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

;PROJECT :Shiftout7Segment

;PURPOSE :Demonstration of indirect addressing to display digits on the ADC Shield

;AUTHOR :C. D'Arcy

;DATE :2020 04 24

;DEVICE :Dolgin Development Platform + ADC Shield

;MCU :ATtiny84

;COURSE :ICS4U

;STATUS :Working

;REFERENCE :https://mail.rsgc.on.ca/~cdarcy/Datasheets/doc0856.pdf

.def util = r16 ;readability is enhanced through 'use' aliases for GP Registers

.def value = r17 ;holds the value to be displayed

.def mask = r18 ;byte with one set bit to act as a mask

.def dir = r19 ;shift direction: LSBFIRST:0, MSBFIRST:1

.def n = r20 ;make the a register for this app

.equ DDR = DDRA ;typically, we'll need the use of PortA

.equ PORT = PORTA ;both its data direction register and output register and, eventually,

.equ thou = PA1 ;these are the port pins...

.equ hund = PA2 ; connected to the base pins of

.equ tens = PA3 ; each of the transistors that

.equ units = PA4 ; ground the respective displays

.equ active = units ; pick one to make active

.equ DATA = PA5 ;595 data pin

.equ LATCH = PA6 ;595 latch pin

.equ CLOCK = PA7 ;595 clock pin

.equ FLAGS = 1<<active|1<<DATA|1<<LATCH|1<<CLOCK

;.equ LSBFIRST= 0 ;same familiar constants from Arduino days

.equ MSBFIRST= 1 ;ditto

; CODE Segment (default)

.cseg ;locate for Code Segment (FLASH)

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

.org 0x0000 ;start of Interrupt Vector Table (IVT) aka. Jump Table

 rjmp reset ;lowest interrupt address == highest priority!

;

.org 0x0020

;\*\*\*\*\*\*\*\* LookUp Table (LUT) of Segments \*\*\*\*\*\*\*\*\*\*

segStart: ;MSBFIRST: ABCDEFGx

.db 0b11111100,0b01100000,0b11011010,0b11110010

.db 0b01100110,0b10110110,0b10111110,0b11100000

.db 0b11111110,0b11110110

segEnd:

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

.org 0x0100 ;safe standard address for start of executable code

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

 ldi util,FLAGS ;load the OR'ed flag set for output pins

 out DDR,util ;set 'em

 sbi PORT,active ;turn on the active (units) display

 ldi XL,low(segStart<<1) ;position the 10-bit registers

 ldi XH,high(segStart<<1) ; to the start and the end

 ldi YL,low(segEnd<<1) ; of the array of segments (aka. LUT)

 ldi YH,high(segEnd<<1) ;...

;ready to shift digit, simply request shift order: either LSBFIRST or MSBFIRST

 ldi dir,MSBFIRST ;ABCDEFGx order and ADC Shield **requires MSFIRST**

repeat:

 movw Z,X ;position Z at the **start** of the LUT

next:

 lpm n,Z+ ;obtain the next segment map and **autoincrement**

 rcall shiftout ;invoke the shiftout function

 rcall delay1s ;admire...

 cp ZL,YL ;are we at the end of the array?

 brne next ;if not, keep going...

 rjmp repeat ;if so, start all over again...

shiftout: ;shifts constant n into the 595

 ldi mask,0x80 ;assume order is MSBFIRST

 sbrs dir,0 ;if bit 0 is set, it's MSBFIRST

 ldi mask,0x01 ;OK, it's LSBFIRST so redefine the mask

 cbi PORT,LATCH ;pull LATCH pin LOW

again:

 cbi PORT,CLOCK ;pull CLOCK pin LOW

 mov value,n ;reload the value to be presented

 and value,mask ;mask off the target bit

 breq lo ;was it 0?

 sbi PORT,DATA ;no, so pull DATA pin HIGH

 rjmp clockit ;ready to clock the 1

lo:

 cbi PORT,DATA ;else, it was a 0, so pull DATA pin LOW

clockit:

 sbi PORT,CLOCK ;pull CLOCK pin HIGH

; hmmm, must decide what direction to shift the mask...

 sbrs dir,0 ;if bit 0 is set, it's MSBFIRST

 rjmp shiftLeft ;OK, it's LSBFIRST

 lsr mask ;MSBFIRST so shift the mask right

 brne again ;repeat if there are still more bits to stuff in

 rjmp done ;we're done, so only one more thing to do

shiftLeft:

 lsl mask ;LSBFIRST, so shift the mask left

 brne again ;repeat if there are still more bits to stuff in

done:

 sbi PORT,LATCH ;pull LATCH pin HIGH to present 595's internal latches on output pins

 ret ;finished, return.

delay1s:

; Assembly code auto-generated

; by utility from Bret Mulvey

; Delay 8 000 000 cycles

; 1s at 8.0 MHz

 ldi r21, 41

 ldi r22, 150

 ldi r23, 128

L1:

 dec r23

 brne L1

 dec r22

 brne L1

 dec r21

 brne L1

 ret

# Arduino IDE (Shiftout7SegmentADC.S)

;PROJECT    :Shiftout7SegmentADC

;PURPOSE    :Demonstration of indirect addressing to display digits on the ADC Shield

;AUTHOR     :C. D'Arcy

;DATE       :2020 04 24

;DEVICE     :Dolgin Development Platform

;MCU        :ATtiny84

;COURSE     :ICS4U

;STATUS     :Working

;REFERENCE  :https://mail.rsgc.on.ca/~cdarcy/Datasheets/doc0856.pdf

#include <avr/io.h>

#define     util   r16           //readability is enhanced through 'use' aliases for GP Registers

#define     value  r17           //holds the value to be displayed

#define     mask   r18           //byte with one set bit to act as a mask

#define     dir    r19           //shift direction: LSBFIRST:0, MSBFIRST:1

#define     n      r20           //make the a register for this app

.equ        OFFSET, 0x20         ;avr-as requires offset address for PORTs

.equ        PORT, PORTA-OFFSET   ;0x3B – 0x20 = 0x1B

.equ        DDR, DDRA-OFFSET     ;0x3A – 0x20 = 0x1A

.equ        DIG1, PA1            ;these are the port pins...

.equ        DIG2, PA2            ;  connected to the base pins of

.equ        DIG3, PA3            ;  each of the transistors that

.equ        DIG4, PA4            ;  ground the respective displays

.equ        active, 1<<DIG4      ;pick one to make active

.equ        DATA, PA5            ;595 data pin

.equ        LATCH, PA6           ;595 latch pin

.equ        CLOCK, PA7           ;595 clock pin

.equ        LSBFIRST, 0          ;same familiar constants from Arduino days

.equ        MSBFIRST, 1          ;ditto

; CODE Segment (default)

.text                            ;locate for Code Segment (FLASH)

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

.org        0x0000               ;start of Interrupt Vector Table (IVT) aka. Jump Table

rjmp    main                     ;lowest interrupt address == highest priority!

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

.org 0x0020

segStart:                        ;MSBFIRST: ABCDEFGx

.byte   0b11111100, 0b01100000, 0b11011010, 0b11110010

.byte   0b01100110, 0b10110110, 0b10111110, 0b11100000

.byte   0b11111110, 0b11110110

segEnd:

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

.global main

.org        0x0100               ;safe standard address for start of executable code

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

   ldi    util, active | 1 << DATA | 1 << LATCH | 1 << CLOCK     ; create an OR'ed flag set for output pins

   out    DDR,util               ;set 'em

   ldi    util, active           ;activate the digit display of choice

   out    PORT, util             ;set it

   ldi    XL, lo8(segStart)      ;position the 16 - bit registers

   ldi    XH, hi8(segStart)      ;  to the start and the end

   ldi    YL, lo8(segEnd)        ;  of the array of segments

   ldi    YH, hi8(segEnd)        ;  ...

; ready to shift digit, simply request shift order: either LSBFIRST or MSBFIRST

   ldi    dir, MSBFIRST          ;change this manually (M or L)

repeat:

   movw   Z, X                   ;position Z (the loop index) at the start

next:

   lpm    n, Z+                  ;obtain the next segment map and autoincrement

   rcall  shiftout               ;invoke the shiftout function

   rcall  delay1s                ;admire...

   cp     ZL, YL                 ;are we at the end of the array ?

   brne   next                   ; if not, keep going...

   rjmp   repeat                 ; if so, start all over again...

; \*\*\*\*\* FUNCTIONS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

shiftout:                        ;shifts constant n into the 595

   ldi   mask, 0x80              ;assume order is MSBFIRST

   sbrs  dir, 0                  ;if bit 0 is set, it's MSBFIRST

   ldi   mask,0x01               ;OK, it's LSBFIRST so redefine the mask

   cbi   PORT, LATCH             ;pull LATCH pin LOW

again:

   cbi   PORT, CLOCK             ;pull CLOCK pin LOW

   mov   value, n                ;reload the value to be presented

   and   value, mask             ;mask off the target bit

   breq  lo                      ;was it 0 ?

   sbi   PORT, DATA              ;no, so pull DATA pin HIGH

   rjmp  clockit                 ;ready to clock the 1

lo:

   cbi   PORT, DATA              ;else, it was a 0, so pull DATA pin LOW

clockit:

   sbi   PORT, CLOCK             ;pull CLOCK pin HIGH

; hmmm, must decide what direction to shift the mask...

   sbrs  dir, 0                  ;if bit 0 is set, it's MSBFIRST

   rjmp  shiftLeft               ;OK, it's LSBFIRST

   lsr   mask                    ;MSBFIRST so shift the mask right

   brne  again                   ;repeat if there are still more bits to stuff in

   rjmp  done                    ; we're done, so only one more thing to do

shiftLeft:

   lsl   mask                    ;LSBFIRST, so shift the mask left

   brne    again                 ;repeat if there are still more bits to stuff in

done:

   sbi   PORT,LATCH              ;pull LATCH pin HIGH to present 595's internal latches on output pins

   ret                           ;finished, return.

delay1s:

; Assembly code auto - generated

; by utility from Bret Mulvey

; Delay 8 000 000 cycles

; 1s at 8.0 MHz

ldi  r21, 41

ldi  r22, 150

ldi  r23, 128

L1: dec  r23

brne L1

dec  r22

brne L1

dec  r21

brne L1

ret