Computer Science Department, Technion – Israel Institute of Technology

Flash chips experiment conduction

User guide

Author: Hila Arobas
Superviser: Gala Yadgar

Table of contents

1. Introduction

Flash memory is an electronic non-volatile computer storage which is widely used in various modern technologies.
The popularity of flash memory motivates the search for methods to increase flash reliability and lifetime. An example for research on this subject can be found in the article "The Devil is in the Details: Implementing Flash Page Reuse with WOM Codes" by Dr. Gala Yadgar and Prof. Eitan Yaakobi.

This document is a full user guide, containing all the necessary information required for conducting experiments on flash chips. The rest of the paper is organized as follows: in section 2 the board is introduced, including instructions for flash chip exchange. In section 3 there are detailed explanations about some flash chips and the differences between them. Section 4 introduces the Sig software, its abilities and some examples of experiments conduction. Sections 4 and 5 are dealing with output processing using Excel and Gnuplot respectively. In section 7 you can find a terms glossary that can help you while reading this document.

2. SigNAS-II

2.1. General information and technical details

SigNAS-II is an analysis kit, which can be used to analyse NAND flash chips from wide range of manufacturers.

When writing data to a flash chip, the physical location of the data on the chip is determined using sophisticated algorithms. Examine the effects of cells programming on different variables, such as flash reliability and lifetime, require controlling the physical location of the programmed cells. Such ability is supplied by SigNAS-II.

In this document 'the board' is the notation for the SigNAS-II's physical component. The board is an electronic device dedicated for writing and reading one up to four flash chips simultaneously, using designated software. Using the board, one can conduct complicated experiments on flash chips and examine their results. The board contains four channels, one for each flash chip, allowing perform read and write operation to more than one flash chip simultaneously.

Board's component list (see figure1):

1. On/Off power switch.
2. Four flash chip channel frames. Note that all channels are empty except of the top right one.
3. Four sets of indicating lights. One set for each flash chip channel.
4. A flash chip.

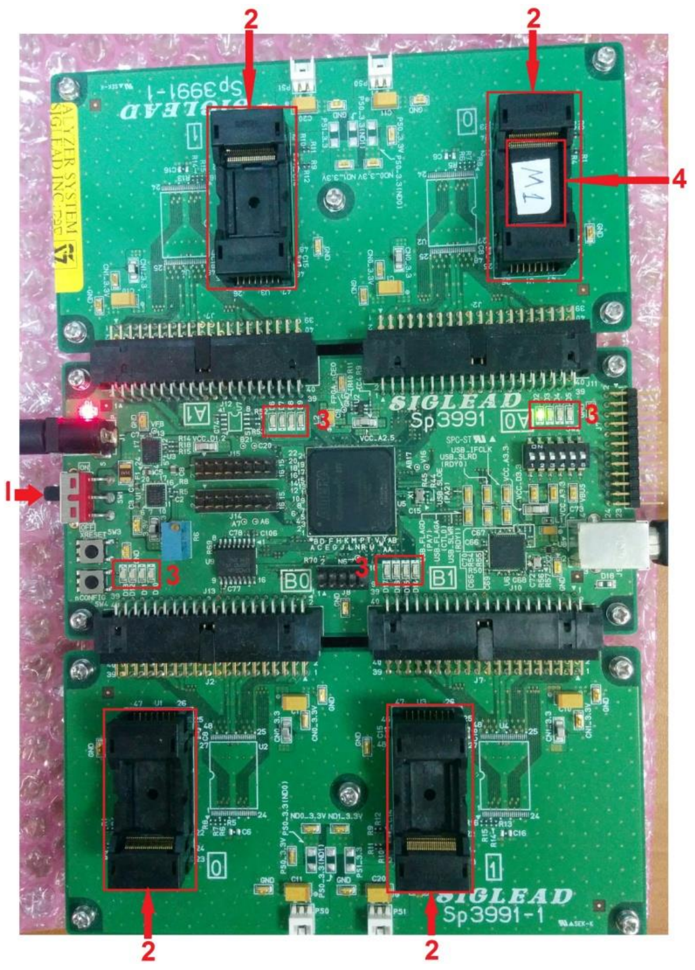


Figure 1: the board

2.2. Chip exchange instructions

Setting up the environment for a new experiment, often requires exchanging the flash chip on which the read and write operations are performed. This is done by the following steps:

- Shut down the power to the board using the power switch on the side of the device. The green indicating lights and the red power light next to the power switch will turn off.
- Select the new flash chip to be inserted to the channel and find the correct way to place it in the channel frame, according to the following instructions:
  - Find the lower side of the chip, by touching the pins on the side of the chip and pointing them down. By doing that, the lower side of the chip will face down, See figure2.

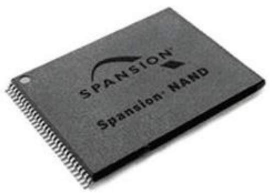


Figure 2: flash chip pointing down (the picture is for illustration only)

  - Find the chip corner where you can spot a hollow point on the upper side of the chip, figure2.
  - Find the channel frame corner, where you can spot a small arrow on the board itself next to the channel frame, see figure3.

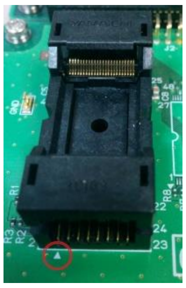


Figure 3: flash channel frame

  - The correct way to insert the flash chip to the cannel frame is to place the corners you just spotted, one on top of the other, while the lower side of the flash chip is facing down.
- Press down the channel frame and hold it down. Place the flash chip inside the channel frame in the correct way as explained in the previous paragraph. When the chip is placed in the channel frame, hold it down with your finger while loosening the channel frame. See figure4.

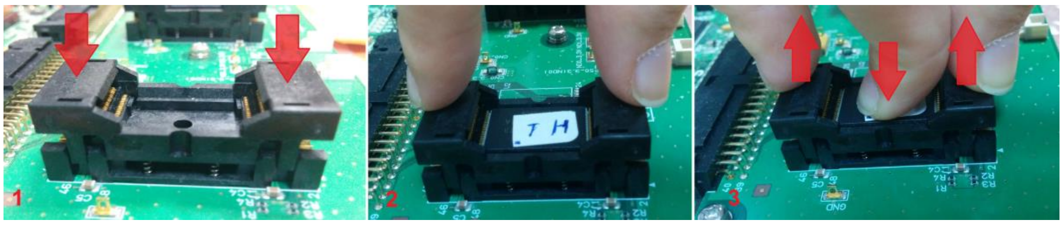


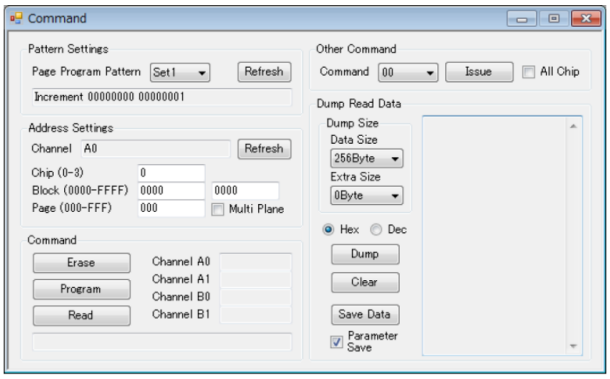
Figure 4: flash chip exchange process

- Turn on the power to the board using the power switch. If the indicating lights are turned green, the board is turned on successfully. Otherwise there was a problem and you need to repeat the whole process.
- Removing the current flash chip from its channel frame is done in the same way, i.e. by pressing down on the channel frame and pulling up the chip.

When the flash chip exchange completed successfully, you can start your experiment on the new flash chip using the designated softwares, as described in section 2.3 and section 4.

2.3. The designated software

SigNAS-II has designated software, providing the ability to perform various operations on the current flash chips located in the board channels. The command window is shown in figure5.


Command

Pattern Settings
Page Program Pattern: Set1
Increment 00000000 00000001

Address Settings
Channel: A0
Chip (0-3): 0
Block (0000-FFFF): 0000 0000
Page (000-FFF): 000
Multi Plane: ☐

Command
Channel A0
Channel A1
Channel B0
Channel B1

Other Command
Command: 00
All Chip: ☐

Dump Read Data
Dump Size
Data Size: 256Byte
Extra Size: 0Byte
[✓ Hex] Dec
Parameter Save: ✓

Figure 5: the SigNAS-II designated software

However, the designated software is lack of some abilities required for our experiments. Therefore, it is not suitable for executing our experiments.

A full manual for the SigNAS-II software can be found in a file called "SigNASII_OperationManual_Ver2_8.pdf". If you want to use the software ask the lab team for this file. The full manual contains installation instructions, settings, operations menu and tests explanations. The software is already installed on the lab computers, so you can skip the installation and settings chapters.

3. The flash chips

3.1. Flash chip technical differences

This section refers to the following differences between flash chips:

- Number of bytes per page.
- Number of pages per block.
- Chip lifetime in P/E cycles.
- Writing methods.
- Low-High coupling scheme.

3.2. Flash chips technical details

The changes described in section 3.1 are summarized in the following table for 3 types of chips: Hynix model A, Hynix model B and Micron:

Chip manufacturer	*Hynix model A*	*Hynix model B*	*Micron*
Bytes per page	8832	18048	4320
Pages per block	256	256	256
Chip lifetime	5,000	30,000	10,000
Writing method	normal	normal	XOR
Low-High coupling scheme	0	1	0

3.3. Technical differences regarding to the Sig program

Some of the technical details summarized at the above table are taken into consideration by the Sig program, using the environment variables. Those technical details are: bytes per page, pages per block and low-high coupling scheme (for more information see section 4.2).

The chip lifetime is an empirical parameter, specifying the number of erasures a block can endure before its reliability deteriorates below an acceptable level. The Sig program doesn't take this parameter into consideration, so make sure you set the correct number of P/E cycles according to the chip lifecycle when conducting your experiment.

Normal writing method indicates that you can conduct experiment involving multiple flash chips simultaneously. XOR writing method, as for the Micron flash chip, indicates that writing a vector of bits to the chip is done by scrambling this vector with an internal vector using XOR function. The Sig program first calculates the internal vector and alters the bits vector it wishes to write in a way that the scrambling doesn't interrupt the experiment. As a result, only one flash chip can participate in an experiment simultaneously, since the internal vector is unique to the chip. If you conducting an experiment on a flash chip with XOR writing method you should NOT use more than one chip simultaneously on the board! Otherwise you can conduct your experiment on up to four flash chips of the same type simultaneously.

4. The Sig program

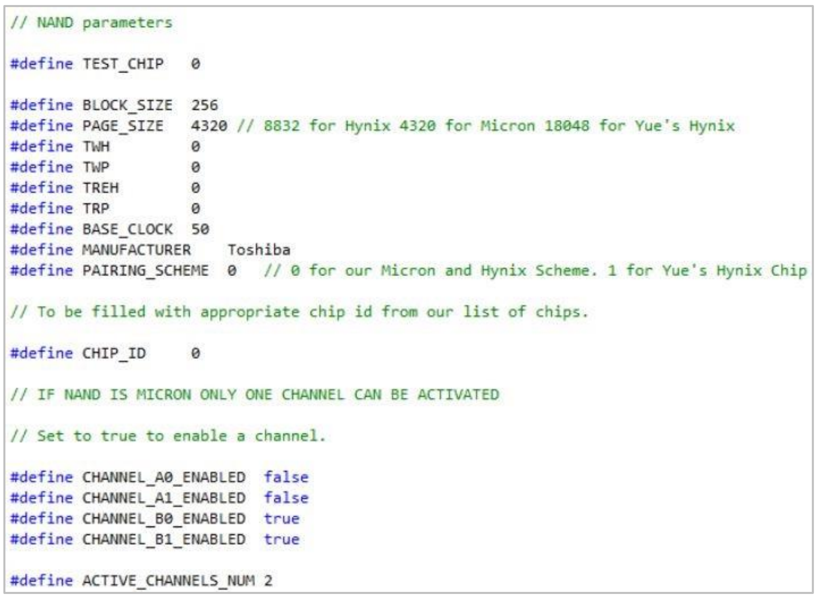
4.1. Introduction

Conducting experiments involving a long series of read and write operations on the board flash chips, requires designated software providing this ability. Since the built-in software doesn't provide this ability, there was a need to write such software. The Sig software is a C++ program written by Alex Yucovich, which provides this ability along with other advance operations.

The Sig program is the main software used in the experiments conduction. Therefore you required to know it well and be familiar with its abilities. In order to work with the program you need to contact Alex and ask him for access to the Sig program, and the location of the Sig.cpp and Sig.h files.

4.2. Environment variables and preparations

Running an experiment using the Sig program is done by several C++ commands described in the Sig API as you can see in section 4.3. Before running your experiment you are required to set up the environment for the Sig program. Setting up the experiment environment is done using the environment variables, located in the Sig.h file shown in figure6.



```
// NAND parameters

#define TEST_CHIP   0

#define BLOCK_SIZE  256
#define PAGE_SIZE   4320 // 8832 for Hynix 4320 for Micron 18048 for Yue's Hynix
#define TWH         0
#define TWP         0
#define TREH        0
#define TRP         0
#define BASE_CLOCK  50
#define MANUFACTURER    Toshiba
#define PAIRING_SCHEME  0   // 0 for our Micron and Hynix Scheme. 1 for Yue's Hynix Chip

// To be filled with appropriate chip id from our list of chips.

#define CHIP_ID     0

// IF NAND IS MICRON ONLY ONE CHANNEL CAN BE ACTIVATED

// Set to true to enable a channel.

#define CHANNEL_A0_ENABLED  false
#define CHANNEL_A1_ENABLED  false
#define CHANNEL_B0_ENABLED  true
#define CHANNEL_B1_ENABLED  true

#define ACTIVE_CHANNELS_NUM 2
```

Figure 6: the Sig program – Sig.h file

The first variable to be considered is the active channels. The board can support up to 4 flash chips simultaneously, physically located in 4 channels, denoted by A0, A1, B0 and B1, as you can see in figure7. Set CHANNEL_XX_ENABLED variables to true or false according to your experiment setup, as you can see in figure6. Set ACTIVE_CHANNELS_NUM variable to be the number of flash chips in your experiment setup, as you can see in figure6.

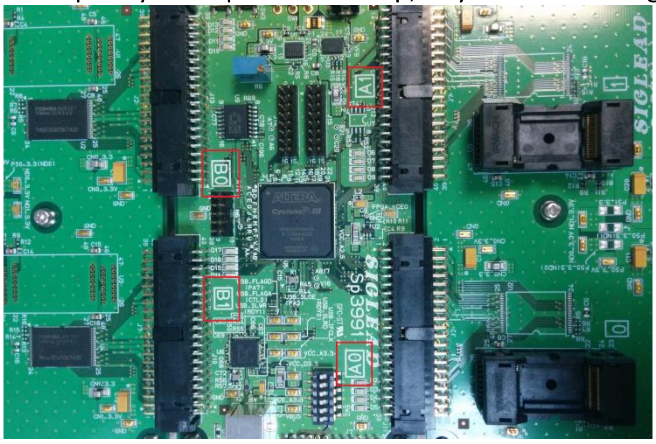


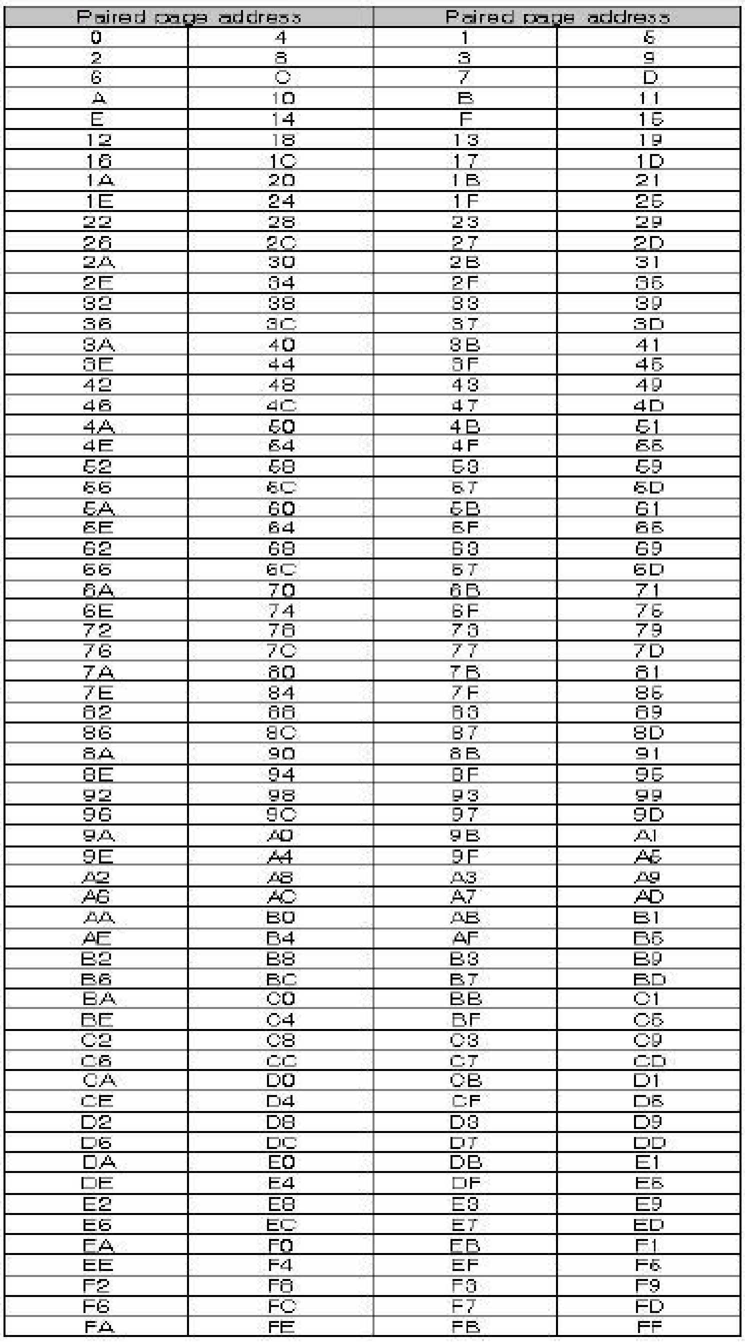
Figure 7: the board channels are surrounded by red squares

The following list detailed the additional environment variables to be set according to the current flash chip technical details as you can see in section 3.2:

• BLOCK_SIZE	the number of pages per block.
• MANUFACTURER	the flash chip manufacturer, such as Hynix, Micron, etc.
• PAGE_SIZE	the number of bytes per page.
• PAIRING_SCHEME	the coupling technique of low and high pages.

It is highly important to enter the correct parameters, otherwise the experiment won't yield reliable results, or an error will occur.

Pairing scheme variable refers to the coupling technique of low and high pages. For example, the table in figure8 details the page coupling of the Hynix model A falsh chip. If you run your experiment on a flash chip that doesn't present in the table in section 3.2, you should take into consideration that it may has a different pairing scheme. In that case, it is best to check the flash chip technical specification for the correct coupling technique, but you can also find that out by running the Sig tests on "dirty" blocks (explained later in this section) and detecting a failure in one or more tests.



Paired page address		Paired page address	
0	4	1	5
2	8	3	9
6	C	7	D
A	10	B	11
E	14	F	15
12	18	13	19
16	1C	17	1D
1A	20	1B	21
1E	24	1F	25
22	28	23	29
26	2C	27	2D
2A	30	2B	31
2E	34	2F	35
32	38	33	39
36	3C	37	3D
3A	40	3B	41
3E	44	3F	45
42	48	43	49
46	4C	47	4D
4A	50	4B	51
4E	54	4F	55
52	58	53	59
56	5C	57	5D
5A	60	5B	61
5E	64	5F	65
62	68	63	69
66	6C	67	6D
6A	70	6B	71
6E	74	6F	75
72	78	73	79
76	7C	77	7D
7A	80	7B	81
7E	84	7F	85
82	88	83	89
86	8C	87	8D
8A	90	8B	91
8E	94	8F	95
92	98	93	99
96	9C	97	9D
9A	A0	9B	A1
9E	A4	9F	A5
A2	A8	A3	A9
A6	AC	A7	AD
AA	B0	AB	B1
AE	B4	AF	B5
B2	B8	B3	B9
B6	BC	B7	BD
BA	C0	BB	C1
BE	C4	BF	C5
C2	C8	C3	C9
C6	CC	C7	CD
CA	D0	CB	D1
CE	D4	CF	D5
D2	D8	D3	D9
D6	DC	D7	DD
DA	E0	DB	E1
DE	E4	DF	E5
E2	E8	E3	E9
E6	EC	E7	ED
EA	F0	EB	F1
EE	F4	EF	F5
F2	F8	F3	F9
F6	FC	F7	FD
FA	FE	FB	FF

Figure 8: pairing scheme for Hynix model A flash chip

Another important preparation for your experiment is selecting a "clean" blocks to work with. A clean block is a block which has never been programmed before. Selecting a "dirty" block will undermine the credibility of the results. The lab team maintains a record of the dirty blocks for each of the flash chips in the lab. You should select clean blocks for your experiment, i.e. blocks that are not included in this record. The responsibility for updating this record is on you, so it is highly important you don’t forget to do so. The record can be found in files called "<chip manufacturer> Used Blocks.txt". Ask the lab team for this files location.

The blocks of a flash ship are organized in planes in a way that even blocks are physically located in plane 0 and odd blocks are physically located in plane 1. When conducting your experiment you should include both even and odd blocks in the tests, in order to include both planes in your tests.

4.3. The Sig program API

The Sig program was designed to execute complicated experiment on flash chips using the board. You can find examples for an experiment conducted by the Sig program in section 4.4. The Sig API functions are described below:

- bool BiasedBlockTest (int block, double bias, int iterNum);
  This function runs a standard test.
  Parameters:
  - block – the block number on which the Sig program will run your experiment.
  - bias – the probability of writing '0' (a number between 0 to 1).
  - iterNum – the number of P/E cycles.

- bool BiasedBlockTest (int block, double lowBias, double highBias, int iterNum);
  This function runs a standard test with different probability for writing '0' to the low and the high pages.
  Parameters:
  - block – the block number on which the Sig program will run your experiment.
  - lowBias – the probability of writing '0' to the low pages (a number between 0 to 1).
  - highBias – the probability of writing '0' to the high pages (a number between 0 to 1).
  - iterNum – the number of P/E cycles.

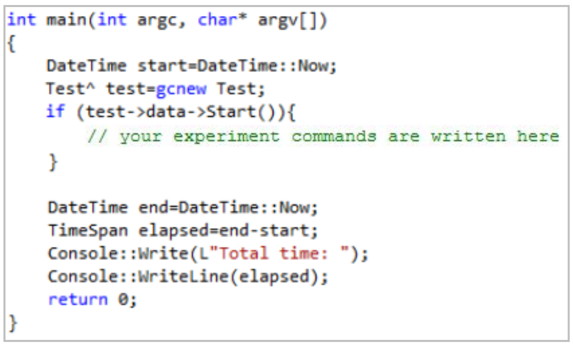
- bool L_LHTest (int block, int iterNum);
  This function runs a test with LLH reprogramming scheme.
  Parameters:
  - block – the block number on which the Sig program will run your experiment.
  - iterNum – the number of P/E cycles.

Be aware that your experiment will not necessarily require you to use all of the above functions. You should select only the necessary function after fully understanding the experiment goal.

After you finish entering your experiment commands to the Sig.cpp file, you should run it using Ctrl+F5. This will open a black console window which will remain open during the whole time the program is running, and let you to view the test status (success or failure).

4.4. Experiment examples

Your code should start with creating a new test. The Sig API functions should be executed only if the test was started successfully. You can see an example in the Sig.cpp file, shown in figure9.



```
int main(int argc, char* argv[])
{
    DateTime start=DateTime::Now;
    Test^ test=gcnew Test;
    if (test->data->Start()){
        // your experiment commands are written here
    }

    DateTime end=DateTime::Now;
    TimeSpan elapsed=end-start;
    Console::Write(L"Total time: ");
    Console::WriteLine(elapsed);
    return 0;
}
```

Figure 9: the Sig program – Sig.cpp file

The rest of this section describe a set of experiments conducted using the Sig program. The commands for those experiment are written inside the "IF" block shown in figure9.

You can find detailed explanation about this set of experiments in the article "The Devil is in the Details: Implementing Flash Page Reuse with WOM Codes" by Dr. Gala Yadgar and Prof. Eitan Yaakobi.

This set of experiments evaluated the long term effects of increased V_{th} on block's BER. Each experiment had two parts: programming the block with probability of 0.75 for '0' and 0.5 for '1', in the first part for a portion of its lifetime, and with probability of 0.5 for '0' and 0.5 for '1' in the second part, which consists of the remaining cycles. The length of the first part is varied between 20%, 40% and 60% of the block's lifetime.

The Sig program runs this experiment on a flash chip with lifecycle of 5,000 cycles using the following commands:

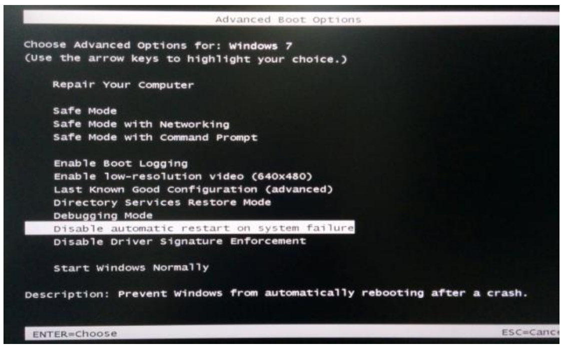
```
test->BiasedBlockTest (602, 0.75, 0.5, 1000);
test->BiasedBlockTest (602, 0.5, 4000);
test->BiasedBlockTest (604, 0.75, 0.5, 2000);
test->BiasedBlockTest (604, 0.5, 3000);
test->BiasedBlockTest (606, 0.75, 0.5, 3000);
test->BiasedBlockTest (606, 0.5, 2000);
```

The Sig program will perform biased writing with different low and high pages probability to block 602 for 1,000 P/E cycles and after that biased writing with the same low and high pages probability to block 602 for additional 4,000 P/E cycles. It then will perform the same writing for blocks 604 and 606 for different portion of the ship lifecycles.

4.5. Possible errors

If you followed all the orders carefully, you should not see errors. However, some errors may occur although all the instructions were done properly.

Chip exchange error:
After exchanging the flash chip successfully and verifying the board green lights are on, the board may still not initialized successfully. You will notice this error after trying running an experiment and receiving an error message right at the beginning, notifying that the Sig program could not establish connection. This error probably caused by gentle electricity contacts. In order to solve this problem shut down the board using the switch button, restart the computer, turn on the board and see if the problem was fixed. If the error still appears repeat the chip exchange process in addition to the above instructions. Note that when restarting the computer you need to disable automatic restart on system failure. This is done by pressing F8 immediately after power on, until the screen shown if figure10 is shown, and selecting this option.



```
Advanced Boot Options

Choose Advanced Options for: Windows 7
(Use the arrow keys to highlight your choice.)

    Repair Your Computer

    Safe Mode
    Safe Mode with Networking
    Safe Mode with Command Prompt

    Enable Boot Logging
    Enable low-resolution video (640x480)
    Last Known Good Configuration (advanced)
    Directory Services Restore Mode
    Debugging Mode
    Disable automatic restart on system failure
    Disable Driver Signature Enforcement

    Start Windows Normally

Description: Prevent Windows from automatically rebooting after a crash.

ENTER=Choose                                                  ESC=Canc
```

Figure 10: restart screen

Test failed error:
Programming errors are rare and not supposed to happen. At the end of each test performed by the Sig program, an indication regarding the test status appears in the console window. In case of an error, the Sig program will stop its running and will not proceed to the rest of the tests. There is no straightforward solution for this error. You should contact Alex Yucovich in order to analyze the problem.

Wrong parameters error:
This is the most risky error, since it mostly stays undetected. This error derives from incorrect parameters inserted to the Sig environment variables. The way to avoid this error is to double check the environment variables and to maintain order in your experiments.

4.6. Outputs

The Sig program output is DAT file that contain data in text format. The output files are located under the Sig folder (where the Sig program is located). Each one of the Sig API functions described in section 4.3 generate a DAT file, so when running a complex experiment combined of two tests, you should manually merge the two output DAT files. The suggested way to do this merge is using Excel program, as explained in section 5.

The name of the DAT file depends on the Sig test as follow:

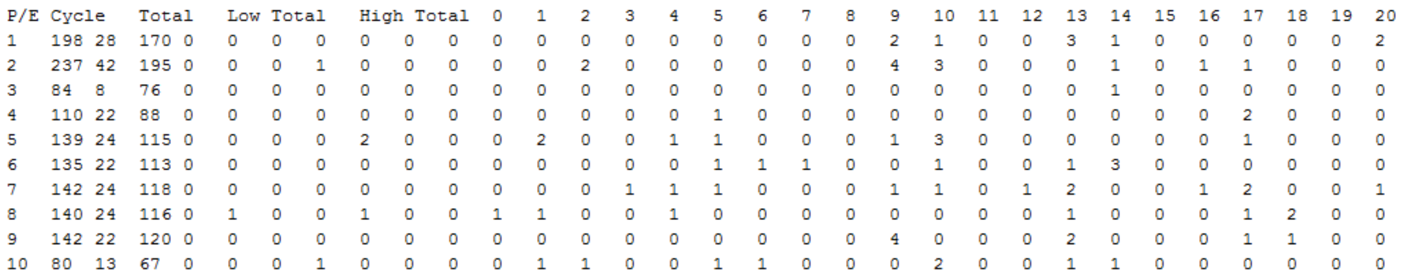
- For BiasedBlockTest the DAT file name is: *blockNum_probability_cycleNum_channelNum*, where *blockNum* is the block number, *probability* is the bias inserted to the test, *cycleNum* is the number of P/E cycles performed, and *channelNum* is 0 or 1 according to the flash chip channel.

- For BiasedBlockTest (second format) the DAT file name is: *blockNum_lowProbability_highProbability_cycleNum_channelNum*, where *blockNum* is the block number, *lowProbability* is the lowBias inserted to the test, *highProbability* is the highBias inserted to the test, *cycleNum* is the number of P/E cycles performed and *channelNum* is 0 or 1 according to the flash chip channel.

- For L_LHTest the DAT file name is: *blockNum_probability_cycleNum_channelNum*, where *blockNum* is the block number, *probability* is the probability for writing '0' to all pages (set by default to 0.5 for this test), *cycleNum* is the number of P/E cycles performed, and *channelNum* is 0 or 1 according to the flash chip channel.

The DAT file summarizes the number of bit errors per P/E cycle in several levels as follows:
- The entire block - number of bit errors in the entire block.
- The low/ high pages level - number of bit errors in all of the block's low/ high pages.
- The page level - number of bit errors in each one of the block's 256 pages.

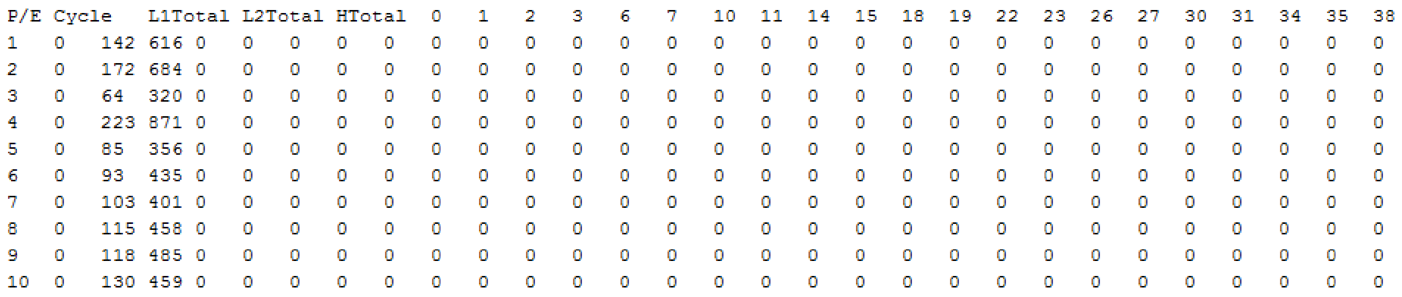
You can see the results for the first P/E cycles in figure11 (in the page level the results are partial due to space limits). Note that the column headers were shifted in this figure.



```
P/E Cycle   Total   Low Total   High Total  0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16  17  18  19  20
1   198 28  170 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2   1   0   0   3   1   0   0   0   0   0   2
2   237 42  195 0   0   0   1   0   0   0   0   0   2   0   0   0   0   0   4   3   0   0   0   1   0   1   1   0   0   0
3   84  8   76  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0
4   110 22  88  0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   2   0   0   0
5   139 24  115 0   0   0   0   2   0   0   0   2   0   0   1   1   0   0   0   1   3   0   0   0   0   0   0   1   0   0   0
6   135 22  113 0   0   0   0   0   0   0   0   0   0   0   0   1   1   1   0   0   1   0   0   1   3   0   0   0   0   0   0
7   142 24  118 0   0   0   0   0   0   0   0   0   0   1   1   1   0   0   0   1   1   0   1   2   0   0   1   2   0   0   1
8   140 24  116 0   1   0   0   1   0   0   1   1   0   0   1   0   0   0   0   0   0   0   0   1   0   0   0   1   2   0   0
9   142 22  120 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   4   0   0   0   2   0   0   0   1   1   0   0
10  80  13  67  0   0   0   1   0   0   0   0   1   1   0   0   1   1   0   0   0   2   0   0   1   1   0   0   0   0   0   0
```

Figure 11: output DAT file (column headers are shifted)

For the LLH test, the results has different format as you can see in figure12 (in the page level the results are partial due to space limits), since the bit errors are counted separately for each of the writes Low-Low-High. Note that in this case the bit errors in the page level are summarized using two columns for each of the low pages, one for the bit errors after the first write and one for the bit errors after the second write.



```
P/E Cycle   L1Total L2Total HTotal  0   1   2   3   6   7   10  11  14  15  18  19  22  23  26  27  30  31  34  35  38
1   0   142 616 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
2   0   172 684 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
3   0   64  320 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
4   0   223 871 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
5   0   85  356 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
6   0   93  435 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
7   0   103 401 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
8   0   115 458 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
9   0   118 485 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
10  0   130 459 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

Figure 12: output DAT file for LLH programming scheme (column headers are shifted)

When performing a complex experiment combined of two tests or more, each test will yield a DAT file. It is highly recommended to merge those file using Excel program as described in section 5. When merging two DAT file, you should pay attention to the differences between the output DAT file formats, i.e. different columns headers. An example for an experiment composed of two kind of test is the "Long term effects of reprogramming on a flash chip" described in the article "The Devil is in the Details: Implementing Flash Page Reuse with WOM Codes" by Dr. Gala Yadgar and Prof. Eitan Yaakobi. The Excel files that summarized the experiment results are called

"L_LH-<CHIP NAME> -<BLOCK PARITY>.xlsx"

Where <CHIP NAME> is the flash chip manufacturer and <BLOCK PARITY> can be "even" for even blocks, "odd" for odd blocks or "odd-even" for averaging over even and odd blocks.

5. Results processing using excel

5.1. Importing DAT files to Excel

As specified in section 4.6, the Sig program results are organized in DAT files. Those files can be imported to the Excel program and be processed using functions. Importing DAT file to Excel sheet is done according to the following steps:

- Open a new Excel sheet, and on the "Data" tab select "From text" in the "Get External Data" menu on the top left side of the screen. See figure13.

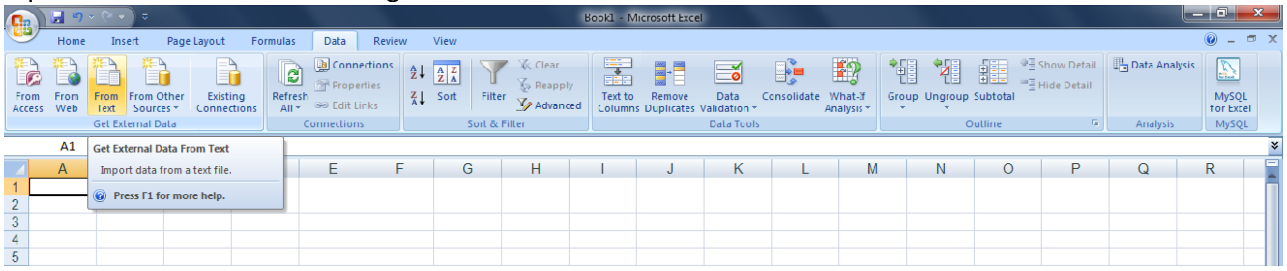
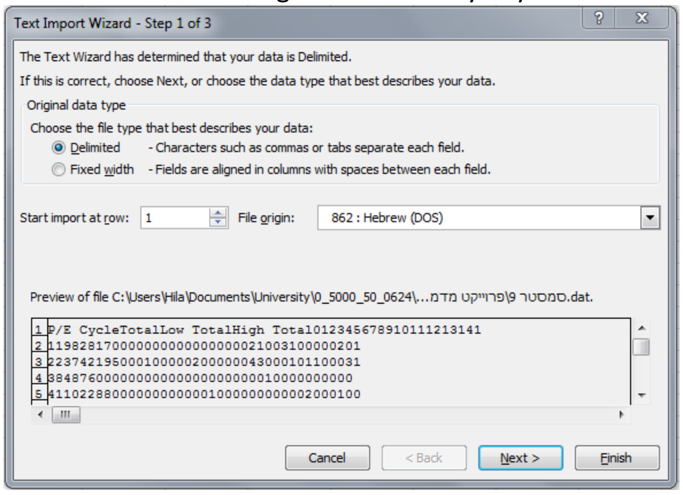


Figure 13: importing DAT file to Excel

- Choose the desired DAT file from your file system and click "Import". Note that files with DAT extension are not displayed by default, so make sure you set "All Files (*.*)" in the file system display option.

- In the "Text Import Wizard" window that opened choose "Delimited" in the "Original data type" section and click "Next >". This makes sure the data will be organized correctly in your excel sheet. See figure14.



Text Import Wizard - Step 1 of 3

The Text Wizard has determined that your data is Delimited.

If this is correct, choose Next, or choose the data type that best describes your data.

Original data type

Choose the file type that best describes your data:

[✓ Delimited] - Characters such as commas or tabs separate each field.

Fixed width - Fields are aligned in columns with spaces between each field.

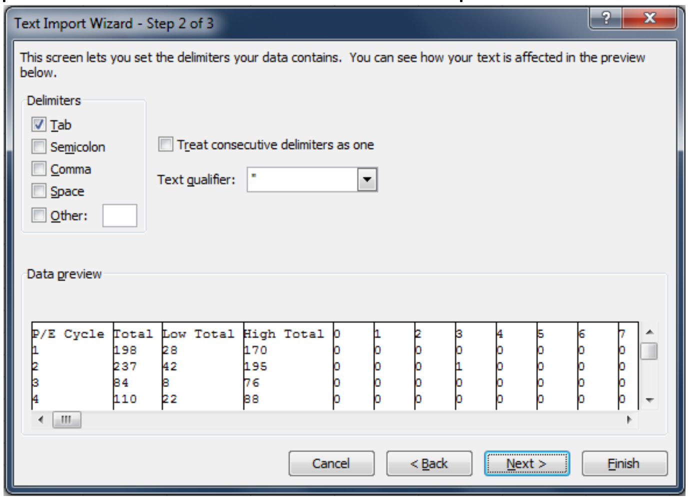
Start import at row: 1 File origin: 862 : Hebrew (DOS)

Preview of file C:\Users\Hila\Documents\University\0_5000_50_0624\...פרוייקט מדמ\9 סמסטר.dat.

```
1 P/E CycleTotalLow TotalHigh Total01234567891011213141
2 11982817000000000000000000210031000002O1
3 22374219500010000020000004300010110003l
4 384876000000000000000000000010000000000
5 41102288000000000000000100000000000002000100
```

Figure 14: importing DAT file to Excel – setting import configuration

- In the next window that opens select "Tab" in the "Delimiters" option and click "Finish". See figure15.



Text Import Wizard - Step 2 of 3

This screen lets you set the delimiters your data contains. You can see how your text is affected in the preview below.

Delimiters

[✓ Tab]

Semicolon

Comma

Space

Other:

Treat consecutive delimiters as one

Text qualifier: "

Data preview

```
P/E Cycle Total Low Total High Total 0  1  2  3  4  5  6  7
1         198   28        170        0  0  0  0  0  0  0  0
2         237   42        195        0  0  0  1  0  0  0  0
3         84    8         76         0  0  0  0  0  0  0  0
4         110   22        88         0  0  0  0  0  0  0  0
```

Figure 15: importing DAT file to Excel – setting import configuration

- In the next window that open select the specific cell you want to locate the data and click "OK". See figure16.

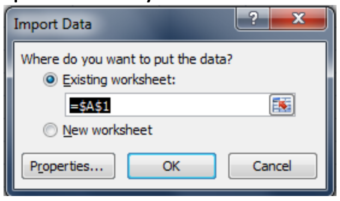
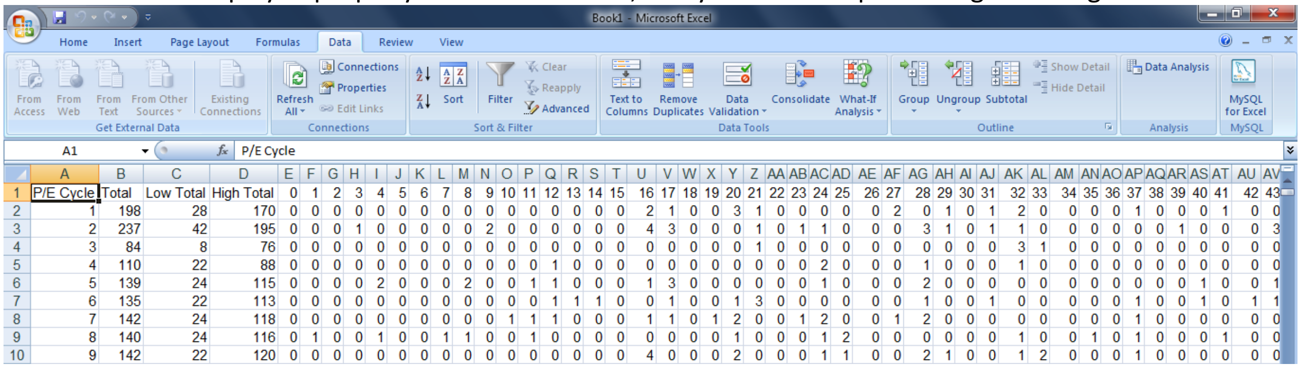


Figure 16: importing DAT file to Excel – locating the data

- The data is now displayed properly in the Excel sheet, and you can start processing it. See figure 17.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	P/E Cycle	Total	Low Total	High Total	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	1	198	28	170	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	2	237	42	195	0	0	0	1	0	0	0	0	0	2	0	0	0	0	0	0
4	3	84	8	76	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	4	110	22	88	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
6	5	139	24	115	0	0	0	0	2	0	0	0	2	0	0	1	1	0	0	0
7	6	135	22	113	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0
8	7	142	24	118	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0
9	8	140	24	116	0	1	0	0	1	0	0	1	1	0	0	1	0	0	0	0
10	9	142	22	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR	AS	AT	AU	AV
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
2	2	1	0	0	3	1	0	0	0	0	0	2	0	1	0	1	2	0	0	0	0	1	0	0	0	1	0	0
3	4	3	0	0	0	1	0	1	1	0	0	0	3	1	0	1	1	0	0	0	0	0	0	1	0	0	0	3
4	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	3	1	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	2	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
6	1	3	0	0	0	0	0	0	1	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
7	0	1	0	0	1	3	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	1	0	1	1
8	1	1	0	1	2	0	0	1	2	0	0	1	2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
9	0	0	0	0	1	0	0	0	1	2	0	0	0	0	0	0	1	0	0	1	0	1	0	0	0	1	0	0
10	4	0	0	0	2	0	0	0	1	1	0	0	2	1	0	0	1	2	0	0	0	1	0	0	0	0	0	0

Figure 17: DAT file displaying using Excel

When conducting complex experiment that runs two different tests on the same block, you are required to make some adaptations. For example, an experiment start by running test1 for 60% of the chip lifetime and then running test2 for the rest 40% of the chip lifetime on the same block of a chip with lifetime of 10,000 P/E cycles. The Sig program will generate two DAT files, one for each test. The P/E cycle column in each of the DAT files will start with counter 1, although it supposed to start with 6,001 in the result of test2, since both tests were performed on the same block. This means that you have no way to know which one of the tests was first and which one was second. You can cope with that by maintaining order in your experiments and results.

In order to combine the two DAT files in the above example, using Excel, you can import them to two different Excel sheets, and concatenate one to another using copy-paste. Note that the P/E cycle column will not be updated accordingly. However this allows you to analyze the result of the entire experiment at once.

5.2. Bit error rate (BER) calculation

Bit error rate (BER) is a mathematic measure indicating the average number of bit errors per page. The DAT files contain the number of bit errors in several levels inside a block: in a single page, in all low pages, in all high pages and in all block's pages. Therefore in order to calculate the BER measure, you need to divide the number of bit errors by the number of bits in the matching level. The number of bits per block is calculated using this formula:

$$bits\ \ per\ block = (pages\ \ per\ block)\cdot(bytes\ \ per\ \ page)\cdot(bits\ \ per\ byte)$$

Note that the number of low/ high pages in a block is exactly half of the total number of pages in a block.

BER calculation examples:

- BER calculation for the entire block: $BER(all\ block\ \ pages) = \dfrac{total\ \ number\ \ of\ \ bit\ errors}{number\ \ of\ \ bits\ \ per\ block}$
- BER calculation for the low pages: $BER(low\ \ pages) = \dfrac{low\ \ pages\ bit\ errors}{0.5\cdot(number\ \ of\ \ bits\ \ per\ block)}$

5.3. Useful excel functions

This section details some Excel functions that can help you analyze the results efficiently. You can always use other Excel functions that meet the requirements.

Analyzing the experiment result is usually done by grouping together every X consecutive P/E cycles, where X is 1% of the chip lifecycles. The Excel program provides the ability to activate a function on that group of results. For example for flash chip with lifecycle of 5,000 P/E cycle, we use the following format:

<*Excel function*>(INDEX(B:B,2+50*(ROW()-ROW(G2))):INDEX(B:B,1+50*(ROW()-ROW(G2)+1)))

The <*Excel function*> will be activated on every consecutive 50 rows in column B starting from cell B2, and will write the results to column G starting from cell G2.

Usage examples:

- Averaging over every 50 consecutive P/E cycles:

  AVERAGE(INDEX(B:B,2+50*(ROW()-ROW(G2))):INDEX(B:B,1+50*(ROW()-ROW(G2)+1)))

- Calculating the standard deviation of every 50 consecutive P/E cycles:

  STDEV(INDEX(B:B,2+50*(ROW()-ROW(K2))):INDEX(D:D,1+50*(ROW()-ROW(K2)+1)))

- Calculating the variance of every 50 consecutive P/E cycles:

  VAR(INDEX(B:B,2+50*(ROW()-ROW(L2))):INDEX(D:D,1+50*(ROW()-ROW(L2)+1)))

- Calculating the 90-precentile of every 50 consecutive P/E cycles:

  PERCENTILE(INDEX(B:B,2+50*(ROW()-ROW(M2))):INDEX(D:D,1+50*(ROW()-ROW(M2)+1)),0.9)

6. Results processing using Gnuplot

6.1. Introduction

Gnuplot is a portable command-line driven graphing utility for Windows, Linux and many other platforms. Gnuplot provide advanced abilities for graphical analysis of large scale data. This section will provide you basic information for using Gnuplot to analyze your experiments results.

6.2. Download and install Gnuplot

Gnuplot can be downloaded for free from the following link https://sourceforge.net/projects/gnuplot/files/gnuplot/
Notice that this document refers to Gnuplot5.0, so if you download any other version you should check for updates. Installing Gnuplot is straightforward and doesn't require any special notes.

Gnuplot output is an EPS file (Encapsulated Postscript) which can contain text as well as graphics. In order to view EPS files properly you are required to download two additional softwares:

- Ghostscript – download from this link http://www.ghostscript.com/download/
- GSview – download from this link http://pages.cs.wisc.edu/~ghost/

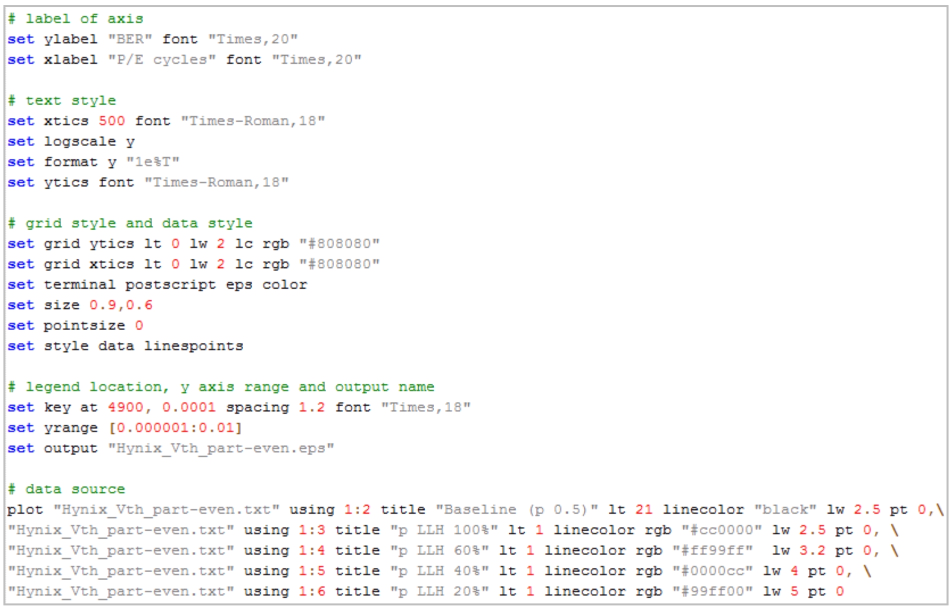
Complete user manual for Gnuplot program can be found in this link: http://www.gnuplot.info/docs_5.0/gnuplot.pdf
In addition a complete Gnuplot script is shown in figure18 under section 6.3

6.3. Recommended work plan with Gnuplot

It is highly recommended to crate Gnuplot script containing set of selected commands for creating uniform appearance results and saving time. The same script can be later updated for each of the experiments results separately.

Gnuplot script can is a PLT file that reads data from a text file. Therefore you should create a text file that will contain all of the data you wish to present in your graph. Copying data from Excel sheet to a text file is done by selecting the desired cells and pasting in the text file. The columns may be shifted, but this should not concern you. Save the text file without changing anything else.

Your Gnuplot script can now refer to the data located in a specific column of the text file. For example, the Gnuplot script that shown in figure18, reads data from "Hynix_Vth_part-even.txt" file and generate a graph containing 5 separate lines, one for each column from 2 to 6 (the columns are counted starting from 1). This script also gives each line a title and different color, and set the output file name to be "Hynix_Vth_part-even.eps". Note that the Gnuplot script and the text file should be located in the same location in your file system, and that the output EPS file will be generated at the same location as well.



```
# label of axis
set ylabel "BER" font "Times,20"
set xlabel "P/E cycles" font "Times,20"

# text style
set xtics 500 font "Times-Roman,18"
set logscale y
set format y "1e%T"
set ytics font "Times-Roman,18"

# grid style and data style
set grid ytics lt 0 lw 2 lc rgb "#808080"
set grid xtics lt 0 lw 2 lc rgb "#808080"
set terminal postscript eps color
set size 0.9,0.6
set pointsize 0
set style data linespoints

# legend location, y axis range and output name
set key at 4900, 0.0001 spacing 1.2 font "Times,18"
set yrange [0.000001:0.01]
set output "Hynix_Vth_part-even.eps"

# data source
plot "Hynix_Vth_part-even.txt" using 1:2 title "Baseline (p 0.5)" lt 21 linecolor "black" lw 2.5 pt 0,\
"Hynix_Vth_part-even.txt" using 1:3 title "p LLH 100%" lt 1 linecolor rgb "#cc0000" lw 2.5 pt 0, \
"Hynix_Vth_part-even.txt" using 1:4 title "p LLH 60%" lt 1 linecolor rgb "#ff99ff"  lw 3.2 pt 0, \
"Hynix_Vth_part-even.txt" using 1:5 title "p LLH 40%" lt 1 linecolor rgb "#0000cc" lw 4 pt 0, \
"Hynix_Vth_part-even.txt" using 1:6 title "p LLH 20%" lt 1 linecolor rgb "#99ff00" lw 5 pt 0
```

Figure 18: Gnuplot script

After creating EPS file using the Gnuplot script you can now open it using the Ghostscript and GSview programs, by selecting it, clicking on the right click, choosing "Open with" option and selecting the corresponding program. In the window that opens, you will be able to see the graph created by Gnuplot, as you can see in figure19.

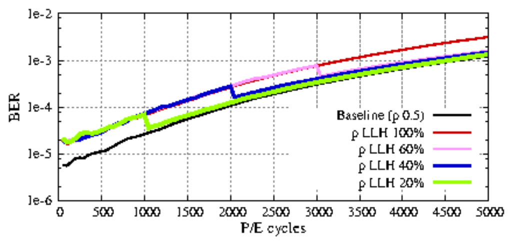


Figure 19: Gnuplot outcome

Note that you may need to adapt your script to your data, for example by changing the Y axis range, placing the legend in different location, etc…

There are some predefined Gnuplot scripts from previous experiments you can use as templates. Those scripts are named as follows:

BER_<CHIP NAME> - <BLOCK PARITY>

90Percentile_<CHIP NAME> - <BLOCK PARITY>

The CHIP NAME is the chip manufacturer (with the exception of Hynix model B, which is called "Hynix-Yue"). The BLOCK PARITY can be "even" or "odd" according to the block number, or "odd-even" for averaging over odd blocks and even blocks results. Each script was designed for three types of experiments described in the article "The Devil is in the Details: Implementing Flash Page Reuse with WOM Codes" by Dr. Gala Yadgar and Prof. Eitan Yaakobi. The experiments are referred in the article with the following descriptions:

- Long term effects of increased Vth on a flash chip.
- Long term effects of reprogramming on a flash chip.
- Short term effects of reprogramming on a flash chip.

7. Terms glossary

- ***The board:***
  The SigNAS-II's physical component.

- ***Flash chip:***
  A flash chip, placed inside the board channel during the experiments.

- ***Board channel:***
  The board contain up to four channels. The board channels are shown in figure1, denoted by the number 2, and surrounded by red square.

- ***Board channel frame:***
  The physical component of the board channel, that keeps the flash chip in place.

- ***P/E cycle:***
  Program/Erase (P/E) cycle is one cycle of programming and erasing a flash chip block.

- ***Environment variables:***
  Sig program variables that determine the set up for the experiment conducted using this software. For more information see section 4.2.

- ***Bit error rate (BER):***
  A mathematic measure indicating the average number of bit errors.