AVR assembly external SRAM usage

Asked 7 years ago Modified 6 years, 10 months ago Viewed 2k times

0

On Atmega 128 how to enable external SRAM, and how to copy a text into external SRAM? For example, I have some data SomeData: .db 0x01,0x02,0x03, ... , 0x.25 in .dseg, how can I copy this text into external SRAM begins at address 2100h?

Thanks in advance.

assembly external avr atmega atmel

Share Improve this question Follow

asked May 21, 2016 at 16:13

L. Kvri
1,422 4 23 41

are you sure you have an AtMega128? because i am quite sure that it has no external RAM interface (but i do not have a datasheet lying around -- took just a quick look into the atmega1284 (a successor) DS) – vlad_tepesch May 24, 2016 at 13:56

You can do that in software -- find a datasheet of the SRAM you want to interface, it should contain a description of the protocol that is used to read from/write to the chip. Then implement it on atmega. It wouldn't be easy nor efficient but is doable. – Aleksander Z. Jun 2, 2016 at 10:34

1 Answer

Sorted by: Highest score (default)

2

First you read the manual

On page 19 there is a figure that shows how the external memory maps into the memory space.

The description of the External Memory Interface starts on page 25.

the memory behaves like an internal memory would do.

Following assembly code shows how to read bytes from internal SRAM and copy them to external SRAM memory.

```
.cseg
copy:
      ldi r17, 4 ; counter for how many bytes to copy
      ldi YL, low(data) ; load pointer to data into Y pointer
      ldi YH, high(data)
      ldi ZL, low(0x2100) ; load pointer to external memory into Z
```

```
      ldi ZH, high(0x2100)
copy_loop:
      ld r16, Y+ ; r16 <- [Y], Y++
      st Z+, r16 ; [Z] <- r16, Z++

      dec r17
      brne copy_loop
      <other code>

data: .byte 4
```

Also have a look on the *st*, *ld*, *sts* and *lds* instructions that are used to access memory.

The pointers X,Y and Z of the AVR are 16 bit wide.

Be careful!

you wrote something like:

```
.dseg
data: .db 0x1, 0x2
```

This doesn't work. you can not write data using .db into the internal SRAM because it's volatile and has to be initialized using program code!

You can write

```
.cseg
<CODE>
data_const: .db 0x1,0x2,0x3,0x4

.dseg
data: .byte 4
```

this will create a four byte initialized data array in FLASH and reserve 4 bytes of space in the internal SRAM. you can now copy the data from data_const to data in your code.

By the way: Pre-initialized variables in C code are also located in flash and will be copied into the SRAM by the startup code before the execution of main() begins.

Share Improve this answer Follow

edited Jul 4, 2016 at 6:09

answered Jul 1, 2016 at 12:04

GNA
429 3 18