# **ATMEL Studio 7 (Variables.asm)**

;PROJECT :Variables84

;PURPOSE :INITIAL introduction to variables in SRAM

;AUTHOR :C. Darcy

;DATE :2020 04 16

;DEVICE :Dolgin Development Platform

;MCU :ATtiny84

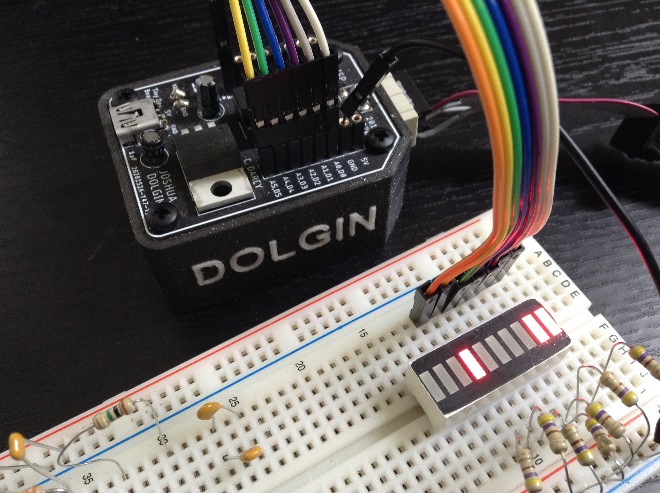
;COURSE :ICS4U

;STATUS :Working

.def util = r16 ;use aliases Registers

.def alt = r17 ;alternate register

.equ DDR = DDRA ;we'll need DDRA and

.equ PORT = PORTA ;PORTA

; DATA Segment declarations

.dseg

.org 0x0100 ;locate for SRAM

vars: .BYTE 4 ;reserve 4 bytes

; CODE Segment (default)

.cseg ;locate for FLASH

; \*\*\*\*\* INTERRUPT VECTOR TABLE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.org 0x0000 ;start of (IVT)

rjmp main ;highest priority!

.org 0x0011 ;locate just beyond 84's 17 (word) size IVT

varp:

.db 'A' ;AVR-ASM2 uses .DB instead of .BYTE

; \*\*\*\*\* START OF CODE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.org 0x0100 ;well clear of IVT

main: ;PC jumps to here (start of code) on reset interrupt...

ser util ;prepare to set ALL port bits for output

out DDR,util ;set ddr pins for output

;let's assign a value to an SRAM Variable...

ldi util,'A' ;note acceptable use of character constants

sts vars,util ;assign the register contents to the SRAM address

ldi util,'C' ;load a different constant

sts vars+1,util ;store it in the byte of the address in SRAM

;OK, let's read the contents of an SRAM location

lds alt,vars ;load a register with the value in the SRAM address

out PORT,alt ;dump it on the PORT to confirm it visually

rcall delay1s ;admire...

lds alt,vars+1 ;load a register with the value in a different SRAM address

out PORT,alt ;dump ACSII value on the PORT to confirm it visually

rcall delay1s ;admire...

rjmp main ;repeat...

ret

delay1s:

ldi r18, 41

ldi r19, 150

ldi r20, 128

L1: dec r20

brne L1

dec r19

brne L1

dec r18

brne L1

ret

# Arduino IDE (Variables.S)

// PROJECT  :Variables

// PURPOSE  :INITIAL introduction to variables in Data Memory(SRAM)

// COURSE   :ICS4U

// AUTHOR   :C. D'Arcy

// DATE     :2020 04 16

// MCU      :84

// STATUS   :Working

// REFERENCE:http://darcy.rsgc.on.ca/ACES/TEI4M/Assembly/AVR8AssemblyLanguage.html

#include  <avr/io.h>            //required to reference ports (-0x20 offset required)

#define util r16                //improve readability with aliases

#define alt  r17                //alternate register

.equ    OFFSET, 0x20            //avr-as requires offset address for PORTS

.equ    PORT,PORTA-OFFSET       //328P: 0x28 (C)

.equ    DDR, DDRA-OFFSET        //328P: 0x27 (C)

.equ    PIN, PINA-OFFSET        //328P: 0x26 (C)

.section .data    ;SRAM segment directive (.section is optional)

.org  0x060       ;0x060 is first available address beyond register space

vars:

.byte 4           ;reserves # of byte(s) of SRAM storage

.section .text    ;FLASH (PROGMEM) directive. (.section is optional)

.org  0           ;set the Program Counter (PC) to the start of program memory (IVT)

 rjmp  main      ;jump to the beginning of executable code

.org  0x11        ;set the Program Counter (PC) beyond the start of the 84's IVT

varp:             ;create a label (address) in which to place constants

.byte 'A'         ;example of a variable stored in Program memory

.org  0x100       ;places code at a specific location (this is sufficient)

.global main      ;eliminates Arduino C requirement of setup & loop functions

main:

 ser   util       ;prepare to set ALL port bits for output

 out   DDR,util   ;set ddr pins for output

;let's assign a value to an SRAM Variable...

 ldi   util,'A'      ;note acceptable use of character constants

 sts   vars,util     ;assign the register contents to the SRAM address

 ldi   util,'C'      ;load a different constant

 sts   vars+1,util   ;store it in the byte of the address in SRAM

;OK, let's read the contents of an SRAM location

 lds   alt,vars      ;load a register with the value in the SRAM address

 out   PORT,alt      ;dump it on the PORT to confirm it visually

 rcall delay1s       ;admire...

 lds   alt,vars+1    ;load a register with the value in a different SRAM address

 out   PORT,alt      ;dump ACSII value on the PORT to confirm it visually

 rcall delay1s       ;admire...

 rjmp  main          ;repeat...

 ret

delay1s:

   ldi  r18, 41

   ldi  r19, 150

   ldi  r20, 128

L1: dec  r20

   brne L1

   dec  r19

   brne L1

   dec  r18

   brne L1

   ret