info.cu

```
#include <cuda.h>
#include <cuda_runtime_api.h>
#include <stdio.h>
#include <stdlib.h>
// This function wraps the CUDA Driver API into a template function
template <class T>
inline void getCudaAttribute(T *attribute, CUdevice_attribute device_attribute,
                                                        int device)
{
    CUresult error = cuDeviceGetAttribute(attribute, device_attribute, device);
    if(CUDA_SUCCESS != error) {
        fprintf(stderr, "cuSafeCallNoSync( ) Driver API error = %04d from file <%s>, line %i.\n", err
or, __FILE__, __LINE__);
        exit(-1);
    }
}

int main()
{
    printf("CUDA Version %i\n", CUDA_VERSION);
    printf("CUDA Version %s\n", VERSION_STATUS);
    printf("\nDriver\n");

    cuInit(0);

    int driverVersion, runtimeVersion;
    cudaDriverGetVersion(&driverVersion);
    cudaRuntimeGetVersion(&runtimeVersion);
    printf("   CUDA Version : %i\n", driverVersion);
    printf("   Runtime version : %i\n", runtimeVersion);

    int deviceCount = 0;
    cudaError_t error_id = cudaGetDeviceCount(&deviceCount);
    printf("   Number of device : %i\n", deviceCount);

    int value;
    cudaDeviceProp prop;
    for(int id = 0; id < deviceCount; id++) {
        cudaGetDeviceProperties(&prop, id);
        printf("      Device Name : %s\n", prop.name);
        printf("      Total global mem : %ld\n", prop.totalGlobalMem);
        printf("      Total Constant Mem : %ld\n", prop.totalConstMem);
        printf("\n      Attribute(%i)\n", id+1);

        getCudaAttribute<int>(&value, CU_DEVICE_ATTRIBUTE_MAX_THREADS_PER_BLOCK, id);
        printf("         Max Threads per Block : %i\n", value);
        getCudaAttribute<int>(&value, CU_DEVICE_ATTRIBUTE_MAX_BLOCK_DIM_X, id);
        printf("         Max Block DIM(x) : %i\n", value);
        getCudaAttribute<int>(&value, CU_DEVICE_ATTRIBUTE_MAX_BLOCK_DIM_Y, id);
        printf("         Max Block DIM(y) : %i\n", value);
        getCudaAttribute<int>(&value, CU_DEVICE_ATTRIBUTE_MAX_BLOCK_DIM_Z, id);
        printf("         Max Block DIM(z) : %i\n", value);
        getCudaAttribute<int>(&value, CU_DEVICE_ATTRIBUTE_TOTAL_CONSTANT_MEMORY, id);
        printf("         Total Constant Memory : %i\n", value);
        getCudaAttribute<int>(&value, CU_DEVICE_ATTRIBUTE_WARP_SIZE, id);
        printf("        Warp size : %i\n", value);
        getCudaAttribute<int>(&value, CU_DEVICE_ATTRIBUTE_MEMORY_CLOCK_RATE, id);
        printf("        Clock Rate : %i\n", value);
        getCudaAttribute<int>(&value, CU_DEVICE_ATTRIBUTE_GLOBAL_MEMORY_BUS_WIDTH, id);
        printf("        Memory Bus Width : %i\n", value);
        getCudaAttribute<int>(&value, CU_DEVICE_ATTRIBUTE_L2_CACHE_SIZE, id);
        printf("        L2 Cache Size : %i\n", value);
    }
    return EXIT_SUCCESS;
}
```