```c
#include "_threadsCore.h"

//global variables
const uint32_t MAX_STACK = 0x2000;
int numStacks = 0;

//obtain the initial location of MSP by looking it up in the vector table
uint32_t* getMSPInitialLocation (void){
  uint32_t* MSP_ptr = (uint32_t*) 0x0; //define a pointer to a pointer that points to initial MSP
  printf("%08x\n", *MSP_ptr);
  return (uint32_t*) *MSP_ptr; //dereference so that it returns just the pointer to initial MSP
}

//return address of new a PSP with offset of "offset" bytes from MSP
uint32_t* getNewThreadStack (uint32_t offset){
  //check if we are exceeding the max stack size
  if (MAX_STACK < offset*(numStacks+1)){
    printf("ERROR: Offset too large");
    return NULL; //make sure to look for a NULL return in future functions to check if getNewThreadStack failed or not
  }
  ++numStacks;

  //calculate address of PSP from MSP
  uint32_t* MSP_ptr = getMSPInitialLocation();
  uint32_t PSP_adr = (uint32_t) MSP_ptr - offset; //do arithemtic in integers

  //check if PSP address is a number divisible by 8
  if(PSP_adr%8 != 0){
    PSP_adr = PSP_adr+sizeof(uint32_t); //add 4 to address to ensure valid address for the stack
  }

  //assign PSP_ptr to point to PSP_adr
  uint32_t* PSP_ptr = (uint32_t*) PSP_adr;
  printf("%08x\n", (uint32_t) PSP_ptr);
  return PSP_ptr;
}


//set the value of PSP to threadStack and ensure that the microcontroller is using that value by changing the CONTROL register
void setThreadingWithPSP (uint32_t* threadStack){
  __set_PSP((uint32_t) threadStack);
  __set_CONTROL(1<<1);
}
```