Урок 1. Мигаем светодиодами и нажимаем кнопочку.

Цель: Получить первое представление о целях использования RTOS, понять какая польза от их использования, начать осознавать, какой потенциальный ад в коде будет происходить, если разрабатывать сложное встроенное ПО без RTOS.

Доп.цель: Получить представления как разрабатывать приложения без RTOS, так называемые bare-metal приложения.

Средства: среда разработки DAVE и отладочная (Relax Kit 4500 или любая другая) плата, на которой есть два светодиода и кнопка.

Задача: На плате есть два светодиода и кнопка. Необходимо написать программу, которая будет мигать светодиодами. Период мигания одного диода в два раза больше второго, допустим, 500 мс и 1000 мс. То есть мигание по схеме:

|  |  |  |
| --- | --- | --- |
| Led 1 | Led 2 | Timeout |
| Off | Off | 500 мс |
| Off | On | 500 мс |
| On | Off | 500 мс |
| On | On | 500 мс |

При нажатии на кнопку в течение от 3 с до 5 с – светодиод Led 1 перестает мигать, при повторном нажатии такой длительности – снова начинает мигать (в фазе с с Led 2, ну то есть как мигал ранее).

При нажатии на кнопку в течение от 5 с до 8 с – светодиод Led 2 перестает мигать, при повторном нажатии такой длительности – снова начинает мигать (в фазе с с Led 1, ну то есть как мигал ранее).

Задание 1.

|  |  |
| --- | --- |
| Создаем GIT-репозиторий |  |
| добавляем файл .gitignore |  |
| создает DAVE workspace и пустой DAVE-CE project. |  |
| Компилируем, проверяем что бинарный файл загружается в плату и есть возможность «ходить» по шагам в режиме отладки. |  |

Сделать все так, чтобы получилось, как в репозитории

<https://github.com/ra9obw/xmc4_rtos_lsns> tag less1\_task1

Задание 2.

Реализовать схему мигания светодиодами в функции main() на основе компонентов DAVE App.

Реализовав свою функцию задержки с прототипом void my\_usleep (uint32\_t timeout\_in\_mks); на основе цикла и макроса \_\_NOP().

Выполнение:

Создаем DAVE App для управление процессором, тактовым генератором и DIGITAL\_IO для светодиодов.

Указываем выводы и генерируем код.

Пишем функцию **my\_usleep** ()

и в цикл while(1) фнукции main добавляем код мигания лампочек:

**my\_usleep** (500000);

DIGITAL\_IO\_ToggleOutput(&LED\_1);/\* Endless loop \*/

**my\_usleep** (500000);

DIGITAL\_IO\_ToggleOutput(&LED\_1);/\* Endless loop \*/

Убедится, что период мигания примерно 500 мс. Если нет – подгоняем внутренности функции my\_delay, чтобы примерно совпадало.

Ход выполнения.

|  |
| --- |
| Создаем DAVE App CPU\_CTRL\_XMC4 |
|  |
| CLOCK\_XMC4 и настраиваем на параметры тактового сигнала, который есть на нашей плате:  Для RelaxLite – это кварц 12 МГц.  Для сейсмоплаты – 16М ГЦ внешний тактовый генератор. |
|  |
| DIGITAL\_IO два раза. Переименовываем их в LED\_0 и LED\_1, настраиваем на выходы.  Для сейсмоплаты это P2.10 и P1.14 |
|  |
| Пишем код функции  **void** **my\_usleep**(uint32\_t dly)  {  //scaling factor  dly \*= 12;  // dly /= 10;  //  **for**(; dly > 0; --dly)  {  \_\_NOP();  }  }  Пишем код в цикле while(1) функции main:  /\* Placeholder for user application code. The while loop below can be replaced with user application code. \*/  **while**(1U)  {  my\_sleep(500000);  DIGITAL\_IO\_ToggleOutput(&LED\_0);  my\_sleep(500000);  DIGITAL\_IO\_ToggleOutput(&LED\_0);  DIGITAL\_IO\_ToggleOutput(&LED\_1);  } |
| Собираем, загружаем, работает. |

Обратим внимание на строку dly \*= 12; Считаем, что \_\_NOP() пропускает 1 такт, накладными на вход в функцию и можно принебречь, плюс есть накладные на организация цикла на каждой итерации. Надо, чтобы наша функция задержки принимала на вход примерно микросекунды. Частота процессора 120 МГц (120 тиков в микросекунде), стало быть надо входную задержку умножить на 120 и поделить на число тактов в одной итерации цикла. Сколько это – мы не знаем. Можно примерно подобрать или измерить осциллографом. У меня получилось 12 без оптимизации –O0 (с –O2 будет другая величина).

На что следует обратить внимание?

1. Наш процессор фактически все время тратит на то, чтобы ничего не делать (выполняет \_\_NOP, чтобы сгенерировать задержку).
2. Величина задержки не точна и зависит от компиляции к компиляции.

Задание 3.

Давайте начнем с того, чтобы поточнее отмерять задержку. Для этого предлагается включить отсчет системного времени с помощью таймера.

Немного теории.

Многие (практически, все) микроконтроллеры содержат в своем составе периферийные модули таймеров. Таймер периодически отсчитывает интервал времени, после чего может генерировать некоторое событие, например, электрический сигнал на выходе или прерывание. После события таймаута таймер, в зависимости от конфигурации может перезапускаться заново. Периферийные модули-таймеры могут быть очень сложными устройствами с большим числом каналов и возможностью генерировать ШИМ-сигнал. Например, можно ознакомиться с описанием блоков-таймеров для МК XMC4500 в соответствующем разделе HW Manual на МК.

Задача – реализовать логику функции my\_sleep(), используя время, отсчитываемое с помощью «системного» таймера.

Само время нужно отсчитывать в прерываниях таймера с периодом таймаута 1 мс. Значение текущего времени в микросекундах можно брать из регистра таймера, а число миллисекунд и секунд формировать с помощью собственно реализованного обработчика прерываний и своей структуры данных, хранящей системное время.

|  |
| --- |
| Добавляем DAVE App SYSTIMER и задаем период 1000 мкс. |
|  |