## Aufgabe 1:

***erase 1:2-127:***
Der Einser gibt die Flash Bank an in der gelöscht werden soll. 2-127 geben die Sektoren an die gelöscht werden. In den Sektoren 2-127 befinden sich der Kernel und das Filesystem, diese werden gelöscht, damit später Windows CE auf das Board gespielt werden kann.

**SREC File Format:**
Das SREC Format kann im Gegensatz zu Binärdateien in einem Texteditor gelesen werden, da es aus ASCII Zeichen besteht. Zusätzlich enthält es noch eine Checksumme zur Überprüfung, ob die Übertragung auf ein Board gelungen ist. Im Gegensatz zu Binärdateien braucht die SREC Datei mehr Speicher.

***loads 0xa0000000:***
Der Parameter *0xa0000000* gibt den Offset im Speicher an, an dem die Datei gesendet werden soll.

***erase 1:0:***
Der gelöschte Sektor liegt im Adressbereich 0x00000000 - 0x00080000

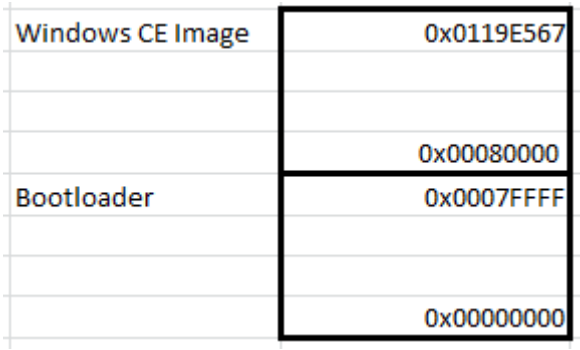
***cp.b 0xa0000000 0x00000000 2b30c:***
Das *.b* steht dafür, dass Byteweise kopiert wird. 0xa0000000 ist die Startadresse, 0x00000000 ist die Zieladresse und 2b30c ist die Größe des zu kopierenden Bereichs. Der Bootloader ist 0x2b30c groß.

## Aufgabe 2:

**Bootload und Windows CE:**

*Src IP 192.168.113.100 Port 0400  Dest IP 192.168.113.101 Port D6F3*
*EthDown::TFTPD_OPEN::boot.bin*
*-EbootSendBootmeAndWaitForTftp*
*FLASH download [ 0x00080000 ==> 0x0119E567 ]*
*<\r><\n>*
*INFO: FlashErase: 0x00080000 to 0x011BFFFF.  Please wait:  0%  1%  2%  4%  5%  7%*
*8%  10%  11%  13%  14%  16%  17%  18%  20%  21%  23%  24%  26%  27%*
*29%  30%  32%  33%  35%  36%  37%  39%  40%  42%  43%  45%  46%  48%*
*49%  51%  52%  54%  55%  56%  58%  59%  61%  62%  64%  65%  67%  68%*
*70%  71%  73%  74%  75%  77%  78%  80%  81%  83%  84%  86%  87%  89%*
*90%  92%  93%  94%  96%  97%  99%  100%*
*INFO: FlashWrite: 0x00080000 to 0x0119E567.  Please wait:  0%  1%  2%  4%  5%  7%*
*8%  10%  11%  13%  14%  16%  17%  18%  20%  21%  23%  24%  26%  27%*
*29%  30%  32%  33%  35%  36%  37%  39%  40%  42%  43%  45%  46%  48%*
*49%  51%  52%  54%  55%  56%  58%  59%  61%  62%  64%  65%  67%  68%*
*70%  71%  73%  74%  75%  77%  78%  80%  81%  83%  84%  86%  87%  89%*
*90%  92%  93%  94%  96%  97%  99%  100%*
*ROMHDR at Address 80100044h*
*WinCE image*
*Jumping to image at 0xA0101000*
*Got EDBG_CMD_JUMPIMG*
*Got EDBG_CMD_CONFIG, flags:0x00000000*

Der Bootloader liegt im Bereich 0x00000000 - 0x00080000. Das Windows CE Image wird in den Bereich 0x00080000 - 0x0119E567 kopiert.




## Aufgabe 3:

**Konsolenausgabe:**

Hello World!

## Aufgabe 4:

Es muss darauf geachtet werden, dass alle verwendeten Funktionen auf Windows CE zur Verfügung stehen. In diesem Beispiel sind uns jedoch keine Besonderheiten aufgefallen.

**EProcessData.h:**

```
#ifndef EPROCESSDATA_H
#define EPROCESSDATA_H

enum ProcessDataType {Speed, Temp, Voltage};

struct ProcessData
{
	ProcessDataType type;
	int value;
};

int const RPMLow = 500;
int const RPMHigh = 3000;
size_t const RPMIntervall = 1000;

int const TempLow = 20;
int const TempHigh = 125;
size_t const TempIntervall = 5000;

int const VoltageLow = -500;
int const VoltageHigh = 500;
size_t const VoltageIntervall = 500;

#endif
```

**SensorThread.h:**

```
#ifndef ThreadImpl_INCLUDED
#define ThreadImple_INCLUDED

#include "ThreadBase.h"
#include "EProcessData.h"
#include "TSQueue.h"

class SensorThread : public ThreadBase
{

public:
	SensorThread();
	virtual ~SensorThread();

	virtual void Init(int Number, ProcessDataType type, TSQueue* queue, int
ThreadPrio = THREAD_PRIORITY_NORMAL);

	virtual int Run();

private:
	int ThreadPriority;
	int ThreadNum;
	bool IsInitialized;

	ProcessDataType Type;
	TSQueue* Queue;
};

#endif
```

**SensorThread.cpp:**

```
#include <cassert>
#include <iostream>
#include "SensorThread.h"
#include "RandomGen.h"

SensorThread::SensorThread()
{
	assert(ThreadBase::IsThreadCreated());

	IsInitialized = false;
	ThreadNum = 0;

	ThreadPriority = GetThreadPriority(GetThreadHdl());
	assert(ThreadPriority != THREAD_PRIORITY_ERROR_RETURN);
}

SensorThread::~SensorThread()
{

}

void SensorThread::Init(int Number, ProcessDataType type, TSQueue* queue, int Prio)
{
	ThreadNum = Number;
	ThreadPriority = Prio;

	SetThreadPriority(GetThreadHdl(), ThreadPriority);

	Type = type;
	Queue = queue;

	IsInitialized = true;
}

int randLimited(int limit)
{
	return rand() % limit;
}

int SensorThread::Run()
{
	assert(IsInitialized);
	if (!IsInitialized)
	{
		return 0;				// specify error code
	}

	// ---- enter your personal code here ----- //

	int Low = 0;
	int High = 0;
	size_t Intervall = 0;

	switch(Type)
	{
	case Speed:
		{
			Low = RPMLow;
			High = RPMHigh;
			Intervall = RPMIntervall;
			break;
		}
```

```
	case Temp:
		{
			Low = TempLow;
			High = TempHigh;
			Intervall = TempIntervall;
			break;
		}
	case Voltage:
		{
			Low = VoltageLow;
			High = VoltageHigh;
			Intervall = VoltageIntervall;
			break;
		}
	}

	while (true)
	{
		Queue->PushBack(rgen::GetRandVal(Low, High));
		Sleep(Intervall);
	}

	// ---- end of personal code

	delete this;

	return 0;
}
```

**main.cpp:**

```
#include <windows.h>
#include <iostream>
#include "SensorThread.h"

using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
	SensorThread * pThread1 = new SensorThread;
	SensorThread * pThread2 = new SensorThread;
	SensorThread * pThread3 = new SensorThread;

	TSQueue RPMQueue;
	TSQueue TempQueue;
	TSQueue VoltageQueue;

	pThread1->Init(1, Speed, &RPMQueue);
	pThread2->Init(2, Temp, &TempQueue);
	pThread3->Init(3, Voltage, &VoltageQueue);

	int const n = 3;
	HANDLE Hdl[n];
	Hdl[0] = pThread1->GetDuplicateHdl();
	Hdl[1] = pThread2->GetDuplicateHdl();
	Hdl[2] = pThread3->GetDuplicateHdl();

	pThread1->Start();
	pThread2->Start();
	pThread3->Start();
```

```
// do something in main thread
	while(true)
	{
		if (RPMQueue.GetSize() > 0)
		{
			cout << "Speed [rpm]: " << RPMQueue.GetFront() << endl;
			RPMQueue.PopFront();
		}
		if (TempQueue.GetSize() > 0)
		{
			cout << "Temp [deg C]: " << TempQueue.GetFront() << endl;
			TempQueue.PopFront();
		}
		if (VoltageQueue.GetSize() > 0)
		{
			cout << "V diff. [mV]: " << VoltageQueue.GetFront() << endl;
			VoltageQueue.PopFront();
		}

		Sleep(50);
	}

	return 0;
}
```

**Konsolenausgabe:**

Temp [deg C]: 111
V diff. [mV]: 467
V diff. [mV]: 183
Speed [rpm]: 1481
V diff. [mV]: -347
V diff. [mV]: 378
Speed [rpm]: 1165
V diff. [mV]: 322
V diff. [mV]: 82
Speed [rpm]: 1243
V diff. [mV]: -309
V diff. [mV]: -322