

Programme mit dem lcc18 für den SpartanMC 18 erstellen

Der LCC18 kann für drei unterschiedliche Speicherkonfigurationen ein C-Programm übersetzen. Dabei wird in Variante eins mit dem Schalter **-target=spartan/hw** eine Konfiguration erstellt, die direkt auf der Hardware und ohne zusätzliche Hilfsprogramme abgearbeitet werden kann. Sie muss alle benötigten Interrupts behandeln und kann auch eigene TRAP Programme enthalten. Eine zweite Variante entsteht mit dem Schalter **-target=spartan/mon** in der dem Anwender alle Systemaufrufe und die Interrupt Behandlung des Testmonitors zur besseren Einarbeitung zur Verfügung stehen. In der dritten Variante, die mit dem Schalter

-target=spartan/moni gestartet wird, kann der Programmierer noch auf die Systemaufrufe des Monitor zugreifen, muss aber eine eigene Behandlung der benötigten Interrupts realisieren. Die Varianten zwei und drei sind vor allem für die Testphase bei der Erstellung eines Programms gedacht, während die Variante eins zur Erstellung der endgültigen Software geeignet ist, die beim Einschalten des SpartanMC 18 oder nach RESET immer selbstständig gestartet wird. Im folgenden sollen nun die Speicheraufteilungen der drei Varianten genauer dargestellt werden.

1. Variante (-target=spartan/hw)

Es wird Code ab der Adresse 0x00000 erstellt. Auf der Adresse 0 befindet sich ein JMP zum Start der Applikation. Ihm folgen bis zu 28 JMP Befehle für den Einsprung in TRAP Programme, die für die Applikation erstellt wurden. Die benötigten TRAP Einsprünge müssen in der Datei **conf_con.inc** in der Konstante **en_trap** gewählt werden. Für jeden der 28 TRAP Einsprünge ist in der Konstanten ein Bit vorgesehen, welches die Übersetzung des JMP freigibt, wenn es den Wert 1 hat. Damit kann jeder TRAP individuell freigegeben werden. Für jeden freigegebenen TRAP muss aber im C-Programm eine Funktion vorhanden sein. Die Namen der Funktionen sind fest vorgegeben und müssen **trap01** bis **trap28** lauten. Im Assemblerprogramm, welches bei der Übersetzung des C-Programms entsteht sind das dann die Namen **_trap01** bis **_trap28**. Nach den TRAP Einsprünge wird der Systemstart von Adresse 0 fortgesetzt. Hier wird Register 0 des SpartanMC 18 als Stackpointer initialisiert und danach zur **main** Funktion des C-Programms gesprungen. Hinter dem JMP zu der Assemblermarke **_main** folgt dann die Interrupt Behandlung für das Programm. Es können bis zu 16 Interrupt Einsprünge realisiert werden. Die benötigten Interrupts müssen auch in der Datei **conf_con.inc** freigegeben werden. Dafür gibt es dort die Konstante **i_bits**, welche die Anzahl der implementierten Interrupt Quellen einschließlich des Signals **interrupt_lowaktiv** enthalten muss. Diese Konstante wird im SystemBuilder bei der Konfiguration des Interruptkontrollers festgelegt. Werden keine Interrupts benötigt, dann wird auch im Zielprogramm kein einziger Befehl installiert. Die Konstante **i_bits** muss dann den Wert 0 haben. Hinter den Befehlen zur Interrupt Behandlung beginnen dann die Datenfelder des C-Programms gefolgt vom Stack. Nach dem Stack beginnen dann die Funktionen des C-Programms gefolgt von den Macros, die der Compiler mit der Datei **compiler.inc** immer mit einbindet. Zum Test wurde das Programm **quicksort_hw.c** übersetzt. Neben dem eigentlichen Programm wurden noch drei leere Funktionen zum Test der Verwendung von ISR und TRAP im Programm erstellt.

```
/*#include <stdio.h>*/
```

```
/* Vom Quicksort zu sortierende Daten */
```

```
unsigned int data[20] = {3,2,10,11,5,4,7,14,16,8,1,9,17,18,19,6,12,20,13,15};
```

```
/* alle nicht benoetigten isr und trap funktionen muessen im Include  
no_action.inc abgefangen werden. Dieser wird beim Start des Compiler  
mit dem Schalter -wf-include-asm=no_action.inc mit eingebunden. Oder sie  
muessen in der Datei conf_con.inc in den Konstanten i_bits und en_trap  
gesperrt werden. */
```

Programme mit dem lcc18 für den SpartanMC 18 erstellen

```

/* Funktionen zur Realisierung von TRAP Einspruengen. */

unsigned int trap01() {
    return 0;
}

/* Funktionen zur Interrupt Behandlung */

void isr00() {
}

void isr01() {
}

/* Programme fuer die Quicksort Funktion */

void swap(int x, int y) {
    unsigned int t;

    t=data[x];
    data[x]=data[y];
    data[y]=t;
}

void quicksort (unsigned int l, unsigned int r) {
    unsigned int pivot, i, j;
    i = l;
    j = r;
    pivot = data[(l+r)/2];

    while (i <= j) {
        /*Suche erstes Element welches größer ist als das Pivotelement */
        while (data[i] < pivot) i++;
        /*Suche erstes Element welches kleiner ist als das Pivotelement */
        while (data[j] > pivot) j--;

        if (i <= j) swap(i++,j--);
    }
    /*if (data[i] > pivot) swap(r,i);*/

    if (l < j) quicksort (l, j);
    if (i < r) quicksort (i, r);
}

unsigned int data[];

int main() {
    /*printf ("Input data[] =
(%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d)\r\n",
    data[0],data[1],data[2],data[3],data[4],
    data[5],data[6],data[7],data[8],data[9],
    data[10],data[11],data[12],data[13],data[14],
    data[15],data[16],data[17],data[18],data[19]);*/

    quicksort(0, 19);
    /*while(1); */

    /*printf ("Output data[] =
(%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d)\r\n",
    data[0],data[1],data[2],data[3],data[4],
    data[5],data[6],data[7],data[8],data[9],
    data[10],data[11],data[12],data[13],data[14],
    data[15],data[16],data[17],data[18],data[19]);*/
    return 0;
}

/*
*Quelle Wikipedia:
*
* funktion quicksort(links, rechts)
* falls links < rechts dann
* teiler := teile(links, rechts)
* quicksort(links, teiler-1)
* quicksort(teiler+1, rechts)
* ende
*ende
*
* funktion teile(links, rechts)

```

Programme mit dem lcc18 für den SpartanMC 18 erstellen

```

*      i := links
*      // Starte mit j links vom Pivotelement
*      j := rechts - 1
*      pivot := daten[rechts]
*
*      wiederhole
*
*          // Suche von links ein Element, welches größer dem Pivotelement ist
*          wiederhole solange daten[i] ≤ pivot und i < rechts
*              i := i + 1
*          ende
*
*          // Suche von rechts ein Element, welches kleiner dem Pivotelement ist
*          wiederhole solange daten[j] > pivot und j > links
*              j := j - 1
*          ende
*
*          falls i < j dann tausche daten[i] mit daten[j]
*
*      solange i < j // solange i an j nicht vorbeigelaufen ist
*
*      // Tausche Pivotelement (daten[rechts]) mit neuer endgültigen Position (daten[i])
*
*      wenn daten[i] > pivot
*          tausche daten[i] mit daten[rechts]
*      ende
*
*      // gib die Position des Pivotelements zurück
*
*      antworte i
*  ende
*
*/

```

In der folgenden Tabelle wird die Belegung des Speichers für dieses Programm aufgelistet. Die Daten sind der Datei **quicksort_hw.1st** zu entnehmen, welche durch den Assembler erzeugt wird.

Programmadresse	Datenadresse	Belegt mit ...	Bemerkung
0x00000 - 0x0001c	0x00000 - 0x00038	TRAP00 = System Start und TRAP01 – TRAP28	Nur TRAP01 wird genutzt.
0x0001d - 0x0001f	0x0003a - 0x0003e	Laden r0 mit Stack Anfang und Start MAIN.	
0x00020 - 0x00037	0x00040 - 0x0006e	Aufrufe für ISR00 und ISR01	ISR00 und ISR01 werden verwendet.
0x00038 - 0x0004b	0x00070 - 0x00097	unsigned int data[20]	Zu sortierendes Daten Feld.
0x0004c - 0x00240	0x00098 - 0x00481	STACK	
0x00241 - 0x00242	0x00482 - 0x00484	unsigned int trap01()	
0x00243	0x00486	void isr00()	
0x00244	0x00488	void isr01()	
0x00245 - 0x0025a	0x0048a - 0x004b4	void swap(...)	
0x0025b - 0x0029e	0x004b6 - 0x0053c	void quicksort (...)	
0x0029f - 0x002a5	0x0053e - 0x0054a	int main()	Wird mit einer Endlosschleife abgeschlossen.
0x002a6 - 0x002b6	0x0054c - 0x0056c	DIVU Macro	Aus compiler.inc
0x002b7 - 0x002e6	0x0056e - 0x005cc	DIV Macro	Aus compiler.inc
0x002e7 - 0x002ff	0x005ce - 0x005fe	SLL Macro	Aus compiler.inc
0x00300 - 0x00318	0x00600 - 0x00630	SRL Macro	Aus compiler.inc
0x00319 - 0x00331	0x00632 - 0x00662	SRA Macro	Aus compiler.inc

Programme mit dem lcc18 für den SpartanMC 18 erstellen

2. Variante (-target=spartan/mon)

Bei dieser Variante sind keine eigenen Interrupt - oder TRAP - Programme möglich. Es können die TRAP Systemaufrufe des Monitors genutzt werden, und sollte versehentlich ein Interrupt ausgelöst werden, so wird er vom Monitor angezeigt. Beim betätigen des Tasters für das Signal **interrupt_lowactiv** wird die aktuelle Programmschleife mit Ihren Registerinhalten protokolliert. Zum Test wurde das Programm **quicksort.c** übersetzt. Die **quicksort.spho** Datei wird im Monitor mit dem Kommando **L0** geladen und mit **G + ENTER** gestartet.

```
/*#include <stdio.h>*/

unsigned int data[20] = {3,2,10,11,5,4,7,14,16,8,1,9,17,18,19,6,12,20,13,15};

void swap(int x, int y) {
    unsigned int t;

    t=data[x];
    data[x]=data[y];
    data[y]=t;
}

void quicksort (unsigned int l, unsigned int r) {
    unsigned int pivot, i, j;
    i = l;
    j = r;
    pivot = data[(l+r)/2];

    while (i <= j) {
        /*Suche erstes Element welches größer ist als das Pivotelement */
        while (data[i] < pivot) i++;
        /*Suche erstes Element welches kleiner ist als das Pivotelement */
        while (data[j] > pivot) j--;

        if (i <= j) swap(i++,j--);
    }
    /*if (data[i] > pivot) swap(r,i);*/

    if (l < j) quicksort (l, j);
    if (i < r) quicksort (i, r);
}

unsigned int data[];

int main() {
    /*printf ("Input data[] =
(%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d)\r\n",
    data[0],data[1],data[2],data[3],data[4],
    data[5],data[6],data[7],data[8],data[9],
    data[10],data[11],data[12],data[13],data[14],
    data[15],data[16],data[17],data[18],data[19]);*/

    quicksort(0, 19);
    /*while(1); */

    /*printf ("Output data[] =
(%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d)\r\n",
    data[0],data[1],data[2],data[3],data[4],
    data[5],data[6],data[7],data[8],data[9],
    data[10],data[11],data[12],data[13],data[14],
    data[15],data[16],data[17],data[18],data[19]);*/
    return 0;
}

/*
*Quelle Wikipedia:
*
* funktion quicksort(links, rechts)
* falls links < rechts dann
*     teiler := teile(links, rechts)
*     quicksort(links, teiler-1)
*     quicksort(teiler+1, rechts)
* ende
*ende
*
*
*/
```

Programme mit dem lcc18 für den SpartanMC 18 erstellen

```

* funktion teile(links, rechts)
*   i := links
*   // Starte mit j links vom Pivotelement
*   j := rechts - 1
*   pivot := daten[rechts]
*
*   wiederhole
*
*       // Suche von links ein Element, welches größer dem Pivotelement ist
*       wiederhole solange daten[i] ≤ pivot und i < rechts
*           i := i + 1
*       ende
*
*       // Suche von rechts ein Element, welches kleiner dem Pivotelement ist
*       wiederhole solange daten[j] > pivot und j > links
*           j := j - 1
*       ende
*
*       falls i < j dann tausche daten[i] mit daten[j]
*
*   solange i < j // solange i an j nicht vorbeigelaufen ist
*
*   // Tausche Pivotelement (daten[rechts]) mit neuer endgültigen Position (daten[i])
*
*   wenn daten[i] > pivot
*       tausche daten[i] mit daten[rechts]
*   ende
*
*   // gib die Position des Pivotelements zurück
*
*   antworte i
* ende
*/

```

In der folgenden Tabelle wird die Belegung des Speichers für dieses Programm aufgelistet. Die Daten sind der Datei **quicksort.lst** zu entnehmen, welche durch den Assembler erzeugt wird.

Programmadresse	Datenadresse	Belegt mit ...	Bemerkung
0x00000 - 0x0001f	0x00000 - 0x0003e	TRAP00 = System Start und TRAP01 – TRAP28	Systemaufrufe des Monitors.
0x00020 - 0x0004e	0x00040 - 0x0009d	Aufrufe für ISR00 bis ISR15	ISR Behandlung des Monitors.
0x00050	0x000a0	Start MAIN.	
0x00051 - 0x00064	0x000a2 - 0x000c8	unsigned int data[20]	Zu sortierendes Daten Feld.
0x00065 - 0x0007a	0x000ca - 0x000f4	void swap(...)	
0x0007b - 0x000be	0x000f6 - 0x0017c	void quicksort (...)	
0x000bf - 0x000c6	0x0017e - 0x0018c	int main()	
0x000c7 - 0x000d7	0x0018e - 0x001ae	DIVU Macro	Aus compiler.inc Aus compiler.inc Aus compiler.inc Aus compiler.inc Aus compiler.inc Monitor User Stack
0x000d8 - 0x00107	0x001b0 - 0x0020e	DIV Macro	
0x00108 - 0x0011f	0x00210 - 0x0023e	SLL Macro	
0x00120 - 0x00137	0x00240 - 0x0027e	SRL Macro	
0x00138 - 0x0014f	0x00270 - 0x0029e	SRA Macro	
0x00150 - 0x00c00	0x002a0 - 0x01800	STACK	
0x00c01 - 0x00d20	0x01802 - 0x01a41	Monitor Daten	
0x00d21 - 0x01045	0x01a42 - 0x0208a	Monitor Programm	

Programme mit dem lcc18 für den SpartanMC 18 erstellen

3. Variante (-target=spartan/moni)

Bei dieser Variante sind eigenen Interrupt – aber keine TRAP - Programme möglich. Es können die TRAP Systemaufrufe des Monitors genutzt werden. Zum Test wurde das Programm **quicksort_moni.c** übersetzt. Die **quicksort_moni.spho** Datei wird im Monitor mit dem Kommando **LO** geladen und mit **G + ENTER** gestartet.

```
/*#include <stdio.h>*/

/* Vom Quicksort zu sortierende Daten */

unsigned int data[20] = {3,2,10,11,5,4,7,14,16,8,1,9,17,18,19,6,12,20,13,15};

/* alle nicht benoetigten isr und trap funktionen muessen im Include
no_action.inc abgefangen werden. Dieser wird beim Start des Compiler
mit dem Schalter -wf-include-asm=no_action.inc mit eingebunden. Oder sie
muessen in der Datei conf_con.inc in den Konstanten i_bits und en_trap
gesperrt werden. */

/* Funktionen zur Interrupt Behandlung */

void isr00() {
}

void isr01() {
}

/* Programme fuer die Quicksort Funktion */

void swap(int x, int y) {
    unsigned int t;

    t=data[x];
    data[x]=data[y];
    data[y]=t;
}

void quicksort (unsigned int l, unsigned int r) {
    unsigned int pivot, i, j;
    i = l;
    j = r;
    pivot = data[(l+r)/2];

    while (i <= j) {
        /*Suche erstes Element welches größer ist als das Pivotelement */
        while (data[i] < pivot) i++;
        /*Suche erstes Element welches kleiner ist als das Pivotelement */
        while (data[j] > pivot) j--;

        if (i <= j) swap(i++,j--);
    }
    /*if (data[i] > pivot) swap(r,i);*/

    if (l < j) quicksort (l, j);
    if (i < r) quicksort (i, r);
}

unsigned int data[];

int main() {
    /*printf ("Input data[] =
(%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d)\r\n",
    data[0],data[1],data[2],data[3],data[4],
    data[5],data[6],data[7],data[8],data[9],
    data[10],data[11],data[12],data[13],data[14],
    data[15],data[16],data[17],data[18],data[19]);*/

    quicksort(0, 19);
    /*while(1); */

    /*printf ("Output data[] =
(%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d)\r\n",
    data[0],data[1],data[2],data[3],data[4],
    data[5],data[6],data[7],data[8],data[9],
    data[10],data[11],data[12],data[13],data[14],
    data[15],data[16],data[17],data[18],data[19]);*/
    return 0;
}
```

Programme mit dem lcc18 für den SpartanMC 18 erstellen

```
/*
*Quelle Wikipedia:
*
* funktion quicksort(links, rechts)
*   falls links < rechts dann
*     teiler := teile(links, rechts)
*     quicksort(links, teiler-1)
*     quicksort(teiler+1, rechts)
*   ende
*ende
*
* funktion teile(links, rechts)
*   i := links
*   // Starte mit j links vom Pivotelement
*   j := rechts - 1
*   pivot := daten[rechts]
*
*   wiederhole
*
*     // Suche von links ein Element, welches größer dem Pivotelement ist
*     wiederhole solange daten[i] ≤ pivot und i < rechts
*       i := i + 1
*     ende
*
*     // Suche von rechts ein Element, welches kleiner dem Pivotelement ist
*     wiederhole solange daten[j] > pivot und j > links
*       j := j - 1
*     ende
*
*     falls i < j dann tausche daten[i] mit daten[j]
*
*   solange i < j // solange i an j nicht vorbeigelaufen ist
*
*   // Tausche Pivotelement (daten[rechts]) mit neuer endgültigen Position (daten[i])
*
*   wenn daten[i] > pivot
*     tausche daten[i] mit daten[rechts]
*   ende
*
*   // gib die Position des Pivotelements zurück
*
*   antworte i
* ende
*/
```

Programme mit dem lcc18 für den SpartanMC 18 erstellen

In der folgenden Tabelle wird die Belegung des Speichers für dieses Programm aufgelistet. Die Daten sind der Datei **quicksort_moni.lst** zu entnehmen, welche durch den Assembler erzeugt wird.

Programmadresse	Datenadresse	Belegt mit ...	Bemerkung
0x00000 - 0x0001e	0x00000 - 0x0003c	TRAP00 = System Start und TRAP01 – TRAP28	Systemaufrufe des Monitors.
0x0001f 0x00020 - 0x00037	0x0003e 0x00040 - 0x0006e	Start MAIN. Aufrufe für ISR00 und ISR01	ISR Behandlung.
0x00038 - 0x0004b	0x00070 - 0x00097	unsigned int data[20]	Zu sortierendes Daten Feld.
0x0004c 0x0004d 0x0004e - 0x00063 0x00064 - 0x000a7 0x000a8 - 0x000af	0x00098 0x0009a 0x0009c - 0x000c6 0x000c8 - 0x0014e 0x00150 - 0x0015e	void isr00() void isr01() void swap(...) void quicksort (...) int main()	Wird mit einem Rücksprung in den Monitor abgeschlossen.
0x000b0 - 0x000c0 0x000c1 - 0x000f0 0x000f1 - 0x00108 0x00109 - 0x00120 0x00121 - 0x00138	0x00160 - 0x00180 0x00182 - 0x001e0 0x001e2 - 0x00210 0x00212 - 0x00240 0x00242 - 0x00270	DIVU Macro DIV Macro SLL Macro SRL Macro SRA Macro	Aus compiler.inc Aus compiler.inc Aus compiler.inc Aus compiler.inc Aus compiler.inc
0x00139 - 0x00c00 0x00c01 - 0x00d20 0x00d21 - 0x01045	0x00272 - 0x01800 0x01802 - 0x01a41 0x01a42 - 0x0208a	STACK Monitor Daten Monitor Programm	Monitor User Stack