Programming microcontrollers

Microcontrollers are similar to processors. The difference is, microcontrollers contain integrated memory and specific hardware periphery modules, which can perform Digital and Analog functions like Binary IO, PWM, Counters, Timers, Digital to Analog Converters and Analog to Digital Converters. Sometimes peripheral modules for communication are also integrated, e.g. CAN, FlexRay, Ethernet, SENT, I2C, and SPI. These modules can be configured by registers.

Microcontrollers are generally programmed in Assembler or in C language. In this practical, the higher level language C is used to program the ECUs. This kind of programming is similar to programming software for PCs. But in embedded systems, the programs access the registers directly to configure the peripheral modules by setting certain bits to activate a desired peripheral function on a particular microcontroller pin. Once the pins are configured for a particular peripheral function, then we use another register either to control or to receive a value from real world. For this purpose the memory addresses of the registers are used, which are defined in the reference manual of the microcontroller. Hence we use register addresses rigorously to access information from peripheral modules.

To ease this process, header files with macros are often delivered by the manufacturer of the microcontroller. In this practical macros are defined in the “jdp.h” header file, which can be found in the project folder.

In general, the pins of the microcontroller are organized by “PORT”. Every port of the microcontroller has a set of pins. Each port could only perform certain tasks due to limited support for remapping peripheral functions. More details of it can be found in the Reference manual.

Figure 1 shows an extract of the reference manual of the microcontroller, which can be found at [1]. It shows the structure of the PCR registers of the SIUL module, which are used to configure digital in- and output.

Base + 0x0040 (PCR0)

Address: ... Access: User read/write

Base + 0x0116 (PCR107) 108 registers

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	SMC	APC	0	PA[1:0]		OBE	IBE	0	0	ODE	0	0	SRC	WPE	WPS
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1: SIUL Pad Configuration Registers (PCR) [1]

To access the PCR registers in C code, it is possible to directly access the base address of the registers or to write “SIU.PCR[X].R“, which is a macro defined in the “jdp.h“ header file. The meaning of bit fields is also described in the reference manual.

Example

Objective: Make pin 94 of a SPC560L5 (100 pin package) to a Digital Output and then set it low.

We look in signals descriptions and find the possibility of GPIO (General Purpose IO) for pin 94 and corresponding PCR number which is in 9, as shown in Figure 2.

Table 7. Pin muxing (continued)

Port pin	Pad configuration register (PCR)	Alternate function(1), (2)	Functions	Peripheral(3)	I/O direction (4)	Pad speed(5)		Pin No.	
						SRC = 0	SRC = 1	100-pin	144-pin
A[7]	PCR[7]	ALT0 ALT1 ALT2 ALT3 —	GPIO[7] SOUT — — EIRQ[7]	SIUL DSPI_1 — — SIUL	I/O O — — I	Slow	Medium	4	10
A[8]	PCR[8]	ALT0 ALT1 ALT2 ALT3 — —	GPIO[8] — — — SIN EIRQ[8]	SIUL — — — DSPI_1 SIUL	I/O — — — I I	Slow	Medium	6	12
A[9]	PCR[9]	ALT0 ALT1 ALT2 ALT3 —	GPIO[9] CS1 — B[3] FAULT[0]	SIUL DSPI_2 — FlexPWM_0 FlexPWM_0	I/O O — O I	Slow	Medium	94	134

Figure 2: Pin List and Multiplexing

To enable the output buffer of an I/O pin the “output buffer enable” bit (OBE) must be set. In figure 1 the OBE bit is at position 6. To set this bit, 0x0200 could be written to the register. In C it looks like this for port A pin 9 of the STM SPC560L5 microcontroller:

```
SIU.PCR[9].R = 0x0200;
```

To activate the output as “low signal”, we write to corresponding pad register of function which is GPDO.

```
SIU.GPDO[9].R = 0x0200;
```

Hexadecimal numbers are not mandatory, but make it a cleaner code.

Every module on the microcontroller can be configured in a similar way. In the reference manual all functionality and registers are described. These are the basics of microcontroller and driver programming.

Note: Every microcontroller has its own individual modules and registers. That’s why it’s necessary to read the reference manual and to program new drivers, if the hardware is changed.

[1] http://www.st.com/st-web-ui/static/active/en/resource/technical/document/reference_manual/CD00204027.pdf