#### ИНСТРУКЦИЯ 1.0

**Установка системы**

**Настройка системы**

Для использования табеля учета рабочего времени требуется система со слотом PCIe.

**Настройки BIOS**

Включите виртуализацию ввода-вывода вашего процессора в BIOS:

* Для процессоров Intel: Включить Vt-D или Vt-x
* Для процессоров AMD: Включить OMMU

# CentOS

Для сборки последней версии драйвера и использования всех передовых функций Qantum Card мы рекомендуем создавать ядро с нуля.

### Загружаем исходник в каталог

сd

~git clone https://github.com/Qantum

### Установите последние ключи ядра

sudo dnf -y install <https://www.elrepo.org/elrepo-release-8.el8.elrepo.noarch.rpm>

sudo rpm --import <https://www.elrepo.org/RPM-GPG-KEY-elrepo.org>

dnf --enablerepo=elrepo-kernel install kernel-ml

reboot

### Установите необходимые пакеты

yum install -y ncurses-devel make gcc bc bison flex elfutils-libelf-devel openssl-devel grub2 i2c-tools git

### Загрузите полный исходный код ядра в Интернете.

cd /usr/src/kernels

wget <https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.17.4.tar.xz>

tar -xvf linux-5.17.4.tar.xz

rm linux-5.17.4.tar.xz

mv linux-5.17.4/ 5.17.4/

cd 5.17.4/

### Скопируйте существующую конфигурацию ядра в исходный код ядра

cp -v /boot/config-5.17.4-1.el8.elrepo.x86\_64 .config

vim .config

Вам нужно будет отредактировать .config и указать вновь установленное ядро.

### Измените параметры сборки ядра в .config

CONFIG\_I2C\_XILINX=m

CONFIG\_MTD=m

CONFIG\_MTD\_SPI\_NOR=m

CONFIG\_SPI=y

CONFIG\_SPI\_ALTERA=m

CONFIG\_SPI\_BITBANG=m

CONFIG\_SPI\_MASTER=y

CONFIG\_SPI\_MEM=y

CONFIG\_SPI\_XILINX=m

CONFIG\_I2C=y

CONFIG\_I2C\_OCORES=m

CONFIG\_IKCONFIG=y

CONFIG\_EEPROM\_AT24=m

Затем сделайте это, make oldconfig чтобы добавить CONFIG\_IKCONFIG\_PROC=y.

Ответьте на y или n на все вопросы.

make oldconfig

**ПРИМЕЧАНИЕ:**

**ПЕРЕХОДИТЕ К этапу временной отладки 5.17.4**, начиная с 17.03.2022, исправьте ядро этими двумя исправлениями, чтобы устранить проблемы, связанные с обновлением Flash FPGA SPI.

patch -p1 < ~/Time-Appliance-Project/Time-Card/DRV/0001-spi-xilinx-Inhibit-transmitter-while-filling-TX-FIFO.patch

patch -p1 < ~/Time-Appliance-Project/Time-Card/DRV/0001-spi-nor-Send-soft-reset-before-probing-flash.patch

### Соберите и установите это ядро

make bzImage -j4; make modules -j4; make -j4;

make INSTALL\_MOD\_STRIP=1 modules\_install; make install;

### Перезагрузка

reboot

### Включить все периферийные устройства UART

grubby --update-kernel=ALL --args=8250.nr\_uarts=8

reboot

Подтвердите перечисление всех включенных периферийных устройств UART при следующей перезагрузке,

cat /proc/cmdline[root@localhost Binaries]# cat /proc/cmdlineBOOT\_IMAGE=(hd1,gpt2)/vmlinuz-5.16.0 root=/dev/mapper/cs-root ro crashkernel=auto resume=/dev/mapper/cs-swap rd.lvm.lv=cs/root rd.lvm.lv=cs/swap rhgb quiet 8250.nr\_uarts=8

### Установите драйверы из репозитория проекта Qantum

cd ~/Qantum/Time-Card/DRV/./remakemodprobe ptp\_ocp

### Подтверждение установки

Просмотрите выходные данные dmesg, чтобы подтвердить, что устройство Time Card установлено и драйвер ядра успешно перечислил устройство.

Пример вывода, подтверждающий, что ptp\_ocp драйвер ядра перечисляет устройство карты времени.

[ 1491.440223] 0000:02:00.0: ttyS5 at MMIO 0x70161000 (irq = 180, base\_baud = 3125000) is a 16550A[ 1491.440278] 0000:02:00.0: ttyS6 at MMIO 0x70171000 (irq = 181, base\_baud = 3125000) is a 16550A[ 1491.440322] 0000:02:00.0: ttyS7 at MMIO 0x70181000 (irq = 182, base\_baud = 3125000) is a 16550A[ 1491.440362] 0000:02:00.0: ttyS0 at MMIO 0x70191000 (irq = 187, base\_baud = 3125000) is a 16550A[ 1491.441702] xilinx\_spi xilinx\_spi.1024: at [mem 0x70310000-0x7031ffff], irq=186[ 1491.442678] spi-nor spi1024.0: n25q128a13 (16384 Kbytes)[ 1491.447310] pps pps1: new PPS source ptp2[ 1491.449374] ptp\_ocp 0000:02:00.0: Version 1.2.0, clock PPS, device ptp2[ 1491.449388] ptp\_ocp 0000:02:00.0: Time: 1647548640.775286647, in-sync[ 1491.449390] ptp\_ocp 0000:02:00.0: version 8005[ 1491.449392] ptp\_ocp 0000:02:00.0: regular image, version 32773[ 1491.449393] ptp\_ocp 0000:02:00.0: GNSS: /dev/ttyS5 @ 115200[ 1491.449394] ptp\_ocp 0000:02:00.0: GNSS2: /dev/ttyS6 @ 115200[ 1491.449396] ptp\_ocp 0000:02:00.0: MAC: /dev/ttyS7 @ 57600[ 1491.449399] ptp\_ocp 0000:02:00.0: NMEA: /dev/ttyS0 @ 115200

# Вычислительный модуль Raspberry Pi

**В настоящее время ведется работа**

### Сборка

Параметры сборки,

CONFIG\_NET\_DEVLINK=y CONFIG\_SERIAL\_8250\_NR\_UARTS=8 CONFIG\_IRQ\_POLL=y

### Обновления Kconfig

Измените Kconfig в net, чтобы включить DEVLINK

### Создать

make oldconfig перед выполнением build, но после добавления вручную DEVLINK

# Использование

Карта подключается через sysfs.

### Sysfs

Для просмотра всех sysfs представленных параметров,

ls /sys/class/timecard/ocp0/

Дополнительные устройства для учета рабочего времени будут перечислены в порядке, ocpN где N указан порядок устройств.

### **Последовательные порты и номер PHC**

**Последовательные порты и номер PHC можно найти через dmesg или ls -l**

## Использование ввода-вывода Qantum Card

### Выходные данные

Все возможные результаты доступны через

cat available\_sma\_outputs

Отобразит все возможные типы выходных данных, такие как,

10Mhz PHC MAC GNSS1 GNSS2 IRIG DCF GEN1 GEN2 GEN3 GEN4 GND VCC

Ниже показано несколько примеров настройки выходных сигналов,

1. Вывод PPS ПЛИС на SMA

echo OUT: PHC >> sma1

1. Вывод PPS атомных часов на SMA

echo OUT: MAC >> sma1

1. Вывод PPS модуля GPS на SMA

echo OUT: GNSS1 >> sma1

### Входные данные

Все возможные входные данные доступны через available\_sma\_inputs.

cat available\_sma\_inputs

Отобразит все возможные типы входных данных, такие как,

10Mhz PPS1 PPS2 TS1 TS2 IRIG DCF TS3 TS4 FREQ1 FREQ2 FREQ3 FREQ4 None

Ниже показано несколько примеров настройки входных сигналов,

#### Настройка портов SMA

1. Используйте порт для ввода временных меток входящих сигналов, это настроено SMA1 с помощью timestamper-1 TS1

echo IN: TS1 >> sma1

Для обратного считывания сигналов с отметкой времени используйте testptp.

#### Загрузить testptp исходный код

Источник доступен в этих местах

1. <https://www.mjmwired.net/kernel/Documentation/ptp/testptp.c>
2. <https://github.com/torvalds/linux/blob/master/tools/testing/selftests/ptp/testptp.c>

#### Считывание сигналов с отметкой времени

**Примечание:** -1 для начальных показаний используйте положительное число для фиксированного количества событий, начиная с Timestamper 1 -i 1 с временной карты /dev/ptp1

./testptp -d /dev/ptp1 -e -1 -i 1

# Инструмент прошивки

Это простой инструмент для добавления заголовка к исходному двоичному файлу прошивки. Заголовок используется драйвером для проверки совместимости оборудования с образом. Заголовок имеет длину всего 16 байт и содержит значения идентификатора поставщика PCI, идентификатора устройства PCI и идентификатора версии оборудования PCI для проверки. Он также содержит значение CRC необработанного изображения. Причина добавления заголовка заключается в том, чтобы быть уверенным, что несовместимое firware не будет непреднамеренно записано на устройство через devlink.

## Формат заголовка

| **Волшебные байты** | **Идентификатор поставщика PCI** | **Идентификатор устройства PCI** | **Размер изображения** | **Идентификатор версии HW** | **CRC16** |
| --- | --- | --- | --- | --- | --- |
| 4 байта | 2 байта | 2 байта | 4 байта | 2 байта | 2 байта |

Заголовок firmware состоит из 6 полей, все значения приведены в сетевом порядке, чтобы быть согласованными в разных архитектурах:

1. Волшебный заголовок (4 байта, 32 бита) - постоянное значение, ‘OCTC’
2. Идентификатор поставщика PCI (2 байта, 16 бит) - идентификатор поставщика устройства PCI, совместимый с этим изображением
3. Идентификатор устройства PCI (2 байта, 16 бит) - идентификатор устройства PCI, совместимый с этим изображением
4. Размер изображения (4 байта, 32 бита, без знака) - размер самой прошивки без заголовка (и нижнего колонтитула, если он у нас есть)
5. Версия HW (2 байта, 16 бит) - Информация, предоставляемая регистром HW для различения версий одной и той же платы.
6. CRC16 (2 байта, 16 бит) - проверка значения реализации CRC16 с полиномом по умолчанию, реализованным в ядре

## Использование

Инструмент имеет несколько опций:

* -input <filename> - Обязательный параметр, предоставляет имя файла необработанного двоичного файла.
* -output <filename> - Обязательный параметр, предоставляет имя файла для записи нового файла прошивки с заголовком. Если файл уже существует, он будет перезаписан.
* -vendor <int> - Обязательный параметр, предоставляет идентификатор поставщика PCI для добавления в заголовок.
* -device <int> - Обязательный параметр, предоставляет идентификатор устройства PCI для добавления в заголовок.
* -hw <int> - Необязательно, используется для указания идентификатора обновления оборудования PCI. По умолчанию равно 0.
* -apply - Необязательно. Используется для фактического создания нового (или перезаписи) выходного файла с заголовком в начале.

## Примеры

./tft -input Time-Card/FPGA/Binary/Production/Binaries/TimeCardProduction.bin -output TimeCardProduction\_Celestica.bin -vendor 0x18d4 -device 0x1008 -apply

Этот вызов создаст TimeCardProduction\_Qantum.bin с заголовком для Qantum Card