Programování mikrokontrolerů

AVR Self Programming

Atmel Application Note 109

- AVR109: Self programming
  - for AVR devices with SPM instruction
  - communicates via UART
    - possible counterpart as of AVR911 Appnote
  - Flash & EEPROM programming
  - read/write lock bits
  - read signature & fuse bits

Main idea

- AVR boots into bootloader section
- bootloader checks if programming is desired
  - no – jumps to $0000 as if ordinary reset occurred
  - yes – enters simple state machine:
    - read single-letter command from UART
    - react accordingly

Commands

	Host Writes		Host Reads	
	ID	Data	Data	
Enter Programming Mode	“P”			13d
Auto Increment Address	“a”		dd	
Set Address	“A”	ah al		13d
Write Program Memory, Low Byte	“c”	dd		13d
Write Program Memory, High Byte	“C”	dd		13d
Issue Page Write	“m”			13d
Read Lock Bits	“r”		dd	
Read Program Memory	“R”		2*dd	
Read Data Memory	“d”		dd	
Write Data Memory	“D”	dd		13d
Chip Erase	“e”			13d
Write Lock Bits	“l”	dd		13d
Read Fuse Bits	“F”		dd	
Read High Fuse Bits	“N”		dd	
Read Extended Fuse Bits	“Q”		dd	
Leave Programming Mode	“L”			13d
Select Device Type	“T”	dd		13d
Read Signature Bytes	“s”		3*dd	

Commands

	Host Writes		Host Reads	
	ID	Data	Data	
Return Supported Device Codes	“t”		n*dd	00d
Return Software Identifier	“S”		s[7]	
Return Software Version	“V”		dd dd	
Return Programmer Type	“p”		dd	
Set LED	“x”	dd		13d
Clear LED	“y”	dd		13d
Exit Bootloader	“E”			13d
Check Block Support	“b”		“Y”	
Start Block Flash Load	“B”	2*dd “F” 2*dd n*dd	13d	
Start Block EEPROM Load	“B”	2*dd “E” n*dd		13d
Start Block Flash Read	“g”	2*dd “F”	n*dd	
Start Block EEPROM Read	“g”	2*dd “E”	n*dd	


```
C_TASK void main(void)
{
    ADDR_T address;
    unsigned int temp_int;
    unsigned char val;
    /* Initialization */
    void (*funcptr)( void ) = 0x0000; // Set up function pointer to RESET vector.
    PROGPORT |= (1<<PROG_NO); // Enable pull-up on PROG_NO line on PROGPORT.
    initbootuart(); // Initialize UART.
    /* Branch to bootloader or application code? */
    if( !(PROGPIN & (1<<PROG_NO)) ) // If PROGPIN is pulled low, enter programmingmode.
    {
        /* Main loop */
        for(;;)
        {
        ...
            // Exit bootloader.
            else if(val=='E')
            {
                _WAIT_FOR_SPM();
                _ENABLE_RWW_SECTION();
                sendchar('\r');
                funcptr(); // Jump to Reset vector 0x0000 in Application Section.
            }
         ...
         } // end: for(;;)
    }
    else
    {
        _WAIT_FOR_SPM();
        _ENABLE_RWW_SECTION();
```

```
/* Main loop */
        for(;;)
        {
            val=recchar(); // Wait for command character.
            // Check autoincrement status.
            if(val=='a')
            {
                sendchar('Y'); // Yes, we do autoincrement.
            }
            // Set address.
            else if(val=='A') // Set address...
            { // NOTE: Flash addresses are given in words, not bytes.
                address=(recchar()<<8) | recchar(); // Read address high and low byte.
                sendchar('\r'); // Send OK back.
            }
            // Chip erase.
            else if(val=='e')
            {
                for(address = 0; address < APP_END;address += PAGESIZE)
                { // NOTE: Here we use address as a byte-address, not word-address,
                  // for convenience.
                    _WAIT_FOR_SPM();
...
```


```
…
            // Return programmer identifier.
            else if(val=='S')
            {
                sendchar('A'); // Return 'AVRBOOT'.
                sendchar('V'); // Software identifier (aka programmer signature) is
                               // always 7 characters.
                sendchar('R');
                sendchar('B');
                sendchar('O');
                sendchar('O');
                sendchar('T');
            }


            // Return software version.
            else if(val=='V')
            {
                sendchar('1');
                sendchar('5');
            }


            // Return signature bytes.
            else if(val=='s')
            {
                sendchar( SIGNATURE_BYTE_3 );
                sendchar( SIGNATURE_BYTE_2 );
                sendchar( SIGNATURE_BYTE_1 );
            }
```