CMSIS-SVD environment and scripts

Stable: 21.02.2020 - 08:34 | Revision: 20.02.2020 - 11:34

A quality version of this page, approved on *21 February 2020*, was based off this revision.

Contents

1 Article purpose

This article describes how to read or write to the STM32MPU internal peripherals registers using CMSIS-SVD file through GDB.

2 Information about CMSIS-SVD files

Extracted from Arm® Keil® web page[1]:

> *The CMSIS System View Description format(CMSIS-SVD) formalizes the description of the system contained in Arm Cortex-M processor-based microcontrollers, in particular, the memory mapped registers of peripherals. The detail contained in system view descriptions is comparable to the data in device reference manuals. The information ranges from high level functional descriptions of a peripheral all the way down to the definition and purpose of an individual bit field in a memory mapped register.*

Arm® also specifies on an other web page that CMSIS-SVD[2] is not restricted to only for Cortex®-M:

> ***For every supported microcontroller**, debuggers can provide detailed views to the device peripherals that display the current register state.*

3 Python scripts environment for GDB

Below information is related to the Android™ distribution
Not yet applicable for Android

3.1 CMSIS-SVD file parser

A CMSIS-SVD file parser is proposed by Paul Osborne and shared is on github: https://github.com/posborne/cmsis-svd.

This package is reused as is. Add STM32MPU microprocessor if it is missing from the CMSIS-SVD files.

For licensing, following information is provided:
In general, the following rules apply:

- *Under data, the license from each Vendor is provided along with the SVDs from that vendor. Please review this license before use of the SVDs contained therein. Look for files named the following for license information.*
- *All other code is licensed under the terms of the Apache License v2.0 (See LICENSE-APACHE).*

In our case, we will only add new files in data/STMicro sub-directory, so STMicroelectronics license is applied.

This package is part of the OpenSTLinux Developer Package.

3.2 Python scripts for GDB

svd-tools[3] github repository provides a gdb-svd python module for gdb which adds some instructions to:

- get information on peripherals | registers | fields
- get registers values | register fields
- set registers values | register fields

For more information, refer to the README[4] file of the repository.

This package is part of the OpenSTLinux Developer Package.

3.3 Setting environment

As pre-requisites, the gdb setup must be installed and running.

As CMSIS-SVD scripts are also part of the OpenSTLinux Developer Package, the environment is set by using the instructions below :

```
# Get the sysroot path from your SDK
(gdb) show environment OECORE_NATIVE_SYSROOT
 OECORE_NATIVE_SYSROOT = <path_to_sysroot>
(gdb) cd <path_to_sysroot>
(gdb) source usr/share/svd-tools/gdb-svd.py
```

4 Python scripts usage for GDB

4.1 Load svd file for the requested SoC

The **svd** instruction loads the appropriate SOC svd files.

Here is an example for the STM32MP15xxx:

```
(gdb) svd usr/share/cmsis-svd/cmsis_svd/data/STMicro/STM32MP15xxx.svd
```

Note: for all svd instructions described below, instruction completion for register name is working.

4.2 Getting peripheral / register / field information

This instruction displays any peripheral, register, and field information.
The last parameter can be part of the name, a filter is applied with the string.

```
(gdb) svd info <peripheral> [register] [field]
```

Please refer to the following examples:

- Getting information on all available ADC peripherals

(gdb) svd info ADC

```
+Peripherals-+------------+--------+-----------------------------+
| name       | base       | access | description                 |
+------------+------------+--------+-----------------------------+
| ADC2       | 0x48003100 | None   | ADC2                        |
| ADC        | 0x48003000 | None   | ADC                         |
| ADC_common | 0x48003300 | None   | Analog-to-Digital Converter |
+------------+------------+--------+-----------------------------+
```

- Getting information on all ADC2 registers beginning by S

```
(gdb) svd info ADC2 ADC_S
+Registers--+------------+------------+---------------------------------+
| name      | address    | access     | description                     |
+-----------+------------+------------+---------------------------------+
| ADC_SMPR1 | 0x48003114 | read-write | ADC sample time register 1      |
| ADC_SMPR2 | 0x48003118 | read-write | ADC sample time register 2      |
| ADC_SQR1  | 0x48003130 | read-write | ADC regular sequence register 1 |
| ADC_SQR2  | 0x48003134 | read-write | ADC regular sequence register 2 |
| ADC_SQR3  | 0x48003138 | read-write | ADC regular sequence register 3 |
| ADC_SQR4  | 0x4800313c | read-write | ADC regular sequence register 4 |
+-----------+------------+------------+---------------------------------+
```

- Getting information on all fields (peripheral: ADC2 register: CR) beginning by J

```
(gdb) svd info ADC2 ADC_CR J
+Fields----+----------+--------+-------------+
| name     | [msb:lsb] | access | description |
+----------+----------+--------+-------------+
| JADSTART | [3:3]     | None   | JADSTART    |
| JADSTP   | [5:5]     | None   | JADSTP      |
+----------+----------+--------+-------------+
```

4.3 Getting register(s) value(s)

This instruction provides the register list and field values.

```
(gdb) svd get [peripheral] [register]
```

Please refer to the following examples:

- Getting all the information for an entire peripheral

```
(gdb) svd get DLYBSD1
+Registers---+------------+------------
+-------------------------------------------------------------+
| name       | address    | value      |
fields                                                       |
+------------+------------+------------
+-------------------------------------------------------------+
| DLYB_CR    | 0x58006000 | 0x00000000 | DEN[0:0]=0x0 SEN[1:1]
=0x0                                   |
| DLYB_CFGR  | 0x58006004 | 0x80000000 | SEL[3:0]=0x0 UNIT[14:8]=0x0 LNG[27:16]=0x0 LNGF
[31:31]=0x1 |
| DLYB_VERR  | 0x580063f4 | 0x00000000 | MINREV[3:0]=0x0 MAJREV[7:4]
=0x0                              |
| DLYB_IPIDR | 0x580063f8 | 0x00000000 | ID[31:0]
=0x0                                                |
| DLYB_SIDR  | 0x580063fc | 0x00000000 | SID[31:0]
=0x0                                               |
+------------+------------+------------
+-------------------------------------------------------------+
```

- Getting information for just one register

```
(gdb) svd get DLYBSD1 DLYB_CFGR
+Registers--+-----------+-----------
+-------------------------------------------------------------+
| name      | address    | value      |
fields                                                       |
+-----------+------------+------------
+-------------------------------------------------------------+
| DLYB_CFGR | 0x58008004 | 0x00000000 | SEL[3:0]=0x0 UNIT[14:8]=0x0 LNG[27:16]=0x0 LNGF
[31:31]=0x0 |
+-----------+------------+------------
+-------------------------------------------------------------+
```

4.4 Writing register value

This instruction sets the register value.

```
(gdb) svd set <peripheral> <register> [field] <value>
```

Please refer to the following examples.

- Set register values

```
(gdb) svd set QUADSPI QUADSPI_LPTR 0x50
```

- Set field values

```
(gdb) svd set QUADSPI QUADSPI_LPTR TIMEOUT 0x10
```

5 References

- http://www.keil.com/pack/doc/CMSIS/SVD/html/index.html
- https://developer.arm.com/embedded/cmsis
- https://github.com/1udo6arre/svd-tools
- https://github.com/1udo6arre/svd-tools/blob/master/README.md