

MICROCHIP POLARFIRE SOC

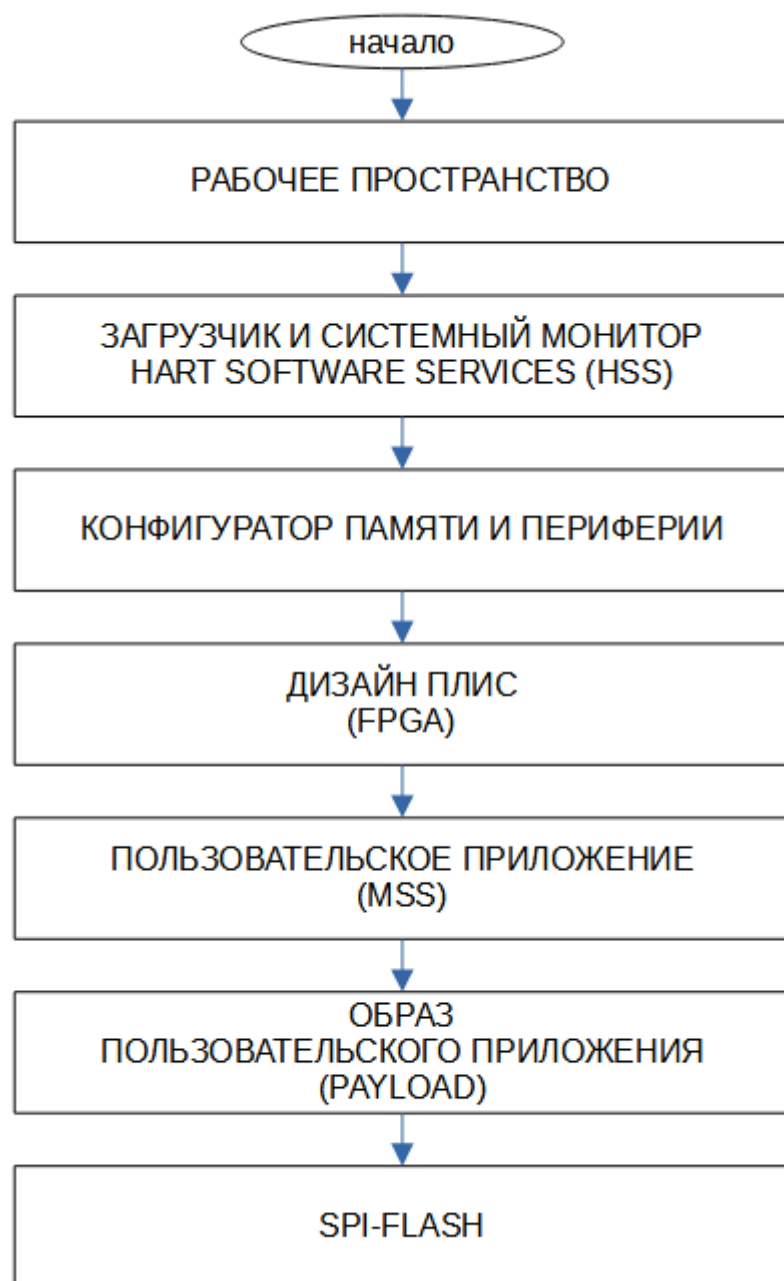
ПРОЦЕСС РАЗРАБОТКИ

2023-09-25

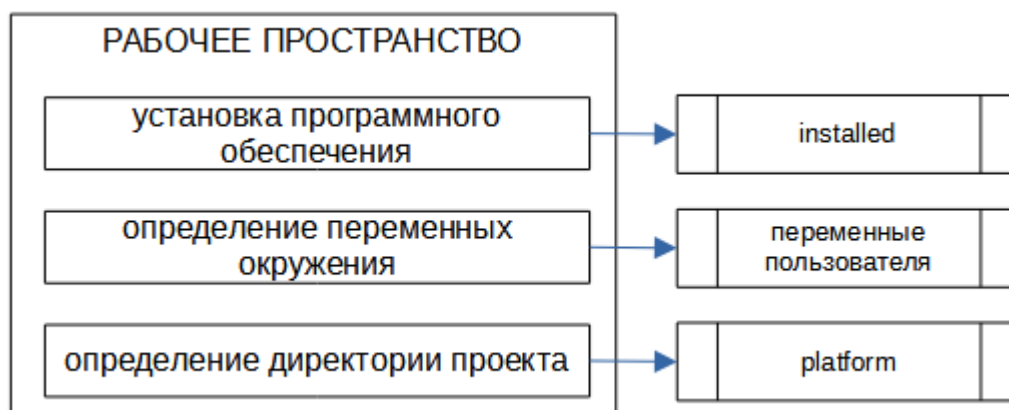
ОГЛАВЛЕНИЕ

1	ЭТАПЫ РАЗРАБОТКИ	3
2	РАБОЧЕЕ ПРОСТРАНСТВО	4
2.1	Программное обеспечение	4
2.2	Переменные среды пользователя	4
2.3	Директория проекта	5
2.4	Файловая структура проекта.....	5
2.4.1	application	6
2.4.2	board.....	7
2.4.3	middleware	8
2.4.4	platform	9
3	ЗАГРУЗЧИК И СИСТЕМНЫЙ МОНИТОР.....	11
3.1	Конфигурирование.....	12
3.2	Сборка	14
3.3	Загрузка в eNVM	14
3.4	Сервис TinyCLI.....	15
4	КОНФИГУРАТОР ПАМЯТИ И ПЕРИФЕРИИ	17
4.1	Запуск	17
4.2	Конфигурирование.....	18
4.3	Сохранение	18
4.4	Генерирование	18
4.5	Интеграция в проект пользовательского приложения	19
5	ДИЗАЙН ПЛИС (FPGA)	20
5.1	Генерирование эталонного дизайна	20
6	ПОЛЬЗОВАТЕЛЬСКОЕ ПРИЛОЖЕНИЕ (MSS)	22
6.1	Настройка сборки.....	22
6.2	Очистка сборки.....	23
6.3	Сборка	23
7	ОБРАЗ ПОЛЬЗОВАТЕЛЬСКОГО ПРИЛОЖЕНИЯ (PAYLOAD).....	24
7.1	Настройка рабочего пространства.....	24
7.2	Конфигурирование.....	25
7.3	Сборка	25
7.4	Конвертация в hex	26
8	SPI-FLASH	27
8.1	Настройка памяти.....	28
8.2	Добавление образа	29
8.3	Запись	30
	БАЗА ЗНАНИЙ.....	31

1 ЭТАПЫ РАЗРАБОТКИ



2 РАБОЧЕЕ ПРОСТРАНСТВО



2.1 Программное обеспечение

1) Скачать и установить следующее программное обеспечение

Libero_SoC_v2022.2

SoftConsole-v2022.2-RISC-V-747

Python 3.10

+ pip

+ pip install windows-curses

2.2 Переменные среды пользователя

1) Определить переменные среды пользователя

Свойства системы / Переменные среды / Переменные среды пользователя

Path

D:\MPFS\SoftConsole-v2022.2-RISC-V-747\python3

путь к Python 3 (комплект SoftConsole)

D:\MPFS\Soft\SoftConsole-v2022.2-RISC-V-747\build_tools\bin

путь к инструментам сборки (комплект SoftConsole)

D:\MPFS\SoftConsole-v2022.2-RISC-V-747\riscv-unknown-elf-gcc\bin

путь к компилятору GCC RISC-V (комплект SoftConsole)

D:\MPFS\Libero_SoC_v2022.2\Designer\bin64

путь к Дизайнеру (комплект Libero)

SC_INSTALL_DIR
D:\MPFS\SoftConsole-v2022.2-RISC-V-747 <i>путь к SoftConsole</i>
JAVA_BINARY
D:\MPFS\SoftConsole-v2022.2-RISC-V-747\eclipse\jre\bin\java.exe <i>путь к Java (комплект SoftConsole)</i>
JAVA_HOME
D:\MPFS\SoftConsole-v2022.2-RISC-V-747\eclipse\jre/ <i>путь к Java Runtime (комплект SoftConsole)</i>
PYTHONPATH
D:\MPFS\SoftConsole-v2022.2-RISC-V-747\python3 <i>путь к Python (комплект SoftConsole)</i>
MACRO_PYTHON_BINARY_EXECUTABLE
python.exe <i>имя исполняемого файла Python (комплект SoftConsole)</i>
MACRO_PYTHON_BINARY_PATH
D:\MPFS\SoftConsole-v2022.2-RISC-V-747\python3 <i>путь к Python (комплект SoftConsole)</i>
MACRO_PYTHON_BINARY_PATH_AND_EXECUTABLE
D:\MPFS\SoftConsole-v2022.2-RISC-V-747\python3\python.exe <i>путь к Python (комплект SoftConsole)</i>

2.3 Директория проекта

- 1) Создать директорию рабочего пространства (например, polarfire / platform)

```
mkdir D:\projects\polarfire\platform
```

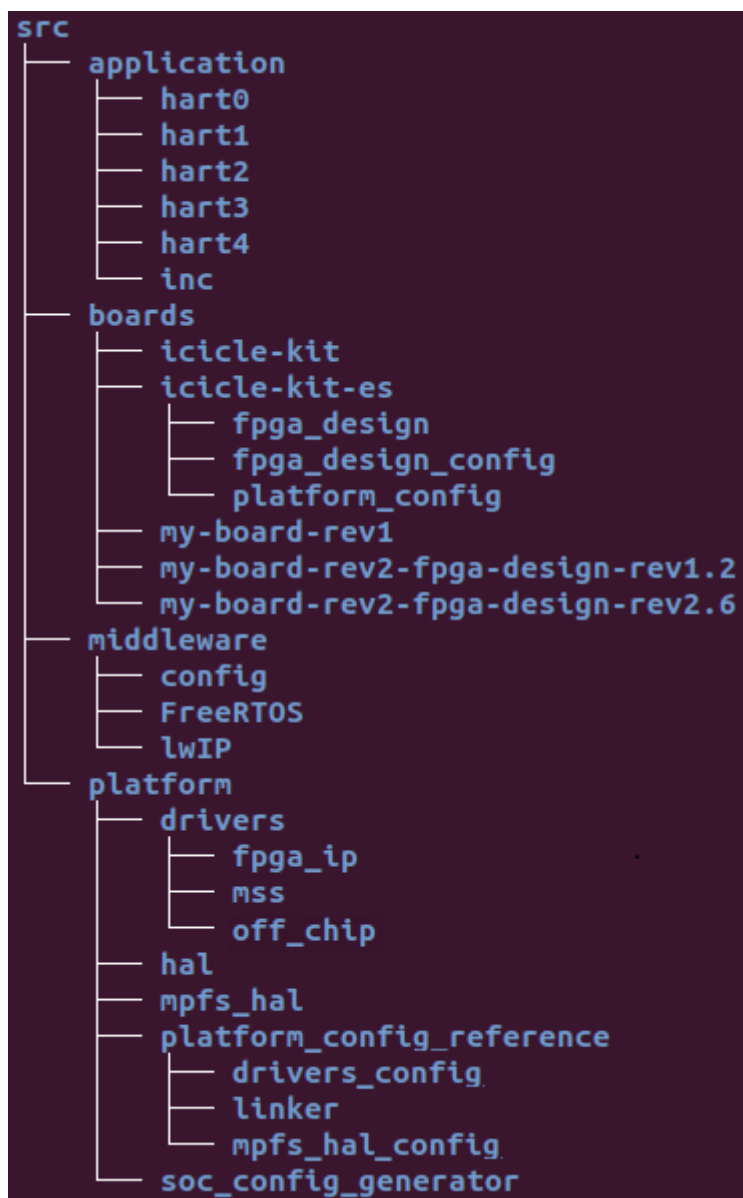
2.4 Файловая структура проекта

- 1) Сформировать файловую структуру проекта

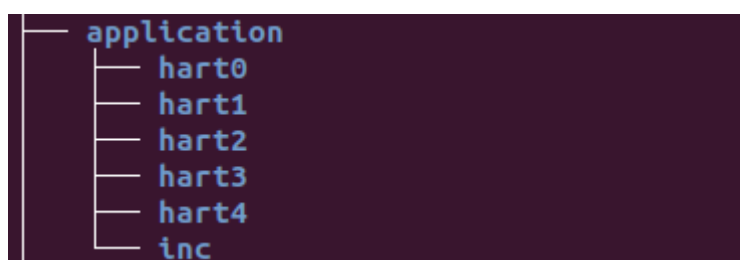
вручную

или

взять в качестве шаблона один из [демонстрационных проектов](#)



2.4.1 application



Нижний уровень абстракции (HAL).

Базовый код пользовательского приложения (MSS Bare Metal) конкретного ядра. Допускается изменение содержимого.

- **hart0**

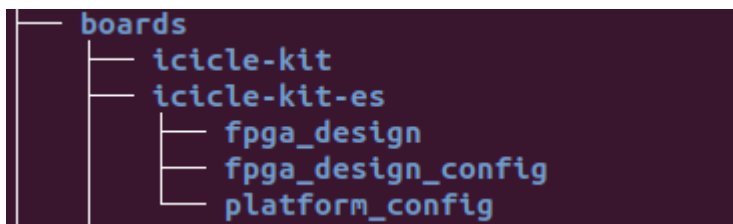
Пользовательский код для Ядра 0 (e51) [*.c *.h]

- **hart1**
Пользовательский код для Ядра 1 (u54_1) [*.c *.h]
- **hart2**
Пользовательский код для Ядра 2 (u54_2) [*.c *.h]
- **hart3**
Пользовательский код для Ядра 3 (u54_3) [*.c *.h]
- **hart4**
Пользовательский код для Ядра 4 (u54_4) [*.c *.h]
- **inc**
Общие заголовочные файлы (для всех ядер) [*.h]

Для соблюдения принципов «Hardware Abstraction Layer», разделять код на:

- аппаратно-зависимый (работа с периферией конкретной платформы)
- аппаратно-независимый

2.4.2 board



Нижний уровень абстракции (HAL).

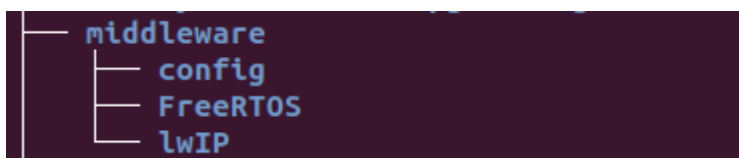
Конфигурации, линковщики и заголовочные файлы для конкретной аппаратной платформы (например, icicle-kit-es).

Допускается изменение содержимого (кроме автогенерируемого).

- **fpga_design**
Конфигурация FPGA и MSS
 - **libero_tcl**
Копия или ссылка на TCL-скрипт Libero
(не используется при сборке пользовательского приложения).
 - **mss_configuration**
Конфигурация (CFG), загружаемая в программу MSS-Configurator
(не используется при сборке пользовательского приложения).

- design_description
Конфигурация (XML), сгенерированная программой MSS-Configurator
(используется при сборке пользовательского приложения на этапе «pre-build» - генерирует содержимое директории *fpga_design_config*).
- **fpga_design_config**
Заголовочные файлы [**.h*] с описанием конфигурации периферии для конкретной аппаратной платформы, автоматически генерируемые при сборке пользовательского приложения на этапе «pre-build» (основа - это конфигурация из *design_description/*.xml*). Эти заголовочные файлы используются *platform/mss_hal*.
 - clocks
 - ddr
 - general
 - io
 - memory_map
 - sgmi
 - fpga_design_config.h
(included in *platform/mpfs_hal/mss_hal.h*)
- **platform_config**
Заголовочные файлы с описанием конфигурации драйверов и HAL (например, определение рабочих аппаратных ядер и загрузчика).
 - drivers_config
Заголовочные файлы [**.h*] с описанием конфигурации драйверов, реализованных в *platform/drivers*.
 - linker
Файлы линковщика [**.ld*].
 - mpfs_hal_config
Заголовочные файлы [**.h*] с описанием конфигурации MSS HAL.

2.4.3 middleware



Средний уровень абстракции (HAL).

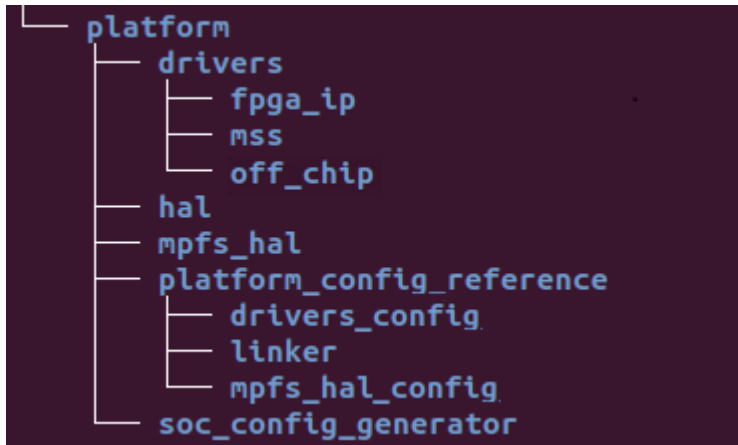
Код (обычно аппаратно-независимый):

- RTOS/ОСРВ (ядро и его компоненты),

- стек протоколов (например, TCP/IP, ModBus RTU/TCP),
- прочий пользовательский код (например, описание задач, работающих в режиме ядра ОСРВ).

Допускается изменение содержимого (кроме стороннего кода).

2.4.4 platform



Нижний уровень абстракции (HAL).

Код библиотеки HAL, код для работы с аппаратной платформой (драйвера периферии), а также код для запуска пользовательской программы (startup-code).

Не допускается изменение содержимого.

- **drivers**

Код драйверов периферии

- fpga_ip
Периферия ядер FPGA
- mss
Периферия MSS
- off_chp
Внешняя периферия
(например, подключаемые датчики, внешняя память и т.п.)

- **hal**

Код библиотеки HAL (универсальная часть)

(позволяет использовать один код для различных аппаратных платформ)

- **mpfs_hal**

Код библиотеки HAL (специфичная для Microchip Polarfire SoC часть)

(включая код для запуска пользовательского приложения – startup-code)

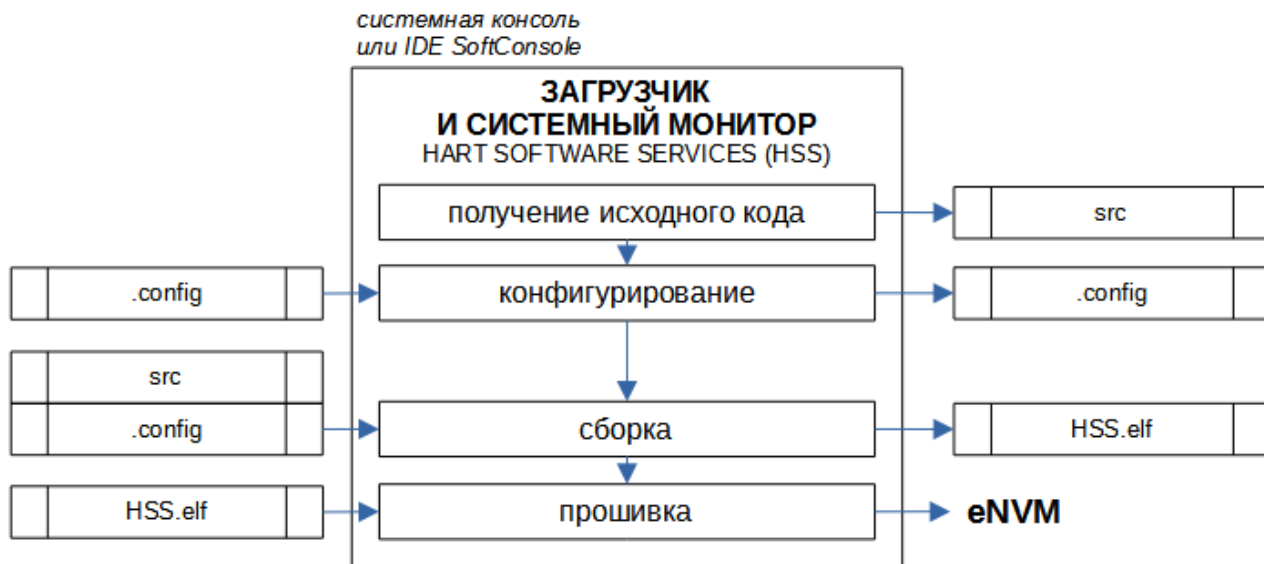
- **platform_config_reference**

Исходная и иные версия platform_config
(шаблоны линковщиков и конфигураций)

- **soc_config_generator**

Python-скрипт для автоматической генерации содержимого директории boards/fpga_design_config при сборке пользовательского приложения на этапе «pre-build».

3 ЗАГРУЗЧИК И СИСТЕМНЫЙ МОНИТОР



HART SOFTWARE SERVICE (HSS)

Готовый программный продукт с [открытым исходным кодом](#).

Хранится (в сжатом виде) во встроенной flash-памяти eNVM (используется режим Boot Mode 1-Direct Boot from eNVM)

При подаче питания

- код HSS выгружается из eNVM в L2-scratchpad

Работает на ядре E51. Ядра U54 свободны для пользовательских приложений.

Включает в себя:

- 0-stage bootloader
 - запускает (стартует) пользовательское приложение, предварительно выгрузив его из SPI-FLASH|eMMC|SD в DDR|LIM
- superloop-monitor
 - получает и обрабатывает запросы от приложений пользовательских ядер U54
 - получает и обрабатывает запросы от внешних приложений через последовательный интерфейс UART0 (115200 8 N 1, если HSS был собран с поддержкой TinyCLI)
- machine-mode software interrupt trap handler
 - позволяет формировать программные прерывания для ядер U54 с передачей им полезных данных

Пользовательское приложение

- представляет собой специально оформленный образ (payload), включающий Bare Metal приложение или загрузчик OCPB или ОС (например, U-Boot)
- хранится во внешней SPI-FLASH|eMMC|SD
- перед запуском выгружается из хранимой памяти в DDR|LIM

3.1 Конфигурирование

1) Перейти в директорию исходного кода

```
cd D:\projects\polarfire\hart-software-services
```

2) Сформировать файл конфигурации одним из следующих способов:

а) использовать готовый шаблон конфигурации (например, для ICICLE-KIT):

из директории шаблонов

```
copy boards\mpfs-icicle-kit-es\def_config .config
```

из заранее подготовленного файла (см. ниже)

```
copy .config-icicle-kit .config
```

б) использовать программу-конфигуратор (псевдографический интерфейс):

со значениями по-умолчанию

```
make menuconfig
```

со значениями из шаблонной конфигурации

```
make BOARD=mpfs-icicle-kit-es menuconfig
```

где, BOARD – наименование системы (платы)

указание BOARD= mpfs-icicle-kit-es равносильно команде

```
copy boards\mpfs-icicle-kit-es\def_config .config
```

```
(Top)
Build Options
Board/Design Configuration Options --->
Services --->
General Configuration Options --->
Build Options --->
Compression --->
Crypto --->
Debug Options --->
SSMB Options --->

[Space/Enter] Toggle/enter  [ESC] Leave menu          [S] Save
[O] Load                   [?] Symbol info           [/] Jump to symbol
[F] Toggle show-help mode  [C] Toggle show-name mode [A] Toggle show-all mode
[Q] Quit (prompts for save) [D] Save minimal config (advanced)
```

Для работы Конфигуратора требуются:

+ *python3*

+ *pip*

+ *pip install windows-curses*

Краткий обзор .config-icicle-kit

```
(Top) → Board/Design Configuration Options → Icicle-Kit Design Configuration Options
Build Options
(boards/mpfs-icicle-kit-es/soc_fpga_design/xml/ICICLE_MSS_mss_cfg.xml) Enter path to Libero XML file
```

```
(Top) → Services → Boot Service
Build Options
(0x400) Copy payload from SPI FLASH at offset (NEW)
[ ] Use Payload file
[ ] Use Custom Boot Flow
(0x103FC0000) Target Base address for BOOT to DDR copy
```

```
(Top) → Services
Build Options
[*] Bus Error Unit support
[*] Enable booting
    Boot Service --->
[*] GOTO support
[*] Health Monitoring support
[*] IPI Polling support
[ ] MMC support
[*] OpenSBI IHC ecall support
[*] Remote Proc ecall service support
    SBI Extension Support --->
[ ] Low Power Mode support
[ ] Boot from QSPI NAND flash support
[*] RAM Scrubbing support
    RAM Scrubbing Service --->
[*] SGDMA support
[*] Boot from System Controller SPI flash support
[*] TinyCLI support
    Tiny Command Line Interface --->
[ ] UART support
[*] Virtual Watchdog support
    Watchdog Service --->
[ ] YMODEM support
```

```
(Top) → Build Options
Build Options
[ ] Color log output
[ ] Display Logo
    Logo ----
[ ] Enable strong stack protection
[ ] Dump stack sizes
[ ] Use RISC-V Relaxing and the Global Pointer, GP
[ ] Enable make dependencies in build system
[ ] Enable GNU Build ID in image
[*] Use inttypes.h
[ ] Display Compiler and Linker tool version information at startup.
[ ] Display function names in console log messages
```

```
(Top) → Compression
Build Options
[*] Compression support
[*] miniz (DEFLATE)
```

3.2 Сборка

1) Перейти в директорию проекта

```
cd D:\projects\polarfire\hart-software-services
```

2) Очистить директорию проекта от файлов предыдущей сборки

```
make BOARD=mpfs-icicle-kit-es clean
```

3) Запустить сборку

```
make BOARD=mpfs-icicle-kit-es
```

4) Дождаться завершения процесса сборки

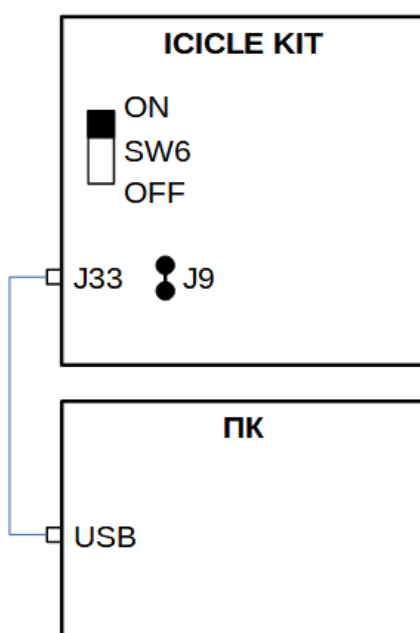
```
16:09:53 INFO - Selected boot mode "1 - non-secure boot from eNVM" and working in directory "platform\hart-software-services\Default".
16:09:53 INFO - Generating BIN file...
16:09:54 INFO - Generating header...
16:09:54 INFO - Generating HEX file...
16:09:54 INFO - Preparing for bitstream generation...
16:09:54 INFO - Generating bitstream...
16:10:03 INFO - Programming/verifying the target skipped because --dryrun was specified.
16:10:03 INFO - mpfsBootmodeProgrammer completed successfully.
text    data    bss     dec     hex filename
78964    416    149632  229012  37e94 Default/hss-envm-wrapper.elf
```

3.3 Загрузка в eNVM

1) Включить встроенный в ICICLE KIT программатор FLASH PRO6, установив переключку J9

2) Установить связь между программатором FLASH PRO6 и ПК посредством кабеля microUSB-USB

3) Включить ICICLE KIT, переведя переключатель SW6 в положение ON



4) Перейти в директорию проекта

```
cd D:\projects\polarfire\hart-software-services
```

5) Запустить загрузку

```
make BOARD=mpfs-icicle-kit-es program
```

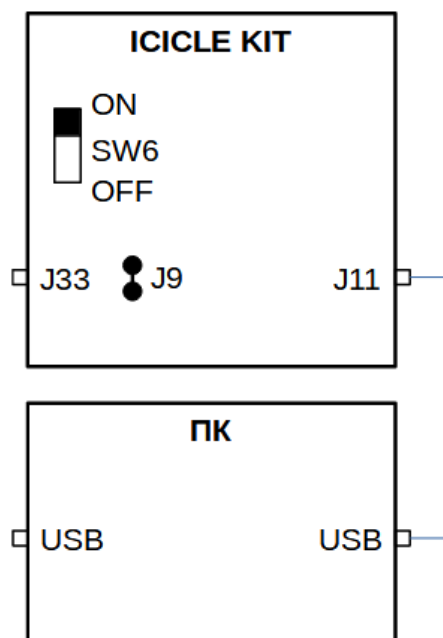
6) Дождаться завершения процесса загрузки

```
16:13:43 INFO - mpfsBootmodeProgrammer completed successfully.
```

3.4 Сервис TinyCLI

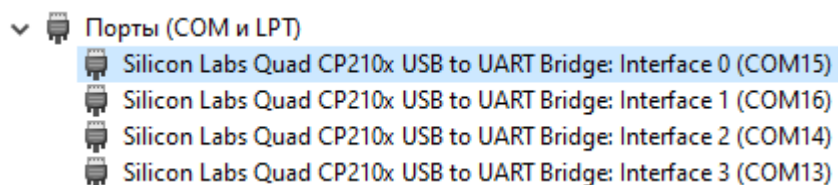
1) Установить связь между ICICLE KIT и ПК посредством кабеля microUSB-USB

2) Включить ICICLE KIT, переведя переключатель SW6 в положение ON



3) Запустить консоль (Putty, Terminal и т.п.) с настройками:

COM = соответствует Interface 0 (см. Диспетчер устройств)



Speed = 115200

Data bits = 8

Stop bits = 1

Parity = None

Flow control = None

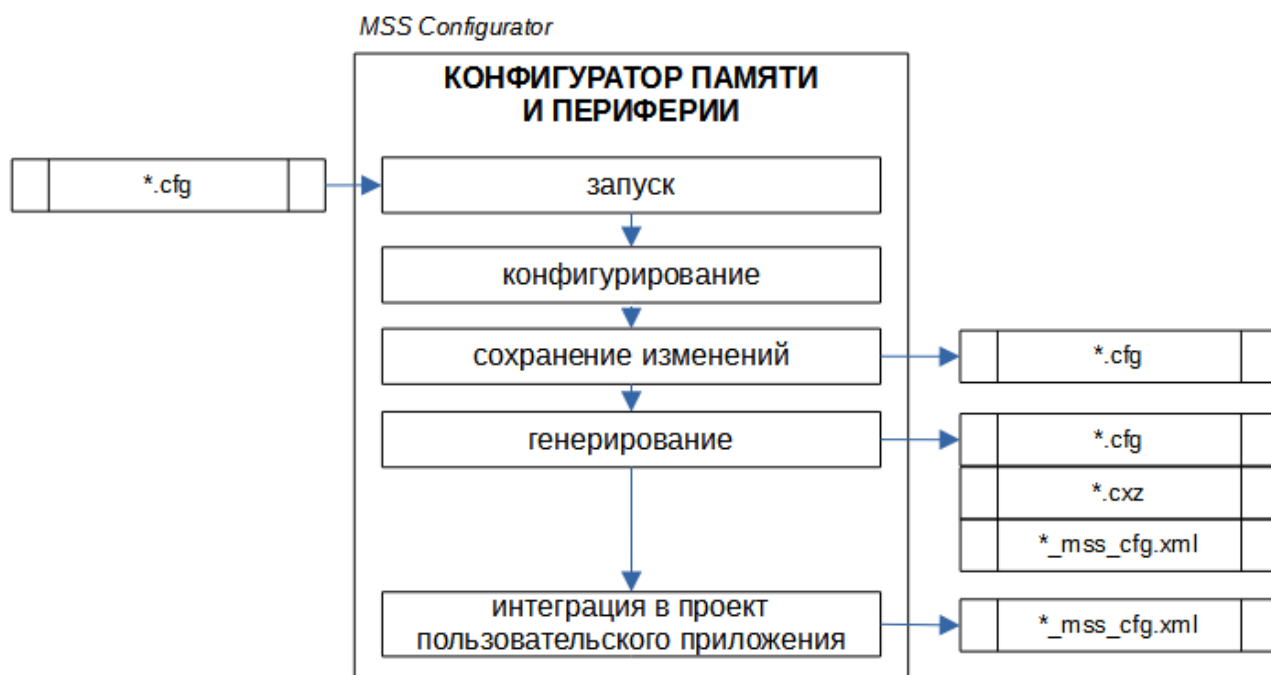
- 4) В консоли дождаться подключения и далее нажать клавишу ВВОД (будет выведена строка приглашения для ввода команд)

```
>> █
```

- 5) Команда **HELP** выводит список поддерживаемых команд

```
>> HELP  
Supported commands:  
QUIT BOOT RESET HELP VERSION UPTIME DEBUG MEMTEST QSPI EMMC MMC SDCARD PAYLOAD SPI SCRUB
```


4 КОНФИГУРАТОР ПАМЯТИ И ПЕРИФЕРИИ



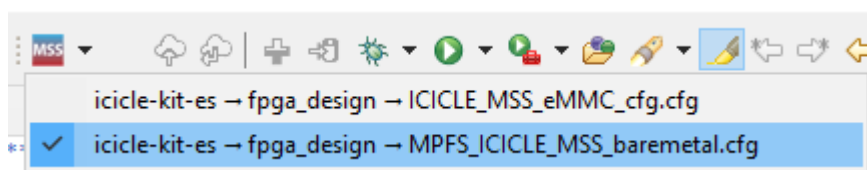
MSS Configurator (Polarfire SoC Configuration Generator)


Готовый программный продукт, интегрированный в SoftConsole и Libero (2022).

Предоставляет графический интерфейс для конфигурирования периферии и памяти Polarfire SoC.

4.1 Запуск

Из SoftConsole:



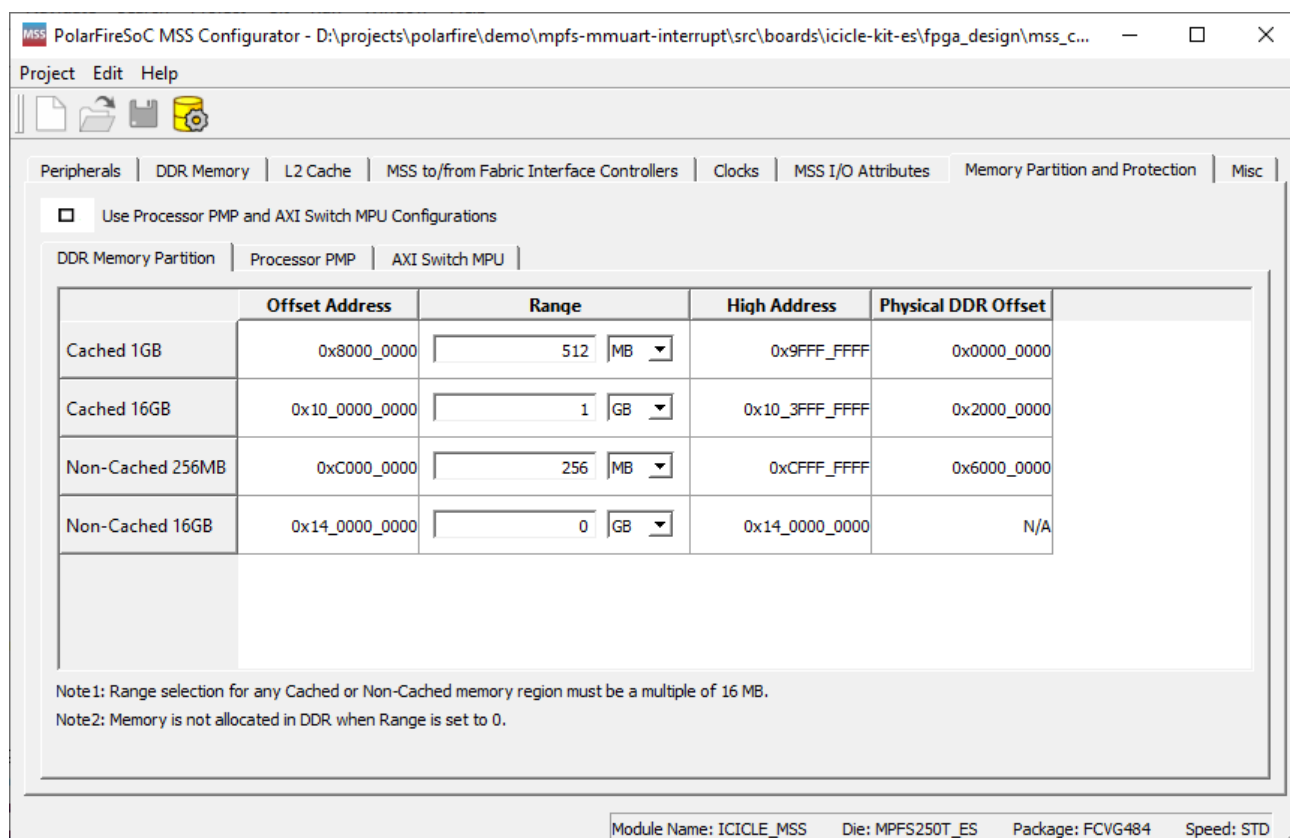
По-умолчанию, загружается конфигурация из проекта пользовательского приложения (*.cfg), но далее можно выбрать другую конфигурацию самостоятельно (Project / Open ).

*.cfg – исходная конфигурация:


`boards/icicle-kit-es/fpga_design/mss_configuration/MPFS_ICICLE_MSS_baremetal.cfg`

4.2 Конфигурирование

Пример окна конфигуратора:




4.3 Сохранение

По-умолчанию, при сохранении () перезаписывается загруженная конфигурация, но можно сохранить конфигурацию под иным, заданным вручную, именем (Project / Save As).

*.cfg – измененная конфигурация:

boards/icicle-kit-es/fpga_design/mss_configuration/MPFS_ICICLE_MSS_baremetal.cfg

4.4 Генерирование

Генерирование () формирует и сохраняет конфигурацию в нескольких форматах (пакет генерации):

boards/_generated/current/ICICLE_MSS.cfg

boards/_generated/current/ICICLE_MSS.cxz

boards/_generated/current/ICICLE_MSS_mss_cfg.xml

Пакет генерации, по-умолчанию, сохраняется в директории boards/_generated/current, но можно выбрать иное место.

Конфигурация в формате CXZ используется в проекте Дизайна ПЛИС.

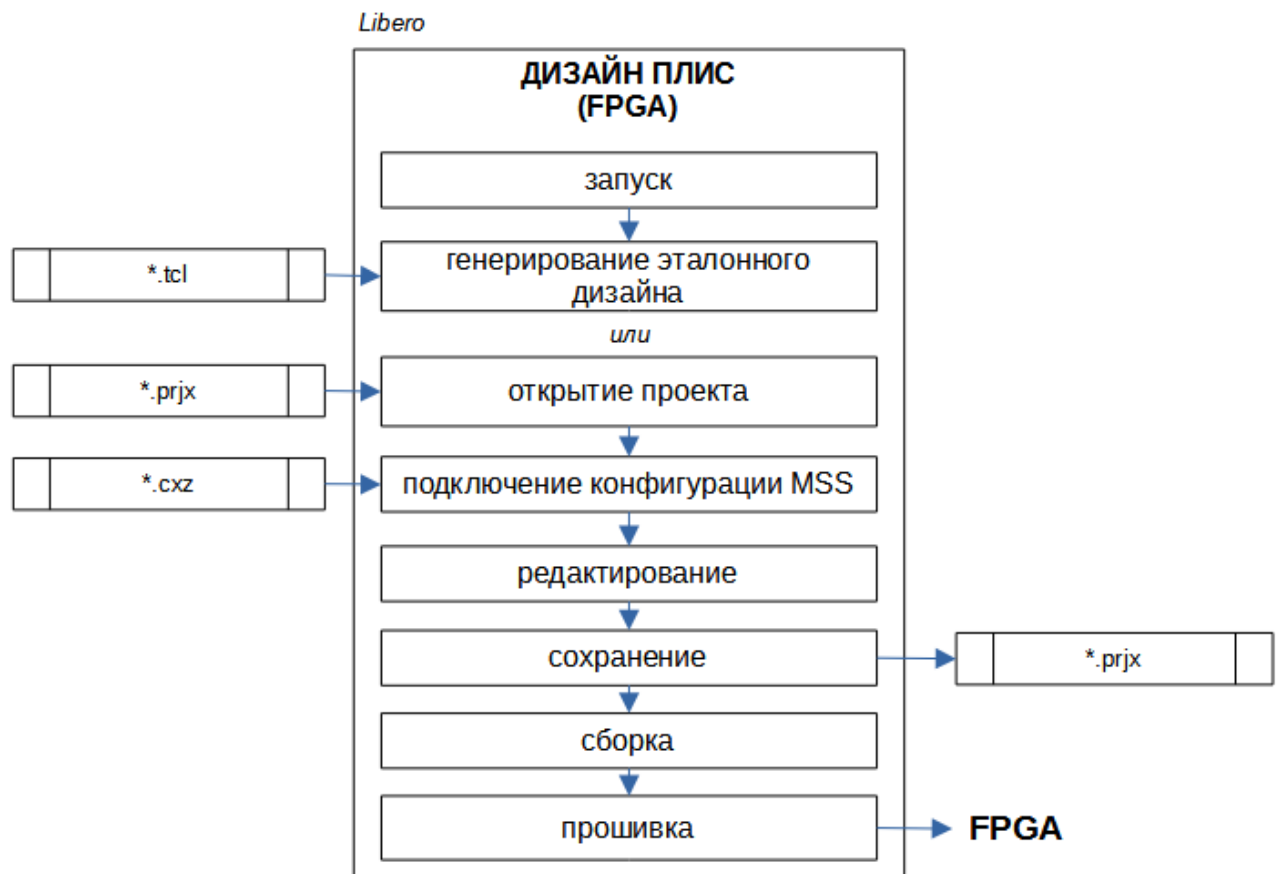
4.5 Интеграция в проект пользовательского приложения

Полученный на этапе генерации XML-файл необходимо *вручную* скопировать в одну из директорий проекта пользовательского приложения:

```
copy boards/_generated/current/ICICLE_MSS_mss_cfg.xml  
boards/icle-kit-es/fpga_design/design_description/ICICLE_MSS_mss_cfg.xml
```

Конфигурация в формате XML используется при сборке (этап «pre-build») пользовательского приложения.

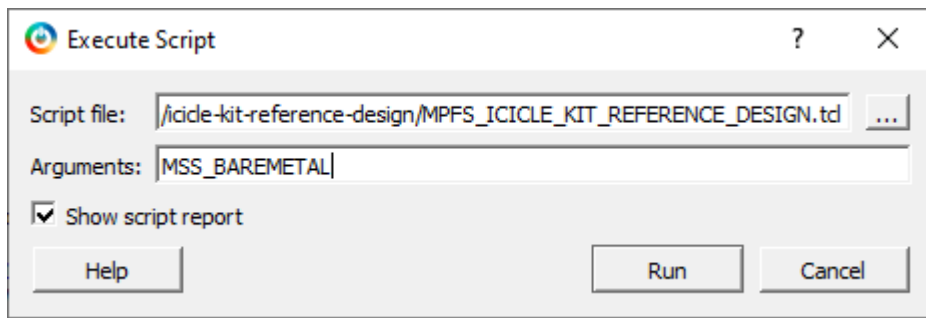
5 ДИЗАЙН ПЛИС (FPGA)



Libero (2022).

5.1 Генерирование эталонного дизайна

- 1) Перейти в директорию рабочего пространства
`cd D:\projects\polarfire\platform`
- 2) Получить исходники эталонного дизайна (например, для [ICICLE-KIT](#))
- 3) Запустить Libero (2022)
* при необходимости задать настройки Proxy
`Project / Preferences / Proxy`
- 4) Выбрать скрипт генерации эталонного дизайна:
`Project / Execute Script`



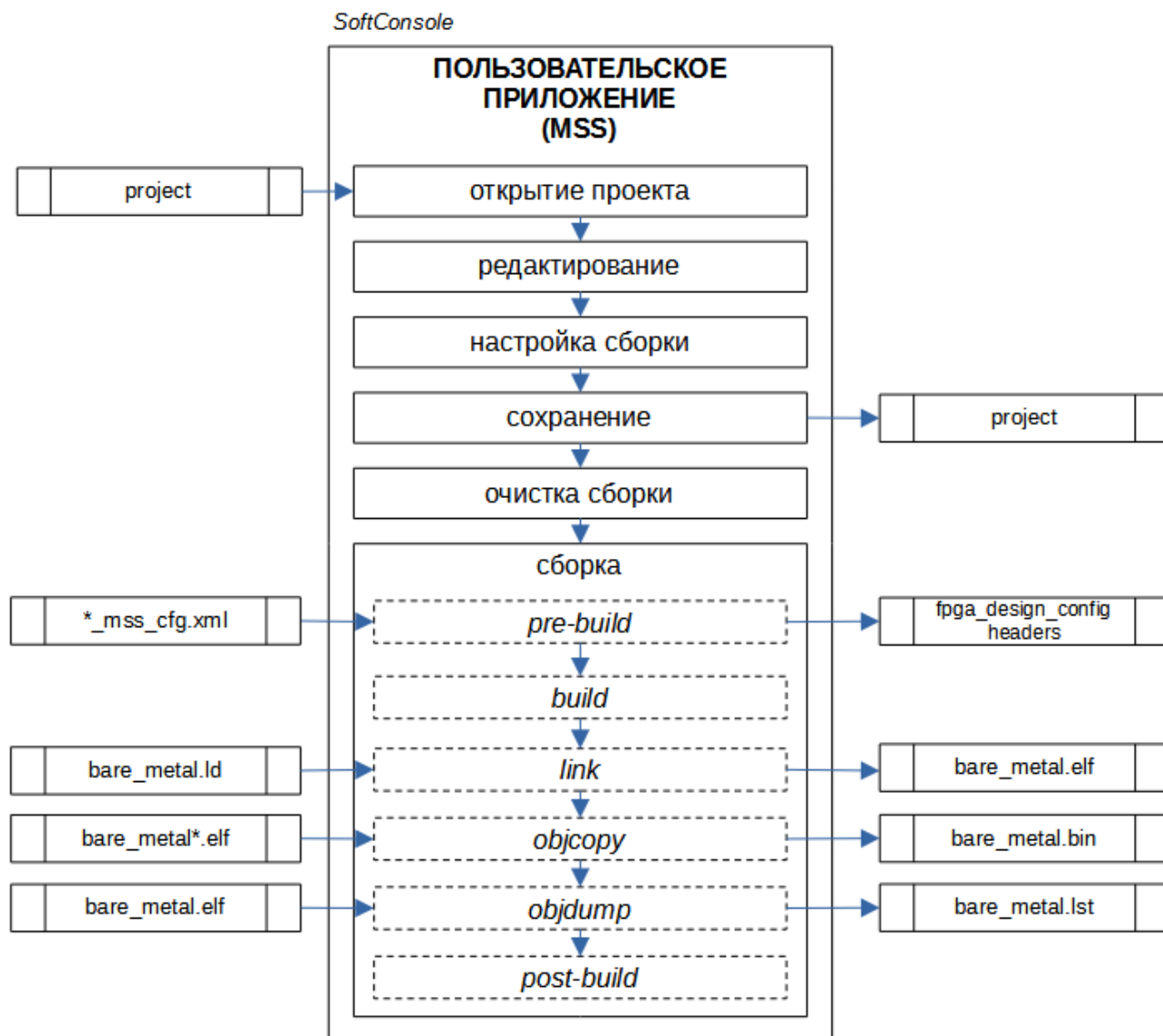
* аргумент `MSS_BAREMETAL` генерирует конфигурацию MSS с поддержкой проектов Bare Metal – без дополнительного функционала, ОЗУ DDR 1 ГБ.

5) Генерировать дизайн, нажав на кнопку Run

* в процессе генерации дизайна будет выполнена автоматическая загрузка всех необходимых библиотек (ядра), поэтому на рабочем ПК должен быть обеспечен доступ к сети Интернет

* дождаться завершения процесса загрузки

6 ПОЛЬЗОВАТЕЛЬСКОЕ ПРИЛОЖЕНИЕ (MSS)



SoftConsole (2022).

6.1 Настройка сборки

Project / Properties / C/C++ Build / Settings / Build Steps

Pre-build Command:

```
${env_var:MACRO_PYTHON_BINARY_PATH_AND_EXECUTABLE}
../src/platform/soc_config_generator/mpfs_configuration_generator.py
../src/boards/icicle-kit-es/fpga_design/design_description/
../src/boards/icicle-kit-es
```

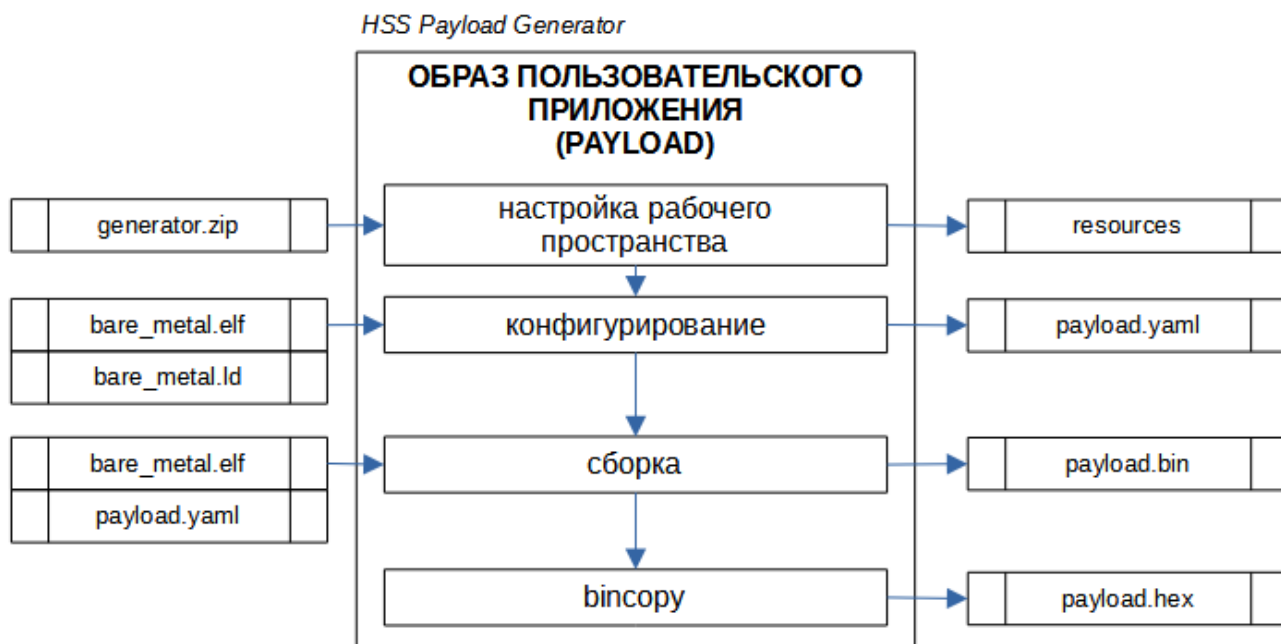
6.2 Очистка сборки

Project / Clean...

6.3 Сборка

Project / Build All

7 ОБРАЗ ПОЛЬЗОВАТЕЛЬСКОГО ПРИЛОЖЕНИЯ (PAYLOAD)



HSS PAYLOAD GENERATOR

Готовый программный продукт с [открытым исходным кодом](#).

Программа без графического интерфейса (работает из консоли).

В схеме с использованием HSS:

- HSS-загрузчик хранится в eNVM (встроена в Polarfire SoC)
- Пользовательское приложение хранится во внешней SPI-FLASH (SD, MMC) в виде специально оформленного образа (**Payload**)
- При включении питания (или перезагрузке):
 - сначала запускается HSS-загрузчик
 - далее загрузчик загружает пользовательское приложение из внешней памяти (например, в ОЗУ) и запускает его.

7.1 Настройка рабочего пространства

Сборка образа пользовательского приложения выполняется в директории «resources», которая находится в директории проекта – на одном уровне с директорией «src»:

```
resources
src
```

В директорию «resources» необходимо скопировать исполняемый файл самого генератора (со всеми прилагаемыми библиотеками). Здесь же будет находиться файл конфигурации (payload.yaml) и результат сборки – образ пользовательского приложения (payload.bin и payload.hex).


```
hss-payload-generator  exe
msys-2.0              dll
msys-crypto-1.1       dll
msys-elf-0            dll
msys-gcc_s-seh-1      dll
msys-yaml-0-2         dll
msys-z                dll
payload               yaml
```

7.2 Конфигурирование

Образ пользовательского приложения (payload.bin, .hex) содержит:

- строковое имя
- адреса «стартовых точек» (entry points) для каждого ядра
- описание пользовательского приложения:
 - исполняемый код (bare_metal.elf)
 - адрес «стартовой точки» (exec address)
 - имя основного ядра и пр.

Выше перечисленная структура образа описывается конфигурационным файлом формата YAML (*payload.yaml*), где:

1) задается имя образа

2) задаются адреса «стартовых точек»

```
hart-entry-points: {  u54_1: '<from *.ld>', ... }
                   *.elf: {exec-addr: '<from *.ld>'}
```

* адреса берутся из файла линковщика
(bare_metal.ld проекта пользовательского приложения)

3) исполняемый код определен как путь к исполняемому файлу (bare_metal.elf)

7.3 Сборка

Команда консоли:

```
hss-payload-generator.exe -c payload.yaml payload.bin
```

где,

hss-payload-generator.exe – исполняемый файл генератора образа

payload.yaml – файл конфигурации образа

payload.bin – образ пользовательского приложения в формате bin

7.4 Конвертация в hex

Команда консоли:

```
binscopy.exe convert -i binary -o ihex payload.bin payload.hex
```

или

```
objcopy.exe -I binary -O ihex payload.bin payload.hex
```

где,

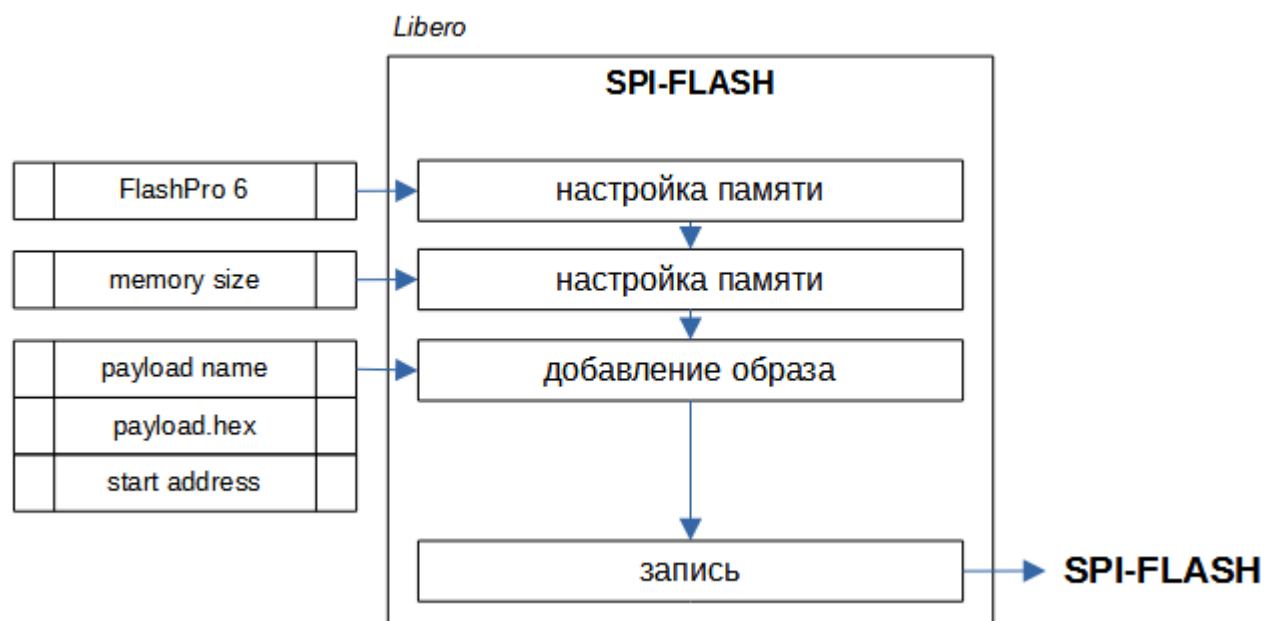
binscopy.exe – утилита конвертации
(например, из состава python3: `pip install binscopy`)

objcopy.exe – утилита конвертации
(например, riscv64-unknown-elf-objcopy.exe из состава SoftConsole 2022)

payload.bin – образ пользовательского приложения в формате bin (вход)

payload.hex – образ пользовательского приложения в формате hex (результат)

8 SPI-FLASH



Libero (2022).

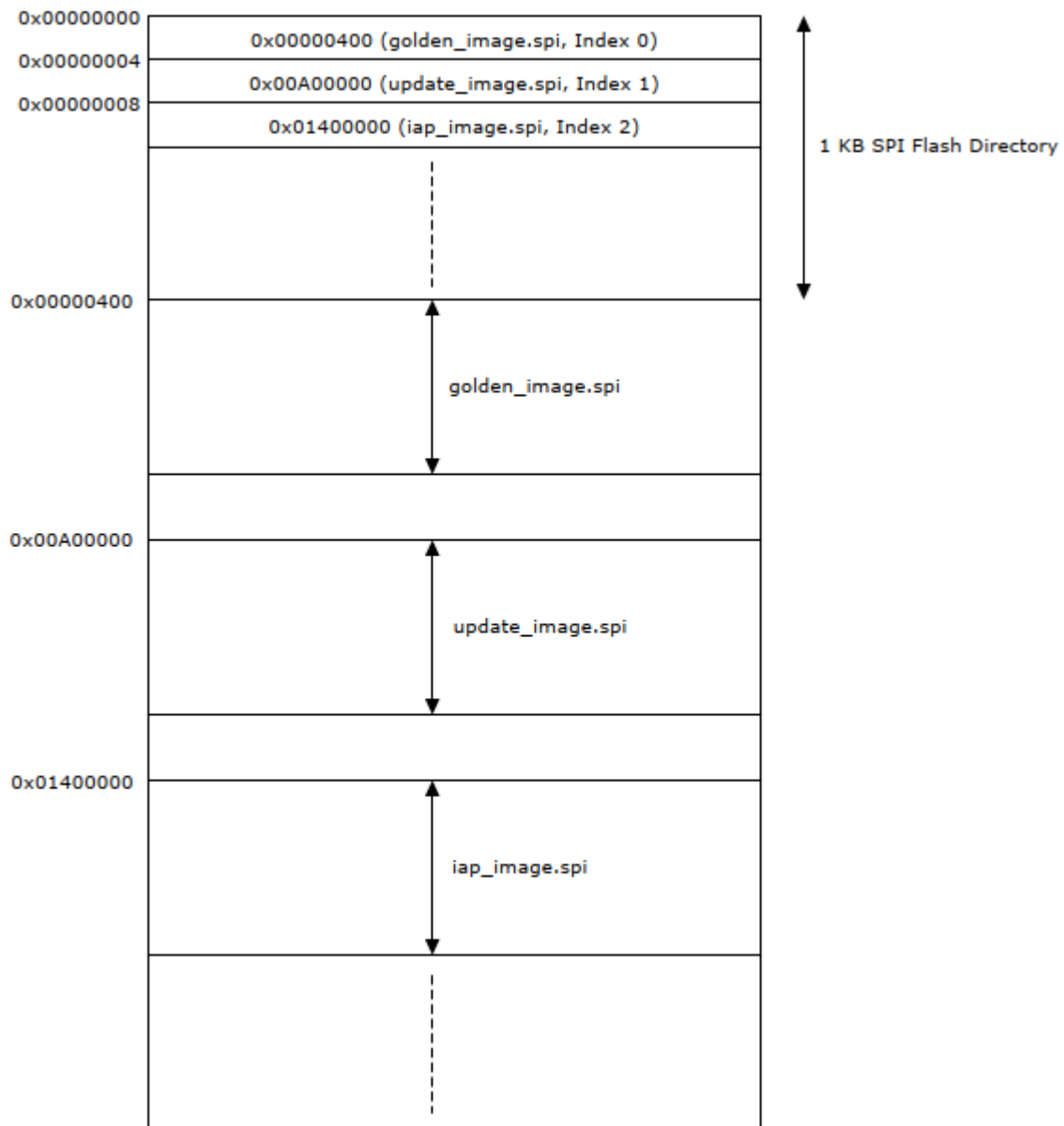
Образ пользовательского приложения хранится в SPI-Flash в отдельной директории.

Первый Килобайт файловой системы:

- таблица адресов директорий SPI-Flash

Далее располагаются директории с образами пользовательских приложений:

- Golden Image – заводской образ
- Update Image – образ обновления
- IAP Image – новый образ



8.1 Настройка памяти

1) В конфигурации дизайна (Libero) выбрать:
Program Design / Configure Design Initialization Data and Memories

2) Выбрать вкладку:
SPI Flash

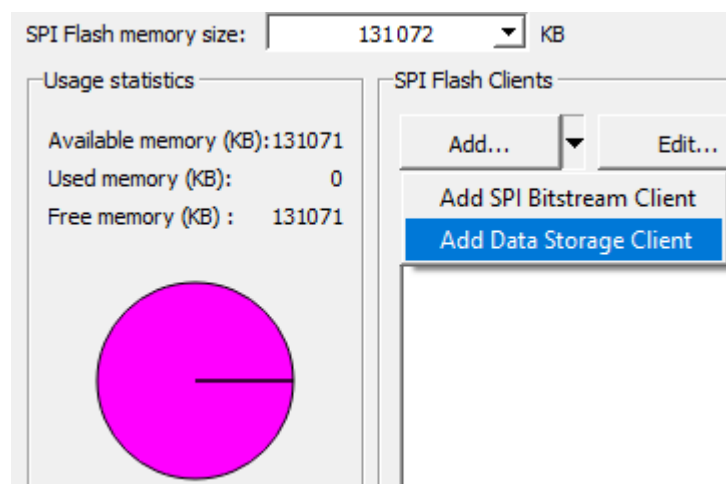
3) Указать размер используемой SPI-Flash (например, для ICICLE-KIT):
SPI Flash memory size: 131 072 KB

SPI Flash memory size: KB

8.2 Добавление образа

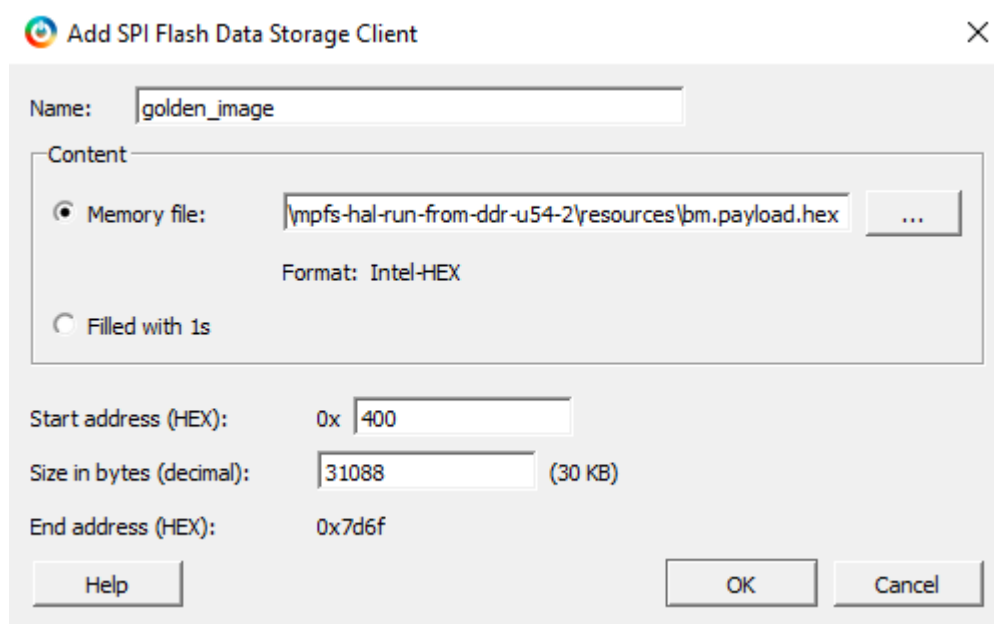
1) Добавить образ пользовательского приложения:

Add... Data Storage Client



2) Указать:

- имя директории (например, golden_image)
- файл образа (payload.hex)
- стартовый адрес директории в SPI-Flash, куда будет записан образ



3) Применить изменения:

OK ... Apply ... Project/Save

8.3 Запись

1) Записать сформированную файловую структуру в SPI-Flash:
`Run PROGRAM Action/Run PROGRAM_SPI_IMAGE Action`

- * во всех всплывающих диалоговых окнах нажать «Yes»
- * дождаться завершения процесса записи

БАЗА ЗНАНИЙ

[Bare Metal Software Projects Structure](#)

[Hart Software Services](#)

[Memory Configuration](#)

[Creating Bare Metal Payload For HSS](#)

[Hart Software Services Payloads](#)

[Programming the SPI Flash on a PolarFire SoC Board and Booting via HSS](#)

[Icicle Kit Bring Up Design Bitstream Builder Readme](#)

[PolarFire SoC MSS Driver User Guides](#)