11015 MS2

MPLAB® Simulators
Advanced Stimulus

Class Objective

- **When you finish this class you will be able to:**
  - Create complex, parallel clock signals using Stimulus
  - Create and stimulate multiple A/D waveform inputs
  - Monitor and control parameters in firmware using DMCI ★
  - Know how to use Stimulus to log data to verify program functionality

Expectations

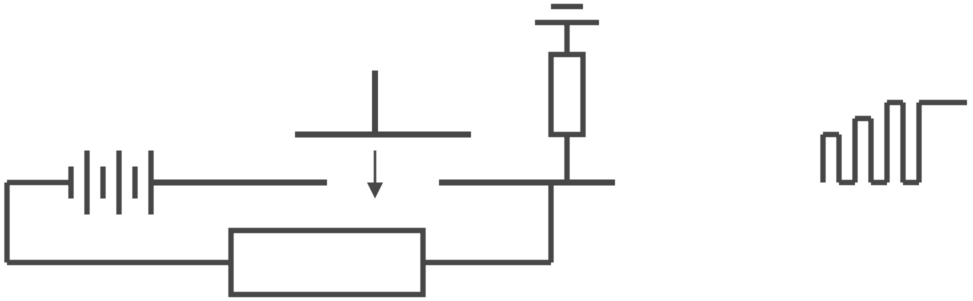
- **I will not cover fundamental operations of**
  - Simulator
  - Stimulus dialog
- **Details will be in lab handouts**
- **Teaching methods**
  - Run through functionality “How to”
  - Lab

Agenda

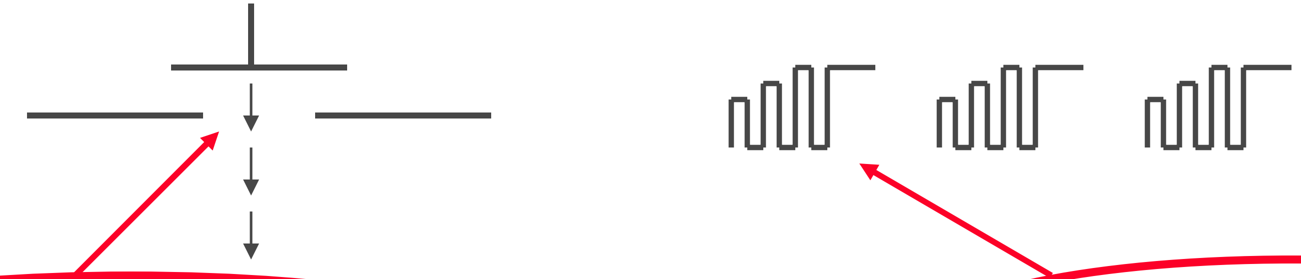
- **Use Stimulus for complex signals**
  - Repeating, periodic clock (Keybounce) Lab 1
  - Conditional stimulus injection (Encoder) Lab 2
- **Create and use multiple A/D channel signal injection** Lab 3a
- **Monitor signal injection and control a firmware parameter with DMCI** Lab 3b
- **Find out more**

Key Bounce

- Short clock burst

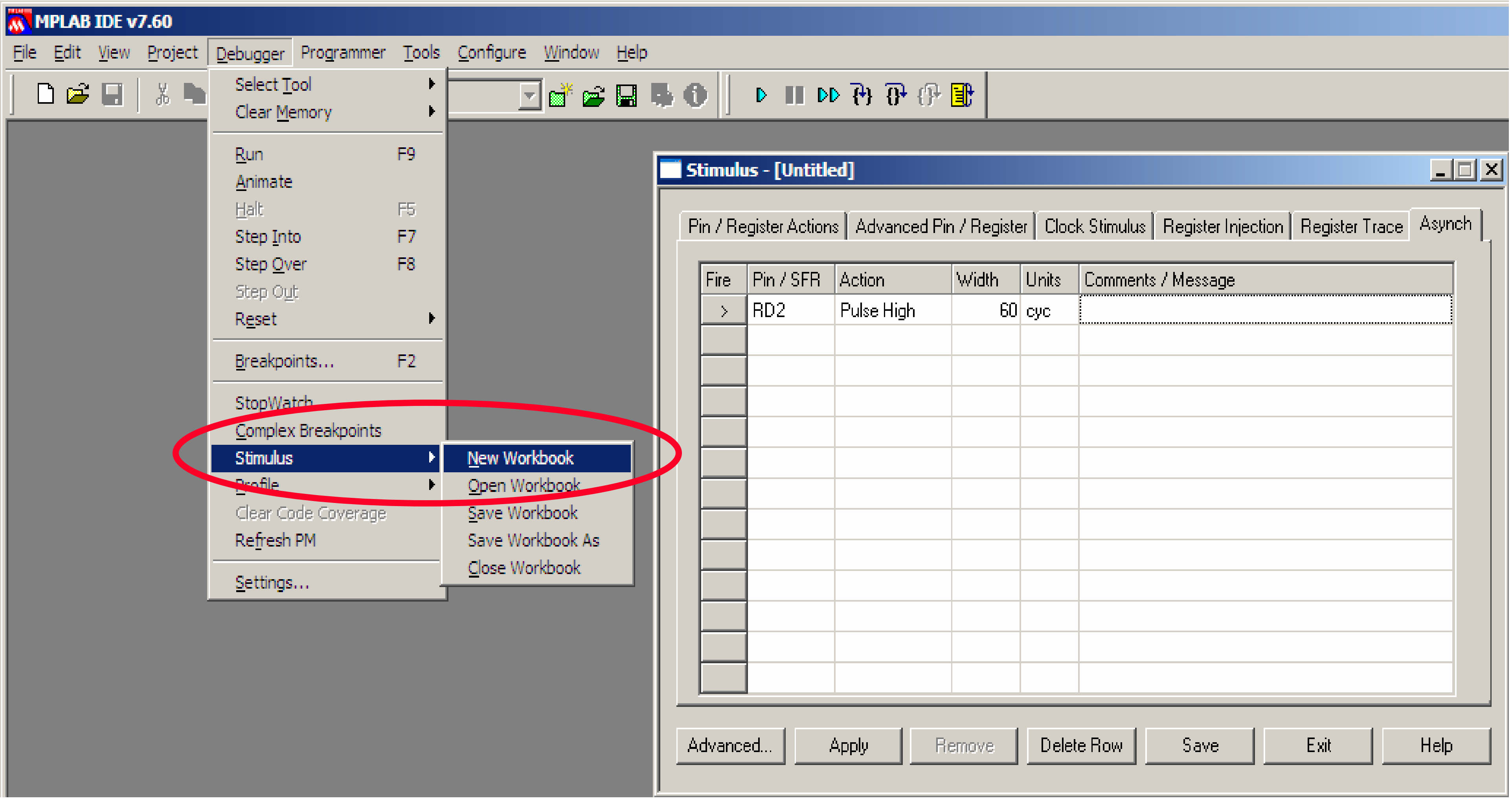


- Repeat when desired



Asynchronous event to trigger a Synchronous clock

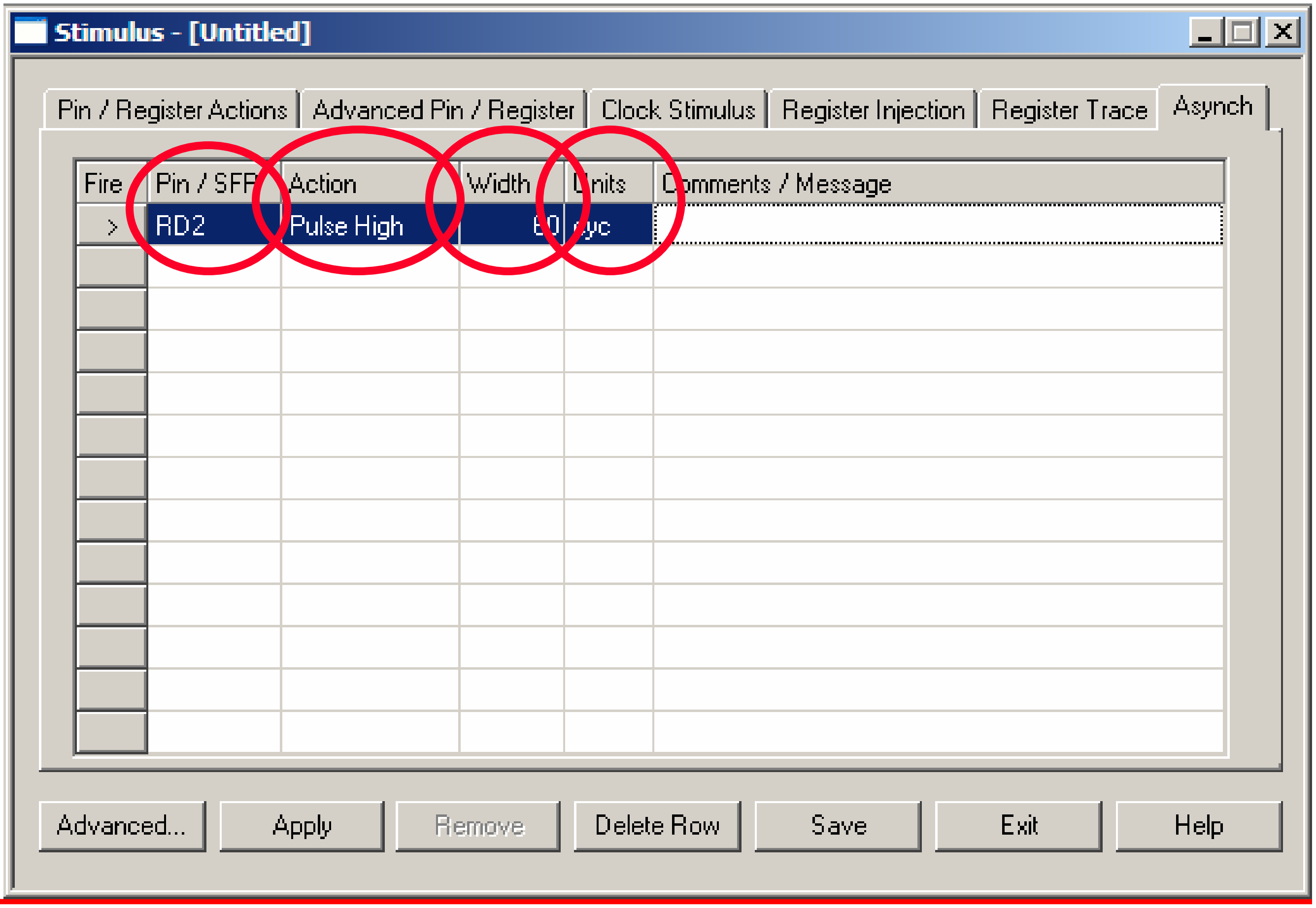
Stimulus Select



Fire	Pin / SFR	Action	Width	Units	Comments / Message
>	RD2	Pulse High	60	cyc	

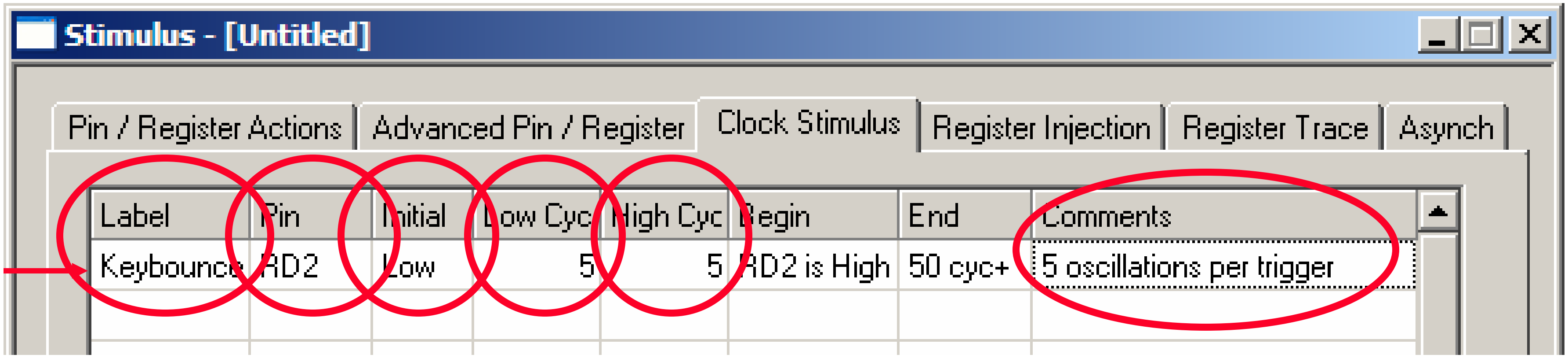
Define Asynchronous Button

- **Select Pin**
  - Rx?
  - TxCLK
- **Select Action**
  - Pulse
  - High
  - Low
  - Toggle
- **Select Width**
  - Dec value
- **Units**
  - cyc, ns
    us, ms
    sec



Fire	Pin / SFR	Action	Width	Units	Comments / Message
>	RD2	Pulse High	60	cyc	

Synchronous Clock

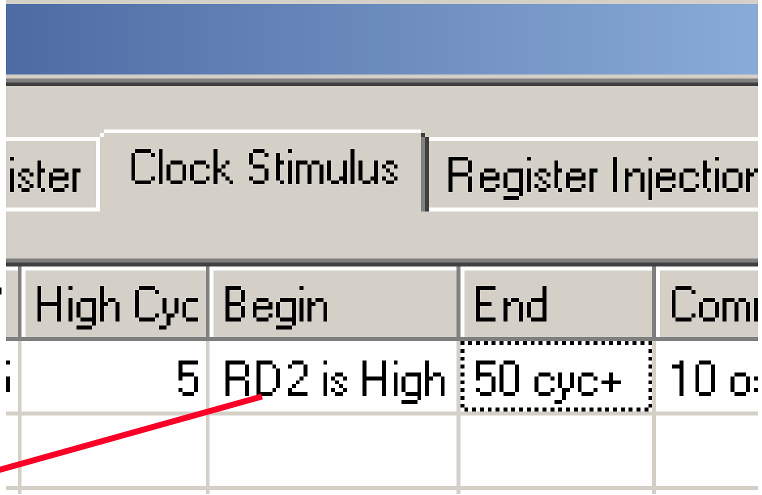


Label	Pin	Initial	Low Cyc	High Cyc	Begin	End	Comments
Keybounce	RD2	Low	5	5	RD2 is High	50 cyc+	5 oscillations per trigger

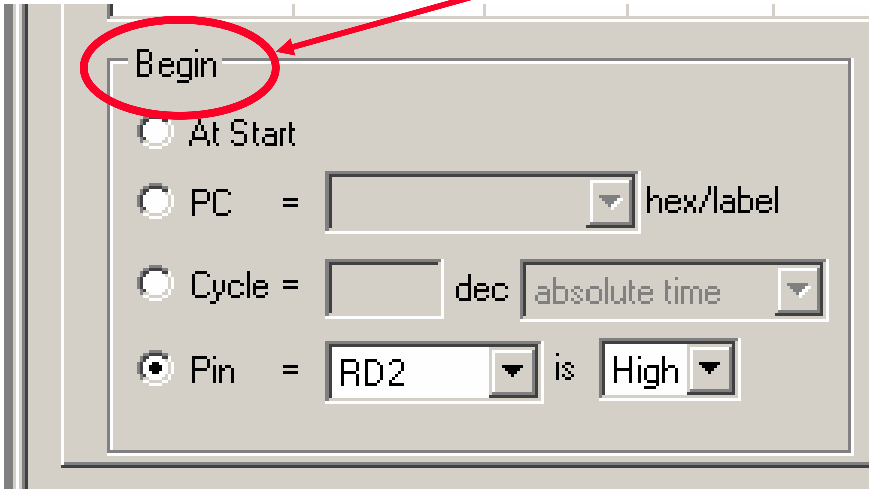
- Each row provides a separate clock
- Optional label
- Select pin to inject the clock into
- Initial start state
- Low time
- High time
- Optional comments

Synchronous Clock

- Begin column edit area determines when the clock will start
- “At Start” starts at initial execution. This is the default
- “PC” starts at PC value or a specific label



High Cyc	Begin	End	Comi
5	RD2 is High	50 cyc+	10 o:



Begin

○ At Start

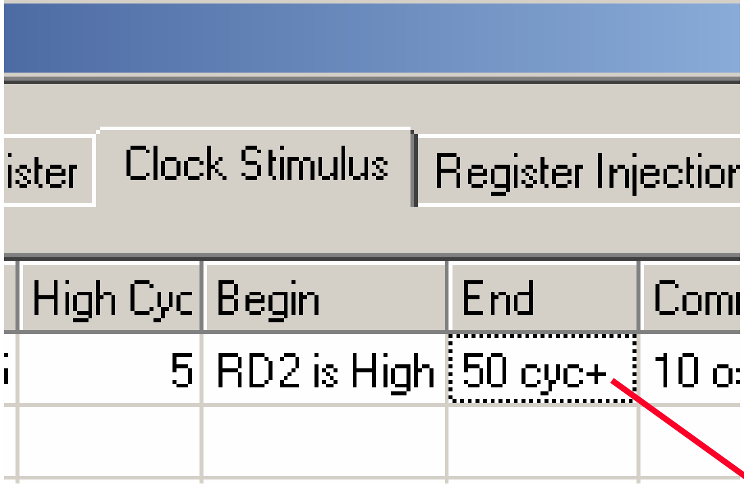
○ PC = hex/label

○ Cycle = dec absolute time

● Pin = RD2 is High

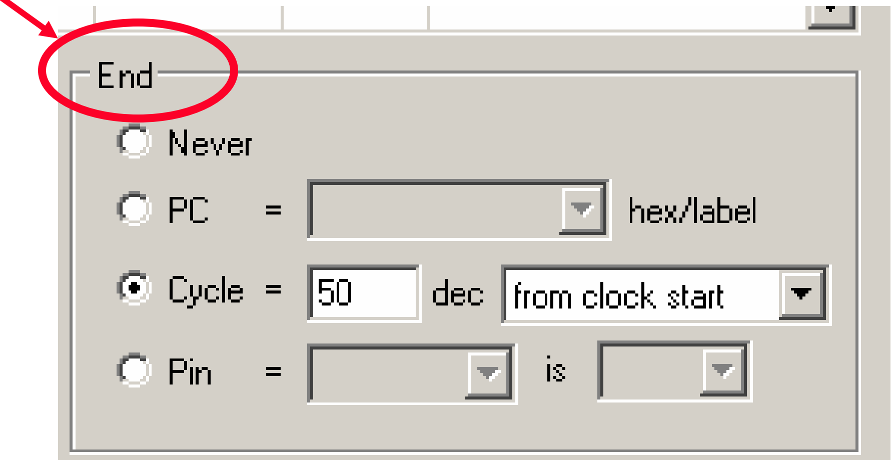
- “Cycle” starts at an absolute cycle count from initial program execution. Or starts a relative cycle count after the last clock
- “Pin” starts when the selected pin changes to the selected state. RD2 :- Port D bit 2

Synchronous Clock



High Cyc	Begin	End	Comi
5	RD2 is High	50 cyc+	10 c:

- End column edit area determines when the clock will stop
- “Never” the clock will never stop. This is the default.
- “PC” stops at the PC value or a specific label
- “Cycle” stops at an absolute cycle count from initial program execution. Or stops a relative cycle count after the start
- “Pin” stops when the selected pin changes to the selected state



End

○ Never

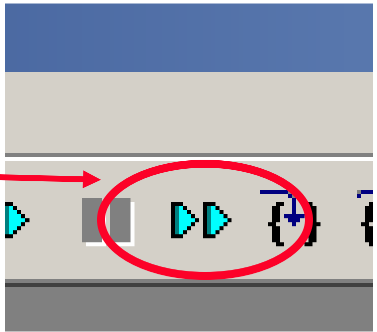
○ PC = hex/label

● Cycle = 50 dec from clock start

○ Pin = is

Testing…

- **Do you need code?** ***Sometimes.***
  - For animation create simple .asm file
  - First line nop, Second line goto 0
  - Third line end directive
- **Use Quick Build (*Project>Quickbuild*)**
- **Ensure trace enabled**
- **Run in animation**
- **Fire the button**

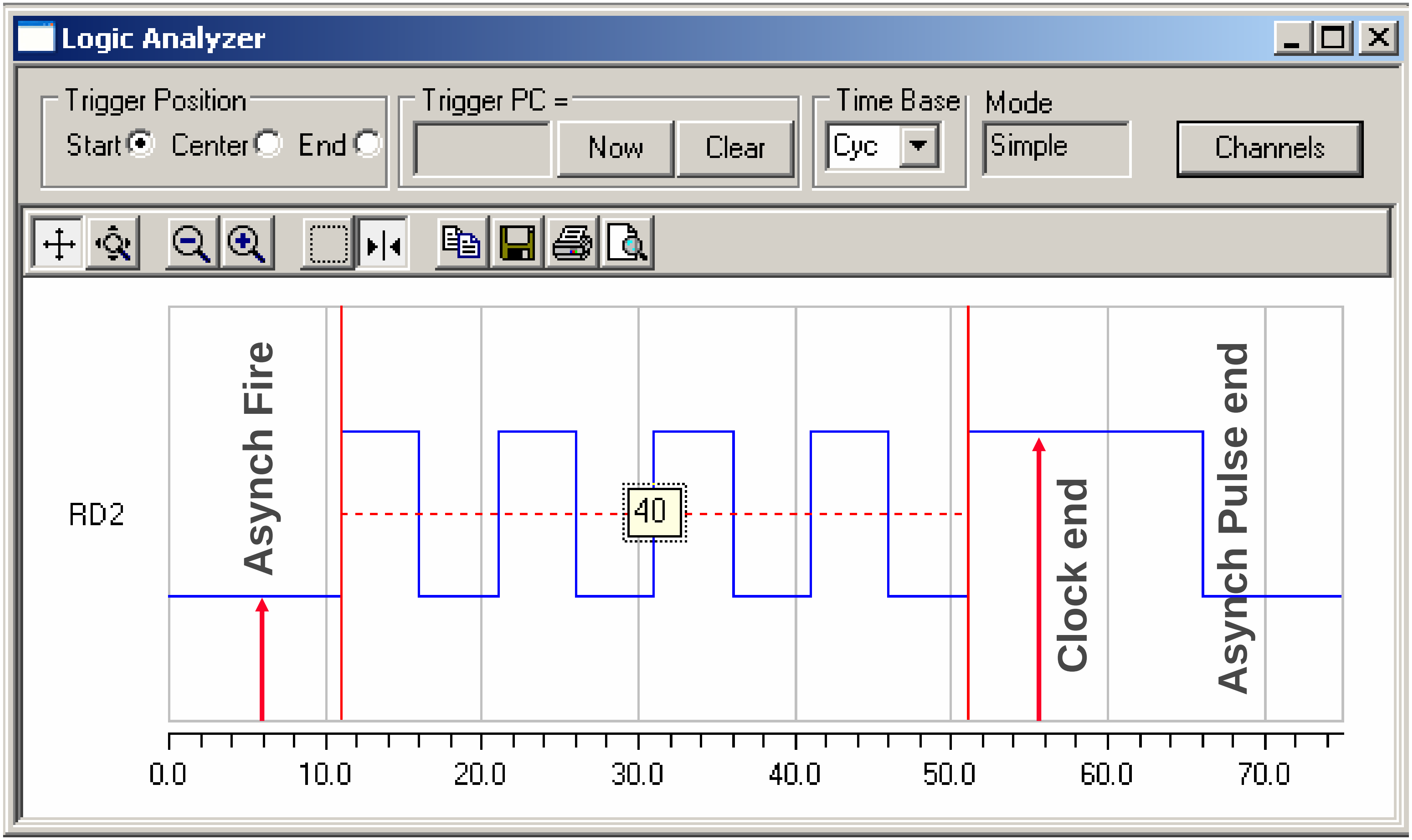


- **Halt animation and view the logic analyzer. Select RD2**

Let’s do it

- **Open Stimulus (Asynch tab)**
  - Define a high asynch pulse on RD2
- **Select Clock tab**
  - Define clock burst
- **Apply** (click button)
- **Test it**

Results

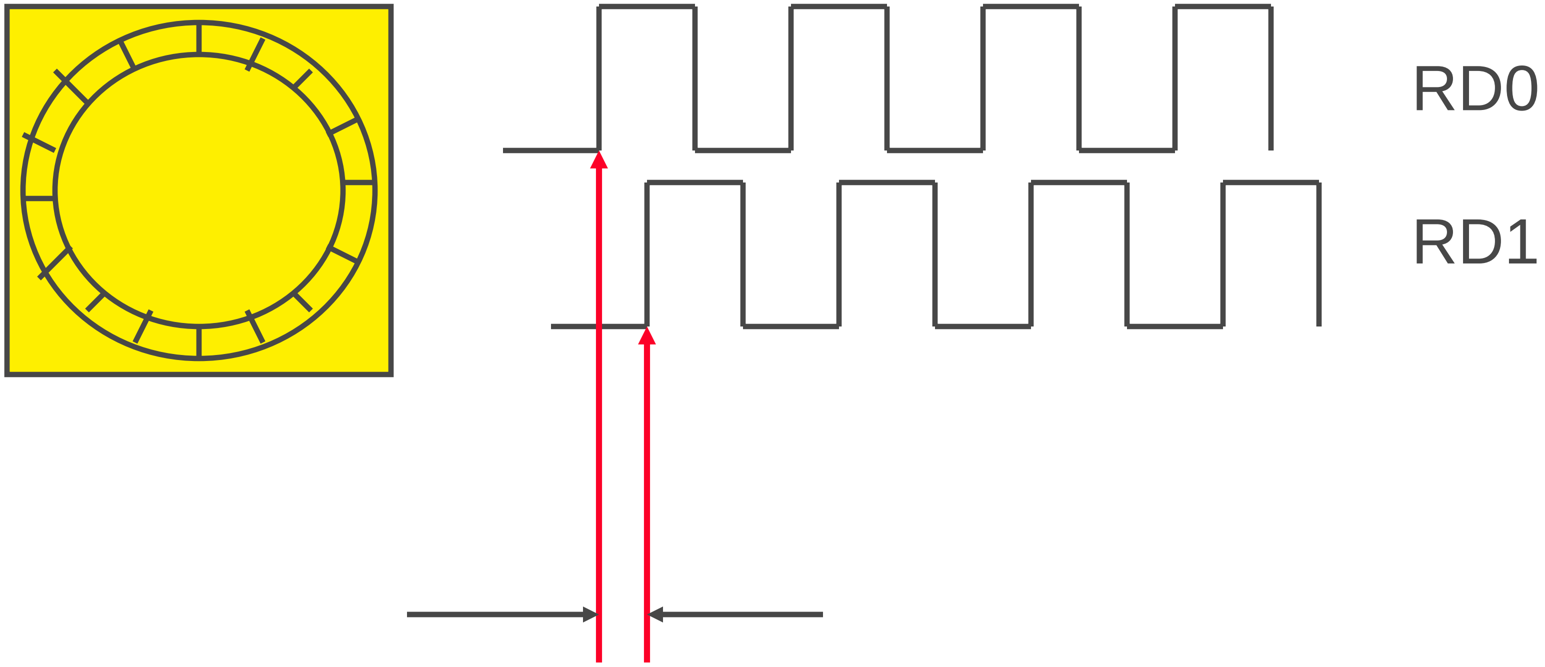


Logic Analyzer
Trigger Position
Start ● Center ○ End ○
Trigger PC =
Now
Clear
Time Base
Cyc
Mode
Simple
Channels
RD2
Asynch Fire
40
Clock end
Asynch Pulse end
0.0
10.0
20.0
30.0
40.0
50.0
60.0
70.0

Encoder

Encoder

- Two pulse trains



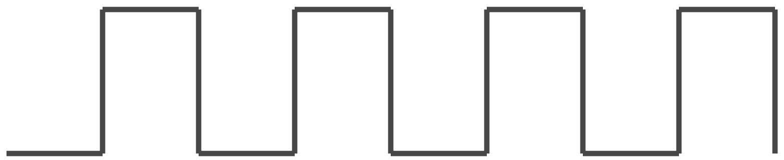
RD0

RD1

90 degree phase shift

Encoder

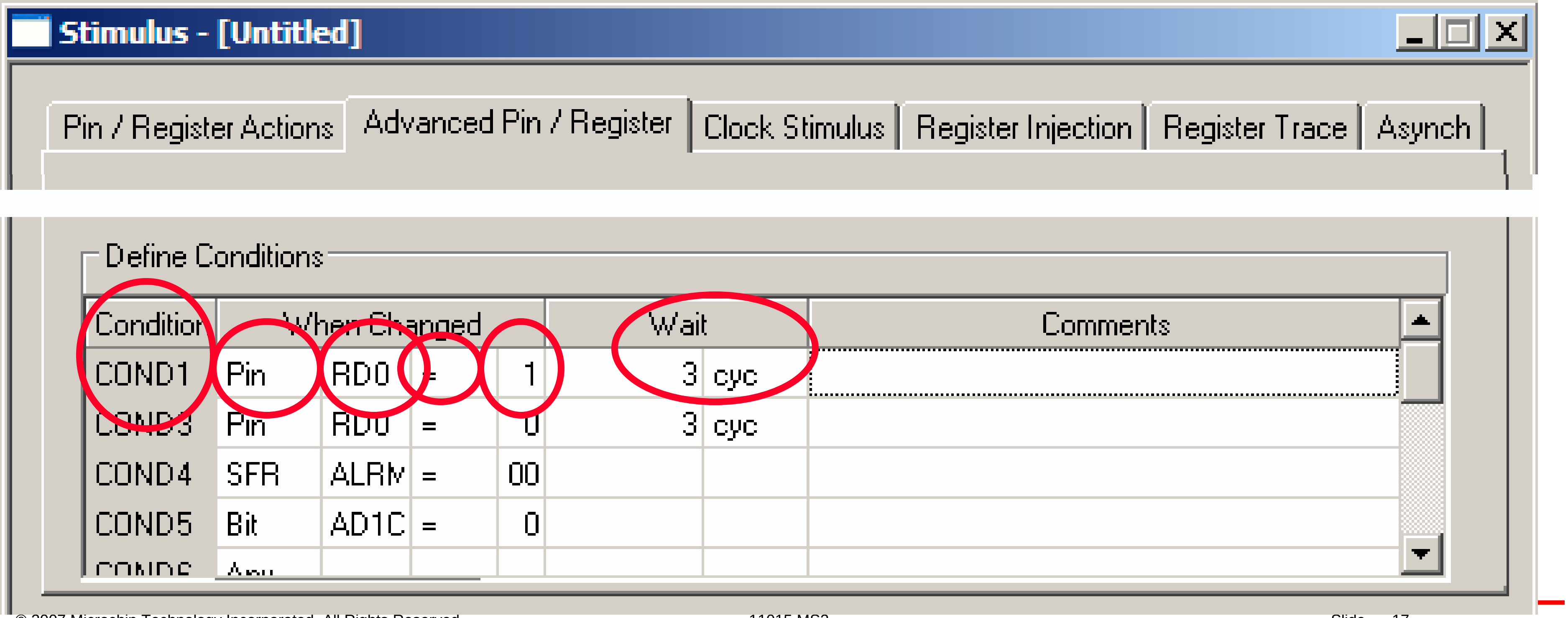
- **First pulse train**
  - Identical method to key bounce
  - Use “Clock” stimulus (init high 6L 6H)
  - Trigger on pin state
  - Run for predetermined time



RD0

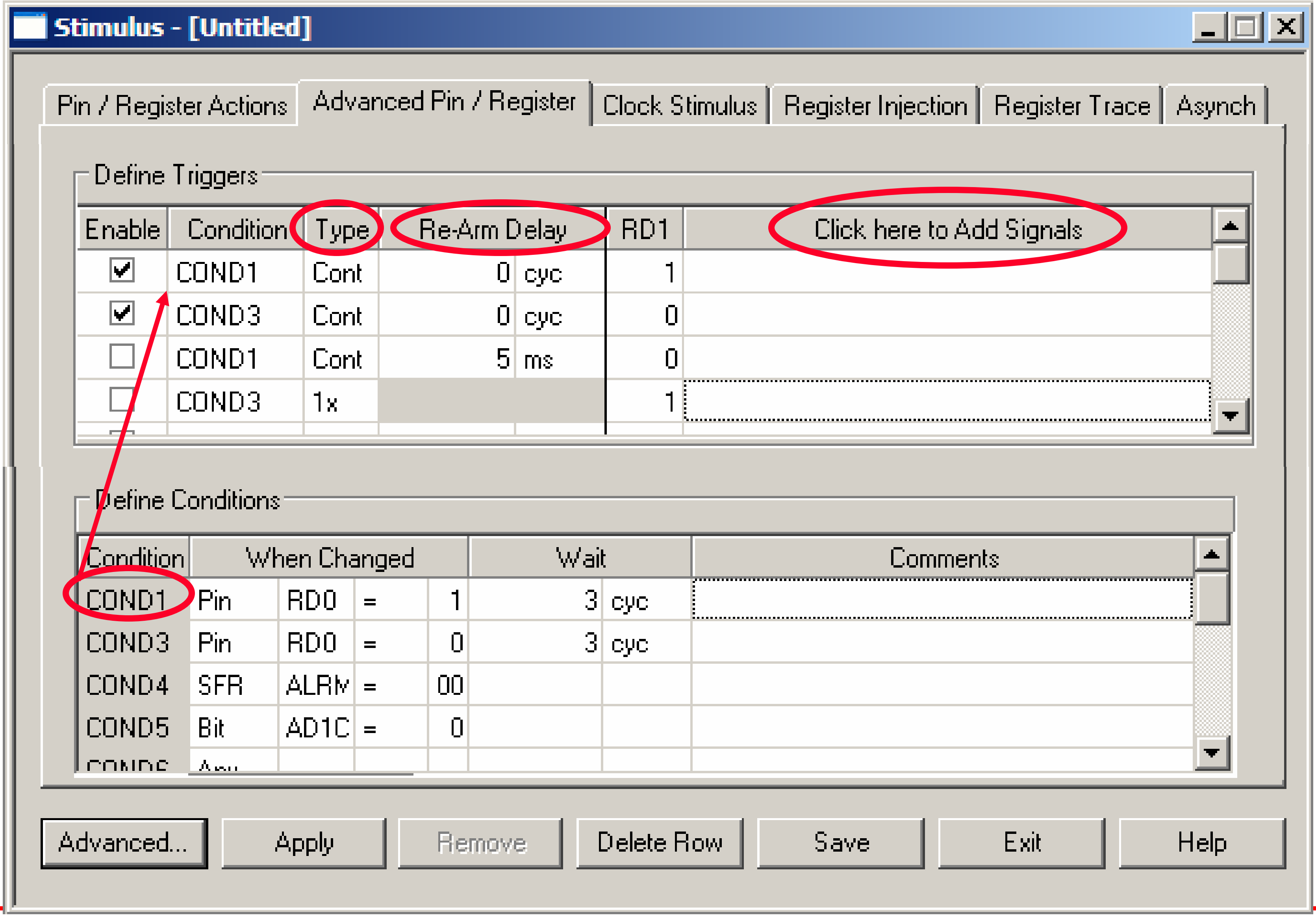
Encoder

- **Second pulse train**
  - Use “Advanced Pin / Register” tab
  - Generate a pulse train based on the first pulse train, using conditional statements



Conditior	When Changed				Wait		Comments
COND1	Pin	RD0	=	1	3	cyc	
COND3	Pin	RD0	=	0	3	cyc	
COND4	SFR	ALRM	=	00			
COND5	Bit	AD1C	=	0			
COND6	Any						

Advanced Pin/Register



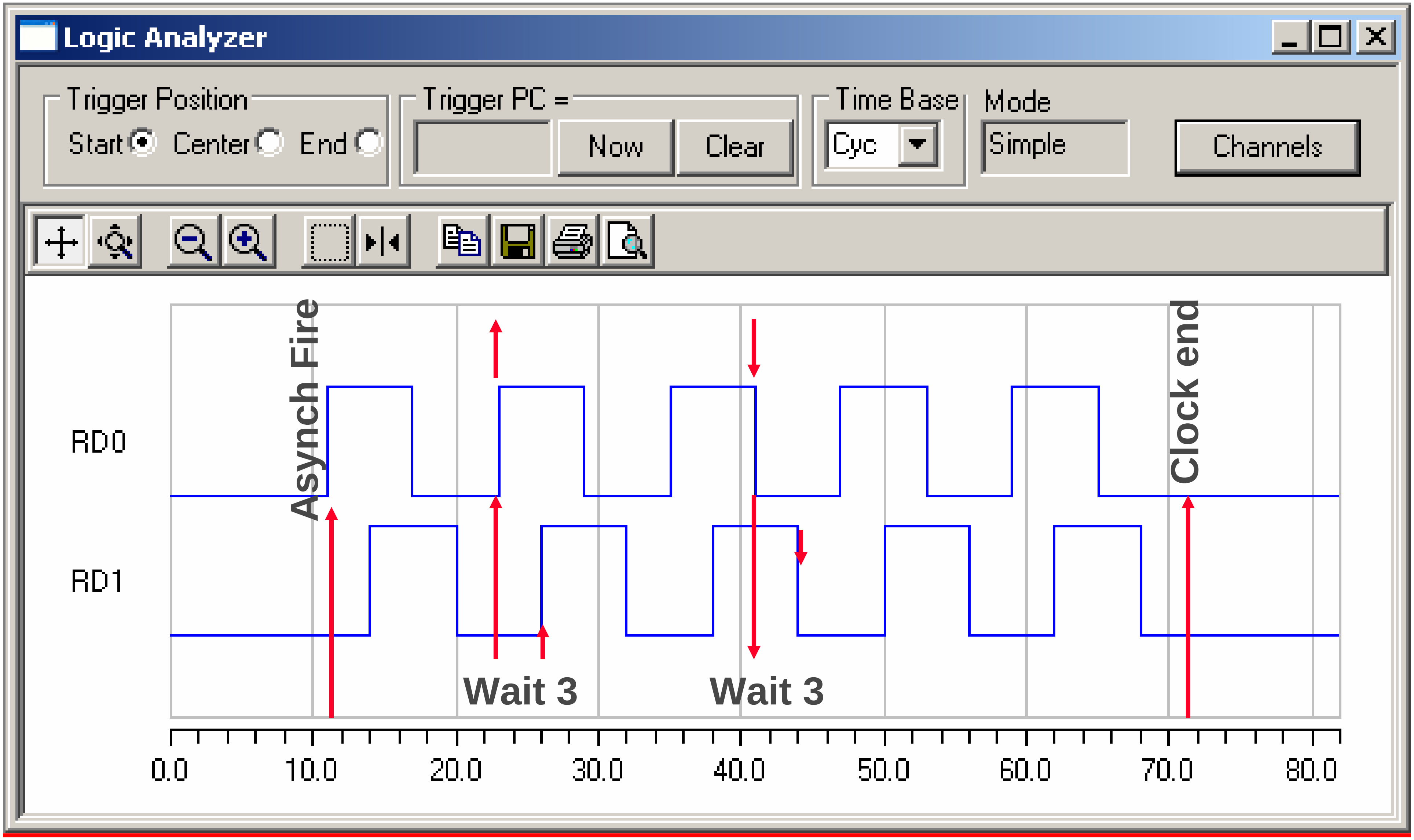
Define Triggers

Enable	Condition	Type	Re-Arm Delay		RD1	Click here to Add Signals
☑	COND1	Cont	0	cyc	1	
☑	COND3	Cont	0	cyc	0	
□	COND1	Cont	5	ms	0	
□	COND3	1x			1	

Define Conditions

Condition	When Changed				Wait		Comments
COND1	Pin	RD0	=	1	3	cyc	
COND3	Pin	RD0	=	0	3	cyc	
COND4	SFR	ALRM	=	00			
COND5	Bit	AD1C	=	0			
COND6	Any						

Results


Logic Analyzer
Trigger Position
Start ● Center ○ End ○
Trigger PC =
Now
Clear
Time Base
Cyc
Mode
Simple
Channels
RD0
RD1
Asynch Fire
Clock end
Wait 3
Wait 3
0.0
10.0
20.0
30.0
40.0
50.0
60.0
70.0
80.0

Let’s do it

- **Open Stimulus (Asynch tab)**
  - Define a set high asynch event on RD0
- **Select Clock tab**
  - Define clock burst with 12 cycle period on RD0
- **Select Advanced Pin/Register tab**
  - Define 2nd clock on RD1 based on first
- **Apply**
- **Test it**

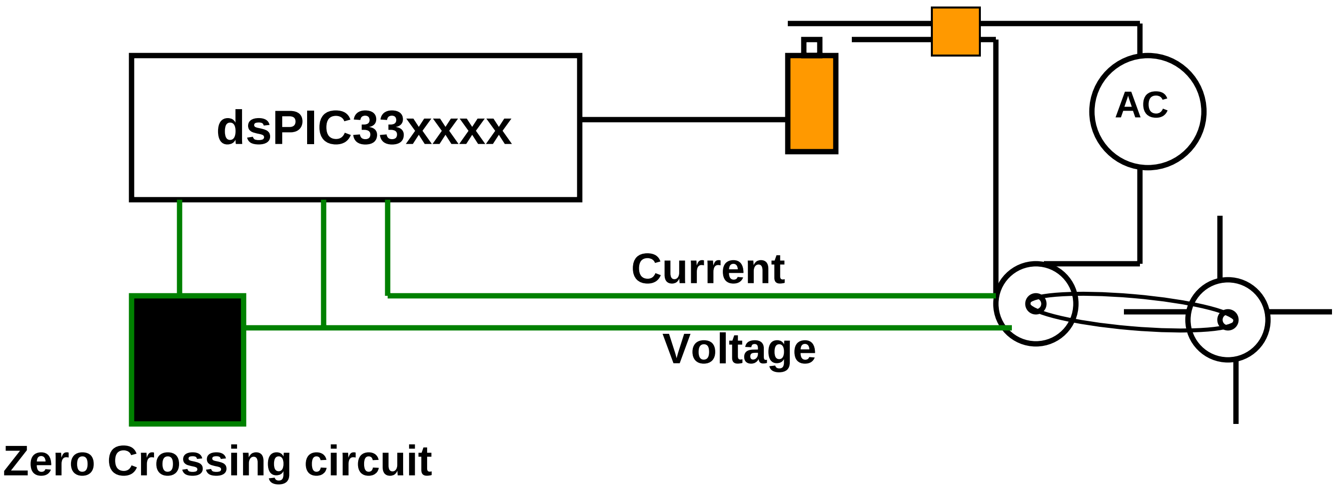
Circuit Breaker Project with Zero Crossing and Multiple A/D Inputs

Zero Crossing Applications

- **Used in**
  - AC Power Metering
  - Circuit Breaker (Lab 3)
  - UPS Systems
  - AC Motors
  - Appliance Speed Control Applications
- **Used to synchronize A/D conversions and calculations with current line cycle**
  - Line frequency, phase and A/D timing derived from Zero Crossing

Circuit Breaker Project

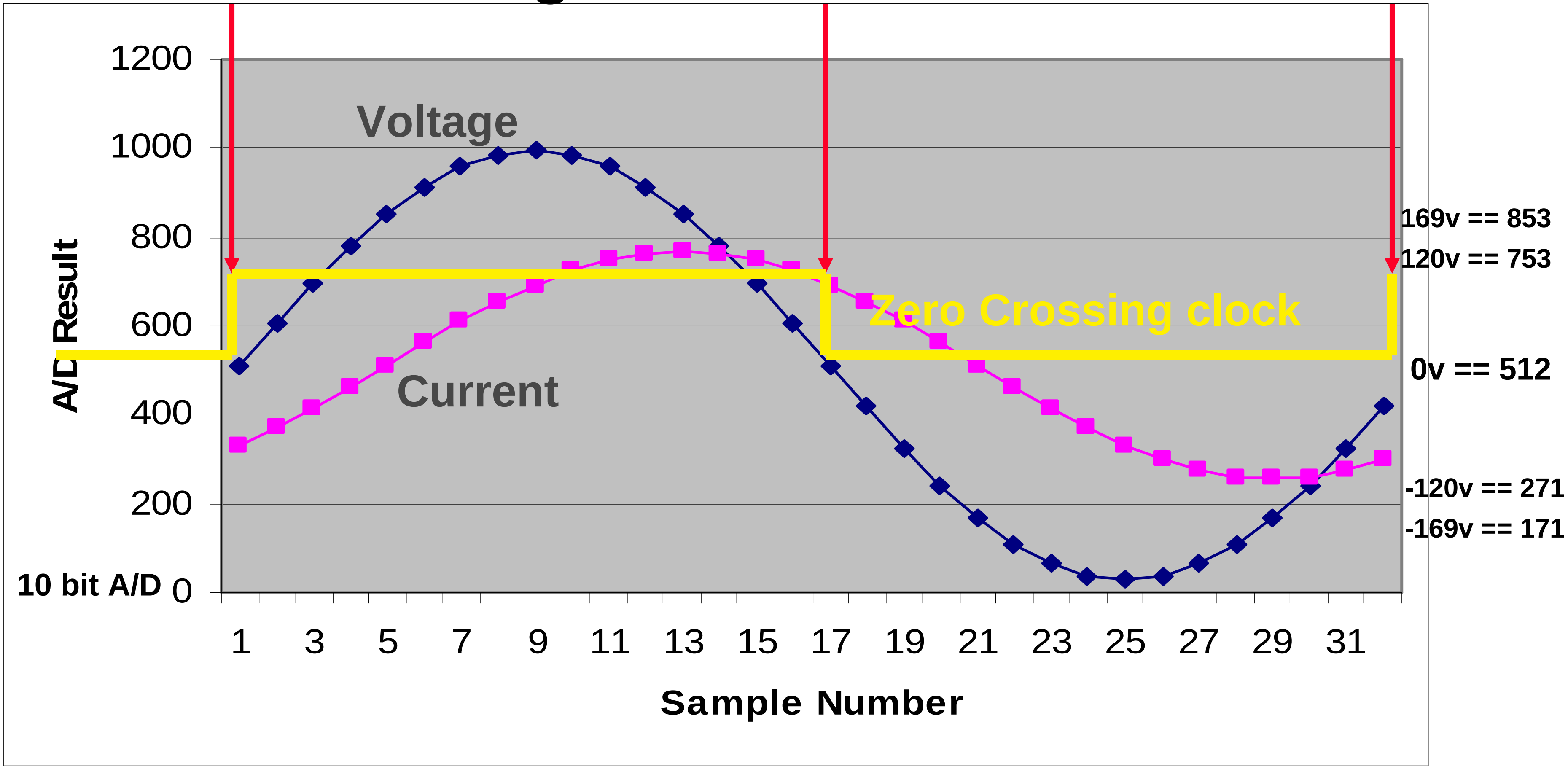
- What are we trying to do? What is the end goal?
- Verify the code works and the Breaker trips



- What do we need to verify the code
  - Firmware complete (Lab project code is done for you)
  - Input:- Simulated A/D values for voltage and current
  - Input:- Simulated zero crossing clock
  - Output:- Ability to monitor trip output for the circuit breaker
  - Output:- Ability to monitor or extract data to compare results

Stimulus Inputs

- Two waveforms Voltage & Current
- Zero Crossing



Circuit Breaker Stimulus Requirements Lab 3

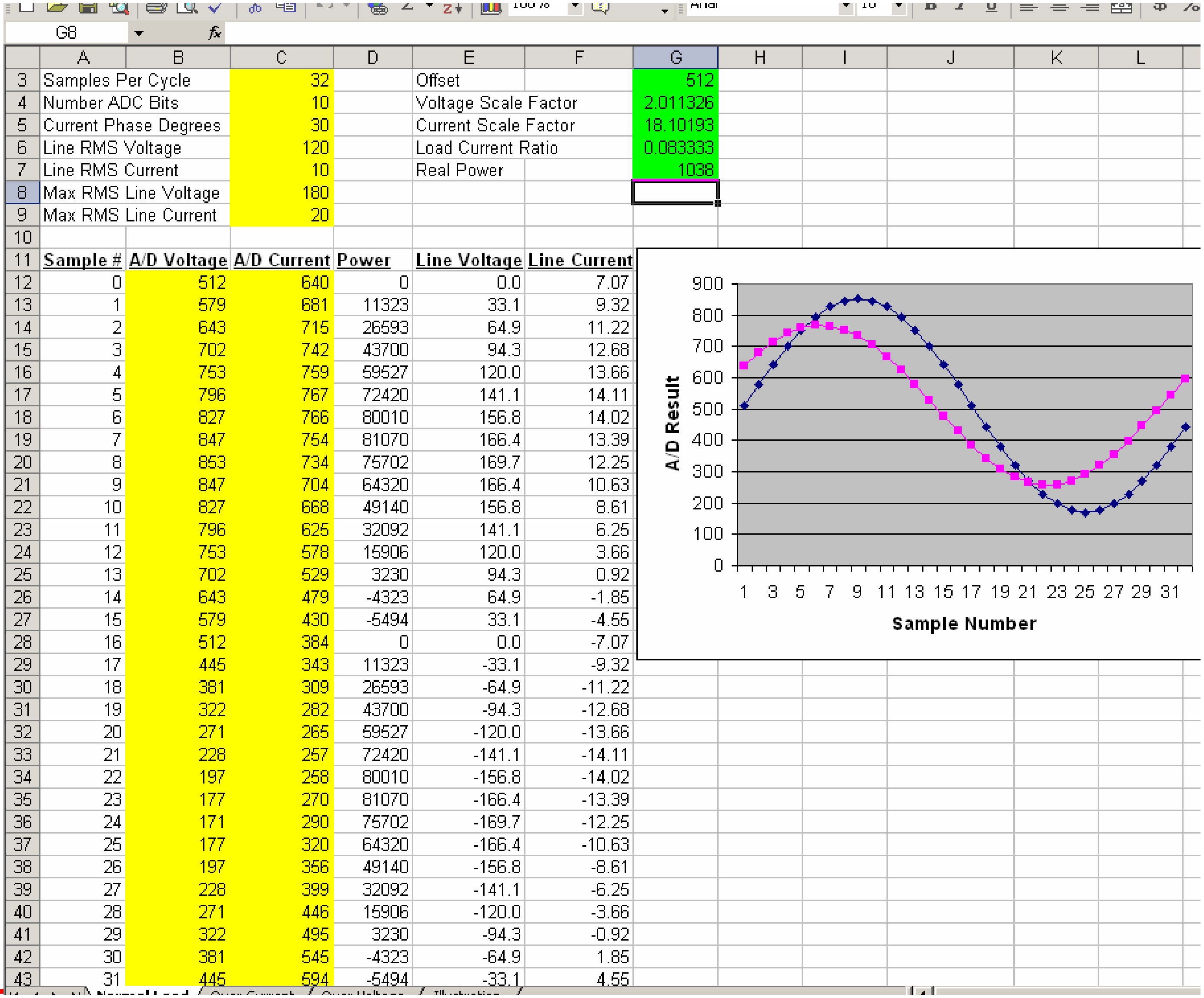
- A/D **voltage and current values**:- Register Stimulus one file two data columns inject into AD1BUF0
  - Scaled A/D inputs :- Excel spread sheet
- Zero Crossing **60Hz line frequency clock** :- Clock Stimulus inject into IC1
- **Solenoid** Trip output:- Watch window LATD/PORTD [Bit 1]
- Verification:
  - Method **A**
    - Trace intermediate Power values
    - Verify in Excel spread sheet VoltageCurrent.xls
  - Method **B**
    - Monitor power readings with DMCI
    - Adjust power level to test breaker with DMCI

General A/D File Data

- A/D reads a new value into the result buffer from the file whenever it is requested to perform a conversion
- The format of rows and columns is irrelevant. White space is used as delimiters for each value.
- For 8-bit PIC® microcontrollers this is one value each time
- For 16-bit PIC microcontrollers there can be simultaneous samples. In this case the number of samples is the number of values read

A/D Data:- EXCEL Spreadsheet

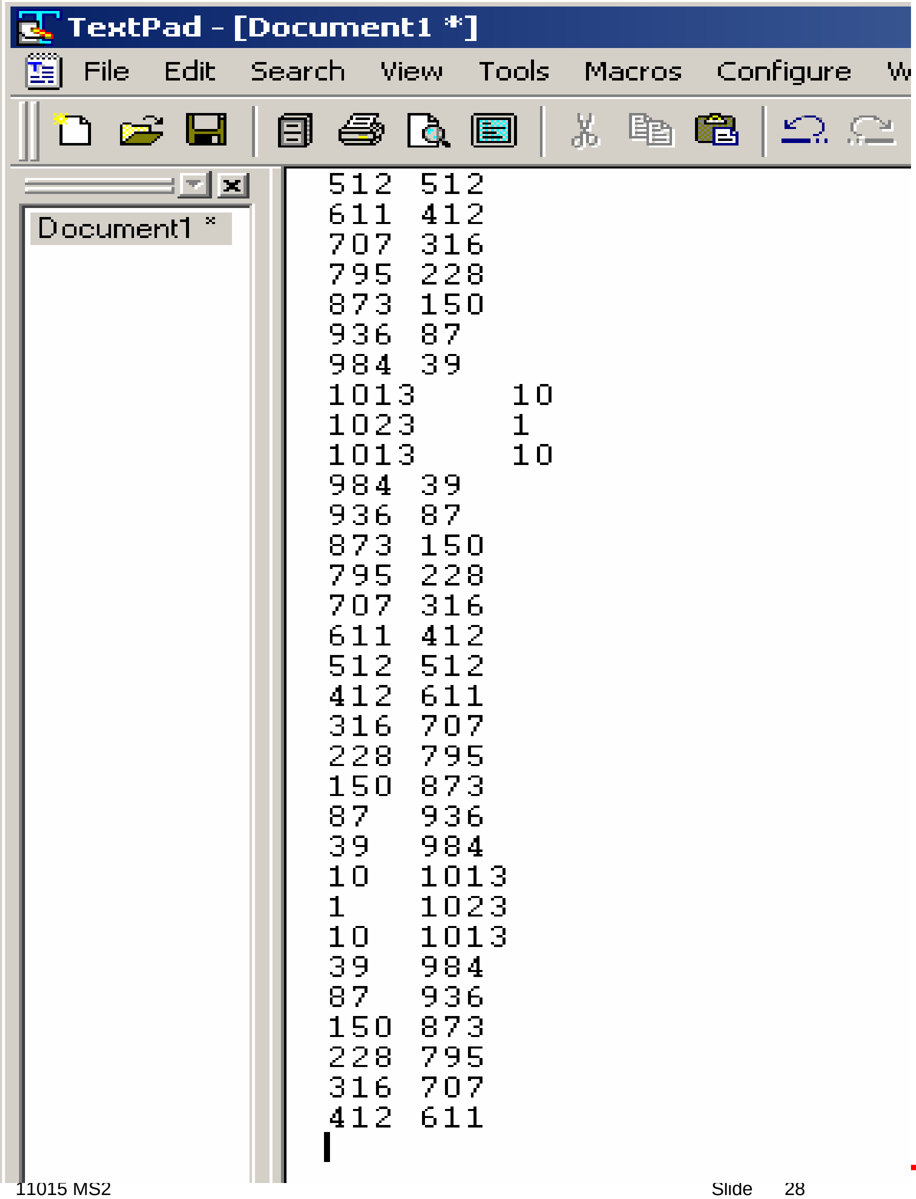
- Create 2 columns of data, charting results
- Copy the 2 columns of data and paste into data file xxx.txt



	A	B	C	D	E	F	G
3	Samples Per Cycle		32		Offset		512
4	Number ADC Bits		10		Voltage Scale Factor		2.011326
5	Current Phase Degrees		30		Current Scale Factor		18.10193
6	Line RMS Voltage		120		Load Current Ratio		0.083333
7	Line RMS Current		10		Real Power		1038
8	Max RMS Line Voltage		180				
9	Max RMS Line Current		20				
10							
11	Sample #	A/D Voltage	A/D Current	Power	Line Voltage	Line Current	
12	0	512	640	0	0.0	7.07	
13	1	579	681	11323	33.1	9.32	
14	2	643	715	26593	64.9	11.22	
15	3	702	742	43700	94.3	12.68	
16	4	753	759	59527	120.0	13.66	
17	5	796	767	72420	141.1	14.11	
18	6	827	766	80010	156.8	14.02	
19	7	847	754	81070	166.4	13.39	
20	8	853	734	75702	169.7	12.25	
21	9	847	704	64320	166.4	10.63	
22	10	827	668	49140	156.8	8.61	
23	11	796	625	32092	141.1	6.25	
24	12	753	578	15906	120.0	3.66	
25	13	702	529	3230	94.3	0.92	
26	14	643	479	-4323	64.9	-1.85	
27	15	579	430	-5494	33.1	-4.55	
28	16	512	384	0	0.0	-7.07	
29	17	445	343	11323	-33.1	-9.32	
30	18	381	309	26593	-64.9	-11.22	
31	19	322	282	43700	-94.3	-12.68	
32	20	271	265	59527	-120.0	-13.66	
33	21	228	257	72420	-141.1	-14.11	
34	22	197	258	80010	-156.8	-14.02	
35	23	177	270	81070	-166.4	-13.39	
36	24	171	290	75702	-169.7	-12.25	
37	25	177	320	64320	-166.4	-10.63	
38	26	197	356	49140	-156.8	-8.61	
39	27	228	399	32092	-141.1	-6.25	
40	28	271	446	15906	-120.0	-3.66	
41	29	322	495	3230	-94.3	-0.92	
42	30	381	545	-4323	-64.9	1.85	
43	31	445	594	-5494	-33.1	4.55	

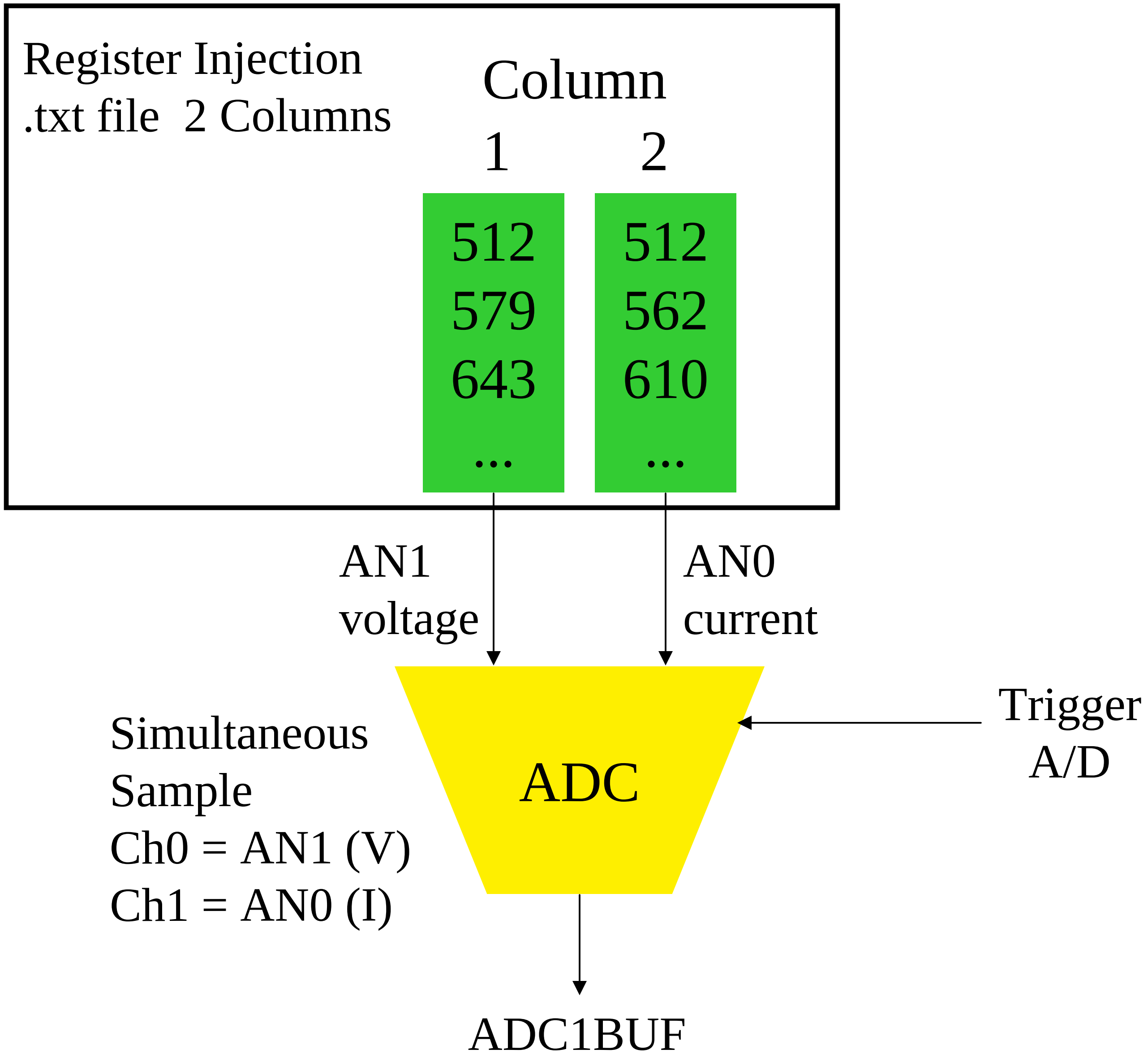
A/D Data File

- White space delimited values
- One line per set of A/D results
- N channels of A/D sampling equates to N columns of data



```
512	512
611	412
707	316
795	228
873	150
936	87
984	39
1013	10
1023	1
1013	10
984	39
936	87
873	150
795	228
707	316
611	412
512	512
412	611
316	707
228	795
150	873
87	936
39	984
10	1013
1	1023
10	1013
39	984
87	936
150	873
228	795
316	707
412	611
```

A/D Value Injection



Register Injection
.txt file 2 Columns

Column 1	Column 2
512	512
579	562
643	610
…	…

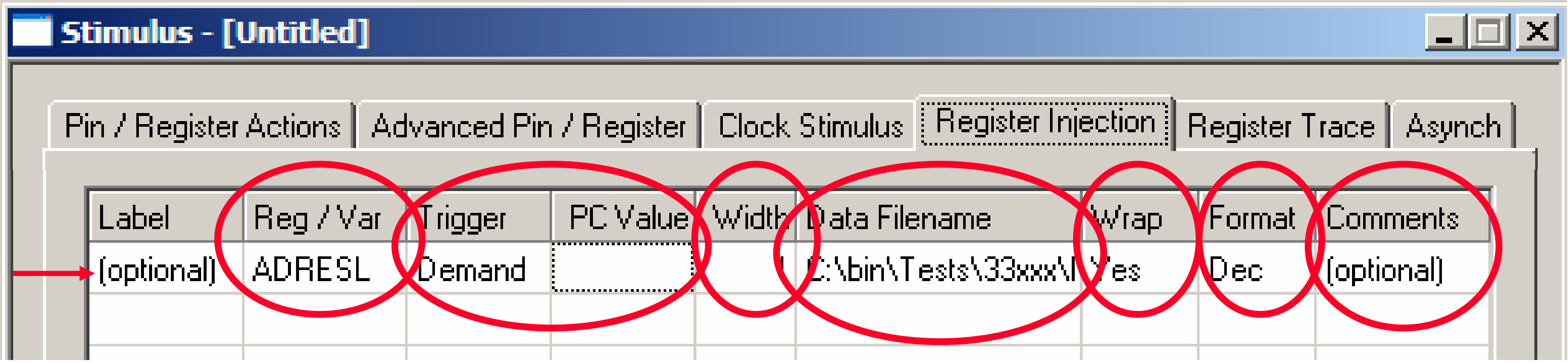
AN1
voltage

AN0
current

Trigger
A/D

Simultaneous
Sample
Ch0 = AN1 (V)
Ch1 = AN0 (I)

Register Injection

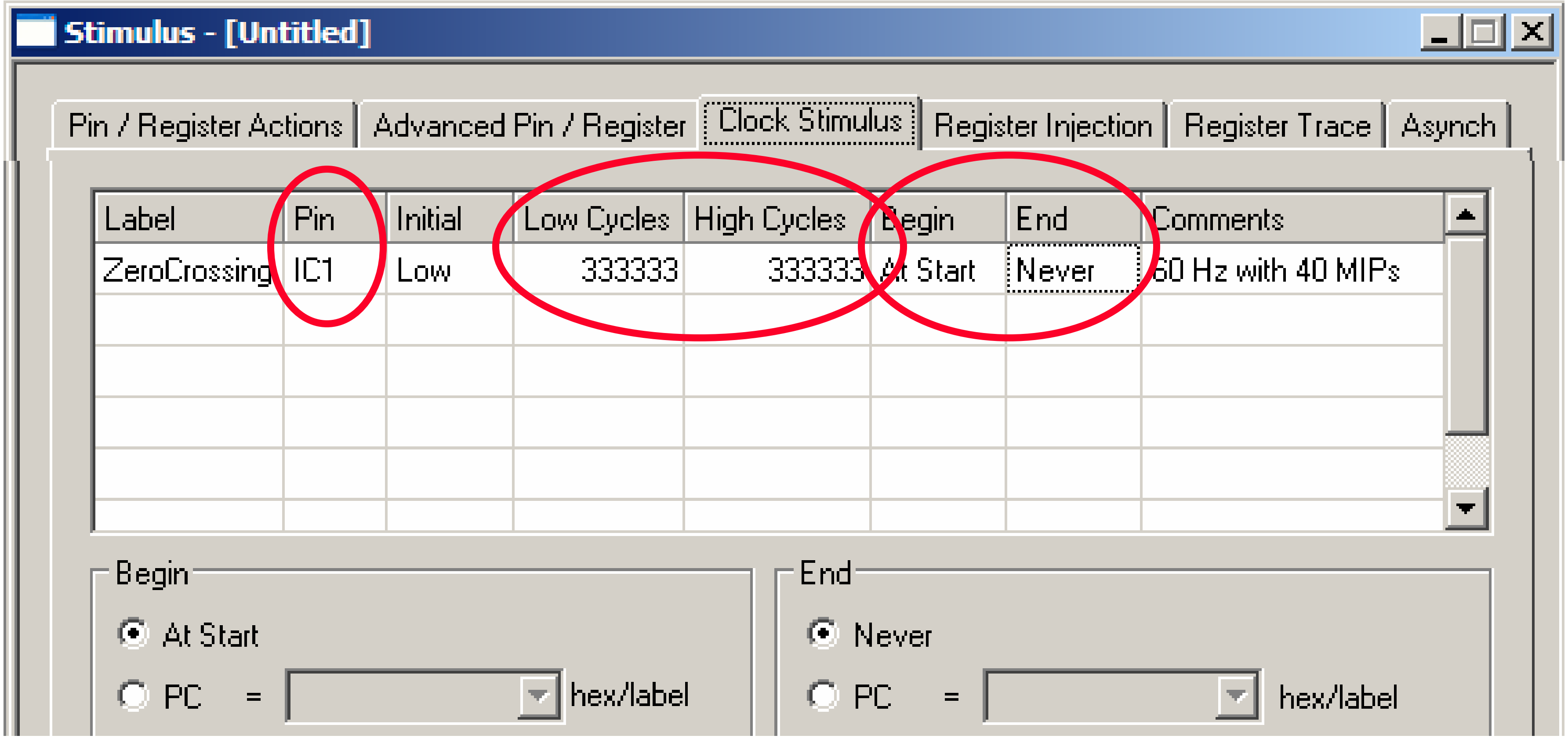


Stimulus - [Untitled]

Label	Reg / Var	Trigger	PC Value	Width	Data Filename	Wrap	Format	Comments
(optional)	ADRESL	Demand		1	C:\bin\Tests\33xxx\I	Yes	Dec	(optional)

- Each row provides a separate Register Injection
- Register or Variable to inject the data into
- Trigger type Demand (when read) or PC= ‘Label’
- Data width
- File name of data
- Wrap around to start (continues until stopped by user)
- Format
- Optional comments

Zero Crossing Clock



Label	Pin	Initial	Low Cycles	High Cycles	Begin	End	Comments
ZeroCrossing	IC1	Low	333333	333333	At Start	Never	60 Hz with 40 MIPs

Begin

● At Start

○ PC = hex/label

End

● Never

○ PC = hex/label

- **Create a clock to simulate the zero crossing detection**
- **Use clock stimulus start always end never**

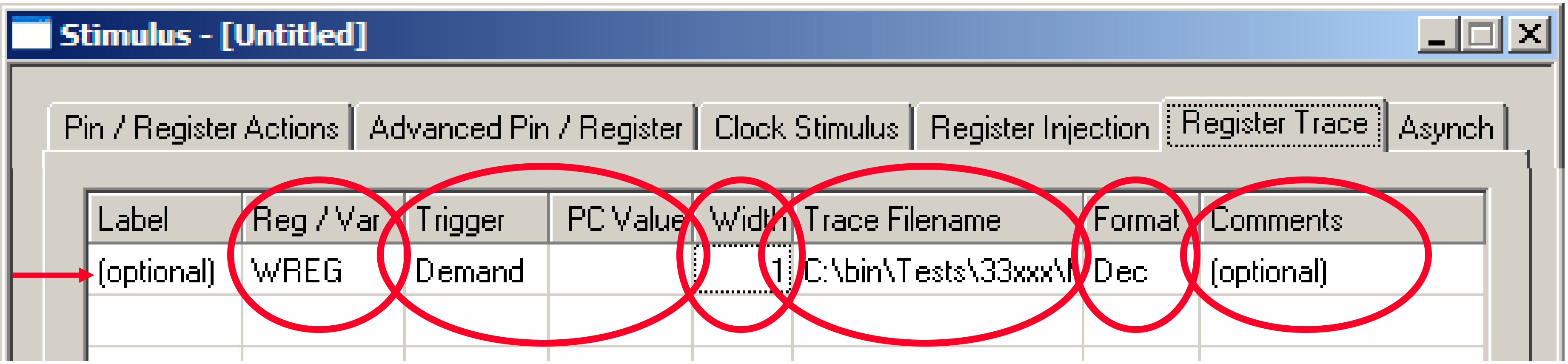
Let’s do it Lab 3a

- Create A/D data file using:- VoltageCurrent.xls
- Use Workspace CircuitBreaker.mcw
- Create 60hz stimulus ZC clock, attach A/D data file within Stimulus dialog
- Apply stimulus
- Build, execute code
- View watch window, shows “Power”
  Should be equal to real power in spread sheet
- Optionally view file registers, which shows data results in DMA ram

Verification Method A

Register Trace

Verification Method A Register Trace



Label	Reg / Var	Trigger	PC Value	Width	Trace Filename	Format	Comments
(optional)	WREG	Demand		1	C:\bin\Tests\33xxx\I	Dec	(optional)

- Each row provides a separate Register trace
- Optional Label
- Register or Variable to trace data from
- Trigger type Demand (when written) or PC= ‘Label’
- Data width
- File name for data

Verification Method A using EXCEL

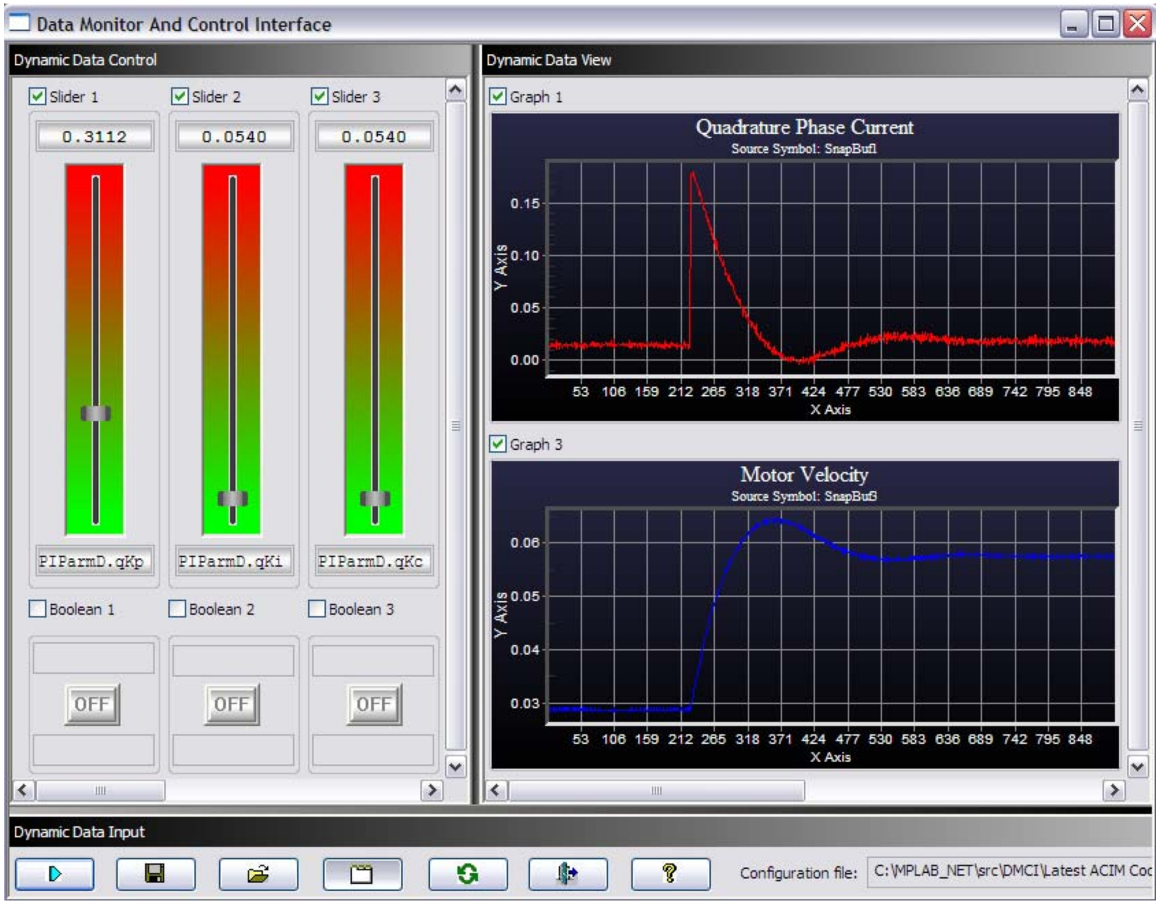
- Open Excel VoltageCurrent.xls
- Cut and Paste the traced data from file created during execution
- Compare with the computed Power values within the spread sheet

Verification Method B

Data Monitor and Control Interface (DMCI)

DMCI

- Plug-in tool shipped with MPLAB® IDE
- Tightly integrated into the IDE
- Use to control application parameters and/or monitor data (graphically)
- Debug tool independent



☑ Slider 1 — 0.3112 — PIParmD.qKp

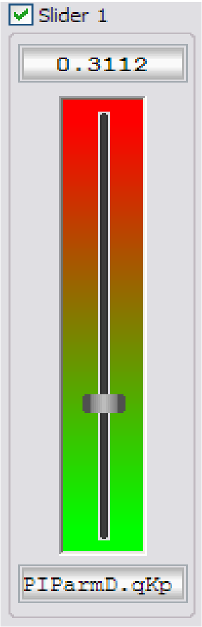
☑ Slider 2 — 0.0540 — PIParmD.qKi

☑ Slider 3 — 0.0540 — PIParmD.qKc

Configuration file: C:\MPLAB_NET\src\DMCI\Latest ACIM Coc

DMCI
Three types of control

- Sliders (9) provide variable control between min/max limits
- Built-in fractional conversion for easy PID control

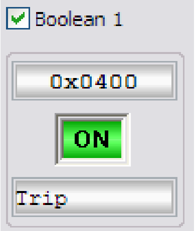


☑ Slider 1

0.3112

PIParmD.qKp

- Booleans (9) provide on/off state control
- Built-in support for bit fields



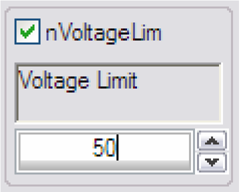
☑ Boolean 1

0x0400

Trip

DMCI
Three types of control (continued)

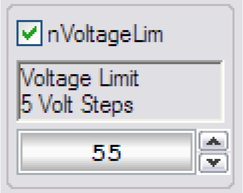
- User defined groups (6) have 5 direct edit entry controls
- Can specify inc/dec steps using spin button
- Can define ‘C’ Enum style step entries using spin button
- Float input mode (built-in fractional conversion)



☑ nVoltageLim

Voltage Limit

50

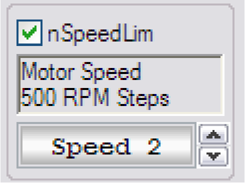


☑ nVoltageLim

Voltage Limit
5 Volt Steps

55

Increment/Decrement Value = 5

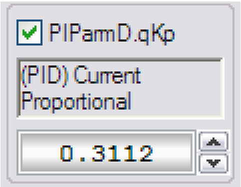


☑ nSpeedLim

Motor Speed
500 RPM Steps

Speed 2

Current Enum List Value = 0x03E8

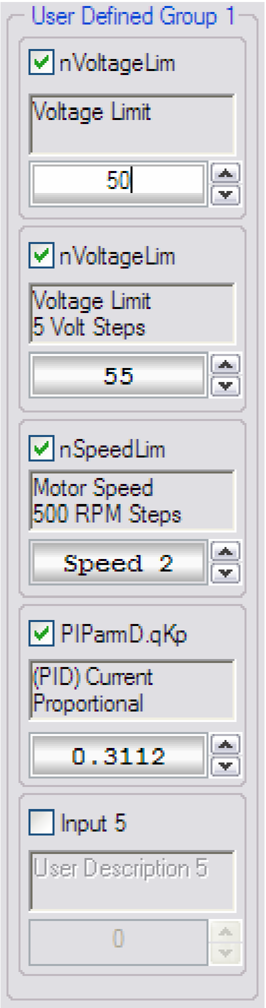


☑ PIParmD.qKp

(PID) Current
Proportional

0.3112

Increment/Decrement Factor = 0.0100, Target Value = 0x27D5



User Defined Group 1

☑ nVoltageLim

Voltage Limit

50

☑ nVoltageLim

Voltage Limit
5 Volt Steps

55

☑ nSpeedLim

Motor Speed
500 RPM Steps

Speed 2

☑ PIParmD.qKp

(PID) Current
Proportional

0.3112

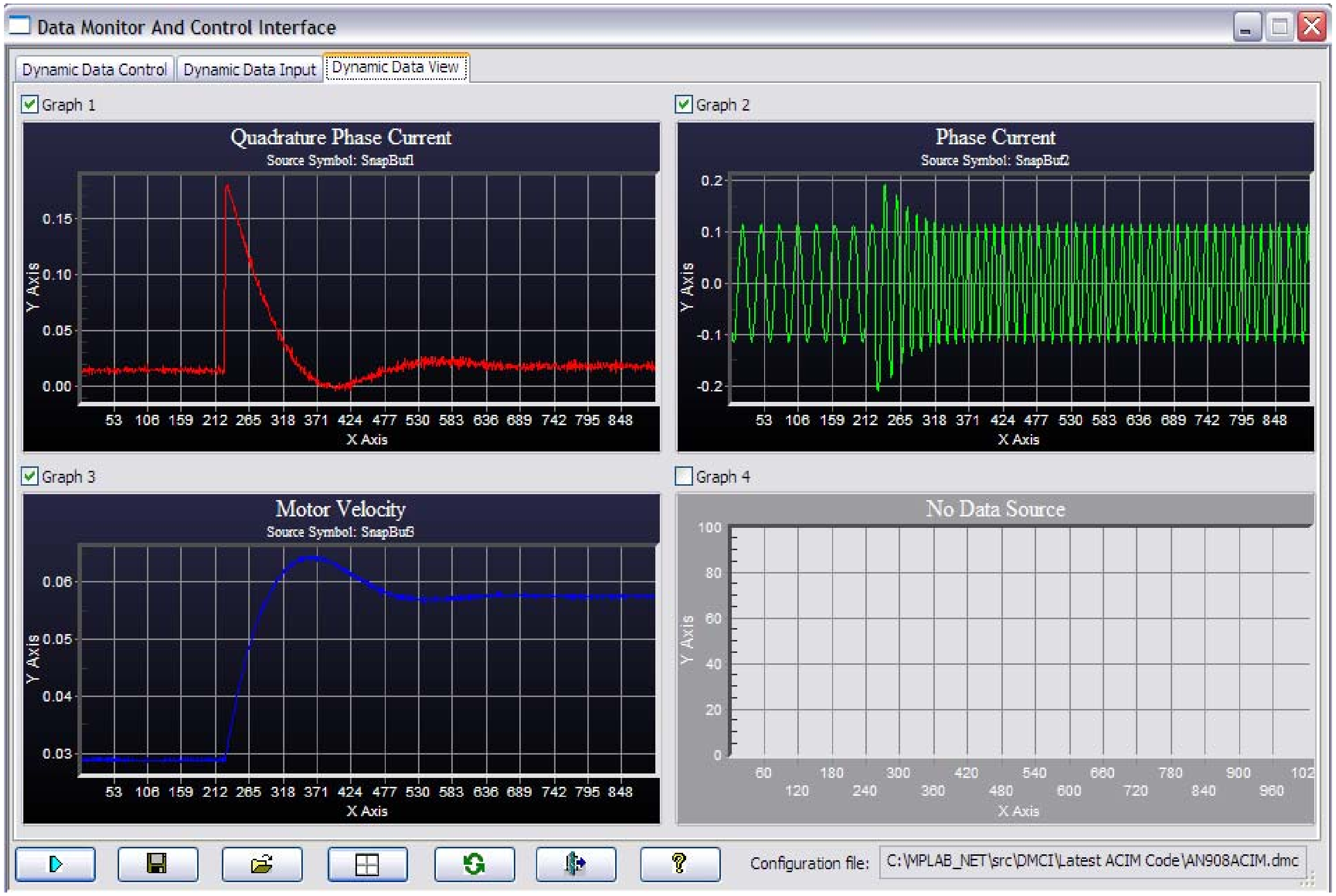
□ Input 5

User Description 5

0

DMCI
Four Graphs

- **View vector based performance history**
- **MPLAB® SIM Real-Time updates**
- **MPLAB REAL ICE™ In-Circuit Emulator**



Data Monitor And Control Interface

Dynamic Data Control | Dynamic Data Input | Dynamic Data View

☑ Graph 1

☑ Graph 2

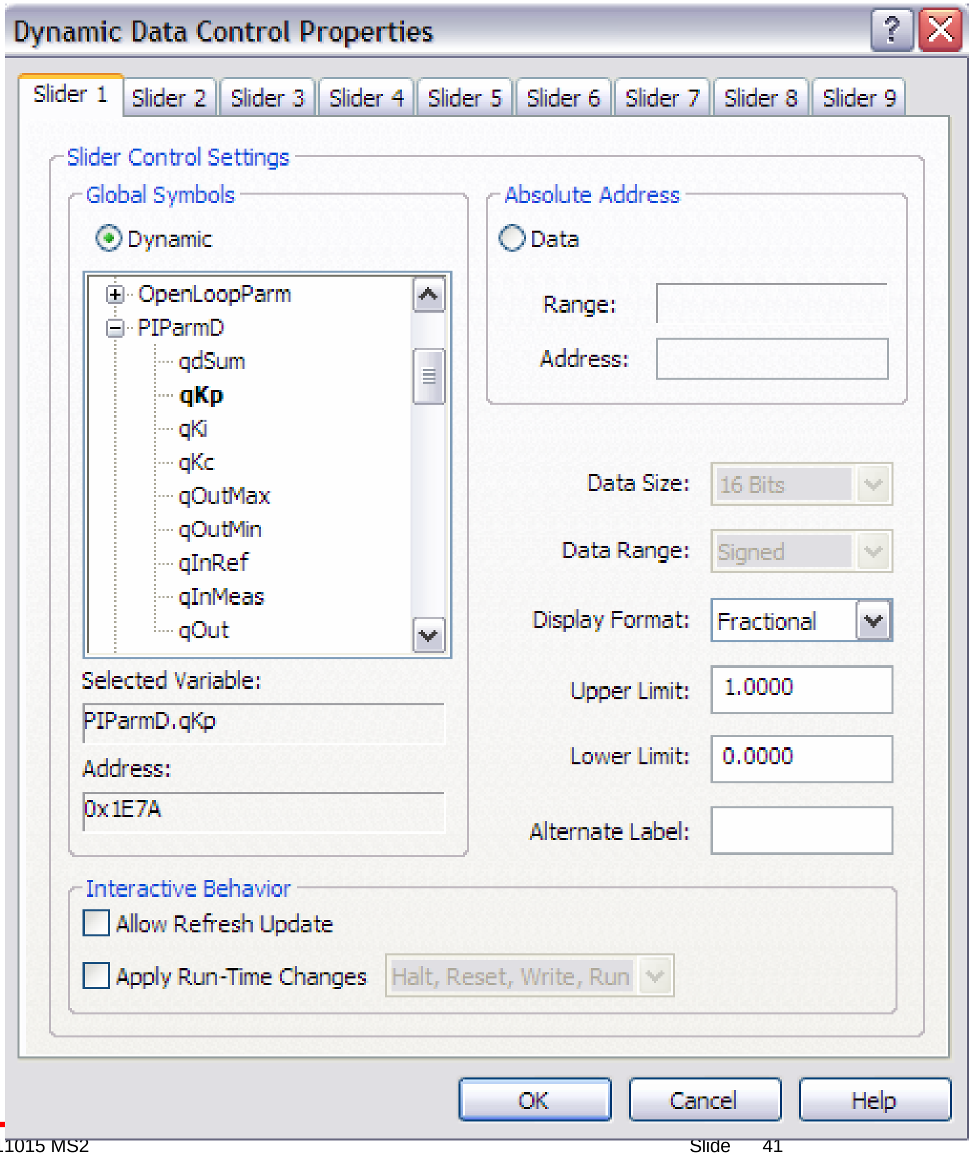
☑ Graph 3

□ Graph 4

Configuration file: C:\MPLAB_NET\src\DMCI\Latest ACIM Code\AN908ACIM.dmc

DMCI
Control configuration

- Context based property dialogs to hook a control to a software variable or memory address
- Global symbols identified as you add them and re-compile
- Dynamic selection provides address, data size, and data range automatically
- Data Absolute allows manual entry of Address, data size and range



Slider Control Settings

Global Symbols

● Dynamic

Selected Variable: PIParmD.qKp

Address: 0x1E7A

Absolute Address

○ Data

Range:

Address:

Data Size: 16 Bits

Data Range: Signed

Display Format: Fractional

Upper Limit: 1.0000

Lower Limit: 0.0000

Alternate Label:

Interactive Behavior

□ Allow Refresh Update

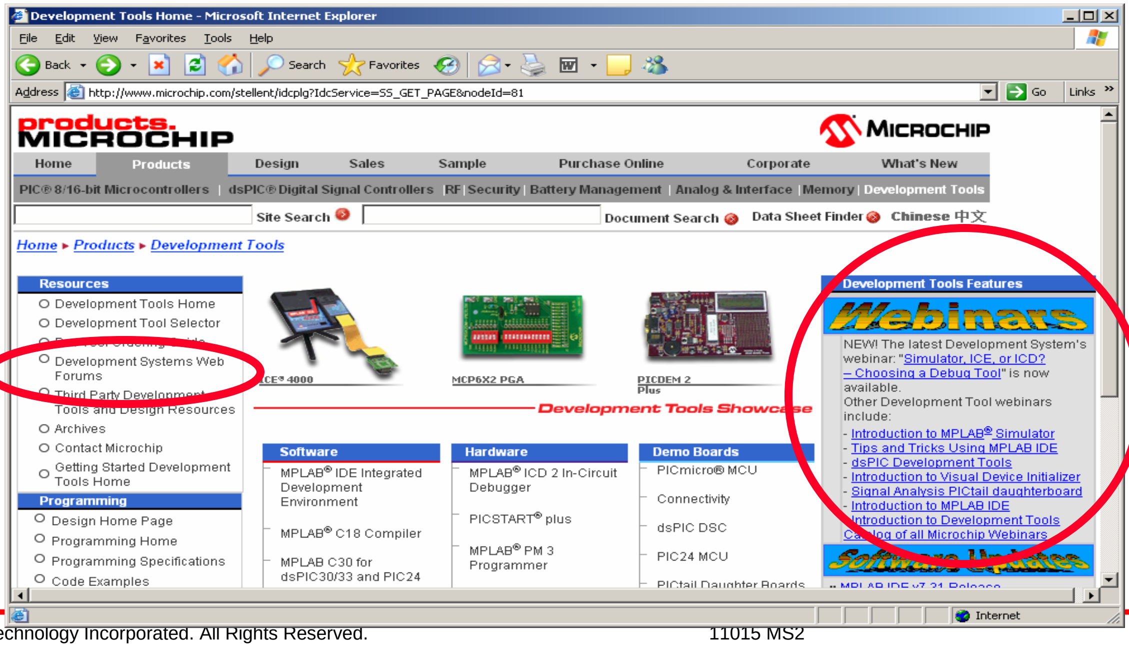
□ Apply Run-Time Changes Halt, Reset, Write, Run

Let’s do it Lab 3b

- **Open DMCI from tools menu (*Tools>Data Monitor and Control Interface*)**
- **Enable 1 slider and 4 graphs**
- **Right click on slider**
- **Select parameter to control**
  - Load
- **Right click on each graph**
- **Select parameters to monitor**
  - Current
  - Voltage
  - Solenoid
  - Power
- **Run the application and adjust Load using slider**

Where to find out more

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


Development Tools Home - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address
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
ICE® 4000
MCP6X2 PGA
PICDEM 2 Plus
Development Tools Showcase
Webinars
Software Updates
Internet

http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=81

Home ▸ Products ▸ Development Tools

Resources

- Development Tools Home
- Development Tool Selector
- Dev Tool Ordering Guide
- 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

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

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

MPLAB IDE v7.31 Release

Summary

- **Complex signals**
  - Key bounce
  - Encoder
- **A/D stimulus on multiple channels**
- **Verification using DMCI or Log data**
- **More…**

THANK YOU!

Appendix

- **Useful Links**
- **Lab 1**
  - Key bounce
- **Lab 2**
  - Encoder
- **Lab 3**
  - CircuitBreaker with Zero Crossing

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

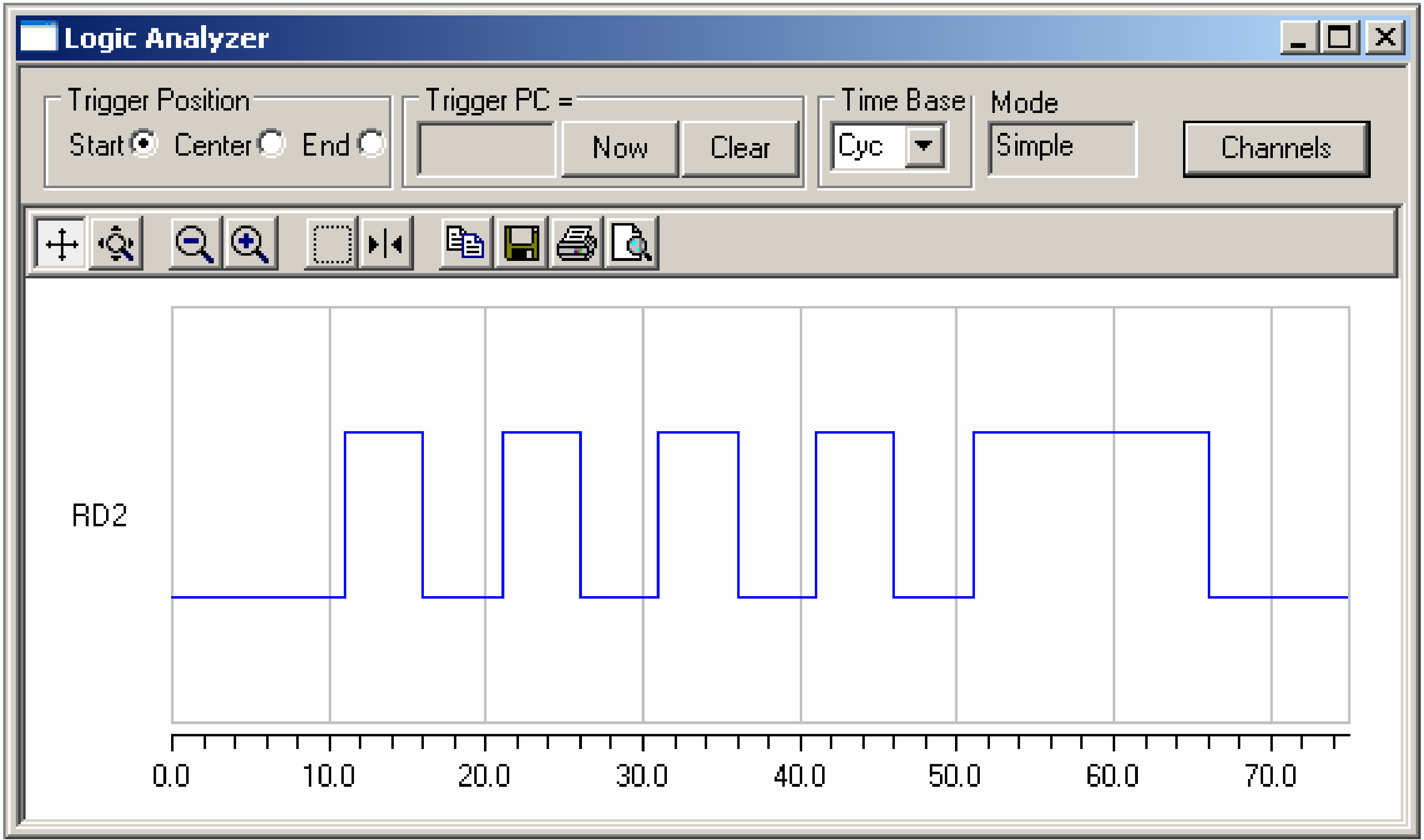
11015_MS2

MPLAB® Simulators Advanced Stimulus

Lab 1

Key Bounce Lab 1

- **Objective: Create pulse train triggered by Asynch stimulus**



Logic Analyzer

Trigger Position
Start ● Center ○ End ○

Trigger PC =

Time Base: Cyc

Mode: Simple

Key Bounce Lab 1

- **Open MPLAB® IDE**
  - Select menu item "*Configure>Select Device*"
  - Select a "pic18C442" device "OK"
  - Select menu item "*Debugger>Select Tool>MPLAB SIM*"
- **Open Stimulus window. ASYNCH event**
  - Select "*Debugger>Stimulus>New Workbook*"
  - Select "Asynch" tab
  - Select pin with drop down list "RD2" (Port D bit 2)
  - Select action "Pulse High"
  - Enter pulse width "60" cycles

Key Bounce Lab 1

- **Stimulus. SYNCH event**
  - Select the “Clock Stimulus” tab at the top
  - Enter an optional label if desired
  - Select the “RD2” pin from drop down under “Pin” column
  - Select “Low” from drop down under “Initial” state column
  - Set “Low Cycles” to “5”. Set “High Cycles” to “5”
  - Select the “Begin” box on the stimulus row. By default this will be set to “At Start”
  - Move down to the “Begin” dialog edit area (lower left)
  - Change from the “At Start” to the “Pin” option (radio button)
  - Select “RD2” in drop down list for the “Pin” to use

Key Bounce Lab 1

- **Stimulus. SYNCH event cont…**
  - Select “High” in the adjacent drop down box
  - The clock will begin when RD2 goes to a high state and the selections in the Begin area will be reflected within the “Begin” column above
  - Select the “End” box on the stimulus row. By default this will be set to “Never”
  - Move to the “End” dialog edit area (lower right)
  - Select the option to end on “Cycle” (radio button)
  - Set cycles to “50”
  - Select “from clock start” in the adjacent drop down box
  - The clock will end 50 cycles after the starting trigger and the selections in the End area will be reflected within the “End” column above
  - Optionally enter a comment within the stimulus row

Key Bounce Lab 1

- **Apply Synchronous Stimulus**
  - Select the “Apply” button at the bottom of the stimulus window
  - Select the “Asynch” tab in preparation to fire the asynchronous stimulus.
  - You are now ready to test

Key Bounce Lab 1

- **Testing. Write code for animation (simple loop)**
  - Open a new file, "*File>New*". No real code needed for testing
  - Enter a tab then a "nop" on the first line
  - Enter a tab then a "goto 0" on the second line
  - Enter a tab then "end" directive on the third line
  - Select "Save", give the file a name with an 'asm' extension
  - Select the menu "*Project>Quickbuild*" (file must be in focus)
- **Testing. Enable trace**
  - Select "*Debugger>Settings*" and check the box "Trace All"
  - Select the "Animation / Realtime Updates" tab. Set "animate step time" to 100 ms
  - Select OK at the bottom to close the settings dialog
  - This allows tracing of IO pin data so we can view it in the logic analyzer

Key Bounce Lab 1

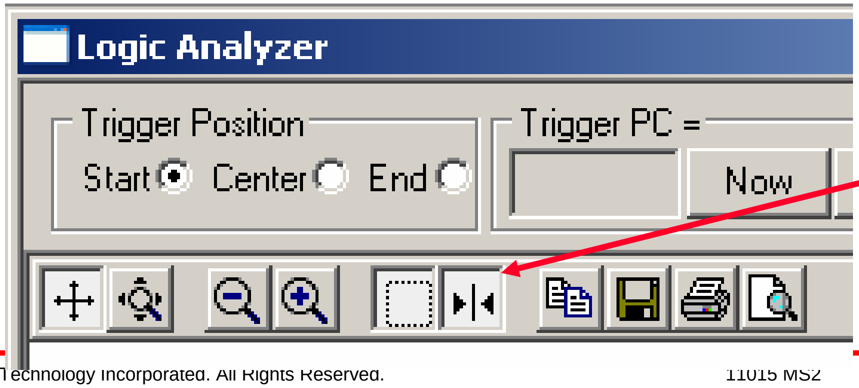
- **Testing. Execution**
  - Select reset and then animate (double arrow icon in toolbar)
  - Fire the Asynch stimulus “RD2 pulse high 60 cycles” from the Asynch tab in the Stimulus window
  - Due to the animate speed being 0.1 seconds per step halt after about 6 seconds to allow the synch clock to complete

- **Testing. Verify input pulses**
  - Open the Logic Display “*View>Simulator Logic Analyzer*”
  - Select the “Channels” button and select the “RD2” signal
  - Press the “Add” button to add it to selected signals
  - Click OK

Key Bounce Lab 1

- **Testing. Verify input pulses**
  - View the output of the RD2 wave form
  - If the Logic Analyzer is already open, you will see it update on each step during animation

- **Extra Objective**
  - Turn the cursors on within the Logic Analyzer and measure between the first and last rising edges
  - Is it what you expect?



Cursors

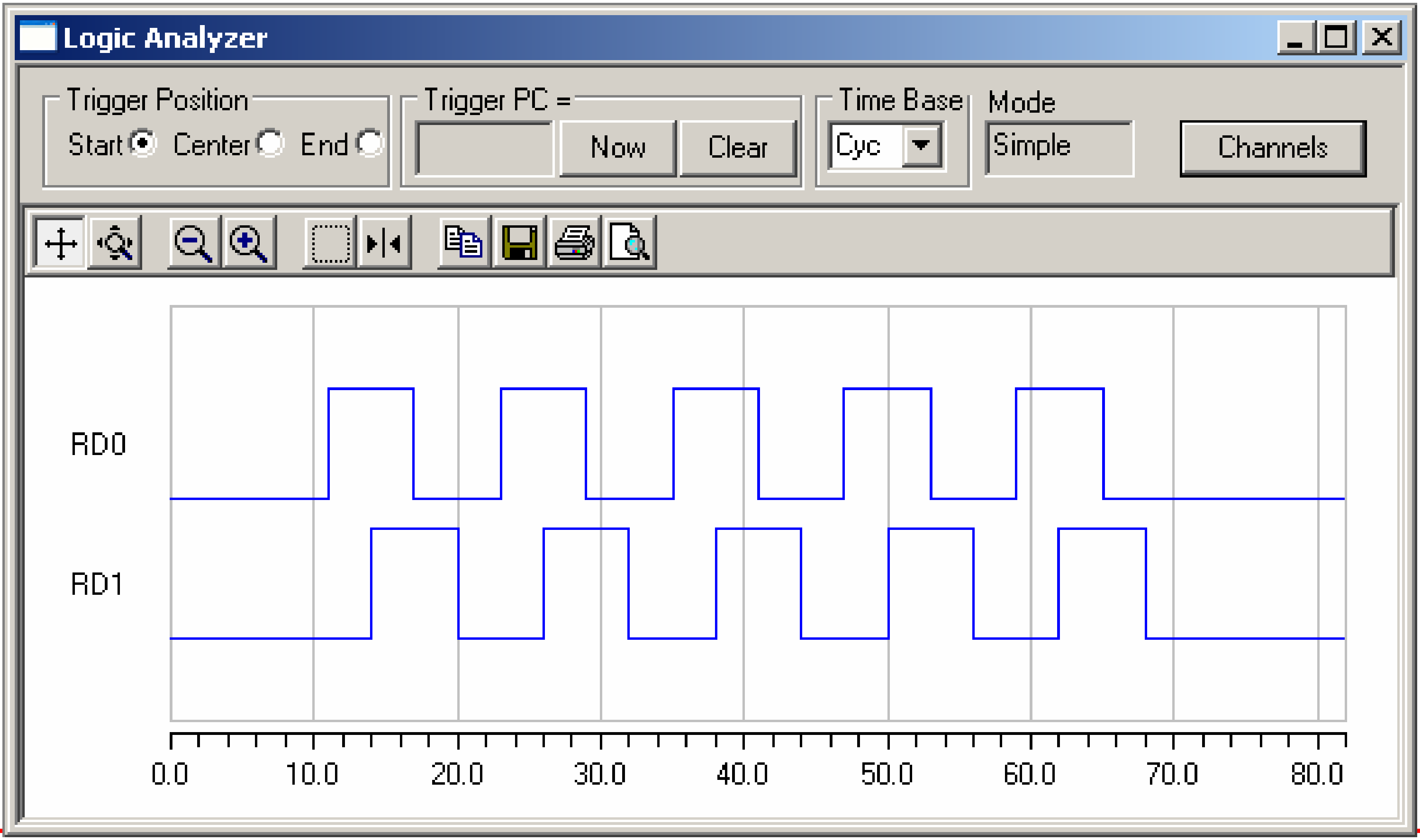
11015_MS2

MPLAB® Simulators Advanced Stimulus

Lab 2

Encoder Lab 2

- **Objective: Create 2 pulse trains RD1 based on RD0**



Logic Analyzer

Trigger Position: Start ● Center ○ End ○

Trigger PC =

Time Base: Cyc

Mode: Simple

Encoder Lab 2

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

- **Open Stimulus window. ASYNCH event**
  - Select “*Debugger>Stimulus>New Workbook*”
  - Select pin with drop down list “RD0” (Port D bit 0)
  - Select action “Set High”
    (The Synch clock will drive this low on completion of clock)

Encoder Lab 2

- **Stimulus. SYNCH clock 1**
  - Select the “Clock Stimulus” tab at the top
  - Enter an optional label if desired
  - Select the “RD0” pin from drop down under “Pin” column
  - Select “High” from drop down under “Initial” state column
  - Set “Low Cycles” to “6”. Set “High Cycles” to “6”
  - Select the “Begin” box on the stimulus row. By default this will be set to “At Start”
  - Move down to the “Begin” dialog edit area (lower left)
  - Change from the “At Start” to the “Pin” option (radio button)
  - Select “RD0” in drop down list for the “Pin” to use

Encoder Lab 2

- **Stimulus. SYNCH clock 1**
  - Select “High” in the adjacent drop down box
  - The clock will begin when RD0 goes to a high state and the selections in the Begin area will be reflected within the “Begin” column above
  - Select the “End” box on the stimulus row. By default this will be set to “Never”
  - Move to the “End” dialog edit area (lower right)
  - Select the option to end on “Cycle” (radio button)
  - Set cycles to “60”
  - Select “from clock start” in the adjacent drop down box
  - The clock will end 60 cycles after the starting trigger and the selections in the End area will be reflected within the “End” column above
  - Optionally enter a comment within the stimulus row

Encoder Lab 2

- **Stimulus. SYNCH clock 2**
  - Select the "Advanced Pin / Register" tab
  - Under "Define Conditions"
  - Select the first box marked "Any" on COND1. From the drop down list select "Pin"
  - Under the next column select "RD0" (The "Pin" the Condition will be based on)
  - Leave the "=" comparison, Set the next box value to "1"
  - Select the "Wait" column and enter "3"
  - Select units to be "cycles"
  - Create a "COND2" the same way by doing the following
  - Select "Pin", select "RD0" again on the next row
  - Leave the "=" comparison, Set the next box value to "0"
  - Select the "Wait" column and enter "3"
  - Select units to be "cycles"

Encoder Lab 2

- **Stimulus. SYNCH clock 2**
  - Under the “Define Triggers”, the first row is enabled
  - Select the “Condition” column and select COND1
  - Select “Type” to be “Continuous”
  - Enter “0” in the “Re-Arm Delay”
  - Click on the column header “Click here to Add Signals”
  - Select “RD1” and click “Add” then “OK”. This adds the RD1 signal to allow it to be changed on a condition
  - Select the column “RD1” and enter “1”
  - Select the “Condition” column on the next row and select COND2
  - Select “Type” to be “Continuous”
  - Enter “0” in the “Re-Arm Delay”
  - Select the column “RD1” and enter “0”
  - The enable check will come on as you enter data

Encoder Lab 2

- **Apply Synchronous Stimulus**
  - Select the “Apply” button at the bottom of the stimulus window
  - Select the “Asynch” tab in preparation to fire the asynchronous stimulus.
  - You are now ready to test

Encoder Lab 2

- **Testing. Write code for animation (simple loop)**
  - Open a new file, “*File>New*”. No real code needed for testing
  - Enter a tab then a “nop” on the first line
  - Enter a tab then a “goto 0” on the second line
  - Enter a tab then “end” directive on the third line
  - Select “Save”, give the file a name with an ‘asm’ extension
  - Select the menu “*Project>Quickbuild*” (file must be in focus)
- **Testing. Enable trace**
  - Select “*Debugger>Settings*” and check the box “Trace All”
  - Select the “Animation / Realtime Updates” tab. Set “animate step time” to 100 ms
  - Select OK at the bottom to close the settings dialog
  - This allows tracing of IO pin data so we can view it in the logic analyzer

Encoder Lab 2

- **Testing. Execution**
  - Select “reset” and then “animate” (double arrow icon in toolbar)
  - Fire the Asynch stimulus “RD0 Set High” from the Asynch tab in the Stimulus window
  - Due to the animate speed being 0.1 seconds per step halt after about 6 seconds to allow the synch clock to complete

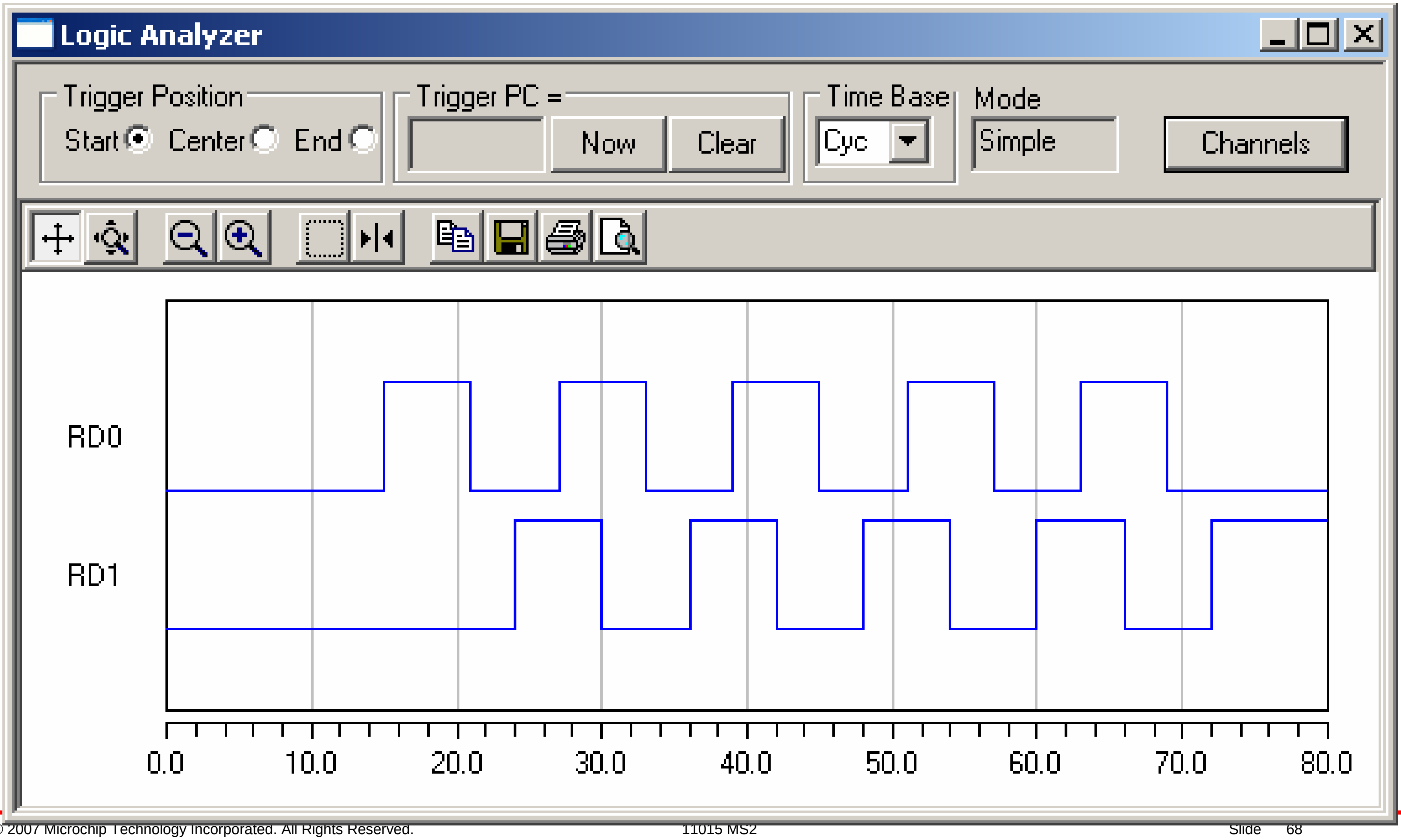
- **Testing. Verify input pulses**
  - Open the Logic Display “*View>Simulator Logic Analyzer*”
  - Select the “Channels” button and select the RD0 & RD1 signals
  - Press the “Add” button to add it to selected signals
  - Click OK

Encoder Lab 2

- **Testing. Verify input pulses**
  - View the output of the RD0 and RD1 wave forms
  - If the Logic Analyzer is already open, you will see it update on each step during animation
- **Extra Objective**
  - Define 2 more Triggers using the existing conditions and make RD1 lead RD0 by 90 deg
(switch the pulse train around)
  - Use the enable check boxes to turn one set off and the other on

Encoder Lab 2

- **Extra Objective Result**



Trigger Position: Start ● Center ○ End ○

Trigger PC =

Time Base: Cyc

Mode: Simple

11015_MS2

MPLAB® Simulators Advanced Stimulus

Lab 3

Circuit Breaker
Stimulus Requirements Lab 3

- **A/D voltage and current values**:- Register Stimulus one file two data columns inject into AD1BUF0
  - AC voltage scaled A/D input:- Excel spread sheet
  - AC current scaled A/D input:- Excel spread sheet
- **Zero Crossing 60Hz line frequency clock**:- Clock Stimulus inject into IC1
- **Asynch Test button**:- Stimulus Controller pulse high RD3
- **Asynch Reset button**:- Stimulus Controller pulse high RD2
- **Solenoid Trip output**:- Watch window LATD [Bit 1]

Circuit Breaker
Zero Crossing Lab 3a

- **Open MPLAB® IDE**
  - Select menu “*File>Open Workspace…*”
  - Select the “11015 MS2 / Lab3 / CircuitBreaker.mcw”
  - OR Select menu “*File>Recent Workspaces>CircuitBreaker”*
  - Build the project

- **Preparing stimulus**
  - Open the Excel spread sheet “VoltageCurrent.xls”
  - View the data and graph representations
  - Copy the 2 columns of data to be used as A/D readings
  - Within the MPLAB® IDE, open “*File>New”* and paste them into a new file
  - Save and name the file “xxxxxxxx.txt”

Circuit Breaker Zero Crossing Lab 3a

- **Open Stimulus and attach A/D file**
  - Select “*Debugger>Stimulus>New Workbook*”
  - Select the “Register Injection” tab at the top
  - Enter an optional label if desired
  - Select Register “AD1BUF0” to inject data
  - Select Trigger type “Demand”
  - Width will be “2” bytes
  - Add the data file name as specified in the first step
  - Select “Yes” for wrap
  - Select “Dec” for decimal data type
  - Add optional comment

Circuit Breaker
Zero Crossing Lab 3a

- **Stimulus define ZC clock**
  - Select the “Clock Stimulus” tab at the top
  - Enter an optional label if desired
  - Select “IC1” from drop down under “Pin” Column for InputCapture 1
  - Select “Low” as “Initial” state from drop down
  - Set “Low Cycles” to “333333”. Set “High Cycles” to “333333”   60hz clock at 40 MIPS (six 3’s in each)
  - Select the “Begin” box. Leave at default “At Start”
  - Select the “End” box. Leave at default “Never”
  - Add optional comment

Circuit Breaker
Zero Crossing Lab 3a

- **Apply Synchronous Stimulus**
  - Select the “Apply” button at the bottom of the stimulus window
  - You are now ready to test

Circuit Breaker
Zero Crossing Lab 3a

- **Testing. Execution**
  - Select “Reset” and then “Run”
  - Watch the variables in the watch window. Once the “Power” value has changed, stop the program

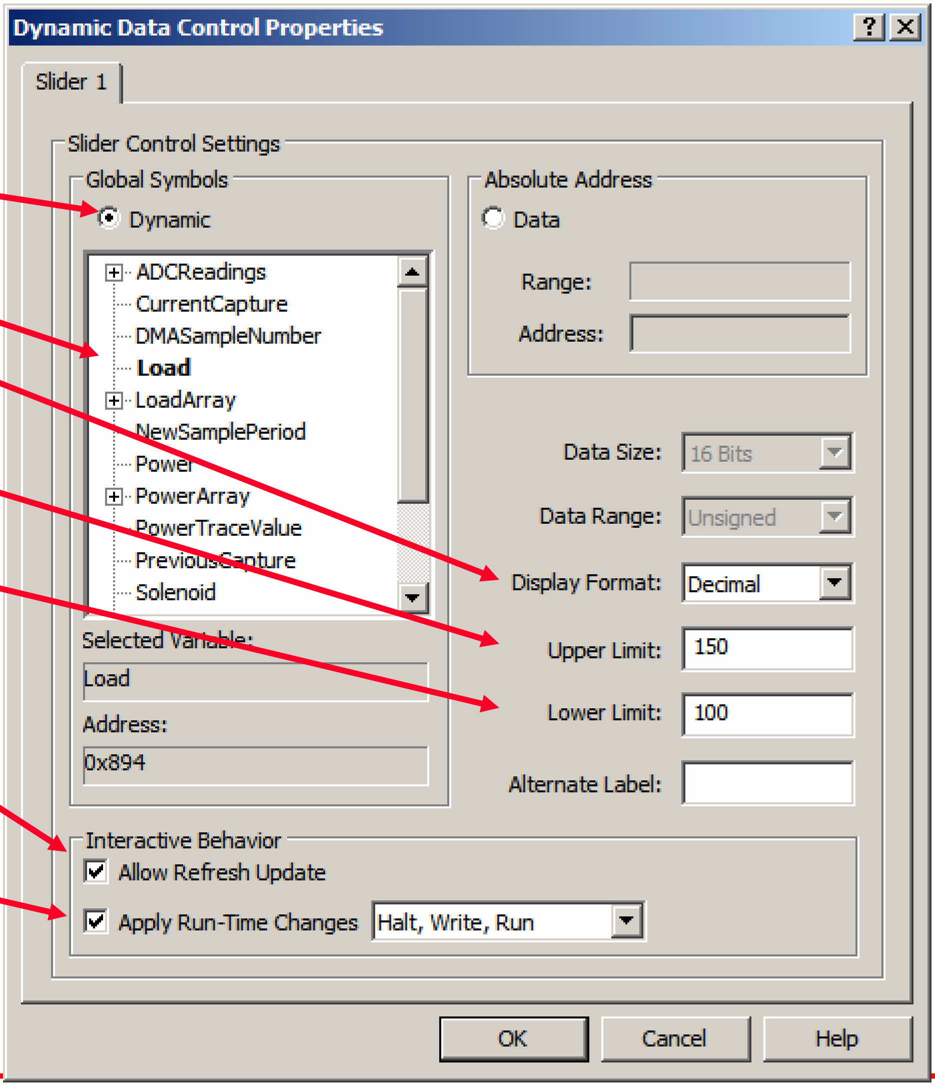
- **Testing. Verify Power**
  - Verify the Power value is equal to the Power value in the Excel spread sheet for the injected data. (One tab in the spread sheet for different test data)
  - View the File Register window at address 0x4780. Note the A/D data is placed here using the DMA and peripheral indirect address mode. (Handled totally by hardware within the silicon)

Circuit Breaker Zero Crossing Lab 3b

- **Verification using DMCI. Slider setup**
  - Select "*Tools>Data Monitor and Control Interface*"
  - Click "Tiled window view" button (bottom 4th button)
  - Adjust the tiles so you have 4 graphs and 1slider visible
  - Enable the slider by setting the check box in upper left
  - Right click in colored area of slider to bring up the configuration
  - Set the configuration up as displayed on next page

Circuit Breaker
Zero Crossing Lab 3b

- **Slider configuration**
  - Dynamic selection
  - “Load” variable
  - Display format Decimal
  - Upper limit 150
  - Lower limit 100
  - Allow refresh
  - Apply Run-Time changes as “Halt, Write, Run”



Dynamic Data Control Properties

Slider 1

Slider Control Settings

Global Symbols

● Dynamic

- ⊞ ADCReadings
- CurrentCapture
- DMASampleNumber
- **Load**
- ⊞ LoadArray
- NewSamplePeriod
- Power
- ⊞ PowerArray
- PowerTraceValue
- PreviousCapture
- Solenoid

Selected Variable: Load

Address: 0x894

Absolute Address

○ Data

Range:

Address:

Data Size: 16 Bits

Data Range: Unsigned

Display Format: Decimal

Upper Limit: 150

Lower Limit: 100

Alternate Label:

Interactive Behavior

☑ Allow Refresh Update

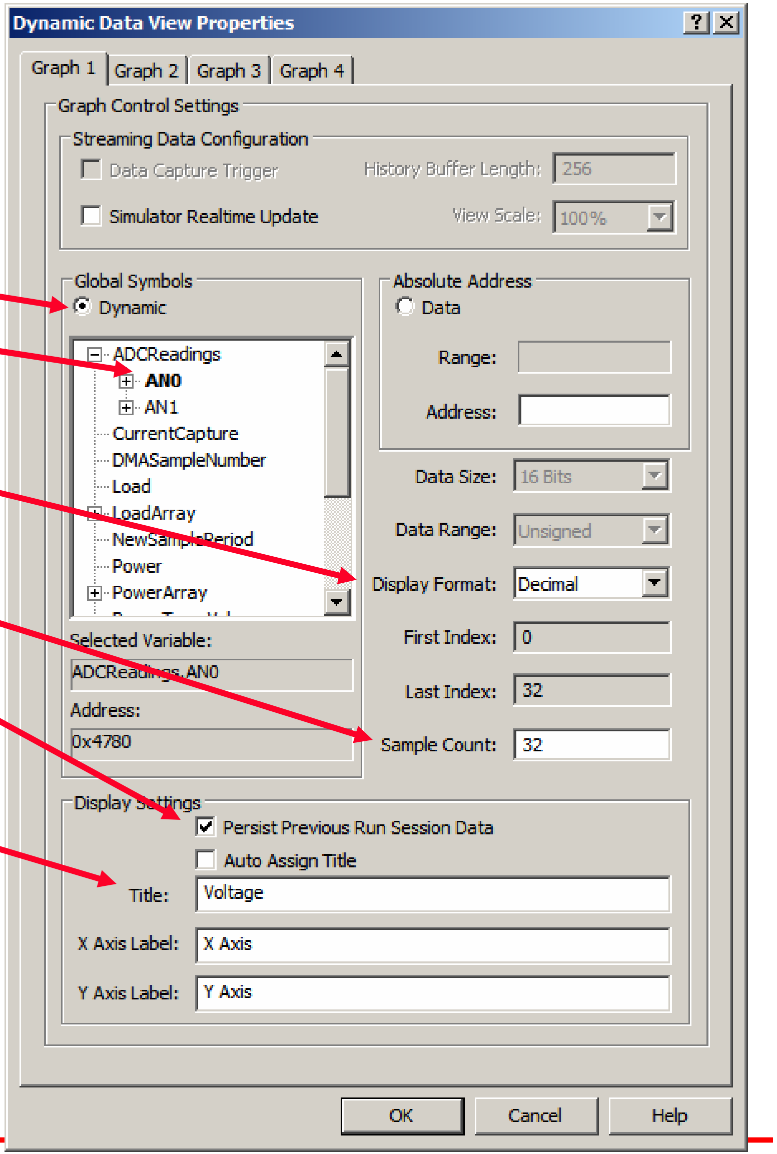
☑ Apply Run-Time Changes Halt, Write, Run

Circuit Breaker
Zero Crossing Lab 3b

- **Verification using DMCI. Graph Setup**
  - Enable the 4 graphs by setting the check box in upper left of each.
  - Right click in the center of the first graph
  - Select the top item “Configure Data Source”
  - Go through each tab, one for each graph and set them up as shown on the next 4 pages

Circuit Breaker
Zero Crossing Lab 3b

- **Graph 1 configuration**
  - Dynamic selection
  - “ADCReadings.AN0” variable
  - Display format “Decimal”
  - Sample count “32”
  - Persist Previous Run data
  - Title “Voltage”



Dynamic Data View Properties

Graph Control Settings

Streaming Data Configuration

□ Data Capture Trigger

History Buffer Length: 256

□ Simulator Realtime Update

View Scale: 100%

Global Symbols

● Dynamic

Selected Variable: ADCReadings.AN0

Address: 0x4780

Absolute Address

○ Data

Range:

Address:

Data Size: 16 Bits

Data Range: Unsigned

Display Format: Decimal

First Index: 0

Last Index: 32

Sample Count: 32

Display Settings

☑ Persist Previous Run Session Data

□ Auto Assign Title

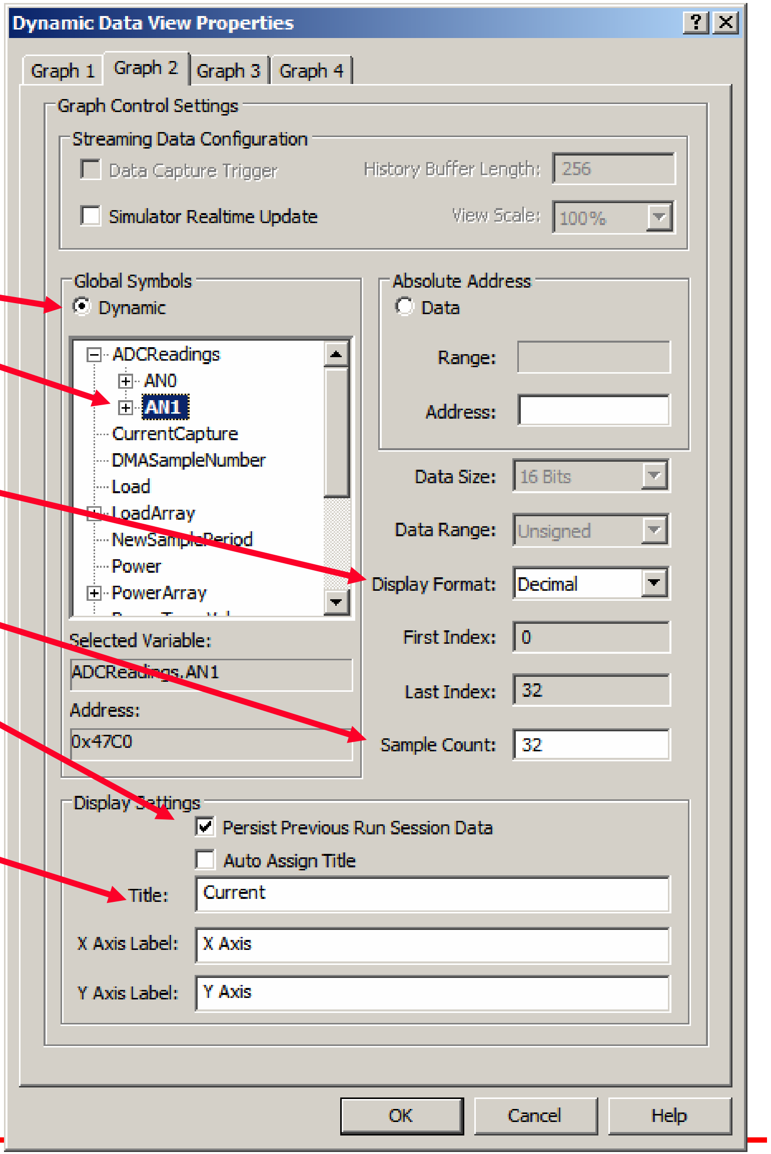
Title: Voltage

X Axis Label: X Axis

Y Axis Label: Y Axis

Circuit Breaker
Zero Crossing Lab 3b

- **Graph 2 configuration**
  - Dynamic selection
  - "ADCReadings.AN1" variable
  - Display format "Decimal"
  - Sample count "32"
  - Persist Previous Run data
  - Title "Current"



Dynamic Data View Properties

Graph 1 | Graph 2 | Graph 3 | Graph 4

Graph Control Settings

Streaming Data Configuration

□ Data Capture Trigger — History Buffer Length: 256

□ Simulator Realtime Update — View Scale: 100%

Global Symbols

● Dynamic

- ADCReadings
  - AN0
  - AN1
- CurrentCapture
- DMASampleNumber
- Load
- LoadArray
- NewSamplePeriod
- Power
- PowerArray

Selected Variable: ADCReadings.AN1

Address: 0x47C0

Absolute Address

○ Data

Range:

Address:

Data Size: 16 Bits

Data Range: Unsigned

Display Format: Decimal

First Index: 0

Last Index: 32

Sample Count: 32

Display Settings

☑ Persist Previous Run Session Data

□ Auto Assign Title

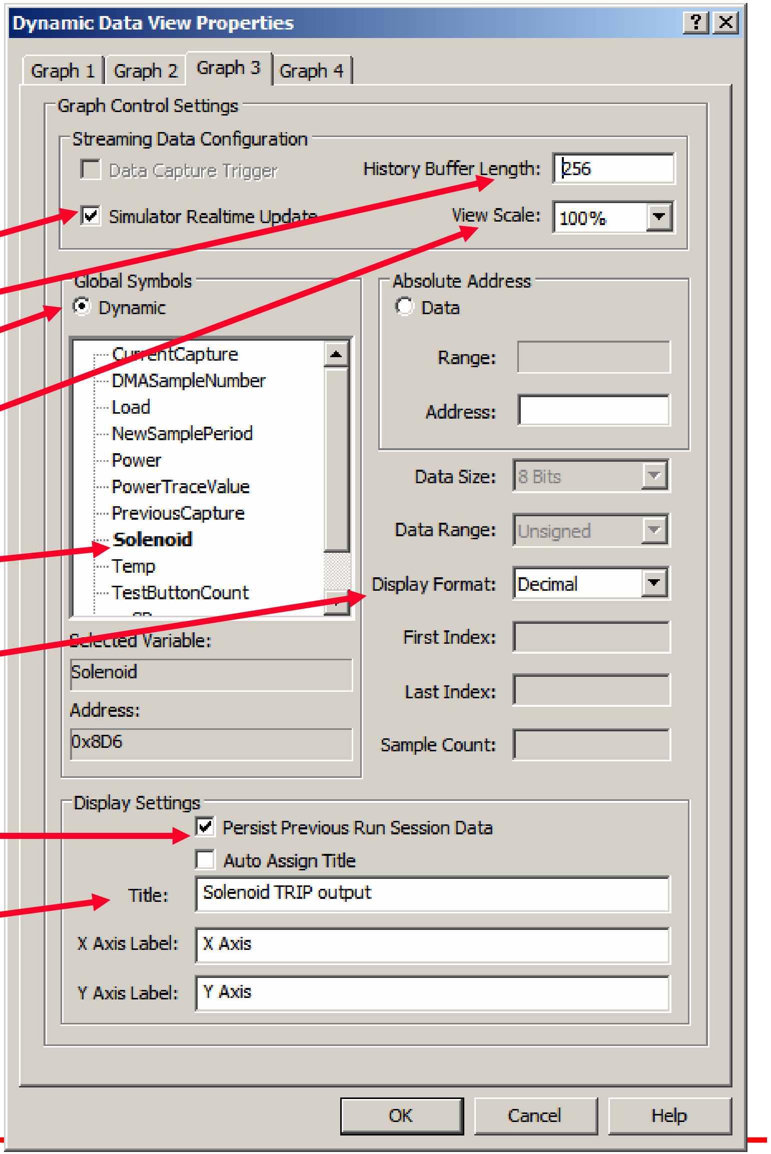
Title: Current

X Axis Label: X Axis

Y Axis Label: Y Axis

Circuit Breaker
Zero Crossing Lab 3b

- **Graph 3 configuration**
  - Simulator Realtime Update
  - History Buffer "256"
  - Dynamic selection
  - View Scale "100%"
  - "Solenoid" variable
  - Display format "Decimal"
  - Persist Previous Run data
  - Title "Solenoid TRIP output"



Dynamic Data View Properties

Graph 1 | Graph 2 | Graph 3 | Graph 4

Graph Control Settings

Streaming Data Configuration

□ Data Capture Trigger

History Buffer Length: 256

☑ Simulator Realtime Update

View Scale: 100%

Global Symbols

● Dynamic

CurrentCapture
DMASampleNumber
Load
NewSamplePeriod
Power
PowerTraceValue
PreviousCapture
Solenoid
Temp
TestButtonCount

Selected Variable: Solenoid

Address: 0x8D6

Absolute Address

○ Data

Range:

Address:

Data Size: 8 Bits

Data Range: Unsigned

Display Format: Decimal

First Index:

Last Index:

Sample Count:

Display Settings

☑ Persist Previous Run Session Data

□ Auto Assign Title

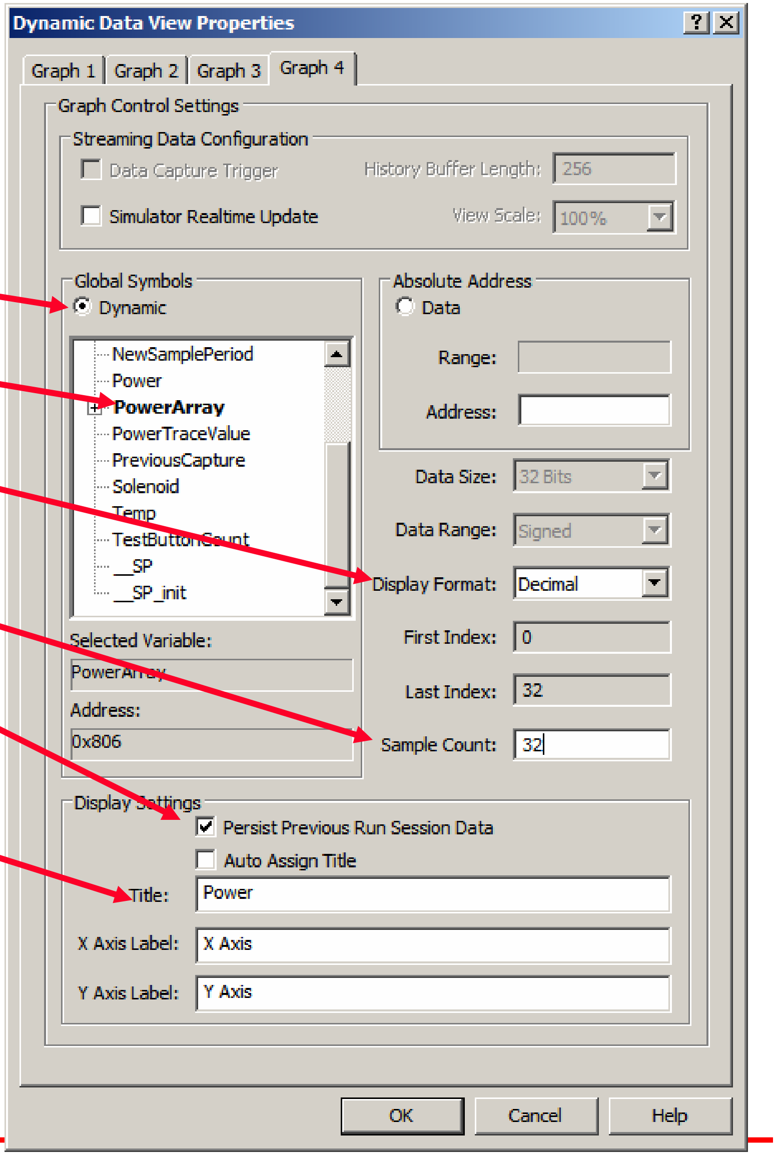
Title: Solenoid TRIP output

X Axis Label: X Axis

Y Axis Label: Y Axis

Circuit Breaker Zero Crossing Lab 3b

- **Graph 4 configuration**
  - Dynamic selection
  - “PowerArray” variable
  - Display format “Decimal”
  - Sample count “32”
  - Persist Previous Run data
  - Title “Power”



Dynamic Data View Properties

Graph 1 | Graph 2 | Graph 3 | Graph 4

Graph Control Settings

Streaming Data Configuration

□ Data Capture Trigger — History Buffer Length: 256

□ Simulator Realtime Update — View Scale: 100%

Global Symbols

● Dynamic

NewSamplePeriod
Power
PowerArray
PowerTraceValue
PreviousCapture
Solenoid
Temp
TestButtonCount
__SP
__SP_init

Selected Variable: PowerArray

Address: 0x806

Absolute Address

○ Data

Range:

Address:

Data Size: 32 Bits

Data Range: Signed

Display Format: Decimal

First Index: 0

Last Index: 32

Sample Count: 32

Display Settings

☑ Persist Previous Run Session Data

□ Auto Assign Title

Title: Power

X Axis Label: X Axis

Y Axis Label: Y Axis

Circuit Breaker
Zero Crossing Lab 3b

- **Testing using the DMCI**
  - Reset the application
  - Start execution
  - Select the slider control button with left mouse button
  - Adjust the slider keeping the mouse button down until you have the desired value.
  - Release the mouse and the selected value will be applied into the Load variable
  - When you raise the value above 116% the trip will occur
  - Set an Asynch stimulus to reset the breaker after you lower the Load percentage

Circuit Breaker
Extended reach **Lab 3b**

- **Verify by tracing data**
  - Focus on Stimulus workbook
  - Select the “Register Trace” tab at the top
  - Enter an optional label if desired
  - Select Register “PowerTraceValue” to monitor
  - Trigger type “PC=” will be the default for data variables
  - Select the label “TracePower” for the PC value
  - Set width to “4” as the variable is a long (4 bytes)
  - Provide the file name to log the data into
  - Select “Dec” for decimal data type
  - Add optional comment

Circuit Breaker
Extended reach Lab 3b

- **Apply updated Stimulus**
  - Select the “Apply” button at the bottom of the stimulus window

- **Testing. Execution**
  - Clear the “Power” value in the watch window
  - Select “Reset” and then “Run”
  - Watch the variables in the watch window. Once the “Power” value has changed, stop the program
  - Select the “Remove” button at the bottom of the stimulus window to allow the trace file to be closed

Circuit Breaker
Extended reach Lab 3b

- **Open the trace data file. “Select All” data within the file and “Copy”**
- **Open the Excel spread sheet “VoltageCurrent.xls”**
- **Highlight an empty cell in a free column next to the highest cell of calculated power values, that you will compare the data to**
- **“Paste” the data. This will fill the column adjacent to the column you are going to compare the data with**
- **Verify at each row (one set of A/D data) that the power traced out, matches the spread sheet calculations**

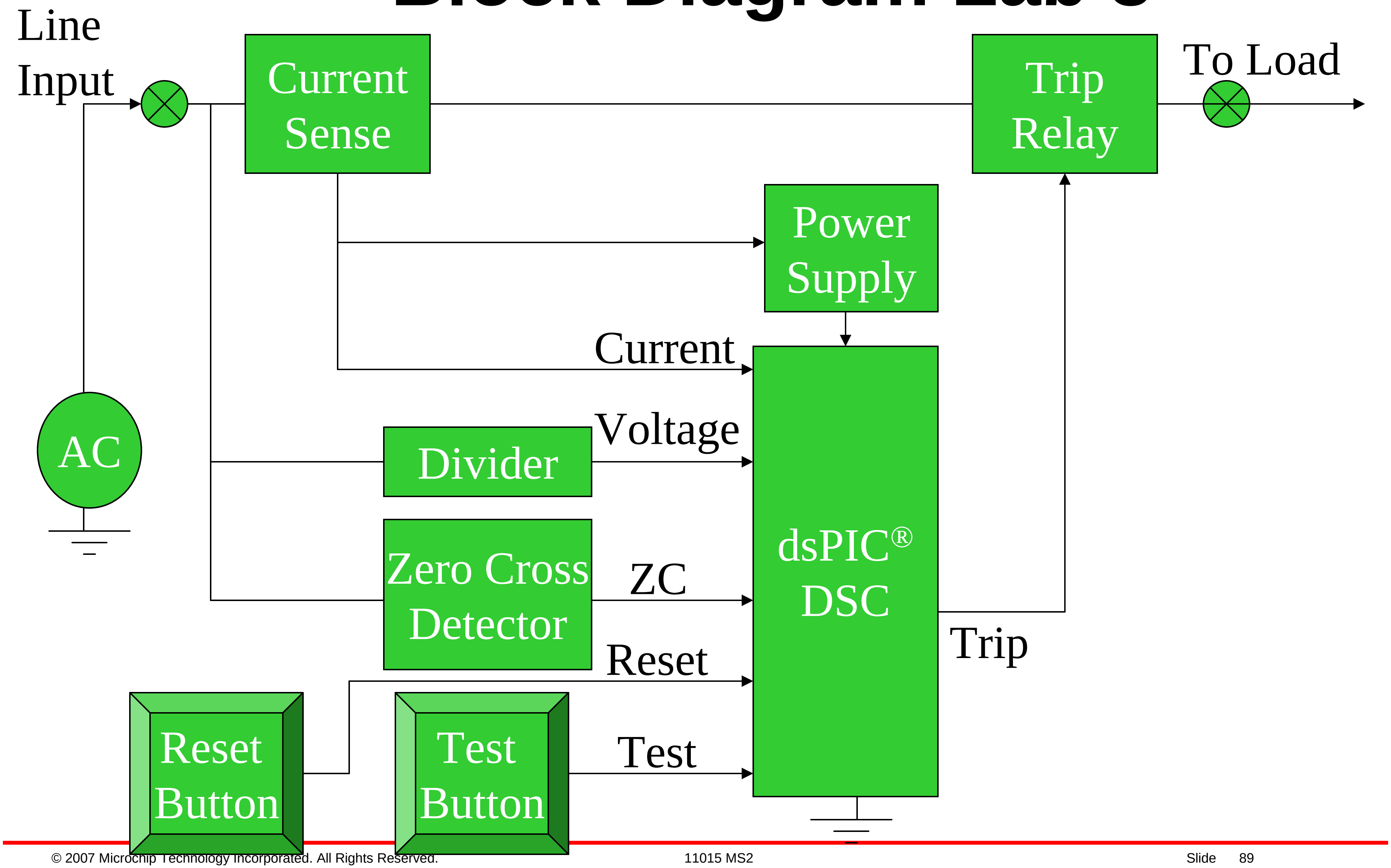
Circuit Breaker *Extended reach* Lab 3b

- **Additional Extra Objective**
  - Create different A/D input files using the different tabs within the EXCEL spread sheet, and test each set of data.
  - Use “Over Current” and “Over Voltage” and check if the “Trip” output is triggered.
  - The “Trip” pin is RD1, shown as “LATD [Bit 1]” in watch window
  - Create asynch button for “Test” (RD3) and test
  - Create asynch button for “Reset” (RD2) and test

Circuit Breaker

- **Following are block diagrams to explain how the application has been designed and how the peripherals are operating within the application**
  - Overall block diagram
  - ZC Input Capture block diagram
  - Timer 3 block diagram
  - ADC block diagram
  - DMAC block diagram

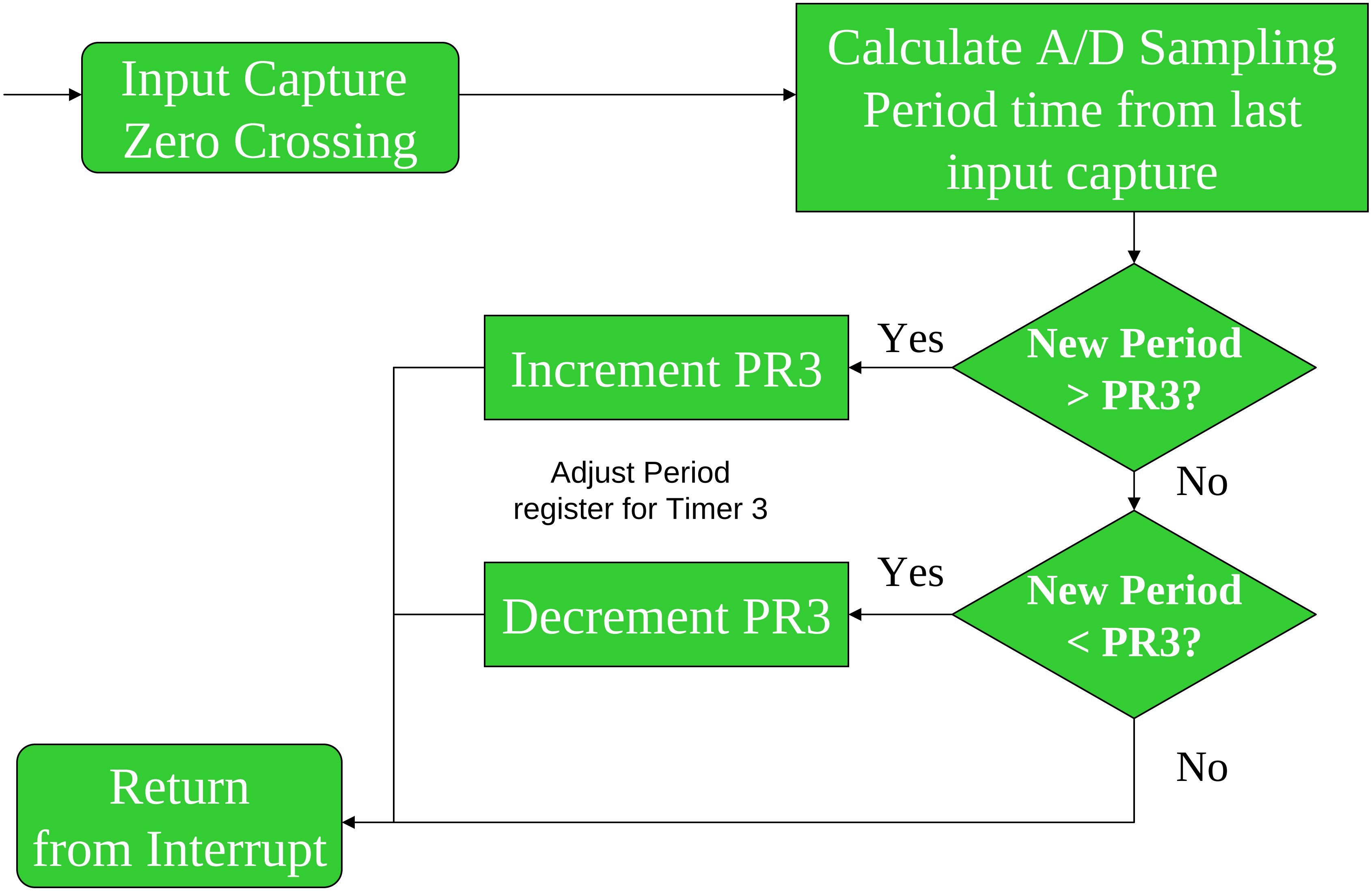
Circuit Breaker
Block Diagram Lab 3



Circuit Breaker Hardware Configuration Lab 3

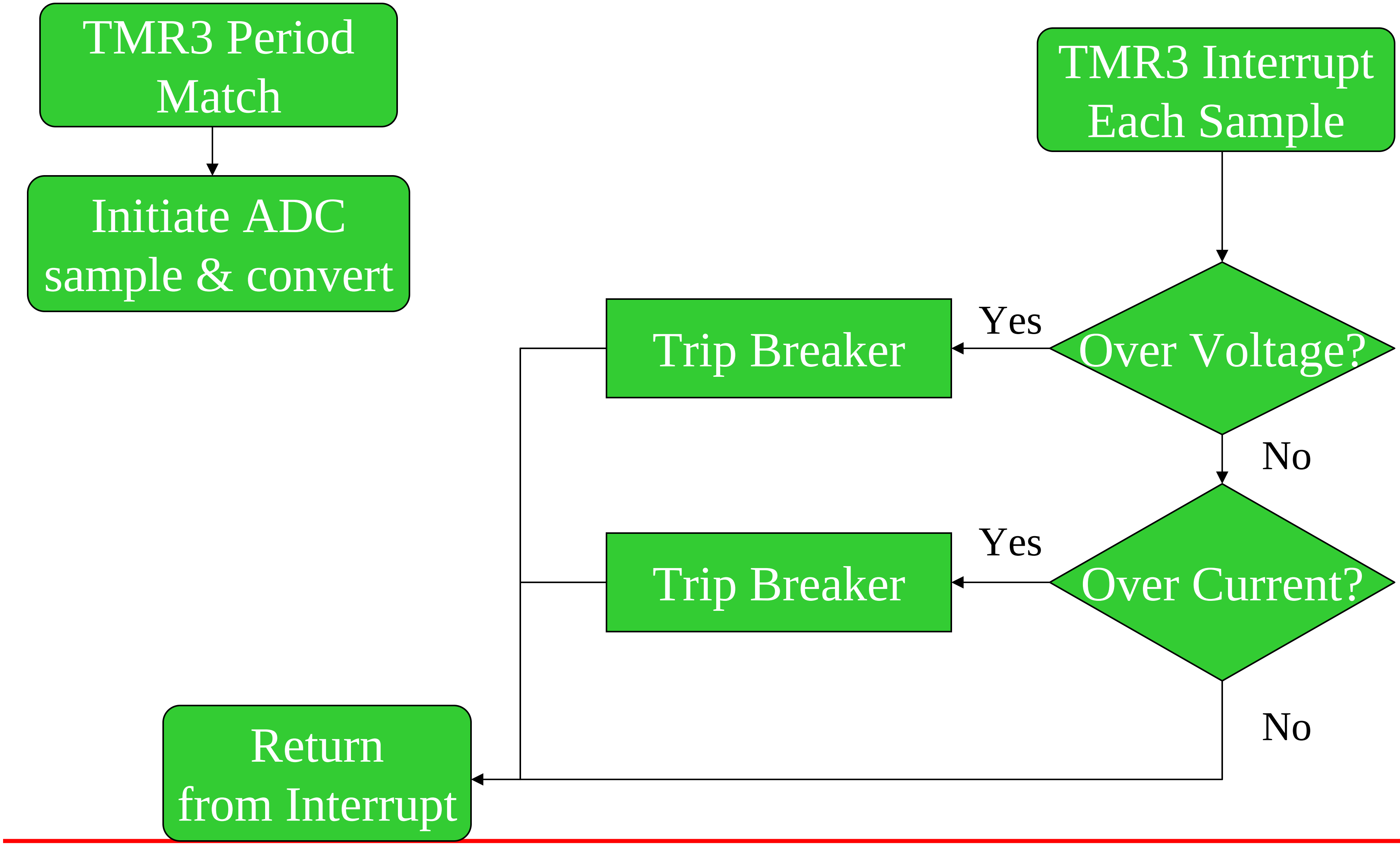
- **Input Capture**
  - AC Zero Crossing voltage triggers IC1 on rising edge
  - Uses TMR2 as time base, free running 16-bit mode period is 1/16 of TMR3 rate, no interrupts
  - IC1 Interrupts firmware to re-calculate A/D sampling period (TMR3 Period value)
  - Maintains phase lock with AC line
- **Timer 3 configuration**
  - Period is set to 1/32 of AC line period
  - Creates 32 identically spaced samples per line cycle
  - Period is adjusted by IC1 to compensate for Phase and line frequency shifts
  - Triggers ADC conversions for both voltage and current

Circuit Breaker
Input Capture Interrupt Lab 3



Adjust Period register for Timer 3

Circuit Breaker
TMR3 Interrupt Lab 3

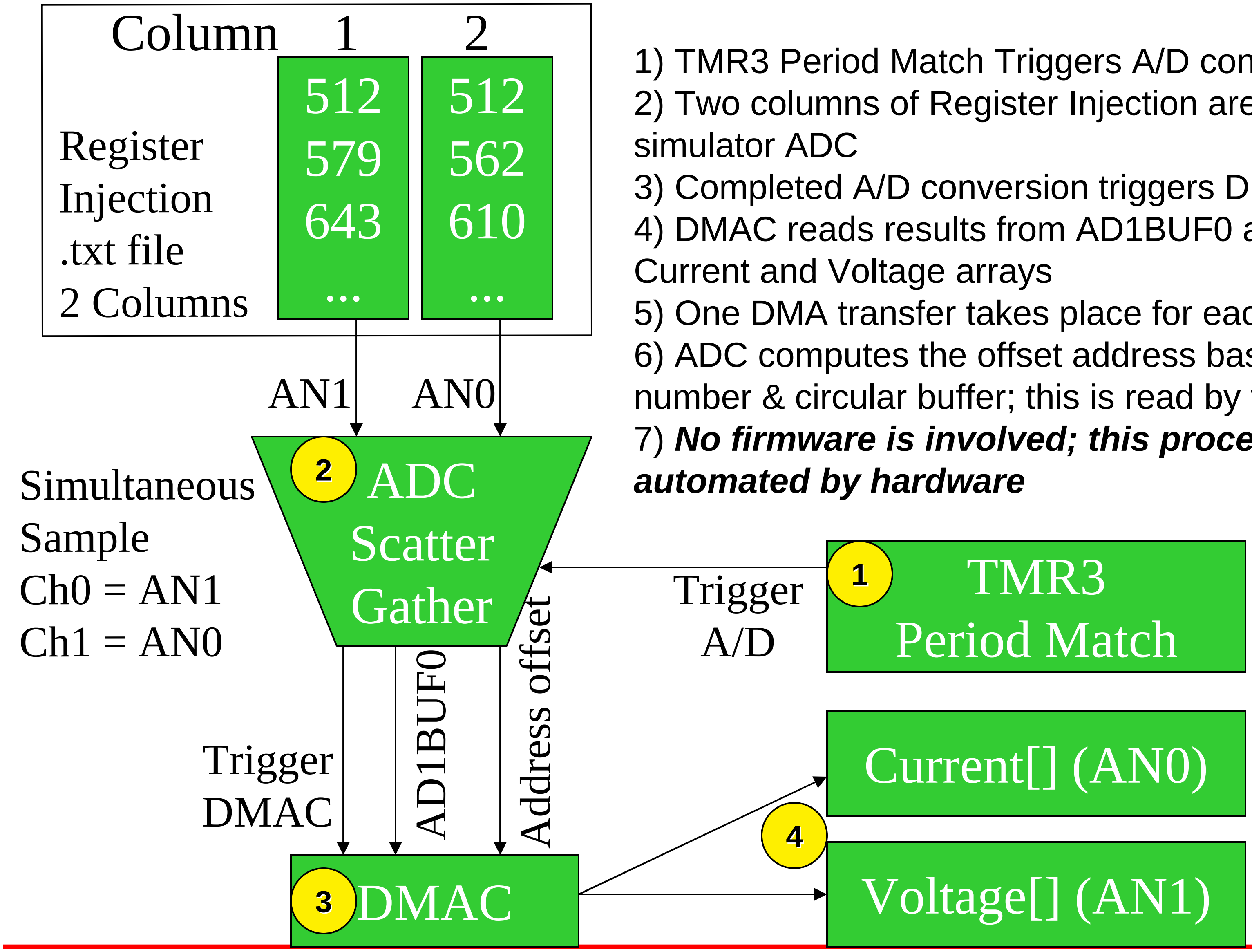


Circuit Breaker Hardware Configuration Lab 3

- **A/D configuration**
  - Simultaneous sampling CH0=AN1 and CH1=AN0
  - Conversion Triggered by TMR3 period match
  - Uses scatter / gather offset address generation for DMAC use, maintaining circular buffer computations
  - Interrupt detected and handled by DMAC hardware
- **DMAC configuration**
  - Services ADC conversion completion
  - Computes final destination address for ADC results
  - Moves data from AD1BUF0 to either Current[] or Voltage[] dual port RAM array
  - Interrupts firmware when both arrays are completely full with 32 A/D samples (64 transfers)

Circuit Breaker Simulator ADC Lab 3

1) TMR3 Period Match Triggers A/D conversion
2) Two columns of Register Injection are read by the simulator ADC
3) Completed A/D conversion triggers DMAC
4) DMAC reads results from AD1BUF0 and writes them to Current and Voltage arrays
5) One DMA transfer takes place for each conversion
6) ADC computes the offset address based on channel number & circular buffer; this is read by the DMAC
7) ***No firmware is involved; this process is completely automated by hardware***



Register Injection .txt file 2 Columns

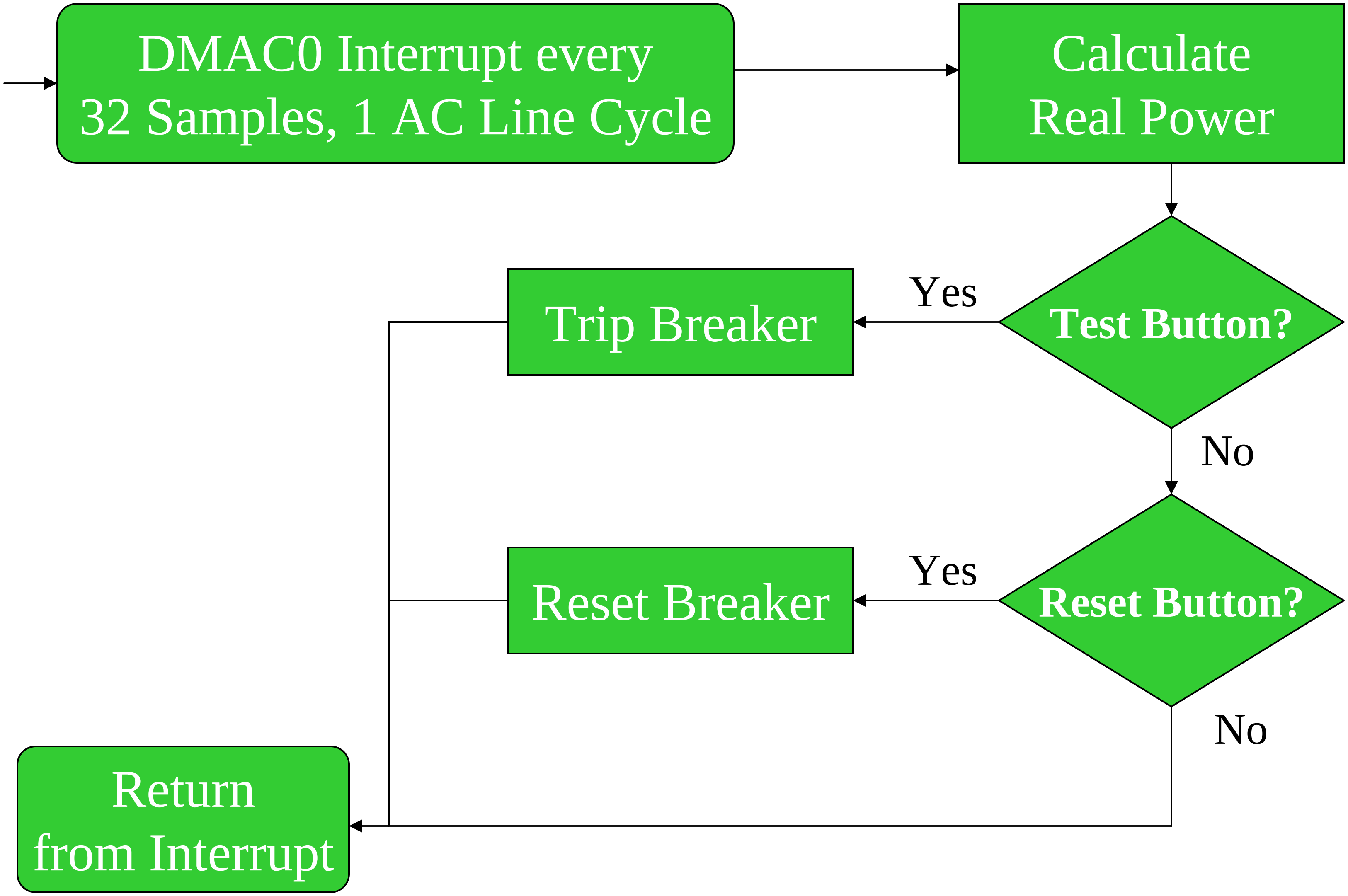
Column	1	2
	512	512
	579	562
	643	610
	…	…

Simultaneous Sample
Ch0 = AN1
Ch1 = AN0

Arrays in dual ported RAM can be accessed by user code at any time

UNIVERSITY OF MICROCHIP
U OF M
MASTERs 2007

Circuit Breaker DMAC Interrupt Lab 3



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.