Modular Optical Laser Control Systems

An Approach Based on EPICS and System-on-Module Controllers

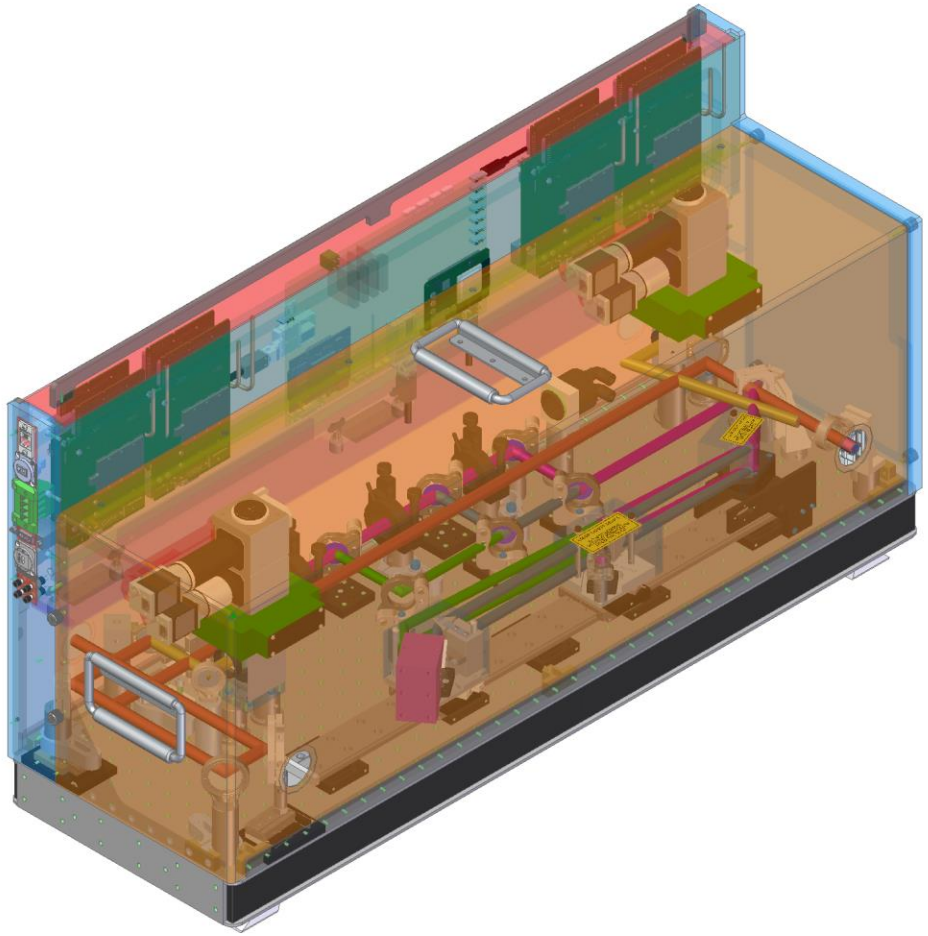
Tyler Johnson / LCLS Experiment Control Systems
Jeremy Lorelli / SLAC Technology Innovation Directorate
Leonid Sapozhnikov / SLAC Technology Innovation Directorate

EPICS Collaboration Meeting, ENS Paris-Saclay
April 22, 2026

SLAC NATIONAL ACCELERATOR LABORATORY
Stanford University
U.S. DEPARTMENT of ENERGY

Outline

1. Introduction and Background
2. Embedded Hardware
3. Software and Build System
4. Future Work
5. Questions



Organization - Team

Tyler Johnson
Lead ECS Engr

Brice Arnold
System Lead/
Lead Mech Engr

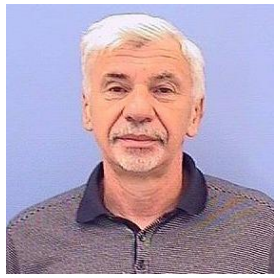
Donny Magna
Laser Systems Lead

Barry Fishler
ECS Lead

Joe Robinson
Laser SRD Lead

Mike Glownia
Laser SRD

Ryan Herbst
TID Lead



Leonid Sapozhnikov
TID

Larry Ruckman
TID

Jeremy Lorelli
TID

Amy Liang
Mech Engr

Haufai Auyeung
Mech Dsgn

The TILES V3 project is a large multidisciplinary effort; today we will only focus on the embedded computing portion!

Organization - Team

Tyler Johnson
Lead ECS Engr

Brice Arnold
System Lead/
Lead Mech Engr

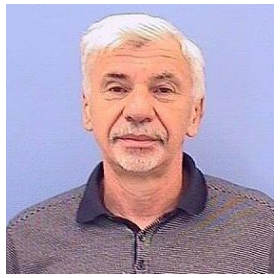
Donny Magna
Laser Systems Lead

Barry Fishler
ECS Lead

Joe Robinson
Laser SRD Lead

Mike Glownia
Laser SRD

Ryan Herbst
TID Lead



Leonid Sapozhnikov
TID

Larry Ruckman
TID

Jeremy Lorelli
TID

Amy Liang
Mech Engr

Haufai Auyeung
Mech Dsgn

The TILES V3 project is a large multidisciplinary effort; today we will only focus on the embedded computing portion!

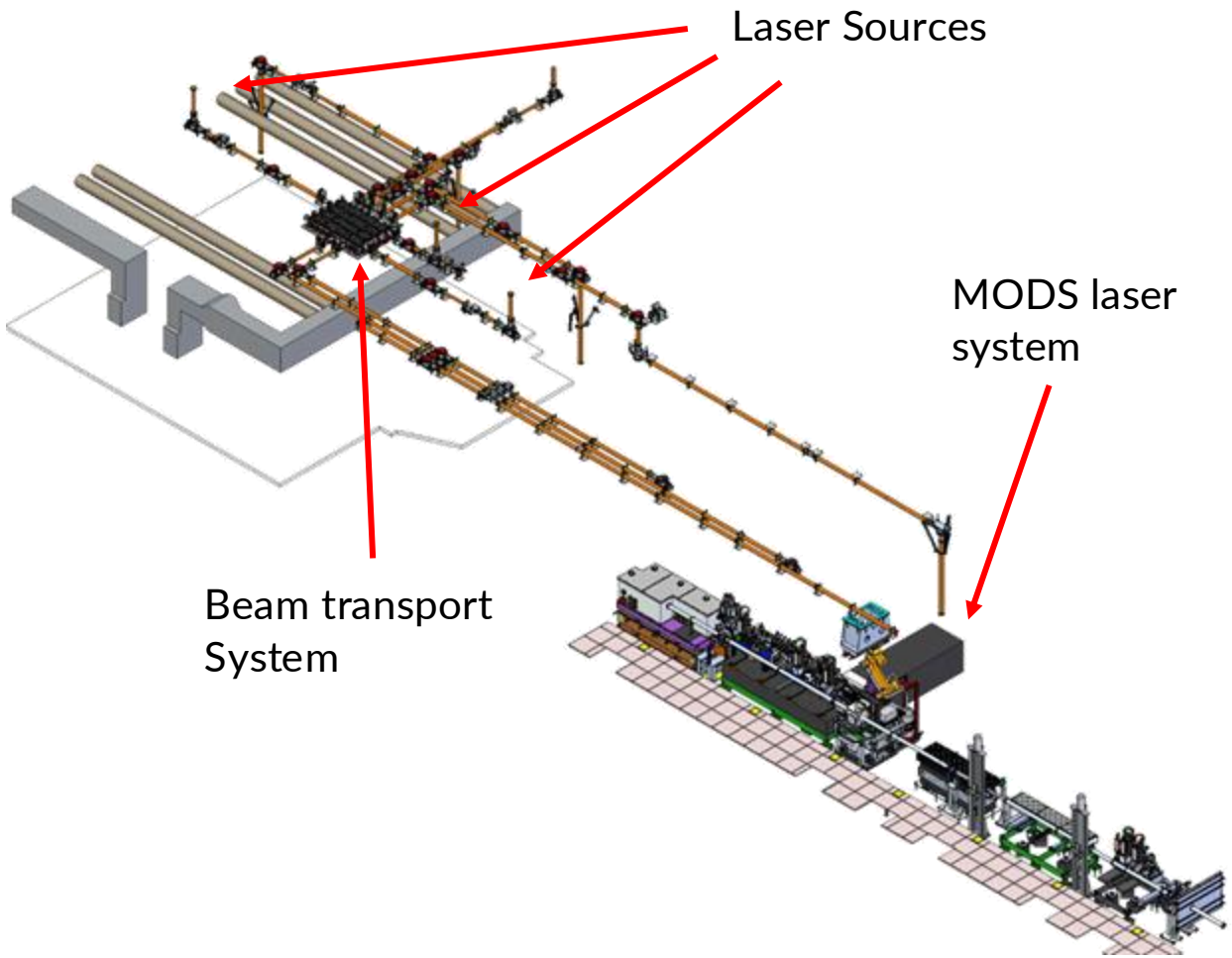
1

Introduction and Background

LCLS Laser Systems Integration

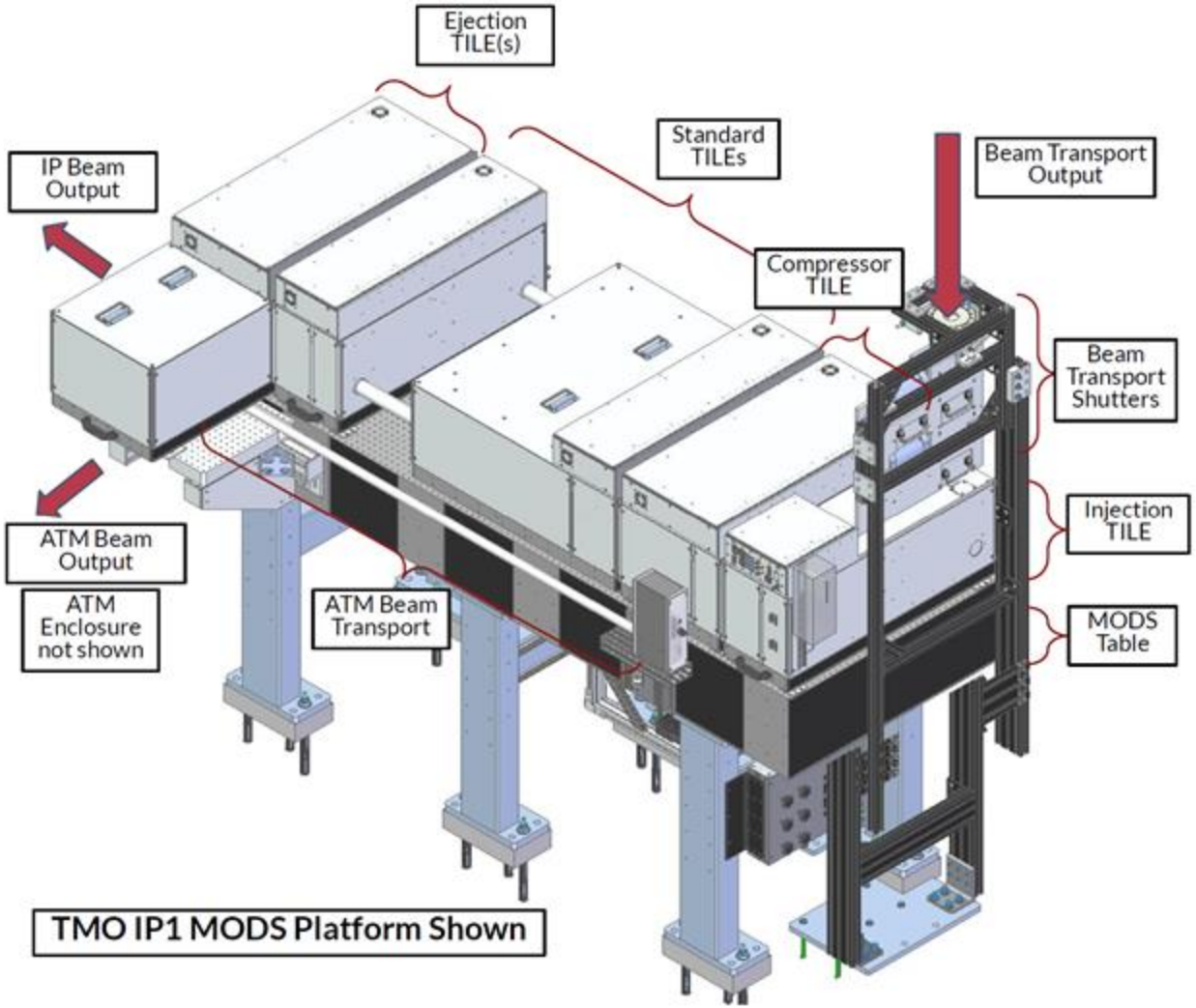
LCLS Laser Delivery

- LCLS-II hutches use shared laser systems, distributed from NEH laser hall
- Laser systems are passed into a large, evacuated beam transport system
- Motorized optical switch yard allows routing of any supported source to any supported destination
- Mixture of 800 nm and 1030 nm systems provide stable, reliable "base" input beams
- The delivered base beam is further modified in the hutch
- The specific configuration in the hutch is tailored to meet the requirements from the instrument and the experiment at hand



In the Hutch

- In the hutch, a modular laser table system ("MODS") accepts the input beam and tailors it to through a series of modules with specific functionality
  - Beam conditioning (energy, wavefront, spot size)
  - Compression
  - Harmonics generation
  - OPA
  - etc.
- Each module has an array of control points and diagnostics for hands-free remote operation
- The constituent components of the MODS are called “TILEs”



What's in a TILE?

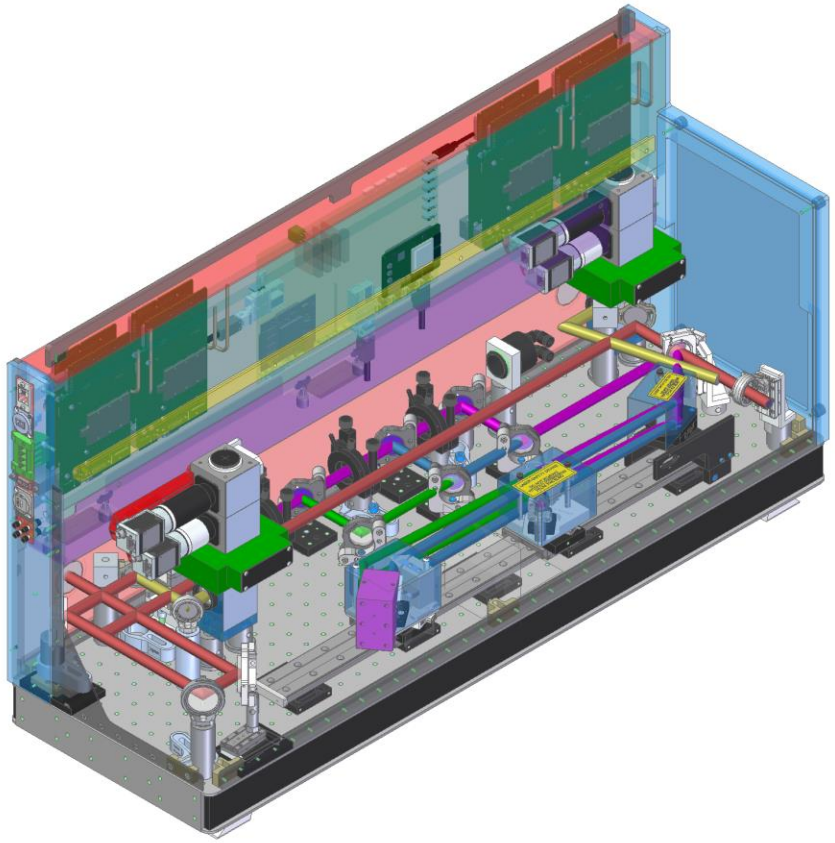
A lot!

(Current) Maximum device/interface counts

- Motorized axes: 26
- Cameras: 6
- Spectrometers: 1
- Analog I/O: 4
- Digital I/O: 2
- Environment sensor (P / T / %RH): 1
- RS-232: 1

Standard TILE footprint: 1.5' x 3'

Minimum TILE footprint: 1' x 3'



Harmonics TILE

Goals for the TILES system

High reliability

- Hardware *and* software
- Reduce downtime and troubleshooting efforts

Minimize time to exchange TILEs

- Reduce time from failure to operation
- Make the delivery system more re-configurable and agile

Small system footprint

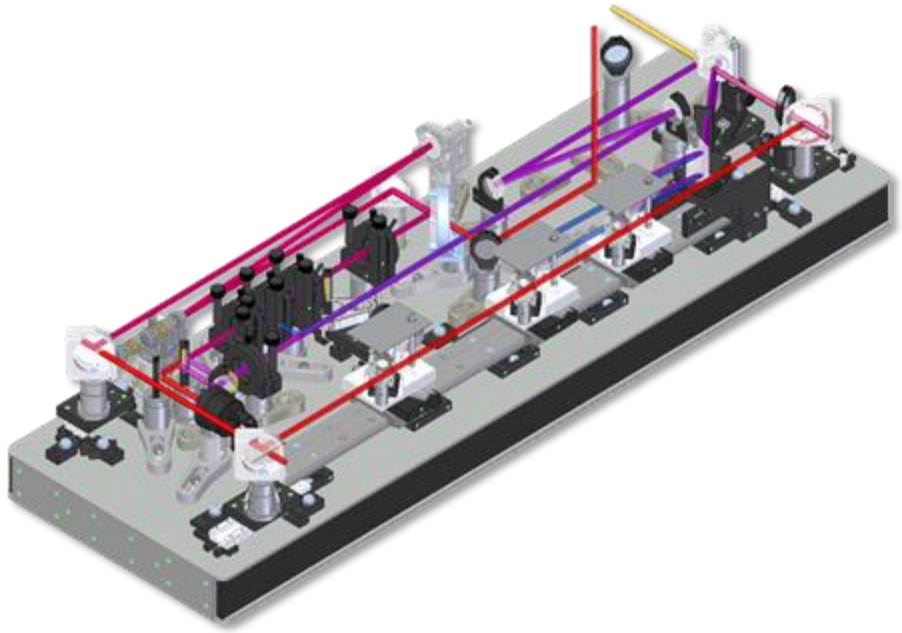
- Reduce space claim in already crowded hutches
- Provide more room for optics

Integration path flexibility

- Provide access to wide variety of interfaces, devices, etc
- Allow for rapid adaptation to science needs

Standalone Operation

- Benchtop operation for system build and debug
- Reduce reliance on facility systems (NFS)



Objective

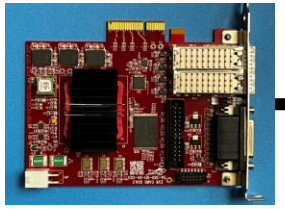
Minimal external interfaces, maximum utility

Cooling Water

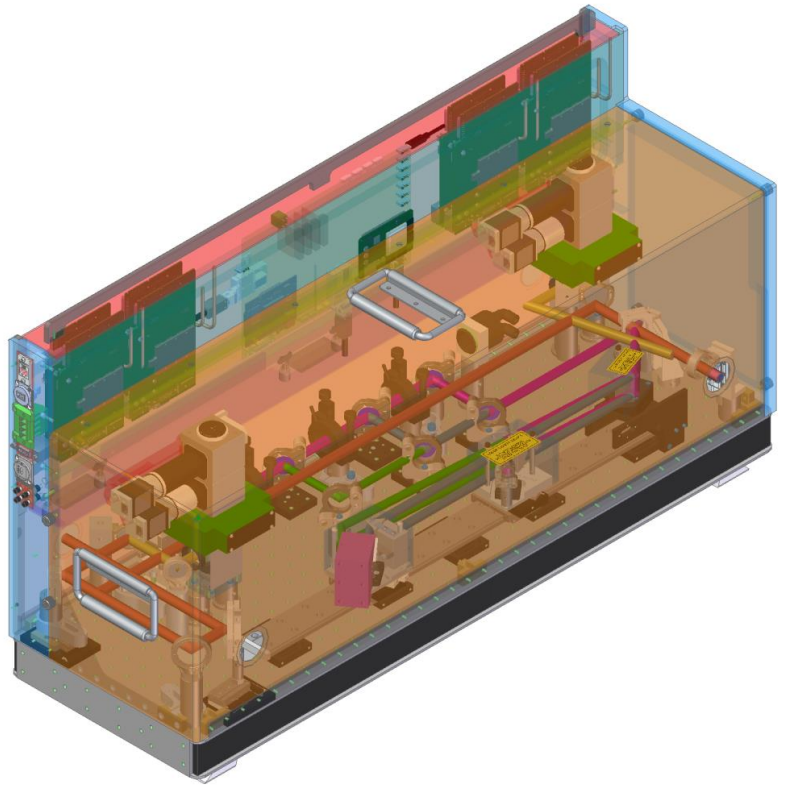
Network

Power

Triggers
(optional)



Fast ADC
(optional)



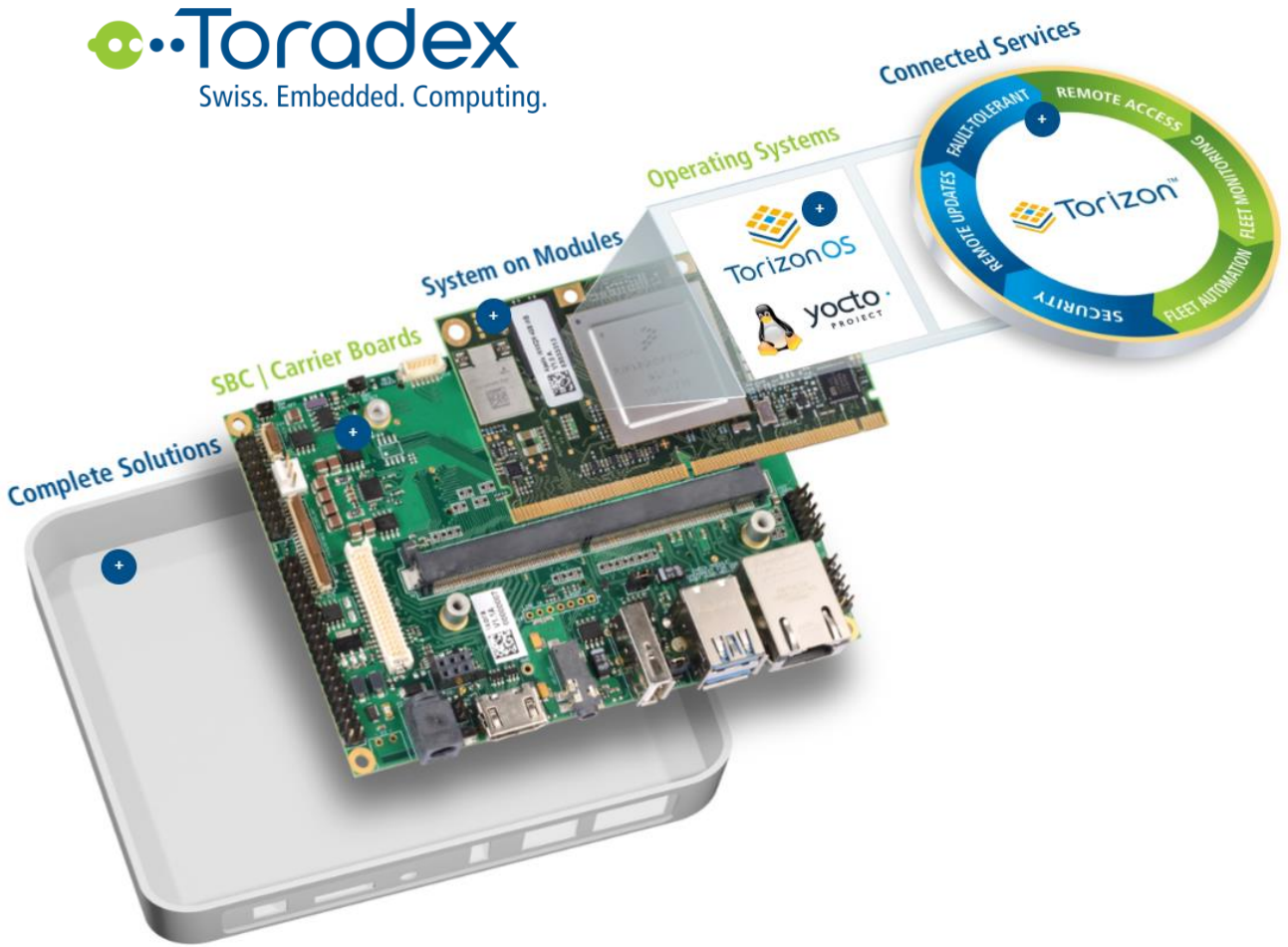
2

Embedded Hardware

Embedded hardware

Toradex

- Founded in 2003, headquartered in Switzerland
- Sell a wide range of embedded hardware at different performance levels
- Provide their own operating system as well as support for custom OSes via the Yocto project
- Provide connectivity services for e.g. OTA updates, fleet monitoring
- Good system documentation and software support
- Design files for their commercially available carriers are made freely available
- Have internal experience using this vendor in a very similar application



Embedded hardware

Toradex Aquila AM69 System-on-Module (SOM)

- Based on TI AM 69
  - 8x cores @ 2.0 GHz
  - Up to 32 GB RAM
  - Up to 256 GB flash storage
  - 2x Integrated Arm M5 MCUs
- 6x Ethernet interfaces
  - 2x 10 Gb
- Up to 3 lanes of PCIe Gen3
- 8x integrated ADC channels, 4 Msps max
- 2x USB 3.2
- Lots of peripheral interfaces (I2C, SPI, UART, QSPI, I3C, ...)
- Most powerful product from Toradex to date
- Minimum product commitment through 2034

FRONT

BACK

CPU Details	
CPU Name	TI AM 69
CPU Type	Up to 8x Arm Cortex™-A72
Microcontroller	2x Arm Cortex™-R5F
CPU Clock	2.0 GHz (A72) 1.0 GHz (R5F)

Memory	
RAM	Up to 32GB LPDDR4 (128 Bit)
Flash	Up to 256GB eMMC

This data is preliminary and is subject to change.

Connectivity	
USB 3.2	1x DRD (Gen 1) 1x Host (Gen 1)
Ethernet	1x Gigabit (PHY with TSN) 4x 2.5 Gigabit (SGMII with TSN) 1x 10 Gigabit (USXGMII with TSN)
Wi-Fi	2.4/5/6 GHz Tri-band 2x2 Wi-Fi 7 (802.11be)
Bluetooth	Bluetooth Classic / BLE 5.3
PCIe	2x (x2 Gen 3) 1x (x1 Gen 3)
I2C	8x
SPI	7x
QSPI	1x
UART	11x
PWM	12x
Analog Input	8x
SDIO/SD/MMC	1x
CAN FD	19x
GPIO	110x
JTAG	1x

Multimedia	
Neural Processing Unit (NPU)	32 TOPS
Image Signal Processor (ISP)	2x
Depth And Motion Processing Accelerator (DMPAC)	Yes
Display Controller	Triple
2D Acceleration	Yes
3D Acceleration	Yes
Video Decoder	H.264/AVC and H.265/HEVC
Video Encoder	H.264/AVC and H.265/HEVC
Display Serial Interface	2x Quad Lane MIPI DSI
Display Port	1x (up to 4k)
Digital Audio	5x McASP: I2S or TDM
Camera Serial Interface	3x Quad Lane MIPI CSI-2

Operating System	
Torizon™	Details
yocto	Details
Toradex Easy Installer	Details
FreeRTOS	Coming Soon
QNX	Contact Us
Android	Contact Us

Physical	
Size	85 x 55 x 9 mm
Temperature	-40° to +85° C
Shock / Vibration	EN 60068-2-6/50g 20ms
Power Dissipation	Approx. 5W - 25W
Product Availability	2034+

Embedded Hardware

Carrier Boards

- SOM is a complete computer
- A carrier board is necessary to break out the available I/O to connectors
- Creates a more flexible hardware architecture where interfaces can be added/removed, form factor changed by exchanging the carrier while keeping the CPU the same
- Custom carrier boards can be designed to fit needs

FRONT

BACK

Connectivity	
Supported Modules	Entire Aquila Family
USB 3.0	1x Dual-Role on USB-C 2x Host on USB-A 1x Downstream Facing Port (DFP) on USB-C
USB 2.0 Debug	1x USB-C
Ethernet	2x Gigabit
PCIe	1x M.2 Key M 1x M.2 Key B (+ Micro-SIM card connector)
I2C	4x
SPI	2x
QSPI	1x
OSPI	1x
UART	4x
PWM	3x
GPIO	Up to 90
Analog Input	4x
RS232	1x
RS485	1x
SDIO/SD/MMC	1x 4 Bit (Full Size)
CAN	4x
Tamper	2x
JTAG	1x

Multimedia	
Display Serial Interface	1x Quad Lane MIPI DSI
LVDS	1x (via optional Verdin DSI to LVDS Adapter)
HDMI	1x (via included Verdin DSI to HDMI Adapter)
Digital Audio	2x I2S
Display Port	1x
Camera Serial Interface	2x Quad Lane MIPI CSI
Digital Mic In	2x PDM Mics
Analog Audio Line in	1x (Stereo)
Analog Audio Mic in	1x (Stereo)
Analog Audio Headphone out	1x (Stereo)
Analog Audio Line out	1x (Stereo)

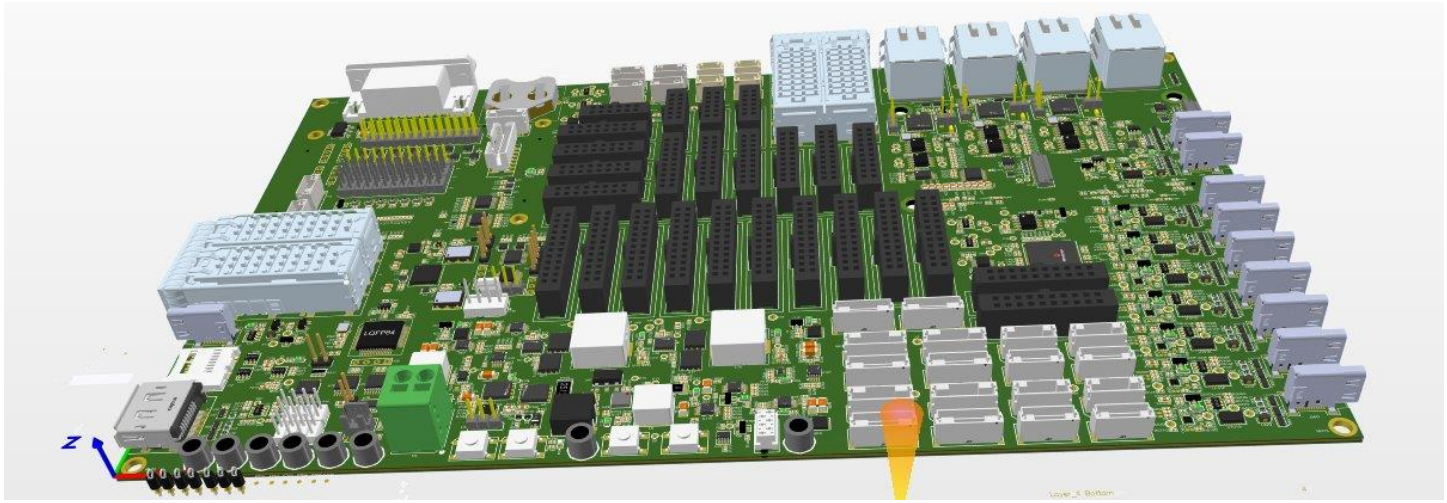
Physical	
Size	250 x 200mm
Temperature	0° to 70° C
Power Supply - Primary	100W USB-C PD
Power Supply - Secondary	9-24V (+/-10%)

Aquila Development Board

Aquila Clover SBC Carrier (120mmx120mm)

Embedded Hardware

SLAC-designed carrier board



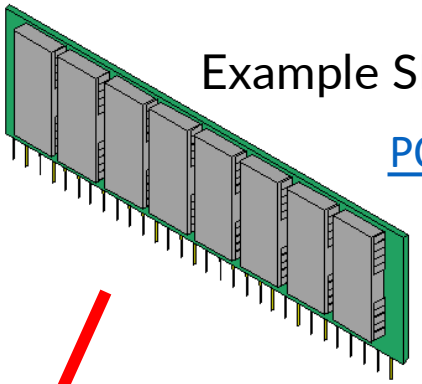
- First revision of the board is complete
- Final board dimensions: 6.5" x10.5"
- SOM is connected to the bottom of the board
- Fabricated 4x first revision boards for testing

Embedded Hardware

Flexible Design

SIP Modules

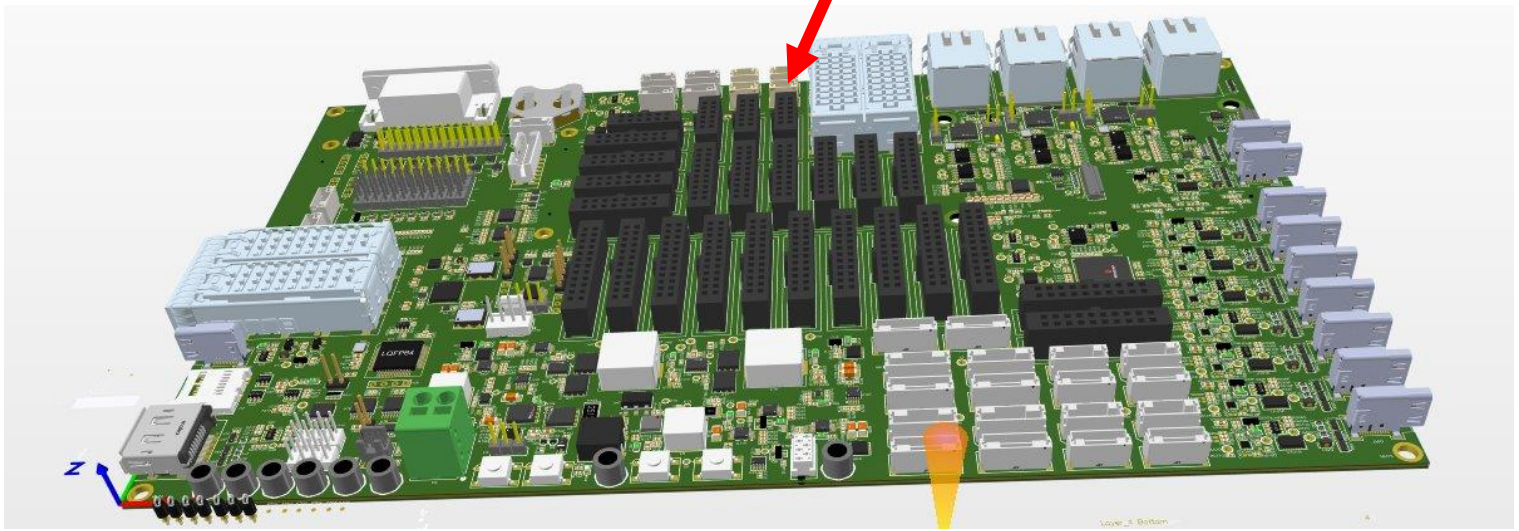
- "System-in-package": pluggable modules that add/modify functionality
- Example: level shifting GPIO outputs



Example SIP module

PCMag

SIP modules plug into keyed headers on main board

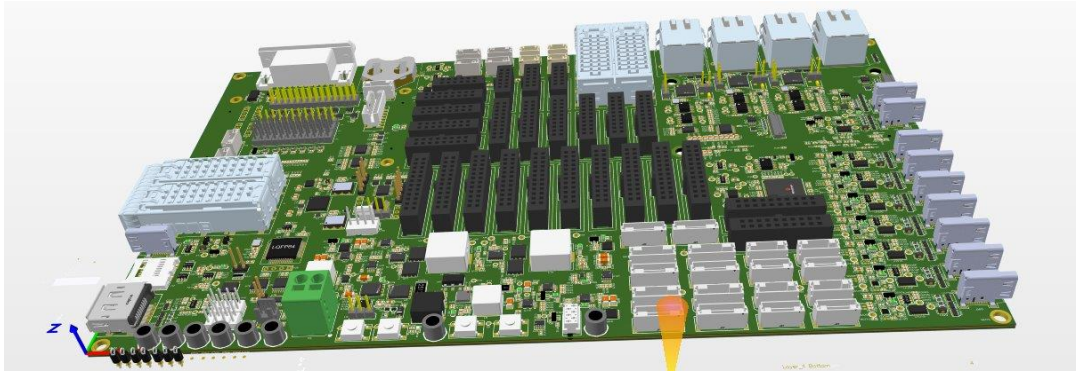


Used for:

- ADCs
- GPIO
- UARTs
- I2C
- SPI
- Trigger outputs

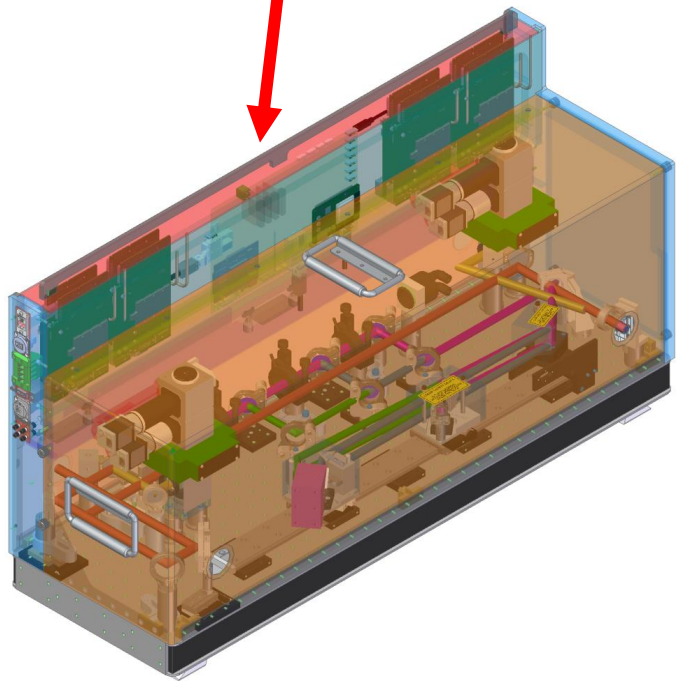
Embedded Hardware

Custom Carrier Board – By the Numbers



Interfaces and Subsystems

- Power conditioning and management
- 1x EEPROM
- 1x SD Card
- 12x hardware trigger outputs
- 40x GPIO
- 4x I2C
- 2x SPI
- 6x Ethernet (2x 10Gb, 4x 1Gb)
- 1x Display Port
- 9x USB-C (1x USB 3.2, 8x USB 3.1)
- 6x UART (RS-232, RS485, …)
- 8x ADC
- 1x JTAG
- 1x Serial debug interface
- 2x CSI
- PCIe (used to interface with FPGA SOM)
- 4x SFP+ slots (event timing, network)



3

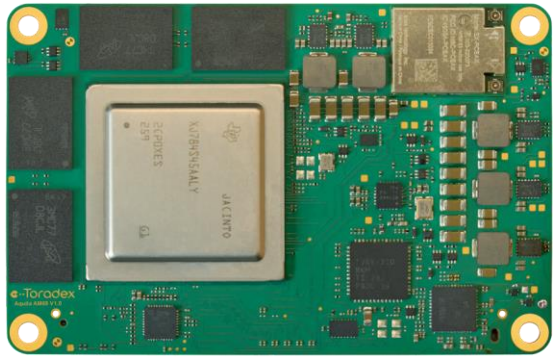
Software and Build System

Embedded EPICS

How do we get to *embedded* EPICS?

Yocto Project

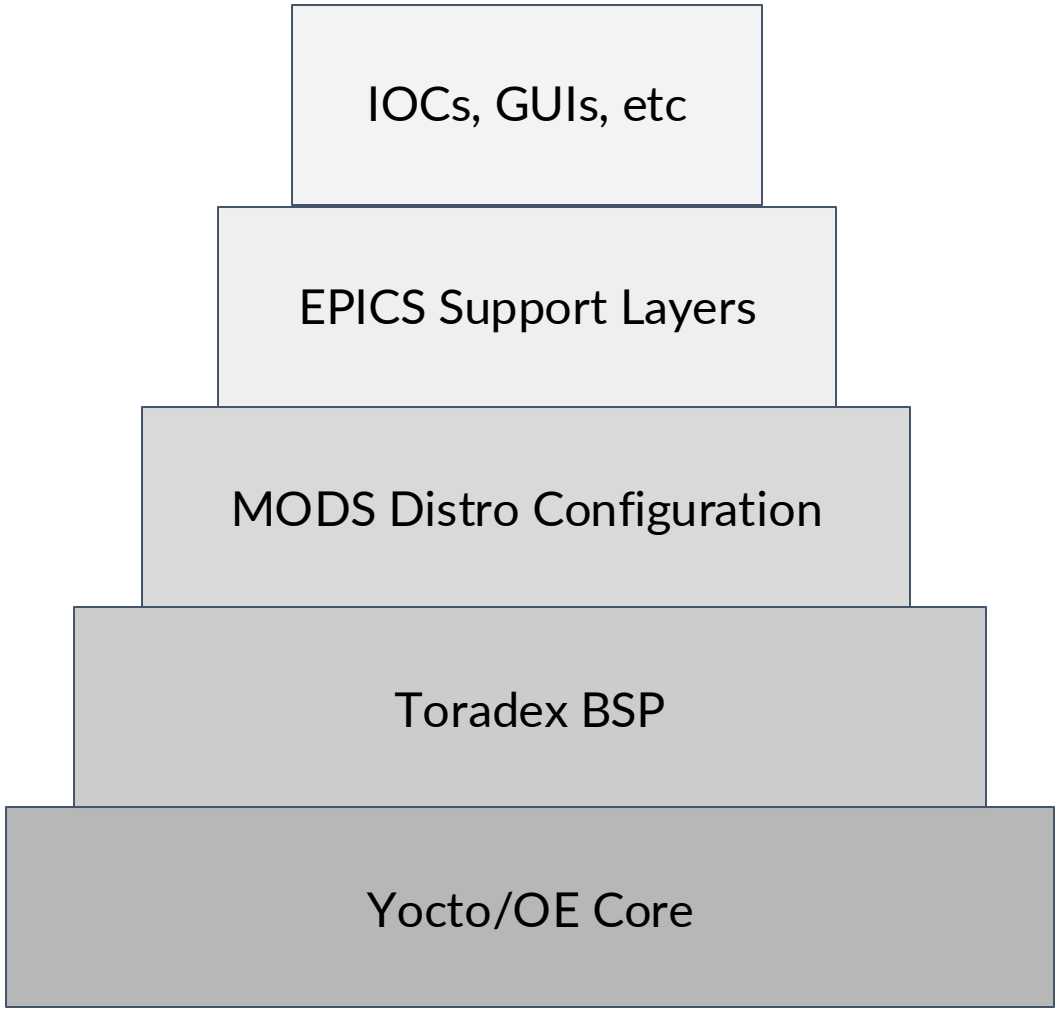
- Utilizes OpenEmbedded “bitbake” build tool and metadata system
  - Based on collections of metadata ("layers")
  - Basic unit of metadata is called a "recipe"
- Generates reproducible, reusable builds of embedded software, completely from source
  - Software re-use is a core feature
- Focused on versatility and providing tools for all use cases
- Good for complex projects, easily supports different project variations
- Has become the industry standard with support from TI, ARM, NXP, Microchip, ST, Renesas, …



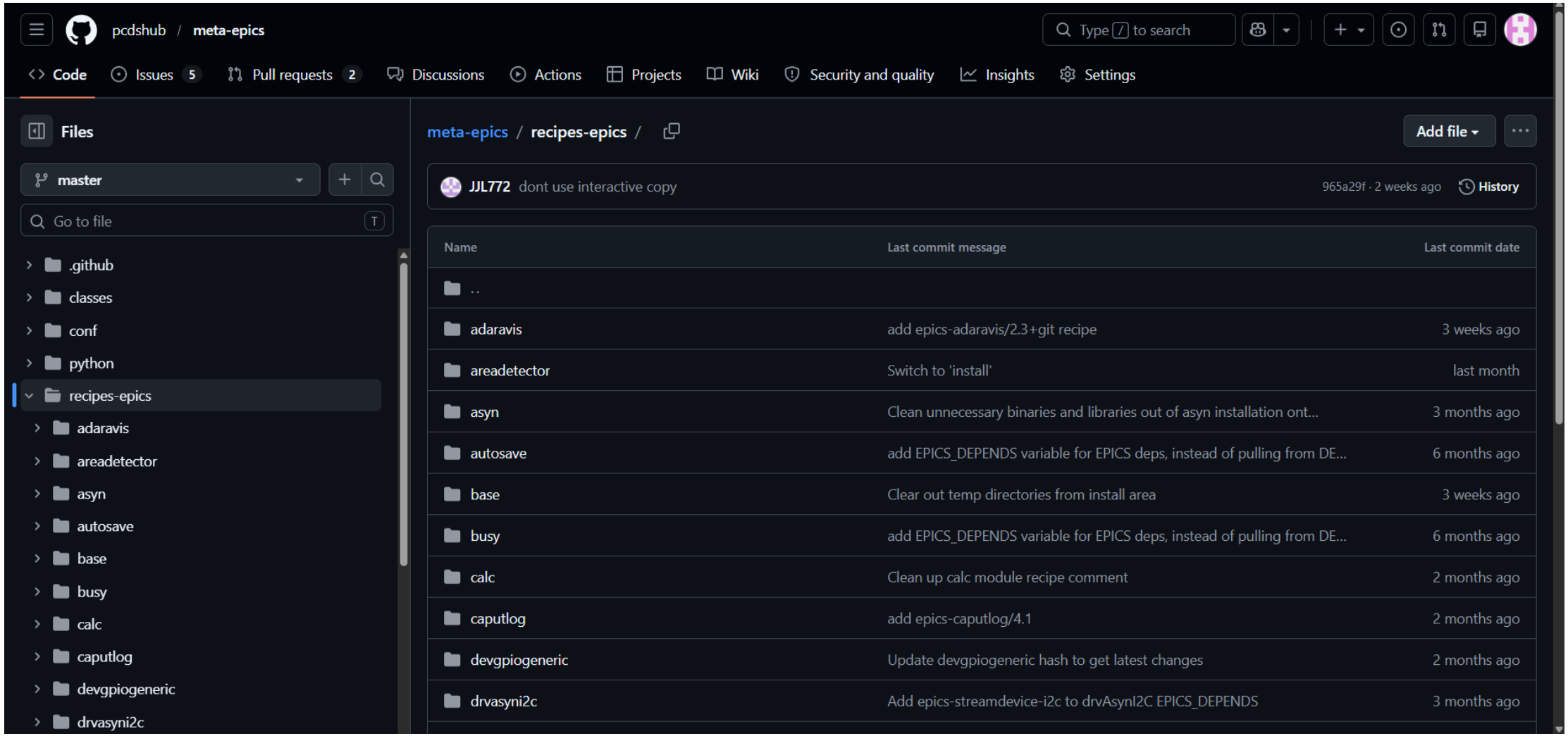
Yocto has a layered architecture

MODS System Layers

- Yocto/OE core
  - Core build system and Linux infrastructure
- Toradex BSP
  - Hardware support, some drivers
- MODS distro configuration
  - Choices about kernel, OS, and development environment, etc.
- EPICS support
  - Libraries and modules for traditional or custom integrations
- Application layers
  - IOCs, GUIs, automation, vendor device SDKs, etc.



meta-epics!



github.com/pcdshub/meta-epics

A simple recipe

EPICS calc module

- Put significant effort into the build system, making EPICS easy(-er) in Yocto
- Most EPICS components are simple to add recipes for
- Inherit a lot of functionality from the epics-module support class
- Need:
  - Basic description of the package
  - Licensing information
  - Source location and revision
  - Dependencies
  - Special configuration information

```
inherit epics-module

SUMMARY = "Calc recipe"
DESCRIPTION = "Recipe for building Calc for the EPICS control system."

LICENSE = "synApps"
LIC_FILES_CHKSUM = "file://LICENSE;md5=a2c259c010f2152379d7769be894bf4a"
LICENSE_PATH += "${S}"
NO_GENERIC_LICENSE[synApps] = "LICENSE"

SRCREV = "2f5b175f260bc3fe35bc25a3f6c204e9d6f628c9"
SRC_URI = "git://github.com/epics-modules/calc;protocol=https;branch=master;rev=${SRCREV}"

DEPENDS += "${EPICS_DEPENDS}"

# We decide not to build with SSCAN here; this can be changed with a bbappend
disable_sscan () {
    echo "SSCAN=" >> "${S}/configure/RELEASE.local"
}

do_configure[postfuncs] += "disable_sscan"

S = "${WORKDIR}/git"
```

A more complicated recipe...

areaDetector

- Builds areaDetector and all enabled submodules
- EPICS dependencies are managed by adding them to a variable, which is later parsed during the configuration step
- Any non-EPICS dependencies can be added to the build as well
- Additional scripts and procedures can be appended or prepended to any part of the build and installation process

```
# Recipe for areaDetector master branch

inherit epics-module

SUMMARY = "areaDetector recipe"
DESCRIPTION = "Recipe for building the areaDetector superpackage for the EPICS control system."

LICENSE = "EPICS"
LIC_FILES_CHKSUM = "file://LICENSE;md5=f58d4a2e30a77fa9529619ccce87dd8b"
LICENSE_PATH += "${S}"

SRCREV = "ed51ecf4f5d781f8edb525cd4219902c82b0be94"
SRC_URI = "gitsm://github.com/areaDetector/areaDetector;protocol=https;branch=master;rev=${SRCREV}"

EPICS_DEPENDS += "epics-autosave epics-calc epics-asyn epics-pvxs epics-sscan"
DEPENDS += "${EPICS_DEPENDS} python3-native zlib tiff ffmpeg"

configure_libs() {
    # Note on RELEASE_PRODS/RELEASE_LIBS:
    # - Must use absolute paths because $(TOP) is different between the two
    # - RELEASE_PRODS is included by test IOCs in the downstream modules. TOP=../..
    # - RELEASE_LIBS is included by downstream modules themselves. TOP=..

    echo "AREA_DETECTOR=${S}" >> "${S}/configure/RELEASE.local"

    # Point at ADSupport/ADCore
    echo 'ADSUPPORT=$(AREA_DETECTOR)/ADSupport' >> "${S}/configure/RELEASE.local"
```

... recipe continues on

IOCs & IOC management

procServ + systemd

- ECS uses a python application for managing IOCs on standard production systems
  - Requires access to NFS
- For this project, we're exploring managing IOCs using simple systemd services + procServ
- Example bbclass for generating service files for installed IOCs
- Plan to use simple scripts to manage IOCs on target
- Using an additional layer (meta-slac-epics) to integrate SLAC-specific EPICS modules and IOCs

meta-epics / classes / epics-ioc-systemd.bbclass

JJL772 Move update_env_paths to epics-functions.bbclass

Code Blame 81 lines (62 loc) · 2.75 KB ·

```
#
# BitBake class for EPICS IOCs that use systemd and procServ.
# Does the following:
#  - Builds and installs the IOC to /opt/epics like other EPICS modules
#  - Installs a systemd unit to start the IOC on system boot using procServ
#

inherit epics-module

# procServ is used to manage the IOCs, we'll also need telnet to access the IOC later
DEPENDS += "procserv"
RDEPENDS:${PN} += "procserv"

# --- User provided settings --- #

# Port to run procServ on
PS_PORT ?= "30000"

# IOC application name (located within this package's bin/<arch>)
# If left empty, it will rely on the shebang line in the st.cmd
IOC_APP_NAME ?= ""

# Path to the IOC, relative to the package root. ex: iocBoot/sioc-my-example
IOC_PATH ?= ""

# List of additional variable names to append to envPaths for the IOC.
# These will be expanded with the suffix _ENV. So, IOC_ENV += "PV_PREFIX" will be set to the value of ${PV_PREFIX_ENV} in Yocto.
IOC_ENV ?= ""

# Name of the IOC's st.cmd (usually just st.cmd)
IOC_ST_CMD ?= "st.cmd"

# --- End user provided settings --- #

# Installs a systemd unit to automatically start the IOC
install_systemd_unit() {
    U="${D}/etc/systemd/system/${PN}.service"
    SHS="${D}/opt/epics/${MODNAME}/ioc-start.sh"

    mkdir -p "$(dirname "${U}")"

```

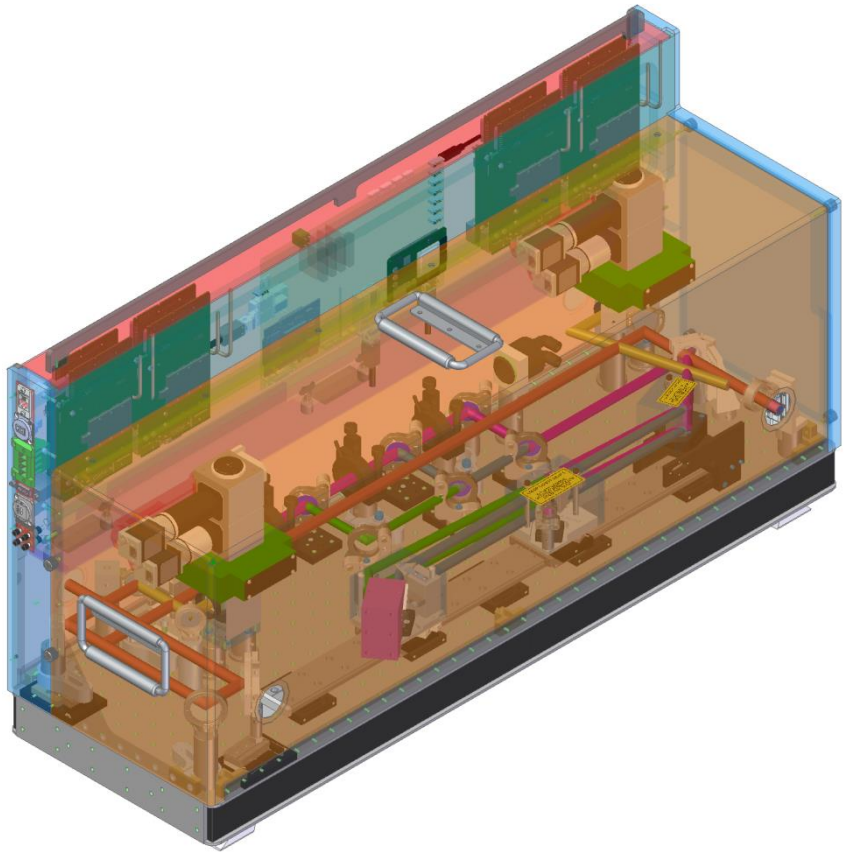
4

Future Work

What's next?

Full steam ahead to first article demonstration

- PCBs have been fabricated, need to be tested before integration in larger system
- First full demonstration will be a compressor TILE
    - Most fully-loaded TILE design thus far
- Following successful first article demonstration we will look to integrate these systems into production designs



meta-epics

- Continue to update and refine support
- Looking forward to any community engagement
    - GH Issues & PRs welcome! :)

Questions?