In Program 6.1 the assumption was the software module had access to all of Port F. In other words, this software owned all pins of Port F. In most cases, a software module needs access to only some of the port pins. If two or more software modules access the same port, a conflict will occur if one module changes modes or output values set by another module. It is good software design to write **friendly** software, which only affects the individual pins as needed. Friendly software does not change the other bits in a shared register. Conversely, **unfriendly** software modifies more bits of a register than it needs to. The difficulty of unfriendly code is each module will run properly when tested by itself, but weird bugs result when two or more modules are combined.

Consider the problem that a software module needs to output to just Port A bit 7. After enabling the clock for Port A, we use read-modify-write software to initialize just pin 7. The following initialization does not modify the configurations for the other 7 bits in Port A. Unlocking is not required for PA7 (just PD7 and PF0 require unlocking)

```
SYSCTL_RCGC2_R |= 0x00000001;     // 1) activate clock for Port A
delay = SYSCTL_RCGC2_R;           // allow time for clock to start
GPIO_PORTA_AMSEL_R &= ~0x80;      // 3) disable analog on PA7
GPIO_PORTA_PCTL_R &= ~0xF0000000; // 4) PCTL GPIO on PA7
GPIO_PORTA_DIR_R |= 0x80;         // 5) PA7 out
GPIO_PORTA_AFSEL_R &= ~0x80;      // 6) disable alt funct on PA7
GPIO_PORTA_DEN_R |= 0x80;         // 7) enable digital I/O on PA7
```

There is no conflict if two or more modules enable the clock for Port A. There are two ways on TM4C microcontrollers to access individual port bits. The first method is to use read-modify-write software to change just one pin. A read-or-write sequence can be used to set bits.

```
LDR R1, =GPIO_PORTA_DATA_R
LDR R0, [R1]      ; previous
ORR R0, R0, #0x80 ; set bit 1
STR R0, [R1]
```

```
// make PA7 high
GPIO_PORTA_DATA_R |= 0x80;
```

A read-and-write sequence can be used to clear one or more bits.

`LDR R1, =GPIO_PORTA_DATA_R`
 	
`LDR R0, [R1]  ; previous`
 	
`BIC R0, R0, #0x80 ; clear bit 1`
 	
`STR R0, [R1]`
	`// make PA7 low`
	
	`GPIO_PORTA_DATA_R &= ~0x80;`