Микроконтроллер BE-U1000
Рекомендации по использованию
Настройка тактовых сигналов

АО «БАЙКАЛ ЭЛЕКТРОНИКС», ИНН: 7707767484
АО «БАЙКАЛ ЭЛЕКТРОНИКС» оставляет за собой право вносить
любые изменения в настоящий документ без дополнительного уведомления

Содержание

1 Введение....2
2 Сокращения и термины....3
3 Общая информация....4
4 Источники тактового сигнала....6
4.1 Внешний источник тактового сигнала....6
4.2 Внутренний генератор....6
4.3 Выбор основного источника сигнала SYS CLK....7
4.4 Отслеживание наличия внешнего тактового сигнала и переход на внутренний генератор....8
5 Блок PLL....11
5.1 Общая информация....11
5.2 Расчет частоты блока PLL....11
5.3 Настройка блока PLL....12
6 Домены тактирования подсистем микроконтроллера....17
6.1 Общая информация....17
6.2 Делитель частоты блока PLL для домена....17
6.3 Выбор источника тактового сигнала для домена....19
6.4 Управление тактированием периферии....20
6.5 Сброс....23
7 Микросекундный таймер ядра....25
8 Примеры настройки системы тактирования....27
8.1 Общая информация....27
8.2 Конфигурация тактирования от SYS CLK....27
8.3 Конфигурация тактирования от PLL на максимальные частоты....28
История изменений....30
Контактные данные....31

1 Введение

1.1 В настоящем документе представлена подробная информация, примеры и разъяснения по настройке системы тактирования микроконтроллера BE-U1000.

1.2 Документ снабжен примерами кода на языке Си.

1.3 Перед прочтением настоящего документа рекомендуется ознакомиться с документами: «BE-U1000. Краткое описание», «BE-U1000. Руководство пользователя», «BE-U1000. Техническая спецификация», «BE-U1000. Reference Manual».

2 Сокращения и термины

2.1 В документе используются сокращения и термины, указанные в таблице 1.

Таблица 1 — Сокращения и термины

Обозначение	Расшифровка
ADC	АЦП (аналого-цифровой преобразователь) – электронное устройство, преобразующее непрерывный аналоговый сигнал в дискретный цифровой код.
Core 0/1/2	Ядро 0/1/2 микроконтроллера BE-U1000.
CLIC	Аппаратный блок в архитектуре RISC-V, предназначенный для обработки прерываний на уровне отдельного процессорного ядра.
CRU	Control Registers Unit – блок микроконтроллера, отвечающий за общие настройки: тактирование, сброс, выбор режимов работы ядер Core 1 и Core 2, GPIO и т.д.
eFlash	Встроенная флэш-память микроконтроллера. Содержит две области – Main и NVR.
GF MUX	Glitch-free Multiplexer – устройство, предназначенное для переключения между несколькими входными тактовыми сигналами без возникновения паразитных импульсов на выходе.
PLL	Phase-Locked Loop (блок фазовой автоподстройки частоты, блок ФАПЧ, блок PLL) – устройство, используемое для формирования тактовых сигналов с заданной частотой.
PWMG	Pulse Width Modulation Generator – генератор ШИМ
QSPI	Quad SPI – расширение классического SPI, позволяющее передавать данные по четырём линиям параллельно.
SPI	Serial Peripheral Interface (последовательный периферийный интерфейс) – это синхронный последовательный интерфейс для обмена данными между микроконтроллерами и периферией (датчики, дисплеи, SD-карты).
VCO	Voltalge Controlled Oscilator (генератор, управляемый напряжением) – электронное устройство, генерирующее периодический сигнал, частота которого изменяется в зависимости от величины управляющего напряжения.

3 Общая информация

3.1 Схема тактирования микроконтроллера представлена на рисунке 1 .

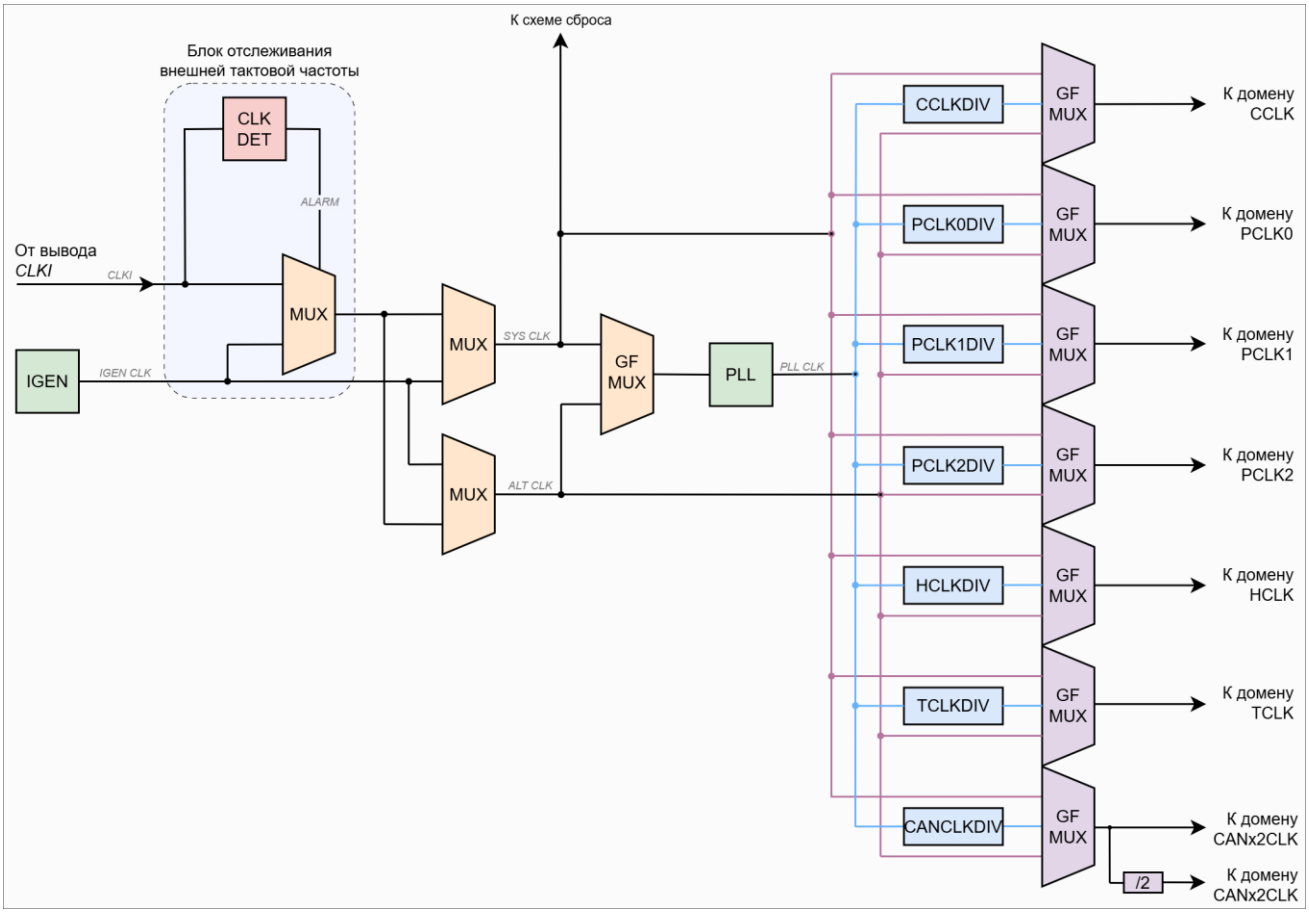


Рисунок 1 – Схема тактирования

3.2 Микроконтроллер BE-U1000 для работы может использовать следующие источники тактирования:

- внешний генератор, подключенный к выводу CLKI;
- внутренний генератор IGEN.

3.3 При помощи системы мультиплексоров можно требуемым образом коммутировать промежуточные сигналы. Сигнал SYS CLK также участвует в процессе снятия сигнала NRESET с микроконтроллера.

3.4 Выходной тактовый сигнал PLL (PLL CLK) может использоваться в качестве синхросигнала для различных доменов тактирования микроконтроллера наравне с сигналами SYS CLK и ALT CLK.

3.5 Схема тактирования включает в себя CLK DET – блок отслеживания внешнего тактового сигнала CLKI. В случае пропадания сигнала CLKI блок отслеживания осуществляет переключение на сигнал внутреннего генератора IGEN (IGEN CLK).

3.6 Микроконтроллер имеет 8 настраиваемых доменов тактирования для различных потребителей тактовых сигналов. Их перечень приведен в таблице 2.

Таблица 2 — Домены тактирования МК

Домен	Максимальная частота	Потребители
CCLK	200 МГц	Основная коммутационная матрица, ядра Core 0/1, eFlash, ROM MicroPython, блок трассировки.
PCLK0	100 МГц	GPIO0, CANFD0, PWMG0, UART0/1/2, SPI0/1, QSPI0, I2C0/1, TIM0, I2S0, WDT0.
PCLK1	100 МГц	GPIO1, CANFD1, PWMG1, I2S1, UART3/4/5, SPI2/3, QSPI1, I2C2/3, TIM1, WDT1.
PCLK2	100 МГц	GPIO2, UART6/7, ADC0/1/2, PWMA0/1/2/3.
HCLK	100 МГц	USB 2.0, DMA0/1, блок Mailbox, ядро Core 2, блок трассировки.
TCLK	100 МГц	Блок трассировки.
CANx2CLK	200 МГц	CANFD0/1 CANx2.
CANCLK	100 МГц	CANFD0/1 CAN.

4 Источники тактового сигнала

4.1 Внешний источник тактового сигнала

4.1.1 Внешним тактовым сигналом должен быть периодический сигнал, подаваемый на вывод CLKI. Частота внешнего тактового сигнала не должна превышать 50 МГц. Для использования CLKI в качестве опорного сигнала PLL, его частота должна быть 10,2 – 50 МГц.

4.2 Внутренний генератор

4.2.1 Использование внутреннего генератора предполагается в качестве источника:

– тактового сигнала низкой частоты в приложениях, не предъявляющих требований к точности тактовой частоты;

– тактового сигнала при старте микроконтроллера в случае отсутствия внешнего тактового сигнала на выводе CLKI;

– аварийного тактирования микросхемы в случае пропадания внешнего тактового сигнала на выводе CLKI.

4.2.2 Типовая зависимость частоты генератора от напряжения в домене питания V1P2 и температуры приведена на рисунке 2.

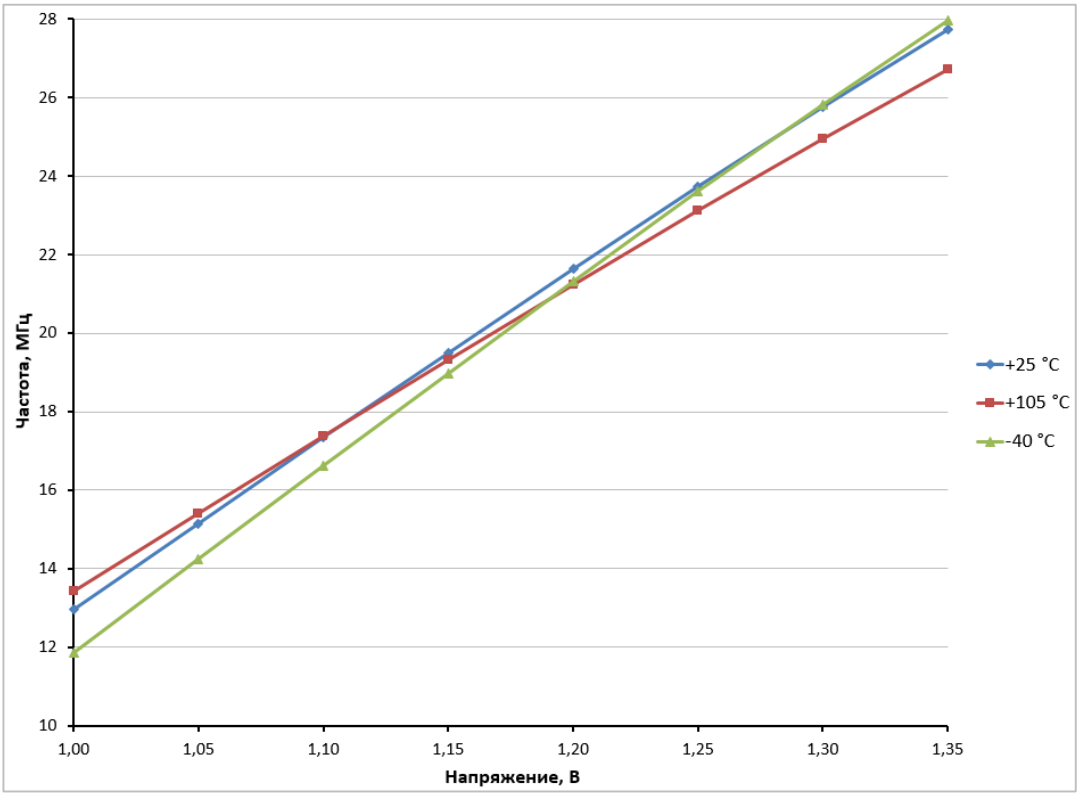


Рисунок 2 – Зависимость частоты внутреннего генератора от температуры и напряжения

4.2.3 Значение частоты внутреннего генератора в кГц можно прочитать из памяти NVR:

```
#define RC_FREQ_PTR ((uint16_t *)(EFLASH_BASE + 0x28)) //1
uint32_t RC_FrequencyFromNVRGet(void){
    uint32_t sysctrl0 = CRU->SYSCR0;
    CRU->SYSCR0 |= CRU_SYSCR0_NVR_Msk;  //2
    uint16_t freq = *(RC_FREQ_PTR);     //3
    CRU->SYSCR0 = sysctrl0;
    return (uint32_t)freq * 1000;       //4
}
```

① адрес ячейки памяти NVR, в которой хранится значение частоты сигнала внутреннего генератора;

② включение перенаправления прямого доступа в память из eFlash в NVR;

③ считывание значения номинальной частоты внутреннего генератора в кГц;

④ перевод частоты внутреннего генератора из кГц в Гц.

4.2.4 По умолчанию внутренний генератор включен. Включение и выключение внутреннего генератора производится записью бита `OSCGENEN` регистра `CLKCR0`:

- выключить: `0`;
- включить: `1`.

```
void RC_TurnOff(){
    CRU->CLKCR0 &= ~CRU_CLKCR0_RCGENEN_Msk;
}
void RC_TurnOn(){
    CRU->CLKCR0 |= CRU_CLKCR0_RCGENEN_Msk;
}
```

4.3 Выбор основного источника сигнала SYS CLK

4.3.1 Какой сигнал (CLKI или IGEN CLK) будет использоваться в качестве основного тактового сигнала SYS CLK зависит от:

а) состояния конфигурационного вывода PC[6];

б) битового поля `BOOTCLKSEL` регистра `CLKCR0`.

Выбор сигнала в зависимости от состояния конфигурационного вывода PC[6] и бита `BOOTCLKSEL` показана в таблице 3.

Таблица 3 — Выбор источника основного сигнала SYS CLK

Источник SYS CLK	PC[6]	CLKCR0.BOOTCLKSEL
CLKI	0	0
IGEN CLK	1	0
CLKI	0	1
CLKI	1	1

При начальной загрузке микроконтроллера бит `BOOTCLKSEL` устанавливается в `0`.

4.3.2 Состояние конфигурационных выводов, считанное при снятии сигнала NRESET, можно прочитать в битовом поле `STRAP` регистра `SYSSR0`:

```
uint32_t CLK_Strap_Get(){
    return (CRU->SYSSR0 & CRU_SYSSR0_STRAP_CLK_Msk) >> CRU_SYSSR0_STRAP_CLK_Pos;
}
```

Таким образом, если при старте микроконтроллера конфигурационный вывод PC[6] в состоянии логической единицы, то в процессе работы возможно изменить источник основного синхросигнала SYS CLK.

4.4 Отслеживание наличия внешнего тактового сигнала и переход на внутренний генератор

4.4.1 В микроконтроллере реализовано устройство отслеживания наличия внешнего тактового сигнала. При его отсутствии осуществляется переход на сигнал IGEN CLK.

Если в течение установленного (поле `ALARM_LVL` регистра `SYSCR0`) количества тактов внутреннего генератора устройство отслеживания не фиксирует наличие сигнала CLKI, оно отправляет сигнал, который переводит SYS CLK и ALT CLK на тактирование от внутреннего генератора, а PLL переводит в режим вывода опорного сигнала (BYPASS). Также по данному сигналу будет сформировано прерывание на ядрах Core 0 и Core 1.

Таким образом, если включен CLKI ALARM и в течение установленного количества тактов внутреннего генератора не было зафиксировано ни одного такта от внешнего генератора, то микроконтроллер будет принудительно переведен на тактирование от внутреннего генератора.

Включить устройство отслеживания можно записью `1` в бит `ALARM_EN` регистра `SYSCR0` (по умолчанию блок выключен):

```
uint8_t CLKI_Mon_TurnOn(uint8_t alarm_level){
    alarm_level = alarm_level > 15 ? 15 : alarm_level;           //1
    CRU->SYSCR0 &= ~CRU_SYSCR0_CLKI_ALARM_EN_Msk;                //2
    CRU->SYSCR0 &= ~CRU_SYSCR0_CLKI_ALARM_LVL_Msk;               //3
    CRU->SYSCR0 |= alarm_level << CRU_SYSCR0_CLKI_ALARM_LVL_Pos; //4
    CRU->SYSCR0 |= CRU_SYSCR0_CLKI_ALARM_EN_Msk;                 //5
    return alarm_level;                                          //6
}
void CLKI_Mon_TurnOff(){
    CRU->SYSCR0 &= ~CRU_SYSCR0_CLKI_ALARM_EN_Msk;
}
```

① проверка переданного параметра на превышения возможного значения. В случае превышения устанавливается максимально возможное;

② отключение блока монитора частоты CLKI;

③ сброс ранее установленного порога срабатывания монитора частоты;

④ установка нового порога срабатывания монитора частоты;

⑤ включение блока монитора частоты CLKI;

⑥ возврат реально установленного порогового значения.

4.4.2 По событию переключения источника тактового сигнала может быть настроено прерывание. Сигнал от устройства отслеживания заведен на 74-ю линию маскируемых прерываний Core 0 и Core 1. Для сброса прерывания необходимо записать `0` в бит `ALARM_EN` регистра `SYSCR0`.

```
void CLKI_Mon_IrqTurnOn(){
    RC_TurnOn();                //1
    CLKI_Mon_TurnOn(15);        //2
    CLIC_SetCfgNlbits(0);       //3
    CLIC_SetLevelThreshold(0);
    CLIC_SetLevel(CLIC_CLKI_Mon_IRQn, 1);
    CLIC_SetVectorMode(CLIC_CLKI_Mon_IRQn, CLIC_INTATTR_SHV_VECTORED);
    CLIC_SetTrigType(CLIC_CLKI_Mon_IRQn, CLIC_INTATTR_TRIG_TYPE_LEVEL);
    CLIC_SetTrigPolarity(CLIC_CLKI_Mon_IRQn, CLIC_INTATTR_TRIG_POL_P);
    CLIC_SetPrivMode(CLIC_CLKI_Mon_IRQn, CLIC_INTATTR_MODE_MACHINE);
    CLIC_EnableIRQ(CLIC_CLKI_Mon_IRQn);
    __enable_irq();             //4
}
__attribute__((interrupt))
```

```
void CLKI_Mon_IRQHandler(){
    CRU->SYSCR0 &= ~CRU_SYSCR0_CLKI_ALARM_EN;                //5
    CLIC_DisableIRQ(CLIC_CLKI_Mon_IRQn);                     //6
    uint32_t rc_freq = RC_FrequencyFromNVRGet();             //7
    con_ctx.freq = rc_freq;                                  //8
    XCP_PROTO_GET(console)->init(&con_ctx);                  //9
    XCP_PROTO_GET(stdio)->printf("CLKI Monitor. Alarm!\n"); //10
}
```

① включение внутреннего генератора;

② включение устройства отслеживания наличия внешнего тактового сигнала;

③ настройка и включение прерывания от монитора частоты CLKI;

④ глобальное разрешение прерываний;

⑤ сброс флага прерывания и выключение монитора частоты CLKI;

⑥ выключение обработки прерывания от монитора частоты CLKI;

⑦ чтение частоты внутреннего генератора из NVR;

⑧ используется ROM XCP Protocol. Установка частоты внутреннего генератора как частоты тактирования UART-консоли;

⑨ используется ROM XCP Protocol. Инициализация UART-консоли с учетом новой частоты тактирования;

⑩ используется ROM XCP Protocol. Вывод в консоль сообщения о прерывании.

5 Блок PLL

5.1 Общая информация

5.1.1 Блок PLL предназначен для синтеза сигналов тактирования в широком диапазоне частот. Блок PLL имеет конфигурируемые делители:

NR – делитель частоты опорного сигнала;

NF – делитель частоты сигнала в цепи обратной связи

OD – делитель выходной частоты.

5.1.2 Блок PLL имеет следующие сигналы управления:

POWER DOWN – отключает питание аналоговой схемы PLL, работоспособной остается только цифровая часть модуля (делители). В результате снижается энергопотребление модуля.

RESET – управляет перезапуском PLL. Отключает опорный сигнал VCO от его входа, устанавливает для всех делителей коэффициент деления 1.

BYPASS – управляет выходным мультиплексором. Если BYPASS активен, опорный сигнал PLL поступает напрямую на выход, если не активен – опорный сигнал поступает на делитель выходной частоты.

5.1.3 Значение делителей частоты и комбинации сигналов управления определяет наличие выходного сигнала PLL и его частоту.

5.1.4 После снятия сигнала NRESET сигналы PLL RESET и BYPASS переходят в активное состояние. Исключением является случай старта микроконтроллера с использованием USB.

5.2 Расчет частоты блока PLL

5.2.1 Режим, в котором все сигналы управления находятся в неактивном состоянии, является основным рабочим режимом блока PLL. Выходная частота блока PLL в этом режиме определяется по формуле:

$$F_{out} = F_{ref} \times (\frac{NF}{NR \times OD})$$, где:

F_{ref} – опорная частота блока PLL;

NR – коэффициент деления частоты опорного сигнала;

NF – коэффициент деления частоты обратной связи,

OD – коэффициент деления частоты VCO.

5.2.2 Опорная частота блока PLL F_{ref} и частота фазового детектора $\frac{F_{ref}}{NR}$ должны находиться в диапазоне от 10,2 до 50 МГц. Частота VCO PLL F_{VCO} должна находиться в диапазоне от 260 до 1300 МГц:

$$F_{VCO} = \frac{F_{ref} \times NF}{NR},$$

при этом коэффициенты деления блока PLL могут находиться в пределах:

NR: от 1 до 16;

NF: от 1 до 64;

OD: от 1 до 16.

5.2.3 После снятия сигнала NRESET значение всех делителей равно `1` (кроме варианта загрузки в режиме «USB»).

5.2.4 В режиме передачи опорного сигнала (сигнал BYPASS активен) опорный сигнал проходит в обход блока PLL и подается непосредственно на выход:

$$F_{out} = F_{ref}.$$

Аналоговая цепь PLL при этом остается включенной, но не используется. Ввиду этого использование этого сигнала можно комбинировать с сигналом POWER DOWN для уменьшения потребления.



При активном сигнале RESET VCO блок PLL продолжает работать, и на выходе блока может формироваться сигнал произвольной частоты. Во избежание этого рекомендуется переводить блок PLL в режим POWER DOWN.

5.3 Настройка блока PLL

5.3.1 В качестве источника опорного сигнала с помощью мультиплексора может быть выбран сигнал SYS CLK или ALT CLK. Управление мультиплексором осуществляется записью бита `I_PLL_CLK_SEL` регистра `CLKSEL`.

```
typedef enum {
    SYS_CLK=0,
    ALT_CLK=1
} REF_CLK_e;
void PLL_RefSourceSet(REF_CLK_eref_clk){ //1
    if (ref_clk == SYS_CLK){
        CRU->CLKSEL &=~CRU_CLKSEL_REFCLKSEL_Msk;
    } else if (ref_clk == ALT_CLK){
        CRU->CLKSEL |= CRU_CLKSEL_REFCLKSEL_C1;
    }
```

```
}
```

① функция принимает параметр: `SYS_CLK` или `ALT_CLK`, в результате чего в бит регистра управления мультиплексором записывается `0` или `1`.

Настройка коэффициентов NR, NF и OD осуществляется с помощью битовых полей `CLKR`, `CLKF` и `CLKOD` регистра `PLLSET`. Значения рассчитываются по формулам:

$$CLKR = NR - 1,$$
$$CLKR = NR - 1,$$
$$CLOD = OD - 1.$$

5.3.2 Битовое поле `BWADJ` регистра `PLLSET` определяет ширину полосы пропускания PLL. Как правило, значение `BWADJ` совпадает со значением `CLKF`.

```
void PLL_DividersSet(uint8_t nr, uint8_t nf, uint8_t od){
    uint32_t clkr = (nr - 1) << CRU_PLLSET_CLKR_Pos; // 1
    uint32_t clkf = (nf - 1) << CRU_PLLSET_CLKF_Pos;
    uint32_t clkod = (od - 1) << CRU_PLLSET_CLKOD_Pos;
    uint32_t clkbw = (nf - 1) << CRU_PLLSET_BWADJ_Pos;
    clkr &= CRU_PLLSET_CLKR_Msk;            // 2
    clkf &= CRU_PLLSET_CLKF_Msk;
    clkod &= CRU_PLLSET_CLKOD_Msk;
    clkbw &= CRU_PLLSET_BWADJ_Msk;
    uint32_t pllset = CRU->PLLSET;          // 3
    pllset &= ~(CRU_PLLSET_CLKR_Msk | CRU_PLLSET_CLKF_Msk | CRU_PLLSET_CLKOD_Msk|
CRU_PLLSET_BWADJ_Msk);                      // 4
    pllset |= clkr | clkf | clkod | clkbw; // 5
    CRU->PLLSET = pllset;                   // 6
}
```

① сдвиг значений делителей на позиции, занимаемые в регистре;

② наложение маски на значения делителей для лишних бит;

③ запись значения регистра `PLLSET` в переменную;

④ сброс старых значений делителей в переменной;

⑤ установка новых значений делителей в переменной;

⑥ сохранение нового значения регистра `PLLSET` из переменной.

5.3.3 Битовое поле `INTFB` регистра `PLLSET` необходимо установить в `1`:

```
void PLL_FeedbackSet(){
    CRU->PLLSET |= CRU_PLLSET_INTFB_Msk;
}
```

5.3.4 Включить сигнал RESET можно с помощью установки в `1` бита `RESET` регистра `PLLSET`:

```
void PLL_ResetSet(){
    CRU->PLLSET |= CRU_PLLSET_RESET_Msk;
}
void PLL_ResetRelease(){
    CRU->PLLSET &= ~CRU_PLLSET_RESET_Msk;
}
```

5.3.5 Включить сигнал POWER DOWN можно с помощью установки в `1` бита `PWRDN` регистра `PLLSET`:

```
void PLL_PowerDown(){
    CRU->PLLSET |= CRU_PLLSET_PWRDN_Msk;
}
void PLL_PowerUp(){
    CRU->PLLSET &= ~CRU_PLLSET_PWRDN_Msk;
}
```

5.3.6 Включить сигнал BYPASS возможно с помощью установки в `1` бита `BYPASS` регистра `PLLSET`:

```
void PLL_BypassModeEnable(){
CRU->PLLSET |= CRU_PLLSET_BYPASS_Msk;
}
```

Отключить сигнал BYPASS возможно с помощью установки в `0` бита `BYPASS` регистра `PLLSET`:

```
void PLL_BypassModeDisable(){
CRU->PLLSET &= ~CRU_PLLSET_BYPASS_Msk;
}
```

5.3.7 Состояние блока PLL отображается в битах регистра `SYSSR0`:

`PLL_RFSLIP` – если бит установлен в `1`, то выходная частота выше ожидаемой;

`PLL_FBSLIP` – если бит установлен в `1`, то выходная частота ниже ожидаемой;

`PLL_NLOCK` – если бит установлен в `1`, то PLL потерял возможность захвата сигнала.

Чтение состояния PLL:

```
uint32_t PLL_RFSlipGet(){
    return CRU->SYSSR0 & CRU_SYSSR0_PLL_RFSLIP_Msk;
}
uint32_t PLL_FBSlipGet(){
    return CRU->SYSSR0 &= CRU_SYSSR0_PLL_FBSLIP_Msk;
}
uint32_t PLL_NLockGet(){
    return CRU->SYSSR0 &= CRU_SYSSR0_PLL_NLOCK_Msk;
}
```

Биты `PLL_RFSLIP` и `PLL_FBSLIP` отображают состояние PLL на текущий момент. Бит `PLL_NLOCK_RSTN` регистра `PLLSET` управляет отслеживанием потери захвата сигнала.

Если `PLL_NLOCK_RSTN` установлен в `1` – отслеживание включено, если в `0` – отслеживание отключено и бит `PLL_NLOCK` будет иметь значение `0`, а сигнал PLL NLOCK будет сброшен.

По умолчанию отслеживание состояния блока PLL отключено.

Пример включения отслеживания, отключения отслеживания и сброса:

```
void PLL_NLockEnable(){
    CRU->PLLSET |= CRU_PLLSET_PLL_NLOCK_RSTN_Msk;
}
void PLL_NLockDisable(){
    CRU->PLLSET &= ~CRU_PLLSET_PLL_NLOCK_RSTN_Msk;
}
void PLL_NLockReset(){
    CRU->PLLSET &= ~CRU_PLLSET_PLL_NLOCK_RSTN_Msk;
    CRU->PLLSET |= CRU_PLLSET_PLL_NLOCK_RSTN_Msk;
}
```

5.3.8 Сигнал PLL NLOCK заведен на 61-ю линию маскируемых прерываний Core 0 и Core 1. Для сброса прерывания необходимо записать `0` в бит `PLL_NLOCK_RSTN` регистра PLLSET.

```
void PLL_Unlock_IrqTurnOn(){
    PLL_NLockEnable();                  // 1
    CLIC_SetCfgNlbits(0);
    CLIC_SetLevelThreshold(0);
    CLIC_SetLevel(CLIC_PLL_Unlock_IRQn, 1);
    CLIC_SetVectorMode(CLIC_PLL_Unlock_IRQn, CLIC_INTATTR_SHV_VECTORED);
    CLIC_SetTrigType(CLIC_PLL_Unlock_IRQn, CLIC_INTATTR_TRIG_TYPE_LEVEL);
    CLIC_SetTrigPolarity(CLIC_PLL_Unlock_IRQn, CLIC_INTATTR_TRIG_POL_P);
    CLIC_SetPrivMode(CLIC_PLL_Unlock_IRQn, CLIC_INTATTR_MODE_MACHINE);
    CLIC_EnableIRQ(CLIC_PLL_Unlock_IRQn);
    __enable_irq();
```

```
}
__attribute__((interrupt))
void PLL_Unlock_IRQHandler(){
    XCP_PROTO_GET(stdio)->printf("PLL Unlocked. Alarm!\n");  // 2
    PLL_NLockReset();    // 3
}
```

① настройка прерывания по выходу блока PLL из установившегося режима;

② вывод сообщения о прерывании в консоль, используется ROM XCP Protocol;

③ сброс флага прерывания PLL_NLOCK.

6 Домены тактирования подсистем микроконтроллера

6.1 Общая информация

6.1.1 Каждая подсистема тактирования (CCLK, PCLK0, PCLK1, PCLK2, HCLK, TCLK, CANx2CLK) состоит из:

- мультиплексора выбора источника синхросигнала;
- делителя частоты сигнала PLL.

6.2 Делитель частоты блока PLL для домена

6.2.1 До поступления на доменный мультиплексор синхросигнал PLL проходит через делитель частоты. Делитель позволяет уменьшить частоту PLL для каждого домена до 16 раз с шагом 0,5. Управление делителем осуществляется с помощью регистров `CLKCR0` и `CLKCR1`. На управление каждым делителем выделено по 6 бит в регистре: 5 бит отвечают за коэффициент деления и 1 бит - за включение делителя.

С помощью регистра CLKCR0 настраивается и включается делитель для доменов CCLK, PCLK0, PCLK1, PCLK2 и HCLK. Битовые поля для включения и настройки домена показаны в таблице 4.

Таблица 4 — Битовые поля для включения и настройки делителя для доменов CCLK, PCLK0, PCLK1, PCLK2 и HCLK

Домен	Бит настройки	Бит активации
CCLK	`CCLKDIV`	`CCLKDIVEN`
PCLK0	`PCLK0DIV`	`PCLK0DIVEN`
PCLK1	`PCLK1DIV`	`PCLK1DIVEN`
PCLK2	`PCLK2DIV`	`PCLK2DIVEN`
HCLK	`HCLKDIV`	`HCLKDIVEN`

С помощью регистра CLKCR1 настраивается делитель для доменов TCLK и CANx2CLK. Битовые поля для включения и настройки показаны в таблице 5.

Таблица 5 — Битовые поля для включения и настройки делителя доменов TCLK и CANx2CLK

Домен	Бит настройки	Бит активации
TCLK	`CCLKDIV`	`CCLKDIVEN`
CANx2CLK	`CCLKDIV`	`CCLKDIVEN`

Значение для настройки делителя рассчитывается по формуле:

$$DIV = (K_{div} - 1) \times 2,$$

где:

K_{div} – коэффициент деления частоты PLL,

DIV – значение, записываемое в битовое поле регистра.

```
#define CLK_DOMAIN_DIVIDER_MASK ((1 << 5) - 1)                  // 1
void ClkDomain_PLLDividerEnable(clk_domain_e domain){
    uint32_t *reg_ptr;
    uint32_t first_domain;
    if (domain < CLK_DOMAIN_TCLK){                              // 2
        reg_ptr = (uint32_t *)(&CRU->CLKCR0);
        first_domain = CLK_DOMAIN_CCLK;
    } else {
        reg_ptr = (uint32_t *)(&CRU->CLKCR1);
        first_domain = CLK_DOMAIN_TCLK;
    }
    uint32_t shift_size = (domain - first_domain) * 3 + 5;      // 3
    *reg_ptr |= 1 << shift_size;                                // 4
}
void ClkDomain_PLLDividerDisable(clk_domain_e domain){
    uint32_t *reg_ptr;
    uint32_t first_domain;
    if (domain < CLK_DOMAIN_TCLK){
        reg_ptr = (uint32_t *)(&CRU->CLKCR0);
        first_domain = CLK_DOMAIN_CCLK;
    } else {
        reg_ptr = (uint32_t *)(&CRU->CLKCR1);
        first_domain = CLK_DOMAIN_TCLK;
    }
    uint32_t shift_size = (domain - first_domain) * 3 + 5;
    *reg_ptr &= ~(1 << shift_size);
}
void ClkDomain_PLLDividerSet(clk_domain_e domain, float divider){
    uint32_t *reg_ptr;
    uint32_t first_domain;
    if (domain < CLK_DOMAIN_TCLK){
        reg_ptr = (uint32_t *)(&CRU->CLKCR0);
        first_domain = CLK_DOMAIN_CCLK;
    } else {
    reg_ptr = (uint32_t *)(&CRU->CLKCR1);
    first_domain = CLK_DOMAIN_TCLK;
    }
```

```
    uint32_t shift_size = (domain - first_domain) * 3;       // 5
    uint32_t clk_cr = *reg_ptr;
    clk_cr &= ~(CLK_DOMAIN_DIVIDER_MASK << shift_size);      // 6
    uint32_t div_val = (uint32_t)((divider + 0.25 - 1) * 2); // 7
    clk_cr |= div_val << shift_size;                         // 8
    *reg_ptr = clk_cr;
}
```

① маска для делителя частоты PLL для домена;

② определение в каком регистре находятся конфигурационные биты для делителя частоты PLL для домена;

③ расчет позиции бита, отвечающего за включение делителя частоты PLL для выбранного домена;

④ включение делителя частоты PLL для выбранного домена;

⑤ расчет позиции битов в регистре, отвечающее за значение делителя частоты PLL для выбранного домена;

⑥ сброс предыдущего значения делителя;

⑦ расчет нового значения делителя с округлением до 0,5 и преобразование данного значения в значение битов для записи в регистр;

⑧ запись нового значения делителя частоты PLL для выбранного домена в регистр.

6.3 Выбор источника тактового сигнала для домена

6.3.1 Мультиплексор в каждом домене позволяет выбрать источник синхросигнала: SYS CLK, ALT CLK или PLL. Управление мультиплексорами осуществляется с помощью битовых полей регистра `CLKSEL`:

CCLK_SEL – управление доменом CCLK;

PCLK0_SEL – управление доменом PCLK0;

PCLK1_SEL – управление доменом PCLK1;

PCLK2_SEL – управление доменом PCLK2;

HCLK_SEL – управление доменом HCLK;

TCLK_SEL – управление доменом TCLK;

CANX2CLKSEL – управление доменами CANx2CLK, CANCLK.

Выбор источника определяется следующими значениями:

`0b00` – синхросигнал SYS CLK;

`0b01` – выходной сигнал PLL, пропущенный через делитель;

`0b10` – синхросигнал ALT CLK;

`0b11` – зарезервировано.

```
typedef enum{         // 1
    CLK_SRC_C0 = 0,
    CLK_SRC_PLL = 1,
    CLK_SRC_C1 = 2,
    CLK_SRC_MASK = 3
}clk_src_e;
typedef enum{         // 2
    CLK_DOMAIN_CCLK = 1,
    CLK_DOMAIN_PCLK0 = 3,
    CLK_DOMAIN_PCLK1 = 5,
    CLK_DOMAIN_PCLK2 = 7,
    CLK_DOMAIN_HCLK = 9,
    CLK_DOMAIN_TCLK = 11,
    CLK_DOMAIN_CANX2CLK = 13
}clk_domain_e;
void ClkDomain_SourceSet(clk_domain_e domain, clk_src_e src){
    uint32_t clk_sel = CRU->CLKSEL;
    clk_sel &= ~(CLK_SRC_MASK << domain);
    clk_sel |= (src & CLK_SRC_MASK) << domain;
    CRU->CLKSEL = clk_sel;
}
```

① список источников тактового сигнала для доменов тактирования;

② список доменов тактирования микроконтроллера.

После снятия сигнала NRESET все домены тактируются от SYS CLK, кроме случая старта в режиме USB.

6.4 Управление тактированием периферии

6.4.1 Возможно раздельное управление подачей тактового сигнала каждого периферийного модуля в частотных доменах PCLK0, PCLK1 и PCLK2. Для этого используются регистры `PCLK0EN`, `PCLK1EN` и `PCLK2EN`. Даже если модуль не тактируется, его регистровое пространство доступно для обращения к нему.

Периферия в доменах PCLK0 и PCLK1 идентична и регистры `PCLK0EN` и `PCLK1EN` имеют одинаковые битовые поля.

Битовые поля `[устройство]CLKEN` отвечают за включение тактирования:

- QSPI0 и QSPI1 – `QSPI0CLKEN` и `QSPI1CLKEN`;
- SPI0/1 и SPI2/3 – `SPI[0/1]CLKEN` и `SPI[0/1]CLKEN`;
- I2C0/1 и I2C2/3 – `I2C[0/1]CLKEN` и `I2C[2/3]CLKEN`;
- TIMER0 и TIMER1 – `TIMER0CLKEN` и `TIMER1CLKEN`;
- WDT0 и WDT1 – `WDT0CLKEN` и `WDT1CLKEN`;
- PWMG0 и PWMG1 – `PWMG0CLKEN` и `PWMG1CLKEN`;
- CAN0 и CAN1 – `CAN0CLKEN` и `CAN1CLKEN`.

6.4.2 При помощи регистра `PCLK2EN` можно настроить тактирование следующей периферии:

ADC0 – `ADC0CLKEN`;

ADC1 – `ADC1CLKEN`;

ADC2 – `ADC2CLKEN`;

PWMA0 – `PWMA0CLKEN`;

PWMA1 – `PWMA1CLKEN`;

PWMA2 – `PWMA2CLKEN`;

PWMA3 – `PWMA3CLKEN`.

```
typedef enum{              // 1
    PERIPH_QSPI0 = 0,
    PERIPH_SPI0 = 1,
    PERIPH_SPI1 = 2,
    PERIPH_UART0 = 3,
    PERIPH_UART1 = 4,
    PERIPH_UART2 = 5,
    PERIPH_I2C0 = 6,
    PERIPH_I2C1 = 7,
    PERIPH_I2S0 = 8,
    PERIPH_TIMER0 = 9,
    PERIPH_WDT0 = 10,
    PERIPH_GPIO0 = 11,
    PERIPH_PWMG0 = 12,
    PERIPH_CAN0 = 13,
    PERIPH_QSPI1 = 32,
    PERIPH_SPI2 = 33,
    PERIPH_SPI3 = 34,
    PERIPH_UART3 = 35,
    PERIPH_UART4 = 36,
    PERIPH_UART5 = 37,
```

```
    PERIPH_I2C2 = 38,
    PERIPH_I2C3 = 39,
    PERIPH_I2S1 = 40,
    PERIPH_TIMER1 = 41,
    PERIPH_WDT1 = 42,
    PERIPH_GPIO1 = 43,
    PERIPH_PWMG1 = 44,
    PERIPH_CAN1 = 45,
    PERIPH_ADC0 = 64,
    PERIPH_ADC1 = 65,
    PERIPH_ADC2 = 66,
    PERIPH_PWMA0 = 67,
    PERIPH_PWMA1 = 68,
    PERIPH_PWMA2 = 69,
    PERIPH_PWMA3 = 70,
    PERIPH_GPIO2 = 71,
    PERIPH_UART6 = 72,
    PERIPH_UART7 = 73,
}periph_e;
void PERIPH_ClockEnable(periph_e periph){
    switch(periph){
        case PERIPH_UART0:
        case PERIPH_UART1:
        case PERIPH_UART3:
        case PERIPH_UART4:
        case PERIPH_UART5:
        case PERIPH_UART6:
        case PERIPH_UART7:
        case PERIPH_I2S0:
        case PERIPH_I2S1:
        case PERIPH_GPIO0:
        case PERIPH_GPIO1:
        case PERIPH_GPIO2:
            break;              // 2
        default: {
            uint32_t * pclk_en_regs = (uint32_t *)(&CRU->PCLK0EN);
            pclk_en_regs[periph / 32] |= (1 << (periph % 32));
            break;
        }
    }
}
void PERIPH_ClockDisable(periph_e periph){
    switch(periph){
```

```
        case PERIPH_UART0:
        case PERIPH_UART1:
        case PERIPH_UART3:
        case PERIPH_UART4:
        case PERIPH_UART5:
        case PERIPH_UART6:
        case PERIPH_UART7:
        case PERIPH_I2S0:
        case PERIPH_I2S1:
        case PERIPH_GPIO0:
        case PERIPH_GPIO1:
        case PERIPH_GPIO2:
            break;
        default: {
            uint32_t * pclk_en_regs = (uint32_t *)(&CRU->PCLK0EN);
            pclk_en_regs[periph / 32] &= ~(1 << (periph % 32));  // 3
            break;
        }
    }
}
```

① список периферии микроконтроллера на шинах PCLK;

② у перечисленной периферии отсутствует возможность управлять тактовым сигналом;

③ поскольку регистры управления тактированием и сбросом периферии располагаются друг за другом, то обращаться к ним можно как к массиву.

6.5 Сброс

6.5.1 Для некоторых периферийных модулей предусмотрен сигнал программного сброса. Сброс осуществляется при помощи регистров `PCLK0EN`, `PCLK1EN`, и `PCLK2EN`. Установка сброса – `0`, снятие сброса – `1`.

Для сброса периферии используются следующие битовые поля регистров `PCLK0EN`, `PCLK1EN`:

QSPI0 и QSPI1 – `QSPI0RSTN` и `QSPI1RSTN`;

SPI0/1 и SPI2/3 – `SPI[0/1]RSTN` и `SPI[0/1]RSTN`;

I2S0 и I2S1 – `I2S0RSTN` и `I2S1RSTN`;

I2C0/1 и I2C2/3 – `I2C[0/1]RSTN` и `I2C[2/3]RSTN`;

UART0/1/2 и UART3/4/5 – `UART[0/1/2]RSTN` и `UART[3/4/5]RSTN`;

TIMER0 и TIMER1 – `TIMER0RSTN` и `TIMER1RSTN`;

WDT0 и WDT1 – `WDT0RSTN` и `WDT1RSTN`;

PWMG0 и PWMG1 – `PWMG0RSTN` и `PWMG1RSTN`;

CAN0 и CAN1 – `CAN0RSTN` и `CAN1RSTN`;

GPIO0 и GPIO1 – `GPIO0RSTN` и `GPIO1RSTN`,

а также битовые поля регистра `PCLK2EN`:

ADC0/1/2 – `ADC[0/1/2]RSTN`;

PWMA0/1/2/3 – `PWMA[0/1/2/3]RST`;

GPIO2 – `GPIO2RSTN`;

UART6 – `UART[6/7]RSTN`.

После сброса тактирование всех модулей периферии включено, сигнал сброса неактивен:

```
void PERIPH_ResetRelease (periph_e periph){
    uint32_t * pclk_en_regs = (uint32_t *)(&CRU->PCLK0EN);
    pclk_en_regs[periph / 32] |= (1 << (periph % 32 + 16)); // 1
}
void PERIPH_ResetSet(periph_e periph){
    uint32_t * pclk_en_regs = (uint32_t *)(&CRU->PCLK0EN);
    pclk_en_regs[periph / 32] &= ~(1 << (periph % 32 + 16));
}
```

① Т.к. регистры управления тактированием и сбросом периферии располагаются друг за другом, то можно обращаться к ним как к массиву. Управление сбросом доступно для всей периферии. Расположение битов управления сбросом периферии идентично расположению битов управления синхросигналом периферии, но смещены на 16 бит.

7 Микросекундный таймер ядра

7.1.1 В каждом ядре (Core 0, Core 1 и Core 2) имеется 64-разрядный микросекундный таймер, который считает микросекунды с момента старта ядра. Для таймера Core 0 и Core 1 тактовый сигнал формируется из сигнала CCLK путем пропускания его через делитель частоты, для таймера ядра Core 2 тактовый сигнал формируется из сигнала HCLK, прошедшего через делитель частоты. Эти делители целочисленные и конфигурируются с помощью битовых полей `CCLK1MHZ` и `HCLK1MHZ` регистра `CLKSEL`.

Значения этих полей рассчитываются по формулам:

$$CCLK1MHZ = \frac{F_{cclk}}{10^6} - 1,$$

$$HCLK1MHZ = \frac{F_{hclk}}{10^6} - 1,$$

где:

F_{cclk} – частота шины CCLK в Гц,

F_{hclk} – частота шины HCLK в Гц.

```
void CCLK_1MHzDividerSet(uint32_t divider){
    uint32_t clk_sel = CRU->CLKSEL;
    clk_sel &= ~(CRU_CLKSEL_CCLK1MHZ_Msk);    // 1
    clk_sel |= ((divider - 1) << CRU_CLKSEL_CCLK1MHZ_Pos) &
    CRU_CLKSEL_CCLK1MHZ_Msk;                  // 2
    CRU->CLKSEL = clk_sel;
}
void HCLK_1MHzDividerSet(uint32_t divider){
    uint32_t clk_sel = CRU->CLKSEL;
    clk_sel &= ~(CRU_CLKSEL_HCLK1MHZ_Msk);
    clk_sel |= ((divider - 1) << CRU_CLKSEL_HCLK1MHZ_Pos) &
    CRU_CLKSEL_CCLK1MHZ_Msk;
    CRU->CLKSEL = clk_sel;
}
```

① сброс предыдущего значения делителя для 1 МГц таймера;

② запись нового значения делителя в регистр.

7.1.2 Чтение значения микросекундного таймера ядра из регистров `MTIMEL` и `MTIMEH`:

```
uint64_t CoreTimer_UsGet(){
    volatile uint32_t msb, lsb, msb2;
    do
    {
        msb = *(volatile uint32_t *)CLINT_MTIMEH;    // 1
        lsb = *(volatile uint32_t *)CLINT_MTIMEL;    // 2
        msb2 = *(volatile uint32_t *)CLINT_MTIMEH;   // 3
    } while (msb != msb2);                           // 4
    return ((((uint64_t)msb) << 32) | lsb);
}
```

① считывание старших 32 бит таймера;

② считывание младших 32 бит таймера;

③ повторное считывание старших 32 бит таймера;

④ в случае, если значение старших 32 бит таймера, считанных в первом и во втором случае, отличаются (произошло переполнение младших 32 бит таймера) повторить процедуру считывания.

8 Примеры настройки системы тактирования

8.1 Общая информация

8.1.1 В этом разделе будет приведено два варианта настройки системы тактирования:

а) все домены работают от внешнего синхросигнала *CLKI*;

б) все домены работают от PLL и настроены на максимально допустимые спецификацией частоты.

8.2 Конфигурация тактирования от SYS CLK

```
clk_domain_e domains[] = {      // 1
CLK_DOMAIN_CCLK, CLK_DOMAIN_PCLK0, CLK_DOMAIN_PCLK1, CLK_DOMAIN_PCLK2,
CLK_DOMAIN_HCLK, CLK_DOMAIN_TCLK, CLK_DOMAIN_CANX2CLK
};
int System_ConfigCLKI(){
21
for (uint32_t i = 0; i < sizeof(domains) / sizeof(domains[0]); i++){ // 2
ClkDomain_SourceSet(domains[i], CLK_SRC_C0);
}
CCLK_1MHzDividerSet(25);        // 3
HCLK_1MHzDividerSet(25);        // 4
PLL_NLockDisable();             // 5
PLL_ResetSet();                 // 6
PLL_PowerDown();                // 7
return 0;
}
```

① Список всех доменов тактирования микроконтроллера

② Настройка тактирования каждого домена тактирования от CLKI

③ Установка значения делителя для 1 МГц таймера для Core 0 и Core 1 (при условии, что частота внешнего источника синхросигнала 25 МГц)

④ Установка значения делителя для 1 МГц таймера для Core 2 (при условии, что частота внешнего источника синхросигнала 25 МГц)

⑤ Остановка PLL: отключение схемы слежения за выходом PLL из установившегося режима

⑥ Остановка PLL: установка сигнала RESET для PLL

⑦ Остановка PLL: отключение питания аналоговой схемы PLL

8.3 Конфигурация тактирования от PLL на максимальные частоты

```
#define PLL_TIMEOUT_US 20       // 1
int System_ConfigMaxFreqs(){
    System_ConfigCLKI();        // 2
    PLL_TestModeDisable();      // 3
    PLL_BypassModeDisable();    // 4
    PLL_DividersSet(1, 16, 2);  // 5
    PLL_PowerUp();              // 6
    PLL_ResetRelease();         // 7
    uint64_t pll_start = CoreTimer_UsGet(); // 8
    while(CRU->SYSSR0 & (CRU_SYSSR0_PLL_RFSLIP_Msk | CRU_SYSSR0_PLL_FBSLIP_Msk)){
       if (CoreTimer_UsGet() - pll_start > PLL_TIMEOUT_US){
           PLL_ResetSet();
           PLL_PowerDown();
           return -1;
       }
    }
    CCLK_1MHzDividerSet(200);                       // 9
    ClkDomain_PLLDividerSet(domains[0], 1);         // 10
    ClkDomain_PLLDividerEnable(domains[0]);         // 11
    ClkDomain_SourceSet(domains[0], CLK_SRC_PLL); // 12
    HCLK_1MHzDividerSet(100);                       // 13
    for (uint32_t i = 1; i < sizeof(domains) / sizeof(domains[0]); i++){
        ClkDomain_PLLDividerSet(domains[i], 2);   // 14
        ClkDomain_PLLDividerEnable(domains[i]);   // 15
        ClkDomain_SourceSet(domains[i], CLK_SRC_PLL); // 16
    }
    PLL_Unlock_IrqTurnOn();                             // 17
    return 0;
}
```

① таймаут для выхода PLL в установившийся режим;

② переход на тактирование от CLKI;

③ конфигурирование PLL: отключение тестового режима;

④ конфигурирование PLL: отключение режима передачи опорного сигнала;

⑤ конфигурирование PLL: настройка делителей – делитель опорной частоты в 1, делитель частоты в цепи обратной связи в 16, делитель выходной частоты в 2. Таким образом частота VCO равна 400 МГц, выходная частота PLL – 200 МГц;

⑥ конфигурирование PLL: включение питания аналоговой схемы;

⑦ конфигурирование PLL: снятие сигнала RESET;

⑧ ожидание выхода PLL в установившийся режим. В случае не установки данного режима – выход из функции с ошибкой;

⑨ настройка CCLK на тактирование от PLL: настройка делителя частоты для 1 МГц таймера Core 0 и Core 1;

⑩ настройка CCLK на тактирование от PLL: установка делителя частоты PLL для домена CCLK в 1;

⑪ настройка CCLK на тактирование от PLL: включение делителя частоты PLL для домена CCLK;

⑫ настройка CCLK на тактирование от PLL: выбор сигнала PLL в качестве тактового сигнала для домена CCLK, домен CCLK тактируется с частотой PLL (200 МГц);

⑬ настройка остальных доменов на тактирование от PLL: настройка делителя частоты для 1 МГц таймера Core 2;

⑭ настройка остальных доменов на тактирование от PLL: установка делителя частоты PLL в 2;

⑮ настройка остальных доменов на тактирование от PLL: включение делителя частоты PLL;

⑯ настройка остальных доменов на тактирование от PLL: выбор сигнала PLL в качестве тактового сигнала, домены PCLK0, PCLK1, PCLK2, HCLK, TCLK и CANX2CLK тактируются с частотой в 2 раза меньшей частоты PLL - 100 МГц;

⑰ включение прерывания по выходу PLL из установившегося режима.

История изменений

Версия	Дата	Описание
1.0	29.04.2026	Начальная версия
1.2	13.05.2026	Корректировки и уточнения
3	04.06.2026	Корректировки и уточнения
4	18.06.2026	Корректировки и уточнения
5	22.06.2026	Корректировки и уточнения
6	06.07.2026	Корректировки и уточнения
6.1	06.10.2026	Корректировка блока кода «Чтение состояния PLL». Исправлена техническая ошибка в названиях битов сброса ADC регистра PCLK2EN. В подразделе 6.2 списки заменены таблицами 4 и 5.
6.2	05.11.2026	Исправление ошибки в формуле в п. 5.2.2.

Контактные данные

Официальный сайт
www.baikalelectronics.ru

Главный офис
www.baikalelectronics.ru/contacts

Электронная почта
info@baikalelectronics.ru

Телефон
+7 495 221-39-47