Chapter 1. Writing a Renegade-i™ Test Pattern

A renegade test pattern defines the stimulus that will drive the Device Under Test (DUT), the pattern should be designed so that the DUT will produce a specific output. The pattern also contains definitions of expected output from the DUT. The pattern is composed of vectors and instructions that will be executed by the tester to generate the desired stimulus for the DUT.

Vector Symbols

0 : drive a logic low to a DUT pin
1 : drive a logic high to a DUT pin
X : do not drive a logic level to the DUT pin (tristate)
L : expect a logic low output from the DUT
H : expect a logic high output from the DUT

Examples:

```
@ VEC 0X0LHL
@VEC 1L0HLL
@ VEC 0L0LHL
@VEC 1H1LLH
```

VECTORMAP

The vectormap defines which vector-symbol goes to which DUT pin. The vectormap contains a list of pin names, the order in the list should match the position of the vector symbols so that the pattern will work properly. A vectormap starts with an '@' then followed by the keyword VECTORMAP.

Example:

```
@VECTORMAP( clk, a_out, d_in, b_out, c_out, e_out).

@VEC 0X0LHL.
@ VEC 1L0HLL.
@VEC 0L0LHL.
@ VEC 1H1LLH.
```

In this example, the DUT have 6 I/O pins named clk, a_out, d_in, b_out, c_out, e_out & f_out. The pins clk & d_in are input pins that should be driven, and the remaining pins are output pins that should not be driven.

The vector symbols in the first column will drive the clk pin,
the vector symbols in the second column will produce fail results if it does not match the output of a_out,
the vector symbols in the third column will drive the d_in pin,
the vector symbols in the fourth column will produce fail results if it does not match the output of b_out,
the vector symbols in the fifth column will produce fail results if it does not match the output of c_out,
the vector symbols in the sixth column will produce fail results if it does not match the output of e_out.

A vectormap should be placed early in the pattern before any vectorline. Multiple vectormaps can be placed anywhere in the pattern if re-arrangement of the vector symbol is desired:

```
@VECTORMAP( clk, a_out, d_in, b_out, c_out, e_out)
@VEC 0X0LHL
@VECTORMAP( a_out, b_out, c_out, e_out, d_in, clk)
@VEC LHLL10
@VECTORMAP( clk, a_out, d_in, b_out, c_out, e_out)
@VEC 0L0LHL
@VEC 1H1LLH
```

VECTORLINE ™

Vectorlines defines the vectors and instructions that will be executed at each pattern cycle. A vectorline always start with the symbol '@' and followed by one or more statements, an optional label can be included at the beginning of the vectorline. Example:

```
@myLabel:  <statement1>. <statement2>. <statement3>. <statement4>. <statement5>.
@next:
            <statement1>. <statement2>. <statement3>.
            <statement4>.
@            <statement1>. <statement2>. <statement3>.
@            <statement1>. <statement2>. <statement3>.
```

Each statement is separated by a period '.' and are case sensitive. Valid statements at the moment are:

```
VEC 01XLH
TSETnn
FUNCSETnn
REPEAT <integer value>
LOAD COUNTERnn <integer value>
INCR COUNTERnn
DECR COUNTERnn
SELECT COUNTERnn
[IF <CONDITIONAL> ] JUMPTO <label>
[IF <CONDITIONAL> ] CALL <label>
[IF <CONDITIONAL> ] RETURN
```

Not yet available: CLEARFAIL, CLEARFLAGnn, SETFLAGnn, CLEARUBITnn, SETUBITnn

Not yet available: GLOBAL <label> (allows a pattern to access a label declared in a different pattern file)
(also allows the test program to begin/stop execution at GLOBAL labels)

Where nn represents an integer, and the items inside the [ ] brackets are optional.

Example:

```
@yourLabel: VEC 01010X. TSET0. FUNCSET0.
@           VEC 10101H. TSET1. FUNCSET5. JUMPTO myLabel.
@           VEC 1110XX. TSET9. FUNCSET0. LOAD COUNTER2 1.
@           VEC 1111LL. TSET1. FUNCSET1. DECR COUNTER2. IF COUNTISZERO CALL subr444
@           VEC 0000HL. TSET5. FUNCSET9. SELECT COUNTER8. IF COUNTNOTZERO RETURN.
```

At the moment, pattern execution will begin at the first vectorline and will only stop when the last vectorline is reached.

The VEC statement

Valid Examples:

```
VEC 10HLX.
VEC 10 HLX .
VEC 10 H L X  .
VEC 1 0 H L X.
VEC 10HLX.
```

Invalid Examples:

```
VEC10HLX.
VEC 1,0,H,L,X.
VEC (10) H (L X).
Vec 1 0 H L X.
vec 10HLX.
```

The TSETnn statement

Examples:

```
TSET0.
TSET1.
TSET2.
TSET3.
---
---
TSET15.
```

TSET is discussed further in Chapter 3.

The FUNCSETnn statement

Examples:

```
FUNCSET0.
FUNCSET1.
FUNCSET2.
FUNCSET3.
---
---
FUNCSET31.
```

FUNCSET is discussed further in Chapter 2.

The REPEAT statement

Examples:

```
REPEAT 2.
REPEAT 3. INCR COUNTER6. //cool!
------
REPEAT 67108865.
REPEAT 0. is invalid
REPEAT 1. Is invalid
```

The repeat statement repeats the current vectorline nn times, where nn is the specified integer.

The LOAD statement

Example:

```
LOAD COUNTER1 0.
LOAD COUNTER2 1.
LOAD COUNTER3 2.
-----
-----
LOAD COUNTER31 67108863. //maximum value that “LOAD” statement can handle
```

There are 32 counters named COUNTER0 ….. COUNTER31. These are 32-bit counters, count from 0 to 4294967295

The INCR , DECR and SELECT statements

Example:

```
DECR COUNTER3.
INCR COUNTER11.
SELECT COUNTER9.
```

The DECR statement subtracts 1 from the specified counter, if decremented from zero, it will wrap around to 4294967295. INCR adds 1 to the specified counter, it wraps around to zero when 4294967295 is exceeded. The INCR and DECR statement also makes the specified counter available for the CONDITIONAL statement. The SELECT statement holds the specified counter to its current value, but it is made available for the CONDITIONAL statement.

The JUMPTO statement

Examples:

```
JUMPTO label555.
IF <CONDITIONAL> JUMPTO label555.
```

The JUMPTO statement transfers control to the vectorline in the specified label.

The CALL statement

Examples:

```
CALL subr555.
IF <CONDITIONAL> CALL subr555.
```

The CALL statement transfers control to the vectorline in the specified label, and pushes the return address into the stack. CALLs can be nested up to 8, exceeding the limit of 8 will produce unpredictable results.

The RETURN statement

Examples:

```
RETURN.
IF <CONDITIONAL> RETURN.
```

The RETURN statement transfers control to the vectorline following the recent CALL statement. If a RETURN statement was executed without a prior CALL statement, unpredictable results will happen.

The CONDITIONAL statements

Valid CONDITIONALs at the moment are:

COUNTNOTZERO
COUNTISZERO

Not yet available: PASSED, FAILED, FLAGnn, NOTFLAGnn, TIMEDOUT, NOTTIMEDOUT

COUNTNOTZERO executes the accompanied JUMPTO/CALL/RETURN statement if the "current" value of the specified counter is not zero.
COUNTISZERO executes the accompanied JUMPTO/CALL/RETURN statement if the "current" value of the specified counter is zero.

"current" value means the contents of the counter before the INCR/DERC/LOAD operation

Example1:

```
@LOAD COUNTER5 0.
@SELECT COUNTER5. IF COUNTISZERO JUMPTO exit. //the condition is true, jump is made.
```

Example2:

```
@forever: LOAD COUNTER5 2.
@DECR COUNTER5. IF COUNTISZERO JUMPTO exit. //current value is 2, jump not made.
@DECR COUNTER5. IF COUNTISZERO JUMPTO exit. //current value is 1, jump not made.
@DECR COUNTER5. IF COUNTISZERO JUMPTO exit. //current value is 0, jump is made.
@JUMPTO forever. //this vectorline is never reached
@exit:
 SELECT COUNTER5. IF COUNTISZERO JUMPTO forever. //current value is 67108863, no jump
```

Inserting COMMENTS to the pattern file

Comments can be inserted anywhere in the pattern file. There are two ways to insert comments:
// way
/* way */

When the "//" is encountered, all texts up to the linefeed will be ignored .
When the "/*" is encountered, all texts up to the matching "*/" will be ignored.

Examples:

```
@forever: VEC 1111XXXX. TSET3. FUNCSET5. //JUMPTO forever. Loopback was commented out
@VEC 1111 /*this is a comment*/ XXXX. TSET3. FUNCSET5.
@VEC 1111XXXX. TSET3. FU/*try for yourself*/NCSET5.
@VEC 1111XXXX. TSET3. FUNCSET5.
/*
@VEC 1111XXXX. TSET3. FUNCSET5.
@VEC 1111XXXX. TSET3. FUNCSET5. //multiline comments
@VEC 1111XXXX. TSET3. FUNCSET5.
*/
@VEC 1111XXXX. /* TSET3. FUNCSET5.
@VEC 1111XXXX. TSET3. FUNCSET5.
@VEC 1111XXXX. TSET3. FUNCSET5.
@VEC 1111XXXX. */ TSET3. FUNCSET5.
```

Mini-APG statements set

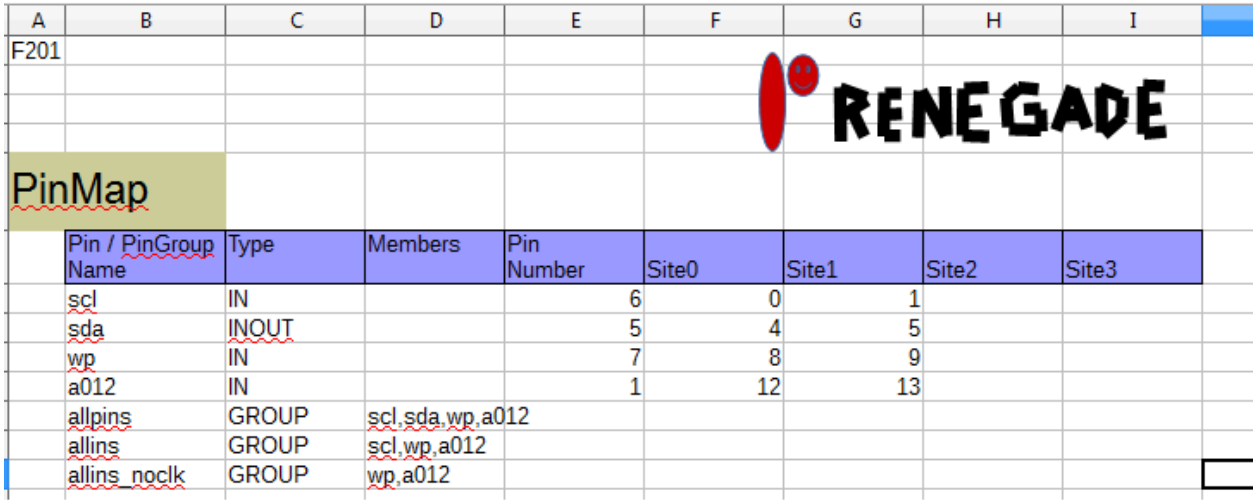
Renegade-i™ includes a mini APG, just enough to test a simple memory device. The mAPG have the following:

10-bit Y address register
10-bit X address register
10-bit data register
That's all for the moment

To operate the mAPG resources, the following statements were included:

```
CLEAR DATREG.
CLEAR YMAIN.
CLEAR XMAIN.
INCR DATREG.
INCR YMAIN.
INCR XMAIN.
DECR DATREG.
DECR YMAIN.
DECR XMAIN.
LOAD DATREG nn.
LOAD YMAIN nn.
LOAD XMAIN nn.
DATA DRIVE. //drive the contents of data register to the pins routed to the data register
DATA TRISTATE. //tristate the pins routed to the data register
DATA READ. // tristate the pins routed to the data register and produce a fail if data don't match
DATA DRIVE XORBCK. //same as DATA DRIVE, but inverts the data when LSB of Y-addr & X-addr not same
DATA READ XORBCK. //same as DATA READ, but inverts the data when LSB of Y-addr & X-addr not same
ABUFF SHIFT. //(for serial pin capture) shifts-in the value of the assigned pin to the 16-bit serial register
ABUFF STORE. //store the contents of the serial register into the next capture memory location, 256 deep
ABUFF SHIFTNSTORE. //same as SHIFT followed by STORE
```

Chapter 2. The PinMap Worksheet



F201

RENEGADE

PinMap

Pin / PinGroup Name	Type	Members	Pin Number	Site0	Site1	Site2	Site3
scl	IN		6	0	1		
sda	INOUT		5	4	5		
wp	IN		7	8	9		
a012	IN		1	12	13		
allpins	GROUP	scl,sda,wp,a012					
allins	GROUP	scl,wp,a012					
allins_noclk	GROUP	wp,a012					

Shown above is an example of a simple PinMap. Column B lists the names of the DUT's pins (or a PinGroup name). Column C describes the type of the given name, valid entries are (uppercase) IN, OUT and INOUT or GROUP.

A type "OUT" pin is an output-only pin from the DUT, the tester channels associated to this pin type will be tristated while the test program is loaded. A program is considered loaded after a successful "Verify".

A type "IN" pin is an input-only pin of the DUT. The tester channels associated to this pin type will be driven while the test program is loaded.

A type "INOUT" pin is bidirectional, the tester channels associated to this pin type will be driven or tristated according to the vector symbols in the running test pattern, the last executed vector symbol will determine the resting condition of this pin.

A type "GROUP" is not a pin, it is a collection of OUT/IN/INOUT pins.

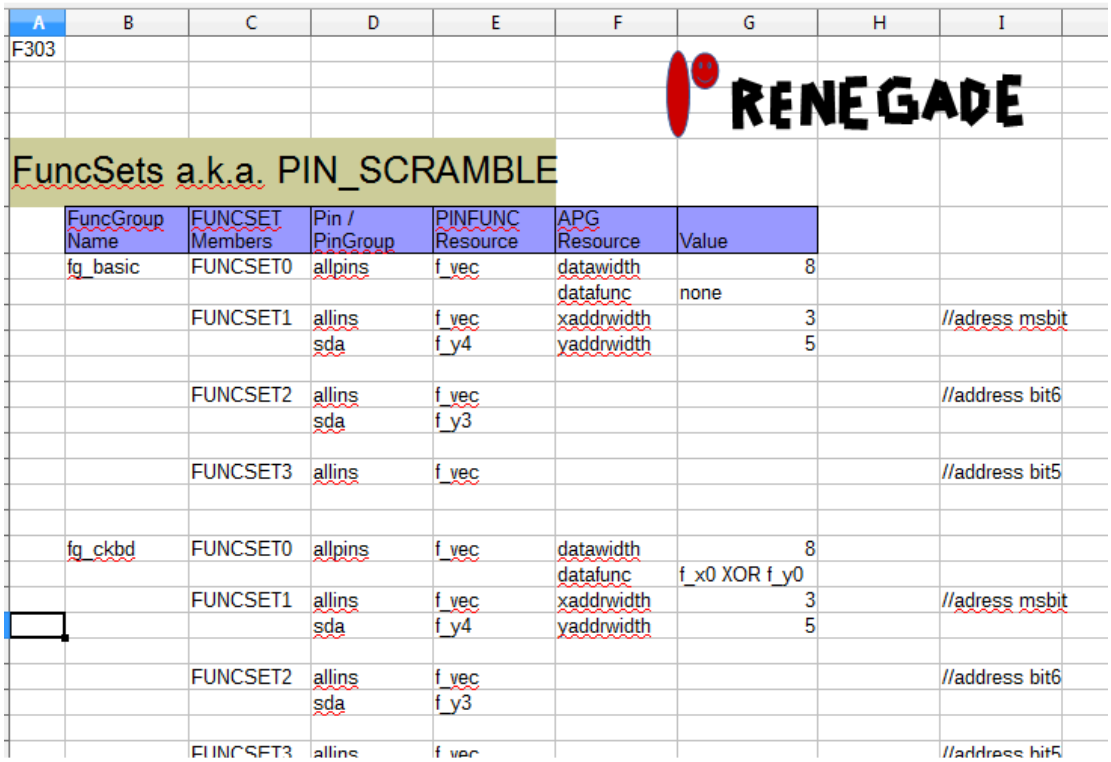
Column D declares the members of a PinGroup. The entries in this column will be ignored if the type in column C is not "GROUP".

Column E is the Pin number of the DUT's pin being described. In this example, the DUT comes in an 8-pin PDIP package, and the numbers in this column correspond to the numbers as specified in the datasheet. This column will be ignored if the type in column C is "GROUP".

Columns F, G, H & I declares the tester channel connected to the described pin name. These columns will be ignored if the type in column C is "GROUP". A blank entry means the DUT associated to that Site is not used. Column E for Site0 are the tester channels used for the first DUT, column F for the second DUT.

It is very important that the entries in PinMap Worksheet matches the wirings in the loadboard (the board that is used to plug in the DUTs).

Chapter 2. The FuncSets Worksheet



F303

FuncSets a.k.a. PIN_SCRAMBLE

FuncGroup Name	FUNCSET Members	Pin / PinGroup	PINFUNC Resource	APG Resource	Value	
fg_basic	FUNCSET0	allpins	f_vec	datawidth	8	
				datafunc	none	
	FUNCSET1	allins	f_vec	xaddrwidth	3	//adress msbit
		sda	f_y4	yaddrwidth	5	
	FUNCSET2	allins	f_vec			//address bit6
		sda	f_y3			
	FUNCSET3	allins	f_vec			//address bit5
fg_ckbd	FUNCSET0	allpins	f_vec	datawidth	8	
				datafunc	f_x0 XOR f_y0	
	FUNCSET1	allins	f_vec	xaddrwidth	3	//adress msbit
		sda	f_y4	yaddrwidth	5	
	FUNCSET2	allins	f_vec			//address bit6
		sda	f_y3			
	FUNCSET3	allins	f_vec			//address bit5

The FuncSets Worksheet describes how the tester channels are routed into the tester resources. Tester resources are described in next page.

Column B in the FuncSets Worksheet specifies the name of a FUNCSET group. Any group should contain all the FUNCSETs that is being used by the running patterns. For example, if your patterns are using FUNCSET0 & FUNCSET1 only, then it is safe to declare these two funcsets only, FUNCSET2 … FUNCSET31 can be left undefined. If the running pattern used a FUNCSET that was not defined in the current active group, then unpredictable results will happen.
Only one FUNCSET group can be active at a time, the group should be activated before running a Test Method as entered in the TestFlow Worksheet.

Column C lists the members of the FUNCSET group named in column B. This list continues until the next groupname entered in column B.

Column D contains the pin name (or PinGroup name) being routed to the tester resource define in column E.

Column E defines the tester resource routed to the tester channel (or channels) asscociated to the name specified in column D. See next page for a list of available tester resource.

Column F is an mAPG resource parameter, and Column G is the parameter setting:

datawidth	specifies the width of the data register
datafunc	specifies the background function to operate the data inversion logic
xaddrwidth	specifies the width of the X address register
yaddrwidth	specifies the width of the Y address register

Valid tester resources are:

f_y9	bit 9 of the Y address register
f_y8	bit 8 of the Y address register

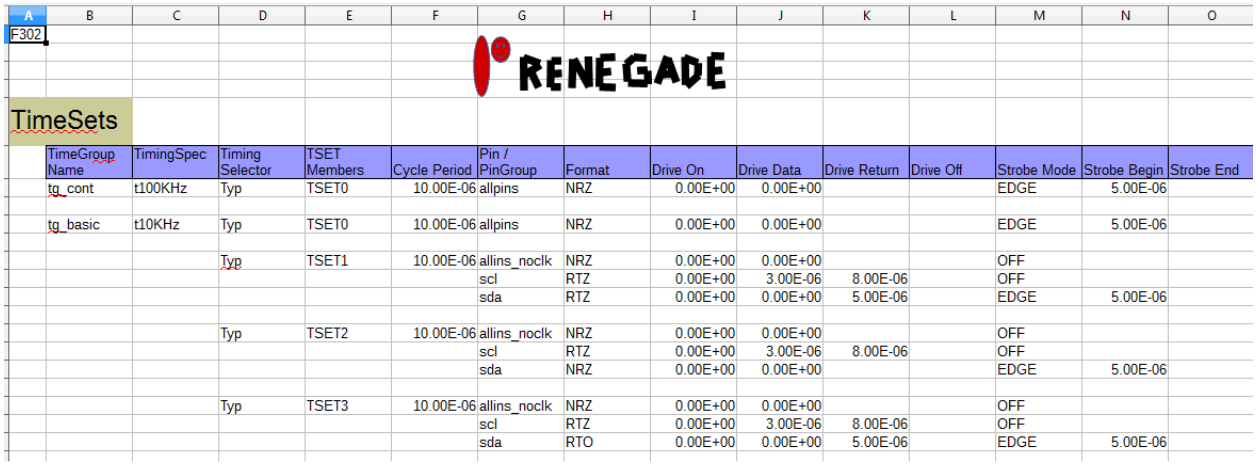
--	
f_y0	bit 0 of the Y address register
f_x9	bit 9 of the X address register
f_x8	bit 8 of the X address register

--	
f_x0	bit 0 of the X address register
f_d9	bit 9 of the data register
f_d8	bit 8 of the data register

--	
f_d0	bit 0 of the data register
f_vec	the vector symbol in the corresponding column in the VECTORMAP

The pattern is allowed to have different FUNCSETnn per vectorline.

Chapter 3. The TimeSets Worksheet



TimeSets

TimeGroup Name	TimingSpec	Timing Selector	TSET Members	Cycle Period	Pin / PinGroup	Format	Drive On	Drive Data	Drive Return	Drive Off	Strobe Mode	Strobe Begin	Strobe End
tg_cont	t100KHz	Typ	TSET0	10.00E-06	allpins	NRZ	0.00E+00	0.00E+00			EDGE	5.00E-06	
tg_basic	t10KHz	Typ	TSET0	10.00E-06	allpins	NRZ	0.00E+00	0.00E+00			EDGE	5.00E-06	
		Typ	TSET1	10.00E-06	allins_noclk	NRZ	0.00E+00	0.00E+00			OFF		
					scl	RTZ	0.00E+00	3.00E-06	8.00E-06		OFF		
					sda	RTZ	0.00E+00	0.00E+00	5.00E-06		EDGE	5.00E-06	
		Typ	TSET2	10.00E-06	allins_noclk	NRZ	0.00E+00	0.00E+00			OFF		
					scl	RTZ	0.00E+00	3.00E-06	8.00E-06		OFF		
					sda	NRZ	0.00E+00	0.00E+00			EDGE	5.00E-06	
		Typ	TSET3	10.00E-06	allins_noclk	NRZ	0.00E+00	0.00E+00			OFF		
					scl	RTZ	0.00E+00	3.00E-06	8.00E-06		OFF		
					sda	RTO	0.00E+00	0.00E+00	5.00E-06		EDGE	5.00E-06	

Column B (TimeGroup) is the name of a TSET group.
Only one TSET group can be active at a time as entered in the TestFlow Worksheet.

Column C & Column D are not yet implemented.

Column E (TSET members) lists the members of a TSET group in column B. The list continues until the next entry in column B. Valid members are TSET0 …. TSET15. All TSETs used in the running pattern should be a member of the current active TSET group, or unpredictable result will happen.

Column F (Cycle Period) is the cycle period in seconds. To make simple entries, you can use “=10*us” in place of “10.0e-6” sec which is equivalent to 10 microsec. Acceptable cycle periods are up to 1.2 millisec only.

Column G (Pin/PinGroup) is the name of a pin or a PinGroup that will receive the timing definition.

Column H (Format) describes the encoding format, valid formats are NRZ, RTZ and RTO.

Column I (Drive On) is the edge position where the pin drivers are turned on. It is recommended not to leave this blank.

Column J (Drive Data) is the edge position where the channel start driving the corresponding high or low.

Column K (Drive Return) is the edge position where the channel returns to its resting point. This edge definition has a meaming only if the format is RTZ or RTO, meaningless for NRZ.

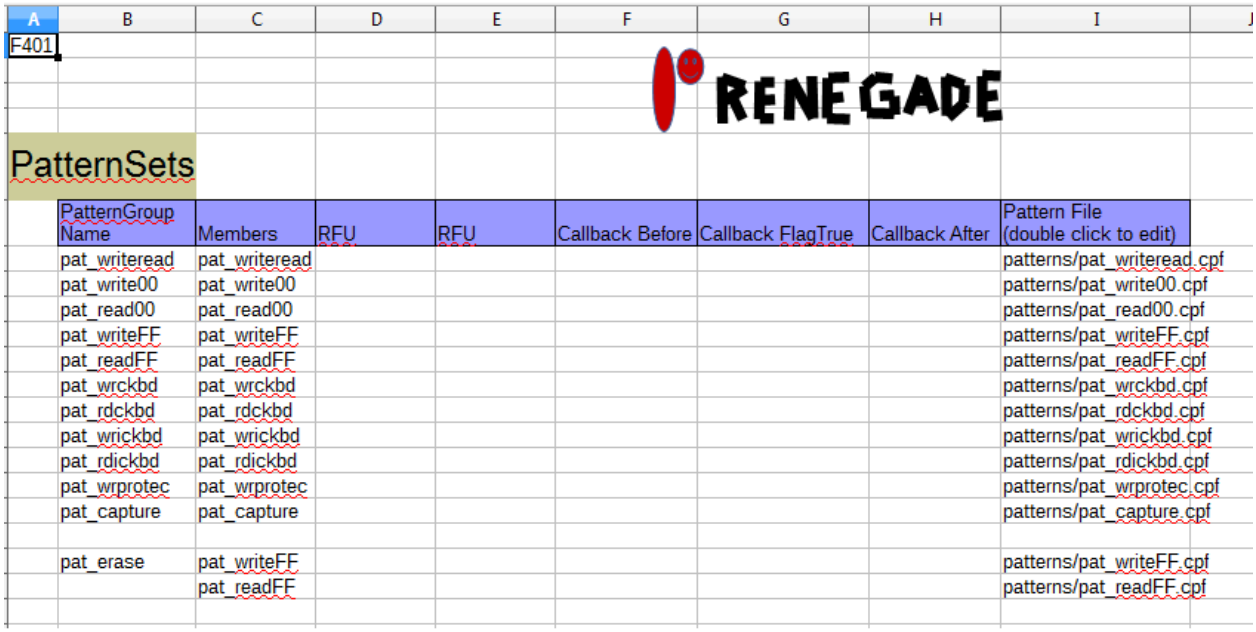
Column L (Drive Off) is the edge position where the channel turns off the drivers. This column is left blank for most application, the drivers automatically turns off the drivers when X, L, or H is encountered in the vector symbols.

Column M (Strobe Mode) is the strobe mode. EDGE, WINDOW, or OFF. At the moment this column is ignored. Defaults to EDGE.

Column N (Strobe Begin) is the strobe position. The DUT’s outputs are compared to expected values at this point in time.

Column O (Strobe End) is ignored at the moment.

Chapter 4. The PatternSets Worksheet



F401

RENEGADE

PatternSets

PatternGroup Name	Members	RFU	RFU	Callback Before	Callback FlagTrue	Callback After	Pattern File (double click to edit)
pat_writeread	pat_writeread						patterns/pat_writeread.cpf
pat_write00	pat_write00						patterns/pat_write00.cpf
pat_read00	pat_read00						patterns/pat_read00.cpf
pat_writeFF	pat_writeFF						patterns/pat_writeFF.cpf
pat_readFF	pat_readFF						patterns/pat_readFF.cpf
pat_wrckbd	pat_wrckbd						patterns/pat_wrckbd.cpf
pat_rdckbd	pat_rdckbd						patterns/pat_rdckbd.cpf
pat_wrickbd	pat_wrickbd						patterns/pat_wrickbd.cpf
pat_rdickbd	pat_rdickbd						patterns/pat_rdickbd.cpf
pat_wrprotec	pat_wrprotec						patterns/pat_wrprotec.cpf
pat_capture	pat_capture						patterns/pat_capture.cpf
pat_erase	pat_writeFF						patterns/pat_writeFF.cpf
	pat_readFF						patterns/pat_readFF.cpf

The PatternSets Worksheet assign names to the pattern that is used in the Test Methods. The pattern names can be collected into a Pattern Group for convenience.

Column B (PatternGroup Name) assign the group name to the list of members in column C.

Column C (Members) is the list of member patterns to a group defined in column B. This list continues until the next group name entered in column B.

Column D & E are not yet available. It is reserved for Start/Stop labels to be implemented in the future.

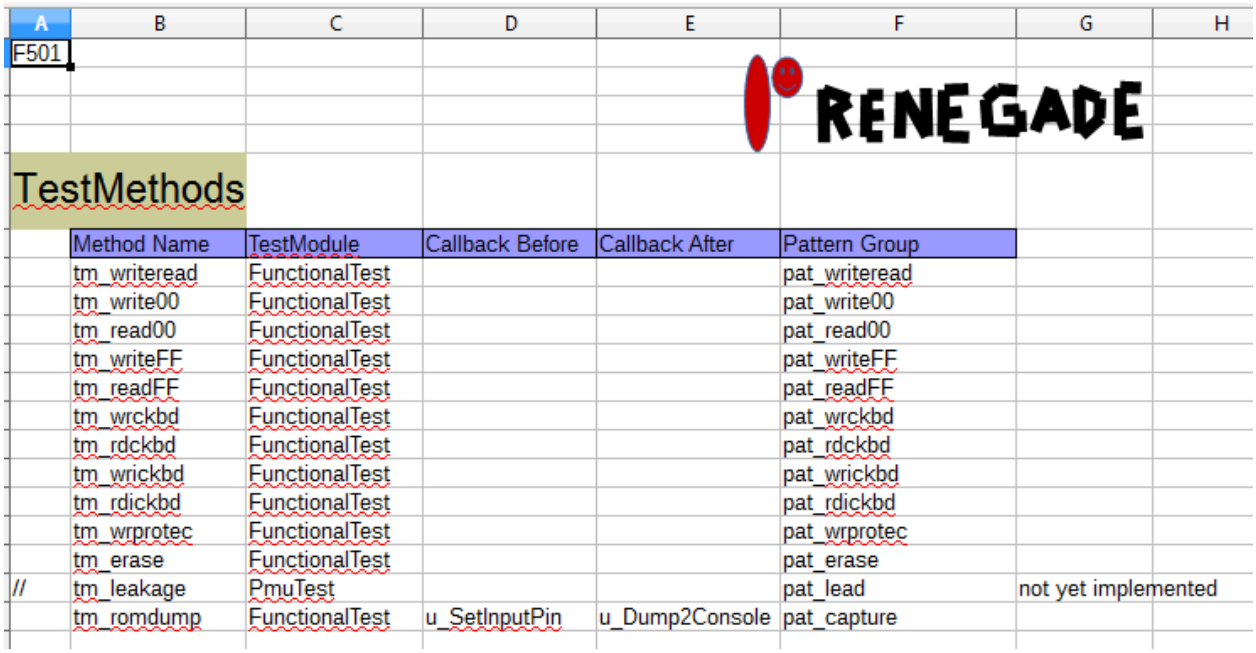
Column F (Callback Before) could be used to enter the name of a “Basic” macro that will be executed before running the member pattern.

Column G (Callback FlagTrue) could be used to enter the name of a “Basic” macro that will be executed when a pattern flag toggled to TRUE while the member pattern is running.

Column H (Callback After) could be used to enter the name of a “Basic” macro that will be executed after the member pattern has stopped.

Column I (Pattern File) is the path of the pattern file associated to this member pattern. A full path of the file is expected here, however, a path relative to the location of the loaded .ods file can be used for convenience. To check for path validity, you can double click the the cell and verify that the correct file opened up in notepad.

Chapter 5. The TestMethods Worksheet



F501

TestMethods

	Method Name	TestModule	Callback Before	Callback After	Pattern Group		
	tm_writeread	FunctionalTest			pat_writeread		
	tm_write00	FunctionalTest			pat_write00		
	tm_read00	FunctionalTest			pat_read00		
	tm_writeFF	FunctionalTest			pat_writeFF		
	tm_readFF	FunctionalTest			pat_readFF		
	tm_wrckbd	FunctionalTest			pat_wrckbd		
	tm_rdckbd	FunctionalTest			pat_rdckbd		
	tm_wrickbd	FunctionalTest			pat_wrickbd		
	tm_rdickbd	FunctionalTest			pat_rdickbd		
	tm_wrprotec	FunctionalTest			pat_wrprotec		
	tm_erase	FunctionalTest			pat_erase		
//	tm_leakage	PmuTest			pat_lead	not yet implemented	
	tm_romdump	FunctionalTest	u_SetInputPin	u_Dump2Console	pat_capture		

A test program is composed of test blocks that are executed one after the other as described in the TestFlow worksheet. Each test is executed according to the method described in this TestMethods Worksheet.

Column B (Method Name) assigns the name for the test method being described.

Column C (Test Module) select one from the predefined Modules in the Test Executive. At the moment, FunctionalTest is the only module available because this tester don't have a PMU (yet).

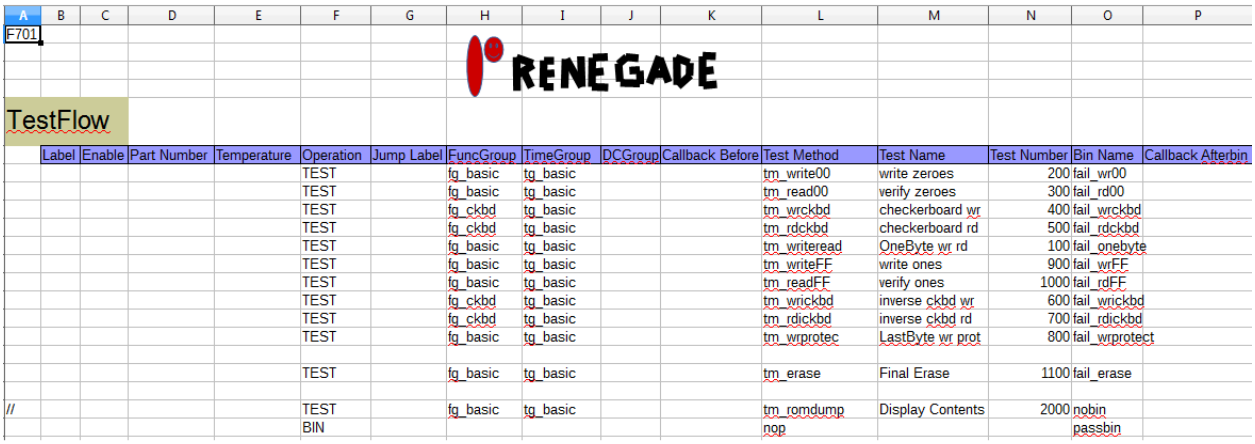
Column D (Callback Before) could be used to enter the name of a "Basic" macro to be executed before the selected module begins execution. The macro should be written in the UserCallbacks Module.

Column E (Callback After) could be used to enter the name of a "Basic" macro to be executed after the selected module has stopped execution. The macro should be written in the UserCallbacks Module.

Column F (Pattern Group) is the name of a pattern group that will be used by the selected module.

A "//" in the first column ignores all entries in that row. This applies to other worksheets.

Chapter 6. The TestFlow Worksheet



F701

RENEGADE

TestFlow

	Label	Enable	Part Number	Temperature	Operation	Jump Label	FuncGroup	TimeGroup	DCGroup	Callback Before	Test Method	Test Name	Test Number	Bin Name	Callback Afterbin
					TEST		fg_basic	tg_basic			tm_write00	write zeroes	200	fail_wr00	
					TEST		fg_basic	tg_basic			tm_read00	verify zeroes	300	fail_rd00	
					TEST		fg_ckbd	tg_basic			tm_wrckbd	checkerboard wr	400	fail_wrckbd	
					TEST		fg_ckbd	tg_basic			tm_rdckbd	checkerboard rd	500	fail_rdckbd	
					TEST		fg_basic	tg_basic			tm_writeread	OneByte wr rd	100	fail_onebyte	
					TEST		fg_basic	tg_basic			tm_writeFF	write ones	900	fail_wrFF	
					TEST		fg_basic	tg_basic			tm_readFF	verify ones	1000	fail_rdFF	
					TEST		fg_ckbd	tg_basic			tm_wrickbd	inverse ckbd wr	600	fail_wrickbd	
					TEST		fg_ckbd	tg_basic			tm_rdickbd	inverse ckbd rd	700	fail_rdickbd	
					TEST		fg_basic	tg_basic			tm_wrprotec	LastByte wr prot	800	fail_wrprotect	
					TEST		fg_basic	tg_basic			tm_erase	Final Erase	1100	fail_erase	
//					TEST		fg_basic	tg_basic			tm_romdump	Display Contents	2000	nobin	
					BIN						nop			passbin	

When the “Run” button is clicked, the Test Methods in Column L will be executed one after the other. Execution will start at row 7 downwards, a “//” in the first column is useful in skipping that row if desired.

Column F specifies the operation to be done for that row, valid entries are “TEST” & “BIN” (“JUMP” will be coming soon). TEST operation will execute the Test Method entered in Column L. BIN operation will assign the bin name in Column O to the Sites that are still standing.

Column H specifies the FUNCSET Group that will be used for the current Test Method. The User must make sure that the group name entered in this column matches one of the group names entered in FuncSets Worksheet. At the moment, the software does not validate the spelling of this entry, no warnings will be issued if the group name was not found, it will just ignore the entry and retain the previously loaded FUNCSET Group.

Column I specifies the TSET Group that will be used for the current Test Method. The User must make sure that the group name entered in this column matches one of the group names entered in TimeSets Worksheet. At the moment, the software does not validate the spelling of this entry, no warnings will be issued if the group name was not found, it will just ignore the entry and retain the previously loaded TSET Group.

Column L specifies the Test Method that will be executed for that row. The name of the Test Method entered in this column must match one of the entries in TestMethods Worksheet. No warnings will be issued on wrong spelling or if the method name was not found.

Column M can be used to give a name for the test operation in that particular row.

Column N assigns a test number for that particular row.

Column O specifies the bin name that will be assigned to the Sites that failed the test for the particular row.

The rest of the Columns not mentioned in this page is still under development, and would be available in the next versions of the software.