A MODEL-BASED APPROACH TO MASTERING COMPLEXITY IN FPGA-SoC SOFTWARE DEVELOPMENT

MPSI
TECHNOLOGIES

Alexander Wirthmüller
aw@mpsitechnologies.com

Introduction

Me and MPSI Technologies

- Diploma in Electrical Engineering
- Based in Munich

- R&D Engineer at Mynaric (FPGA-based error-correction algorithms for free-space optical laser communications)

- Founder and Director at MPSI Technologies
- MPSI Technologies: make Embedded Software development more fun by replacing repetitive tasks by model-based source code generation

FPGA-SoC landscape

Devices and applications

Product lines

Selection discussed here: CPU complex can run Embedded Linux

AMD XILINX ZYNQ

- from 2011: Zynq 7000 with Dual 32-bit ARM CPU and SRAM-based FPGA, internally connected via AXI
- from 2016: Zynq UltraScale+ with Quad 64-bit ARM CPU and additional real-time cores / accelerators

intel

intel CYCLONE

intel STRATIX

- from 2012: CycloneV with Dual 32-bit ARM CPU
- from 2016: Stratix 10 with Quad 64-bit ARM CPU

MICROCHIP Microsemi Product Portfolio

MICROCHIP PolarFire FPGA PolarFire SoC

RISC-V

- from 2019: PolarFire SoC with Quad 64-bit RISC-V CPU and antifuse-based FPGA

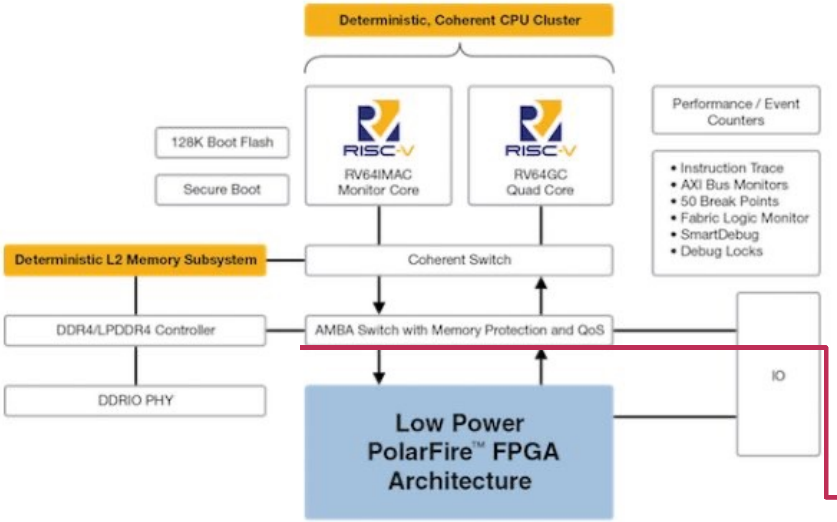
Typical applications

- classical FPGA applications where additional high-level control is of advantage
- ”data reduction” or pre-processing of high-bandwidth sources
- cameras: binning, pixel-level filters, compression, feature and object detection
- ADC’s: spectral analysis, DSP filters
- clock-precise signaling for mixed-signal ASIC’s
- not considered here: data center and hardware acceleration applications

The FPGA-SoC software full stack

From VHDL to C++ to HTTPS and XML

Microchip PolarFire SoC Block Diagram



Deterministic, Coherent CPU Cluster

128K Boot Flash

Secure Boot

RISC-V RV64IMAC Monitor Core

RISC-V RV64GC Quad Core

Performance / Event Counters

- Instruction Trace
- AXI Bus Monitors
- 50 Break Points
- Fabric Logic Monitor
- SmartDebug
- Debug Locks

Deterministic L2 Memory Subsystem

Coherent Switch

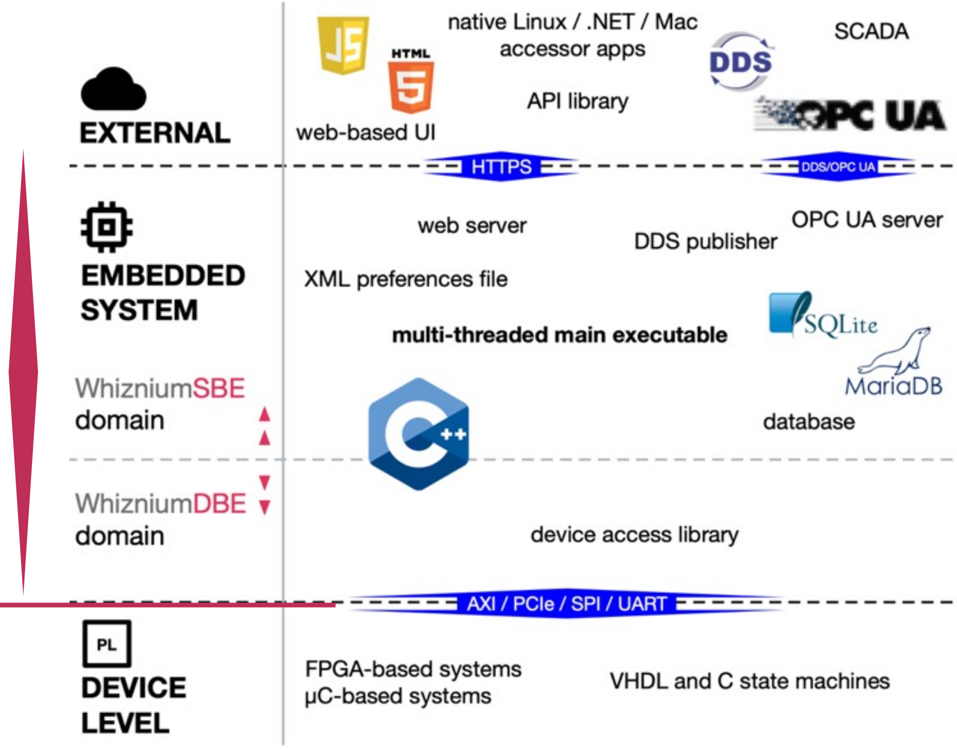
DDR4/LPDDR4 Controller

AMBA Switch with Memory Protection and QoS

DDRIO PHY

IO

Low Power PolarFire™ FPGA Architecture



EXTERNAL

web-based UI

native Linux / .NET / Mac accessor apps

API library

SCADA

HTTPS

DDS/OPC UA

EMBEDDED SYSTEM

web server

DDS publisher

OPC UA server

XML preferences file

multi-threaded main executable

database

WhizniumSBE domain

WhizniumDBE domain

device access library

AXI / PCIe / SPI / UART

DEVICE LEVEL

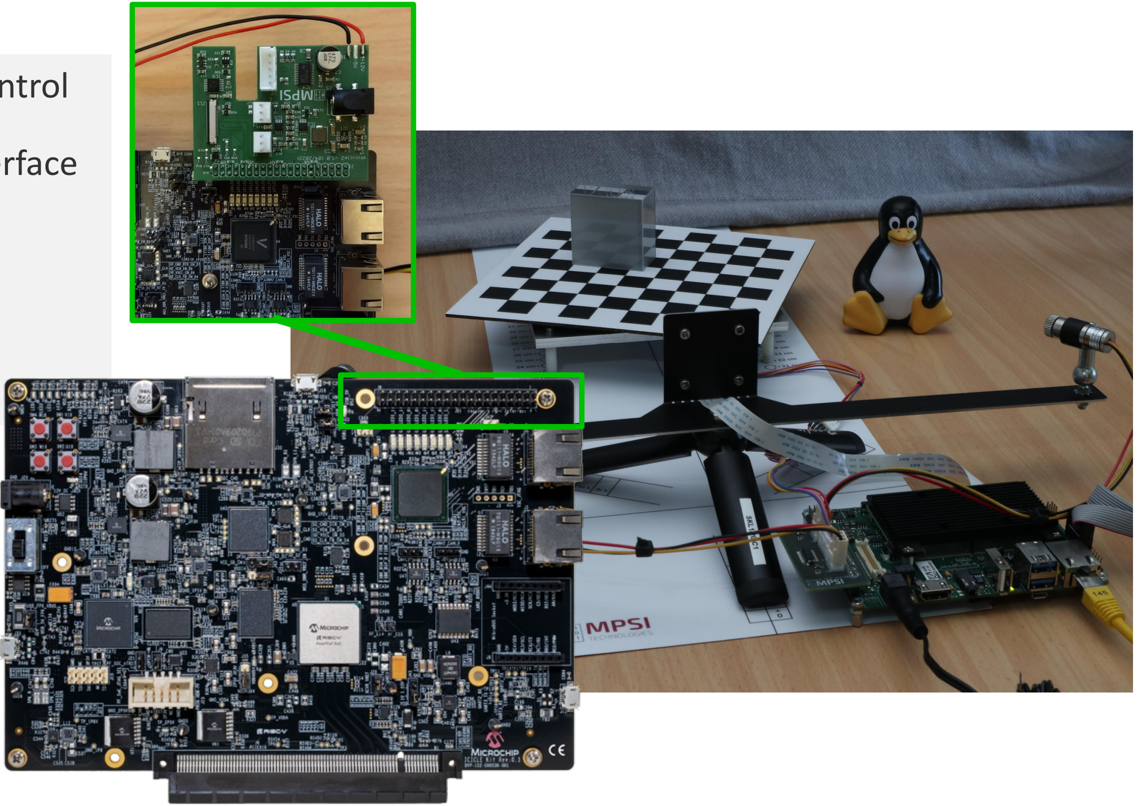
FPGA-based systems
µC-based systems

VHDL and C state machines

Demo project: hardware

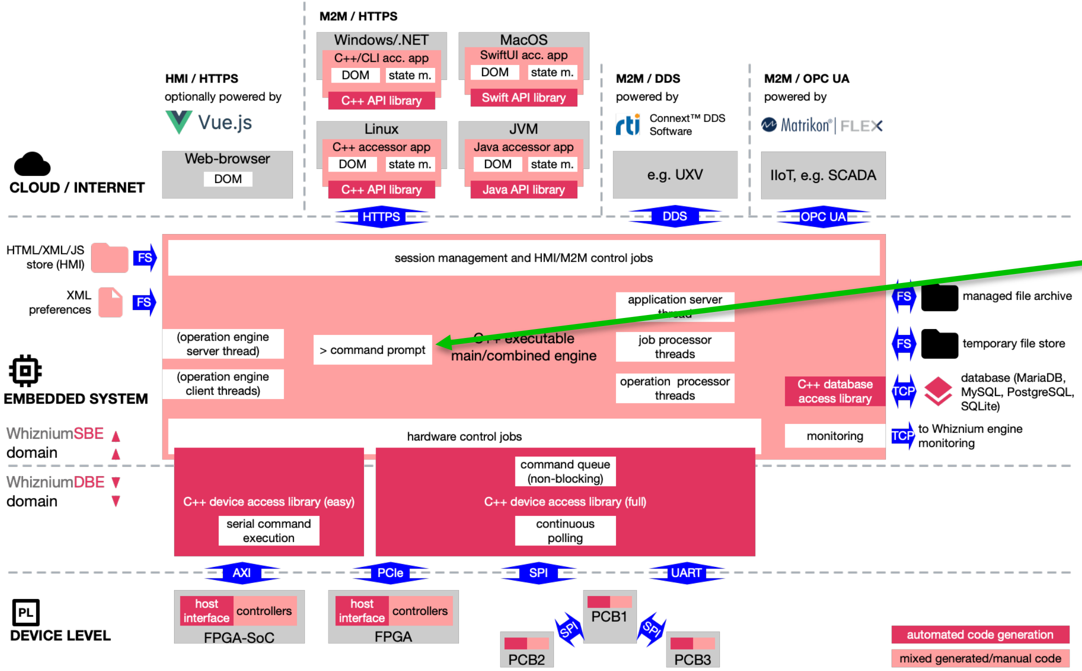
Tabletop 3D laser scanner

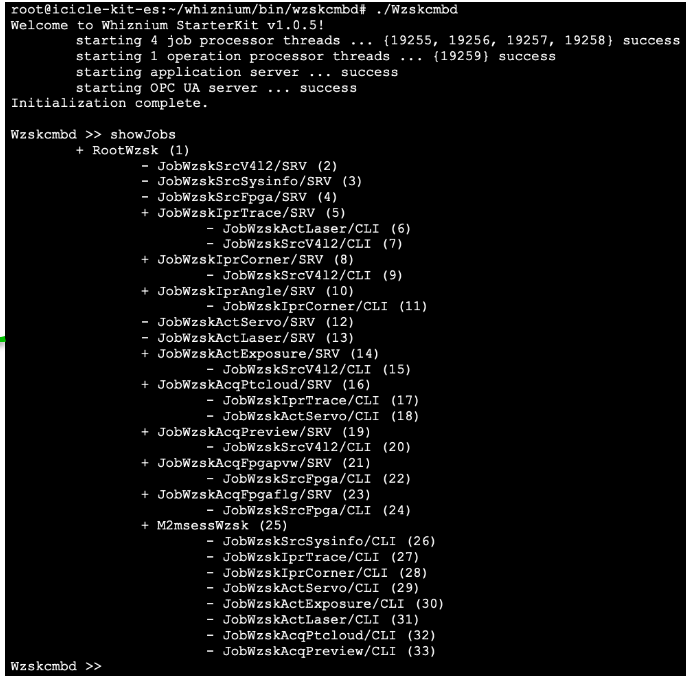
- turntable with with stepper motor control
- 5 megapixel camera with MIPI CSI interface
- two intensity-modulated line lasers
- Microchip PolarFire SoC Icicle kit with adapter PCB



Demo project: software

From camera raw data to point cloud display in web browser



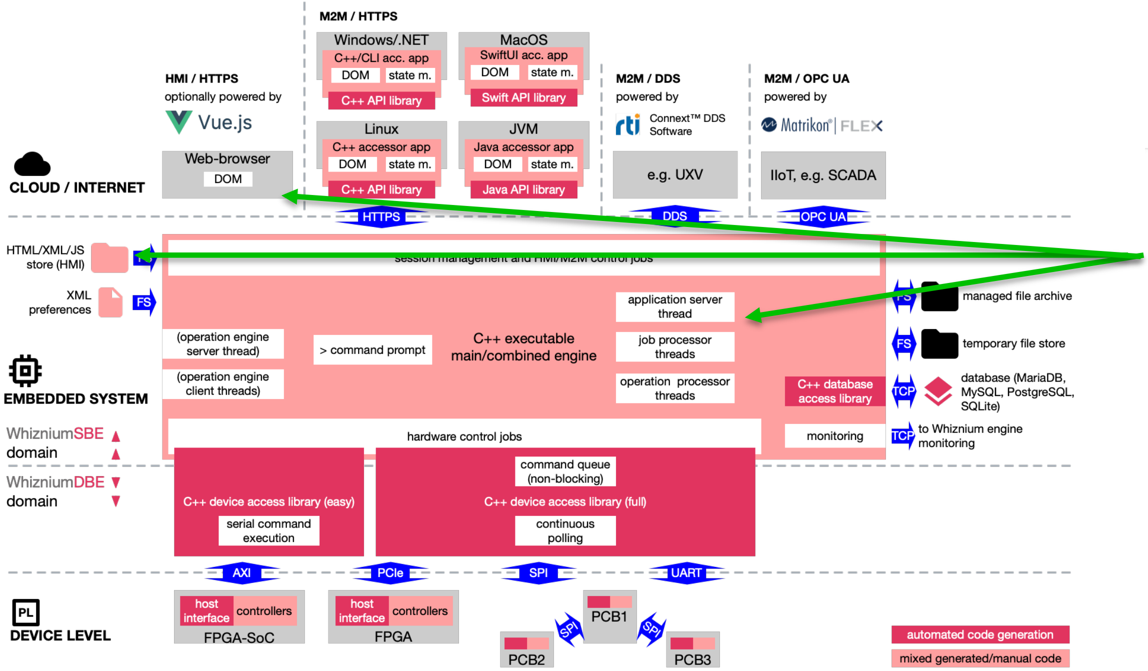


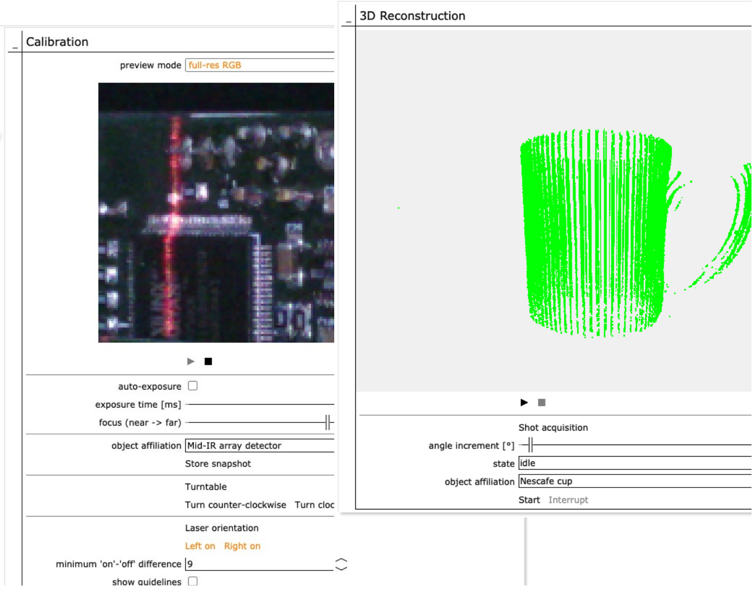
```
root@icicle-kit-es:~/whiznium/bin/wzskcmbd# ./Wzskcmbd
Welcome to Whiznium StarterKit v1.0.5!
        starting 4 job processor threads ... {19255, 19256, 19257, 19258} success
        starting 1 operation processor threads ... {19259} success
        starting application server ... success
        starting OPC UA server ... success
Initialization complete.

Wzskcmbd >> showJobs
        + RootWzsk (1)
                - JobWzskSrcV4l2/SRV (2)
                - JobWzskSrcSysinfo/SRV (3)
                - JobWzskSrcFpga/SRV (4)
                + JobWzskIprTrace/SRV (5)
                        - JobWzskActLaser/CLI (6)
                        - JobWzskSrcV4l2/CLI (7)
                + JobWzskIprCorner/SRV (8)
                        - JobWzskSrcV4l2/CLI (9)
                + JobWzskIprAngle/SRV (10)
                        - JobWzskIprCorner/CLI (11)
                - JobWzskActServo/SRV (12)
                - JobWzskActLaser/SRV (13)
                + JobWzskActExposure/SRV (14)
                        - JobWzskSrcV4l2/CLI (15)
                + JobWzskAcqPtcloud/SRV (16)
                        - JobWzskIprTrace/CLI (17)
                        - JobWzskActServo/CLI (18)
                + JobWzskAcqPreview/SRV (19)
                        - JobWzskSrcV4l2/CLI (20)
                + JobWzskAcqFpgapvw/SRV (21)
                        - JobWzskSrcFpga/CLI (22)
                + JobWzskAcqFpgaflg/SRV (23)
                        - JobWzskSrcFpga/CLI (24)
                + M2msessWzsk (25)
                        - JobWzskSrcSysinfo/CLI (26)
                        - JobWzskIprTrace/CLI (27)
                        - JobWzskIprCorner/CLI (28)
                        - JobWzskActServo/CLI (29)
                        - JobWzskActExposure/CLI (30)
                        - JobWzskActLaser/CLI (31)
                        - JobWzskAcqPtcloud/CLI (32)
                        - JobWzskAcqPreview/CLI (33)
Wzskcmbd >>
```

Demo project: software

From camera raw data to point cloud display in web browser





Calibration

preview mode: full-res RGB

auto-exposure □

exposure time [ms]

focus (near -> far)

object affiliation: Mid-IR array detector

Store snapshot

Turntable

Turn counter-clockwise Turn cloc

Laser orientation

Left on Right on

minimum 'on'-'off' difference: 9

show guidelines □

3D Reconstruction

Shot acquisition

angle increment [°]

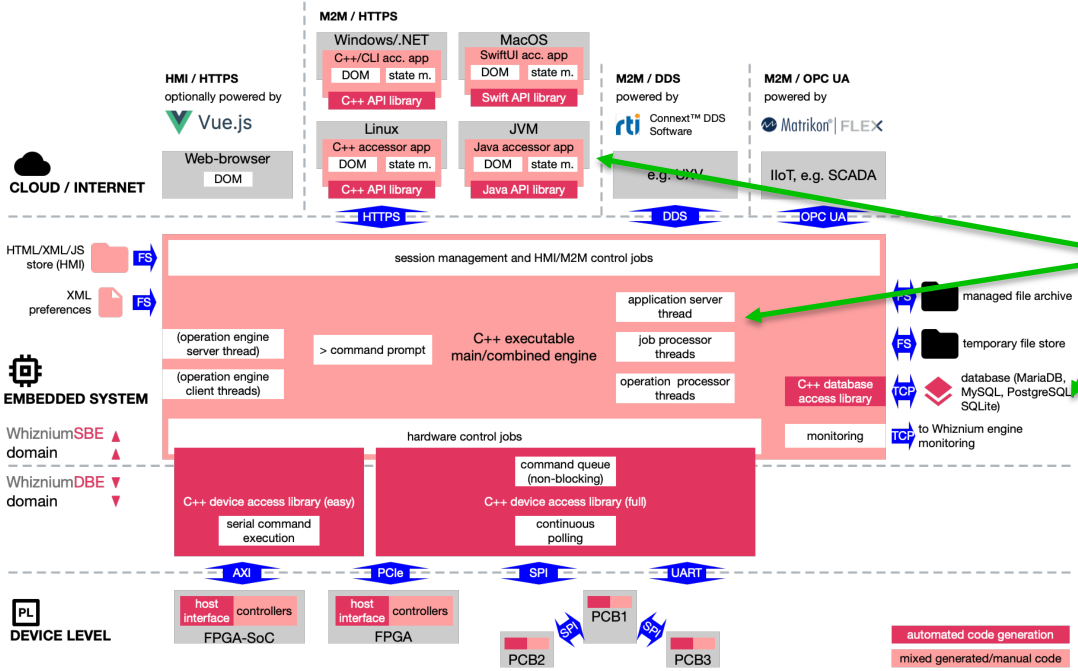
state: idle

object affiliation: Nescafe cup

Start Interrupt

Demo project: software

From camera raw data to point cloud display in web browser



CLOUD / INTERNET

HMI / HTTPS
optionally powered by
Vue.js

Web-browser
DOM

M2M / HTTPS

Windows/.NET
C++/CLI acc. app
DOM | state m.
C++ API library

MacOS
SwiftUI acc. app
DOM | state m.
Swift API library

Linux
C++ accessor app
DOM | state m.
C++ API library

JVM
Java accessor app
DOM | state m.
Java API library

M2M / DDS
powered by
Connext™ DDS Software
e.g. UXV

M2M / OPC UA
powered by
Matrikon® | FLEX
IIoT, e.g. SCADA

HTTPS | DDS | OPC UA

HTML/XML/JS store (HMI) — FS
XML preferences — FS

EMBEDDED SYSTEM

session management and HMI/M2M control jobs
(operation engine server thread)
(operation engine client threads)
> command prompt
C++ executable main/combined engine
application server thread
job processor threads
operation processor threads
C++ database access library
monitoring
hardware control jobs

FS — managed file archive
FS — temporary file store
TCP — database (MariaDB, MySQL, PostgreSQL, SQLite)
TCP — to Whiznium engine monitoring

WhizniumSBE domain
WhizniumDBE domain

C++ device access library (easy)
serial command execution
command queue (non-blocking)
C++ device access library (full)
continuous polling

AXI | PCIe | SPI | UART

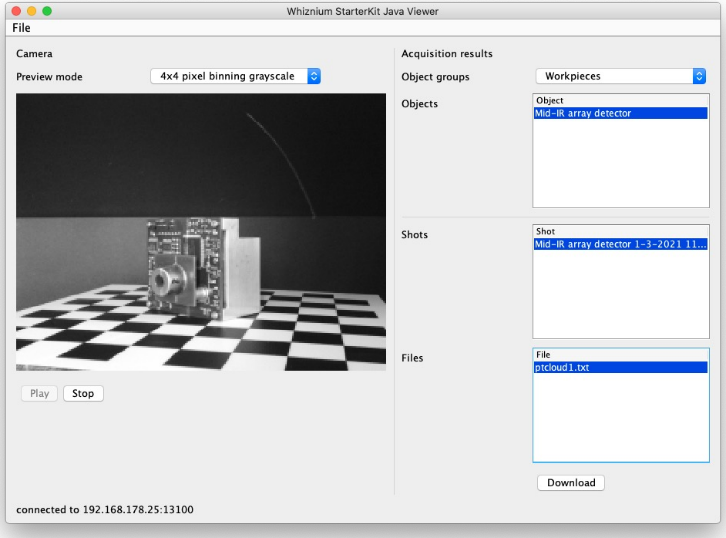
DEVICE LEVEL

host interface | controllers
FPGA-SoC

host interface | controllers
FPGA

PCB1
PCB2
PCB3
SPI
SPI

automated code generation
mixed generated/manual code



Whiznium StarterKit Java Viewer

File

Camera

Preview mode: 4x4 pixel binning grayscale

Play | Stop

Acquisition results

Object groups: Workpieces

Objects

Object
Mid-IR array detector

Shots

Shot
Mid-IR array detector 1-3-2021 11...

Files

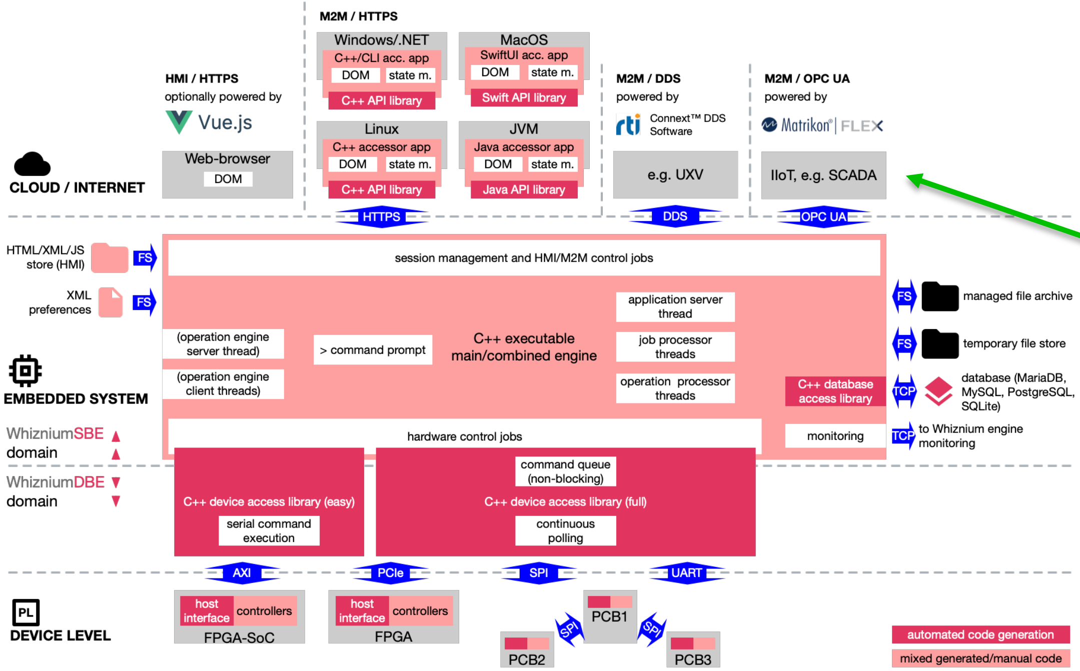
File
ptcloud1.txt

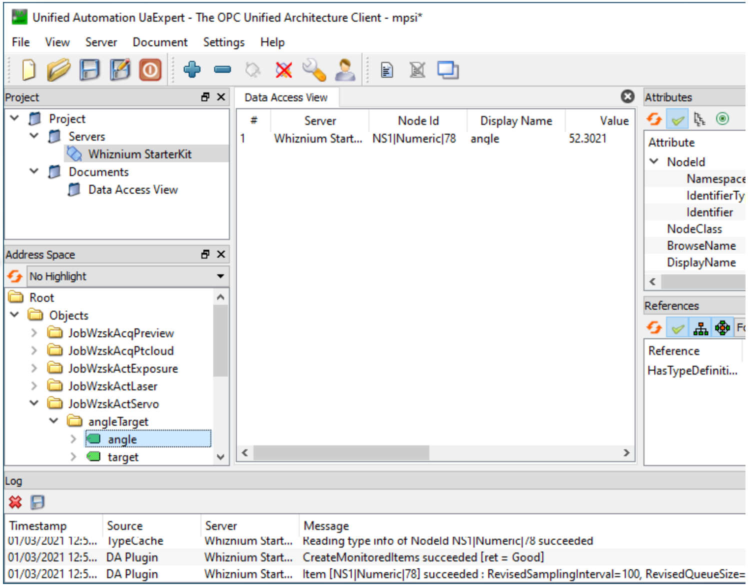
Download

connected to 192.168.178.25:13100

Demo project: software

From camera raw data to point cloud display in web browser





#	Server	Node Id	Display Name	Value
1	Whiznium Start...	NS1|Numeric|78	angle	52.3021

```
Timestamp            Source        Server              Message
01/03/2021 12:5...   TypeCache     Whiznium Start...   Reading type info of NodeId NS1|Numeric|78 succeeded
01/03/2021 12:5...   DA Plugin     Whiznium Start...   CreateMonitoredItems succeeded [ret = Good]
01/03/2021 12:5...   DA Plugin     Whiznium Start...   Item [NS1|Numeric|78] succeeded : RevisedSamplingInterval=100, RevisedQueueSize=
```

Embedded software model description

WhizniumDBE for FPGA layer, WhizniumSBE for Linux application software

- successive model composition within an SQL database using import (I) and generation (G) steps
- output of source code trees only thereafter
- text-based model files (”diffable”)

WhizniumDBE (Device Builder’s Edition)

- Modular structure (I)
- Command set and buffer transfers (I)
- Data flows and algorithms (I)
- Fine structure (G)
- Custom fine structure (I)
- Finalization (G)

WhizniumSBE (Service Builder’s Edition)

- Deployment information (I)
- Global features (I)
- Database structure (I)
- Basic user interface structure (I)
- Import/export structure (I)
- Operation pack structure (I)
- Custom jobs (I)
- User interface (G)
- Custom user interface features (I)
- Job tree (G)
- Custom job tree features (I)
- Finalization (G)

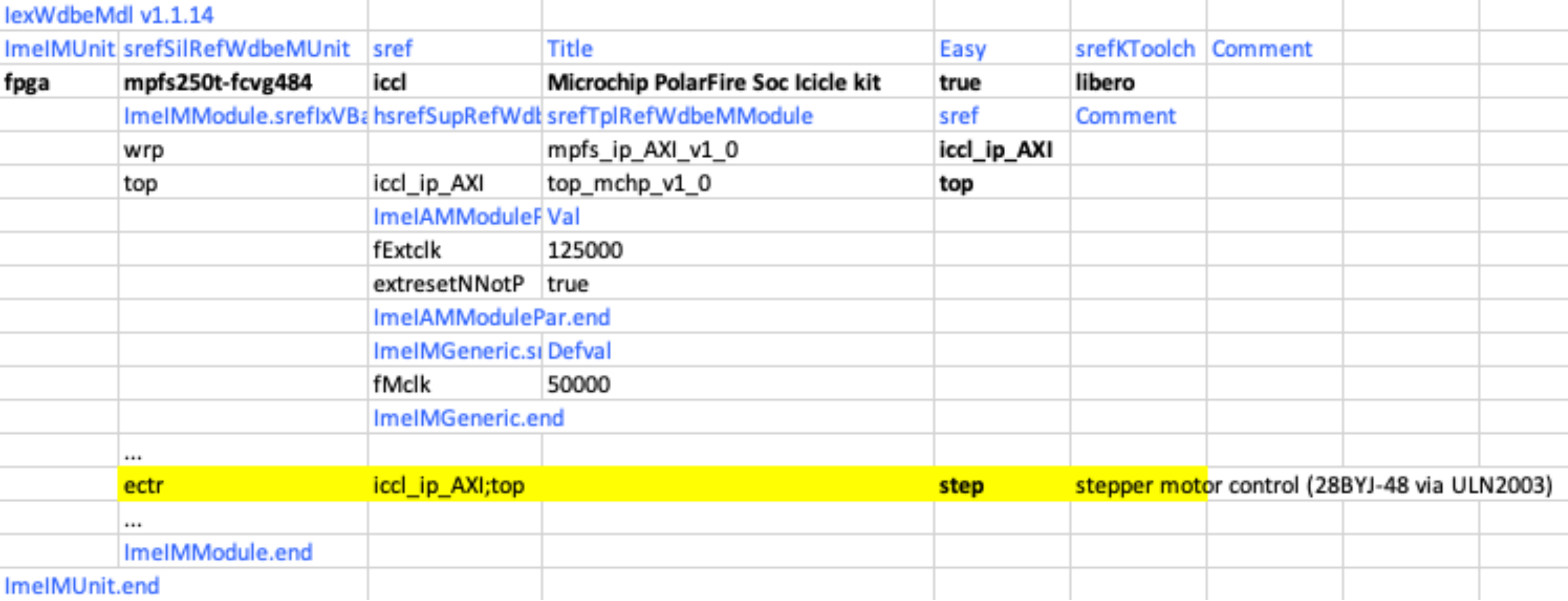
Deep dive I: from C++ command to RTL finite state machine

Control of turntable stepper motor

- module definition, command definition, fine structure

Deep dive I: from C++ command to RTL finite state machine

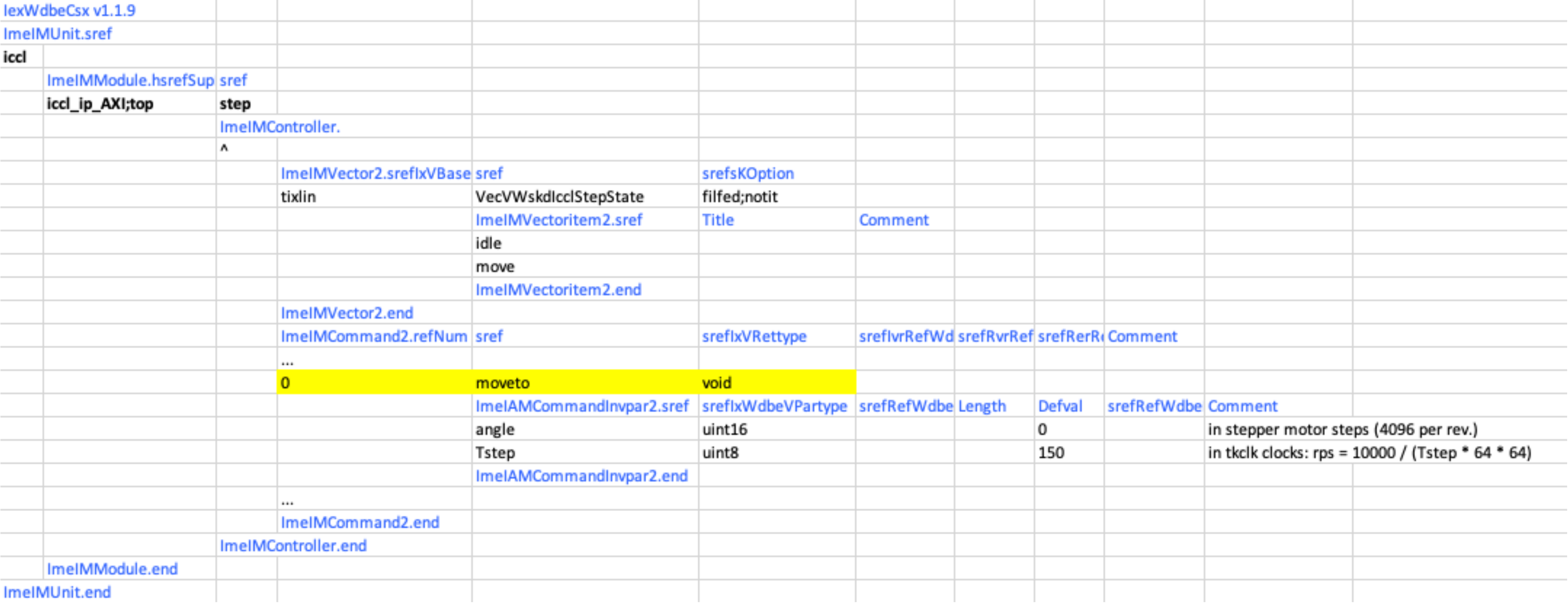
Control of turntable stepper motor



IexWdbeMdl v1.1.14								
ImelMUnit	srefSilRefWdbeMUnit	sref	Title	Easy	srefKToolch	Comment		
fpga	**mpfs250t-fcvg484**	**iccl**	**Microchip PolarFire Soc Icicle kit**	**true**	**libero**			
	ImelMModule.srefIxVBa	hsrefSupRefWdb	srefTplRefWdbeMModule	sref	Comment			
	wrp		mpfs_ip_AXI_v1_0	**iccl_ip_AXI**				
	top	iccl_ip_AXI	top_mchp_v1_0	**top**				
		ImelAMModuleP	Val					
		fExtclk	125000					
		extresetNNotP	true					
		ImelAMModulePar.end						
		ImelMGeneric.s	Defval					
		fMclk	50000					
		ImelMGeneric.end						
	...							
	ectr	iccl_ip_AXI;top		**step**	stepper motor control (28BYJ-48 via ULN2003)			
	...							
	ImelMModule.end							
ImelMUnit.end								

Deep dive I: from C++ command to RTL finite state machine

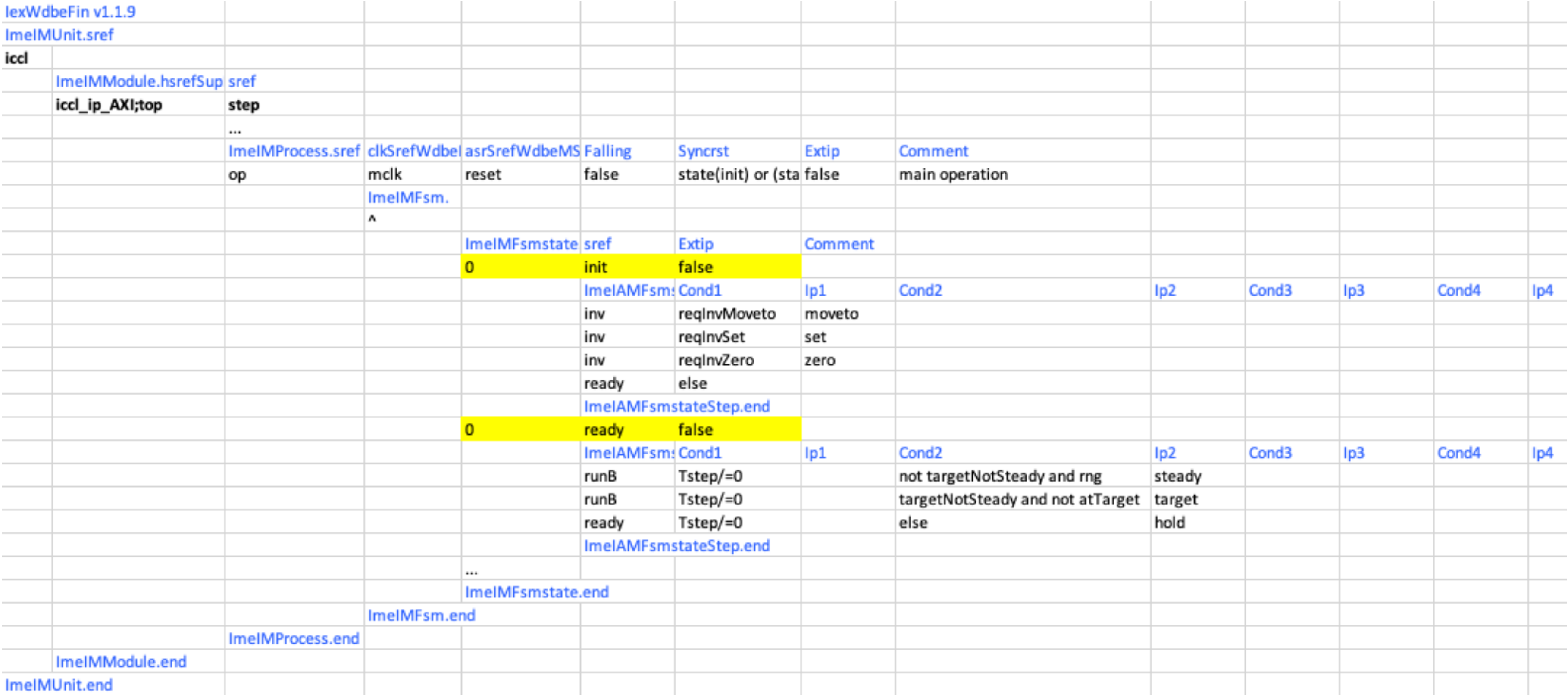
Control of turntable stepper motor



IexWdbeCsx v1.1.9										
ImeIMUnit.sref										
iccl										
	ImeIMModule.hsrefSup	sref								
	iccl_ip_AXI;top	step								
		ImeIMController.								
		^								
			ImeIMVector2.srefIxVBase	sref	srefsKOption					
			tixlin	VecVWskdIcclStepState	filfed;notit					
				ImeIMVectoritem2.sref	Title	Comment				
				idle						
				move						
				ImeIMVectoritem2.end						
			ImeIMVector2.end							
			ImeIMCommand2.refNum	sref	srefIxVRettype	srefIvrRefWd	srefRvrRef	srefRerR	Comment	
			...							
			0	moveto	void					
				ImeIAMCommandInvpar2.sref	srefIxWdbeVPartype	srefRefWdbe	Length	Defval	srefRefWdbe	Comment
				angle	uint16			0		in stepper motor steps (4096 per rev.)
				Tstep	uint8			150		in tkclk clocks: rps = 10000 / (Tstep * 64 * 64)
				ImeIAMCommandInvpar2.end						
			...							
			ImeIMCommand2.end							
		ImeIMController.end								
	ImeIMModule.end									
ImeIMUnit.end										

Deep dive I: from C++ command to RTL finite state machine

Control of turntable stepper motor



IexWdbeFin v1.1.9													
ImeIMUnit.sref													
iccl													
	ImeIMModule.hsrefSup	sref											
	iccl_ip_AXI;top	step											
		...											
		ImeIMProcess.sref	clkSrefWdbeI	asrSrefWdbeMS	Falling	Syncrst	Extip	Comment					
		op	mclk	reset	false	state(init) or (sta	false	main operation					
			ImeIMFsm.										
			^										
				ImeIMFsmstate	sref	Extip	Comment						
				0	init	false							
					ImeIAMFsm:	Cond1	Ip1	Cond2	Ip2	Cond3	Ip3	Cond4	Ip4
					inv	reqInvMoveto	moveto						
					inv	reqInvSet	set						
					inv	reqInvZero	zero						
					ready	else							
					ImeIAMFsmstateStep.end								
				0	ready	false							
					ImeIAMFsm:	Cond1	Ip1	Cond2	Ip2	Cond3	Ip3	Cond4	Ip4
					runB	Tstep/=0		not targetNotSteady and rng	steady				
					runB	Tstep/=0		targetNotSteady and not atTarget	target				
					ready	Tstep/=0		else	hold				
					ImeIAMFsmstateStep.end								
				...									
				ImeIMFsmstate.end									
			ImeIMFsm.end										
		ImeIMProcess.end											
	ImeIMModule.end												
ImeIMUnit.end													

Deep dive I: from C++ command to RTL finite state machine

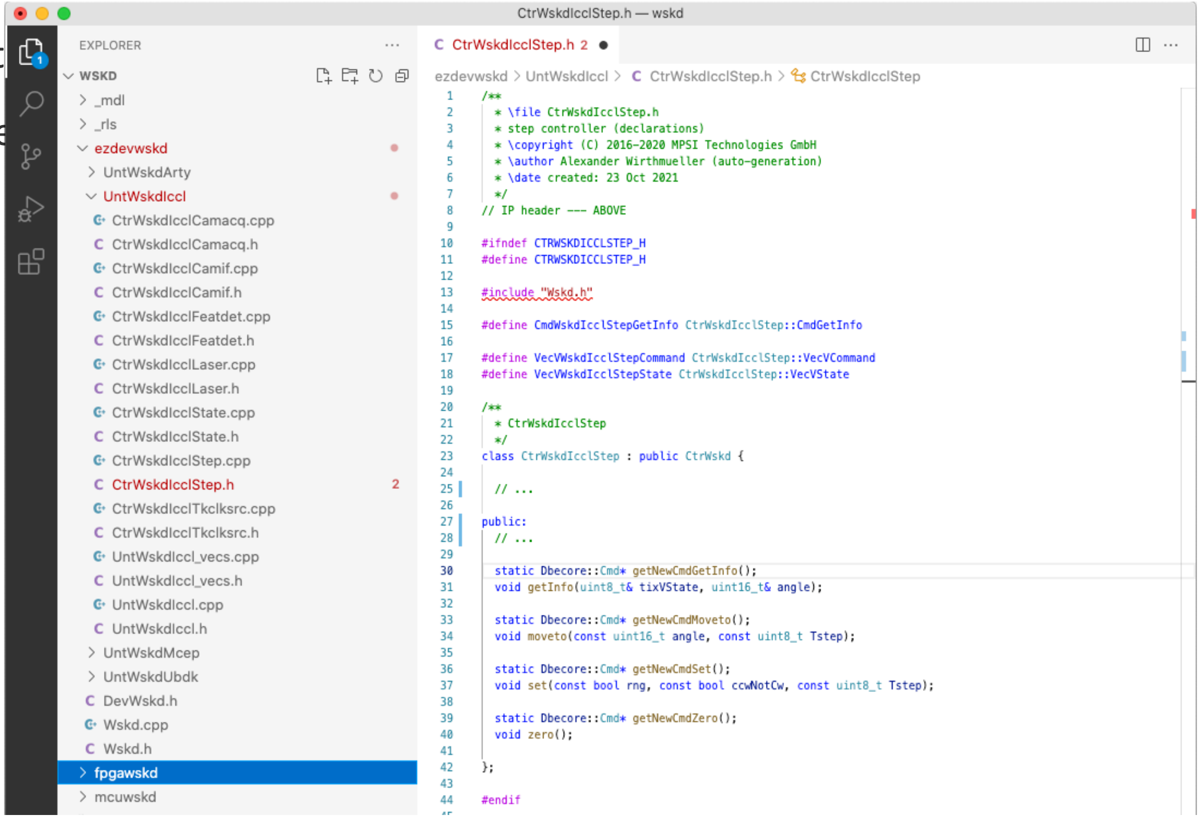
Control of turntable stepper motor

- module definition, command definition, fine structure
- Linux side developer-facing: executable API method

Deep dive I: from C++ command to RTL finite state machine

Control of turntable stepper motor

- module definit
- Linux side deve



```cpp
/**
  * \file CtrWskdIcclStep.h
  * step controller (declarations)
  * \copyright (C) 2016-2020 MPSI Technologies GmbH
  * \author Alexander Wirthmueller (auto-generation)
  * \date created: 23 Oct 2021
  */
// IP header --- ABOVE

#ifndef CTRWSKDICCLSTEP_H
#define CTRWSKDICCLSTEP_H

#include "Wskd.h"

#define CmdWskdIcclStepGetInfo CtrWskdIcclStep::CmdGetInfo

#define VecVWskdIcclStepCommand CtrWskdIcclStep::VecVCommand
#define VecVWskdIcclStepState CtrWskdIcclStep::VecVState

/**
  * CtrWskdIcclStep
  */
class CtrWskdIcclStep : public CtrWskd {

  // ...

public:
  // ...

  static Dbecore::Cmd* getNewCmdGetInfo();
  void getInfo(uint8_t& tixVState, uint16_t& angle);

  static Dbecore::Cmd* getNewCmdMoveto();
  void moveto(const uint16_t angle, const uint8_t Tstep);

  static Dbecore::Cmd* getNewCmdSet();
  void set(const bool rng, const bool ccwNotCw, const uint8_t Tstep);

  static Dbecore::Cmd* getNewCmdZero();
  void zero();

};

#endif
```

Deep dive I: from C++ command to RTL finite state machine

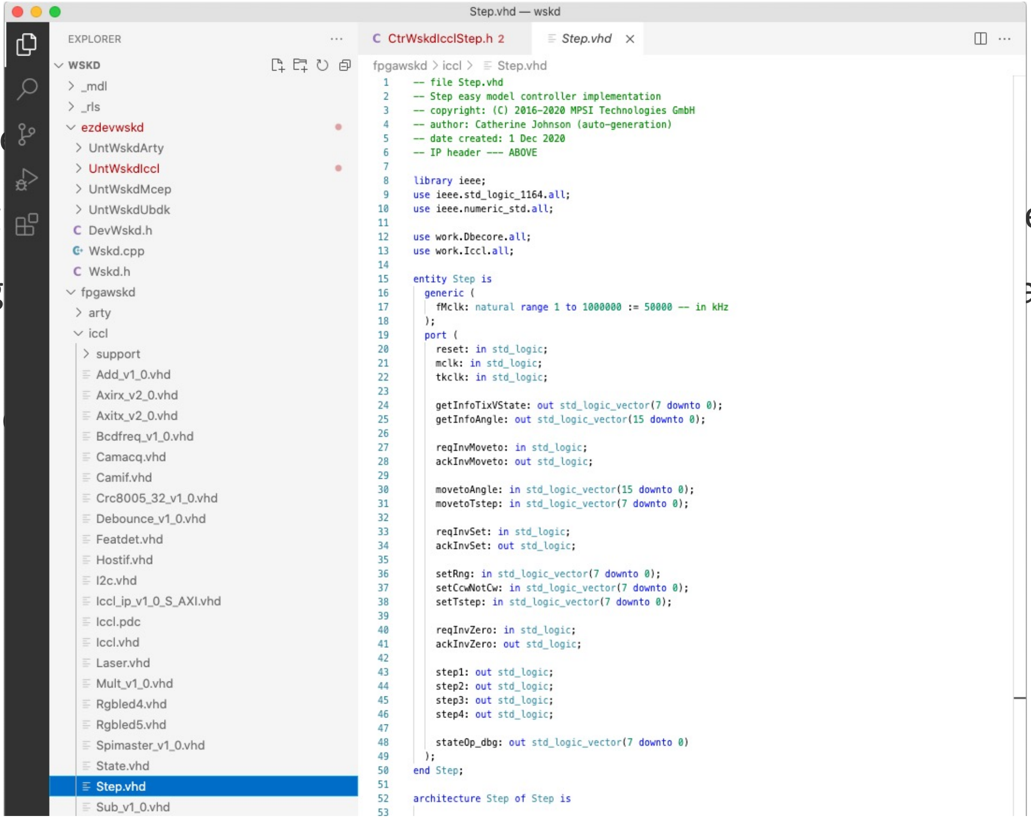
Control of turntable stepper motor

- module definition, command definition, fine structure
- Linux side developer-facing: executable API method
- Linux side in background: translation into byte code and invocation of character device driver (AXI)
- FPGA side in background: reception and decoding of byte code in “host interface” module, CRC evaluation
- FPGA side developer-facing: handshake signals

Deep dive I: from C++ command to RTL finite state machine

Control of turntable stepper motor

- module definition,
- Linux side develope
- Linux side in backg er device driver (AXI)
- FPGA side in backg ace” module, CRC evaluation
- FPGA side develope



```
-- file Step.vhd
-- Step easy model controller implementation
-- copyright: (C) 2016-2020 MPSI Technologies GmbH
-- author: Catherine Johnson (auto-generation)
-- date created: 1 Dec 2020
-- IP header --- ABOVE

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

use work.Dbecore.all;
use work.Iccl.all;

entity Step is
  generic (
    fMclk: natural range 1 to 1000000 := 50000 -- in kHz
  );
  port (
    reset: in std_logic;
    mclk: in std_logic;
    tkclk: in std_logic;

    getInfoTixVState: out std_logic_vector(7 downto 0);
    getInfoAngle: out std_logic_vector(15 downto 0);

    reqInvMoveto: in std_logic;
    ackInvMoveto: out std_logic;

    movetoAngle: in std_logic_vector(15 downto 0);
    movetoTstep: in std_logic_vector(7 downto 0);

    reqInvSet: in std_logic;
    ackInvSet: out std_logic;

    setRng: in std_logic_vector(7 downto 0);
    setCcwNotCw: in std_logic_vector(7 downto 0);
    setTstep: in std_logic_vector(7 downto 0);

    reqInvZero: in std_logic;
    ackInvZero: out std_logic;

    step1: out std_logic;
    step2: out std_logic;
    step3: out std_logic;
    step4: out std_logic;

    stateOp_dbg: out std_logic_vector(7 downto 0)
  );
end Step;

architecture Step of Step is
```

Deep dive I: from C++ command to RTL finite state machine

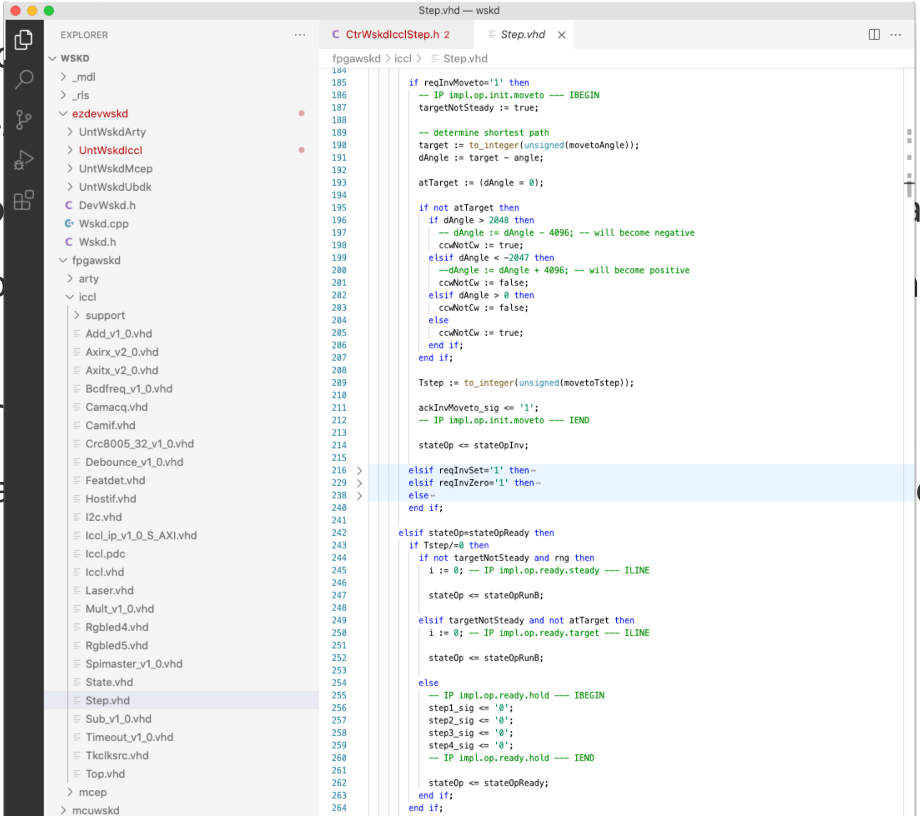
Control of turntable stepper motor

- module definition, command definition, fine structure
- Linux side developer-facing: executable API method
- Linux side in background: translation into byte code and invocation of character device driver (AXI)
- FPGA side in background: reception and decoding of byte code in “host interface” module, CRC evaluation
- FPGA side developer-facing: handshake signals
- FPGA side left for manual implementation: finite state machine reacting to command invocation

Deep dive I: from C++ command to RTL finite state machine

Control of turntable stepper motor

- module definition, co
- Linux side developer
- Linux side in backgro … aracter device driver (AXI)
- FPGA side in backgro … nterface” module, CRC evaluation
- FPGA side developer
- FPGA side left for ma … o command invocation



```vhdl
        if reqInvMoveto='1' then
          -- IP impl.op.init.moveto --- IBEGIN
          targetNotSteady := true;

          -- determine shortest path
          target := to_integer(unsigned(movetoAngle));
          dAngle := target - angle;

          atTarget := (dAngle = 0);

          if not atTarget then
            if dAngle > 2048 then
              -- dAngle := dAngle - 4096; -- will become negative
              ccwNotCw := true;
            elsif dAngle < -2047 then
              --dAngle := dAngle + 4096; -- will become positive
              ccwNotCw := false;
            elsif dAngle > 0 then
              ccwNotCw := false;
            else
              ccwNotCw := true;
            end if;
          end if;

          Tstep := to_integer(unsigned(movetoTstep));

          ackInvMoveto_sig <= '1';
          -- IP impl.op.init.moveto --- IEND

          stateOp <= stateOpInv;

        elsif reqInvSet='1' then⋯
        elsif reqInvZero='1' then⋯
        else⋯
        end if;

      elsif stateOp=stateOpReady then
        if Tstep/=0 then
          if not targetNotSteady and rng then
            i := 0; -- IP impl.op.ready.steady --- ILINE

            stateOp <= stateOpRunB;

          elsif targetNotSteady and not atTarget then
            i := 0; -- IP impl.op.ready.target --- ILINE

            stateOp <= stateOpRunB;

          else
            -- IP impl.op.ready.hold --- IBEGIN
            step1_sig <= '0';
            step2_sig <= '0';
            step3_sig <= '0';
            step4_sig <= '0';
            -- IP impl.op.ready.hold --- IEND

            stateOp <= stateOpReady;
          end if;
        end if;
```

Deep dive II: camera preview images on the move

FPGA-based binning, processing in C++ code and forwarding to the UI

- pixels arrive over four-lane MIPI CSI at 576Mbps per lane and get deserialized into a 10-bit RAW10 data stream at about 20fps
- four preview modes (2560 x 1280 to 160 x 120 RGB vs. 2048 x 1536 to 256 x 192 grayscale), manual implementation using finite state machines and 2/4kB buffers
- ”buffer transfers” are besides “commands” the second functionality which can be generated for the host interface
- polling in separate thread on host, insertion into ”job tree” via “call”
- generation of XML block containing image data, Base64 coded transmission
- reception in web browser via HTTPS/1.1 “long polling”, rendering in HTML5 <canvas/>

Deep dive III: meta data and the “lowering” process in Whiznium

From SQL database structure to code and UI elements to XML/JSON blocks

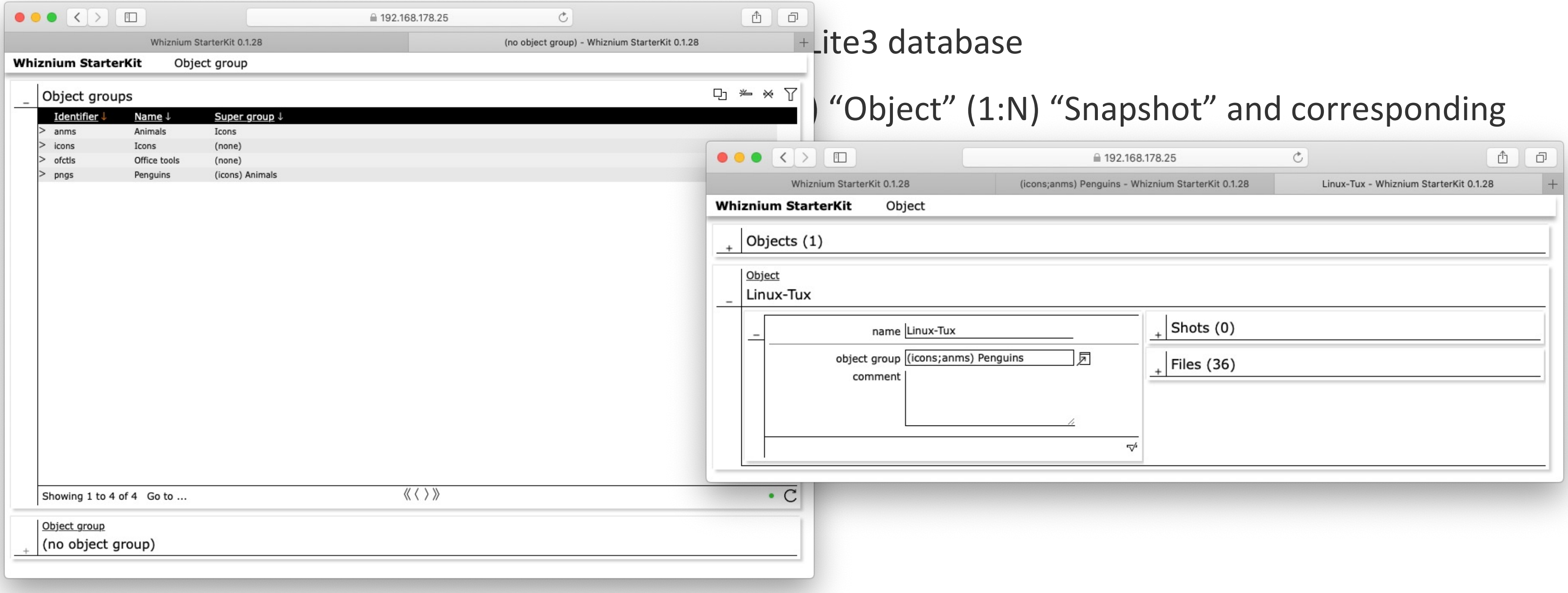
- all WhizniumSBE applications are backed by a SQLite3 database
- “Database structure”: tables “Object group” (1:N) “Object” (1:N) “Snapshot” and corresponding ”Basic user interface”

Deep dive III: meta data and the “lowering” process in Whiznium

From SQL database structure to code and UI elements to XML/JSON blocks

ite3 database

) “Object” (1:N) “Snapshot” and corresponding



192.168.178.25

Whiznium StarterKit 0.1.28 | (no object group) - Whiznium StarterKit 0.1.28

Whiznium StarterKit Object group

Object groups

Identifier ↓	Name ↓	Super group ↓
anms	Animals	Icons
icons	Icons	(none)
ofctls	Office tools	(none)
pngs	Penguins	(icons) Animals

Showing 1 to 4 of 4 Go to ...

Object group
(no object group)

192.168.178.25

Whiznium StarterKit 0.1.28 | (icons;anms) Penguins - Whiznium StarterKit 0.1.28 | Linux-Tux - Whiznium StarterKit 0.1.28

Whiznium StarterKit Object

Objects (1)

Object
Linux-Tux

name: Linux-Tux
object group: (icons;anms) Penguins
comment:

Shots (0)

Files (36)

Deep dive III: meta data and the “lowering” process in Whiznium

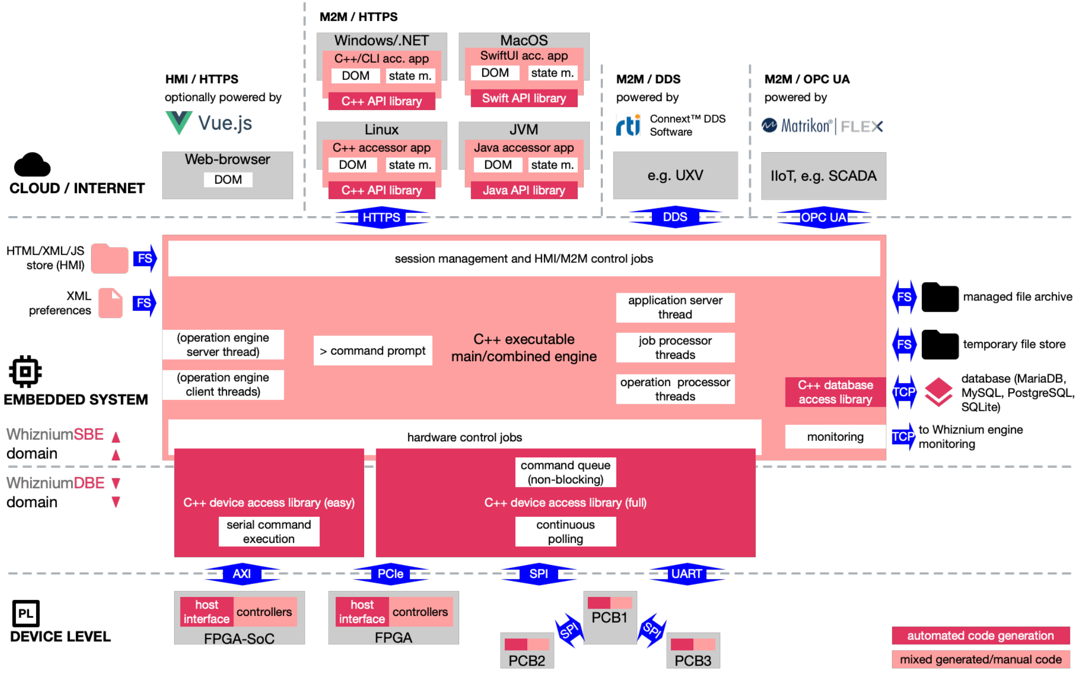
From SQL database structure to code and UI elements to XML/JSON blocks

- all WhizniumSBE applications are backed by a SQLite3 database
- “Database structure”: tables “Object group” (1:N) “Object” (1:N) “Snapshot” and corresponding ”Basic user interface”
- first “lowering” step: multi-locale UI elements, “queries”, “panels” and “controls”
- second “lowering” step: “blocks” and “dispatches” for engine <-> app communication
- code generation “database access library”
- code generation on engine side: classes for “cards”, ”panels” and ”queries” which at runtime dynamically generate objects responsible for web UI sessions and react to user input
- code generation app side: HTML and JavaScript

Conclusion demo project

The model-based approach pays off

- the maze of software technologies relevant for FPGA-SoC’s is cleanly covered by a single, coherent method



Whiznium concepts

Modularity, transparency and re-usability

- Whiznium is Open Source; the generated code is subject to no license restrictions
- Whiznium generates well-organized, human-readable source code trees which can be synthesized / compiled “out-of-the-box”
- manual modifications are enabled through the concept of “insertion points”
- upon source code iteration (e.g. following model extension) manual modifications are carried over to the next version
- generated code relies on few, well-proven external libraries, all of which are Open Source. Standards are strictly followed
- WhizniumDBE features parametrized “module templates”. Besides corresponding VHDL files, template-specific intervention in the WhizniumDBE master database through C++ code is possible
- WhizniumSBE features parametrized “capability templates”. Also here, template-specific intervention in the WhizniumSBE master database through C++ code is possible

Whiznium tools

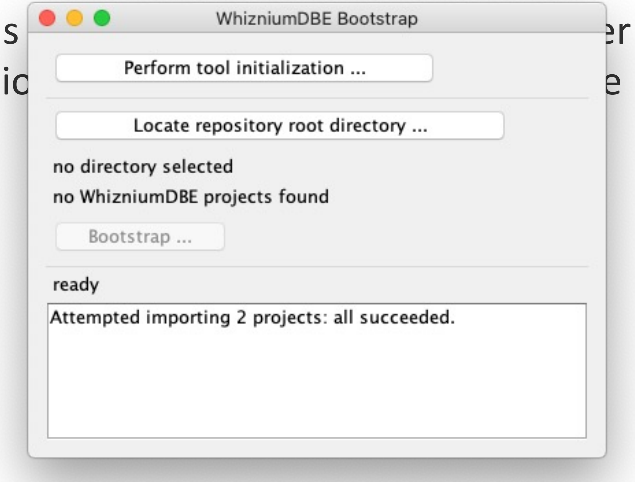
Incorporation into existing developer workflows

- WhizniumSBE and WhizniumDBE are Linux-based ”daemons” (and [fun fact] WhizniumSBE projects), which receive model information and send source code trees via HTTPS
- Java tools WhizniumDBE/SBE Bootstrap offer initialization of WhizniumDBE/SBE with project information stored in a local folder structure

Whiznium tools

Incorporation into existing developer workflows

- WhizniumSBE and WhizniumDBE are Linux-based ”daemons” (and [fun fact] WhizniumSBE projects), which receive model information and send source code trees via HTTPS
- Java tools er initializa oject informatio e

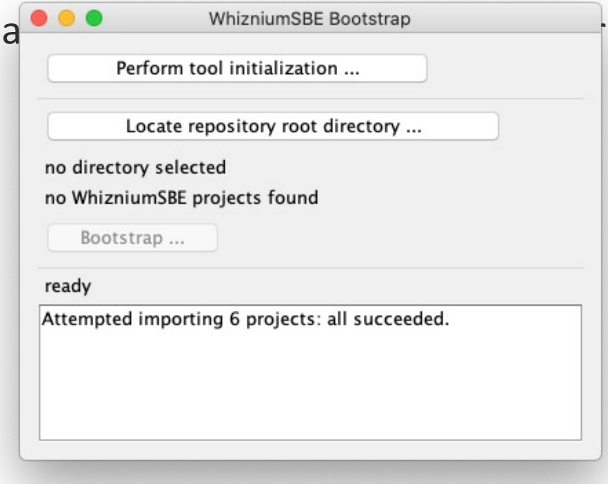

WhizniumDBE Bootstrap
Perform tool initialization ...
Locate repository root directory ...
Bootstrap ...

no directory selected

no WhizniumDBE projects found

ready

```
Attempted importing 2 projects: all succeeded.
```


WhizniumSBE Bootstrap
Perform tool initialization ...
Locate repository root directory ...
Bootstrap ...

no directory selected

no WhizniumSBE projects found

ready

```
Attempted importing 6 projects: all succeeded.
```

Whiznium tools

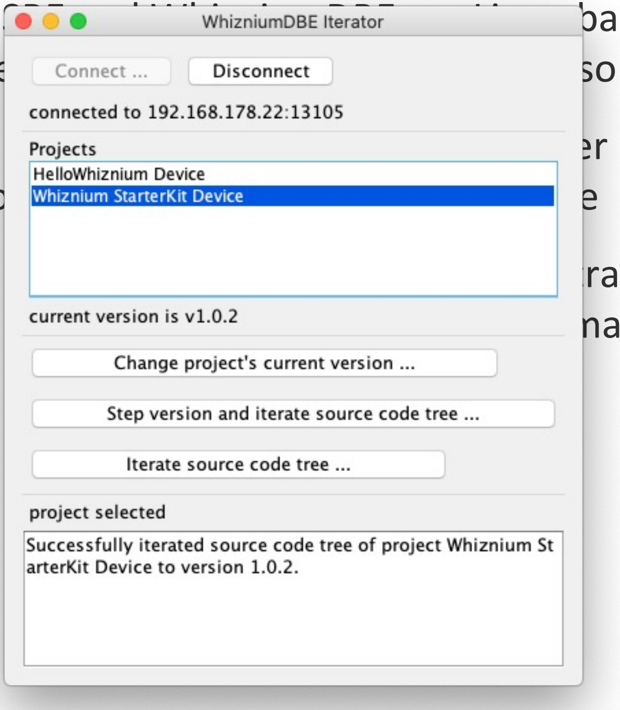
Incorporation into existing developer workflows

- WhizniumSBE and WhizniumDBE are Linux-based ”daemons” (and [fun fact] WhizniumSBE projects), which receive model information and send source code trees via HTTPS
- Java tools WhizniumDBE/SBE Bootstrap offer initialization of WhizniumDBE/SBE with project information stored in a local folder structure
- Java tools WhizniumDBE/SBE Iterator help transform local source code trees from the current version to the next. Here, API calls replace manual UI clicks

Whiznium tools

Incorporation into existing developer workflows

- Whiznium based ”dae SBE projects), which rece source coc
- Java tools er initializa roject informatic e
- Java tools ransform l urrent version to nanual UI


WhizniumDBE Iterator
Connect ...
Disconnect

connected to 192.168.178.22:13105

Projects

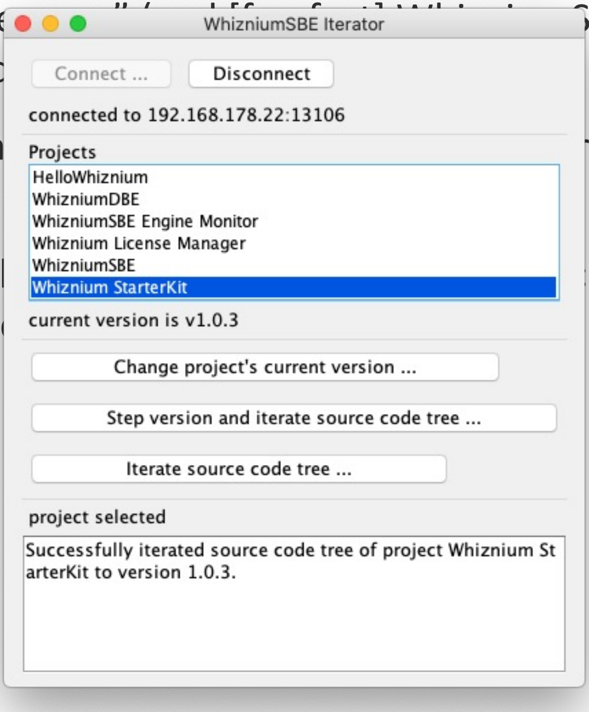
HelloWhiznium Device
Whiznium StarterKit Device

current version is v1.0.2

Change project's current version ...
Step version and iterate source code tree ...
Iterate source code tree ...

project selected

Successfully iterated source code tree of project Whiznium StarterKit Device to version 1.0.2.


WhizniumSBE Iterator
Connect ...
Disconnect

connected to 192.168.178.22:13106

Projects

HelloWhiznium
WhizniumDBE
WhizniumSBE Engine Monitor
Whiznium License Manager
WhizniumSBE
Whiznium StarterKit

current version is v1.0.3

Change project's current version ...
Step version and iterate source code tree ...
Iterate source code tree ...

project selected

Successfully iterated source code tree of project Whiznium StarterKit to version 1.0.3.

Whiznium tools

Incorporation into existing developer workflows

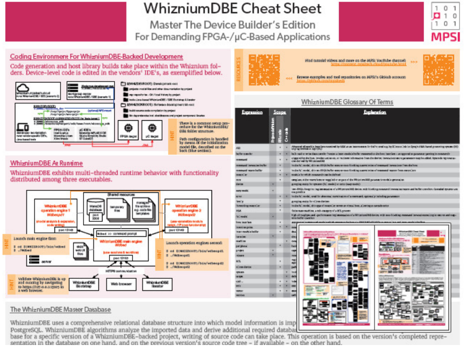
- WhizniumSBE and WhizniumDBE are Linux-based ”daemons” (and [fun fact] WhizniumSBE projects), which receive model information and send source code trees via HTTPS
- Java tools WhizniumDBE/SBE Bootstrap offer initialization of WhizniumDBE/SBE with project information stored in a local folder structure
- Java tools WhizniumDBE/SBE Iterator help transform local source code trees from the current version to the next. Here, API calls replace manual UI clicks
- WhizniumDBE code can be developed using the vendor-provided tools, e.g. Vivado, Quartus, Libero SoC or Simplicity Studio
- WhizniumSBE code can be (cross-)compiled using the industry-standard tool chains gcc/Clang. (Remote-)Debugging can be done using e.g. VS Code
- the Yocto project helps building custom Embedded Linux distributions for each FPGA-SoC platform. WhizniumSBE projects run on those distributions

Resources

- both Whiznium tools are available free of charge on GitHub, including installation instructions

  https://github.com/mpsitech/The-Whiznium-Documentation

- the Open Source StarterKit ist available for various hardware platforms, with vendor-specific instructions also available on GitHub
- "The Whiznium Developer Experience" on YouTube is an ongoing Webinar series on Whiznium
- for advanced users WhizniumSBE/DBE cheat sheets are available which serve as reference for writing model files



WhizniumDBE Cheat Sheet

Master The Device Builder's Edition

For Demanding FPGA-/µC-Based Applications

Coding Environment For WhizniumDBE-Backed Development

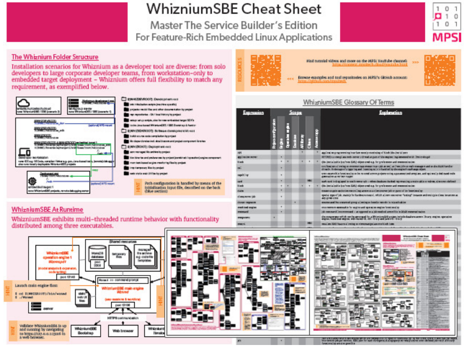
Code generation and host library builds take place within the Whiznium folders. Device-level code is edited in the vendors' IDE's, as exemplified below.

WhizniumDBE Glossary Of Terms

WhizniumDBE At Runtime

WhizniumDBE exhibits multi-threaded runtime behavior with functionality distributed among three executables.

The WhizniumDBE Master Database



WhizniumSBE Cheat Sheet

Master The Service Builder's Edition

For Feature-Rich Embedded Linux Applications

The Whiznium Folder Structure

Installation scenarios for Whiznium as a developer tool are diverse: from solo developers to large corporate developer teams, from workstation-only to embedded target deployment – Whiznium offers full flexibility to match any requirement, as exemplified below.

WhizniumSBE Glossary Of Terms

WhizniumSBE At Runtime

WhizniumSBE exhibits multi-threaded runtime behavior with functionality distributed among three executables.

Thank You!

Don’t hesitate to reach out
aw@mpsitechnologies.com