Hello,

I have some troubles with getting started my atmega32L . Same Sources run on atmega8.

0. My toolset is GNU-cc from download WINAVR 200030424_bin_install.exe from avrfreaks.net,
   modified by download of 2002-06-25_FREAKS.exe from avrfreaks.net,
   avr-studio 3.56 target-system is STK500

1. creating a small testprogramm using Timer0-int is running with this toolset for the target atmega8, but not atmega32.
The Problems is, that i suppose (because of the map-file) generated, that, when, in case of the atmega8-device the lib crtm32.0 is included. Then in the map file is before the 'ctors'-section a space left, i suppose for the vector-table. In case of the atmega32, there isn't, the section ctors is placed at location 0. I suppose, that thereforwe no int-function will be called.
Supposing that the switch from using the crtm8.o onto crtm32.o depends on the OUTPUT-ARCH given in the linker-script file, this seems to be problem in this lib. Can i rebuild it or is there a bug-fix or must I do anything else.

2. the makefile for this is

```
#
#
# simple make file for ucos-ii avr-gcc port
#
# AVR-GCC port version : 1.0    2001-04-02  Jesper Hansen (jesperh@telia.com)
#
#
#

CC              = avr-gcc
AS              = avr-gcc -x assembler-with-cpp
RM              = rm -f
RN              = mv
BIN             = avr-objcopy
OUT     = coff
ELFCOF  = elfcoff
ELFCOFF = objtool
CP      = cp



TRG     = test
TRGDIR  = ../OBJ
LSTDIR  = ../LST
INCDIR  = .
IDIR2   = $(AVR)/avr/include/avr
#INCDIR  = $(AVR)/avr/include/avr
LIBDIR  = $(AVR)/avr/lib
SHELL   = $(AVR)/bin/sh.exe
FORMAT  = srec
```

```
#
# Select the processor and appropriate linker script below
#

#
# At90S8515
#

#MCU    = at90s8515
#LDFLAGS = -T../source/avr85xx.x


#############

#
# Atmega103
#

#  MCU  = atmega103
#  #LDFLAGS = -T../source/avrmega103.x
#  LDFLAGS = -T../source/avr3.x

#############

#
# Atmega8
#

  MCU   = atmega8
  LDFLAGS = -T../source/avrmega8.x


##############################################################################
##############################################################################

#CPFLAGS = -E -g -O3 -Wall -Wstrict-prototypes -Wa,-ahlms=$(LSTDIR)/$(*F).lst
-mmcu=$(MCU)
#CPFLAGS = -g -Os -Wall -Wstrict-prototypes -Wa,-ahlms=$(LSTDIR)/$(*F).lst -mmcu=$(MCU)
CPFLAGS = -Os -Wall -Wstrict-prototypes -Wa,-ahlms=$(LSTDIR)/$(*F).lst -mmcu=$(MCU)
ASFLAGS = -Wa,-gstabs -mmcu=$(MCU)
LDFLAGS += -Wl,-Map=test.map,--cref -mmcu=$(MCU)

##############################################################################

CSRC    = test.c os_cpu_c.c ucos_ii.c
ASRC    = os_cpu_a.asm
OBJ     = $(CSRC:%.c=$(TRGDIR)/%.o) $(ASRC:%.asm=$(TRGDIR)/%.o)

##############################################################################

all:    $(TRG).obj $(TRG).elf $(TRG).rom $(TRG).eep $(TRG).hex $(TRG).cof

$(TRGDIR)/$(TRG).o: $(TRG).c
        $(CC) -c $(CPFLAGS) -I$(INCDIR) -o $@ $<

$(TRGDIR)/$(TRG).s : $(TRG).c
        $(CC) -S $(CPFLAGS) -I$(INCDIR) $< -o $@
```

```
$(TRGDIR)/os_cpu_c.o: ../source/os_cpu_c.c
	$(CC) -c $(CPFLAGS) -I$(INCDIR) -o $@ $<

$(TRGDIR)/os_cpu_a.o: ../source/os_cpu_a.asm
	$(AS) -c $(ASFLAGS) -I$(INCDIR) -o $@ $<

$(TRGDIR)/ucos_ii.o: ../../../source/ucos_ii.c
	$(CC) -c $(CPFLAGS) -I$(INCDIR) -o $@ $<

#############################################################################


%elf: $(OBJ)
	$(CC) $(OBJ) $(LIB) $(LDFLAGS) -o $@

%obj: %elf
	$(BIN) -O avrobj $< $@

%rom: %elf
	$(BIN) -O $(FORMAT) $< $@

%.hex: %.elf
	$(BIN) -O ihex -R .eeprom $< $@

%eep: %elf
	$(BIN) -j .eeprom --set-section-flags=.eeprom="alloc,load" -O $(FORMAT) $< $@

#%.cof: %.elf
#	$(ELFCOF) $< $(OUT) $@ $*.sym
#	$(CP) $(OUT)/$@ .
#	$(CP) $(OUT)/* .
##	$(CP) $(OUT)\\*sym .
##	$(CP) $(OUT)\\*S .

#%.cof: %.elf
#  objtool loadelf  test.elf  mapfile  test.map  writecof  test.cof
# WARUM AUCH IMMER : MANUELL FUNKTIONIERT DAS



#############################################################################

clean:
		rm -f $(TRGDIR)/*.o
		rm -f $(LSTDIR)/*.lst

cleanall:
		rm -f $(TRGDIR)/*.o
		rm -f $(LSTDIR)/*.lst
		rm -f *.map
		rm -f *.obj
		rm -f *.elf
		rm -f *.rom
		rm -f *.eep
```

3. Contents of avrmega8.x :

```
/* Default linker script, for normal executables */
OUTPUT_FORMAT("elf32-avr","elf32-avr","elf32-avr")
OUTPUT_ARCH(avr:4)
MEMORY
{
  /*
  text   (rx)   : ORIGIN = 0, LENGTH = 128K
  data   (rw!x) : ORIGIN = 0x800060, LENGTH = 0xffa0
  eeprom (rw!x) : ORIGIN = 0x810000, LENGTH = 64K
  */

  text   (rx)   : ORIGIN = 0, LENGTH = 8K
  data   (rw!x) : ORIGIN = 0x800060, LENGTH = 1K
  eeprom (rw!x) : ORIGIN = 0x810000, LENGTH = 512

}
SECTIONS
{
  /* Read-only sections, merged into text segment: */
  .hash          : { *(.hash)           }
  .dynsym        : { *(.dynsym)         }
  .dynstr        : { *(.dynstr)         }
  .gnu.version   : { *(.gnu.version)    }
  .gnu.version_d   : { *(.gnu.version_d)        }
  .gnu.version_r   : { *(.gnu.version_r)        }
  .rel.init      : { *(.rel.init)               }
  .rela.init     : { *(.rela.init)      }
  .rel.text      :
    {
      *(.rel.text)
      *(.rel.text.*)
      *(.rel.gnu.linkonce.t*)
    }
  .rela.text     :
    {
      *(.rela.text)
      *(.rela.text.*)
      *(.rela.gnu.linkonce.t*)
    }
  .rel.fini      : { *(.rel.fini)               }
  .rela.fini     : { *(.rela.fini)      }
  .rel.rodata    :
    {
      *(.rel.rodata)
      *(.rel.rodata.*)
      *(.rel.gnu.linkonce.r*)
    }
  .rela.rodata   :
    {
      *(.rela.rodata)
      *(.rela.rodata.*)
      *(.rela.gnu.linkonce.r*)
    }
  .rel.data      :
    {
      *(.rel.data)
      *(.rel.data.*)
      *(.rel.gnu.linkonce.d*)
    }
```

```
.rela.data     :
  {
    *(.rela.data)
    *(.rela.data.*)
    *(.rela.gnu.linkonce.d*)
  }
.rel.ctors     : { *(.rel.ctors)      }
.rela.ctors    : { *(.rela.ctors)     }
.rel.dtors     : { *(.rel.dtors)      }
.rela.dtors    : { *(.rela.dtors)     }
.rel.got       : { *(.rel.got)                }
.rela.got      : { *(.rela.got)               }
.rel.bss       : { *(.rel.bss)                }
.rela.bss      : { *(.rela.bss)               }
.rel.plt       : { *(.rel.plt)                }
.rela.plt      : { *(.rela.plt)               }
/* Internal text space or external memory */
.text :
{
  *(.vectors)
   __ctors_start = . ;
   *(.ctors)
   __ctors_end = . ;
   __dtors_start = . ;
   *(.dtors)
   __dtors_end = . ;
  *(.progmem.gcc*)
  *(.progmem*)
  . = ALIGN(2);
  *(.init0)  /* Start here after reset.  */
  *(.init1)
  *(.init2)  /* Clear __zero_reg__, set up stack pointer.  */
  *(.init3)
  *(.init4)  /* Initialize data and BSS.  */
  *(.init5)
  *(.init6)  /* C++ constructors.  */
  *(.init7)
  *(.init8)
  *(.init9)  /* Call main().  */
  ../OBJ/os_cpu_a.o (.text)
  ../OBJ/os_cpu_c.o (.text)
  *(.text)
  . = ALIGN(2);
  *(.text.*)
  . = ALIGN(2);
  *(.fini9)  /* _exit() starts here.  */
  *(.fini8)
  *(.fini7)
  *(.fini6)  /* C++ destructors.  */
  *(.fini5)
  *(.fini4)
  *(.fini3)
  *(.fini2)
  *(.fini1)
  *(.fini0)  /* Infinite loop after program termination.  */
   _etext = . ;
}  > text
.data   : AT (ADDR (.text) + SIZEOF (.text))
{
   PROVIDE (__data_start = .) ;
  *(.data)
```

```
    *(.gnu.linkonce.d*)
    . = ALIGN(2);
     _edata = . ;
     PROVIDE (__data_end = .) ;
  }  > data
  .bss  SIZEOF(.data) + ADDR(.data) :
  {
     PROVIDE (__bss_start = .) ;
    *(.bss)
    *(COMMON)
    . += 50;
     PROVIDE (__stack = . );
     PROVIDE (__bss_end = .) ;
  }  > data
   __data_load_start = LOADADDR(.data);
   __data_load_end = __data_load_start + SIZEOF(.data);
  /* Global data not cleared after reset.  */
  .noinit  SIZEOF(.bss) + ADDR(.bss) :
  {
     PROVIDE (__noinit_start = .) ;
    *(.noinit*)
     PROVIDE (__noinit_end = .) ;
     _end = . ;
     PROVIDE (__heap_start = .) ;
  }  > data
  .eeprom  :
        AT (ADDR (.text) + SIZEOF (.text) + SIZEOF (.data))
  {
    *(.eeprom*)
     __eeprom_end = . ;
  }  > eeprom
  /* Stabs debugging sections.  */
  .stab 0 : { *(.stab) }
  .stabstr 0 : { *(.stabstr) }
  .stab.excl 0 : { *(.stab.excl) }
  .stab.exclstr 0 : { *(.stab.exclstr) }
  .stab.index 0 : { *(.stab.index) }
  .stab.indexstr 0 : { *(.stab.indexstr) }
  .comment 0 : { *(.comment) }
  /* DWARF debug sections.
     Symbols in the DWARF debugging sections are relative to the beginning
     of the section so we begin them at 0.  */
  /* DWARF 1 */
  .debug          0 : { *(.debug) }
  .line           0 : { *(.line) }
  /* GNU DWARF 1 extensions */
  .debug_srcinfo  0 : { *(.debug_srcinfo) }
  .debug_sfnames  0 : { *(.debug_sfnames) }
  /* DWARF 1.1 and DWARF 2 */
  .debug_aranges  0 : { *(.debug_aranges) }
  .debug_pubnames 0 : { *(.debug_pubnames) }
  /* DWARF 2 */
  .debug_info     0 : { *(.debug_info) *(.gnu.linkonce.wi.*) }
  .debug_abbrev   0 : { *(.debug_abbrev) }
  .debug_line     0 : { *(.debug_line) }
  .debug_frame    0 : { *(.debug_frame) }
  .debug_str      0 : { *(.debug_str) }
  .debug_loc      0 : { *(.debug_loc) }
  .debug_macinfo  0 : { *(.debug_macinfo) }
}
```

4. Try to port to/run on atmega32L :
I modified the makefile by replacing the section for atmega8 by a section for atmega32, which lokks like this :
#############

```
#
# Atmega32
#

#  MCU  = atmega32
#  LDFLAGS = -T../source/avrmega32.x
```

well i think this isn't the error

the linker script file avrmega32.x looks like the avrmega3.x sshown above, except for this lines :

```
/* Default linker script, for normal executables */
/* Ursprung avr5.x */
OUTPUT_FORMAT("elf32-avr","elf32-avr","elf32-avr")
OUTPUT_ARCH(avr:5)
MEMORY
{
  /*
  text   (rx)   : ORIGIN = 0,    LENGTH = 16K
  data   (rw!x) : ORIGIN = (0x800000 + 0x60), LENGTH = 1K
  eeprom (rw!x) : ORIGIN = 0x810000, LENGTH = 512
  */
  text   (rx)   : ORIGIN = 0,    LENGTH = 32K
  data   (rw!x) : ORIGIN = (0x800000 + 0x60), LENGTH = 2K
  eeprom (rw!x) : ORIGIN = 0x810000, LENGTH = 1K
}
```

5. The Problems is, that i suppose (because of the map-file) generated, that, when, in case of the atmega8-device the lib crtm32.0 is included. Then in the map file is before the 'ctors'-section a space left, i suppose for the vector-table. In case of the atmega32, there isn't, the section ctors is placed at location 0. I suppose, that thereforwe no int-function will be called.
Supposing that the switch from using the crtm8.o onto crtm32.o depends on the OUTPUT-ARCH given in the linker-script file, this seems to be problem in this lib. Can i rebuild it or is there a bug-fix or must I do anything else.

6. I give the beginnings of the map-files :
a. for the at-mega8 :

```
.text           0x00000000     0x168c
 *(.vectors)
 .vectors       0x00000000       0x26
C:\avrgcc\bin\..\lib\gcc-lib\avr\3.2\..\..\..\..\avr\lib\avr4\crtm8.o
                0x00000000                __vectors
                0x00000000                __vector_default
                0x00000026                __ctors_start = .
 *(.ctors)
                0x00000026                __ctors_end = .
                0x00000026                __dtors_start = .
 *(.dtors)
                0x00000026                __dtors_end = .
 *(.progmem.gcc*)
 *(.progmem*)
                0x00000026                . = ALIGN (0x2)
```

```
 *(.init0)



b. for the at-mega32
.text           0x00000000     0x1766
 *(.vectors)
                0x00000000                __ctors_start = .
 *(.ctors)
                0x00000000                __ctors_end = .
                0x00000000                __dtors_start = .
 *(.dtors)
                0x00000000                __dtors_end = .
 *(.progmem.gcc*)
 *(.progmem*)
                0x00000000                . = ALIGN (0x2)
 *(.init0)
 .init0         0x00000000       0x50
C:\avrgcc\bin\..\lib\gcc-lib\avr\3.2\..\..\..\..\avr\lib\avr5\crtm32.o
```