**8.5 Типы данных и руководство по стилю кодирования.**

**Типы данных.**

Каждый порт FreeRTOS имеет уникальный заголовочный файл portmacro.h, который содержит (среди прочего) определения для двух специальных типов данных, portTickType и portBASE\_TYPE. Эти типы данных описаны в таблице 30.

Таблица 30. Специальные типы данных, используемые FreeRTOS.

|  |  |
| --- | --- |
| Используемый макрос или typedef | Реальный тип |
| portTickType | Этот тип используется для хранения значения счетчика тиков и для определения времен блокирования. portTickType может быть либо 16-битным типом без знака, или 32-битным типом без знака, в зависимости от установки опции configUSE\_16\_BIT\_TICKS в файле FreeRTOSConfig.h. Использование 16-битного типа может великолепно улучшить эффективность на 8-битных и 16-битных архитектурах, но сильно ограничивает период максимального времени блокировки. Нет причин использовать 16-битный тип на 32-битной архитектуре. Так что для них configUSE\_16\_BIT\_TICKS должен быть установлен в 0. |
| portBASE\_TYPE | Этот тип всегда определен как наиболее эффективный тип данных для архитектуры. Обычно это 32-битный тип на 32-битной архитектуре, 16-битный тип на 16-битной архитектуре и 8-битный тип на 8-битной архитектуре. portBASE\_TYPE вообще говоря, используется для возвращаемых значений, которые могут принимать только очень ограниченный диапазон значений, и булевы значения. Порты Cortex-M3 определяют portBASE\_TYPE как «long». |

Некоторые компиляторы делают все неквалифицированные переменные char беззнаковыми, в то время как другие делают их знаковыми. По этой причине исходный код FreeRTOS явно квалифицирует каждое использование char либо с «signed», либо с «unsigned». Простые int типы никогда не используются – только long и short.

**Имена переменных.**

Переменные имеют префикс, который обозначает тип: «c» для char, «s» для short, «l» для long, и «x» для portBASE\_TYPE и прочих типов (структур, дескрипторов задач, дескрипторов очередей и пр.). Если переменная беззнаковая, то к префиксу добавляется «u». Если переменная – указатель, то к префиксу добавляется «p». Таким образом, переменная типа unsigned char будет иметь префикс «uc», а переменная типа указатель на char будет иметь префикс «pc».

**Имена функций.**

Префиксы функций состоят из типа возвращаемого значения и имени файла, в котором они определены. Например:

* v**Task**PrioritySet() возвращает void и определен в файле «**task**.c».