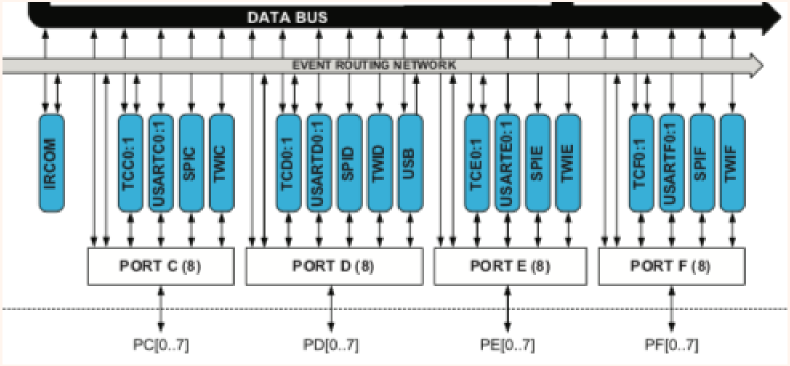
Chip Programming Model

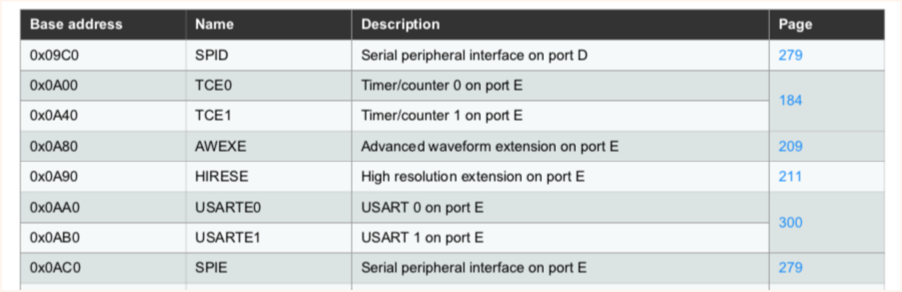
- Modules, Registers and Bits
  - Xmega comprised of modules: CPU, SPI, UART, I2C, etc.
  - More than one instance of a module may exist.
  - Name reflects the function of the module.
  - Each module instance has a suffix to identify it:
    - USARTC0: USART 0 on port C
    - SPIE1: SPI 1 on port E



Chip Programming Model - Modules

- Each module has a fixed address in I/O memory.
- For SPI on port D:

```
#define SPID (*(SPI_t *) 0x09C0)  /*(iox64a1u.h line 2683)*/
```

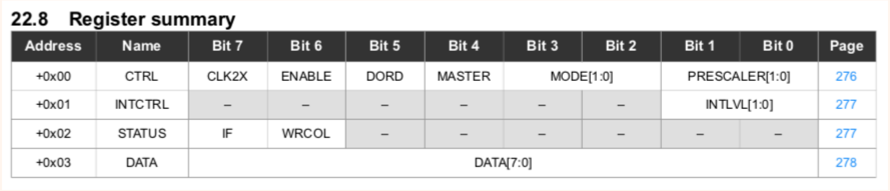


Base address	Name	Description	Page
0x09C0	SPID	Serial peripheral interface on port D	279
0x0A00	TCE0	Timer/counter 0 on port E	184
0x0A40	TCE1	Timer/counter 1 on port E	
0x0A80	AWEXE	Advanced waveform extension on port E	209
0x0A90	HIRESE	High resolution extension on port E	211
0x0AA0	USARTE0	USART 0 on port E	300
0x0AB0	USARTE1	USART 1 on port E	
0x0AC0	SPIE	Serial peripheral interface on port E	279

Chip Programming Model - Registers

- Module registers are located a fixed offset from the module base address.
- Register offsets are equal for all instances of that module type.
- Each module has status and control registers.

```
#/*(iox64a1u.h line 3584)*/
/* SPID - Serial Peripheral Interface D */
#define SPID_CTRL   _SFR\MEM8(0x09C0)
#define SPID_INTCTRL   _SFR_MEM8(0x09C1)
#define SPID_STATUS   _SFR_MEM8(0x09C2)
#define SPID_DATA   _SFR_MEM8(0x09C3)
```



22.8 Register summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRL	CLK2X	ENABLE	DORD	MASTER	MODE[1:0]		PRESCALER[1:0]		276
+0x01	INTCTRL	–	–	–	–	–	–	INTLVL[1:0]		277
+0x02	STATUS	IF	WRCOL	–	–	–	–	–	–	277
+0x03	DATA	DATA[7:0]								278

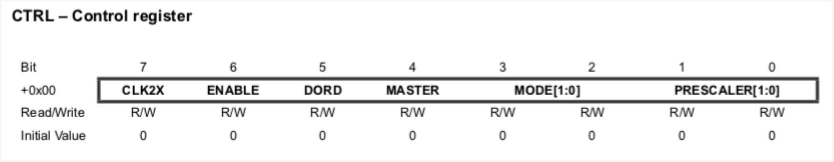
Chip Programming Model - Registers

- Modules of the same type have identical sets of status and control registers.
- Otherwise, there would be multiple structs for multiple SPI modules.

```
#/*(iox64a1u.h line 2539)*/
/* Serial Peripheral Interface */
typedef struct SPI_struct
{
    register8_t CTRL;    /*Control Register*/
    register8_t INTCTRL; /*Interrupt Control Register*/
    register8_t STATUS;  /*Status Register*/
    register8_t DATA;    /*Data Register*/
} SPI\_t;
```

Chip Programming Model - Bits

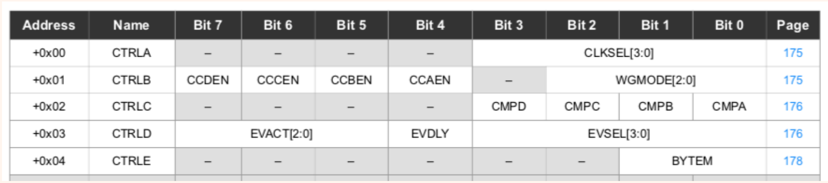
- Modules of the same type (SPI, USART, etc.), within the same part family (A,B,C) have identical sets of status and control bits.



CTRL – Control register

Bit	7	6	5	4	3	2	1	0
+0x00	CLK2X	ENABLE	DORD	MASTER	MODE[1:0]		PRESCALER[1:0]	
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

Figure: SPI Port B,C,D or F Control Regs



Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRLA	–	–	–	–	CLKSEL[3:0]				175
+0x01	CTRLB	CCDEN	CCCEN	CCBEN	CCAEN	–	WGMODE[2:0]			175
+0x02	CTRLC	–	–	–	–	CMPD	CMPC	CMPB	CMPA	176
+0x03	CTRLD	EVACT[2:0]			EVDLY	EVSEL[3:0]				176
+0x04	CTRLE	–	–	–	–	–	–	BYTEM		178

Figure: Timer Counter 0 or 1 Control Regs

Chip Programming Model - Bit Masks

- Bits in a register can be have an individual function or be part of a bit group that have a joint function.
- A bit with an individual function could be the single enable bit for a module. For example, bit six of the SPID control register enables that SPI device.
- We would access this bit like this:

```
     SPIC.CTRL = SPI_ENABLE_bm
```

- The _bm suffix indicates a bit mask.
- In `iox64a1u.h`, line 7014, we see:

```
     #define SPI_ENABLE_bm  0x40  /* Enable Module bit mask. */.
```

  Thus, the SPID enable bit is set to one.

Chip Programming Model - Bit Groups

- A bit could also be part of a bit group.
- The bit would be part of a group of bits that choose upto 2^n selections where n is the number of bits in the group.
- For example, the group of bits that choose which of four modes a SPI module is in:

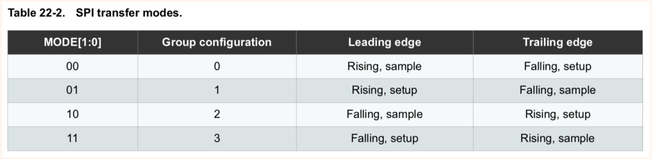


Table 22-2. SPI transfer modes.

MODE[1:0]	Group configuration	Leading edge	Trailing edge
00	0	Rising, sample	Falling, setup
01	1	Rising, setup	Falling, sample
10	2	Falling, sample	Rising, setup
11	3	Falling, setup	Rising, sample

Figure: Four SPI Transfer Modes

Chip Programming Model - Bit Groups

- We would access the SPI mode group configureation as:

```
    SPIC.CTRL = SPI_MODE_0_gc;
```

- The _gc suffix indicates group configureation.
- SPI_MODE_0_gc is defined as 0x00 in `iox64a1u.h, line 2548.`

```
/* SPI Mode */
typedef enum SPI_MODE_enum
{
    SPI_MODE_0_gc = (0x00<<2),  /* SPI Mode 0 */
    SPI_MODE_1_gc = (0x01<<2),  /* SPI Mode 1 */
    SPI_MODE_2_gc = (0x02<<2),  /* SPI Mode 2 */
    SPI_MODE_3_gc = (0x03<<2),  /* SPI Mode 3 */
} SPI_MODE_t;
```