ACCESSING IAP ROUTINES ON P89V66X

All 8051 based microcontrollers from NXP come with some sort of built-in flash memory. The microcontrollers provide multiple methods to program the flash. The most common method is In-System Programming (ISP). In ISP, the code is transferred to the flash, through the serial port, by a built-in ISP bootloader. The other method is In-Application Programming (IAP). In IAP, the user application receives the code through whatever mechanism (like say SPI or I²C) and programs the flash. A set of built-in routines, called the IAP routines, are available to the application for manipulating the flash.

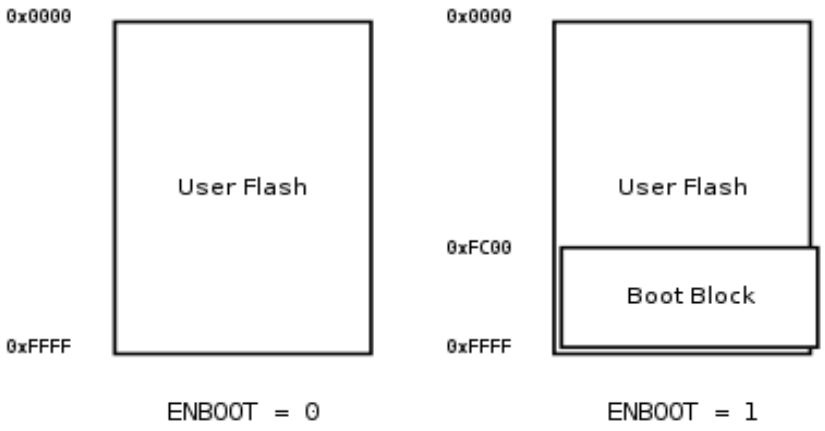
This article shows how to access the IAP routines on P89V66x based devices. The routines will be accessed from C, and the free and open source SDCC compiler will be used.

The Boot Block

On the P89V66x devices the bootloader IAP routines are located within the boot block. The boot block is usually not accessible. But by setting ENBOOT bit in the AUXR1 register, the boot block gets mapped to the address range 0xFC00 to 0xFFFF.

The ENBOOT bit is located at bit position 5 in the AUXR1 register. For some unknown reason, this is an undocumented flag!

So to access the IAP routines, the boot block must be mapped first.



BootBlock and ENBOOT

IAP routines

The IAP routines and the calling conventions are provided in the product data sheet. All the IAP routines are invoked through a common entry point at 0xFFF0. The operation to perform is specified using the R1 register. For example to erase erase a block, R1 should be set to 0x1 and to program a byte, R1 should be set to 0x2. The arguments to the operation are passed in DPH, DPL and in some cases ACC. After the operation completes status and return values are passed on ACC.

The assembly code skeleton for invoking the IAP routine, will look something like the following.

```
    mov R1,  <op>    ; Specify the operation
    mov DPL, <arg1>  ; Specify the first argument
    mov DPH, <arg2>  ; Specify the second argument
    PGM_MTP = 0xFFF0
    lcall PGM_MTP    ; Invoke the IAP entry point
```

IAP and Interrupts

It should be noted that interrupts should not occur while executing IAP routines. Hence, it is best to disable interrupts before invoking them, and restore the interrupt state after invocation.

Invoking IAP routines from C

The goal is to write a C function that takes in the values for R1, DPH and DPL as arguments, invokes the IAP routine, and returns the value in ACC. The prototype of the funtion is shown below.

```
unsigned char iap_call(unsigned char r1, unsigned char dph, unsigned char
dpl);
```

To acheive this, we will need to use inline assembly to access the IAP routines. We also need to know SDCC's calling convention, so that we know in which registers the arguments are passed to the C function. This information can be obtained by compiling a C program and looking and the generated assembly code!

According to the generated assembly for the small memory model, the first function parameter is stored inDPL, and the remaining parameters are stored in _iap_call_PARAM_n, where n is th parameter no.

Combining all the information we have so far, the following code listing shows the implementation of iap_call.

```
unsigned char iap_call(unsigned char r1, unsigned char dph, unsigned char
dpl)
{
    bit save_ea;

    /* Too keep the compiler quiet. */
    r1, dph, dpl;

    /* Save and disable interrupts. */
```

```
    save_ea = EA;
    EA = 0;

    /* Enable boot block. */
    AUXR1 |= 0x20;

    /* Invoke the IAP routine. */
    _asm ;
    mov R1,  DPL                ; /* First argument is in DPL */
    mov DPH, _iap_call_PARM_2   ; /* Second argument from memory */
    mov DPL, _iap_call_PARM_3   ; /* Third argument from memory */

    PGM_MTP = 0xFFF0            ;

    lcall PGM_MTP               ; /* IAP entry point is 0xFFF0 */
    mov DPL, A                  ; /* C function return value should be in DPL */
    _endasm;

    /* Disable the boot block. */
    AUXR1 &= ~0x20;

    /* Restore interrupt enable flag. */
    EA = save_ea;
}
```

Source: http://www.zilogic.com/blog/iap-p89v66x.html