

// Code Modified by Jerome, 2012 .

/*

Copyright (c) 2002, Marek Michalkiewicz <marekm@amelek.gda.pl>

Copyright (c) 2007, 2008 Eric B. Weddington

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

* Neither the name of the copyright holders nor the names of contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. */

/*
vector __vector_94
vector __vector_95
vector __vector_96

```

vector __vector_97
vector __vector_98
vector __vector_99
vector __vector_100
vector __vector_101
vector __vector_102
vector __vector_103
vector __vector_104
vector __vector_105
vector __vector_106
vector __vector_107
vector __vector_108
vector __vector_109
vector __vector_110
vector __vector_111
vector __vector_112
vector __vector_113
vector __vector_114
vector __vector_115
vector __vector_116
vector __vector_117
vector __vector_118
vector __vector_119
vector __vector_120
vector __vector_121
vector __vector_122
vector __vector_123
vector __vector_124

vector __vector_127*/

#include "macros.inc"

.macro vector name
.if (. - __vectors < _VECTORS_SIZE)
.weak \name
.set \name, __bad_interrupt
XJMP \name
.endif
.endm

.section .vectors,"ax",@progbits
.global __vectors
.func __vectors

__vectors:

```

XJMP __init

vector __vector_1
vector __vector_2
vector __vector_3
vector __vector_4
vector __vector_5
vector __vector_6
vector __vector_7
vector __vector_8
vector __vector_9
vector __vector_10
vector __vector_11
vector __vector_12
vector __vector_13
vector __vector_14
vector __vector_15
vector __vector_16
vector __vector_17
vector __vector_18
vector __vector_19
vector __vector_20
vector __vector_21
vector __vector_22
vector __vector_23
vector __vector_24
vector __vector_25
vector __vector_26
vector __vector_27
vector __vector_28
vector __vector_29
vector __vector_30
vector __vector_31
vector __vector_32
vector __vector_33
vector __vector_34
vector __vector_35
vector __vector_36
vector __vector_37
vector __vector_38
vector __vector_39
vector __vector_40
vector __vector_41
vector __vector_42
vector __vector_43
vector __vector_44

vector __vector_45
vector __vector_46
vector __vector_47
vector __vector_48
vector __vector_49
vector __vector_50
vector __vector_51
vector __vector_52
vector __vector_53
vector __vector_54
vector __vector_55
vector __vector_56
vector __vector_57
vector __vector_58
vector __vector_59
vector __vector_60
vector __vector_61
vector __vector_62
vector __vector_63
vector __vector_64
vector __vector_65
vector __vector_66
vector __vector_67
vector __vector_68
vector __vector_69
vector __vector_70
vector __vector_71
vector __vector_72
vector __vector_73
vector __vector_74
vector __vector_75
vector __vector_76
vector __vector_77
vector __vector_78
vector __vector_79
vector __vector_80
vector __vector_81
vector __vector_82
vector __vector_83
vector __vector_84
vector __vector_85
vector __vector_86
vector __vector_87
vector __vector_88
vector __vector_89
vector __vector_90

```

vector __vector_91
vector __vector_92
vector __vector_93

vector __vector_125
vector __vector_126

.endfunc

.text
.global __bad_interrupt
.func __bad_interrupt
__bad_interrupt:
.weak __vector_default
.set __vector_default, __vectors
XJMP __vector_default
.endfunc

.section .init0,"ax",@progbits
.weak __init
; .func __init
__init:

#ifdef __AVR_ASM_ONLY__
.weak __stack

/* By default, malloc() uses the current value of the stack pointer
   minus __malloc_margin as the highest available address.

   In some applications with external SRAM, the stack can be below
   the data section (in the internal SRAM - faster), and __heap_end
   should be set to the highest address available for malloc(). */
.weak __heap_end
.set __heap_end, 0
.set __stack, RAMEND
.section .init2,"ax",@progbits
clr __zero_reg__
out AVR_STATUS_ADDR, __zero_reg__
ldi r28,lo8(__stack)
ldi r29,hi8(__stack)
out AVR_STACK_POINTER_HI_ADDR, r29
out AVR_STACK_POINTER_LO_ADDR, r28

#ifdef __AVR_3_BYTE_PC__

```

```

        ldi    r16, hh8(pm(__vectors))
        out   _SFR_IO_ADDR(EIND), r16
#endif /* __AVR_3_BYTE_PC__ */

        out   AVR_RAMPD_ADDR, __zero_reg__
        out   AVR_RAMPX_ADDR, __zero_reg__
        out   AVR_RAMPY_ADDR, __zero_reg__
        out   AVR_RAMPZ_ADDR, __zero_reg__

/* // This part not needed for parts with < 64K

#if defined(__GNUC__) && ((__GNUC__ <= 3) || (__GNUC__ == 4 &&
__GNUC_MINOR__ <= 3))
#if BIG_CODE
    // Only for >64K devices with RAMPZ, replaces the default code
    // provided by libgcc.S which is only linked in if necessary.

    .section .init4,"ax",@progbits
    .global __do_copy_data
__do_copy_data:
    ldi    r17, hi8(__data_end)
    ldi    r26, lo8(__data_start)
    ldi    r27, hi8(__data_start)
    ldi    r30, lo8(__data_load_start)
    ldi    r31, hi8(__data_load_start)

    // On the enhanced core, "elpm" with post-increment updates RAMPZ
    // automatically. Otherwise we have to handle it ourselves.

#ifdef __AVR_ENHANCED__
        ldi    r16, hh8(__data_load_start)
#else
        ldi    r16, hh8(__data_load_start - 0x10000)
.L__do_copy_data_carry:
        inc    r16
#endif
        out   AVR_RAMPZ_ADDR, r16
        rjmp  .L__do_copy_data_start
.L__do_copy_data_loop:
#ifdef __AVR_ENHANCED__
        elpm  r0, Z+
#else
        elpm
#endif
        st    X+, r0
#endifdef __AVR_ENHANCED__

```

```

        adiw  r30, 1
        brcs  .L__do_copy_data_carry
#endif
.L__do_copy_data_start:
        cpi   r26, lo8(__data_end)
        cpc   r27, r17
        brne  .L__do_copy_data_loop
#ifdef __AVR_HAVE_RAMPD__
        out   AVR_RAMPZ_ADDR, __zero_reg__
#endif // __AVR_HAVE_RAMPD__

#endif // ** BIG_CODE **
#endif /* defined(__GNUC__) && ((__GNUC__ <= 3) || (__GNUC__ == 4 &&
__GNUC_MINOR__ <= 3)) *****/

*/

#endif /* !__AVR_ASM_ONLY__ */

        .section .init9,"ax",@progbits
#ifdef __AVR_ASM_ONLY__
        XJMP main
#else /* !__AVR_ASM_ONLY__ */
        XCALL    main
        XJMP exit
#endif /* __AVR_ASM_ONLY__ */

```