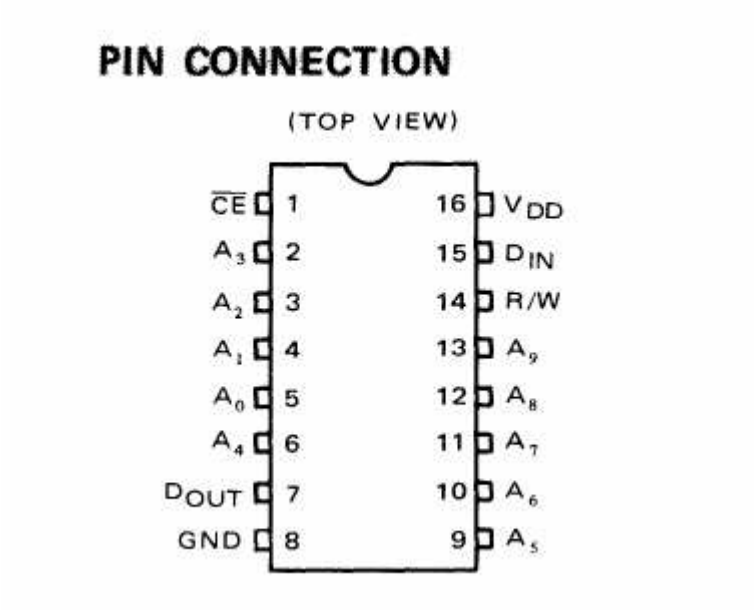
Casio PRO-101 TC5006P RAM Replacement

This modification to the Casio PRO-101 replaces the TC5006P RAM ICs with an RP2040 microcontroller. The TC5006P devices are used to store the program steps of the calculator. They are low power devices and can be powered by three button cells when the main circuitry of the calculator is powered off. This provides the continuous memory function of the PRO-101.

The RP2040 is a dual core ARM based microcontroller. It is not a low power device so storage of program steps cannot be done using the three button cells as the original circuitry did.

Program Memory

The PRO-101 has 256 steps. Each step is stored in a byte. The TC5006P is a 1024 x 1 bit device. It can store 128 bytes and so two devices are used to store the 256 steps. There is little information about the T5006P to be found, but it turns out that the later TC5508P device has the same pinout. A datasheet is available and the pinout is shown here:



The device has a data in signal and a data out signal, which is not unusual for devices made in the 70s and 80s. There are address lines, a R/W signal and a chip select. The RP2040 reads the address lines, CE line, data in line and R/W line to work out what data to put on the data out signal, and when. It monitors both of the CE signals that went to the two original chips and emulates both of them. This isn’t difficult as they are on the same bus and are never accessed at the same time.

The RP2040 has 264K of RAM and so can easily emulate the 256 bytes of program space. The RP2040 RAM is accessible by both ARM cores, but simultaneous access can cause contention and stalling of a core.

Hardware Modifications

The hardware of the calculator has been altered as little as possible. The two TC5006P chips have been removed as they were faulty and could no longer store programs. One of the TC5006P footprints has been fitted with a custom PCB with an RP2040 Zero type module. There is a flying lead from the second TC5006P footprint to provide the second CE signal.

The RP2040 Zero is powered either from the PRO-101 TC5006P supply or from USB. The modified unit does not use the button cells and they should not be fitted to this modified calculator.

The power consumption of the modified calculator is higher than the original by a factor of about two to three. I have used rechargeable AA batteries in the calculator for many hours while developing the code for the RP2040, so this calculator is still portable.

Interaction with the code running on the RP2040 is either through the calculator keyboard (details in another section) or a USB command line interface (CLI). There are not very many ways to get the USB cable out of the calculator. I ended up removing the A adapter socket and running the cable through the resulting hole. The battery compartment couldn’t be used to run the cable out as the battery cover couldn’t then be fitted and the batteries tended to fall out. A wireless (Wi-Fi) interface was prototyped but the module used couldn’t be powered by the calculator. The power in the calculator is quite complicated and no easy, safe way to run the Wi-Fi or Bluetooth interfaces could be found. I didn’t want to run the risk of damaging the power circuitry of the calculator. The Wi-Fi interface code is still in the source, but compiled out at the moment.

Memory Emulation

One of the RP2040s performs the emulation of the TC5006P. It has interrupts turned off as it has to respond to the signals on the bus promptly. It does not perform any task other than RAM emulation.

The second core performs all of the interface functions for USB, Wi-Fi and the calculator keyboard interface.

The second core (core0) accesses the emulation RAM of the emulation core (core1) very carefully. If it reads the RAM in bursts then core1 can be stalled and corruption of the emulation RAM can result. For this reason there can be a delay in the user interface before changes to the emulation RAM are seen. This is of the order of 2 or 3 seconds. This must be borne in mind when interacting with the RP2040.

Flash Storage

The same flash memory of the RP2040 that is used to store its firmware is used to store programs for the PRO-101. Flash is non volatile so these programs will still be stored even with power off and no batteries in the calculator.

The RP2040 Zero has 2M bytes of flash, 1000K bytes of this is used to store programs. There are 250 4K slots that can hold a program and a text label.

Emulation memory can be stored in one of these slots and recalled later. Labels for the slots can be set up over USB.

Programs can be saved and loaded to and from flash using keyboard commands, but that does involve using a few steps as a command. To save and load the full 256 bytes the USB interface has to be used.

Flash slots can be erased if needed. There is no way to erase all of the slots in this version of the firmware. Updating the firmware itself does not erase the data stored in the flash slots.

USB Interface

The USB interface starts with a sign-on message and is then menu driven. The sign on is here:

```
/------------------------------\
| Casio PRO-101                |
| TC5006P Pico                 |
/------------------------------/

Setting GPIOs...
Initialising UART RX...
Starting PIO UART RX interrupt...
UART RX running...
UART TX initialised...
```

This is a list of parts of the firmware as they initialise.

After the sign on message, press either ‘h’ or ‘?’ for the menu help display:

```
h: Serial command help
?: Serial command help
s: Save emulation RAM to flash slot
u: Send text to uart
I: Information
l: Load emulation RAM from flash slot
z: Zero parameter
E: Erase program slot
D: Display program slot
S: Display slot buffer
d: Dump RAM
L: List Programs
#: Set Slot Number
A: Set Address
a: Display All Programs
T: Toggle Write Protect
W: Write Byte
t: Enter Text Parameter
B: Label Slot
b: List Slot Labels
@: Display trace buffer
```

```
!: Boot to mass storage
```

Each of the other options is described below:

s: Save emulation RAM to flash slot

Saves the emulation RAM contents to the flash slot number ‘parameter’. The parameter value can be set up by typing numeric characters at the menu prompt. It’s current value is displayed in the menu prompt. The parameter can be cleared using the ‘z’ command.

This command should be issued 2-3 seconds after any changes have been made to the emulation RAM.

u: Send text to uart

This sends the text parameter to the UART. It can be used to send commands to a wireless module (Wi-Fi and/or Bluetooth) if it is fitted. This is not used at the moment.

I: Information

This option displays various information about the firmware.

l: Load emulation RAM from flash slot

This option loads emulation RAM from the slot whose number is in the numeric parameter.

z: Zero parameter

Sets the numeric parameter to zero. Digits can then be used to update the numeric parameter.

So, for instance, to put 1328 in the numeric parameter you would use the keys:

z 1 3 2 8

The numeric parameter is displayed in the menu prompt.

E: Erase program slot

Erases the flash slot whose number is in the numeric parameter.

D: Display program slot

Displays the flash slot whose number is in the numeric parameter.

For example, to display the contents of slot 200 use the keys sequence:

z 2 0 0 D

S: Display slot buffer

This displays an internal buffer.

d: Dump RAM

This displays the emulation RAM as hex program tokens and also as keystrokes. The entire 256 bytes of emulated memory is displayed. The order of the bytes is not as they are stored in memory, the ‘reversed 8 bytes’ scheme is reversed to make it easier to view the memory.

L: List Programs

Displays the emulation RAM as keystrokes.

#: Set Slot Number

This is not used in the current firmware.

A: Set Address

Sets the Address parameter to the value of the numeric parameter. The address parameter is used when writing bytes to the emulation RAM.

a: Display All Programs

Displays all program slots in flash. Blank entries are listed as ‘Blank’. Any slot with a program has the program displayed as keystrokes, if there is a label then it is displayed with the slot number.

T: Toggle Write Protect

A write protect feature for the program was added but this revealed the fact that the PRO-101 re-writes the program memory for a lot of reasons. Setting the write protect to ON will cause errors on the PRO-101 quite easily. I have left it in, in case it is useful in the future.

W: Write Byte

Writes a single byte to the emulation RAM. The address to write to is set up in the Address parameter and the data is in the numeric parameter. All parameters are decimal.

For example, to write 170 to address 128, the following keys would be used:

z 1 2 8 A z 1 7 0 W

There is no ‘Read Byte’ option as the ‘d’ option dumps all of the emulation RAM so performs that function.

t: Enter Text Parameter

Allows a text parameter to be entered. This is used when labelling slots. The label entry is ended when the ENTER key is pressed. I have noticed that this menu option sometimes ‘sticks’. I do not know why, but pressing a key on the PRO-101 sometimes ‘unsticks’ the entry of a label.

B: Label Slot

Labels the slot whose number is in the numeric parameter with the string that is in the text parameter.

For example, to label slot 123 with ‘Test slot’ the following keys are used:

z 1 2 3 t T e s t s l o t <ENTER> B

b: List Slot Labels

Displays all of the slots that have labels. Blank slots are not displayed at all in this list.

@: Display trace buffer

Displays the trace buffer that is continuously storing accesses to the emulation RAM. Reads and writes are stored and the buffer is currently 1000 accesses long. An access is a single bit read or write, as that is the atomic action for the RAM. These one bit accesses are coalesced into byte accesses when displayed with this menu option.

This option can be used to view programs while they are executing, and also to view the memory re-arrangement that the PRO-101 does at various times when viewing programs.

!: Boot to mass storage

Boots the RP2040 and puts it into mass storage mode, ready to accept a UF2 firmware file. Once copied the new firmware is flashed and the RP2040 reboots.

Calculator Interface

There is an interface to the T5006P replacement that can be accessed using the calculator keyboard. As the replacement circuit replaces the program memory the interface uses that memory to communicate. Commands are entered as a program (in PR15) and the command is executed once the full command ‘program’ has been entered. There is no way to indicate success on the screen, so once the END key is pressed wait about 3 seconds and the command should have been executed. Due to the way the RP2040 hardware works, the commands will not be recognised by the firmware very quickly, so the 3 second delay is important.

If you have the USB cable attached, you will see a message on the terminal when a command is recognised and executed.

Commands are coded as programs in PR15 of the form:

```
PR#15 : 9 = K ccsnnn : END
```

where:

cc : Command code

s : Status

nnn : Numeric parameters

Once the END is entered the firmware will see the command and execute it. Up to that point nothing will happen.

Commands

The commands that are available are:

Code	Mnemonic	Name	Description
11	**1**oad **1**oad (‘Load: load’)	LOAD	Loads the flash slot whose number is nnn into emulation RAM.
15	**1**oad **5**ave (Load: Save)	SAVE	Saves emulation RAM to the slot whose number is nnn
25	C**2**um **5**lot (‘Csum Slot’)	CSUM_SLOT	Checksums the flash slot whose number is nnn
20	C**2**um Emulati**0**n RAM	CSUM_RAM	Checksums the emulation RAM
30	S**3**t	SET_ADDR	Sets the address parameter to the value nnn
31	S**3**t (‘Set’)	SET_DATA	Sets the data parameter
32	S**3**t **2**lot (‘Set Slot’)	SET_SLOT	Sets the slot parameter (not used)
40	**4**mend **0**ut (‘Amend’)	WRITE_BYTE	Writes the value of the data parameter to the address given by the address parameter
41	**4**mend **1**n (‘Amend’)	READ_BYTE	Reads the value of the byte at the address given by the address parameter. The result is left in the PR15 constant.

I have tried to use mnemonic numbers, so, for instance,

There are two checksum commands, but due to PR15 being used to issue the command, there are problems with using them. The USB interface shows checksums and is a better way to use them. These commands leave the returned checksum in the PR15 constant.

NOTE: When issuing commands it is important to wait about 3s for the command to be recognised and executed. The USB interface displays a message when this happens.

Command Status

Each command attempts to leave status in the PR15 program after it is executed. Some commands cannot do this, for instance the LOAD command will overwrite the entire emulation RAM including the PR15 program and so no status can be left.

Status codes are listed below. Initially they were numeric but I switched to a more obvious alphabetic version with a later code revision. Numeric codes may still be left in the code.

Numeric Code	Character Code	Mnemonic	Name	Description
0			NOT_EXEC	This is the code to use when creating a PR15 program. The firmware knows that if it sees a command with this code it has to execute it.
1	C	**C**ommand done	COMMAND_DONE	The command has been successfully executed with no errors.
8	F	can't **F**ind command	COMMAND_UNKNOWN	The command code is unknown.
9	E	**E**rror	COMMAND_ERROR	The command is known but there is an error with it. For instance, an invalid slot number.

Wireless Interface

The wireless interface is in the code as a prototype, but is not accessible unless a Wi-Fi module is wired to the RP2040 Zero PCB. The power supply circuitry in the PRO-101 cannot supply the power needed to operate a Wi-Fi or Bluetooth module. When running off USB there is sufficient power, but if this is done, there is the danger of removing the USB cable and running off the PRO-101 power supply. This may damage the supply if done for long periods of time.