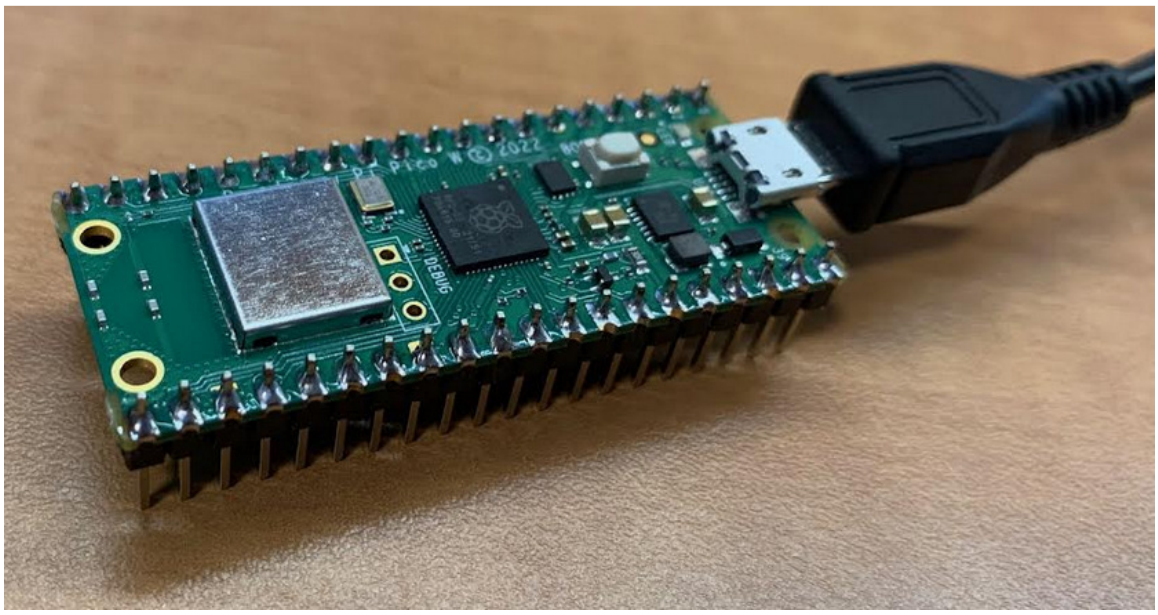
Pico-Flash-Utility



Firmware Version 2.00
User Guide
Updated January 4th, 2023

IMPORTANT :
This User Guide is about Pico-Flash-Utility firmware Version 2.00 from Andre St. Louys.

Join our Pico-Flash-Utility discussion group on:
https://github.com/astlouys/Pico-Flash-Utility/discussions

Pico-Flash-Utility User Guide

Index

Index 2
About Pico-Flash-Utility Version 2.00 3
Introduction 3
Operation 5
1) Display Pico's manufacturing test results. 7
2) Display Pico's flash memory specific sector. 8
3) Display Pico's complete flash address space. 10
4) Display Pico's complete RAM address space. 11
5) Display firmware functions address. 12
6) Erase all flash and generate Pico's complete log. 13
7) Erase a specific sector of Pico`s flash. 14
8) Erase Pico`s whole flash address space. 15
9) Flash memory blank check. 16
10) Flash memory test. 17
11) Clear screen. 18
Appendix A – Configuring a terminal emulator program 19
Pico to computer, USB-to-USB 19

Do you want to share your experience with the Pico-Flash-Utility and help other users?
Join our discussion group on:
https://github.com/astlouys/Pico-Flash-Utility/discussions

If you want to send me a personal email (as long as it is something constructive),
here is my email address:

Andre St. Louys
(Quebec, Canada)
astlouys@gmail.com

Pico-Flash-Utility User Guide

About Pico-Flash-Utility Version 2.00

This release of the Pico-Flash-Utility adds a flash test option. Some cleanup and optimization has also been done in the code.

This utility is for the Raspberry Pi Pico (and / or PicoW). You will find more information throughout this User Guide on how to use it and what you can do with it.

Introduction

As you may guess by its name, the Pico-Flash-Utility is a Firmware allowing some basic operations on the Raspberry Pi Pico flash memory.

The Firmware runs on both the Raspberry Pi Pico and the Raspberry Pi PicoW with only one minor difference that will be discussed later. So, for the sake of simplicity, we'll use the term "Pico" throughout this guide, but it applies to both the Pico and the PicoW.

The only equipment required to use this Firmware is the Raspberry Pi Pico itself, along with an external PC running a terminal emulator program. I personally use TeraTerm and it is perfect for the job. Moreover, it is an open source and freeware running on Windows. The Pico will communicate with the PC for input and output by the way of CDC USB communication. You can find basic information about that in Appendix A and there are also many sources of information on the Internet.

You will also need a text editor to navigate through the log files generated by the terminal emulator program. I use the free version of EmEditor (free for personal use). It is very fast, supports large file and have many features... In short, this is an excellent tool. Please support the developers and pay for the EmEditor Professional license if you use it for other than personal use.

The Pico-Flash-Utility Firmware runs from Pico`s RAM memory so that you can even erase all the flash memory if you want. This is useful if you want to give away a microcontroller unit but you know that your WiFi access credentials have been saved in its non volatile memory (or other confidential information). To build a firmware to run from RAM on the Pico, open a Linux terminal session, go to the "build" directory and write the following command:
cmake –DPICO_COPY_TO_RAM=1
This command needs to be executed only once. This assumes that you write with Visual Studio Code on a Raspberry Pi platform running Raspian. Instructions may differ if you use another development platform.

Pico-Flash-Utility User Guide

During the Pico production (manufacturing) process, “Pico Manufacturing Tests” are performed on the microcontroller. The 107 bytes of the test results are saved to the Pico`s flash at address 0x1007F000. Those bytes will not be overwritten by the Pico-Flash-Utility, even if you select to erase the sector that contains them. The sector will be recognized by the Firmware and the 107 bytes will be protected. Obviously, if you happen to load a program that is so large that it overwrites those bytes, the Firmware will not touch whatever bytes have been saved in the 107 bytes of those flash memory locations.

Basically, the features / operations supported by the Pico-Flash-Utility are the following:

- Display the type of microcontroller (auto-detection of Pico or PicoW).
- Display the “Pico Unique Number” (extracted from the flash IC).
- Display the Pico’s Manufacturing Test results.
- Display a specific sector of the flash memory.
- Display the complete flash memory address space.
- Display the complete RAM memory address space.
- Display the Firmware functions address (to confirm they run from RAM).
- Erase a specific sector of the flash memory.
- Erase the complete flash memory address space (will be reset to 0xFF).
- Perform a “Blank check” of the flash memory space.
- Perform a write test on the whole flash memory space.
- Automate many of the operations above by selecting a single menu choice.

As you can see, many operations are based on “flash sector”. Be aware that a sector is 4096 bytes long on the Pico (0x1000 in hex).

Operation

First of all, you need to connect your Pico to a computer in order to transfer the Pico-Flash-Utility firmware to the microcontroller. This guide assumes that you already know how to proceed to do so.

When the Pico-Flash-Utility has been transferred, the Firmware will be launched automatically. At this point, the LED on the Pico will blink, indicating that it is waiting for the CDC USB communication to be established.

NOTE: The PicoW has changed the way the local Pico's LED is controlled by software. To be able to blink the local LED on the PicoW, system libraries for the CYW43 are required and loading those libraries would extend the size of the Firmware to about 575 kBytes, making it too big to be run from RAM (which is 264 kBytes). As a consequence, If you use a PicoW, you won't see thePico`s local LED blinking while it is waiting to establish CDC USB communication. But you still have to proceed with this connection.
This is the only difference you need to know if you use a PicoW.

When the CDC USB communication has been established and the terminal emulator program launched, the Firmware will automatically detect it and display the Firmware menu.

Pico-Flash-Utility main menu

This screenshot has been taken from the TeraTerm terminal emulator software, and as can be seen at the top of the screen, the type of microcontroller has already been auto-detected and is displayed (Pico or Pico W), along with the "Pico`s Unique ID".

The auto-detection is based on the fact that some changes have been done to PicoW's power supply and local LED architectures. By changing the level of GPIO25 (used on the Pico for local LED) and reading back the voltage from one of the ADC GPIO, we can determine if we are running on a Pico or a PicoW.

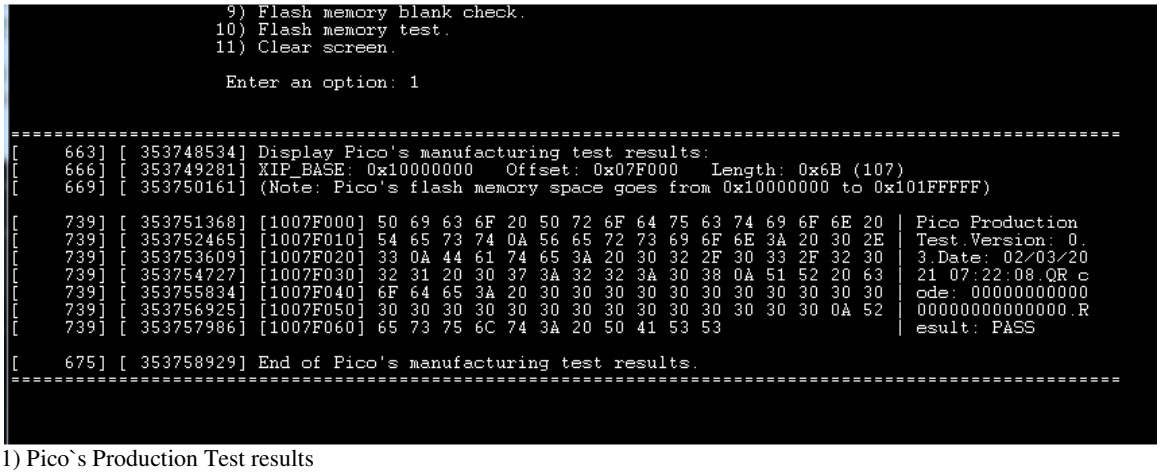
The microcontroller Unique ID is also displayed on screen. It is made of 16 hexadecimal digits taken from the flash memory integrated circuit ("IC") on the Pico. As you can understand from its name, this Unique ID is different from one flash IC to another. If you connect another Pico and display the Unique ID again, you will see that this number has changed. Even if it not the exact truth, we can consider this number as a "serial number" unique to each Pico board.

The eleven (11) menu options are also displayed and each one is explained in the next sections of this guide.

1) Display Pico's manufacturing test results.

This function will display the results of the Manufacturing Test of your Raspberry Pi Pico. This data is saved at offset 0x7F000 in the flash and is 107 bytes long. This data may have already been overwritten by one of the programs you previously loaded on the Pico. However, its offset in flash is 0x7F000. It requires a relatively large program to extend up to that point in the flash.

As mentioned in the introduction, this data will be preserved by the Pico-Flash-Utility even if you select to erase that sector offset (0x7F000) or if you select to erase the complete flash memory.



1) Pico`s Production Test results

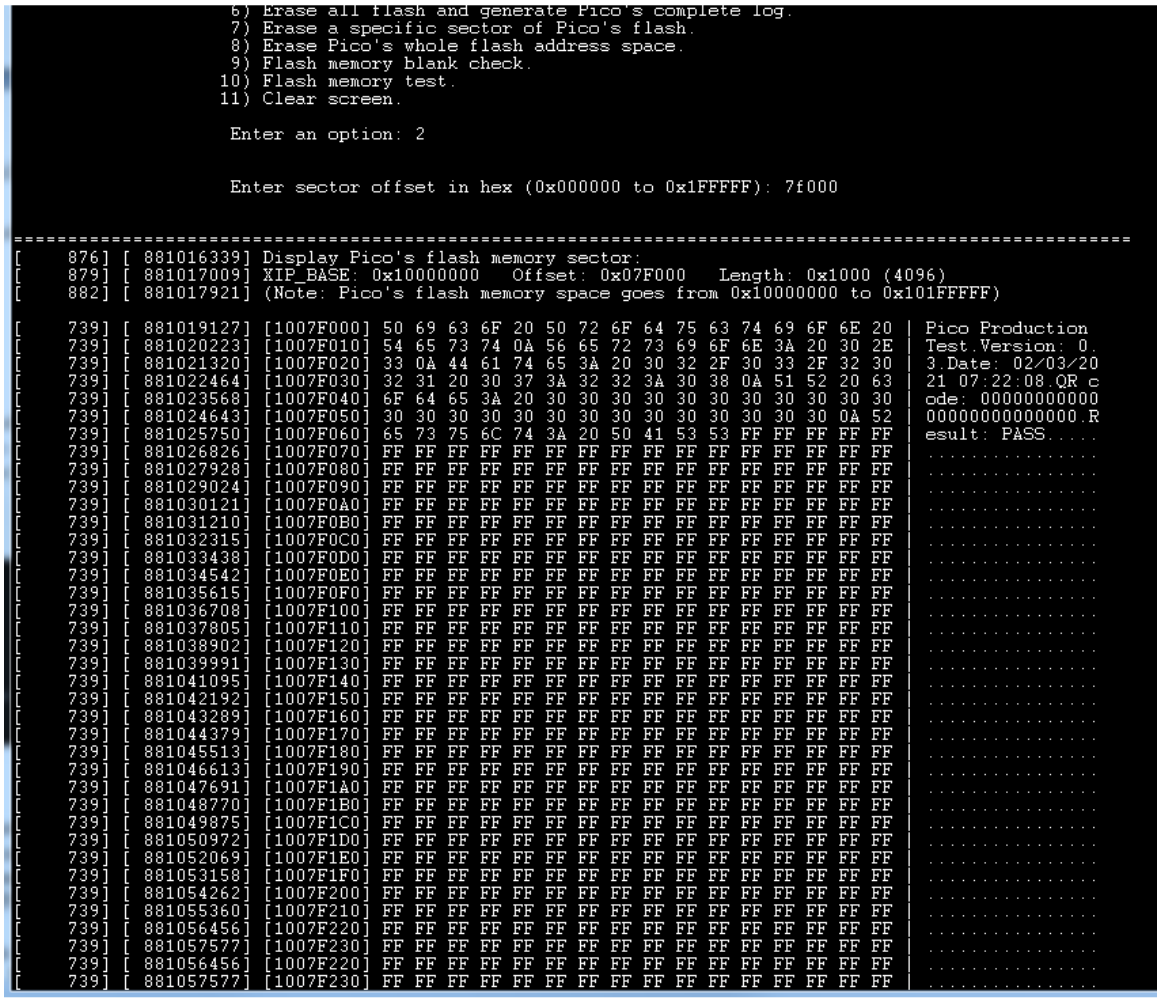
The display format of the Production Test results is similar to the other display options of the Pico-Flash-Utility. The top of the display (just below the horizontal bar) gives some technical information about the display that follows. Then, on the left of the screen, the source code line number is given, followed by the Pico`s current timer value, then the address of the first data byte of that line. After that, each line contains 16 bytes in hex, a vertical separator, and the same 16 bytes in ASCII. The bytes that are not displayable in ASCII are replaced by a dot (.) in the ASCII display part.

The timer value (in microseconds) is shown to give an idea of the relative delay between different lines in the display.

2) Display Pico's flash memory specific sector.

This function will display the content of a specific sector of the flash memory. If you select this option (2), the system will ask you which sector offset you want to display. The sector offset must be entered in hex (see example below) and the usual leading "0x" header is optional

.



2) Display a specific sector of flash memory

NOTES:

1) Keep in mind that you must enter the sector *offset*, not the sector address. So, for example, you may want to display the flash memory content from 0x1007F000 up to 0x1007FFFF (containing the Manufacturing Test results). 0x1000000 being the base address of the flash memory you must enter 7F000 (leading 0x is optional) which represents the offset of the sector.
2) All the data is not shown in the picture above. In fact, displaying a complete sector (4096 bytes) will scroll out of the screen. You must turn on the log

capabilities of your terminal emulator program so that the data is saved to a log file. Then, with your file editor, you can open this file and analyze the data at will.

3) If the offset specified is not on a sector boundary (a multiple of 0x1000), the system will decrease the given offset (until reaching a sector boundary) and display the sector containing the address you specified. In other words, the sector offset you enter should always end with three zeroes “0” (hex 0`s), like in B000, 1A000, 1FC000, or similar values.

3) Display Pico's complete flash address space.

This function will display the content of all flash memory space, from 0x10000000 up to 0x101FFFFF. For the same reason than the one given in section 3 above, you will need a text editor to review and analyze the data. Since there are 2MBytes of data to display, it may take a few minutes to display it all (2 to 5 minutes, depending on the serial speed, system performance, terminal emulator program used, etc).

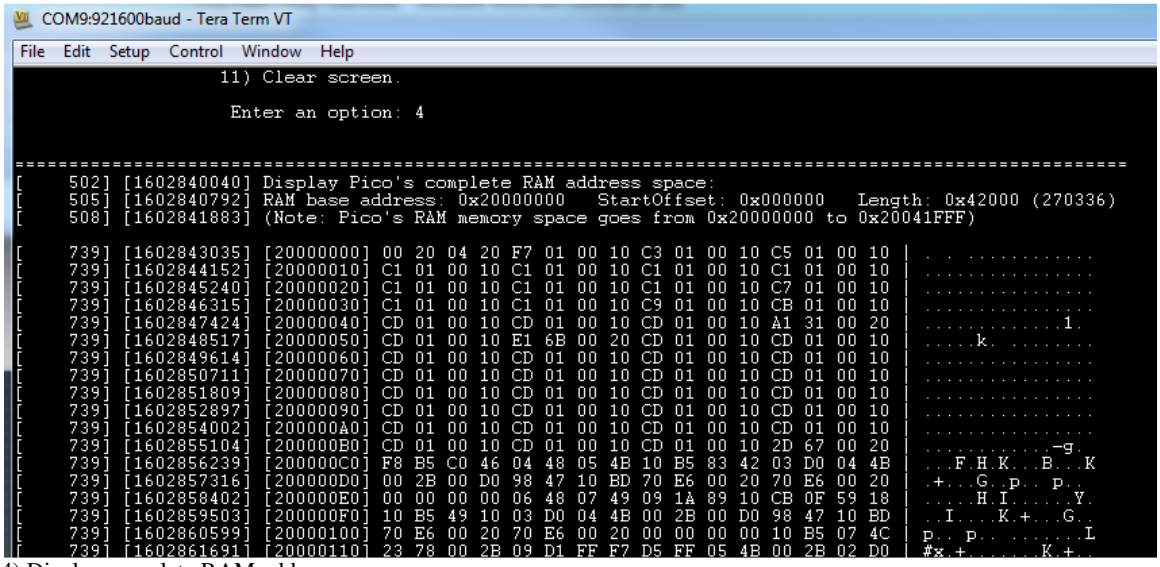
```
                  9) Flash memory blank check.
                 10) Flash memory test.
                 11) Clear screen.

                  Enter an option: 3


================================================================================================
[    461] [1138275766] Display Pico's complete flash address space:
[    464] [1138276540] XIP_BASE: 0x10000000   StartOffset: 0x00000000   Length: 0x00200000 (2097152)
[    467] [1138277602] (Note: Pico's flash memory space goes from 0x10000000 to 0x101FFFFF)

[    739] [1138278794] [10000000] 00 B5 32 4B 21 20 58 60 98 68 02 21 88 43 98 60 | ..2K! X`.h.!.C.`
[    739] [1138279941] [10000010] D8 60 18 61 58 61 2E 4B 00 21 99 60 02 21 59 61 | .`.aXa.K.!.`.!Ya
[    739] [1138281027] [10000020] 01 21 F0 22 99 50 2B 49 19 60 01 21 99 60 35 20 | .!."P+I.`.!.`5
[    739] [1138282111] [10000030] 00 F0 44 F8 02 22 90 42 14 D0 06 21 19 66 00 F0 | ..D.."B...!.f..
[    739] [1138283212] [10000040] 34 F8 19 6E 01 21 19 66 00 20 18 66 1A 66 00 F0 | 4..n.!.f. .f.f..
[    739] [1138284309] [10000050] 2C F8 19 6E 19 6E 19 6E 05 20 00 F0 2F F8 01 21 | ,..n.n.n. ../..!
[    739] [1138285399] [10000060] 08 42 F9 D1 00 21 99 60 1B 49 19 60 00 21 59 60 | .B...!.`.I.`.!Y`
[    739] [1138286505] [10000070] 1A 49 1B 48 01 60 01 21 99 60 EB 21 19 66 A0 21 | .I.H.`.!.`.!.f.!
[    739] [1138287597] [10000080] 19 66 00 F0 12 F8 00 21 99 60 16 49 14 48 01 60 | .f.....!.`.I.H.`
[    739] [1138288693] [10000090] 01 21 99 60 01 BC 00 28 00 D0 00 47 12 48 13 49 | .!.`...(...G.H.I
[    739] [1138289779] [100000A0] 08 60 03 C8 80 F3 08 88 08 47 03 B5 99 6A 04 20 | .`.......G...j.
[    739] [1138290916] [100000B0] 01 42 FB D0 01 20 01 42 F8 D1 03 BD 02 B5 18 66 | .B... .B.......f
[    739] [1138292001] [100000C0] 18 66 FF F7 F2 FF 18 6E 18 6E 02 BD 00 00 02 40 | .f.....n.n.....@
[    739] [1138293071] [100000D0] 00 00 00 18 00 00 07 00 00 03 5F 00 21 22 00 00 | .........._.!"..
[    739] [1138294170] [100000E0] F4 00 00 18 22 20 00 A0 00 01 00 10 08 ED 00 E0 | ...." ..........
[    739] [1138295268] [100000F0] 00 00 00 00 00 00 00 00 00 00 00 00 74 B2 4E 7A | ............t.Nz
[    739] [1138296369] [10000100] 00 20 04 20 F7 01 00 10 C3 01 00 10 C5 01 00 10 | . . ............
[    739] [1138297463] [10000110] C1 01 00 10 C1 01 00 10 C1 01 00 10 C1 01 00 10 |
```

3) Display Pico's complete flash address space.

This time, the screen shot has been taken from EmEditor after opening the log file created with TeraTerm. We can see the beginning of the flash memory content. Keep in mind that even is the firmware is run from RAM, it has been saved to flash on the Pico in order to be available on next power-up. Obviously, if you erase all flash, Pico will automatically fall in "USB-drive" mode on next power-up.

4) Display Pico's complete RAM address space.

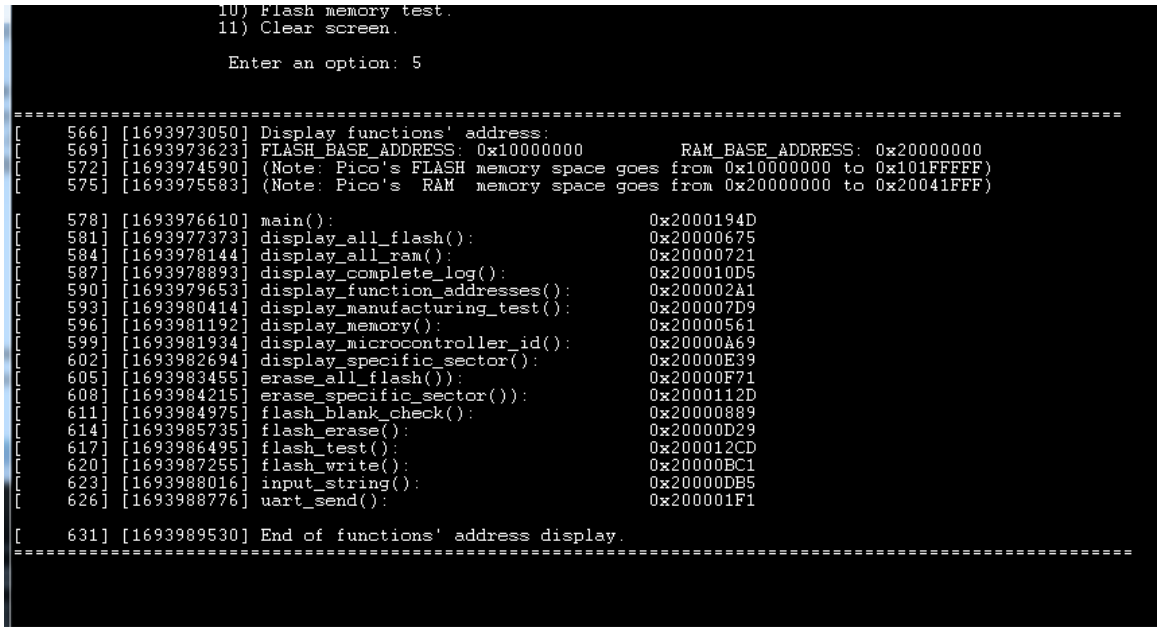
This function is kind of "out-of-context" in a flash utility, but since it was easy to add, I simply put a few more lines in the source code to cover it. Pico`s RAM is 264kBytes long and goes from 0x20000000 up to 0x20041FFF.



4) Display complete RAM address space

5) Display firmware functions address.

This function will display the start address of a most functions of the Pico-Flash-Utility. The goal is to confirm that the firmware runs from RAM (and not from flash). The obvious reason why we must run the firmware from flash is that if ever we erase all flash memory space (while we execute the code from flash), we are self destructing the code and we can expect strange behavior (and a crash) to occur at some point. Even if we don`t erase the complete flash memory space, if we erase a specific flash sector containing parts of firmware code, we can expect code instability and a crash, eventually.



5) Firmware functions address

NOTE: Even if running from RAM is required for the reasons given above, the Firmware will accept to run from flash after giving a warning to the user on power-up. However, it will refuse to proceed with a complete flash erase. (But accept to erase a sector containing its own code. Be aware of potential problems!).

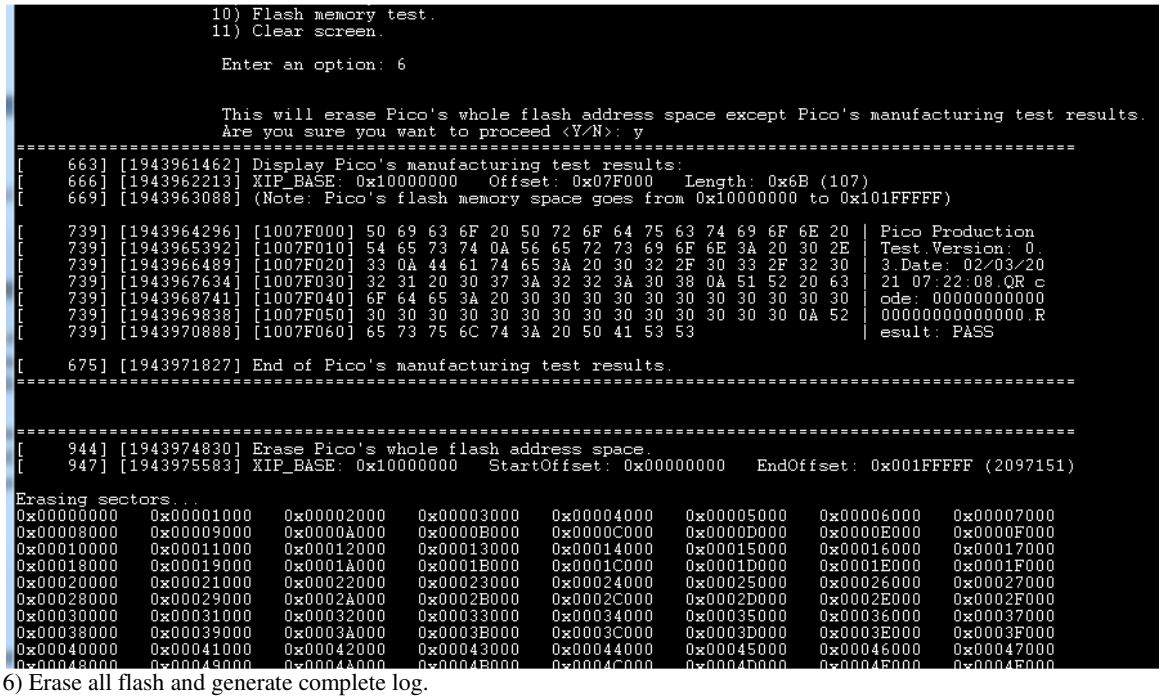
6) Erase all flash and generate Pico's complete log.

This function is like a "macro function" that will automate other functions of the menu. You can customize it to your needs by changing the source code and building the executable. Currently, this option will proceed with the following functions and in this order:

- Ask user to confirm that he wants to erase all flash memory space. (Once this has been confirmed, the other tasks can be run unattended).
- Display Pico's Manufacturing Test results.
- Erase all flash memory space.
- Flash memory blank check.
- Display all flash memory space.
- Display functions start address.

Since all the data will be saved in the terminal emulator program's log file, it can be used as a reference later.

NOTE: If user does not confirm his choice (first bullet in the list above), the system simply returns to the main menu without any further action.



6) Erase all flash and generate complete log.

7) Erase a specific sector of Pico`s flash.

This function will erase a specific sector of the Pico's flash memory. User is first asked which sector he wants to erase. As with other functions, sector offset must be enter in hex. The specified sector will first be displayed and user must confirm that he wants to proceed erasing the sector displayed.

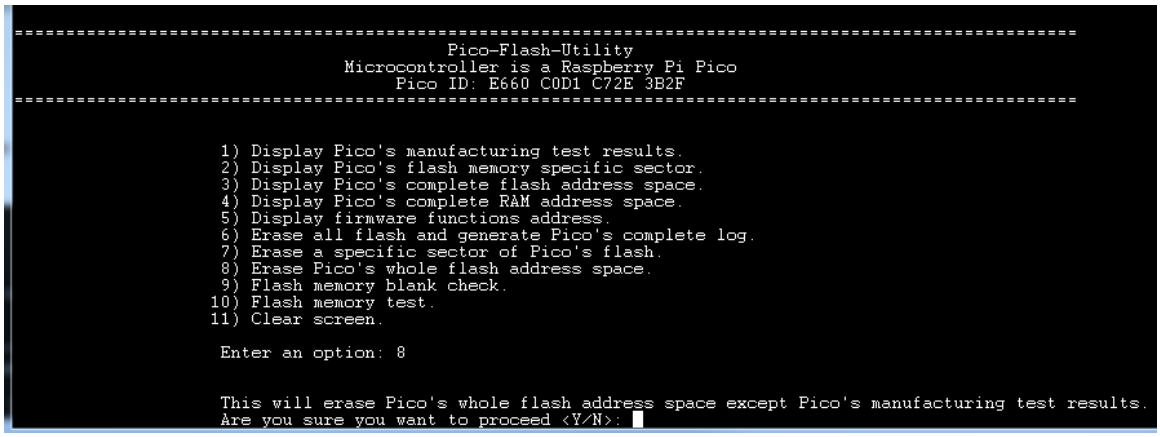
NOTE: As opposed to option 3, if user does not specify a valid sector offset (a multiple of 4096, or more simply, ending with three zeroes, the system will return an error message and ask to enter a valid sector offset.

```
 9) Flash memory blank check.
10) Flash memory test.
11) Clear screen.

 Enter an option: 7


 Enter offset of the sector to erase in hex (or <Enter> to return to menu): 7F000

================================================================================================
[   1002] [2139362550] Erase Pico's specific flash memory sector at offset 0x0007F000
[   1005] [2139363489] XIP_BASE: 0x10000000   Offset: 0x07F000   Length: 0x1000 (4096)
[   1008] [2139364391] (Note: Pico's flash memory space goes from 0x10000000 to 0x101FFFFF)

[    739] [2139365628] [1007F000] 50 69 63 6F 20 50 72 6F 64 75 63 74 69 6F 6E 20 | Pico Production
[    739] [2139366733] [1007F010] 54 65 73 74 0A 56 65 72 73 69 6F 6E 3A 20 30 2E | Test.Version: 0.
[    739] [2139367806] [1007F020] 33 0A 44 61 74 65 3A 20 30 32 2F 30 33 2F 32 30 | 3.Date: 02/03/20
[    739] [2139368920] [1007F030] 32 31 20 30 37 3A 32 32 3A 30 38 0A 51 52 20 63 | 21 07:22:08.QR c
[    739] [2139370018] [1007F040] 6F 64 65 3A 20 30 30 30 30 30 30 30 30 30 30 30 | ode: 00000000000
```

7) Erase a specific of Pico`s flash.

8) Erase Pico`s whole flash address space.

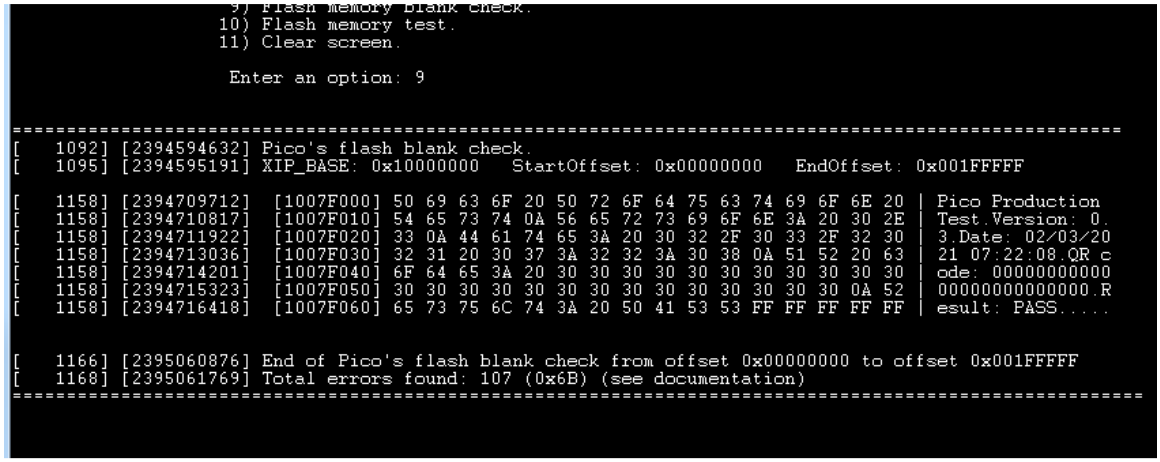
This function will erase all Pico`s flash memory space, after asking a confirmation from user.



8) Erase whole flash memory space

9) Flash memory blank check.

This function will check every byte of the flash memory space to make sure it is in a "blank" state (that is, equals to 0xFF). When some bytes are found not equal to 0xFF, they will be displayed with the same format as other displays seen so far. All consecutive bytes will be displayed on consecutive lines. If "blocks" of non-erased bytes are found, a blank line will be inserted on the display so that we can more easily track the location of those "blocks" of memory.



9) Flash memory blank check

10) Flash memory test.

This function will perform a flash memory test by executing five (5) write cycles on the whole flash memory.
For each cycle, those actions will be performed:

1) Flash memory space will be erased (with the exception of manufacturing test result).
2) A blank check will be done.
3) Byte 0x00 will be written to all flash memory space.
4) Flash will be read back to check if all bytes are 0x00 as expected.

5) Flash memory space will be erased (with the exception of manufacturing test result).
6) A blank check will be done.
7) Byte 0x55 will be written to all flash memory space.
8) Flash will be read back to check if all bytes are 0x55 as expected.

9) Flash memory space will be erased (with the exception of manufacturing test result).
10) A blank check will be done.
11) Byte 0xAA will be written to all flash memory space.
12) Flash will be read back to check if all bytes are 0xAA as expected.

13) Flash memory space will be erased (with the exception of manufacturing test result).
14) A blank check will be done.
15) Consecutive bytes 0x55 and 0xAA will be written to all flash memory space.
16) Flash will be read back to check if all bytes are 0x55 and 0xAA as expected.

17) Flash memory space will be erased (with the exception of manufacturing test result).
18) A blank check will be done.
19) Consecutive bytes 0xAA and 0x55 will be written to all flash memory space.
20) Flash will be read back to check if all bytes are 0xAA and 0x55 as expected.

NOTES:

- If an error occurs at any step during the process, it will be reported (in the log file) and the process will continue.
- Memory writes to Pico`s flash memory is limited to more or less 100,000 cycles. So, you can use the Pico-Flash-Utility as you want. However, it is probably not a good idea to modify the code and make it a “burn-in” process by letting it loop and runs for hours continuously!! No wear levelling algorithm has been implemented in the Pico-Flash-Utility since its purpose is precisely to test the good working order of every byte of flash.
- As already mentioned, the 107 bytes of the Pico’s manufacturing test results will be kept intact in all functions, including flash erase and flash writes. As a consequence, it is normal that the firmware reports 107 errors at each flash erase and each pattern flash write during the test. Globally, 107 errors for each flash erase, 107 errors for each pattern flash writes, 5 patterns to write per cycle, for 5

cycles will give 107 * 2 * 5 * 5 = 5350 errors total. So, it is normal that you see 5350 errors reported at the end. Also, if ever you have overwritten the Pico`s manufacturing test result flash area with some other code in the past, it is possible that this contains 0x00s, 0x55s, 0xAAs or 0xFFs. If ever the case, blank check and / or match check may give error numbers different than 107 for each step...

- While the flash memory test is executing, if you use a Pico (not applicable to the Pico W), you will see the Pico`s LED blinking a number of times indicating what is write cycle currently executing (out of the 5 write cycles). This blinking will occur more or less every 15 seconds. (The callback is set to 5 seconds, but since flash erase and flash write disable interrupts - and there are a lot of flash erase and flash write in this piece of code – I've seen the period vary from 15 up to 20 seconds...)

11) Clear screen.

This function does not require a lot of explanations... Simply a few words, however, about the "cls" tag that can be seen in the uart_send() function. When used, this "cls" tag works fine, but there are side effects that appear later on. I suggest that you use the code that I put in the "switch" statement (a simple "for" loop with linefeeds).

Appendix A – Configuring a terminal emulator program

Pico to computer, USB-to-USB

It is relatively easy to communicate between the Pico and your computer with a USB-to-USB connection. If you modify the source code, you must make sure that the Pico-Flash-Utility "make" file (CmakeLists.txt), contains the following lines:

Pico_enable_stdio_uart(Pico-Flash-Utility 0)
Pico_enable_stdio_usb(Pico-Flash-Utility 1)

The first line will stop sending the displayed data through the Pico's UART and the second line will make it sent through the Pico`s USB port.

IMPORTANT: For the PC to "see" a CDC USB port active, the Pico must be connected and the Firmware running (the PC will not "see" a CDC USB port if the Pico is in "USB drive" mode – that is, after you pressed the Bootsel button and it is waiting for the Firmware to be uploaded). For this reason, you must upload the Firmware to the Pico and let it starts before starting the terminal emulator program. If you start TeraTerm while the Pico is in "USB drive" mode, TeraTerm will display an error message since no serial device is found.

Then start a terminal emulator program on a PC. As indicated in the Introduction section, I use TeraTerm on Windows that I find very useful (and which is freeware and open source), but there are terminal emulator programs for all popular platforms (Windows, Linux, Mac).

https://download.cnet.com/Tera-Term/3000-2094_4-75766675.html

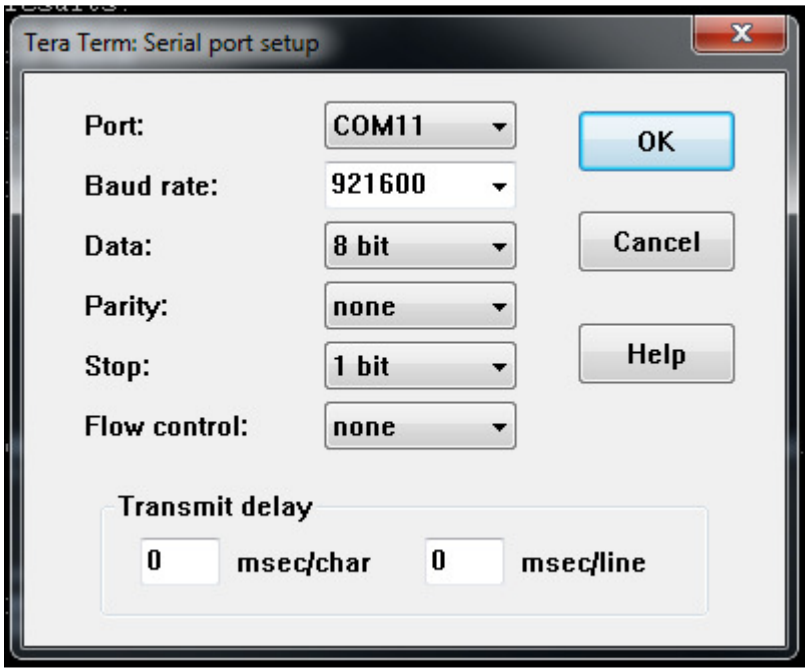
If you use TeraTerm, start it and go to the "File / New connection" menu. You will see that there is a "Serial" option and a serial port number should be assigned to the serial-to-USB (or CDC USB) connection (you may also check in Windows Control panel / Device manager / Ports (com and lpt). You should see an entry for CDC, along with a COM port number assigned to it.

NOTE: On some older Windows versions, you may need to install a special CDC driver for the terminal program to "see" the CDC USB. Complete explanations are beyond the scope of this Guide, but you can find information about this on the Internet.

NOTE: If you don't see a COM port in Windows Device manager, you may try to search for "Zidag" on the Internet and get more information.

NOTE: Be aware that if you change the Pico unit for a different one (both being Picos or both being PicoWs or both being mixed), the COM port number may change from what it was before. It's usually not complicated to change the COM port number in the terminal emulator, but this may cause a surprise the first times you realize that.

NOTE: You must make sure that both ends (the Pico`s Firmware on one side and the terminal emulator program on the other side) are configured the same way for the serial communication. If you use TeraTerm, click on "Setup / Serial port". You will see this popup with the serial parameters. They must conform to those used in Pico`s Firmware. (the parameters in the figure below are the actual settings of the uf2 version posted in GitHub, with the exception of COM11 who must rather match the CDC USB that has been identified by your system).



Serial port settings of the Pico-Flash-Utility Firmware on GitHub

Once the serial settings have been configured, you may "Setup / Save setup" (on TeraTerm), so that what has been configured remains the same next time you start the terminal emulator program.

If you take a look at the source code, you will see that when the Firmware starts, it waits for the serial communication to be established before sending data to the terminal emulator. Once the communication has been established, the firmware automatically detects it and display the menu on the terminal.