Lab 03 Tips and General Information

Course : ET032G, Elektroteknik GR (A), Mikrodatorteknik at Mittuniversitetet, Sweden.

Extracted by : Muhammad Imran, Mid Sweden University, Sweden.

Starting with a basic, about function prototype, function call and function declaration/definition since I have used these terms in the lab instructions

Example of function calling and parameter passing

```
#include<stdio.h>
int add(int , int);                    // function prototype/declaration
int main()
    {
    Int num1, num2;
    value_return=add(num1, num2)  // function call and getting return value
    printf("addition=%d", value_return);
    }

Int add(int var1, int var2)            //function definition (Int add(int var1, int var2) is declarator and body is definition
{
 Int x,y, sum;
 sum=a+b;
 return sum
}
```

I/O memory and virtual port discussion

I/O memory and general purpose registers

The data memory space in Xmega AVR is divided into I/O registers, SRAM, and external RAM. In addition, the EEPROM can be memory mapped in the data memory. All I/O status and control registers reside in the lowest 4KB addresses of the data memory. This is referred to as the **I/O memory** space. All I/O locations can be accessed by the load (LD/LDS/LDD) and store (ST/STS/STD) instructions.

The lowest **64 addresses of** I/O memory can be accessed directly, or as the data space locations from 0x00 to 0x3F. The **lowest 4** I/O memory addresses are reserved as general purpose I/O registers. These registers can be used for storing global variables and flags, as they are directly bit-accessible using the SBI, CBI, SBIS, and SBIC instructions. The rest is the extended I/O memory space, ranging from 0x0040 to 0x0FFF [1].

Xmega has 8/16 bit AVR RISC CPU and has 32x8-bit registers directly connected to ALU. The 32 x 8-bit general purpose working registers all have single clock cycle access time allowing single-cycle arithmetic logic unit operation between registers or between a register and an immediate.

The six registers R26-R31 have added function in addition to their general purpose usage. These six 32 registers, shown in *Figure 1*, can be used as three **16-bit address pointers for program and data space addressing**, enabling efficient address calculations. One of these register (Z-register) can also be use as an address pointer to read from and/or write to flash program memory, signature rows and lock bits.

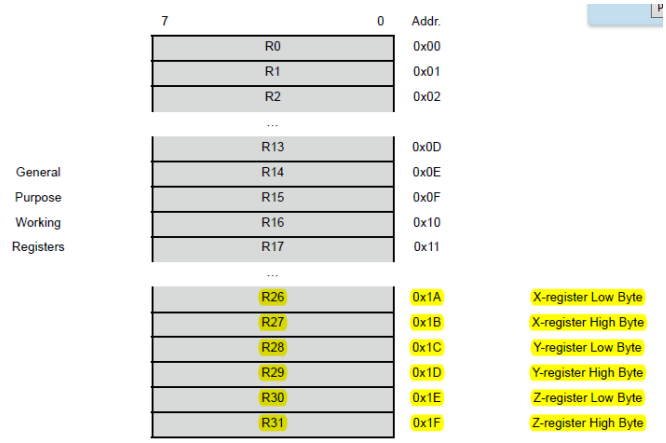

Figure 1. CPU general purpose registers

Virtual port

Some instructions in the AVR® instruction set can only operate on addresses that are within the AVR I/O space. Using these instructions instead of their data space equivalents is both faster and consumes less program memory. All I/O port registers on the XMEGA have addresses outside the I/O space. The solution to this is to use the virtual ports. Up to four of the I/O ports can be mapped into virtual ports that have registers in the I/O space. The virtual ports make the DIR, OUT, IN and INTFLAGS registers of the desired I/O port available in I/O space. The other, less used I/O port registers are still available through the regular port module registers [2].

Virtual port registers allow the port registers to be mapped virtually in the bit-accessible I/O memory space. When this is done, writing to the virtual port register will be the same as writing to the real port register. This enables the use of I/O memory-specific instructions, such as bit-manipulation instructions, on a port register that normally resides in the extended I/O memory space. There are four virtual ports, and so four ports can be mapped at the same time. Virtual 0 and 1 mapping control register is shows here.

12.13.2 VPCTRLA – Virtual Port-map Control register A

Bit	7	6	5	4	3	2	1	0
+0x02	VP1MAP[3:0]				VP0MAP[3:0]			
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:4 – VP1MAP: Virtual Port 1 Mapping**

These bits decide which ports should be mapped to Virtual Port 1. The registers DIR, OUT, IN, and INTFLAGS will be mapped. Accessing the virtual port registers is equal to accessing the actual port registers. See Table 12-7 on page 141 for configuration.

PORTA to PORTR can be mapped in any of the four virtual ports. For example, I want to map PORTB to virtual port 0.

VP0MAP	**PORT**	**Description**
0001	PORTB	PORTB mapped to Virtual Port n

In c code you can write `PORTCFG.VPCTRLA = 0x01;`

References

[1]. Page (7-11) in the XMEGA-B manual
[2]. AVR1313: Using the XMEGA IO Pins and External Interrupts
[3]. Page (132, 140) in the XMEGA-B manual
[4].