ECE 315 Assignment 1

Arun Woosaree
XXXXXXX

February 23, 2020

1

The system will enter any of the low power modes below when a `STOP` instruction is executed. Depending on the `WCR[LPMD]` bits, the MCU will enter either wait, doze, or stop mode. For each of these modes, power is saved by idling the CPU with no active cycles, powering down the system, and stopping all internal clocks. The coldfire core is also disabled in each of the low-power modes.

A wake-up event will cause the MCU to exit any of these low power modes, and return to run mode. A wake-up event can be any type or reset, or any valid, enabled interrupt request. To exit from these low power modes with an interrupt, the interrupt request's priority should be higher than the value programmed in `WCR[LPMD]`, and higher than the value programmed in the interrupt priority mask (I) field of the core's status register. Additionally, the interrupt request should be from a source that is not masked in the interrupt controller's interrupt mask register, and it should have been enabled at the module of the interrupt's origin.

wait

Wait mode saves power by stopping the CPU and memory clocks until a wake-up event is detected. Peripherals can be programmed to continue operating, and can also generate interrupts which will cause the CPU to exit from wait mode. The wait mode is entered when the `STOP` instruction is executed, with the `WCR[LPMD]` bits having a value of `10`.

doze

Doze mode is similar to wait mode, in that it affects the processor in the same way as in wait mode, however, some peripherals define individual operational characteristics in doze mode. Peripherals continuing to run and having the capability of producing interrupts may cause the CPU to exit the doze mode and return to run mode. Stopped peripherals restart operation on exit from

doze mode, as defined for each peripheral The doze mode is entered when the `STOP` instruction is executed with the `WCR[LPMD]` bits having a value of `01`.

stop

Stop mode is also similar to wait and doze mode, in that it affects the processor the same way. However, all system clocks are stopped, and peripherals cease operation. The stop mode is entered when the `STOP` instruction is executed with the `WCR[LPMD]` bits having a value of `11`.

2

Similarities

1. both use harmonic intervals to move between different sections of code
2. each polled input or device is assigned to one loop or state
3. loops/states not giving up CPU until the code is done processing

Differences

1. separate blocks of code represented as a different loop vs. a different states
2. interupts for responding to external events, (Multiple nonpreemptive loops) provide fast and deterministic responce to external events vs. giving up CPU to the kernel when the task is complete, meaning external events are only processed when the state associated with that peripheral is run (periodically state-driven)
3. advancing from state to state happens at predictable intervals, but with multiple nonpreemptive loops with interrupts, an interrupt can vary the time it takes to switch between tasks

3

Advantages

- when responsiveness is desired and fast response times are needed
- allows the programmer to concentrate on the requirements of a specific task without worrying about the effect it has on the system and other tasks
- better modularity, making it easier to maintain and extend the code

When would it be more appropriate to choose a single threaded bare-metal architecture

- when simplicity is desired in the source code and deemed more important than responsiveness. For example, when prototyping
- when there are only a few, not complicated tasks
- if the overhead of multitasking is too high. For example, multitasking environments require more memory, and context switching might take too much time. Also, debugging becomes more difficult due to complex interactions, and control of shared resources
- if efficient operation with the simplest hardware possible is desired

4

Idle time can be implemented using an idle task that executes `NOP` in an infinite loop until a hardware timer signals the star of the next looping interval.

5

6

7

8

9