**Summary of Xmega C code Functionality**

* **Preprocessor commands**: basic library and macro calls, function and variable declarations
* **Main**: begins the program and is executed only once at startup and reset
  + Initialize hardware pins and software settings:
    - System clock to 32MHz
    - 2-wire serial communication for debug (USART)
    - Hardware pins with pull-up/pull-down resistors for stability where necessary
    - Overflow interrupt timer to maintain sample frequency of control loop
    - PWM timer for hardware connection to H-bridge and motor
  + Enter “Manual Position Mode” where buttons on breadboard may be used to position the platform prior to execution; wait for user to toggle a switch on the breadboard to enable user inputs via the UI
  + Enter SPI mode to transfer user inputs from Raspberry Pi
    - Initialize SPI hardware and software, clear SPI buffer
    - Accept 6 user inputs from the UI via 16-bit SPI transactions (comprised of 2 post-processed 8-bit transactions) and send the one transaction, the calculated sample frequency, back to RPi to signal completion. User inputs include:
      * Amplitude Motor 1
      * Amplitude Motor 2
      * Frequency Motor 1
      * Frequency Motor 2
      * Number of Cycles
      * Input Mode (Oscillation or Step input)
    - Store input variables according to their “control bits” designated as the first 4 binary bits in the 16-bit transaction
  + Depending on user inputs, calculate sample frequency, proportional gain for control equation, and total time of program execution
  + Initialize quadrature decode to process encoder signals using a specialized event system which handles this function outside of the CPU, preserving processing speed
  + Enable global interrupts to initiate the control loop interrupt timer and begin program execution
* **While Loop**: In embedded programming, a never-ending while loop is often used simply to keep the program going while the interrupt fires according to the timing required for the control loop. Here, the loop keeps the program running until it has ended according to the user inputs then shuts down the control loop interrupt and provides a software reset.
* **Control Loop Interrupt**: This part of the code executes the closed loop control equation and sends commands to the motor at the specified sample frequency.
  + If the user inputs “Oscillation” mode, the platform command position is a sine wave calculated from the user inputs of amplitude and frequency
  + If the user inputs “Step” mode, the platform command position is simply the amplitude according to the user input
  + Encoder values are read and stored, and the error signal is calculated
  + The control equation is executed
  + The command position and actual position are sent to the RPi via 16-bit SPI transaction with command bits, as well as certain bits specifying whether the value is positive or negative
  + All variables from the current cycle are stored for use in the control equation for the next cycle
  + The control signal is converted to a voltage and sent to the motors via a PWM signal to the motor drivers
  + Program time is incremented according to the sample frequency
* **End of Program**: When program time runs out, the user flips the toggle switch to execute the software reset contained in the while loop. The code executes from the top, which in the user experience, takes them directly back into “Manual Positioning Mode,” and ready for the next test.