CERN ABT’s lazy git workflow for equipment-specific designs

Pieter Van Trappen, Léa Strobino

2nd FPGA Developers’ Forum – https://indico.cern.ch/event/1467417

Presentation content

1. **Context**
2. **Project structure**
3. **Examples and use cases**
4. **Future**

Motivation: present the current, sub-optimal but relatively ~~lazy~~ efficient, git workflow hoping for ideas for improvement. Or maybe inspire others. All sources available to CERN colleagues, open to share publicly.

Context – CERN SY-ABT

Accelerator Systems, Beam Transfer Group (SY-ABT)

- The ABT group is in charge of the design, development, construction, installation, exploitation and maintenance of injection and extraction related equipment and beam-transfer systems.

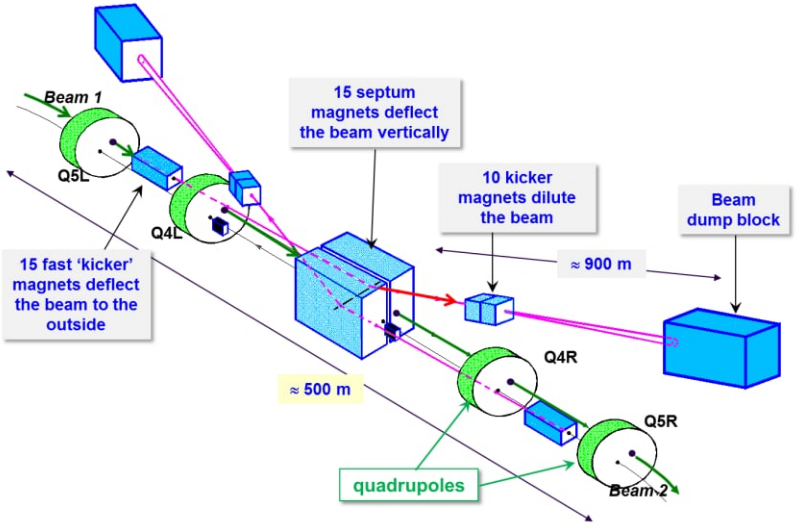
The electronics section designs, among others, cards and gateware for such ‘fast pulsed transfer systems’.

Wide variety of hardware of the ‘70s being consolidated and newer hardware; due to protection latency and timing synchronisation FPGAs in use since 20+ years.

ABT handles about 50 installations (aka equipment) and strives for reusable, modular designs for obvious reasons (limited time, knowledge retention, reduce support vector).

Main focus here will be on two projects, though there’s more with similar structure

- A Fast Interlock Detection System for high-power switch protection (FIDS) - paper
- A Generic Timing Software/Gateware for fast pulsed magnet systems (KiTS/R) - paper


15 septum magnets deflect the beam vertically

10 kicker magnets dilute the beam

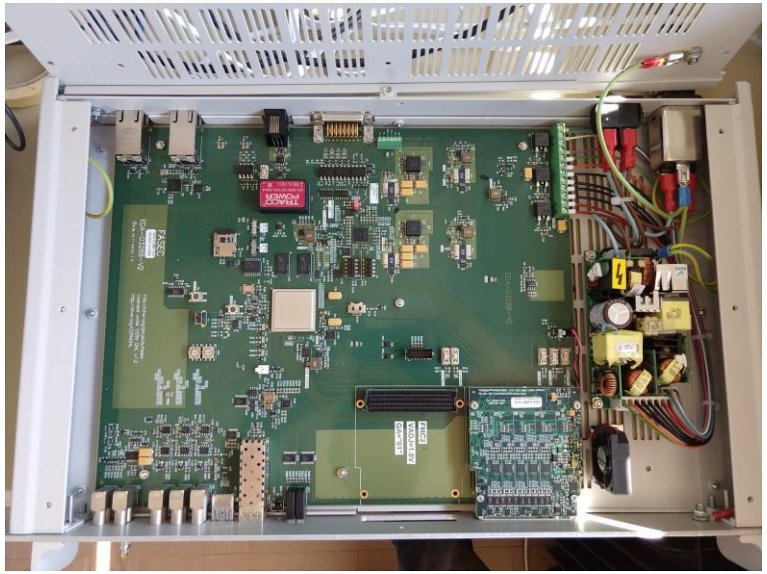
Beam dump block

15 fast ‘kicker’ magnets deflect the beam to the outside

Context – Hardware

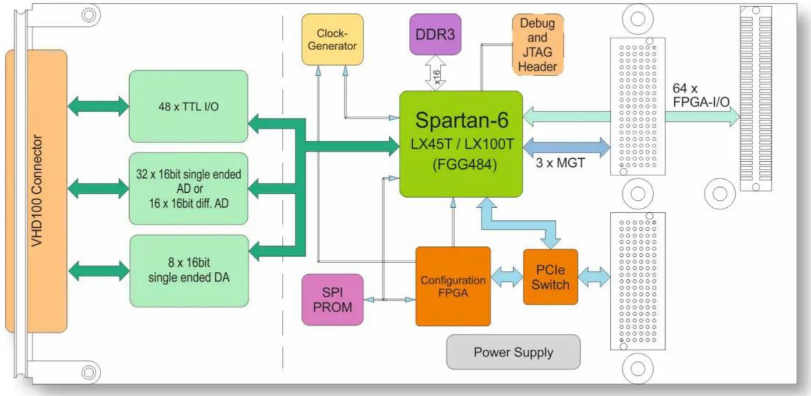
FIDS

- OHWR FASEC (FPGA & ARM SoC FMC Carrier) platform
- AMD Zynq-7000 SoC

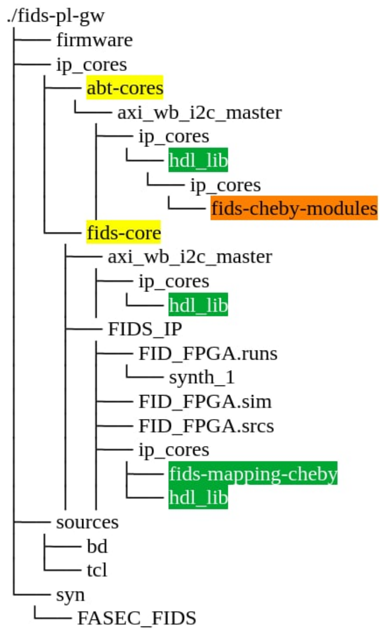


KiTR

- Uses TEWS' TXMC635 'mixed GPIO' XMC, hosted on VME or PCIe carriers
- Reconfigurable FPGA with Analog Input (16 bit), Analog Output (16 bit), and TTL I/O
- AMD Spartan-6 FPGA



Project structure

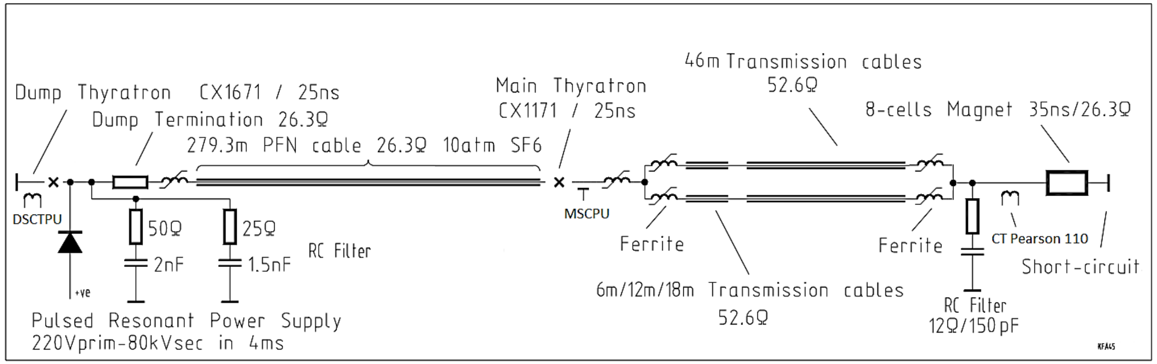


```
./fids-pl-gw
├── firmware
├── ip_cores
│   ├── abt-cores
│   │   └── axi_wb_i2c_master
│   │       ├── ip_cores
│   │           └── hdl_lib
│   │               └── ip_cores
│   │                   └── fids-cheby-modules
│   └── fids-core
│       ├── axi_wb_i2c_master
│       │   ├── ip_cores
│       │       └── hdl_lib
│       ├── FIDS_IP
│       │   ├── FID_FPGA.runs
│       │   │   └── synth_1
│       │   ├── FID_FPGA.sim
│       │   ├── FID_FPGA.srcs
│       │   ├── ip_cores
│       │   │   ├── fids-mapping-cheby
│       │   │   └── hdl_lib
├── sources
│   ├── bd
│   └── tcl
├── syn
    └── FASEC_FIDS
```

- **Our typical gateware project structure is composed as follows**
  - one or multiple common and dedicated IP/HDL cores
  - a top-level design, specific for each installation
    - specific: HW IO interconnections, cores instantiation, (cheby) memmap, AXI4 interconnects, etc.
    - common: HW IO interface, abt_card_header, submodules, etc.
- **Some projects predate wider ABT/CERN git adaptation**
  - No use of git or any other version control system; common files were copied around (though often modified).
  - Some had an IP-based structure, but such ‘common cores’ were managed at the level of the designer’s HDD with a common, single folder for multiple projects.
    - No use of git submodules thus once such a dependency had advanced for one project, all remaining projects required implementing similar API changes (e.g. modules’ port); zero reproducibility.
  - Recently we’ve seen substantial efforts to migrate to the common ‘modern’ approach i.e. **git submodules with a GitLab CI/CD pipeline** to obtain (~reproducible) bitstreams.

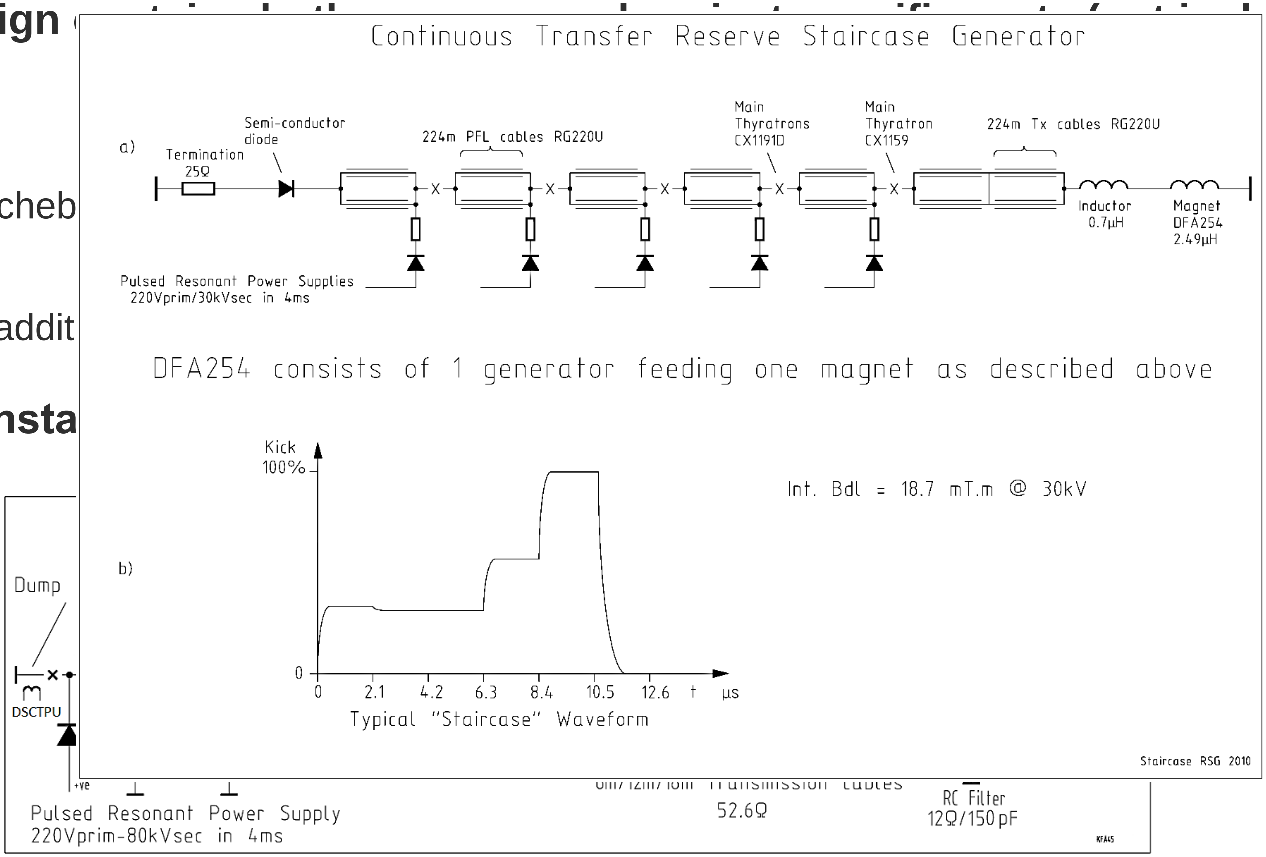
Project structure

- **ABT gateware projects are very similar but still different!**
- **The project’s top-level design contains both common and <u>project-specific parts</u> (not including the submodules), such as**
    - HW IO interconnections
    - cores instantiation, through (cheby) memmap
    - AXI4 interconnects
    - project peculiarities such as additional gating, muxing, etc.
- **As such, for each project/installation there’s a dedicated git branch.**


Dump Thyratron CX1671 / 25ns
Dump Termination 26.3Ω
279.3m PFN cable 26.3Ω 10atm SF6
Main Thyratron CX1171 / 25ns
46m Transmission cables 52.6Ω
8-cells Magnet 35ns/26.3Ω
DSCTPU
MSCPU
50Ω
2nF
25Ω
1.5nF
RC Filter
+ve
Ferrite
Ferrite
CT Pearson 110
Short-circuit
6m/12m/18m Transmission cables 52.6Ω
RC Filter 12Ω/150pF
Pulsed Resonant Power Supply 220Vprim-80kVsec in 4ms
KFA45

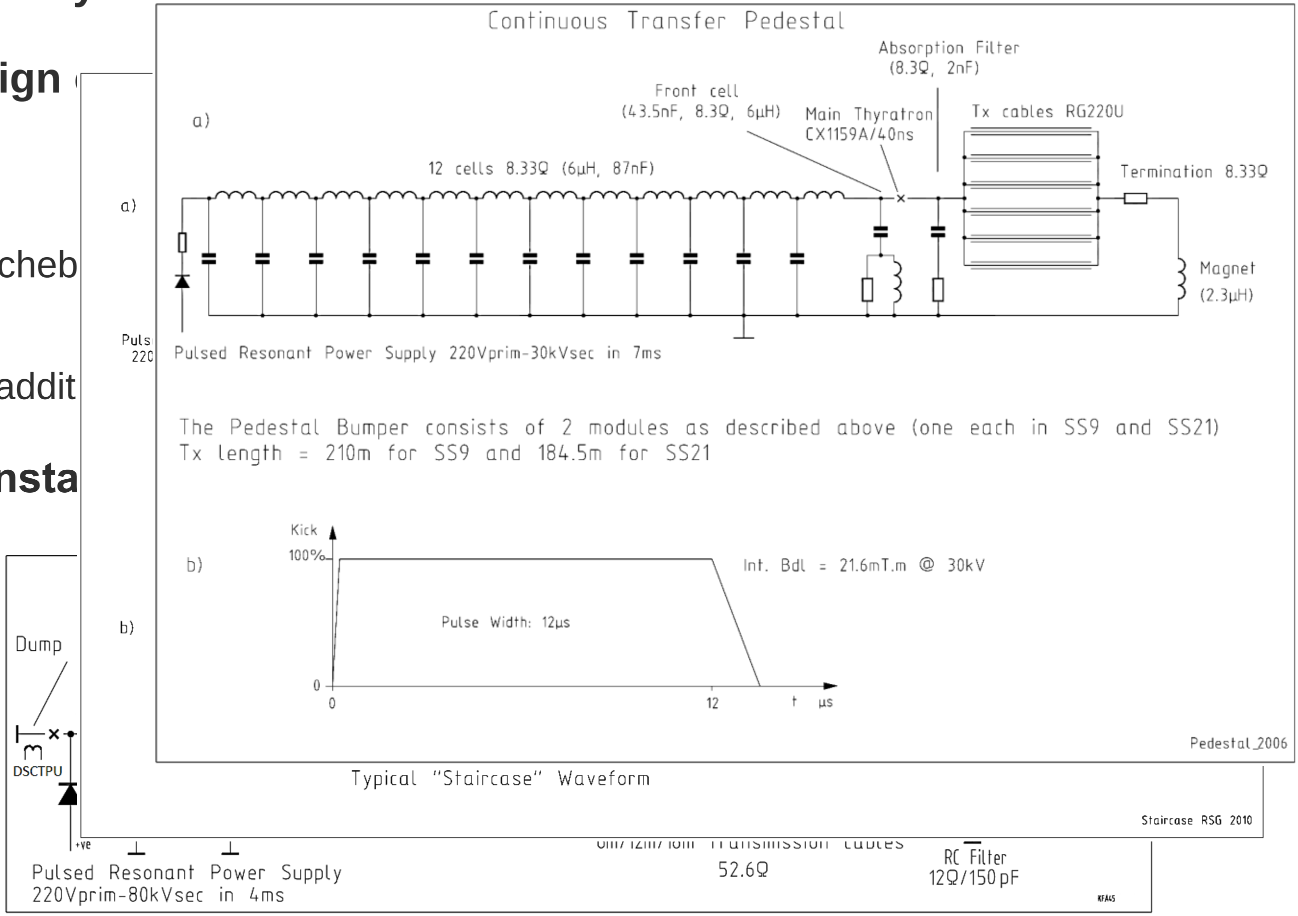
Project structure

- **ABT gateware projects are very similar but still different!**
- **The project's top-level design c… …uding the submodules), such as**
  - HW IO interconnections
  - cores instantiation, through (cheb
  - AXI4 interconnects
  - project peculiarities such as addit
- **As such, for each project/insta**



Project structure

- **ABT gateware projects are very similar but still different!**
- **The project’s top-level design** **ling the submodules), such as**
  - HW IO interconnections
  - cores instantiation, through (cheb
  - AXI4 interconnects
  - project peculiarities such as addit
- **As such, for each project/insta**



The Pedestal Bumper consists of 2 modules as described above (one each in SS9 and SS21)
Tx length = 210m for SS9 and 184.5m for SS21

Project structure – *branch-per-installation*

Pros

- **Apply changes to the common part relatively fast using `git cherry-pick`**
- **No need to abstract away all peculiarities into a generic module (in a submodule)**
- **Common GitLab CI/CD repository pipeline configuration**

Cons

- **Cherry-pick not as fast as rebase and still manual, time consuming e.g. 10+ branches for FIDS**
- **Not all commits apply cleanly, need to pay attention to git commit best practices**
- **Difficult to use git tags**
  - Tags are unique per repo, thus tags à la \`v1.2.3\` for such a *branch-per-installation* IP repo cannot be used; workaround being branch-name prefix à la \`foo_v1.2.3\`.
  - GitLab's **multi-project pipeline** does not work reliably with **tag pipelines** (`if: $CI_COMMIT_TAG`) which one commonly would use to trigger the downstream build → downstream will be triggered but <u>fetch the latest BRANCH artifacts, not necessarily the TAG ones</u>.

Use case – TXMC635 semi-tag pipeline

- **Project where gateware commits/tags trigger downstream softcore firmware compilation, to be included in bitstream**

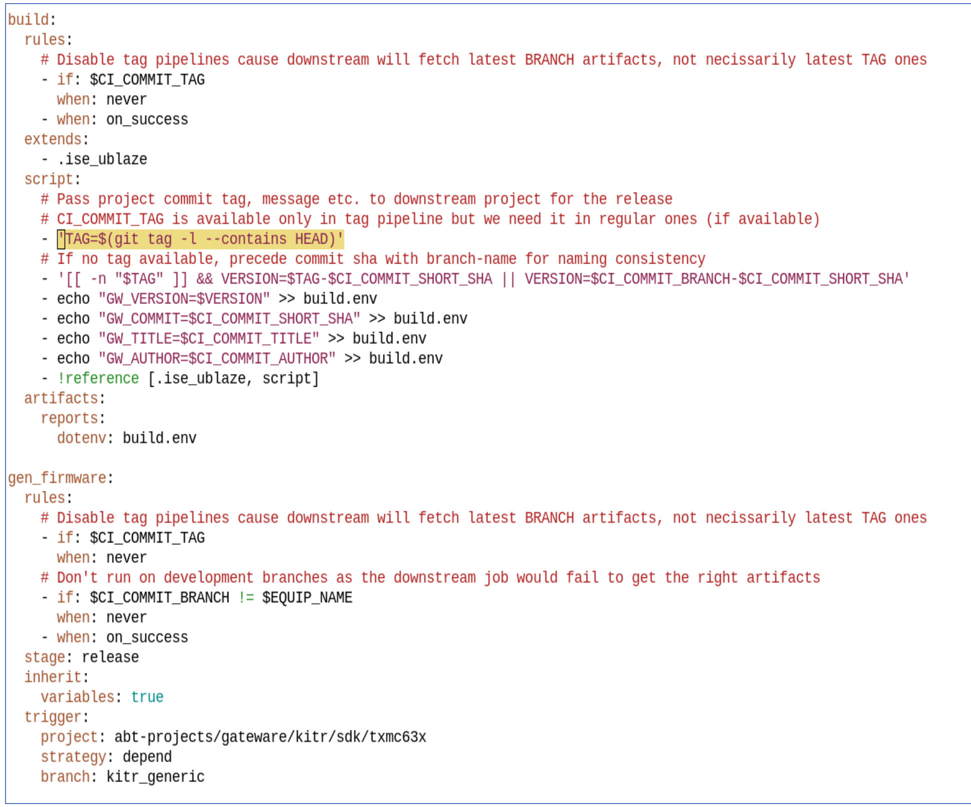
- **GitLab's multi-project pipeline incompatible with tag pipelines → downstream will be triggered but fetch the latest BRANCH artifacts, not necessarily the latest TAG ones**

- **Current workaround for such multi-project pipelines**
  - Disable tag pipelines
  - Custom job rules
  - Push atomically
  - `git push --atomic origin <branch> <tag>`

- **To-do: make a GitLab feature request for tag support for `needs: ref:`, similarly to `$CI_MERGE_REQUEST_REF_PATH` (docs)**

Use case – TXMC635 semi-tag pipeline

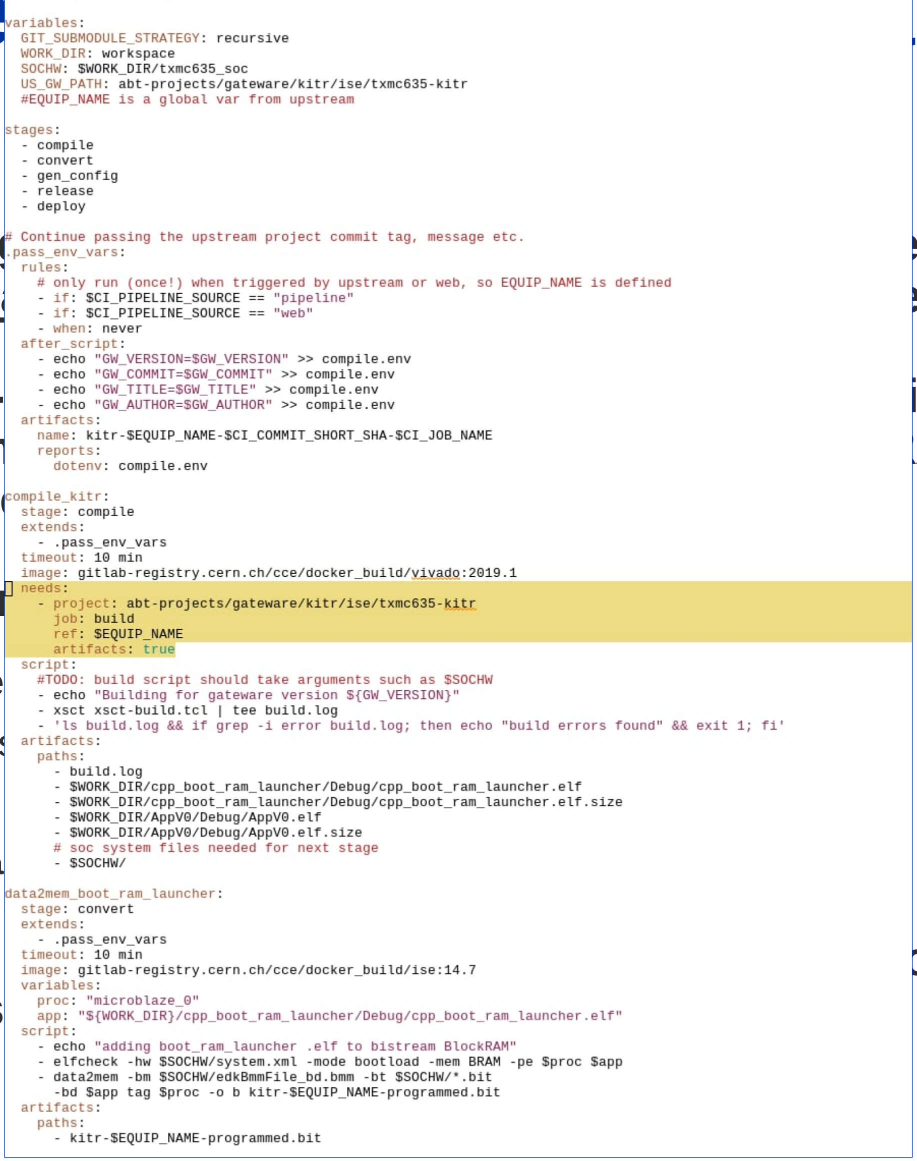
- **Project where gateware commits/tags trigger downstream softcore firmware compilation, to be included in bitstream**
- **GitLab's multi-project pipeline incompatible with tag pipelines → downstream will be triggered but fetch the latest BRANCH artifacts, not necessarily the latest TAG ones**
- **Current workaround for such multi-project pipelines**
  - Disable tag pipelines
  - Custom job rules
  - Push atomically
  - `git push --atomic origin <branch> <tag>`
- **To-do: make a GitLab feature request for tag support for `needs: ref:`, similarly to `$CI_MERGE_REQUEST_REF_PATH` (docs)**



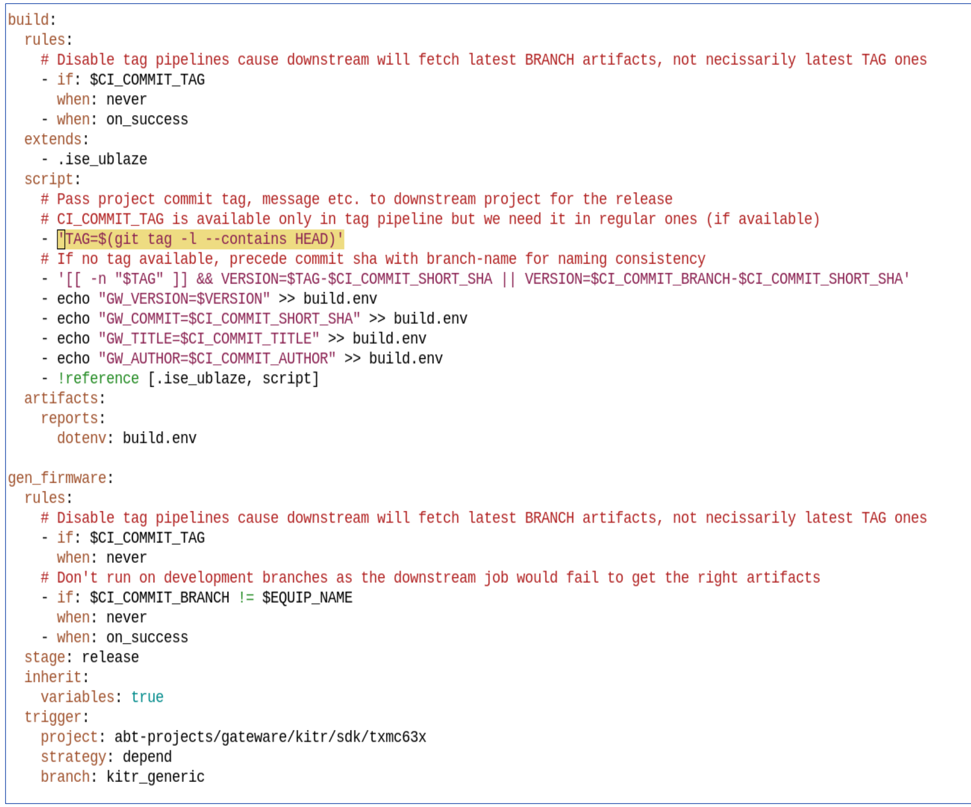
```
build:
  rules:
    # Disable tag pipelines cause downstream will fetch latest BRANCH artifacts, not necissarily latest TAG ones
    - if: $CI_COMMIT_TAG
      when: never
    - when: on_success
  extends:
    - .ise_ublaze
  script:
    # Pass project commit tag, message etc. to downstream project for the release
    # CI_COMMIT_TAG is available only in tag pipeline but we need it in regular ones (if available)
    - 'TAG=$(git tag -l --contains HEAD)'
    # If no tag available, precede commit sha with branch-name for naming consistency
    - '[[ -n "$TAG" ]] && VERSION=$TAG-$CI_COMMIT_SHORT_SHA || VERSION=$CI_COMMIT_BRANCH-$CI_COMMIT_SHORT_SHA'
    - echo "GW_VERSION=$VERSION" >> build.env
    - echo "GW_COMMIT=$CI_COMMIT_SHORT_SHA" >> build.env
    - echo "GW_TITLE=$CI_COMMIT_TITLE" >> build.env
    - echo "GW_AUTHOR=$CI_COMMIT_AUTHOR" >> build.env
    - !reference [.ise_ublaze, script]
  artifacts:
    reports:
      dotenv: build.env

gen_firmware:
  rules:
    # Disable tag pipelines cause downstream will fetch latest BRANCH artifacts, not necissarily latest TAG ones
    - if: $CI_COMMIT_TAG
      when: never
    # Don't run on development branches as the downstream job would fail to get the right artifacts
    - if: $CI_COMMIT_BRANCH != $EQUIP_NAME
      when: never
    - when: on_success
  stage: release
  inherit:
    variables: true
  trigger:
    project: abt-projects/gateware/kitr/sdk/txmc63x
    strategy: depend
    branch: kitr_generic
```

Use case g pipeline

- Project where eam
  softcore firmwa eam
- GitLab's multi- ipelines
  → downstream ANCH
  artifacts, not n
- Current workar
  - Disable tag pipe
  - Custom job rules
  - Push atomically
  - `git push --a`
- To-do: make a or
  `needs: ref:`, s (docs)



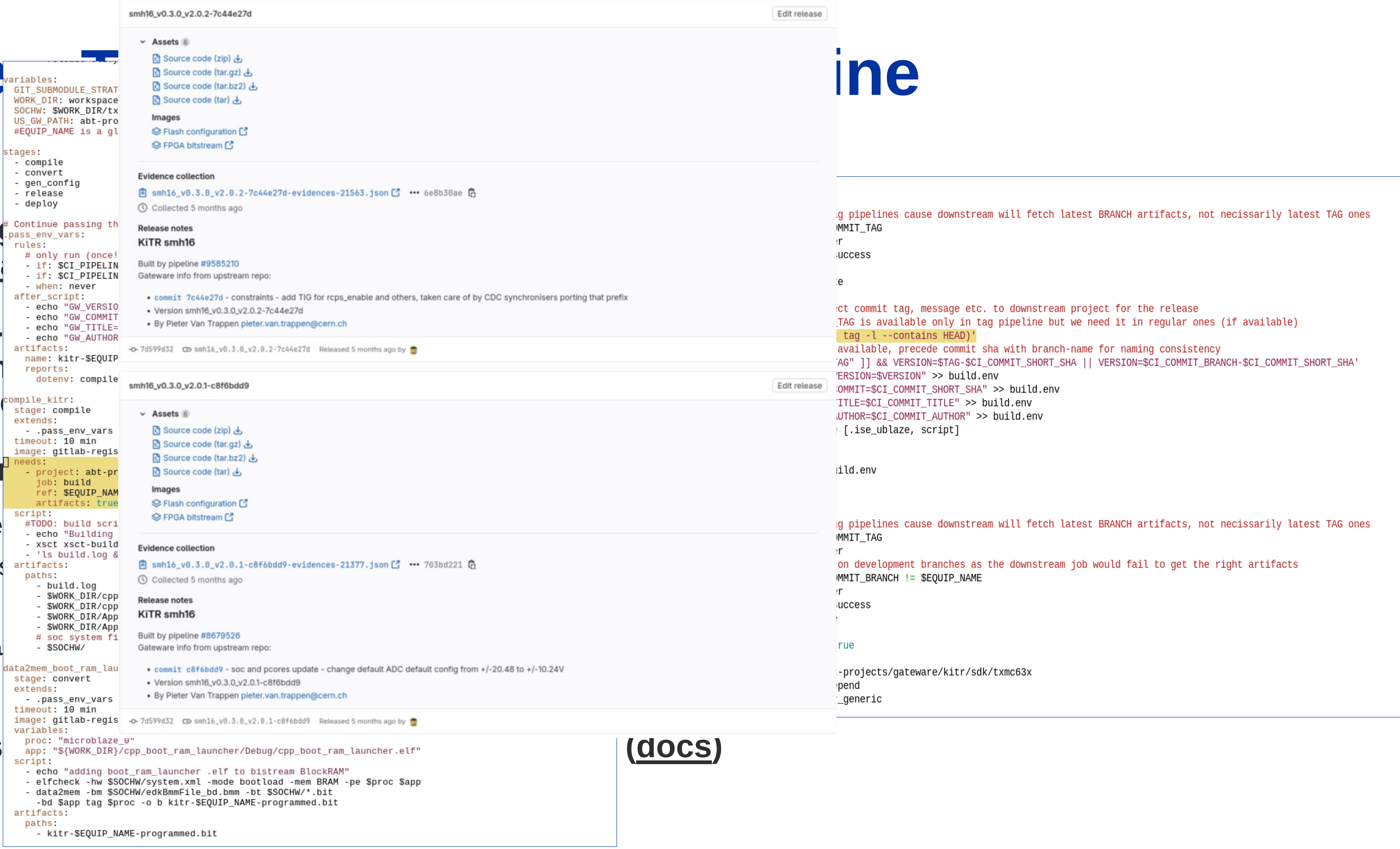
```yaml
variables:
  GIT_SUBMODULE_STRATEGY: recursive
  WORK_DIR: workspace
  SOCHW: $WORK_DIR/txmc635_soc
  US_GW_PATH: abt-projects/gateware/kitr/ise/txmc635-kitr
  #EQUIP_NAME is a global var from upstream

stages:
  - compile
  - convert
  - gen_config
  - release
  - deploy

# Continue passing the upstream project commit tag, message etc.
.pass_env_vars:
  rules:
    # only run (once!) when triggered by upstream or web, so EQUIP_NAME is defined
    - if: $CI_PIPELINE_SOURCE == "pipeline"
    - if: $CI_PIPELINE_SOURCE == "web"
    - when: never
  after_script:
    - echo "GW_VERSION=$GW_VERSION" >> compile.env
    - echo "GW_COMMIT=$GW_COMMIT" >> compile.env
    - echo "GW_TITLE=$GW_TITLE" >> compile.env
    - echo "GW_AUTHOR=$GW_AUTHOR" >> compile.env
  artifacts:
    name: kitr-$EQUIP_NAME-$CI_COMMIT_SHORT_SHA-$CI_JOB_NAME
    reports:
      dotenv: compile.env

compile_kitr:
  stage: compile
  extends:
    - .pass_env_vars
  timeout: 10 min
  image: gitlab-registry.cern.ch/cce/docker_build/vivado:2019.1
  needs:
    - project: abt-projects/gateware/kitr/ise/txmc635-kitr
      job: build
      ref: $EQUIP_NAME
      artifacts: true
  script:
    #TODO: build script should take arguments such as $SOCHW
    - echo "Building for gateware version ${GW_VERSION}"
    - xsct xsct-build.tcl | tee build.log
    - 'ls build.log && if grep -i error build.log; then echo "build errors found" && exit 1; fi'
  artifacts:
    paths:
      - build.log
      - $WORK_DIR/cpp_boot_ram_launcher/Debug/cpp_boot_ram_launcher.elf
      - $WORK_DIR/cpp_boot_ram_launcher/Debug/cpp_boot_ram_launcher.elf.size
      - $WORK_DIR/AppV0/Debug/AppV0.elf
      - $WORK_DIR/AppV0/Debug/AppV0.elf.size
      # soc system files needed for next stage
      - $SOCHW/

data2mem_boot_ram_launcher:
  stage: convert
  extends:
    - .pass_env_vars
  timeout: 10 min
  image: gitlab-registry.cern.ch/cce/docker_build/ise:14.7
  variables:
    proc: "microblaze_0"
    app: "${WORK_DIR}/cpp_boot_ram_launcher/Debug/cpp_boot_ram_launcher.elf"
  script:
    - echo "adding boot_ram_launcher .elf to bistream BlockRAM"
    - elfcheck -hw $SOCHW/system.xml -mode bootload -mem BRAM -pe $proc $app
    - data2mem -bm $SOCHW/edkBmmFile_bd.bmm -bt $SOCHW/*.bit
      -bd $app tag $proc -o b kitr-$EQUIP_NAME-programmed.bit
  artifacts:
    paths:
      - kitr-$EQUIP_NAME-programmed.bit
```



```yaml
build:
  rules:
    # Disable tag pipelines cause downstream will fetch latest BRANCH artifacts, not necissarily latest TAG ones
    - if: $CI_COMMIT_TAG
      when: never
    - when: on_success
  extends:
    - .ise_ublaze
  script:
    # Pass project commit tag, message etc. to downstream project for the release
    # CI_COMMIT_TAG is available only in tag pipeline but we need it in regular ones (if available)
    - 'TAG=$(git tag -l --contains HEAD)'
    # If no tag available, precede commit sha with branch-name for naming consistency
    - '[[ -n "$TAG" ]] && VERSION=$TAG-$CI_COMMIT_SHORT_SHA || VERSION=$CI_COMMIT_BRANCH-$CI_COMMIT_SHORT_SHA'
    - echo "GW_VERSION=$VERSION" >> build.env
    - echo "GW_COMMIT=$CI_COMMIT_SHORT_SHA" >> build.env
    - echo "GW_TITLE=$CI_COMMIT_TITLE" >> build.env
    - echo "GW_AUTHOR=$CI_COMMIT_AUTHOR" >> build.env
    - !reference [.ise_ublaze, script]
  artifacts:
    reports:
      dotenv: build.env

gen_firmware:
  rules:
    # Disable tag pipelines cause downstream will fetch latest BRANCH artifacts, not necissarily latest TAG ones
    - if: $CI_COMMIT_TAG
      when: never
    # Don't run on development branches as the downstream job would fail to get the right artifacts
    - if: $CI_COMMIT_BRANCH != $EQUIP_NAME
      when: never
    - when: on_success
  stage: release
  inherit:
    variables: true
  trigger:
    project: abt-projects/gateware/kitr/sdk/txmc63x
    strategy: depend
    branch: kitr_generic
```

Use case ine

- Project where softcore firmw
- GitLab's multi- → downstream artifacts, not n
- Current workar
  - Disable tag pipe
  - Custom job rules
  - Push atomically
  - `git push --a`
- To-do: make a `needs: ref:`, s (docs)



```
variables:
  GIT_SUBMODULE_STRAT
  WORK_DIR: workspace
  SOCHW: $WORK_DIR/tx
  US_GW_PATH: abt-pro
  #EQUIP_NAME is a gl

stages:
  - compile
  - convert
  - gen_config
  - release
  - deploy

# Continue passing th
.pass_env_vars:
  rules:
    # only run (once!
    - if: $CI_PIPELIN
    - if: $CI_PIPELIN
    - when: never
  after_script:
    - echo "GW_VERSIO
    - echo "GW_COMMIT
    - echo "GW_TITLE=
    - echo "GW_AUTHOR
  artifacts:
    name: kitr-$EQUIP
    reports:
      dotenv: compile

compile_kitr:
  stage: compile
  extends:
    - .pass_env_vars
  timeout: 10 min
  image: gitlab-regis
  needs:
    - project: abt-pr
      job: build
      ref: $EQUIP_NAM
      artifacts: true
  script:
    #TODO: build scri
    - echo "Building
    - xsct xsct-build
    - 'ls build.log &
  artifacts:
    paths:
      - build.log
      - $WORK_DIR/cpp
      - $WORK_DIR/cpp
      - $WORK_DIR/App
      - $WORK_DIR/App
      # soc system fi
      - $SOCHW/

data2mem_boot_ram_lau
  stage: convert
  extends:
    - .pass_env_vars
  timeout: 10 min
  image: gitlab-regis
  variables:
    proc: "microblaze_0"
    app: "${WORK_DIR}/cpp_boot_ram_launcher/Debug/cpp_boot_ram_launcher.elf"
  script:
    - echo "adding boot_ram_launcher .elf to bistream BlockRAM"
    - elfcheck -hw $SOCHW/system.xml -mode bootload -mem BRAM -pe $proc $app
    - data2mem -bm $SOCHW/edkBmmFile_bd.bmm -bt $SOCHW/*.bit
      -bd $app tag $proc -o b kitr-$EQUIP_NAME-programmed.bit
  artifacts:
    paths:
      - kitr-$EQUIP_NAME-programmed.bit
```

smh16_v0.3.0_v2.0.2-7c44e27d

Assets 6

- Source code (zip)
- Source code (tar.gz)
- Source code (tar.bz2)
- Source code (tar)

Images

- Flash configuration
- FPGA bitstream

Evidence collection

smh16_v0.3.0_v2.0.2-7c44e27d-evidences-21563.json ••• 6e8b30ae

Collected 5 months ago

Release notes

KiTR smh16

Built by pipeline #9585210
Gateware info from upstream repo:

- commit 7c44e27d - constraints - add TIG for rcps_enable and others, taken care of by CDC synchronisers porting that prefix
- Version smh16_v0.3.0_v2.0.2-7c44e27d
- By Pieter Van Trappen pieter.van.trappen@cern.ch

7d599d32 smh16_v0.3.0_v2.0.2-7c44e27d Released 5 months ago by

smh16_v0.3.0_v2.0.1-c8f6bdd9

Assets 6

- Source code (zip)
- Source code (tar.gz)
- Source code (tar.bz2)
- Source code (tar)

Images

- Flash configuration
- FPGA bitstream

Evidence collection

smh16_v0.3.0_v2.0.1-c8f6bdd9-evidences-21377.json ••• 703bd221

Collected 5 months ago

Release notes

KiTR smh16

Built by pipeline #8679526
Gateware info from upstream repo:

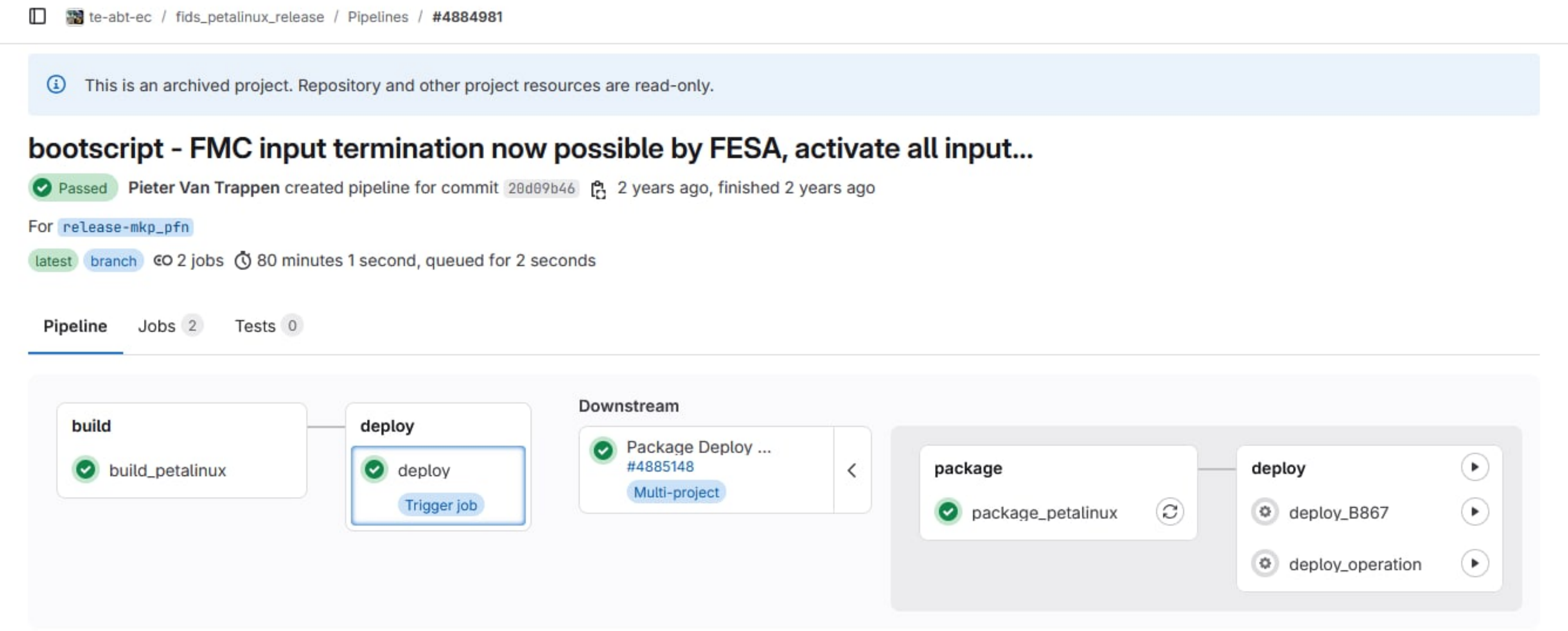
- commit c8f6bdd9 - soc and pcores update - change default ADC default config from +/-20.48 to +/-10.24V
- Version smh16_v0.3.0_v2.0.1-c8f6bdd9
- By Pieter Van Trappen pieter.van.trappen@cern.ch

7d599d32 smh16_v0.3.0_v2.0.1-c8f6bdd9 Released 5 months ago by

```
g pipelines cause downstream will fetch latest BRANCH artifacts, not necissarily latest TAG ones
MMIT_TAG
r
uccess

e

ct commit tag, message etc. to downstream project for the release
TAG is available only in tag pipeline but we need it in regular ones (if available)
 tag -l --contains HEAD)'
available, precede commit sha with branch-name for naming consistency
AG" ]] && VERSION=$TAG-$CI_COMMIT_SHORT_SHA || VERSION=$CI_COMMIT_BRANCH-$CI_COMMIT_SHORT_SHA'
ERSION=$VERSION" >> build.env
OMMIT=$CI_COMMIT_SHORT_SHA" >> build.env
ITLE=$CI_COMMIT_TITLE" >> build.env
UTHOR=$CI_COMMIT_AUTHOR" >> build.env
 [.ise_ublaze, script]


ild.env



g pipelines cause downstream will fetch latest BRANCH artifacts, not necissarily latest TAG ones
MMIT_TAG
r
 on development branches as the downstream job would fail to get the right artifacts
MMIT_BRANCH != $EQUIP_NAME
r
uccess


rue

-projects/gateware/kitr/sdk/txmc63x
pend
_generic
```

Use case – fids-ps-sw (old pipeline)

- Downstream pipeline from gateware, one branch per installation
- **80 minutes** to build the software, must run <u>for each branch</u> (10+)
- Triggers one downstream pipeline to package and deploy
- Not reproducible


te-abt-ec / fids_petalinux_release / Pipelines / #4884981

This is an archived project. Repository and other project resources are read-only.

bootscript - FMC input termination now possible by FESA, activate all input...

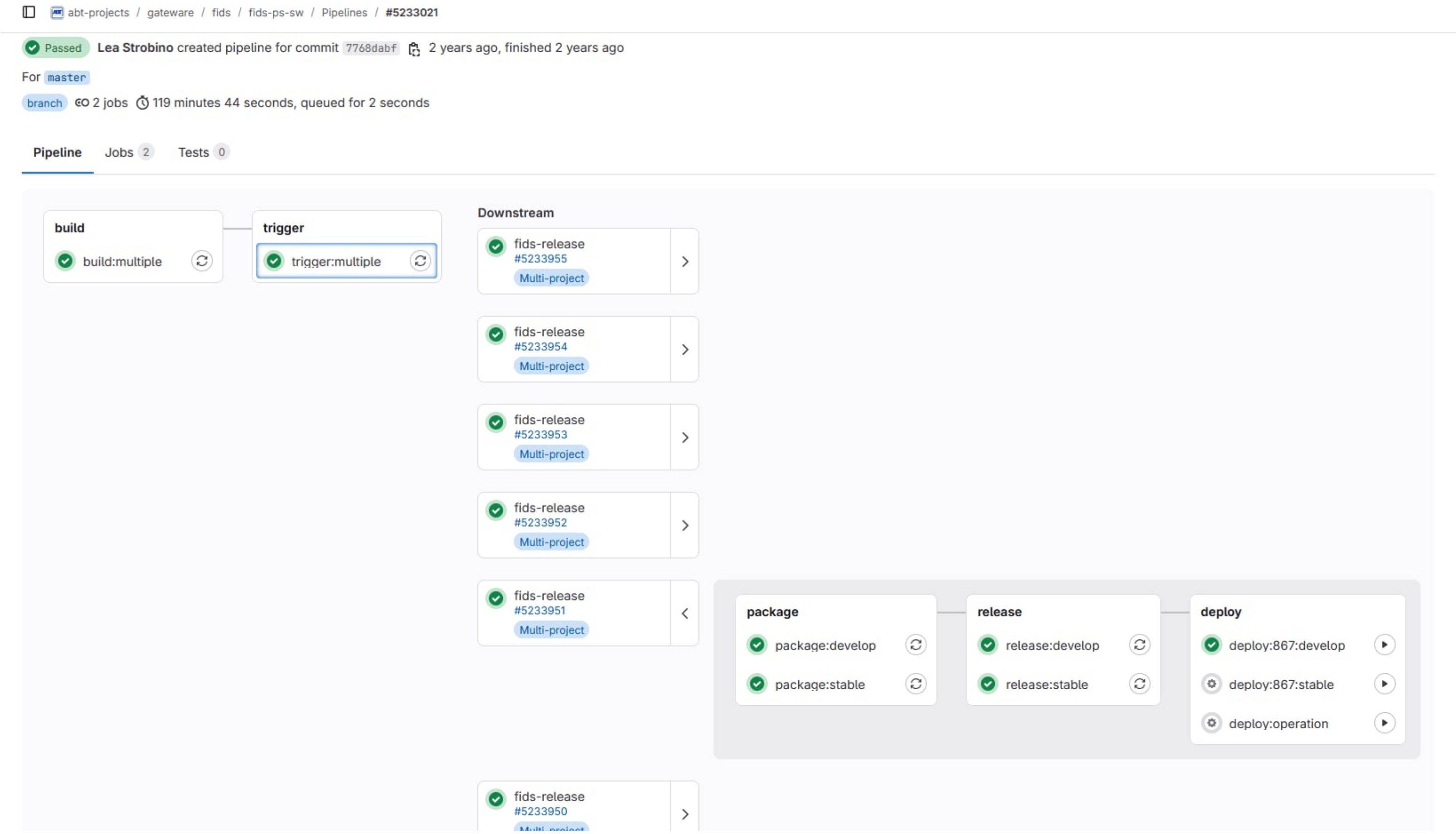
Passed **Pieter Van Trappen** created pipeline for commit `20d09b46` 2 years ago, finished 2 years ago

For `release-mkp_pfn`

latest branch 2 jobs 80 minutes 1 second, queued for 2 seconds

Use case – fids-ps-sw (new pipeline)

- Single branch, loops over a list of installations
- Configures the Yocto recipes and submodules branches within the recipes
- Saves identification files with git hashes of all dependencies
- **120 minutes** to build all 9 “branches”
- Triggers one downstream pipeline per “branch” to package and deploy
- More reproducibility (docs)



abt-projects / gateware / fids / fids-ps-sw / Pipelines / #5233021

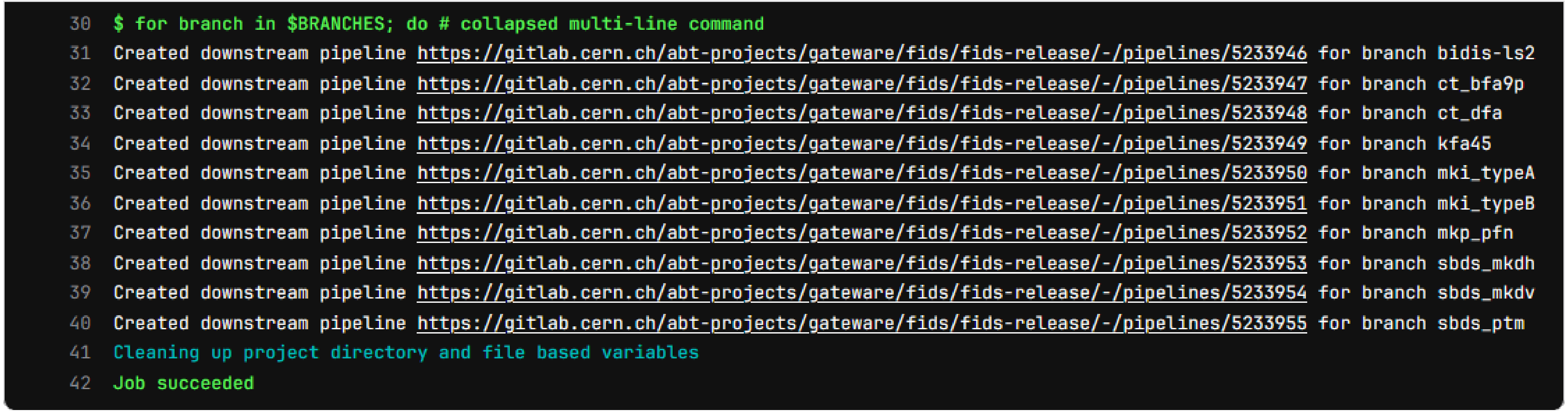
Passed Lea Strobino created pipeline for commit 7768dabf 2 years ago, finished 2 years ago

For master

branch 2 jobs 119 minutes 44 seconds, queued for 2 seconds

Use case – fids-ps-sw (new pipeline)

- Additional ‘trigger’ stage
- Downstream pipelines are **created dynamically using GitLab API**, authenticated with the job token
- Also possible to pass variables

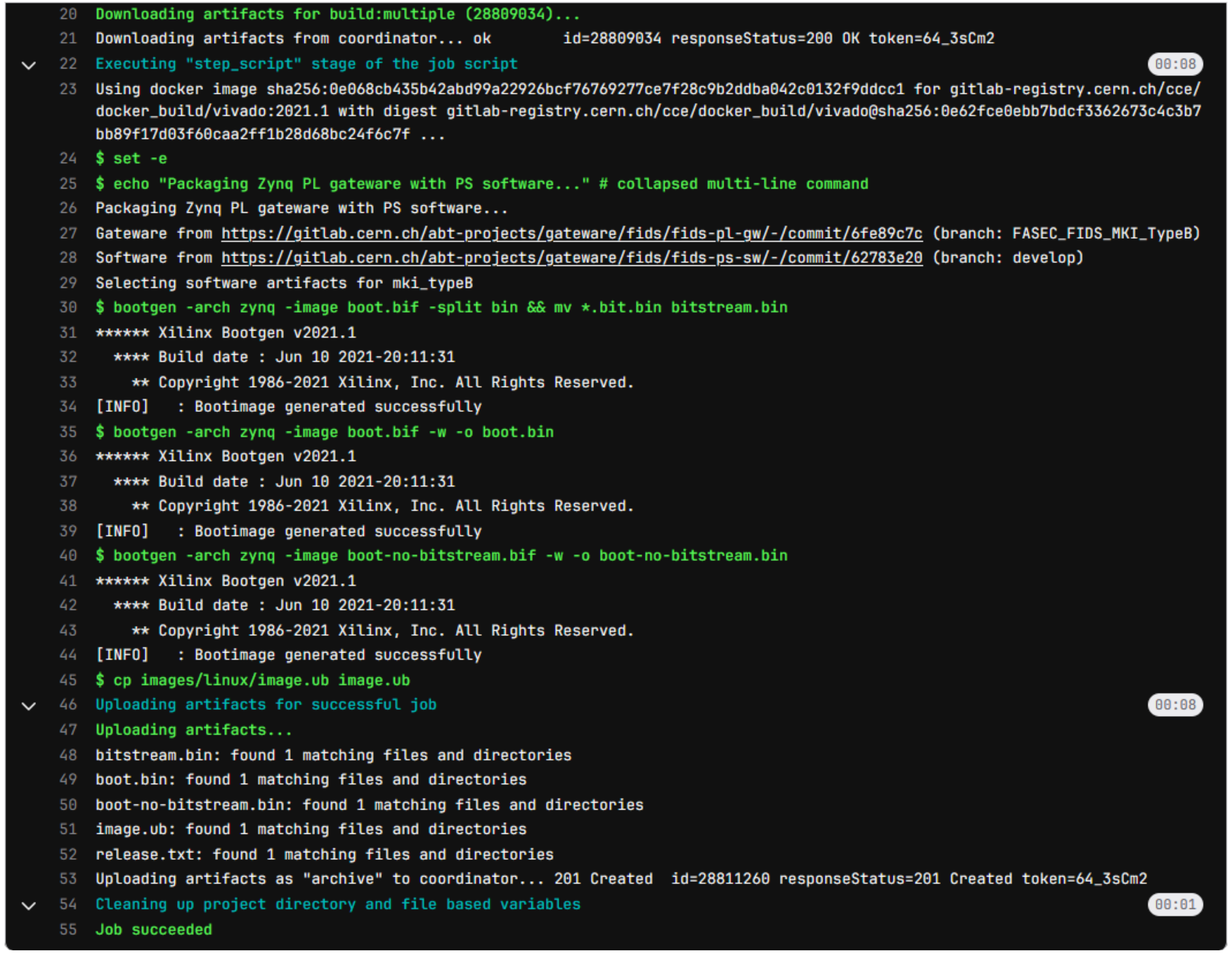


```
$ for branch in $BRANCHES; do # collapsed multi-line command
Created downstream pipeline https://gitlab.cern.ch/abt-projects/gateware/fids/fids-release/-/pipelines/5233946 for branch bidis-ls2
Created downstream pipeline https://gitlab.cern.ch/abt-projects/gateware/fids/fids-release/-/pipelines/5233947 for branch ct_bfa9p
Created downstream pipeline https://gitlab.cern.ch/abt-projects/gateware/fids/fids-release/-/pipelines/5233948 for branch ct_dfa
Created downstream pipeline https://gitlab.cern.ch/abt-projects/gateware/fids/fids-release/-/pipelines/5233949 for branch kfa45
Created downstream pipeline https://gitlab.cern.ch/abt-projects/gateware/fids/fids-release/-/pipelines/5233950 for branch mki_typeA
Created downstream pipeline https://gitlab.cern.ch/abt-projects/gateware/fids/fids-release/-/pipelines/5233951 for branch mki_typeB
Created downstream pipeline https://gitlab.cern.ch/abt-projects/gateware/fids/fids-release/-/pipelines/5233952 for branch mkp_pfn
Created downstream pipeline https://gitlab.cern.ch/abt-projects/gateware/fids/fids-release/-/pipelines/5233953 for branch sbds_mkdh
Created downstream pipeline https://gitlab.cern.ch/abt-projects/gateware/fids/fids-release/-/pipelines/5233954 for branch sbds_mkdv
Created downstream pipeline https://gitlab.cern.ch/abt-projects/gateware/fids/fids-release/-/pipelines/5233955 for branch sbds_ptm
Cleaning up project directory and file based variables
Job succeeded
```

Use case – fids-release

▸ package & release jobs

- Main branch containing global CI config
- One branch per installation with installation-specific mappings (gw & sw branches → devices)
- Combines gateware and software artifacts, creates boot images, uploads to GitLab package registry
  - Selects one software artifact from upstream pipeline (all included in the same archive because of the loop)
- Generates release text from identification files of each artifact
- Creates a release using the **job ID as tag**
- Passes this tag to next job in .env

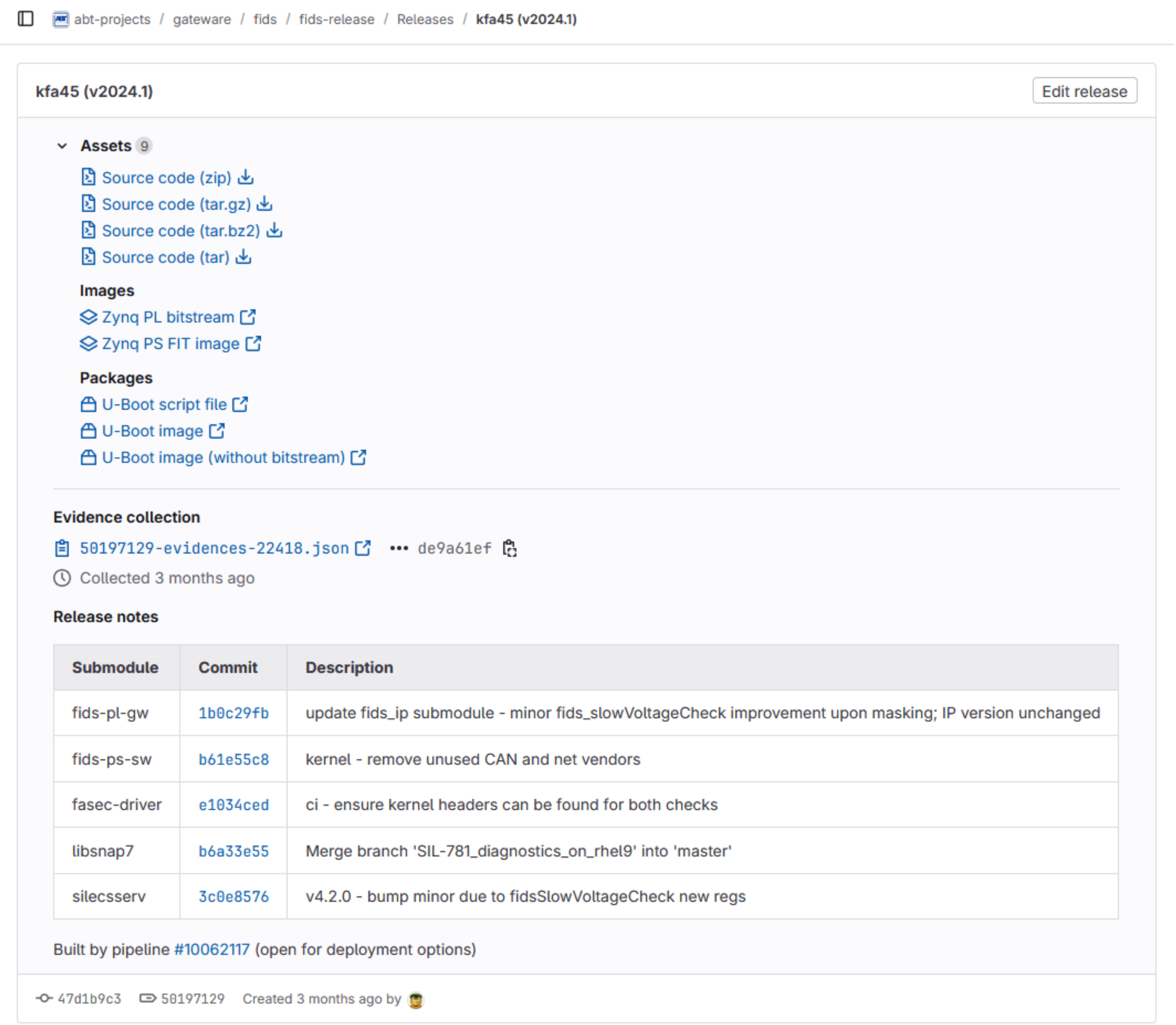


```
Downloading artifacts for build:multiple (28809034)...
Downloading artifacts from coordinator... ok        id=28809034 responseStatus=200 OK token=64_3sCm2
Executing "step_script" stage of the job script                                                   00:08
Using docker image sha256:0e068cb435b42abd99a22926bcf76769277ce7f28c9b2ddba042c0132f9ddcc1 for gitlab-registry.cern.ch/cce/
docker_build/vivado:2021.1 with digest gitlab-registry.cern.ch/cce/docker_build/vivado@sha256:0e62fce0ebb7bdcf3362673c4c3b7
bb89f17d03f60caa2ff1b28d68bc24f6c7f ...
$ set -e
$ echo "Packaging Zynq PL gateware with PS software..." # collapsed multi-line command
Packaging Zynq PL gateware with PS software...
Gateware from https://gitlab.cern.ch/abt-projects/gateware/fids/fids-pl-gw/-/commit/6fe89c7c (branch: FASEC_FIDS_MKI_TypeB)
Software from https://gitlab.cern.ch/abt-projects/gateware/fids/fids-ps-sw/-/commit/62783e20 (branch: develop)
Selecting software artifacts for mki_typeB
$ bootgen -arch zynq -image boot.bif -split bin && mv *.bit.bin bitstream.bin
****** Xilinx Bootgen v2021.1
  **** Build date : Jun 10 2021-20:11:31
    ** Copyright 1986-2021 Xilinx, Inc. All Rights Reserved.
[INFO]   : Bootimage generated successfully
$ bootgen -arch zynq -image boot.bif -w -o boot.bin
****** Xilinx Bootgen v2021.1
  **** Build date : Jun 10 2021-20:11:31
    ** Copyright 1986-2021 Xilinx, Inc. All Rights Reserved.
[INFO]   : Bootimage generated successfully
$ bootgen -arch zynq -image boot-no-bitstream.bif -w -o boot-no-bitstream.bin
****** Xilinx Bootgen v2021.1
  **** Build date : Jun 10 2021-20:11:31
    ** Copyright 1986-2021 Xilinx, Inc. All Rights Reserved.
[INFO]   : Bootimage generated successfully
$ cp images/linux/image.ub image.ub
Uploading artifacts for successful job                                                            00:08
Uploading artifacts...
bitstream.bin: found 1 matching files and directories
boot.bin: found 1 matching files and directories
boot-no-bitstream.bin: found 1 matching files and directories
image.ub: found 1 matching files and directories
release.txt: found 1 matching files and directories
Uploading artifacts as "archive" to coordinator... 201 Created  id=28811260 responseStatus=201 Created token=64_3sCm2
Cleaning up project directory and file based variables                                            00:01
Job succeeded
```

Use case – fids-release

▶ GitLab release

- All releases attached to the ‘fids-release’ repository, which doesn't contain any code (just config files)
- Direct links to artifacts stored in GitLab package registry
- Table summarizing gw & sw git hashes, incl. dependencies, with links to commits in their respective repository
- Link to pipeline containing deployment jobs


abt-projects / gateware / fids / fids-release / Releases / kfa45 (v2024.1)

kfa45 (v2024.1)

Assets 9

- Source code (zip)
- Source code (tar.gz)
- Source code (tar.bz2)
- Source code (tar)

Images

- Zynq PL bitstream
- Zynq PS FIT image

Packages

- U-Boot script file
- U-Boot image
- U-Boot image (without bitstream)

Evidence collection

`50197129-evidences-22418.json` ••• `de9a61ef`

Collected 3 months ago

Release notes

Submodule	Commit	Description
fids-pl-gw	`1b0c29fb`	update fids_ip submodule - minor fids_slowVoltageCheck improvement upon masking; IP version unchanged
fids-ps-sw	`b61e55c8`	kernel - remove unused CAN and net vendors
fasec-driver	`e1034ced`	ci - ensure kernel headers can be found for both checks
libsnap7	`b6a33e55`	Merge branch 'SIL-781_diagnostics_on_rhel9' into 'master'
silecsserv	`3c0e8576`	v4.2.0 - bump minor due to fidsSlowVoltageCheck new regs

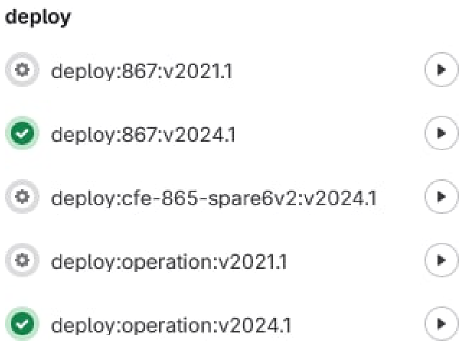
Built by pipeline #10062117 (open for deployment options)

`47d1b9c3` `50197129` Created 3 months ago by

Use case – fids-release

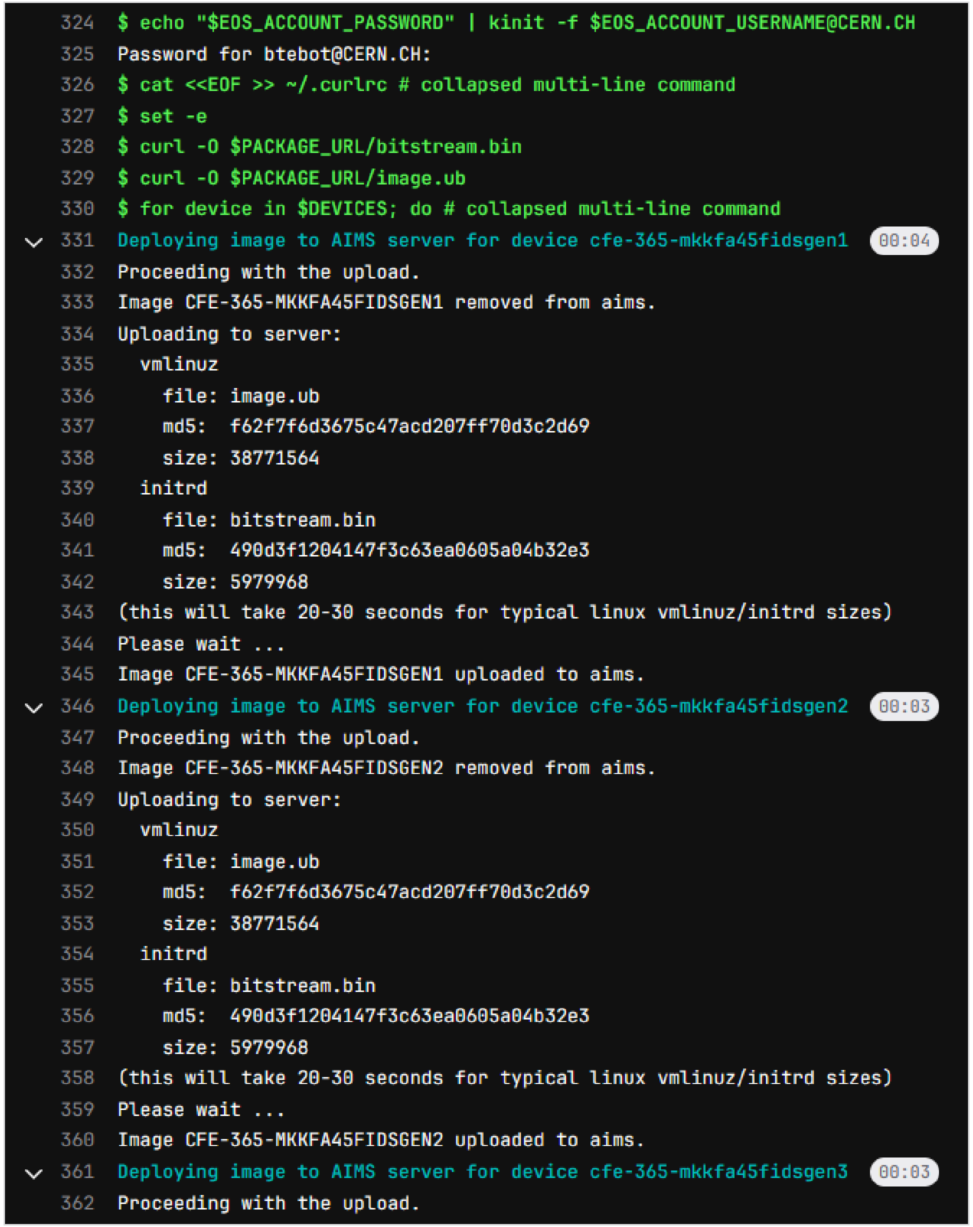
▸ deploy job

- 1-click deployment to operational and test devices, from pre-configured mappings: **no risk to deploy a development version in operation**
- Downloads artifacts from GitLab package registry, using release tag passed from previous job in .env
- Uploads the bitstream and boot image to CERN's TFTP server, for each configured device
  - Example here: same image for devices cfe-365-mkkfa45fidsgen1 to cfe-365-mkkfa45fidsgen4



deploy

- ⚙ deploy:867:v2021.1 ▶
- ✓ deploy:867:v2024.1 ▶
- ⚙ deploy:cfe-865-spare6v2:v2024.1 ▶
- ⚙ deploy:operation:v2021.1 ▶
- ✓ deploy:operation:v2024.1 ▶



```
$ echo "$EOS_ACCOUNT_PASSWORD" | kinit -f $EOS_ACCOUNT_USERNAME@CERN.CH
Password for btebot@CERN.CH:
$ cat <<EOF >> ~/.curlrc # collapsed multi-line command
$ set -e
$ curl -O $PACKAGE_URL/bitstream.bin
$ curl -O $PACKAGE_URL/image.ub
$ for device in $DEVICES; do # collapsed multi-line command
Deploying image to AIMS server for device cfe-365-mkkfa45fidsgen1 00:04
Proceeding with the upload.
Image CFE-365-MKKFA45FIDSGEN1 removed from aims.
Uploading to server:
  vmlinuz
    file: image.ub
    md5:  f62f7f6d3675c47acd207ff70d3c2d69
    size: 38771564
  initrd
    file: bitstream.bin
    md5:  490d3f1204147f3c63ea0605a04b32e3
    size: 5979968
(this will take 20-30 seconds for typical linux vmlinuz/initrd sizes)
Please wait ...
Image CFE-365-MKKFA45FIDSGEN1 uploaded to aims.
Deploying image to AIMS server for device cfe-365-mkkfa45fidsgen2 00:03
Proceeding with the upload.
Image CFE-365-MKKFA45FIDSGEN2 removed from aims.
Uploading to server:
  vmlinuz
    file: image.ub
    md5:  f62f7f6d3675c47acd207ff70d3c2d69
    size: 38771564
  initrd
    file: bitstream.bin
    md5:  490d3f1204147f3c63ea0605a04b32e3
    size: 5979968
(this will take 20-30 seconds for typical linux vmlinuz/initrd sizes)
Please wait ...
Image CFE-365-MKKFA45FIDSGEN2 uploaded to aims.
Deploying image to AIMS server for device cfe-365-mkkfa45fidsgen3 00:03
Proceeding with the upload.
```

Use case – fids-release

▸ identification

From AIMS boot server:
deployed image with release tag

```
[lstrobin@cwe-513-vml321 ~]$ aims2client showimage cfe-ua*-*fids*

Image Name           Arch   UEFI  Description
-----------------------------------------------------------------
CFE-UA23-MKIFIDSA1   armv7  N     FIDS-RELEASE-MKI_TYPEA-44971639
CFE-UA23-MKIFIDSA2   armv7  N     FIDS-RELEASE-MKI_TYPEA-44971639
CFE-UA23-MKIFIDSB1   armv7  N     FIDS-RELEASE-MKI_TYPEB-44971650
CFE-UA23-MKIFIDSB2   armv7  N     FIDS-RELEASE-MKI_TYPEB-44971650
CFE-UA87-MKIFIDSA1   armv7  N     FIDS-RELEASE-MKI_TYPEA-44971639
CFE-UA87-MKIFIDSA2   armv7  N     FIDS-RELEASE-MKI_TYPEA-44971639
CFE-UA87-MKIFIDSB1   armv7  N     FIDS-RELEASE-MKI_TYPEB-44971650
CFE-UA87-MKIFIDSB2   armv7  N     FIDS-RELEASE-MKI_TYPEB-44971650
```

From FASEC devices (SSH):
gw & sw git hashes (incl. submodules)

```
[lstrobin@cwe-513-vml321 ~]$ ssh cfe-867-mkfa45fids

================================================================================
cfe-867-mkfa45fids - 172.18.210.112 - FASEC FIDS v10.6.2 - kfa45_gpiod
================================================================================
FIDS gateware            commit:  1b0c29fb    date:  Fri Jan 31 13:02:25 UTC 2025
PetaLinux                commit:  b61e55c8    date:  Mon Oct 21 15:24:00 UTC 2024
FASEC driver             commit:  e1034ced    date:  Tue Jun 11 13:11:22 UTC 2024
SILECS server            commit:  3c0e8576    date:  Thu Dec 12 15:18:42 UTC 2024
Snap7 library            commit:  b6a33e55    date:  Mon May  6 14:52:47 UTC 2024
================================================================================
Xilinx Zynq(R)-7000 XC7Z030, Kintex(TM)-7 FPGA, 125K logic cells
2x ARM(R) Cortex(R)-A9 MPCore(TM) CPU @ 333 MHz, 1006 MB memory
PetaLinux 2024.1
Linux cfe-867-mkfa45fids.cern.ch 6.6.10-xilinx-v2024.1-g62eec70a4258 armv7l
Up since: Sat Mar 22 07:02:46 UTC 2025

cfe-867-mkfa45fids:~$ sudo peek 0x43c10008

0x1b0c29fb

cfe-867-mkfa45fids:~$ ls -l /etc/revision/

-rw-r--r--    1 root     root            78 Oct 21  2024 fasec-driver
-rw-r--r--    1 root     root            59 Oct 21  2024 libsnap7
-rw-r--r--    1 root     root            53 Oct 21  2024 petalinux
-rw-r--r--    1 root     root            64 Oct 21  2024 silecsserv
```

Future

- **Migration of both projects to DI/OT, little impact on project structure**
  - Distributed Kicker Fast Control (DKFC), using CERN common DI/OT hardware and ABT specific hardware
  - ZynqMP System Board allows for a user script (block design compatible?) – docs
    - *Branch-per-installation* approach still required...
- **Possible improvements to the *branch-per-installation* approach**
  - To be investigated but with considerable design effort, all gateware variations between projects/installations could be embedded in the (bigger) ZynqMP PL/FPGA.
    - Software could configure MUXs and enable/disable modules to cater for installation differences.
    - Software in any case already today requires a hand-crafted (FESA3) Deploy Unit i.e. the instances.
  - Better separation between generic and specific parts so git rebase can be used? Technically not feasible since git uses snapshots and not differences. New version control systems could help:
    - Jujutsu VCS (`jj`) – 'experimental' but stable and well-advanced, most promising
      - can be hosted on git forges
      - simultaneous multi-branch rebasing would allow for more ~~laziness~~ efficiency
      - submodule support lacking but can still be done using git native commands

Thank you for your attention!

Questions?

CERN

ABT

ACCELERATOR BEAM TRANSFER

home.cern