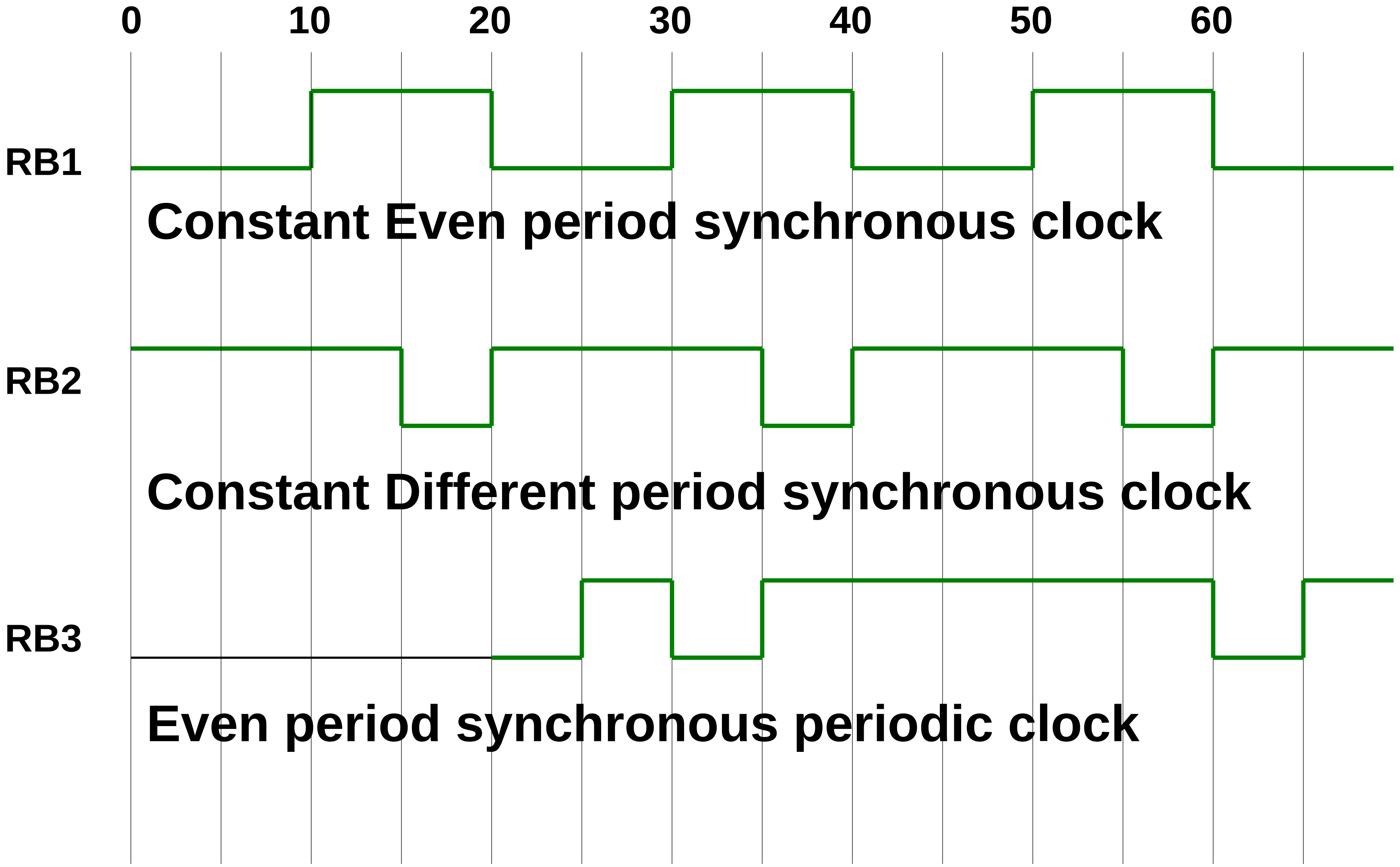
Time Units cyc

Repeat □ after: 1 (dec) restart at: (dec)

Time	PORTB	RA0	RA1	RA2	INT1	T1CKI	Click here to Add Signals
(dec)	(hex)	(bin)	(bin)	(bin)	(bin)	(bin)	
0	3F	0	1	1	0	1	

Stimulus Wave Form Generation

Waveform Generation Synch.



Constant Even period synchronous clock

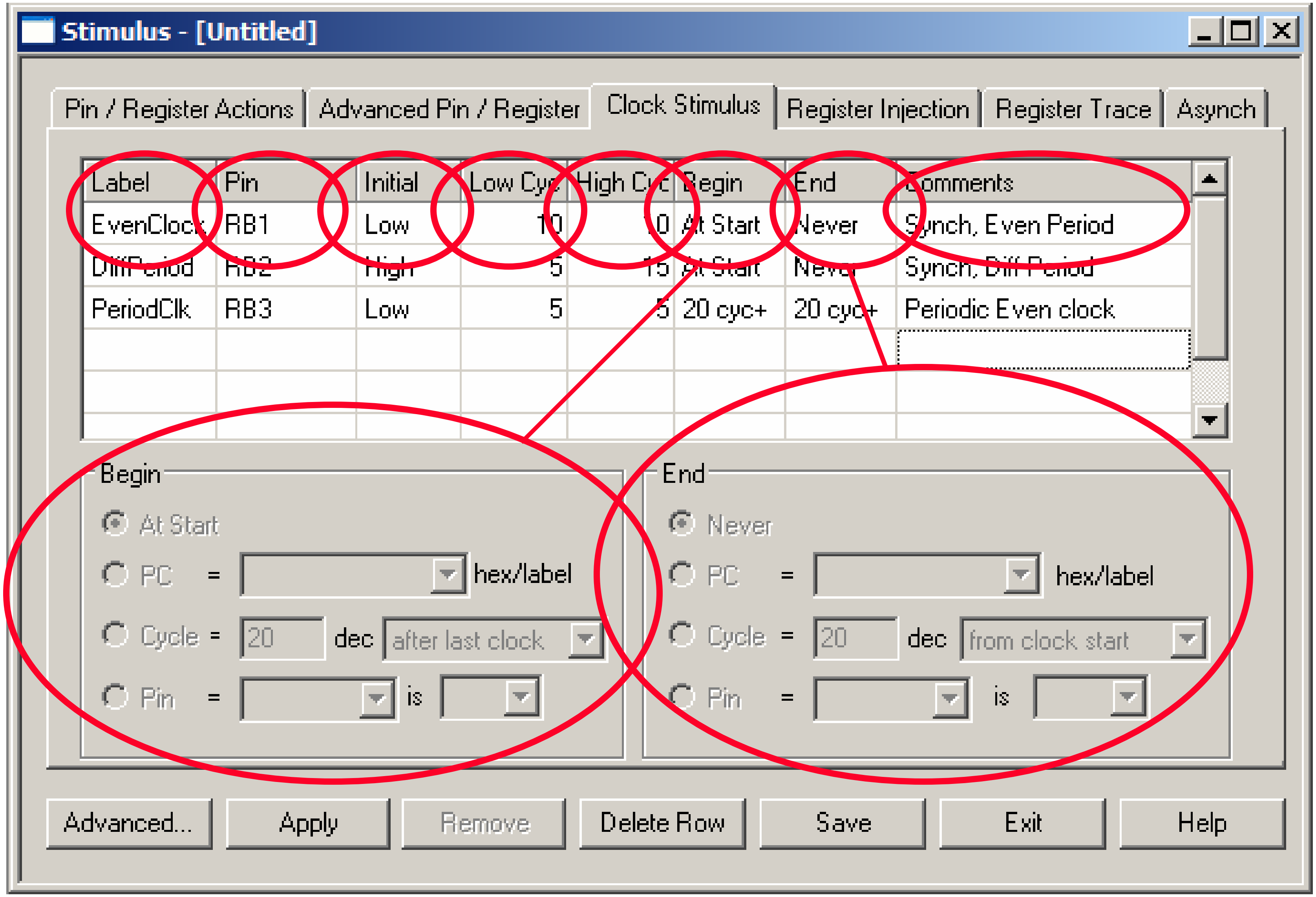
Constant Different period synchronous clock

Even period synchronous periodic clock

Waveform Generation Synch.

- Synchronous Wave forms are looked at as clocks
- Defined in Cycles (Instructions)
- Different periods
- Periodic pulses
- Most clocks or waveforms are defined in the Clock Stimulus tab
- Even clocks that are triggered asynchronously (later)

Waveform Generation Synch.



Label	Pin	Initial	Low Cyc	High Cyc	Begin	End	Comments
EvenClock	RB1	Low	10	10	At Start	Never	Synch, Even Period
DiffPeriod	RB2	High	5	15	At Start	Never	Synch, Diff Period
PeriodClk	RB3	Low	5	5	20 cyc+	20 cyc+	Periodic Even clock

Begin

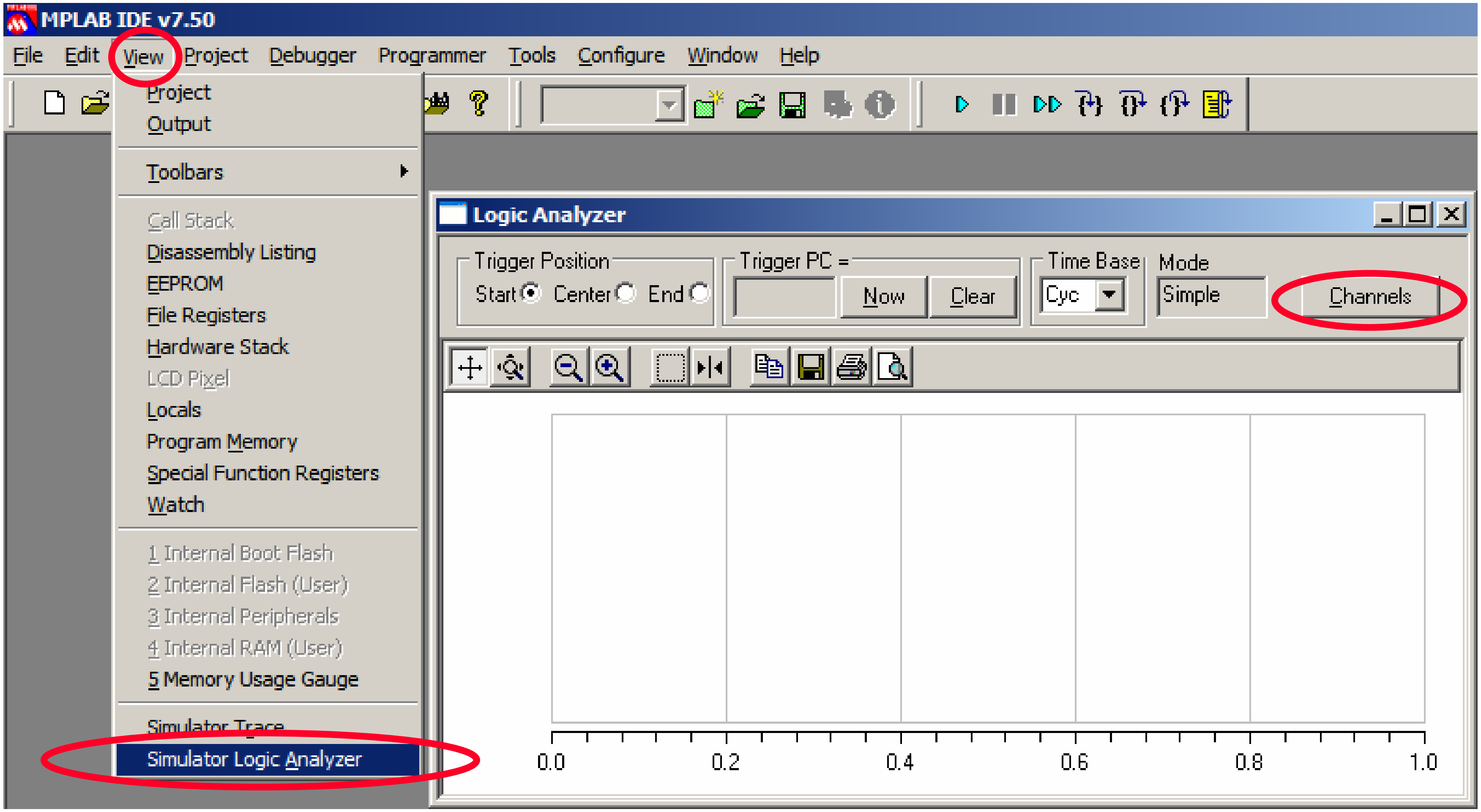
- ● At Start
- ○ PC = hex/label
- ○ Cycle = 20 dec after last clock
- ○ Pin = is

End

- ● Never
- ○ PC = hex/label
- ○ Cycle = 20 dec from clock start
- ○ Pin = is

Waveform Generation Synch.

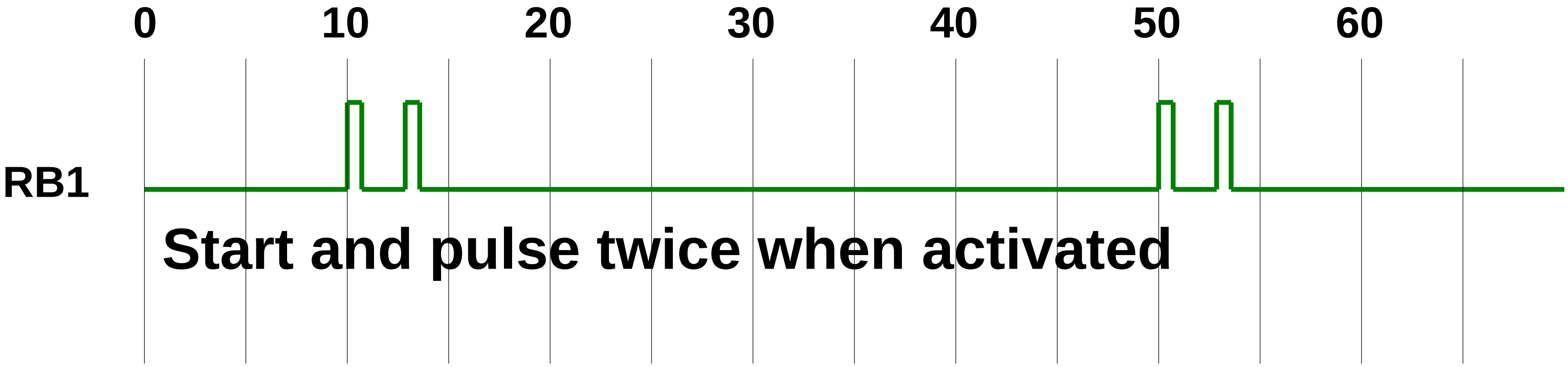
- Use Logic Analyzer to view IO pin changes
- Trace must be enabled through simulator settings



Demo 1

- **Synchronous Clock Generation**

Waveform Generation Asynch.

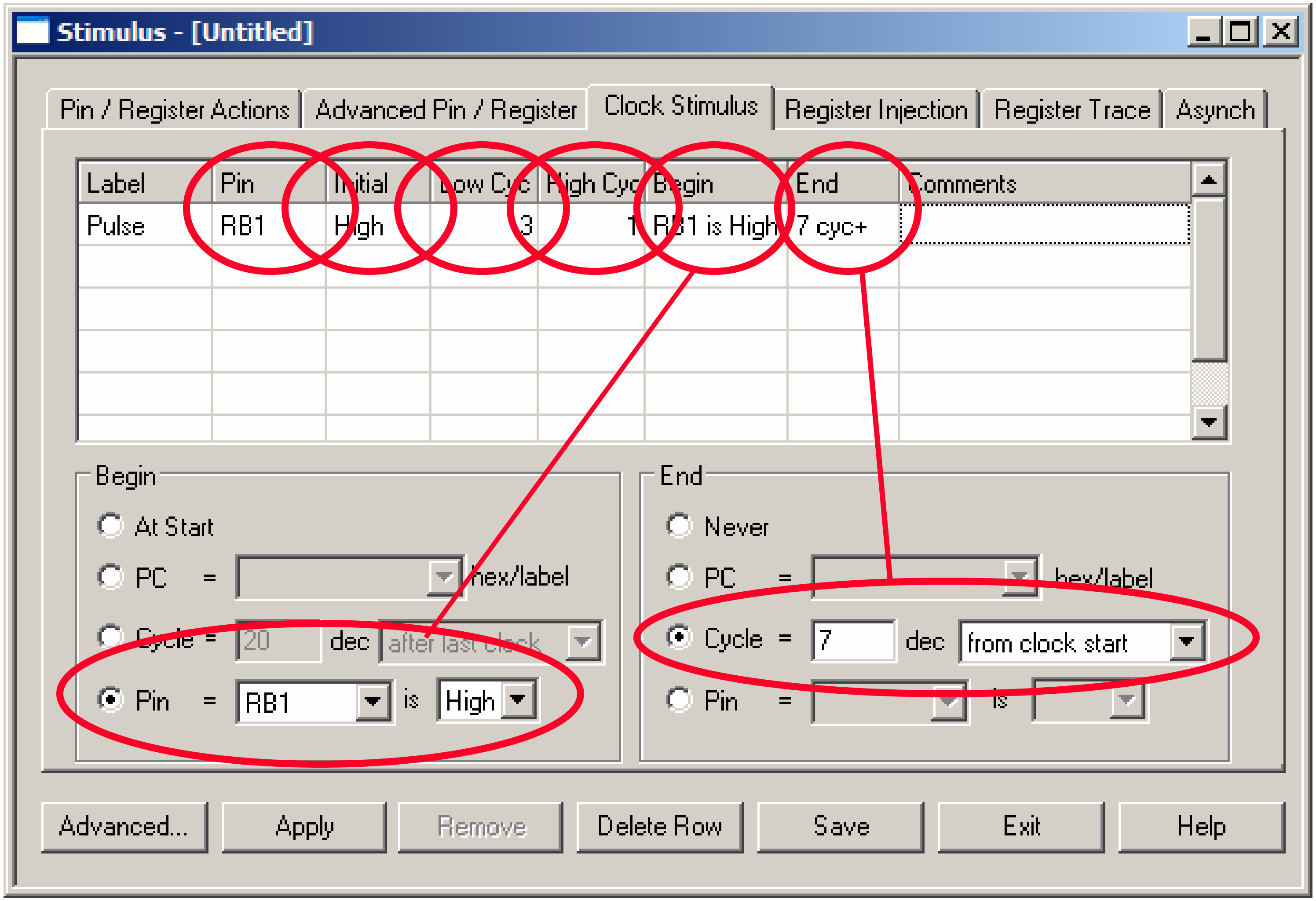


Start and pulse twice when activated

- Asynchronous triggered, synchronous pulses
  - Define Synchronous pulses in Clock Stimulus
  - Use asynchronous stimulus to trigger the start

Waveform Generation Asynch.

- **Start and pulse twice when activated**



Label	Pin	Initial	Low Cyc	High Cyc	Begin	End	Comments
Pulse	RB1	High	3	1	RB1 is High	7 cyc+	

Begin

○ At Start

○ PC = hex/label

○ Cycle = 20 dec after last clock

● Pin = RB1 is High

End

○ Never

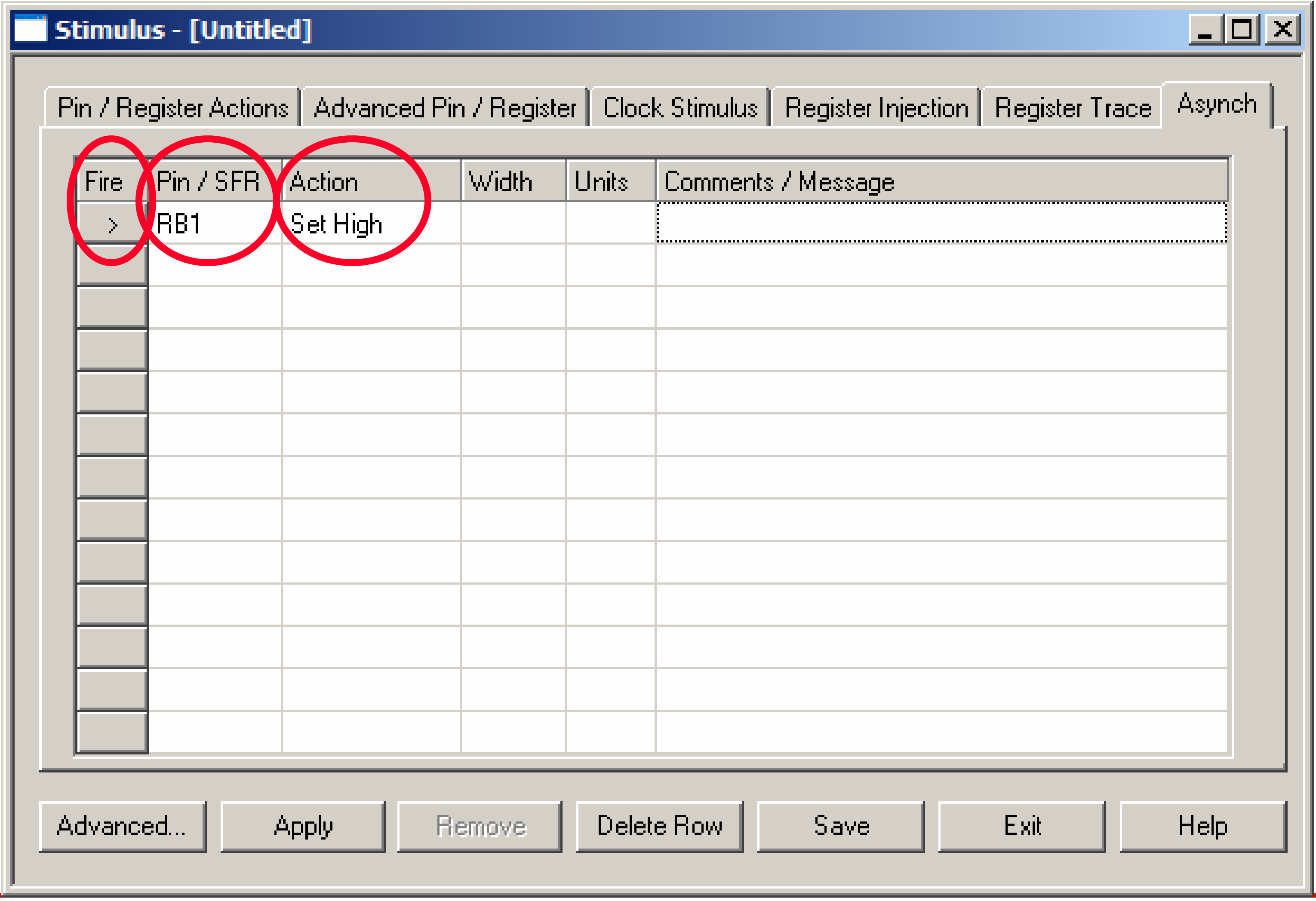
○ PC = hex/label

● Cycle = 7 dec from clock start

○ Pin = is

Waveform Generation Asynch.

- **Start and pulse twice when activated**

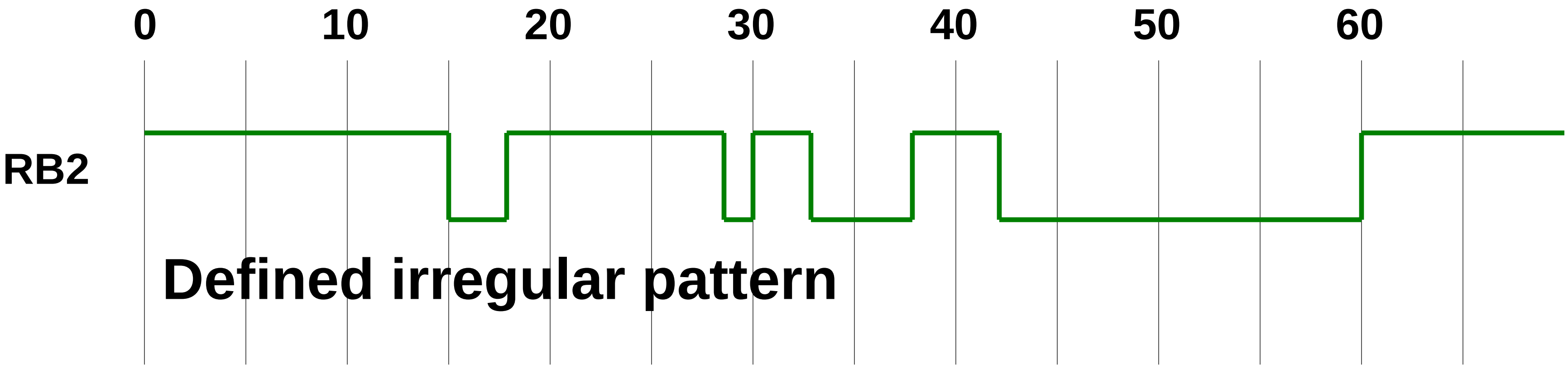


Fire	Pin / SFR	Action	Width	Units	Comments / Message
>	RB1	Set High			

Demo 2

- **Asynchronous trigger for a Synchronous Clock**

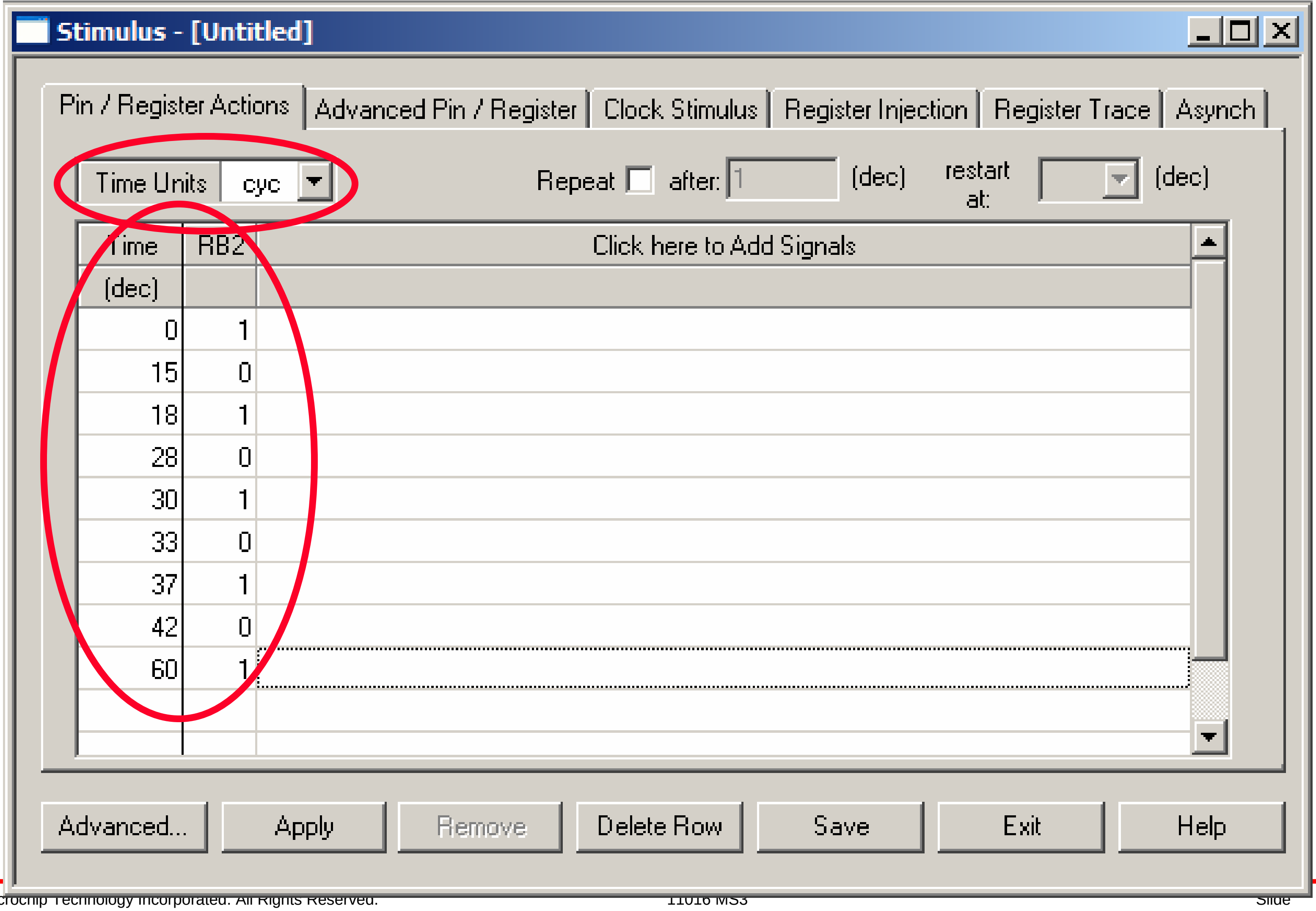
Waveform Generation Irregular



- **Predefined irregular waveform based on run time**
  - **Define transitions using the Pin/Register tab of stimulus at predefined execution time**

Waveform Generation Irregular

- **Defined irregular pattern**

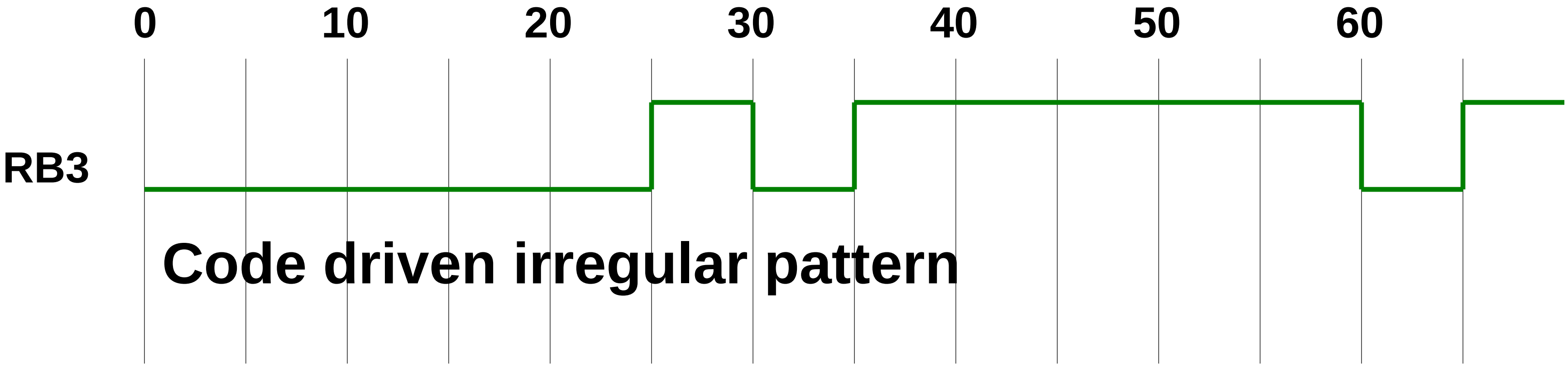


Time Units: cyc

Repeat □ after: 1 (dec) restart at: (dec)

Time (dec)	RB2
0	1
15	0
18	1
28	0
30	1
33	0
37	1
42	0
60	1

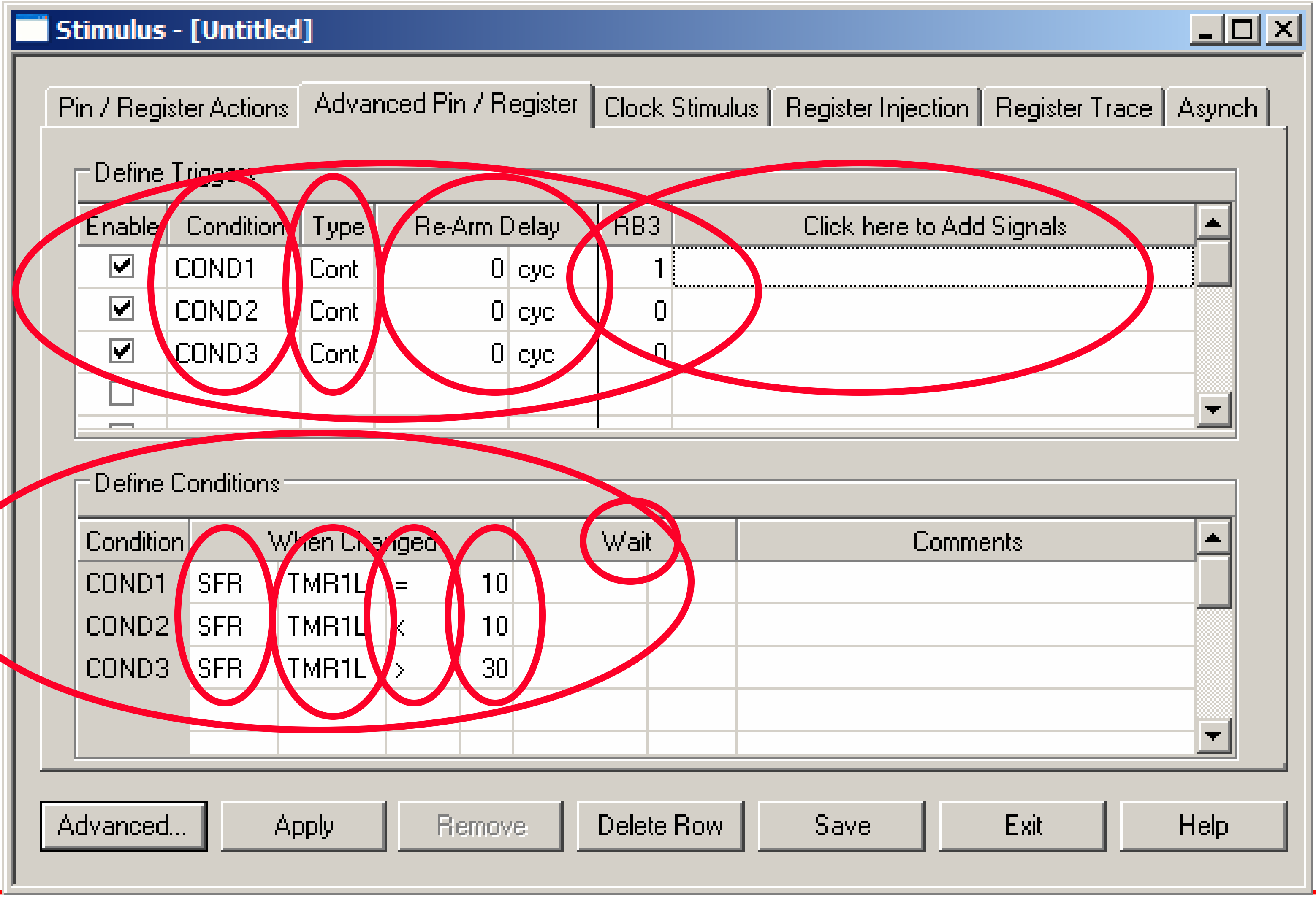
Waveform Generation Irregular



- **Execution code can affect the clock generation**
  - **Use Advanced Pin/Register to define conditions**
  - **Conditions can be based on Pin state, SFR value or bit value**

Waveform Generation Irregular

- **Code driven irregular pattern using conditions**



Define Triggers

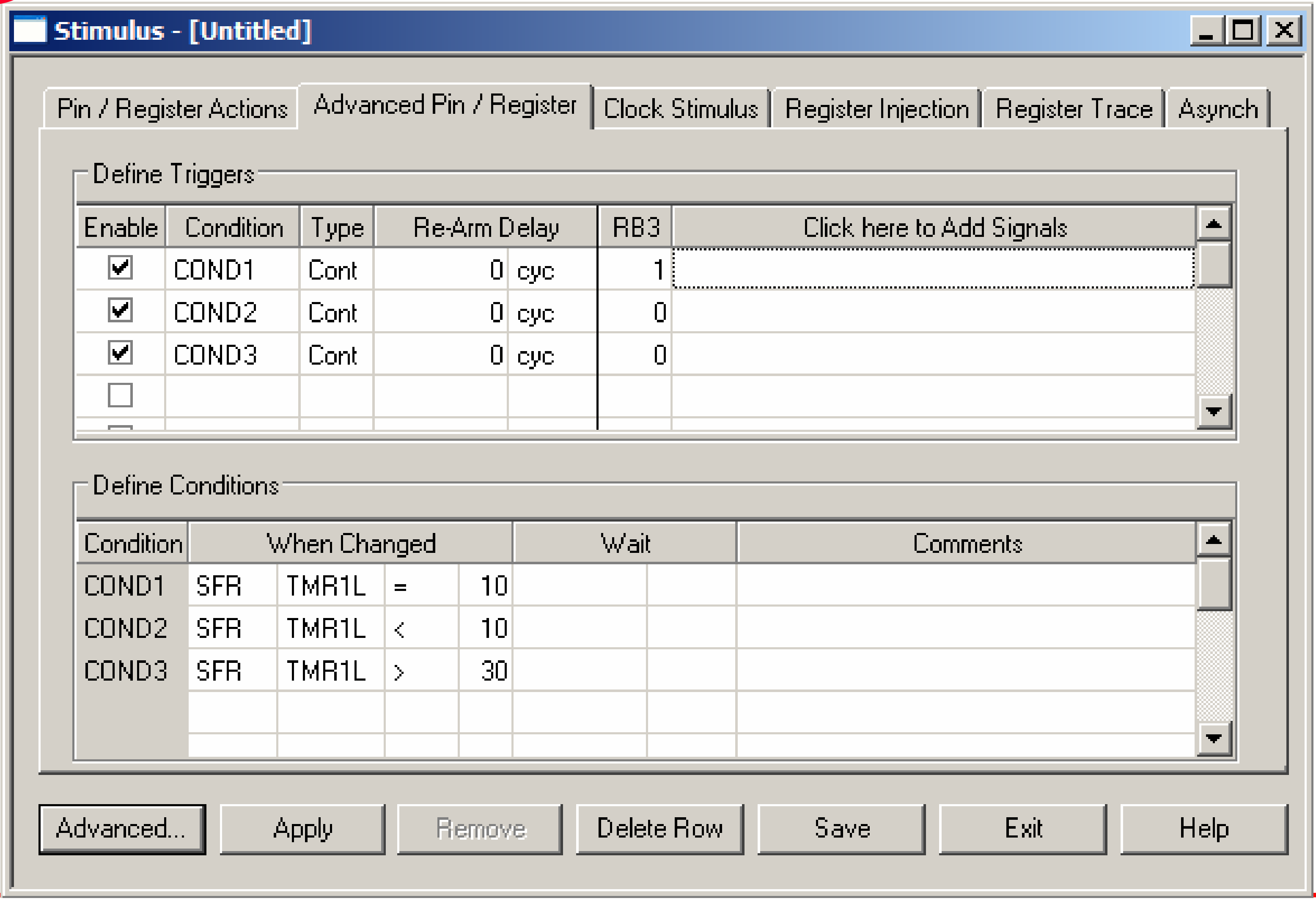
Enable	Condition	Type	Re-Arm Delay		RB3	Click here to Add Signals
☑	COND1	Cont	0	cyc	1	
☑	COND2	Cont	0	cyc	0	
☑	COND3	Cont	0	cyc	0	
□						

Define Conditions

Condition	When Changed				Wait	Comments
COND1	SFR	TMR1L	=	10		
COND2	SFR	TMR1L	<	10		
COND3	SFR	TMR1L	>	30		

Waveform Generation Synch.

- OR Use the conditional trigger to control a synch clock



Define Triggers

Enable	Condition	Type	Re-Arm Delay		RB3	Click here to Add Signals
☑	COND1	Cont	0	cyc	1	
☑	COND2	Cont	0	cyc	0	
☑	COND3	Cont	0	cyc	0	
□						

Define Conditions

Condition	When Changed				Wait		Comments
COND1	SFR	TMR1L	=	10			
COND2	SFR	TMR1L	<	10			
COND3	SFR	TMR1L	>	30			

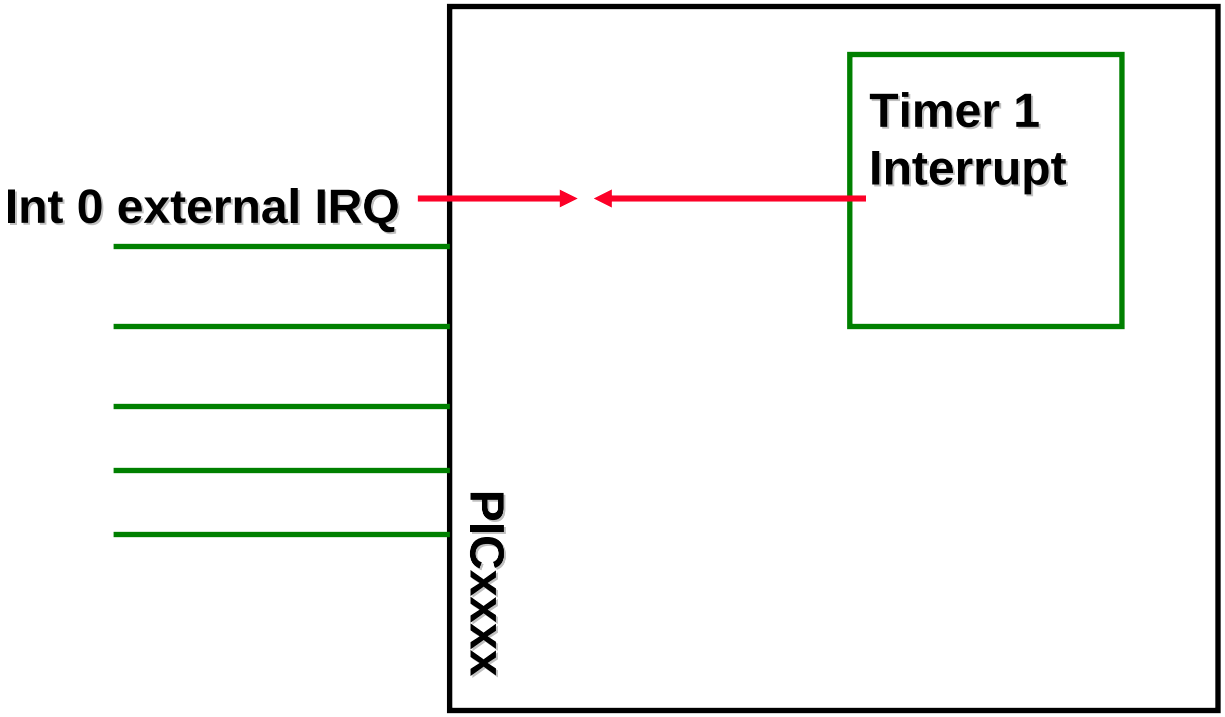
Conditional Stimulus

- **Conditional stimulus is an advanced feature useful for:**
  - Modifying bits or bit fields
  - Modifying IO pins
  - Modifying SFRs
- **Conditions based on**
  - SFR values
  - IO pin values
  - Bit or multi bit field values
  - True always

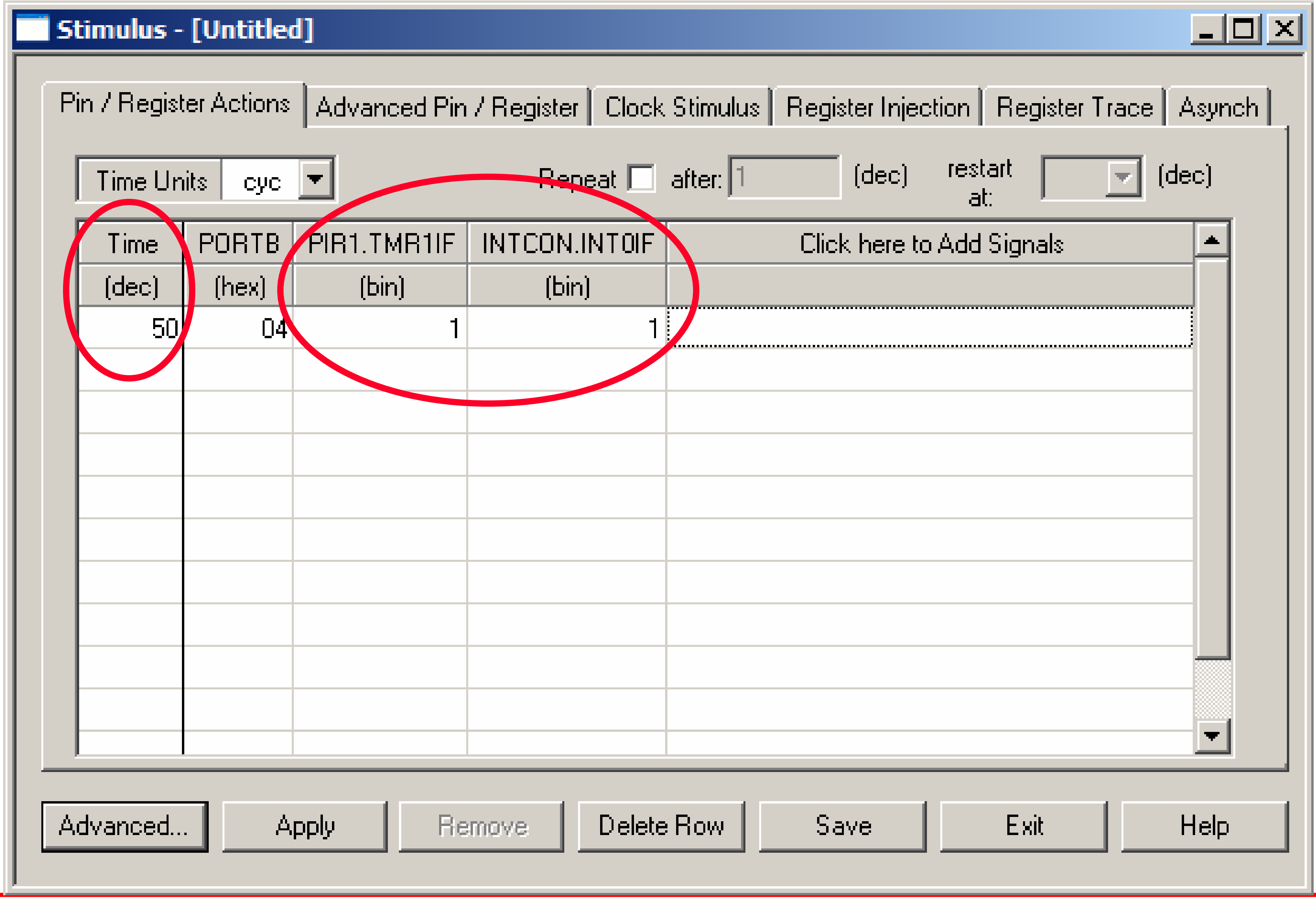
Stimulus for Load Testing

Load Testing

- Service multiple Interrupts and ensure priority is handled correctly



Load Testing

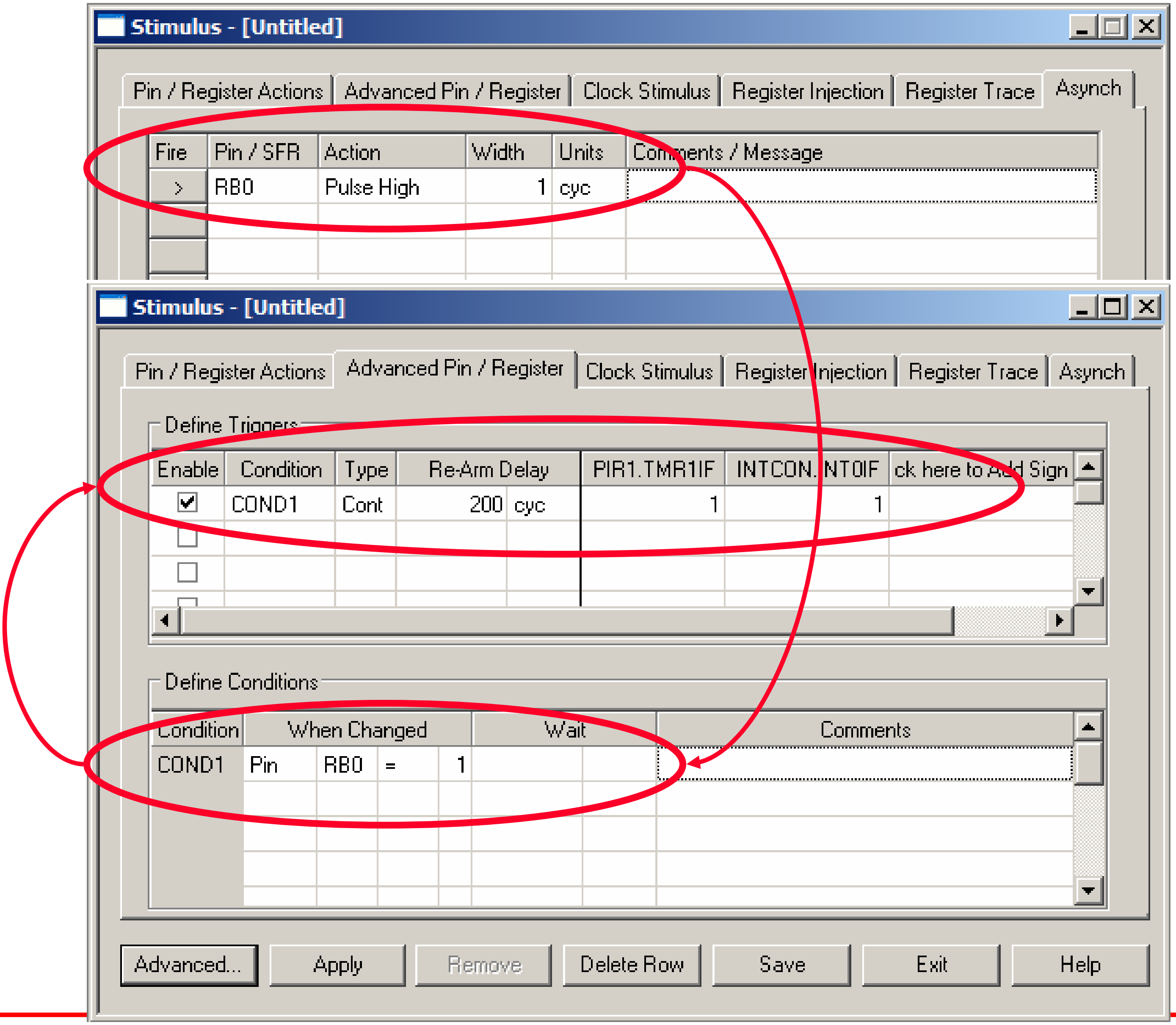


Time Units: cyc

Repeat ☐ after: 1 (dec) restart at: (dec)

Time	PORTB	PIR1.TMR1IF	INTCON.INT0IF	Click here to Add Signals
(dec)	(hex)	(bin)	(bin)	
50	04	1	1	

Load Testing



Fire	Pin / SFR	Action	Width	Units	Comments / Message
>	RB0	Pulse High	1	cyc	

Define Triggers

Enable	Condition	Type	Re-Arm Delay		PIR1.TMR1IF	INTCON.INT0IF	ck here to Add Sign
☑	COND1	Cont	200	cyc	1	1	
□							
□							
□							

Define Conditions

Condition	When Changed				Wait		Comments
COND1	Pin	RB0	=	1			

Demo 3

- **Simultaneous Interrupts for load testing**

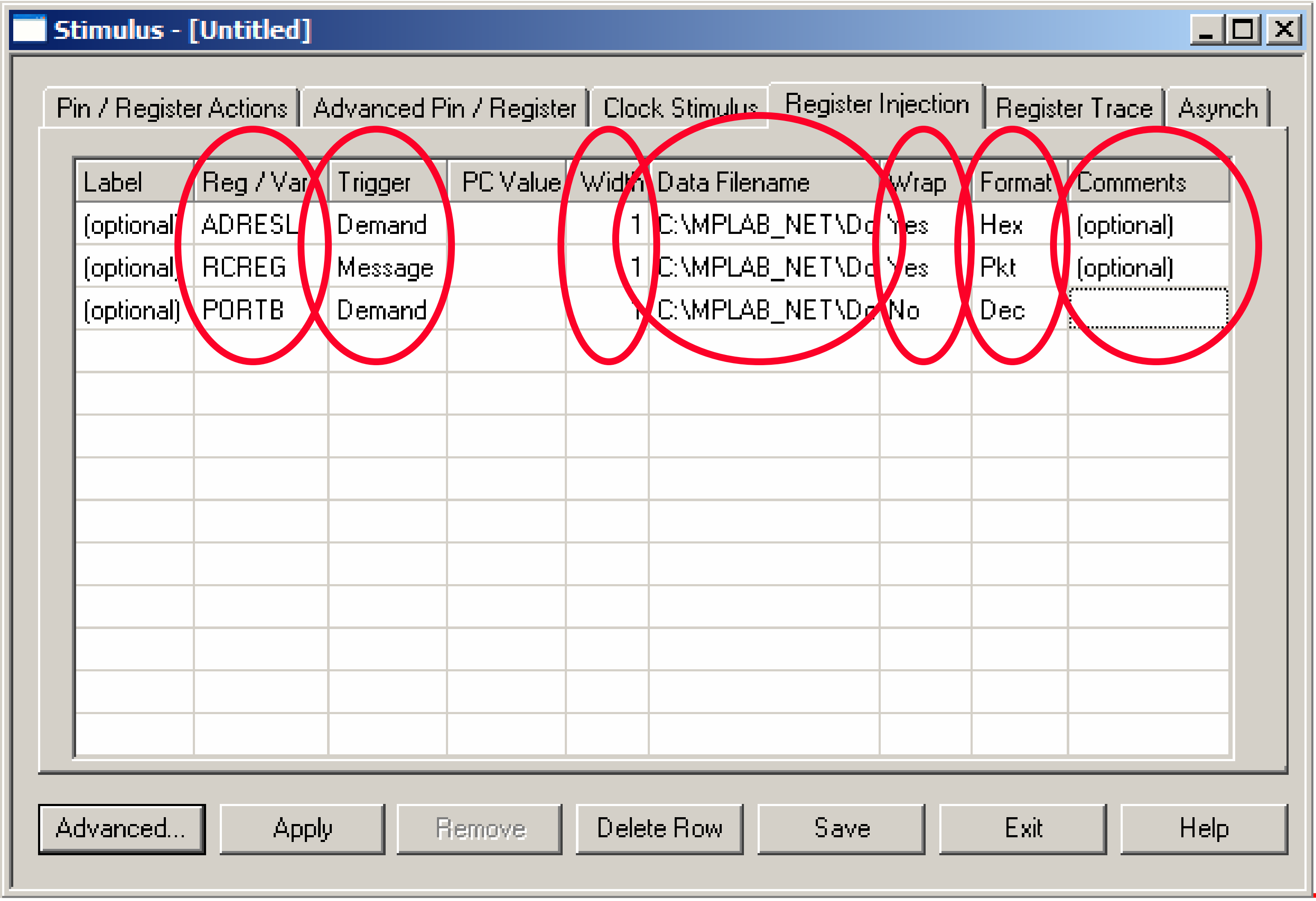
Stimulus Peripheral Data Injection

Peripheral Data Injection

- ADC data injection to test sensor input values
- SPI data injection for external modules
- UART data injection for communication
- Provide different values on IO ports for each read
- In all cases the data files are white space delimited text files

Peripheral Data Injection

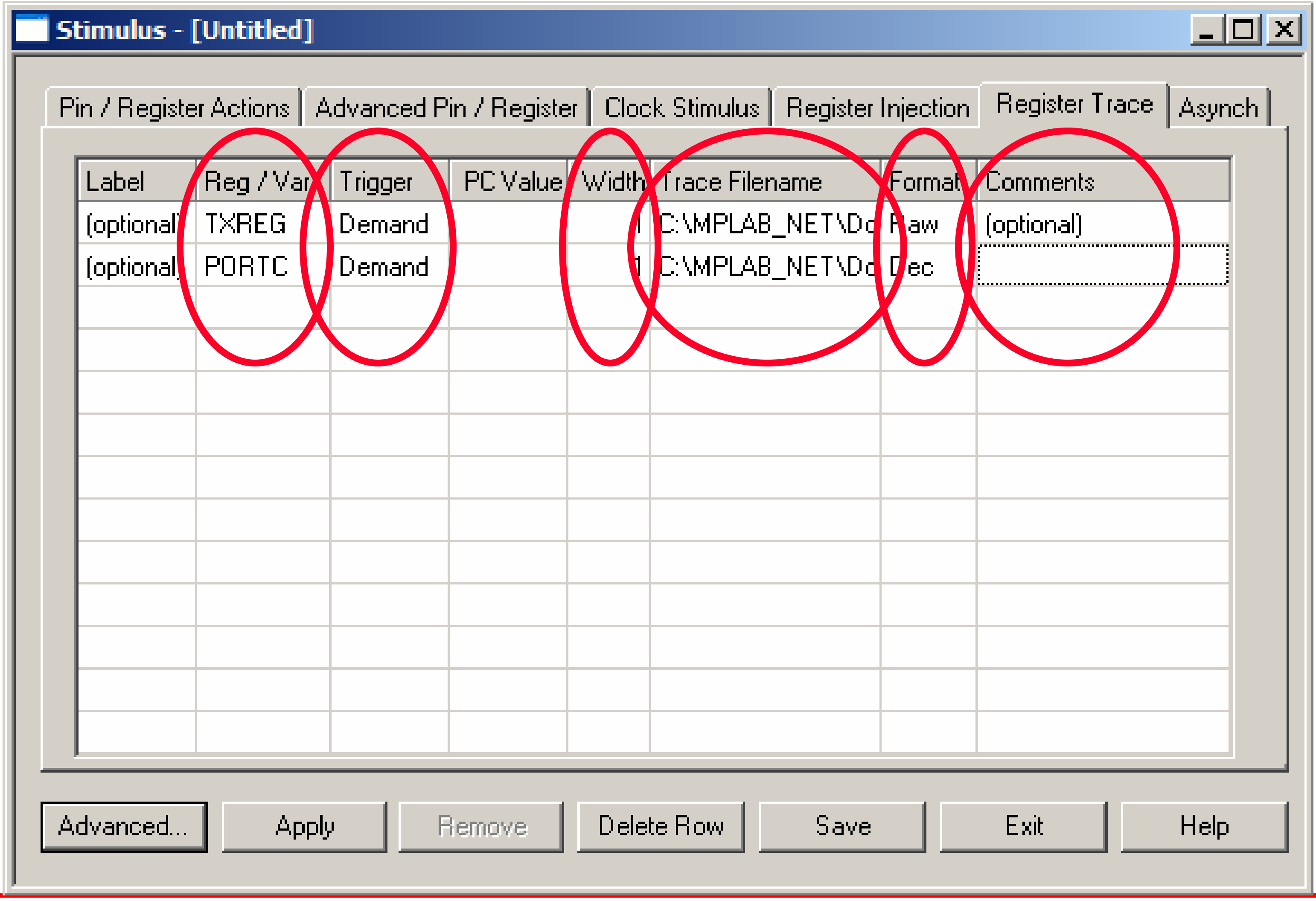
- **Attach data file to the SFR of the peripheral or to a Port**



Label	Reg / Var	Trigger	PC Value	Width	Data Filename	Wrap	Format	Comments
(optional)	ADRESL	Demand		1	C:\MPLAB_NET\Do	Yes	Hex	(optional)
(optional)	RCREG	Message		1	C:\MPLAB_NET\Do	Yes	Pkt	(optional)
(optional)	PORTB	Demand		1	C:\MPLAB_NET\Do	No	Dec	

Peripheral Data Trace

- **Attach data file to the SFR of the peripheral or to a Port**



Label	Reg / Var	Trigger	PC Value	Width	Trace Filename	Format	Comments
(optional)	TXREG	Demand		1	C:\MPLAB_NET\Dc	Raw	(optional)
(optional)	PORTC	Demand		1	C:\MPLAB_NET\Dc	Dec	

Demo 4

- Data injection into SPI BUFF
  Trace onto UART output
- Asynch message to UART
  Start conversion on ADC
  Data injection into ADC

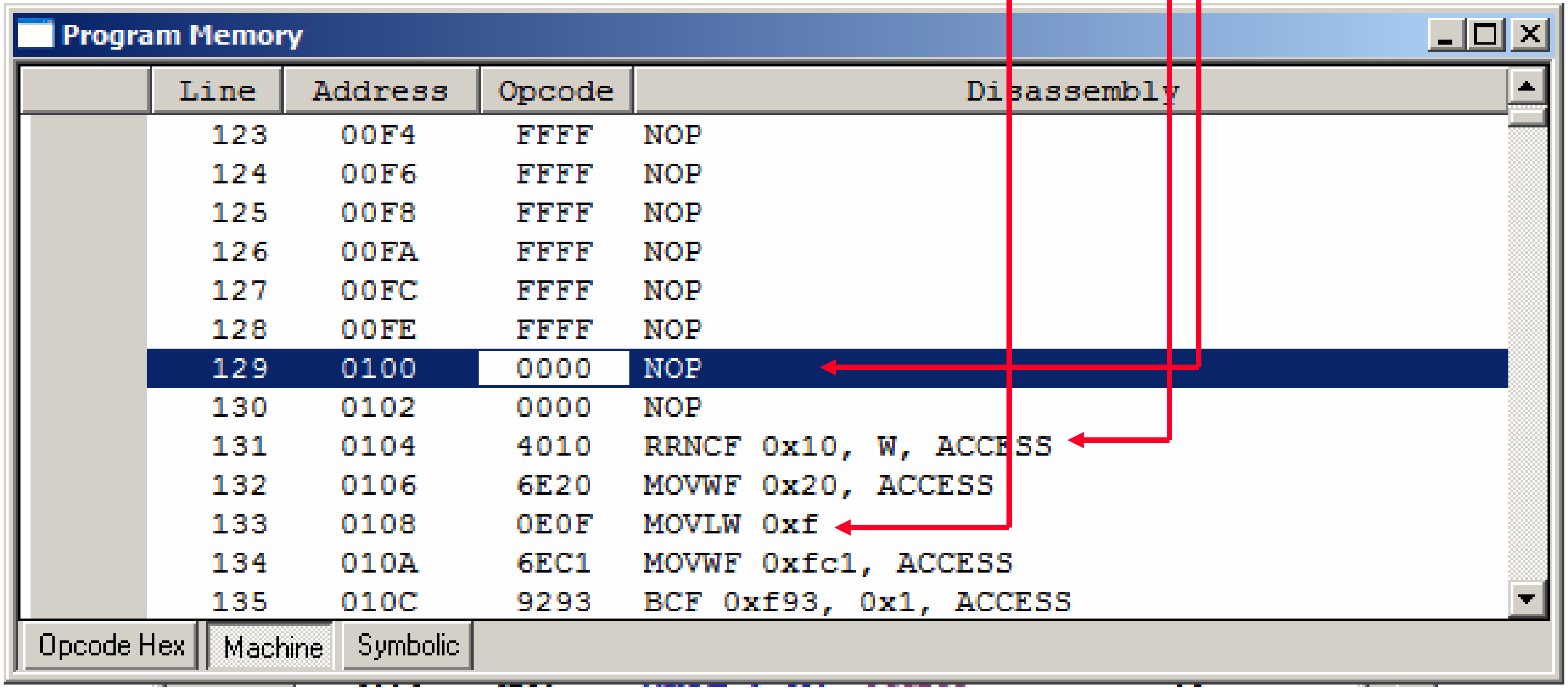
Stimulus Algorithm Verification

Algorithm Verification

- Inject data into a general purpose register at a specific time
- Log output from a general purpose register at a specific time
- Between each injection and log perform some function which needs to be verified

Algorithm Verification

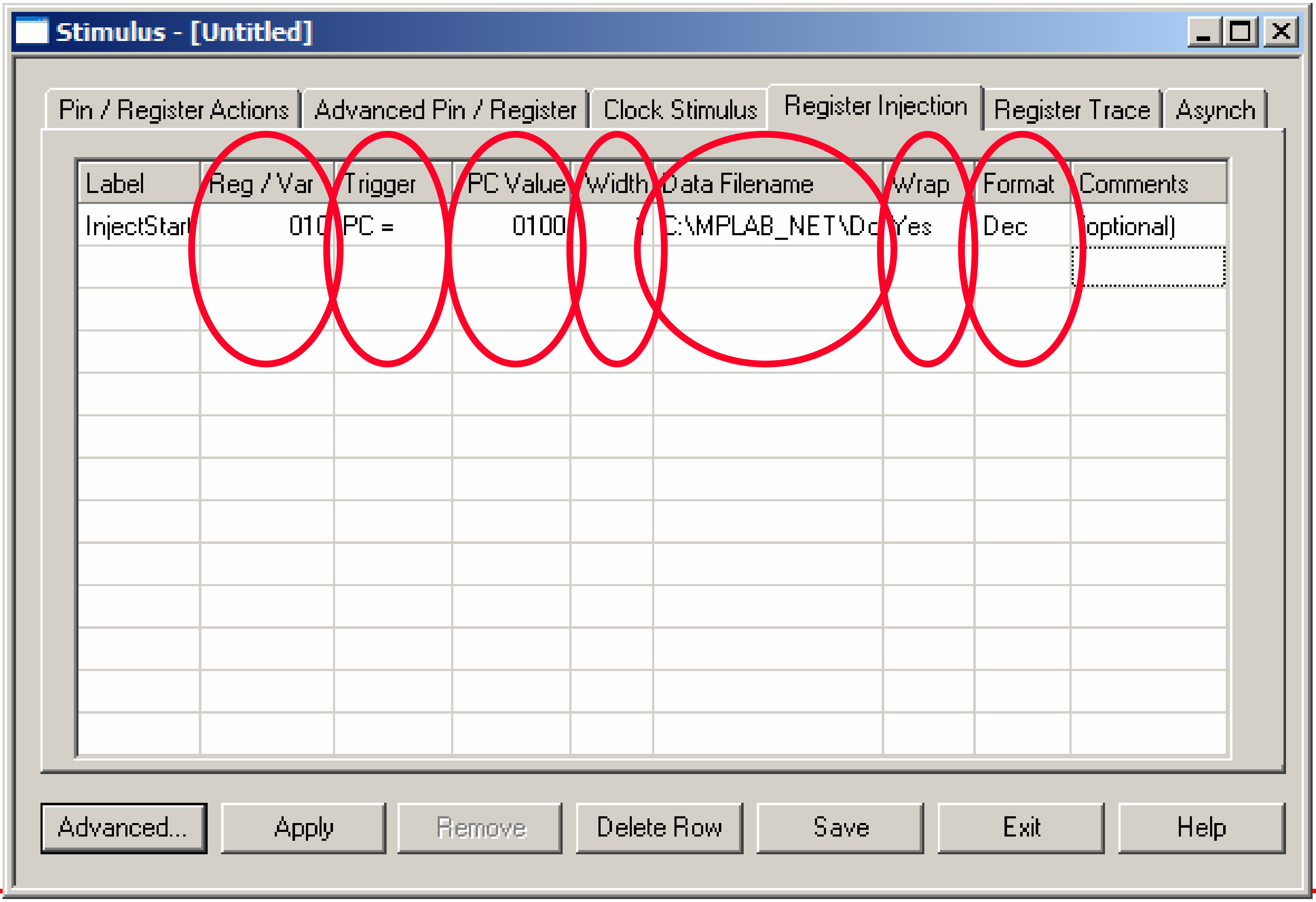
- Inject data at address 0x100
- Perform algorithm (RRNCF) 0x104
- Trace data at address 0x108



Line	Address	Opcode	Disassembly
123	00F4	FFFF	NOP
124	00F6	FFFF	NOP
125	00F8	FFFF	NOP
126	00FA	FFFF	NOP
127	00FC	FFFF	NOP
128	00FE	FFFF	NOP
129	0100	0000	NOP
130	0102	0000	NOP
131	0104	4010	RRNCF 0x10, W, ACCESS
132	0106	6E20	MOVWF 0x20, ACCESS
133	0108	0E0F	MOVLW 0xf
134	010A	6EC1	MOVWF 0xfc1, ACCESS
135	010C	9293	BCF 0xf93, 0x1, ACCESS

Algorithm Verification

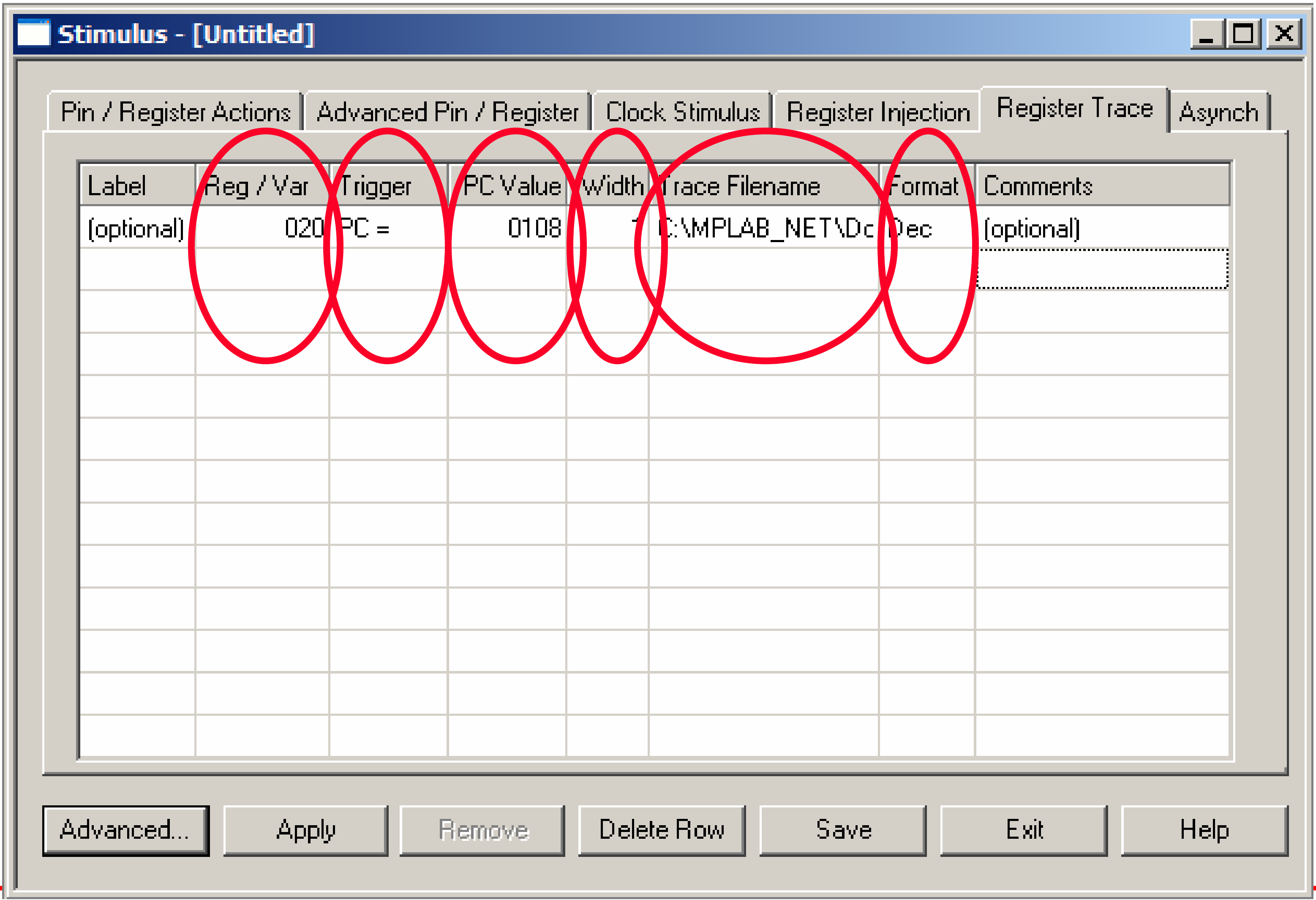
- Attach data input file to input GPR address or symbol



Label	Reg / Var	Trigger	PC Value	Width	Data Filename	Wrap	Format	Comments
InjectStart	010	PC =	0100	1	C:\MPLAB_NET\Dc	Yes	Dec	(optional)

Algorithm Verification

- **Attach output data file to output GPR address or symbol**



Label	Reg / Var	Trigger	PC Value	Width	Trace Filename	Format	Comments
(optional)	020	PC =	0108	1	C:\MPLAB_NET\Dc	Dec	(optional)

Demo 5

- Algorithm Verification

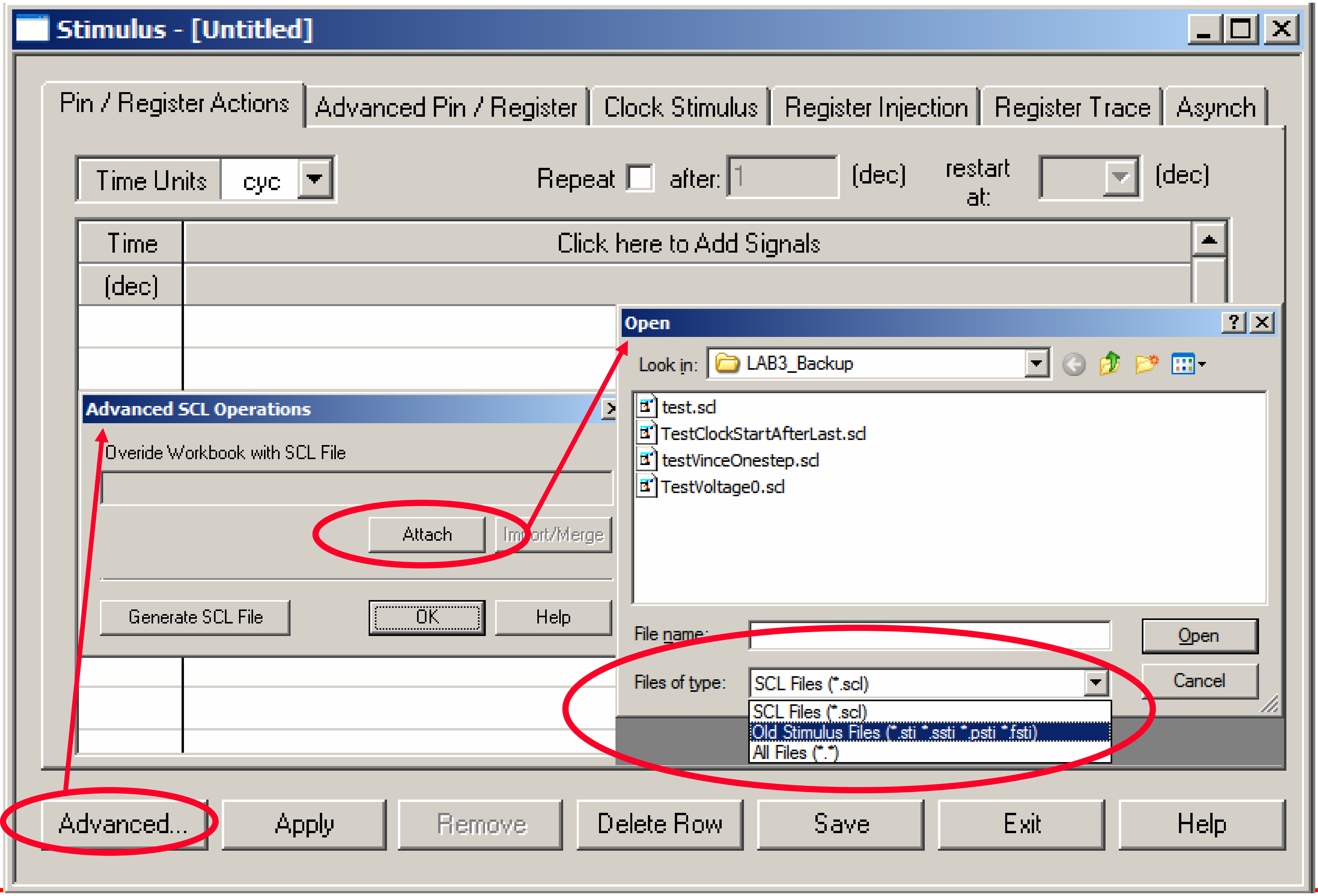
Simple
Pin Stimulus File
Format

Other Formats for Pin Injection

- Pin stimulus supports a simpler format
- Use a .sti extension to allow conversion

```
cycle  RD3      RD2     RD1     RD0     RB0 ;Use this file for 18F458
5        0      0       0       0       0    ; Initialize pin
360      0      0       0       1       0    ; Set C1Vin+,C1Vin+ > C1Vin-
370      0      0       0       1       0    ; comp1
400    0  0     0       1       0       ;
410    0  0     1       0       0   ; Clear C1Vin+,C1Vin+ < C1Vin-
…………..
520    0  0     1       0       0
530    0        0       1       0       0    ; comp2
540    0  1     0       0       0   ; Set C2Vin+,C2Vin+ > C2Vin-
```

Using Stimulus Files



Stimulus - [Untitled]

Time Units: cyc

Repeat □ after: 1 (dec) restart at: (dec)

Advanced SCL Operations

Overide Workbook with SCL File

Open

Look in: LAB3_Backup

- test.scl
- TestClockStartAfterLast.scl
- testVinceOnestep.scl
- TestVoltage0.scl

File name:

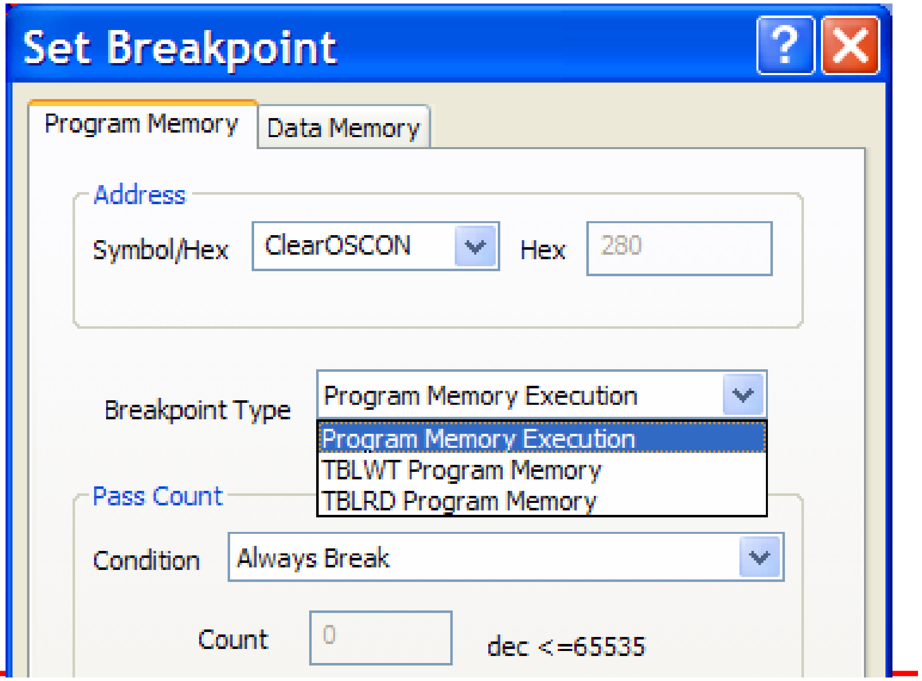
Files of type: SCL Files (*.scl)

- SCL Files (*.scl)
- Old Stimulus Files (*.sti *.ssti *.psti *.fsti)
- All Files (*.*)

Complex Breakpoints

Definition of a Complex Breakpoint

- Complex breakpoints in simulator are modeled as in real silicon. This makes operation easy when switching debuggers
- Traditional breakpoints halt the simulator prior to execution of an opcode at an address in program space
- Complex breakpoints can do that, but can also halt the simulator on read/writes of program or data space



Set Breakpoint

Address

Symbol/Hex: ClearOSCON

Hex: 280

Breakpoint Type: Program Memory Execution

- Program Memory Execution
- TBLWT Program Memory
- TBLRD Program Memory

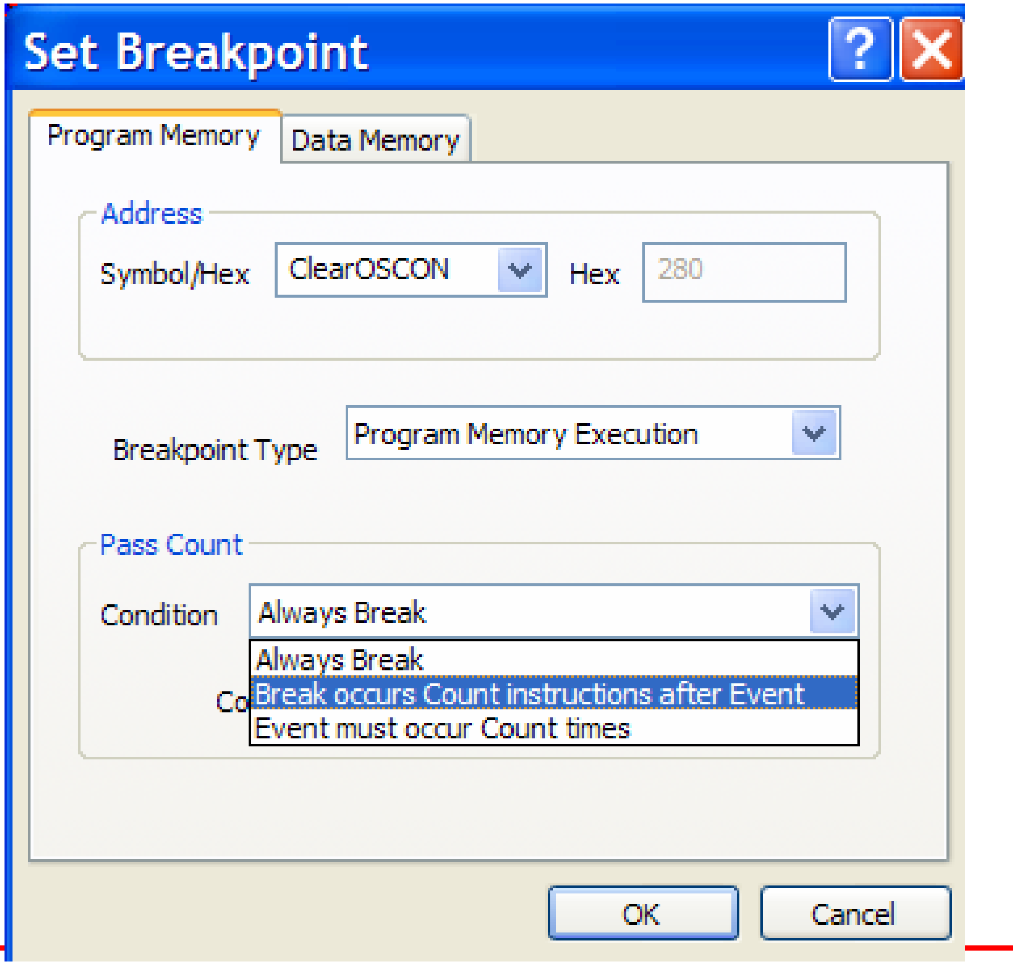
Pass Count

Condition: Always Break

Count: 0 dec <=65535

Definition of a Complex Breakpoint

- **Complex breakpoints have a count associated with them that can be used in two different ways**
  - break N instructions after the event is seen
  - break when this event is seen N times



Address

Symbol/Hex: ClearOSCON

Hex: 280

Breakpoint Type: Program Memory Execution

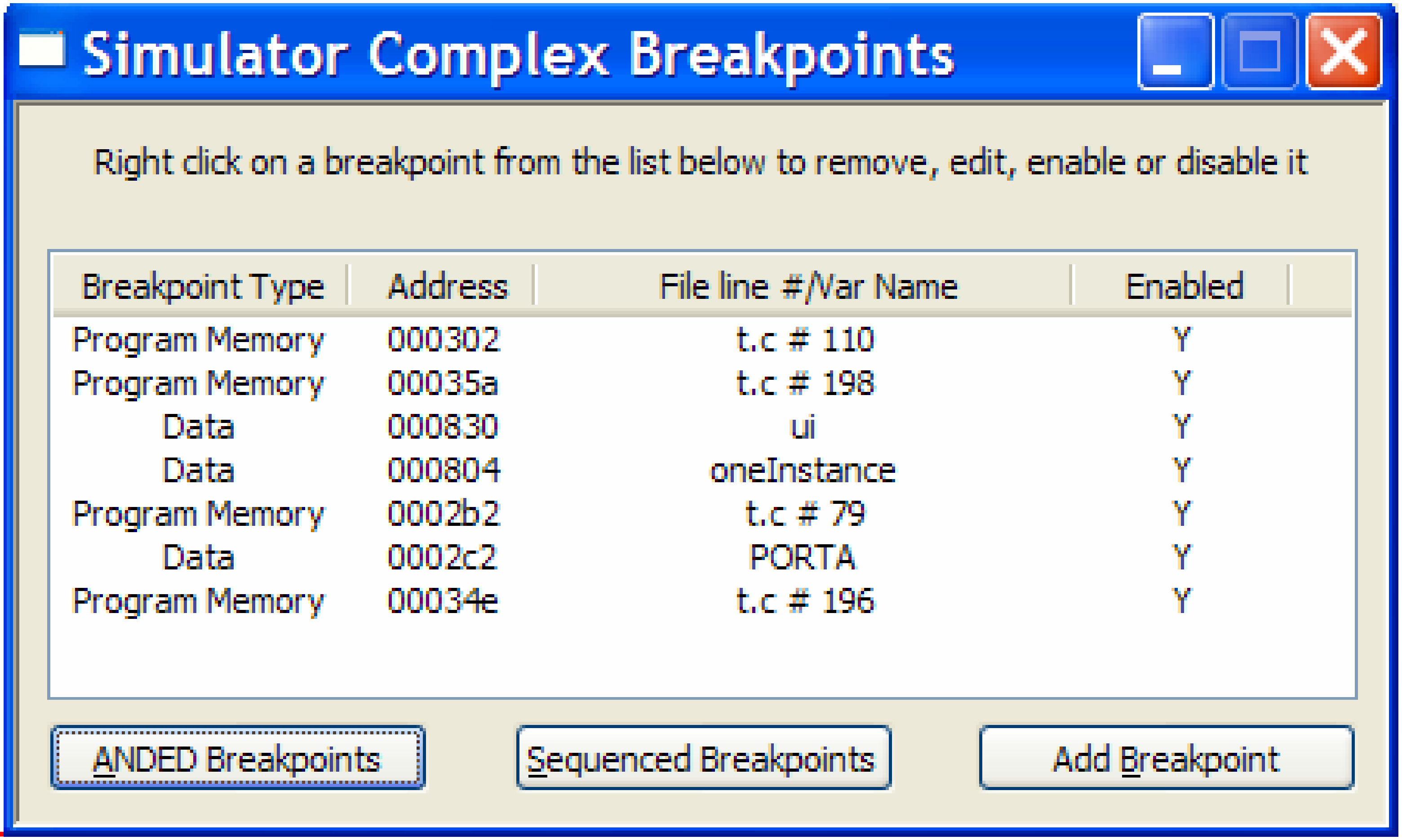
Pass Count

Condition: Always Break

Always Break
Break occurs Count instructions after Event
Event must occur Count times

Breakpoints are part of the workspace

- **Any defined breakpoint will halt CPU execution when the breakpoint conditions are met**



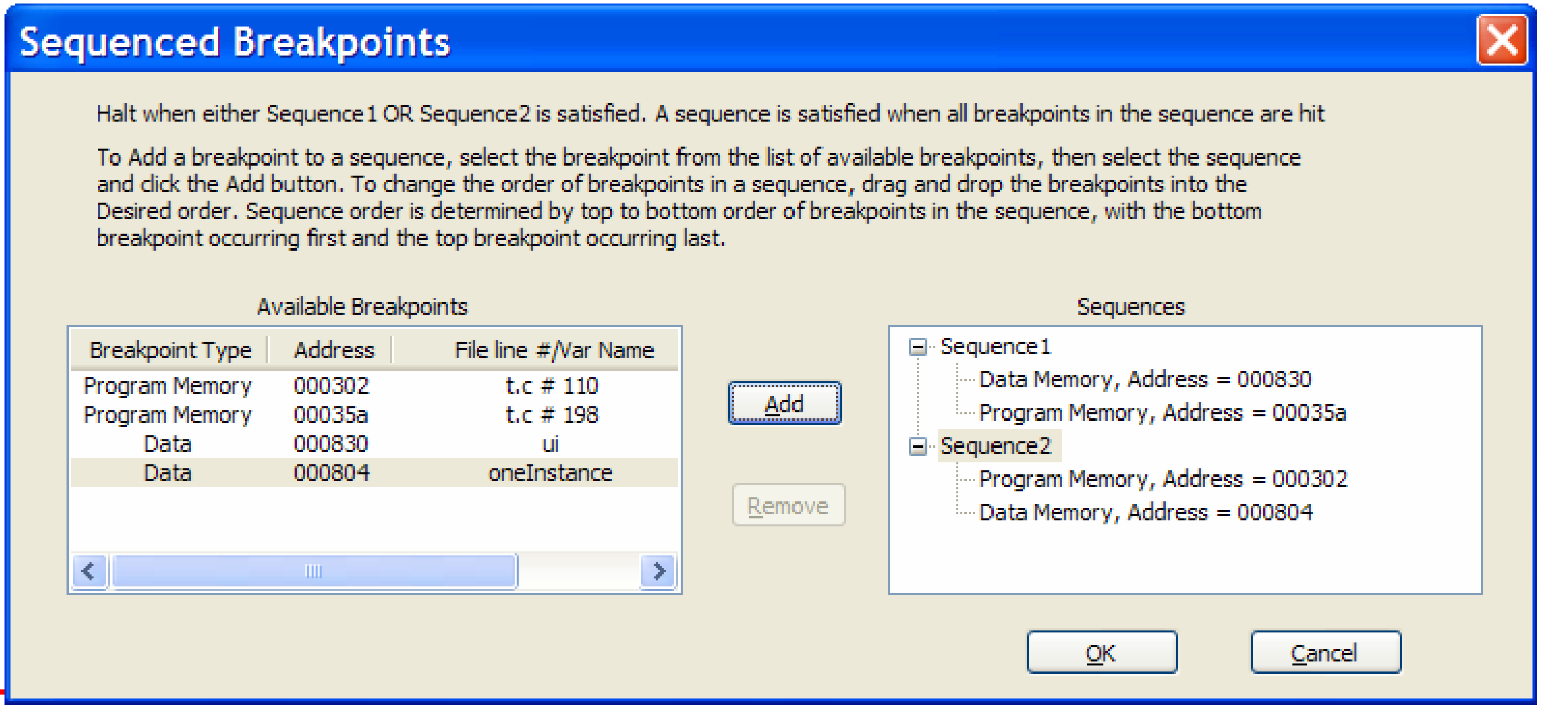
Simulator Complex Breakpoints

Right click on a breakpoint from the list below to remove, edit, enable or disable it

Breakpoint Type	Address	File line #/Var Name	Enabled
Program Memory	000302	t.c # 110	Y
Program Memory	00035a	t.c # 198	Y
Data	000830	ui	Y
Data	000804	oneInstance	Y
Program Memory	0002b2	t.c # 79	Y
Data	0002c2	PORTA	Y
Program Memory	00034e	t.c # 196	Y

Group Breakpoints into a Sequence

- **Multiple complex breakpoints can be combined into sequences: break only when all the breakpoints have been seen in a given sequence**
- **Two sequences are monitored simultaneously. If either sequence is satisfied, the simulator breaks**



Sequenced Breakpoints

Halt when either Sequence1 OR Sequence2 is satisfied. A sequence is satisfied when all breakpoints in the sequence are hit

To Add a breakpoint to a sequence, select the breakpoint from the list of available breakpoints, then select the sequence and click the Add button. To change the order of breakpoints in a sequence, drag and drop the breakpoints into the Desired order. Sequence order is determined by top to bottom order of breakpoints in the sequence, with the bottom breakpoint occurring first and the top breakpoint occurring last.

Available Breakpoints

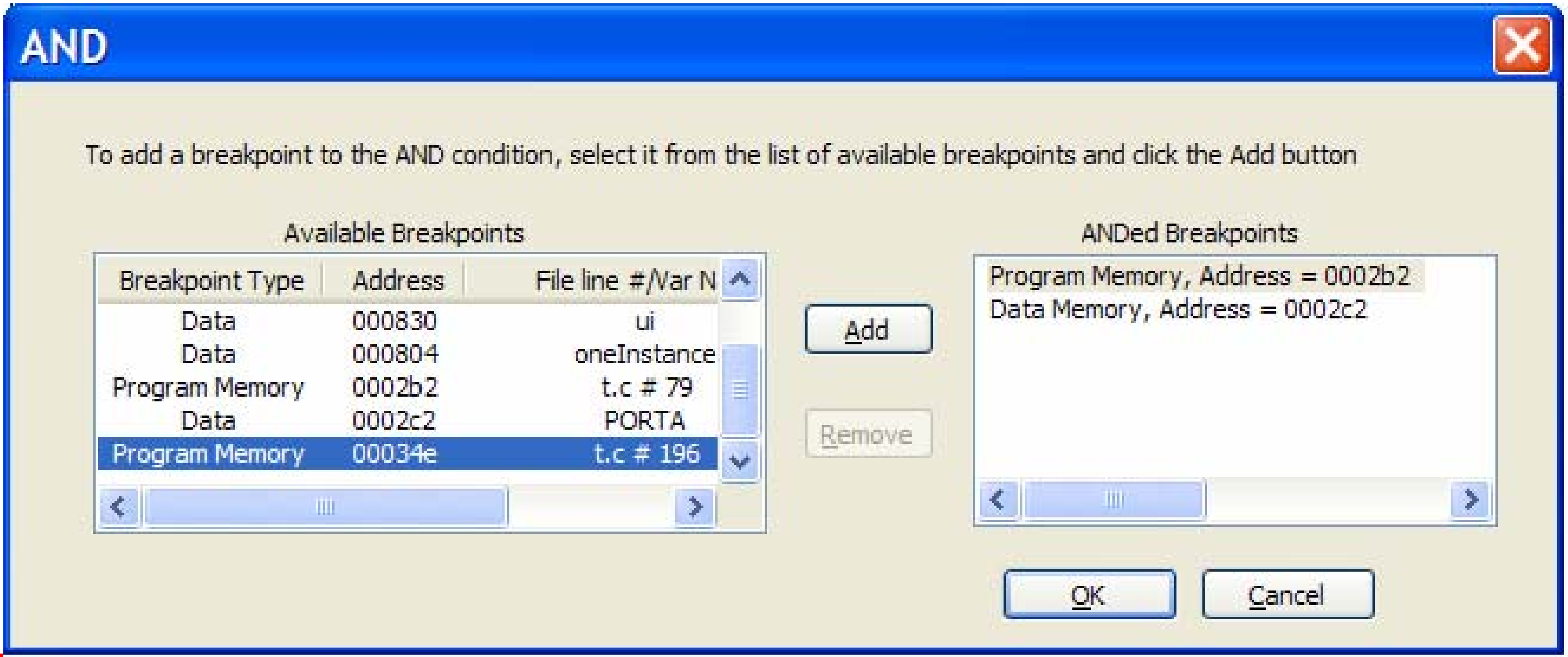
Breakpoint Type	Address	File line #/Var Name
Program Memory	000302	t.c # 110
Program Memory	00035a	t.c # 198
Data	000830	ui
Data	000804	oneInstance

Sequences

- Sequence1
  - Data Memory, Address = 000830
  - Program Memory, Address = 00035a
- Sequence2
  - Program Memory, Address = 000302
  - Data Memory, Address = 000804

AND Breakpoints Together

- **Two breakpoints can halt execution when they happen at exactly the same time. This can only happen when a read/write to either program/data space is combined with the execution of a given address**



AND

To add a breakpoint to the AND condition, select it from the list of available breakpoints and click the Add button

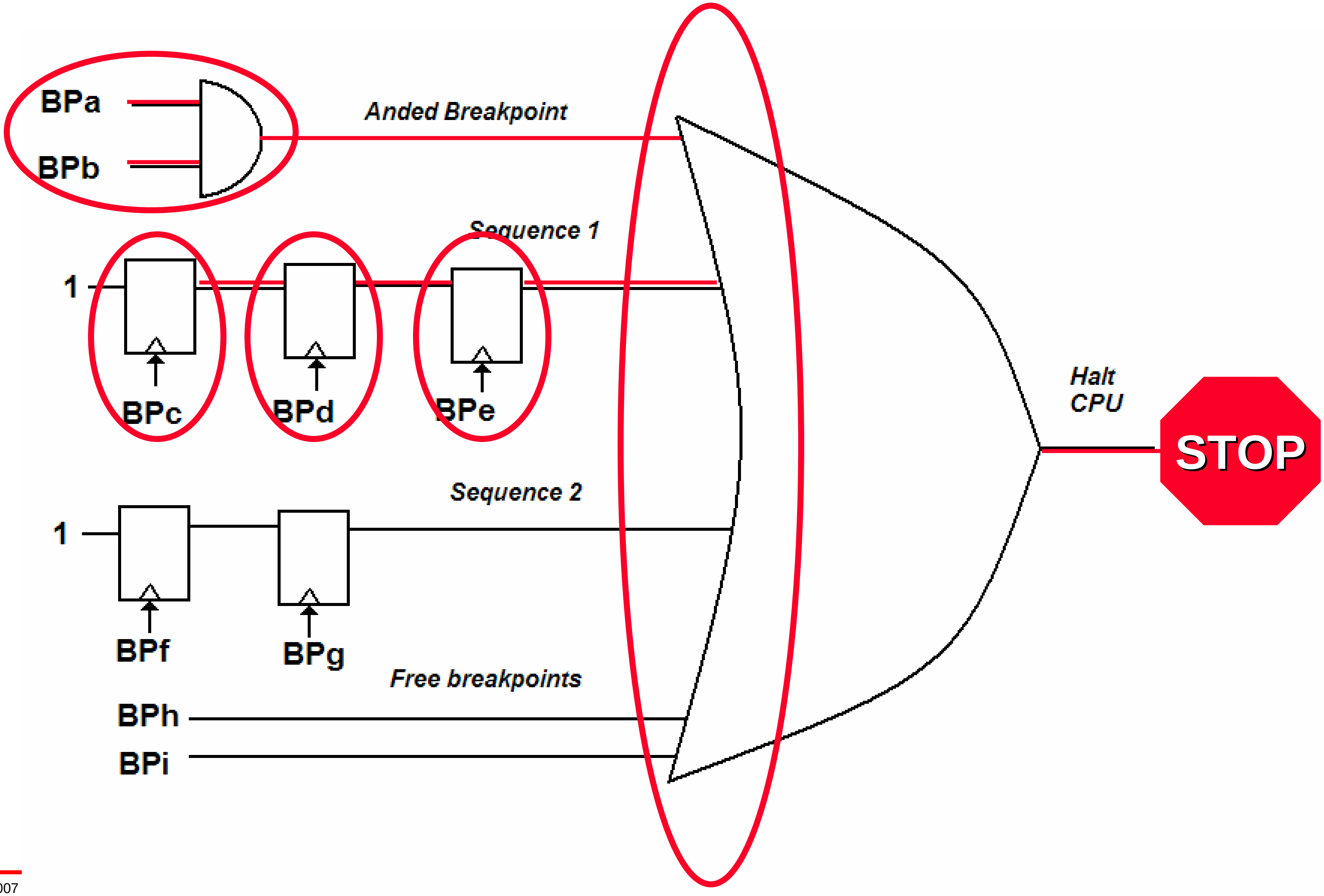
Available Breakpoints

Breakpoint Type	Address	File line #/Var N
Data	000830	ui
Data	000804	oneInstance
Program Memory	0002b2	t.c # 79
Data	0002c2	PORTA
Program Memory	00034e	t.c # 196

ANDed Breakpoints

Program Memory, Address = 0002b2
Data Memory, Address = 0002c2

Example: halt when



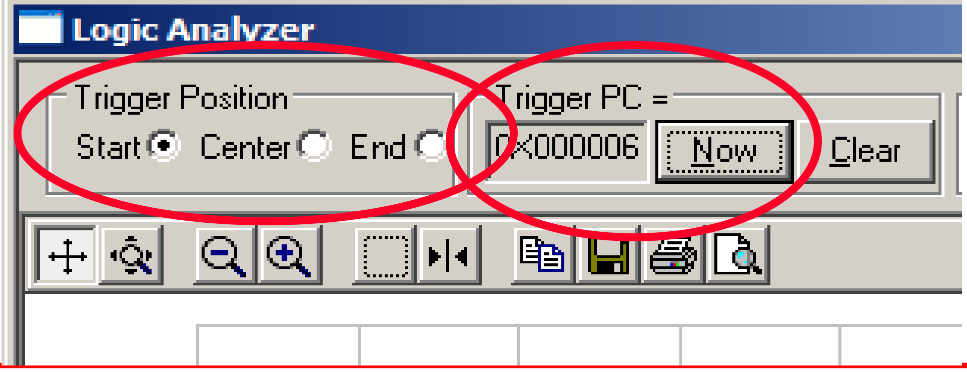
Demo 6

- Complex Breakpoint using Table Write

Using Instruction Trace Effectively

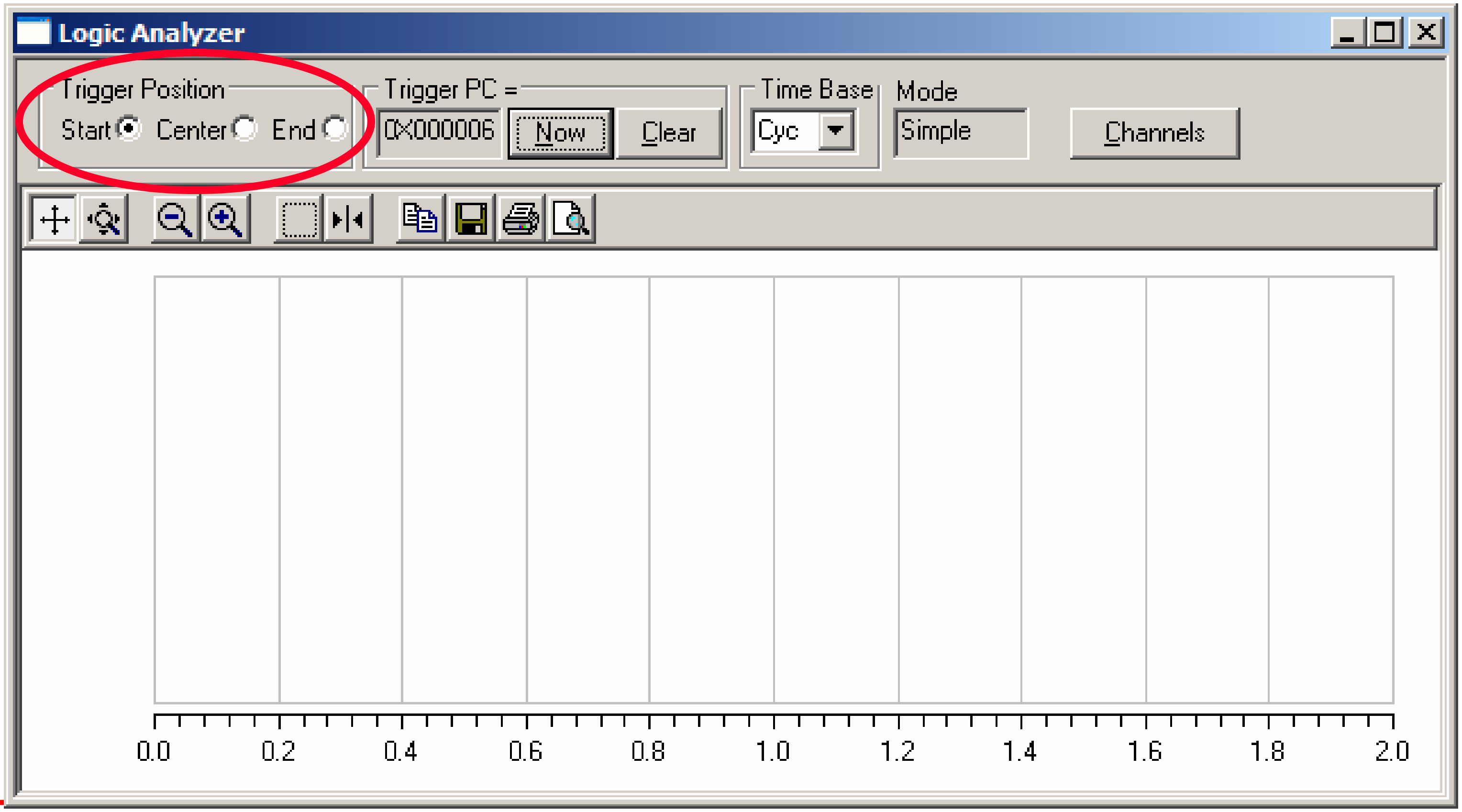
Effectively Capturing Instruction Trace

- **Triggers**
  - Currently possible to set a simple trigger (PC) to start Instruction trace capture
  - Access from Logic Analyzer
  - Set a breakpoint on trigger location in code OR right click in editor, use “Set PC at Cursor”
  - Set trigger
  - Set buffer collection format
  - Clear breakpoint from code



Instruction Trace (Trigger)

- **Start: {10 records, Trigger, remainder of Buffer}**
- **Center: {half buffer, Trigger, half buffer}**
- **End: {most of buffer, Trigger, 10 records}**



Logic Analyzer

Trigger Position: Start ● Center ○ End ○

Trigger PC = 0X000006

Time Base: Cyc

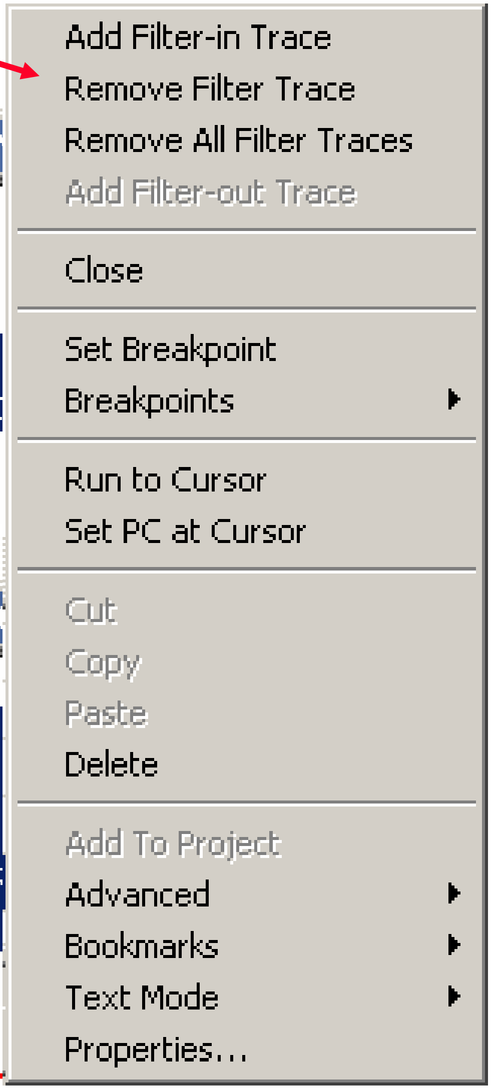
Mode: Simple

Effectively Capturing Instruction Trace

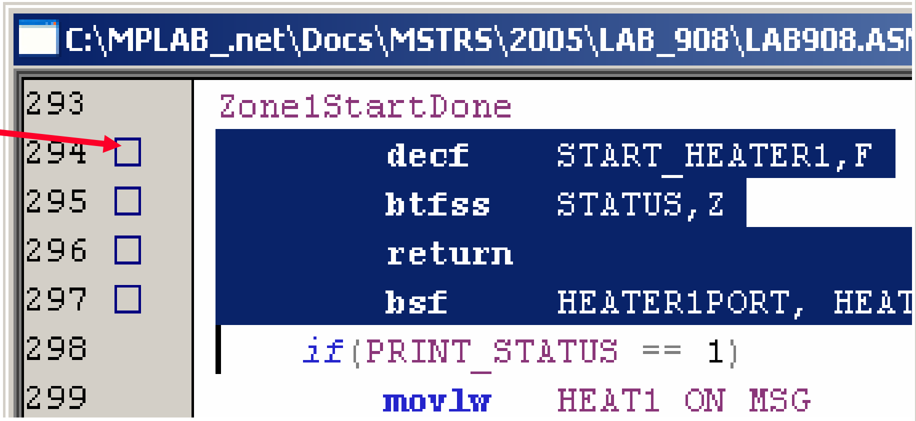
- **Filters**
  - Use when there are large delay loops or specific areas to focus on
  - Eliminates the need to have large trace buffers
  - Filter Out will trace all EXCEPT the selected lines
  - Filter In will only trace the SELECTED lines

Instruction Trace (Filters)

- **Right mouse click in editor for context menu**

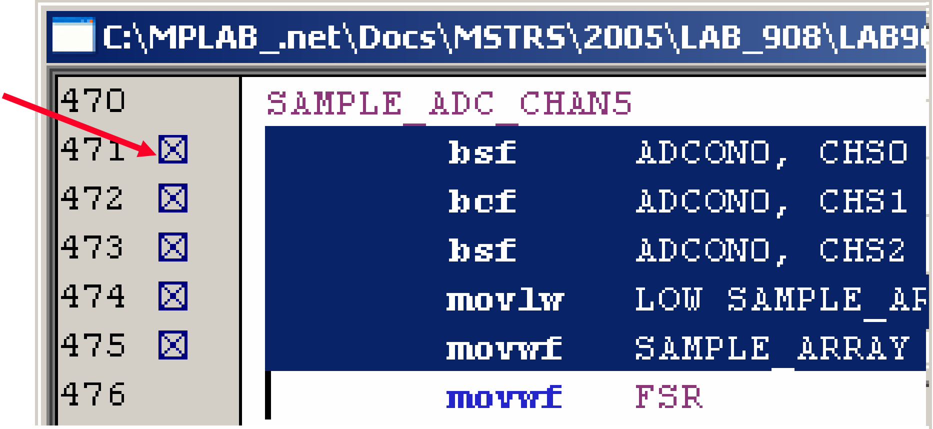


- **Filter-In** **Use to trace specific loops**



```
Zone1StartDone
        decf     START_HEATER1,F
        btfss    STATUS,Z
        return
        bsf      HEATER1PORT, HEAT
    if(PRINT_STATUS == 1)
        movlw    HEAT1 ON MSG
```

- **Filter-Out** **Use to eliminate wait loops**



```
SAMPLE_ADC_CHAN5
        bsf      ADCON0, CHS0
        bcf      ADCON0, CHS1
        bsf      ADCON0, CHS2
        movlw    LOW SAMPLE_AF
        movwf    SAMPLE_ARRAY
        movwf    FSR
```

Demo 7

- **Simple Trace Trigger**
- **Filter Trace**
  - Filter out loop
  - Filter In specific function

Log Text Data Without UART

Log Data Without UART

- **Currently under C18 and C30 there is ‘printf’ (sprintf) support to output log data using the UART**
- **Occasionally there may be no UARTS available**
    - Either the application uses both and the engineer wants to test them
    - A device doesn’t have a UART
- **Using Register trace an engineer can create a function to output log data**

UNIVERSITY OF MICROCHIP
UofM
MASTERs 2007

Log Data Without UART

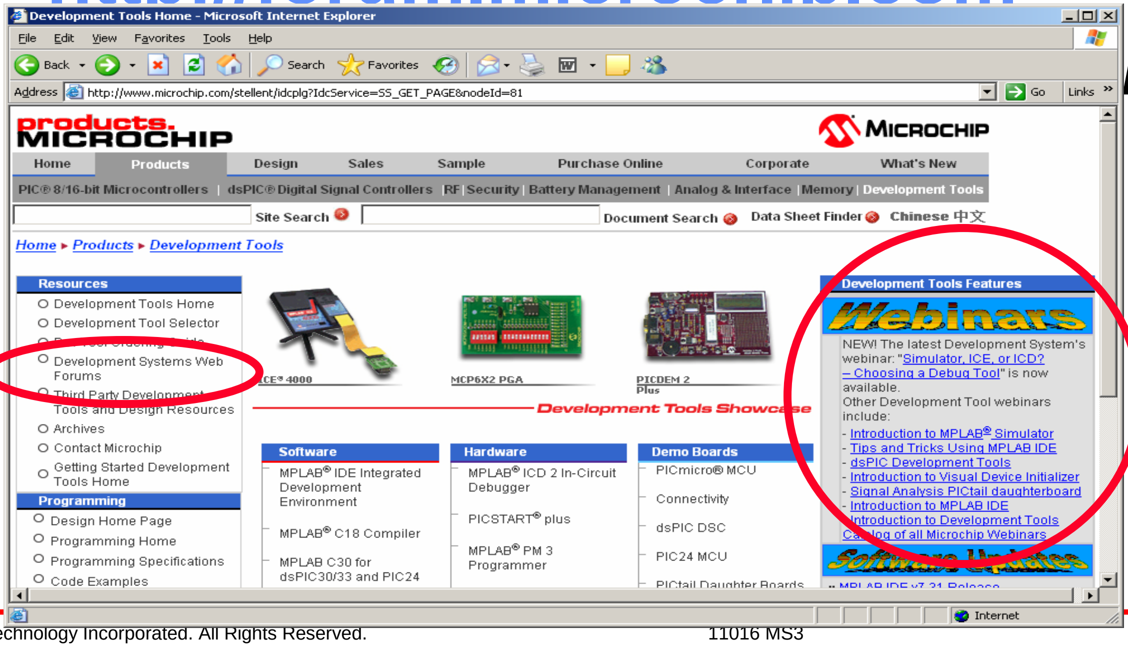
- **Easiest: Use an unused SFR in the application**
  - Set Register Trace on SFR for Demand and Raw format
    - SSPBUF
    - EEDATA
  - Write directly to SFR
  - Limitation on this is one Byte at a time (SFR data width)
- **More complex: Create a string in GPR**
  - Write data to the string then call null() function to trigger
  - Set Register Trace on GPR (string address) for PC = null() function. Set width to size of string and Raw format
  - When calling null function the Trace is triggered and the GPR of size width is written to the file

Demo 8

- **Logging data without UART**

Where to find out more

- MPLAB® IDE Help
- Appendix – Useful links & Demo Instructions
- Forums / Webinars
  - http://forum.microchip.com
  - /webseminars


Development Tools Home - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=81
Go Links
products. MICROCHIP
MICROCHIP
Home Products Design Sales Sample Purchase Online Corporate What's New
PIC® 8/16-bit Microcontrollers | dsPIC® Digital Signal Controllers | RF | Security | Battery Management | Analog & Interface | Memory | Development Tools
Site Search
Document Search
Data Sheet Finder
Chinese 中文
Home ▸ Products ▸ Development Tools
Resources
Development Tools Home
Development Tool Selector
Development Systems Web Forums
Third Party Development Tools and Design Resources
Archives
Contact Microchip
Getting Started Development Tools Home
Programming
Design Home Page
Programming Home
Programming Specifications
Code Examples
ICE® 4000
MCP6X2 PGA
PICDEM 2 Plus
Development Tools Showcase
Software
MPLAB® IDE Integrated Development Environment
MPLAB® C18 Compiler
MPLAB C30 for dsPIC30/33 and PIC24
Hardware
MPLAB® ICD 2 In-Circuit Debugger
PICSTART® plus
MPLAB® PM 3 Programmer
Demo Boards
PICmicro® MCU
Connectivity
dsPIC DSC
PIC24 MCU
PICtail Daughter Boards
Development Tools Features
Webinars
Software Updates
MPLAB IDE v7.31 Release
Internet

NEW! The latest Development System's webinar: "Simulator, ICE, or ICD? – Choosing a Debug Tool" is now available.

Other Development Tool webinars include:

- Introduction to MPLAB® Simulator
- Tips and Tricks Using MPLAB IDE
- dsPIC Development Tools
- Introduction to Visual Device Initializer
- Signal Analysis PICtail daughterboard
- Introduction to MPLAB IDE
- Introduction to Development Tools

Catalog of all Microchip Webinars

Summary

- Using Stimulus within the simulator combined with the Logic Analyzer to verify Stimulus generation
- Peripheral data injection
- Complex Breakpoints
- Using Instruction Trace Effectively
- Logging data
- Find out more

UNIVERSITY OF MICROCHIP
U OF M
MASTERS 2007

THANK YOU!

Appendix

- Uart packet data injection format
- Useful Links
- Demo Instructions
- SPI Setup

Uart data injection format

File data	**Asynch behavior**	**Synch behavior**
//comment line		
wait 5 ms	Wait for fire	Wait for 5 ms
“***the quick brown*** ”	Msg1 ***the quick brown fox***	Msg1 ***the quick brown fox***
“***fox***”		
wait 20 ms	Wait for fire	Wait for 20 ms
71 72 73 74 13 10	Msg2 ***GHIJ<CR><LF>***	
//comment line	Wait for fire	Msg2 ***GHIJ<CR><LF>KL***
75 76	Msg3 ***KL***	
wait 10 ns	Wait for fire	Wait 10 ns
32 33 34	Msg4 ***234mix***	Msg3 ***234mix***
“***mix***”		
rand 15 20 sec	Wait for fire	Wait Random time (15-20 seconds)
34 33 24 32 34	Msg5 ***43$24***	Msg4 ***43$24***
rand 0 100 min	Wait for fire	Wait Random time (0-100 minutes)
“***message x***”	Msg6 ***message x***	Msg5 ***message x***

Where to find out more

- **Other useful Links:**
  - Microchip Change Notification (good way to keep up to date on latest MPLAB® IDE and C18/C30 releases, as well as important Dev Tool notifications):
  - http://cn.microchip.com/sales/product_change.nsf
  - Microchip Dev Tools Getting Started (series of many tutorials and overviews):
  - http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2122
  - Microchip archives:
  - http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en023073
  - Development Tool Selector (to find out tool support, accessories, adapteres, etc.):
  - http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1496
  - Third Party Development Tools:
  - http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1926&type=-1&label=A
  - MPLAB IDE download page:
  - http://www.microchip.com/mplab

Demo 1 Synch clock

- **Open MPLAB® IDE**
  - Select “*Configure>Select Device*” menu item
  - Select a “pic18C442” device
  - Select “*Debugger>Select Tool>MPLAB SIM*”
- **Open Stimulus window**
  - Select “*Debugger>Settings*”
  - Select Trace All check box on “Osc\Trace” tab Select OK
  - Select “*Debugger>Stimulus>New Workbook*”
  - Select “Clock Stimulus” tab
  - Select “pin” with drop down list on first row “RB1” (Port B bit 1)
  - Select “initial” drop down list as “Low”
  - Enter “10” in “Low cycles”, enter “10” in “High cycles”
  - Select “Start” and “End” boxes to set the default
  - Add labels and comments if desired

Demo 1 Synch clock cont…

- Select “pin” with drop down list on second row “RB2” (Port B bit 2)
- Select “initial” drop down list as “High”
- Enter “5” in “Low cycles”, enter “15” in “High cycles”
- Select “Start” and “End” boxes to set the default
- Add labels and comments if desired
- Select “pin” with drop down list on Third row “RB3” (Port B bit 3)
- Select “initial” drop down list as “Low”
- Enter “5” in “Low cycles”, enter “5” in “High cycles”
- Select “Start” box, select the “Cycle” radio button in Start edit area
- Enter “20” in the first box and select “after last clock” in adjacent box
- Select “End” box, select the “Cycle” radio button in End edit area
- Enter “20” in the first box and select “from clock start” in adjacent box
- Add labels and comments if desired

Demo 1 Synch clock cont…

- Click "Apply" on bottom of Stimulus workbook
- Open new file Type "<Tab> nop <Enter>"
- Type " goto 0 <Enter>; end <Enter>
- Select "*File>Save as…*" menu item
- Save file as "testdemo.asm"
- Select "*Project>Quickbuild testdemo.asm*" menu item
- Select "*View>Simulator Logic Analyzer*" menu item
- Select "Channels" on the logic analyzer
- Add signals RB1,RB2 and RB3; select "OK"
- Select "Reset" then "Animate" on the debugger tool bar
- Watch signals generated on Logic Analyzer

University of Microchip UofM MASTERs 2007

Demo 2 Asynch Trigger for synch clock

- **Open MPLAB® IDE**
  - Select “*Configure>Select Device*” menu item
  - Select a “pic18C442” device
  - Select “*Debugger>Select Tool>MPLAB SIM*”

- **Open Stimulus window**
  - Select “*Debugger>Settings*”
  - Select Trace All check box on “Osc\Trace” tab Select OK
  - Select “*Debugger>Stimulus>New Workbook*”
  - Select “Clock Stimulus” tab
  - Select “pin” with drop down list on first row “RB1” (Port B bit 1)
  - Select “initial” drop down list as “High”
  - Enter “3” in “Low cycles”, enter “1” in “High cycles”
  - Select “Start” box, select the “Pin” radio button in Start edit area
  - Select “RB1” in the pin drop down list and “High” in adjacent box
  - Select “End” box, select the “Cycle” radio button in End edit area
  - Enter “7” in the first box and select “from clock start” in adjacent box

Demo 2 Asynch Trigger for synch clock cont…

- Select the “Asynch” tab
- Select “Pin/SFR” drop down list and select “RB1”
- Select “Set High” for the “Action”
- Click “Apply” on bottom of Stimulus workbook
- Open new file Type “<Tab> nop <Enter>”
- Type “ goto 0 <Enter>; end <Enter>
- Select “*File>Save as…*” menu item
- Save file as “testdemo.asm”
- Select “*Project>Quickbuild testdemo.asm*” menu item
- Select “*View>Simulator Logic Analyzer*” menu item
- Select “Channels” on the logic analyzer
- Add signals RB1; select “OK”
- Select “Reset” then “Animate” on the debugger tool bar
- Fire the Asych stimulus and watch signal generated on Logic Analyzer
- Ensure RB1 is low when you start or else nothing will change

Demo 3 Simultaneous Interrupts

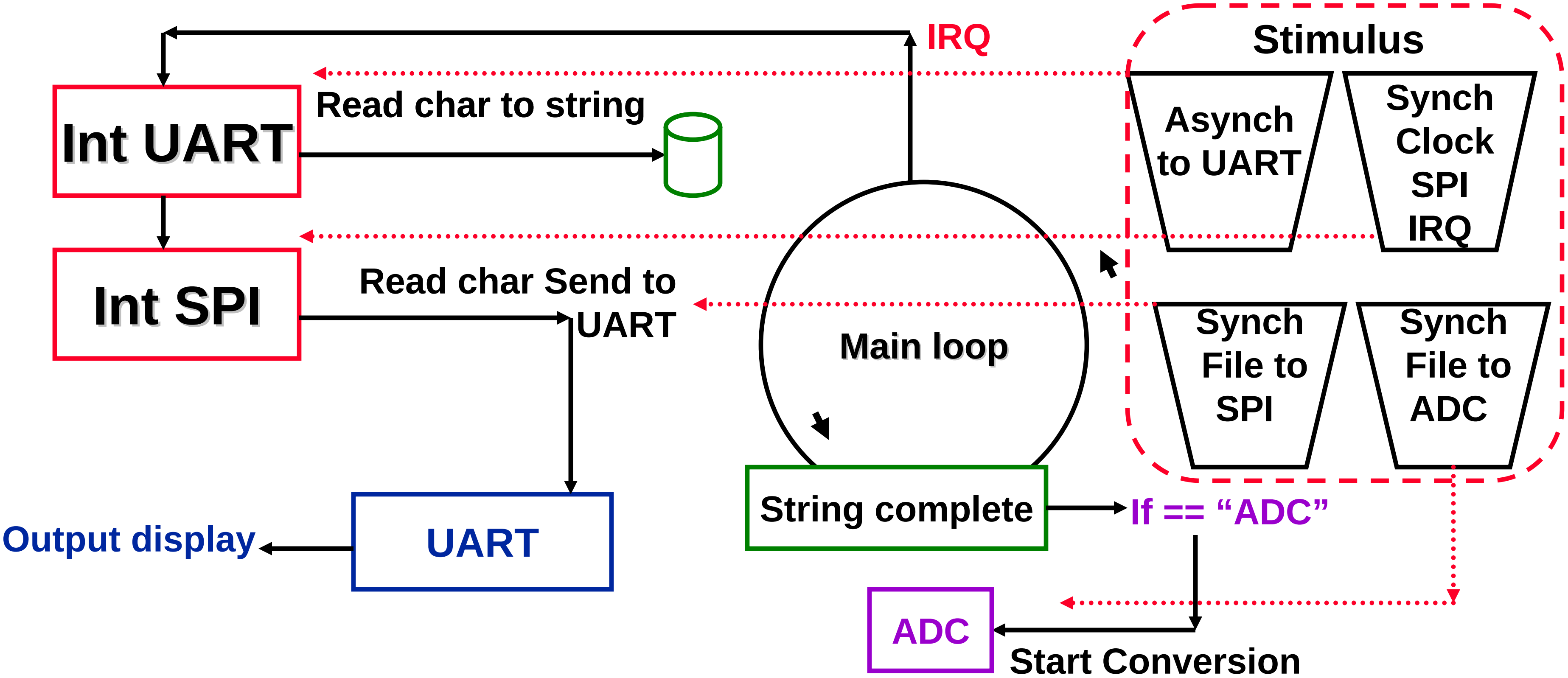
- **Open MPLAB® IDE**
  - Select “*Configure>Select Device*” menu item
  - Select a “pic18C442” device
  - Select “*Debugger>Select Tool>MPLAB SIM*”
- **Open Stimulus window**
  - Select “*Debugger>Stimulus>New Workbook*”
  - Select the “Asynch” tab
  - Select “Pin/SFR” drop down list and select “RB0”
  - Select “Pulse High” for the “Action”
  - Enter “1” for pulse width in “cycles”
  - Select the “Advanced Pin/Register” tab
  - Create a condition in the lower “Define Conditions” edit area
  - Click “Pin” in the first drop down box of the first row
  - Select “RB0” as the pin, leave the “=“ test and change the value to “1”

Demo 3 Simultaneous Interrupts cont…

- Define a Trigger in the “Define Triggers” edit area above
- Select “Cond1” from the drop down list in the first box of first row
- Select “Cont” for the “Type”
- Enter “200” for the Re-Arm delay in cycles
- Click the title bar to add signals which one wants to change
- Select bit fields “INTCON.INT0IF” and “PIR1.TMR1IF”
- Select “Add” and “OK” to add them as columns in trigger area
- Enter a “1” under each column “INTCON.INT0IF” and “PIR1.TMR1IF” in the first row
- Click “Apply” on bottom of Stimulus workbook
- Open a watch window “*View>Watch*”
- Select SFRs “INTCON” and “PIR1” adding them to the watch window
- Reset processor, step a few times
- Fire the Asynch RB0 pulse high action
- Step once, view the watch window and see both IF flags are set in the same step

Demo 4 Peripheral Injection and Trace

- Open MPLAB® IDE
  - Select “*File>Open Workspace…*” menu item
  - Select “C:\MASTERS\11016\Demo4\RegisterInjectTrace.mcw”
  - Code is written to read SPI, UART and ADC; write to UART



Demo 4 Peripheral Injection and Trace

- **Open Stimulus window**
  - Select “*Debugger>Stimulus>New Workbook*”
  - Select “Clock Stimulus” tab
  - Select pin with drop down list on first row “RC3” (Port C bit 3)SPI SCK
  - Select initial drop down list as “Low”
  - Enter “800” in Low cycles, enter “2” in High cycles
  - Select “Start” and “End” boxes to set the default
  - Add labels and comments if desired
  - Select “Advanced Pin/Register” tab
  - Create COND with “Pin” “RC3” “=“ “1”
  - Create Trigger using COND type “Cont” re-arm “700” “cyc”
  - Select title to add signal “PIR1.SSPIF”
  - Enter “1” below signal
  - The SSPIF flag will now be set approximately every 800 cycles
  - Select “Register Injection” tab
  - Select register “SSPBUF” Trigger set to “Demand”

Demo 4 Peripheral Injection and Trace

- Select Data Filename and browse for SPIinput.txt
- Select wrap “Yes” and set format to “Raw”
- When the SPI receives an interrupt set by clock, it will branch in code and read a value injected from the file, then send it out on the UART
- On the next row select register “ADRESL” Trigger set to “Demand”
- Select Data Filename and browse for ADCinput.txt
- Select wrap “Yes” and set format to “Hex”
- Select “Apply” at bottom of window to apply Synchronous stimulus
- Select “Asynch” tab
- Create 3 entires, All Pin/SFR “RCREG1” and action “Direct Message”
- Set first message ‘ “ADC” ‘
- Set second message ‘41 43 44 D’
- Set third message ‘A’
- Select “*Debugger>Settings*”
- Select “UART 1 IO” tab
- Enable the UART 1 IO check box
- In the “Output” area select the “window” radio button
- Select OK

Demo 4 Peripheral Injection and Trace

- This will create a UART 1 IO tab on the output window
- Build and execute the code
- The output window will display the message “Input from SPI shows up on UART IO display” This is the SPI read echoed onto the UART IO
- Fire the “ADC” Asynch message
- Fire the A to end the string (0xA)
- This will start and ADC conversion
- ADC will read data from file on completion and load it into the Result register
- Summary…
- The SPI is reading raw data from the one file and sending it to the UART
- The UART IO is enabled through the simulator settings window to output data to the display. This shows the data injected into the SPI whenever the IRQ flag is set on the SPI
- When firing the correct string “ADC” asynchronously into the UART receive and ending the string with an 0xA or 0xD the main loop starts an ADC conversion
- The ADC reads the next value from the ADC data file and after the correct conversion time passes, it places the value into the Result register and clears the Done bit

Demo 5 Algorithm Verification

- **Open MPLAB® IDE**
  - Select “*File>Open Workspace…*” menu item
  - Select “C:\MASTERS\11016\Demo5\Algtest.mcw”
  - Code is written to perform a RRNC
- **Open Excel Algorithm**
  - Open “C:\MASTERS\11016\Demo5\AlgTestData.xls”
  - Select rows 4 to 41 in column ‘A’ only and copy.
  - Open a new file within MPLAB IDE “*File>New*” and paste the data into it
  - Select “File>*Save As…*” and save the file under Demo 5 as DataInput.txt

Demo 5 Algorithm Verification

- **Open Stimulus window**
  - Select “*Debugger>Stimulus>New Workbook*”
  - Select “Register Injection” tab
  - Select “Register/Var” in first row, and select BSR to force a PC= trigger selection
  - Within the Register box enter the value “0x10”
  - Enter “0x100” in the “PC Value” box
  - Ensure Data width is set to one
  - Select the browse button on the Data filename box, and select the DataInput.txt file created earlier
  - Select “Yes” under the wrap
  - Select “Dec” for the file format
  - Select the “Register Trace” tab
  - Select “Register/Var” in first row, and select BSR to force a PC= trigger selection

Demo 5 Algorithm Verification

- **Open Stimulus window cont...**
  - Within the Register box enter the value “0x20”
  - Enter “0x108” in the “PC Value” box
  - Ensure Data width is set to one
  - Select the browse button on the Data filename box, and enter DataOutput.txt
  - Select “Dec” for the file format
  - Select the “Apply” button to activate the stimulus
  - Reset the processor
  - Execute “Run” for a short time 1 second
  - Execute Halt
  - Select “Remove” on the stimulus window to allow closing of the output file
  - Open the DataOutput.txt file, select the first 38 entries and copy them
  - Open the Excel spread sheet and paste the values into the column next to the results so a comparison can be made for each line

Demo 6 Complex Breakpoint

- **Open MPLAB® IDE**
  - Select "*File>Open Workspace…*" menu item
  - Select "C:\MASTERS\11016\Demo6\Breakpoints.mcw"
  - Code is written to perform Tblwth to location 0x1000 in program memory
- **Open Complex Breakpoints**
  - Open *"Debugger>Complex Breakpoints*"
  - Select "Add Breakpoint" from dialog
  - Enter in "0x1000" in the Address Symbol/Hex edit box
  - Select breakpoint type "TBLWT Program Memory"
  - Select "Always Break"
  - Execute code, it will break after every TBLWT

Demo 6 Complex Breakpoint

- Right click on breakpoint in Complex breakpoint dialog box, select Edit
- Change “Always Break” to “Event must occur count times” enter 2 in count box
- Reset and Run. Program will break on halt instruction then after second TBLWT
- Right click on breakpoint in Complex breakpoint dialog box, select Edit
- Change “Event must occur count times” to “Break occurs count instructions after event”
- Enter 5 in count box
- Reset and Run. Program will break 5 instructions after the first TBLWT occurs

Demo 7 Trace

- **Open MPLAB® IDE**
  - Select "*File>Open Workspace…*" menu item
  - Select "C:\MASTERS\11016\Demo7\Tracetest.mcw"
- **Simple Trace Trigger**
  - Set a Breakpoint in the Interrupt handler at line 26
  - Open the Logic Analyzer window "*View>Simulator Logic Analyzer*"
  - Build the project, and execute
  - Fire the Asynch button for RB0 stimulus
  - The program should halt at the break point
  - Select the "Now" button under the Trigger PC = label on the Logic Analyzer
  - A PC = value of 0x00080 should be entered into the read only box
  - Remove the Break point, reset the processor and execute
  - Halt the processor open the Trace window "*View>Simulator Trace*"
  - You will notice it states "No items to display"
  - Execute again, Fire the Asynch RB0 wait 2 seconds and Halt the processor. The buffer stops collecting when full
  - View the data and start point of the trace buffer
  - Try different formats of the buffer using start, center and end

Demo 7 Trace

- **Simple Trace Trigger *Or*** (not on PIC18xxx prior to 7.62)
  - Open the Logic Analyzer window "*View>Simulator Logic Analyzer*"
  - Click in editor window where trigger needs to be set
  - Right click to get context menu
  - Select "Set PC at Cursor"
  - Select the "Now" button under the Trigger PC = label on the Logic Analyzer
  - A PC = value of 0x00080 should be entered into the read only box
  - Reset the processor and execute
  - Fire the Asynch RB0 wait 2 seconds and Halt the processor. The buffer stops collecting when full
  - View the data and start point of the trace buffer
  - Try different formats of the buffer using start, center and end

Demo 7 Trace

- **Filter Trace**
  - Select Clear in the Logic Analyzer window under the Trigger PC= to remove the trigger
  - Select the source file, highlight the Interrupt routine and right click
  - Select the “Add Filter-in Trace”
  - Reset the processor and execute. Fire the Asynch stimulus RB0 button a few times
  - View the Trace output
  - Right click in the editor select “Remove all Filter Traces”
  - Highlight the “While loop”, right click
  - Select the “Add Filter-out Trace”
  - Reset and execute. Fire the Asynch stimulus RB0
  - View the Trace output

Demo 8 Logging data without UART

- **Open MPLAB® IDE**
  - Select “*File>Open Workspace…*” menu item
  - Select “C:\MASTERS\11016\Demo8\LoggingInC.mcw”
  - Notice within C file a function “void LogError (void)” (Any Function name can be used, C symbol)
  - Build the project

- **Open Stimulus window**
  - Select “*Debugger>Stimulus>New Workbook*”
  - Select the “Register Trace” tab
  - Select “Register/Var” in first row, and select BSR to force a PC= trigger selection
  - Within the Register box enter the value “0x100” the address of the “readbyte_int” string
  - Select from the drop down list for the PC value the “LogError” function name (C Symbol)
  - Change the data width to 100

Demo 8 Logging data without UART

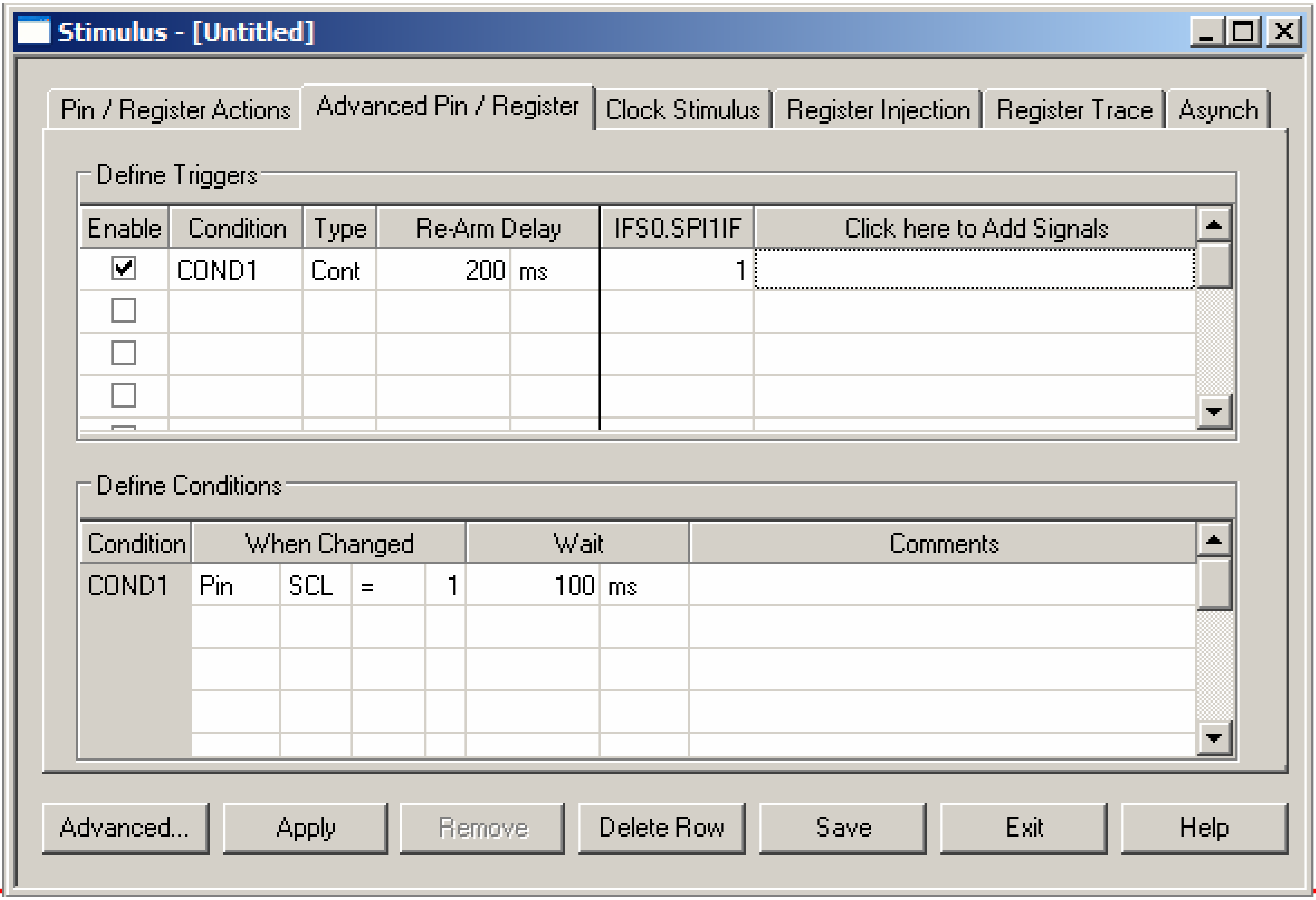
- **Open Stimulus window cont...**
  - Select the browse button on the Data filename box, and create or select the file ErrorLog.txt
  - Select “Raw” for the file format
  - Select the “Apply” button to activate the stimulus
  - Reset the processor
  - Execute “Run”
  - Wait 2 seconds Select “Halt”
  - Select the “Remove” button to allow the log file to close
  - Open the ErrorLog file from the project window and note the errors

SPI

- **SPI requires multiple stimulus**
  - Clock to trigger the Interrupt flag
  - Data Injection into the buffer
- **Clock can be driven by Firmware or stimulus**
  - Create a clock based on time or firmware and use conditional stimulus to monitor this, using it to set the SPIIF flag
- **Data injected when firmware reads the SPI buffer**
  - Attaching a register stimulus file to SPIBUF on demand will inject on each read
  - Raw text or hex if you need control codes

SPI

- **Condition based on Clock out triggers IF bit**



Define Triggers

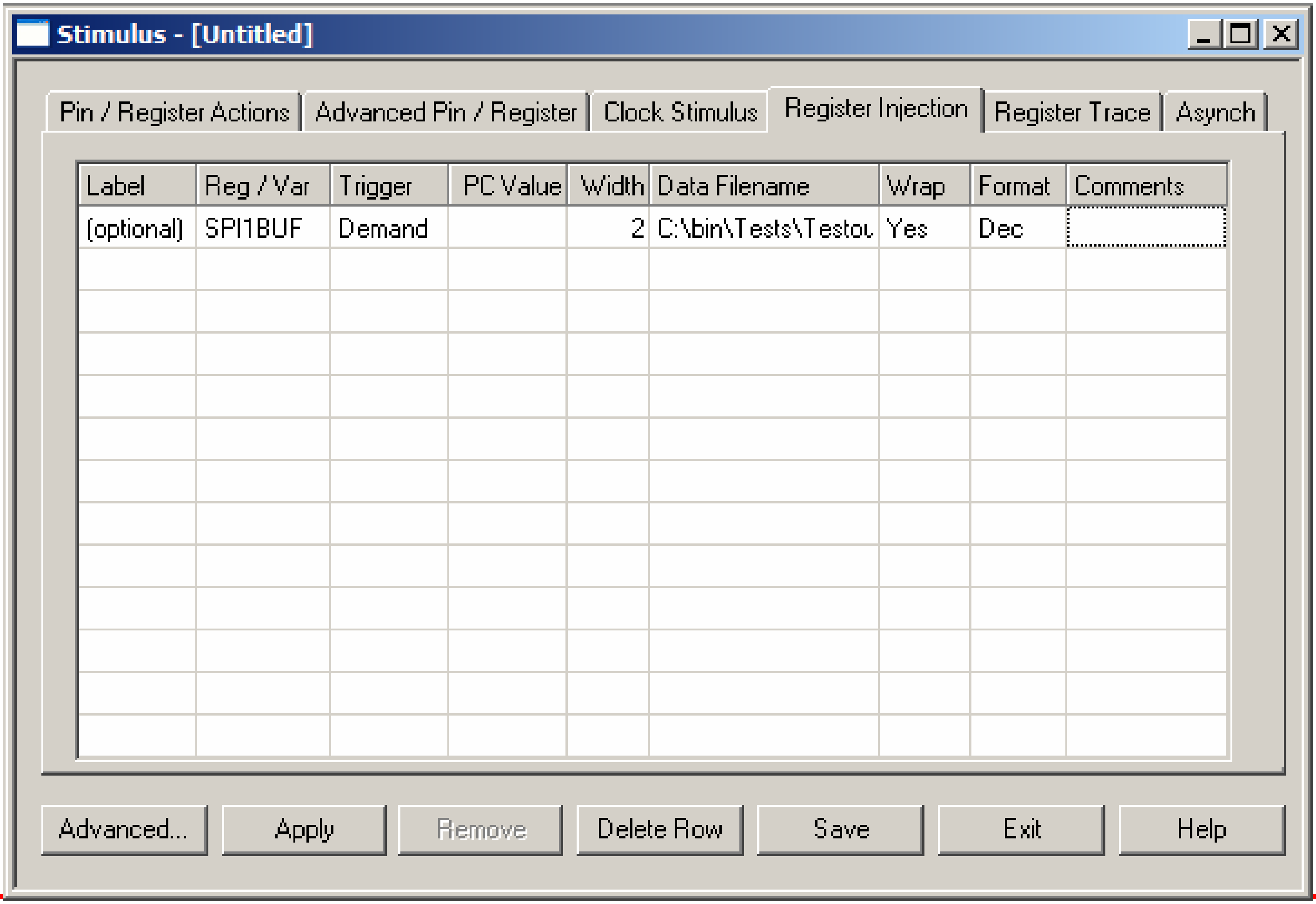
Enable	Condition	Type	Re-Arm Delay		IFS0.SPI1IF	Click here to Add Signals
☑	COND1	Cont	200	ms	1	
□						
□						
□						

Define Conditions

Condition	When Changed				Wait		Comments
COND1	Pin	SCL	=	1	100	ms	

SPI

- **Data file attached to SFR SPI1Buf feeds data into SFR when read by firmware**



Label	Reg / Var	Trigger	PC Value	Width	Data Filename	Wrap	Format	Comments
(optional)	SPI1BUF	Demand		2	C:\bin\Tests\Testou	Yes	Dec	

Trademarks

The Microchip name and logo, the Microchip logo, Accuron, dsPIC, KeeLoq, KeeLoq logo, microID, MPLAB, PIC, PICmicro, PICSTART, PRO MATE, rfPIC and SmartShunt are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AmpLab, FilterLab, Linear Active Thermistor, Migratable Memory, MXDEV, MXLAB, SEEVAL, SmartSensor and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, ECAN, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, PICkit, PICDEM, PICDEM.net, PICLAB, PICtail, PowerCal, PowerInfo, PowerMate, PowerTool, REAL ICE, rfLAB, Select Mode, Smart Serial, SmartTel, Total Endurance, UNI/O, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.