EMBEDDED SYSTEM PROGRAMMING - 1.1

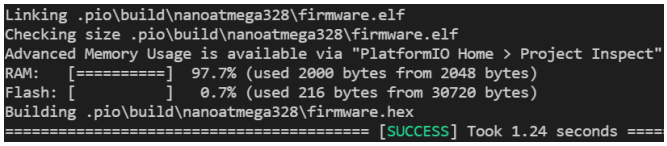
August 20, 2021

Sindri Thor Jonsson

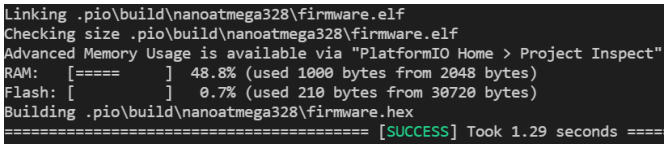
1 PART

TASK: Measure the memory footprint of an array of 1000 integers of default size (int) as well as size 8, 16, 32, 64, using the size information printed at the end of the compilation. In each case, indicate whether or not the available processor memory is exceeded. From the above measurements, infer how many bytes the compiler allocates for a single int variable is on this processor.

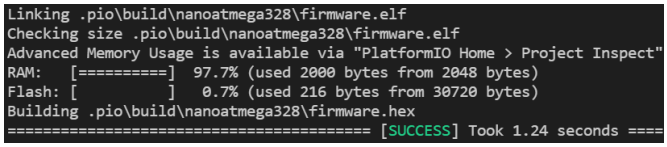
SOLUTION: In C++ we can declare which size of integer we want to use (int8_t, int16_t, int32_t, int64_t). When using platform IO the memory usage is written to the terminal after building.
The available processor memory (RAM) on the Arduino Nano is 2048 bytes.
These are the results of creating an array of 1000 integers of the sizes (8,16,32,64bits):



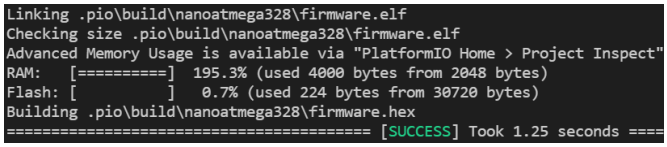
(a) Default size int



(b) 8-bit int



(a) 16-bit int



(b) 32-bit int

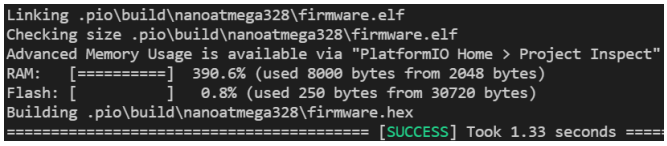


Figure 3: 64-bit int

From these results we can see that the memory footprint from the int types, the compiler allocates 1 byte for each the 8 bit int. 2 bytes for the default int size which is the 16bit int and 4 bytes for each int of 64b it size.

When creating the array of 1000 integers in 32bit or 64bit size the available processor memory is exceeded.

2 PART

GLOBAL: Using standard integer array of 1000 integers and global declaration with a return statement of the array in the main function the RAM allocation is 2000bytes + the fash memory usage increases by 2000bytes.
Using constant double did not add anything to the flash nor RAM usage

LOCAL: Using standard integer array of 1000 integers and local declaration with a return statement of the array in the main function the RAM and flash memory allocated does not change. Static int array declared locally added 2000bytes of RAM usage. Dynamically allocated integer array of 1000 integers allocated only 10bytes of RAM but around 800bytes of flash memory. Im pretty sure these results are not correct or making sense but i did not find a fix unfortunately.