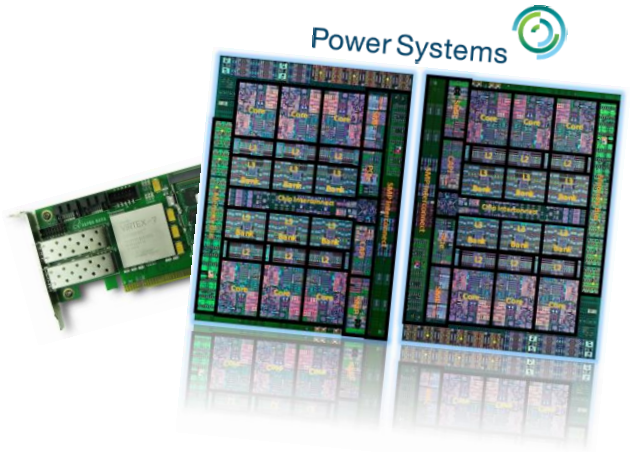
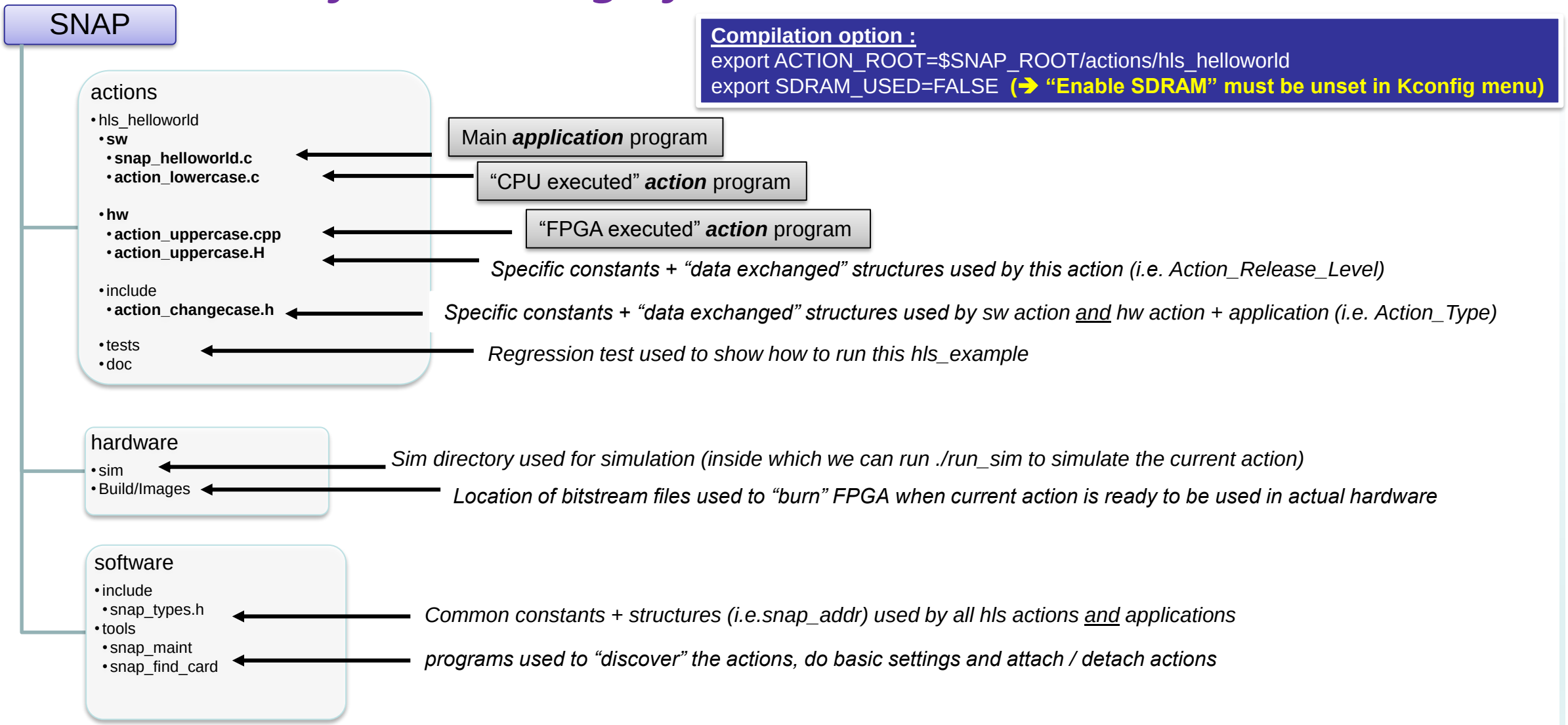
CAPI SNAP Education Series:
User Guide

CAPI SNAP Education
hls_helloworld : howto?
V2.2



Architecture of the SNAP git files



Action overview

Purpose: Providing to a 1st SNAP user a simple example to let him understand how different files work together.
Access to external interfaces are :

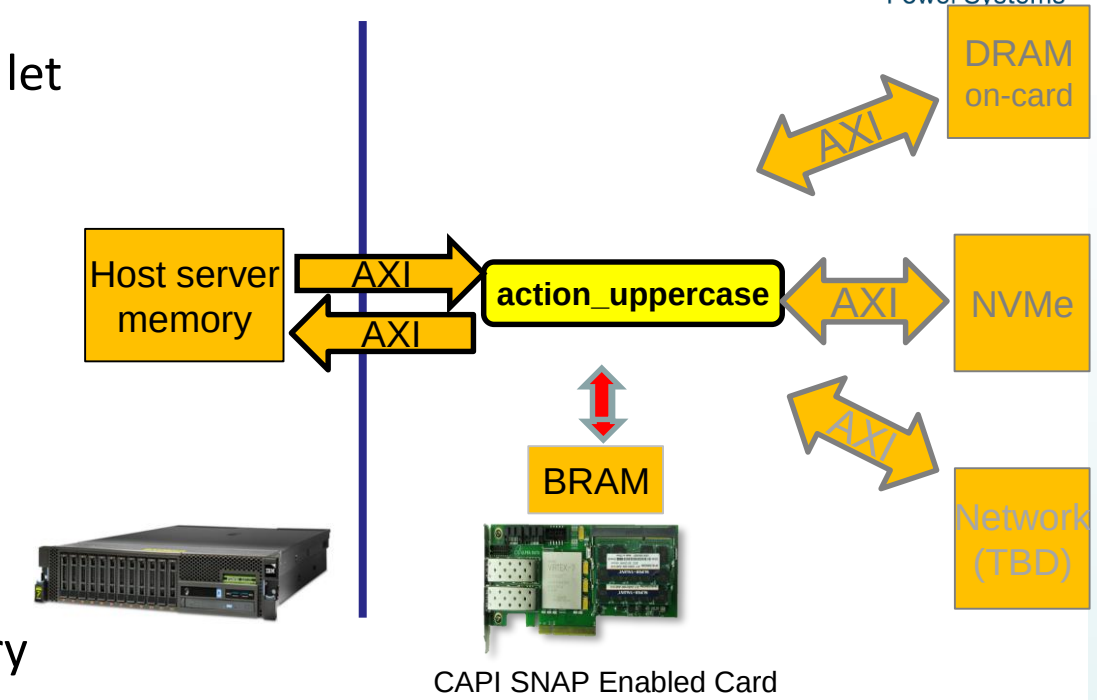
- Host memory server

When to use it:

- Understand Basic access

Memory management:

- Application is managing address of Host memory
- Data are read 64B words one after the other



Known limitations:

- HLS requires transfers to be 64 byte aligned and a size of multiples of 64 bytes
- DDR simulation model reads will return wrong values if non 64 bytes words or non initialized words are read (this is due to the simulation model only)

Action usage

Usage:

```
./snap_helloworld [-h] [-v, --verbose] [-V, --version]
        -C, --card <cardno> can be (0...3)
        -i, --input <file.bin>    input file.
        -o, --output <file.bin>   output file.
        -A, --type-in <CARD_DRAM, HOST_DRAM, ...>.
        -a, --addr-in <addr>      address e.g. in CARD_RAM.
        -D, --type-out <CARD_DRAM, HOST_DRAM, ...>.
        -d, --addr-out <addr>     address e.g. in CARD_RAM.
        -s, --size <size>         size of data.
        -t, --timeout             timeout in sec to wait for done.
        -X, --verify              verify result if possible
        -N, --no-irq              disable Interrupts
```

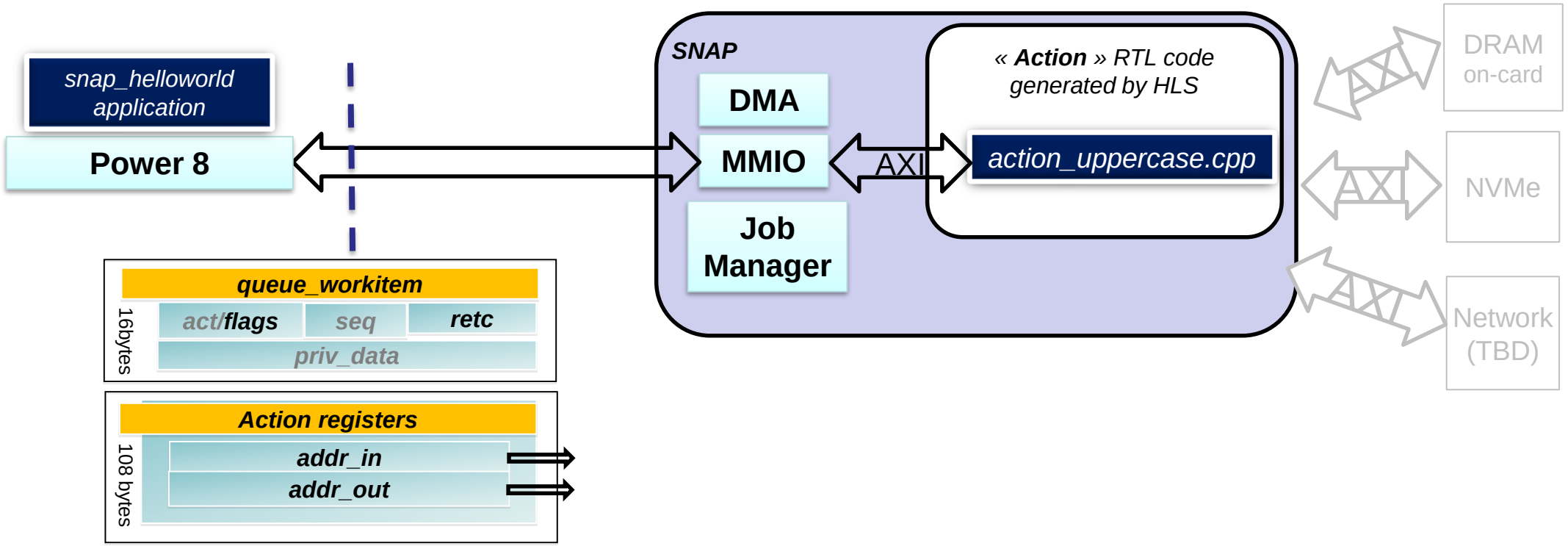
Options: *(default option in **bold**)*

SNAP_TRACE = 0x0 ➔ no debug trace
SNAP_TRACE = 0xF ➔ full debug trace
SNAP_CONFIG = FPGA ➔ hardware execution
SNAP_CONFIG = CPU ➔ software execution

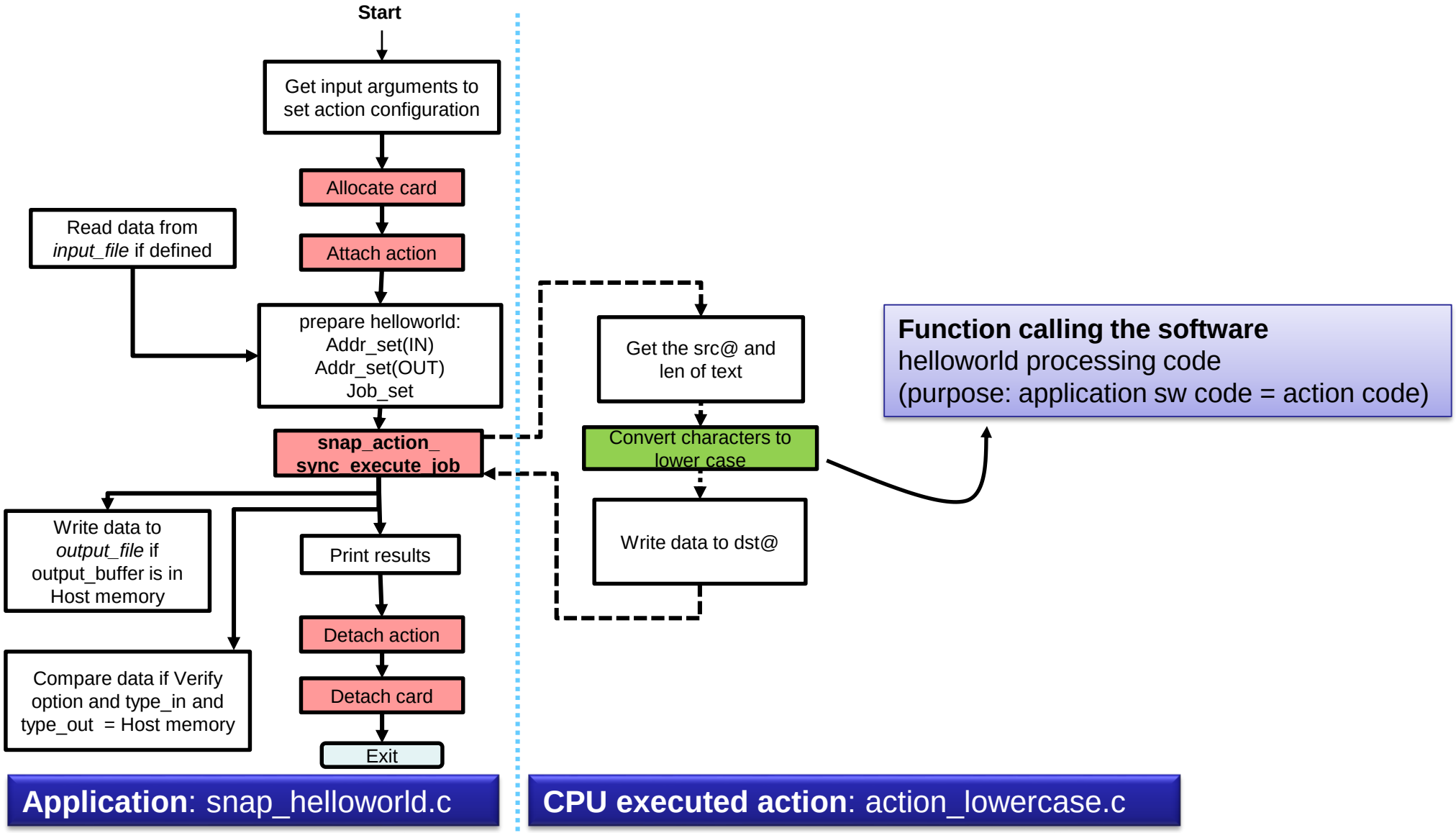
Example :

```
export SNAP_TRACE=0x0
snap_maint -vvv

rm /tmp/t2; rm /tmp/t3
vi /tmp/t1
     Hello world. This is my first CAPI SNAP experience. It's real fun!

$SNAP_CONFIG=FPGA snap_helloworld -i /tmp/t1 -o /tmp/t2
$SNAP_CONFIG=CPU  snap_helloworld -i /tmp/t1 -o /tmp/t3
echo "Display  input file"; cat /tmp/t1
 Hello world. This is my first CAPI SNAP experience. It's real fun!
echo "Display output file from FPGA EXECUTED ACTION"; cat /tmp/t2
 HELLO WORLD. THIS IS MY FIRST CAPI SNAP EXPERIENCE. IT'S REAL FUN!
echo "Display output file from CPU EXECUTED ACTION"; cat /tmp/t3
 hello world. this is my first capi snap experience. it's real fun!
```

helloworld registers



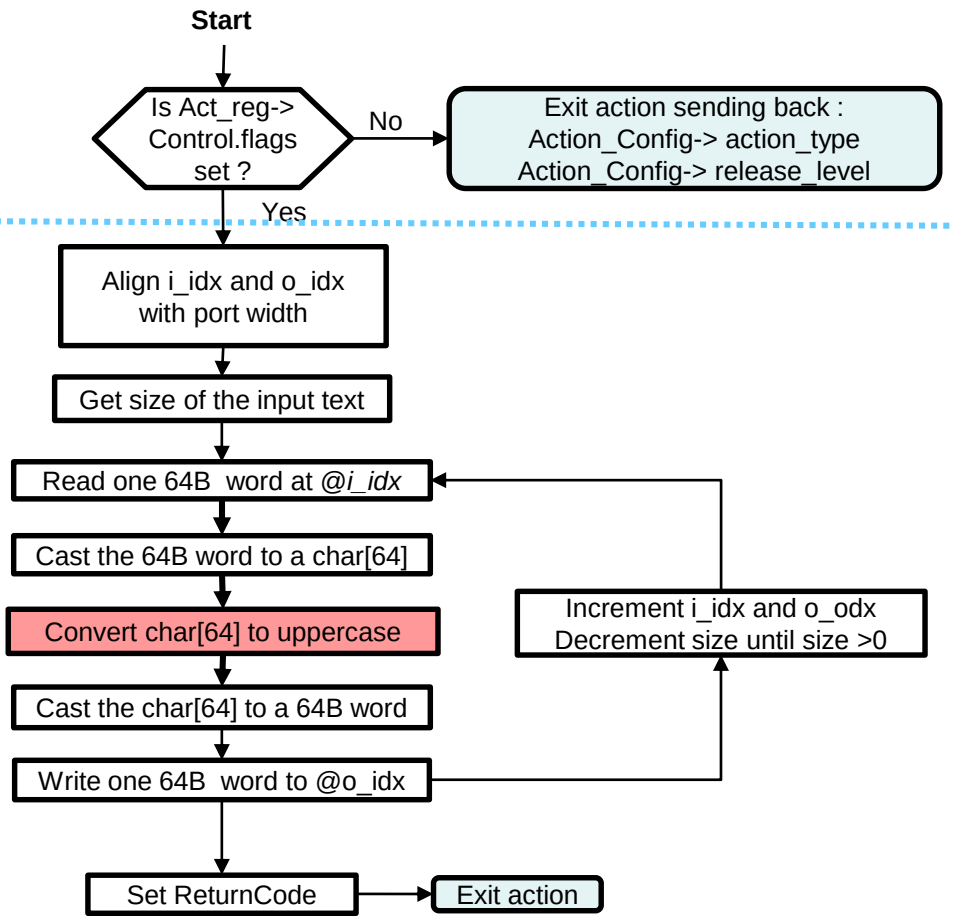
Application Code + software action code: what's in it?



Hardware action Code : what's in it?

hls_action

process_action



Used during discovery phase only

FPGA executed Action: action_uppercase.cpp

Constants - Ports

Constants: ➔ $ACTION_ROOT = snap/actions/hls_helloworld

Constant name	Value	Type	Definition location	Usage
HELLOWORLD_ACTION_TYPE	0x10141008	Fixed	$ACTION_ROOT/include/action_changecase.h	helloworld ID - list is in snap/ActionTypes.md
RELEASE_LEVEL	0x00000022	Variable	$ACTION_ROOT/hw/action_uppercase.**H**	release level – user defined

Ports used:

Ports name	Description	Enabled
din_gmem	Host memory data bus input Addr : 64bits - Data : 512bits	Yes
dout_gmem	Host memory data bus output Addr : 64bits - Data : 512bits	Yes
d_ddrmem	DDR3 - DDR4 data bus in/out Addr : 33bits - Data : 512bits	NOT used
nvme	NVMe data bus in/out Addr : 32bits - Data : 32bits	*No (soon)*

MMIO Registers

Read and Write are considered from the application / software side

act_reg.Control CONTROL	***This header is initialized by the SNAP job manager. The action will update the Return code and read the flags value. If the flags value is 0, then action sends only the action_RO_config_reg value and exit the action, otherwise it will process the action***									
Simu - WR	Write@	Read@	3		2	1	0	Typical Write value		Typical Read value	
0x3C40	0x100	0x180	sequence			**flags**	short action type	f001_01_00			
0x3C41	0x104	0x184	**Retc** (return code 0x102/0x104)					0		0x102 - 0x104	SUCCESS/FAILURE
0x3C42	0x108	0x188	Private Data					c0febabe			
0x3C43	0x10C	0x18C	Private Data					deadbeef			

action_reg.Data memcopy_job_t	***Action specific - user defined - need to stay in 108 Bytes This is the way for application and action to exchange information through this set of registers***									
	Write@	Read@	3		2	1	0	Typical Write value		Typical Read value	
0x3C44	0x110	0x190	snap_addr.**addr_in** (LSB)								
0x3C45	0x114	0x194	snap_addr.**addr_in** (MSB)								
0x3C46	0x118	0x198	snap_addr_in.**size**								
0x3C47	0x11C	0x19C	snap.addr_in.**flags** (SRC, DST, ...)			snap.addr_in.**type** (HOST, DRAM, NVME,..)					
0x3C48	0x120	0x1A0	snap_addr.**addr_out** (LSB)								
0x3C49	0x124	0x1A4	snap_addr.**addr_out** (MSB)								
0x3C4A	0x128	0x1A8	snap.addr_out.**size**								
0x3C4B	0x12C	0x1AC	snap.addr_out.**flags** (SRC, DST, ...)			snap.addr_out.**type** (HOST, DRAM, NVME,..)					

$ACTION_ROOT/hw/action_uppercase.H

```
typedef struct {
    CONTROL Control;       /*  16 bytes */
    helloworld_job_t Data;  /* 108 bytes */
    uint8_t padding[SNAP_HLS_JOBSIZE - sizeof(helloworld_job_t)];
} action_reg;
```

$SNAP_ROOT/actions/include/hls_snap.H

```
typedef struct {
    snapu8_t sat; // short action type
    snapu8_t flags;
    snapu16_t seq;
    snapu32_t Retc;
    snapu64_t Reserved; // Priv_data
} CONTROL;
```

$ACTION_ROOT/include/action_changecase.h

```
typedef struct helloworld_job {
    struct snap_addr in;    /* input data */
    struct snap_addr out;   /* offset table */
} helloworld_job_t;
```

$SNAP_ROOT/software/include/snap_types.h

```
typedef struct snap_addr {
    uint64_t addr;
    uint32_t size;
    snap_addrtype_t type;        /* DRAM, NVME, ... */
    snap_addrflag_t flags;       /* SRC, DST, EXT, ... */
} snap_addr_t;
```

Path of improvements

History of this document and of the action release level

V2.0: initial document
V2.1: new files directory structure applied
V2.2: simplified the code removing the circumvention of issue #320