6. Printing Bytes as Hexadecimal Values

To easily debug and view numbers, we can create a subroutine that outputs numbers into serial. Since our architecture uses 8 bits, it is easier to print bytes as two hexadecimal values 0x00 - 0xFF by splitting the upper and lower nibbles. In this page, we will create a subroutine that prints R16 as hexadecimal to serial.

Given the following subroutines to use the serial:

```
; Initializes USART
; Registers Affected: R24
SER_init:
  CLR   R24
  STS   UCSR0A, R24                     ; clear UCSR0A register
  STS   UBRR0H, R24                     ; clear UBRR0H register
  LDI   R24, 103                        ; store in UBRR0L 103 value
  STS   UBRR0L, R24                     ; to set baud rate 9600
  LDI   R24, 1 << RXEN0 | 1 << TXEN0    ; enable RXB & TXB
  STS   UCSR0B, R24
  LDI   R24, 1 << UCSZ00 | 1 << UCSZ01  ; asynch, no parity, 1 stop, 8 bits
  STS   UCSR0C, R24
  RET
```

```
; Prints character in R16 to USART
; Blocks while UDRE0 is not ready.
; Registers Affected: R24
SER_send_byte:
  LDS   R24, UCSR0A
  SBRS  R24, UDRE0                      ; test data buffer if data can be sent
  RJMP  SER_send_byte                   ; loop back if not ready
  STS   UDR0, R16                       ; sends R16 to USART
  RET
```

Printing Nibbles

A 4 bit nibble, the lower parts of a byte, can be mapped into hexadecimal values 0 - F. Adding '0' to the nibble would map values 0 - 9 to its respective '0' - '9' ASCII characters.

```
SER_nibble:
  ANDI  R16, 0x0F ; mask that removes the higher nibble
  SUBI  R16, -'0' ; add '0' to R16 to represent ASCII '0' - '9'.
  ...
```

Just simply doing this, however, wouldn't work on values A - F since they are not continuously mapped right after '9' which would instead prints ':' for 10. To do so, we can check whether the resulting ASCII is greater than '9'. If so, we can then add it by 7 since 'A' is located 7 characters after '9'.

```
SER_nibble:
  PUSH  R16           ; preserves R16
  ANDI  R16, 0x0F     ; mask that removes the higher nibble
  SUBI  R16, -'0'     ; add '0' to R16 to represent ASCII '0' - '9'.
  CPI   R16, '9' + 1  ; compare with '9' + 1 = ':'
  BRLT  print_nibble  ; if no problem just simply print the character
  SUBI  R16, -7       ; otherwise add with 7 first to adjust for 'A'
print_nibble:
  RCALL SER_send_byte
  POP   R16           ; retrieves R16
  RET
```

With this subroutine, we can print the lower nibble of R16

```
main:
  RCALL SER_init

  LDI   R16, 0x0C
  RCALL SER_nibble      ; prints C

  LDI   R16, 0x14       ; prints 4
  RCALL SER_nibble
```

Print Nibbles

Printing Bytes

We can expand this further by printing entire bytes by first printing the upper nibble followed by the lower nibble. In AVR there is an instruction `SWAP Rd` that swaps the upper 4 bits with the lower 4 bits of Rd.

```
LDI R16, 0x14
SWAP R16
RCALL SER_nibble       ; prints 1 instead
```

With this, we can complete the hexadecimal printing subroutine.

```
; Prints R16 as HEX to USART
; R16 is preserved.
SER_hex:
  SWAP R16            ; swap to get upper nibble first
  RCALL SER_nibble
  SWAP R16            ; revert the nibbles back for the lower one
  RCALL SER_nibble
  RET

main:
  RCALL SER_init
  LDI   R16, 0x3C
  RCALL SER_hex      ; prints 3C
```

Print Bytes

Revision #1
Created 2026-02-25 09:14:17 UTC by MF
Updated 2026-02-25 09:17:02 UTC by MF