MCU DATA MODEL GAPS BETWEEN NXP AND OPEN CMSIS

PUBLIC

Holt Sun
Software Engineer
MARCH 2022

SECURE CONNECTIONS
FOR A SMARTER WORLD

- NXP data model introduction
- Gaps

NXP data model introduction

- NXP data model aims to cover all kinds SDK data
  - Component
  - Project
  - Configuration
  - Device
  - Board/kit
  - Dependency
  - MISC

- All our products: project/files/pack you download from NXP web or GitHub are auto generated based on NXP data.

Gaps

- Key concepts/features
  - ? Data driven project tree
    - Standalone/Linked project
- Component definition
  - ? Invisible component
  - ? Component type
- Project definition
  - On-going

And the situation here is that we may have gaps.

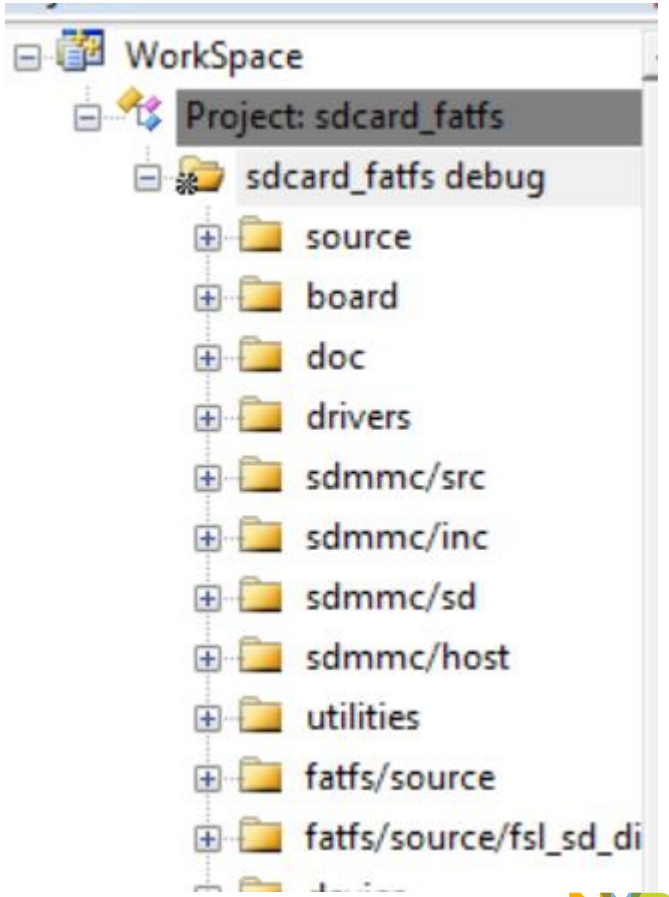
Gaps: key concepts/features

```
driver.gpio:
  contents:
    cc-include:
    - path: devices/MK64F12/drivers
      project_path: drivers
    files:
    - source: devices/MK64F12/drivers/fsl_gpio.h
      project_path: drivers
    - source: devices/MK64F12/drivers/fsl_gpio.c
      project_path: drivers
```

- Data driven project tree
- https://github.com/Open-CMSIS-Pack/Open-CMSIS-Pack-Spec/issues/95

In our data model, there are 2 paths for a source of both component/project.

1. Physical path of the source, tell where the file is
2. project_path of the source, tell where the file is in project tree



For example, NXP middlewares are usually put under “middlewares/<>/<>”, but there is no need to show “middleware” in the project tree, so our developers usually remove “middlewares” from “project_path”

In open cmsis

- For project
  - In cprj: “group” can work as “project_path”.
  - In cproject: “GroupType” can work as “project_path”
- **For component pdsc definition**
  - **No dedicated/similar attribute**

Gaps: key concepts/features

- Data driven project tree
- https://github.com/Open-CMSIS-Pack/Open-CMSIS-Pack-Spec/issues/95

We don’t think it is a good way to use Cclass/Cgroup/Csub to construct project tree like the in #95.

Under Cclass/Cgroup/Csub for this component, there will be all files flat which have following defects:
1. It totally hide hierarchy organization information for the source of a component which may confuse users
2. It looks not tidy/nice.

The key is that from perspective of data model definition, Cclass/Cgroup/Csub is mainly used for classification of component while "project_path" serves source organization.

They are not same concept.

Gaps: key concepts/features

- Standalone/linked project

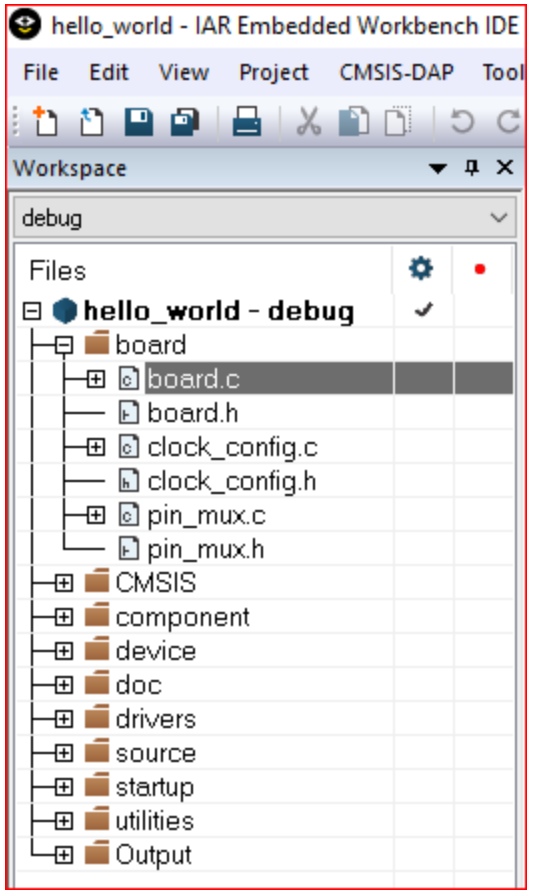
In the delivered packages from NXP to the market, we support 2 kinds projects for each example/demo

Linked Project And Copied Project

Linked project: project explorer view doesn't reflect physical source location. They are "virtual".

Copied project: project explorer view is exactly same as the physical source location. They are real.

NXP IDE/tool needs "project_path" to achieve this.



Gaps: component definition

- Invisible component
- https://github.com/Open-CMSIS-Pack/Open-CMSIS-Pack-Spec/issues/100

There are some cases that developers especially middleware owners want to intentionally hide certain intermediate component. For example, USB stack can be implemented into usb.controller required by usb.stack.common required by usb.stack.device/usb.stack.host/usb.stack.otg. Owner may only want to expose usb.stack.device/usb.stack.host/usb.stack.otg such end point components for users to select. There is not too much meaning to show intermediate components like usb.controller and usb.stack.common.

Gaps: component definition

- Component type
- https://github.com/Open-CMSIS-Pack/Open-CMSIS-Pack-Spec/issues/98

Every component should have its own type. Common types can be driver, middleware, OS, config. IDE/tools can has specific operations on certain component type.

Existing pdsc supports file level “attr” which can also be supported on component level.