Identification d’images de micrologiciels

Ambre IOOSS

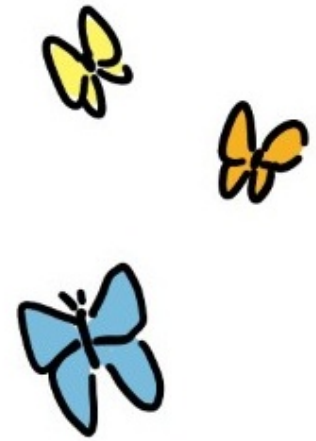
SSTIC 2025 — 4 juin 2025

Qui suis-je ?

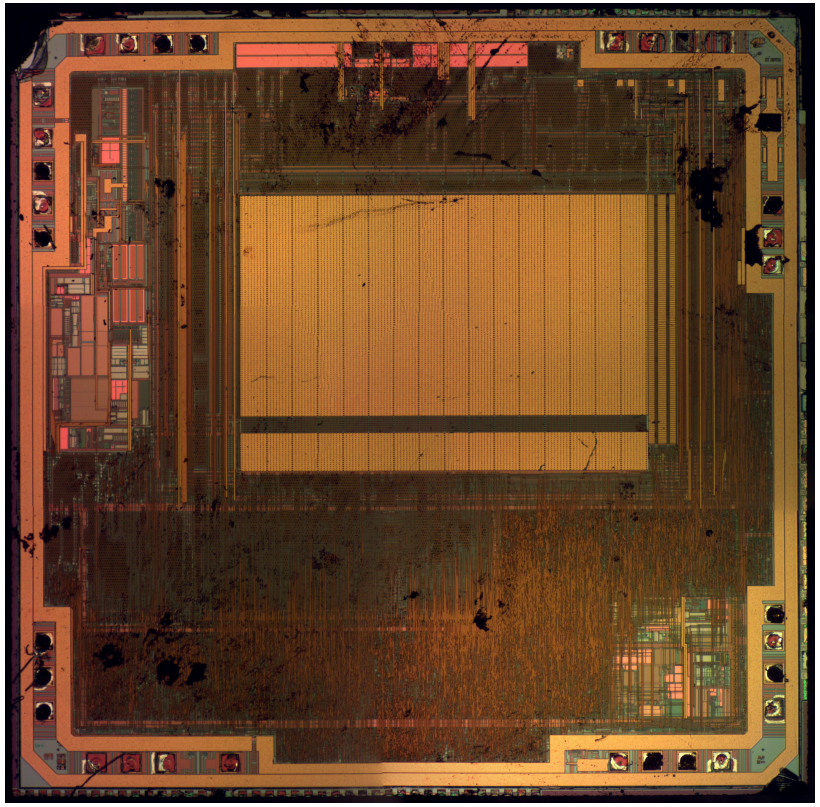
Experte en sécurité matérielle chez **Synacktiv**, anciennement à l'**ANSSI**.

Régulièrement exposée à du micrologiciel en CTF à *The Flat Network Society* et en coaching *TeamFrance*.

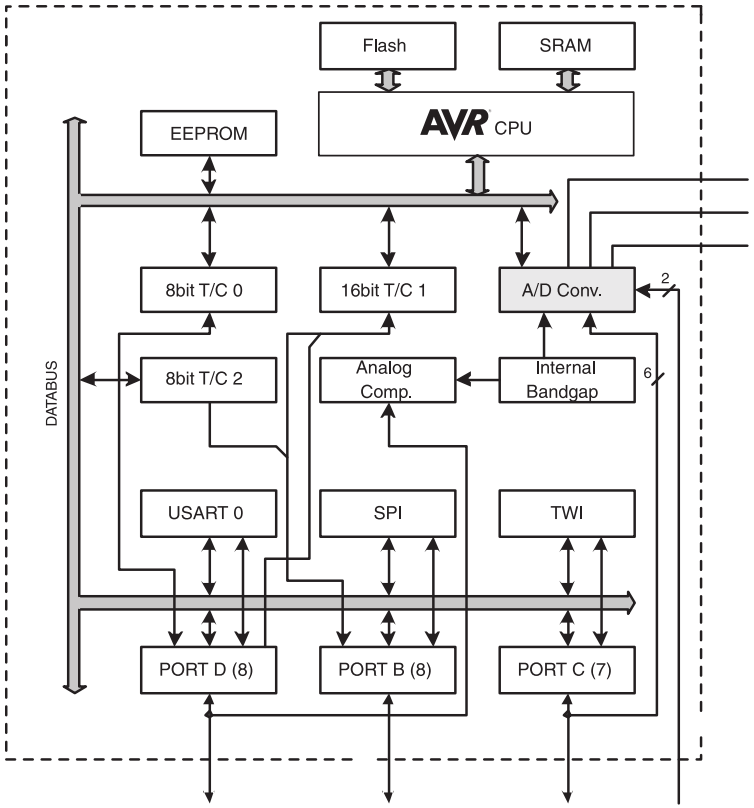
Observation : outillage parfois limité pour démarrer une épreuve de sécurité matérielle.



Microcontrôleur : Microchip ATmega328P



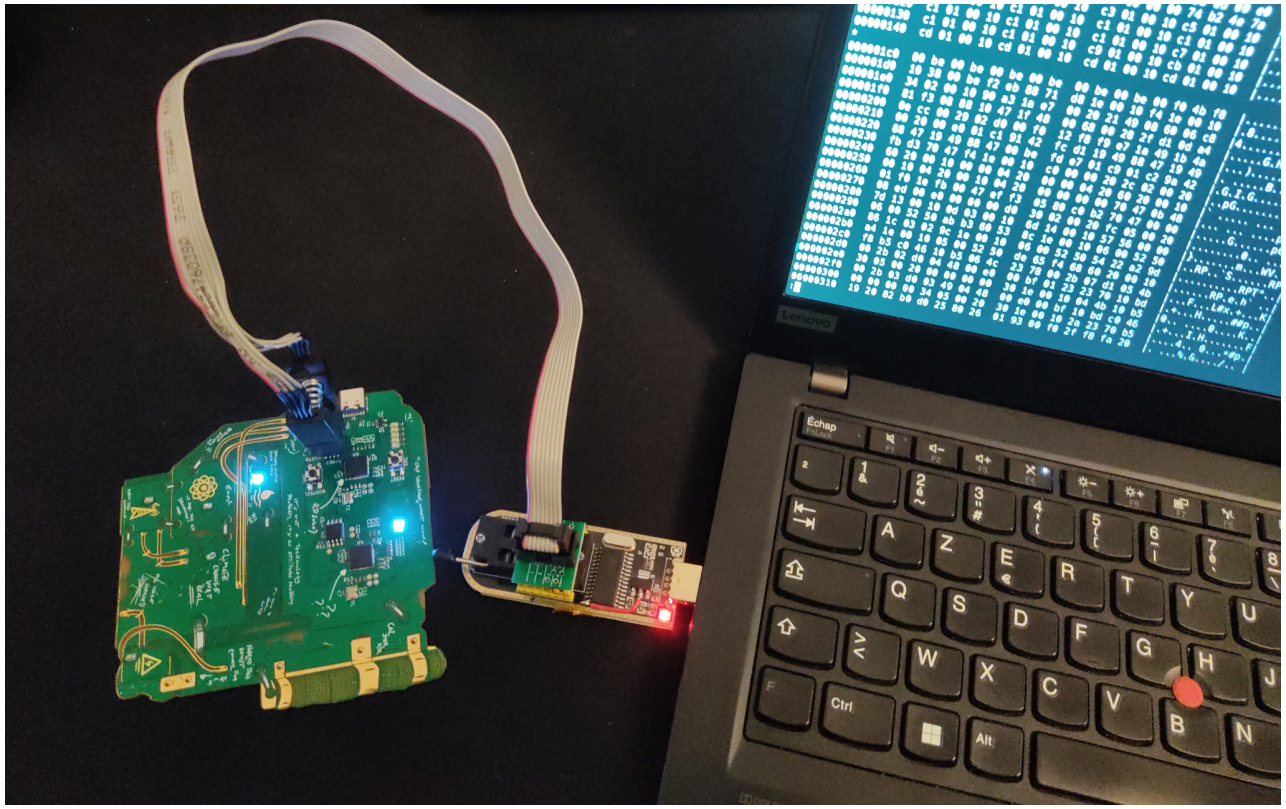
2990 µm × 2950 µm, Andrew Zonenberg



Extrait de la fiche technique

Scénario

Épreuve hardware de CTF, microcontrôleurs poncés / sous époxy.



Lecteur de Flash SPI CH341A branché à l'épreuve hardware de l'ECSC 2022

Tentatives d'identification du dump

```
$ file dump.bin
dump.bin: data
```

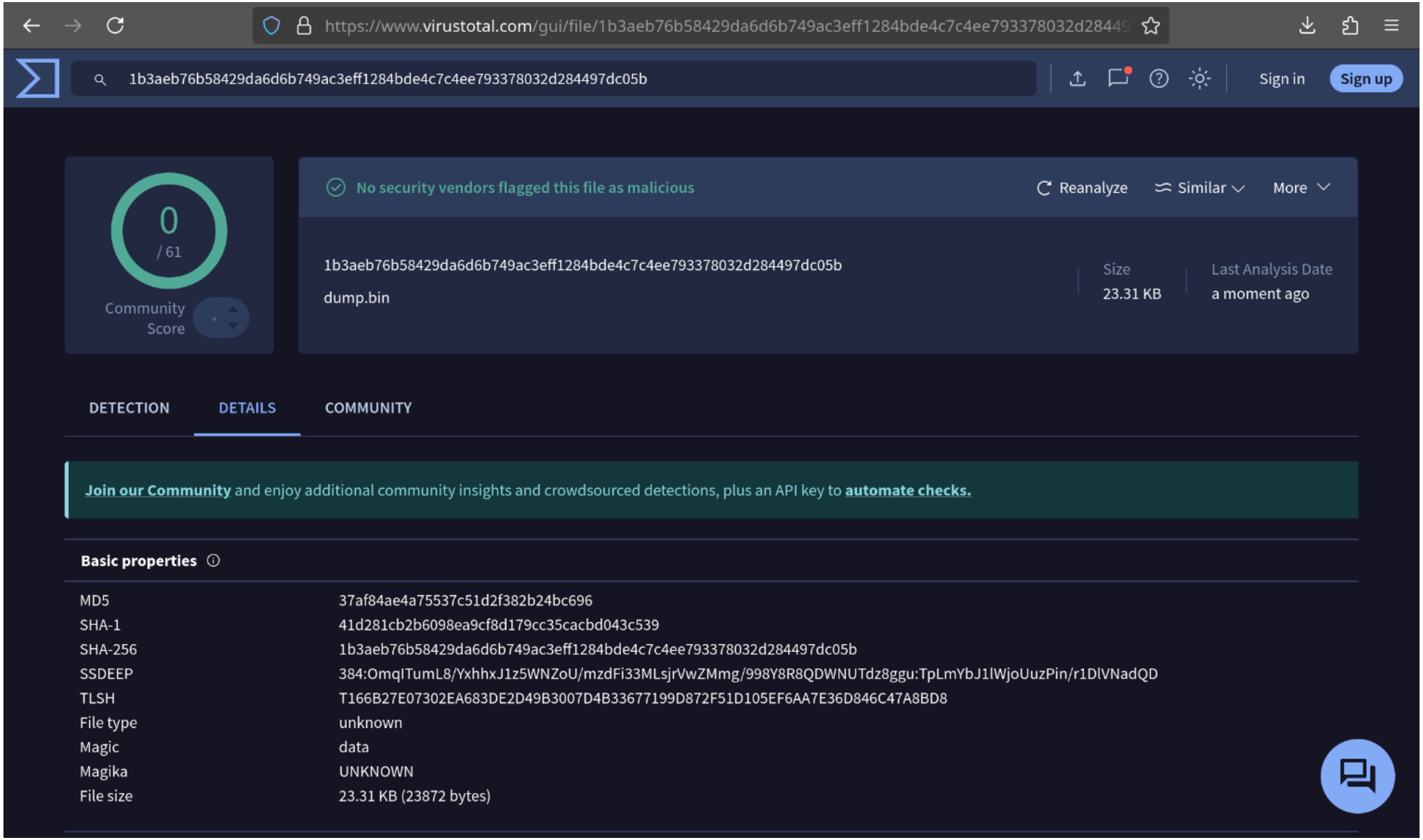
file 5.39 sous Debian Bullseye

Tentatives d'identification du dump

```
$ binwalk dump.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
```

binwalk 2.3.4 sous Debian Bullseye


https://www.virustotal.com/gui/file/1b3aeb76b58429da6d6b749ac3eff1284bde4c7c4ee793378032d28449
1b3aeb76b58429da6d6b749ac3eff1284bde4c7c4ee793378032d284497dc05b
Sign in
Sign up
0
/ 61
Community Score
No security vendors flagged this file as malicious
Reanalyze
Similar
More
1b3aeb76b58429da6d6b749ac3eff1284bde4c7c4ee793378032d284497dc05b
dump.bin
Size
23.31 KB
Last Analysis Date
a moment ago
DETECTION
DETAILS
COMMUNITY
Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.
Basic properties
MD5 37af84ae4a75537c51d2f382b24bc696
SHA-1 41d281cb2b6098ea9cf8d179cc35cacbd043c539
SHA-256 1b3aeb76b58429da6d6b749ac3eff1284bde4c7c4ee793378032d284497dc05b
SSDEEP 384:OmqlTumL8/YxhhxJ1z5WNZoU/mzdFi33MLsjrVwZMmg/998Y8R8QDWNUTdz8ggu:TpLmYbJ1lWjoUuzPin/r1DlVNadQD
TLSH T166B27E07302EA683DE2D49B3007D4B33677199D872F51D105EF6AA7E36D846C47A8BD8
File type unknown
Magic data
Magika UNKNOWN
File size 23.31 KB (23872 bytes)

Tentatives d'identification du dump

```
$ cpu_rec.py dump.bin
dump.bin            full(0x5d40)   ARMhf            chunk(0x2400;18)    ARMhf
```

Louis Granboulan, SSTIC 2017

Tentatives d'identification du dump

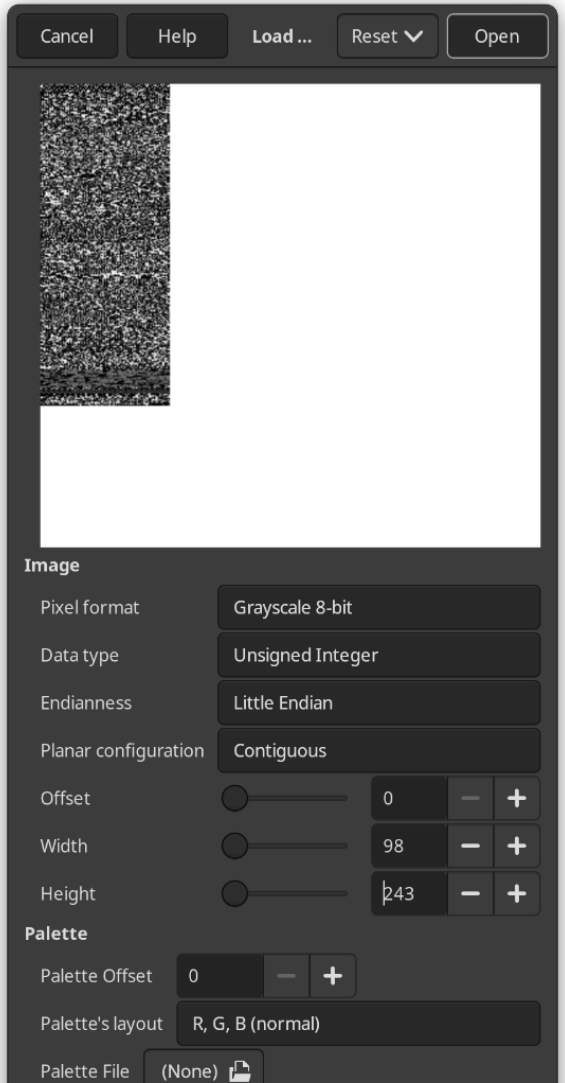
```
$ binbloom dump.bin
[i] File read (23872 bytes)
[i] Endianness is LE
[i] 55 strings indexed
[i] Found 80 base addresses to test
[i] Base address found: 0x10000000.
 More base addresses to consider (just in case):
  0x1fffb000 (0.03)
```

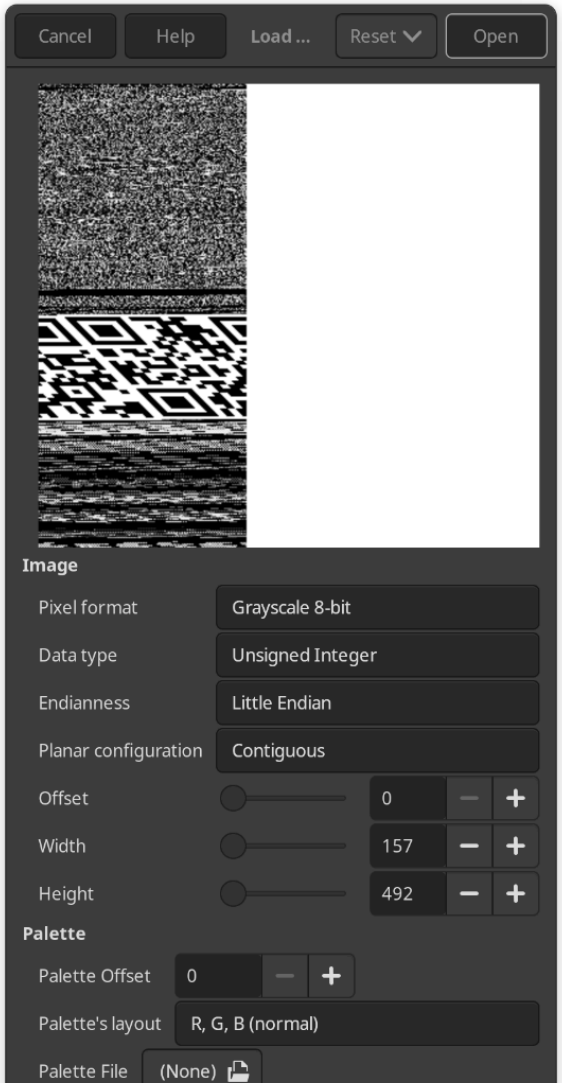
Damien Cauquil, SSTIC 2022

Tentative d'identification du dump

```
$ cp dump.bin dump.data
$ gimp dump.data
```

Technique si le microcontrôleur gère un écran.





Informations obtenues : **ARMhf**, chargé en **0x10000000**

Informations souhaitées : **ARM Cortex-M** pour **RP2040**, chargé en **0x10000000**

Buts de cette présentation

1. améliorer l'outillage libre pour mieux identifier les micrologiciels
2. montrer que c'est possible d'identifier précisément ces blobs binaires

Idée 1. Identification avec `file` / `libmagic`

Idée 2. Identification avec les accès aux périphériques

Introduction au format de `file` / `libmagic`

`file` opensource depuis 1986 par Ian Darwin, maintenu par Christos Zoulas.

```
0       string          DfuSe\x01       Image DFU (variante STM)
>6      ulelong         x               \b, taille: %d octets
# suffixe optionnel, spécification 0x011A
>-10    string          \x1A\x01UFD
>>-12   uleshort        x               \b, pour périphérique %04X:
>>-14   uleshort        x               \b%04X
```

Exemple de signature pour détecter un fichier `.dfu`

Format compatible `binwalk` et `polyfile`. En savoir plus : `man magic`

Pour s'entraîner : épreuve "*The Colour of Magic*" du FCSC 2023, sur hackropole.fr

Identification de l'architecture CPU

3 cas observés :

1. Structure attendue par la ROM

   Espressif ESP8266/ESP32, NXP FlexSPI

2. Code d'initialisation à une position connue fixe

   Raspberry Pi RP2040, SiFive Freedom E, Blackfin

3. Table de vecteurs d'interruption à une position connue fixe

   ARM Cortex-M, Microchip AVR et PIC, TI MSP430, 6502, NXP Coldfire, AMD MicroBlaze

Heuristiques implémentables dans `file` / `libmagic` !

Cas 1 : Structure attendue par la ROM

esptool.py

ESPRESSIF

ESP32

master (latest)

Search docs

- Installation
- Esptool
- Espefuse
- Espsecure
- Remote Serial Ports
- Advanced Topics
- Firmware Image Format
  - File Header
  - Extended File Header
  - Segment

File Header

Firmware image header

The image header is 8 bytes long:

Byte	Description
0	Magic number (always `0xE9`)
1	Number of segments
2	SPI Flash Mode (`0` = QIO, `1` = QOUT, `2` = DIO, `3` = DOUT)
3	High four bits - Flash size (`0` = 1MB, `1` = 2MB, `2` = 4MB, `3` = 8MB, `4` = 16MB) Low four bits - Flash frequency (`0` = 40MHz, `1` = 26MHz, `2` = 20MHz, `0xf` = 80MHz)
4-7	Entry point address

Cas 1 : Structure attendue par la ROM

```
0          byte          0xE9
>2        byte          <4
>>7       byte          0x40              Image firmware ESP
>>>12     uleshort      0x0000            pour ESP32
>>>12     uleshort      0x0002            pour ESP32-S2
>>>12     uleshort      0x0005            pour ESP32-C3
>>>12     uleshort      0x0009            pour ESP32-S3
>>>4      ulelong       x                 \b, entry point 0x%08X
```

Extrait simplifié de la signature pour Espressif ESP32/ESP8266 dans `file` / `libmagic`

Même stratégie pour NXP FlexSPI (Teensy)

Cas 2 : Code d'initialisation

```
_stage2_boot:
    push {lr}
    ldr r3, =PADS_QSPI_BASE
    movs r0, #(2 << PADS_QSPI_GPIO_QSPI_SCLK_DRIVE_LSB | PADS_QSPI_GPIO_QSPI_SCLK_SLEWFAST_BITS)
    str r0, [r3, #PADS_QSPI_GPIO_QSPI_SCLK_OFFSET]
    ldr r0, [r3, #PADS_QSPI_GPIO_QSPI_SD0_OFFSET]
    movs r1, #PADS_QSPI_GPIO_QSPI_SD0_SCHMITT_BITS
[...]

check_return:
    pop {r0}
    cmp r0, #0
[...]
```

Extrait du `_stage2_boot` de `pico-sdk` pour Raspberry Pi RP2040

Cas 2 : Code d'initialisation

```
# Bootloader stage 2, sources dans le pico-sdk
# boot2_*.S code (_stage2_boot)
0               ulelong                 0x4B32B500
>4              ulelong                 0x60582021
>>8             ulelong                 0x21026898
# exit_from_boot2.S code (check_return) `pop {r0}; cmp r0, #0`
>>>148         ulelong                 0x2800bc01
```

Extrait simplifié de la signature pour Raspberry Pi RP2040 dans `file` / `libmagic`

Même stratégie pour SiFive Freedom E, Blackfin

Cas 3 : Table de vecteurs d'interruption

Position	Vecteur
00	Pointeur de pile
04	Reset
08	NMI
0C	HardFault
10	MemoryManageFault
14	BusFault
18	UsageFault
1C - 28	(Réservé)
2C	SVCall

Début des tables de vecteurs ARM Cortex-M

Cas 3 : Table de vecteurs d'interruption

MicroPython pour ARM Cortex-M (nRF52)

```
00000000: 00 00 04 20  c1 d0 02 00  31 9c 02 00  7b f1 01 00
00000010: 31 9c 02 00  31 9c 02 00  31 9c 02 00  00 00 00 00
00000020: 00 00 00 00  00 00 00 00  00 00 00 00  31 9c 02 00
```

Lecture en 32-bit little-endian :

```
           SP init      Reset        NMI          HardFault
00000000: 20040000     0002d0c1     00029c31     0001f17b
           MemFault     BusFault     UsageFault
00000010: 00029c31     00029c31     00029c31     00000000
                                                 SVCall
00000020: 00000000     00000000     00000000     00029c31
```

Cas 3 : Table de vecteurs d'interruption

Adresses **impaires** :

```
           SP init     Reset       NMI         HardFault
00000000: 20040000     0002d0c1    00029c31    0001f17b
           MemFault    BusFault    UsageFault
00000010: 00029c31     00029c31    00029c31    00000000
                                               SVCall
00000020: 00000000     00000000    00000000    00029c31
```

Cas 3 : Table de vecteurs d'interruption

Contraintes sur les **plages d'adresses** :

```
           SP init     Reset       NMI         HardFault
00000000:  20040000    0002d0c1    0029c31     0001f17b
           MemFault    BusFault    UsageFault
00000010:  0029c31     0029c31     0029c31     00000000
                                               SVCall
00000020:  00000000    00000000    00000000    0029c31
```

Cas 3 : Table de vecteurs d'interruption

```
# Octet de poids fort de la pile
3               byte                    0x20
# Les pointeurs sont impairs et avant 0x20000000
>4              ulelong&0xE0000001      0x00000001
>>8             ulelong&0xE0000001      0x00000001
>>>12           ulelong&0xE0000001      0x00000001
>>>>44          ulelong&0xE0000001      0x00000001
>>>>>56         ulelong&0xE0000001      0x00000001   ARM Cortex-M
```

Extrait simplifié de la signature pour ARM Cortex-M dans `file` / `libmagic`

Cas 3 : Table de vecteurs d'interruption

Plus simple et robuste si utilisation d'opcode.

```
# JMP (4 octets) pour Reset, Int0-2, PcInt0-3 et WDT
0           uleshort&0xFE0E     0x940C
>4          uleshort&0xFE0E     0x940C
>>8         uleshort&0XFE0E     0x940C
>>>12       uleshort&0XFE0E     0x940C
>>>>16      uleshort&0XFE0E     0x940C
>>>>>20     uleshort&0XFE0E     0x940C
>>>>>>24    uleshort&0XFE0E     0x940C
>>>>>>>28   uleshort&0XFE0E     0x940C
>>>>>>>>32  uleshort&0XFE0E     0x940C          AVR firmware
```

Extrait simplifié de la signature pour AVR dans `file` / `libmagic`

Même stratégie pour PIC, TI MSP430, 6502, Coldfire et MicroBlaze

Démonstration

```
$ file dump.bin
dump.bin: ARM Cortex-M firmware, initial SP at 0x20040000,
    reset at 0x0002d0c0, NMI at 0x00029c30, HardFault at 0x0001f17a,
    SVCall at 0x00029c30, PendSV at 0x00029c30
$ file dump2.bin
dump2.bin: NXP i.MX RT bootable image, version 1.4.0, ARM Cortex-M,
    initial SP at 0x20002000, reset at 0x6000de18, NMI at 0x6000ddd4,
    HardFault at 0x6000de50, SVCall at 0x6000dd5c, PendSV at 0x6000ddd4
$ file dump3.bin
dump3: AVR firmware, reset at 0x0174
```

Tests sur 3000+ échantillons, github.com/erdnaxe/firmware-samples

Limitation : imprécision sur la famille de microcontrôleur

Idée 1. Identification avec `file`/`libmagic`

Idée 2. Identification avec les accès aux périphériques

Identification avec les accès aux périphériques

Contexte : jeu d'instructions connu, mais chip inconnu

Idée : désassemblage du code et identification des accès aux périphériques

```
LDR   r3, [pc, #0x2c]    // pc+0x2c pointe le périphérique GPIOB
MOV.W r2, #0x800
STRH  r2, [r3, #0x18]    // 0x18 est l'offset du registre BSRR
```

Exemple d'écriture d'un GPIO en ARM Thumb

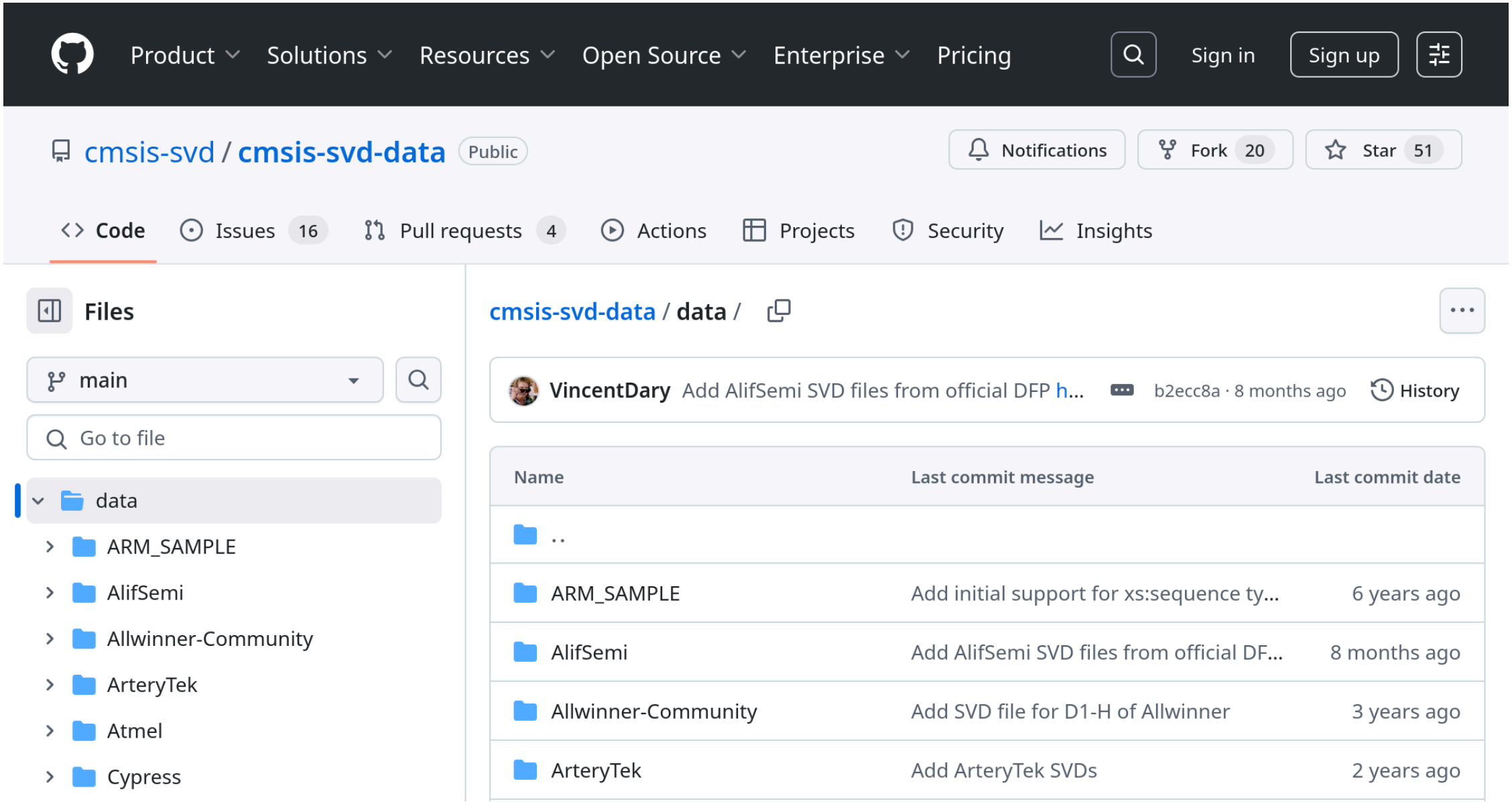
Collecte de descriptions de microcontrôleurs

Architecture	Format de description	Exemples d'outils
ARM Cortex-M	System View Description (SVD)	CMSIS, svdtools
AVR	AVR Tools Device File (ATDF)	avr-rust, atdf2svd
Risc-V	System View Description (SVD)	riscv-rust
TI MSP430	DSLite (folder of XML files)	msp430_svd
Xtensa	System View Description (SVD)	esp-rs

Le format SVD est devenu un dénominateur commun.

```
<device schemaVersion="1.3">
  <name>LPC54S005</name>
  <peripherals>
    <peripheral>
      <name>SYSCON</name>
      <baseAddress>0x40000000</baseAddress>
      <registers>
        <register>
          <name>AHBMATPRIO</name>
          <addressOffset>0x10</addressOffset>
          <fields>
            <field>
              <name>PRI_ICODE</name>
              <description>I-Code bus priority.</description>
              <bitOffset>0</bitOffset>
              <bitWidth>2</bitWidth>
              <access>read-write</access>
            </field>
```

Extrait de fichier System View Description (SVD)


Product
Solutions
Resources
Open Source
Enterprise
Pricing
Sign in
Sign up
cmsis-svd / cmsis-svd-data
Public
Notifications
Fork 20
Star 51
Code
Issues 16
Pull requests 4
Actions
Projects
Security
Insights
Files
main
Go to file
data
ARM_SAMPLE
AlifSemi
Allwinner-Community
ArteryTek
Atmel
Cypress
cmsis-svd-data / data /
VincentDary Add AlifSemi SVD files from official DFP h...
b2ecc8a · 8 months ago
History
Name
Last commit message
Last commit date
..
ARM_SAMPLE
Add initial support for xs:sequence ty...
6 years ago
AlifSemi
Add AlifSemi SVD files from official DF...
8 months ago
Allwinner-Community
Add SVD file for D1-H of Allwinner
3 years ago
ArteryTek
Add ArteryTek SVDs
2 years ago

CMSIS Packs

CMSIS-Pack is a distribution format that makes it easy to use software components, device drivers and middleware in CMSIS development tools including Keil MDK and Keil Studio.

Search by name or vendor

Vendor | Content type | Include deprecated packs

Results (1303)

Sort by: Relevance

32L4R9IDISCOVERY_BSP Keil — Download version 1.0.0

STMicroelectronics STM32L4 Series 32L4R9IDISCOVERY Board Support Pack

1 Board

A31G1xx Series ABOV Semiconductor — Download version 2.5.0

Traitement des données et identification

```
$ ./chiprec.py ./dump3.bin
Found addresses: 0x58024428 (read), 0x58024454 (read) [...]

STM32H742x (STM32H742x.svd):
    [...]

STM32H753 (STM32H753.svd):
    read register PLLCKSELR of RCC
    read register D2CCIP2R of RCC
    read register D3CCIPR of RCC
    read register DIER of TIM6
    read register CR of CRS
    read register D3CR of PWR
    read register CR of RCC
    [...]
```

Extrait de l'exécution de `chiprec.py` sur un micrologiciel

Démonstrations

Merci pour votre attention