Linker Symbole in C/C++ verwenden

Werden im Linker File (Linker Control File) Symbole für z.B. bestimmte Adresse definiert, so können diese nicht nur im Assembler, sondern auch im C/C++ Code zugegriffen und verwendet werden. Allerdings muss hierbei ein wichtiges Detail berücksichtigt werden.

Um im Linker-File definierte Symbole verwenden zu können, müssen folgende Schritte berücksichtigt werden:

1. **Definition des Symbols im Linker-File**
   Im Linker File können Symbole ohne vorherige (Typ-)Deklaration erstellt werden. Hierzu wird einfach an der entsprechenden Stelle in einer Section einem Symbolname ein Wert zugewiesen. (z.B. die aktuelle Adresse)
   ```
   _top_of_stack = .;
   ```
   Hier wird z.B. dem Symbol _top_of_stack die aktuelle Adresse in der Section Verarbeitung zugewiesen. Aber es können auch außerhalb der Sections Symbole definiert werden. Wie z.B. die Stackgröße in folgendem Beispiel:
   ```
   _size_of_stack = 0x100;
   ```
2. **Deklaration der Variablen an entsprechender Stell im C/C++ Code**
   Nach der Definition der Symbole im Linker File, müssen diese Symbole noch als Variablen innerhalb des C/C++ Code bekannt gemacht werden. Hierzu wird zunächst eine extern-Variable deklariert
   ```
   extern uint32_t _top_of_stack;
   ```
   Damit ist das Symbol auf für den C/C++ Compiler deklariert. Als Datentyp wird uin32_t verwendet, also ein vorzeichenloser 32-Bit Integer, da das Symbol eine 32 Bit Adresse aus dem Link-Prozess darstellt.
3. **Deklaration eines Zeigers auf den Speicher der Adresse**
   Das Symbol aus dem Linker-File ist zunächst wirklich nur ein Symbol und stellt einen Namen für einen Adress-Wert dar. Damit diese Adresse in C/C++ verwendet werden kann, ist es zweckmäßig einen Pointer zu deklarieren. Dieser bekommt als Wert die „Adresse des Symbols“ zu gewiesen und zeigt damit auf die vom Symbol definierte Adresse:
   ```
   uint32_t* pTopOfStack = (uint32_t*)&_top_of_stack;
   ```
   Nun kann mit dem Pointer direkt auf den von dem Symbol repräsentierten Speicher zugegriffen werden. Zu beachten ist bei der Pointer Deklaration, dass hier ebenfalls uint32_t als Datentyp verwendet wird, und zusätzlich noch ein Type-Cast auf uint32_t* erfolgt. Grundsätzlich könnte hier .z.B. auch ein Pointer auf ein uint8_t (also Byte) deklariert werden. Dies ist natürlich abhängig von der Art des Zugriffs.