SMH Technologies®

30/08/2024

Driver v. 5.05

Moreno Ortolan

Interfacing FlashRunner 2.0 with INFINEON PSoC4

SMH Technologies S.r.l.

UNIVERSAL PRODUCTION IN-SYSTEM PROGRAMMING

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

INFINEON PSoC4 Introduction

32-bit PSoC™ Arm® Cortex® Microcontroller

PSoC™ Microcontrollers are the world's only programmable embedded System-on-Chip solutions based on the ARM® Cortex®-M processor, high-performance programmable analog blocks, PLD-based programmable digital blocks, programmable interconnect and routing, and CapSense™.

PSoC™ 4 is an ARM® -based PSoC™, featuring the low-power **Cortex®-M0** and **Cortex®-M0+** cores combined with PSoC's unique programmable mixed-signal hardware IP and CapSense™, resulting in the industry's most flexible and scalable low-power mixed-signal architecture.

32-bit PSoC™ Arm® Cortex® Microcontroller subcategories

- − 32-bit PSoC™ 4 Arm® Cortex®-M0/M0+
  - › Overview
  - › PSoC™ 4000 - Entry-Level
  - › PSoC™ 4100 - Intelligent Analog
  - › PSoC™ 4200 - Programmable Digital
  - › PSoC™ 4700 - Sense Anything
  - › PSoC™ 4 MCU with AIROC™ Bluetooth LE
- − 32-bit PSoC™ 4 Automotive Arm® Cortex®-M0/M0+
  - › Overview
  - › Automotive PSoC™ 4 Series
  - › Automotive PSoC™ 4 S-Series
  - › Automotive PSoC™ 4 M-Series
  - › Automotive PSoC™ 4 L-Series
- − 32-bit PSoC™ 4 HV Arm® Cortex®-M0+
  - › Overview
  - › PSoC™ 4 HV PA

32-bit PSoC™ 4 Arm® Cortex®-M0/M0+

PSoC™ 4 has tackled some of the complex portions of embedded system design making it easier for you to get your product to market.
Functions such as analog sensor integration, capacitive touch, and wireless connectivity have been integrated and optimized in PSoC™ 4 to just work.

The PSoC™ 4 portfolio consists of several families of Arm® Cortex®-M0 and Cortex-M0+ microcontrollers. Most devices in the portfolio include Infineon's CAPSENSE™ technology for capacitive-sensing applications.
Other key features in the PSoC™ 4 portfolio include a customizable analog front end through programmable analog blocks as well as wired and wireless connectivity options such as USB, CAN, and Bluetooth Low Energy.
These unique features make PSoC™ 4 the industry's most flexible and scalable low-power mixed-signal architecture.

32-bit PSoC™ 4 Arm® Cortex®-M0/M0+ subcategories

- + PSoC™ 4000 - Entry-Level
- + PSoC™ 4100 - Intelligent Analog
- › PSoC™ 4200 - Programmable Digital
- › PSoC™ 4700 - Sense Anything
- › PSoC™ 4 MCU with AIROC™ Bluetooth LE

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

SMH Technologies®

PSoC™ 4 families

PSoC™ 4000 family - CAPSENSE™
The PSoC™ 4000 family is a cost-optimized, entry-level family of Arm® Cortex®-M0 and M0+ microcontrollers. The PSoC™ 4000 family delivers the industry's best capacitive-sensing technology, CapSense, to implement buttons, sliders, and proximity sensors.

PSoC™ 4100 family - CAPSENSE™ + programmable analog blocks
The PSoC™ 4100 family adds intelligent analog integration through programmable analog blocks. Programmable analog blocks include analog-to-digital converters (ADCs), digital-to-analog converters (DACs), low-power comparators, and operational amplifiers (opamps).

PSoC™ 4200 family - CAPSENSE™ + programmable analog blocks + programmable digital blocks
The PSoC™ 4200 family boosts the flexibility and performance of the PSoC 4 portfolio by adding programmable, Universal Digital Blocks (UDBs). UDBs can be configured to set up custom digital interfaces, state machines, and custom logic functions. The PSoC 4200BL includes an integrated Bluetooth® Low Energy radio and subsystem.

PSoC™ 4700 family - CAPSENSE™ + inductive sensing + advanced sensing technologies
The PSoC™ 4700 family adds sensing technologies to sense anything including advanced capacitive sensing, inductive sensing, heart-rate sensing, and more. These advanced sensing technologies leverage specialized analog blocks in the PSoC™ 4 portfolio to deliver innovative solutions to next-generation designs.

PSoC™ 4 Bluetooth® Low Energy (BLE) - Bluetooth® smart
PSoC™ 4 Bluetooth® Low Energy enables the universal connectivity consumers to look for in their wireless devices by adding a royalty-free Bluetooth® Low Energy Protocol Stack to the popular PSoC™ 4 portfolio.

32-bit PSoC™ 4 Automotive: Arm® Cortex®-M0

Automotive PSoC™ 4 is a scalable and reconfigurable platform architecture for a family of programmable embedded system controllers with an ARM® Cortex®-M0+ CPU while being AEC-Q100 compliant.

The family combines programmable and reconfigurable analog and digital blocks with flexible automatic routing.
It is a combination of a microcontroller with standard communication and timing peripherals, a capacitive touch-sensing system (CapSense) with best-in-class performance, programmable general-purpose continuous-time, and switched-capacitor analog blocks, and programmable connectivity.

Automotive PSoC™ 4: 32-bit Arm® Cortex®-M0 MCU subcategories

+ Automotive PSoC™ 4 Series
+ Automotive PSoC™ 4 M-Series
+ Automotive PSoC™ 4 S-Series
+ Automotive PSoC™ 4 L-Series

32-bit PSoC™ 4 HV Arm® Cortex®-M0+

PSoC™ 4 HV (High Voltage) is a family of products providing one-chip solutions for smart sensor designs for automotive.
It integrates high precision and programmable analog blocks, programmable digital blocks along with high-voltage (12V) operation integrating communication transceivers such as LIN / CXPI PHY, and an ARM Cortex M0+ CPU with embedded memory.

PSoC™ 4 HV family devices are designed to be compliant with the automotive functional safety level of ASIL-B according to ISO 26262.

32-bit PSoC™ 4 HV Arm® Cortex®-M0+ subcategories

+ PSoC™ 4 HV PA

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

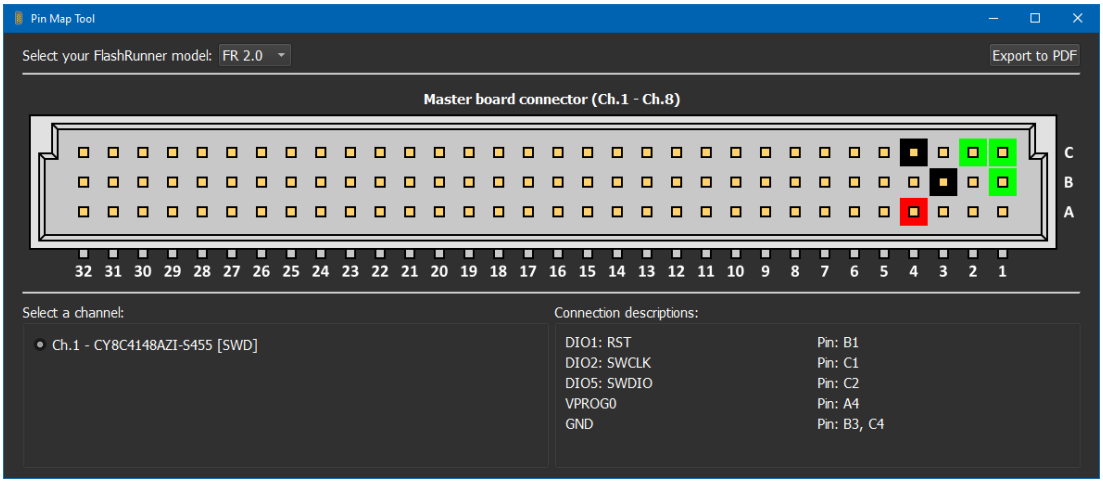
→smh-tech.com
info@smh-tech.com

INFINEON PSoC4 Protocol and PIN map

PSoC4 devices support the SWD protocol.

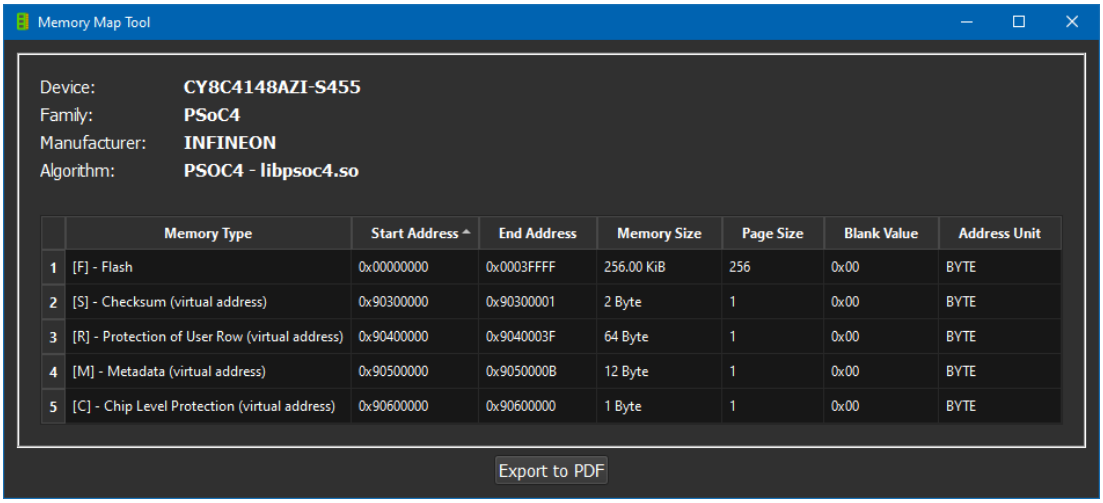
```
#TCSETPAR CMODE <SWD>
```

INFINEON PSoC4 PIN MAP



INFINEON PSoC4 Memory Map

Memory Type	Start Address	End Address	Memory Size	Page Size	Blank Value	Address Unit
[F] - Main Flash	0x00000000	0x0003FFFF	256.00 KiB	256	0x00	BYTE
[S] - Checksum (virtual address)	0x90300000	0x90300001	2 Byte	1	0x00	BYTE
[R] - Protection of User Row (virtual address)	0x90400000	0x9040003F	64 Byte	1	0x00	BYTE
[M] - Metadata (virtual address)	0x90500000	0x9050000B	12 Byte	1	0x00	BYTE
[C] - Chip Level Protection (virtual address)	0x90600000	0x90600000	1 Byte	1	0x00	BYTE



	Memory Type	Start Address	End Address	Memory Size	Page Size	Blank Value	Address Unit
1	[F] - Flash	0x00000000	0x0003FFFF	256.00 KiB	256	0x00	BYTE
2	[S] - Checksum (virtual address)	0x90300000	0x90300001	2 Byte	1	0x00	BYTE
3	[R] - Protection of User Row (virtual address)	0x90400000	0x9040003F	64 Byte	1	0x00	BYTE
4	[M] - Metadata (virtual address)	0x90500000	0x9050000B	12 Byte	1	0x00	BYTE
5	[C] - Chip Level Protection (virtual address)	0x90600000	0x90600000	1 Byte	1	0x00	BYTE

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com
info@smh-tech.com

INFINEON PSoC4 Driver Parameters

The standard parameters are used to configure some specific options inside PSoC4 driver.

#TCSETPAR ENTRY_CLOCK

Syntax: `#TCSETPAR ENTRY_CLOCK <Frequency>`

`<Frequency>` Accepted parameters 4000000, 2000000, 1000000, 500000, 100000 Hz

Description: Set the SWD frequency used in the Connect procedure before raising the PLL of the device, if the device PLL is available

Note: Default value 4.00 MHz

#TCSETPAR ACQUIRING_SEQUENCE

Syntax: `#TCSETPAR ACQUIRING_SEQUENCE`

Description: This parameter defines the entry mode
The acquiring chip procedures are defined by Infineon and you can find them in the Reference Manuals of PSoC4 devices
The *ACQUIRE_CHIP* is a procedure that uses the reset line to establish the communication between the FlashRunner and the target device
The other procedure, *ALTERNATE_ACQUIRE_CHIP* does not use the reset line to establish the communication.

Note: By default, the driver uses the *ACQUIRE_CHIP* method

#TCSETPAR SAMPLING_POINT

Syntax: `#TCSETPAR SAMPLING_POINT <Value>`

`<Value>` Accepted values are in the range 1-15

Description: Use this parameter to permanently set the sampling point of the FPGA
It is recommended to leave this parameter with the default value

Note: Default value 17

INFINEON PSoC4 Driver Commands

Here you can find the complete list of all available commands for PSoC4 driver.

```
F → Main Flash
S → Checksum (virtual address)
R → Protection of User Row (virtual address)
M → Metadata (virtual address)
C → Chip Level Protection (virtual address)
```

#TPCMD CONNECT

`#TPCMD CONNECT`
This function performs the entry and is the first command to be executed when starting the communication with the device.
There are two types of connect, regarding this please read `#TCSETPAR ACQUIRING_SEQUENCE`.
Here you can the log of a standard connect with the Acquire Chip procedure selected:

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com
info@smh-tech.com

```
---#TPSTART
Load SWD FPGA version 0x00001215.
Detected PSoC4 device: PSoC 4100 HVPA family.
Selected PSoC4 Acquire Sequence.
>|
---#TPCMD CONNECT
Protocol selected SWD.
Toggling XRES pin to execute Acquire Chip procedure.
ID-Code read correctly at 4.00 MHz after 52 retries.
JTAG-SWD Debug Port enabled.
Move PSoC4 internal state to Test Mode.
PSoC4 enter into Test Mode.
Read device informations:
 * Silicon ID: 0x320B10C2 - 0x320BxxC2.
 * Chip protection state: OPEN [0x1].
Checking device Silicon Id:
> Check PSoC4 Silicon ID passed from SMH database [0x320BxxC2].
 * Check PSoC4 Silicon ID from source file not available.
Scanning AP map to find all APs:
 * AP[0] IDR: 0x04770031, Type: AMBA AHB3 bus.
Scanning AP to find all cores:
 * AP[0] Found Cortex M0+ revision r0p1.
         CPUID: 0x410CC601.
         Implementer Code: 0x41 - [ARM].
         ROM table base address 0xF0000000.
Try to halt the Cortex M0+ core:
 * AP[0] Cortex M0+ Core halted [0.002 s].
Program counter value is 0x10000028.
Try to execute the Acquire Chip method procedure:
 * Acquire Chip method procedure completed.
Requested Clock is 37.50 MHz.
Generated Clock is 37.50 MHz.
Good samples: 4 [Range 4-7].
IDCODE: 0x0BC11477.
Designer: 0x23B, Part Number: 0xBC11, Version: 0x0.
ID-Code read correctly at 37.50 MHz.
Set Internal Main Oscillator at 48 MHz.
Start check device protections:
 * All Flash rows are unprotected.
Time for Connect: 0.123 s.
>|
```

ID-Code read correctly at 4.00 MHz after 42 retries

This is normal behaviour because when the PSoC4 device is powered on the SWD/JTAG port is not immediately available.
The PSoC4 driver tries to read the ID code several times before the SWD/JTAG port is activated.

Check PSoC4 Silicon ID from source file not available

This warning means that the FRB file is not present or the Silicon ID is not available inside FRB file.
Therefore, it is not possible to compare the Silicon ID read from the device with the one present in the file to be programmed.

#TPCMD MASSERASE

#TPCMD MASSERASE <F>
This function performs a masserase for Main Flash Memory.
Here you can the log of a standard masserase:

```
---#TPCMD MASSERASE F
Chip protection state: Open [0x1].
Start Standard Masserase operation:
 * Completed Masserase operation.
Start Checksum Privileged operation:
 * Checksum Privileged 28bit is 0x0000000.
Time for Masserase F: 0.260 s.
>|
```

Start Checksum Privileged operation: is a special 28bit checksum performed internally by PSoC4 device through specific API to check if all Flash memory is erased.

UNIVERSAL PRODUCTION IN-SYSTEM PROGRAMMING

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com
info@smh-tech.com

#TPCMD BLANKCHECK

`#TPCMD BLANKCHECK <F>`
Blankcheck is only available for Main Flash Memory.
Verify if all memory is erased.

`#TPCMD BLANKCHECK <F> <start address> <size>`
Blankcheck is only available for Main Flash Memory.
Verify if selected part of memory is erased.
Enter the Start Address and Size in hexadecimal format.

#TPCMD PROGRAM

`#TPCMD PROGRAM <F>`
Program available for Main Flash Memory.
Programs all memory of the selected type based on the data in the FRB file.

`#TPCMD PROGRAM <F> <start address> <size>`
Program available for Main Flash Memory.
Programs selected part of memory of the selected type based on the data in the FRB file.
Enter the Start Address and Size in hexadecimal format.

Programs the selected memory using the data loaded into the FRB file.

If no `#TPCMD MASSERASE` or `#TPCMD BLANKCHECK` command was issued before this command, the driver will assume that the device is not blank, and thus it will use a slower programming procedure, that also incudes an erasing routine.

This can be useful to partially re-program a device without the need to erase it completely.

#TPCMD VERIFY

`#TPCMD VERIFY <F> <R>`
R: Readout Mode.
Verify Readout available for Main Flash Memory.
Verify all memory of the selected type based on the data in the FRB file.

`#TPCMD VERIFY <F> <R> <start address> <size>`
R: Readout Mode.
Verify Readout available for Main Flash Memory.
Verify selected part of memory of the selected type based on the data in the FRB file.
Enter the Start Address and Size in hexadecimal format.

`#TPCMD VERIFY <F> <S>`
S: Checksum 32 Bit Mode.
Verify Checksum available for Main Flash Memory.
Verify all memory of the selected type based on the data in the FRB file.

`#TPCMD VERIFY <F> <S> <start address> <size>`
S: Checksum 32 Bit Mode.
Verify Checksum available for Main Flash Memory.
Verify selected part of memory based on the data in the FRB file.
Enter the Start Address and Size in hexadecimal format.

#TPCMD READ

`#TPCMD READ <F>`
Read function for Main Flash Memory.
The result of the read command will be visible into the Terminal.

UNIVERSAL PRODUCTION IN-SYSTEM PROGRAMMING

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

`#TPCMD READ <F> <start address> <size>`
Read function for Main Flash Memory.
The result of the read command will be visible into the Terminal.
Enter the Start Address and Size in hexadecimal format.

#TPCMD DUMP

`#TPCMD DUMP <F>`
Dump command for the Main Flash Memory.
The result of the dump command will be stored in the FlashRunner 2.0 internal memory.

`#TPCMD DUMP <F> <start address> <size>`
Dump command for the Main Flash Memory.
The result of the dump command will be stored in the FlashRunner 2.0 internal memory.
Enter the Start Address and Size in hexadecimal format.

#TPCMD SET_PROTECTION

Syntax: `#TPCMD SET_PROTECTION`

Prerequisites: none

Description: This function set the device protections based on data present into FRB file
[R] - Protection of User Row (virtual address): FRB section where you can st the protection user rows
[C] - Chip Level Protection (virtual address): FRB section where you can set the Chip Level Protection byte

Note: This command prints into Real Time Log

Examples: Correct command execution: 😊

When FRB data is not present into **[R]** and **[C]** the PSoC4 driver use default values for **Row-level protection** and for **Chip-level protection**

```
---#TPCMD SET_PROTECTION
Row-level protection not found inside FRB file.
Using default value. All rows are un-protected.
Chip-level protection not found inside FRB file.
Using default value. Set device to protection state OPEN [0x1].
Start program device macros:
 * Device macros programmed correctly.
Start verify device macros:
 * Device macros verified correctly.
Time for Set Protection: 0.044 s
>|
```

```
---#TPCMD GET_PROTECTION
Start check device protections:
Row protections: "X" means that the corresponding row (0x80 bytes) is protected.
Row protections: "-" means that the corresponding row (0x80 bytes) is unprotected.
 Checking Macro number 1:
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
 Checking Macro number 2:
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
```


HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

```
 * All Flash rows are unprotected.
Time for Get Protection: 0.003 s
>|
```

When FRB data is present into **[R]** and **[C]** the PSoC4 driver use the values inserted for **Row-level protection** and for **Chip-level protection**
Correct command execution: 😊

Here and example using `#DYNMEMSET2`:

`#DYNMEMSET2 0x90600000 1 01` (Chip-level protection set to **OPEN**)
`#DYNMEMSET2 0x90400010 16 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF` (Row-level protections)
`#DYNMEMSET2 0x90400030 16 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF` (Row-level protections)

```
---#TPCMD SET_PROTECTION
Start program device macros:
 * Device macros programmed correctly.
Start verify device macros:
 * Device macros verified correctly.
Time for Set Protection: 0.044 s
>|
```

```
---#TPCMD GET_PROTECTION
Start check device protections:
Row protections: "X" means that the corresponding row (0x80 bytes) is protected.
Row protections: "-" means that the corresponding row (0x80 bytes) is unprotected.
 Checking Macro number 1:
   * Protected Flash section from address 0x00004000 to 0x00007FFF.
   * Protected Flash section from address 0x0000C000 to 0x0000FFFF.
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX
   XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX
   XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX
 Checking Macro number 2:
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
 * Some Flash rows are protected.
Time for Get Protection: 0.003 s
>|
```

#TPCMD GET_PROTECTION

Syntax: `#TPCMD GET_PROTECTION`

Prerequisites: none

Description: This function gets the device protections: **Protection of User Row** and **Chip Level Protection**

Note: This command prints into Real Time Log

Examples: Correct command execution: 😊

```
---#TPCMD GET_PROTECTION
Start check device protections:
Row protections: "X" means that the corresponding row (0x80 bytes) is protected.
Row protections: "-" means that the corresponding row (0x80 bytes) is unprotected.
 Checking Macro number 1:
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
```


HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

```
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
 Checking Macro number 2:
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
 * All Flash rows are unprotected.
Time for Get Protection: 0.003 s
>|
```

Correct command execution: 😊

```
---#TPCMD GET_PROTECTION
Start check device protections:
Row protections: "X" means that the corresponding row (0x80 bytes) is protected.
Row protections: "-" means that the corresponding row (0x80 bytes) is unprotected.
 Checking Macro number 1:
   * Protected Flash section from address 0x00004000 to 0x00007FFF.
   * Protected Flash section from address 0x0000C000 to 0x0000FFFF.
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX
   XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX
   XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX
 Checking Macro number 2:
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
 * Some Flash rows are protected.
Time for Get Protection: 0.003 s
>|
```

#TPCMD GET_CHECKSUM

Syntax: `#TPCMD GET_CHECKSUM`

Prerequisites: none

Description: This function gets the 28bit Privileged checksum from PSoC4 device

Note: This command prints into Real Time Log

Examples: Correct command execution: 😊

With PSoC4 Flash memory completely erased:

```
---#TPCMD GET_CHECKSUM
Start Checksum Privileged operation:
* Checksum Privileged 28bit is 0x0000000.
Time for Get Checksum Privileged 28bit: 0.031 s
>|
```

Correct command execution: 😊

With PSoC4 Flash memory programmed:

```
---#TPCMD GET_CHECKSUM
Start Checksum Privileged operation:
```


HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

```
* Checksum Privileged 28bit is 0x1006CF0.
Time for Get Checksum Privileged 28bit: 0.031 s
>|
```

#TPCMD RUN

Syntax: `#TPCMD RUN <Time [s]>`

<Time [s]>	Time in seconds (i.e., 2 s). This time is an optional parameter.

Prerequisites: none

Description: Move the Reset line up and down quickly if no parameter <Time [s]> is inserted.
`#TPCMD RUN <Time [s]>` instead moves the Reset line down and high, waits for the entered time.
This command typically can be used to execute the firmware programmed in the device.

#TPCMD READ_MEM8

Syntax: `#TPCMD READ_MEM8 <Address> <Byte Count>`

<Address>	Address in HEX format (i.e., 0x52002020)
<Byte Count>	Byte count in decimal format (i.e., 8 -> eight bytes)

Prerequisites: none

Description: Read memory byte per byte from target PSoC4 device

Note: This command prints into Terminal and Real Time Log

Examples: Correct command execution: 😊

```
---#TPCMD READ_MEM8 0x52002020 8
Read[0x52002020]: 0xF0
Read[0x52002021]: 0xAA
Read[0x52002022]: 0x16
Read[0x52002023]: 0x14
Read[0x52002024]: 0x00
Read[0x52002025]: 0x00
Read[0x52002026]: 0x00
Read[0x52002027]: 0x00
Time for Read Mem: 0.002 s
```

#TPCMD READ_MEM16

Syntax: `#TPCMD READ_MEM16 <Address> <16-bit Word Count>`

<Address>	Address in HEX format (i.e., 0x52002020)
<16-bit Word Count>	16-bit Word count in decimal format (i.e., 4 -> four 16-bit words)

Prerequisites: none

Description: Read memory 16-bit word per 16-bit word from target PSoC4 device

Note: This command prints into Terminal and Real Time Log

Examples: Correct command execution: 😊

```
---#TPCMD READ_MEM16 0x52002020 4
Read[0x52002020]: 0xAAF0
Read[0x52002022]: 0x1416
Read[0x52002024]: 0x0000
Read[0x52002026]: 0x0000
Time for Read Mem: 0.002 s
```


HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

#TPCMD READ_MEM32

Syntax: `#TPCMD READ_MEM32 <Address> <32-bit Word Count>`

<Address>	Address in HEX format (i.e., 0x52002020)
<32-bit Word Count>	32-bit Word count in decimal format (i.e., 2 -> two 32-bit words)

Prerequisites: none

Description: Read memory 32-bit word per 32-bit word from target PSoC4 device

Note: This command prints into Terminal and Real Time Log

Examples: Correct command execution: 😊

```
---#TPCMD READ_MEM32 0x52002020 2
Read[0x52002020]: 0x1416AAF0
Read[0x52002024]: 0x00000000
Time for Read Mem: 0.002 s
```

#TPCMD DISCONNECT

`#TPCMD DISCONNECT`

Disconnect function. Power off and exit.

INFINEON PSoC4 Driver Examples

Here you can see a complete example of INFINEON PSoC4 projects.

1 – INFINEON PSoC4 128 KB example Commands

```
#TCSETPAR ACQUIRING_SEQUENCE ACQUIRE_CHIP
#TCSETPAR PROTCLK 37500000
#TCSETPAR ENTRY_CLOCK 4000000
#TCSETPAR PWDOWN 100
#TCSETPAR PWUP 100
#TCSETPAR RSTDOWN 100
#TCSETPAR RSTDRV OPENDRAIN
#TCSETPAR RSTUP 100
#TCSETPAR VPROG0 3300
#TCSETPAR CMODE SWD
#TPSETSRC 128KB.frb
#TPSTART
#TPCMD CONNECT
#TPCMD GET_PROTECTION
#TPCMD SET_PROTECTION
#TPCMD MASSERASE F
#TPCMD BLANKCHECK F
#TPCMD PROGRAM F
#TPCMD VERIFY F R
#TPCMD VERIFY F S
#TPCMD DISCONNECT
#TPEND
```

1 – INFINEON PSoC4 128 KB example Real Time Log

```
---#TPSTART
Load SWD FPGA version 0x00001215.
Detected PSoC4 device: PSoC 4100 HVPA family.
Selected PSoC4 Acquire Sequence.
>|
```


HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

```
---#TPCMD CONNECT
Protocol selected SWD.
Toggling XRES pin to execute Acquire Chip procedure.
ID-Code read correctly at 4.00 MHz after 52 retries.
JTAG-SWD Debug Port enabled.
Move PSoC4 internal state to Test Mode.
PSoC4 enter into Test Mode.
Read device informations:
 * Silicon ID: 0x320B10C2 - 0x320BxxC2.
 * Chip protection state: OPEN [0x1].
Checking device Silicon Id:
> Check PSoC4 Silicon ID passed from SMH database [0x320BxxC2].
 * Check PSoC4 Silicon ID from source file not available.
Scanning AP map to find all APs:
 * AP[0] IDR: 0x04770031, Type: AMBA AHB3 bus.
Scanning AP to find all cores:
 * AP[0] Found Cortex M0+ revision r0p1.
         CPUID: 0x410CC601.
         Implementer Code: 0x41 - [ARM].
         ROM table base address 0xF0000000.
Try to halt the Cortex M0+ core:
 * AP[0] Cortex M0+ Core halted [0.002 s].
Program counter value is 0x10000028.
Try to execute the Acquire Chip method procedure:
 * Acquire Chip method procedure completed.
Requested Clock is 37.50 MHz.
Generated Clock is 37.50 MHz.
Good samples: 4 [Range 4-7].
IDCODE: 0x0BC11477.
Designer: 0x23B, Part Number: 0xBC11, Version: 0x0.
ID-Code read correctly at 37.50 MHz.
Set Internal Main Oscillator at 48 MHz.
Start check device protections:
 * All Flash rows are unprotected.
Time for Connect: 0.123 s.
>|
---#TPCMD GET_PROTECTION
Start check device protections:
Row protections: "X" means that the corresponding row (0x80 bytes) is protected.
Row protections: "-" means that the corresponding row (0x80 bytes) is unprotected.
 Checking Macro number 1:
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
 Checking Macro number 2:
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- -------- -------- --------
 * All Flash rows are unprotected.
Time for Get Protection: 0.003 s
>|
---#TPCMD SET_PROTECTION
Row-level protection not found inside FRB file.
Using default value. All rows are un-protected.
Chip-level protection not found inside FRB file.
Using default value. Set device to protection state OPEN [0x1].
Start program device macros:
 * Device macros programmed correctly.
Start verify device macros:
 * Device macros verified correctly.
Time for Set Protection: 0.044 s
>|
---#TPCMD MASSERASE F
Chip protection state: Open [0x1].
Start Standard Masserase operation:
 * Completed Masserase operation.
Start Checksum Privileged operation:
 * Checksum Privileged 28bit is 0x0000000.
Time for Masserase F: 0.260 s.
>|
```


HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

```
---#TPCMD BLANKCHECK F
Start Blankcheck operation.
Time for Blankcheck F: 0.009 s.
>|
---#TPCMD PROGRAM F
Start Program operation.
Time for Program F: 2.769 s.
>|
---#TPCMD VERIFY F R
Start Verify Readout operation.
Time for Verify Readout F: 0.056 s.
>|
---#TPCMD VERIFY F S
Start Verify Checksum 32bit operation.
Time for Verify Checksum 32bit F: 0.008 s.
>|
---#TPCMD DISCONNECT
>|
```

1 – INFINEON PSoC4 128 KB example Programming Times

Operation	Timings FlashRunner 2.0
Time for Connect	0.123 s
Get Protection	0.003 s
Set Protection	0.044 s
Masserase Flash	0.260 s
Blankcheck Flash	0.009 s
Program Flash	2.769 s
Verify Readout Flash	0.056 s
Verify Checksum Flash	0.008 s
Cycle Time	**00:03.322 s**

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

INFINEON PSoC4 Driver Changelog

Info about driver versions prior to 5.00
All driver versions prior to 5.00 are to be considered obsolete, please update your driver to the latest version.

Info about driver version 5.00 - 04/12/2023
Upgraded PSoC4 driver, supported all PSoC4 devices already available in the market.

Info about driver version 5.01 - 05/12/2023
Added PSoC4 CYPD7xx devices of CCG7D family.

Info about driver version 5.02 - 21/03/2024
Upgraded PSoC4 driver to manage retro compatibility for a specific project case.

Info about driver version 5.03 - 28/03/2024
Added PSoC4 4100 HVMS devices.

Info about driver version 5.04 - 22/04/2024
Added PSoC4 4100 S Max devices.

Info about driver version 5.05 - 30/08/2024
Added PSOC4 CY8C414xxxx-HVSxxx devices.

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com
info@smh-tech.com