Table of contents:

Общая информация
- Графическое представление
- Общее описание

Основные характеристики
- Электрические параметры микросхемы
  - Таблица 1. Электрические характеристики (температурный диапазон от – 60°C до +125°C)
- Электростатическая защита
- Предельно-допустимые и предельные режимы эксплуатации

Порты
- Распиновка
  - Таблица выводов

Рекомендуемая схема применения

Выбор схемы питания микросхемы

Типы датчиков
- Датчики типа СКВТ (Resolver)
- Датчики типа сельсин (Synchro)
- Датчики типа ЛРДТ (LVDT)

Архитектура и принцип работы
- Принцип преобразования сигналов СКВТ в код угла
- Архитектура микросхемы

Программируемый генератор
- Программируемый генератор
  - Режимы работы
- Синусоидальный сигнал (EXO_mode = 10)
  - Частота
  - Амплитуда
  - Опорное напряжение ЦАП (REFDAC)
  - Фильтрация выходного сигнала (ФНЧ)
- Меандр (EXO_mode = 01)
- Постоянный уровень (EXO_mode = 11)
- Краткая шпаргалка по настройке

Входные группы преобразователя
  - Включение
  - Компенсатор смещения

Блок масштабирования и преобразования координат

- Коррекция амплитуды: формулы и расчёт коэффициентов

Блок восстановления сигнала опорной частоты

- Настройка и анализ работы блока

Следящий контур

Блок модели датчика

- Модель датчика в режиме СКВТ/сельсин
- Модель ЛРДТ по 5-проводной схеме

Полоса пропускания контура

- Полоса пропускания преобразователя

Блок обработки результатов

Эмуляция квадратурного энкодера

- Эмуляция квадратурного энкодера

Блок определения ошибок подключения

- Регистр состояния и маски
- Условия срабатывания флагов
  - CLIP_SIN, CLIP_COS — переполнение АЦП
  - ADC_OVF — большая постоянная составляющая
  - CORR_OVF — переполнение после коррекции усиления
  - UIN_HIGH, UIN_LOW — пороги амплитуды
  - EX_PH_OUTRANGE — большой сдвиг фазы опоры
  - MISS_EXREF — отсутствие опорного сигнала
  - NLock — ошибка слежения
  - C_LOOP_OVF — переполнение интегратора контура
- Численные значения и расчёт порогов амплитуды
- Диагностика при настройке (на примере канала 1)

Интерфейс SPI

- Описание структуры SPI
- Коды операций
- Временные параметры последовательного интерфейса
- Дополнительные режимы SPI
  - Фиксация значения выходной ячейки для множественного чтения.
  - Чтение ранее переданной транзакции
  - Блокировка записи слов
  - Программный адрес
  - Управление внешним приемо-передатчиком с помощью вывода VC
- Режим параллельной передачи результата
- Назначение выводов в режиме параллельной передачи результата

Тактирование
- Цепочка тактирования
- Выбор источника тактирования
- Настройка ФАПЧ (PLL)
- Настройка частоты АЦП
  - Частоты, участвующие в тактировании АЦП
  - Формирование Fclk_adc
  - Условие Fclk_adc ≤ 2,5 МГц
  - Примеры конфигурации
- Выключение тактирования

Постоянное запоминающее устройство
- Структура
  - ПЗУ с параллельным доступом
  - ПЗУ с последовательным доступом
  - Порядок программирования ПЗУ

Подсистема сброса
- Таблица этапов распространения сигнала сброс

Микровычислители
- Архитектура конвейера
- Регистры общего назначения
- Структура памяти
  - Ячейки обмена данными CPU
  - Ячейки обмена с интерфейсом SPI
  - Буферы
  - Адресация памяти
- Форматы данных
  - 28-битный формат с плавающей запятой
    - Структура формата
    - Кодирование (Float32 → Float28)
    - Float28 → Float32
    - Float28 → Integer
- Система команд
  - Арифметико-логические команды
  - Команды сдвига
  - Команды умножения и преобразования форматов
  - Команды работы с памятью
  - Индексные команды работы с памятью

- Команды условного перехода
- CORDIC-команды
- Канальные регистры
- Команды управления HAND-интерфейсом
- Команды управления и синхронизации
- Взаимодействие с преобразователем
  - Запись данных в преобразователь
  - Запись Coord/Vel
  - Запись Pole_addi
- Синхронизация двух CPU
- Отладчик
  - Управление выполнением
  - Чтение регистров
- Пример программы
  - CPU1: Коррекция угла по 1-й гармонике
- Сводная таблица команд

Регистры конфигурации

- Таблица адресов регистров
  - C1KampS/C2KampS
  - C1KampC/C2KampC
  - C1KbiasS/C2KbiasS
  - C1KbiasC/C2KbiasC
  - C1fbias/C2fbias
  - C1ExPhShft/C2ExPhShft
  - C1ExoStngs/C2ExoStngs
    - Поля регистра
  - C1EXInc/C2EXInc
  - C1Amp_th/C2Amp_th
    - Поля регистра
  - C1InputStngs/C2InputStngs
    - Поля регистра
  - C1Lock_th/C2Lock_th
  - C1Zero/C2Zero
  - C1Mask/C2Mask
    - Поля регистра
  - C1KonturStngs/C2KonturStngs
    - Поля регистра

- C1ResCntrl/C2ResCntrl
  - Поля регистра
- C1Vcnt_bound/C2Vcnt_bound
- C1Coord/C2Coord
- C1CoordHB/C2CoordHB
- C1AdcS/C2AdcS
  - Поля регистра
- C1AdcC/C2AdcC
  - Поля регистра
- C1OutS/C2OutS
  - Поля регистра
- C1VirtualS/C2VirtualS
  - Поля регистра
- C1Err_metric/C2Err_metric
- C1Amp_metric/C2Amp_metric
- C1Vel/C2Vel
- C1VelHB/C2VelHB
- C1PhiS/C2PhiS
  - Поля регистра
- C1PhiC/C2PhiC
  - Поля регистра
- C1OutC/C2OutC
  - Поля регистра
- C1VirtualC/C2VirtualC
  - Поля регистра
- C1Stat/C2Stat
  - Поля регистра
- C1Pole_addi/C2Pole_addi
- IC_addr
- ADC_config
  - Поля регистра
- Mask_Stat
  - Поля регистра
- Flags_delay
- WR_lock
- CMP_lth
- AFE_config

    - Поля регистра
- Mode_config
    - Поля регистра
- NOCLK_stat
    - Поля регистра
- SPI_req
- alive_cnt
- Stat_main
    - Поля регистра
- Dcpu1LB
- Dcpu1HB
- Dcpu2LB
- Dcpu2HB
- PLL_config
    - Поля регистра
- INIT_conf
    - Поля регистра
- Значения, при которых Fclk_rom превышает 1 МГц, использовать нельзя: при FINT = 10 МГц допустимы BOTP_clkdel ≥ 3, при FINT = 60 МГц — BOTP_clkdel ≥ 11 (N = 64, Fclk_rom ≈ 938 кГц).
- UOTP_ctrl
    - Поля регистра
- BUS_addr
- BOTP_addr
- BOTP_data
- BOTP_ctrl
    - Поля регистра
- BOTP_out
- P1BG_ctrl/P2BG_ctrl
    - Поля регистра
- P1BG_data/P2BG_data
    - Поля регистра

Методика подключения и настройки

- 1. Проверка подключения
- 2. Обнаружение микросхемы и проверка SPI
- 3. Настройка тактовой частоты микросхемы
- 4. Настройка опорных уровней аналоговой части
- 5. Настройка сигнала возбуждения датчика (на примере канала 1)

- 6. Настройка входных каскадов и проверка оцифровки сигналов АЦП (на примере канала 1)
  - 6.а Входные усилители: внешние или встроенные
  - 6.б Параметры аналогового сигнала на входах АЦП
  - 6.в Проверка оцифровки: чтение отсчётов АЦП по SPI
- 7. Выбор режима преобразования и настройка опорного сигнала
  - 7.а Выбор режима преобразования (Mode)
  - 7.б Источник опорного сигнала
  - 7.в Фазирование опорного сигнала по цифровым отсчётам АЦП
  - 7.г Блок восстановления опорной частоты и цифровая настройка фазы
- 8. Калибровка коэффициента усиления входных сигналов и смещения сигналов модели датчика
  - 8.а Контрольные сигналы
  - 8.б Настройка амплитуды (размаха): KampS, KampC
  - 8.в Настройка смещения сигналов модели: KbiasS, KbiasC
  - 8.г Порядок настройки
- 12. Почему микросхема не отвечает (проверка на уровне SPI)
  - Адресация микросхем на общей шине (IC_addr)
- 14. Блокировка записи и инициализация из ПЗУ

Демонстрационное ПО

- 📄Описание ПО
- 📄Terminal
- 📄Окно Тест
- 📄Анализатор HandTap
- 📄Энкодер
- 📄GraphView
- 📄Режим детектора
- 📄Отладчик CPU
- 📄ParallelSpiView (параллельный вывод)
- 📄BatchWriter (пакетная запись)
- 📄Record (запись данных)
- 📄Программирование OTP

Описание ПО Workstation4ad3s

- Варианты аппаратной части и запуск
  - Вариант 1 — готовая демонстрационная плата
  - Вариант 2 — своя плата с микросхемой 5400TP065A-022 + модуль ESP32-S3
  - Запуск и подключение
- Быстрый старт

- Структура окна
  - Левая панель
  - Рабочая область (вкладки)
  - Консоль вывода
  - Сохранение layout
- Меню
  - File
  - Options
  - Help
- Состав ПО
  - Terminal
  - GraphView
  - Анализатор HandTap
  - ParallelSpiView (параллельный вывод)
  - Отладчик CPU
  - Окно Тест
  - Энкодер
  - Программирование OTP

Вкладка Terminal

- Структура вкладки
- Таблица регистров IC
  - Контекстное меню
- Основные регистры (вкладки «1» и «2»)
  - Вкладка «1»
  - Вкладка «2»
- Регистры контуров отслеживания (C1 / C2)
  - Колонка 1 (левая)
  - Колонка 2 (центральная)
  - Колонка 3 (правая)
- Взаимодействие элементов
  - Чтение и запись отдельной ячейки
- Actions (кнопки управления)
  - Загрузка и сохранение
  - Обмен с микросхемой
  - Регистры по адресу
  - Управление CPU и конвертерами
  - Программирование OTP

    - Управление питанием
- Типовой порядок работы

Окно Тест

- Когда микросхема не отвечает

Анализатор HandTap

- Расположение элементов
- Органы управления
- Включение и выключение режима
    - Включение (ON)
    - Выключение (OFF)
- Процесс захвата данных
- Адреса блоков данных
- Режим синхронизации (Trig)
- Типичное использование
- Программа CPU1 режима HandTap
    - Карта памяти CPU1
    - Алгоритм работы программы
        - 1. Ожидание команды SPI
        - 2. Декодирование параметров
        - 3. Ожидание синхронизации (режим Trig)
        - 4. Захват первого блока (64 отсчёта)
        - 5. Захват второго блока (128 отсчётов)
        - 6. Сигнализация готовности
    - Взаимодействие CPU1, МК и ПК — полная диаграмма

Энкодер

- Принцип работы
- Начальное состояние и сброс нуля
- Окно Encoder
    - Текущие значения
    - Настройки каналов
- Включение и выключение
    - Start Reading
    - Stop Reading
- Сохранение настроек
- GPIO-подключение

GraphView

- Расположение элементов

- Органы управления
  - Настройка каналов (4 строки)
  - Управление графиками
- Сохранение и восстановление настроек
- Чтение данных
- Адреса по умолчанию
  - Обычный режим
  - Режим детектора H1
  - Режим детектора H2
- Режим детектора

Режим детектора

- Включение и выключение
- Предустановки адресов
  - Канал H1
  - Канал H2
- Настройка амплитуды
  - До настройки
  - Подбор коэффициентов усиления
  - После настройки
- Настройка смещения
- Порядок настройки
- Программа CPU1 режима детектора
  - Принцип работы
  - Алгоритм программы
  - Как работает распаковка данных
  - Карта выходных данных в разделяемой RAM
- Условие корректной работы: синхронизация EX_REF

Отладчик CPU

- Расположение элементов
- Кнопки управления
- Точки останова (Breakpoints)
  - Установка точек останова мышью
  - Принцип работы точек останова
- Регистры CPU
  - Основные регистры
  - Системные регистры и каналы
- Память и буферы

    - Память данных CPU (правая колонка)
    - Буферы BUF1 и BUF2 (нижняя панель)
- Связь с редактором ассемблера (Asm Editor)
- Типовой порядок работы

ParallelSpiView (параллельный вывод)

- Принцип работы
- Расположение элементов
- Органы управления
- Включение и выключение режима
    - Включение (ON)
    - Выключение (OFF)
- Процесс захвата данных
    - Декодирование данных
- Адреса блоков данных
- Режим синхронизации (Trig)
- Типичное использование
- Программа CPU1 режима параллельной выдачи
    - Описание алгоритма
    - Карта памяти CPU1
    - Состав файлов проекта dma_asm
- Взаимодействие CPU1, МК и ПК — полная диаграмма

BatchWriter (пакетная запись)

- Расположение элементов
    - Столбцы таблицы
- Органы управления
- Типичное использование
- Формат данных
- Автосохранение

Record (запись данных)

- Расположение элементов
- Панель управления
- Таблица адресов
    - Значения по умолчанию
- Формат выходного файла
    - Объединение данных
- Взаимодействие с ESP32
- Типичное использование

Программирование OTP-памяти

- Полный сценарий программирования
  - Предварительные условия
- Программирование BOTP (слепок регистров)
- Программирование UOTP (PLL_config, INIT_conf)
- Справочник кнопок прожига
- Если микросхема перестала отвечать
- Связанные ресурсы

Общая информация

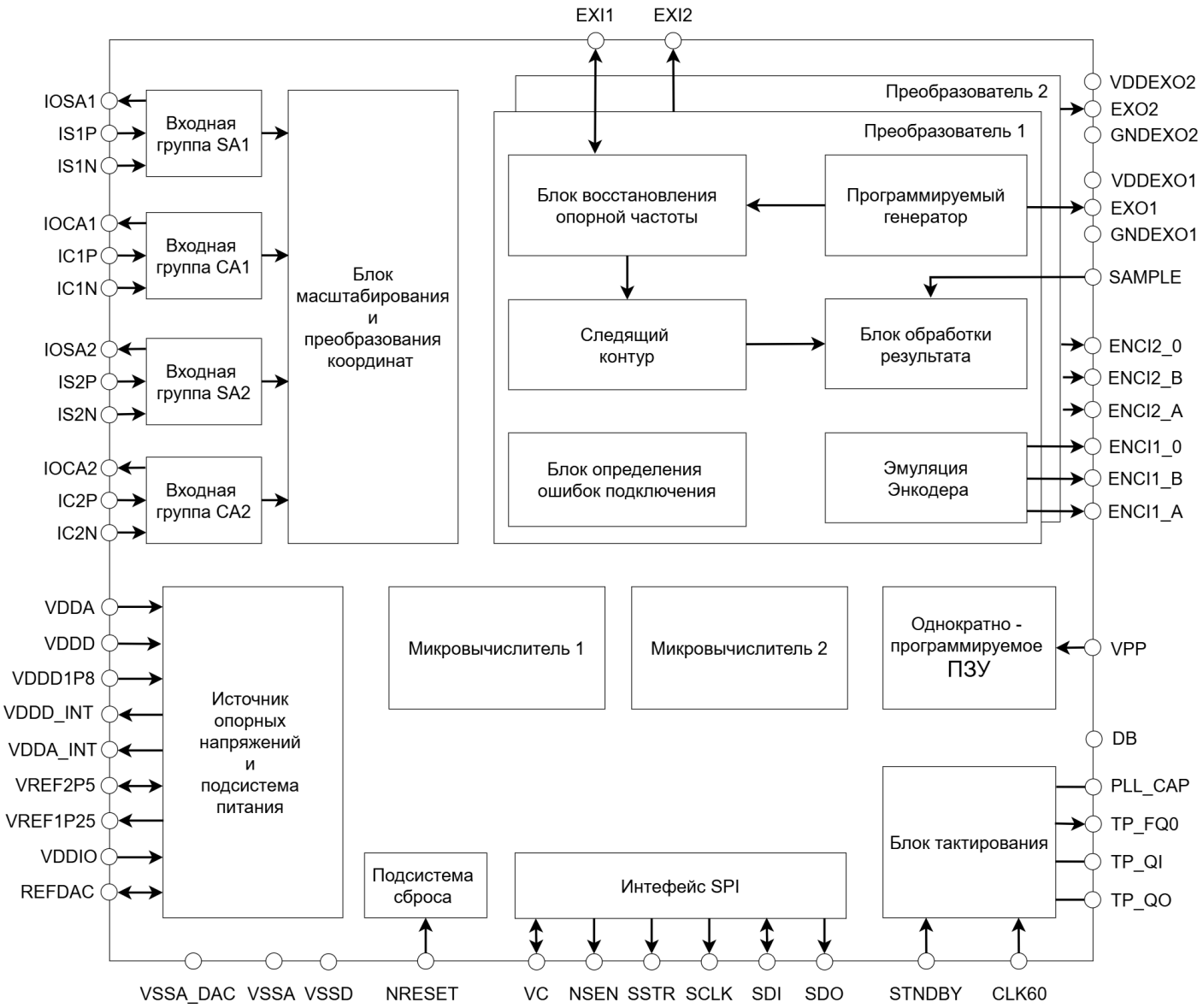
5400TP065A-022 – Микросхема преобразователя сигналов с датчиков типа сельсин, СКВТ и ЛРДТ в цифровой код угла

ДОКУМЕНТАЦИЯ

Скачать pdf версию документации datasheet_5400TP065A-022(03.07.2026).pdf

- Напряжения питания 5,0 В;
- Период обновления информации не более 6,7 мкс;
- Частота возбуждения датчиков от 0 до 30 кГц;
- Разрядность выходной информации от 8 бит до 16 бит;
- Два независимых преобразователя;
- Два независимых генератора опорных сигналов с частотой от 20 Гц до 30 кГц;
- Эмуляция квадратурного энкодера;
- Последовательный (SPI) интерфейс;
- Температурный диапазон от –60°С до +125°С.

Графическое представление



Блок схема

Общее описание

Микросхема предназначена для преобразования сигналов с датчиков типа сельсин, СКВТ (синусно-косинусный вращающийся трансформатор) и датчиков линейного перемещения – ЛРДТ (линейный регулируемый дифференциальный трансформатор) (см. раздел типы датчиков). Микросхема в своем составе содержит программируемый генератор возбуждающего напряжения и два следящих контура, производящих вычисление угла поворота вала или перемещения сердечника ЛРДТ. Микросхема выполнена в 64-выводном металлокерамическом корпусе МК 5153.64-3.

*Последнее обновление **3 июл. 2026 г.***

Основные характеристики

Электрические параметры микросхемы

Таблица 1. Электрические характеристики (температурный диапазон от – 60°С до +125°С)

Параметр, единица измерения	Букв. обозн.	Не менее	Не более	Номер выв.
Статический ток потребления в режиме покоя, мА	$I_{ПОТ1}$	—	10	33, 63
Ток утечки высокого уровня по цифровым выводам[2], мкА	$I_{1УТ}$	—	10	52, 53, 55-59, 62
Ток утечки низкого уровня по цифровым выводам[2], мкА	$I_{0УТ}$	—	10	52, 53, 55-59, 62
Динамический ток потребления, мА	$I_{ПОТ2}$	—	200	33, 63
Время задержки, нс	$t_{ЗД}$	—	80	51, 55
Выходное напряжение высокого уровня по цифровым выводам[3], В	$U_{1ВЫХ}$	2,4	—	41-44, 46-51, 53
Выходное напряжение низкого уровня по цифровым выводам[4], В	$U_{0ВЫХ}$	—	0,4	41-44, 46-51, 53, 62
Выходное напряжение ИОН для формирования опорного напряжения АЦП, В	$U_{ВЫХ_ОП}$	2,25	2,75	11

Параметр, единица измерения	Букв. обозн.	Не менее	Не более	Номер выв.
Выходное напряжение ИОН схемы смещения постоянной составляющей входных сигналов, В	$U_{\text{ВЫХ_СМ}}$	1,125	1,375	12
Выходное напряжение встроенного линейного регулятора 4 В питания цифровой части, В	$U_{\text{ЛР_ЦИФ}}$	3,7	4,3	31
Выходное напряжение встроенного линейного регулятора 4 В питания аналоговой части, В	$U_{\text{ЛР_АН}}$	3,7	4,3	30
Выходное напряжение встроенного линейного регулятора 1,8 В питания цифрового ядра, В	$U_{\text{ЛР_1.8}}$	1,62	1,98	36, 61
Максимальное выходное напряжение по выводу EXO1[5], В	$U_{\text{МАКС1}}$	$U_{\text{ВХ_БУФ1}} - 0,7$	—	6
Максимальное выходное напряжение по выводу EXO2[5], В	$U_{\text{МАКС2}}$	$U_{\text{ВХ_БУФ2}} - 0,7$	—	3
Минимальная частота выходного сигнала (сигнала опорной частоты), основная гармоника[5], Гц	$f_{\text{МИН}}$	—	20	3, 6
Минимальное выходное напряжение по выводам EXO1, EXO2[5], В	$U_{\text{МИН}}$	—	0,7	3, 6
Максимальная частота выходного сигнала (сигнала опорной частоты), основная гармоника[5], кГц	$f_{\text{МАКС}}$	30	—	3, 6

Параметр, единица измерения	Букв. обозн.	Не менее	Не более	Номер выв.
Входной ток на выводах VREF2P5 в режиме подключения внешнего источника напряжения 2,55 В, мА	$I_{\text{ВХ}}$	—	1	11
Полная погрешность определения угла при преобразовании сигналов датчика типа СКВТ при F=1,5 кГц[6], EMP	E_{A}	-20	20	21, 24, 50, 18, 15, 49

[1] Допускается уточнять нормы на параметры до проведения испытаний. [2] Выводы: CLK60, EXI2, EXI1, SDI, VC, SCLK, nSEN, SSTR, STNDBY, SAMPLE. [3] Выводы: EXI1, EXI2, SDO, VC, ENC1_0, ENC1_A, ENC1_B, ENC2_0, ENC2_A, ENC2_B, DB, SDI. [4] Выводы: EXI1, EXI2, SDO, VC, ENC1_0, ENC1_A, ENC1_B, ENC2_0, ENC2_A, ENC2_B, DB, SDI, NRESET. [5] В режиме выдачи синусоидального напряжения. [6] При Fclk_adc = 2,5 МГц, Fclk = Fclk_adc/16.

Электростатическая защита

Микросхема имеет встроенную защиту от электростатического разряда до 1000 В по модели человеческого тела. Требует мер предосторожности.

Предельно-допустимые и предельные режимы эксплуатации

Таблица 3. Предельно-допустимые и предельные режимы эксплуатации микросхем

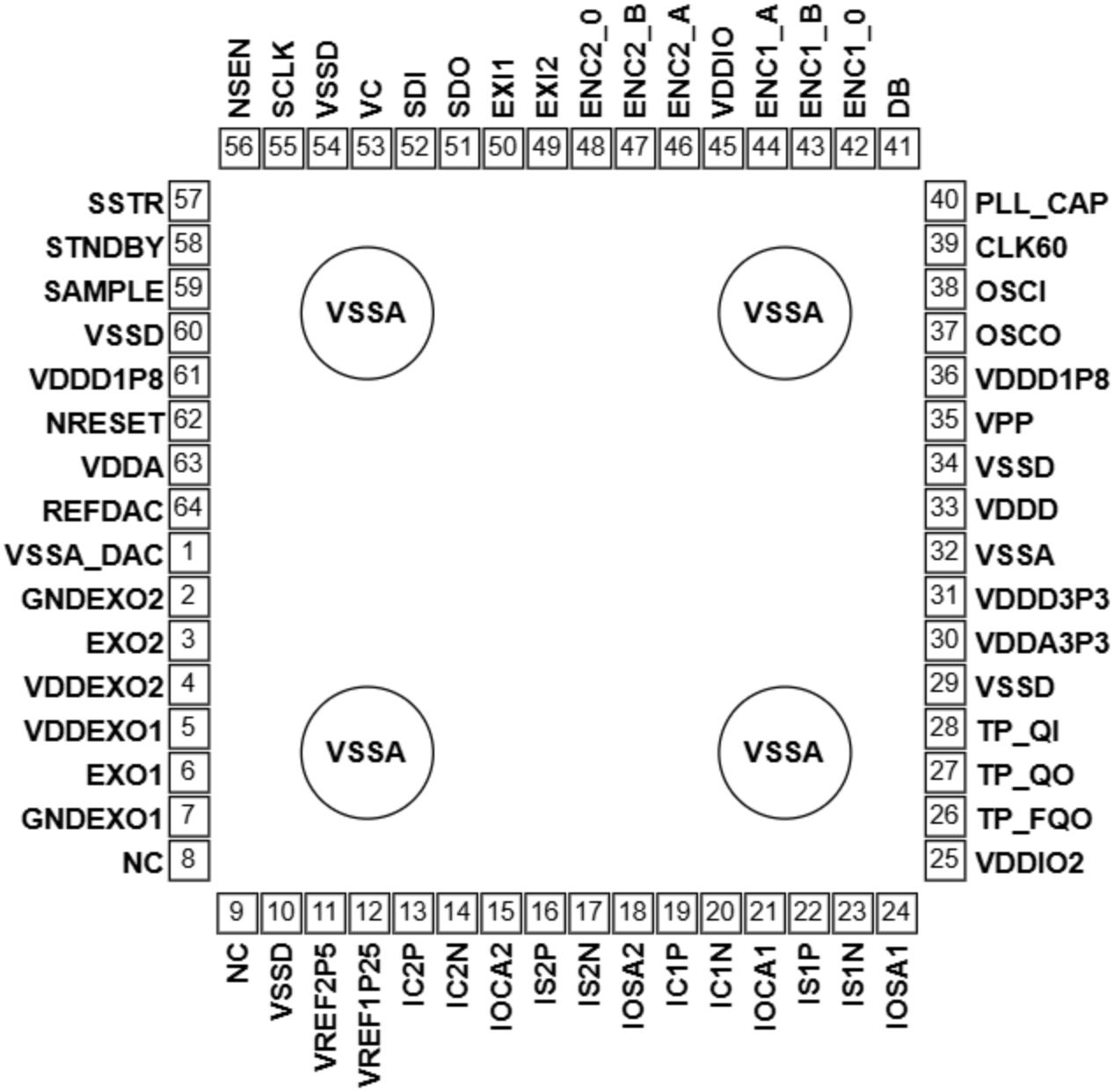
Параметр, единица измерения	Предельно-допустимый режим		Предельный режим	
	не менее	не более	не менее	не более

Напряжение питания аналоговой части (Uпит1), В	4,5	5,25	–0,3	5,35
Напряжение питания цифровой части (Uпит2), В	4,5	5,25	–0,3	5,35
Напряжение питания интерфейсной части (Uпит3), В	3	3,6	–0,3	4
Входное опорное напряжение АЦП (Uвх_оп), В	2,125	2,875	0	5,35
Входное напряжение высокого уровня цифровых выводов, В	0,8·Uпит3	Uпит3+0,3[2]	–0,3	4
Входное напряжение низкого уровня цифровых выводов, В	0	0,6	–0,3	4
Входное напряжение питания выходного буфера EXO1 (Uвх_буф1), В	3	5,25	-0,3	5,35
Входное напряжение питания выходного буфера EXO2 (Uвх_буф2), В	3	5,25	-0,3	5,35
Напряжение на выводе VPP в режиме чтения ПЗУ, В	3,0	3,6	–0,3	4,0
Напряжение на выводе VPP в режиме программирования ПЗУ, В	9,5	10,1	–0,3	10,5
Температура эксплуатации, °С	–60	+125	–60	+150

[1] Допускается уточнять электрические режимы после проведения испытаний. [2] Не более 5,25 В.

*Последнее обновление **3 июл. 2026 г.***

Порты



Распиновка 5400TP065A-022

Распиновка

Таблица выводов

№ вывода	Наименование вывода	Тип вывода	Описание
1	VSSA_DAC	PWR	Общий вывод опорного напряжения генератора
2	GNDEXO2	PWR	Общий вывод генератора опорной частоты 2
3	EXO2	AO	Выход генератора опорной частоты 2
4	VDDEXO2	PWR	Питание генератора опорной частоты 2
5	VDDEXO1	PWR	Питание генератора опорной частоты 1
6	EXO1	AO	Выход генератора опорной частоты 1
7	GNDEXO1	PWR	Общий вывод генератора опорной частоты 1
8, 9	NC	–	Не используется
10, 29, 34, 54, 60	VSSD	PWR	Общий вывод питания цифровой части
11	VREF2P5	AI/AO	Вход-выход источника опорного напряжения АЦП. Подключен к шине «общий» через конденсатор
12	VREF1P25	AO	Выход источника опорного напряжения АЦП. Подключен к шине «общий» через конденсатор

№ вывода	Наименование вывода	Тип вывода	Описание
13	IC2P	AI	Прямой вход усилителя cos2
14	IC2N	AI	Инверсный вход усилителя cos2
15	IOCA2	AO	Выход буферного усилителя cos2 (вход АЦП cos2)
16	IS2P	AI	Прямой вход усилителя sin2
17	IS2N	AI	Инверсный вход усилителя sin2
18	IOSA2	AO	Выход буферного усилителя sin2 (вход АЦП sin2)
19	IC1P	AI	Прямой вход усилителя cos1
20	IC1N	AI	Инверсный вход усилителя cos1
21	IOCA1	AO	Выход буферного усилителя cos1 (вход АЦП cos1)
22	IS1P	AI	Прямой вход усилителя sin1
23	IS1N	AI	Инверсный вход усилителя sin1
24	IOSA1	AO	Выход буферного усилителя sin1 (вход АЦП sin1)
25	VDDIO2	PWR	Питание генератора на кварцевом резонаторе
26	TP_FQO	DO	Выход частоты генератора на кварцевом резонаторе

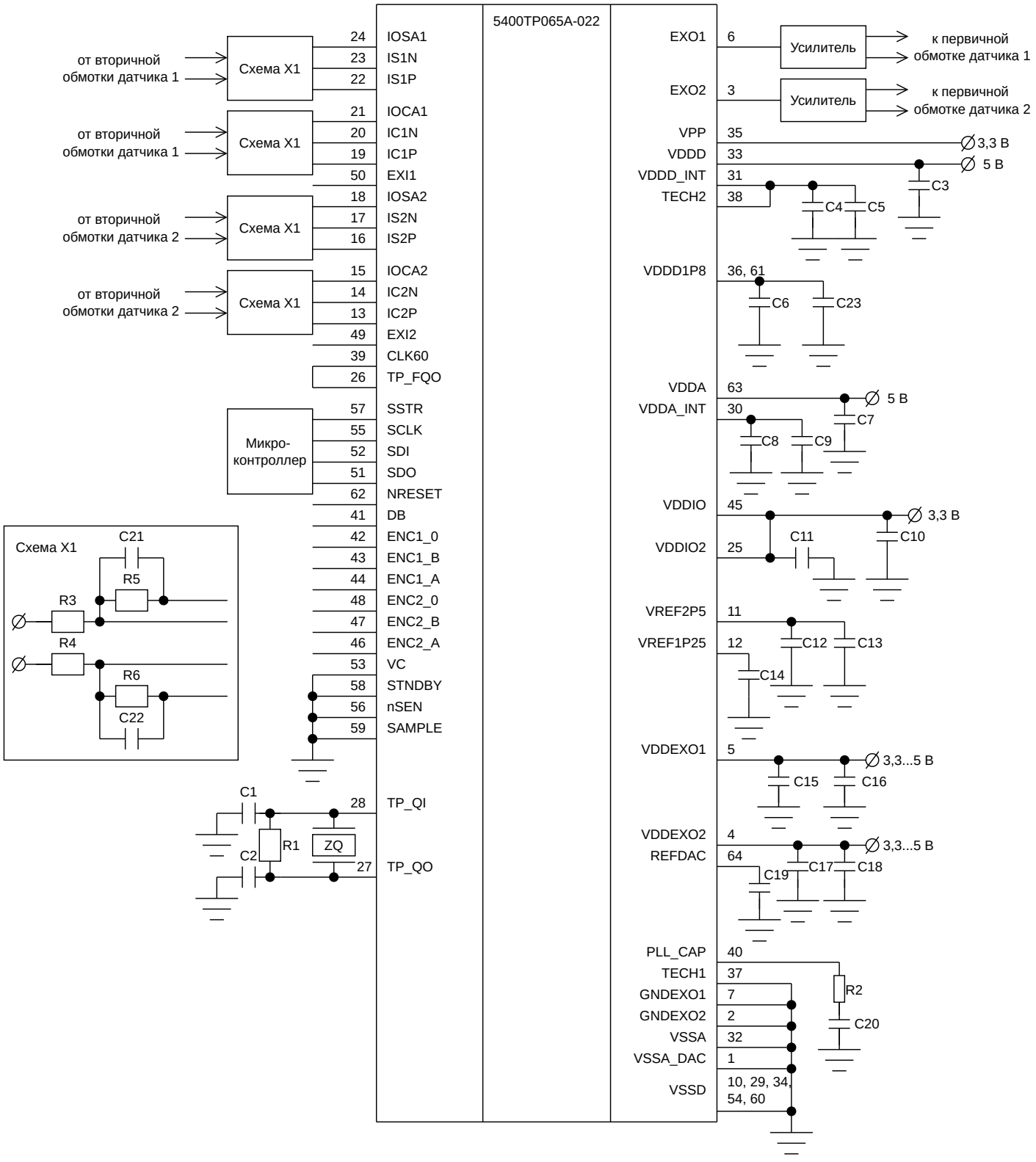
№ вывода	Наименование вывода	Тип вывода	Описание
27	TP_QO	AO	Вывод для подключения кварцевого резонатора
28	TP_QI	AI	Вывод для подключения кварцевого резонатора
30	VDDA_INT	PWR	Вывод питания аналоговой части
31	VDDD_INT	PWR	Вывод питания цифровой части
32	VSSA	PWR	Общий вывод питания аналоговой части
33	VDDD	PWR	Вход линейного регулятора питания цифровой части
35	VPP	AI	Вывод напряжения программирования
36, 61	VDDD1P8	PWR	Вывод питания ядра 1,8 В
37	TECH1	AO	Служебный вывод
38	TECH2	AI	Служебный вывод
39	CLK60	AI	Вход тактового сигнала
40	PLL_CAP	AO	Вывод петлевого фильтра умножителя частот
41	DB	DO	Вывод отладки режимов и программ
42	ENC1_0	DO	Выход эмуляции квадратурного энкодера, датчик 1, канал 0. Выход дополнительного канала SPI

№ вывода	Наименование вывода	Тип вывода	Описание
43	ENC1_B	DO	Выход эмуляции квадратурного энкодера, датчик 1, канал B. Выход дополнительного канала SPI
44	ENC1_A	DO	Выход эмуляции квадратурного энкодера, датчик 1, канал A. Выход дополнительного канала SPI
45	VDDIO	PWR	Питание портов ввода-вывода
46	ENC2_A	DO	Выход эмуляции квадратурного энкодера, датчик 2, канал A. Выход дополнительного канала SPI
47	ENC2_B	DO	Выход эмуляции квадратурного энкодера, датчик 2, канал B. Выход дополнительного канала SPI
48	ENC2_0	DO	Выход эмуляции квадратурного энкодера, датчик 2, канал 0. Выход дополнительного канала SPI
49	EXI2	DI/DO	Выход опорного напряжения, датчик 2.
50	EXI1	DI/DO	Выход опорного напряжения, датчик 1. Вход опорного напряжения для датчика 1 и 2
51	SDO	DO	Выход данных SPI
52	SDI	DI/DO	Вход/выход данных SPI
53	VC	DI/DO	Вход/выход. В режиме прямой/ параллельной передачи угла выбирает

№ вывода	Наименование вывода	Тип вывода	Описание
			угол или скорость. В остальных режимах управляет внешними приемопередатчиками
55	SCLK	DI	Тактовый сигнал SPI
56	NSEN	DI	Выбор микросхемы. Активный уровень – лог. «0»
57	SSTR	DI	Строб SPI
58	STNDBY	DI	Режим пониженного энергопотребления: Активный уровень – лог. «1»
59	SAMPLE	DI	Строб сэмплирования координаты и скорости для SPI
62	NRESET	DI	Сброс микросхемы. Активный уровень – лог. «0»
63	VDDA	PWR	Вход регулятора питания аналоговой части
64	REFDAC	AI/AO	Вход/выход опорного напряжения генератора
EP	VSSA	PWR	Общий вывод

*Последнее обновление **19 июн. 2026 г.***

Рекомендуемая схема применения



Рекомендуемая схема применения

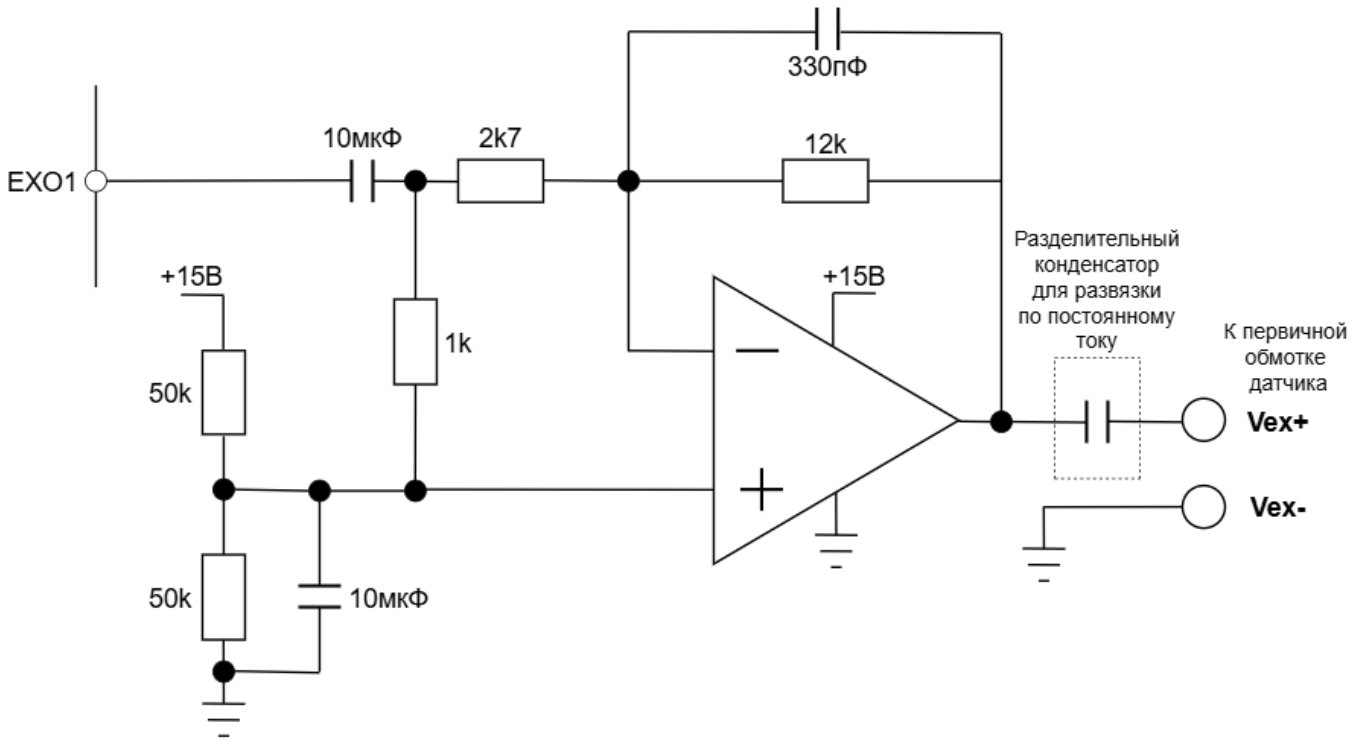
Таблица 4. Таблица внешних компонентов

Обозначение	Номинал	Примечание
R1	510 кОм	
R2	20 Ом	
R3, R4	27 кОм	
R5, R6	10 кОм	
C1, C2	12 пФ	
C3, C4, C6, C7, C8, C10, C11, C12, C14, C16, C18, C19, C20	0,1 мкФ	
C5, C9, C13, C15, C17	4,7 мкФ	Не обязательны, рекомендуются для выравнивания заряда
C21, C22	330 пФ	
ZQ	10 МГц	Кварцевый резонатор

Конденсаторы либо высокочастотные керамические, либо сдвоенные. В случае сдвоенных конденсаторов, один из них обязательно должен быть высокочастотный керамический емкостью не менее 10 нФ. Шунтирующие конденсаторы должны располагаться на плате в непосредственной близости к соответствующим выводам микросхемы.

Примечания: Допускается подключение VDDIO (45) к питанию VDDD (33) или VDDD_INT (31). Для программирования ячейки OTP памяти вывод VPP (35) запитывается 9,8 В ± 0,3 В, для чтения ячеек вывод VPP (35) запитывается 3,3 В. При VREF_en = 0 (регистр AFE_config) вывод VREF2P5 (11) должен быть запитан опорным уровнем 2,5 В. При использовании

внешнего тактового генератора подключенного к выводу CLK60 (39) вывод TP_QI (28) подключить к VDDD_INT, вывод TP_QO (27) подключить к GND.



Типовая схема усиления сигнала возбуждения

ЗАЧЕМ В СХЕМЕ УСИЛЕНИЯ НУЖЕН ФНЧ

Сигнал на EXO1/EXO2 — ступенчатый: он формируется 8-битным ЦАП без внутреннего сглаживания. Полосоограничивающий каскад на операционном усилителе в этой схеме выполняет роль **реконструкционного ФНЧ** — пропускает синусоиду возбуждения и подавляет ступеньки и образы дискретизации. Обоснование — в разделе Программируемый генератор.

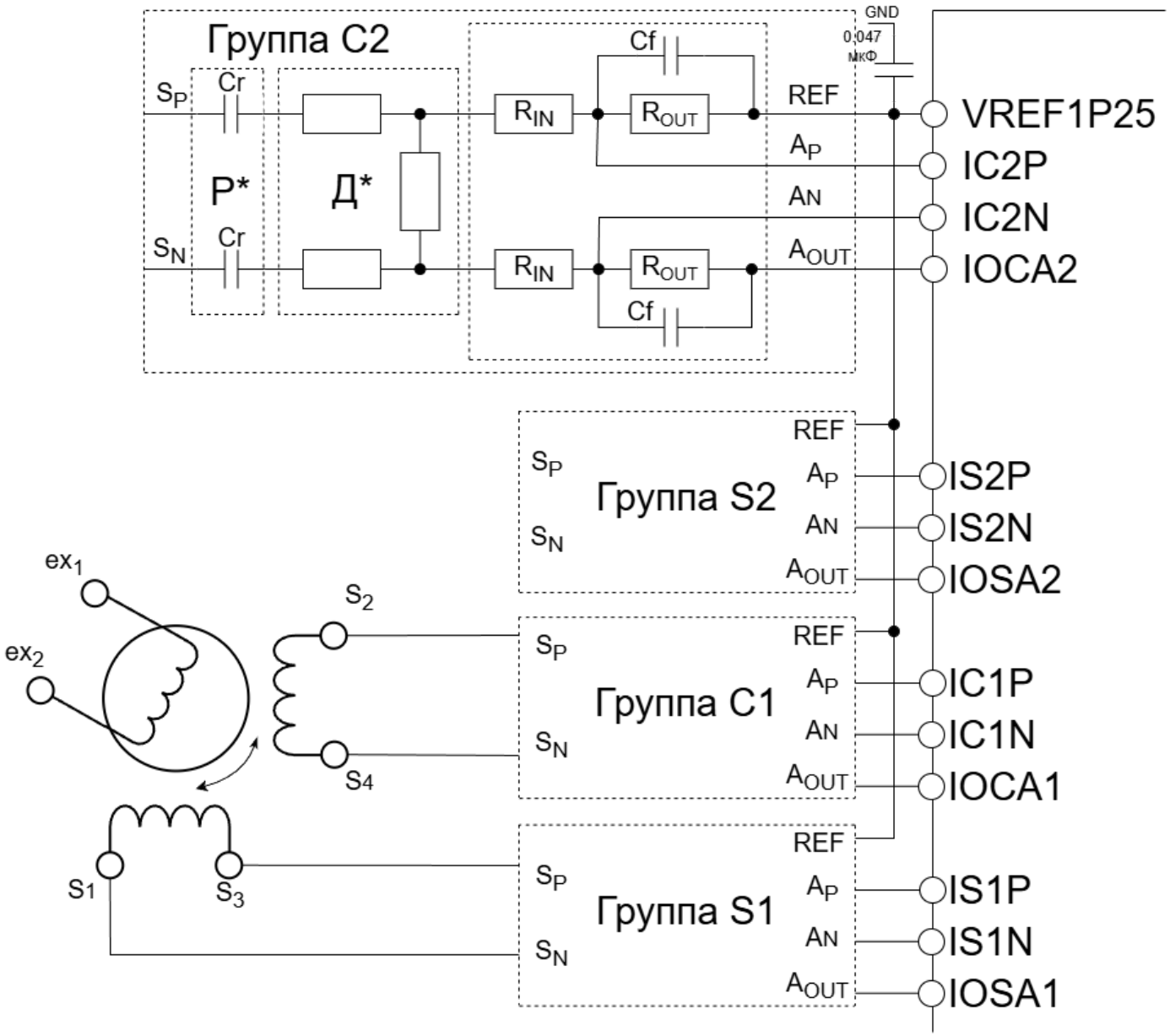


Схема подключения входов преобразователя для датчиков типа СКВТ с развязкой по постоянному напряжению и дополнительным делителем на входах

Д - Делитель напряжения для выравнивания амплитуды (опционально)

Р - Разделительные конденсаторы С1, С2 для развязки по постоянному току (опционально)

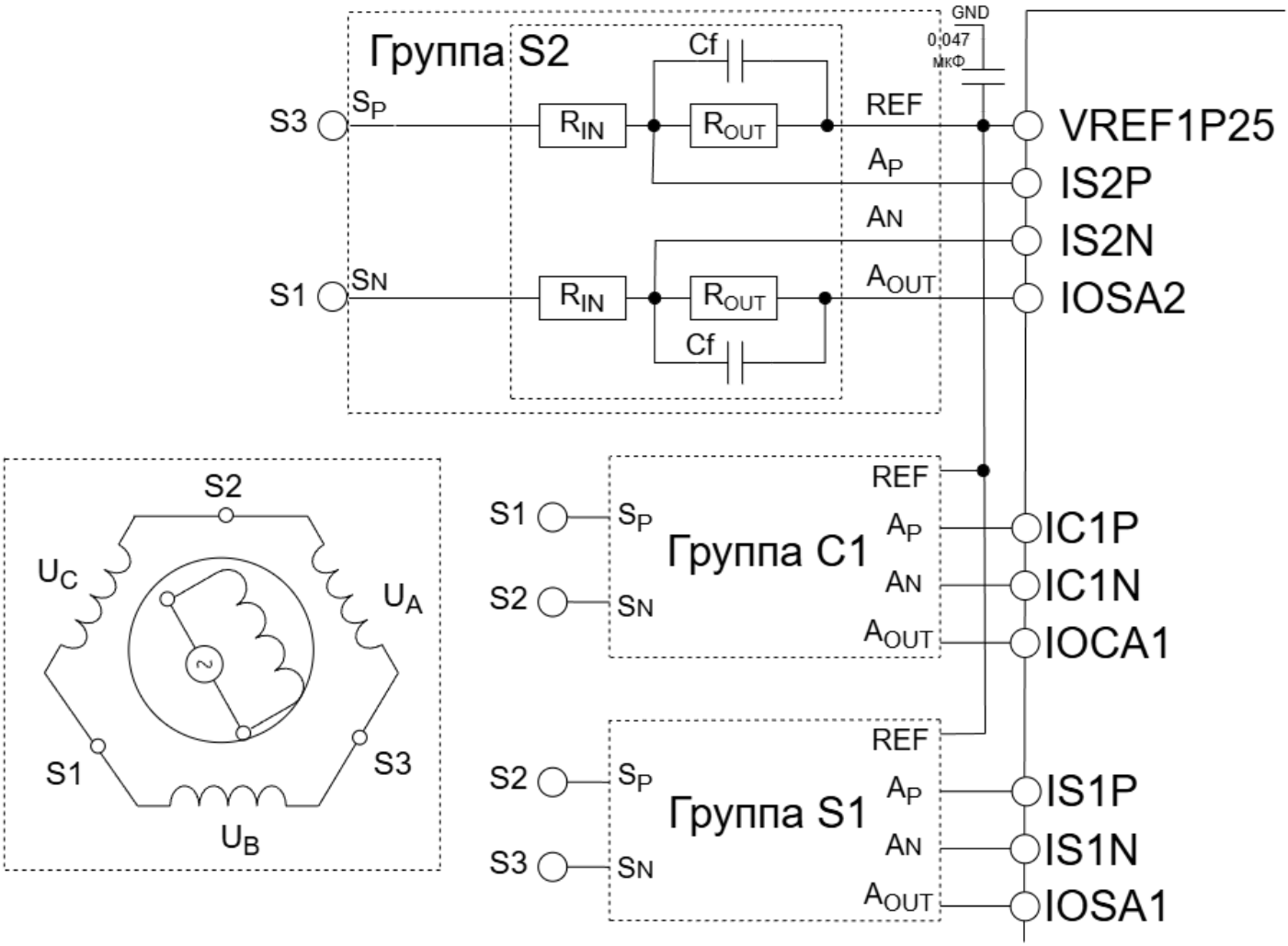
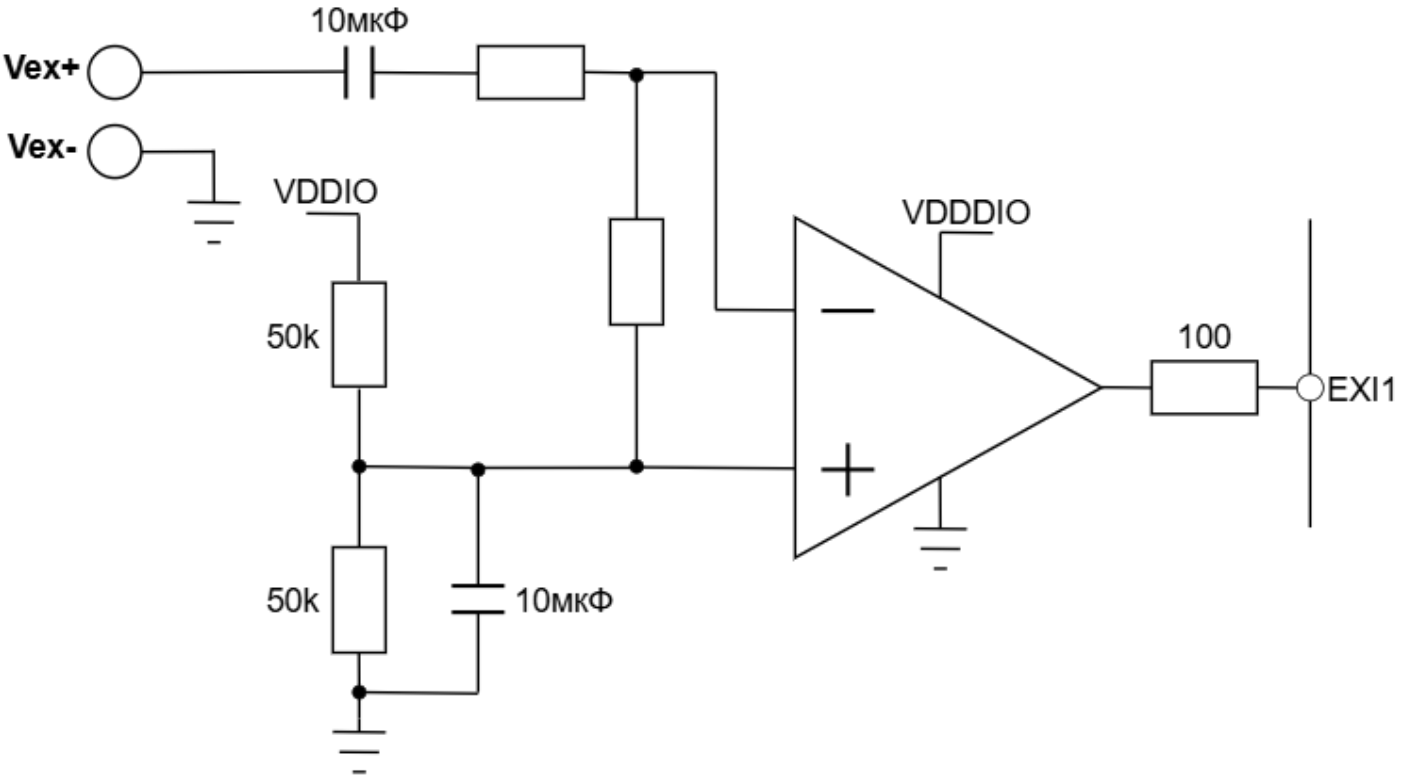


Схема подключения входов преобразователя для датчиков типа сельсин



Типовая схема подключения входа [EXI1](/docs/5400TP065A-022/ad3s-pins#EXI1) при Ex_source=1

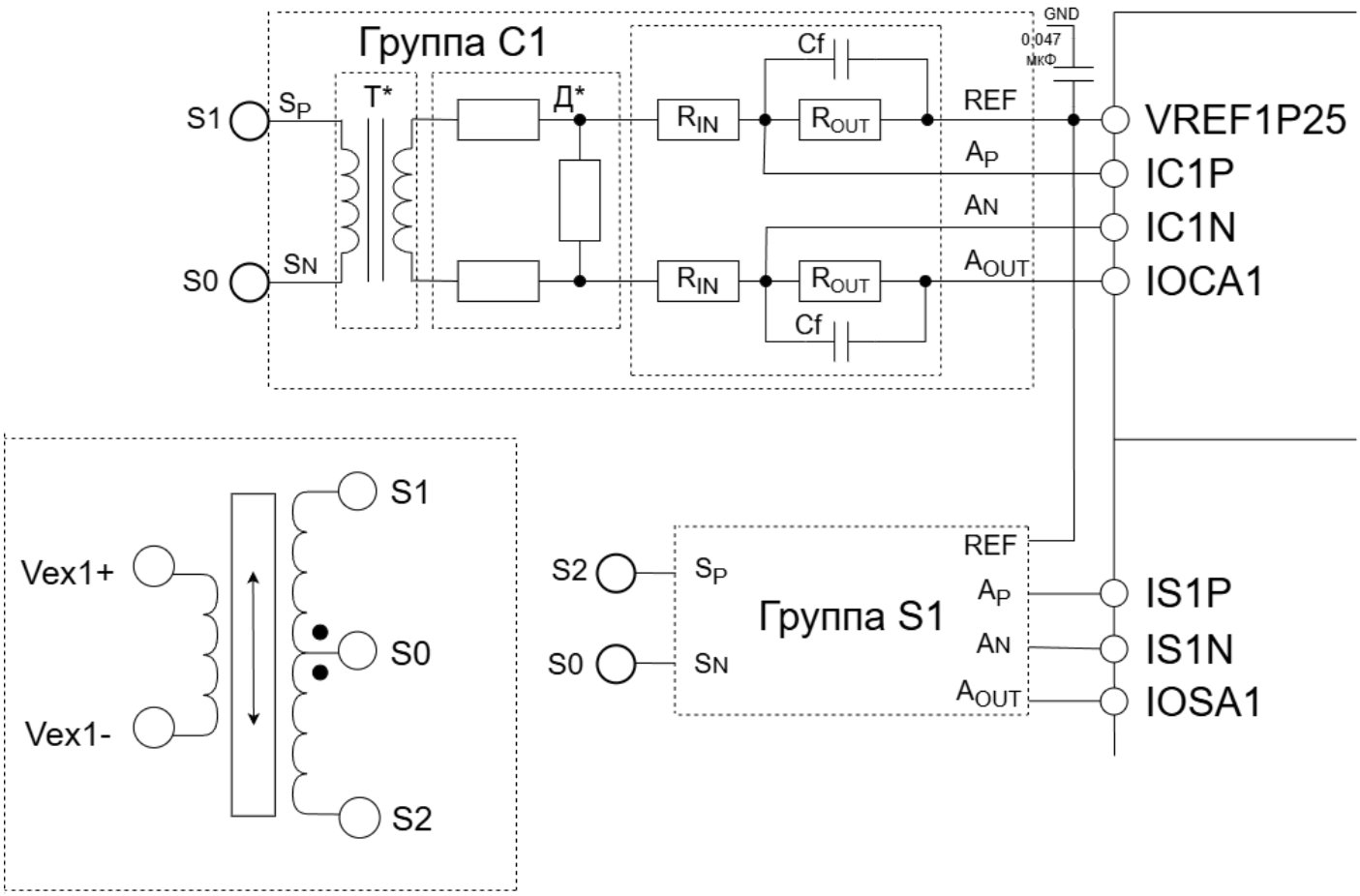


Схема подключения входов преобразователя для датчиков типа ЛРДТ, подключаемых по 5-ти проводной схеме, с дополнительным делителем и трансформаторной развязкой по постоянному напряжению при задании sensor_mode=1

Т - Трансформаторная развязка по постоянному напряжению

Д - Делитель напряжения для выравнивания амплитуды (опционально)

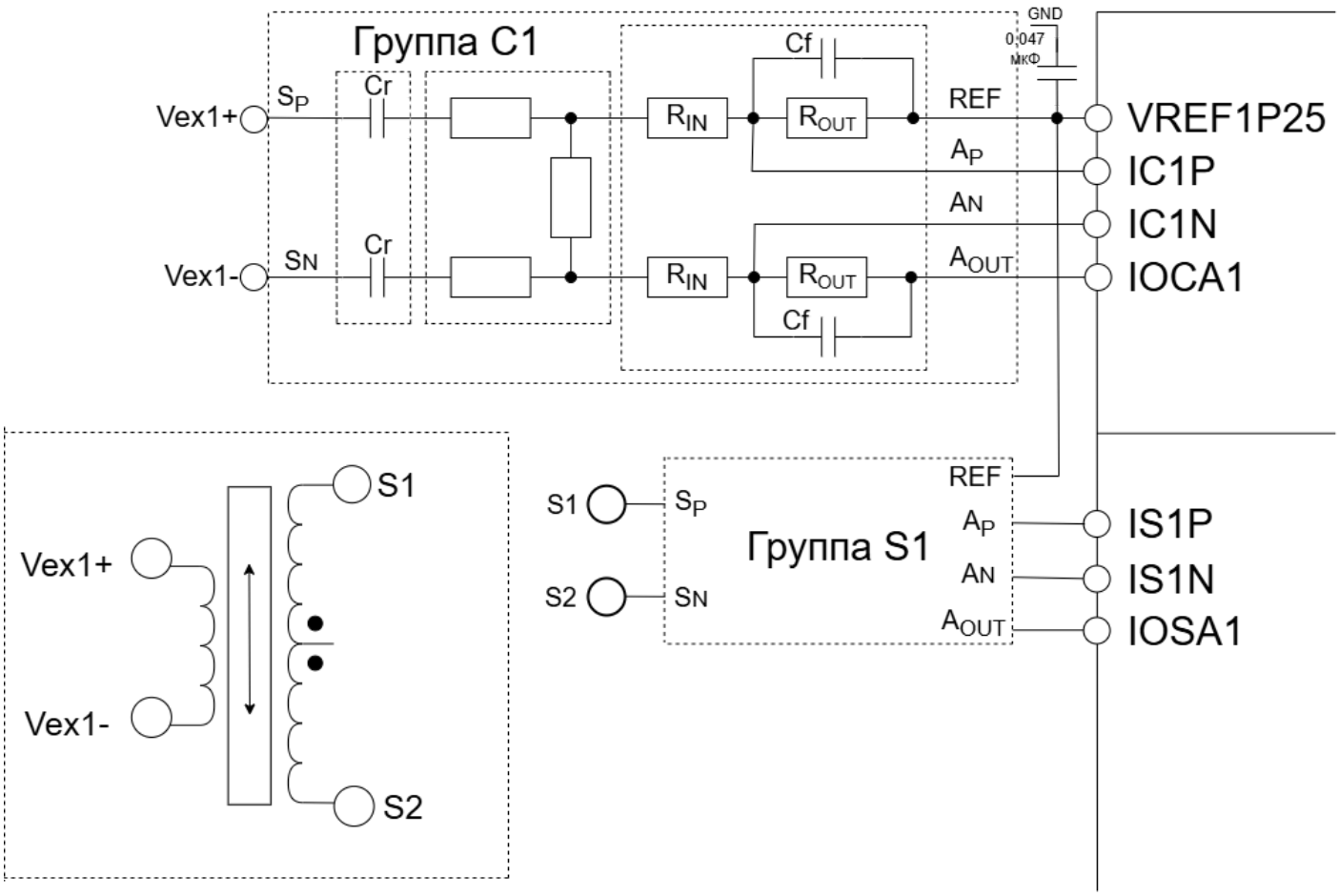
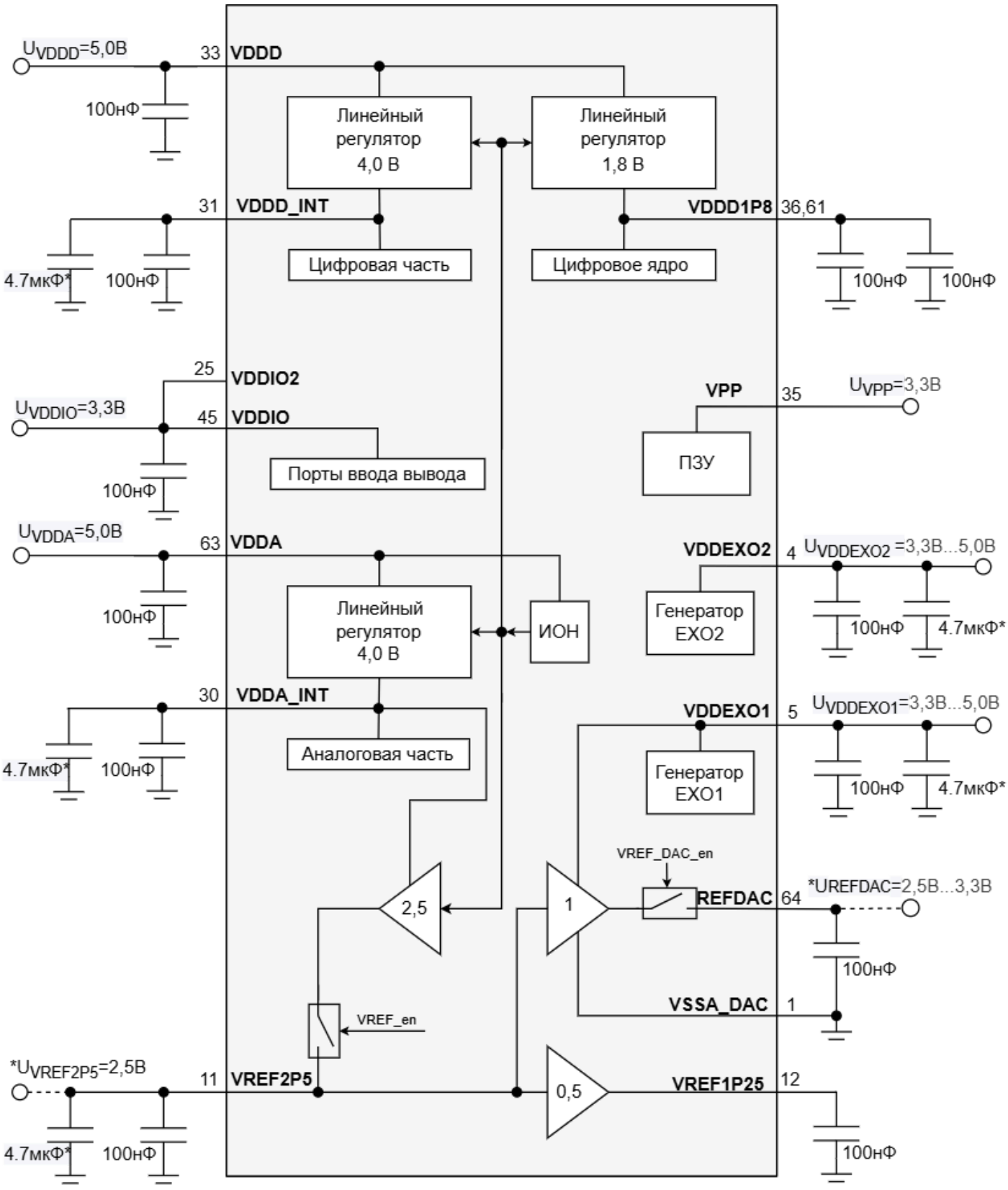


Схема подключения входов преобразователя для датчиков типа ЛРДТ с последовательно соединенными обмотками при задании Sensor_mode=2

*Последнее обновление **21 сент. 2026 г.***

Выбор схемы питания микросхемы



Структура питания микросхемы

- Конденсаторы 4.7мкФ не обязательны, но рекомендуются - для выравнивания заряда и снижения пульсаций. Чем меньше размер корпуса конденсаторов 100нФ, тем лучше он справляется с пульсациями.

Организация питания микросхемы:

- **Аналоговая и цифровая части** питаются от встроенных в микросхему линейных регуляторов, выходное напряжение которых настроено на 4,0 В ±10%.
- **Цифровое ядро** получает питание от отдельного встроенного линейного регулятора на 1,8 В.
- **Порты ввода-вывода** запитываются от вывода VDDIO — внешнее напряжение 3,3 В ±10%.
- **Генератор на кварце** запитывается от вывода VDDIO2 — внешнее напряжение 3,3 В ±10%.
- **Блок ПЗУ** запитывается от вывода VPP — внешнее напряжение 3,3 В ±10% (при программировании ПЗУ напряжение на VPP повышается до 9,8 В ± 0,3 В, см. рекомендуемую схему применения).

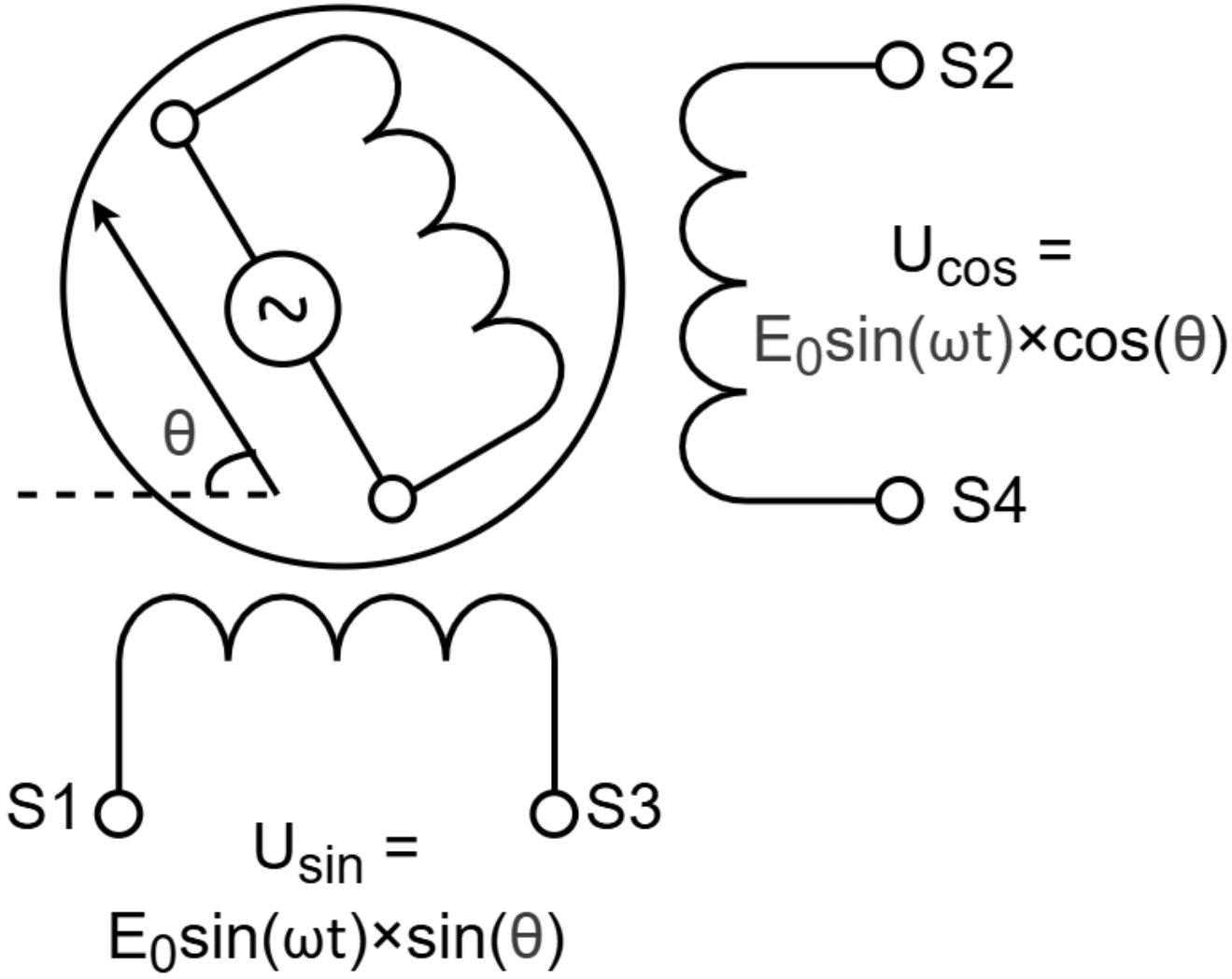
Источники опорного напряжения (ИОН):

- **Опорное напряжение для АЦП (2,5 В)** микросхема может обеспечивать сама при установке бита VREF_en регистра AFE_config. Также допускается его подача на вывод VREF2P5 с внешнего прецизионного регулятора (по умолчанию данный бит равен 0).
- **Опорное напряжение схемы смещения постоянной составляющей входных сигналов (вывод VREF1P25, пин 12; типовое 1,25 В, диапазон 1,125…1,375 В); используется при автокалибровке АЦП.
- **Опорное напряжение для ЦАП** микросхема может формировать самостоятельно при включении бита VREF_DAC_en регистра AFE_config. Кроме того, оно может быть подано от внешнего источника на вывод REFDAC (в этом случае бит VREF_DAC_en должен быть равен 0). Конкретное значение выбирается пользователем в диапазоне [2,5 В…3,3 В].

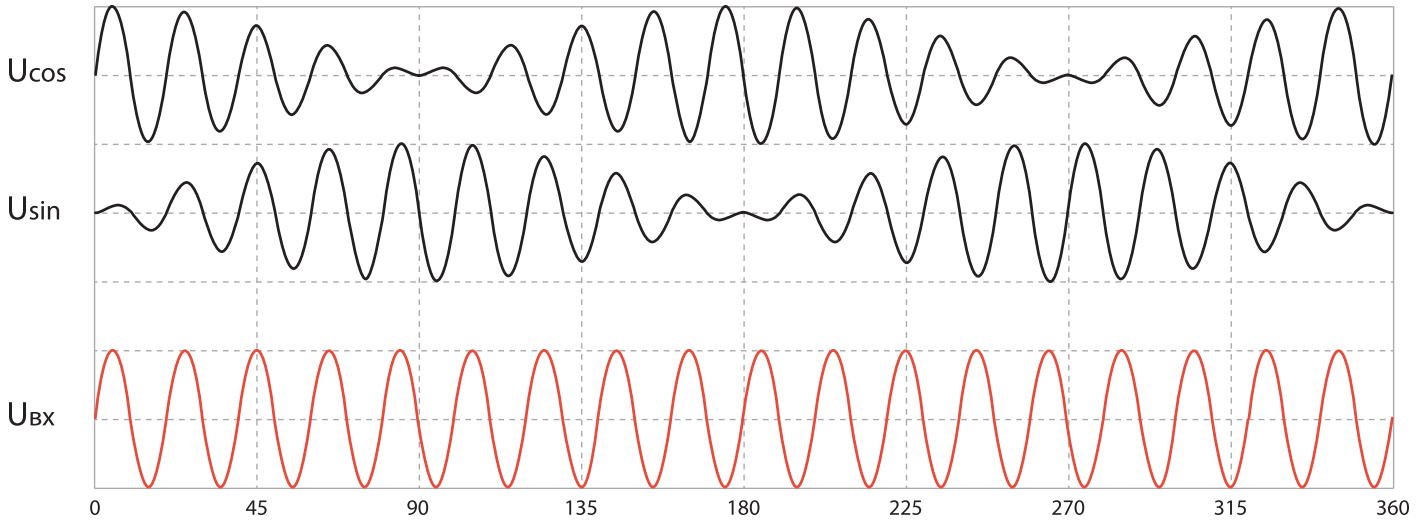
*Последнее обновление **3 июл. 2026 г.***

Типы датчиков

Датчики типа СКВТ (Resolver)



Датчик типа СКВТ



Осциллограмма сигналов датчика типа СКВТ

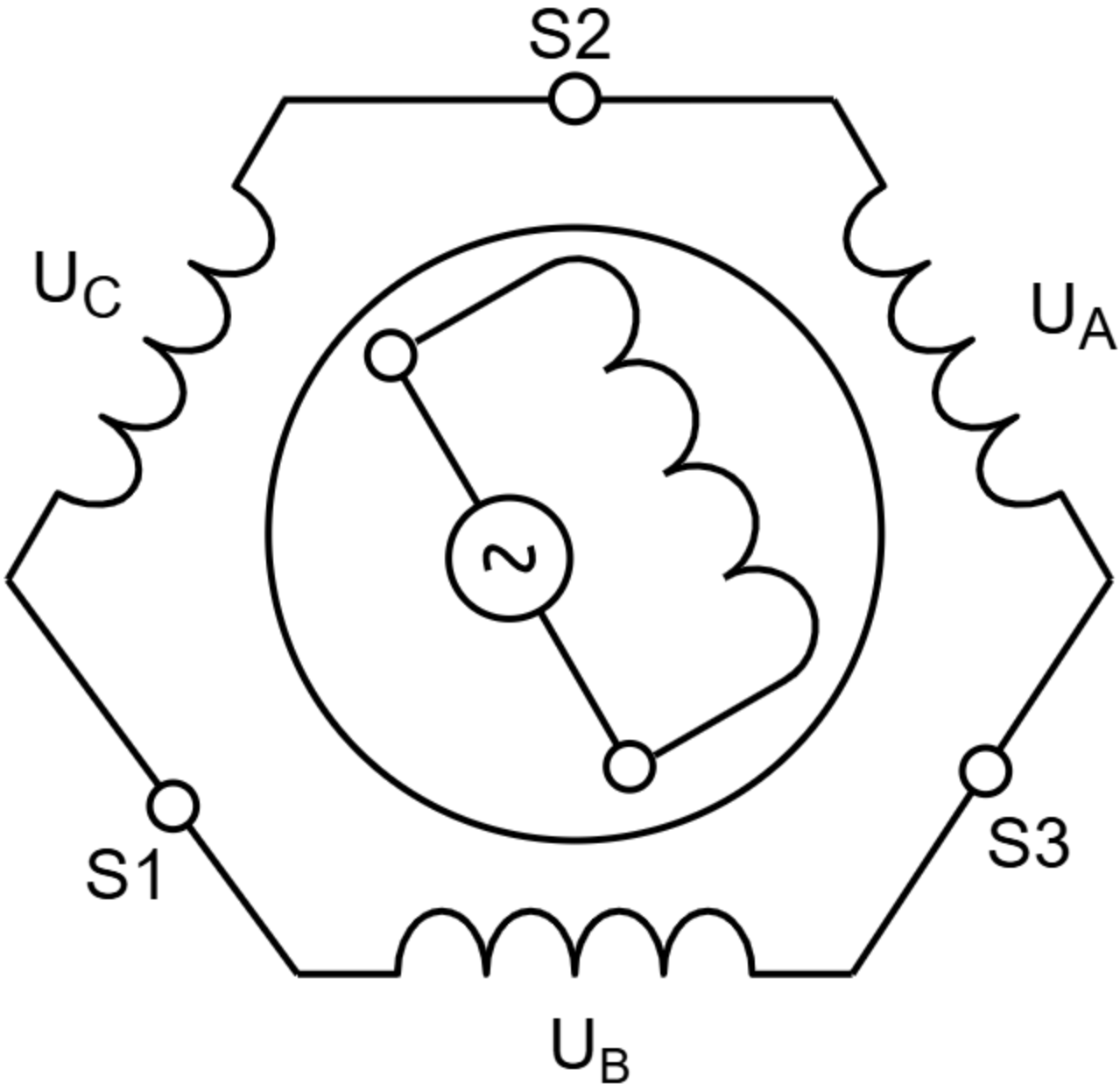
Для классического СКВТ датчика статорные обмотки механически расположены под углом 90°. Первичная обмотка возбуждается переменным опорным (эталонным) напряжением. Индуктивная связь наводит ЭДС на вторичные обмотки в соответствии с функцией положения ротора (вала) относительно статора. Таким образом, датчик вырабатывает два выходных напряжения Ucos, Usin, модулированных синусом и косинусом угла поворота вала. Выходные напряжения имеют схожие уравнения:

$$S3 - S1 = E_0 \sin \omega t \times \sin \theta = U_{\sin}$$
$$S2 - S4 = E_0 \sin \omega t \times \cos \theta = U_{\cos}$$

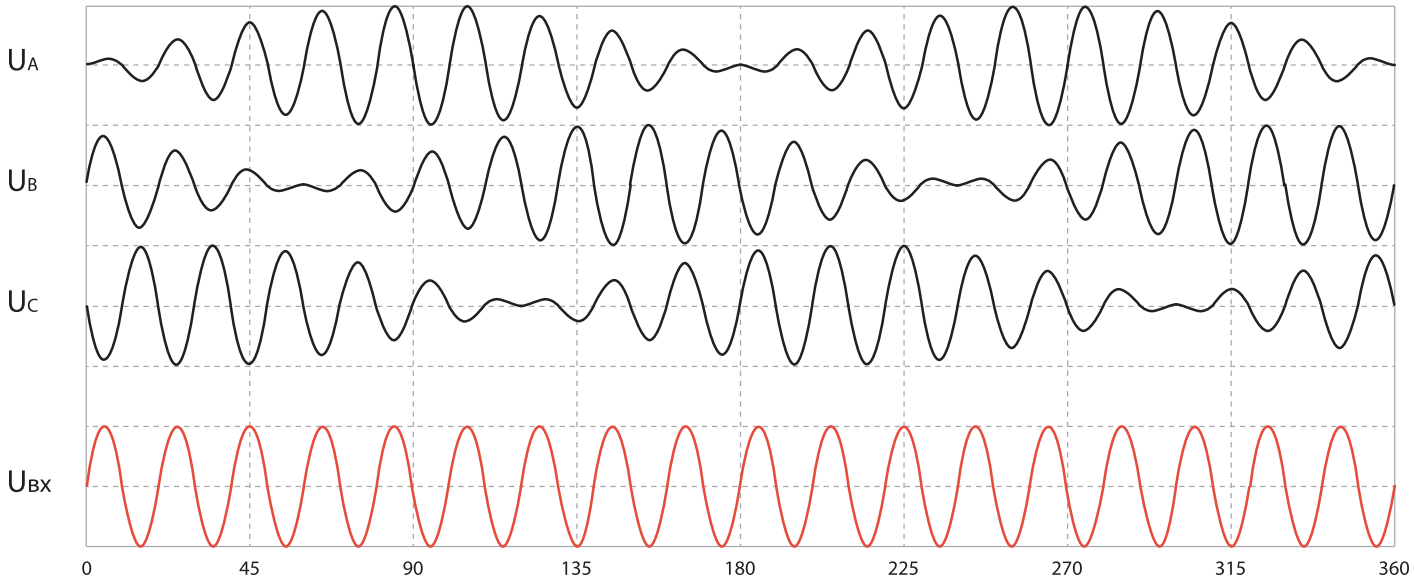
где:

- θ — угол ротора резольвера
- $\sin \omega t$ — частота сигнала возбуждения
- E_0 — амплитуда сигнала возбуждения

Датчики типа сельсин (Synchro)



Датчик типа Сельсин



Осциллограмма сигналов датчика типа Сельсин

Сельсин – вращающийся трансформатор, состоящий из одной первичной обмотки (ротор) и тремя вторичными обмотками (статор). Три обмотки статора расположены под углом 120° друг к другу.

При подаче переменного напряжения возбуждения Uвх на первичную обмотку сельсина в машине создается электромагнитное поле, ориентированное перпендикулярно плоскости ротора. Данное поле вызывает индуцирование электродвижущей силы во вторичных обмотках сельсина. Амплитуда напряжения, возникающего во вторичных обмотках, определяется углом поворота ротора θ относительно соответствующей вторичной обмотки и рассчитывается согласно выражениям:

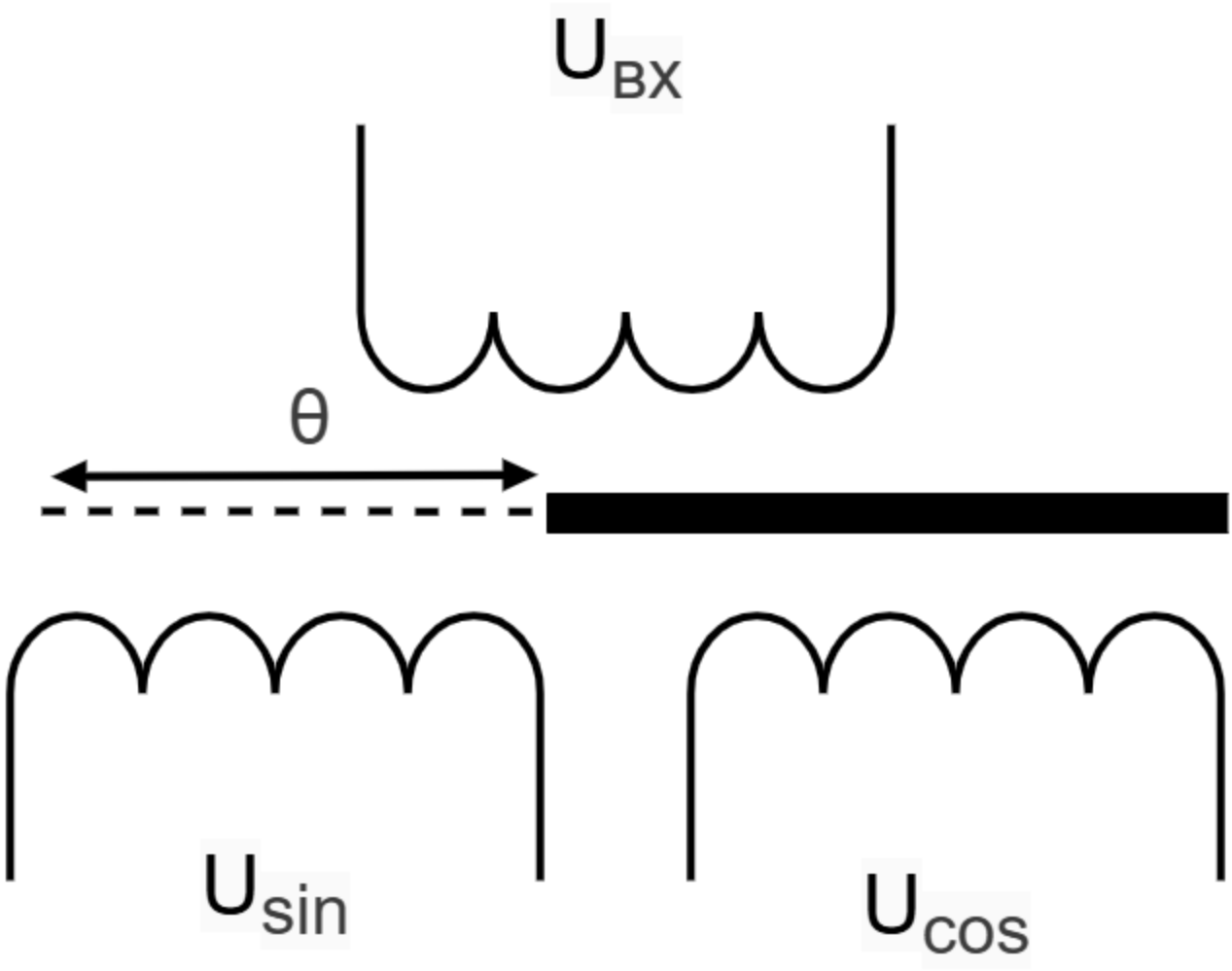
$$U_A = k \cdot U_{Bx} \cdot \sin(\theta),$$

$$U_B = k \cdot U_{Bx} \cdot \sin(\theta + 120^\circ),$$

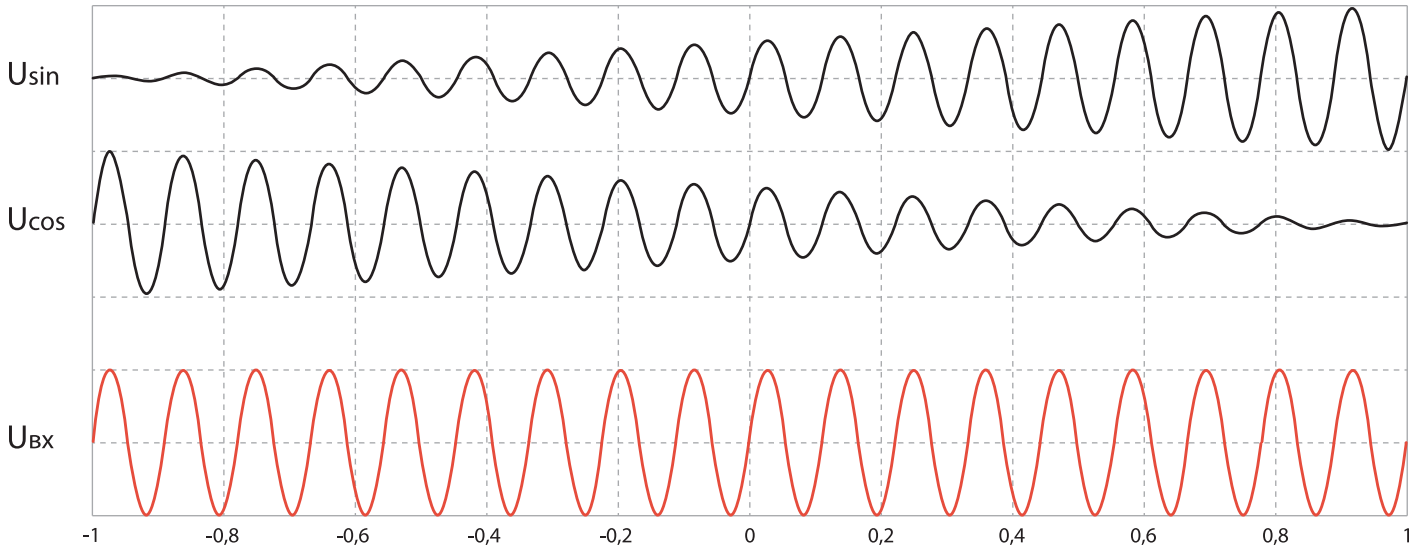
$$U_C = k \cdot U_{Bx} \cdot \sin(\theta - 120^\circ).$$

При подключении сельсина к микросхеме производится преобразование координат из трехфазной в двухфазную (преобразование alpha–beta–zero). Результирующие сигналы после преобразования соответствуют формату сигналов, поддерживаемых преобразователем в режиме СКВТ. Использование встроенного преобразователя координат позволяет подключить к микросхеме только один сельсин. В случае необходимости одновременного подключения двух сельсинов преобразование координат должно быть реализовано схемотехническими средствами на печатной плате, а микросхема должна работать в режиме СКВТ.

Датчики типа ЛРДТ (LVDT)



Датчик типа ЛРДТ



Осциллограмма сигналов датчика типа ЛРДТ

Линейный регулируемый дифференциальный трансформатор – электрическая машина обычно с одной первичной обмоткой и двумя вторичными обмотками, расположенными на одной линии. ЛРДТ используются в качестве датчиков перемещений.

При подаче сигнала переменного напряжения Uвх на первичную обмотку ЛРДТ в машине создается электромагнитное поле, которое наводит ЭДС во вторичных обмотках ЛРДТ. Амплитуда напряжения во вторичных обмотках зависит от смещения сердечника θ относительно вторичных обмоток и определяется по формулам

$$U_S = k \cdot U_{Bx} \cdot (1 + \theta),$$

$$U_C = k \cdot U_{Bx} \cdot (1 - \theta).$$

Часто используется последовательное соединение вторичных обмоток ЛРДТ. При этом происходит сложение сигналов UA и UB в противофазе. При таком включении сигналы на входе микросхемы имеют зависимость, которая задается по формулам

$$U_S = k \cdot U_{Bx} \cdot \theta,$$

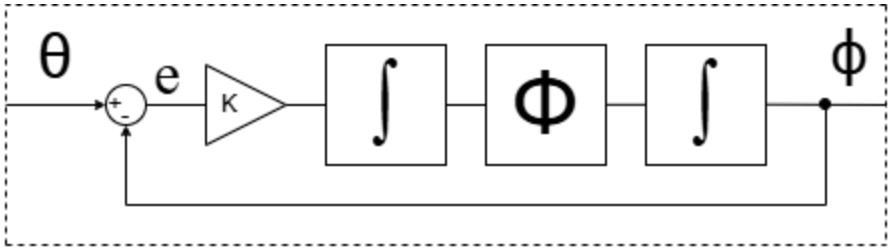
$$U_C = k \cdot U_{Bx} \cdot 1.$$

*Последнее обновление **21 сент. 2026 г.***

Архитектура и принцип работы

Принцип преобразования сигналов СКВТ в код угла

В основе преобразователя 5400TP065A-022 лежит система регулирования с замкнутой обратной отрицательной связью по углу поворота.

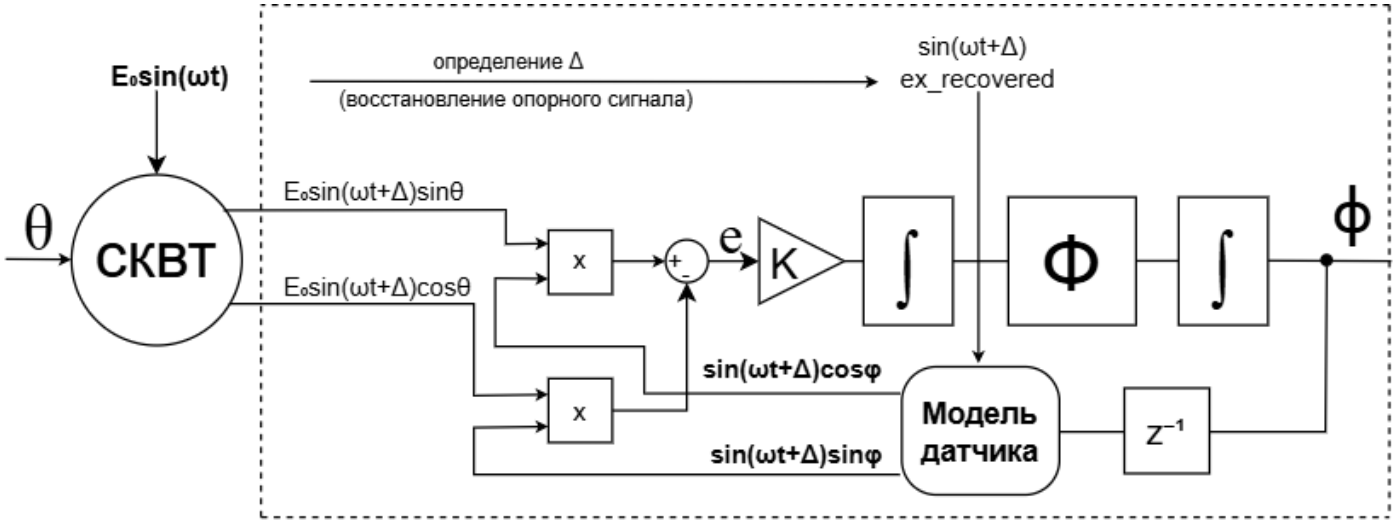


Контур 1

Следящий контур стремится свести разницу выходного и входного значения к нулю, делая свой выход равным входной координате. Контур содержит следующие элементы:

- вычислитель ошибки
- усилитель
- интегратор для демодуляции сигнала ошибки
- петлевой фильтр и интегратор для компенсации запаздывания и получения нулевой ошибки при равномерном движении датчика.

Для получения ошибки **e** контур в обратной связи содержит модель датчика, которая осуществляет преобразование вычисленной на предыдущем шаге координаты в виртуальные сигналы датчика $\sin\phi$, $\cos\phi$.



Контур 2

Виртуальные сигналы умножаются на входные сигналы с датчика:

$$E_0 \sin(\omega t + \Delta) \sin\theta \times \sin(\omega t + \Delta) \cos\phi$$

$$E_0 \sin(\omega t + \Delta) \cos\theta \times \sin(\omega t + \Delta) \sin\phi$$

Вычисляется разность сигналов:

$$E_0 \sin^2(\omega t + \Delta) \times (\sin\theta\cos\phi - \cos\theta\sin\phi) \tag{1}$$

Высокочастотная составляющая $\sin^2(\omega t + \Delta) = \frac{1}{2} - \frac{1}{2}\cos(2\omega t + 2\Delta)$ подавляется на первом интегрирующем звене контура, выполняющим роль ФНЧ.

Оставшееся выражение можно преобразовать с использованием тригонометрического тождества:

$$E_0(\sin\theta\cos\phi - \cos\theta\sin\phi) = E_0\sin(\theta - \phi) \tag{2}$$

При корректной работе контура разница $(\theta - \phi)$ становится малым числом, для которого верно следующее выражение:

$$E_0 \sin(\theta - \phi) \approx E_0(\theta - \phi) \tag{3}$$

Полученное после первого интегрирования значение $E_0(\theta - \phi)$ является разницей между истинным положением ротора и вычисленной следящим контуром координатой, и это значение получено без проведения вычислений, обратных к функции СКВТ. Модель функции СКВТ в обратной связи может быть усложнена параметрами и естественными

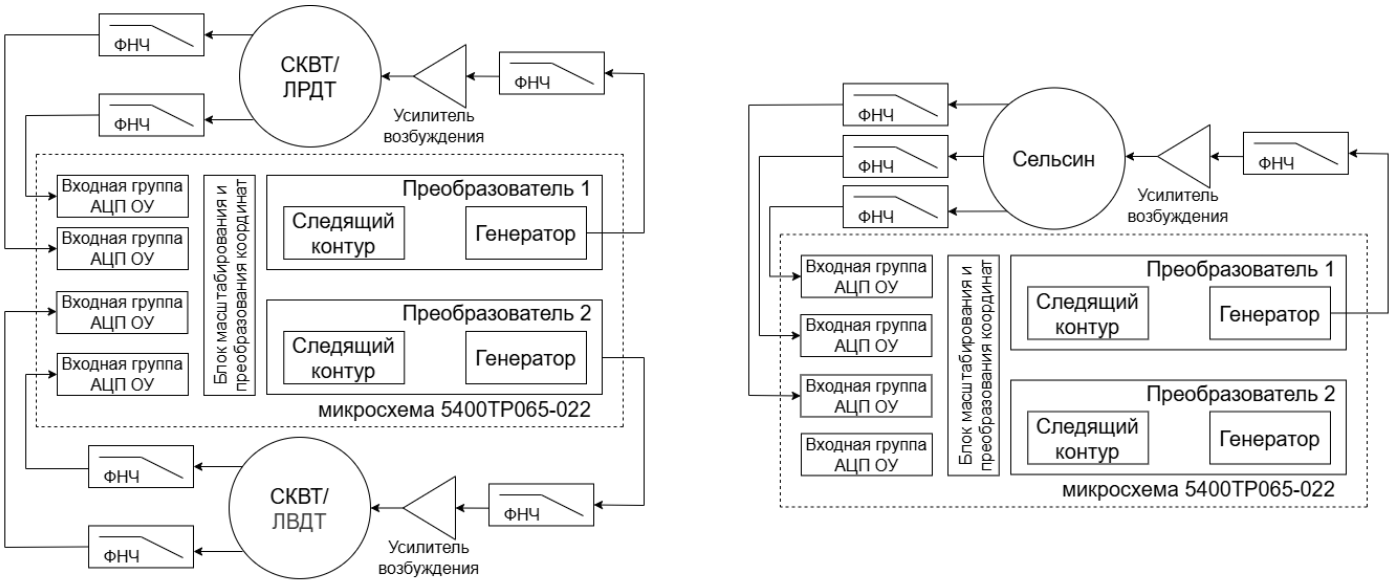
искажениями порождаемых сигналов в целях увеличения соответствия виртуальных к реальным сигналам с датчика.

Взятие разницы двух последовательных по времени координат эквивалентно дифференцированию угла, а результат является угловой скоростью, из которой в контуре восстанавливается координата с помощью второго интегратора.

$$\omega[n] = \theta[n] - \phi[n-1]$$
$$\phi[n] = \int \omega[n] \tag{4}$$

Архитектура микросхемы

Микросхема содержит два независимых блока преобразователя координата/код (далее “преобразователь 1” и “преобразователь 2”), которые могут использоваться как независимо, так и совместно для повышения точности или надежности преобразования. В обозначении регистров, принадлежащих каждому преобразователю, используется префикс “C1” и “C2“ соответственно. Каждый блок преобразователя реализует следящий принцип измерения. Для подачи возбуждающего напряжения на датчик микросхема содержит два генератора сигналов возбуждения датчиков. Сигналы с вторичных обмоток датчика преобразуются в цифровую форму и подаются на вход преобразователя, который осуществляет нахождение координаты, используя виртуальную модель датчика и контур с обратной связью.



А Б

Варианты использования микросхемы: А - два независимых СКВТ датчика, Б - подключение датчика типа Сельсин

Наличие 4 входных групп и 2 преобразователей позволяет использовать микросхему в различных включениях:

- один или два датчика СКВТ (с синхронным или независимым возбуждением)
- СКВТ с 4-мя выходными обмотками и одной обмоткой возбуждения
- Сельсин (3 выходных обмотки)
- один или два датчика ЛРДТ (с синхронным или независимым возбуждением)

Микросхема позволяет использовать внешний независимый генератор возбуждения. Микросхема имеет возможность обрабатывать демодулированные сигналы.

*Последнее обновление **22 мая 2026 г.***

Программируемый генератор

Программируемый генератор

Микросхема содержит **два независимых программируемых генератора** — они формируют опорный (возбуждающий) сигнал, который после внешнего усиления подаётся на первичную обмотку датчика (СКВТ, сельсин, ЛРДТ). Сигналы выдаются на выводы EXO1/EXO2.

Питание выходного буфера каждого генератора подается с отдельных выводов — VDDEXO1/VDDEXO2 (плюс) и GNDEXO1/GNDEXO2 (общий). Это питание **не связано** с VDDD/VDDA и выбирается под конкретную схему усилителя возбуждения — в диапазоне **от 3,3 В до 5,0 В**.

Режимы работы

Режим задаётся полем EXO_mode:

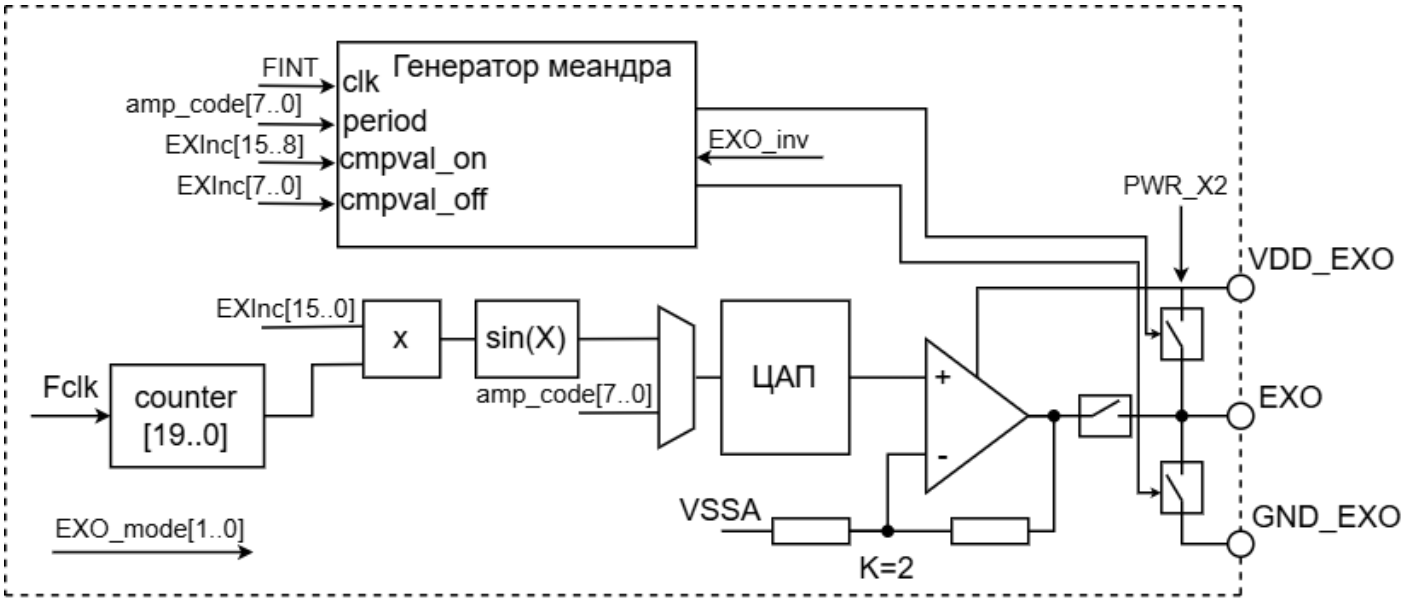
EXO_mode	Режим	Что выдаётся на EXO1/EXO2	Чем настраивается
`00`	Выключено	0 В (или VDDEXO1 при EXO_inv=1)	—
`01`	Меандр	импульсный сигнал	частота и скважность
`10`	Синусоида	синусоидальный сигнал (формируется ЦАП)	частота и амплитуда
`11`	Постоянный уровень	постоянное напряжение	уровень

> **ВАЖНО**
>
> Поле Amp_code в разных режимах отвечает за разное: в режиме **синусоиды** — это **амплитуда**, в режиме **меандра** — **делитель частоты**, в режиме **постоянного уровня** — само **выходное напряжение**.

Чтобы генератор заработал, нужно выбрать режим в EXO_mode и установить бит включения EXO1_en/EXO2_en регистра Mode_config.

Синусоидальный сигнал (EXO_mode = 10)

На рисунке ниже — структурная схема генератора.



Структурная схема генератора синусоидальных сигналов

У синусоиды **частота и амплитуда настраиваются отдельно**: частота — регистром EXInc, амплитуда — полем Amp_code.

Частота

В цифровом формирователе ведётся 20-битный счётчик фазы: на каждом такте `Fclk` он увеличивается на значение регистра EXInc. Когда счётчик переполняется (через $2^{20} = 1\,048\,576$ шагов), начинается новый период синусоиды. Чем больше `EXInc`, тем быстрее «бежит» фаза и тем выше частота:

$$f_{ex} = \frac{EXInc}{1\,048\,576} \cdot f_{clk} \tag{5}$$

Для нужной частоты возбуждения f_{ex} значение регистра вычисляется как

$$EXInc = \frac{f_{ex} \cdot 1\,048\,576}{f_{clk}}.$$

СТАРТ ПОСЛЕ СБРОСА

После сброса счётчик фазы инициализируется значением 1 048 000 ($\approx 2^{20} - 576$).

Амплитуда

Амплитуда задаётся 8-битным полем Amp_code (0...255): чем больше значение, тем больше размах сигнала. Внутри микросхемы каждый отсчёт синуса из таблицы (8-битное знаковое значение) умножается на коэффициент $(176 + AMP_{CODE})$; произведение масштабируется сдвигом на 9 разрядов (то есть делится на 512), а для компенсации нелинейности выходного буфера из результата вычитается поправка $(63 - \lfloor AMP_{CODE}/4 \rfloor)$ (деление на 4 — целочисленное, младшие 2 бита отбрасываются). Итоговое 8-битное значение поступает на ЦАП; напряжение на выводе EXO:

$$V_{EXO} = 2 \cdot V_{REF2P5} \cdot \frac{\sin\left(\frac{2\pi}{f_{ex}}\right) \cdot \frac{176 + AMP_{CODE}}{512} - \left(63 - \left\lfloor \frac{AMP_{CODE}}{4} \right\rfloor\right)}{256} \tag{6}$$

где $V_{REF2P5} = 2{,}5$ В — опорное напряжение ЦАП, множитель 2 — коэффициент усиления выходного буфера.

НА ПРАКТИКЕ

Формулу (6) считать вручную не нужно: амплитуду подбирают по осциллографу, постепенно увеличивая `Amp_code` до нужного размаха сигнала. Главное ограничение — **пиковое значение сигнала должно быть ниже питания выходного буфера** VDDEXO1/VDDEXO2 **минимум на 0,4 В**. Если превысить — буфер уйдёт в насыщение и появятся искажения.

Опорное напряжение ЦАП (REFDAC)

Опорное напряжение V_{REF2P5} для ЦАП берётся либо от внутреннего ИОН (бит VREF_DAC_en=1), либо подаётся извне на вывод REFDAC в диапазоне **2,5...3,3 В**. В обоих случаях вывод REFDAC шунтируется конденсатором 0,1 мкФ.

Фильтрация выходного сигнала (ФНЧ)

Сигнал на выводах EXO1/EXO2 формируется **8-битным ЦАП**: по уровню это 256 градаций, по времени — отсчёты, обновляемые каждый такт `Fclk` (формула (5)). На выходе микросхемы — **ступенчатый (дискретный) сигнал, без внутреннего сглаживания**.

Чем выше частота возбуждения, тем **меньше отсчётов приходится на один период** синусоиды, и тем заметнее ступенчатость и образы дискретизации (спектральные составляющие вблизи частоты обновления и её гармоник). Они порождают искажения формы сигнала и высокочастотные помехи в тракте усиления.

Поэтому **между выводом EXO и усилителем требуется внешний реконструкционный фильтр нижних частот (ФНЧ)**, который пропускает полезную синусоиду возбуждения и подавляет ступеньки и образы дискретизации. Типовой тракт (ЦАП → ФНЧ → усилитель → первичная обмотка датчика) приведён в разделе Рекомендуемая схема применения (типовая схема усиления сигнала возбуждения) — фильтр реализован прямо в усилительном каскаде.

> **ЭТО КАСАЕТСЯ ТОЛЬКО СИНУСОИДЫ**
>
> Ступенчатость и потребность в ФНЧ относятся к режиму синусоиды (`EXO_mode = 10`). В режиме меандра (`01`) и постоянного уровня (`11`) сигнал формируется иначе и ФНЧ для реконструкции не требуется.

Меандр (EXO_mode = 01)

В режиме меандра работает 8-битный счётчик, тактируемый системной частотой `FINT`.

Частота импульсов определяется значением Amp_code — это делитель частоты (счётчик считает от 0 до `Amp_code` включительно, то есть период составляет `Amp_code + 1` тактов

FINT):

$$F_{imp} = \frac{FINT}{Amp_code + 1} \tag{7}$$

Скважность (соотношение высокого/низкого уровня) задаётся двумя пороговыми значениями регистра EXInc: при достижении счётчиком значения `EXInc[15:8]` выход переключается на уровень VDDEXO1/VDDEXO2, при достижении `EXInc[7:0]` — на GNDEXO1/GNDEXO2. Это два независимых порога компаратора, задающих моменты переключения.

Дополнительные настройки регистра C1ExoStngs/C2ExoStngs:

- EXO_inv — инвертирует выход (действует только в режимах «выключено» `00` и «меандр» `01`; в режимах синуса и постоянного уровня игнорируется);
- PWR_X2 — увеличивает выходной ток коммутации в 2 раза (для низкоомной нагрузки);
- EXO_sync (в AFE_config) — синхронизирует счётчики двух генераторов.

Постоянный уровень (EXO_mode = 11)

В этом режиме на выходе формируется постоянное напряжение, пропорциональное Amp_code (0...255):

$$V_{EXO} = 2 \cdot V_{REF2P5} \cdot \left(\frac{AMP_{CODE}}{255} \right) \tag{8}$$

Режим удобен для настройки и калибровки: можно подать известное постоянное напряжение и проверить коэффициент усиления внешнего тракта. Как и для синусоиды, нужно следить, чтобы расчётное `Vexo` не превышало питание VDDEXO1/VDDEXO2.

Краткая шпаргалка по настройке

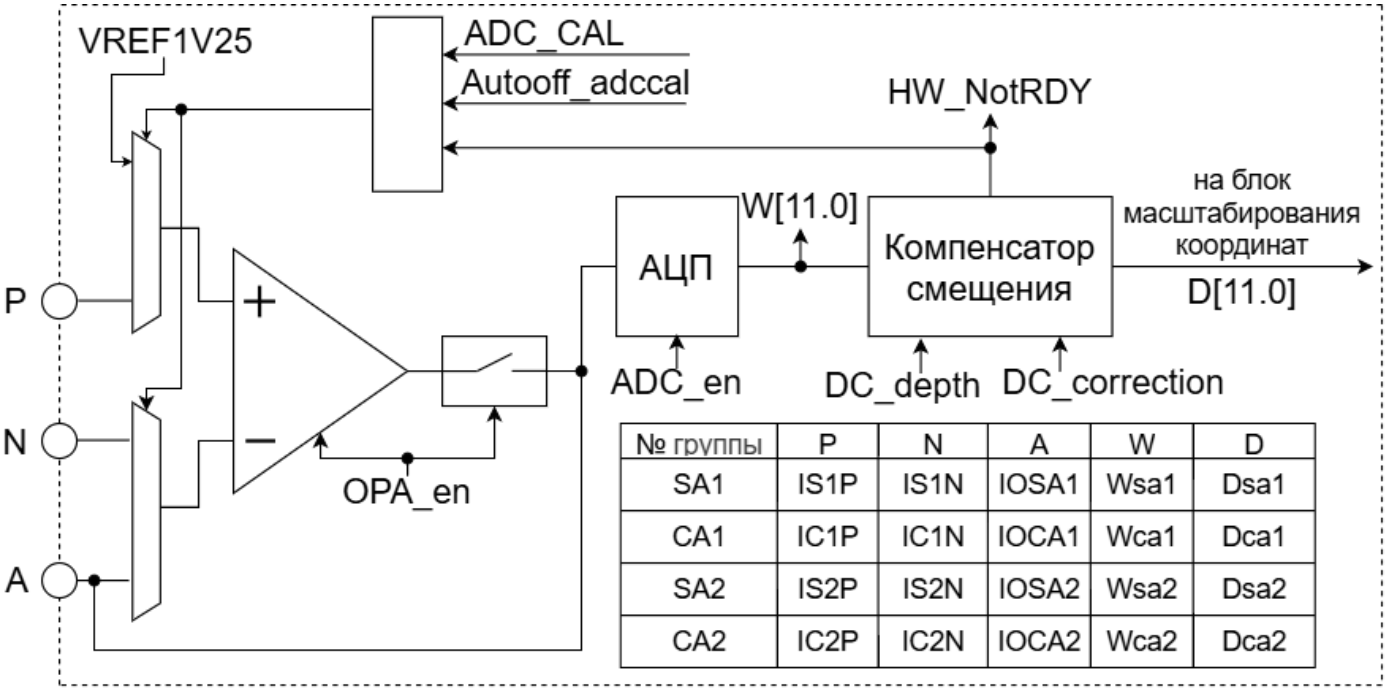
Что настроить	Как	Где
Включить генератор 1/2	бит `EX01_en` / `EX02_en` = 1	Mode_config
Режим (выкл/меандр/синус/постоянный)	`EXO_mode` = 00/01/10/11	ExoStngs
Частота синусоиды	`EXInc` (формула 5)	EXInc
Амплитуда синусоиды	`Amp_code` (формула 6, подбор по осциллографу)	ExoStngs
Частота меандра	`Amp_code` (формула 7)	ExoStngs
Скважность меандра	`EXInc[15:8]` / `EXInc[7:0]`	EXInc
Постоянный уровень	`Amp_code` (формула 8)	ExoStngs
Питание буфера	3,3…5,0 В на VDDEXO	VDDEXO1/VDDEXO2
Опора ЦАП	внутр. ИОН (`VREF_DAC_en=1`) либо внешний 2,5…3,3 В	REFDAC

> ⚠ **ГЛАВНОЕ ОГРАНИЧЕНИЕ ПО АМПЛИТУДЕ**
>
> Пиковое выходное напряжение генератора должно быть **минимум на 0,4 В ниже** питания выходного буфера (VDDEXO1/VDDEXO2). Превышение ведёт к насыщению буфера и искажению сигнала.

*Последнее обновление **21 сент. 2026 г.***

Входные группы преобразователя

В микросхеме реализовано 4 идентичные группы (`SA1`, `CA1`, `SA2`, `CA2`) преобразования аналогового сигнала в цифровой код.



№ группы	P	N	A	W	D
SA1	IS1P	IS1N	IOSA1	Wsa1	Dsa1
CA1	IC1P	IC1N	IOCA1	Wca1	Dca1
SA2	IS2P	IS2N	IOSA2	Wsa2	Dsa2
CA2	IC2P	IC2N	IOCA2	Wca2	Dca2

Входная группа преобразователя

Включение

По умолчанию буферные ОУ выключены, в этом случае подача сигналов на АЦП осуществляется через выводы IOSA1, IOCA1, IOSA2, IOCA2. Для использования встроенных ОУ требуется установить бит OPA_en. Для работы АЦП требуется установить бит ADC_en в единицу.

Компенсатор смещения

В преобразователе реализовано 4 идентичных компенсатора постоянной составляющей (по одному на входную группу). Каждый компенсатор — это **CIC-фильтр (каскадный интегратор-гребёнка) переменной глубины**: он накапливает заданное число выборок АЦП, вычисляет по ним среднее значение (постоянную составляющую смещения) и

вычитает его из последующих выборок. Удаление постоянной составляющей выполняется **до** масштабирования и коррекции амплитуды, поэтому настройка усиления не зависит от смещения среднего уровня.

Глубина усреднения (объём выборки для расчёта смещения) задаётся полем DC_depth: накопитель усредняет $2^{7+\text{ext}}$ отсчётов, где `ext` определяется значением `DC_depth` — при `DC_depth[3]=0` ext = `DC_depth[2:0]` = 0...7, при `DC_depth[3]=1` ext = 9 + `DC_depth[2:0]` = 9...16. Объём выборки меняется от $2^7 = 128$ (`DC_depth=0`) до 2^{23} (`DC_depth=15`). Чем больше объём и время калибровки, тем точнее определяются коэффициенты смещения.

Компенсатор включается/выключается битом DC_correction: при `DC_correction=1` смещение вычисляется и вычитается; при `0` компенсатор выключен, и сигнал АЦП проходит без удаления постоянной составляющей.

Компенсатор настоятельно рекомендуется использовать для модулированных сигналов, т.к. смещения вызывают шум и погрешности в работе блока восстановления частоты и в определении выходной координаты следящим контуром.

Автокалибровка и готовность контура

Для преобразования сигналов без модуляции необходимо установить бит DC_carrier в единицу. В этом случае рекомендуется включить режим автокалибровки АЦП — установить биты ADC_CAL и Wait_1offset. Автокалибровка определяет и компенсирует смещение нуля входных трактов (ОУ и АЦП) при включении питания: в режиме калибровки входы ОУ отключаются от выводов микросхемы и подключаются к напряжению `VREF1P25`, а коды АЦП поступают на компенсатор, пока бит ADC_CAL установлен в 1. Для автоматического выключения калибровки по завершении служит бит Autooff_adccal.

Цепочка готовности следящего контура:

- расчёт первого смещения длится один цикл объёма выборки (см. выше); пока он не завершён, флаг HW_NotRDY = 1;
- при Wait_1offset = 1 блок восстановления опорной частоты и следящий контур удерживаются в сбросе до расчёта первых коэффициентов смещения (до сброса `HW_NotRDY`);
- для модулированных сигналов автокалибровка не требуется — компенсатор работает непрерывно в процессе преобразования.

*Последнее обновление **3 июл. 2026 г.***

Блок масштабирования и преобразования координат

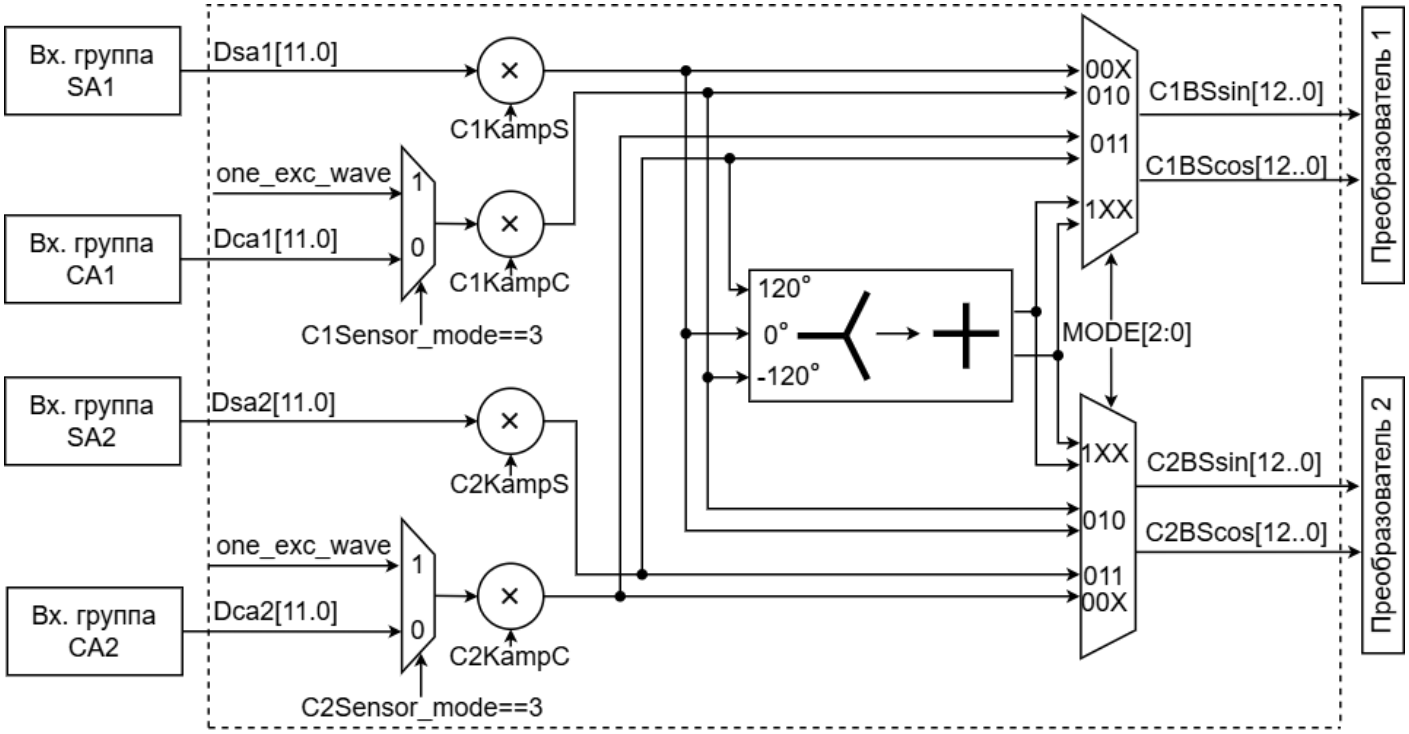


Схема блока масштабирования и преобразования координат

Блок масштабирования позволяет изменять масштаб сигналов после аналого-цифрового преобразования. Масштабирование предназначено для коррекции ошибки усиления ALIT и аналоговых цепей на плате, а также погрешностей датчика. Масштаб сигналов задается в регистрах C1KampS (вход *IOSA1*), C1KampC (вход *IOCA1*), C2KampS (вход *IOSA2*), C2KampC (вход *IOCA2*). Значения по умолчанию в этих регистрах предполагают сигнал на входе микросхемы амплитудой 2,5 В (пик-пик). Блок производит преобразование координат из трёхфазной системы координат в декартову систему координат для режима Сельсин. Выбор режима Сельсин производится установкой Mode**[2:0]=4** (регистр AFE_config). В данном режиме используются входы, приведенные в таблице 5.

Таблица 5 – Использование входов микросхемы в зависимости от режима преобразования

Вход	Режим Сельсин	Режим СКВТ
IOSA1	Вход преобразователя sin(Θ + 0°)	Вход 1-го преобразователя sin
IOCA1	Вход преобразователя sin(Θ + 120°)	Вход 1-го преобразователя cos
IOSA2	Вход преобразователя sin(Θ - 120°)	Вход 2-го преобразователя sin
IOCA2	Не используется	Вход 2-го преобразователя cos

Значения битов Mode[2:0] регистра AFE_config определяют режимы совместной работы двух преобразователей (таблица ниже).

Таблица 6 – Режимы совместной работы двух преобразователей

Регистр Mode[2:0]	**Режим**
000	Оба преобразователя включены и работают независимо. Первый преобразователь подключен к выводам IOSA1, IOCA1, EXI1. Второй преобразователь подключен к выводам IOSA2, IOCA2, EXI2.
010	Оба преобразователя включены и работают параллельно. Преобразователь 2 использует входы преобразователя 1 (IOSA1, IOCA1, EXI1).

Регистр Mode**[2:0]**	**Режим**
011	Оба преобразователя включены и работают параллельно. Преобразователь 1 использует входы преобразователя 2 (IOSA2, IOCA2, EXI2).
100	Режим Сельсин. Оба преобразователя включены и работают параллельно. На входы преобразователей подается сигнал со входов IOSA1, IOCA1, IOSA2, EXI1. Включено преобразование координат из трёхфазной в декартову.
Примечание. В режиме подачи внешнего опорного сигнала Ex_source==01 преобразователь 2 использует опорный сигнал с вывода EXI1.	
При использовании двух преобразователей параллельно пользователь может задать в регистре CMP_lth максимально допустимое различие между каналами. Если результаты двух преобразователей отличаются на величину большую, чем записана в регистр CMP_lth, выставляется бит Not_Equal в регистре Stat_main. В зависимости от настройки в регистре Mask_Stat это также может установить в состояние логического нуля выход **Ready** микросхемы.	

Для экономии энергии любой из преобразователей может быть отключен установкой битов CONV1_en, CONV2_en в регистре Mode_config в состояние логического нуля. При этом для корректной работы выхода **Ready** необходимо бит MSK_Not_Equal также установить в состояние логического нуля.

Коррекция амплитуды: формулы и расчёт коэффициентов

Цифровой аргумент, поступающий в следящий контур (arg_sin_kontur1, arg_cos_kontur1), получается умножением оцифрованного сигнала АЦП на коэффициент усиления. Безразмерный **коэффициент усиления** каждого канала равен значению регистра, делённому на 1024:

$$k = \frac{\mathrm{KampS}}{1024} \quad \left(k = \frac{\mathrm{KampC}}{1024}\right), \qquad \mathrm{arg} = \mathrm{Dadc} \cdot k$$

где arg — аргумент контура соответствующего канала, Dadc — оцифрованный сигнал канала АЦП (Dadc_sin, Dadc_cos) **после удаления постоянной составляющей** (знаковый). Аппаратно умножение выполняется как произведение 13-разрядного знакового сигнала на значение регистра с последующим делением на 1024 (взятие битов [22:10] произведения). Постоянная составляющая удаляется отдельным DC-блоком **до** масштабирования, поэтому настройка амплитуды не зависит от смещения среднего — отсюда двухэтапная процедура калибровки (амплитуда, затем смещение), описанная в разделе 8 Методики.

При значении по сбросу KampS = KampC = `0x0400` (1024) усиление $k = 1{,}0$.

Соответствие «напряжение → код». АЦП — 12-разрядный, опорное напряжение VREF2P5 = 2,5 В, диапазон входа 0…2,5 В:

Параметр	Значение
Младший разряд (1 LSB)	2,5 В / 4096 ≈ **0,610 мВ**
Синфазный уровень (среднее)	1,25 В → код **2048**
Номинальный сигнал 2,0 В пик-пик (±1 В)	коды 410…3686 (±1638 от среднего) — это **0,8** дифференциального диапазона АЦП (±1 В из доступных ±1,25 В)

Значения по умолчанию (`0x0400`, $k = 1{,}0$) рассчитаны именно на номинальный входной сигнал 2,0 В пик-пик. Соответственно, в терминах нормированной амплитуды контура

номинальный сигнал составляет ±0,8 (см. раздел Полоса пропускания контура).

Расчёт коэффициента при коррекции амплитуды. Если измеренный размах сигнала на входе контура R (по arg_sin_kontur1/arg_cos_kontur1; методика измерения — в разделе 8.6 Методики) отличается от целевого (размах модели 4096, ±2048), коэффициент пересчитывается пропорционально — штрихом обозначено новое (рассчитанное) значение:

$$\mathrm{KampS}' = \mathrm{KampS} \cdot \frac{4096}{R_S}, \qquad \mathrm{KampC}' = \mathrm{KampC} \cdot \frac{4096}{R_C}, \qquad k' = \frac{\mathrm{KampS}'}{1024}$$

Пример. Размах входного сигнала ±600 отсчётов ($R = 1200$) при исходном `0x0400` (1024):

$$\mathrm{KampS}' = 1024 \cdot \frac{4096}{1200} \approx 3495 \quad (k' \approx 3{,}41)$$

На практике достаточно округлённого значения — например, KampS = 3072 ($k = 3{,}0$) даёт размах ≈ ±1800; визуальный пример совмещения входных сигналов с моделью приведён в разделе Режим детектора.

> **КОНТРОЛЬ ПЕРЕПОЛНЕНИЯ И ПОРОГОВ**
>
> При больших значениях KampS произведение может выйти за разрядность аргумента контура (13 бит, ±4095) — при этом выставляется флаг переполнения коррекции амплитуды. Также необходимо следить за пороговыми компараторами амплитуды (флаги **UIN_High/UIN_Low** по метрике C1Amp_metric, номинальное значение 400) и за тем, чтобы мгновенный сигнал на входах АЦП не выходил за диапазон 0…2,5 В (раздел 6 Методики подключения и настройки).

*Последнее обновление **3 июл. 2026 г.***

Блок восстановления сигнала опорной частоты

Точность определения координаты в следящем контуре базируется на соответствии виртуальных сигналов входным сигналам датчика. Индуктивная природа кодирования угла поворота в СКВТ подразумевает использование переменных сигналов. В типовой схеме фаза сигнала сформированная генератором проходит ряд преобразований - фильтр в усилителе, механический сдвиг фазы в СКВТ, фильтры и усилители входных каскадов, АЦП и задержки в цифровых вычислениях, что приводит к значительному изменению фазы принимаемых сигналов IOSA1/IOSA2 и IOCA1/IOCA2 по сравнению с EXO1/EXO2. Изменение фазы сигналов имеет температурную зависимость и при изменении температуры вносит значительную погрешность в результат преобразования (при отключенном блоке восстановления сигнала опорной частоты).

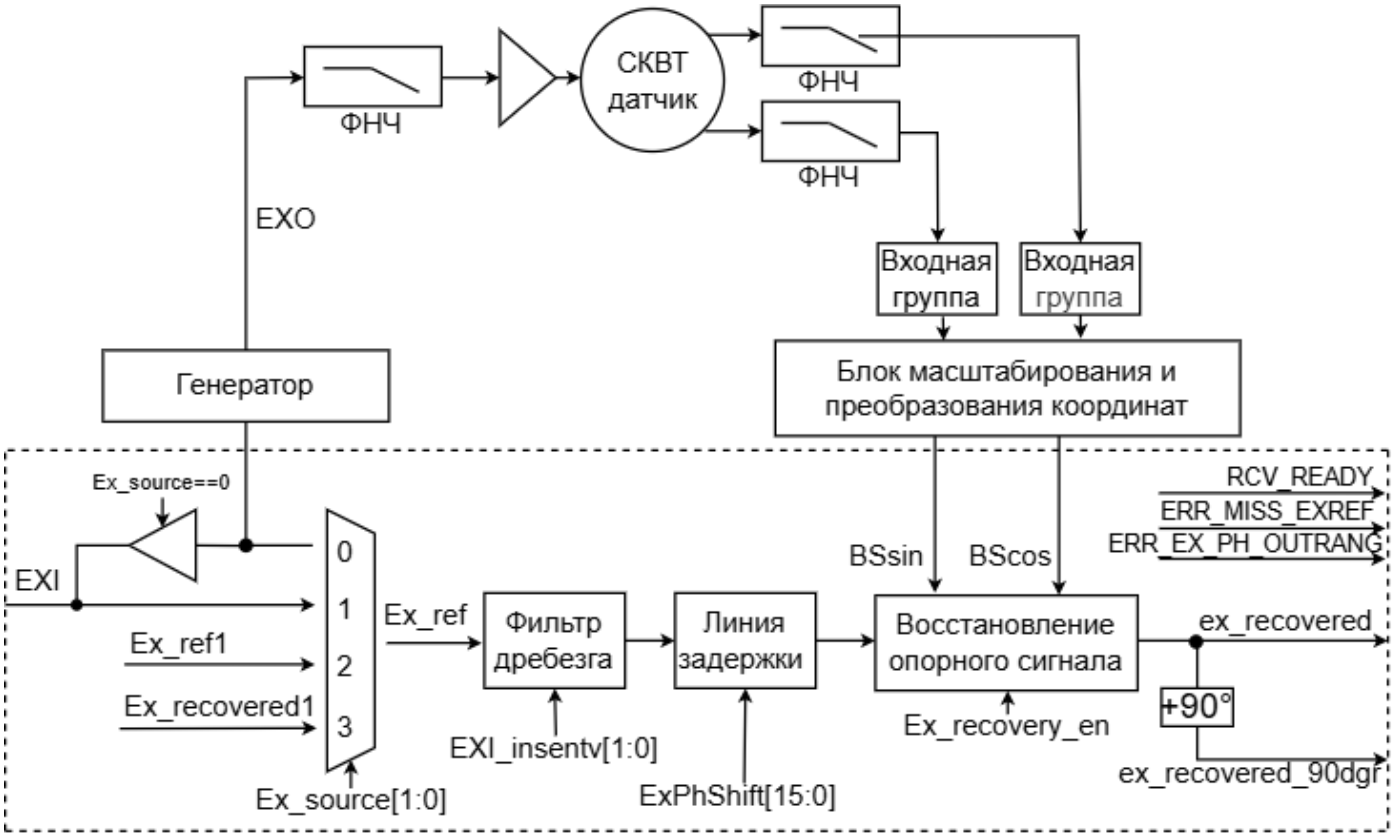


Схема блока восстановления опорного сигнала в следящем контуре

Чтобы избавиться от погрешности, вызванной этой задержкой, а также подавить квадратурные составляющие сигналов, возникающие при движении вала датчика, вместо

сигнала возбуждения микросхема использует в модели датчика восстановленный сигнал опорной частоты **ex_recovered**. Сигнал **ex_recovered** вычисляется из принимаемых сигналов IOSA1/IOSA2, IOCA1/IOCA2 и вспомогательного сигнала Ex_ref, который необходим для определения квадранта, в котором находится СКВТ. Суть восстановления частоты сводится к автоматическому сдвигу Ex_ref к моментам перехода через нуль большего из сигналов IOSA1/IOSA2, IOCA1/IOCA2. Поэтому для корректной работы блока сдвиг фазы между сигналом Ex_ref и сигналами IOSA1/IOSA2, IOCA1/IOCA2 должен находиться в пределах одного квадранта ($\pm 45^\circ$). При сильной зависимости сдвига фазы сигналов от температуры, рекомендуется устанавливать в пределах $\pm 20^\circ$. Для проведения настройки в таблице 7 указано соответствие фазы сигнала ex_ref к входным с СКВТ сигналам, чтобы выходная координата следящего контура совпадала с положением ротора СКВТ.

Таблица 7 – Полярность сигналов СКВТ в зависимости от угла поворота вала

Квадрант, в котором находится СКВТ/сельсин	**Знак сигнала** IOSA1/IOSA2	**Знак сигнала** IOCA1/IOCA2
I ($0 - 90^\circ$)	Совпадает с Ex_ref	Совпадает с Ex_ref
II ($90^\circ - 180^\circ$)	Совпадает с Ex_ref	Инверсный к Ex_ref
III ($180^\circ - 270^\circ$)	Инверсный к Ex_ref	Инверсный к Ex_ref
IV ($270^\circ - 360^\circ$)	Инверсный к Ex_ref	Совпадает с Ex_ref

Контроль разницы фаз между ex_ref и сигналами IOSA1/IOSA2 IOCA1/IOCA2 лучше осуществлять программным способом с помощью буферизированного считывания ячейки C1AdcS/C2AdcS, в которой содержится старшие разряды входных и опорных сигналов поступающих непосредственно на преобразователь.
Для точной коррекировки сдвига фазы Ex_ref должен использоваться регистр C1ExPhShft/C2ExPhShft, который в тактах **Fclk** (частота работы следящего контура, АЦП и ЦАП) задает задержку Ex_ref. Выбор источника Ex_ref осуществляется с помощью битов Ex_source в регистре C1InputStngs/C2InputStngs (таблица 8):

Таблица 8 – Источники опорного сигнала Ex_ref

№ преобразователя	Ex_Source	Источник ex_ref
1	0	с генератора опоры 1
1	1	со входа EXI1
1	2	с генератора опоры 1
1	3	со входа EXI1
2	0	с генератора опоры 2
2	1	со входа EXI1
2	2	ex_ref с преобразователя 1
2	3	ex_recovered с преобразователя 1

Примечание. Преобразователь 2 не может принять внешний опорный сигнал с вывода EXI2, при выборе режима Ex_source=01 для преобразователя 2, сигнал опоры будет браться с вывода EXI1. При использовании внутренних источников Ex_ref (Ex_source!=1) вывод EXI1/EXI2 становится выходом сигнала Ex_ref. Этот выход также может использоваться для работы нескольких микросхем от одного источника опорной частоты.

Для введения нечувствительности к ложным импульсам ex_ref используется поле EXI_insentv (биты [14:13] регистра C1InputStngs/C2InputStngs): `0` — фильтр выключен, `1` — усреднение за 2 отсчёта, `2` — за 4 отсчёта, `3` — за 8 отсчётов.

Если опорный сигнал ex_ref расходится с входным сигналом более чем 40°, то сформируется флаг неисправности EX_PH_OUTRANGE. При отсутствии переключений сигнала ex_ref в течение 65535 тактов Fclk сформируется флаг ошибки MISS_EXREF (он сбрасывается при появлении опоры).

При наличии большого смещения уровня входных сигналов рекомендуется отложить включение блока восстановления опорной частоты до момента, когда будут рассчитаны первые коэффициенты смещений. Для этого требуется записать в бит Wait_1offset значение "1". В этом случае блок восстановления частоты начнет расчет с первого полного периода входного сигнала с момента наступления флага HW_NotRDY. При

включении преобразователя и до расчета восстановленного сигнала флаг готовности RCV_notRDY удерживается в единице. При включенном блоке восстановления частоты и отсутствии его готовности следящий контур удерживается в сбросе.

При установке бита DC_carrier восстановление опоры фактически отключено: сигнал **ex_recovered** = 0, а флаг готовности RCV_NotRDY принудительно сброшен (готов) — поэтому следящий контур не ждёт захвата опоры и сразу обрабатывает немодулированный сигнал. Если частота вращения датчика больше $\frac{1}{8}$ от частоты опорного сигнала, схема восстановления должна быть отключена (Ex_recovery_en=0), и использован внешний или внутренний сигнал опорной частоты. При этом сдвиг фазы этого сигнала должен быть точно подогнан к фазе IOSA1/IOSA2 и IOCA1/IOCA2. Остаточный сдвиг фазы влияет на шум преобразования, а также погрешность при движении датчика. Данный режим работы не рекомендуется использовать, т.к. возникает зависимость результата преобразования от температуры.

Внимание! При неправильной настройке блока восстановления опорного сигнала преобразователь может давать ошибку 180°, при этом флаги ошибок будут сброшены.

Настройка и анализ работы блока

Практическая процедура включения, контроля и подстройки блока — полностью по цифровым отсчётам и флагам, без осциллографа. Пошаговая методика в составе общей настройки приведена в разделе 7.г Методики подключения.

1. Подготовка и включение.

a. Выберите источник опорного сигнала Ex_ref битами Ex_source (таблица 8) и убедитесь, что опора присутствует (иначе выставится MISS_EXREF). b. При большом смещении уровня входных сигналов установите Wait_1offset = 1, чтобы блок начал расчёт после вычисления первых коэффициентов смещений. c. Включите блок: Ex_recovery_en = 1. Включать можно сразу — работе он не мешает.

2. Контроль захвата фазы (цифровой).

a. Считывайте флаг RCV_NotRDY регистра C1Stat. В течение первого периода входного сигнала RCV_NotRDY = 1 и следящий контур удерживается в сбросе; при RCV_NotRDY = 0 фаза захвачена. b. Восстановленную опору читайте как бит Ex_recovered_vs (бит 14 регистра C1OutS, адрес 20). c. Знак входного сигнала и исходную опору читайте из

регистра C1AdcS (адрес 18) — лучше буферизованным считыванием: в одном слове содержатся Ex_ref (бит 0), Ex_shifted (бит 14), Dadc_sin и старшие биты msb_arg_sin/msb_arg_cos. d. Критерий корректной работы: переходы Ex_recovered_vs совпадают с переходами входного сигнала через нуль (сменой знака msb_arg_sin/msb_arg_cos) — восстановленная опора синхронна со знаком сигнала АЦП.

3. Подстройка фазы.

a. Флаг EX_PH_OUTRANGE регистра C1Stat показывает, что исходная опора Ex_ref разошлась с восстановленным сигналом более чем на ±40°. b. Регистром C1ExPhShft (задержка Ex_ref в тактах Fclk) сдвигайте опору, пока EX_PH_OUTRANGE = 0. c. Дополнительно можно контролировать совмещение Ex_shifted со старшим битом сигнала АЦП (msb_arg_sin/msb_arg_cos) по буферизованному чтению C1AdcS.

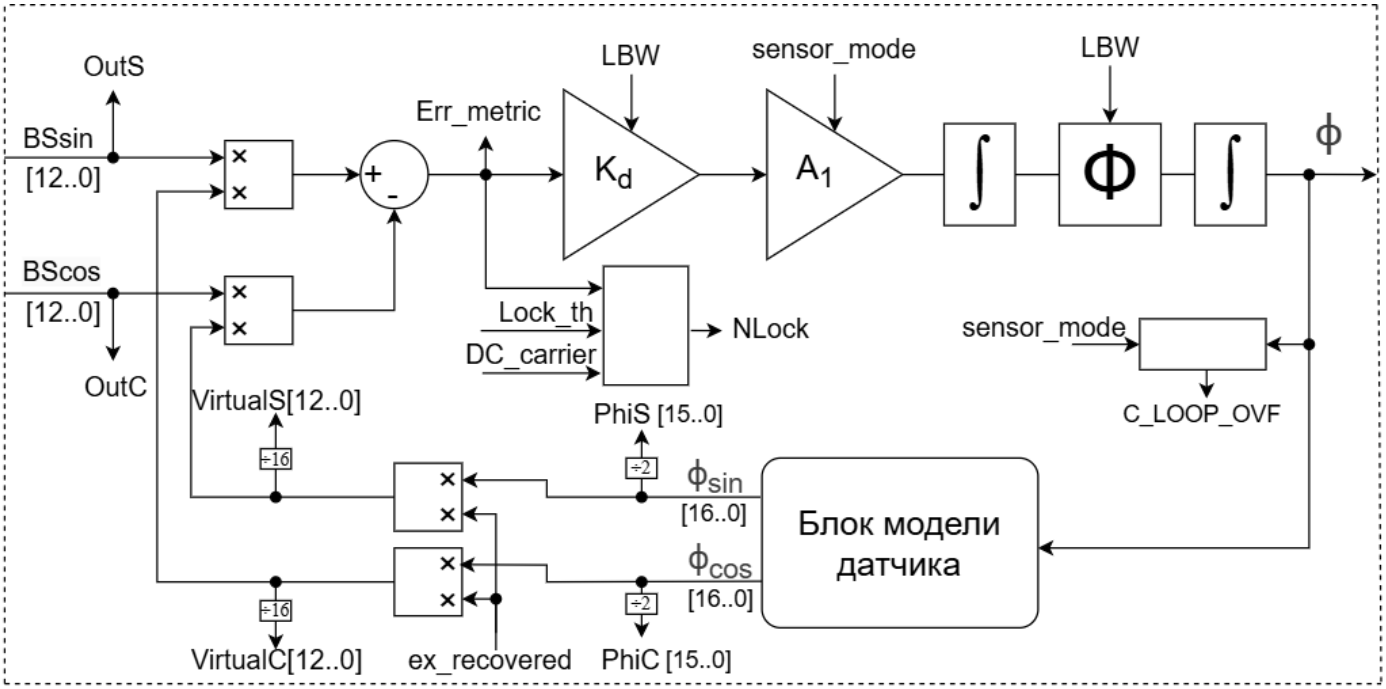
4. Контроль неисправностей (регистр C1Stat).

Флаг	Значение = 1
RCV_NotRDY	захват фазы не завершён (следящий контур в сбросе)
EX_PH_OUTRANGE	сдвиг фазы между Ex_ref и восстановленным сигналом > ±40° — подстройте C1ExPhShft
MISS_EXREF	опорный сигнал отсутствует — проверьте источник Ex_ref (таблица 8)

5. Ограничения. При частоте вращения датчика больше 1/8 от частоты опорного сигнала блок должен быть отключён (Ex_recovery_en = 0) — в этом случае используется внешний или внутренний Ex_ref с точной фазировкой (см. выше); при DC_carrier = 1 восстановление опоры отключено (контур работает с немодулированным сигналом, без захвата опоры).

*Последнее обновление **3 июл. 2026 г.***

Следящий контур



Следящий контур

Для анализа качества внесенных настроек и точности работы следящего контура выведено множество контрольных точек, доступных для чтения по интерфейсу SPI:

- Регистры OutC/OutS - вход следящего контура, выход блока масштабирования координат. В этой точке размах изменения амплитуды должен быть равен двум единичным амплитудам, т.е. значению 4096 (настраивается коэффициентами KampC/KampS). Для анализа амплитуды модулированных сигналов рекомендуется использовать микропрограмму детектора амплитуды на CPU и натуральное вращение датчика. В контур поступают 13 битные значения, поэтому наличие смещений не переполняет разрядную сетку.
- Регистры PhiC/PhiS- выход модели датчика. В этой точке смещение среднего уровня должно быть равно смещению OutC/OutS (настраивается коэффициентами).
- Регистры VirtualC/VirtualS - выход модели датчика, модулированные сигналом восстановленной частоты. В этой точке сигналы должны быть очень близки к входным сигналам по фазе и амплитуде. Для анализа использовать буферизированную запись пар первичных сигналов `[OutS, VirtualS]` или `[OutC, VirtualC]`.

- Регистр Err_metric - ошибка рассогласования виртуальных и реальных датчиков. Может быть использован для анализа уровня соответствия коэффициентов реальным сигналам датчика, а также для анализа уровня ошибки следящего контура при угловых ускорениях вращения датчика. Сигнал Err_metric используется для автоматического определения неисправности и формирования флага NLock. NLock формируется если Err_metric превышает заданный порог Lock_th (для немодулированных сигналов). Для модулированных сигналов для сравнения берется модуль максимальной амплитуды Err_metric за каждый период.

Для датчиков с конечным диапазоном угла в контуре отслеживается переполнение интегратора; для этого введён флаг C_LOOP_OVF, который сигнализирует о переполнении суммы второго интегрирующего звена. Возможна некорректная работа микросхемы.

*Последнее обновление **3 июл. 2026 г.***

Блок модели датчика

В микросхему заложено несколько моделей вычисления виртуальных сигналов в зависимости от типа датчика. Выбор и настройка параметров модели дает возможность приблизить виртуальные сигналы к реальным, сокращая разницу между ними, что снижает нелинейности и шум координаты на выходе следящего контура. Если погрешности датчика и схемы не известны, регистры должны оставаться в значениях по умолчанию. Выбор типа модели осуществляется битами Sensor_mode.

Модель датчика в режиме СКВТ/сельсин

При подключении СКВТ или сельсина используется модель СКВТ/сельсин (Sensor_mode==0). Для сельсина сигналы преобразуются в декартовую систему координат (аналогичную СКВТ) блоком масштабирования и преобразования координат.

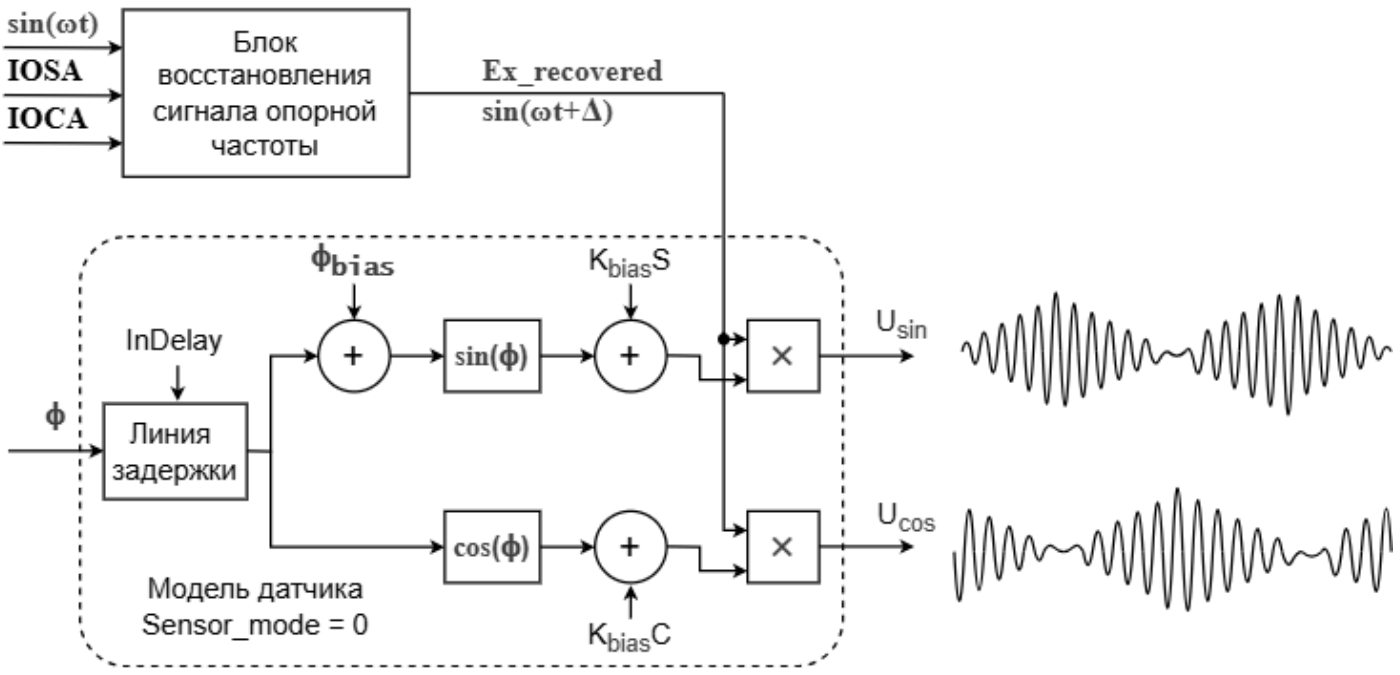


Схема модели СКВТ/сельсин (Sensor_mode==0)

Модель датчика в режиме СКВТ/сельсин использует восстановленный сигнал опорной частоты **Ex_recovered** (см. подраздел 5.6 «Блок восстановления сигнала опорной частоты») и вычисленный угол ϕ для расчета сигналов по формулам:

$$U_S = Ex_recovered \cdot \left(\sin\left(\phi' + \phi bias \cdot \frac{\pi}{65536} \right) + KbiasS \cdot \frac{0,25}{32768} \right), \quad (9)$$

$$U_C = Ex_recovered \cdot \left(\cos(\phi') + KbiasC \cdot \frac{0,25}{32768} \right), \quad (10)$$

где:

- InDelay – фиксированная дополнительная погрешность к групповой задержке сигналов в фильтрах на плате и в датчике. Основная часть групповой задержки сигнала Uex в фильтрах и датчике имитируется с помощью операции восстановления до Ex_recovered в блоке восстановления сигнала опорной частоты.
- ϕ' – угол в диапазоне $[0 \div 2\pi]$, вычисленный на предыдущем шаге, задержанный на InDelay тактов f_{clk}.
- $K_{\text{bias}}S, K_{\text{bias}}C$ – коэффициенты смещения нуля по каналам sin и cos (знаковое число в дополнительном коде);
- ϕ_{bias} – неортогональность осей X и Y.

Модель ЛРДТ по 5-проводной схеме

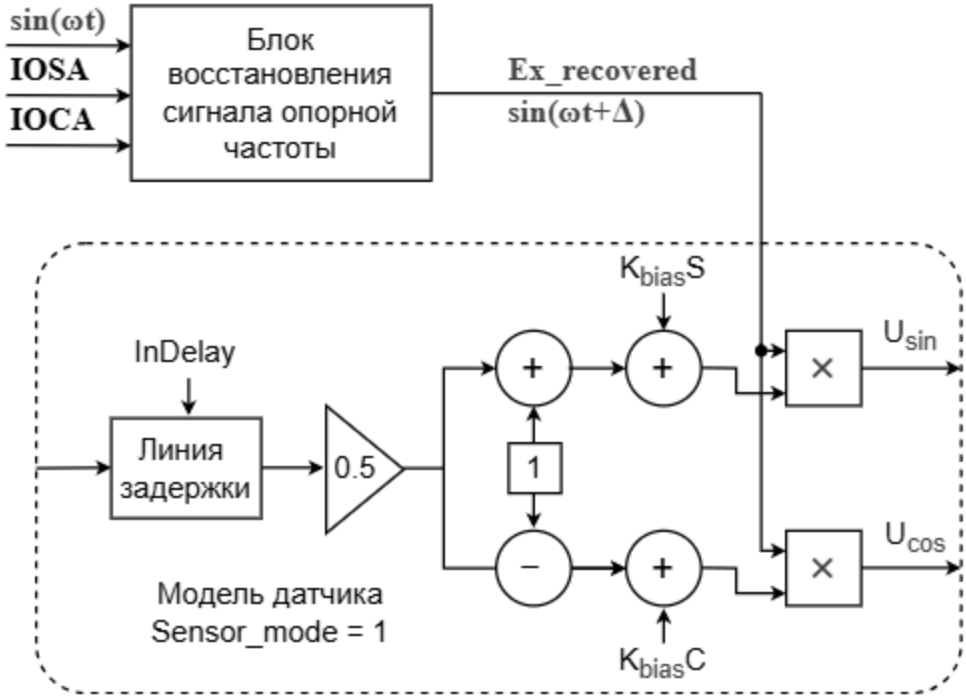


Схема модели ЛРДТ с двумя независимыми выходами (Sensor_mode==1)

$$U_S = Ex_recovered \cdot \left(1 + 0,5\phi + KbiasS \cdot \frac{0,25}{32768}\right), \tag{11}$$

$$U_C = Ex_recovered \cdot \left(1 - 0,5\phi + KbiasC \cdot \frac{0,25}{32768}\right), \tag{12}$$

где:

- **KbiasS, KbiasC** – коэффициенты смещения нуля по каналам S и C соответственно (знаковое число в дополнительном коде);
- ϕ – координата в диапазоне от -1 до $+1$, вычисленная на предыдущем шаге, задержанная на InDelay тактов f_{clk}.
- Умножение координаты ϕ на 0.5 растягивает выход контура на рабочий диапазон датчика, у которого наведенная ЭДС выходных независимых обмоток не меняет своей полярности.

---

*Последнее обновление **22 мая 2026 г.***

Полоса пропускания контура

Полоса пропускания преобразователя

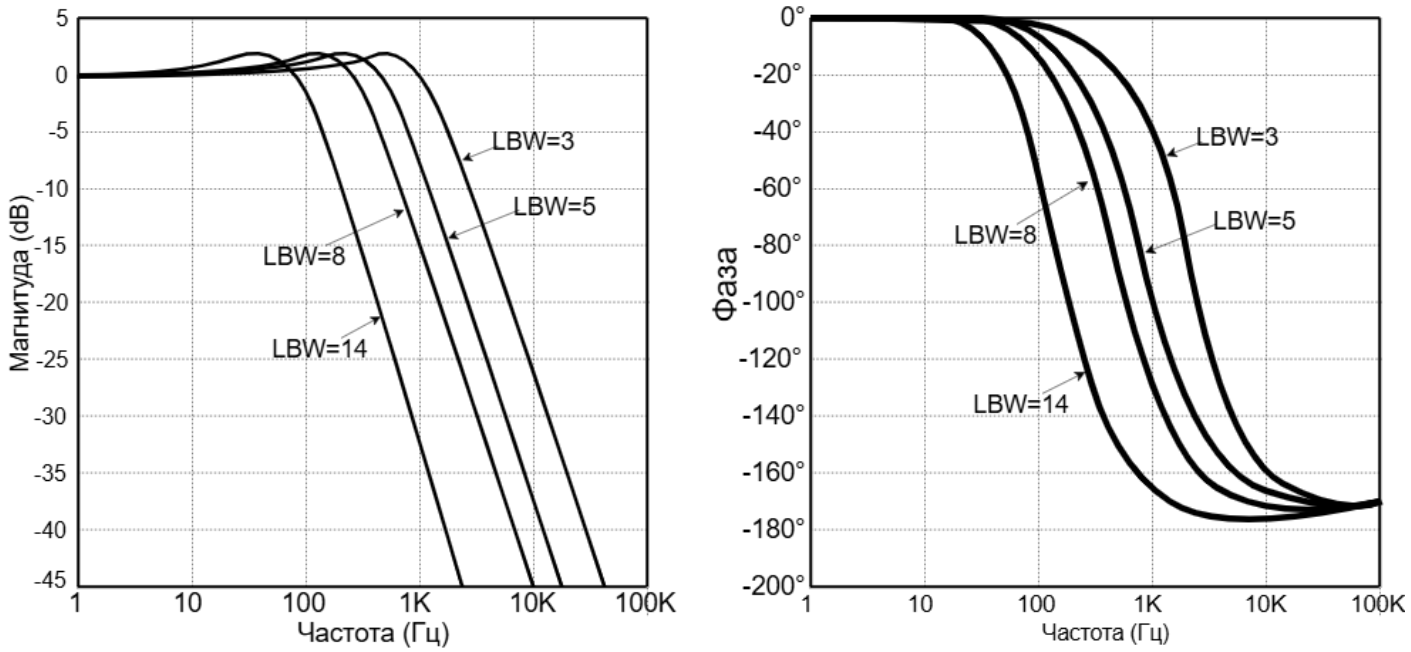
Полоса пропускания контура настраивается пользователем путем установки значения в битах LBW регистров C1KonturStngs/C2KonturStngs.

Таблица 9 – Полоса пропускания преобразователя

LBW	Полоса пропускания, Гц	Разрядность, бит (в лабораторных условиях, fex = 10 кГц)	Длительность переходного процесса, мс
3	5617	10	0.8
4	3984	12	1
5	2840	12	1.5
6	2015	12	2
7	1428	12	3
8	1011	13	4
9	716	13	6
10	507	14	8
11	358	14	12
12	253	14	16

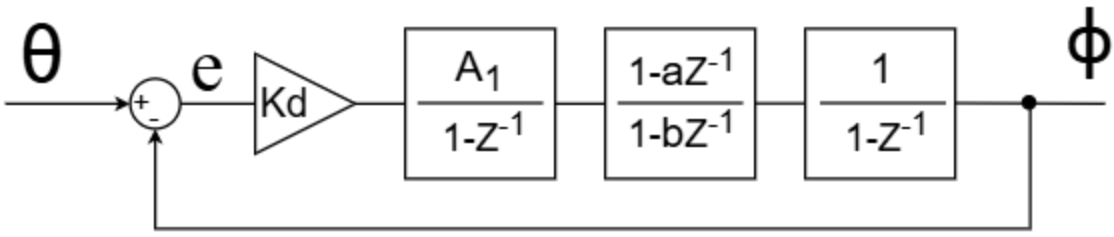
LBW	Полоса пропускания, Гц	Разрядность, бит (в лабораторных условиях, fex = 10 кГц)	Длительность переходного процесса, мс
13	179	14	24
14	127	15	32
15	89	15	48
16	63	16	64
17	45	16+	96
18	32	16+	128
19	22.4	16+	192
20	15.9	16+	256

Выбор значения LBW меняет коэффициенты петлевого фильтра таким образом, чтобы обеспечить переходной процесс, близкий по форме к критическому, при номинальных сигналах на входах микросхемы. При выборе полосы пропускания контура LBW необходимо найти компромисс между требуемой разрядностью и динамикой преобразования. Чем уже полоса пропускания контура, тем большая ошибка возникает при ускорении вала, и тем длительнее переходной процесс первоначального нахождения угла. В то же время, чем шире полоса пропускания контура, тем меньшую разрядность можно получить на выходе преобразователя. Значение полосы пропускания контура, записываемое в регистр LBW, выбирается из таблицы 9 или настраивается экспериментально, в зависимости от используемой схемы подключения преобразователя, необходимой динамики преобразования и разрядности. Если частота цикла преобразования fclk отличается от номинальной (1024 кГц), полоса пропускания контура меняется пропорционально изменению частоты.



АЧХ и ФЧХ контура для разных LBW

Форма переходного процесса также остается постоянной, изменяется только время переходного процесса. Вид переходного процесса в контуре при номинальной амплитуде входного сигнала близок к критическому.



Контур 1

Передаточная функция открытого контура (с незамкнутой обратной связью) выражается следующей функцией

$$W(z) = \frac{A_1 \cdot K_d}{(1 - z^{-1})^2} \cdot \frac{(1 - a \cdot z^{-1})}{(1 - b \cdot z^{-1})} = \frac{A_1 \cdot \left[\frac{\pi}{2^{8+LBW}}\right]}{(1 - z^{-1})^2} \cdot \frac{1 - \left[\frac{2^{6+0.5LBW} - 1}{2^{6+0.5LBW}}\right] \cdot z^{-1}}{1 - \left[\frac{2^{6+0.5LBW} - 10}{2^{6+0.5LBW}}\right] \cdot z^{-1}}, \quad (13)$$

где LBW – значение, записанное в регистр LBW[4:0].

A_1 принимает следующие значения:

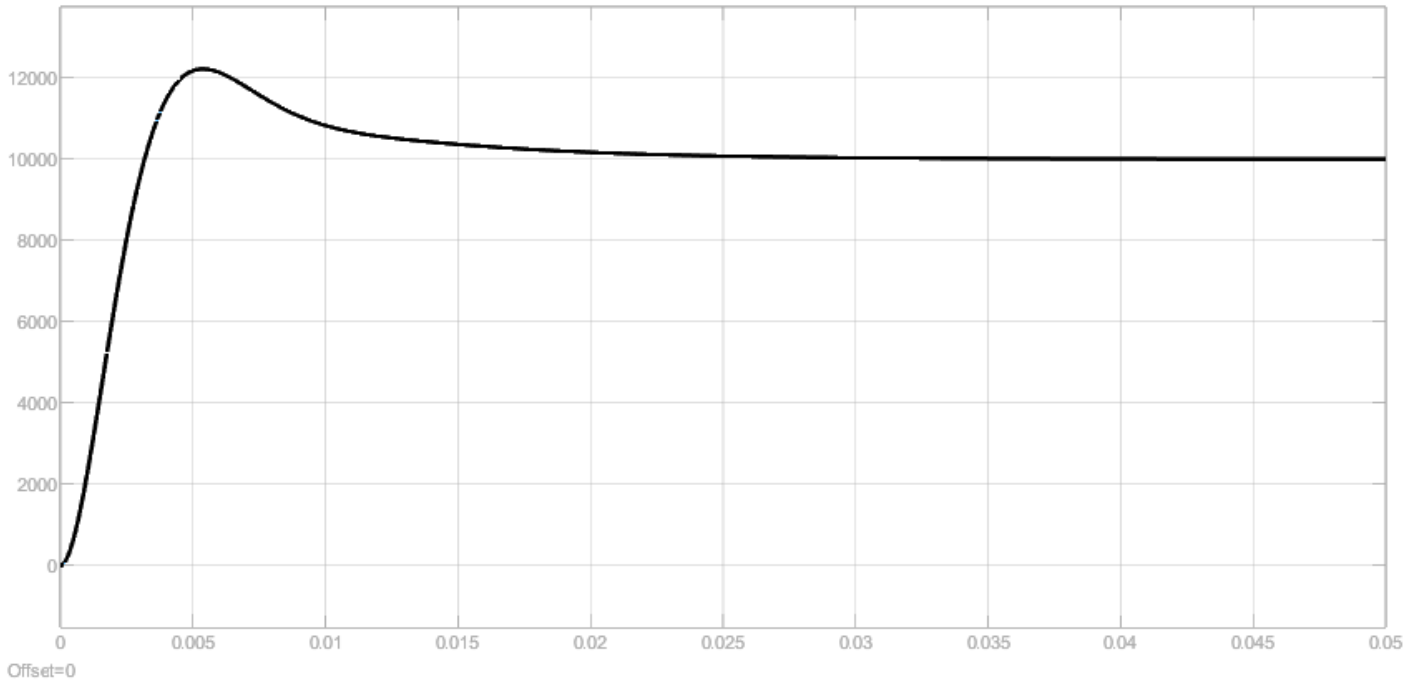
- $A_1 = 1$ при $Sensor_mode = 0$;
- $A_1 = 6\ (\sim 4 \cdot \sqrt{2})$ при $Sensor_mode == 1$;

- $A_1 = 3$ $(\sim 2 \cdot \sqrt{2})$ при $Sensor_mode == 2$.

Если установлен бит $DC_carrier$, то:

- $A_1 = 0,75$ $(\sim \frac{1}{\sqrt{2}})$ при $Sensor_mode == 0$;
- $A_1 = 4$ при $Sensor_mode == 1$;
- $A_1 = 2$ при $Sensor_mode == 2$.

Формула (13) верна для максимальной амплитуды сигнала на входах преобразователя $\pm 0,8$ (в микросхеме $\pm 0,8$ соответствует сигналу на входе микросхемы $2,0$ В [пик-пик] и значениям в регистрах юстировки по умолчанию). Масштабирование сигналов осуществляется в блоке масштабирования и преобразования координат.



Переходной процесс в замкнутом контуре (размерность времени для LBW = 14)

Преобразуя K_d из формулы (13), можно рассчитать максимальную ошибку при ускорении:

$$K_a = \frac{\text{Ускорение } \left(\frac{\text{рад}}{c^2}\right)}{\text{Ошибка (рад)}} = \frac{A_1 \cdot K_d \cdot (f_{\text{clk}})^2}{10} = \frac{A_1 \cdot 2 \cdot \pi \cdot (f_{\text{clk}})^2}{10 \cdot 2^{8+LBW}} \left(\frac{1}{c^2}\right). \qquad (14)$$

Полоса преобразователя выбирается пользователем в зависимости от соотношения сигнал/шум на входе микросхемы, максимальной допустимой ошибки при ускорении и времени переходных процессов. Например, при повышенном шуме и помехах внешней схемы возможно получение необходимой разрядности путем настройки меньшей полосы

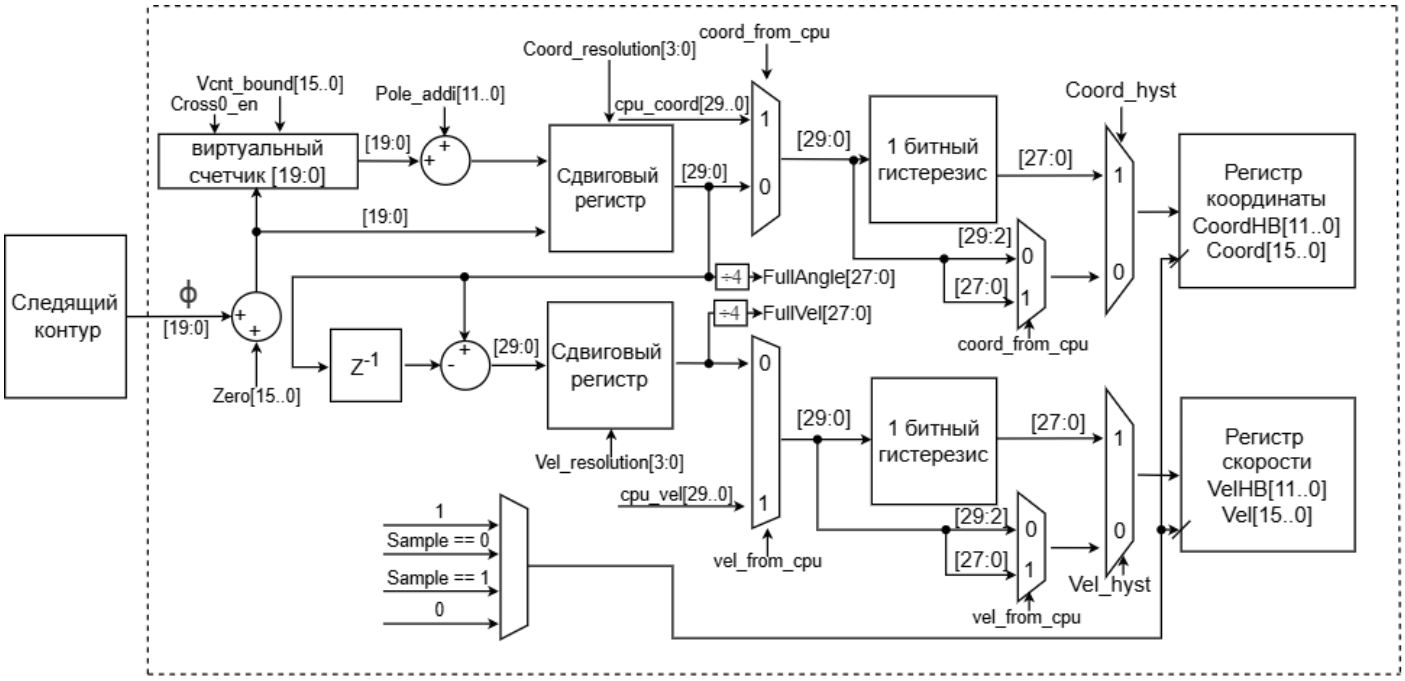
пропускания контура. При этом время переходных процессов и ошибка при ускорении увеличиваются.

Последнее обновление ***3 июл. 2026 г.***

Блок обработки результатов

Блок позволяет представить результат преобразования в формате удобном для пользователя.

При выборе ***Sensor_mode==0*** результат преобразования беззнаковый и соответствует углу от 0 до 2π. При выборе ***Sensor_mode != 0*** результат преобразования знаковый и соответствует координате от -1 до +1.



Структура блока обработки результата

Координата, вычисляемая в следящем контуре, всегда имеет разрядность 20 бит (младшие 2 бита требуются для гистерезиса). Координата может быть скорректирована на постоянную величину, задаваемую в регистре Zero. Коррекция производится путем знакового сложения 20-битной координаты следящего контура со значением, заданным в регистре Zero, помноженной на 16.

В некоторых применениях СКВТ требуется отслеживание переходов через нуль выходного угла (многополюсные СКВТ, недостаточный темп опроса датчика при высокой частоте вращения, малые ресурсы для вычислений и т.д.), для чего в преобразователь

введен 20 битный виртуальный счетчик переходов через нуль выходной координаты следящего контура. Счетчик в темпе работы преобразователя добавляет единицу если значение двух старших разрядов [19:18] координаты изменилось из 3 в 0, и наоборот счетчик декрементируется если значение старших разрядов изменилось из 0 в 3. Для включения счетчика требуется установить бит En_cross0 регистра KonturStngs в единицу (при En_cross0 значение счетчика равно нулю при Sensor_mode==0 или размножено знаком старшего [19] разряда выходной координаты следящего контура при Sensor_mode != 0). Значение 20 битного виртуального счетчика переполняется как двоичное число (C1Vcnt_bound==65535), однако для датчиков с количеством полюсов некратным степени двойки возможно введение другого порога с помощью записи в регистр C1Vcnt_bound/C2Vcnt_bound значения, которое умножается на 16 и добавляется 15. В этом случае координата может быть приведена к двоичной простым умножением на коэффициент.

20 разрядов координаты следящего контура дополняются слева 20 разрядами виртуального счетчика и получившееся значение подается на регистр сдвига "вправо". Величина сдвига задается битами Coord_resolution в регистре ResCntrl (по умолчанию равно 2, что дает 16 битный код угла следящего контура в регистрах C1Coord/C2Coord). 28 битов [29:2] с регистра сдвига могут быть считаны по SPI по адресам C1CoordHB, причем старшая часть (C1CoordHB) всегда синхронизирована с младшей на уровне интерфейса (значения тактируемого регистра с нечетным адресом фиксируется регистра с адресом на единицу меньше (четного)).

При необходимости может быть включен гистерезис координаты (с помощью установки бита Coord_hist), который для своей работы использует младшие два бита с выхода регистра сдвига. При включенной схеме гистерезиса границы переключения значения выхода координаты сдвигаются на ± 0,75 от предыдущего значения. Если отклонение нового значения координаты составило менее ± 0,75 от значения, вычисленного на предыдущем шаге, регистр Coord не меняется. Таким образом, при неподвижном датчике, и, если позволяет соотношение сигнал шум на входе преобразователя, значение в регистре Coord будет постоянным. Даже, если координата датчика находится на границе между соседними значениями, гистограмма кодов будет содержать только одно значение. При использовании постобработки измеренной координаты (например, фильтрации) следует отключить схему гистерезиса, т.к. она является нелинейным элементом и уменьшает точность преобразования. При отключенной схеме гистерезиса, если координата датчика находится посредине между двумя значениями, гистограмма кодов будет содержать примерно одинаковое количество этих значений.

Скорость вычисляется в следящем контуре как приращение координаты на каждом такте преобразования f_{clk}.

В зависимости от настройки битов Sample_src (регистр Mode_config) возможно запоминание вычисленных преобразователем значений координаты, скорости.

Таблица 10 – Настройка режима запоминания

Значение битов Sample_src[1:0]	**Режим запоминания значений в регистрах Coord, Vel, Stat**
00	Сохраняются предыдущие значения
01	Вход Sample == 1 – защелкиваются новые значения Вход Sample == 0 – сохраняются предыдущие значения
10	Вход Sample == 1 – сохраняются предыдущие значения Вход Sample == 0 – защелкиваются новые значения
11 (по умолчанию)	Защелкивание в регистры происходит во время транзакции чтения по интерфейсу SPI

Биты Sample_src[1:0] и вход **Sample** не оказывают влияния на блок эмуляции квадратурного энкодера, работающий на тактовой частоте микросхемы.

*Последнее обновление **3 июл. 2026 г.***

Эмуляция квадратурного энкодера

Эмуляция квадратурного энкодера

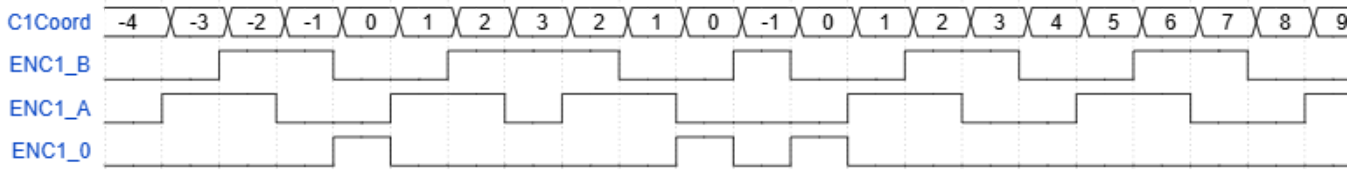
Микросхема содержит выходы эмуляции квадратурного энкодера для замены датчиков типа энкодер на СКВТ, а также для подключения микросхемы к устройствам, принимающим на вход сигналы типа энкодер.

Эмуляция квадратурного энкодера использует те же разряды координаты, что попадают в регистр C1Coord/C2Coord .

Младшие N разрядов координаты и скорости могут быть замаскированы (занулены) с помощью настройки N = Vel_resolution[3:0] регистров C1ResCntrl/C2ResCntrl (предполагается при использовании режима энкодера значение скорости рассчитанное контуром не будет использоваться).

Сигнал **`ENC_0`** устанавливается в состояние логической единицы, когда вычисленная координата равна 0.

После сброса микросхемы **`ENC_A`**, **`ENC_B`** устанавливаются в состояние логического нуля, **`ENC_0`** устанавливается в состояние логической единицы.



Временная диаграмма сигналов эмуляции квадратурного энкодера

Максимальная скорость переключения сигналов **`ENC_A`**, **`ENC_B`**, **`ENC_0`** задается битами **`Enc_presc[2:0]`** в регистрах C1ResCntrl/C2ResCntrl путём деления тактовой частоты генератора микросхемы **`FINT`** (см. таблицу ниже). **`FINT`** определяется настройкой PLL.

Таблица 11 – Временные характеристики сигналов эмуляции квадратурного энкодера

`Enc_presc[2:0]`	**Частота переключения сигналов `ENC_A`, `ENC_B`, `ENC_0`**
`000`	`FINT/2`
`001`	`FINT/3`
`010`	`FINT/4`
`011`	`FINT/5`
`100`	`FINT/8`
`101`	`FINT/16`
`110`	`FINT/32`
`111`	`FINT/64`

*Последнее обновление **3 июл. 2026 г.***

Блок определения ошибок подключения

Микросхема производит определение корректности подключения датчика и детектирование ошибок преобразования. Причина появления ошибки записывается в регистры C1Stat/C2Stat. При наличии ошибки в регистрах C1Stat/C2Stat значения скорости и координаты могут принимать некорректные значения.

Регистр состояния и маски

Все флаги ошибок и состояния преобразователя собраны в регистре C1Stat (адрес 30, преобразователь 1) и C2Stat (адрес 62, преобразователь 2). Распределение бит:

Бит	Флаг	Назначение
15	NLock	Контур в неустановившемся режиме (ошибка слежения превышает порог)
14..13	quadrant	Квадрант результата (не влияет на Ready)
12	—	зарезервирован (всегда 0)
11	Kontur_NotENA	Контур ожидает условий запуска
10	RCV_NotRDY	Опорная частота не восстановлена
9	MISS_EXREF	Отсутствие опорного сигнала EX_REF
8	EX_PH_OUTRANGE	Большой сдвиг фазы опоры (≥ ±40°)
7	C_LOOP_OVF	Переполнение интегратора следящего контура

Бит	Флаг	Назначение
6	UIN_HIGH	Амплитуда сигналов слишком велика
5	UIN_LOW	Амплитуда сигналов слишком мала
4	CORR_OVF	Переполнение после коррекции усиления
3	ADC_OVF	Переполнение из-за большой постоянной составляющей
2	CLIP_COS	Переполнение АЦП по каналу cos
1	CLIP_SIN	Переполнение АЦП по каналу sin
0	HW_NotRDY	Коррекция АЦП не выполнена

> **ЗНАЧЕНИЕ ПО СБРОСУ**
>
> По сбросу регистр `Stat = 0x0801` — установлены биты **0 (HW_NotRDY)** и **11 (Kontur_NotENA)**; остальные сброшены.

Регистр C1Mask (адрес 12) / C2Mask (адрес 44) управляет вычислением флагов. Биты регистра Mask соответствуют битам Stat «один к одному»:

- **Mask.бит = 0** — вычисление соответствующего бита Stat **блокируется** и он принимается равным 0 (флаг выключен);
- **Mask.бит = 1** — бит Stat **отражает действительное состояние** флага.

Флаги в Stat **защёлкиваются** («липкие»): установившись в 1, бит удерживается до прохождения цикла обновления (`REFRESH_ERR`), в течение которого новых срабатываний не было. После этого бит сбрасывается. Период обновления зависит от источника флага: для быстрых флагов (CLIP, CORR_OVF, UIN_*, EX_PH_OUTRANGE) он составляет ≈ 2,5 мс, для ADC_OVF (связанного с длинным циклом CIC-усреднения) — ≈ 18 с. Сигналы **Ready1**/**Ready2** формируются в 1, когда все немаскированные биты Stat из набора равны 0. Биты 14..13 (quadrant), 12 и 10 (MISS_EXREF) **не влияют** на Ready.

Условия срабатывания флагов

Ниже для каждого флага приведено точное условие срабатывания, извлечённое из кода RTL. Значения сигналов — в единицах АЦП: откорректированный 12-битный код (знак в бите [11], диапазон −2048…+2047 LSB), что соответствует нормированной амплитуде ±1.

CLIP_SIN, CLIP_COS — переполнение АЦП

Вычисляются в `ad3s_handler_calcclip` на синхронизированном 12-битном коде АЦП. Флаг срабатывает при точном равенстве кода одному из двух крайних (насыщающих) значений:

$$\mathrm{CLIP} = 1 \;\Leftrightarrow\; (\mathrm{ADC} = 0\mathrm{x}7\mathrm{FF} = +2047) \;\text{ или }\; (\mathrm{ADC} = 0\mathrm{x}800 = -2048). \quad (14\mathrm{a})$$

Срабатывание — на двух конкретных кодах (не на диапазон «около шкалы»): даже 0x7FE/+2046 флага не вызывает. Маскируются битами MSK_CLIP_SIN (бит 1) и MSK_CLIP_COS (бит 2). Сбрасываются, если в течение ≈ 2,5 мс не было новых срабатываний.

ADC_OVF — большая постоянная составляющая

Вычисляется в `ad3s_handler_cic` на сигнале **после удаления постоянной составляющей** `res = ADC − offset`, где `offset` — среднее значение, накопленное CIC-фильтром. Флаг срабатывает, когда результат выходит за пределы 12-битного диапазона:

$$\mathrm{ADC_OVF} = 1 \;\Leftrightarrow\; |res_{sin}| \geq 2048 \;\text{ или }\; |res_{cos}| \geq 2048. \quad (14\mathrm{b})$$

Аппаратно это проверяется как несовпадение знаковых бит: `res[12] XOR res[11]`. Срабатывание означает, что постоянная составляющая слишком велика — сигнал невозможно отцентровать внутри диапазона АЦП (либо АЦП находится в насыщении). Маскируется битом MSK_ADC_OVF (бит 3). Сбрасывается, если в течение ≈ 18 с не было выхода за диапазон.

CORR_OVF — переполнение после коррекции усиления

Вычисляется в `ad3s_handler_ampcorrector` при масштабировании сигнала коэффициентами C1KampS/C1KampC. Внутреннее преобразование:

$$out = \frac{Sin1 \cdot Kamp}{1024}, \qquad Kamp \in [0,\ 65535]. \tag{14c}$$

Произведение `Sin1·Kamp` (30 бит) проверяется на переполнение 13-битного выходного окна; флаг срабатывает, когда защитные старшие биты не являются чистым расширением знака, то есть когда скорректированный сигнал выходит за внутреннюю разрядность:

$$\mathrm{CORR_OVF} = 1 \ \Leftrightarrow\ \left|\frac{Sin1 \cdot KampS}{1024}\right| > 4095 \ \text{ или } \ \left|\frac{Cos1 \cdot KampC}{1024}\right| > 4095 \tag{14d}$$

Единичное усиление соответствует `Kamp ≈ 1024` (значение по умолчанию). С учётом этого переполнение возникает примерно при `Kamp > 2048` для сигнала, близкого к полной шкале (запас ≈ ×2 от единичного усиления); при меньшей амплитуде сигнала допустимы большие `Kamp`. Маскируется битом MSK_CORR_OVF (бит 4). Сбрасывается, если в течение ≈ 2,5 мс не было переполнений.

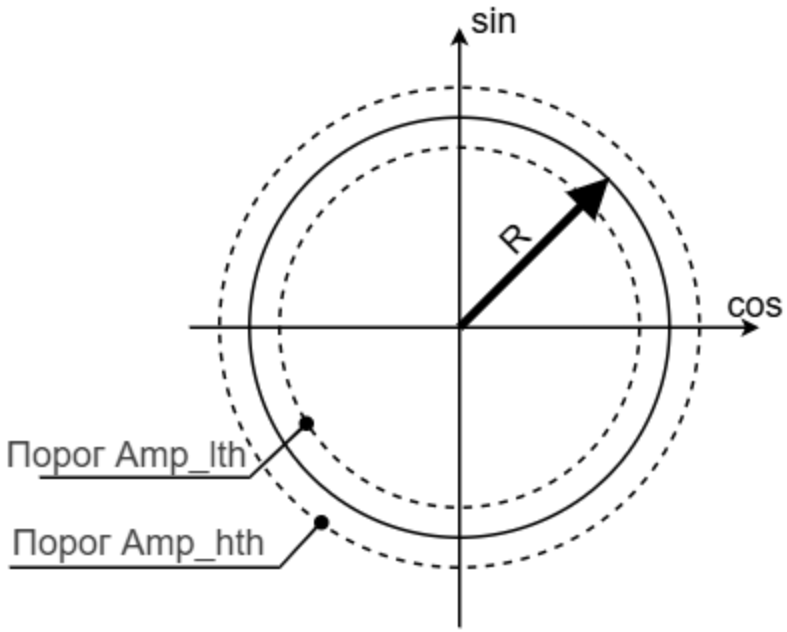
UIN_HIGH, UIN_LOW — пороги амплитуды

Микросхема определяет корректность амплитуды сигнала на входе следящего контура путём сравнения амплитуды **R**, представленной в полярных координатах, с порогами Amp_hth и Amp_lth. Вычисление R (`ad3s_handler_calcuin`) зависит от Sensor_mode:

В режиме Sensor_mode**==0** амплитуда вычисляется по формуле

$$R = \sqrt{Sin1^2 + Cos1^2}, \tag{15}$$

где **Sin1** и **Cos1** – сигналы на входе следящего контура (амплитудой от -1 до +1).

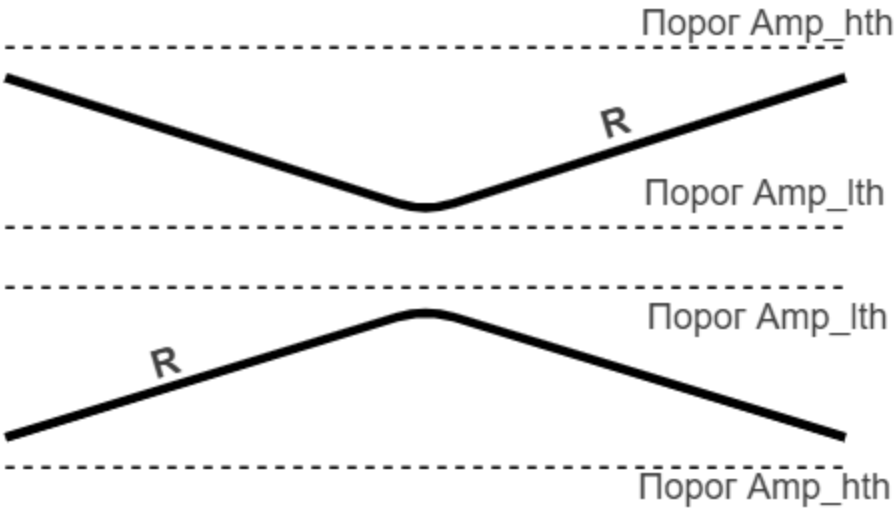


Определение амплитуды и наличия сигналов на входе преобразователя в режиме Sensor_Mode=0

В режимах Sensor_mode**==1** и Sensor_mode**==2** амплитуда вычисляется по формуле

$$R = |Sin1 + Cos1|, \tag{16}$$

где **Sin1** и **Cos1** – сигналы на входе следящего контура (амплитудой от -1 до +1).



Определение амплитуды и наличия сигналов на входе преобразователя в режимах Sensor_Mode=1, Sensor_Mode=2

Внутреннее сравнение выполняется в масштабе ×16. Текущая метрика R доступна для чтения в регистре C1Amp_metric (адрес 23; для канала 2 — C2Amp_metric, адрес 55). Регистр хранит `Amp_metric = R/4` в единицах АЦП (номинальное значение 400); метрика

пиково удерживается по полупериоду опоры (кроме режима DC_carrier, где используется мгновенное значение):

- UIN_HIGH = 1 при `Amp_metric ≥ Amp_hth × 16`;
- UIN_LOW = 1 при `Amp_metric < Amp_lth × 16`.

Поля Amp_hth (биты [15:8]) и Amp_lth (биты [7:0]) регистра C1Amp_th — по 8 бит (0…255). Маскируются битами MSK_UIN_HIGH (бит 6) и MSK_UIN_LOW (бит 5).

EX_PH_OUTRANGE — большой сдвиг фазы опоры

Вычисляется в `ad3s_exrecover` сравнением логических уровней задержанной опоры `ex_shifted` и восстановленной опоры `ex_recovered`. Внутренний счётчик `diff_cnt` подсчитывает в пределах восстановленного периода T число тактов, на котором уровни различаются (XOR). Порог срабатывания:

$$\mathrm{EX_PH_OUTRANGE} = 1 \;\Leftrightarrow\; diff_cnt \geq \frac{7}{32}\,T. \tag{14e}$$

Для двух меандров равной скважности доля тактов с несовпадением уровней равна `2δ/360°`, где δ — фазовый сдвиг. Отсюда порогу `7/32` соответствует

$$\delta \geq 180^\circ \cdot \tfrac{7}{32} \approx 39{,}4^\circ \approx \pm 40^\circ. \tag{14f}$$

Маскируется битом MSK_EX_PH_OUTR (бит 9). Сдвиг задаётся регистром C1ExPhShft (задержка EX_REF в тактах Fclk) — им подстраивают опору до исчезновения флага.

MISS_EXREF — отсутствие опорного сигнала

Вычисляется в `ad3s_exrecover`. Внутренний счётчик `in_cnt` (16 бит) считает такты между спадами EX_REF. Флаг срабатывает, когда счётчик достигает максимума, ни разу не обнаружив спада:

$$\mathrm{MISS_EXREF} = 1 \;\Leftrightarrow\; in_cnt = \mathrm{0xFFFF} = 65535 \;\; (\text{без спада } \mathrm{EX_REF}). \tag{14g}$$

То есть опорный сигнал отсутствует или застопорен дольше 65535 отсчётов (≈ 1 с; норма — переключения EX_REF чаще 10 Гц). Сбрасывается при обнаружении спада EX_REF. Маскируется битом MSK_MISS_EXREF (бит 10).

NLock — ошибка слежения

Вычисляется в `ad3s_kontur` по демодуляционной угловой ошибке — разности между входными сигналами и сигналами внутренней модели. Метрика ошибки доступна в регистре C1Err_metric (адрес 22; канал 2 — C2Err_metric, адрес 54) и равна

$$Err_metric = PHI_{COS} \cdot Sin1 - PHI_{SIN} \cdot Cos1 \;\propto\; \sin(\theta_{\text{вх}} - \theta_{\text{модели}}), \quad (14\text{h})$$

где PHI_{SIN}, PHI_{COS} — сигналы внутренней модели (выходы контура). Флаг сравнивает модуль метрики с 16-битным порогом Lock_th (регистр C1Lock_th, адрес 10):

$$\text{NLock} = 1 \;\Leftrightarrow\; |Err_metric| > Lock_th. \quad (14\text{i})$$

На старте NLock удерживается в 1 до получения двух достоверных полупериодов опоры (первичный захват контура). Сравнение выполняется по пиковому значению $|Err_metric|$ за полупериод опоры (для модулированных сигналов; в режиме DC_carrier — по мгновенному значению). Lock_th — безразмерная величина в тех же единицах, что и `Err_metric` (значение по умолчанию 300); её подбирают так, чтобы при нормальной работе срабатываний не было. Флаг может использоваться для определения большого отклонения вычисленной координаты от координаты датчика (например, из-за ускорения вала). Маскируется битом MSK_NLock (бит 15).

C_LOOP_OVF — переполнение интегратора контура

Вычисляется в `ad3s_kontur` и сигнализирует о насыщении второго (интегрирующего) звена следящего контура типа 2. Когда накопленная сумма выходит за разрядность, контур насыщается до ±полной шкалы и выставляется C_LOOP_OVF. Флаг актуален для датчиков с конечным диапазоном угла (Sensor_mode ≠0 — потенциометры и т. п.). В режиме Sensor_mode==0 (СКВТ/сельсин) срабатывание на упорах является вариантом нормальной работы и указывает на достижение координатой границы диапазона. Маскируется битом MSK_C_LOOP_OVF (бит 7).

Численные значения и расчёт порогов амплитуды

Текущее значение амплитуды R, с которым сравниваются пороги, доступно для чтения в регистре C1Amp_metric (адрес 23; для канала 2 — C2Amp_metric, адрес 55). Внутреннее

сравнение выполняется в масштабе ×16:

- UIN_HIGH = 1 при `Amp_metric ≥ Amp_hth × 16`;
- UIN_LOW = 1 при `Amp_metric < Amp_lth × 16`.

Расчёт порогов под реальный сигнал:

1. Подайте рабочий сигнал и считайте `C1Amp_metric` (адрес 23) — получите значение `M`.
2. Amp_hth (адрес 8, биты [15:8]) = `ceil(M_верх / 16)`, где `M_верх` — граница «амплитуда слишком велика» (на 10–20 % выше `M`).
3. Amp_lth (биты [7:0]) = `floor(M_ниж / 16)`, где `M_ниж` — граница «амплитуда слишком мала» (на 10–20 % ниже `M`).

Оба поля 8-битные (0…255). Значения по умолчанию (Amp_hth = 187, Amp_lth = 62) соответствуют амплитудам выше полного диапазона АЦП и фактически выключают эти флаги — для реального сигнала пороги следует пересчитать.

> **ПРИМЕР**
>
> При номинальном сигнале `M = C1Amp_metric = 400`:
>
> - верхняя граница (амплитуда +20 %): `M_верх = 480` → `Amp_hth = ceil(480 / 16) = 30`;
> - нижняя граница (амплитуда −20 %): `M_ниж = 320` → `Amp_lth = floor(320 / 16) = 20`.

Диагностика при настройке (на примере канала 1)

a. Запишите C1Mask (адрес 12) = `0xFFFE`. Запишите тип датчика в поле C1KonturStngs.Sensor_mode (адрес 13, биты [11:10]). При подключении сельсина установите AFE_config.Mode = `100`. Включите АЦП и преобразователь: Mode_config.ADC_en (адрес 71, бит 14) = 1, CONV1_en (бит 10) = 1.

b. Запишите C1KonturStngs.LBW (биты [4:0]) = 4 (значение по умолчанию; при необходимости — по таблице полос пропускания). Запишите C1ResCntrl.Vel_resolution

(биты [8:5]) = 7, Coord_resolution (биты [3:0]) = 2 (16-битная координата).

c. Двигая датчик, считывайте регистр C1Stat (адрес 30). В первую очередь убедитесь, что флаги CLIP_COS = 0, CLIP_SIN = 0, ADC_OVF = 0 всегда.

d. Флаги UIN_LOW, UIN_HIGH, CORR_OVF служат для диагностики сигналов после коррекции. При подключении СКВТ/сельсина и корректной настройке входного тракта они должны быть равны 0. При изменении регистров C1KampC, C1KampS, C1ExPhShft, C1KbiasC, C1KbiasS, C1fbias убедитесь, что новые флаги ошибки не появились. Если флаг UIN_HIGH или UIN_LOW ложный — пересчитайте пороги Amp_hth/Amp_lth по методике выше.

e. Вращая/двигая датчик, считывайте значения координаты C1Coord (адрес 16) и скорости C1Vel (адрес 24).

- Если координата изменяется случайным образом по всему диапазону при неподвижном датчике: выключите преобразователь (CONV1_en = 0), затем снова включите (CONV1_en = 1). Если не помогло — повторите настройку сначала.
- Если угол отличается на 180° или ЛРДТ находится на упоре: проверьте сигнал EXI1 — вероятно, он инверсный.
- В режиме сельсина при нелинейном изменении угла (например, поворот на 90° даёт изменение только на 60°): неправильно подключены обмотки сельсина.
- Для сдвига значения координаты относительно получаемой с датчика запишите значение в регистр C1Zero (адрес 11).

*Последнее обновление **3 июл. 2026 г.***

Интерфейс SPI

Описание структуры SPI

Взаимодействие управляющего микроконтроллера с микросхемой 5400TP065A-022 осуществляется через последовательный SPI интерфейс с помощью четырех сигналов: SCLK, SSTR, SDO, SDI.

- SSTR — сигнал разрешения работы контроллера. Активный уровень "0". Переход из 1 в 0 является событием начала транзакции (или цепочки транзакций). При SSTR = 1 производится сброс счетчиков синхроимпульсов SCLK.
- SCLK — сигнал синхронизации. Микросхема защелкивает входные данные (вход SDI) по переднему фронту SCLK и выставляет выходные данные SDO по заднему фронту SCLK. Ведущее устройство должно принимать данные с входа SDO по переднему фронту и передавать SDI по заднему фронту SCLK.
- SDI — порт для приема и выдачи данных. Выдача данных на порт производится только при однократном переводе контроллера SPI в режиме полудуплексного чтения.
- SDO — порт выдачи данных, при неактивном уровне SSTR, SDO всегда переводится в состояние высокого импеданса ("Z")

Контроллер SPI работает только по синхроимпульсам SCLK и не зависит от наличия системной частоты FINT. Взаимодействие с контроллером осуществляется через последовательности кадров. Каждый кадр состоит из **16 полных синхроимпульсов** SCLK (имеющих и передний, и задний фронт).

- Контроллер имеет два 4-битных счетчика:
  - Счетчик синхроимпульсов запроса (инкрементируется передними фронтами SCLK)
  - Счетчик синхроимпульсов ответа (инкрементируется задними фронтами SCLK)
- При SSTR **= 1** счетчики сброшены.
- При наступлении **17-го синхроимпульса** счетчики переполняются, что сигнализирует о начале нового кадра (позволяет использовать DMA).

Существуют три вида кадра

- прием на SDI командного слова и одновременная выдача на SDO данных по предыдущему адресу
- прием на SDI данных для записи (на выводе SDO выдается существующее данное по адресу записи)
- выдачей данных на SDI в полудуплексном режиме (вывод SDO дублирует выдачу SDI)

Формат командного слова: **M_2,M_1,M_0, A10,A9,A8, A7, A6, A5, A4, A3, A2, A1, A0, 0, P**

- **M_2, M_1, M_0** — код операции (см. таблицу ниже)
- **A_{10}...A_0** — адрес
- **P** — бит четности

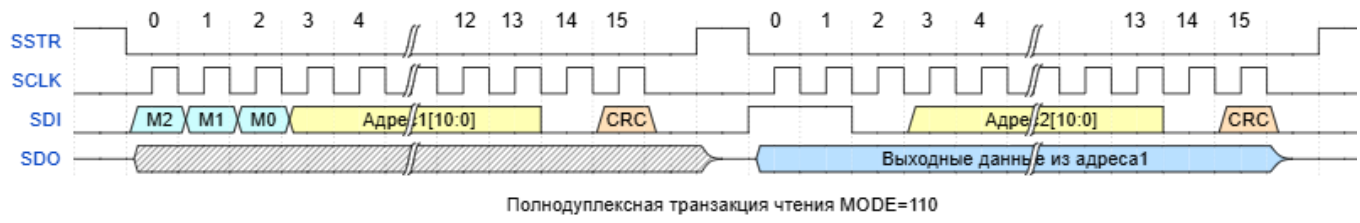
Коды операций

Код операции (M_2,M_1,M_0)	Описание
100	Запись по адресу значением из следующего кадра
110	Вывод значения ячейки по адресу в следующем кадре (полнодуплексная транзакция)
001	Вывод значения ячейки по адресу в следующем кадре (полудуплексная транзакция)
010	Команда блокировки значения в выходном регистре
101	Команда разброкировки записи выходного регистра

Контроллер SPI может находиться в одном из 3 состояний:

1. Прием командного слова
2. Прием слова данных (если предыдущее командное слово интерпретировалось как запись)
3. Выдача слова данных в полудуплексном режиме

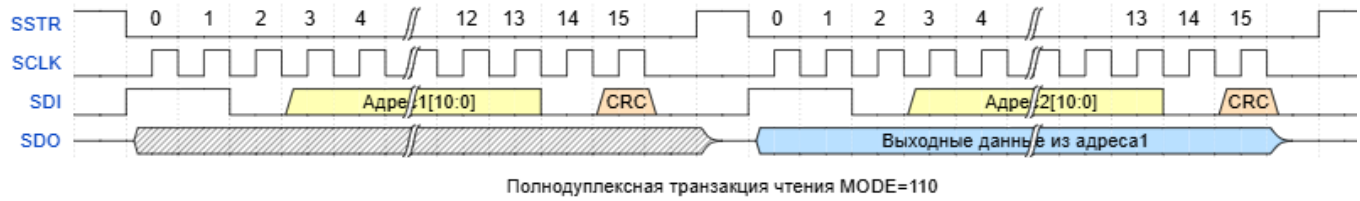
Коды операции "110", "010", "101" оставляют контроллер SPI в 1 состоянии, следующий кадр может содержать командное слово, а на выводе SDO будет передаваться значение адреса зафиксированного в предыдущем кадре.



Полнодуплексная транзакция чтения MODE=110

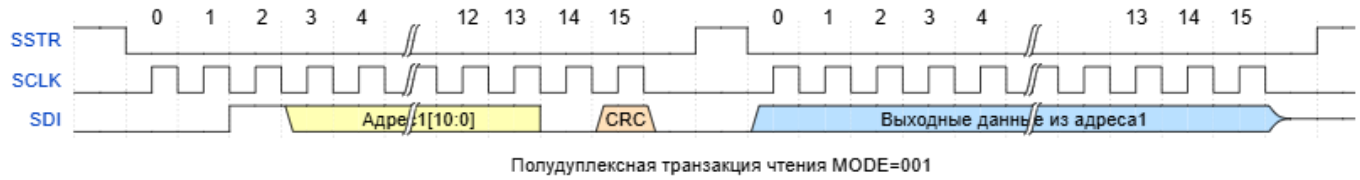
Полнодуплексная транзакция чтения режим 110

Переход в 2,3 состояние из состояния 1 возможен при корректно принятом командном слове с кодом операции "100" или "001". При нахождении контроллера в 2 и 3 состоянии данные на выводе SDI не интерпретируются как командное слово, по окончании кадра контроллер переходит в состояние 1.



Полнодуплексная транзакция чтения MODE=110

Транзакция записи — режим 100



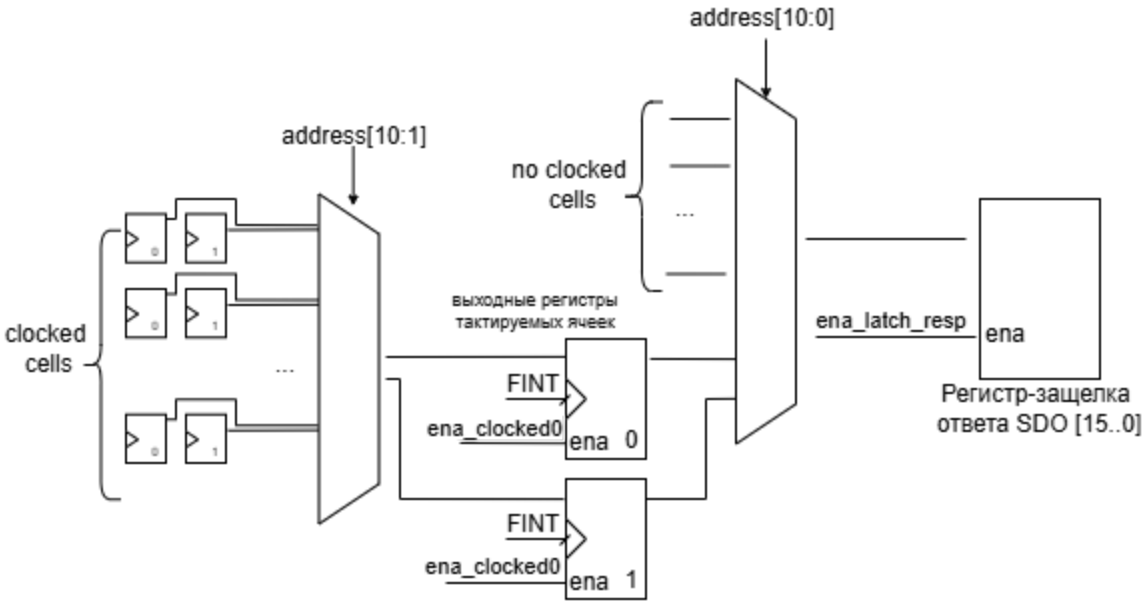
Полудуплексная транзакция чтения MODE=001

Полудуплексная транзакция чтения режим 001

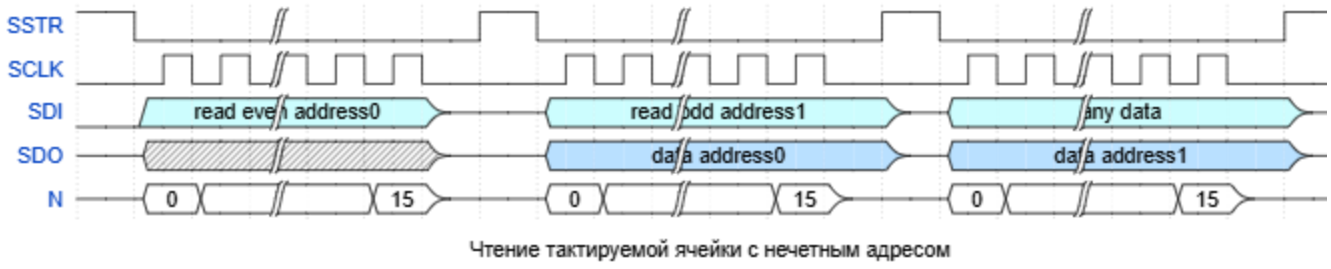
Доступ к ячейкам памяти — параллельный: один мультиплексор подключает нужную ячейку к контроллеру SPI по заданному адресу.

Фиксация адреса, мультиплексирование по адресу регистра для чтения производится при приеме 14-го переднего фронта командного слова, не дожидаясь завершения окончания транзакции и приема бита четности. Однако контроллер перейдет в другое состояние только если поступило 16 передних фронтов SCLK, 14-й бит регистра запроса равен 0 и корректен бит четности (или отключен контроль).

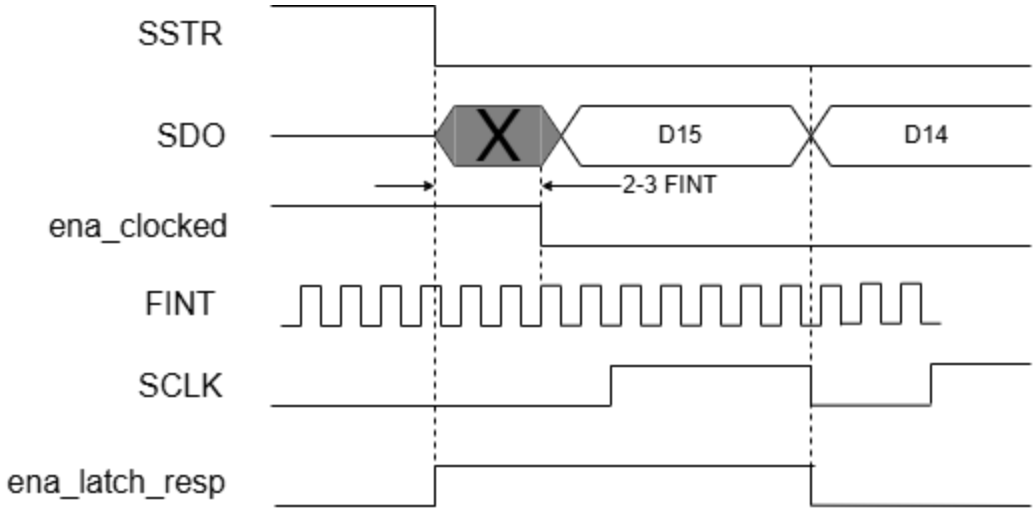
Для тактируемых FINT ячеек выход мультиплексора подключен к промежуточному регистру. Разрешение копирования данных на него снимается "ena_clocked" при заднем фронте SSTR или 16-м заднем фронте SCLK (при непрерывной циклической передаче). Сигнал "ena_clocked" (см. рис. ниже) пересинхронизируется двумя триггерами на FINT, что требует паузы до первого фронта SCLK длиной в 3 периода FINT (параметр данной задержки конфигурируется в большинстве архитектур микроконтроллеров). Нечетные адреса всех тактируемых ячеек защелкиваются в промежуточный регистр только при чтении четного адреса на единицу меньше. При повторном чтении нечетного адреса значение в промежуточном регистре не обновится. Для чтения актуального значения нечетного адреса возможно при последовательной передаче транзакций команд чтения сначала четного адреса, затем нечетного адреса на единицу больше.



Формат командного слова SPI



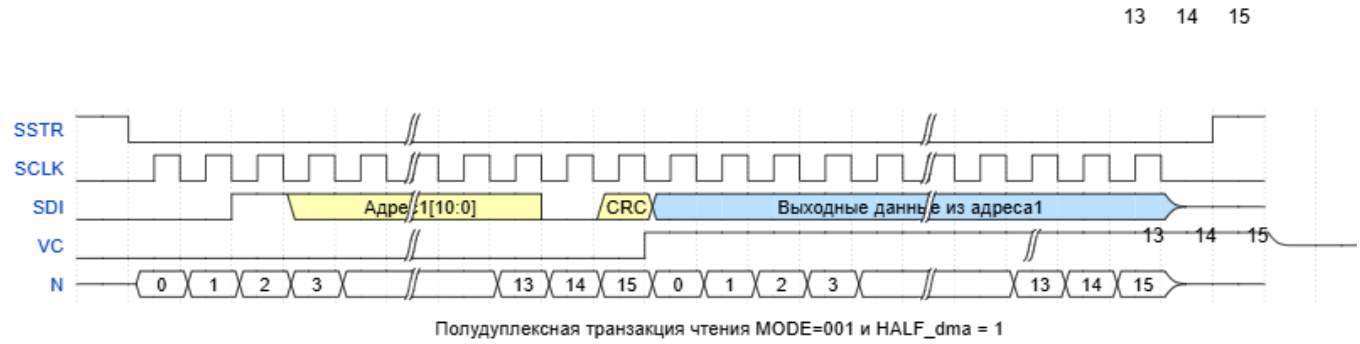
Чётный/нечётный адрес SPI



Временная диаграмма

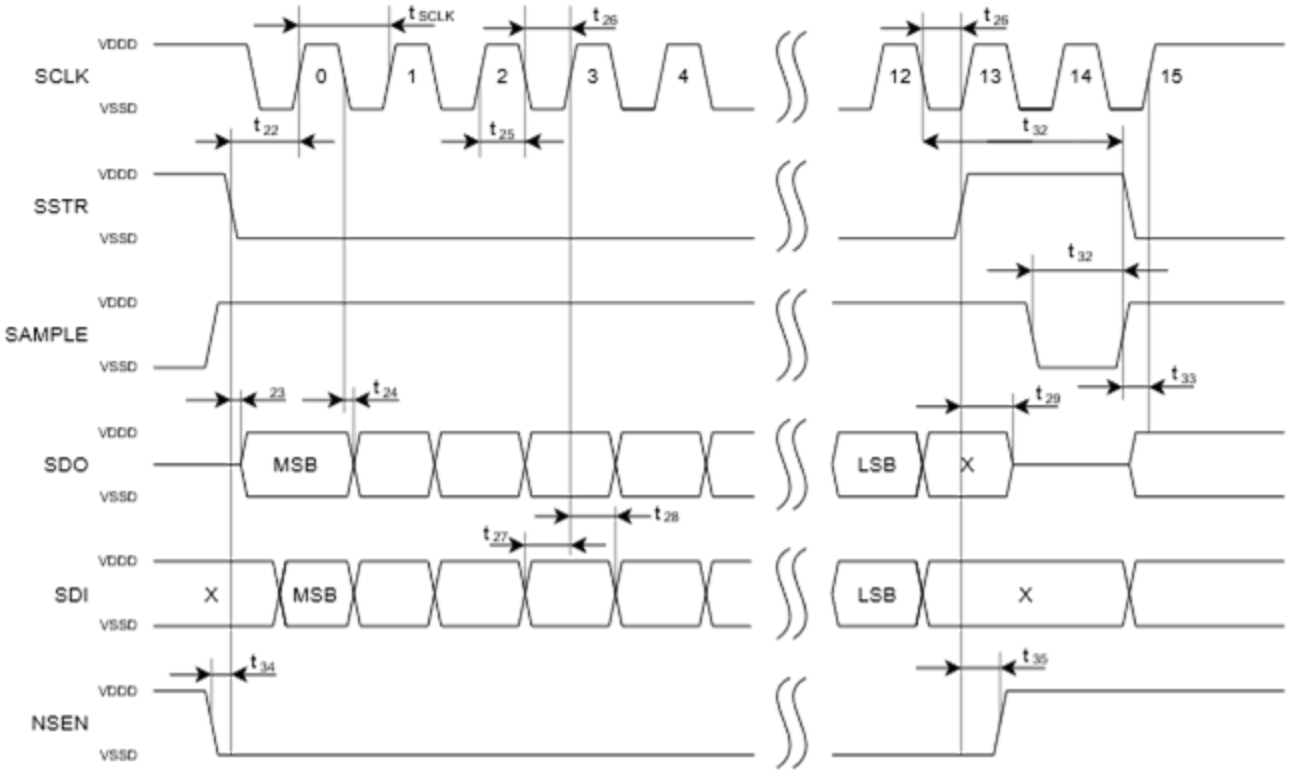
Выходной регистр-защелка не может поменять свое значение начиная с 1 по 15 заднего фронта SCLK.

Контроллер SPI может быть переведен в режим полудуплексной передачи без перевода SSTR в 1 (циклический режим). В этом случае требуется установить бит HALF_dma = 1, что даст указание контроллеру переводить вывод SDI и приемо-передатчик сигналом VC в активное состояние выдачи на спаде 16 SCLK текущего кадра приема командного слова.



Полудуплексная транзакция чтения режим 001 и HALF_dma=1

Вывод **nSEN** маскирует сигнал SSTR. При **nSEN = 1** интерфейс SPI отключается (используется при подключении нескольких микросхем). Если обмен только с одной микросхемой, nSEN должен быть подтянут к 0.



Временная диаграмма

Временные параметры последовательного интерфейса

Параметр, единица измерения	не менее	типовое	не более
Период SCLK (t_SCLK), нс	50		
Задержка от заднего фронта SSTR до переднего фронта SCLK (t_22), нс	6,0		
Задержка от заднего фронта SSTR до занятия линии SDO (t_23), нс			39
Задержка от заднего фронта SCLK до переключения SDO (t_24), нс			41
Длительность лог. «1» на SCLK (t_25), нс	16		

Параметр, единица измерения	не менее	типовое	не более
Длительность лог. «0» на SCLK (t_26), нс	16		
Задержка от выставления SDI до переднего фронта SCLK (t_27), нс	6,0		
Задержка от переднего фронта SCLK до изменения SDI (t_28), нс	6,0		
Задержка от заднего фронта SSTR до освобождения SDO (t_29), нс			29
Задержка от заднего фронта SCLK до переднего фронта SSTR (t_30), нс	6,0		
Задержка от заднего фронта SCLK при SSTR=0 до переднего фронта Sample (t_31), нс	6,0		
Длительность лог. «0» на Sample (t_32), нс	50		
Задержка от переднего фронта Sample до первого переднего фронта SCLK при SSTR=0 (t_33), нс	4,0		
Задержка от заднего фронта nSEN до заднего фронта SSTR (t_34), нс	0		
Задержка от переднего фронта SSTR до переднего фронта nSEN (t_35), нс	0		

Дополнительные режимы SPI

Целевое применение 5400TP065A-022 - индуктивные датчики угла и положения в аппаратуре специального назначения. Для снижения уровня помех и искажений рекомендуется размещение 5400TP065A-022 в непосредственной близости от источника ЭДС. Для обеспечения цифрового обмена рекомендуется использование

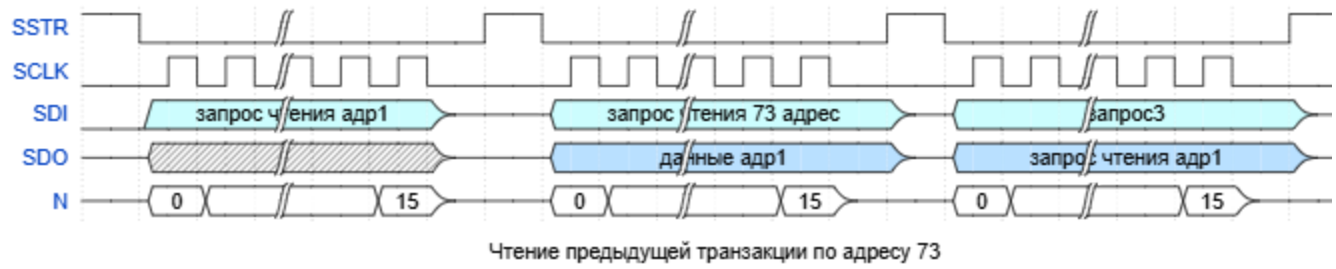
дифференциальных пар - передача сигналов по току (RS-485, M-LVDS, LVDS и др.). Для повышения надежности передачи данных и упрощения использования множества преобразователей в микросхеме 5400TP065A-022 реализовано несколько дополнительных функций.

Фиксация значения выходной ячейки для множественного чтения.

При передаче кода операции "010" сигнал разрешения с регистра (четный и нечетный адрес) тактируемых ячеек маскируется, значение его не обновляется. Это дает возможность прочитать значения несколько раз и сравнить целостность передачи данных. Разблокировка производится кодом операции "101".

Чтение ранее переданной транзакции

Чтение адреса 73 SPI_req возвращает ранее переданный в контроллер SPI кадр. Это позволяет проверить соответствие принятого значения с запрошенным адресом.



Чтение предыдущей транзакции по адресу 73

Чтение предыдущей транзакции по адресу 73

Блокировка записи слов

При записи в регистр WR_Lock значения отличного от нуля, запись в другие регистры блокируется (кроме BUS_addr), только чтение.

Программный адрес

При наличии нескольких микросхем подключенных к одной общей шине для обеспечения индивидуального доступа к каждой микросхеме в их регистры IC_addr могут быть предварительно записаны различные номера в диапазоне [1..255]. Чтобы команда чтения или записи сработала для конкретной микросхемы мастер устройство должно предварительно записать в регистр BUS_addr значение номера необходимой микросхемы, и затем следующие необходимые транзакции записи и чтения. Запись в регистр BUS_addr не блокируется значениями регистров WR_Lock и IC_addr.

Контроллер SPI конкретной микросхемы реагирует на команды только в следующих случаях

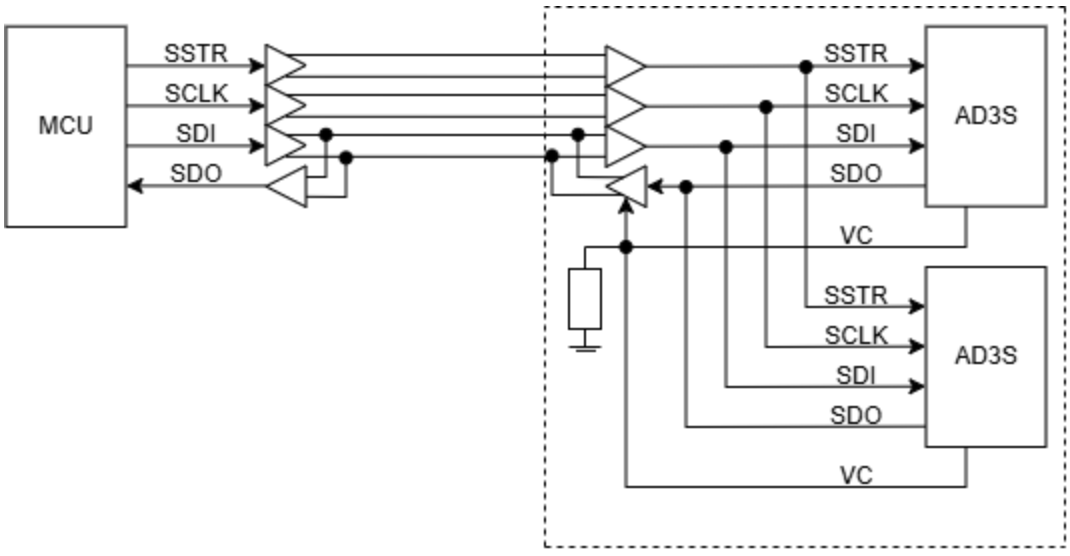
- значение IC_addr равно 0
- IC_addr != 0, BUS_addr == IC_addr
- IC_addr != 0, BUS_addr == 0, флаг BUS0_mode == 1

В остальных случаях контроллер не реагирует (за исключением перезаписи BUS_addr).

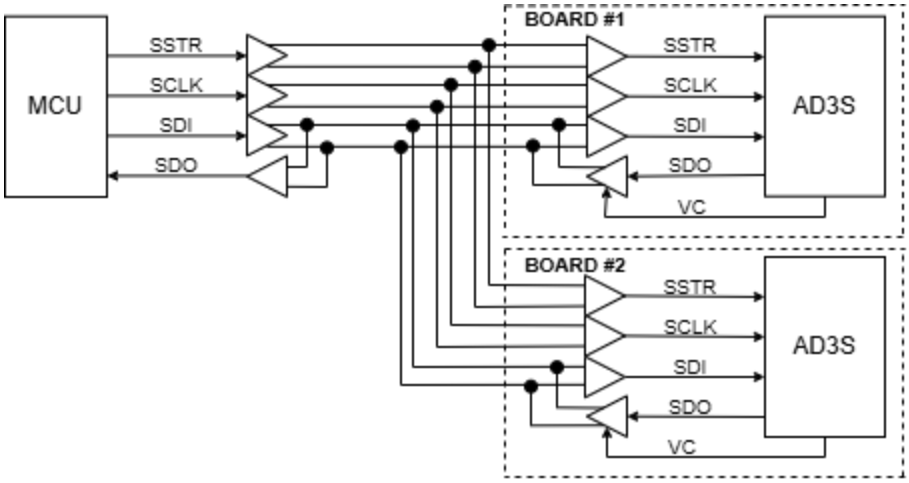
Подразумевается, что конкретное значение IC_addr записывается в ПЗУ микросхемы до установки в общий интерфейс аппаратуры.

Управление внешним приемо-передатчиком с помощью вывода VC

Установка бита VC_mode в 1 регистра AFE_config переводит вывод VC в режим управления внешним приемо-передатчиком. Бит DE_half определяет когда вывод VC переходит в активное состояние: при DE_half = 1 - только для ответа при полудуплексных запросах, при DE_half = 0 при всех запросах когда микросхема должна отвечать (когда SSTR = 0, nSEN = 0 и есть соответствие адреса микросхемы и шины адреса). Бит DE_inv определяет какой уровень VC является активным для передатчика: при DE_inv = 0 активный 1, при DE_inv = 1 активный уровень 0. Установка бита DE_turnZ в 1 переводит вывод VC в состояние высокого импеданса ("Z") между передачами, а сигнал VC должен быть доопределен подтяжкой резистора (требуется при подключении нескольких микросхем к одному передатчику). На рисунках ниже представлены некоторые варианты включения микросхемы.



Подключение двух преобразователей на одной плате с единым интерфейсом в полудуплексном режиме передачи



Подключение двух независимых преобразователей на один интерфейс в полудуплексном режиме передачи

Режим параллельной передачи результата

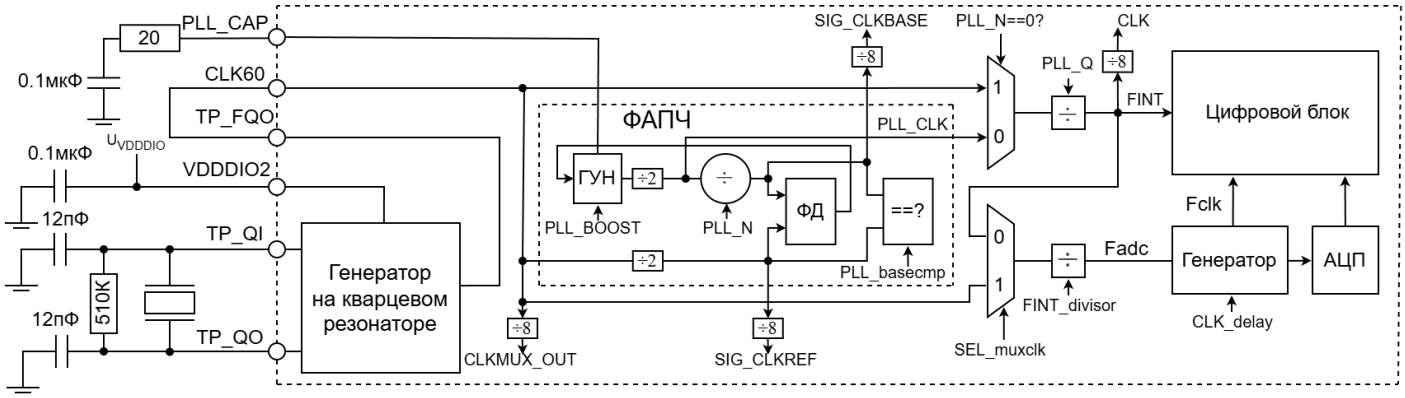
Режим включается при SPI_ext_en **= 1**. Выводы ENC1_A**,** ENC1_B**,** ENC1_0**,** ENC2_A**,** ENC2_B**,** ENC2_0 переключаются параллельно с SDO и передают данные из регистров C1Coord/C2Coord, C1CoordHB/C2CoordHB, C1Vel/C2Vel, C1VelHB/C2VelHB.

Назначение выводов в режиме параллельной передачи результата

SDI	**VC**	ENC1_A	ENC1_B	ENC1_0	SDO	ENC2_A	ENC2_B	EI
0	0	C1Coord	C1Vel	C1CoordHB	C1VelHB	C2Coord	C2Vel	C2C
0	1	C1Vel	C1Coord	C1VelHB	C1CoordHB	C2Vel	C2Coord	C2V
1	0	C2Coord	C2Vel	C2CoordHB	C2VelHB	C1Coord	C1Vel	C1C
1	1	C2Vel	C2Coord	C2VelHB	C2CoordHB	C1Vel	C1Coord	C1V

*Последнее обновление **3 июл. 2026 г.***

Тактирование



Организация тактирования

Цепочка тактирования

Тактовый сигнал проходит через несколько ступеней обработки:

1. **Источник** — кварцевый резонатор (2–20 МГц) или внешний генератор (подается на CLK60).
2. **Умножитель PLL** — умножает входную частоту на коэффициент PLL_N (если включён). Результат умножения не должен превышать 60 МГц.
3. **Делитель PLL** — делит умноженную частоту на коэффициент PLL_Q + 1.
4. **FINT** — итоговая системная частота: FINT = f_вх × PLL_N / (PLL_Q + 1). При PLL_N = 0 умножитель выключен, FINT = f_вх.
5. **Делитель АЦП** — FINT_divisor: Fclk_adc = FINT / (FINT_divisor + 1).

ОГРАНИЧЕНИЯ В ТЕКУЩЕЙ ВЕРСИИ МИКРОСХЕМЫ

- **Fclk_adc ≤ 2,5 МГц** — частота тактирования АЦП не должна превышать 2,5 МГц (номинальное значение — до 20 МГц).
- **FINT ≥ 2 × Fclk_adc** — для корректной обработки сэмплов АЦП преобразователями HAND1 и HAND2.
- **Частота выборок ≤ 150 кВыб/с** — период обновления результата Tclk ≥ 6,7 мкс (Tclk = 16 / Fclk_adc при DELAY_cycles = 0).

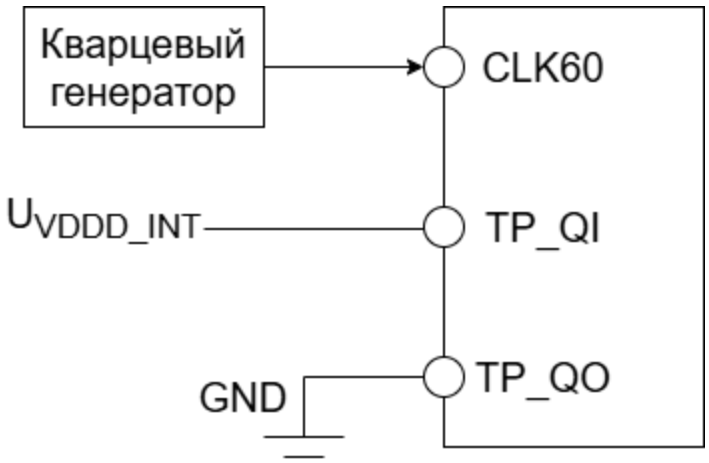
- **Результат умножения PLL ≤ 60 МГц** — частота с выхода умножителя не должна превышать 60 МГц.

Выбор источника тактирования

Источник тактирования определяется схемотехнически одним из двух вариантов:

1. **Встроенный генератор на кварцевом резонаторе** — кварцевый резонатор подключается к выводам TP_QI(28) и TP_QO(27). Выход генератора TP_FQO(26) подключается на вход CLK60(39). Генератор поддерживает кварцевые резонаторы с частотой **от 2 МГц до 20 МГц**. Время выхода на режим — от нескольких миллисекунд. Для работы генератора на вывод VDDIO2(25) необходимо подать питание, равное VDDIO(45). Резистор 510 кОм, подключённый параллельно резонатору, обеспечивает стабильный запуск генерации.

2. **Внешний активный генератор** — сигнал подаётся непосредственно на вход CLK60(39). При этом вывод TP_QI(28) должен быть подключён к VDDD_INT(31), вывод TP_QO(27) — к GND.



Подключение внешнего активного генератора

РЕКОМЕНДУЕМЫЕ ЧАСТОТЫ НА ВХОДЕ CLK60

Для последующего умножения с помощью PLL частота на CLK60 должна быть **не более 30 МГц** (при внешнем генераторе) или **не более 20 МГц** (при кварцевом резонаторе). Это позволяет умножить частоту на 2–3 раза до целевых 50–60 МГц FINT, не превышая предел 60 МГц на выходе PLL.

Настройка ФАПЧ (PLL)

Блок ФАПЧ состоит из двух независимых ступеней: **умножителя** (поле PLL_N) и **делителя** (поле PLL_Q).

Умножитель частоты — может быть задействован, если входная частота не более 20 МГц. Задаётся полем PLL_N в диапазоне `2...74`. Частота на выходе умножителя CLK_PLL = f_вх × PLL_N **не должна превышать 60 МГц**. Если PLL_N = 0, умножитель выключен — тактирование производится напрямую от CLK60.

Делитель частоты — поле PLL_Q в диапазоне `0...15`. Итоговая системная частота: FINT = CLK_PLL / (PLL_Q + 1). Если деление не требуется, задать PLL_Q = 0.

Контроль готовности:

- Для ускорения выхода PLL на режим — бит PLL_BOOST[7] = 1 (предзаряд петлевого фильтра); используется по необходимости.
- Для контроля захвата частоты — установите PLL_basecmp[15:14] ≠ 00. Готовность — по биту CLKCOMP_rdy.
- Для задержки запуска цифровой логики — поле CLK_delay[13:12]: `00` = без задержки (по умолчанию, запуск сразу после готовности тактовой), `01` = 2048 тактов, `10` = 4096 тактов, `11` = 8192 тактов FINT. Задержка удерживает сброс цифровой части до стабилизации тактовой частоты.

Петлевой фильтр — состоит из внутреннего резистора 10 кОм и внешнего конденсатора на выводе PLL_CAP(40). Номинал конденсатора 0,1 мкФ является типовым. Увеличение ёмкости уменьшает пульсации, но увеличивает время установления, и наоборот. Уровень пульсаций не должен превышать 5% от периода тактовой частоты (ограничение джиттера).

Для отладки ФАПЧ на выводе DB(41) можно наблюдать внутренние сигналы через поле DB_mode регистра Mode_config. Все сигналы выводятся поделёнными на 8:

DB_mode	Сигнал на DB
000	CLKMUX_OUT — мультиплексированный тактовый сигнал (значение по умолчанию)

DB_mode	Сигнал на DB
001	DB_SIG_PLLBASE — частота с делителя PLL_BASE
010	DB_SIG_CLKREF — опорная частота CLKREF
011	**CLK — результирующая частота FINT**

Для проверки итоговой частоты тактирования FINT установите DB_mode = `011` и измерьте частоту на выводе DB — она должна быть равна FINT / 8.

Настройка частоты АЦП

Fclk_adc — частота тактирования АЦП. В текущей версии микросхемы она **не должна превышать 2,5 МГц**; превышение приводит к некорректной работе преобразователя.

Частоты, участвующие в тактировании АЦП

Обозначение	Что это	Как получается
f_CLK60	частота на входе CLK60	задаётся внешним генератором или кварцевым резонатором
FINT	системная тактовая частота цифрового блока (CPU, HAND1/HAND2 и др.)	при включённом умножителе: `f_CLK60 × PLL_N / (PLL_Q + 1)`; при PLL_N = 0: `f_CLK60 / (PLL_Q + 1)`
f_adc_src	источник делителя АЦП	FINT либо f_CLK60 — выбирается битом SEL_muxclk (см. ниже)
Fclk_adc	частота тактирования АЦП	`f_adc_src / (FINT_divisor + 1)`
Fclk	частота выдачи выборок (преобразования)	`Fclk_adc / 16`

Формирование Fclk_adc

Делитель АЦП — поле FINT_divisor регистра ADC_config (4 бита, диапазон `0...15`, значение по умолчанию = 3). Он делит не FINT напрямую, а **частоту источника `f_adc_src`, выбранного битом** SEL_muxclk:

SEL_muxclk	Источник делителя `f_adc_src`	Что тактирует
`0`	**FINT** — частота цифрового блока	та же FINT, что и цифра
`1`	**f_CLK60** — вход CLK60 напрямую, минуя PLL	исходная опорная частота

```
Fclk_adc = f_adc_src / (FINT_divisor + 1)
```

Условие Fclk_adc ≤ 2,5 МГц

> **FCLK_ADC ≤ 2,5 МГц — ОБЯЗАТЕЛЬНОЕ УСЛОВИЕ**
>
> Поскольку `FINT_divisor` 4-битное (максимальный коэффициент деления 16), для получения 2,5 МГц **источник делителя должен быть не выше 40 МГц**: `f_adc_src ≤ 40 МГц`.
>
> Дополнительно: частота тактирования цифрового блока должна быть **минимум в 2 раза больше** Fclk_adc — это требуется для обработки сэмплов АЦП преобразователями HAND1 и HAND2.

Выполнить `f_adc_src ≤ 40 МГц` можно двумя путями. Выбор зависит от того, нужна ли цифровому блоку высокая тактовая частота.

Путь A — `SEL_muxclk = 0`, АЦП от FINT. FINT делится полем PLL_Q, поэтому её всегда можно опустить до ≤ 40 МГц. *Пример:* 60 МГц на CLK60 + PLL_Q = 1 → FINT = 30 МГц → `FINT_divisor = 11` → Fclk_adc = 2,5 МГц. Оборотная сторона — **цифровой блок работает на той же поделенной FINT** (30 МГц), а не на 60 МГц.

Путь B — `SEL_muxclk = 1`, АЦП от CLK60. Делитель работает от входа CLK60 напрямую, а цифровой блок умножается полем PLL_N до 50–60 МГц. Для этого на CLK60 подаётся

низкая опорная частота (10–30 МГц), и `FINT_divisor` подбирается под неё. Это единственный способ одновременно держать цифру на 50–60 МГц и АЦП на 2,5 МГц.

> ⓘ **КАКОЙ ПУТЬ ВЫБРАТЬ**
>
> - **Путь B** (низкая опорная на CLK60 + умножение PLL_N + `SEL_muxclk = 1`) — рекомендуемый, если нужна производительная математика на CPU.
> - **Путь A** проще, но ограничивает тактовую цифры значением FINT ≤ 40 МГц.
> - Подача большой частоты напрямую на CLK60 **не является препятствием**: она просто делится полем PLL_Q, ценой снижения тактовой частоты цифрового блока (Путь A).

Примеры конфигурации

Сценарий	f_CLK60	Путь	PLL_N	PLL_Q	FINT (цифра)	SEL_muxclk	FINT_divisc
По умолчанию (кварц 10 МГц, PLL выкл)	10 МГц	A	0	0	10 МГц	0	3
Внешний 10 МГц, без PLL	10 МГц	A	0	0	10 МГц	0	3
Высокая цифра: кварц 10 МГц, PLL на 60 МГц	10 МГц	B	6	0	60 МГц	1	3
Высокая цифра: внешний	20 МГц	B	3	0	60 МГц	1	7

Сценарий	f_CLK60	Путь	PLL_N	PLL_Q	FINT (цифра)	SEL_muxclk	FINT_divisc
20 МГц, PLL на 60 МГц							
Высокая частота на CLK60, с делением PLL_Q	60 МГц	A	0	1	30 МГц	0	11

Подбор делителя: `FINT_divisor = f_adc_src / 2,5 МГц − 1` (округлить вниз; результат должен быть в диапазоне `0...15`).

Период преобразования — для одного преобразования АЦП требуется 16 тактов Fclk_adc. Период работы преобразователя:

Tclk = (16 + DELAY_cycles) × Tclk_adc

При необходимости замедлить АЦП (например, если цикл следящего контура короче времени преобразования) добавляются пустые такты полем DELAY_cycles (8 бит, `0...255`).

Выключение тактирования

Для снижения энергопотребления, когда микросхема запитана, но не используется, имеется возможность выключить тактирование с цифрового блока с помощью бита OSC_DIS или подачей активного уровня `1` на вывод STNDBY. Состояние включено/выключено тактирование может быть определено по биту CLK_rdy.

*Последнее обновление **3 июл. 2026 г.***

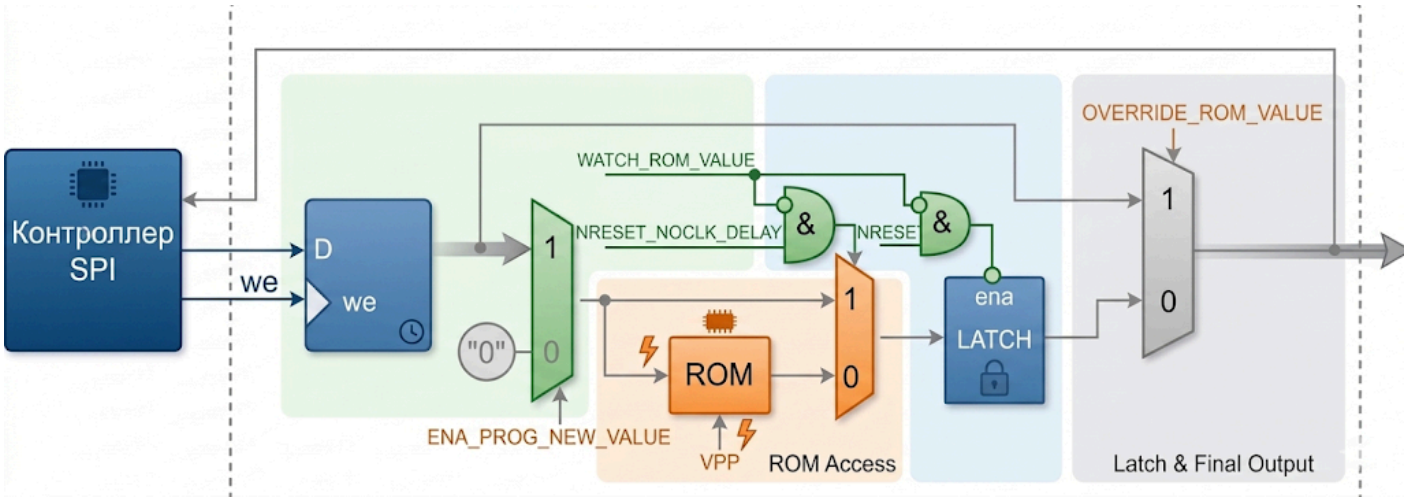
Постоянное запоминающее устройство

Структура

Микросхема содержит два сегмента ПЗУ для хранения пользовательских настроек, необходимых для автоматической инициализации регистров после подачи напряжения питания или сигнала сброса.

ПЗУ с параллельным доступом

ПЗУ с параллельным доступом активируется непосредственно при включении питания и содержит 2 ячейки, которые неразрывно связаны с управляющими регистрами PLL_config и INIT_conf. На рисунке ниже представлена структура ячейки ПЗУ.



Структура ячейки ПЗУ с параллельным доступом

Доступ к настройкам PLL_config и INIT_conf осуществляется через управляющий регистр UOTP_ctrl, который содержит три бита: WATCH_ROM_UVAL, PROG_NEW_UVAL и OVERRIDE_UVAL. По умолчанию значения регистров PLL_config и INIT_conf равны нулю.

Для временного переопределения этих значений (до снятия питания) необходимо записать требуемые данные в соответствующие регистры (PLL_config и INIT_conf), а затем установить бит OVERRIDE_UVAL в логическую «1». При этом заданные значения

сохраняются даже при выполнении программного сброса с помощью бита MANUAL_NRST_PLL (путем записи нуля, а затем единицы).

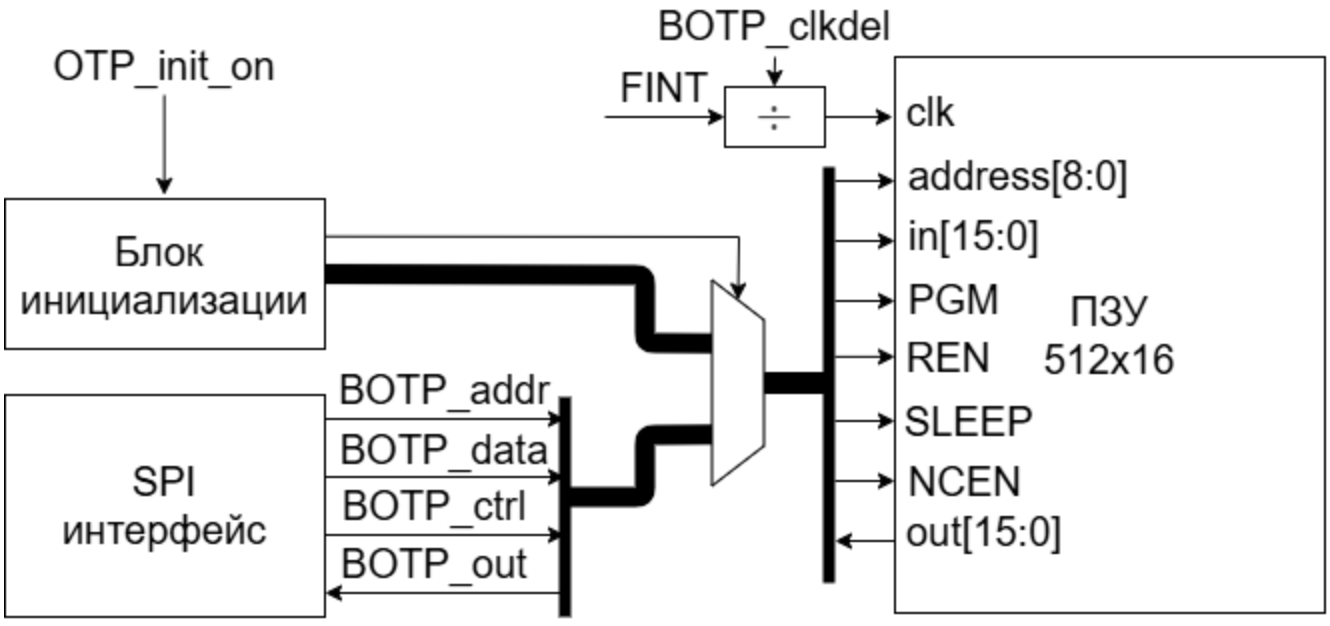
Для программирования этих значений в энергонезависимую память ПЗУ необходимо выполнить следующую последовательность действий:

1. Записать в регистры PLL_config и INIT_conf нужные параметры.
2. Убедиться в корректности записи путем обратного чтения (при `OVERRIDE_UVAL = 1`).
3. Установить бит PROG_NEW_UVAL и подать напряжение программирования 9,8 В ± 0,3 В на вывод `VPP`.
4. По истечении заданного времени снять напряжение `VPP` и сбросить бит PROG_NEW_UVAL.
5. Установить бит WATCH_ROM_UVAL в логическую «1».
6. Считать значения из PLL_config и INIT_conf, чтобы подтвердить успешность записи.

Если считанные значения не совпадают с заданными, операцию программирования следует повторить.

Обратите внимание: в состав регистра INIT_conf входит бит OTP_init_on — разрешение автозагрузки конфигурации из BOTP при включении питания. Прожигать его в значение «1» допускается **только после полного прожига и успешной проверки BOTP** — см. Порядок программирования ПЗУ.

ПЗУ с последовательным доступом



Второй сегмент ПЗУ предназначен для инициализации регистров конфигурации и ячеек памяти микровычислителей при включении питания. ПЗУ содержит 512 16-битных слов. Если выставлен бит OTP_init_on, после снятия сигнала сброс (при наличии тактирования), контроллер инициализации последовательно считывает данные ячеек ПЗУ и записывает в соответствующие регистры. Порядок записи жестко определен и неизменен.(см таблицу 12).

Таблица 12. Соответствие адресов ячеек памяти ПЗУ и конфигурационных регистров

+	2 x 0	2 x 1	2 x 2	2 x 3	2 x 4	2 x 5
2 x 0	0 cpu1	1 cpu1	2 cpu1	3 cpu1	4 cpu1	5 cpu1
2 x 8	8 cpu1	9 cpu1	10 cpu1	11 cpu1	12 cpu1	13 cpu1
2 x 16	16 cpu1	17 cpu1	18 cpu1	19 cpu1	20 cpu1	21 cpu1
2 x 24	24 cpu1	25 cpu1	26 cpu1	27 cpu1	28 cpu1	29 cpu1
...	...	...	...	...	...	...
2 x 232	C1KampS	C1KampC	C1KbiasS	C1KbiasC	C1fbias	C1ExPhShft
2 x 240	C1Amp_th	C1InputStngs	C1Lock_th	C1Zero	C1Mask	C1KonturStngs
2 x 248	233 cpu1	234 cpu1	235 cpu1	236 cpu1	233 cpu2	234 cpu2

+	2 x 0	2 x 1	2 x 2	2 x 3	2 x 4	2 x 5
2 x 256	0 cpu2	1 cpu2	2 cpu2	3 cpu2	4 cpu2	5 cpu2
…	…	…	…	…	…	…
2 x 488	C2KampS	C2KampC	C2KbiasS	C2KbiasC	C2fbias	C2ExPhShft
2 x 496	C2Amp_th	C2InputStngs	C2Lock_th	C2Zero	C2Mask	C2KonturStngs
2 x 504	IC_addr	ADC_config	Mask_Stat	Flags_delay	MR_lock	CMP_lth

Запись и чтение ПЗУ производится с помощью управляющих регистров BOTP_addr, BOTP_data, BOTP_out, BOTP_ctrl.

Для корректной работы ПЗУ (записи и чтения, в т.ч. при инициализации после включения питания) требуется настройка тактирования. Исходя из значения частоты `FINT` требуется выбрать значение BOTP_clkdel в соответствии с таблицей делителей в описании регистров так, чтобы частота тактирования ПЗУ Fclk_rom = FINT / N не превышала **1 МГц**. Например, при FINT = 10 МГц значение BOTP_clkdel должно быть не меньше 3: при BOTP_clkdel = 2 делитель N = 8 даёт Fclk_rom = 1,25 МГц (превышает 1 МГц), а при BOTP_clkdel = 3 делитель N = 12 даёт Fclk_rom ≈ 833 кГц. При FINT = 60 МГц минимально допустимое значение — BOTP_clkdel = 11 (N = 64, Fclk_rom ≈ 938 кГц): при BOTP_clkdel = 10 делитель N = 56 даёт Fclk_rom ≈ 1,07 МГц, что превышает ограничение.

Для записи данных в ячейку ПЗУ по указанному адресу требуется записать целевой адрес в регистр BOTP_addr, а записываемое значение — в регистр BOTP_data. Установить бит NCEN в лог. «0», бит SLEEP в лог. «0», бит PGM в лог. «1». Затем поднять напряжение на выводе `VPP` с уровня чтения 3,3 В до напряжения программирования 9,8 В ± 0,3 В, выдержать 200 мс, после чего вернуть напряжение `VPP` на уровень чтения 3,3 В, сбросить бит PGM, установить бит REN и проверить записанное значение с помощью считывания ячейки BOTP_data.

Внимание! Если Вы прожгли бит OTP_init_on, при подаче питания микросхема не отвечает, то вероятной причиной является отсутствие надлежащей настройки делителя частоты. Для того чтобы микросхема ответила в этом состоянии нужно найти её IC_addr, путём последовательной записи в регистр BUS_addr всех 255 возможных значений от 1 до 255 (см. раздел Методика подключения → Обнаружение микросхемы). После нахождения адреса, нужно запрограммировать делитель BOTP_clkdel на значение, обеспечивающее Fclk_rom ≤ 1 МГц (подробно в описании BOTP_clkdel).

Внимание! Чтение ячеек последовательной ПЗУ после инициализации по событию подачи питания или сброса будет давать нулевые значения. Для того чтобы прочитать необходимо выставить бит OVERRIDE_UVAL, снять бит OTP_init_on, далее снять MANUAL_NRST_PLL, далее установить MANUAL_NRST_PLL, и тогда ячейки ПЗУ будут читаться.

Порядок программирования ПЗУ

Автозагрузка конфигурации при включении питания включается битом OTP_init_on регистра INIT_conf. Этот бит принадлежит ПЗУ с параллельным доступом (UOTP) и должен прожигаться **в последнюю очередь** — только после полного прожига и успешной проверки BOTP. Правильная последовательность программирования ПЗУ:

1. **Тактирование.** Настройте PLL_config и выберите значение BOTP_clkdel по таблице делителей (см. описание регистров) так, чтобы частота тактирования ПЗУ Fclk_rom = FINT / N не превышала 1 МГц.
2. **Прожиг UOTP без автозагрузки.** Запишите и прожгите в UOTP регистры PLL_config и INIT_conf с выбранным значением BOTP_clkdel; бит OTP_init_on при этом должен быть равен **0** (последовательность прожига — в разделе ПЗУ с параллельным доступом).
3. **Полный прожиг BOTP.** Прожгите все 512 ячеек BOTP, обязательно включая область регистров конфигурации — ячейки 504–511 (IC_addr, ADC_config, Mask_Stat, Flags_delay, MR_lock, CMP_lth, AFE_config, Mode_config — см. таблицу 12). Пропускать ячейки нельзя: незаполненная (пустая) ячейка читается как ноль, и при автозагрузке соответствующий регистр будет инициализирован нулём.
4. **Проверка BOTP.** Прочитайте все 512 ячеек и убедитесь, что содержимое совпадает с заданным (о порядке чтения — предупреждение выше). При несовпадении повторите прожиг.
5. **Включение автозагрузки.** Повторно прожгите INIT_conf в UOTP со значением OTP_init_on = **1** (до этого бит должен быть прожжён нулём). Значение BOTP_clkdel и

остальные биты изменять нельзя.

6. **Проверка автостарта.** Снимите питание (или подайте сброс) и убедитесь, что микросхема самостоятельно инициализировалась из ПЗУ и работоспособна.

Внимание! Если прожечь бит OTP_init_on до программирования ВОТР (как минимум — области регистров конфигурации, ячейки 504–511), то при каждом включении питания контроллер инициализации запишет во все регистры содержимое пустой ПЗУ — нули: микровычислители будут выключены (Mode_config = 0), микросхема не будет отвечать по последовательному интерфейсу (IC_addr = 0) и окажется полностью неработоспособной. Микросхема при этом не повреждается, но будет находиться в неизвестном состоянии — для её возвращения к работоспособности потребуются значительные усилия.

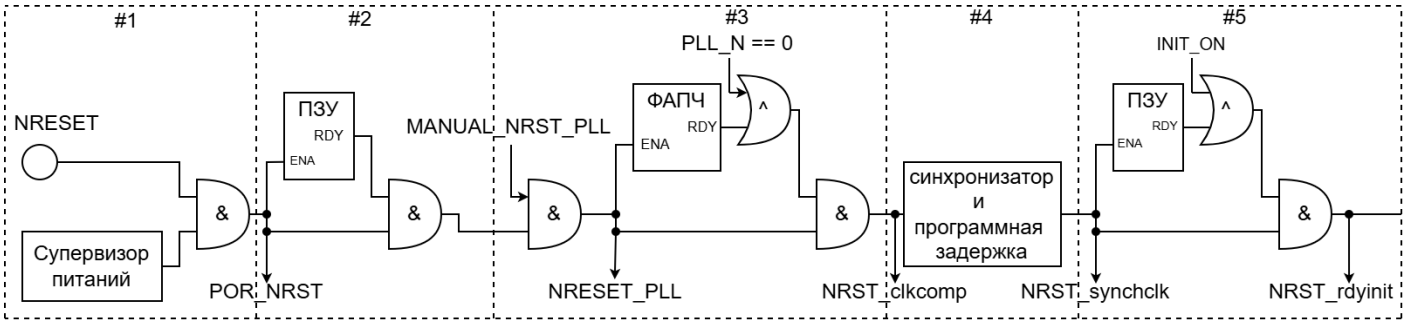
*Последнее обновление **21 сент. 2026 г.***

Подсистема сброса

Система сброса решает следующие задачи:

- Блокирование работы микросхемы до установления требуемых уровней питания, блокирование работы цифрового блока до установления тактирования и инициализации начальными данными.
- Очистка памяти и установка всех регистров в начальное состояние по внешнему сигналу "сброс"

Сигнал "сброс" может быть подан сигналом низкого уровня на вывод RESET



Цепочка распространения сигнала сброс

Таблица этапов распространения сигнала сброс

#	Бит SPI	Этап	Описание
1	NRST_pll	Формирование	сигнал с вывода NRESET объединяется с сигналом готовности супервизора питания
2		Чтение настроек ПЗУ с параллельным доступом	Производится перезапись из ПЗУ с параллельным доступом настроек ИОН, ФАПЧ, и режимов инициализации.

#	Бит SPI	Этап	Описание
3	NRST_clkcomp	ФАПЧ	Ожидание готовности аналоговой части ФАПЧ
4	NRST_synchclk	Программная задержка	Синхронизация сигнала сброса и программная задержка
5	NRST_rdyinit	Чтение настроек ПЗУ с последовательным доступом	Ожидание готовности ФАПЧ и программной задержки

*Последнее обновление **22 мая 2026 г.***

Микровычислители

Микросхема содержит два независимых специализированных микровычислителя (CPU1 и CPU2), предназначенных для:

- коррекции статических нелинейностей датчика угла;
- согласования отсчётов в многополюсных датчиках;
- расчёта угловой скорости, ускорения и фильтрации;
- буферизации внутренних сигналов преобразователей.

Наличие двух CPU позволяет одному выполнять коррекцию нелинейностей, а другому — расчёт скорости и ускорения, что обеспечивает средний темп вычислений 50–150 мкс.

Каждый CPU имеет нестандартную разрядность 14 бит, сокращённый набор из 48 команд, 8 регистров общего назначения (R0–R5 по 14 бит, R6 и R7 по 28 бит) и 512 ячеек памяти программ и данных по 14 бит.

Архитектура конвейера

Каждый CPU реализован на основе 4-ступенчатого конвейера:

Ступень	Назначение
IF	Выборка инструкции из памяти по адресу счётчика команд (PC)
ID	Декодирование инструкции, чтение регистров
EX	Исполнение: АЛУ, CORDIC, умножитель, вычисление адреса, запись в память
MEM/WB	Выбор результата, запись в регистр

Базовое время выполнения одной инструкции — 4 такта FINT. Результат умножения или операции CORDIC может быть готов позже, но не блокирует выполнение следующих

инструкций, если они не используют регистры R6 или R7.

Исключения:

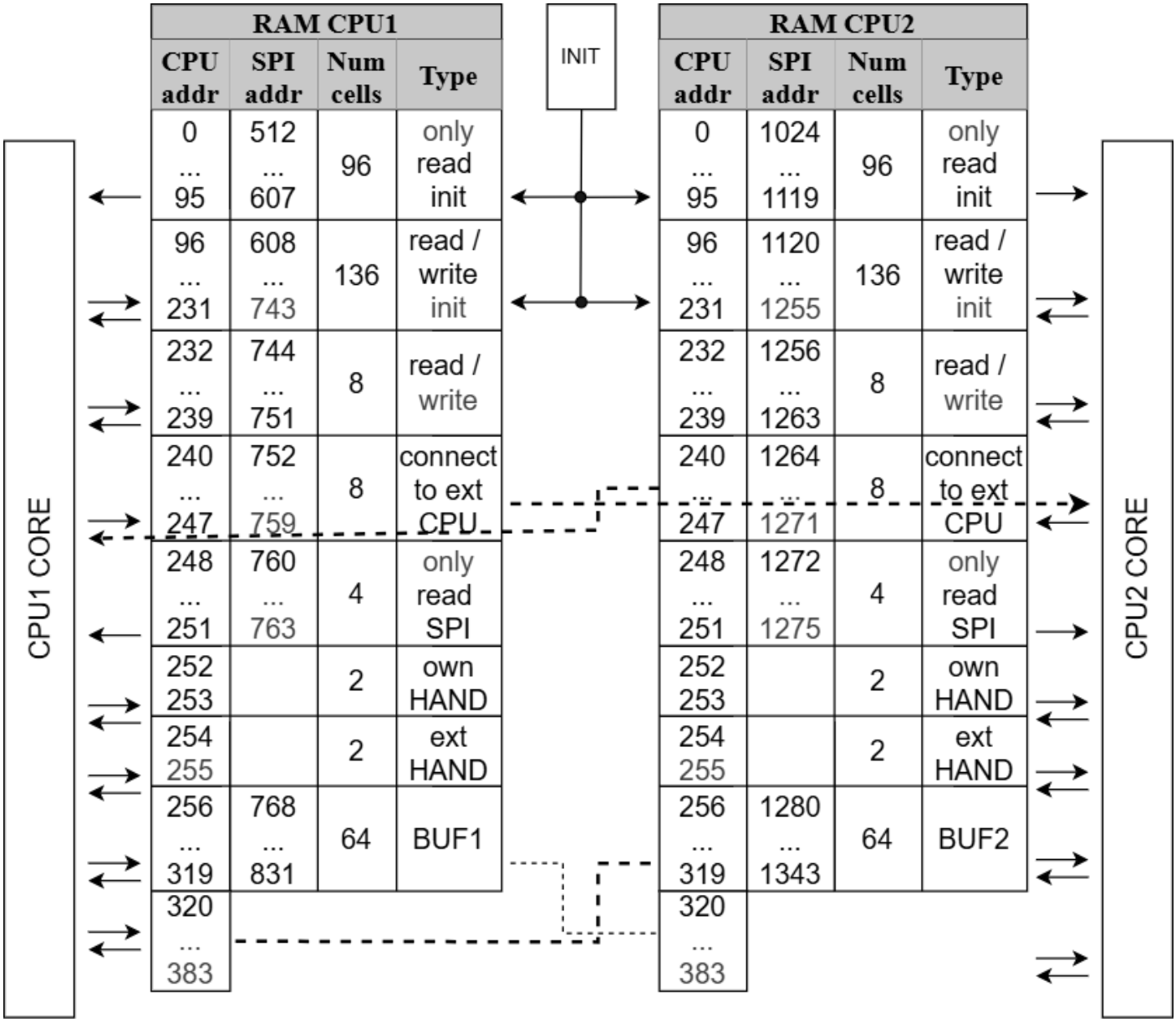
- **CORDIC** (SIN, COS, ATAN) — 16 итераций, ~17 тактов;
- **Умножение** (MULTI, MULTF и др.) — 5–10 тактов;
- **NOP N** — N+1 тактов (задержка);
- **WAIT**_* — блокировка до наступления события.

Регистры общего назначения

Регистр	Разрядность	Назначение
R0–R5	14 бит	Регистры общего назначения. При чтении знакорасширяются до 28 бит
R6	28 бит	Результат умножения/сдвига. Всегда перезаписывается командами MULTI, MULTI_ACC, MULTF, MULTF_ACC, MULTCUBE_ACC, DEC2FLOAT
R7	28 бит	Аккумулятор. Всегда перезаписывается командами ACC, ACC2, MULTI_ACC, MULTF_ACC, MULTCUBE_ACC

Загрузка и сохранение R6 и R7 возможна только форматами по 28 бит (2 ячейки) или 56 бит (4 ячейки). Загрузить или сохранить только младшие 14 бит R6/R7 нельзя. Для операций с R6/R7 использовать команды LOAD32/STORE32, CLOAD32/CSTORE32.

Структура памяти



Структура памяти CPU

Объём памяти каждого CPU — 512 ячеек по 14 бит. Карта адресов:

Адреса	Размер	Назначение	Доступ CPU
0–95	96	Программный код (инструкции)	Только чтение
96–239	144	Данные	Чтение/запись

Адреса	Размер	Назначение	Доступ CPU
240–247	8	Обмен между CPU1 и CPU2	Чтение/запись (кросс)
248–251	4	Ячейки SPI	Чтение/запись
252–253	2	HAND (данные преобразователя)	Чтение/запись
254–255	2	HAND ext (данные внешнего преобразователя)	Чтение/запись
256–383	128	Буфер (BUF)	Чтение/запись

При включённой функции инициализации (бит INIT_on) в первые 232 ячейки обоих CPU переписываются данные из ПЗУ с последовательным доступом. Поэтому ВОТР должна быть полностью запрограммирована до включения бита OTP_init_on — при пустой ПЗУ ячейки CPU и регистры конфигурации инициализируются нулями (см. ПЗУ → Порядок программирования ПЗУ).

Ячейки обмена данными CPU

Ячейки с адресами 240–247 предназначены для передачи данных между CPU1 и CPU2. Шины чтения подключены кросс-образованием: чтение CPU1 по адресам 240–247 возвращает данные, записанные CPU2, и наоборот.

Ячейки обмена с интерфейсом SPI

Ячейки памяти CPU доступны для чтения и записи через интерфейс SPI через регистры Dcpu1LB/Dcpu1HB и Dcpu2LB/Dcpu2HB. Адрес ячейки задаётся в составе транзакции SPI. При записи данные фиксируются в соответствующей ячейке памяти CPU. При чтении возвращается текущее значение ячейки.

Буферы

В микросхему добавлены два дополнительных блока памяти BUF1 и BUF2 по 64 ячейки каждый (адреса 256–319). По умолчанию BUF1 расширяет память CPU1, а BUF2 — CPU2. При необходимости буфер может быть переключён на другой CPU с помощью битов SHRD_RAM и SHRD_CPU2, что позволяет одному CPU работать с 128 дополнительными ячейками (адреса 256–383).

Адресация памяти

Команды загрузки и сохранения используют 7-битное поле адреса. Фактический адрес зависит от типа команды и регистра:

Команда	Регистр	Ячейки	Диапазон адресов	Формула адреса
STORE/CSTORE	R0–R5	1	128–255	addr + 128
LOAD/CLOAD	R0–R5	1	128–255	addr + 128
STORE32/CSTORE32	R0–R5	2	64–318 (чётные)	addr × 2 + 64
LOAD32/CLOAD32	R0–R5	2	64–318 (чётные)	addr × 2 + 64
STORE/CSTORE	R6, R7	4	64–318 (чётные)	addr × 2 + 64
LOAD/CLOAD	R6, R7	4	64–318 (чётные)	addr × 2 + 64
STORE32/CSTORE32	R6, R7	4	64–318 (чётные)	addr × 2 + 64
LOAD32/CLOAD32	R6, R7	4	64–318 (чётные)	addr × 2 + 64

Для 32-битных операций (STORE32, LOAD32) и операций с R6/R7 адрес всегда умножается на 2. Ячейка с чётным адресом помещается в младшую часть регистра, с нечётным (адрес +1) — в старшую. Для 4-ячеистых операций: `{data[addr+3], data[addr+2], data[addr+1], data[addr]}`.

Форматы данных

28-битный формат с плавающей запятой

Формат является усечённым IEEE754 float32. Нет NaN, Inf, denormal.

Структура формата

Поле	Биты	Длина	Описание
Sign	27	1	0 = +, 1 = –
Exp	26..20	7	Порядок, bias = 63
Mant	19..0	20	Мантисса (IEEE754 fraction >> 3)

Кодирование (Float32 → Float28)

1. `v = single(x)` (IEEE754 float32)
2. Извлечь поля IEEE754: `Sign = bit31`, `Exp32 = bits30..23`, `Frac32 = bits22..0`
3. Порядок: если `Exp32 == 0`, то `Exp28 = 0`, иначе `Exp28 = Exp32 − 127`
4. `Exp = Exp28 + 63`
5. `Mant = Frac32 >> 3`
6. Упаковка: `Float28 = (Sign << 27) | (Exp << 20) | Mant`

Float28 → Float32

1. Извлечь: `Sign = bit27`, `Exp = bits26..20`, `Mant = bits19..0`
2. Восстановление: `Exp32 = Exp − 63 + 127`, `Frac32 = Mant << 3`
3. `Value = Sign × (1 + Frac32 / 2^23) × 2^(Exp − 63)`

Float28 → Integer

`INTvalue = Sign × Mant × 2^(Order - 63)`, где `Sign = +1` если бит 27 = 0, иначе `−1`.

Система команд

Все инструкции кодируются 14-битным словом. Биты [5:3] кодируют регистр-источник B (SEL_B), биты [2:0] — регистр-приёмник A (SEL_A). Регистры R0–R7 кодируются значениями 000–111.

Арифметико-логические команды

Команда	Код	Аргументы	Описание
NOP	`00000000000`	[N]	Нет операции. При N > 0 — задержка N+1 тактов
CLR	`00000000001`	A	A = 0
INC	`00000000010`	A	A = A + 1
DEC	`00000000011`	A	A = A – 1
ABS	`00000000100`	A	A = |A|
ACC	`00000000111`	A	R7 = R7 + A
ACC2	`00000001`	B, A	R7 = R7 + A + B
ADD	`00000011`	B, A	A = A + B
SUB	`00000100`	B, A	A = A – B
ORL	`00000101`	B, A	A = A | B
ANL	`00000110`	B, A	A = A & B
MOVE	`00000111`	B, A	A = B
CONST	`1101`	val, A	A = знакорасширенное val (–64…+63)

Команды сдвига

Команда	Код	Аргументы	Описание
LSL	`000010`	count, A	Логический сдвиг влево: A = A << (count + 1)
ASR	`000011`	count, A	Арифметический сдвиг вправо: A = A >> (count + 1)

Команды умножения и преобразования форматов

Команда	Код	Аргументы	Описание
MULTI	`11110100`	B, A	R6 = A × B (целое, A — 28 бит, B — 24 бит, R6 — 52 бит)
MULTI_ACC	`11110101`	B, A	R6 = A × B; R7 = R7 + R6 (целое MAC)
MULTF	`11110110`	B, A	R6 = A × B (A — целое 28 бит, B — float 28 бит, R6 — 56 бит)
MULTF_ACC	`11110111`	B, A	R6 = A × B; R7 = R7 + R6 (float MAC)
MULTK24	`11111011`	B, A	A = RES[51:22], RES = A × B (целое, 24-битная константа)
MULTK14	`11111100`	B, A	A = RES[40:13], RES = A × B (целое, 14-битная константа)
MULTCUBE_ACC	`11111101`	B, A	R6 = (ch1+1)^(ch7+1) × (B + A); R7 = R7 + R6
DEC2FLOAT	`11111110000`	A	R6 = float(A) — преобразование целого в формат float28

Команды работы с памятью

Команда	Код	Аргументы	Описание
STORE	`0001`	addr, A	Запись 14 бит: ram[addr] = A[13:0]
STORE32	`0010`	addr, A	Запись 28 бит: ram[addr] = A[13:0], ram[addr+1] = A[27:14]
LOAD	`0011`	addr, A	Чтение 14 бит: A = знакорасширение(ram[addr])
LOAD32	`1100`	addr, A	Чтение 28 бит: A = `{ram[addr+1], ram[addr]}`

Индексные команды работы с памятью

Индексные команды CLOAD/CSTORE используют дополнительное смещение rel (см. раздел Канальные регистры). Для работы этих команд необходимо включить режим командой LOADSTORE_ON.

Команда	Код	Аргументы	Описание
CSTORE	`1001`	addr, A	Запись 14 бит: ram[addr + rel] = A[13:0]
CSTORE32	`1010`	addr, A	Запись 28 бит: ram[addr + rel] = A[13:0], ram[addr + rel + 1] = A[27:14]
CLOAD	`1000`	addr, A	Чтение 14 бит: A = знакорасширение(ram[addr + rel])
CLOAD32	`1011`	addr, A	Чтение 28 бит: A = `{ram[addr + rel + 1], ram[addr + rel]}`

Команды условного перехода

Все переходы используют относительное смещение от текущего PC.

Команда	Код	Аргументы	Описание
EQUAL	`010`	rel, B, A	PC = PC + rel если A == B (rel: −16…+15)
MORE	`011`	rel, B, A	PC = PC + rel если A > B, знаковое (rel: −16…+15)
DJNZ	`11100`	rel, A	A = A − 1; PC = PC + rel если A ≠ 0 (rel: −32…+31)
CDJNZ	`111010`	rel, CH#	CH# = CH# − 1; PC = PC + rel если CH# ≠ 0 (rel: −16…+15)
EQUAL0	`111011`	rel, A	PC = PC + rel если A == 0 (rel: −16…+15)
JUMP	`111100`	rel	PC = PC + rel (rel: −128…+127)
RJUMP	`11111111100`	A	PC = A (абсолютный переход по регистру)
STOREPC	`11111111101`	A	A = PC + 1 (сохранение адреса возврата)

CORDIC-команды

Аппаратный CORDIC-сопроцессор выполняет 16 итераций за ~17 тактов. SIN и COS вычисляются одновременно.

Команда	Код	Аргументы	Описание
SIN	`00000000101`	A	Вычисление sin(A). Результат: A = знакорасширенный cordic_sin[15:0]. Аргумент — 18-битный угол (2 старших бита — квадрант)
COS	`00000000110`	A	Вычисление cos(A). Результат: A = знакорасширенный cordic_cos[15:0]

Команда	Код	Аргументы	Описание
ATAN	`00000010`	B, A	Вычисление atan2(A, B). Результат: A — 18-битный угол, B — аргумент cos, A — аргумент sin

Канальные регистры

Канальные регистры используются для индексной адресации памяти и организации циклов.

Регистр	Разрядность	Команда загрузки	Источник данных
ch0	9 бит	CFIX0	A[8:0]
ch1	8 бит	CFIX1	A[7:0]
ch2	7 бит	CFIX2	A[6:0]
ch3	6 бит	CFIX3	A[5:0]
ch4	5 бит	CFIX4	A[4:0]
ch7	2 бит	CFIX7	A[1:0]

Смещение rel для индексных команд вычисляется как взвешенная сумма: `rel = ch0 + ch1 × 2 + ch2 × 4 + ch3 × 8 + ch4 × 16`

Команда	Код	Аргументы	Описание
CFIX0	`11111110100`	A	ch0 = A[8:0]
CFIX1	`11111110101`	A	ch1 = A[7:0]
CFIX2	`11111110110`	A	ch2 = A[6:0]

Команда	Код	Аргументы	Описание
CFIX3	`11111110111`	A	ch3 = A[5:0]
CFIX4	`11111111000`	A	ch4 = A[4:0]
CFIX7	`11111111001`	A	ch7 = A[1:0]
CLR_CH	`11111111010`	CH#	CH# = 0. При CH# = 5 — очистка всех канальных регистров
CLOAD_CUBE	`11111111011`	A	A = (ch1+1)^(ch7+1) — полиномиальный базис

NUMCUBE: значение (ch1+1)^(ch7+1) определяет порядок полинома:

ch7	Значение	Базис
0	ch1 + 1	Линейный
1	$(ch1 + 1)^2$	Квадратичный
2	$(ch1 + 1)^3$	Кубический
3	1	Константа

Команды управления HAND-интерфейсом

Команда	Код	Аргументы	Описание
SET_A_HAND	`11111110001`	const	Выбор номера пары ячеек преобразователя (0–7)
HFIX	`11111110010`	A	Установка номера HAND из регистра A[2:0]

Команды управления и синхронизации

Команда	Код	Описание
SETB_STP	`11111111110000`	Установка флага STP = 1
CLRB_STP	`11111111110001`	Сброс флага STP = 0
SW_HTE1	`11111111110010`	Переключение мультиплексора HAND1 на другой CPU
SW_HTE2	`11111111110011`	Переключение мультиплексора HAND2 на другой CPU
WAIT_EXT_CPU	`11111111111000`	Ожидание импульса от другого CPU
PULSE_EXT_CPU	`11111111111001`	Генерация импульса для другого CPU
WAIT_OWN_HAND	`11111111111010`	Ожидание данных от «родного» преобразователя
WAIT_EXT_HAND	`11111111111011`	Ожидание данных от «внешнего» преобразователя
LOADSTORE_ON	`11111111111100`	Разрешение работы команд CLOAD/CSTORE
LOADSTORE_OFF	`11111111111101`	Запрет работы команд CLOAD/CSTORE
IDLE	`11111111111111`	Остановка CPU (бесконечный NOP)

Взаимодействие с преобразователем

Подключение CPU1 и CPU2 к преобразователям (HAND1 и HAND2) осуществляется симметрично: для CPU1 «родным» (own) является HAND1, а внешним (ext) — HAND2. Для CPU2 — наоборот. Считывание данных производится с двойной шириной слова (28 бит): с own по адресам 252–253, с ext по адресам 254–255.

Выбор номера пары ячеек преобразователя выполняется командой SET_A_HAND.

#ADDR	Содержимое
0	`FullAngle[27:0]`
1	`Wca[11:0], BScos[12], ex_shifted, Wsa[11:0], BSsin[12], ex_ref`
2	`VirtualS[12:0], ex_recovered, BSsin[12:0], ex_recovered_90dgr`
3	`Amp_metric[11:0], Err_metric[15:0]`
4	`FullVel[27:0]`
5	`PhiC[15:2], PhiS[15:2]`
6	`VirtualC[12:0], ex_recovered, BScos[12:0], ex_recovered_90dgr`
7	`Pole_addi, STAT`

Запись данных в преобразователь

Доступ к записи в каждый преобразователь есть только у одного CPU. Выбор CPU с доступом на запись определяется конфигурационным битом HandToEXT. CPU может инвертировать действие флага HandToEXT с помощью команд SW_HTE1 и SW_HTE2 (цифра — номер преобразователя).

Запись Coord/Vel

Результаты вычислений CPU могут быть записаны в блок гистерезиса скорости и координаты. Для этого установите флаг Coord_from_cpu или Vel_from_cpu и запишите данные, установив командой SET_A_HAND соответствующий адрес (0 для координаты, 4 для скорости).

Запись Pole_addi

Для записи добавочного значения полюса `Pole_addi` выставите адрес 3. Запись возможна, если источник определён как CPU с помощью бита PoleAddi_src.

Синхронизация двух CPU

CPU1 и CPU2 могут обмениваться данными и синхронизировать работу:

- **Обмен данными** — через ячейки 240–247 (кросс-подключение шин чтения).
- **Программная синхронизация** — WAIT_EXT_CPU ожидает импульс от другого CPU, PULSE_EXT_CPU генерирует импульс.
- **Синхронизация с преобразователем** — WAIT_OWN_HAND ожидает обновление данных от «родного» преобразователя, WAIT_EXT_HAND — от «внешнего».
- **Флаг STP** — SETB_STP/CLRB_STP управляют однобитным выходом CPU, может использоваться для внешней сигнализации.

Отладчик

Каждый CPU имеет встроенный отладчик, управляемый через SPI-регистры DBG1_ctrl/DBG1_data и DBG2_ctrl/DBG2_data.

Управление выполнением

Регистр DBG_ctrl:

Биты	Назначение
[12:11]	Команда: 0 = RUN, 1 = STOP, 2 = STEP
[10]	Триггер обработки команды (по фронту)
[9]	Разрешение точки останова 1
[8:0]	Адрес точки останова 1

Регистр DBG_data:

Биты	Назначение
[9]	Разрешение точки останова 2
[8:0]	Адрес точки останова 2

Чтение регистров

Чтение внутренних регистров CPU производится через SPI-команду READ_CPU_REGS. Поле sel_reg (биты [13:10]) выбирает данные:

sel_reg	Данные
0–5	R0–R5 (14 бит)
6–7	R6/R7 младшие 14 бит
8–9	R6/R7 старшие 14 бит
10	Статус: DBG_STOP, STP, PC, инструкция
11	Каналы: a_hand, ch1, ch4, ch0
12	Каналы: ch7, ch_rel, ch2, ch3

Пример программы

CPU1: Коррекция угла по 1-й гармонике

Программа читает угол из преобразователя, вычисляет sin(угол) через CORDIC, умножает на float-коэффициент из памяти и вычитает полученную коррекцию из исходного угла. Результат передаётся CPU2 через ячейки обмена.

```
SET_A_HAND 0                          // Выбрать FullAngle

LOOP: WAIT_OWN_HAND                   // Ждать обновление данных
LOAD32 HAND1_L R0                     // R0 = текущий угол (28 бит,
адреса 252-253)

CLR R7                                // R7 = 0 (аккумулятор коррекции)
SIN R0                                // CORDIC: R0 = sin(угол)
LOAD32 228 R1                         // R1 = коэффициент (float28,
адреса 228-229)
MULTF_ACC R1 R0                       // R6 = R0(целое) × R1(float); R7
+= R6

ASR 15 R7                             // Масштабирование: R7 >>= 16
MOVE R7 R2                            // R2 = коррекция

LOAD32 HAND1_L R0                     // R0 = исходный угол (перечитать)
SUB R2 R0                             // R0 = угол − коррекция

STORE32 EXTCPU_0 R0                   // Исправленный угол → CPU2 (адреса
240-241)
PULSE_EXT_CPU                         // Сигнал готовности CPU2
JUMP LOOP                             // Следующий цикл
```

Используемые переменные (из файла cpu1_ram.txt):

Переменная	Адрес	Описание
HAND1_L	252	Младшие 14 бит OWN HAND
HAND1_H	253	Старшие 14 бит OWN HAND
EXTCPU_0	240	Младшие 14 бит обмена CPU1→CPU2
EXTCPU_1	241	Старшие 14 бит обмена CPU1→CPU2
(адрес 228)	228–229	Float28 коэффициент коррекции

Пошаговое описание:

1. `SET_A_HAND 0` — выбирает пару ячеек FullAngle для чтения из преобразователя.
2. `WAIT_OWN_HAND` — останавливает CPU до появления новых данных от «родного» преобразователя.
3. `LOAD32 HAND1_L R0` — загружает 28-битный угол из ячеек 252 (младшие) и 253 (старшие).
4. `CLR R7` — обнуляет аккумулятор.
5. `SIN R0` — CORDIC вычисляет sin(R0), результат записывается в R0. CORDIC одновременно вычисляет cos, результат доступен для следующей команды COS.
6. `LOAD32 228 R1` — загружает 28-битный float-коэффициент из ПЗУ.
7. `MULTF_ACC R1 R0` — R6 = R0 (целое sin) × R1 (float коэфф.); R7 += R6.
8. `ASR 15 R7` — арифметический сдвиг R7 вправо на 16 бит (масштабирование накопленной суммы).
9. `MOVE R7 R2` — R2 = коррекция.
10. `LOAD32 HAND1_L R0` — перечитывает исходный угол (данные в HAND защёлкнуты).
11. `SUB R2 R0` — R0 = исходный угол − коррекция.
12. `STORE32 EXTCPU_0 R0` — записывает 28-битный результат в ячейки 240–241 для CPU2.
13. `PULSE_EXT_CPU` — генерирует импульс, по которому CPU2 узнаёт о готовности данных.

Сводная таблица команд

Имя	Код операции	Аргументы	Описание
NOP	`00000000000`	[count]	Нет операции; задержка count+1 тактов
CLR	`00000000001`	A	A = 0
INC	`00000000010`	A	A = A + 1
DEC	`00000000011`	A	A = A − 1
ABS	`00000000100`	A	A = |A|

Имя	Код операции	Аргументы	Описание
SIN	`00000000101`	A	A = sin(A), CORDIC 16 итераций
COS	`00000000110`	A	A = cos(A), CORDIC 16 итераций
ACC	`00000000111`	A	R7 = R7 + A
ACC2	`00000001`	B, A	R7 = R7 + A + B
ATAN	`00000010`	B, A	A = atan2(A, B)
ADD	`00000011`	B, A	A = A + B
SUB	`00000100`	B, A	A = A – B
ORL	`00000101`	B, A	A = A | B
ANL	`00000110`	B, A	A = A & B
MOVE	`00000111`	B, A	A = B
LSL	`000010`	count, A	A = A << (count + 1)
ASR	`000011`	count, A	A = A >> (count + 1)
STORE	`0001`	addr, A	ram[addr] = A[13:0]
STORE32	`0010`	addr, A	ram[addr..addr+1] = A[27:0]
LOAD	`0011`	addr, A	A = знакорасширение(ram[addr])
EQUAL	`010`	rel, B, A	PC += rel если A == B
MORE	`011`	rel, B, A	PC += rel если A > B

Имя	Код операции	Аргументы	Описание
CLOAD	`1000`	addr, A	A = знакорасширение(ram[addr + rel])
CSTORE	`1001`	addr, A	ram[addr + rel] = A[13:0]
CSTORE32	`1010`	addr, A	ram[addr+rel..addr+rel+1] = A[27:0]
CLOAD32	`1011`	addr, A	A = `{ram[addr+rel+1], ram[addr+rel]}`
LOAD32	`1100`	addr, A	A = `{ram[addr+1], ram[addr]}`
CONST	`1101`	val, A	A = знакорасширение(val), –64…+63
DJNZ	`11100`	rel, A	A--; PC += rel если A ≠ 0
CDJNZ	`111010`	rel, CH#	CH#--; PC += rel если CH# ≠ 0
EQUAL0	`111011`	rel, A	PC += rel если A == 0
JUMP	`111100`	rel	PC += rel
MULTI	`11110100`	B, A	R6 = A × B, целое
MULTI_ACC	`11110101`	B, A	R6 = A × B; R7 += R6, целое
MULTF	`11110110`	B, A	R6 = A × B, A — целое, B — float
MULTF_ACC	`11110111`	B, A	R6 = A × B; R7 += R6, float
MULTK24	`11111011`	B, A	A = (A × B)[51:22]

Имя	Код операции	Аргументы	Описание
MULTK14	`11111100`	B, A	A = (A × B)[40:13]
MULTCUBE_ACC	`11111101`	B, A	R6 = (ch1+1)^(ch7+1) × (B+A); R7 += R6
DEC2FLOAT	`11111110000`	A	R6 = float(A)
SET_A_HAND	`11111110001`	const	Выбор адреса HAND (0–7)
HFIX	`11111110010`	A	Установка HAND из A[2:0]
CFIX0	`11111110100`	A	ch0 = A[8:0]
CFIX1	`11111110101`	A	ch1 = A[7:0]
CFIX2	`11111110110`	A	ch2 = A[6:0]
CFIX3	`11111110111`	A	ch3 = A[5:0]
CFIX4	`11111111000`	A	ch4 = A[4:0]
CFIX7	`11111111001`	A	ch7 = A[1:0]
CLR_CH	`11111111010`	CH#	CH# = 0; при CH#=5 очистка всех
CLOAD_CUBE	`11111111011`	A	A = (ch1+1)^(ch7+1)
RJUMP	`11111111100`	A	PC = A
STOREPC	`11111111101`	A	A = PC + 1
SETB_STP	`11111111110000`		STP = 1
CLRB_STP	`11111111110001`		STP = 0

Имя	Код операции	Аргументы	Описание
SW_HTE1	`11111111110010`		Переключение HAND1
SW_HTE2	`11111111110011`		Переключение HAND2
WAIT_EXT_CPU	`11111111111000`		Ожидание импульса от CPU
PULSE_EXT_CPU	`11111111111001`		Импульс для CPU
WAIT_OWN_HAND	`11111111111010`		Ожидание own HAND
WAIT_EXT_HAND	`11111111111011`		Ожидание ext HAND
LOADSTORE_ON	`11111111111100`		Включить CLOAD/CSTORE
LOADSTORE_OFF	`11111111111101`		Выключить CLOAD/CSTORE
IDLE	`11111111111111`		Остановка CPU

*Последнее обновление **21 сент. 2026 г.***

Регистры конфигурации

Таблица адресов регистров

+	2 x 0	2 x 1	2 x 2	2 x 3	2 x 4	2 x 5
2 x 0	C1KampS	C1KampC	C1KbiasS	C1KbiasC	C1fbias	C1ExPhShft
2 x 8	C1Amp_th	C1InputStngs	C1Lock_th	C1Zero	C1Mask	C1KonturStng
2 x 16	C1Coord	C1CoordHB	C1AdcS	C1AdcC	C1OutS	C1VirtualS
2 x 24	C1Vel	C1VelHB	C1PhiS	C1PhiC	C1OutC	C1VirtualC
2 x 32	C2KampS	C2KampC	C2KbiasS	C2KbiasC	C2fbias	C2ExPhShft
2 x 40	C2Amp_th	C2InputStngs	C2Lock_th	C2Zero	C2Mask	C2KonturStng
2 x 48	C2Coord	C2CoordHB	C2AdcS	C2AdcC	C2OutS	C2VirtualS

+	2 x 0	2 x 1	2 x 2	2 x 3	2 x 4	2 x 5
2 x 56	C2Vel	C2VelHB	C2PhiS	C2PhiC	C2OutC	C2VirtualC
2 x 64	IC_addr	ADC_config	Mask_Stat	Flags_delay	WR_lock	CMP_lth
2 x 72	NOCLK_stat	SPI_req	alive_cnt	Stat_main	Dcpu1LB	Dcpu1HB
2 x 80	PLL_config	INIT_conf	UOTP_ctrl	BUS_addr	BOTP_addr	BOTP_data
2 x 88	-	-	-	-	P1BG_ctrl	P1BG_data
2 x 96	-	-	-	-	-	-

C1KampS/C2KampS

Адрес: 0/32

Описание: KampS [15:0] – коэффициент усиления по каналу АЦП IOSA1(преобразователь 1), IOSA2 (преобразователь 2). Беззнаковое значение, всегда положительное. Амплитуда сигналов на этих входах микросхемы умножается на значение из данного регистра и делится на 1024. Значение по умолчанию соответствует амплитуде сигнала на входе равному входному диапазону АЦП (0 ÷ 2,5 В) для режима СКВТ. При изменении значений в

данном регистре необходимо следить за флагами переполнения, а также за срабатыванием компараторов порогов.

Тип доступа: Чтение/Запись (R/W)

C1KampC/C2KampC

Адрес: 1/33

Описание: KampC [15:0] – коэффициент усиления по каналу АЦП IOCA1 (преобразователь 1), IOCA2 (преобразователь 2). Беззнаковое значение, всегда положительное. Амплитуда сигналов на этих входах микросхемы умножается на значение из данного регистра и делится на 1024. Значение по умолчанию соответствует амплитуде сигнала на входе равному входному диапазону АЦП (0 ÷ 2,5 В) для режима СКВТ. При изменении значений в данном регистре необходимо следить за флагами переполнения, а также срабатыванием компараторов порогов. В режиме Sensor_mode=01 максимальное значение 1024.

Тип доступа: Чтение/Запись (R/W)

C1KbiasS/C2KbiasS

Адрес: 2/34

Описание: KbiasS [15:0] – смещение нуля по каналу АЦП IOSA1 (преобразователь 1), IOSA2 (преобразователь 2). Знаковое значение в дополнительном коде. Максимум +32767, минимум -32767.

Тип доступа: Чтение/Запись (R/W)

C1KbiasC/C2KbiasC

Адрес: 3/35

Описание: KbiasC [15:0] – смещение нуля по каналу АЦП IOCA1 (преобразователь 1), IOCA2 (преобразователь 2). Знаковое значение в дополнительном коде. Максимум +32767, минимум -32767.

Тип доступа: Чтение/Запись (R/W)

C1fbias/C2fbias

Адрес: 4/36

Описание: fbias[15:0] – коррекция неортогональности обмоток СКВТ. Смещение фазы обмотки sin. Знаковое значение в дополнительном коде.

Тип доступа: Чтение/Запись (R/W)

C1ExPhShft/C2ExPhShft

Адрес: 5/37

Описание: ExPhShft[15:0] – задает сдвиг по фазе сигнала с EXO1 (преобразователь 1), EXO2 (преобразователь 2) на плате до входов IOSA1, IOCA1 (преобразователь 1), IOSA2, IOCA2 (преобразователь 2). Используется для определения квадранта положения СКВТ. Знаковое значение в дополнительном коде.

Тип доступа: Чтение/Запись (R/W)

C1ExoStngs/C2ExoStngs

Адрес: 6/38

Описание: Настройки Amp_code

Тип доступа: Чтение/Запись (R/W)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	11..10	**EXO_mode**	R/W	0	EXO_mode - режим формирования опоры возбуждения датчика 00 -

№	Биты	Имя поля	Доступ	Сброс	Описание
					выключено. При EXO_inv=0 на EXO1(6) выводится 0 В, при EXO_inv=1 на EXO1(6) выводится VDDEXO1(5) 01 - вывод меандра 10 - вывод синусоидального напряжения с помощью ЦАП 11 - вывод постоянного значения C1EXInc/C2EXInc По умолчанию EXO_mode = 00
2	9	**PWR_X2**	R/W	0	Добавление усилительного каскада при формировании меандра на EXO1(6) для увеличения выходного тока: 0 - без увеличения 1 - с увеличением По умолчанию 0.
3	8	**EXO_inv**	R/W	0	Инверсия выхода EXO1(6) в состоянии выключено, или при выводе меандра: 0 - без инверсии 1 - с инверсией По умолчанию 0.
4	7..0	**Amp_code**	R/W	0	Коэффициент усиления выходного напряжения ЦАП

C1EXInc/C2EXInc

Адрес: 7/39

Описание: EXInc[15:0] задает приращение фазы синусоидального сигнала на каждом такте ЦАП (Fclk). Значение рассчитывается по формуле (5). Значение по умолчанию соответствует частоте 12 кГц.

Тип доступа: Чтение/Запись (R/W)

C1Amp_th/C2Amp_th

Адрес: 8/40

Описание: Порог компаратора для вычисления флагов UIN в регистре C1Stat (преобразователь 1), C2Stat (преобразователь 2).

Тип доступа: Чтение/Запись (R/W)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	15..8	**Amp_hth**	R/W	187	Порог компаратора для вычисления флага UIN_HIGH в регистре C1Stat (преобразователь 1), C2Stat (преобразователь 2). Amp_hth безразмерная относительная величина. Рекомендуется подобрать значение этой величины такой, чтобы при нормальной работе преобразователя не возникало срабатывание компаратора. Текущее значение величины можно считать из регистров C1Amp_metric, C2Amp_metric.
2	7..0	**Amp_lth**	R/W	62	Порог компаратора для вычисления флага UIN_LOW в регистре C1Stat (преобразователь 1), C2Stat (преобразователь 2). Amp_lth безразмерная относительная величина. Рекомендуется подобрать значение этой величины такой, чтобы при нормальной работе преобразователя не возникало срабатывание компаратора. Текущее значение величины можно считать из регистров C1Amp_metric, C2Amp_metric

C1InputStngs/C2InputStngs

Адрес: 9/41

Описание: Настройки входных групп: источник опорного сигнала, инверсия, фильтрация, калибровка АЦП, компенсация смещения

Тип доступа: Чтение/Запись (R/W)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	14..13	**EXI_insentv**	R/W	0	Глубина фильтра дребезга переключения сигнала EXI. 0 – без фильтра 1 – 2 цикла 2 – 4 цикла 3 – 8 циклов
2	12	**EXI_inv**	R/W	0	Инверсия входа EXI1: 0 – вход EXI поступает в схему без инверсии 1 – вход EXI инвертируется
3	11	**Ex_recovery_en**	R/W	1	Включение блока восстановления сигнала опорной частоты: 0 – блок восстановления сигнала опорной частоты отключен 1 – блок восстановления сигнала опорной частоты включен
4	10..9	**Ex_source**	R/W	0	Биты выбора источника сигнала Ex_ref для блока восстановления сигнала опорной частоты. 3 – Ex_ref преобразователя 2 дублирует Ex_recovered от преобразователя 1 (невозможно установить для преобразователя 1); 2 – Ex_ref преобразователя 2

№	Биты	Имя поля	Доступ	Сброс	Описание
					дублирует Ex_ref от преобразователя 1 (невозможно установить для преобразователя 1); 1 – с входа EXI1 для преобразователь 1 и для преобразователя 2; Примечание – Подача с EXI2 опорного сигнала не предусмотрена. Если включен режим сельсин, то используется вход EXI1 для обоих преобразователей. Если включен один из сдвоенных режимов, используется EXI1. 0 – с генератора опорной частоты
5	8	**OPA_en**	R/W	0	Включение входных операционных усилителей преобразователя: 0 – входные усилители отключены. Сигналы на АЦП подаются со входов IOSA, IOCA 1 – входные усилители включены. Сигналы на АЦП подаются с усилителей
6	7	**ADC_CAL**	R/W	0	Калибровка АЦП: 0 – обычный режим работы 1 – режим калибровки смещения АЦП В режиме калибровки смещения ОУ и АЦП отключаются от выводов микросхемы, и на вход ОУ подается опорное напряжение 1,25 В. Коды, получаемые с АЦП, усредняются для вычисления смещения АЦП, пока бит установлен в 1. После установки

№	Биты	Имя поля	Доступ	Сброс	Описание
					бита в 0, полученное значение смещения вычитается из кодов, получаемых с АЦП. Калибровка АЦП прямо влияет на погрешность микросхемы в режимах с немодулированным сигналом датчика. В режимах с модулированным сигналом датчика калибровка не обязательна. Чем больше время калибровки, тем более точная будет калибровка. Рекомендуемое время калибровки ~350 мс или больше.
7	6	**Autooff_adccal**	R/W	0	Отключение режима калибровки АЦП после первого расчета корректирующих коэффициентов: 0 – режим калибровки не отключается 1 – автоотключение режима калибровки при DC_carrier = 1 после установки флага HW_NotRDY = 0
8	5	**DC_carrier**	R/W	0	Входные сигналы без модуляции: 0 – входные сигналы модулированы 1 – входные сигналы без модуляции
9	4	**DC_correction**	R/W	1	Включение компенсации среднего уровня сигнала для АЦП: 0 – компенсация отключена 1 – компенсация включена При подаче на вход немодулированных сигналов этот

№	Биты	Имя поля	Доступ	Сброс	Описание
					бит должен быть установлен в состояние лог. «0»
10	3..0	**DC_depth**	R/W	0	Глубина коррекции смещения. Задается значение в диапазоне от 0 до 15. Расчет первых коэффициентов смещения производится после 4 циклов работы блока коррекции смещения Tdc_dep. Период одного цикла Tdc_dep определяется следующими зависимостями: (Tclk - период работы контура, для примера взято 6,4 мкс) 0 - 1 128*Tclk ~ 819 мкс 1 - 2 256*Tclk ~ 1,6 мс 2 - 4 512*Tclk ~ 3,3 мс ... Значение по умолчанию 0.

C1Lock_th/C2Lock_th

Адрес: 10/42

Описание: Порог компаратора для вычисления флага NLock в регистре C1Stat (преобразователь 1), C2Stat (преобразователь 2). Lock_th безразмерная относительная величина. Рекомендуется подобрать значение этой величины такой, чтобы при нормальной работе преобразователя не возникало срабатывание компаратора.

Тип доступа: Чтение/Запись (R/W)

C1Zero/C2Zero

Адрес: 11/43

Описание: Zero[15:0] – коррекция вычисленной координаты. Значение Zero прибавляется к вычисленной преобразователем координате. При коррекции угол представлен 16-битным значением, вне зависимости от настроек, заданных в C1ResCntrl или C2ResCntrl.

Тип доступа: Чтение/Запись (R/W)

C1Mask/C2Mask

Адрес: 12/44

Описание: Регистр маски Mask Значение, записанное в регистр маски Mask, включает работу соответствующих бит регистра C1Stat.

Тип доступа: Чтение/Запись (R/W)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	15	**MSK_NLock**	R/W	1	Маска для бита NLock (контур в неустановившемся режиме): 0 – бит NLock устанавливается в состояние лог. «0» 1 – бит NLock отражает действительное состояние флага
2	14..13	**MSK_quadrant**	R/W	0	Маска для бита quadrant (квадрант): 0 – биты quadrant[1:0] устанавливаются в состояние лог. «0» 1 – биты quadrant[1:0] отражают действительное состояние флагов

№	Биты	Имя поля	Доступ	Сброс	Описание
3	12	**MSK_Kontur_NotENA**	R/W	1	Флаг запуска контура сигнализирует, что контур запустился в соответствии с настройками. 0 - бит Kontur_NotENA устанавливается в состояние лог. «0» 1 - бит Kontur_NotENA функционирует
4	11	**MSK_RCV_NotRDY**	R/W	1	Маска для бита RCV_NotRDY: 0 - бит RCV_NotRDY устанавливается в состояние лог. «0» 1 - бит RCV_NotRDY функционирует
5	10	**MSK_MISS_EXREF**	R/W	1	Маска для бита MISS_EXREF: 0 - бит MISS_EXREF устанавливается в состояние лог. «0» 1 - бит MISS_EXREF функционирует
6	9	**MSK_EX_PH_OUTR**	R/W	1	Маска для битов EX_PH_HIGH, MISS_EXREF: 0 – биты EX_PH_HIGH, MISS_EXREF устанавливаются в состояние лог. «0» 1 – биты EX_PH_HIGH, MISS_EXREF отражают действительное состояние флагов
7	7	**MSK_C_LOOP_OVF**	R/W	1	Маска для бита C_LOOP_OVF (переполнение в следящем контуре): 0 – бит C_LOOP_OVF устанавливается в состояние

№	Биты	Имя поля	Доступ	Сброс	Описание
					лог. «0» 1 – бит C_LOOP_OVF отражает действительное состояние флага
8	6	**MSK_UIN_HIGH**	R/W	1	Маска для бита UIN_HIGH (амплитуда сигналов слишком велика): 0 – бит UIN_HIGH устанавливается в состояние лог. «0» 1 – бит UIN_HIGH отражает действительное состояние флага
9	5	**MSK_UIN_LOW**	R/W	1	Маска для бита UIN_LOW (амплитуда сигналов слишком мала): 0 – бит UIN_LOW устанавливается в состояние лог. «0» 1 – бит UIN_LOW отражает действительное состояние флага
10	4	**MSK_CORR_OVF**	R/W	1	Переполнение после коррекции амплитуды сигналов. Маска для бита CORR_OVF (переполнение после коррекции амплитуды сигналов). 0 – бит CORR_OVF устанавливается в состояние лог. «0» 1 – бит CORR_OVF отражает действительное состояние флага
11	3	**MSK_ADC_OVF**	R/W	1	Маска для бита ADC_OVF (переполнение из-за

№	Биты	Имя поля	Доступ	Сброс	Описание
					большой постоянной составляющей сигналов на входе АЦП): 0 – бит ADC_OVF устанавливается в состояние лог. «0» 1 – бит ADC_OVF отражает действительное состояние флага
12	2	**MSK_CLIP_COS**	R/W	1	Маска для бита CLIP_COS (переполнение АЦП по каналу cos): 0 – бит CLIP_COS устанавливается в состояние лог. «0» 1 – бит CLIP_COS отражает действительное состояние флага
13	1	**MSK_CLIP_SIN**	R/W	1	Маска для бита CLIP_SIN (переполнение АЦП по каналу sin): 0 – бит CLIP_SIN устанавливается в состояние лог. «0» 1 – бит CLIP_SIN отражает действительное состояние флага
14	0	**MSK_HW_NotRDY**	R/W	1	Маска для бита HW_NotRDY (коррекция АЦП не произведена): 0 – бит HW_NotRDY устанавливается в состояние лог. «0» 1 – бит HW_NotRDY отражает действительное состояние флага

C1KonturStngs/C2KonturStngs

Адрес: 13/45

Описание: Регистры режимов работы преобразователей

Тип доступа: Чтение/Запись (R/W)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	15	**Wait_1offset**	R/W	0	Режим начала работы блока восстановления частоты при DC_carrier=0, или контура при DC_carrier=1: 0 – включать работу блоков до расчета первых коэффициентов 1 – удерживать блоки в сбросе пока не рассчитаются первые коэффициенты смещений
2	14	**Vel_hist**	R/W	0	Гистерезис значений координаты: 0 – гистерезис отключен 1 – гистерезис включен
3	13	**Coord_hist**	R/W	0	Гистерезис значений координаты: 0 – гистерезис отключен 1 – гистерезис включен
4	12	**HandToEXT**	R/W	0	Адресная шина мультиплексора подключается не к своему CPU: 0 - HAND подключается к OWN CPU 1 - HAND подключается к EXT CPU
5	11..10	**Sensor_mode**	R/W	0	00 – режим СКВТ или сельсин. В модели датчика используется тригонометрическая функция (СКВТ,

№	Биты	Имя поля	Доступ	Сброс	Описание
					сельсин).01 – режим ЛРДТ с подключением по 5-ти проводной схеме. В модели датчика используется линейная функция. Модель датчика — ЛРДТ. 10 – режим ЛРДТ с подключением по схеме с последовательным соединением обмоток. В модели датчика используется линейная функция. Модель датчика — ЛРДТ.
6	9	**En_cross0**	R/W	0	Режим виртуального грубого отсчета: 0 - режим выключен 1 - режим включен. В зависимости от направления вращения датчика при переходе через "0" координаты, производится инкремент или декремент виртуального счетчика. Значение счетчика может быть перезаписано из CPU1/CPU2 для согласования отсчетов для датчиков с редукцией грубого и точного каналов.
7	8	**PoleAddi_src**	R/W	0	Источник добавки к виртуальному счетчику полюсов: 0 - SPI 1 - CPU
8	7..5	**InDelay**	R/W	0	InDelay [2:0] – компенсация задержки входного тракта микросхемы и фильтров на плате. Задается в тактах частоты Fclk. Беззнаковое положительное значение.

№	Биты	Имя поля	Доступ	Сброс	Описание
9	4..0	**LBW**	R/W	4	Настройка полосы пропускания следящего контура ((см. таблицу 9))

C1ResCntrl/C2ResCntrl

Адрес: 14/46

Описание: Регистр настройки выходной информации преобразователя

Тип доступа: Чтение/Запись (R/W)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	15	**Enc_en**	R/W	0	Включение блока эмуляции квадратурного энкодера: 0 – блок эмуляции квадратурного энкодера отключен 1 – блок эмуляции квадратурного энкодера включен Coord_resolution должен быть равным 2 (16 битный код) Значение Vel_resolution равно количеству младших битов, которые будут замаскированы Например, при Vel_resolution=8, энкодер будет работать 8 битным кодом при Vel_resolution=12, энкодер работает с 4 битным кодом
2	14..12	**Enc_presc**	R/W	0	Делитель частоты для сигналов энкодера: 000 – FINT/2 001 –

№	Биты	Имя поля	Доступ	Сброс	Описание
					FINT/3 010 – FINT/4 011 – FINT/5 100 – FINT/8 101 – FINT/16 110 – FINT/32 111 – FINT/64
3	11	**SPI_ext_en**	R/W	0	Разрешает режим прямой параллельной передачи результата: 0 – режим отключен 1 – режим включен При ENC_en == 1 состояние данного бита не имеет значения
4	9	**Vel_from_cpu**	R/W	0	Значения скорости режима прямой/параллельной передачи переопределить значениями из CPU (соответствующему данному каналу) 0 - использовать значения из контура 1 - переопределить скорость значениями из CPU
5	8..5	**Vel_resolution**	R/W	7	Устанавливает разрешение в регистре C1Vel (преобразователь 1), C2Vel (преобразователь 2).Значение из таблицы 9
6	4	**Coord_from_cpu**	R/W	0	Значения энкодера, режима прямой/параллельной передачи переопределить значениями из CPU (соответствующего данному каналу) 1 - переопределить координату значениями из CPU 0 - использовать значения из контура

№	Биты	Имя поля	Доступ	Сброс	Описание
7	3..0	**Coord_resolution**	R/W	2	Устанавливает разрешение в регистре C1Coord (преобразователь 1), C2Coord (преобразователь 2). Значение может быть вычислено по формуле: Количество бит = 18 shl Coord_resolution[3:0] или из таблицы 9. При получении данных с выхода эмуляции квадратурного энкодера устанавливать эти биты в значение 2 (16-битный код угла).

C1Vcnt_bound/C2Vcnt_bound

Адрес: 15/47

Описание: Порог переполнения виртуального счетчика в старших разрядах [15:0]

Тип доступа: Чтение/Запись (R/W)

C1Coord/C2Coord

Адрес: 16/48

Описание: Coord – координата, вычисленная в преобразователе. Разрядность зависит от настроек в регистрах C1ResCntrl и C2ResCntrl.

Тип доступа: Только чтение (RO)

C1CoordHB/C2CoordHB

Адрес: 17/49

Описание: CoordHB[11..0] - старшие 12 разрядов координаты

Тип доступа: Только чтение (RO)

C1AdcS/C2AdcS

Адрес: 18/50

Описание: Выход каналов АЦП и опорных сигналов

Тип доступа: Только чтение (RO)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	15	**msb_arg_cos**	RO	0	Старший разряд данных поступающих на контур канала COS
2	14	**Ex_shifted**	RO	0	Опорный сигнал EX_REF после сдвига на ExPhShift
3	13..2	**Dadc_sin**	RO	0	Выход АЦП канал SIN
4	1	**msb_arg_sin**	RO	0	Старший разряд данных поступающих на контур канала SIN
5	0	**Ex_ref**	RO	0	Опорный сигнал EX_REF

C1AdcC/C2AdcC

Адрес: 19/51

Описание: Выход каналов АЦП и опорных сигналов

Тип доступа: Только чтение (RO)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	15..12	**crc4_adc**	RO	0	Контрольная сумма по [AdcC[13:0], AdcS[15:0]], включается битом HAND_CRC4_en
2	11..0	**Dadc_cos**	RO	0	Выход АЦП канал COS

C1OutS/C2OutS

Адрес: 20/52

Описание: Код канала АЦП SIN после коррекции смещения и амплитуды. Является входным аргументом в контур.

Тип доступа: Только чтение (RO)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	15	**VirtualSin0**	RO	0	Младший бит виртуального сигнала датчика канала синус
2	14	**Ex_recovered_vs**	RO	0	Восстановленный сигнал опорной частоты
3	13..1	**arg_sin_kontur1**	RO	0	Данные 13-бит поступающие на контур

№	Биты	Имя поля	Доступ	Сброс	Описание
					канала SIN
4	0	**Ex_recovered90dgr_vs**	RO	0	Восстановленный сигнал опорной частоты, сдвинутый на 90 градусов

C1VirtualS/C2VirtualS

Адрес: 21/53

Описание: Виртуальные значения Sin участвующие в свертке

Тип доступа: Только чтение (RO)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	15..12	**crc4_vs**	RO	0	Контрольная сумма по битам [VirtualS[13:0], OutS[15:0]], включается битом HAND_CRC4_en
2	11..0	**VirtualSin12_1**	RO	0	Старшие 12 бит виртуального сигнала датчика канала синус

C1Err_metric/C2Err_metric

Адрес: 22/54

Описание: Метрика ошибки угла на выходе микросхемы. По модулю этой метрики срабатывает компаратор флага NLock.

Тип доступа: Только чтение (RO)

C1Amp_metric/C2Amp_metric

Адрес: 23/55

Описание: Метрика амплитуды сигнала на входе микросхемы. По этой метрике срабатывают компараторы флагов UIN_High, UIN_Low. Номинальное значение 400.

Тип доступа: Только чтение (RO)

C1Vel/C2Vel

Адрес: 24/56

Описание: Vel – скорость, вычисленная в преобразователе. Разрядность зависит от настроек в регистрах C1ResCntrl и C2ResCntrl.

Тип доступа: Только чтение (RO)

C1VelHB/C2VelHB

Адрес: 25/57

Описание: Старшие разряды скорости Vel

Тип доступа: Только чтение (RO)

C1PhiS/C2PhiS

Адрес: 26/58

Описание: Выход контура с учетом модели датчика, коэффициентов InDelay, KbiasS, Fbias

Тип доступа: Только чтение (RO)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	15..14	**phimodel_cos1_0**	RO	0	Младшие 2 бит виртуального сигнала датчика канала COS
2	13..0	**phimodel_sin**	RO	0	14 бит виртуального сигнала датчика канала SIN

C1PhiC/C2PhiC

Адрес: 27/59

Описание: Выход контура с учетом модели датчика, коэффициентов InDelay, KbiasC, Fbias

Тип доступа: Только чтение (RO)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	15..12	**crc4_ph**	RO	0	Контрольная сумма по битам [PhiC[13:0], PhiS[15:0]], включается битом HAND_CRC4_en
2	11..0	**phimodel_cos13_2**	RO	0	Старшие 12 бит виртуального сигнала датчика канала COS

C1OutC/C2OutC

Адрес: 28/60

Описание: Код канала АЦП COS после коррекции смещения и амплитуды. Является входным аргументом в контур.

Тип доступа: Только чтение (RO)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	15	**VirtualCos0**	RO	0	Младший бит виртуального сигнала датчика канала COS
2	14	**Ex_recovered_vc**	RO	0	Восстановленный сигнал опорной частоты
3	13..1	**arg_cos_kontur1**	RO	0	Данные 13-бит поступающие на контур канала COS
4	0	**Ex_recovered90dgr_vc**	RO	0	Восстановленный сигнал опорной частоты, сдвинутый на 90 градусов

C1VirtualC/C2VirtualC

Адрес: 29/61

Описание: Виртуальные значения Cos для вычисления ошибки в контуре

Тип доступа: Только чтение (RO)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	15..12	**crc4_vc**	RO	0	Контрольная сумма по битам [VirtualS[13:0], OutS[15:0]],

№	Биты	Имя поля	Доступ	Сброс	Описание
					включается битом HAND_CRC4_en
2	11..0	**VirtualCos12_1**	RO	0	Старшие 12 бит виртуального сигнала датчика канала COS

C1Stat/C2Stat

Адрес: 30/62

Описание: Stat - регистр ошибок/состояния канала преобразователя.

Тип доступа: Только чтение (RO)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	15	**NLock**	RO	1	Контур в неустановившемся режиме: 0 – ошибка в следящем контуре меньше чем значение в регистре C1Lock_th (преобразователь 1), C2Lock_th (преобразователь 2). 1 – ошибка в следящем контуре больше чем значение в регистре C1Lock_th (преобразователь 1), C2Lock_th (преобразователь 2).
2	14..13	**quadrant**	RO	0	Квадрант. Старшие два бита результата (без учета сдвига в блоке обработки результатов). При записи в Coord_resolution 1 или 0 позволяет получить

№	Биты	Имя поля	Доступ	Сброс	Описание
					старший бит или биты результата. Биты не влияют на состояние бит Ready
3	11	**Kontur_NotENA**	RO	0	Флаг запуска контура сигнализирует, что контур запустился в соответствии с настройками: 0 - контур запущен 1 - контур в ожидании наступления заданных условий
4	10	**RCV_NotRDY**	RO	0	Флаг готовности блока восстановления опорной частоты: пока RCV_NotRDY=1, контур удерживается в сбросе: 0 - опорная частота восстановлена 1 - опорная частота не восстановлена
5	9	**MISS_EXREF**	RO	0	Отсутствие опорного сигнала: 0 - EX_REF переключается чаще 10 Гц 1 - за секунду изменений логического уровня EX_REF не обнаружено
6	8	**EX_PH_OUTRANGE**	RO	0	Большой сдвиг фазы: 0 – сдвиг фазы между опорным и восстановленным сигналом меньше ± 40° 1 – сдвиг фазы между опорным и восстановленным сигналом больше ± 40°, что может приводить к избыточному шуму на выходе микросхемы.

№	Биты	Имя поля	Доступ	Сброс	Описание
7	7	**C_LOOP_OVF**	RO	0	Переполнение в следящем контуре: 0 – нет ошибки в следящем контуре 1 – ошибка в следящем контуре. Возможна некорректная работа микросхемы. В режимах Sensor_mode==0 установка данного бита может быть вариантом нормальной работы преобразователя, указывающим на то, что координата датчика достигла максимального/минимального значения.
8	6	**UIN_HIGH**	RO	0	Амплитуда сигналов слишком велика: 0 – амплитуда сигналов на входе следящего контура меньше заданного порога 1 – большая амплитуда сигналов на входе следящего контура. Определяется сравнением с порогом заданном в регистре Amp_hth (преобразователь 1), Amp_hth (преобразователь 2).
9	5	**UIN_LOW**	RO	0	Амплитуда сигналов слишком мала: 0 – амплитуда сигналов на входе следящего контура больше заданного порога 1 – малая амплитуда сигналов на входе следящего контура. Определяется сравнением с порогом заданном в регистре

№	Биты	Имя поля	Доступ	Сброс	Описание
					C1Amp_lth (преобразователь 1), C2Amp_lth (преобразователь 2).
10	4	**CORR_OVF**	RO	0	Переполнение после коррекции амплитуды сигналов: 0 – переполнение после коррекции усиления отсутствует 1 – слишком большой сигнал после коррекции усиления. Ошибка вызвана слишком большими коэффициентами C1KampS, C1KampC или C2KampS, C2KampC. Флаг сбрасывается если в течение 2,5 мс не было переполнений.
11	3	**ADC_OVF**	RO	0	Переполнение из-за большой постоянной составляющей сигналов на входе АЦП: 0 – переполнение из-за большой постоянной составляющей отсутствует 1 – постоянная составляющая сигнала(ов) вне диапазона. Флаг сбрасывается если в течение 18 с не было выхода за диапазон.
12	2	**CLIP_COS**	RO	0	Переполнение АЦП по каналу cos: 0 – амплитуда сигнала в канале cos не выходит за диапазон АЦП 1 – амплитуда сигнала в канале cos выходит за диапазон АЦП. Флаг сбрасывается если в течение

№	Биты	Имя поля	Доступ	Сброс	Описание
					2,5 мс не было выхода за диапазон.
13	1	**CLIP_SIN**	RO	0	Переполнение АЦП по каналу sin: 0 – амплитуда сигнала в канале sin не выходит за диапазон АЦП 1 – амплитуда сигнала в канале sin выходит за диапазон АЦП. Флаг сбрасывается если в течение 2,5 мс не было выхода за диапазон.
14	0	**HW_NotRDY**	RO	0	Коррекция АЦП не произведена: 0 – коррекция АЦП пройдена. Бит устанавливается 0 после первого расчета корректирующих коэффициентов АЦП (происходит за время ~18 с). При установке бита DC_carrier регистра InputStngs, этот бит устанавливается в 0. 1 – устанавливается после сброса

C1Pole_addi/C2Pole_addi

Адрес: 31/63

Описание: Корректировка номера полюса [11:0]. Значение добавляется к виртуальному счетчику.

Тип доступа: Чтение/Запись (R/W)

IC_addr

Адрес: 64

Описание: Текущий адрес запросов к устройству Для того чтобы микросхема принимала и выдавала значения, необходимо установить BUS_addr = 0 или BUS_addr == IC_addr

Тип доступа: Чтение/Запись (R/W)

ADC_config

Адрес: 65

Описание: Настройки периода работы преобразователей и частоты тактирования АЦП. Период работы преобразователя вычисляется по формуле: Tclk = (16 + DELAY_cycles) × Tclk_adc, где Tclk_adc = 1/Fclk_adc, Fclk_adc = f_adc_src / (FINT_divisor+1) (f_adc_src = FINT при SEL_muxclk=0 либо f_CLK60 при SEL_muxclk=1).

> **ВНИМАНИЕ**
>
> Максимальная частота преобразования Fclk ≤ **150 кВыб/с** (Tclk ≥ 6,7 мкс). При Fclk_adc = 2,5 МГц и DELAY_cycles = 0: Tclk = 6,4 мкс → Fclk ≈ 156 кГц. Частота тактирования АЦП Fclk_adc = f_adc_src/(FINT_divisor+1) не должна превышать 2,5 МГц. Так как FINT_divisor — 4-битное поле (максимум 15), при тактировании АЦП от FINT (SEL_muxclk=0) условие выполнимо только для FINT ≤ 40 МГц; для FINT 50–60 МГц установите SEL_muxclk=1 (тактирование АЦП от входа CLK60) и подберите FINT_divisor под частоту CLK60 (например, 3 для 10 МГц, 7 для 20 МГц).

Тип доступа: Чтение/Запись (R/W)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	12	**SEL_muxclk**	R/W	0	Выбор источника тактирования блока формирования синхрочастоты

№	Биты	Имя поля	Доступ	Сброс	Описание
					для АЦП и синусоидального генератора: 0 - частота цифрового блока 1 - частота с мультиплексора MUX_CLK
2	11..8	**FINT_divisor**	R/W	3	FINT_divisor - коэффициент деления частоты для тактирования АЦП: Fclk_adc = f_adc_src/(FINT_divisor+1) = 1/Tclk_adc, где f_adc_src = FINT при SEL_muxclk=0 либо частота входа CLK60 при SEL_muxclk=1. Fclk_adc не должна превышать 2,5 МГц. FINT_divisor - 4-битное беззнаковое значение в диапазоне [0..15] (максимальный коэффициент деления 16).
3	7..0	**DELAY_cycles**	R/W	0	DELAY_cycles - значение добавочного количества тактов Fclk для составления необходимого периода преобразования. Период преобразования микросхемы Tadc = 16 * Tclk_adc + Delay_cycles*Tclk_adc DELAY_cycles - 8 битное безнаковое значение в диапазоне [0..255]

Mask_Stat

Адрес: 66

Описание: маски для регистров C1Stat и C2Stat

Тип доступа: Чтение/Запись (R/W)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	4	**MSK_CLK_not_RDY**	R/W	1	Маска для бита CLK_not_RDY: 0 – бит CLK_not_RDY устанавливается в состояние лог. «0» 1 – бит CLK_not_RDY отражает действительное состояние флага
2	3	**MSK_SPI_err**	R/W	1	Маска для бита SPI_err (ошибка при передаче по SPI): 0 – бит SPI_err устанавливается в состояние лог. «0» 1 – бит SPI_err отражает действительное состояние флага
3	2	**MSK_Not_equal**	R/W	1	Маска для бита Not_Equal (результаты преобразований не совпадают): 0 – бит Not_Equal устанавливается в состояние лог. «0» 1 – бит Not_Equal отражает действительное состояние флага
4	1	**MSK_nReady2**	R/W	1	Маска для бита nReady2 (преобразователь 2 Готов): 0 – бит nReady2 устанавливается в состояние лог. «0» 1 – бит nReady2 отражает действительное состояние флага
5	0	**MSK_nReady1**	R/W	1	Маска для бита nReady1 (преобразователь 1 Готов): 0 –

№	Биты	Имя поля	Доступ	Сброс	Описание
					бит nReady1 устанавливается в состояние лог. «0» 1 – бит nReady1 отражает действительное состояние флага

Flags_delay

Адрес: 67

Описание: Flags_delay [15:0] выполняет следующие функции: задает время обновления регистров Amp_metric и флагов UIN_HIGH, UIN_LOW (с ревизии 3); устанавливает время удержания флагов. Единица времени 4/Fclk. После пропадания ошибки время удержания флагов (3×65535 – 4×65535) мкс. Желательно устанавливать время удержания флагов больше периода сигнала возбуждения датчика, чтобы избежать постоянного сброса и обратной установки флагов. В то же время, установка слишком большого значения нежелательна, т.к. время обновления флагов увеличивается.

Тип доступа: Чтение/Запись (R/W)

WR_lock

Адрес: 68

Описание: Блокировка записи настроек в микросхему. Если WR_lock == 0, запись разрешена. При WR_lock != 0, команды записи не исполняются

Тип доступа: Чтение/Запись (R/W)

CMP_lth

Адрес: 69

Описание: Максимальное допустимое различие результатов преобразования каналов 1 и 2 для выставления флага Not_Equal в регистре Stat_main.

Тип доступа: Чтение/Запись (R/W)

AFE_config

Адрес: 70

Описание: Регистр настройки аналоговых блоков

Тип доступа: Чтение/Запись (R/W)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	15	**OSC_DIS**	R/W	0	Отключение подачи тактового сигнала на микросхему: 0 – частота подается 1 – частота не подается При установлении бита в состояние лог. «1» тактовая частота внутри микросхемы отсутствует, и работа микросхемы останавливается. При этом возможно осуществлять запись регистров по SPI
2	14..12	**Mode**	R/W	0	Выбор режима работы преобразователя. 000 – каналы преобразователя работают независимо 010 – каналы преобразователя работают параллельно и подключены к входам IOSA1, IOCA1, EXI1, EXO1 011 – каналы преобразователя работают параллельно и подключены к входам IOSA2, IOCA2, EXI2, EXO2 100

№	Биты	Имя поля	Доступ	Сброс	Описание
					– каналы преобразователя работают параллельно в режиме Сельсин и подключены к входам IOSA1, IOCA1, IOSA2, EXI1, EXO1
3	11	**VC_mode**	R/W	0	Режим работы пина VC: 0 – VC функционирует согласно описанию в разделе Управление внешним приёмопередатчиком 1 – режим выдачи разрешения выдачи SDO на приемопередатчик
4	10	**DE_inv**	R/W	0	Полярность разрешения выдачи SDO на приемопередатчик: 0 – выдача разрешена при DE = "1", при DE="0" прием 1 – выдача разрешена при DE = "0", при DE="1" прием
5	9	**DE_half**	R/W	0	Режим выдачи сигнала DE: 0 – для полнодуплексного режима, DE выдается при nSEN=0, SSTR=0 1 – для полудуплексного режима, DE=1 только при запросах в режиме 001
6	8	**Bus0_mode**	R/W	0	Отвечать ли при групповом адресе шины BUS_addr = 0 и наличии адреса (IC_addr !=0): 0 – не отвечать 1 – отвечать
7	7	**HALF_dma**	R/W	0	Переводить приемопередатчик и вывод SDI на спаде 16 такта SCLK текущей транзакции. Этот режим необходим для DMA транзакций в полудуплексом режиме. (Трехпроводный режим) 0 –

№	Биты	Имя поля	Доступ	Сброс	Описание
					открывать транзакцию только при новой транзации SPI 1 – переводить приемопередатчик на последней транзакции
8	6	**DE_turnZ**	R/W	0	Перевод в неактивное состояние DE вывод VC в состояние высокого импеданса (high Z): 0 – выводить жесткий 0 (или 1 при инверсном режиме) - повышенная производительность 1 – для выключения передатчика переводить VC в состояние высокого импеданса (необходима подтяжка резистором)
9	4	**EXO_sync**	R/W	0	Синхронизация таймеров счетчиков в режиме Меандр: 0 - синхронизация выключена 1 - синхронизация включена Синхронизация требуется для управления комплементарными транзисторами накачки возбуждения.
10	3	**SHRD_RAM**	R/W	0	Включение режима передачи массива ячеек памяти данных по адресам [256:384] одного из микровычислителей для расширения памяти данных другого вычислителя: 0 – массивы ячеек памяти данных [256:384] подключены к своим микровычислителям и обращение к ним производится независимо 1 –

№	Биты	Имя поля	Доступ	Сброс	Описание
					массив ячеек памяти данных [256:384] одного из микровычислителей подключается к расширение памяти другого
11	2	**SHRD_CPU2**	R/W	0	Выбор микровычислителя для применения режима расширения памяти данных на 128 ячеек. Выбор активизируется при SHRD_RAM = 1. 0 – расширение производится для CPU1 1 – расширение производится для CPU2
12	1	**VREF_DAC_en**	R/W	0	Включение опорного сигнала для генератора 2.5 В синусоидального возбуждения: 0 – опорный сигнал выключен 1 – опорный сигнал включен
13	0	**VREF_en**	R/W	0	Включение источника опорного напряжения 2.5 В: 0 – источник опорного напряжения выключен 1 – источник опорного напряжения включен

Mode_config

Адрес: 71

Описание: Регистр общей настройки микросхемы

Тип доступа: Чтение/Запись (R/W)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	15	**SPI_CRC_en**	R/W	1	Включение бита четности CRC: 0 – бит четности при передаче по SPI игнорируется 1 – бит четности требуется для записи и чтения из SPI
2	14	**ADC_en**	R/W	0	Включение опроса АЦП: 0 - опрос АЦП выключен 1 - опрос АЦП работает
3	13	**CPU2_en**	R/W	0	Включение CPU2: 0 - CPU2 выключен, находится в состоянии сброса 1 - CPU2 включен
4	12	**CPU1_en**	R/W	0	Включение CPU1: 0 - CPU1 выключен, находится в состоянии сброса 1 - CPU1 включен
5	11	**CONV2_en**	R/W	0	Включение преобразователя 2: 0 – преобразователь 2 отключен и находится в состоянии сброса 1 – преобразователь 2 включен
6	10	**CONV1_en**	R/W	0	Включение преобразователя 1: 0 – преобразователь 1 отключен и находится в состоянии сброса 1 – преобразователь 1 включен
7	9	**EXO1_en**	R/W	0	Включение тактирования формирователя частоты на EXO1: 0 – формирователь EXO1 не тактируется 1 – формирователь EXO1 тактируется

№	Биты	Имя поля	Доступ	Сброс	Описание
8	8	**EXO2_en**	R/W	0	Включение тактирования формирователя частоты на EXO2: 0 – формирователь EXO2 не тактируется 1 – формирователь EXO2 тактируется
9	7..6	**Sample_src**	R/W	3	Выбор строба для выборки данных в SPI. 11 – Выборка производится сигналами SPI в момент чтения регистров 10 – при входе Sample равном лог. «0» данные защелкиваются в регистры 01 – при входе Sample равном лог. «1» данные защелкиваются в регистры 00 – регистры координаты, скорости и состояния сохраняют свое предыдущее состояние
10	4..2	**DB_mode**	R/W	0	Управление источником цифрового сигнала DB. 000 - выход DB подключен к CLKMUX_OUT 001 - выход DB подключен к DB_SIG_PLLBASE 010 - выход DB подключен к DB_SIG_CLKREF 011 - выход DB подключен к CLK 100 - выход DB подключен к STP CPU1 101 - выход DB подключен к STP CPU2 110 - выдача на DB лог. «0» 111 - выдача на DB лог. «1»
11	1	**CPU_CRC4_en**	R/W	0	Включение выдачи CRC4 при чтении ячеек CPU: 0 – Четные адреса: [d1[1:0], d0[13:0]], нечетные адреса [2'b00, d1[13:0]]

№	Биты	Имя поля	Доступ	Сброс	Описание
					1 – Четные адреса: [d1[1:0], d0[13:0]], нечетные адреса [crc4[3:0], d1[13:2]]
12	0	**HAND_CRC4_en**	R/W	0	Включение выдачи CRC4 при чтении выходных тактируемых ячеек HAND1 и HAND2: 0 – Четные адреса: d0[15:0], нечетные адреса [4'b00000, d1[13:0]] 1 – Четные адреса: d0[15:0], нечетные адреса [crc4[3:0], d1[13:0]]

NOCLK_stat

Адрес: 72

Описание: Биты состояния микросхемы

Тип доступа: Только чтение (RO)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	10	**SAMPLE_pin**	RO	1	Логический уровень на входе Sample
2	9	**VC_pin**	RO	1	Логический уровень на входе VC
3	8	**CLKDLY_rdy**	RO	1	Clock delay ready
4	7	**CLKCOMP_rdy**	RO	1	Clock comp ready
5	6	**CLK_rdy**	RO	1	CLK ready

№	Биты	Имя поля	Доступ	Сброс	Описание
6	5	**STNDBY**	RO	0	Pad STNDBY
7	4	**POR_NRST**	RO	1	(POR & NRESET pin) after delay
8	3	**NRST_pll**	RO	1	Сигнал сброса логики уровня PLL
9	2	**NRST_clkcomp**	RO	1	Сигнал сброса логики уровня блока сравнения частот
10	1	**NRST_synchclk**	RO	1	Сигнал сброса логики уровня синхронизатора сброса
11	0	**NRST_rdyinit**	RO	1	Сигнал сброса логики уровня готовности блока инициализации

SPI_req

Адрес: 73

Описание: Предыдущая транзакция SPI

Тип доступа: Только чтение (RO)

---

alive_cnt

Адрес: 74

Описание: alive_cnt [15:0] – счетчик считает во время работы микросхемы. Единица времени 32768/Fclk

Тип доступа: Только чтение (RO)

Stat_main

Адрес: 75

Описание: Регистр состояния микросхемы.

Тип доступа: Только чтение (RO)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	4	**CLK_not_RDY**	RO	1	Флаг готовности тактовой частоты: 0 – синхросигнал не подается на цифровой блок 1 – синхросигнал подается на цифровой блок (прошла задержка)
2	3	**SPI_err**	RO	1	Флаг ошибки при передаче по SPI: 0 – при приеме предыдущего кадра по интерфейсу SPI ошибок не было 1 – при приеме предыдущего кадра по интерфейсу SPI возникла ошибка. Необходимо выполнить сброс микросхемы. Флаг может быть сброшен записью лог. «1»
3	2	**Not_equal**	RO	0	Результаты преобразований не совпадают: 0 – результаты преобразований конвертеров 1 и 2 совпадают или различаются не больше, чем на величину, установленную в регистре CMP_lth 1 – результаты преобразований конвертеров 1 и 2 различаются больше, чем на величину, установленную в регистре CMP_lth

№	Биты	Имя поля	Доступ	Сброс	Описание
4	1	**nReady2**	RO	1	Преобразователь 2 готов: 0 – регистр C2Stat не содержит установленных в «1» битов 1 – регистр C2Stat содержит установленные в «1» биты
5	0	**nReady1**	RO	1	Преобразователь 1 готов: 0 – регистр C1Stat не содержит установленных в «1» битов 1 – регистр C1Stat содержит установленные в «1» биты

Dcpu1LB

Адрес: 76

Описание: Выходная шина с регистров CPU1, младшее слово.

Тип доступа: Только чтение (RO)

Dcpu1HB

Адрес: 77

Описание: Выходная шина с регистров CPU1, старшее слово.

Тип доступа: Только чтение (RO)

Dcpu2LB

Адрес: 78

Описание: Выходная шина с регистров CPU2, младшее слово.

Тип доступа: Только чтение (RO)

Dcpu2HB

Адрес: 79

Описание: Выходная шина с регистров CPU2, старшее слово.

Тип доступа: Только чтение (RO)

PLL_config

Адрес: 80

Описание: Регистр настройки режимов тактирования микросхемы

Тип доступа: Чтение/Запись (R/W)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	15..14	**PLL_basecmp**	R/W	0	Параметр критерия готовности умножителя частоты PLL. Если задано ненулевое значение, то производится подсчет импульсов выходной частоты и сравнение с опорной на 7 битном счетчике: если значения расходятся, то происходит удержание сигнала RESET 00 – признак не влияет на удержание сброса 01 – приведенная к базовой частота PLL отличается не более чем на [-1..1] импульсов 10 – приведенная к базовой частота PLL отличается не более чем на [-2..2] импульсов 11 – приведенная к

№	Биты	Имя поля	Доступ	Сброс	Описание
					базовой частота PLL отличается не более чем на [-4..4] импульсов
2	13..12	**CLK_delay**	R/W	0	Количество импульсов синхрочастоты, при которых удерживается сигнал RESET цифровой части: 0 – сброс не удерживается (запуск сразу после готовности тактовой) 1 – 2048 импульса 2 – 4096 импульса 3 – 8192 импульсов
3	11..8	**PLL_Q**	R/W	0	Коэффициент деления для получения внутренней тактовой частоты. FINT = fOSC × PLL_N/(PLL_Q+1) PLL_Q = 0....15, fOSC – частота на входе TECH2 Примечание: значение fOSC/(PLL_Q+1) должно быть в диапазоне от 2 до 16 МГц
4	7	**PLL_BOOST**	R/W	1	Выдача начального напряжения на PLL для ускорения достижения заданной частоты 0 – без начального напряжения 1 – вместе с начальным напряжением
5	6..0	**PLL_N**	R/W	0	Коэффициент умножения для получения внутренней тактовой частоты. FINT = fOSC × PLL_N/(PLL_Q+1) PLL_N = 2 до 74, fOSC – частота на входе TECH2. Результат умножения (fOSC × PLL_N) не должен превышать 60 МГц. Если PLL_N = 0, блок ФАПЧ отключается и переходит

№	Биты	Имя поля	Доступ	Сброс	Описание
					в низко потребляющий режим. Тактирование производится от внешнего тактового генератора

INIT_conf

Адрес: 81

Описание: Регистр управления режимом начальной конфигурации микросхемы

Тип доступа: Чтение/Запись (R/W)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	5	**LVL_pin**	R/W	0	Выключение проставок в цифровых портах: 0 - проставки включены 1 - проставки закорочены
2	4	**OTP_init_on**	R/W	0	Инициализация из ВОТР памяти: 0 - инициализация выключена 1 - инициализация включена. Внимание! Прожигать бит со значением 1 в ПЗУ допускается только после полного прожига и проверки ВОТР — при пустой ВОТР автозагрузка инициализирует все регистры нулями, и микросхема не работает после включения питания (см. ПЗУ → Порядок программирования ПЗУ)
3	3..0	**BOTP_clkdel**	R/W	4	Делитель частоты для работы ПЗУ (ВОТР). Fclk_rom = FINT / N, где N

№	Биты	Имя поля	Доступ	Сброс	Описание
					определяется значением BOTP_clkdel по таблице (см. ниже). Fclk_rom не должна превышать **1 МГц**.

BOTP_clkdel	Делитель N	Fclk_rom при FINT = 10 МГц	Fclk_rom при FINT = 60 МГц
0	4	2,50 МГц	15,00 МГц
1	6	1,67 МГц	10,00 МГц
2	8	1,25 МГц	7,50 МГц
3	12	833 кГц	5,00 МГц
4	16	625 кГц	3,75 МГц
5	20	500 кГц	3,00 МГц
6	24	417 кГц	2,50 МГц
7	32	313 кГц	1,88 МГц
8	40	250 кГц	1,50 МГц
9	48	208 кГц	1,25 МГц
10	56	179 кГц	1,07 МГц
11	64	156 кГц	938 кГц
12	80	125 кГц	750 кГц

BOTP_clkdel	Делитель N	Fclk_rom при FINT = 10 МГц	Fclk_rom при FINT = 60 МГц
13	96	104 кГц	625 кГц
14	128	78 кГц	469 кГц
15	160	63 кГц	375 кГц

Значения, при которых Fclk_rom превышает 1 МГц, использовать нельзя: при FINT = 10 МГц допустимы BOTP_clkdel ≥ 3, при FINT = 60 МГц — BOTP_clkdel ≥ 11 (N = 64, Fclk_rom ≈ 938 кГц).

UOTP_ctrl

Адрес: 82

Описание: Регистр управления записи и чтения пользовательской памяти прямого доступа (UOTP) , отвечает за регистры PLL_CONFIG, INIT_conf

Тип доступа: Чтение/Запись (R/W)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	3	**MANUAL_NRST_PLL**	R/W	1	Переключить выход пользовательской ячейки SPI на реальное значение OTP памяти

№	Биты	Имя поля	Доступ	Сброс	Описание
2	2	**WATCH_ROM_UVAL**	R/W	0	Переключить выход пользовательской ячейки SPI на реальное значение OTP памяти
3	1	**PROG_NEW_UVAL**	R/W	0	Разрешить запись значения из ячейки SPI в память OTP
4	0	**OVERRIDE_UVAL**	R/W	0	Переопределить текущее значение пользовательской настройки значением из SPI

BUS_addr

Адрес: 83

Описание: Текущий адрес запросов к устройству Если задан IC_addr, чтобы микросхема принимала и выдавала значения необходимо установить BUS_addr = 0 или BUS_addr == IC_addr

Тип доступа: Чтение/Запись (R/W)

BOTP_addr

Адрес: 84

Описание: Регистр адреса блока OTP памяти 512x16 бит (BOTP)

Тип доступа: Чтение/Запись (R/W)

BOTP_data

Адрес: 85

Описание: Регистр данных блока OTP памяти 512x16 бит (BOTP)

Тип доступа: Чтение/Запись (R/W)

BOTP_ctrl

Адрес: 86

Описание: Регистр управления записи и чтения блока OTP памяти 512x16 бит (BOTP)

Тип доступа: Чтение/Запись (R/W)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	3	**PGM**	R/W	0	PGM - бит разрешения записи: 0 - запись не производится 1 - запись по адресу, записанному в регистре BOTP_addr и данного, записанного в регистр BOTP_data
2	2	**REN**	R/W	0	REN - режим чтения: 0 - чтение не производится 1 - чтение по адресу, записанному в регистре BOTP_addr
3	1	**NCEN**	R/W	1	NCEN - разрешение транзакции для BOTP: 0 - BOTP принимает команды, адреса и данные 1 - BOTP игнорирует
4	0	**SLEEP**	R/W	1	Сигнал перехода блока BOTP в режим пониженного энергопотребления: 0 - BOTP

№	Биты	Имя поля	Доступ	Сброс	Описание
					включен, готов к чтению/записи 1 - BOTP выключен, недоступен

BOTP_out

Адрес: 87

Описание: Выход блока OTP памяти 512x16 бит (BOTP)

Тип доступа: Только чтение (RO)

P1BG_ctrl/P2BG_ctrl

Адрес: 92/94

Описание: Регистр управления отладкой CPU

Тип доступа: Чтение/Запись (R/W)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	12..11	**command**	R/W	0	00 - запустить CPU до точки останова, если она активна 01 - остановить CPU как есть 10 - выполнить текущую операцию
2	10	**tap_event**	R/W	0	Переход из 0 в 1 активизирует одну из команд (00, 01, 10)

№	Биты	Имя поля	Доступ	Сброс	Описание
3	9	**stop1_ena**	R/W	0	Включение режима отслеживания точки останова 1
4	8..0	**stop1_pc**	R/W	0	Адрес точки останова 1

P1BG_data/P2BG_data

Адрес: 93/95

Описание: Данные управления отладкой CPU

Тип доступа: Чтение/Запись (R/W)

Поля регистра

№	Биты	Имя поля	Доступ	Сброс	Описание
1	13..10	**sel_reg**	R/W	0	Выбор регистра из CPU для чтения
2	9	**stop2_ena**	R/W	0	Включение режима отслеживания точки останова 2
3	8..0	**stop2_pc**	R/W	0	Адрес точки останова 2

---

*Последнее обновление **21 сент. 2026 г.***

Методика подключения и настройки

Настройка должна производиться в порядке, заданном нумерацией разделов.

1. Проверка подключения

a. С помощью осциллографа убедитесь в наличии напряжения питания на выводах VDDA(63) = 5,0 В и VDDD(33) = 5,0 В, а также на выводах VDDIO(45) = 3,3 В, VDDIO2(25) = 3,3 В, VPP(35) = 3,3 В. При использовании генератора опорной частоты убедитесь в наличии напряжения питания на выводах VDDEXO1(5) = 5,0 В и/или VDDEXO2(4) = 5,0 В.

b. Убедитесь в наличии напряжения на выходах встроенных линейных регуляторов: VDDD1P8(36, 61) ≈ 1,8 В, VDDD_INT(31) ≈ 4,0 В, VDDA_INT(30) ≈ 4,0 В.

c. Проверьте подключение цепей тактирования:

- При тактировании с использованием кварцевого резонатора (подключённого к TP_QI(28), TP_QO(27), выход TP_FQO(26) подключён к CLK60(39)) убедитесь в наличии тактовой частоты на выходе генератора TP_FQO(26). Если частота на выходе TP_FQO отсутствует — проверьте наличие резистора 510 кОм, подключённого параллельно кварцевому резонатору (между выводами TP_QI(28) и TP_QO(27)). Отсутствие резистора может приводить к нестабильному запуску генератора или отсутствию генерации. Также проверьте подключение служебных выводов: TECH1(37) должен быть подключён к GND, TECH2(38) — к VDDD_INT(31).

- При тактировании от внешнего активного генератора (подключённого к CLK60(39)): вывод TP_QI(28) должен быть подключён к VDDD_INT, вывод TP_QO(27) — к GND.

- Убедитесь в наличии тактового сигнала на технологическом выводе DB(41) — по умолчанию на нём присутствует сигнал CLKMUX_OUT, поделённый на 8 (поле DB_mode регистра Mode_config = 000 после сброса). Если на выводе DB тактовый сигнал отсутствует — проверьте подключение служебных выводов TECH1(37) и TECH2(38).

2. Обнаружение микросхемы и проверка SPI

а. Убедитесь в работоспособности обмена по SPI: прочитайте регистры с известными значениями сброса и сравните с ожидаемыми:

Адрес	Регистр	Ожидаемое значение (после сброса)
0	C1KampS	`0x0400`
1	C1KampC	`0x0400`
8	C1Amp_th	`0xBB3E`
9	C1InputStngs	`0x0810`

Если считанные значения совпадают с указанными — микросхема обнаружена, обмен по SPI функционирует корректно, переходите к разделу 3. Если значения не читаются или искажены — см. раздел 12.

Адресация нескольких микросхем на общей шине (регистры IC_addr, BUS_addr) и блокировка записи (WR_lock) — опции: адресация описана в разделе 12, блокировка записи — в разделе 14.

3. Настройка тактовой частоты микросхемы

ОГРАНИЧЕНИЕ ЧАСТОТЫ ПРЕОБРАЗОВАНИЯ В ТЕКУЩЕЙ ВЕРСИИ МИКРОСХЕМЫ

Практическое значение частоты тактирования АЦП в существующей версии микросхемы несколько снижено по сравнению с номинальным. Пользователь должен обеспечить частоту тактирования АЦП **не превышающую 2,5 МГц** (Fclk_adc ≤ 2,5 МГц). Для корректной работы преобразователей HAND1 и HAND2, обрабатывающих сэмплы АЦП, частота FINT должна быть минимум в 2 раза больше частоты АЦП: **FINT ≥ 2 × Fclk_adc**.

Для АЦП требуется небольшая частота (Fclk_adc ≤ 2,5 МГц), а для максимизации производительности микровычислителей FINT должна составлять 50–60 МГц. Получить 2,5 МГц непосредственно из FINT 50–60 МГц нельзя — поле FINT_divisor 4-битное (максимальный коэффициент деления 16). Поэтому рекомендуется подать на вход CLK60 опорную частоту 10–30 МГц и использовать её двояко: **для АЦП — поделить** (бит SEL_muxclk = 1, делитель FINT_divisor работает непосредственно от CLK60), **для микровычислителей — умножить** до 50–60 МГц полем PLL_N. Подробно — см. Тактирование.

Пошаговая настройка (при использовании математики на CPU):

1. Подайте на вход CLK60 частоту от внешнего активного генератора (не более 30 МГц) или от встроенного генератора на кварцевом резонаторе (до 20 МГц по его возможностям, см. Тактирование).
2. Умножьте входную частоту до 50–60 МГц с помощью поля PLL_N. Минимальный коэффициент умножения PLL_N = 2. **Следите, чтобы результат умножения не превышал 60 МГц** (контроль частоты FINT — установите DB_mode = `011` и измерьте частоту на выводе DB(41): она должна быть равна FINT/8).
3. Тактируйте АЦП от входа CLK60, а не от умноженной FINT: установите SEL_muxclk = 1. Тогда делитель работает от исходной (низкой) частоты CLK60, и поле FINT_divisor подбирается под неё: Fclk_adc = f_CLK60 / (FINT_divisor + 1) ≤ 2,5 МГц (например, FINT_divisor = 3 для 10 МГц, 7 для 20 МГц, 11 для 30 МГц). Делить саму FINT (50–60 МГц) до 2,5 МГц нельзя — для этого потребовался бы коэффициент 24, что превышает 4-битный диапазон поля FINT_divisor (максимум 16).

> **СТРУКТУРА ФАПЧ**
>
> Блок ФАПЧ (PLL) в микросхеме разделён на две независимые части: **сначала идёт умножитель частоты** (коэффициент PLL_N), **затем делитель** (коэффициент PLL_Q + 1). Итоговая частота: FINT = f_CLK60 × PLL_N / (PLL_Q + 1). Если PLL не используется (PLL_N = 0), тактирование производится напрямую от входа CLK60.

:::

а. Запишите в регистр PLL_config (адрес 80) рассчитанные значения PLL_N[6:0], PLL_Q[11:8]. Чтобы записанные значения переопределили значения, загруженные из ПЗУ,

установите бит OVERRIDE_UVAL = 1 в регистре UOTP_ctrl (адрес 82, бит 0). Для контроля готовности PLL установите PLL_basecmp[15:14] ≠ 00. Если PLL не используется, запишите PLL_N = 0 — тактирование будет производиться напрямую от входа CLK60.

b. Запишите в регистр ADC_config (адрес 65) значения FINT_divisor[11:8] и DELAY_cycles[7:0]. Частота тактирования АЦП Fclk_adc = f_adc_src / (FINT_divisor + 1) (где f_adc_src = FINT при SEL_muxclk=0 или f_CLK60 при SEL_muxclk=1) **не должна превышать 2,5 МГц**. Период работы преобразователя Tclk = (16 + DELAY_cycles) × Tclk_adc.

c. Убедитесь в готовности тактовой частоты: считайте регистр NOCLK_stat (адрес 72) — биты CLK_rdy[6] и CLKCOMP_rdy[7] должны быть равны 1. Также убедитесь, что AFE_config.OSC_DIS[15] = 0. Для проверки итоговой частоты FINT установите DB_mode = `011` (поле регистра Mode_config, биты [4:2]) и измерьте частоту на выводе DB(41) осциллографом — она должна быть равна FINT / 8.

> **ПРИМЕЧАНИЕ**
>
> При некорректной настройке тактовой частоты может неправильно работать чтение регистров SPI. В этом случае произведите внешний сброс микросхемы (вывод NRESET), либо запишите корректные значения в регистры PLL_config, AFE_config (запись в эти регистры функционирует при отсутствующей или слишком большой тактовой частоте).

4. Настройка опорных уровней аналоговой части

a. Выберите источник опорного напряжения АЦП (вывод VREF2P5(11)):

- **Встроенный ИОН (2,5 В):** запишите AFE_config.VREF_en (адрес 70, бит 0) = **1** — при этом на выводе VREF2P5 появится напряжение 2,5 В от встроенного источника.
- **Внешний источник:** подайте внешнее опорное напряжение на вывод VREF2P5, VREF_en должен быть равен **0**.

b. При использовании встроенного синусоидального генератора возбуждения запишите AFE_config.VREF_DAC_en (бит 1) = **1**. При этом внутренний буфер повторяет напряжение с вывода VREF2P5(11) и подаёт его на вывод REFDAC(64). Вывод REFDAC необходимо

зашунтировать конденсатором 0,1 мкФ. Буфер REFDAC питается от вывода VDDEXO1(5) — без питания на VDDEXO1 напряжение на REFDAC(64) отсутствует. Осциллографом убедитесь в наличии напряжения: VREF2P5(11) ≈ 2,5 В и REFDAC(64) ≈ 2,5 В.

5. Настройка сигнала возбуждения датчика (на примере канала 1)

a. При использовании внешнего сигнала возбуждения: C1ExoStngs.EXO_mode (адрес 6, биты [11:10]) = 00 (генератор выключен), C1InputStngs.Ex_source (адрес 9, биты [10:9]) = 01 (внешний сигнал с EXI1). На этом настройка возбуждения окончена, переходите к разделу 6.

b. При использовании встроенного генератора в режиме синусоидального сигнала: запишите C1ExoStngs.EXO_mode (адрес 6, биты [11:10]) = 10, Amp_code (биты [7:0]) — начальное значение амплитуды. Включите тактирование генератора: Mode_config.EXO1_en (адрес 71, бит 9) = 1. Генератор возбуждения работает только при включённом АЦП — установите ADC_en (бит 14 регистра Mode_config) = 1: этот бит разрешает тактирование АЦП и ЦАП, без него формирование сигнала невозможно.

c. Запишите в регистр C1EXInc (адрес 7) значение, соответствующее выбранной частоте возбуждения датчика (см. формулу (5) в разделе Программируемый генератор). Частота рассчитывается по формуле: f_ex = EXInc / 2^20 × Fclk.

d. Используя осциллограф, подключённый к выводу EXO1(6), убедитесь в наличии и параметрах сигнала (амплитуда, частота, отсутствие нелинейных искажений). Амплитуда сигнала должна быть ниже напряжения питания VDDEXO1 как минимум на 0,4 В. При необходимости скорректируйте Amp_code или C1EXInc.

e. Подключите датчик. Осциллографом убедитесь в параметрах сигнала возбуждения на входе датчика (амплитуда, искажения) и сигналов вторичных обмоток, подключённых к микросхеме. Убедитесь в отсутствии насыщения сердечника датчика. При необходимости скорректируйте схему усилителя-фильтра на плате.

6. Настройка входных каскадов и проверка оцифровки сигналов АЦП (на примере канала 1)

В этом разделе настраивается аналоговый входной тракт (внешние или встроенные усилители, тип и уровни сигнала) и проверяется **сам факт оцифровки** — что сигналы датчика действительно преобразуются АЦП в цифровые отсчёты. Проверка выполняется прямым чтением отсчётов АЦП по SPI, без микровычислителей и без специального режима.

6.а Входные усилители: внешние или встроенные

a. Выберите способ подачи сигналов датчика на АЦП — через встроенные операционные усилители или напрямую/через внешний усилитель. За это отвечает бит C1InputStngs.OPA_en (адрес 9, бит 8):

- **OPA_en = 1** — включены встроенные операционные усилители на входах сигналов датчика; сигнал на АЦП поступает с этих усилителей.
- **OPA_en = 0** — встроенные усилители отключены; сигнал подаётся на АЦП непосредственно через выводы IOSA1(24), IOCA1(21) (например, при наличии внешнего усилителя-фильтра на плате).

b. Для модулированных сигналов датчика (СКВТ, сельсин, ЛРДТ) установите DC_carrier (бит 5) = 0, DC_correction (бит 4) = 1 — включена компенсация среднего уровня. Для немодулированных сигналов (DC_carrier = 1) установите DC_correction = 0.

6.б Параметры аналогового сигнала на входах АЦП

a. Вращая/двигая датчик, оцените параметры сигнала на входах АЦП IOSA1(24), IOCA1(21):

- максимальная амплитуда сигналов **2 В ±10 %** (при VREF2P5 = 2,5 В);
- средний уровень (синфазное напряжение) **1,25 В ±10 %** (при VREF2P5 = 2,5 В);
- отсутствие значительных нелинейных искажений.

b. При подключении датчиков типа ЛРДТ с последовательно соединёнными обмотками (Sensor_mode = 10) дополнительно убедитесь в совпадении фаз сигналов IOSA1, IOCA1;

при необходимости скорректируйте сдвиг фаз RC-цепью в одном из каналов.

6.в Проверка оцифровки: чтение отсчётов АЦП по SPI

a. Запустите преобразование и убедитесь, что настроено тактирование АЦП (раздел 3). Необходимые биты:

- ADC_en[14] = 1 — опрос АЦП;
- CONV1_en[10] = 1 — преобразователь 1 (для канала 2 — CONV2_en[11] = 1).

Микровычислители для получения отсчётов не требуются.

b. Оцифрованные сигналы канала 1 считываются из регистров:

Регистр	Адрес	Содержимое
C1AdcS	18	опорный сигнал (Ex_ref, Ex_shifted) и оцифрованный sin (Dadc_sin, ст. бит msb_arg_sin)
C1AdcC	19	оцифрованный cos (Dadc_cos, ст. бит msb_arg_cos; при включённом контроле — CRC4)
C2AdcS	50	то же, канал 2
C2AdcC	51	то же, канал 2

Для получения пары (sin, cos), относящейся к одному моменту, считывайте подряд регистры 18 и 19 (для канала 2 — 50 и 51): при чтении регистра 18 происходит защёлка данных, и следующее чтение регистра 19 относится к тому же отсчёту.

c. **Подтверждение факта оцифровки.** Перемещая датчик, убедитесь, что значения Dadc_sin/Dadc_cos изменяются вслед за аналоговым сигналом. Дополнительно переключите OPA_en (1 ↔ 0) — цифровые отсчёты должны реагировать на это переключение, что подтверждает прохождение сигнала через выбранный входной каскад.

> **ОПЦИОНАЛЬНЫЕ СПОСОБЫ ПРОСМОТРА**

> Для непрерывного наблюдения и захвата осциллограмм предусмотрены: **буферизованный захват** (запись блока отсчётов во внутреннюю память с последующим чтением по SPI) и **параллельный вывод** (непрерывная выдача потока отсчётов через четырёхбитную шину на выводах эмулятора энкодера ENC1_0/A/B, ENC2_0/A/B, тактируемая от SCLK).

d. **Что проверять, если сигнала нет или он неверный:**

- Сигнала нет вообще — проверьте включение АЦП и преобразователя (`ADC_en`[14] = 1, `CONV1_en`[10] = 1), наличие возбуждения (раздел 5) и включение входных усилителей (`OPA_en`[8]).
- Размах оцифрованного sin/cos меньше или больше ожидаемого — скорректируйте коэффициенты KampS/KampC (раздел 8).
- Сигнал есть, но координата нестабильна — проверьте фазу опорного сигнала (раздел 7) и DC_correction.

7. Выбор режима преобразования и настройка опорного сигнала

В этом разделе выбирается режим преобразования (СКВТ или сельсин), источник опорного сигнала возбуждения (внешний или внутренний) и выполняется фазирование опоры относительно оцифрованного сигнала; после этого подключается блок восстановления опорной частоты.

7.a Выбор режима преобразования (Mode)

Запишите в поле Mode регистра AFE_config (адрес 70, биты [14:12]) требуемый режим работы преобразователей:

- 000 — каналы преобразователя работают независимо;
- 010 — каналы работают параллельно, подключены к входам IOSA1, IOCA1, EXI1, EXO1;
- 011 — каналы работают параллельно, подключены к входам IOSA2, IOCA2, EXI2, EXO2;
- 100 — режим **сельсин**: оба канала параллельно с преобразованием 3 фаз в 2 (в декартову систему координат, аналогичную СКВТ).

Соответствие входов режиму — см. таблицу 5 в разделе Блок масштабирования.

7.б Источник опорного сигнала

Опорный сигнал Ex_ref, используемый для демодуляции, берётся из того же источника, что и возбуждение (раздел 5), и выбирается битами Ex_source регистра C1InputStngs (адрес 9, биты [10:9]):

- **внутренний** (Ex_source = 0, при встроенном генераторе возбуждения) — опорный сигнал формируется внутри микросхемы и выводится на EXI1;
- **внешний** (Ex_source = 01) — опора подаётся на вход EXI1(50) через компаратор (см. рекомендуемую схему); сигнал на EXI1 — цифровой с уровнями 0 и 1.

При выборе внутреннего источника (Ex_source ≠1) вывод EXI1 становится выходом опорного сигнала — это позволяет работать нескольким микросхемам от одной опоры.

7.в Фазирование опорного сигнала по цифровым отсчётам АЦП

a. Считывайте показания АЦП **вместе с опорным сигналом**. Регистр C1AdcS (адрес 18) в одном слове содержит и оцифрованный сигнал (Dadc_sin и его старший бит msb_arg_sin; в регистре C1AdcC — msb_arg_cos), и опорные сигналы Ex_ref (бит 0) и Ex_shifted (бит 14) — тем самым опора и выборка привязаны к одному моменту времени.

b. Регистром C1ExPhShft (адрес 5) сдвигайте опору так, чтобы сдвинутый опорный сигнал Ex_shifted стал **почти синхронен со старшим битом сигнала АЦП** — msb_arg_sin/msb_arg_cos: переходы Ex_shifted должны совпадать с переходами старшего бита (сменой знака) оцифрованного сигнала. Наблюдайте и совмещайте именно эти сигналы — Ex_shifted, msb_arg_sin, msb_arg_cos.

c. Требования к точности совпадения фаз опоры и сигнала зависят от режима демодуляции:

- При Ex_recovery_en = 1 (рекомендуется) достаточно **приблизительного** совпадения фаз (в пределах ±45°); температурная зависимость отсутствует, т.к. демодуляция производится сигналом, выделенным из АЦП. Для датчиков типа ЛРДТ при Ex_recovery_en = 1 вход EXI1 можно подтянуть к лог. 0 или 1.

- При Ex_recovery_en = 0 (не рекомендуется) требуется **точное** совпадение фаз; сдвиг фазы в датчике зависит от температуры, несовпадение ведёт к росту шума и падению точности.

7.г Блок восстановления опорной частоты и цифровая настройка фазы

a. После грубого совмещения фаз (7.в) — а можно и сразу, работе он не мешает — включите блок восстановления опорной частоты: C1InputStngs.Ex_recovery_en (бит 11) = 1. Блок сам привязывает опору к моментам перехода через нуль большего из входных сигналов (IOSA1/IOCA1) и формирует восстановленный сигнал (подробнее — Блок восстановления сигнала опорной частоты). До захвата фазы (в течение первого периода входного сигнала) в регистре C1Stat удерживается флаг RCV_NotRDY = 1, и следящий контур удерживается в сбросе.

b. **Цифровая проверка фазы.** Восстановленную опору и знак входного сигнала считывайте в одном цикле: бит Ex_recovered_vs (бит 14 регистра C1OutS, адрес 20) — рядом со знаком входного сигнала: старшим битом msb_arg_sin/msb_arg_cos регистра C1AdcS либо знаком arg_sin_kontur1. При правильном захвате переходы Ex_recovered_vs совпадают с переходами входного сигнала через нуль (сменой знака) — восстановленная опора синхронна со знаком сигнала АЦП.

c. **Подстройка фазы регистром** C1ExPhShft**.** Блок сравнивает исходную опору с восстановленным сигналом; если их фазы расходятся более чем на ±40°, в C1Stat выставляется флаг EX_PH_OUTRANGE. Подбирайте C1ExPhShft (адрес 5) так, чтобы EX_PH_OUTRANGE = 0 и RCV_NotRDY = 0 — это и есть цифровое подтверждение правильной фазы опоры. Если опорный сигнал отсутствует, выставляется флаг MISS_EXREF — проверьте источник опоры (раздел 7.б).

> ⚠ **ВНИМАНИЕ**
>
> При неправильной настройке блока восстановления опорного сигнала (в частности, если EX_PH_OUTRANGE удерживается) преобразователь может давать ошибку 180°, при этом флаги ошибок будут сброшены.

8. Калибровка коэффициента усиления входных сигналов и смещения сигналов модели датчика

Точность следящего контура определяется тем, насколько входные сигналы (с АЦП) совпадают с виртуальными сигналами математической модели датчика. Калибровка выполняется в **два этапа** и строго в указанном порядке:

1. **Настройка амплитуды (размаха)** — коэффициентами KampS/KampC размах входных сигналов приводится к размаху модели (4096, т.е. ±2048). На этом этапе **смещение среднего игнорируется**.
2. **Настройка смещения** — коэффициентами KbiasS/KbiasC смещение сигналов модели подгоняется под смещение входных сигналов.

После обоих этапов входной и модельный сигналы должны практически совпадать: их нулевая (постоянная составляющая) и первая (основная) гармоники совпадают.

> ⓘ **ССЫЛКИ ПО ТЕМЕ**
>
> - Блок масштабирования и преобразования координат — раздел Блок масштабирования.
> - Формулы расчёта виртуальных сигналов — раздел Блок модели датчика.
> - Практическая настройка в графическом режиме GraphView — раздел Режим детектора.

8.a Контрольные сигналы

Все значения доступны по SPI без специального режима. Для канала 1 (для канала 2 адреса — в скобках):

Сигнал	Регистр (адрес)	Биты	Назначение
arg_sin_kontur1	C1OutS (20 / 52)	13..1	Входной SIN **после** коррекции усиления и смещения — входной аргумент контура
arg_cos_kontur1	C1OutC (28 / 60)	13..1	Входной COS после коррекции — входной аргумент контура
VirtualSin12_1	C1VirtualS (21 / 53)	11..0	Виртуальный SIN (сигнал модели)
VirtualCos12_1	C1VirtualC (29 / 61)	11..0	Виртуальный COS (сигнал модели)
Ex_recovered90dgr_vs	C1OutS (20 / 52)	0	Восстановленная опора, сдвинутая на 90° — метка пика несущей

Сигналы `arg_*` — 13-битные знаковые (диапазон ±4095), что оставляет запас под смещение при целевом размахе ±2048 без усечки. Сигналы модели `Virtual*` всегда имеют размах ±2048 (4096).

8.б Настройка амплитуды (размаха): KampS, KampC

a. Коэффициент усиления каждого канала равен значению регистра, делённому на 1024:

$$k = \frac{\mathrm{KampS}}{1024}, \qquad k = \frac{\mathrm{KampC}}{1024}$$

При значении по сбросу KampS = `0x0400` (1024) усиление k = 1,0. (См. также Блок масштабирования.)

b. Целевой размах входных сигналов — **4096** (±2048), т.е. размах модели. Это обеспечит совпадение первой гармоники входного сигнала и модели.

c. Для **переменных (модулированных)** сигналов (СКВТ, сельсин, ЛРДТ) размах определяется по огибающей за полный оборот/ход датчика. Найдите два положения

вала, разнесённые на 180°, в которых огибающая максимальна: в одном сигнал достигает положительного максимума (+A), в противоположном — отрицательного (–A). Сумма этих двух амплитуд и есть размах: $R = A + |-A| = 2A$ (эквивалентно $R = \max - \min$ за оборот). Такой способ измерения **не зависит от смещения среднего**, поэтому на этапе настройки амплитуды смещение не учитывается.

d. Измерьте размах R_S и R_C для каналов SIN и COS (по arg_sin_kontur1 и arg_cos_kontur1) при текущем значении коэффициентов. Если измеренный размах меньше 4096, пересчитайте коэффициенты так, чтобы после умножения сумма (размах) стала равна 4096:

$$\mathrm{KampS}_{\text{нов}} = \mathrm{KampS}_{\text{тек}} \cdot \frac{4096}{R_S}, \qquad \mathrm{KampC}_{\text{нов}} = \mathrm{KampC}_{\text{тек}} \cdot \frac{4096}{R_C}$$

При исходном значении по сбросу (1024): $\mathrm{KampS}_{\text{нов}} = \mathrm{round}(1024 \cdot 4096 / R_S)$. Запишите новые значения в регистры KampS (адрес 0 / 32) и KampC (адрес 1 / 33).

e. **Контроль результата умножения** выполняйте по 13-битным значениям arg_sin_kontur1 / arg_cos_kontur1: после записи их размах должен составлять ≈ 4096 (±2048). Следите за флагами переполнения и пороговыми компараторами (см. описание регистров KampS/KampC).

f. Удобный способ снять отсчёт огибающей — выбрать момент **фронта или спада** сигнала Ex_recovered90dgr_vs (бит 0 регистра C1OutS). Если блок восстановления опорной частоты (Ex_recovery_en = 1) включён и синхронизирован, этот фронт совпадает с пиком несущей, где мгновенное значение модулированного сигнала максимально. Считывая `arg_sin_kontur1`/`arg_cos_kontur1` в этот момент, вы получаете значение огибающей напрямую — это и есть значение +A (при одном положении вала) и –A (при положении, повёрнутом на 180°).

> ⚠ **УСЛОВИЕ КОРРЕКТНОСТИ ВЫБОРКИ ПО ФРОНТУ**
>
> Метод по фронту `Ex_recovered90dgr_vs` корректен **только при работающем блоке восстановления опорной частоты**. Если блок выключен (Ex_recovery_en = 0), фронт не привязан к пику несущей — используйте непосредственное измерение размаха (max – min) за оборот. Подробнее — см. раздел Блок восстановления сигнала опорной частоты.

g. Графический аналог этой процедуры — Режим детектора в GraphView: там выборка входных и виртуальных сигналов уже привязана к фронту опорного сигнала, а амплитуда оценивается совмещением графиков.

> **НЕМОДУЛИРОВАННЫЕ СИГНАЛЫ**
>
> Для сигналов без модуляции (DC_carrier = 1) огибающей несущей нет — размах измеряется непосредственно как разность максимума и минимума сигнала за ход датчика, метод по фронту `Ex_recovered90dgr_vs` не применяется.

8.в Настройка смещения сигналов модели: KbiasS, KbiasC

a. После согласования размаха (раздел 8.б) входные сигналы могут быть смещены относительно модели по постоянной составляющей: графики совпадают по амплитуде, но сдвинуты друг относительно друга по вертикали.

b. Сигналы модели (VirtualSin12_1 / VirtualCos12_1) всегда имеют размах 4096 (±2048). Коэффициенты KbiasS / KbiasC добавляют к модели смещение (см. формулы (9)–(10) в разделе Блок модели датчика), поднимая или опуская её так, чтобы постоянная составляющая модели совпала со смещением входных сигналов.

c. Для анализа считывайте одновременно входные сигналы (arg_sin_kontur1 / arg_cos_kontur1) и модельные (VirtualSin12_1 / VirtualCos12_1) и сравнивайте их средние уровни. Подбирайте KbiasS (адрес 2 / 34) и KbiasC (адрес 3 / 35) — это знаковые значения в дополнительном коде (±32767) — до совпадения постоянных составляющих.

d. **Критерий окончания настройки:** входной и модельный сигналы должны практически совпадать — совпадают нулевая (постоянная составляющая) и первая (основная) гармоники. Остаточная разница в высших гармониках отражает нелинейность самого датчика и аналогового тракта платы и данной калибровкой не устраняется.

8.г Порядок настройки

1. Убедитесь, что АЦП выдаёт осмысленные отсчёты (раздел 6), а фаза опорного сигнала настроена (раздел 7).

2. **Амплитуда:** измерьте размах R_S, R_C сигналов `arg_sin_kontur1` / `arg_cos_kontur1` за полный оборот (методом двух положений, разнесённых на 180°, либо по фронту `Ex_recovered90dgr_vs`); рассчитайте и запишите KampS / KampC по формуле раздела 8.б так, чтобы размах стал ≈ 4096 (±2048).
3. **Смещение:** сравните средние уровни входных (`arg_*`) и модельных (`Virtual*`) сигналов; подберите KbiasS / KbiasC до совпадения постоянных составляющих.
4. Проверьте совпадение сигналов по нулевой и первой гармоникам. При необходимости используйте Режим детектора для визуального контроля.

12. Почему микросхема не отвечает (проверка на уровне SPI)

Если микросхема не отвечает на команды SPI (значения регистров не читаются или искажены), выполните проверки физического уровня интерфейса SPI.

a. Осциллографом проверьте уровни на служебных выводах: **nSEN** должен быть подключён к GND (при работе с одной микросхемой на шине); вывод **STNDBY** должен находиться в лог. «0» (тактирование цифрового блока не выключено).

b. Проверьте кадровую структуру. При **SSTR = 0** должны проходить **16 полных фронтов SCLK** (каждый импульс имеет и передний, и задний фронт). Сигнал **SSTR должен быть в 1 до первого фронта SCLK и снова подниматься в 1 после последнего фронта** транзакции: переход SSTR $1 \to 0$ начинает транзакцию, $0 \to 1$ завершает её и сбрасывает счётчики.

c. Проверьте переключение выхода **SDO**. При опускании SSTR из 1 в 0 вывод SDO должен переходить из состояния высокого импеданса (Z) в активный логический уровень (0 или 1). Если SDO постоянно находится в 0, временно подтяните его к 1 через резистор 10 кОм: если после этого на SDO появится реальный сигнал — микросхема отвечает, причина была в отсутствии подтяжки/чтении; если SDO останется в 0 — микросхема не активирует выход (проверьте пункты a, b, e).

d. Убедитесь в правильности посылки на **SDI**. Командное слово (16 бит, передаётся младшим/старшим разрядом согласно Интерфейс SPI) имеет формат:

`M₂, M₁, M₀, A10, A9, A8, A7, A6, A5, A4, A3, A2, A1, A0, 0, P`

где **$M_2M_1M_0$** — код операции (`110` — полнодуплексное чтение, `001` — полудуплексное чтение, `100` — запись), **A10…A0** — адрес регистра, **P** — бит чётности. Микросхема защёлкивает SDI по переднему фронту SCLK и выдаёт SDO по заднем фронту SCLK.

Пример последовательности чтения регистра 8 (C1Amp_th, ожидается `0xBB3E`):

1. SSTR = 1.
2. SSTR → 0 (начало транзакции).
3. Передать на SDI командное слово: код `110` + адрес `8` (`0b00000001000`) + `0` + бит чётности P, в течение кадра из 16 фронтов SCLK.
4. SSTR → 1 (конец транзакции, сброс счётчиков).
5. Следующим кадром (SSTR 1→0, 16 фронтов SCLK) считать с SDO значение `0xBB3E`.

Если бит чётности P ошибочен, микросхема игнорирует команду; на время отладки контроль чётности можно отключить (SPI_CRC_en = 0). Полное описание кодов операций и временных диаграмм — см. Интерфейс SPI.

e. Проверьте тактовую частоту. При её отсутствии или неправильной настройке чтение регистров может работать нестабильно — см. раздел 3 и примечание о сбросе/записи в PLL_config, AFE_config.

Адресация микросхем на общей шине (IC_addr)

Это опциональный механизм для работы нескольких микросхем на одной шине SPI; для одной микросхемы он не требуется.

Каждая микросхема имеет собственный адрес — регистр IC_addr (адрес 64, 8 бит). Адрес цели задаётся регистром BUS_addr (адрес 83, 8 бит). Контроллер SPI микросхемы реагирует на команды только в случаях:

- IC_addr = 0;
- IC_addr ≠0 и BUS_addr = IC_addr;
- IC_addr ≠0, BUS_addr = 0 и установлен флаг BUS0_mode.

В остальных случаях микросхема не реагирует (за исключением перезаписи BUS_addr).

Для определения IC_addr конкретной микросхемы переберите адреса от 1 до 255. Для каждого кандидата `i`:

1. Запишите `i` в регистр BUS_addr (адрес 83).
2. Прочитайте регистр BUS_addr (адрес 83).
3. Если прочитанное значение равно `i` — микросхема приняла запись, её IC_addr = `i`.
4. Если отлично от `i` — адреса не совпали, переходите к следующему кандидату.

Если ни один адрес не дал совпадения — вернитесь к проверкам a–e данного раздела.

14. Блокировка записи и инициализация из ПЗУ

a. Перед настройкой убедитесь, что регистр WR_lock (адрес 68) = 0: при **WR_lock ≠0 все команды записи блокируются** (за исключением самого WR_lock и BUS_addr). После сброса WR_lock = 0; если конфигурация была ранее заблокирована — запишите 0 для разблокировки.

b. После завершения настройки всех регистров запишите ненулевое значение в регистр WR_lock (адрес 68). Это предотвратит случайное изменение конфигурации. Для разблокировки запишите 0.

c. Для серийного производства конфигурация может быть сохранена в блок OTP-памяти (BOTP, 512×16 бит). При установке INIT_conf.OTP_init_on (бит 4) = 1 инициализация из BOTP включена: конфигурация загружается автоматически при каждом сбросе микросхемы. Подробнее см. раздел ПЗУ. Внимание! Прожигать OTP_init_on = 1 допускается только последним шагом — после полного прожига и проверки BOTP: если BOTP пуста, автозагрузка инициализирует все регистры нулями, и микросхема перестаёт отвечать после включения питания. Порядок программирования и восстановление — в разделе ПЗУ → Порядок программирования ПЗУ.

*Последнее обновление **21 сент. 2026 г.***

Демонстрационное ПО

Демонстрационное ПО Workstation4ad3s для настройки, диагностики и отладки микросхемы 5400TP065A-022

Описание ПО

Программное обеспечение Workstation4ad3s предназначено для настройки, диагностики и отладки ...

Terminal

Вкладка Terminal предназначена для просмотра и редактирования регистров микросхемы в ручном ...

Окно Тест

Окно Test Board (меню → Utilities → Test Board) содержит средства первичной диагностики подключе...

Анализатор HandTap

Окно HandTap (вкладка в нижней панели) предназначено для диагностики работы контуров отслежи...

Энкодер

GraphView

Вкладка GraphView предназначена для осциллографирования регистровых данных микросхемы в ре...

Режим детектора

Режим детектора в GraphView предназначен для настройки амплитуды и смещения входных сигнало...

Отладчик CPU

Вкладка Debugger CPUs предназначена для пошаговой отладки программ микровычислителей CPU1...

ParallelSpiView (параллельный вывод)

Окно ParallelSpiView (меню → Options → ParallelSpiView) предназначено для диагностики работы мик...

BatchWriter (пакетная запись)

Окно BatchWriter (меню → Options → BatchWriter) предназначено для подбора и проверки настроек ...

Record (запись данных)

Программирование OTP

Микросхема содержит энергонезависимую память (OTP) для хранения конфигурации, которая автом...

Описание ПО Workstation4ad3s

Программное обеспечение **Workstation4ad3s** предназначено для настройки, диагностики и отладки микросхемы преобразователя 5400TP065A-022. Связь с микросхемой осуществляется через микроконтроллер МК (ad3s_prog_esp32) по интерфейсу USB CDC (виртуальный COM-порт), который далее общается с микросхемой по SPI.

Варианты аппаратной части и запуск

ПО распространяется в виде дистрибутива (скрипт запуска + JAR) и требует установленной **Java 17** или новее (JDK 17 для скачивания — на FTP, офлайн-вариант для ПК без интернета; либо на сайте Microsoft). Приложение не работает с микросхемой напрямую: связь выполняется через **мост на ESP32-S3** с прошивкой `ad3s_prog_esp32`, который подключается к ПК по USB (виртуальный COM-порт), а к микросхеме — по SPI. Чтобы ПО заработало с преобразователем, возможны два варианта аппаратной части.

Вариант 1 — готовая демонстрационная плата

Готовая отладочная плата уже содержит микросхему 5400TP065A-022, мост ESP32-S3 с залитой прошивкой и всю необходимую обвязку (тактовый генератор, опорные напряжения, фильтры, разъёмы). В этом варианте ничего программировать не нужно — достаточно подключить плату к ПК кабелем USB и запустить приложение. Удобен для быстрого начала работы и демонстрации.

Вариант 2 — своя плата с микросхемой 5400TP065A-022 + модуль ESP32-S3

Если используется собственная плата с микросхемой 5400TP065A-022, к ней подключается отдельный модуль **ESP32-S3**, который необходимо предварительно прошить мостовой прошивкой `ad3s_prog_esp32`. Исходники и инструкции находятся в репозитории

`DCsoyuz/ad3s`. При сборке прошивки в `menuconfig` (**AD3S Board Configuration → Version board**) выберите вариант платы **«Custom User Board»** и в файле `main/ad3s_types.hpp` задайте номера GPIO под свою обвязку (по умолчанию там рабочая разводка образца). Минимально требуются 4 линии SPI:

ESP32-S3	Микросхема	Назначение
MOSI	SDI	данные ПК → микросхема
MISO	SDO	данные микросхема → ПК
SCLK	SCLK	тактовый сигнал SPI
CS	SSTR	строб кадра

Опционально дополнительно подключаются: NRESET, STNDBY, SAMPLE, VC, nSEN, а также два канала квадратурных энкодеров (ENC1, ENC2). Режим SPI — 0, частота по умолчанию 10 МГц (настраивается в приложении).

Запуск и подключение

1. Подключите мост ESP32-S3 (или демо-плату) к ПК кабелем USB **с передачей данных**. На платах с двумя разъёмами type-C обмен с приложением ведётся через разъём **«USB»** (нативный USB, интерфейс CDC). Разъём **«UART»** (USB-UART мост) предназначен для прошивки контроллера.
2. Определите виртуальный COM-порт: «Диспетчер устройств» в Windows или `ls /dev/tty*` в Linux.
3. Запустите приложение: Windows — `Workstation4ad3s-X.X.X.bat`, Linux/macOS — `./Workstation4ad3s-X.X.X.sh` (при первом запуске требуется Java 17+; при её отсутствии приложение сообщит, как установить).
4. В левой панели (ConnectPanel) выберите **CDC-порт разъёма «USB»** (виртуальный COM-порт нативного USB) и нажмите **Open COM**.
5. После подключения читайте и записывайте регистры микросхемы через вкладку **Terminal**.

> ⓘ **ДВА COM-ПОРТА — КАКОЙ ВЫБИРАТЬ**

При одновременном подключении обоих разъёмов в системе регистрируются два независимых COM-порта: мост «UART» (прошивка/лог) и CDC-порт разъёма «USB» (приложение). В Workstation4ad3s используется CDC-порт разъёма «USB», а не порт моста.

СВЯЗАННЫЕ РЕСУРСЫ

- Прошивка моста ESP32-S3 (`ad3s_prog_esp32`): `DCsoyuz/ad3s`

Быстрый старт

ПО поставляется с готовым проектом `user_asm` — он открывается в дереве файлов (File Tree) при первом запуске и **уже настроен под типичный датчик — классический СКВТ** (resolver). Рабочая конфигурация по умолчанию на месте, вручную регистры настраивать не нужно:

1. **Подайте питание** на плату и микросхему.
2. Подключите плату к ПК и нажмите **Open COM** (см. «Запуск и подключение»).
3. Нажмите **Program** — в микросхему зашивается готовый набор регистров (`base_ram.txt`).
4. Нажмите **Loading** — запускается циклическое чтение регистров, и во вкладке **Terminal** в полях `C1Coord`/`C2Coord` появится **считанная координата (угол)** поворота датчика, а в `C1Vel`/`C2Vel` — скорость.

ЧТО ЗАШИТО В `user_asm` ПО УМОЛЧАНИЮ

Конфигурация рассчитана на **классический СКВТ** (resolver), два независимых канала:

- **Тип датчика:** СКВТ, режим sin/cos.
- **Возбуждение:** формируется внутренним генератором микросхемы (выводы EXO1/EXO2), синусоида ~12 кГц, минимальная амплитуда.
- **Входные сигналы:** sin/cos через встроенные ОУ (`OPA_en` = 1); блок восстановления опорного сигнала включён.
- **DC-коррекция:** включена (`DC_correction` = 1) — компенсация среднего уровня сигналов АЦП.

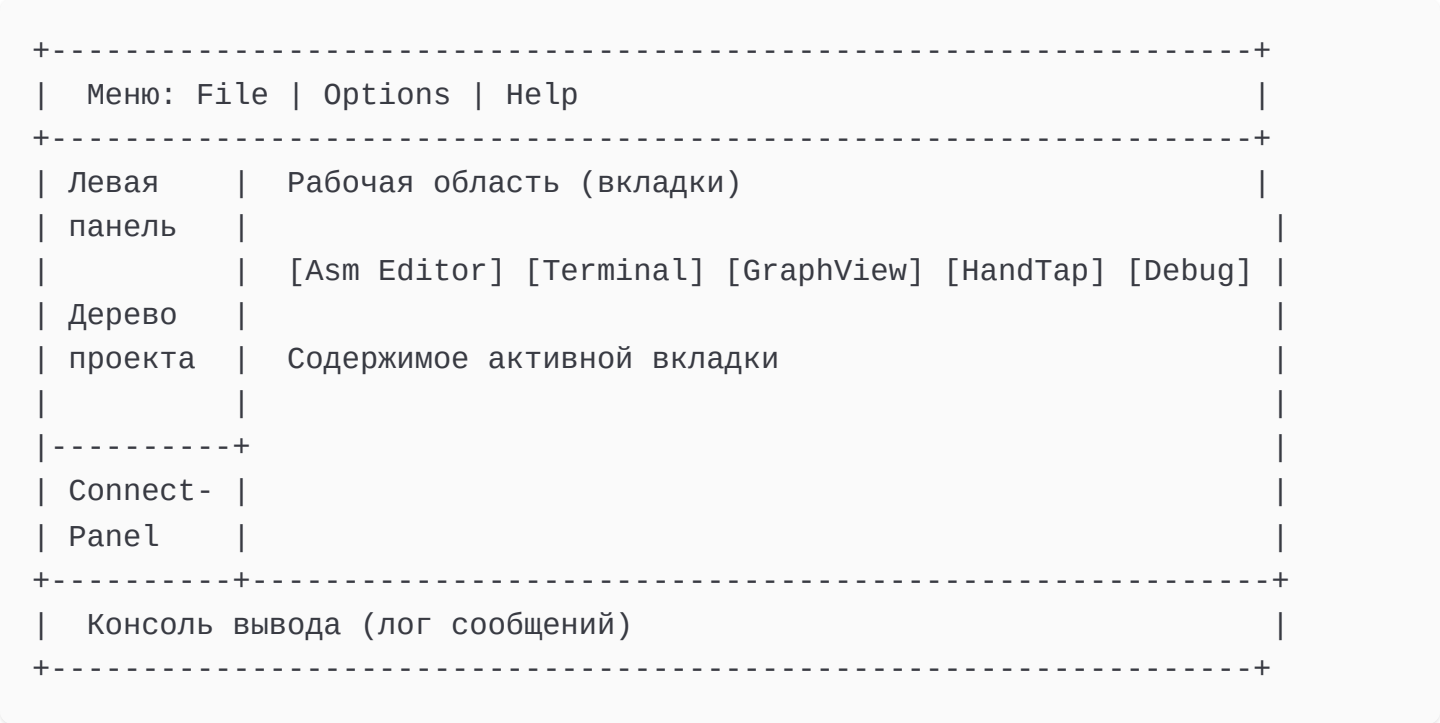
- **Тактирование:** от внешнего резонатора **10 МГц** на входе CLK60 (для демонстрационной платы; внутренний PLL выключен, `Fclk_adc` = 2,5 МГц).
- **Выход:** координата 16 бит и скорость по каждому каналу; эмуляция энкодера выключена.

После запуска доступны инструменты наблюдения и отладки:

- **HandTap** — осциллограммы сигналов с датчика (sin/cos, опора);
- **GraphView** — графики координаты/скорости в реальном времени;
- **Terminal** — чтение/запись отдельных регистров для калибровки;
- **ParallelSpiView** — двухканальный DMA-осциллограф.

Тонкую настройку (частоты, опорные уровни, режимы входных/выходных сигналов, калибровку) выполняйте по Методике подключения и настройки.

Структура окна



Левая панель

- **Меню** — строка меню (File, Options, Help).

- **Дерево проекта** — навигация по файлам проекта (ассемблерные исходники, RAM-файлы, конфигурация). Двойной клик открывает файл в редакторе.
- **ConnectPanel** — кнопки подключения и управления: выбор COM-порта, ParseAll, Program (запись оперативной конфигурации из `base_ram.txt`, **не** прожиг OTP), Prog Flash, Verify, Open COM, Clean log.

Рабочая область (вкладки)

5 стыкуемых вкладок, которые можно переключать, перетаскивать и откреплять в плавающие окна:

Вкладка	Описание
Asm Editor	Редактор ассемблерного кода с подсветкой синтаксиса
Terminal	Просмотр и редактирование регистров микросхемы
GraphView	Осциллограф регистровых данных в реальном времени
HandTap	Анализатор контуров отслеживания
Debugger CPUs	Пошаговый отладчик микровычислителей

Консоль вывода

Нижняя панель отображает системные сообщения, лог операций и ошибки. Контекстное меню (правый клик) — **Copy** и **Clear**.

Сохранение layout

Расположение вкладок автоматически сохраняется в `dock_layout.xml`. Для возврата к стандартному расположению — **Help → Restore Window**.

Меню

File

Пункт	Сочетание	Назначение
Open...	Ctrl+O	Открыть файл проекта
Save	Ctrl+S	Сохранить текущий файл
Save As...	Ctrl+Shift+S	Сохранить файл под новым именем
Exit	Ctrl+Q	Выход из приложения

Options

Пункт	Назначение
Float Helper	Утилита для работы с числами с плавающей точкой
Encoder	Просмотр данных энкодера
Paths	Редактор путей к файлам проекта
Record	Запись и воспроизведение сеансов
ParallelSpiView	Параллельный вывод данных HAND (2 канала, непрерывный поток)
BatchWriter	Пакетная запись регистров
Окно Тест	Диагностика подключения (LED, Find IC, SpiSpd)

Help

Пункт	Назначение
About	Информация о версии приложения
Registers	HTML-документация по регистрам микросхемы
Restore Window	Восстановление layout вкладок по умолчанию. Сбрасывает сохранённые настройки GraphView и детектора к начальным значениям.

Состав ПО

Terminal

Вкладка для просмотра и редактирования регистров микросхемы в ручном режиме. Содержит таблицу всех IC-регистров (адреса 0–95), панели настройки контуров C1/C2 (KampS, KampC, KbiasS, KbiasC и др.) и в правой колонке — панель `Actions:` с кнопками управления (Load/Store, Loading, Write to IC, Verify) и кнопками прожига OTP (Create BOTP, Prog BOTP, Verify BOTP, Create U22B, Prog UOTP).

Подробное описание — в разделе Terminal.

GraphView

Осциллограф регистровых данных реального времени. Считывает значения регистров микросхемы напрямую (до 4 независимых каналов на двух графиках) и отображает их в виде бегущих осциллограмм. Поддерживает настройку адресов регистров, разрядности, знаковости данных, а также режим детектора для анализа сигналов SIN/COS. Все введённые настройки сохраняются автоматически.

Подробное описание — в разделе GraphView.

Анализатор HandTap

Осциллограф-диагностик контуров отслеживания. При включении режима микровычислитель CPU1 перезагружается специальной программой, которая захватывает данные конвертера HAND1 или HAND2 и передаёт их на ПК для отображения в виде осциллограмм. Позволяет анализировать сигналы sin/cos АЦП, координату, скорость, метрики ошибки и другие внутренние сигналы микросхемы в реальном времени.

Подробное описание — в разделе Анализатор HandTap.

ParallelSpiView (параллельный вывод)

Осциллограф-диагностик с непрерывным потоком данных в режиме параллельной выдачи микросхемы 5400TP065A-022 (7-проводный параллельный вывод). В отличие от HandTap, обеспечивает неограниченный захват данных и одновременное отображение двух каналов (Addr1 и Addr2). Поддерживает управление сигналами VC и SDI, режим синхронизации (Trig) и авто-масштабирование. Открывается через меню Options → ParallelSpiView.

Подробное описание — в разделе ParallelSpiView.

Отладчик CPU

Пошаговый отладчик программ микровычислителей CPU1 и CPU2. Позволяет ставить до 2 точек останова, выполнять программу по шагам (Step), запускать до точки останова (Run), просматривать регистры (PC, инструкции, R0–R7, каналы, флаги) и память данных CPU. Листинг программы отображается с синтаксической подсветкой; редактирование выполняется в Asm Editor, изменения автоматически передаются в отладчик при сохранении.

Подробное описание — в разделе Отладчик CPU.

Окно Тест

Средства первичной диагностики подключения к микросхеме: управление светодиодом LED на плате МК, автоматический поиск адреса микросхемы (**Find IC**), настройка скорости

SPI (**SpiSpd**). Также содержит инструкции по восстановлению связи, если микросхема перестала отвечать.

Подробное описание — в разделе Окно Тест.

Энкодер

Окно для настройки и чтения данных квадратурного энкодера через ESP32. Поддерживает два канала (ENC1, ENC2), аппаратный счётчик PCNT и автоматический сброс по нулевой метке (Index). Настройки передаются при нажатии Start Reading.

Подробное описание — в разделе Энкодер.

Программирование OTP

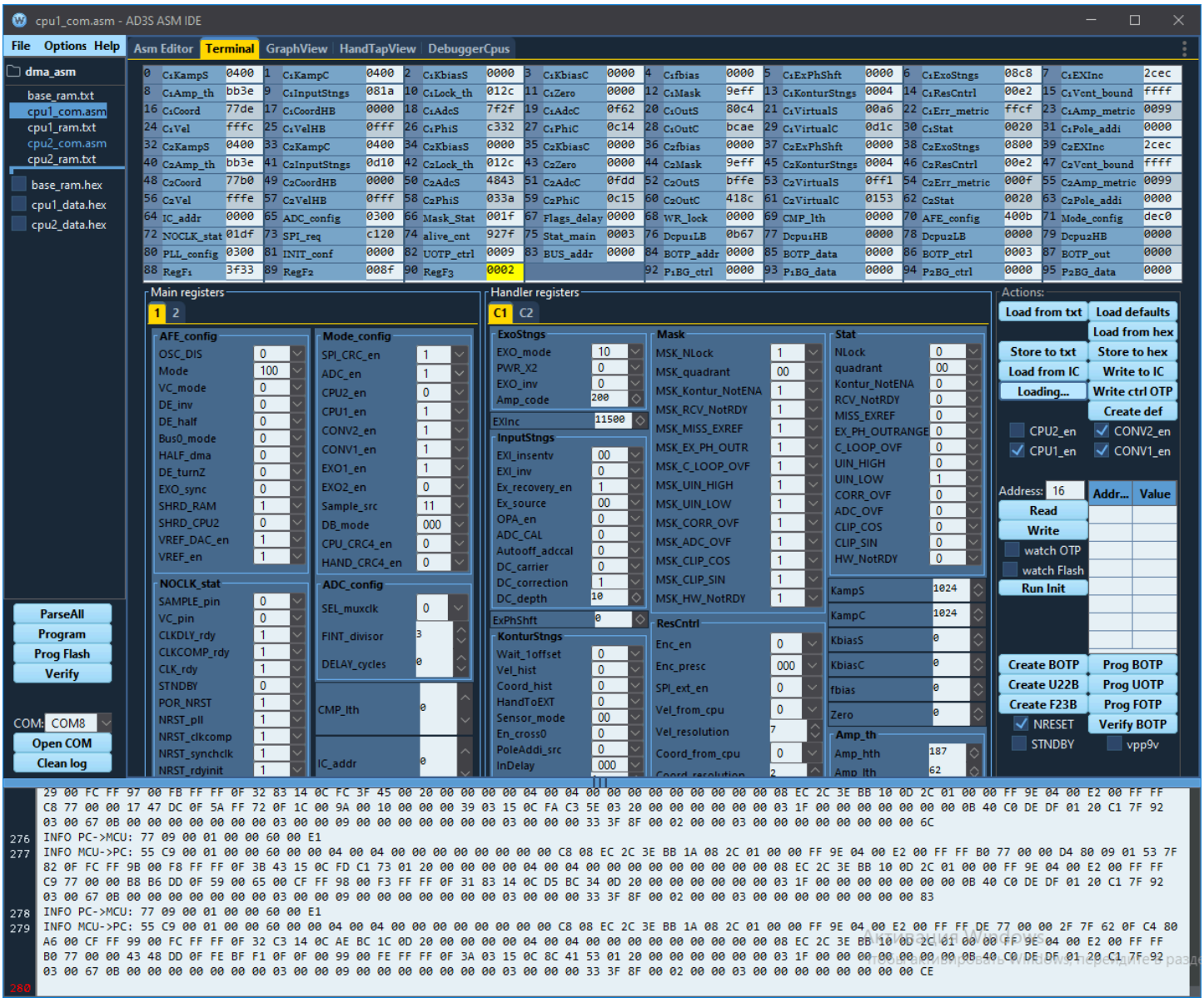
Сохранение конфигурации микросхемы в энергонезависимую память (BOTP/UOTP) для автоматической загрузки при включении или сбросе. Выполняется из вкладки **Terminal**, панель `Actions:`: генерация слепка регистров, прожиг (напряжение VPP подаётся автоматически) и верификация.

Подробное пошаговое описание — в разделе Программирование OTP.

*Последнее обновление **21 сент. 2026 г.***

Вкладка Terminal

Вкладка **Terminal** предназначена для просмотра и редактирования регистров микросхемы в ручном режиме. Это основной инструмент для настройки параметров контуров отслеживания, конфигурации АЦП, управления режимами работы микросхемы.



Структура вкладки

Окно Terminal разделено на две строки:

- **Верхняя часть** — таблица регистров IC (адреса 0–95).

- **Нижняя часть** — три колонки: основные регистры, регистры контуров C1/C2, редактор битовых полей.

```
+-------------------------------------------------------------------+
| Таблица регистров IC (адреса 0–95, 24 столбца)                    |
+-------------------------------------------------------------------+
| Основные регистры    | Регистры контуров   | Actions              |
| (вкладки "1" и "2")  | (вкладки C1 и C2)   | (кнопки управления)  |
+-------------------------------------------------------------------+
```

Таблица регистров IC

В верхней части расположена таблица всех регистров микросхемы (адреса 0–95). Таблица содержит 24 столбца — 8 групп по 3 столбца: **Адрес** | **Имя** | **Значение**.

- **Количество строк**: 12 (отображаются адреса 0–95).
- **Цветовая схема**: синие тона с пользовательским рендерингом ячеек.
- **Отслеживание изменений**: изменённые ячейки подсвечиваются жёлтым цветом.

Контекстное меню

Правый клик на ячейке значения открывает контекстное меню:

- **Read value** — прочитать значение регистра из микросхемы.
- **Write value** — записать введённое значение в регистр микросхемы.

Основные регистры (вкладки «1» и «2»)

Вкладка «1»

Группа	Регистры
Левая колонка	AFE_config (адрес 70), NOCLK_stat (адрес 72)

Группа	Регистры
Правая колонка	Mode_config (адрес 71), ADC_config (адрес 65), CMP_lth (адрес 69), IC_addr (адрес 64)

Вкладка «2»

Группа	Регистры
Левая колонка	Mask_Stat (адрес 66), RegF1–RegF3 (адреса 88–90, фабричные), BOTP_addr / BOTP_data / BOTP_ctrl (адреса 84–86)
Правая колонка	Stat_main (адрес 75), PLL_config (адрес 80), INIT_conf (адрес 81), UOTP_ctrl (адрес 82)

Регистры контуров отслеживания (C1 / C2)

Панель содержит две вкладки — **C1** (контур 1) и **C2** (контур 2), каждая с идентичным набором регистров:

Колонка 1 (левая)

Регистр	Описание
ExoStngs	Настройки генератора возбуждения
EXInc	Приращение амплитуды возбуждения
InputStngs	Настройки входного каскада
ExPhShft	Сдвиг фазы возбуждения

Регистр	Описание
KonturStngs	Настройки контура (включая HandToEXT)

Колонка 2 (центральная)

Регистр	Описание
Mask	Маска статуса
ResCntrl	Управление сбросом/рестартом контура

Колонка 3 (правая)

Регистр	Описание
Stat	Статус контура
KampS	Коэффициент усиления SIN
KampC	Коэффициент усиления COS
KbiasS	Смещение SIN
KbiasC	Смещение COS
fbias	Смещение частоты
Zero	Нулевое значение
Amp_th	Порог амплитуды

Подробное описание каждого регистра — в разделе Регистры.

Взаимодействие элементов

Таблица регистров, панели регистров контуров и редактор битовых полей **связаны двунаправленно**:

- Изменение значения в **ячейке таблицы** IC-регистров автоматически обновляет соответствующие битовые поля на панелях.
- Изменение **бита или поля** на панели контуров немедленно отражается в значении ячейки таблицы.

Таким образом, можно редактировать регистр любым удобным способом: вводя 16-битное значение целиком в таблицу, или управляя отдельными битами через флажки и поля на панелях.

Чтение и запись отдельной ячейки

Для каждой ячейки таблицы IC-регистров доступно контекстное меню **правой кнопкой мыши**:

- **Read value** — прочитать значение одного регистра из микросхемы.
- **Write value** — записать введённое значение в регистр микросхемы.

Это позволяет точечно обновить или проверить конкретный регистр, не затрагивая остальные.

Actions (кнопки управления)

Третья колонка (справа) — панель `Actions:` с кнопками для работы с регистрами и памятью микросхемы. Именно здесь находятся все кнопки прожига OTP (подробно — Программирование OTP). Разделена на несколько групп:

Загрузка и сохранение

Кнопка	Назначение
Load from txt	Загрузить значения регистров из файла `base_ram.txt`

Кнопка	Назначение
Load from hex	Загрузить значения из файла `base_ram.hex`
Load defaults	Загрузить фабричные значения по умолчанию
Store to txt	Сохранить текущие значения в `base_ram.txt`
Store to hex	Сохранить текущие значения в `base_ram.hex`
Create RDL	Сгенерировать RDL-файл с текущими значениями
Create BOTP	Сгенерировать файл `rom_BOTP.hex`
Create U22B	Сгенерировать файл `rom_u22b.hex` (PLL_config, INIT_conf)

Обмен с микросхемой

Кнопка	Назначение
Load from IC	Прочитать все регистры из микросхемы в таблицу
Write to IC	Записать в микросхему **настройки преобразователя** (не все регистры — см. ниже)
Loading... / STOP	Циклическое чтение регистров из микросхемы. При нажатии кнопка меняет текст на **STOP**, при повторном — останавливает цикл.

> **ЧТО ИМЕННО ЗАПИСЫВАЕТ «WRITE TO IC»**
>
> Кнопка **Write to IC** отправляет в микросхему только **конфигурационные регистры преобразователя** — те, что задают режим работы (коэффициенты, пороги, настройки входов/выходов, возбуждения, режим АЦП):
>
> - **Канал 1:** адреса 0–15, 31 (KampS, KampC, KbiasS, KbiasC, InputStngs, KonturStngs, ResCntrl, Amp_th и т.д.);

- **Канал 2:** адреса 32–47, 63;
- **Общие:** адреса 64–71 — IC_addr, ADC_config, Mask_Stat, Flags_delay, WR_lock, CMP_lth, AFE_config, Mode_config.

Write to IC записывает все настраиваемые регистры преобразователя, включая регистры тактирования и OTP-управления — PLL_config (80), INIT_conf (81), UOTP_ctrl (82), BOTP_addr (84), BOTP_data (85), BOTP_ctrl (86). Отдельная кнопка для их записи больше не нужна.

Записи не подлежат только статусные / вычисляемые регистры — координату (C1Coord/C2Coord), скорость (C1Vel/C2Vel), метрики, статусы, отсчёты АЦП (адреса 16–30, 48–62, 72–79, 83, 87, 91–95). Они вычисляются микросхемой в процессе работы.

Одиночный регистр (любой адрес) можно записать точечно через контекстное меню ячейки → **Write value**.

Запись ведётся в **оперативную память** чипа: настройки действуют до сброса питания. Чтобы сохранить их энергонезависимо — см. Программирование OTP.

Регистры по адресу

В нижней части — поле **Address** и таблица на 8 строк (Address | Value), а также кнопки:

Кнопка	Назначение
Read	Прочитать 8 слов из указанного адреса
Write	Записать 8 слов по указанному адресу
watch OTP	Флажок — при чтении читать OTP-область
watch Flash	Флажок — при чтении читать Flash-область ESP32
Run Init	Выполнить инициализацию из Flash ESP32 (NRESET + запись всех блоков)

Управление CPU и конвертерами

Флажок	Назначение
CPU1_en	Включение/выключение микровычислителя CPU1 (бит Mode_config)
CPU2_en	Включение/выключение микровычислителя CPU2
CONV1_en	Включение/выключение конвертера HAND1
CONV2_en	Включение/выключение конвертера HAND2

Программирование OTP

Кнопка	Назначение
Prog BOTP	Прожиг **BOTP** — энергонезависимый слепок регистров (необратимо, требует подтверждения)
Prog UOTP	Прожиг **UOTP** — энергонезависимое хранение PLL_config и INIT_conf (необратимо)
Verify BOTP	Верификация BOTP-памяти

ОПЕРАТИВНАЯ ПАМЯТЬ VS ЭНЕРГОНЕЗАВИСИМАЯ OTP

Write to IC (панель `Actions:`) пишет в **оперативную память** все настраиваемые регистры (включая тактирование и OTP-управление) — настройки действуют только до сброса питания. **Prog BOTP / Prog UOTP** прожигают **энергонезависимую OTP** — значения сохраняются и автоматически загружаются при каждом включении или сбросе. Не путайте с кнопкой **Program** в левой панели (ConnectPanel) — она записывает оперативную конфигурацию из `base_ram.txt` и к прожигу отношения не имеет. Пошаговый сценарий прожига — в разделе Программирование OTP.

Управление питанием

Флажок	Назначение
NRESET	Управление сигналом сброса микросхемы
STNDBY	Перевод микросхемы в режим standby
vpp9v	Ручная подача VPP (override для диагностики, с предупреждением). При штатном прожиге **не нужна** — напряжение VPP подаётся автоматически

> ⚠ **ПРЕДУПРЕЖДЕНИЕ**
>
> Кнопки **Prog BOTP**, **Prog UOTP** выполняют необратимую запись в OTP-память. Перед использованием убедитесь в корректности данных. Операция требует двойного подтверждения.

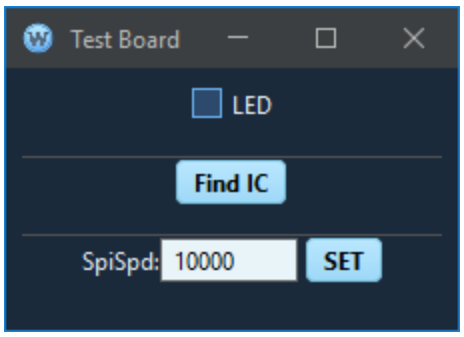
Типовой порядок работы

1. Подключите микросхему (Окно Тест → **Find IC**).
2. Откройте вкладку **Terminal**.
3. Нажмите **Load from IC** — все регистры микросхемы загрузятся в таблицу.
4. Для изменения регистра: правый клик по ячейке → **Write value** (одиночная запись), либо **Write to IC** — записать разом все настройки преобразователя, включая регистры тактирования и OTP-управления (PLL_config, INIT_conf, BOTP_ctrl).
5. Для настройки контура: используйте вкладки C1/C2 для записи коэффициентов KampS, KampC, KbiasS, KbiasC и других параметров.
6. Для наблюдения за изменениями в реальном времени: нажмите **Loading...** — регистры будут циклически обновляться.
7. Для сохранения текущих значений: **Store to txt** или **Store to hex**.
8. Чтобы сохранить конфигурацию **энергонезависимо** (с автозагрузкой при включении/сбросе): выполните прожиг OTP — **Create BOTP** → **Prog BOTP** → **Verify BOTP** и **Create U22B** → **Prog UOTP**. Пошагово — в разделе Программирование OTP.

*Последнее обновление **21 сент. 2026 г.***

Окно Тест

Окно **Test Board** (меню → Utilities → Test Board) содержит средства первичной диагностики подключения к микросхеме.



LED — управление состоянием вывода LED на плате МК. Включает/выключает индикаторный светодиод для визуальной проверки связи ПК с МК.

Find IC — автоматический поиск адреса микросхемы IC_addr. Программа перебирает адреса 1–255, записывая каждое значение в регистр BUS_addr (адрес 83) и читая его обратно. При совпадении записанного и прочитанного значений — адрес найден. Алгоритм подробно описан в разделе Методика подключения.

SpiSpd — задание скорости SPI-обмена между МК и микросхемой (команда МК 0x18). Значение по умолчанию — 10000. Уменьшение скорости может помочь при нестабильном обмене.

Когда микросхема не отвечает

Если при выполнении настройки (разделы 3–7 методики подключения) микросхема перестаёт отвечать на SPI-команды, возможной причиной является неверное значение делителя частоты ROM — поле BOTP_clkdel регистра INIT_conf (адрес 81). Частота тактирования ПЗУ определяется по таблице: **Fclk_rom = FINT / N**, где N зависит от BOTP_clkdel (подробно в описании BOTP_clkdel). Fclk_rom не должна превышать **1 МГц**. Например, при FINT = 10 МГц и BOTP_clkdel = 3 делитель N = 12, Fclk_rom ≈ 833 кГц. При превышении допустимой частоты микросхема может не корректно инициализироваться из BOTP-памяти и не отвечать на SPI. В этом случае:

1. Нажмите кнопку **Find IC** для проверки доступности микросхемы.

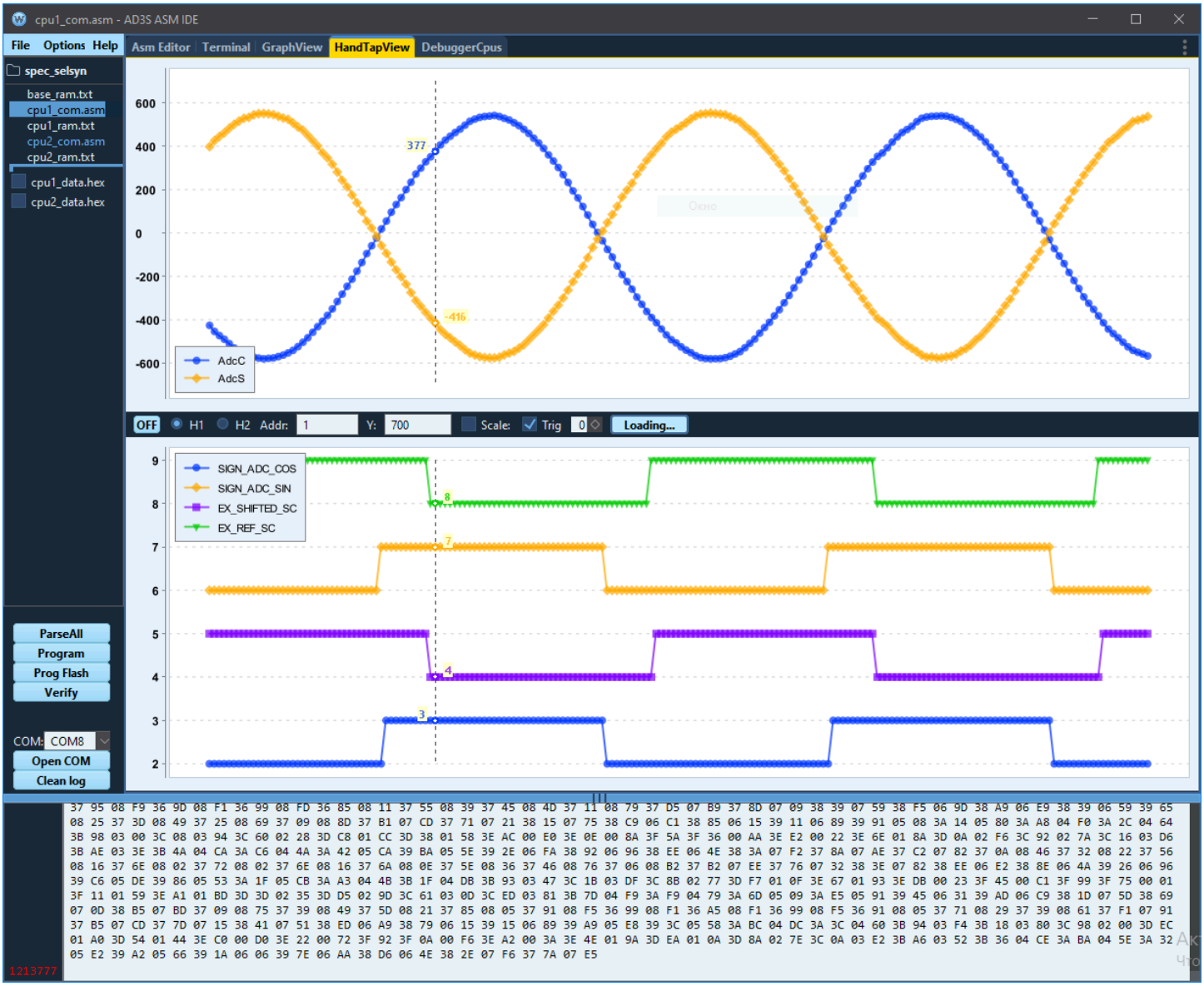
2. Если микросхема обнаружена — запишите в WR_lock (адрес 68) значение 0 для разблокировки записи.
3. Увеличьте BOTP_clkdel или запишите INIT_conf с корректным делителем.
4. При необходимости — перепрограммируйте BOTP (UOTP) с исправленным значением INIT_conf.

Вторая возможная причина — прожиг бита OTP_init_on до программирования BOTP: тогда при каждом включении питания все регистры инициализируются нулями из пустой ПЗУ, и микросхема не отвечает. Восстановление и правильный порядок прожига — в разделе Программирование OTP → Если микросхема перестала отвечать.

*Последнее обновление **21 сент. 2026 г.***

Анализатор HandTap

Окно **HandTap** (вкладка в нижней панели) предназначено для диагностики работы контуров отслеживания микросхемы. В режиме HandTap микровычислитель CPU1 перезагружается специальной программой, которая захватывает данные конвертера HAND1 или HAND2 и передаёт их на ПК для отображения в виде осциллограмм.



Расположение элементов

Окно разделено на три области:

- **Верхний график (Values)** — отображение многоразрядных числовых сигналов (например, sin, cos, координата, скорость). По горизонтали — номер отсчёта, по вертикали — значение.
- **Панель управления** — строка с кнопками и настройками (см. ниже).
- **Нижний график (Signals)** — отображение одноразрядных цифровых сигналов (например, `exi`, `exi_recovered`).

На обоих графиках доступен курсор: клик левой кнопкой устанавливает вертикальную пунктирную линию, на пересечении которой с каждым сигналом отображается кружок и числовое значение. Правый клик убирает курсор.

Органы управления

Элемент	Назначение
ON / OFF	Включение/выключение режима HandTap. Кнопка работает как toggle — при нажатии **ON** выполняется процедура включения (см. ниже), кнопка меняет текст на **OFF**. Повторное нажатие выключает режим.
H1 / H2	Переключатель радиокнопок — выбор канала конвертера: HAND1 или HAND2.
Addr	Раскрывающийся список (0–7) для выбора адреса блока данных. Каждый пункт показывает номер адреса и список сигналов через запятую (например, «1 - Wca, BScos, ex_shifted, Wsa, BSsin, ex_ref»). Выбранный адрес определяет, какие поля будут декодированы из захваченных отсчётов (см. таблицу «Адреса блоков данных»). По умолчанию — 1 (SinCos).
Y	Текстовое поле, диапазон оси Y верхнего графика (± значение). По умолчанию — 2000.
Scale	Флажок авто-масштабирования оси Y. При установке диапазон определяется автоматически по амплитуде данных, поле Y становится недоступным.

Элемент	Назначение
Trig	Флажок режима синхронизации по возбуждению (triggered). При установке захват начинается от отрицательного фронта сигнала EXI (возбуждение). Без флага — захват выполняется немедленно.
Threshold	Спиннер (0–99), количество тактов ожидания между отсчётами. Записывается в регистр 763 перед началом цикла чтения.
Loading... / Stop	Запуск/остановка циклического чтения данных. При нажатии **Loading...** начинается непрерывный захват и обновление графиков. Кнопка меняет текст на **Stop**.

Включение и выключение режима

Включение (ON)

При нажатии кнопки **ON** выполняется следующая последовательность:

1. **Отключение CPU1 и CPU2** — в регистре Mode_config (адрес 71) сбрасываются биты `CPU1_en` и `CPU2_en`.
2. **Настройка разделяемой RAM** — в регистре AFE_config (адрес 70) устанавливается `SHRD_RAM = 1`, `SHRD_CPU2 = 0`.
3. **Загрузка программы** — в программную память CPU1 (адреса 512–1023) загружается скомпилированная ассемблерная программа `handtap_asm/cpu1_data.hex`.
4. **Включение CPU1** — в регистре Mode_config устанавливается бит `CPU1_en`.

Выключение (OFF)

При нажатии кнопки **OFF**:

1. В регистре Mode_config сбрасывается бит `CPU1_en` — CPU1 останавливается.

> ⚠ **ПРЕДУПРЕЖДЕНИЕ**

> При выключении режима HandTap микросхема остаётся с отключённым CPU. Для восстановления штатной работы необходимо перезагрузить микросхему или вручную загрузить рабочие программы CPU.

Процесс захвата данных

После нажатия **Loading...** выполняется циклический опрос:

1. Значение **Threshold** записывается в регистр 763 микросхемы.
2. Из регистра 760 считывается начальное значение счётчика `actionHandValue`.
3. Циклически отправляется команда `READ_HANDTAP` (0x0E) на МК (контроллер LED Console) с параметрами:
   - `actionHandValue` — инкрементируемый счётчик для синхронизации;
   - `address` — адрес блока данных (биты 2:0), флаг Trig (бит 3), флаг H2 (бит 4).
4. МК записывает эти параметры в регистры микросхемы 760–761 и ожидает готовности, опрашивая регистр 748 (флаг готовности CPU1).
5. После готовности МК считывает 384 16-битных слова из трёх областей разделяемой RAM:
   - 128 слов с адреса 608;
   - 128 слов с адреса 768;
   - 128 слов с адреса 1280.
6. Данные передаются на ПК, где декодируются: пары 14-битных слов объединяются в 28-битные значения, переставляются в хронологическом порядке (итого 128 отсчётов) и разбираются на поля в соответствии с выбранным адресом блока.

Адреса блоков данных

Поле **Addr** определяет, какие данные считываются из конвертера HAND. Адресация соответствует команде микровычислителя `SET_A_HAND` (подробнее в разделе CPU):

Addr	Блок	Поля на графике
0	**Coord** (Координата)	`FullAngle` (28 бит)
1	**SinCos** (АЦП)	`Wca` (12 бит, зн.), `BScos`, `ex_shifted`, `Wsa` (12 бит, зн.), `BSsin`, `ex_ref`
2	**OutVirtSin**	`VirtualS` (13 бит, зн.), `ex_recovered_S`, `BSsin_V` (13 бит, зн.), `ex_recovered_90dgr_S`
3	**ErrAmpMetric**	`Amp_metric` (12 бит), `Err_metric` (16 бит, зн.)
4	**Vel** (Скорость)	`FullVel` (28 бит, зн.)
5	**PhiModel**	`PhiC` (14 бит, зн.), `PhiS` (14 бит, зн.)
6	**OutVirtCos**	`VirtualC` (13 бит, зн.), `ex_recovered_C`, `BScos_V` (13 бит, зн.), `ex_recovered_90dgr_C`
7	**Stat** (Статус)	`STAT` (16 бит)

Многоразрядные поля (размер > 1 бит) отображаются на верхнем графике **Values**. Одноразрядные поля — на нижнем графике **Signals** со смещением +2 по оси Y для визуального разделения.

Режим синхронизации (Trig)

При установленном флаге **Trig** программа CPU1 перед захватом данных ожидает отрицательный фронт сигнала возбуждения EXI. Это позволяет привязать осциллограмму к определённой фазе сигнала возбуждения и анализировать поведение контура синхронно с ним.

Без флага Trig захват начинается немедленно после получения команды от МК.

Типичное использование

1. Подключите микросхему и убедитесь в связи (Test Board → Find IC).
2. Откройте вкладку **HandTap**.
3. Нажмите **ON** — загрузится программа захвата в CPU1.
4. Выберите канал **H1** или **H2**.
5. В раскрывающемся списке **Addr** выберите «1 - Wca, BScos, ex_shifted, Wsa, BSsin, ex_ref» для просмотра сигналов Sin/Cos АЦП.
6. При необходимости включите **Trig** для синхронизации от возбуждения.
7. Нажмите **Loading...** — начнётся захват и отображение данных.
8. Используйте курсор (клик на графике) для точного отсчёта значений.
9. Для просмотра других данных выберите другой адрес в списке **Addr**: 0 — координата, 4 — скорость, 3 — метрики ошибки и т.д.
10. По окончании нажмите **Stop**, затем **OFF** для выхода из режима HandTap.

Программа CPU1 режима HandTap

При включении режима HandTap в программную память CPU1 загружается специальная программа из файла `handtap_asm/cpu1_data.hex`. Программа работает в бесконечном цикле, ожидая команды от МК через SPI и выполняя захват данных конвертера HAND в разделяемую RAM. Исходный текст программы — файл `cpu1_com.asm`.

Карта памяти CPU1

Программа использует следующее распределение памяти (адреса указаны в 14-битных словах):

Адреса	Назначение
0–95	Программная память (код `cpu1_com.asm`)
96–223	Буфер захвата `TAPBUF_0` (128 слов = 256 байт) — для данных первого блока

Адреса	Назначение
224–229	Зарезервировано / ROM-константы
230	`CONST_50` = 0x50 (80) — смещение второго блока захвата в буфере
231	`CONST_7F` = 0x7F (127) — количество отсчётов второго блока
235	`PREV_EX_REF` — предыдущее значение бита EXI (для детектирования фронта)
236	`TO_OUT_READY` — флаг готовности данных (отображается в регистре IC 748)
237	`BUFFERED_SPI_0` — буфер предыдущего значения SPI_0 (для детектирования новой команды)
248	`SPI_0` — слово 0 данных SPI от МК (счётчик `actionHandValue`)
249	`SPI_1` — слово 1 данных SPI (адрес + флаги Trig/H2)
251	`SPI_3` / `num_wait_samples` — слово 3 (число тактов ожидания между отсчётами, регистр IC 763)
252	`HAND1_L` — младшие 14 бит данных HAND1
253	`HAND1_H` — старшие 14 бит данных HAND1
254	`HAND2_L` — младшие 14 бит данных HAND2
255	`HAND2_H` — старшие 14 бит данных HAND2

Буфер захвата `TAPBUF_0` отображается в разделяемую RAM микросхемы и доступен для чтения МК через SPI:

- Первый блок (64 отсчёта × 2 слова = 128 слов) — адреса IC **608–735**

- Второй блок (128 отсчётов × 2 слова = 256 слов) — размещается начиная со смещения 80 в буфере, читается как два региона: **768–895** и **1280–1407**

Алгоритм работы программы

Программа выполняется в бесконечном цикле. Ниже описан каждый этап:

1. Ожидание команды SPI

```
WAIT_SPI_START_LOOP:
    LOAD SPI_0 R0              // Читаем текущее значение SPI_0
    LOAD BUFFERED_SPI_0 R1     // Читаем предыдущее значение
    EQUAL WAIT_SPI_START_LOOP R0 R1   // Если совпадают — команда ещё не
поступила, ждём
    STORE BUFFERED_SPI_0 R0    // Запоминаем новое значение
```

CPU1 опрашивает регистр `SPI_0` (адрес 248), сравнивая его с ранее сохранённым значением. Когда МК записывает через SPI новое значение `actionHandValue` (регистр IC 760), оно появляется в `SPI_0` микровычислителя. Изменение `SPI_0` означает поступление новой команды.

2. Декодирование параметров

```
    LOAD SPI_1 R0              // Читаем слово параметров (регистр IC 761)
    CONST 7 R2
    ANL R2 R0                  // R0 = SPI_1 & 0x07 — адрес блока данных
(0–7)
    HFIX R0                    // Устанавливаем адрес чтения HAND
    LOAD SPI_1 R0
    CONST 16 R2
    ANL R2 R0                  // R0 = SPI_1 & 0x10 — флаг H2
    EQUAL0 HANDLER0_PROC R0    // Если H2=0 → HAND1, иначе → HAND2
    JUMP HANDLER1_PROC
```

Из `SPI_1` (регистр IC 761) извлекаются два параметра:

Бит	Маска	Флаг	Назначение
2:0	0x07	`addr`	Адрес блока данных конвертера (команда `SET_A_HAND`)
4	0x10	`H2`	Выбор канала: 0 = HAND1, 1 = HAND2
3	0x08	`Trig`	Режим синхронизации (обрабатывается на следующем шаге)

Команда `HFIX` загружает адрес в мультиплексор конвертера, определяя, какие данные (координата, sin/cos, скорость и т.д.) будут выдаваться через HAND при каждом вызове `WAIT_OWN_HAND`.

В зависимости от флага H2 программа переходит к одному из двух идентичных обработчиков — для HAND1 (`HANDLER0_PROC`) или HAND2 (`HANDLER1_PROC`).

3. Ожидание синхронизации (режим Trig)

Если в параметрах установлен бит `Trig` (0x08), программа ожидает отрицательный фронт сигнала возбуждения EXI:

```
    LOAD SPI_1 R0
    CONST 8 R2
    ANL R0 R2                   // Проверяем бит Trig
    EQUAL0 EXIT_WAIT_NEG R2     // Trig не установлен — пропускаем ожидание

    CONST 1 R0                  // Маска бита EXI (бит 0 в HAND1_L)
WAIT_NEG_LOOP:
    LOAD32 HAND1_L R1           // Читаем младшее слово HAND1
    ANL R0 R1                   // Выделяем бит EXI
    LOAD PREV_EX_REF R2         // Загружаем предыдущее значение
    STORE PREV_EX_REF R1        // Сохраняем текущее значение
    SUB R2 R1                   // R1 = prev - current
    EQUAL EXIT_WAIT_NEG R0 R1   // Если prev=1, current=0 → R1=1 → переход
обнаружен
    JUMP WAIT_NEG_LOOP
```

Бит EXI (сигнал возбуждения) передаётся в бите 0 младшего слова данных HAND. Программа запоминает предыдущее значение бита и сравнивает с текущим. Когда

предыдущее = 1, а текущее = 0 — обнаружен отрицательный фронт. Это точка, от которой начинается захват данных.

Если флаг Trig не установлен — захват начинается немедленно.

4. Захват первого блока (64 отсчёта)

```
EXIT_WAIT_NEG:
    CONST 0 R0                  // Начальный адрес в буфере
    CONST 63 R1                 // Количество отсчётов − 1 (0..63)
LOOP:
    MOVE R0 R3
    ADD R1 R3                   // R3 = адрес в буфере + текущий индекс
    CFIX1 R3                    // Преобразуем в адрес записи в разделяемую
RAM
    LOAD num_wait_samples R4    // Загружаем задержку из SPI_3 (регистр IC
763)
WAIT0_SAMPLES:
    WAIT_OWN_HAND               // Ожидаем обновление данных от конвертера
    DJNZ WAIT0_SAMPLES R4       // Декремент R4, если не 0 — ждём ещё
    LOAD32 HAND1_L R3           // Читаем 28-битное значение HAND
    CSTORE32 TAPBUF_0 R3        // Записываем 32 бит в буфер (2 × 14 бит)
    DJNZ LOOP R1                // Следующий отсчёт
```

Первый блок захватывает 64 отсчёта (индексы 0–63), записывая их последовательно в буфер `TAPBUF_0` начиная с адреса 0. Команда `CSTORE32` записывает 32-битное значение как два 14-битных слова (младшее и старшее) в разделяемую RAM.

Между отсчётами программа выполняет `num_wait_samples` (значение поля **Threshold**, регистр IC 763) циклов ожидания `WAIT_OWN_HAND`, задавая временной интервал между точками захвата.

5. Захват второго блока (128 отсчётов)

```
    LOAD CONST_50 R0            // Смещение = 80 (0x50)
    LOAD CONST_7F R1            // Количество отсчётов − 1 (0..127)
LOOP2:
    MOVE R0 R3
    ADD R1 R3                   // R3 = 80 + текущий индекс
    CFIX1 R3
    LOAD num_wait_samples R4
```

```
WAIT1_SAMPLES:
    WAIT_OWN_HAND
    DJNZ WAIT1_SAMPLES R4
    LOAD32 HAND1_L R3
    CSTORE32 TAPBUF_0 R3
    DJNZ LOOP2 R1
```

Второй блок захватывает 128 отсчётов (индексы 80–207), записывая их в буфер начиная со смещения 80. Общее количество захваченных отсчётов: **64 + 128 = 192** (384 14-битных слова).

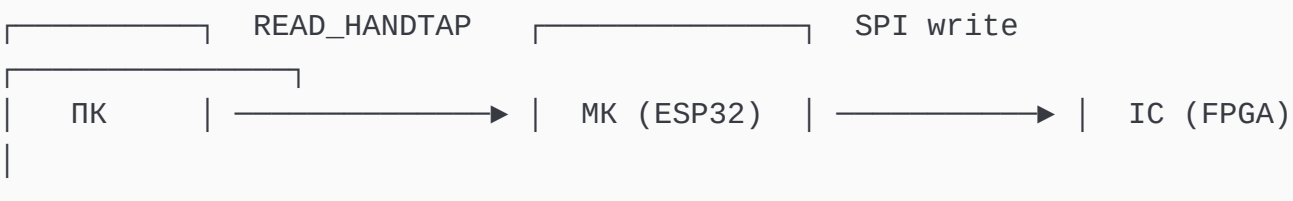
> **К СВЕДЕНИЮ**
>
> Разделение на два блока связано с ограничением адресации команды `CFIX1` — адрес должен вычисляться как база + индекс, при этом первый блок размещается с начала буфера (индексы 0–63), а второй — со смещением 80. Между блоками остаётся «пропуск» в 16 слов (адреса 64–79), которые не заполняются.

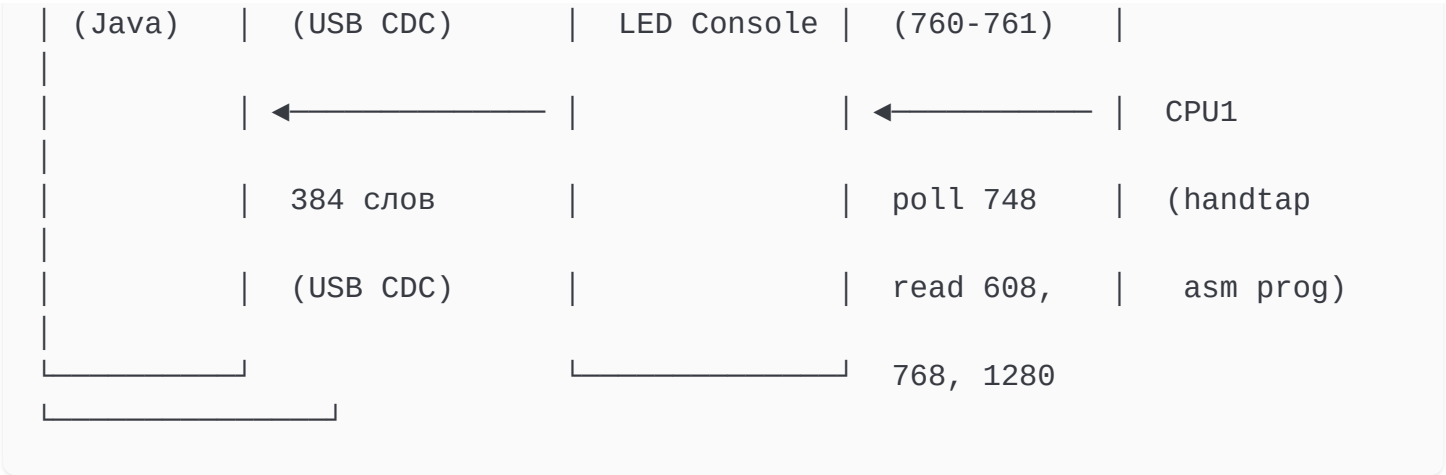
6. Сигнализация готовности

```
    LOAD BUFFERED_SPI_0 R0     // Загружаем полученный actionHandValue
    STORE TO_OUT_READY R0      // Записываем в регистр IC 748 — данные
готовы
    JUMP WAIT_SPI_START_LOOP   // Возвращаемся к ожиданию следующей команды
```

После завершения захвата программа записывает полученное значение `actionHandValue` в регистр `TO_OUT_READY` (отображается как регистр IC **748**). МК, периодически опрашивая этот регистр, обнаруживает совпадение с отправленным значением — это сигнал о том, что CPU1 завершил захват и данные в разделяемой RAM готовы для чтения.

Взаимодействие CPU1, МК и ПК — полная диаграмма





1. **ПК** отправляет команду `READ_HANDTAP` (0x0E) на МК с параметрами: `actionHandValue` + адрес/флаги.
2. **МК** записывает параметры через SPI в регистры IC 760–761.
3. **CPU1** (работает программа `cpu1_com.asm`) обнаруживает изменение `SPI_0`, декодирует параметры, при необходимости ждёт фронт EXI, захватывает 192 отсчёта в разделяемую RAM, записывает `actionHandValue` в регистр 748.
4. **МК** опрашивает регистр 748, обнаруживает совпадение флага готовности, считывает 384 слова (128 × 3 региона) из разделяемой RAM.
5. **МК** отправляет 384 слова на **ПК** через USB CDC.
6. **ПК** декодирует данные и отображает на графиках HandTap.

*Последнее обновление **3 июл. 2026 г.***

Энкодер

Окно **Encoder** (меню → Options → Encoder) предназначено для настройки и чтения данных внешнего квадратурного энкодера через контроллер ESP32. Поддерживаются два независимых канала (ENC1, ENC2), каждый из которых подключён к GPIO-выводам ESP32 и обрабатывается аппаратным счётчиком импульсов (PCNT).

Принцип работы

ESP32 считывает сигналы квадратурного энкодера (A, B, Index) через встроенный периферийный модуль **PCNT (Pulse Counter)**. Счётчик автоматически наращивает/уменьшает значение при каждом фронте сигналов A и B, обеспечивая аппаратное декодирование направления вращения. Сигнал **Index** (нулевая метка энкодера) подключён к GPIO-прерыванию: при нарастающем фронте ISR немедленно сбрасывает счётчик в ноль.

```
Энкодер ──► GPIO (A, B, Index) ──► ESP32 PCNT ──► UART ──► ПК (EncoderView)
                                          │
                                  Index ISR → счётчик = 0
```

Начальное состояние и сброс нуля

По умолчанию настройка идёт на **8-битный код угла** (Coord_resolution = 10, что задаёт разрядность координаты 18 shl 10 = 28 бит, но Vel_resolution = 0 и Enc_presc = 0).

После включения энкодера (нажатие **Start Reading**) счётчик PCNT инициализируется нулём. Это начальное состояние **не соответствует реальному положению датчика**, пока датчик физически не пройдёт через точку нуля (сработает сигнал Index). Только после прохождения нулевой метки счётчик корректно привязывается к углу энкодера.

Если необходимо принудительно сбросить счётчик — выключите и включите энкодер (**Stop Reading** → **Start Reading**), после чего проведите датчик через нулевую точку.

Окно Encoder



Окно разделено на две части:

- **Верхняя панель** — текущие значения: угол ENC1, угол ENC2, координата C1 Coord, координата C2 Coord.
- **Центральная панель** — настройки двух каналов (C1 и C2) с одинаковым набором параметров.

Текущие значения

Поле	Описание
ENC1 Angle	Текущее значение счётчика PCNT энкодера 1 (считывается из ESP32)
ENC2 Angle	Текущее значение счётчика PCNT энкодера 2
C1 Coord	Координата контура 1 из регистра C1Coord микросхемы (SPI адрес 16)
C2 Coord	Координата контура 2 из регистра C2Coord микросхемы (SPI адрес 48)

Настройки каналов

Каждый канал (C1, C2) содержит следующие параметры:

Параметр	Регистр	Биты	По умолчанию	Описание
Enc_presc	ResCntrl	[14:12]	0	Делитель частоты сигналов энкодера. Определяет максимальную частоту переключения: 0 = FINT/2, 1 = FINT/3, ..., 7 = FINT/64
Vel_res	ResCntrl	[8:5]	0	Разрешение скорости. Маскирует (обнуляет) младшие N бит значения скорости
Coord_res	ResCntrl	[3:0]	10	Разрешение координаты. Определяет разрядность: bits = 18 shl Coord_resolution
LBW	KonturStngs	[4:0]	10	Полоса пропускания контура отслеживания

Включение и выключение

Start Reading

При нажатии **Start Reading**:

1. **Чтение текущих регистров** — из микросхемы через ESP32 по SPI считываются регистры C1ResCntrl (адрес 14), C2ResCntrl (адрес 46), C1KonturStngs (адрес 13), C2KonturStngs (адрес 45).
2. **Применение настроек** — значения полей с панели (Enc_presc, Vel_res, Coord_res, LBW) накладываются на считанные значения регистров. При этом **устанавливается бит Enc_en** (бит 15 регистра ResCntrl), включающий блок энкодера в микросхеме.

3. **Запись регистров** — обновлённые значения записываются обратно в микросхему через SPI.
4. **Включение ESP32 PCNT** — на ESP32 отправляется команда `ENC_ON` (0x19), которая инициализирует аппаратные счётчики PCNT, подключает GPIO-прерывания для Index и запускает счёт.
5. **Запуск опроса** — запускается циклическое чтение с периодом 50 мс: счётчики ENC1/ENC2 из ESP32, координаты C1/C2 Coord из микросхемы.

> **К СВЕДЕНИЮ**
>
> Настройки с панели **передаются в микросхему только при нажатии Start Reading**. До этого момента изменённые значения хранятся только в интерфейсе.

Stop Reading

При нажатии **Stop Reading**:

1. Останавливается циклический опрос.
2. В регистры ResCntrl **сбрасывается бит Enc_en** (бит 15 = 0) — блок энкодера в микросхеме выключается.
3. На ESP32 отправляется команда `ENC_OFF` (0x1A) — аппаратные PCNT останавливаются.

Сохранение настроек

Введённые настройки (Enc_presc, Vel_res, Coord_res, LBW) автоматически сохраняются между сеансами работы приложения и восстанавливаются при следующем открытии окна Encoder.

GPIO-подключение

Назначение выводов энкодера на плате ESP32:

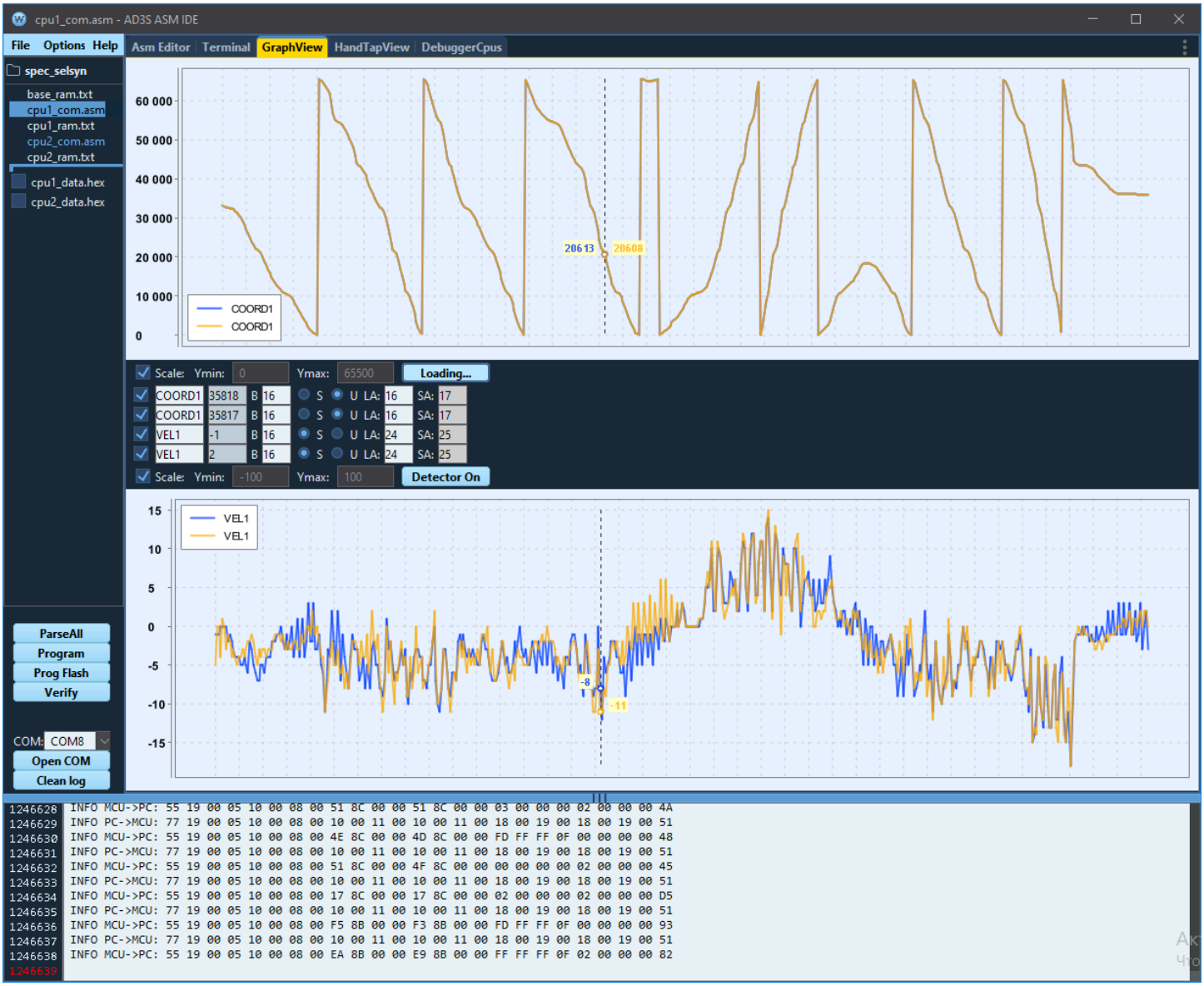
Сигнал	ENC1	ENC2
A (квадратура)	GPIO 47	GPIO 5
B (квадратура)	GPIO 21	GPIO 6
Index (нулевая метка)	GPIO 14	GPIO 4

Сигнал A — основной канал квадратуры, B — сдвинутый на 90°, Index — нулевая метка (один импульс за оборот). Для подавления дребезга используется аппаратный фильтр PCNT (glitch filter = 1000 нс).

*Последнее обновление **3 июл. 2026 г.***

GraphView

Вкладка **GraphView** предназначена для осциллографирования регистровых данных микросхемы в реальном времени. В отличие от анализатора HandTap, GraphView считывает значения регистров напрямую, без загрузки специальной программы в CPU, и может отображать до 4 независимых сигналов на двух графиках.



Расположение элементов

Окно разделено на несколько областей:

- **Верхний график (chart1)** — отображение первых двух каналов (серия `s0` и `s1`). По умолчанию — координата.
- **Строка управления графиком 1** — флажок авто-масштабирования, поля Ymin/Ymax.
- **4 строки настройки каналов** — конфигурация каждого из 4 сигналов (адрес, разрядность, легенда).
- **Строка управления графиком 2** — флажок авто-масштабирования, поля Ymin/Ymax, кнопка детектора, переключатель H1/H2.
- **Нижний график (chart2)** — отображение каналов 2 и 3 (серия `s2` и `s3`). По умолчанию — скорость.

Каналы 0 и 1 выводятся на верхний график, каналы 2 и 3 — на нижний.

На обоих графиках доступен курсор: клик левой кнопкой устанавливает вертикальную пунктирную линию, на пересечении которой с каждым сигналом отображается кружок и числовое значение. Правый клик убирает курсор. Положение курсора синхронизировано между обоими графиками.

Органы управления

Настройка каналов (4 строки)

Каждый канал имеет следующие элементы управления:

Элемент	Назначение
Флажок (✓)	Включение/выключение канала. Адреса активных каналов добавляются в список регистров для циклического чтения.
Legend	Текстовое поле — название серии на графике (например, «COORD1», «VEL1»).
Значение	Текстовое поле (серое, только чтение) — отображает текущее последнее значение канала.

Элемент	Назначение
NumBits	Текстовое поле — разрядность значения (количество младших бит, извлекаемых из 32-битного слова).
S / U	Радиокнопки — знаковый (Signed) или беззнаковый (Unsigned) формат данных. При знаковом формате выполняется расширение знака.
LA	Текстовое поле — младший адрес регистра (Low Address). Старший адрес вычисляется автоматически как LA + 1.

Управление графиками

Элемент	Назначение
Scale	Флажок авто-масштабирования оси Y. При установке диапазон определяется автоматически. При снятии — становятся доступны поля Ymin/Ymax для ручного задания диапазона.
Ymin / Ymax	Поля ввода нижней и верхней границы оси Y. Округляются до сотен.
Loading... / Stop	Запуск/остановка циклического чтения регистров.
Detector On / Off	Включение/выключение режима детектора (см. ниже).
H1 / H2	Переключатель канала детектора (активен только в режиме Detector).

Сохранение и восстановление настроек

Все введённые настройки (адреса, легенды, разрядность, знаковость, состояние флажков каналов) **сохраняются автоматически** при нажатии кнопки **Loading...**. Настройки

хранятся в файле `config.properties` и восстанавливаются при следующем запуске программы.

Для возврата к настройкам по умолчанию выберите в меню **Restore Window** — все параметры GraphView сбросятся к начальным значениям.

Чтение данных

При нажатии **Loading...** выполняется циклическое чтение:

1. Формируется список адресов из активных каналов (для каждого канала — пара адресов: LA и LA+1).
2. Циклически отправляется команда чтения произвольных регистров на МК.
3. МК считывает указанные регистры микросхемы и возвращает значения.
4. Полученные пары 16-битных значений объединяются в 32-битное слово:
   - Для обычных регистров: `value = (high << 16) | low`
   - Для адресов CPU (≥ 512): `value = (high << 14) | low`
5. Извлекаются младшие `numBits` бит, при знаковом формате выполняется расширение знака.
6. Значение добавляется в кольцевой буфер FIFO (глубина 512 отсчётов).
7. Графики обновляются каждые 40 мс.

Адреса по умолчанию

Обычный режим

Канал	Легенда	Адрес (LA)	SA (LA+1)	NumBits	Signed	График
0	COORD1	16	17	16	Нет	chart1
1	COORD1	16	17	16	Нет	chart1
2	VEL1	24	25	16	Да	chart2

Канал	Легенда	Адрес (LA)	SA (LA+1)	NumBits	Signed	График
3	VEL1	24	25	16	Да	chart2

Режим детектора H1

Канал	Легенда	Адрес (LA)	SA	NumBits	Signed
0	SIN1_IN	640	641	13	Да
1	SIN1_VIRT	648	649	13	Да
2	COS1_IN	642	643	13	Да
3	COS2_VIRT	650	651	13	Да

Режим детектора H2

Канал	Легенда	Адрес (LA)	SA	NumBits	Signed
0	SIN2_IN	644	645	13	Да
1	SIN2_VIRT	652	653	13	Да
2	COS2_IN	646	647	13	Да
3	COS2_VIRT	654	655	13	Да

Режим детектора

Кнопка **Detector On/Off** в нижней строке управления включает специальный режим для настройки амплитуды и смещения входных сигналов SIN/COS относительно модели контура. При включении адреса каналов автоматически переключаются на

предустановки детектора (см. таблицы выше). При выключении (**Detector Off**) адреса каналов и настройки **восстанавливаются** из ранее сохранённых значений.

Подробное описание процедуры настройки — в разделе Режим детектора.

*Последнее обновление **3 июл. 2026 г.***

Режим детектора

Режим детектора в GraphView предназначен для настройки амплитуды и смещения входных сигналов датчика относительно опорных сигналов математической модели контура. При включении режим загружает в CPU1 специальную программу, которая обеспечивает доступ к сигналам SIN/COS — как входным (с АЦП), так и виртуальным (с модели) — через разделяемую RAM.

Включение и выключение

При нажатии кнопки **Detector On**:

1. Отключаются CPU1 и CPU2 (регистр Mode_config, адрес 71).
2. Настраивается разделяемая RAM (AFE_config, адрес 70).
3. В память CPU1 (адреса 512–1023) загружается программа `detector_asm/cpu1_data.hex`.
4. CPU1 включается.
5. Адреса каналов GraphView автоматически переключаются на предустановки детектора (SIN/COS входные и виртуальные сигналы).

При нажатии кнопки **Detector Off**:

1. CPU1 отключается (регистр Mode_config).
2. **Адреса каналов и настройки восстанавливаются** из ранее сохранённых значений — GraphView возвращается к тем регистрам, которые были заданы до включения детектора.

> ⚠ **ПРЕДУПРЕЖДЕНИЕ**
>
> При выключении детектора микросхема остаётся с отключённым CPU. Для восстановления штатной работы перезагрузите микросхему или загрузите рабочие программы CPU.

Предустановки адресов

При включении детектора каналы автоматически настраиваются на следующие адреса (13-битные знаковые данные):

Канал H1

Канал	Легенда	Адрес (LA)	SA	Назначение
0	SIN1_IN	640	641	Входной SIN с АЦП контура 1
1	SIN1_VIRT	648	649	Виртуальный SIN (модель контура 1)
2	COS1_IN	642	643	Входной COS с АЦП контура 1
3	COS2_VIRT	650	651	Виртуальный COS (модель контура 1)

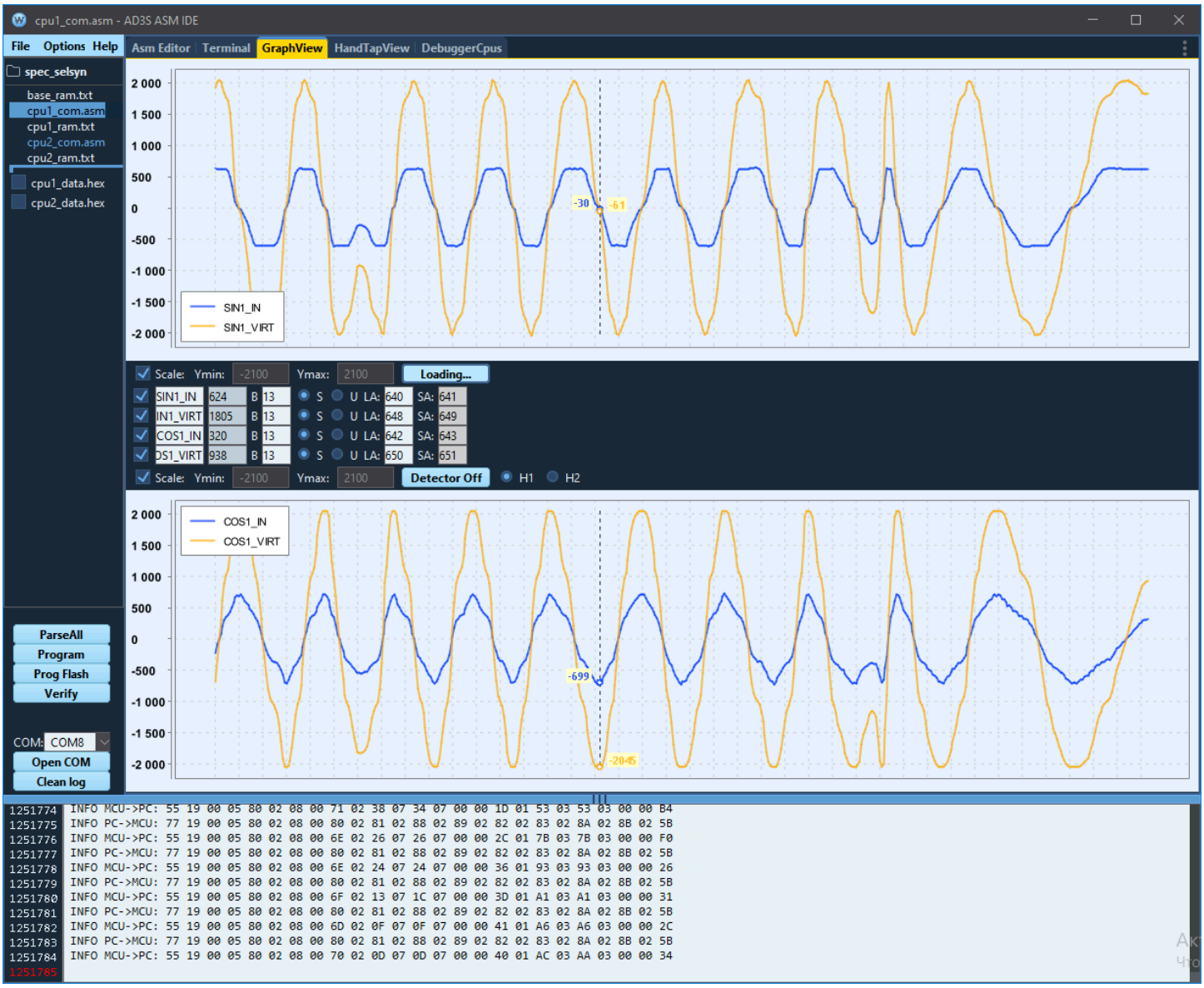
Канал H2

Канал	Легенда	Адрес (LA)	SA	Назначение
0	SIN2_IN	644	645	Входной SIN с АЦП контура 2
1	SIN2_VIRT	652	653	Виртуальный SIN (модель контура 2)
2	COS2_IN	646	647	Входной COS с АЦП контура 2
3	COS2_VIRT	654	655	Виртуальный COS (модель контура 2)

Переключатель **H1/H2** позволяет выбрать контур для анализа.

Настройка амплитуды

До настройки



На рисунке видны входные сигналы с АЦП (SIN_IN, COS_IN) и сигналы математической модели (SIN_VIRT, COS_VIRT). Амплитуда входных сигналов составляет approximately ±600 отсчётов — значительно меньше размаха модели ±2047. Это означает, что коэффициенты преобразования контура ещё не настроены: чувствительность датчика и параметры аналогового тракта не согласованы с масштабом цифровой модели. Несовпадение амплитуд приводит к ошибкам слежения контура — отставанию или опережению фазы, повышенным пульсациям на выходе координаты.

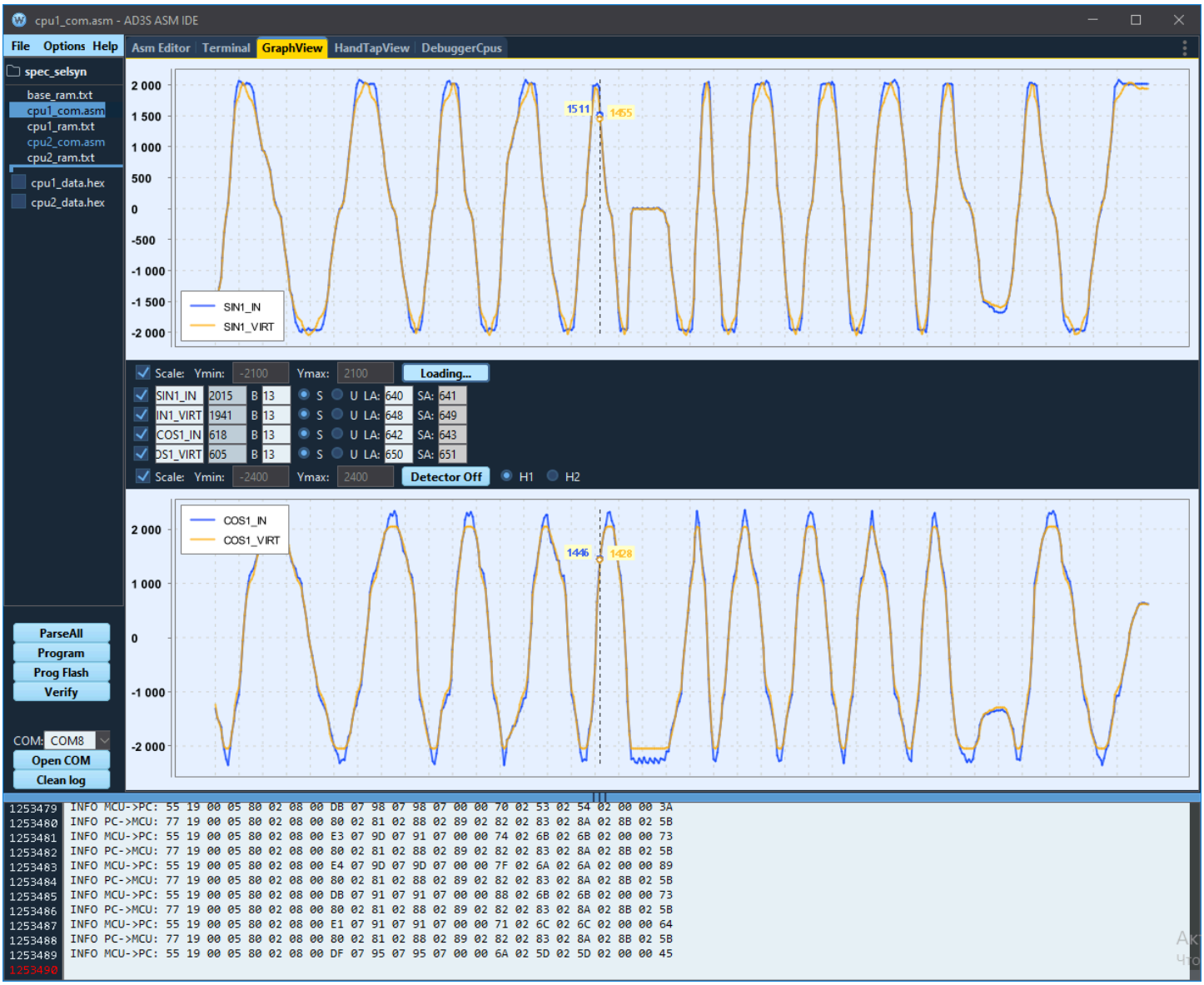
Подбор коэффициентов усиления

Амплитуда входных сигналов приводится к размаху модели коэффициентами KampS и KampC (регистры 0 и 1 контура 1, регистры 32 и 33 контура 2). Коэффициент определяет множитель, на который умножается оцифрованное значение АЦП перед подачей в контур.

В данном случае для согласования амплитуды были записаны значения:

- **KampS** = 3072 (коэффициент ≈ 3.0)
- **KampC** = 3072 (коэффициент ≈ 3.0)

После настройки



После записи коэффициентов KampS = KampC = 3072 входные сигналы по амплитуде практически совпали с размахом модели (±2047). На данном этапе подбирается именно **размах** сигналов — 13-битный формат данных (знаковый, диапазон ±4095) позволяет наблюдать как амплитуду, так и смещение среднего значения без усечки, что важно для следующего этапа настройки.

Настройка смещения

После выравнивания амплитуд может наблюдаться смещение среднего значения входных сигналов относительно сигналов модели — графики SIN_IN и SIN_VIRT совпадают по размаху, но сдвинуты друг относительно друга по вертикали.

Это смещение корректируется коэффициентами KbiasS и KbiasC (регистры 2 и 3 контура 1, регистры 34 и 35 контура 2). Эти коэффициенты поднимают или опускают сигналы математической модели, делая их постоянную составляющую равной смещению входных сигналов. Корректная настройка смещения устраняет статическую ошибку и обеспечивает симметричную работу контура отслеживания.

Порядок настройки

1. Включите режим **Detector On**, выберите канал **H1** или **H2**.
2. Нажмите **Loading...** — наблюдайте входные сигналы и сигналы модели.
3. **Подберите KampS и KampC** — увеличивайте или уменьшайте до совпадения амплитуд входных сигналов с моделью (±2047).
4. **Подберите KbiasS и KbiasC** — скорректируйте смещение так, чтобы средние значения входных сигналов и модели совпадали.
5. Нажмите **Detector Off** — адреса каналов восстановятся, микросхема вернётся к штатному режиму (требуется перезагрузка для восстановления CPU).

Программа CPU1 режима детектора

При включении режима детектора в программную память CPU1 загружается программа из файла `detector_asm/cpu1_data.hex`. Исходный текст — `cpu1_com.asm`.

Принцип работы

Программа работает в бесконечном цикле и выполняет **синхронную выборку** данных конвертеров HAND1 и HAND2. Ключевая особенность: значения записываются в выходные ячейки разделяемой RAM **только в момент отрицательного фронта** сигнала EX_REF (опорный сигнал возбуждения). Это обеспечивает привязку отсчётов к одной и той же фазе сигнала возбуждения, что критично для корректного измерения амплитуды.

Алгоритм программы

```
MAIN_LOOP:
    SET_A_HAND 2                        // Выбираем блок SIN (OutVirtSin)

    // --- Ожидание отрицательного фронта EX_REF ---
    CONST 1 R0                          // Маска бита 0 (EX_REF)
WAIT_NEG_LOOP:
    LOAD32 HAND1_L R1                   // Читаем данные HAND1
    ANL R0 R1                           // Выделяем бит EX_REF
    LOAD PREV_EX_REF R2                 // Предыдущее значение
    STORE PREV_EX_REF R1                // Сохраняем текущее
    SUB R2 R1                           // R1 = prev − current
    EQUAL EXIT_WAIT_NEG R0 R1           // Если prev=1, current=0 → фронт
обнаружен
    JUMP WAIT_NEG_LOOP

EXIT_WAIT_NEG:
    // --- Чтение блока SIN (SET_A_HAND 2) ---
    SET_A_HAND 2
    LOAD32 HAND1_L R1                   // HAND1: VirtualS[12:0], ex,
ArgSin[12:0], ex
    ASR 1 R1                            // Убираем бит 0, сдвигаем
    STORE32 DET1_SIN_L R1               // → IC 640/641: входной SIN контура
1
    ASR 14 R1                           // Сдвигаем ещё на 14
    STORE32 DET1_VSIN_L R1              // → IC 648/649: виртуальный SIN
контура 1

    LOAD32 HAND2_L R1                   // HAND2: аналогично для контура 2
    ASR 1 R1
    STORE32 DET2_SIN_L R1               // → IC 644/645: входной SIN контура
2
    ASR 14 R1
```

```
    STORE32 DET2_VSIN_L R1            // → IC 652/653: виртуальный SIN
контура 2

    // --- Чтение блока COS (SET_A_HAND 6) ---
    SET_A_HAND 6
    LOAD32 HAND1_L R1                 // HAND1: VirtualC[12:0], ex,
ArgCos[12:0], ex
    ASR 1 R1
    STORE32 DET1_COS_L R1             // → IC 642/643: входной COS контура
1
    ASR 14 R1
    STORE32 DET1_VCOS_L R1            // → IC 650/651: виртуальный COS
контура 1

    LOAD32 HAND2_L R1
    ASR 1 R1
    STORE32 DET2_COS_L R1             // → IC 646/647: входной COS контура
2
    ASR 14 R1
    STORE32 DET2_VCOS_L R1            // → IC 654/655: виртуальный COS
контура 2

    JUMP MAIN_LOOP                    // Повторяем цикл
```

Как работает распаковка данных

Конвертер HAND упаковывает два 13-битных значения в одно 28-битное слово:

- **Блок SIN** (SET_A_HAND 2): биты 27:15 — входной SIN (VirtualS), биты 13:1 — виртуальный SIN (ArgSinKontur1)
- **Блок COS** (SET_A_HAND 6): биты 27:15 — входной COS (VirtualC), биты 13:1 — виртуальный COS (ArgCosKontur1)

Программа распаковывает эти данные двумя операциями `ASR`:

1. `ASR 1` — убирает младший бит (служебный флаг), после чего в младших 13 битах остаётся **входной сигнал** (с АЦП, после масштабирования KampS/KampC).
2. `ASR 14` — сдвигает ещё на 14 бит, оставляя в младших 13 битах **сигнал модели** (виртуальный).

Результат записывается в разделяемую RAM через `ST0RE32` (как пара 14-битных слов), откуда GraphView считывает данные по адресам 640–655.

Карта выходных данных в разделяемой RAM

CPU1 addr	IC addr	Метка	Назначение
128–129	640–641	DET1_SIN	Входной SIN контура 1
130–131	642–643	DET1_COS	Входной COS контура 1
132–133	644–645	DET2_SIN	Входной SIN контура 2
134–135	646–647	DET2_COS	Входной COS контура 2
136–137	648–649	DET1_VSIN	Виртуальный SIN контура 1
138–139	650–651	DET1_VCOS	Виртуальный COS контура 1
140–141	652–653	DET2_VSIN	Виртуальный SIN контура 2
142–143	654–655	DET2_VCOS	Виртуальный COS контура 2

Значения обновляются **один раз за период сигнала возбуждения** — строго в момент отрицательного фронта EX_REF.

Условие корректной работы: синхронизация EX_REF

> ⚠ **WARNING**
>
> Детектор корректно измеряет амплитуду только при условии, что опорный сигнал EX_REF **синхронизирован** с реальным сигналом возбуждения датчика. В противном случае показания будут искажены и не будут отражать реальную амплитуду входных сигналов.

Программа CPU1 захватывает данные строго в момент отрицательного фронта EX_REF. Это означает, что точка выборки на синусоиде возбуждения определяется именно фазой EX_REF. Если EX_REF не соответствует реальной фазе возбуждения:

- Выборка происходит в «неправильный» момент периода возбуждения
- Измеренная амплитуда будет меньше реальной (выборка не на пике)
- Или сигнал окажется смещённым (выборка на склоне, а не на экстремуме)

Для обеспечения корректной синхронизации EX_REF:

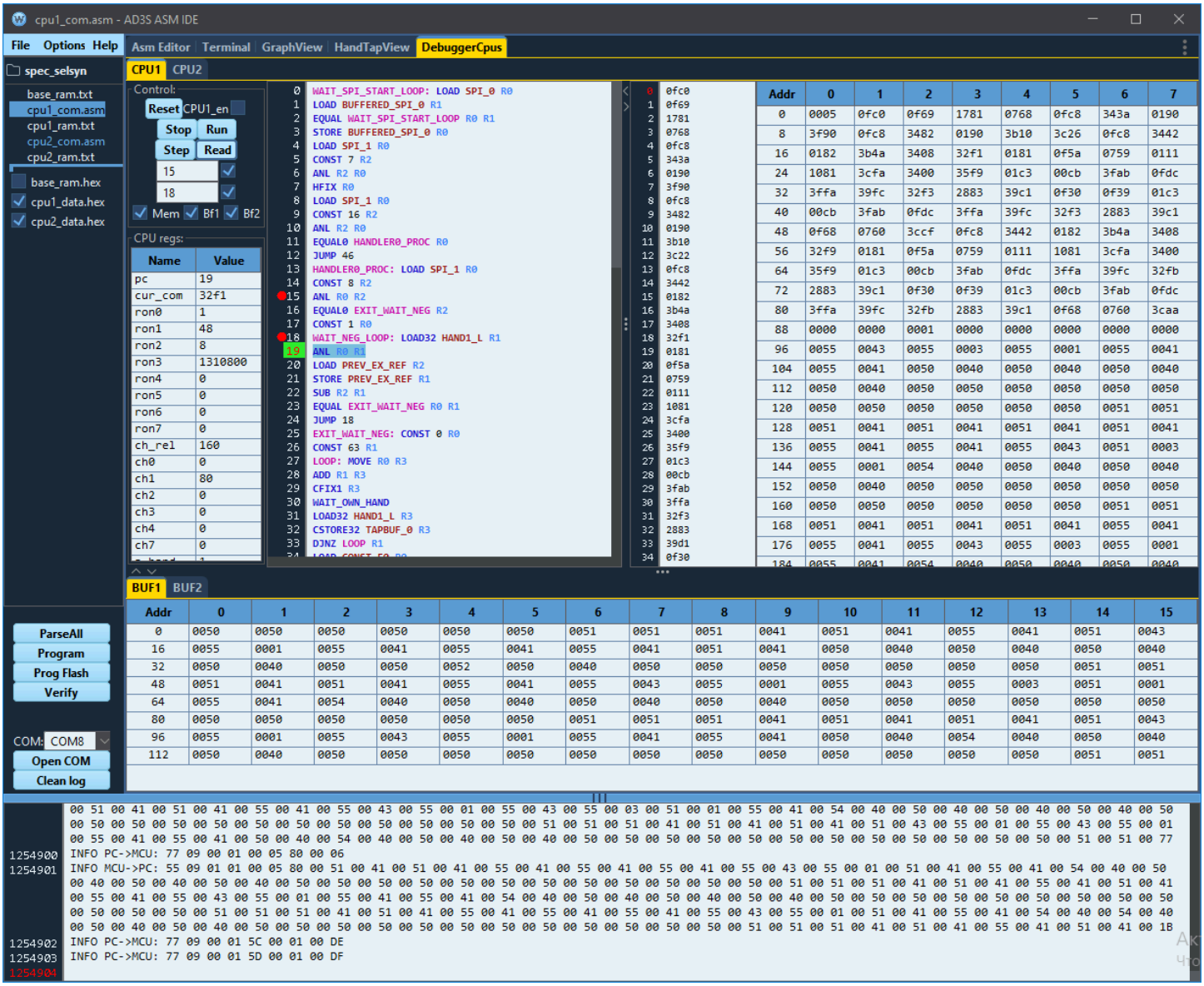
1. **Блок восстановления опорной частоты включён** — микросхема сама формирует EX_REF из входного сигнала, обеспечивая правильную фазу (подробнее в разделе Восстановление возбуждения). Это **рекомендуемый режим** для работы с детектором.

2. **Блок восстановления выключен** — EX_REF подаётся извне. В этом случае пользователь должен **самостоятельно обеспечить точное фазирование** внешнего опорного сигнала относительно реального возбуждения датчика. Неточное фазирование приведёт к ошибкам измерения амплитуды и смещения.

*Последнее обновление **3 июл. 2026 г.***

Отладчик CPU

Вкладка **Debugger CPUs** предназначена для пошаговой отладки программ микровычислителей CPU1 и CPU2 микросхемы. Отладчик позволяет ставить точки останова, выполнять программу по шагам, просматривать и изменять состояние регистров и памяти CPU в реальном времени.



Расположение элементов

Окно отладчика разделено на две части по вертикали:

- **Верхняя панель** — вкладки **CPU1** и **CPU2**, каждая содержит панель отладки соответствующего микровычислителя.
- **Нижняя панель** — вкладки **BUF1** и **BUF2**, отображающие содержимое буферной памяти (по 128 слов в шестнадцатеричном виде).

Панель каждого CPU состоит из трёх колонок:

- **Левая колонка** — кнопки управления, адреса точек останова, таблица регистров CPU.
- **Центральная колонка** — листинг программы на ассемблере (синтаксическая подсветка, только чтение) и шестнадцатеричные коды инструкций.
- **Правая колонка** — память данных CPU (256 слов, 32×8).

Кнопки управления

Кнопка	Действие
Reset	Отключает CPU, записывает адреса точек останова и команду RUN в регистры P1BG_ctrl / P1BG_data, активирует через `tap_event`, заново включает CPU. После — считывает регистры из микросхемы.
Stop	Записывает команду STOP (биты command = 01) в P1BG_ctrl и активирует `tap_event`. CPU останавливается на текущей инструкции.
Run	Записывает команду RUN (биты command = 00) — CPU выполняет программу до следующей активной точки останова.
Step	Записывает команду STEP (биты command = 10) — CPU выполняет одну инструкцию и останавливается.
Read	Считывает регистры CPU из микросхемы. При установленных флажках Mem/Bf1/Bf2 — также считывает память CPU и буферы.

Точки останова (Breakpoints)

Микросхема поддерживает до **2 точек останова** на каждый CPU. Точки останова задаются двумя парами полей:

Поле	Описание
Address 1 + **ena**	Адрес (0–511) и флаг включения точки останова 1. Записывается в биты [8:0] и бит [9] регистра P1BG_ctrl.
Address 2 + **ena**	Адрес (0–511) и флаг включения точки останова 2. Записывается в биты [8:0] и бит [9] регистра P1BG_data.

Установка точек останова мышью

Клик на **полосе номеров строк** (gutter) в листинге ассемблера устанавливает или снимает точку останова на этой строке. Точка останова отображается в виде **красного кружка** слева от номера строки. Максимум — 2 точки; при установке третьей самая старая удаляется.

Текущая позиция программы (PC) подсвечивается **зелёной полосой** в gutter, а номер строки выделяется красным цветом.

Принцип работы точек останова

Точки останова реализованы аппаратно в микросхеме. Когда CPU достигает адрес, совпадающий с `stop1_pc` или `stop2_pc` (при активном соответствующем флаге `stopX_ena`), микровычислитель останавливается, и в регистре статуса устанавливается флаг `DBG_STOP`.

Команда передаётся в микросхему через регистры P1BG_ctrl (адрес 92 для CPU1, 94 для CPU2) и P1BG_data (адрес 93, 95):

Биты	Поле	Назначение
12:11	command	00 = RUN (до точки останова), 01 = STOP, 10 = STEP

Биты	Поле	Назначение
10	tap_event	Переход 0→1 активирует команду
9	stop1_ena	Разрешение точки останова 1
8:0	stop1_pc	Адрес точки останова 1

Регистр P1BG_data содержит аналогичные поля для второй точки останова и поле `sel_reg` для выбора считываемых регистров CPU.

Регистры CPU

После нажатия **Read** (или автоматически после Reset/Step) в левой таблице отображаются регистры микровычислителя:

Основные регистры

Регистр	Разрядность	Описание
pc	9 бит	Счётчик команд (Program Counter). Текущая позиция выполнения программы. Подсвечивается зелёной полосой в листинге.
cur_com	14 бит	Текущая инструкция (отображается в шестнадцатеричном виде).
ron0 — **ron5**	28 бит, знаковые	Регистры общего назначения R0–R5. Состоят из двух 14-битных слов.
ron6, **ron7**	56 бит, знаковые	Расширенные регистры R6–R7. Состоят из четырёх 14-битных слов.

Системные регистры и каналы

Регистр	Разрядность	Описание
stp	1 бит	Флаг пошагового режима
stop	1 бит	Флаг останова отладчиком (DBG_STOP)
a_hand	3 бит	Адрес мультиплексора HAND
ch_rel	9 бит	Состояние канального релиза
ch0 — **ch7**	различные	Состояние каналов микровычислителя

Память и буферы

Память данных CPU (правая колонка)

Отображает 256 слов (32 строки × 8 столбцов) памяти данных CPU. Читается с адреса 512 для CPU1 или 1024 для CPU2. Для отображения установите флажок **Mem** и нажмите **Read**.

Буферы BUF1 и BUF2 (нижняя панель)

Каждый буфер отображает 128 слов (8 строк × 16 столбцов) в шестнадцатеричном виде:

- **BUF1** — адрес IC 768
- **BUF2** — адрес IC 1280

Для чтения установите флажки **Bf1** и/или **Bf2** и нажмите **Read**.

Связь с редактором ассемблера (Asm Editor)

Листинг программы в центре панели отладчика — **только для чтения**. Непосредственно в отладчике редактировать код нельзя.

Для изменения программы:

1. Откройте вкладку **Asm Editor** (редактор ассемблера).
2. Внесите изменения в исходный код.
3. Нажмите **Save** (сохранить файл).
4. При сохранении в Asm Editor программа автоматически парсится, и **обновлённый листинг передаётся в отладчик** DebuggerCpus — синтаксически подсвеченный текст и шестнадцатеричные коды обновляются.

Таким образом, цикл отладки выглядит так: **редактирование в Asm Editor → сохранение → автоматическое обновление в Debugger → Reset/Step/Run для загрузки и выполнения**.

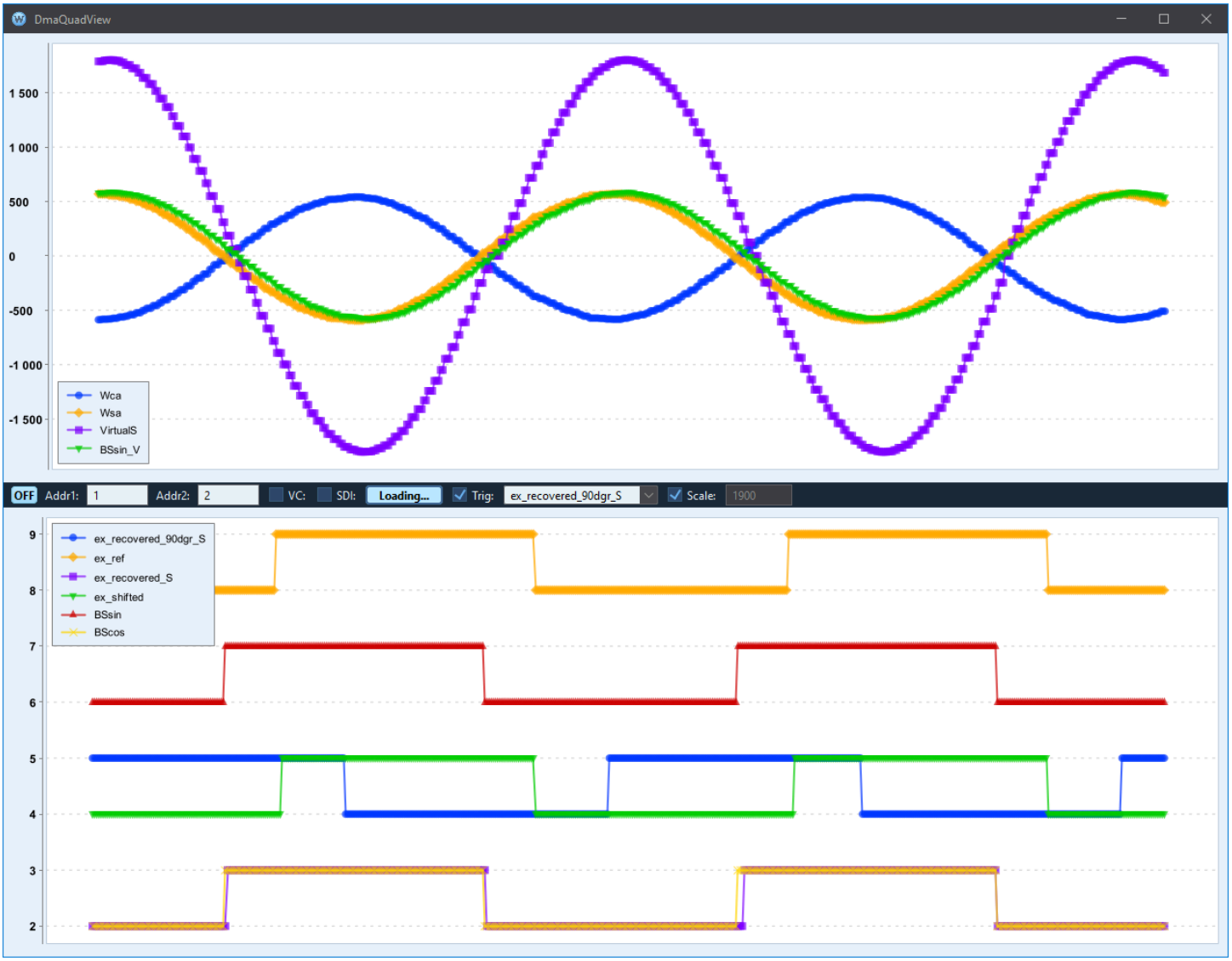
Типовой порядок работы

1. Откройте вкладку **Debugger CPUs**.
2. При необходимости отредактируйте программу в **Asm Editor** и сохраните — листинг обновится в отладчике.
3. Установите точки останова кликом на gutter (красные кружки) или введите адреса вручную в поля **Address 1/2** и включите флажки **ena**.
4. Нажмите **Reset** — CPU перезагрузится с точками останова и начнёт выполнение (команда RUN).
5. При достижении точки останова CPU остановится. Нажмите **Read** для просмотра регистров.
6. Используйте **Step** для пошагового выполнения.
7. Нажмите **Run** для продолжения до следующей точки останова.
8. Для просмотра памяти данных установите флажок **Mem** и нажмите **Read**.

*Последнее обновление **3 июл. 2026 г.***

ParallelSpiView (параллельный вывод)

Окно **ParallelSpiView** (меню → Options → ParallelSpiView) предназначено для диагностики работы микросхемы 5400TP065A-022 в **режиме параллельной выдачи данных** конвертера HAND. В этом режиме микросхема выдаёт данные через 7-проводный параллельный интерфейс (7 сигналов данных), что позволяет принимать **неограниченный поток данных** в реальном времени. В отличие от анализатора HandTap, который захватывает ограниченный буфер (192 отсчёта) через разделяемую RAM, ParallelSpiView обеспечивает непрерывное отображение двух каналов одновременно.



Принцип работы

Микросхема 5400TP065A-022 имеет режим параллельной выдачи, при котором данные конвертера HAND выдаются через 7-проводный параллельный интерфейс (7 сигналов данных). На стороне микроконтроллера захват выполняется в режиме octal-DMA по 8 линиям, одна из которых холостая (микросхема выдаёт только 7 сигналов). Каждые 16 тактовых циклов формируется один набор из четырёх 16-битных слов, которые на стороне ПК объединяются в два 28-битных значения — по одному на каждый канал (Addr1 и Addr2).

Ключевое отличие от HandTap:

Параметр	HandTap	ParallelSpiView
Способ передачи	Разделяемая RAM + SPI	Параллельный вывод (7 сигналов)
Объём данных	192 отсчёта (фиксированный буфер)	Неограниченный поток (циклическое чтение)
Количество каналов	1 (HAND1 или HAND2)	2 одновременно (Addr1 и Addr2)
Обновление	По запросу (кнопка)	Непрерывное (циклический опрос)
Команда МК	`READ_HANDTAP` (0x0E)	`SET_DMA_QUAD` (0x11)

Расположение элементов

Окно ParallelSpiView открывается как отдельное плавающее окно и разделено на три области:

```
+-------------------------------------------------------------+
|  Верхний график (Values)                                    |
|  Многоразрядные числовые сигналы каналов Addr1 и Addr2      |
+-------------------------------------------------------------+
```

```
|  Панель управления:                                              |
|  [ON] Addr1:[▼] Addr2:[▼] [VC] [SDI]                            |
|  [Loading...] [Trig:▼signal] [Scale] [____]                     |
+-----------------------------------------------------------------+
|  Нижний график (Signals)                                         |
|  Одноразрядные цифровые сигналы (exi, ex_ref, BSsin и пр.)      |
+-----------------------------------------------------------------+
```

- **Верхний график (Values)** — отображение многоразрядных числовых сигналов обоих каналов (например, sin, cos, координата, скорость). По горизонтали — номер отсчёта, по вертикали — значение.
- **Панель управления** — строка с кнопками и настройками (см. ниже).
- **Нижний график (Signals)** — отображение одноразрядных цифровых сигналов (например, `exi`, `exi_recovered`).

На обоих графиках доступен курсор: клик левой кнопкой устанавливает вертикальную пунктирную линию, на пересечении которой с каждым сигналом отображается кружок и числовое значение. Правый клик убирает курсор.

Органы управления

Элемент	Назначение
ON / OFF	Включение/выключение режима параллельной выдачи. Кнопка работает как toggle — при нажатии **ON** выполняется процедура включения (см. ниже), кнопка меняет текст на **OFF**. Повторное нажатие выключает режим и возвращает микросхему в исходное состояние.
Addr1	Раскрывающийся список (0–7) для выбора адреса блока данных первого канала. Каждый пункт показывает номер адреса и список сигналов через запятую (например, «1 - Wca, BScos, ex_shifted, Wsa, BSsin, ex_ref»). Определяет, какие поля будут декодированы из первого потока. По умолчанию — 1 (SinCos).

Элемент	Назначение
Addr2	Раскрывающийся список (0–7) для выбора адреса блока данных второго канала. По умолчанию — 2 (OutVirtSin).
VC	Флажок управления сигналом VC (Voltage Control). При установке отправляется команда `SET_VC(1)` на МК, при снятии — `SET_VC(0)`. Управляет коммутацией каналов в параллельном потоке: при включённом VC потоки d/b назначаются на Addr1, а c/a — на Addr2; при выключенном — наоборот.
SDI	Флажок управления сигналом SDI (SPI Data Input). При установке отправляется `SET_SDI(1)`, при снятии — `SET_SDI(0)`. Управляет режимом SPI-интерфейса микросхемы.
Loading... / Stop	Запуск/остановка циклического чтения данных. При нажатии **Loading...** на МК отправляется команда с адресами Addr1/Addr2, затем начинается непрерывный циклический опрос данных. Кнопка меняет текст на **Stop**.
Trig	Флажок режима синхронизации (triggered). При установке данные обрезаются от первого положительного фронта выбранного одноразрядного сигнала (из раскрывающегося списка).
(раскрывающийся список)	Выбор одноразрядного сигнала для синхронизации (Trig). Заполняется автоматически после первого захвата данных на основе доступных 1-битных полей.
Scale	Флажок авто-масштабирования оси Y верхнего графика. При установке диапазон определяется автоматически по амплитуде данных (с округлением до сотен). Поле справа отображает текущее значение.

Элемент	Назначение
(поле Y-диапазона)	Текстовое поле, диапазон оси Y верхнего графика (± значение). Доступно только когда Scale снят.

Включение и выключение режима

Включение (ON)

При нажатии кнопки **ON** выполняется следующая последовательность:

1. **Настройка C1 ResCntrl** — в регистре C1ResCntrl сбрасывается бит `Enc_en`, устанавливаются биты `SPI_ext_en`, `Vel_from_cpu`, `Coord_from_cpu`.
2. **Настройка C2 ResCntrl** — аналогично в регистре C2ResCntrl: `Enc_en` = 0, `SPI_ext_en` = 1, `Vel_from_cpu` = 1, `Coord_from_cpu` = 1.
3. **Отключение CPU1** — в регистре Mode_config (адрес 71) сбрасывается бит `CPU1_en`.
4. **Загрузка программы** — в программную память CPU1 (начиная с адреса 512) загружается скомпилированная программа `dma_asm/cpu1_data.hex`.
5. **Включение CPU1** — в регистре Mode_config устанавливается бит `CPU1_en`.

После успешного включения кнопка меняет текст на **OFF**, кнопка **Loading...** становится доступной.

Выключение (OFF)

При нажатии кнопки **OFF**:

1. В регистре C1 ResCntrl сбрасываются биты `SPI_ext_en`, `Vel_from_cpu`, `Coord_from_cpu`.
2. В регистре C2 ResCntrl сбрасываются биты `SPI_ext_en`, `Vel_from_cpu`, `Coord_from_cpu`.
3. В регистре Mode_config сбрасывается бит `CPU1_en` — CPU1 останавливается.

> ⚠ **ПРЕДУПРЕЖДЕНИЕ**

> При выключении режима параллельной выдачи микросхема остаётся с отключённым CPU1. Для восстановления штатной работы необходимо перезагрузить микросхему или вручную загрузить рабочие программы CPU.

Процесс захвата данных

После нажатия **Loading...** выполняется следующая последовательность:

1. Из полей **Addr1** и **Addr2** формируется составной адрес: `addr2 << 8 | addr1`.
2. На МК отправляется команда `SET_DMA_QUAD` (0x11) с подкомандой **1** и данными `[addr1, addr2]`. МК конфигурирует режим параллельной выдачи микросхемы для захвата двух потоков данных.
3. Выдерживается пауза 200 мс для стабилизации.
4. Запускается **циклический опрос** — в бесконечном цикле на МК отправляется команда `SET_DMA_QUAD` с подкомандой **2** (чтение). МК считывает данные из параллельного интерфейса микросхемы и возвращает их на ПК.
5. Ответы обрабатываются таймером рендеринга (период опроса 40 мс). При получении нового ответа данные декодируются и отображаются на графиках.

Декодирование данных

Данные приходят от МК в упакованном виде. Декодирование выполняется следующим образом:

1. Из ответа удаляются 8 байт заголовка и 1 байт контрольной суммы.
2. Каждый байт разделяется на два 4-битных полубайта (nibble): старший и младший.
3. Каждая группа из 16 полубайтов объединяется в четыре 16-битных слова (a, b, c, d) — по одному биту из каждого полубайта:
   - Бит 0 каждого полубайта → поток **a**
   - Бит 1 → поток **b**
   - Бит 2 → поток **c**
   - Бит 3 → поток **d**
4. Пары потоков объединяются в 28-битные значения:
   - `val = (stream_hi & 0xFFFF) | ((stream_lo & 0xFFF) << 16)`

5. Коммутация потоков зависит от флага **VC**:

Флаг VC	Addr1 — потоки	Addr2 — потоки
Выключен	c (старшие) + a (младшие)	d (старшие) + b (младшие)
Включён	d (старшие) + b (младшие)	c (старшие) + a (младшие)

6. Каждое 28-битное значение разбирается на именованные поля в соответствии с адресом (Addr1 или Addr2) по таблице `HandValueSetting`.

Адреса блоков данных

Поля **Addr1** и **Addr2** определяют, какие данные считываются из конвертера HAND. Адресация соответствует команде микровычислителя `SET_A_HAND` (подробнее в разделе CPU). Адреса одинаковы для обоих каналов:

Addr	Блок	Поля на графике
0	**Coord** (Координата)	`FullAngle` (28 бит)
1	**SinCos** (АЦП)	`Wca` (12 бит, зн.), `BScos`, `ex_shifted`, `Wsa` (12 бит, зн.), `BSsin`, `ex_ref`
2	**OutVirtSin**	`VirtualS` (13 бит, зн.), `ex_recovered_S`, `BSsin_V` (13 бит, зн.), `ex_recovered_90dgr_S`
3	**ErrAmpMetric**	`Amp_metric` (12 бит), `Err_metric` (16 бит, зн.)
4	**Vel** (Скорость)	`FullVel` (28 бит, зн.)
5	**PhiModel**	`PhiC` (14 бит, зн.), `PhiS` (14 бит, зн.)

Addr	Блок	Поля на графике
6	**OutVirtCos**	`VirtualC` (13 бит, зн.), `ex_recovered_C`, `BScos_V` (13 бит, зн.), `ex_recovered_90dgr_C`
7	**Stat** (Статус)	`STAT` (16 бит)

Многоразрядные поля (размер > 1 бит) отображаются на верхнем графике **Values**. Одноразрядные поля — на нижнем графике **Signals**. Данные обоих адресов (Addr1 и Addr2) отображаются одновременно на одних графиках.

Режим синхронизации (Trig)

При установленном флаге **Trig** полученные данные обрезаются от первого обнаруженного положительного фронта выбранного одноразрядного сигнала (из раскрывающегося списка). Алгоритм ищет переход 0→1 в данных выбранного сигнала, и все данные (числовые и одноразрядные) обрезаются, оставляя только часть после фронта. Это позволяет привязать осциллограмму к определённой фазе сигнала.

Распространённый вариант — синхронизация по сигналу `ex_ref` (бит возбуждения) для анализа поведения контура синхронно с возбуждением.

Типичное использование

1. Подключите микросхему и убедитесь в связи (Test Board → Find IC).
2. Откройте **Options → ParallelSpiView** — появится отдельное окно.
3. Нажмите **ON** — выполнится процедура включения (настройка регистров, загрузка программы CPU1).
4. В раскрывающемся списке **Addr1** выберите «1 - Wca, BScos, ex_shifted, Wsa, BSsin, ex_ref» для просмотра сигналов АЦП.
5. В раскрывающемся списке **Addr2** выберите «4 - FullVel» для одновременного просмотра скорости.
6. При необходимости включите **VC** или **SDI** для управления коммутацией.
7. Нажмите **Loading...** — начнётся непрерывный захват и отображение данных.

8. Используйте курсор (клик на графике) для точного отсчёта значений.
9. Для привязки к фазе возбуждения включите **Trig** и выберите сигнал `ex_ref`.
10. По окончании нажмите **Stop**, затем **OFF** для выхода из режима параллельной выдачи.

Программа CPU1 режима параллельной выдачи

При включении режима в программную память CPU1 загружается программа из файла `dma_asm/cpu1_data.hex`. Она не записывает данные в разделяемую RAM. Вместо этого CPU1 работает в тесном цикле, непрерывно переключая адрес конвертера HAND и обеспечивая поток данных через параллельный интерфейс микросхемы. Адреса чтения/записи задаются через SPI-регистры МК, что позволяет динамически менять привязку каналов без перезагрузки программы CPU1.

Исходный текст программы — файл `cpu1_com.asm`:

```
CLR R0              ; R0 = 0 (аргумент для HFIX — фиктивный адрес)
CONST 4 R3          ; R3 = 4 (аргумент для HFIX — фиктивный адрес)
LOOP: LOAD32 SPI_0 R1   ; Прочитать адрес чтения из SPI_0 (Addr1 из UI)
      LOAD32 SPI_2 R2   ; Прочитать адрес записи из SPI_2 (Addr2 из UI)
      HFIX R1            ; Выбрать адрес чтения HAND (из SPI_0)
      WAIT_EXT_HAND      ; Ожидать обновление данных
      LOAD32 HAND1_L R4  ; Прочитать 28-битное значение HAND → R4
      HFIX R0            ; Выбрать фиктивный адрес (R0 = 0)
      STORE32 HAND1_L R4 ; Записать значение в HAND (поток A/B)
      HFIX R2            ; Выбрать адрес чтения HAND (из SPI_2)
      LOAD32 HAND1_L R4  ; Прочитать 28-битное значение HAND → R4
      HFIX R3            ; Выбрать фиктивный адрес (R3 = 4)
      STORE32 HAND1_L R4 ; Записать значение в HAND (поток C/D)
      JUMP LOOP          ; Повторить
```

Описание алгоритма

Программа выполняется в бесконечном цикле (14 инструкций). Каждая итерация:

1. **Читает адрес чтения** из `SPI_0` (регистр IC 760, младший байт — это Addr1, заданный пользователем в UI).

2. **Читает адрес записи** из `SPI_2` (задаётся МК при конфигурации — Addr2 или другой адрес для второго потока).
3. **Выбирает адрес чтения HAND** (HFIX R1), ожидает обновление данных через `WAIT_EXT_HAND`, считывает 28-битное значение в R4.
4. **Переключает на фиктивный адрес** (HFIX R0 = 0), записывает значение через `STORE32` — формирует данные для потока A/B через параллельный интерфейс.
5. **Выбирает второй адрес чтения** (HFIX R2), считывает очередное 28-битное значение.
6. **Переключает на фиктивный адрес** (HFIX R3 = 4), записывает значение — данные для потока C/D.

Таким образом, за каждый проход цикла через параллельный интерфейс проходят данные от двух адресов конвертера, упакованные в параллельный поток.

> **К СВЕДЕНИЮ**
>
> Адреса чтения/записи не жёстко заданы в программе — они поступают через SPI-регистры от МК. Это позволяет менять привязку каналов на лету: пользователь выбирает Addr1/Addr2 в UI, МК записывает их в SPI-регистры, и CPU1 немедленно начинает использовать новые адреса без перезагрузки программы.

Карта памяти CPU1

Программа использует следующее распределение памяти (адреса указаны в 14-битных словах):

Адреса	Назначение
0–14	Программная память (код `cpu1_com.asm`, 15 инструкций)
15–95	Не используется (нули)
96–223	Буфер `TAPBUF_0` (128 слов) — не используется в данном режиме
224–229	ROM-константы
230	`CONST_50` = 0x50 (80)

Адреса	Назначение
231	`CONST_7F` = 0x7F (127)
232–234	Зарезервировано
235	`PREV_EX_REF` — предыдущее значение бита EXI
236	`TO_OUT_READY` — флаг готовности данных
237	`BUFFERED_SPI_0` — буфер SPI
238–239	Константы (50, 7F)
240–247	`EXTCPU_0` … `EXTCPU_7` — регистры внешнего CPU
248–251	`SPI_0` … `SPI_3` — данные SPI от МК
252–255	`HAND1_L`, `HAND1_H`, `HAND2_L`, `HAND2_H` — данные конвертера HAND
256–319	`BUF_L_0`/`BUF_H_0` … `BUF_L_31`/`BUF_H_31` — буфер 32 записи
320–383	`BUF_L_32`/`BUF_H_32` … `BUF_L_63`/`BUF_H_63` — буфер (продолжение)

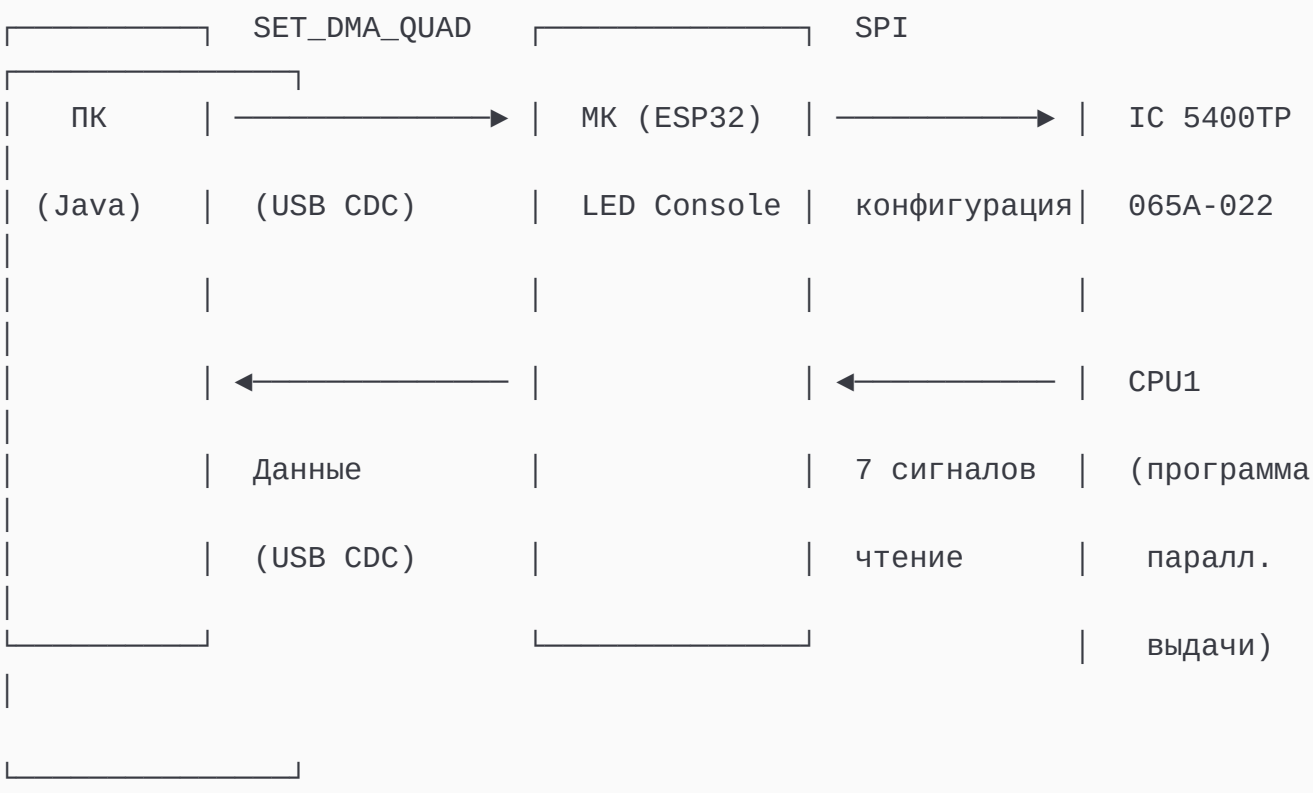
Состав файлов проекта dma_asm

Проект `dma_asm` в директории `dist/risc_programs/dma_asm/` содержит:

Файл	Тип	Описание
`cpu1_com.asm`	ASM	Исходный текст программы CPU1 (15 инструкций)
`cpu1_data.hex`	HEX	Скомпилированный образ CPU1 (512 × 14-битных слов) — загружается по адресу 512
`cpu1_ram.txt`	TXT	Карта памяти CPU1 с символьными именами

Файл	Тип	Описание
`cpu2_com.asm`	ASM	Исходный текст программы CPU2 (не используется в текущей реализации)
`cpu2_data.hex`	HEX	Скомпилированный образ CPU2
`cpu2_ram.txt`	TXT	Карта памяти CPU2
`base_ram.txt`	TXT	Начальные значения базовой RAM микросхемы (96 × 16-битных слов)
`base_ram.hex`	HEX	Скомпилированный образ базовой RAM
`mode.config`	CFG	Конфигурация загрузки: типы файлов, размеры, адреса

Взаимодействие CPU1, МК и ПК — полная диаграмма



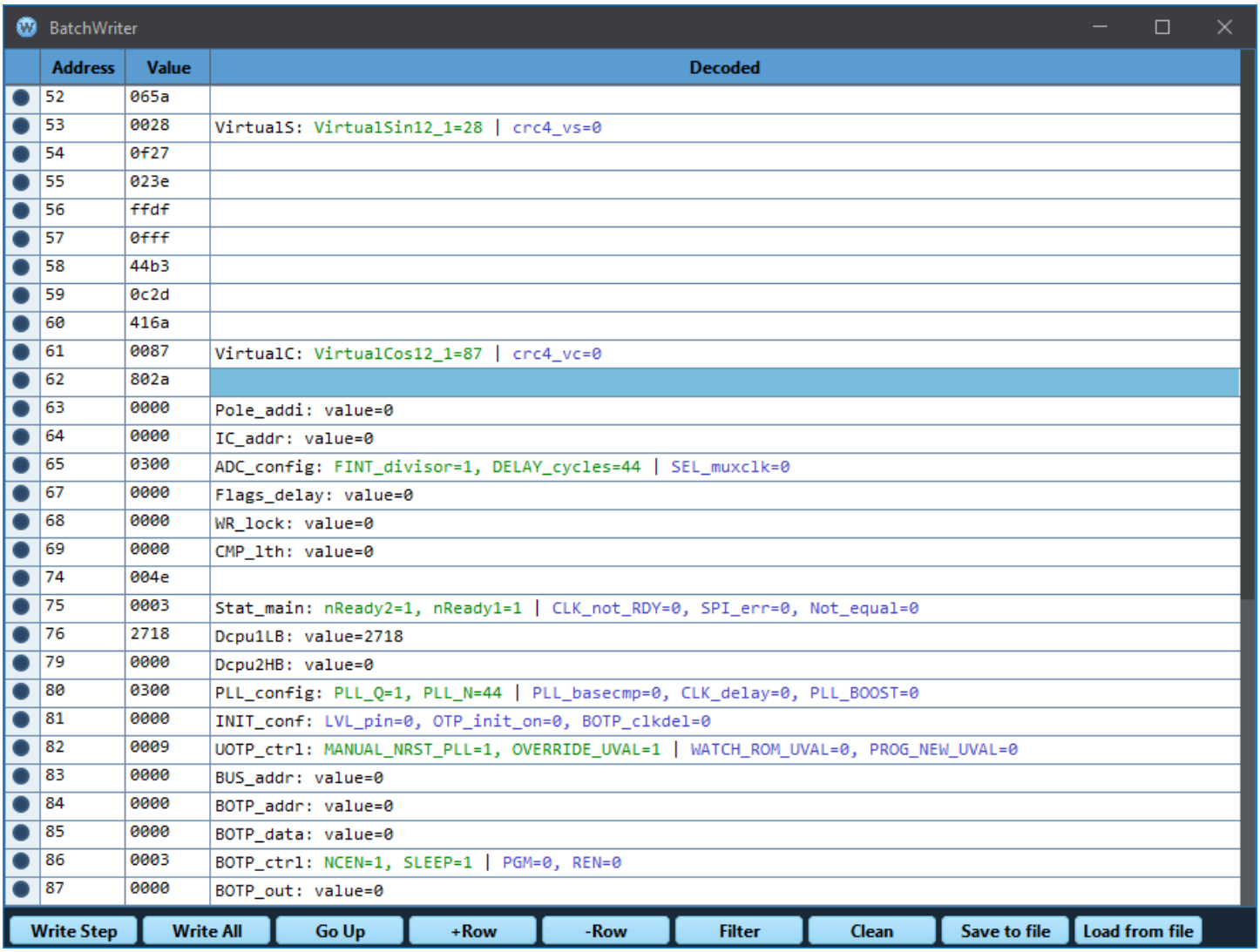
1. **ПК** отправляет команду `SET_DMA_QUAD` (0x11) на МК с подкомандой 1 и адресами `[addr1, addr2]`.
2. **МК** конфигурирует режим параллельной выдачи микросхемы через SPI.
3. **CPU1** (программа `cpu1_com.asm`) выполняет непрерывный цикл чтения/записи данных конвертера HAND, обеспечивая поток данных через параллельный интерфейс.
4. **МК** циклически отправляет `SET_DMA_QUAD` с подкомандой 2, считывает параллельные данные из микросхемы.
5. **МК** передаёт данные на **ПК** через USB CDC.
6. **ПК** декодирует 4 параллельных потока, объединяет в два 28-битных значения, разбирает на поля и отображает на графиках.

> **К СВЕДЕНИЮ**
>
> Режим параллельной выдачи микросхемы 5400TP065A-022 обеспечивает непрерывный поток данных без ограничений на количество отсчётов. В текущей реализации ПО каждый цикл опроса возвращает один кадр данных, который немедленно отображается на графиках. Обновление происходит с периодом таймера рендеринга (40 мс), что обеспечивает частоту обновления около 25 кадров/с.

*Последнее обновление **3 июл. 2026 г.***

BatchWriter (пакетная запись)

Окно **BatchWriter** (меню → Options → BatchWriter) предназначено для подбора и проверки настроек микросхемы по адресам и значениям регистров. При вводе адреса и данных столбец **Decoded** автоматически отображает расшифровку в соответствии с битовыми масками регистров — имя регистра, значения отдельных полей, ненулевые и нулевые биты. Подготовленную таблицу можно сохранить в файл и передать для анализа. При необходимости настройки можно записать в микросхему — последовательно строка за строкой или все разом.



Расположение элементов

Окно разделено на две части:

- **Таблица** — список строк, каждая из которых содержит адрес регистра, значение и расшифровку записанных битов/полей.
- **Панель кнопок** — строка управления в нижней части окна.

Столбцы таблицы

Столбец	Назначение
(радиокнопка)	Указатель текущей строки. Клик по радиокнопке делает строку текущей — именно она будет записана при нажатии Write Step. Подсвечивается белым фоном.
Address	Адрес регистра микросхемы (десятичный или шестнадцатеричный, например `71` или `0x47`).
Value	Значение для записи (десятичное или шестнадцатеричный, например `5` или `0x05`).
Decoded	Автоматическая расшифровка: имя регистра, ненулевые поля выделены зелёным, нулевые — синим. Столбец только для чтения.

Органы управления

Кнопка	Назначение
Write Step	Записать текущую строку (отмеченную радиокнопкой) в микросхему и перейти к следующей. Если адрес пустой — запись пропускается, курсор переходит на следующую строку.
Write All	Записать все строки последовательно сверху вниз; строки с пустым или некорректным адресом пропускаются. В процессе текущая (подсвеченная)

Кнопка	Назначение
	строка перемещается по таблице; по завершении курсор остаётся на последней записанной строке.
Go Up	Переместить курсор и радиокнопку на первую строку.
+Row	Вставить пустую строку перед выделенной (или перед текущей, если ничего не выделено). Курсор переходит на новую строку.
-Row	Удалить выделенную строку. Курсор переходит на следующую доступную строку (или на последнюю, если удалена последняя).
Filter	Убрать строки, которые не нужно записывать: совпадающие со значениями регистров по умолчанию или содержащие адреса только для чтения. Удобно для очистки загруженного списка перед записью.
Clean	Очистить таблицу и создать 20 пустых строк.
Save to file	Сохранить таблицу в текстовый файл (адрес, значение, расшифровка — разделены табуляцией).
Load from file	Загрузить таблицу из текстового файла. Помимо собственного формата (адрес, значение) поддерживается файл `base_ram.txt` (три столбца: индекс, значение, имя регистра) — в этом случае расшифровка выполняется по имени регистра.

Типичное использование

1. Откройте **Options** → **BatchWriter** — появится окно с пустой таблицей (20 строк).
2. Заполните столбцы **Address** и **Value** нужными адресами и значениями регистров. Столбец **Decoded** заполняется автоматически.
3. Пустые строки (без адреса) можно использовать как разделители или комментарии — при Write Step они пропускаются.
4. Для записи одной строки — убедитесь, что радиокнопка стоит на нужной строке, и нажмите **Write Step**. После записи курсор автоматически переходит на следующую

строку.

5. Для записи всех строк разом — нажмите **Write All**. Запись идёт последовательно сверху вниз.
6. Для добавления строк — нажмите **+Row** (вставка перед выделенной).
7. Для удаления строки — выделите её и нажмите **-Row**.
8. Чтобы оставить в таблице только строки, реально меняющие конфигурацию, нажмите **Filter** — строки со значениями по умолчанию и «только для чтения» будут удалены.

Также поддерживается **вставка из буфера обмена** (Excel/других таблиц): скопированные ячейки вставляются в выделенную область таблицы.

Формат данных

Адреса и значения можно вводить в десятичном или шестнадцатеричном формате (префикс `0x` или `0X`). Например:

- `71` — десятичный адрес
- `0x47` — шестнадцатеричный адрес (то же самое)
- `0x05` — значение 5

Столбец **Decoded** автоматически показывает:

- Имя регистра (если адрес известен) — например, `Mode_config`
- Значения отдельных полей — например, `CPU1_en=1, CPU2_en=0`
- Ненулевые поля выделяются зелёным цветом, нулевые — синим

Автосохранение

Таблица автоматически сохраняется между сеансами работы приложения в файл `batchwriter.csv` (адрес и значение, разделены `;`). При следующем открытии BatchWriter загруженная таблица восстанавливается автоматически.

*Последнее обновление **3 июл. 2026 г.***

Record (запись данных)

Окно **Record** (вкладка в нижней панели) предназначено для записи потока данных с микросхемы в текстовый файл. В режиме записи контроллер ESP32 циклически считывает заданные пары регистров с интервалом 1 мс и передаёт значения на ПК, где они сохраняются в файл. Позволяет захватывать динамические процессы для последующего анализа.

Расположение элементов

Окно разделено на три части:

- **Верхняя панель** — кнопка записи, поле пути к файлу и кнопка обзора.
- **Центральная панель** — таблица из 9 пар адресов (четный + следующий нечётный).
- **Нижняя панель** — статус записи.

Панель управления

Элемент	Назначение
Rec / Stop	Кнопка запуска и остановки записи. При нажатии **Rec** отправляется команда `START_RECORD` на ESP32, начинается циклический опрос и запись данных в файл. Кнопка меняет текст на **Stop**. Повторное нажатие останавливает запись (`STOP_RECORD`).
Поле пути	Путь к файлу для сохранения. Путь сохраняется между сеансами в `config.properties`. По умолчанию — `rec1.txt`.
...	Кнопка выбора файла через диалог.
Статус	Текстовая метка в нижней части: «Ready» — ожидание, «Recording... N samples» — идёт запись, «Done. N samples written» — запись завершена.

Таблица адресов

Каждая строка задаёт пару регистров для чтения:

Столбец	Назначение
#	Номер пары (1–9).
Addr (even)	Чётный адрес регистра (редактируемое поле).
Addr (odd)	Нечётный адрес (автоматически = even + 1, только для чтения).
14b14	Флажок режима объединения. При установке значения из двух 16-битных регистров объединяются как 14 старших бит + 14 младших бит (2 старших бита каждой части отбрасываются). По умолчанию — объединение 12 старших + 16 младших бит.

Значения по умолчанию

#	Addr (even)	Описание
1	16	
2	48	
3	162	
4	674	
5	676	
6	678	
7	680	

#	Addr (even)	Описание
8	682	
9	684	

При изменении чётного адреса нечётный обновляется автоматически (even + 1). Настройки пар адресов и флажков 14b14 сохраняются между сеансами.

Формат выходного файла

Каждая строка содержит 9 значений, разделённых табуляцией. Значения записываются в шестнадцатеричном формате (7 цифр, с ведущими нулями):

```
0012345\t0023456\t0034567\t...
```

Объединение данных

Для каждой пары ESP32 считывает два 16-битных слова:

- **14b14 выключен** (по умолчанию): `result = (odd[11:0] << 16) | even` — 12 старших бит из нечётного регистра + 16 младших бит из чётного. Итого 28 бит.
- **14b14 включен**: `result = (odd[13:0] << 14) | even[13:0]` — 14 старших бит + 14 младших бит. Итого 28 бит (4 старших бита каждой половины отбрасываются).

Взаимодействие с ESP32

В режиме записи ESP32 выполняет циклический опрос микросхемы по SPI по заданным адресам и передаёт результаты на ПК через UART. Порядок работы:

1. **Rec** — ПК отправляет команду `START_RECORD` с адресами всех 18 регистров (9 пар × 2).
2. ESP32 начинает циклический опрос микросхемы с интервалом 1 мс.
3. При каждом опросе ESP32 считывает 18 регистров и отправляет пакет на ПК.

4. ПК принимает данные через callback, объединяет пары в значения и записывает в файл.
5. **Stop** — ПК отправляет команду `STOP_RECORD`, ESP32 прекращает опрос.

Типичное использование

1. Настройте пары адресов в таблице.
2. Укажите путь к файлу записи.
3. Нажмите **Rec** — начнётся запись данных.
4. При необходимости включите флажки **14b14** для нужных пар (зависит от формата данных в регистрах).
5. Нажмите **Stop** для остановки записи.
6. Полученный файл можно использовать для анализа в MATLAB, Python или других инструментах.

*Последнее обновление **3 июл. 2026 г.***

Программирование OTP-памяти

Микросхема содержит **энергонезависимую память (OTP)** для хранения конфигурации, которая автоматически загружается в регистры при подаче питания или сбросе. После программирования OTP микросхема самостоятельно инициализируется — ПК-приложение для штатной работы больше не требуется.

OTP состоит из двух независимых частей (подробно — в разделе ПЗУ):

- **BOTP** (последовательная, 512 слов) — загружает при старте все конфигурационные регистры и память микровычислителей. Это «слепок» настроек преобразователя (соответствие ячеек и регистров).
- **UOTP** (параллельная, 2 ячейки) — хранит PLL_config и INIT_conf (настройки тактирования и инициализации, описание).

> **ГДЕ НАЖИМАТЬ**
>
> Все кнопки прожига OTP находятся **в одном месте**: вкладка Terminal → правая колонка, панель с заголовком `Actions:`. Других окон для программирования нет. Если вы не видите кнопок `Create BOTP` / `Prog BOTP` / `Prog UOTP` — значит, вы не в панели `Actions:` вкладки Terminal.

Полный сценарий программирования

Пошагово, от подключения до проверки автозагрузки:

1. **Подключите микросхему.** В левой панели (**ConnectPanel**) откройте COM-порт кнопкой **Open COM**, затем найдите микросхему кнопкой **Find IC** в окне **Test Board** (меню → Utilities → Test Board, подробнее).
2. **Настройте регистры** во вкладке **Terminal** так, как они должны быть при старте микросхемы (контур, тактирование, AFE и т.д.). Для BOTP также подготовьте файлы

проекта (`base_ram.txt`, `cpu1_data.hex`, `cpu2_data.hex`) и нажмите **ParseAll** в левой панели.

3. **Write to IC** (панель `Actions:`) — запишите настроенные регистры в микросхему (в оперативную память), чтобы проверить конфигурацию «вживую» до прожига.
4. **Прожиг UOTP без автозагрузки** (тактирование и делитель ПЗУ): выставьте в таблице регистров `PLL_config` и `INIT_conf` с выбранным BOTP_clkdel (Fclk_rom ≤ 1 МГц) и **сбитым битом** `OTP_init_on` → **Create U22B** → **Prog UOTP**. Прожиг проверяется автоматически (в логе `UOTP programmed: ROM VERIFY PASSED …` либо `UOTP ROM VERIFY FAILED`; при неудаче повторите).
5. **Прожиг BOTP** (слепок регистров): **Create BOTP** (из файлов проекта) → **Prog BOTP** → **Verify BOTP** — должны записаться и совпасть **все 512 слов**.
6. **Включение автозагрузки — строго последний шаг:** установите бит `OTP_init_on` (INIT_conf, бит 4) в таблице регистров → **Create U22B** (пересоберите hex с новым INIT_conf!) → **Prog UOTP**.
7. **Финальная проверка:** снимите питание (или подайте сброс) — микросхема должна сама загрузить настройки из OTP и работать без приложения.

> **НЕ ПРОЖИГАЙТЕ OTP_INIT_ON ДО BOTP!**
>
> Бит `OTP_init_on` включает автозагрузку **всех** регистров из BOTP при каждом включении питания. Если прожечь его **до** полного прожига и проверки BOTP (минимум — области регистров конфигурации, слова 504–511: IC_addr … Mode_config), микросхема при каждом включении будет загружать нули из пустой ПЗУ: микровычислители выключены (`Mode_config` = 0), адрес по интерфейсу сброшен (`IC_addr` = 0) — микросхема не работает и не отвечает. Она будет находиться в неизвестном состоянии, и для её возвращения к работоспособности потребуются значительные усилия. Полный порядок программирования — в разделе ПЗУ → Порядок программирования ПЗУ.

> **НАПРЯЖЕНИЕ ПРОЖИГА ПОДАЁТСЯ АВТОМАТИЧЕСКИ**
>
> Напряжение программирования VPP (9,8 В ± 0,3 В) приложение подаёт **самостоятельно** в момент прожига: для BOTP импульс VPP формирует прошивка ESP32 при каждой записи слова, для UOTP — само приложение. **Флажок** `vpp9v` **вручную ставить не нужно** — это отдельный ручной режим подачи VPP для диагностики (со своим предупреждением).

Предварительные условия

Перед прожигом убедитесь, что:

- микросхема **обнаружена** (Find IC нашёл адрес);
- микросхема **не в standby** (флажок **STNDBY** снят);
- сброс **отпущен** (флажок **NRESET** снят);
- для **BOTP** — файлы проекта распарсены (**ParseAll**); для **UOTP** — значения `PLL_config`/`INIT_conf` выставлены в таблице (см. разделы ниже).

> ⚠ **НЕОБРАТИМАЯ ОПЕРАЦИЯ**
>
> Прожиг OTP **необратим** и рассчитан на ограниченное число перезаписей. Любая кнопка `Prog …` требует **двойного подтверждения** — последовательно появятся диалоги «Подтверждение программирования OTP» и «Окончательное подтверждение». Перед прожигом обязательно проверьте конфигурацию и выполните **Verify**.

Программирование BOTP (слепок регистров)

1. Подключите микросхему (**Find IC**) и подготовьте проект: нужные значения регистров сохраните в `base_ram.txt` (кнопка **Store to txt**), данные микровычислителей должны быть в `cpu1_data.hex`/`cpu2_data.hex`, затем нажмите **ParseAll** (левая панель, ConnectPanel).
2. **Create BOTP** (панель `Actions:`) — приложение соберёт файл `rom_BOTP.hex` (512 слов) из **распарсенных файлов проекта** (`cpu1_data.hex` + `cpu2_data.hex` + `base_ram.txt`), а **не из чипа и не из текущей таблицы** регистров. Если файлы не были распарсены — `Create BOTP` ничего не сделает (в логе: «Before parse all files!»).
3. **Prog BOTP** — прожигает слепок в BOTP. Появятся два диалога подтверждения — ответьте «Да» на оба. Приложение запишет 512 слов и автоматически проверит каждое; при несовпадении сообщит об ошибке и остановится.
4. **Verify BOTP** — перечитает BOTP и сравнит с ожидаемым (сообщение `Verify BOTP PASSED …`). Подтверждения не требует.

Прожигать BOTP нужно **целиком, все 512 слов**: пустая (непрожжённая) ячейка читается как ноль, и если при автозагрузке в область регистров конфигурации попадут нули — микросхема после включения питания будет неработоспособна.

Сама по себе записанная BOTP ещё не включает автозагрузку: регистры начнут загружаться автоматически только после прожига бита `OTP_init_on` (шаг 6 полного сценария выше).

Программирование UOTP (PLL_config, INIT_conf)

1. Установите нужные значения PLL_config и INIT_conf **в таблице регистров** вкладки Terminal (введите вручную или загрузите через **Load from IC** / **Load from txt** / **Load from hex**). Перед прожигом рекомендуется записать их в чип кнопкой **Write to IC** и убедиться, что тактирование работает корректно.
2. **Create U22B** (панель `Actions:`) — сформирует файл `rom_u22b.hex` из значений `PLL_config` и `INIT_conf`, **отображённых в таблице регистров** (Create U22B читает таблицу UI, а не сам чип). Если после изменения значений не нажать `Create U22B` повторно, в hex-файле останутся старые значения. Если файла ещё нет и вы нажмёте `Prog UOTP`, приложение сообщит: «Файл rom_u22b.hex не найден! Сначала нажмите 'Create U22B'.»
3. **Prog UOTP** — прожигает значения в UOTP (напряжение VPP приложение подаст само). Два диалога подтверждения — «Да» на оба. После прожига приложение **автоматически читает фактическое содержимое ПЗУ** и выводит результат: `UOTP programmed: ROM VERIFY PASSED …` либо `UOTP ROM VERIFY FAILED` со списком несовпадений (в этом случае прожиг можно повторить).

> ⓘ **UOTP ПРОЖИГАЕТСЯ ДВАЖДЫ**
>
> Первый проход — **до** прожига BOTP: `PLL_config` + `INIT_conf` с выбранным `BOTP_clkdel` и `OTP_init_on = 0` (гарантирует правильное тактирование ПЗУ к моменту включения автозагрузки). Второй проход — **после** успешного Verify BOTP: тот же `INIT_conf`, но с `OTP_init_on = 1`. Перед вторым прожигом обязательно пересоздайте hex кнопкой **Create U22B** — `Prog UOTP` берёт значения из файла, а не из таблицы.

Проверить UOTP **вручную** (кнопки `Verify UOTP` нет):

1. В регистре UOTP_ctrl **снимите** бит `OVERRIDE_UVAL` (0) и **установите** бит `WATCH_ROM_UVAL` (1) — чтение `PLL_config`/`INIT_conf` станет возвращать фактическое содержимое ПЗУ UOTP. Важно: при `OVERRIDE_UVAL=1` чтение возвращает значения оперативных регистров, а не ПЗУ — «проверка через OVERRIDE» даёт ложный результат.
2. Прочитайте PLL_config и INIT_conf и сравните с тем, что прожигали.
3. Аппаратное описание процедуры — в разделе ПЗУ → параллельный доступ.

Справочник кнопок прожига

Все кнопки — в панели `Actions:` вкладки Terminal:

Кнопка / флажок	Что делает
Create BOTP	Формирует `rom_BOTP.hex` из файлов проекта (`cpu1/cpu2_data.hex` + `base_ram.txt`), 512 слов
Prog BOTP	Прожиг BOTP (необратимо, двойное подтверждение, авто-VPP)
Verify BOTP	Перечитывает BOTP и сравнивает с ожидаемым
Create U22B	Формирует `rom_u22b.hex` из таблицы регистров (PLL_config, INIT_conf)
Prog UOTP	Прожиг UOTP (необратимо, двойное подтверждение, авто-VPP, авто-проверка содержимого ПЗУ)
vpp9v	Ручная подача VPP (override для диагностики); для штатного прожига **не нужна**

> **НЕ ПУТАЙТЕ С PROGRAM**

> **Prog BOTP / Prog UOTP** прожигают **энергонезависимую OTP**. Кнопка **Program** в левой панели (ConnectPanel) к прожигу отношения не имеет — она записывает оперативную конфигурацию из файла `base_ram.txt` (действует до сброса питания).

Если микросхема перестала отвечать

Если после прожига бита `OTP_init_on` (автозагрузка из OTP при старте) микросхема не отвечает — возможны две причины:

1. **`OTP_init_on` прожжён до BOTP (или BOTP прожита не полностью).** При каждом включении питания все регистры инициализируются нулями из пустой ПЗУ: `Mode_config` = 0 (микровычислители выключены), `IC_addr` = 0 (микросхема не отвечает по адресу). Именно так проявляется ошибка порядка прожига. Исправление: найти микросхему (шаги ниже) и прожечь BOTP полностью — как минимум область регистров конфигурации (слова 504–511).
2. **Неверно задан делитель тактирования OTP** (BOTP_clkdel): инициализация выполняется на частоте выше 1 МГц и не завершается. Исправление: найти микросхему, записать корректный `INIT_conf` и перепрограммировать UOTP.

В обоих случаях микросхема не повреждена. Порядок восстановления:

1. Найдите адрес микросхемы, перебирая в регистре `BUS_addr` значения 1…255 до отклика (см. Методика подключения → Обнаружение микросхемы).
2. Выполните исправление по соответствующей причине выше.
3. Проверьте автостарт: снимите питание (или подайте сброс) — микросхема должна инициализироваться и отвечать.

Подробно — в разделах ПЗУ и Порядок программирования ПЗУ.

Связанные ресурсы

- ПЗУ — структура BOTP/UOTP и таблица ячеек
- Вкладка Terminal — где находятся кнопки
- Описание протокола программирования BOTP по SPI — файл `workstation4ad3s/PROG_BOTP_SPI_WRITE.md` репозитория DCsoyuz/ad3s.

*Последнее обновление **21 сент. 2026 г.***