

# J-Link Device Support Kit

Supporting a new device usually requires a flashloader and in some cases also a script defining special connect and reset sequences, required by the device.

While SEGGER maintains updating the J-Link ([https://www.segger.com/products/debug-probes/j-link/?mtm\\_campaign=kb&mtm\\_kwd=DSK](https://www.segger.com/products/debug-probes/j-link/?mtm_campaign=kb&mtm_kwd=DSK)) software with new devices, still devices might be left out in the process.

To enable customers filling in the gaps on their own, the J-Link Device Support Kit (DSK) is offered.

To make creating the support as easy as possible, the J-Link DSK comes with the [SEGGER Flash Loader \(SFL\)](#) as well as a set of example script files for various devices that require special handling.

The development of SEGGER Flash Loader with J-Link DSK and SEGGER Embedded Studio ([https://www.segger.com/products/development-tools/embedded-studio/?mtm\\_campaign=kb&mtm\\_kwd=DSK](https://www.segger.com/products/development-tools/embedded-studio/?mtm_campaign=kb&mtm_kwd=DSK)) does not require a Embedded Studio commercial use license, [SEGGER's Friendly License](#) ([https://www.segger.com/purchase/licensing/license-sfl/?mtm\\_campaign=kb&mtm\\_kwd=DSK](https://www.segger.com/purchase/licensing/license-sfl/?mtm_campaign=kb&mtm_kwd=DSK)) is sufficient.

## Contents

### Adding a new device

[Inheriting from an existing device](#)

### Adding a flash loader

### JLinkDevices folder

### Device XML file

[XML-tag Database](#)

[XML-tag Device](#)

[XML-tag ChipInfo](#)

[XML-tag FlashBankInfo](#)

[XML-tag LoaderInfo](#)

### DSK availability

## Adding a new device

In order to add a new device to the J-Link software (meaning that the device name is selectable and known to the J-Link utilities like J-Link Commander ([https://www.segger.com/products/debug-probes/j-link/tools/j-link-commander/?mtm\\_campaign=kb&mtm\\_kwd=DSK](https://www.segger.com/products/debug-probes/j-link/tools/j-link-commander/?mtm_campaign=kb&mtm_kwd=DSK)), J-Flash ([https://www.segger.com/products/debug-probes/j-link/tools/j-flash/about-j-flash/?mtm\\_campaign=kb&mtm\\_kwd=DSK](https://www.segger.com/products/debug-probes/j-link/tools/j-flash/about-j-flash/?mtm_campaign=kb&mtm_kwd=DSK)), ...) and XML file with the following minimal contents need to be created:

```
<Database>
<Device>
  <ChipInfo Vendor="SEGGER" Name="SEGGER_Device0" Core="JLINK_CORE_CORTX_M0"/>
</Device>
<Device>
  <ChipInfo Vendor="SEGGER" Name="SEGGER_Device1" Core="JLINK_CORE_CORTX_M4"/>
</Device>
</Database>
```

More info, see [ChipInfo-tag](#)

In case the device requires a special connect / reset / ... sequence, a [J-Link script file](#) needs to be specified via [ChipInfo->JLinkScriptFile](#):

```
<Database>
<Device>
  <ChipInfo Vendor="SEGGER" Name="SEGGER_Device0" Core="JLINK_CORE_CORTX_M0" JLinkScriptFile="SEGGER/Example.jlinkscript"/>
</Device>
</Database>
```

## Inheriting from an existing device

Sometimes, multiple devices are added via the same XML file and many of these devices share many of their attributes.

For such cases, it can simplify things to use the [Device->InheritFrom](#) attribute to inherit all properties from the parent device and then overriding selective ones:

```
<Database>
<Device>
  <ChipInfo Vendor="SEGGER" Name="SEGGER_Device0" Core="JLINK_CORE_CORTX_M0" JLinkScriptFile="SEGGER/Example.jlinkscript"/>
</Device>
<Device InheritFrom="SEGGER_Device0">
  <ChipInfo Name="SEGGER_Device1"/>
</Device>
</Database>
```

## Adding a flash loader

More information about how to write an SFL, please refer to the [SEGGER Flash Loader knowledge base article](#)

A device entry may also be associated with 1 or more flash loaders for that device. In general, the following combinations are supported by the J-Link DSK:

- 1:0:0 = 1 device, 0 flash banks, 0 flash loaders associated
- 1:1:1 = 1 device, 1 flash bank, 1 flash loader associated
- 1:1:n = 1 device, 1 flash bank, n flash loaders associated
- 1:m:n = 1 device, n flash banks, m flash loader associated per flash bank (number of loaders may be different for each bank)

The 1:m:n association makes sense for complex devices that provide multiple flash banks and at least one of these banks may provide multiple configurations.

For example: Device with internal code flash, internal data flash, external QSPI flash.

While there is only 1 loader for the code flash and 1 for the data flash, the QSPI flash may be connected to various pin pairs and so the QSPI connection may differ from hardware to hardware.

For such cases, it makes sense to have multiple loaders available to select from, where each loader for example provides support for a specific pin configuration.

Based on the above example, a device entry could look like as follows:

```
<Database>
<Device>
  <ChipInfo Vendor="SEGGER" Name="SEGGER_Device0" WorkRAMAddr="0x20000000" WorkRAMSize="0x8000" Core="JLINK_CORE_CORTEX_M4" />
  <!-- 512 KB internal code flash bank with 1 Loader -->
  <FlashBankInfo Name="Internal code flash" BaseAddr="0x08000000" AlwaysPresent="1" >
    <LoaderInfo Name="Default" MaxSize="0x80000" Loader="Flashloader_Device0_InternalCodeFlash.elf" LoaderType="FLASH_ALGO_TYPE_OPEN" />
  </FlashBankInfo>
  <!-- 4 KB internal data flash bank with 1 Loader -->
  <FlashBankInfo Name="Internal data flash" BaseAddr="0x09000000" AlwaysPresent="1" >
    <LoaderInfo Name="Default" MaxSize="0x1000" Loader="Flashloader_Device0_InternalDataFlash.elf" LoaderType="FLASH_ALGO_TYPE_OPEN" />
  </FlashBankInfo>
  <!-- Max. 16 MB external QSPI flash with 3 Loaders. Real size + sectorization is determined by Loader at runtime -->
  <FlashBankInfo Name="Internal code flash" BaseAddr="0x60000000" AlwaysPresent="1" >
    <LoaderInfo Name="CLK@PIO0_D0@PIO1_D1@PIO2_D2@PIO3_D3@PIO4_CS@PIO5" MaxSize="0x1000000" Loader="Flashloader_Device0_ExternalQSPIFlash_PinConfig0.elf"
    LoaderType="FLASH_ALGO_TYPE_OPEN" />
    <LoaderInfo Name="CLK@PIO20_D0@PIO21_D1@PIO22_D2@PIO23_D3@PIO24_CS@PIO25" MaxSize="0x1000000" Loader="Flashloader_Device0_ExternalQSPIFlash_PinConfig1.elf"
    LoaderType="FLASH_ALGO_TYPE_OPEN" />
    <LoaderInfo Name="CLK@PIO30_D0@PIO31_D1@PIO32_D2@PIO33_D3@PIO34_CS@PIO35" MaxSize="0x1000000" Loader="Flashloader_Device0_ExternalQSPIFlash_PinConfig2.elf"
    LoaderType="FLASH_ALGO_TYPE_OPEN" />
  </FlashBankInfo>
</Device>
</Database>
```

More info, see [ChipInfo-tag](#), [FlashBankInfo-tag](#), [LoaderInfo-tag](#)

## JLinkDevices folder

In order to make the J-Link software aware of the new device, the previously created XML file need to be placed in the central **JLinkDevices** folder.

If the folder does not exist, it may be created.

The location of the central folder depends on the host OS, as well as on the active user on that OS:

OS	Location
Windows	C:\Users\<USER>\AppData\Roaming\SEGGER\JLinkDevices
Linux	\$HOME/.config/SEGGER/JLinkDevices
macOS	\$HOME/Library/Application Support/SEGGER/JLinkDevices

### Example of JLinkDevices folder structure

```
\---JLinkDevices
+---Vendor1
|   |   +---DevFamily1
|   |   |   Devices.xml
|   |   +---DevFamily2
|   |   |   Devices.xml
|   |   \---DevFamily3
|   |       Devices.xml
+---Vendor2
|   |   +---DevFamily1
|   |   |   Devices.xml
|   |   +---DevFamily2
|   |   |   Devices.xml
|   |   \---DevFamily3
|   |       Devices.xml
+---Vendor3
|   |   +---DevFamily1
|   |   |   Devices.xml
|   |   +---DevFamily2
|   |   |   Devices.xml
|   |   \---DevFamily3
|   |       Devices.xml
```

### Note:

1. This feature has been introduced with V7.62.
2. The J-Link DLL will recursively look through the JLinkDevices folder for any \*.xml files and evaluate them as JLinkDevices.xml files. Customers may freely structure the folder into subfolders with a maximum depth of 4 (four).

The process of placing (i.e. copying) the XML file(s) (and potential flash loaders, script files, ...) at the desired locations can be automated by using an installer.

The J-Link DSK comes with an installer template for Microsoft Windows.



The "Welcome" page of the Open Flashloader installer example

## Device XML file

J-Link Device XML files are used with the J-Link DSK to add support for new devices or extend support for existing devices. This section explains the structure, tags, attributes etc. of a J-Link Device XML file.

In the following, the valid XML tags and their possible attributes are explained.

General rules

- Attributes may only occur inside an opening tag
- Attribute values must be enclosed by quotation marks

Tag Description

Tag	Description
<Database>	Opens the XML file top-level tag.
<Device>	Opens the description for a new device.
<ChipInfo>	Specifies basic information about the device to be added, like the core it incorporates etc.
<FlashBankInfo>	Specifies a flash bank for the device.

### XML-tag Database

#### Example

TBD xxx

#### Explanation

Opens the XML file top-level tag. Only present once per XML file.

#### Attributes

This tag has no attributes **Notes**

- Must only occur once per XML file
- Must be closed via </Database>

### XML-tag Device

#### Example

TBD xxx

#### Explanation

Opens the description for a new device.

#### Attributes

Parameter	Meaning
<i>InheritFrom</i>	String that specifies the name of a parent device which the DLL already knows. The new child device will inherit all properties from the parent to use as a base. If the child inherits from a parent that was defined through *.XML, the parent and child must be defined in the same *.XML file and the parent must be defined before the child. Other configurations may lead to unspecified behavior. This attribute is optional. E.g. InheritFrom="STM32F767NI".

## Notes

- Must be closed via </Device> .
- May occur multiple times in an XML file

## XML-tag ChipInfo

### Example

TBD xxx

### Explanation

Specifies basic information about the device to be added, like the core it incorporates etc.

### Attributes

Parameter	Meaning
<i>Vendor</i>	String that specifies the name of the vendor of the device. This attribute is mandatory. E.g. Vendor="ST".
<i>Name</i>	Name of the device. This attribute is mandatory. E.g. Name="STM32F407IE"
<i>WorkRAMAddr</i>	Hexadecimal value that specifies the address of a RAM area that can be used by J-Link during flash programming etc. Should not be used by any DMAs on the device. Cannot exist without also specifying WorkRAMSize. If no flash banks are added for the new device, this attribute is optional. E.g. WorkRAMAddr=" 0x20000000 "
<i>WorkRAMSize</i>	Hexadecimal value that specifies the size of the RAM area that can be used by J-Link during flash programming etc. Cannot exist without also specifying WorkRAMAddr. If no flash banks are added for the new device, this attribute is optional. E.g. WorkRAMSize=" 0x10000 "
<i>Core</i>	Specifies the core that the device incorporates. If a new device added, this attribute is mandatory. E.g. Core="JLINK_CORE_CORTEX_M0 "For a list of valid attribute values, please refer to Attribute values - Core.
<i>JLinkScriptFile</i>	String that specifies the path to a J-Link script file if required for the device. Path can be relative or absolute. If path is relative, it is relative to the location of the JLinkDevices.xml file. This attribute is optional. E.g. JLinkScriptFile="ST/Example.jlinkscript"

## Notes

- No separate closing tag. Directly closed after attributes have been specified: <ChipInfo ... />
- Must not occur outside a <Device> tag.

### Attribute values - Core

For a list of all available values for the *Core* attribute, please expand the following list:

[Collapse]
JLINK_CORE_CORTEX_M1
JLINK_CORE_CORTEX_M3
JLINK_CORE_CORTEX_M3_R1P0
JLINK_CORE_CORTEX_M3_R1P1
JLINK_CORE_CORTEX_M3_R2P0
JLINK_CORE_CORTEX_M3_R2P1
JLINK_CORE_CORTEX_M0
JLINK_CORE_CORTEX_M_V8BASEL
JLINK_CORE_CORTEX_M4
JLINK_CORE_CORTEX_M7
JLINK_CORE_CORTEX_M_V8MAINL
JLINK_CORE_CORTEX_M23
JLINK_CORE_CORTEX_M33
JLINK_CORE_RV32
JLINK_CORE_RV64
JLINK_CORE_ARM7
JLINK_CORE_ARM7TDMI
JLINK_CORE_ARM7TDMI_R3
JLINK_CORE_ARM7TDMI_R4
JLINK_CORE_ARM7TDMI_S
JLINK_CORE_ARM7TDMI_S_R3
JLINK_CORE_ARM7TDMI_S_R4
JLINK_CORE_CORTEX_A8
JLINK_CORE_CORTEX_A7
JLINK_CORE_CORTEX_A9
JLINK_CORE_CORTEX_A12
JLINK_CORE_CORTEX_A15
JLINK_CORE_CORTEX_A17
JLINK_CORE_ARM9
JLINK_CORE_ARM9TDMI_S
JLINK_CORE_ARM920T
JLINK_CORE_ARM922T
JLINK_CORE_ARM926EJ_S
JLINK_CORE_ARM946E_S
JLINK_CORE_ARM966E_S
JLINK_CORE_ARM968E_S
JLINK_CORE_ARM11
JLINK_CORE_ARM1136
JLINK_CORE_ARM1136J
JLINK_CORE_ARM1136J_S
JLINK_CORE_ARM1136JF
JLINK_CORE_ARM1136JF_S
JLINK_CORE_ARM1156
JLINK_CORE_ARM1176
JLINK_CORE_ARM1176J
JLINK_CORE_ARM1176J_S
JLINK_CORE_ARM1176JF
JLINK_CORE_ARM1176JF_S
JLINK_CORE_CORTEX_R4
JLINK_CORE_CORTEX_R5
JLINK_CORE_RX
JLINK_CORE_RX62N
JLINK_CORE_RX62T
JLINK_CORE_RX63N
JLINK_CORE_RX630

JLINK_CORE_RX63T
JLINK_CORE_RX621
JLINK_CORE_RX62G
JLINK_CORE_RX631
JLINK_CORE_RX65N
JLINK_CORE_RX21A
JLINK_CORE_RX220
JLINK_CORE_RX230
JLINK_CORE_RX231
JLINK_CORE_RX23T
JLINK_CORE_RX24T
JLINK_CORE_RX110
JLINK_CORE_RX113
JLINK_CORE_RX130
JLINK_CORE_RX71M
JLINK_CORE_CORTEX_A5
JLINK_CORE_POWER_PC
JLINK_CORE_POWER_PC_N1
JLINK_CORE_POWER_PC_N2
JLINK_CORE_MIPS
JLINK_CORE_MIPS_M4K
JLINK_CORE_MIPS_MICROAPTIV
JLINK_CORE_EFM8_UNSPEC
JLINK_CORE_CIP51

## XML-tag FlashBankInfo

### Example

```
<Database>
<Device>
  <ChipInfo Vendor="SEGGER" Name="SEGGER_Device0" WorkRAMAddr="0x20000000" WorkRAMSize="0x8000" Core="JLINK_CORE_CORTEX_M4" />
  <!-- 512 KB internal code flash bank with 1 Loader -->
  <FlashBankInfo Name="Internal code flash" BaseAddr="0x08000000" AlwaysPresent="1" >
    <LoaderInfo Name="Default" MaxSize="0x80000" Loader="Flashloader_Device0_InternalCodeFlash.elf" LoaderType="FLASH_ALGO_TYPE_OPEN" />
  </FlashBankInfo>
</Device>
</Database>
```

### Explanation

Specifies a flash bank for the device. This allows to use the J-Link flash download functionality with IDEs, debuggers and other software that uses the J-Link DLL (e.g. J-Link Commander) for this device. The flash bank can then be programmed via the normal flash download functionality of the J-Link DLL. For more information about flash download, please refer to [Flash download](#) . For possible limitations etc. regarding newly added flash banks, please refer to [Add. Info / Considerations / Limitations](#) .

### Attributes

Attribute	Implementation	Type	Example	Explanation
Name	Mandatory	String	Name="SPIFI flash"	Specifies the name of the flash bank. Only used for visualization. Can be freely chosen.
BaseAddr	Mandatory	Hexadecimal value	BaseAddr="0x08000000"	Specifies the start address of the flash bank. The J-Link software uses this attribute together with MaxSize to determine which memory write accesses performed by the debugger, shall be redirected to the flash loader instead of being written directly to the target as normal memory access.
AlwaysPresent	Optional	Boolean (0/1)	AlwaysPresent="1"	Specifies if a flash bank is always present (e.g. internal flash). If this element is set to 1, this flash bank will be affected by the "erase" command.

### Notes

- Must not occur outside a <Device> tag

## XML-tag LoaderInfo

### Example

```

<Database>
<Device>
  <ChipInfo Vendor="SEGGER" Name="SEGGER_Device0" WorkRAMAddr="0x20000000" WorkRAMSize="0x8000" Core="JLINK_CORE_CORTEX_M4" />
  <!-- 512 KB internal code flash bank with 1 Loader -->
  <FlashBankInfo Name="Internal code flash" BaseAddr="0x08000000" AlwaysPresent="1" >
    <LoaderInfo Name="Default" MaxSize="0x80000" Loader="Flashloader_Device0_InternalCodeFlash.elf" LoaderType="FLASH_ALGO_TYPE_OPEN" />
  </FlashBankInfo>
</Device>
</Database>

```

## Explanation

Specifies a flash loader for the device.

## Attributes

Attribute	Implementation	Type	Example	Explanation
Name	Mandatory	String	Name="SPIFI flash"	E.g. Name="SPIFI flash loader" <b>Note:</b> Must be different from loader names of already existing loaders.
MaxSize	Optional	Hexadecimal value	MaxSize="0x08000000"	Hexadecimal value that specifies the max. size of the flash bank in bytes. For many flash loader types the real bank size may depend on the actual flash being connected (e.g. SPIFI flash where the loader can handle different SPIFI flashes so size may differ from hardware to hardware). Also, for some flash loaders the sectorization is extracted from the flash loader at runtime. The real size of the flash bank may be smaller than MaxSize but must never be bigger. The J-Link DLL uses this attribute together with BaseAddr to determine which memory write accesses performed by the debugger, shall be redirected to the flash loader instead of being written directly to the target as normal memory access. E.g. MaxSize="0x80000". May <b>NOT</b> be used together with the Sectorization attribute.
Sectorization	Optional	String	Sectorization="[32]0x800"	Specifies a sectorization that will be used to override the sectorization given with the flash loader ELF file. Format of sectorization information is: [NumSectors]SectSize,<Off>; [NumSectors]SectSize,<Off>;... The "<Off>" is optional and may be omitted if the sectors are continuous and there are no gaps between them. Otherwise, <Off> describes an offset relative(!) to <FlashBankBaseAddr> where the sector starts! May <b>NOT</b> be used together with the MaxSize attribute.
Loader	Mandatory	String	Loader="ST/MyFlashLoader.elf"	String that specifies path to the ELF file that holds the flash loader. Path can be relative or absolute. If path is relative, it is relative to the location of the JLinkDevices.xml file. E.g. Loader="ST/MyFlashLoader.elf" For CMSIS flash loaders the file extension is usually FLM, however any extension is accepted by the J-Link DLL.
LoaderType	Mandatory	FLASH_ALGO_TYPE_OPEN	LoaderType="FLASH_ALGO_TYPE_OPEN"	Specifies the type of the loader specified by Loader. E.g. LoaderType="FLASH_ALGO_TYPE_OPEN"

				For a list of valid attribute values, please refer to Attribute values LoaderType.
JLinkScriptFile	Optional	String	JLinkScriptFile="MyScript.JLinkScript"	Specifies a J-Link script file that is associated with the flash loader.

## Notes

- No separate closing tag. Directly closed after attributes have been specified: <FlashBankInfo ... />
- Must not occur outside a <FlashBankInfo> tag

## Attribute values - LoaderType

The following values are valid for the LoaderType attribute:

- FLASH\_ALGO\_TYPE\_OPEN  
Describes that the used algorithm is an Open Flashloader algorithm. CMSIS based algorithms are also supported via the Open Flashloader concept. For additional information, see Add. Info / Considerations / Limitations .

## DSK availability

---

To obtain the DSK with

- templates for SEGGER's high performance flash loaders
- functioning reference SEGGER high performance flash loaders in source code
- templates for device support installers
- ...

please contact [info@segger.com](mailto:info@segger.com)

For a free and open, yet lower performance flash loader option, CMSIS loaders (<https://open-cmsis-pack.github.io/Open-CMSIS-Pack-Spec/main/html/flashAlgorithm.html>) can be used.

Besides the SEGGER high performance flash loaders, also flash loaders in the CMSIS loader format are accepted by the J-Link devices XML files and software.

The procedure for adding a device stays exactly the same just generating the ELF file for the flash loader and its entry points are based on the CMSIS loader.

---

Retrieved from "[https://kb.segger.com/index.php?title=J-Link\\_Device\\_Support\\_Kit&oldid=27105](https://kb.segger.com/index.php?title=J-Link_Device_Support_Kit&oldid=27105)"

---

This page was last edited on 27 November 2025, at 16:58.