**Ports**

**PE2 (Sound Port)** – Only two operations on port are in Sound.c and they are toggle or off. They are never done simultaneously because in main it either toggles OR turns in off, which is dependent on alarm\_Lock flag.

**PF1 (Heartbeat)** –Only toggled in Time.c. Never written to elsewhere.

**PE4 (Select)** – Port is Read-Only

**PB4 (Down)** – Port is Read-Only

**PC4 (Up)** – Port is Read-Only

**Global Variables**

**Clock[], clock1[], clock2[], clock3[], secX[], secY[], minx[], minY[], hourX[], hourY[]** – read only (all constants)

**current\_State** – current\_State is only changed in Time.c (when alarm is triggered) or in Switch.c (when button triggers a state transition). In SysTick\_Handler AND in each GPIOPort”X”\_Handler we use StartCritical() and EndCritical(sr); to ensure that no interrupts occur during these handlers to prevent critical sections when current\_State is read or written to.

**global\_Hour, global\_Min, global\_Sec** – same as above.

**alarm1\_Hour, alarm1\_Min, alarm1\_Sec, alarm2\_Hour, alarm2\_Min, alarm2\_Sec** – Only changed in Switch.c when a button interrupt makes it go up or down. These are atomic accesses and additionally, we use StartCritical() and EndCritical(sr) to ensure no interrupts take place while these increment/decrement operations take place.

**alarm1\_Enable, alarm2\_Enable** – Only enabled/disabled in Switch.c off a button interrupt. Same conditions as the H/M/S as stated previously.

**snooze\_Hour, snooze\_Min, snooze\_Sec** – Only triggered when an alarm is snoozed. This is set in Switch.c and cannot be changed elsewhere. Same as above.

**snooze\_Enable** – Set when an alarm is snoozed. Then disabled when time reaches the snooze time and and alarm\_Lock is set. These two should never overlap because snooze is one minute long. Secondly, we have StartCritical() and EndCritical(sr) just in case.

**clock\_Mode** – changed only in Switch.c and happens on a “Select” button push on the toggle screen. It is used in some of the display methods, but the variable is only being read in the if statements to determine if a white or black background needs to be displayed on the LCD.

**interrupt\_Flag** – interrupt flag is set during every interrupt that requires a screen change to be printed. It is then read in the AlarmClock.c (main) module. Then it clears the flag. We have put a StartCritical() and EndCritical(sr) during this write in main. When the interrupts are set, there will be no critical section because all the interrupts will set it to 1 regardless so there will be no “conflict” between the different interrupts.

**alarm\_Lock** – alarm\_Lock is used in AlarmClock.c, Switch.c, and Time.c. It is only read in AlarmClock.c to change the screen. Time.c and Switch.c writes are in StartCritical() and EndCritical(sr) so there is no critical section because interrupts can’t occur during the time when we are writing to alarm\_Lock.