Pseudocode for InputService.c

This service takes analog inputs from potentiometer and accelerometers and convert the them into digital, the results of which are obtained by calling the query functions defined in this service from different services, which act upon these values.

Data private to the module:
MyPriority
CurrentState
NewResult
Result
Converting// this is a flag to indicate that the AD conversion is in progress

Define query functions for:
RA0 // left accelerometer X value
RA3 // right accelerometer X value
RA7 // tilt sensor for pairing
RB5 // for potentiometer team select

Function : bool InitInputService(uint8_t Priority)
Save MyPriority in Priority
Set the ports and Pins for analog inputs
Initialise ADC hardware by setting ADCON0 to 0x01 and ADCON1 to 0
Set the clock in ADCON1 by setting BIT5HI and BIT7HI for FOSC/32 //this is the conversion clock
Initialize the read_timer to read values from the analog inputs
Set CurrentState to Reading
Post ES_Init to itself

Function : bool PostInputService(ES_Event_t ThisEvent)
Return PostToService

Function : ES_Event_t RunInputService(ES_Event_t ThisEvent)
Set ReturnEvent to ES_NO_EVENT

If CurrentState is Reading: //reading is the only state in this state machine

If the event is ES_TIMEOUT and param is Read_Timer
    If Converting_flag for RA0 is off
        Configure ADCON0 for RA0
        write blocking code for 7.6us
        make Converting_flag for RA0 as high
        clear ADRESH and ADRESHL
        set the Go/NotDone bit
        write blocking code for 20us for conversion
    endif

```
if converting_flag for RA0 is high //ready for read
        Initialize New_Result_RA0 to 0
        Read ADRESH into upper 8 bits of New_Result
        Read ADRESL into Lower 8 bits of New_result
        Take average of Result and New_Result and save in Result  //this
implements moving average of the result values
        Configure ADCON0 for RA3
        //no need to write 7.6us blocking code after this as there are a
number of instructions. If response is slow (updating values slowly), add blocking code
        Set the Converting_Flag for RA3 as high
        Set the Go/Not Done bit to start conversion
        Write blocking code for 20 us for capacitor charging.
endif

if converting_flag for RA3 is high //ready for read
        Initialize New_Result_RA3 to 0
        Read ADRESH into upper 8 bits of New_Result
        Read ADRESL into Lower 8 bits of New_result
        Take average of Result and New_Result and save in Result  //this
implements moving average of the result values
        Configure ADCON0 for RA5
        //no need to write 7.6us blocking code after this as there are a
number of instructions. If response is slow (updating values slowly), add blocking code
        Set the Converting_Flag for RA7 as high
        Set the Go/Not Done bit to start conversion
        Write blocking code for 20 us for capacitor charging.
endif

if converting_flag for RA7 is high //ready for read
        Initialize New_Result_RA07to 0
        Read ADRESH into upper 8 bits of New_Result
        Read ADRESL into Lower 8 bits of New_result
        Take average of Result and New_Result and save in Result  //this
implements moving average of the result values
        Configure ADCON0 for RB5
        //no need to write 7.6us blocking code after this as there are a
number of instructions. If response is slow (updating values slowly), add blocking code
        Set the Converting_Flag for RB5 as high
        Set the Go/Not Done bit to start conversion
        Write blocking code for 20 us for capacitor charging.
endif

If Converting_flag for RB5 is 1
        Turn it flag off
        Read the result
        Take average and store in Result
Endif
```

Endif

If default state
    Break

Function : uint16_t QueryRA0(void)// will be called by TransmitService
Return Result_RA0

Function : uint16_t QueryRA3(void)// will be called by TransmitService
Return Result_RA3

Function : uint16_t QueryRA7(void)// will be called by TransmitService
Return Result_RA7

Function : uint16_t QueryRB5(void)// will be called by TransmitService
Return Result_RB5