APPLE-1 TERMINAL

v1.0

1. PROJECT DESCRIPTION

The *Apple-1 Terminal* is a device that allows you to fully operate your Apple-1 Original, Full-Size Replica or Briel Computers Replica-1 Plus **via a USB serial terminal**.
You will be able to operate your computer without the need for an external keyboard or monitor as the computer's output will be replicated on the terminal.
The board also adds other interesting features such as the **management of ANSI escape codes** and much more.

The possible applications for such a solution are many, for example:
- Troubleshooting, both on the computer and terminal section,
- Remote use of the machine,
- Development of programs and applications by using or not using the additional functionalities.

The board is designed to be inserted on either the vertical or horizontal connector. In the latter case it will be possible to continue using additional cards such as Juke-Box, microSD card, Wi-Fi modem, etc.

If you wish to use this option, the EDGE connector must be mounted in correspondence with the white frame, on the opposite side of the components, which must therefore always face outwards.
For the insertion direction, also refer to the guide numbers/letters 1, A and 22, Z, which must correspond to those on the motherboard.

To function, the board **does not need any specific computer modifications or setup** and is compatible with any memory configuration.

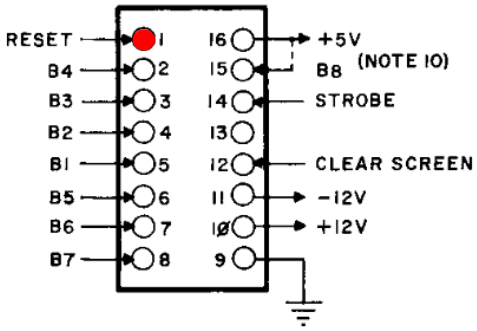
After assembling/building the circuit, program the Arduino Nano microcontroller with the source code available on the reference page. You may be asked to install the EEPROM library.

WARNINGS:

- NEVER leave the USB cable connected with the Apple-1/II/II+ computer switched off.
- Take extreme care to the construction and orientation of the flat cable connectors. Incorrect construction or incorrect insertion will result in **damage to the computer** and the adapter.
- Apple-1 is not a machine designed to handle multiple expansion cards. Use of this card in conjunction with others may under some circumstances result in malfunction and loss of data.

2. KEYBOARD CONNECTION

Apple-1 Terminal interacts with the computer via the existing keyboard interface, which also provides the essential CLEAR SCREEN function. The pin layout of the board matches exactly that on the computer.
With the computer switched off, connect the terminal board and the computer with a properly headed flat cable.



The red dot marks pin number 1. On the computer's motherboard and on the adapter the

A mistake in such connections can lead to immediate and permanent damage to the terminal board and the computer: always check your work more than once.

Once the PCB and flat cable have been assembled, you should check that no errors have been made in the construction by following these steps:

- With the computer switched off: connect the flat cable to the "KEYBOARD" socket,
- Remove microcontroller and all integrated circuits if present,
- Connect the terminal board to the designated connector,
- Connect the other end of the flat cable to the terminal board,
- Switch on the computer,
- Check for the presence of +5V between pin 20 (VCC) and pin 10 (GND) of the 74HCT374.

If this voltage is present, turn off the computer and proceed with assembly, otherwise check the wiring and soldering.

The device is sensitive to static electricity, just like your Apple computer, and may be damaged by it.
Before any operation on your devices, you must discharge the static electricity accumulated by your body and prevent it from building up again.
We do not accept any responsibility for damage, even serious or fatal, caused to people / things / intellectual property during the installation or use of this device.

3. OPERATIONS

Switch on your Apple computer first, then connect the USB cable.
Set up the terminal program on your home computer with the following parameters:

Baud rate: **230400**, **8** data bits, **1** stop bit, No parity.

The reasons for this particularly high baud rate will be explained later.
The following summary message should appear:

```
Apple-1 serial terminal by P-LAB
--------------------------------

CTRL-L ==> Clear screen
CTRL-R ==> Reset
CTRL-O ==> Toggle:
Uppercase for OUTGOING characters: ACTIVE
CTRL-I ==> Toggle:
Uppercase for INCOMING characters: not active
```

(please note: incoming/outgoing status may be different)

From now on, it is possible to use the Apple computer in the traditional way, as if you were in front of its keyboard and monitor.
The commands above are intuitive, but the main two are summarized here:

- CONTROL-L -CLEAR SCREEN- (both native and terminal)
- CONTROL-R -RESET- reset the computer

Then enter the desired commands (typically CONTROL-L and CONTROL-R) and operate normally.
Lowercase/uppercase commands are described in the following section.

4. UPPERCASE and lowercase

The *native* Apple-1 video terminal has several limitations, one of which is certainly the impossibility of displaying lower case text natively on the screen.
Both the computer and INTEGER BASIC, however, are **perfectly capable of distinguishing, storing and using lower case letters**.

On our terminal, and only on it, you will be able to print strings and lowercase text on the screen as you prefer. It will continue to be displayed in uppercase only on the monitor connected to the computer.

Commands to the WOZ monitor, instructions, BASIC statements etc. must always be typed in CAPITAL letters or they will not be interpreted.
To simplify operations, two additional entries have been provided which appear at start-up: CONTROL-I and CONTROL-O. They operate in toggle mode, i.e. they activate or deactivate the desired function with the same command.
The operating status is displayed by a short on-screen message.

Description of functions:

- CONTROL-I -INCOMING- forces everything sent from the computer to the terminal to be displayed in upper case. If any content is stored in lower case, it will appear in upper case on the terminal.
  This function actually mimics the behavior of the original screen terminal.

- CONTROL-O -OUTGOING- forces the upper case translation of alphabetical characters typed in lower case by the terminal before sending them to the Apple computer. This makes it possible to enter, for example, strings containing lowercase characters.

Your preferences are saved in the microcontroller's EEPROM, so the next time you switch the device back on, you will find them as you left them.

The combination of these two commands can lead to errors, especially if you have forced 'lower case' for sending and 'upper case' for receiving.
In this case, commands and instructions will be transmitted in lower case, displayed as upper case, but will not be interpreted or will return an error message.

5. SPEED, SPEED, SPEED

Another strong limitation of the native video terminal is the speed in writing on the screen, which is **1/60 of a second per character** written on the screen.
What we have seen so far, while convenient, makes using the computer via terminal exactly the same as using a native keyboard and monitor. The terminal board, however, offers something more.

It can be much faster than this, if one instructs programs and applications not to use the WOZ monitor's screen-printing routines or, alternatively, not to use the PIA (Peripheral Interface Adapter - the large 6820 chip next to the 6502 microprocessor) at all for screen writing.

The terminal board, in addition to 'eavesdropping' on the characters that the microprocessor asks the PIA to print on the screen, also listens for direct writes, and it is thanks to this possibility that **terminal writing can be speeded up enormously.**
Since the card does not have to wait all that time for a character to be printed on the screen before accepting the next one programs can poke to this address range to display their output.
Needless to say, in this mode of use, the main monitor will not display anything beyond what is already present.

Many tests have been conducted (extreme cases, of no practical use) on the maximum speed that can be achieved, which far exceeds the maximum number of interrupts and writes to the serial port that the microcontroller can handle.
However, programs written in BASIC (therefore interpreted) are very well suited to this optimization since screen writes are the result of many operations and are therefore sufficiently ‘diluted in time by design' not to represent a major problem for the microcontroller.
It was decided to try modifying INTEGER BASIC because of the large number of programs available: by modifying the interpreter, all programs will benefit.

The FTBASIC (Fast Terminal) version of INTEGER BASIC on the repository https://p-l4b.github.io/terminal is the usual BASIC as always, but instead of writing to the screen via the PIA, it writes to the terminal via terminal board.

The modification was implemented by **Antonino Porcino** and the additional routines did not affect the size of the program, which remains at 4096 bytes. He has in fact used memory areas not used by the original code.
The 230400 baud rate is the safest to avoid losing characters even in the most inconvenient conditions

So if you want to use the same games and programs as always, but much faster, simply load FTBASIC from your **microSD card** (or from the audio file provided on the same page) instead of the classic BASIC.

Visit the GitHub repository mentioned above from time to time to find out if other interpreters or languages have been 'speeded up'!

Machine language programs can also be speeded up. Typically, screen writes are performed by calls (JSR -Jump to SubRoutine) to the WOZ monitor's famous ECHO routine, which responds to the address $FFEF:
`JSR $FFEF → 20 EF FF`

Since both the ECHO and the serial card display the contents of the Accumulator (the ASCII code of the character we want to write on the screen) it would be natural to replace these three bytes with a write of the Accumulator to the mapped address of the serial card $DF12:
`STA $DF12 → 8D 12 DF`

However, if the code makes too many consecutive writes, **characters may be lost**, even at high baud rates (e.g. 2000000 bit/s).
As an example: the ASCII character test program needs 20 NOP instructions (No OPeration - for a total of 40 clock cycles) after the STA instruction to avoid overlapping between the interrupt coming from the Address Decoding network and the write to the serial port.
These timings will be explained in more detail in the last section.

In the case of 'speeding up' existing, or new, machine language programs, watch out for possible malfunctions in writing to the screen.
If you have problems with space to allocate these delays, also consider using the PHA (PusH Accumulator) and PLA (PuLL Accumulator) instructions, which together use 7 clock cycles with only two bytes taken up in memory.

6. CURSOR CONTROL – ANSI ESCAPE CODES

The last limitation that the terminal board overcomes is cursor control, namely **the ability to write at a certain position on the screen**.
The native video terminal is more like a teletypewriter than an actual terminal: the only difference is that it does not use paper but otherwise it is equivalent: what has been written cannot be wiped from the screen, the cursor only goes forwards, and there is no function/call for clearing the screen as this is done at hardware level.

ANSI escape codes are predefined and well-coded sequences of ASCII codes that, once interpreted by the terminal, allow the **manipulation of writes and deletions, character font, color and position** of what is being written.
The name 'escape' is derived from the ASCII code of the ESCAPE key, which is 27 in decimal.
In fact, all escape code sequences begin with 27 decimal or $1B hexadecimal.

Your terminal program must be set up to be compatible with this standard if it is not.
Check the settings for compliance with the ANSI standard (or compatibility of the emulation with the VT-100 standard).
The programs: "minicom" for Linux, "PuTTY" and "Teraterm" for Windows are usually compatible by default.
Once the terminal has been set up, the parts of the text transmitted by the Apple computer that represent valid escape codes will not be displayed by the terminal but rather **interpreted to obtain the corresponding effect**.

There are dozens of escape codes, sometimes dedicated to specific types of terminals, and a great amount of documentation exists on the subject. This may be a starting point to explore the subject further:
https://www2.ccs.neu.edu/research/gpc/VonaUtils/vona/terminal/vtansi.htm

By means of the appropriate code sequence, the microcontroller's firmware is therefore able to **reproduce the Clear Screen function** as well, replicating the computer's true behavior.

In this specific case, the microcontroller will send two commands (<ESC> represents escape character 27):

`<ESC>[2J` clears visible screen
`<ESC>[H` set cursor position in the upper left corner (Home)

Keep in mind that there may be differences in behavior between terminals with the same escape codes. The program "minicom", for example, interprets the first command as both "clear screen" and "place cursor in HOME". "PuTTY" and "Teraterm", on the other hand, also need the second command to place the cursor in HOME.

Another example: to position the cursor in a specific *row/column*, the command can be used:

`<ESC>[row;columsf`
(Please note: the final letter 'f' is part of the command)

The ESCAPE character, on Apple-1, can be generated and sent to the serial board by a simple instruction that writes the value $1B to address $DF12 (or, in INTEGER BASIC, pokes the value 27 to location -8446)

This finally makes the **Clear Screen function available to programs**.
In INTEGER BASIC, all that is needed is a few simple instructions, which can be also called be GOSUB/RETURN:

```
10 POKE -8446,27
20 PRINT "[2J";
30 POKE -8446,27
40 PRINT "[H";
50 PRINT "A CLEAR SCREEN"
```

Cursor control is also possible, of course. To write on line 3, column 5 (let us assume the use of the variables `R=3` e `C=5`) instructions will be:

```
10 POKE -8446,27
20 PRINT "[";R;";";C;"f";
30 PRINT "THERE YOU GO"
```

Note that:

- PRINT instructions for escape codes end with a ';' to prevent the 'true' PRINT from carriage return,
- the last letter of the escape code in the cursor control example is a lower case 'f'. It must therefore be written using the appropriate combination of CONTROL-I and CONTROL-O.

The possibility of using escape codes opens many ways for those who wish to write advanced programs and games that go beyond the normal capabilities of the native video terminal.

With the serial terminal, the 40x23 screen constraint also falls away. In fact, you can resize the screen and use it the size you prefer for your programs!

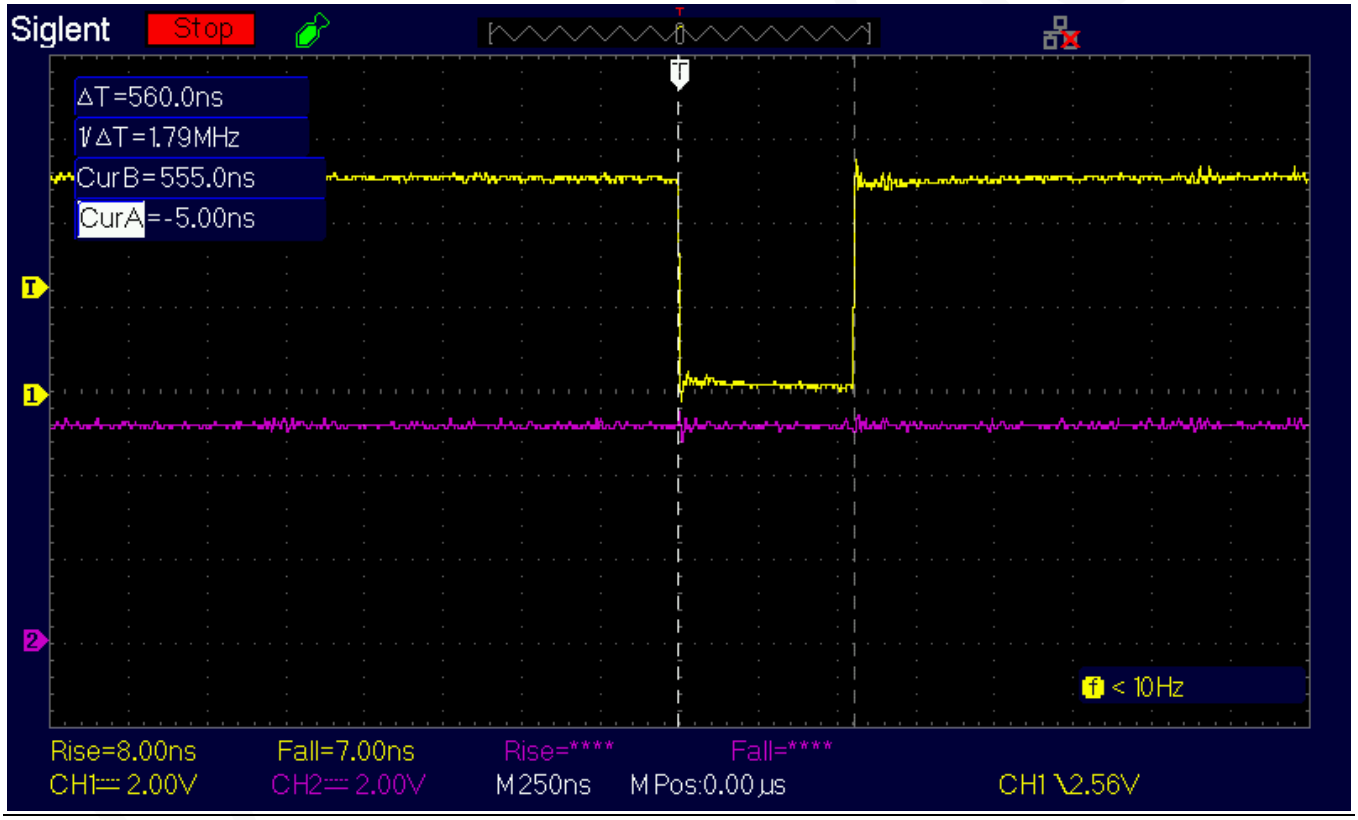
7. AC CHARACTERISTICS

This small collection of pictures and figures is intended to measure the time taken by the board to process the information received.
The measurement points used are:
CHANNEL 1 - YELLOW TRACK, interrupt signal from Address Decoding logic, Arduino pin "D2"
CHANNEL 2 - PURPLE TRACK, serial data transmission, Arduino PIN "TX"

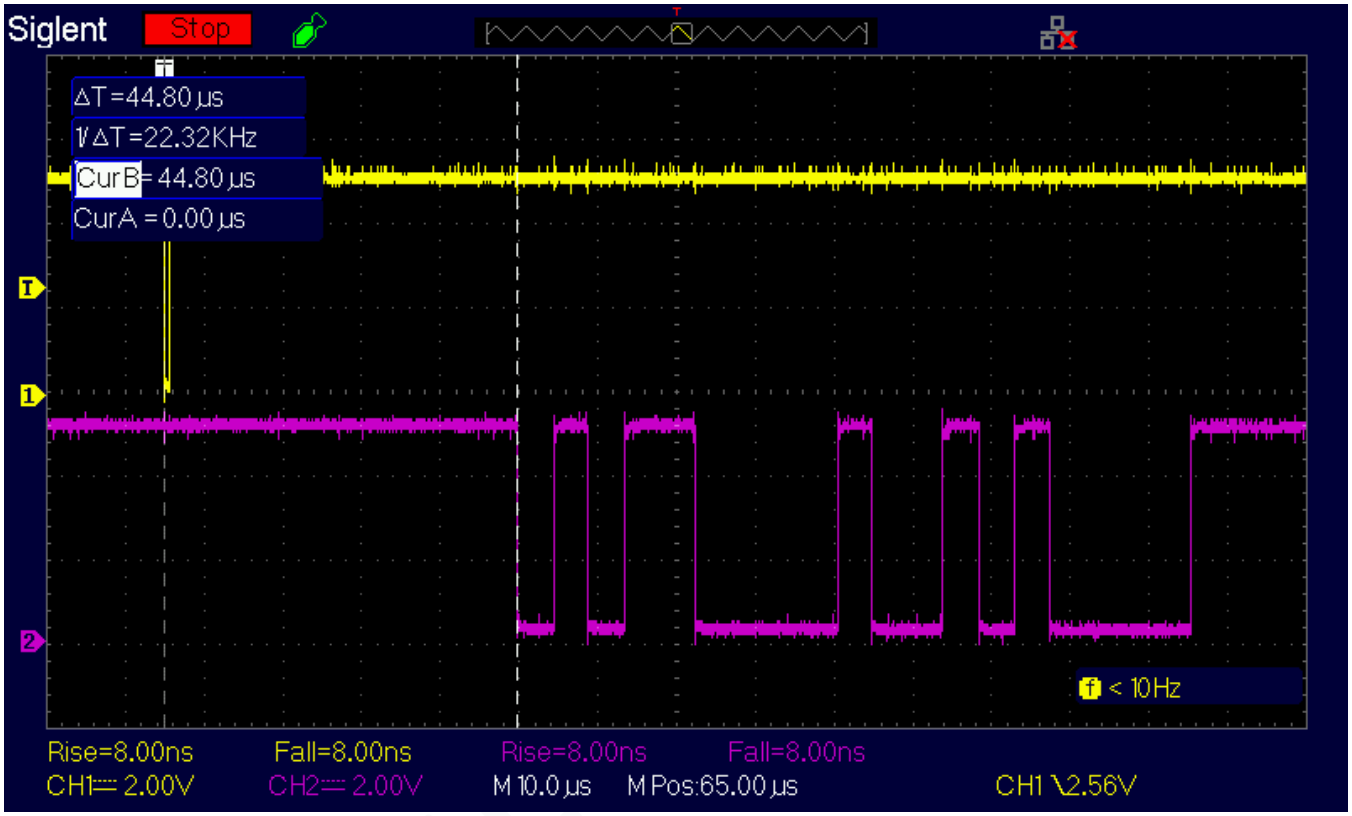
The ATMEL 328P microcontroller specification states that the minimum time to detect an interrupt is 4 clock cycles. At 16 MHz, the microcontroller's clock frequency, this is equal to **250 ns**.

In the following pictures, the vertical dotted lines represent the measured time interval.

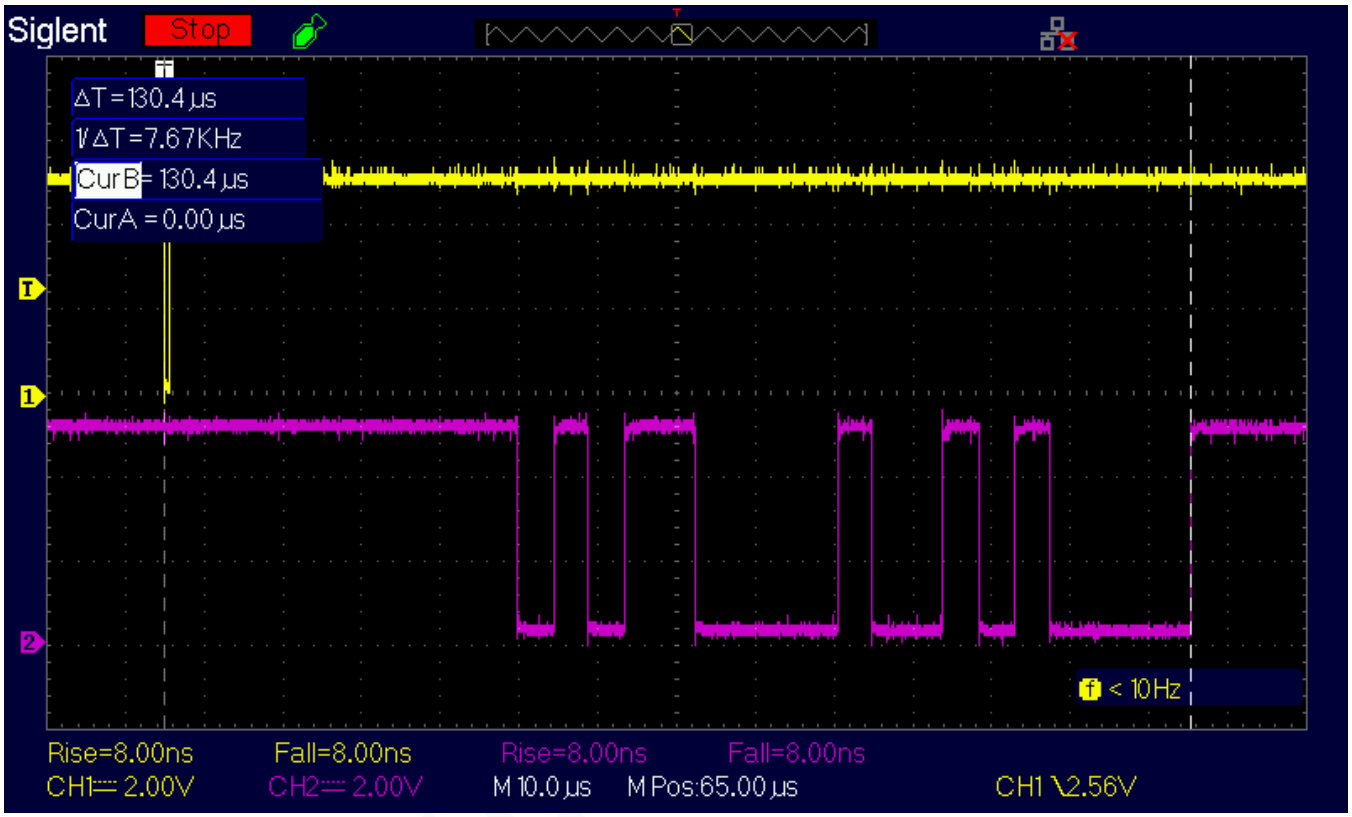
Interrupt signal from Address Decoding logic**: 560 ns** (greater than the minimum threshold of 250ns):



Microcontroller interrupt handling plus processing time, before starting transmission at 230400 baud: **~45 µs**

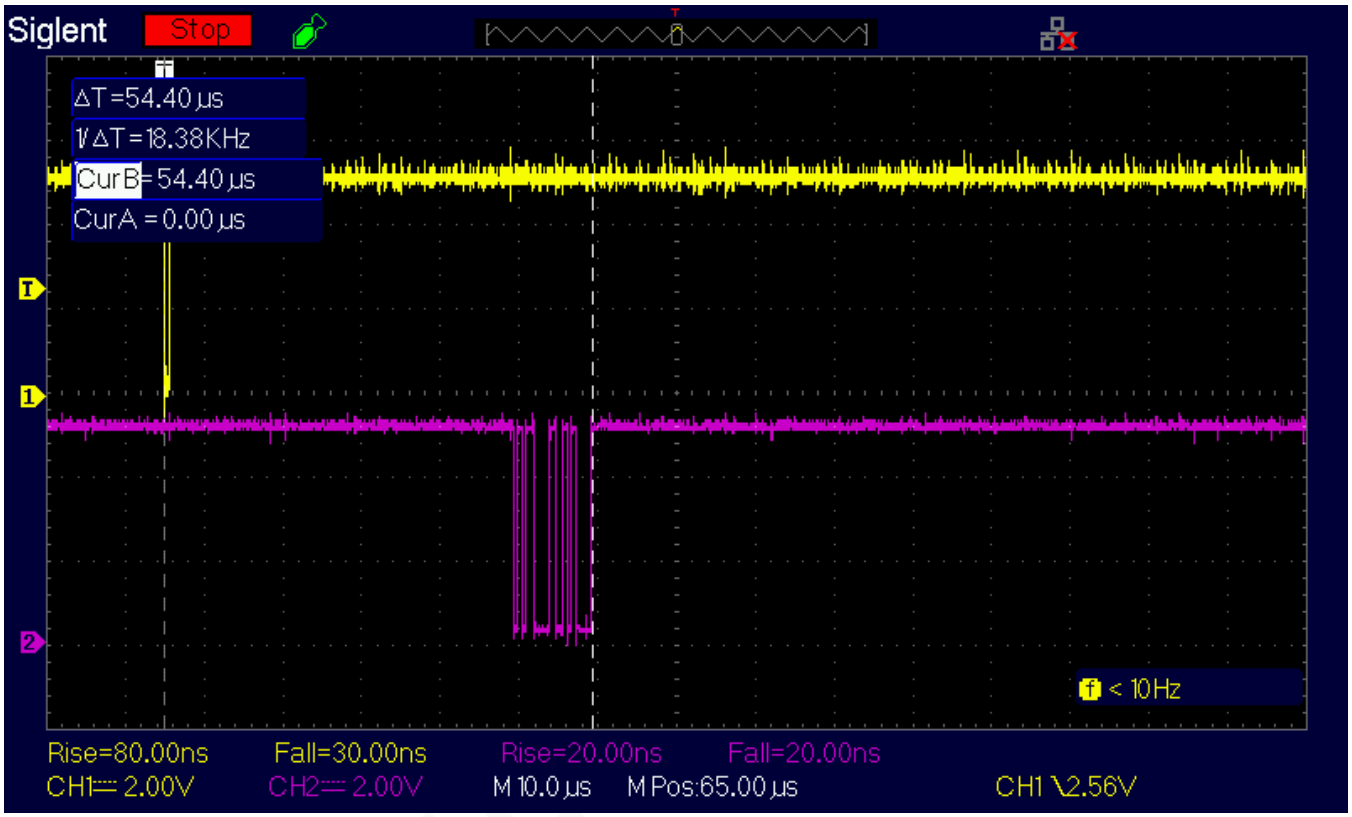


Total interrupt handling time and process time, added to serial transmission at 230400 baud: **130.4 µs**



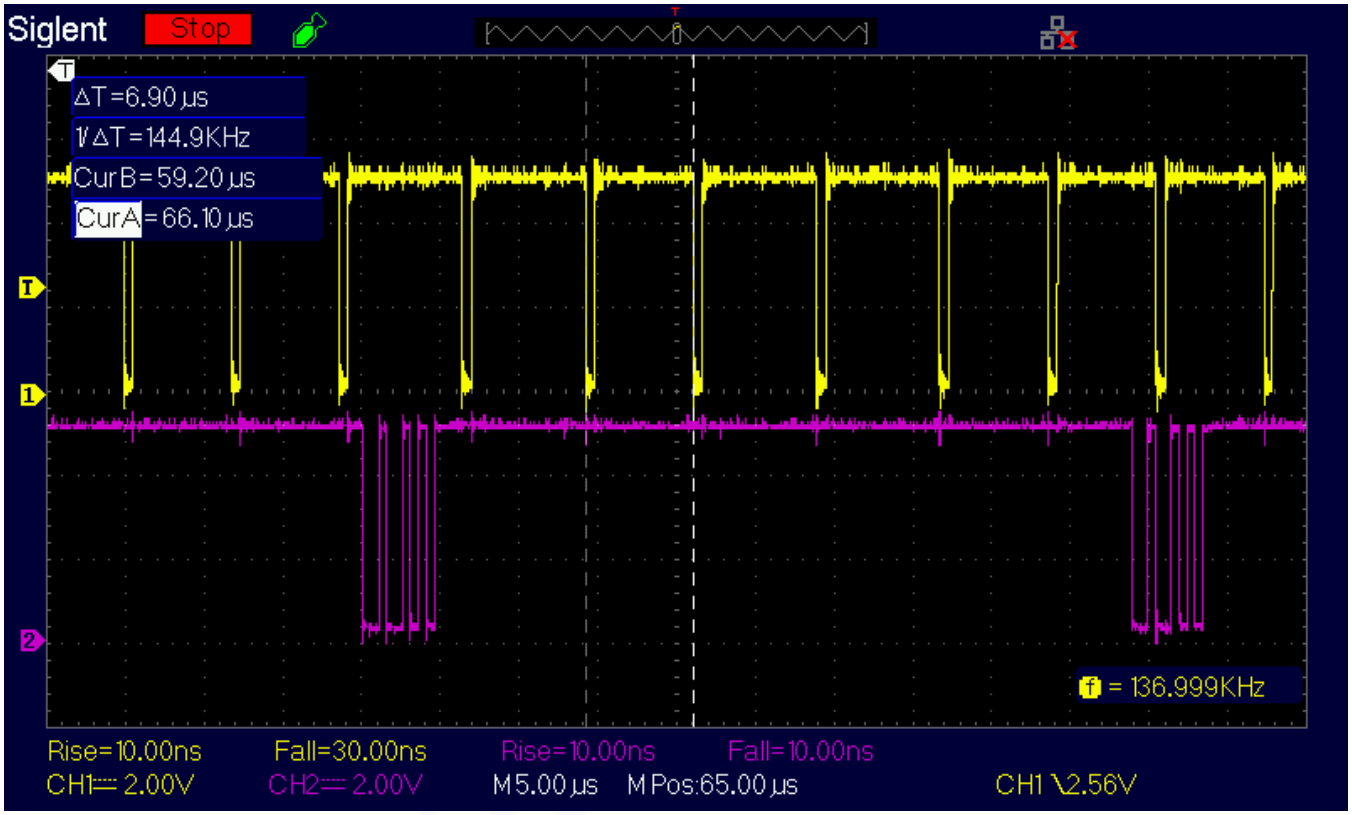
This is the worst case because pressing the ENTER key generates two characters on the serial port: CR (13 decimal, $0D in hexadecimal) and LF (10 decimal, $0A in hexadecimal).

Total interrupt handling time and process time, added to serial transmission at 2000000 baud: **54.40 µs**



This is the worst case because pressing the ENTER key generates two characters on the serial port: CR (13 decimal, $0D in hexadecimal) and LF (10 decimal, $0A in hexadecimal).

Overlap between interrupt signals and serial transmission at 2000000 Baud: loss of characters.



We hope you enjoy using *Apple-1 Terminal!*

APPLE-1 TERMINAL

INFO | ORDINI | SUPPORTO: p-l4b @ protonmail.com

P-L4B @ PROTONMAIL.COM