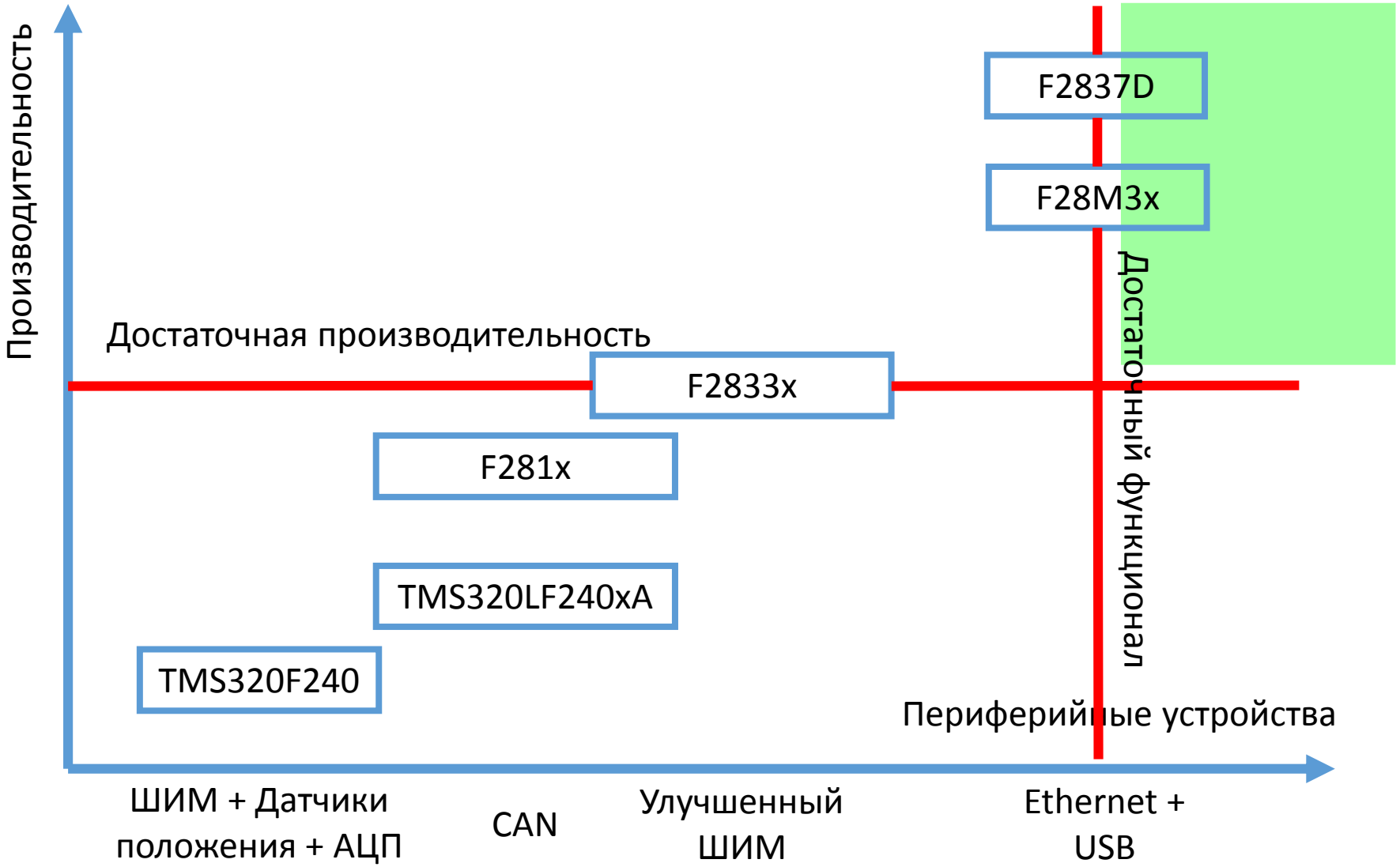
Компания
ООО «НПФ Вектор»
г. Москва

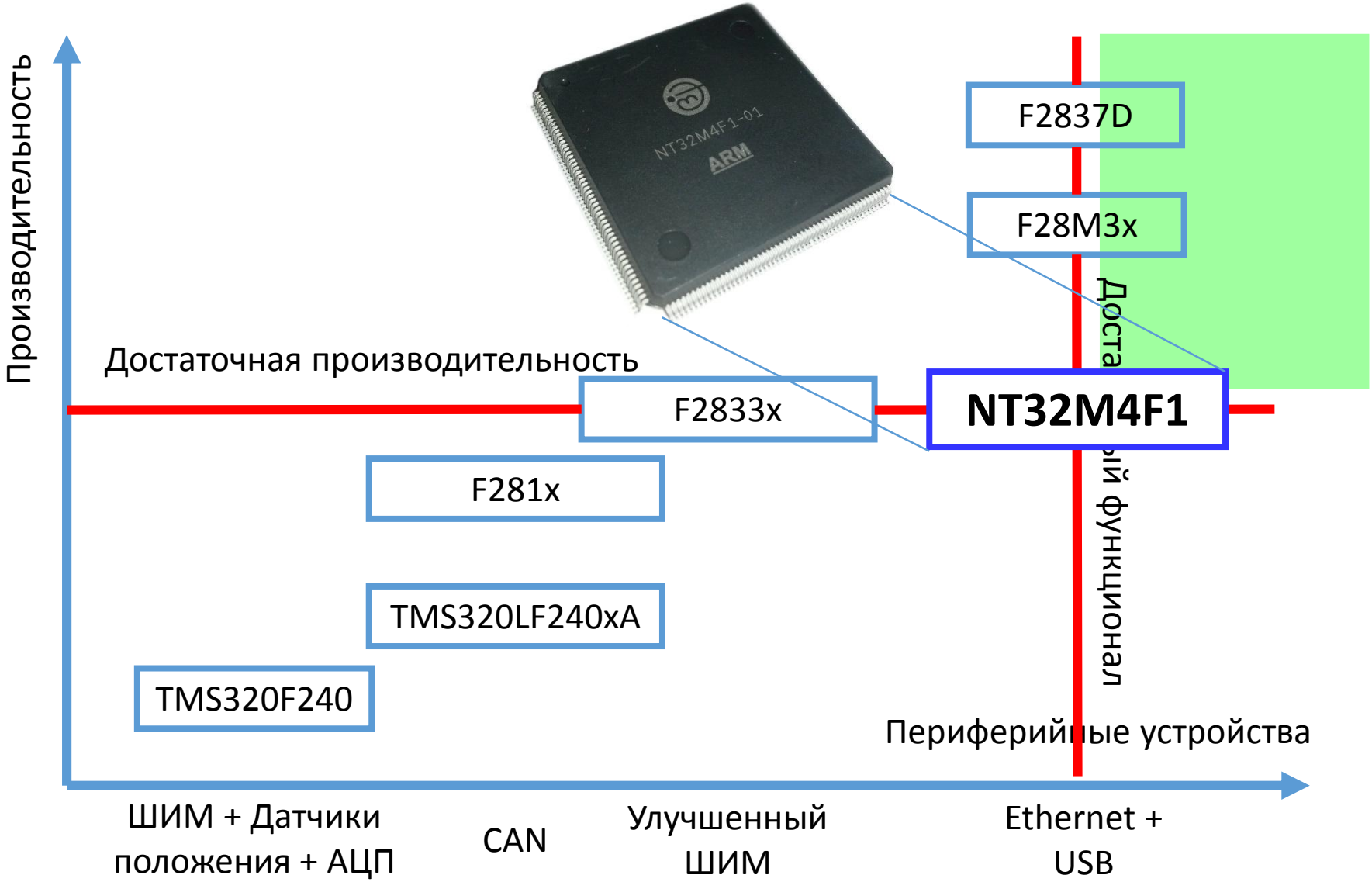
Разработка встраиваемых систем управления
Электропривод и автоматика

Доклад:
опыт работы с ARM микроконтроллером НИИЭТ NT32M4F1

Тенденции развития встроенных систем управления на примере МК Texas Instruments



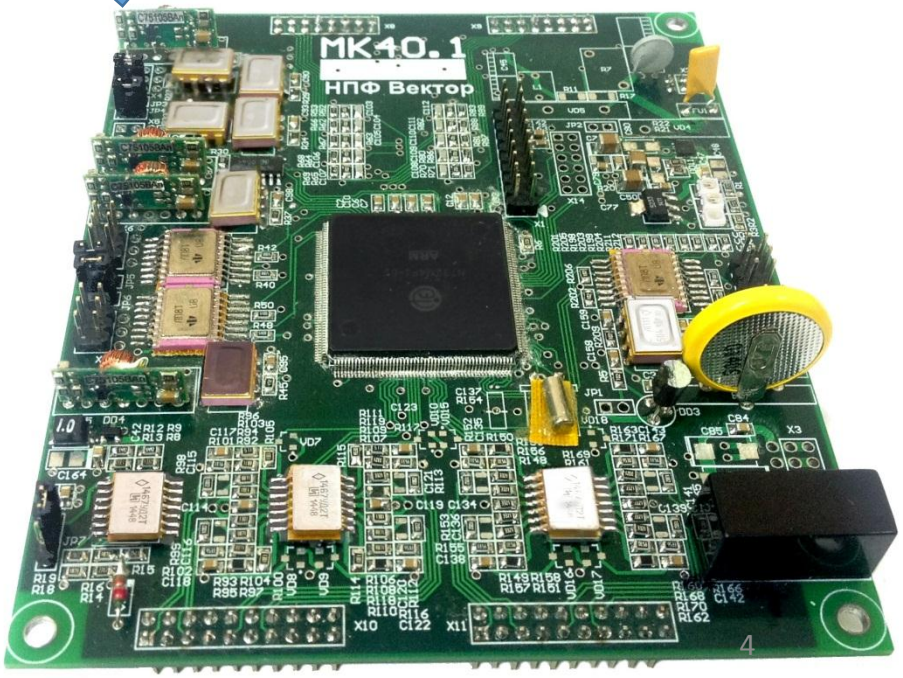
Микроконтроллер NT32M4F1 ARM 100 MHz (НИИЭТ Воронеж)



Сделать контроллер полностью на отечественных компонентах в тех же габаритах и посадочных местах взамен контроллера на импортных комплектующих? Это возможно!

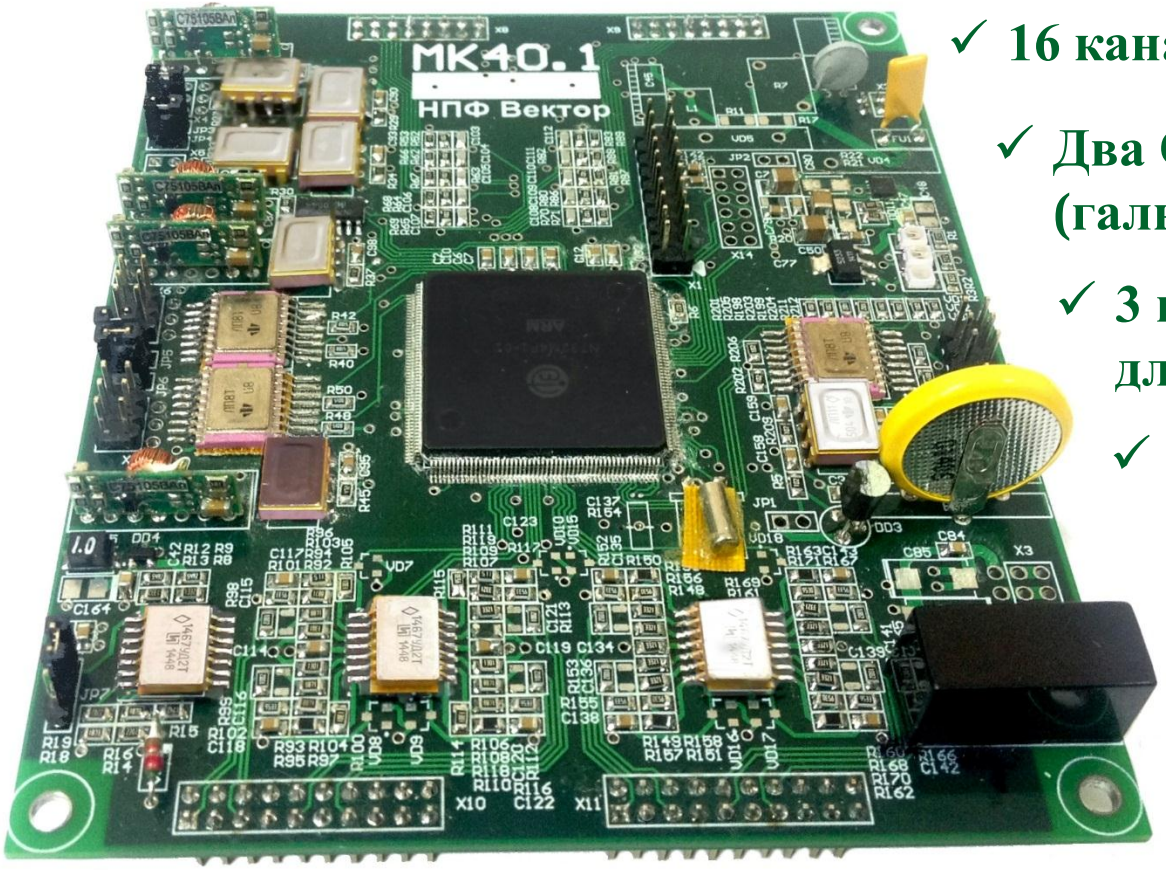
Texas Instruments
TMS320F2810 C28 150МГц

НИИЭТ
NT32M4F1 ARM 100МГц



МК40.1 – контроллер на отечественных компонентах на базе ОАО «НИИЭТ» NT32M4F1 с ARM ядром Cortex-M4F 100МГц

- ✓ Микроконтроллер NT32M4F1 – 100МГц, 1 МБ flash-памяти, 192 КБ ОЗУ



- ✓ 16 каналов ШИМ, 16 каналов АЦП
- ✓ Два CAN, два RS-485 (гальваническая изоляция!)
- ✓ 3 канала модуля захвата для датчика положения
- ✓ 8 каналов приема аппаратных аварий
- ✓ Часы реального времени
- ✓ Пользовательский EEPROM

Контроллер используется для тягового электропривода

Какие эмуляторы JTAG подходят?

Подходят любые JTAG для Cortex-M4F!

Нами были успешно испытаны:

- ✓ J-link v8 оригинального производства Segger
- ✓ Качественный аналог J-link – Jet-link v8
- ✓ Недорогой аналог J-link (за 800р) v8
- ✓ ST-link из отладочной платы по протоколу SWD

Все эмуляторы успешно работают! Отличается только максимальная скорость обмена.

Какую среду разработки и компиляторы выбрать для программирования на языке C/C++?

Подходят любые средства разработки для ARM Cortex-M4F: NT32M4F1 соответствует стандартам

IAR EWARM
Keil uVision

Популярные коммерческие среды разработки с собственными компиляторами. Цена на 2015г – около 400 т.р. за одно рабочее место

GCC+Eclipse+ OpenOCD

Инструментарий с открытым исходным кодом. По удобству превосходит платные аналоги, однако имеет высокий порог вхождения (сложно настроить и разобраться)

Программирование flash-памяти NT32M4F1

Есть одна проблема:

работа с ядром ARM стандартизирована, но процесс программирования flash-памяти у каждого производителя свой собственный – свои регистры

Необходимо научить среду разработки «прожигать» flash нашего МК!

Как обычно проходит процесс программирования flash-памяти?

1. Среда разработки через JTAG грузит в ОЗУ МК «загрузчик» – небольшую предварительно разработанную программу, которая умеет работать с регистрами программирования flash-памяти конкретного МК.
2. Дает через JTAG загрузчику команду на стирание flash-памяти.
3. Загружает через JTAG в ОЗУ МК фрагмент программного кода для прошивки.
4. Дает через JTAG загрузчику команду на «прожиг» фрагмента кода во flash.
5. После окончания процесса грузит следующую часть кода и запускает «прожиг» дальше и т.п.

Как это делается для популярных сред разработки?

IAR EWARM – содержит возможность написать свой программатор. **Программатор реализован** и имеется у ОАО «НИИЭТ»!

Keil uVision – содержит возможность написать свой программатор. **Программатор пока не написан** (можно попробовать написать самим – это не так сложно!)

GCC+Eclipse+OpenOCD – для поддержки своего алгоритма программирования flash необходимо доработать OpenOCD. **Программатор реализован** ООО «НПФ Вектор»!

Заголовочные файлы

Следующий фактор успешной разработки – наличие заголовочных файлов для работы с регистрами периферии МК! В заголовочных файлах должны быть описаны адреса и поля всех регистров.

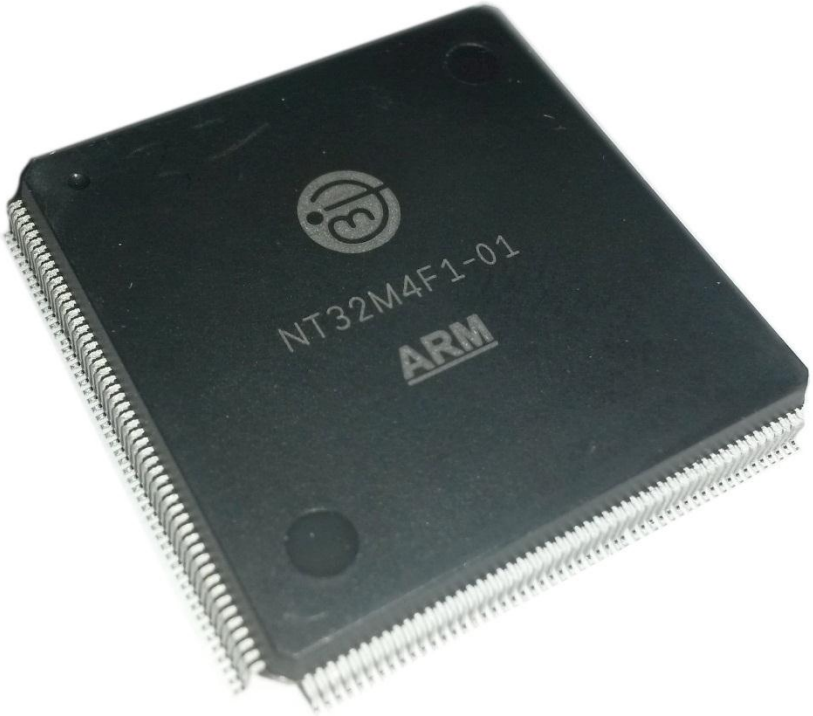
Заголовочные файлы для NT32M4F1 есть у НИИЭТ в двух видах:

1. Все регистры описаны в виде вложенных структур данных и битовых полей (struct и union) по аналогии с заголовочными файлами Texas Instruments – **удобно, но иногда возникают нетривиальные ошибки.**
2. Все регистры описаны в виде констант с адресами регистров и масок их битовых полей **– крайне неудобно, но надежно.**

А что с производительностью?

Сравним типичный Motorcontrol МК с NT32M4F1

VS



Texas Instruments
TMS 320F2810
ядро C28
150МГц

НИИЭТ
NT32M4F1
ядро Cortex M4F
100МГц

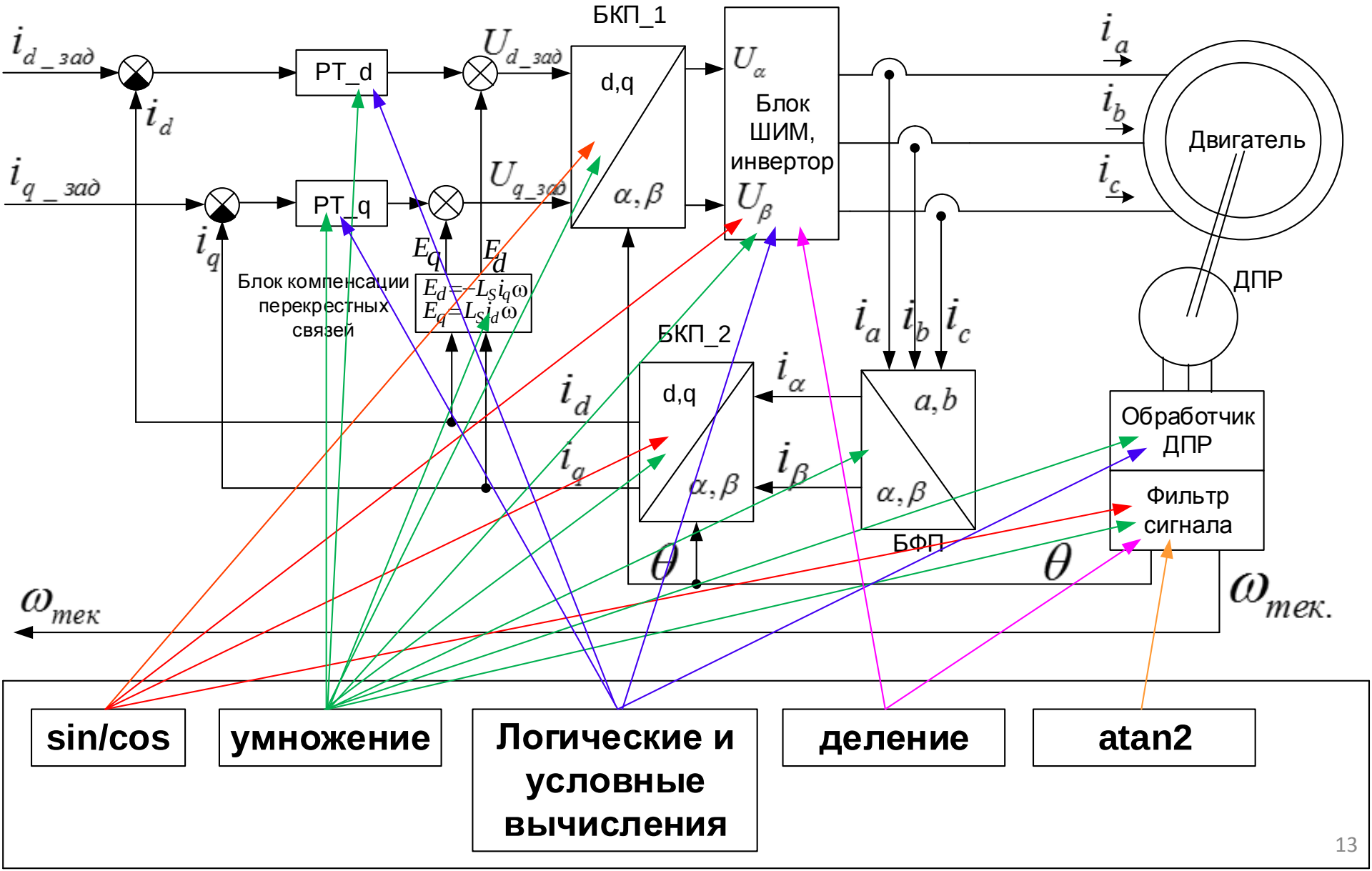
Как сравнить вычислительную производительность?

Всё зависит от используемых компиляторов и вида вычислений.

На какой вычислительной задаче сравнивать?

Рассмотрим классическую структуру векторного управления электродвигателем.

Какие вычислительные задачи решает микроконтроллер для векторного управления электродвигателем?



Какие библиотеки использовать?

Для ядра Texas C28 есть оптимизированная и написанная на ассемблере библиотека целочисленных вычислений в формате Q24 (и других) – IQmath.

Для ядра Cortex-M4F есть библиотека CMSIS-DSP, которую делает ARM.

Но:

1. **Она медленная!**
2. **Из целочисленных там только форматы 1.15 и 1.31 – они неудобны!**

Можно сделать свою библиотеку функций с нужной точностью и эффективностью

В интернете можно найти множество реализаций вычисления различных функций с разной точностью. Большинство реализаций – на Си.

Соответственно, эффективность работы таких функций сильно зависит от оптимизирующих свойств используемого компилятора.

Сравним компиляторы и готовые библиотеки на базе вычисления функции синуса.

Оценка **производительности компиляторов** на функции аппроксимированного **синуса** в целочисленной арифметике с фиксированной точкой в формате 8.24

```
int32 _IQ24sinPU(int32 x)
{
    int32 c, y;
    static const int32 qN= 13, qA= 12, B=19900, C=3516;
    x=x>>9;//from 8.24
    c= x<<(30-qN);              // Semi-circle info into carry.
    x -= 1<<qN;                 // sine -> cosine calc

    x= x<<(31-qN);              // Mask with PI
    x= x>>(31-qN);              // Note: SIGNED shift! (to qN)
    x= x*x>>(2*qN-14);          // x=x^2 To Q14

    y= B - (x*C>>14);           // B - x^2*C
    y= (1<<qA)-(x*y>>16);       // A - x^2*(B-x^2*C)
    y=y<<12;// to 8.24
    return c>=0 ? y : -y;
}
```

http://www.coranac.com/2009/07/sines/ алгоритм работы данной аппроксимации

Честный точный табличный синус!

	GCC	IAR	C28 TI IQmath Q24	CMSIS-DSP Q15
Число ассемблерных команд	18	19	-	-
Число тактов	29	30	45	80

Оценка **производительности компиляторов** на функции целочисленного **умножения** в формате 8.24

```
inline  int32 _IQ24mpy(int32 inArg0, int32 inArg1) {
    return (int32)(((int64)inArg0 * inArg1)>> 24);
}
```

	GCC	IAR	C28 TI IQmath
Число команд	6	6	7
Число тактов	8	8	7

Лучшие компиляторы для ядра Cortex-M4F проигрывают всего один такт ядру C28 Texas Instruments, где умножение реализовано в виде оптимизированной библиотечной функции

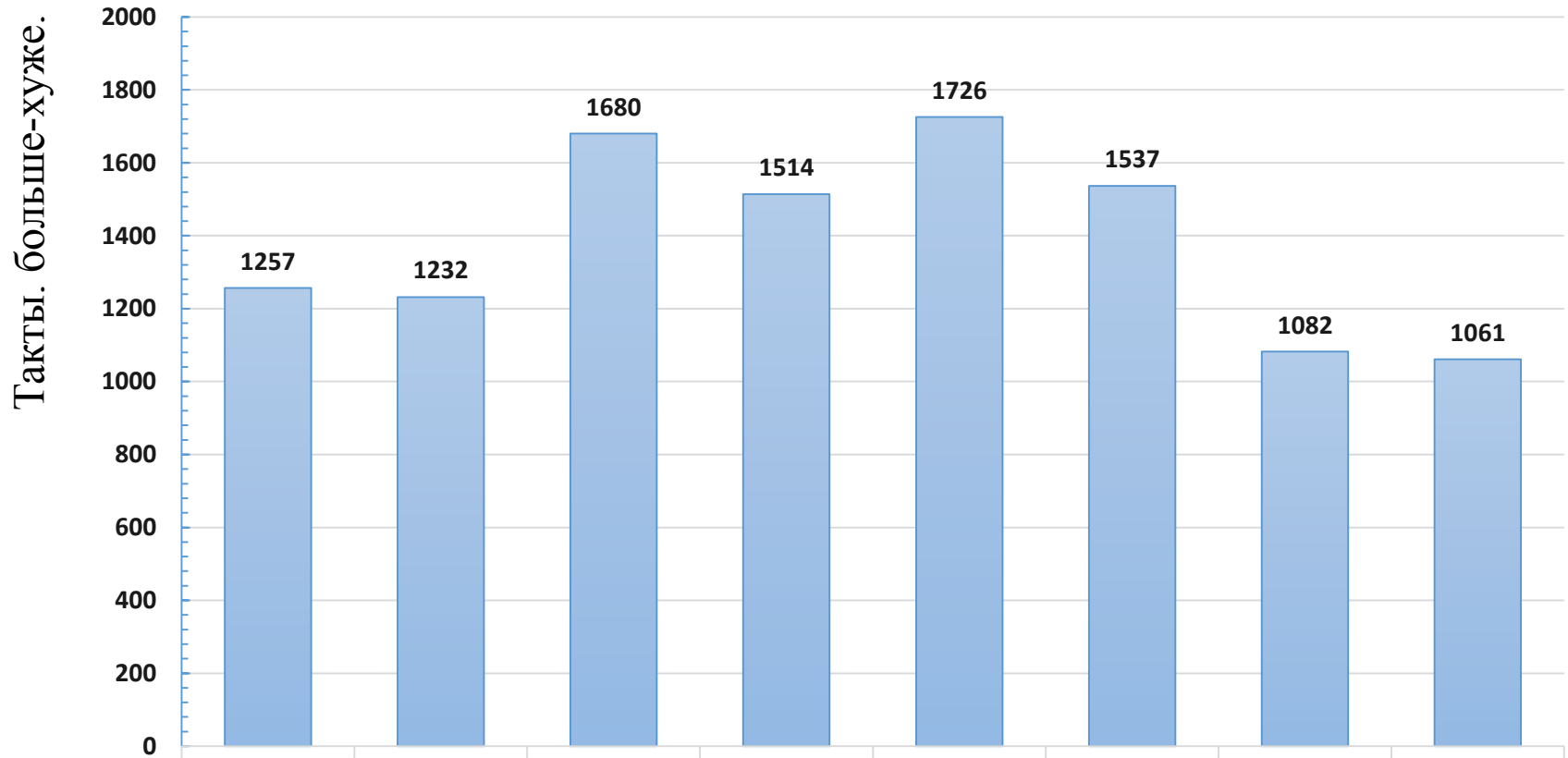
Промежуточные выводы

- **Современные компиляторы очень эффективны**
- **Свою библиотеку функций реализовать можно**
- **Там, где не требуется высокая точность готовых библиотек, своя библиотека будет быстрее!**

Мы реализовали свою библиотеку целочисленных вычислений с функциями умножения, деления, синуса, косинуса, atan2, квадратного корня и других на Си.

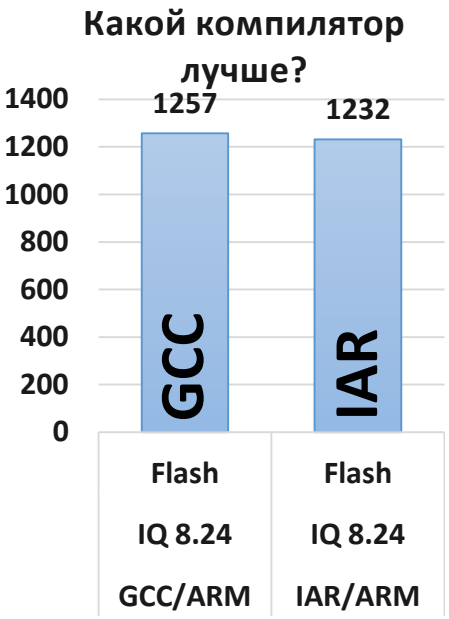
Проведем комплексное сравнение компиляторов и библиотек!

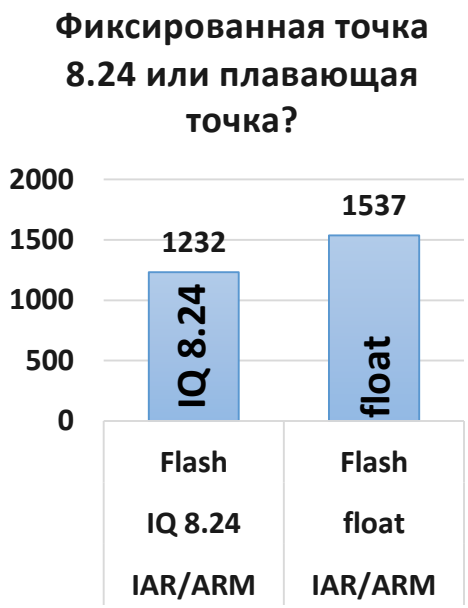
Комплексная задача: вычисление сложного программного модуля, который включает в себя **синус, косинус, множество формул, функцию арктангенса и деление**. Тест имитирует **среднестатистический набор** вычислений, требуемый для задач **векторного управления**.



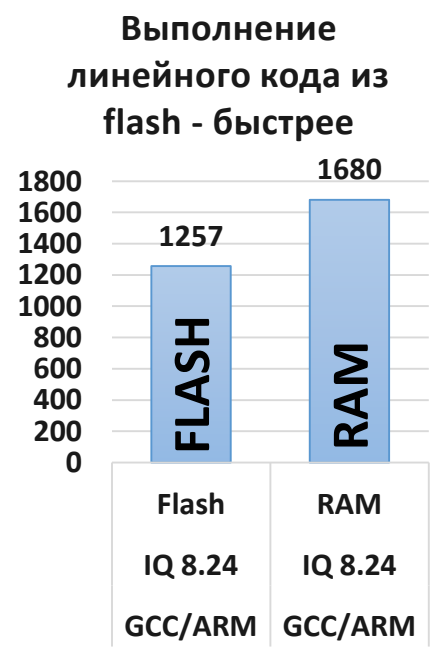
Где расположен код:	Flash	Flash	RAM	RAM	Flash	Flash	Flash	RAM
Тип данных:	IQ 8.24	IQ 8.24	IQ 8.24	IQ 8.24	float	float	IQ 8.24	IQ 8.24
Компилятор/ядро:	GCC/ARM	IAR/ARM	GCC/ARM	IAR/ARM	GCC/ARM	IAR/ARM	Texas/C28	Texas/C28

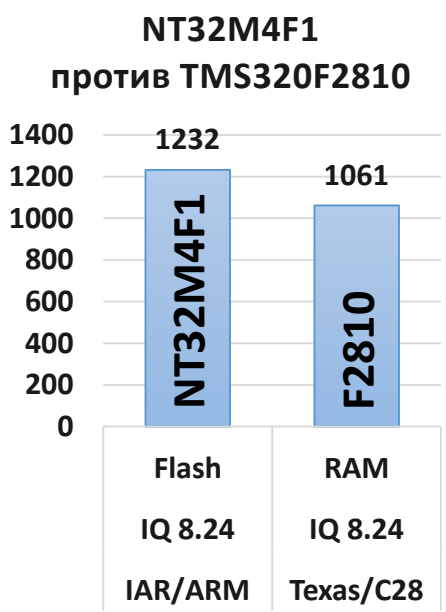
Интерпретация результатов теста





Вычисления во float отстают, несмотря на аппаратную поддержку float в ядре: из-за того, что для вычислений float в этом тесте использовались стандартные библиотеки компиляторов, а для IQ 8.24 – библиотека с быстрыми приближенными вычислениями.

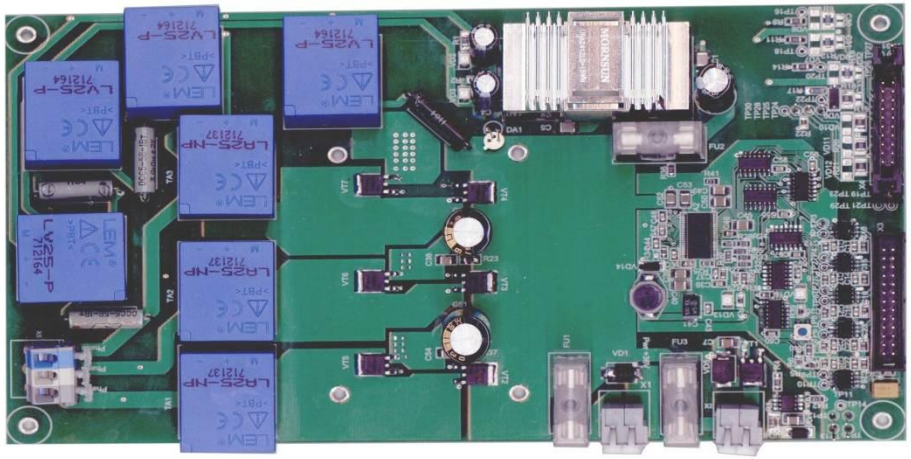




С разработанной «НПФ Вектор» библиотекой целочисленных вычислений МК NT32M4F1 удалось приблизиться к результатам МК TMS320F2810 с фирменной библиотекой IQmath по числу тактов. Однако из-за более низкой частоты итоговое отставание NT32M4F1 больше.

Как мы проверяли микроконтроллер NT32M4F1

Motorcontrol функции проверялись при помощи сервопривода: двигатель с постоянными магнитами, векторное управления с двумя датчиками положения, векторный ШИМ, программный контур тока на базе встроенного АЦП и работа по протоколу CANopen.



Выводы – страница 1

- Можно относительно «безболезненно» **перенести программное обеспечение** с TI TMS320F2810 на НИИЭТ NT32M4F1 **в целочисленной арифметике**, реализовав собственные библиотечные функции на Си – **проигрыш** в вычислениях **не более 20%.** С учетом более низкой частоты ядра, придется оптимизировать ПО.

- При использовании стандартных библиотек работы **с плавающей точкой**, встроенных в компиляторы, вычисление происходит **медленнее**, чем с целыми числами в формате 8.24.

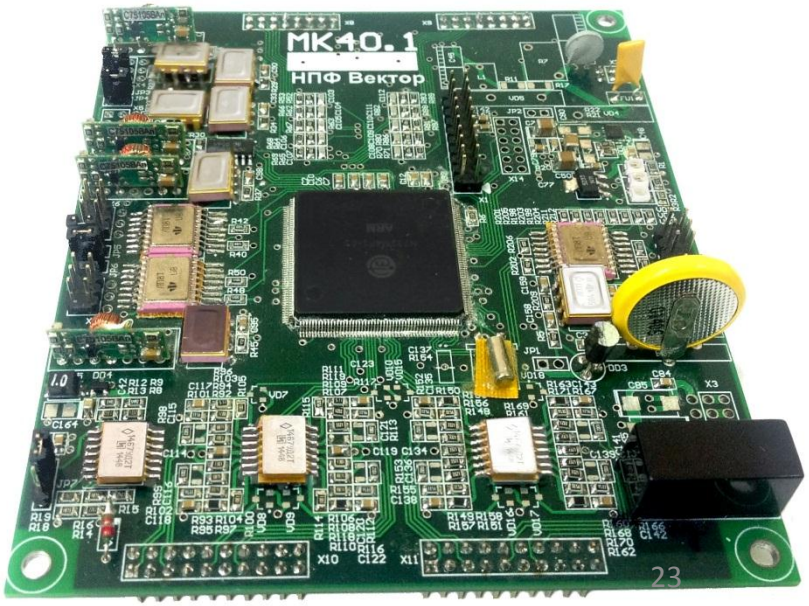
- **Использовать плавающую точку** имеет смысл **при разработке** ПО с **нуля**, при этом требуется **найти «быструю» библиотеку** функций синуса, арктангенса, квадратного корня и т.п.

- Бесплатный компилятор с открытыми исходными кодами **GCC** вместе со средой **Eclipse** представляют собой **превосходный инструментарий** для разработки, однако непростой в конфигурировании.

Выводы – страница 2

Удалось разработать контроллер **полностью на отечественных компонентах** (или сделанных в странах таможенного союза) взамен контроллера на импортных компонентах!

+ В тех же габаритах
+ Выполняет те же функции управления электродвигателем
+ Имеет тот же температурный диапазон работы
+ Разработка ПО в Eclipse: более, чем очень удобно!
– Требуется чуть более оптимизировать вычисления из-за более медленного ядра



Немного рекламы

НПФ Вектор
Москва

http://motorcontrol.ru/
+7-(495)-303-3754

ООО «НПФ Вектор» предлагает решения для микроконтроллера НИИЭТ NT32M4F1

- **Контроллер МК 40.1 на базе NT32M4F1 для задач электропривода и источников питания**
- **Средства разработки для NT32M4F1: оптимизированную сборку GCC+Eclipse+OpenOCD с функциями программирования flash-памяти**
- **Быструю библиотеку приближенных целочисленных вычислений в IQ24**
- **Библиотеку цифрового управления электродвигателями и источниками питания: ПИД-регулятор, цифровой фильтр первого порядка, фазные и координатные преобразования, блок кривой U/f, задатчик интенсивности и другие модули.**
- **Отдельные программные модули: модуль векторной ШИМ, драйвер CANopen, драйвер MODBUS, модуль обработки датчика положения ротора типа энкодер.**
- **Демонстрационное программное обеспечение, включающее сразу несколько описанных выше модулей для скалярного или векторного управления синхронным или асинхронным электродвигателем (по заказу).**