Register Access

Common - Last updated 5/16/19

Register Access

- Block Register Access
  - Each block (peripheral) has a series of registers
    - Configuration
    - Control
    - Inputs
    - Outputs

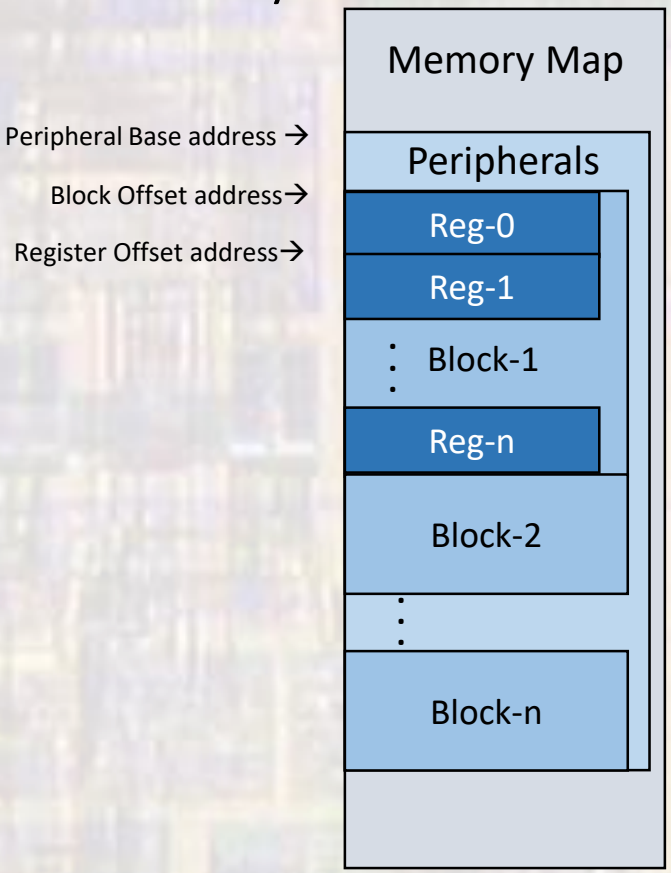
Typical Block Register Set

Table 17-3. Timer_A Registers

Offset	Acronym	Register Name	Section
00h	TAxCTL	Timer_Ax Control	Section 17.3.1
02h to 0Eh	TAxCCTL0 to TAxCCTL6	Timer_Ax Capture/Compare Control 0 to Timer_Ax Capture/Compare Control 6	Section 17.3.3
10h	TAxR	Timer_Ax Counter	Section 17.3.2
12h to 1Eh	TAxCCR0 to TAxCCR6	Timer_Ax Capture/Compare 0 to Timer_Ax Capture/Compare 6	Section 17.3.4
2Eh	TAxIV	Timer_Ax Interrupt Vector	Section 17.3.5
20h	TAxEX0	Timer_Ax Expansion 0	Section 17.3.6

Register Access

- Block Register Access
  - Registers are memory mapped
    - Accessed via memory addresses
    - Actually exist in the blocks



MSP432 Example – Timer_A1

← 0x4000 0000 - Peripherals

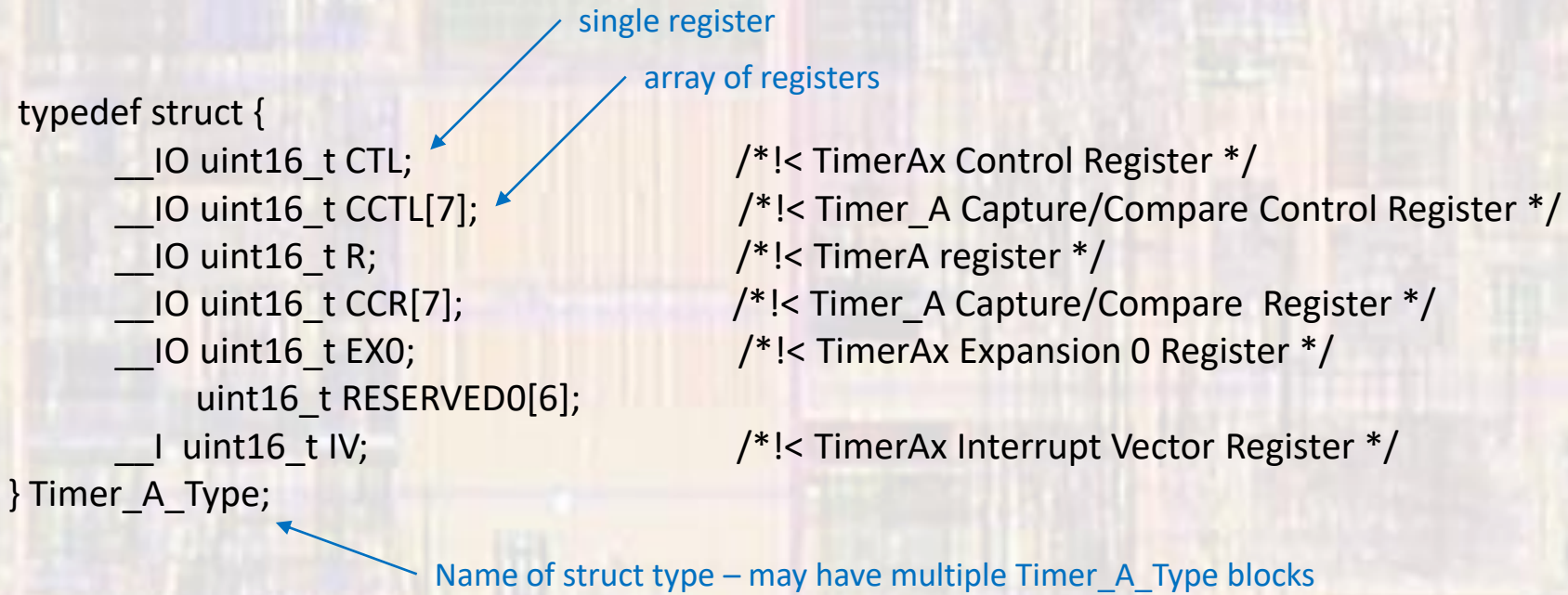
← offset: 0x400 , absolute 0x4000 0400 – Timer_A1

← offset: 0x02 , absolute 0x4000 0402 – Timer_A1, TA1_CCTL0 register

← offset: 0x20 , absolute 0x4000 0420 – Timer_A1, TA1_EX0 register

Register Access

- Peripheral Register Access
  - Use C-Structures to define the registers



Register Access

- Peripheral Register Access
  - MSP432.h (or some lower level file)
    - Define the addresses

```
#define __IO  volatile
#define __I  constant

#define PERIPH_BASE  ((uint32_t)0x40000000)     // Peripherals start address
#define TIMER_A1_BASE  (PERIPH_BASE + 0x00000400)     // Base address of module

#define TIMER_A1  ((Timer_A_Type *) TIMER_A1_BASE)  //  pointer to  Timer_A1 base addr
```

create aliases

Assign base addresses using offsets

Create a pointer to the block base address

Register Access

- Peripheral Register Access
  - Use pointer and structure element names to access individual registers

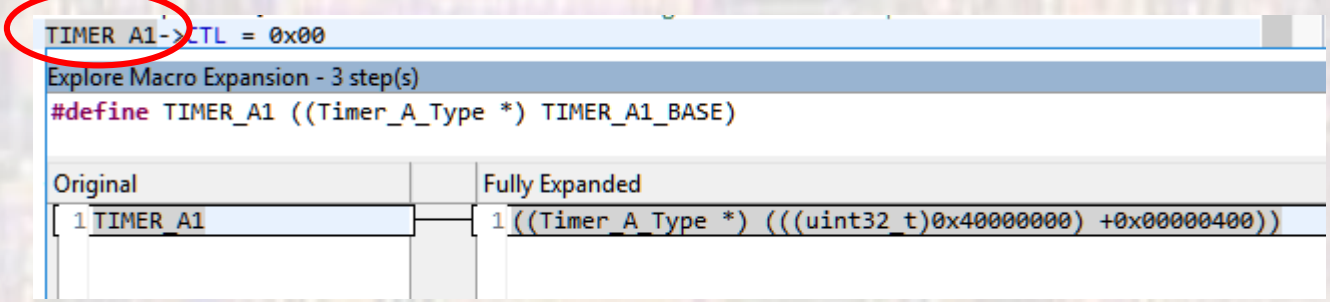
Pointer

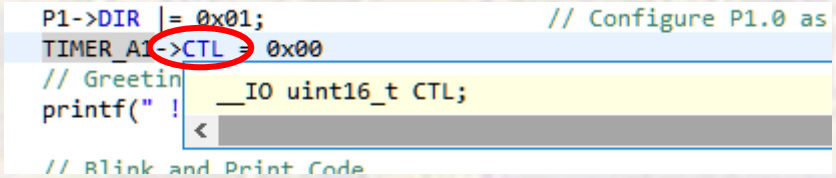
Structure Member Name

Value to store

Note the -> notation: we are accessing the structure via a pointer

```
TIMER_A1 -> CTL = 0x0000;        // Clear Timer_A1 control register
```





Register Access

- Register Configuration

Set bit 0 in register P6->DIR to 1

Need to set this bit to a 1

```
P6->DIR = 0x01;          // set P6->Dir bit 0 to 1
or
P6->DIR = b00000001;     // set P6->Dir bit 0 to 1
or
P6->DIR = 1;             // set P6->Dir bit 0 to 1
```

NO !!!

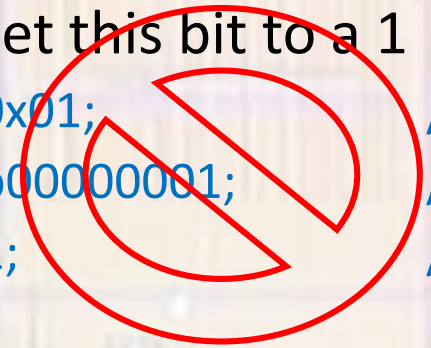
Register Access

- Register Configuration

Set bit 0 in register P6->DIR to 1

Need to set this bit to a 1

```
P6->DIR = 0x01;          // set P6->DIR bit 0 to 1
or
P6->DIR = b00000001;     // set P6->DIR bit 0 to 1
or
P6->DIR = 1;             // set P6->DIR bit 0 to 1
```



These set the other 7 pins to 0
this may not be OK

Register Access

- Register Configuration

  Set bit 0 in register P6->DIR to 1

  ```
  P6->DIR |= 0x01;              // set P6->DIR bit 0 to 1
  ```

  P6->DIR = bABCDEFGH | b00000001
  P6->DIR = bABCDEFG1
  Only the bit we want to change is changed

Register Access

- Register Configuration

Set bit 2 in register P3->DIR to 0

```
P3->DIR &= 0xFB;                // set P3->DIR bit 2 to 0
```
or
```
P3->DIR &= b11111011;           // set P3->DIR bit 2 to 0
```

P3->DIR = bABCDEFGH & b11111011

P3->DIR = bABCDE0GH

Only the bit we want to change is changed

Not intuitive

Register Access

- Register Configuration

Set bit 2 in register P3->DIR to 0

```
P3->DIR &= ~0x04;                // set P3->DIR bit 2 to 0
```
or
```
P3->DIR &= ~b0000100;            // set P3->DIR bit 2 to 0
```

P3->DIR = bABCDEFGH & ~b00000100

P3->DIR = bABCDEFGH & b11111011

P3->DIR = bABCDE0GH

Only the bit we want to change is changed

More intuitive

Register Access

- Register Configuration
  - Change a few bits in a register to 1s and 0s

```
// Modify bits 5 and 7 to 1s, and 3 and 12 to zeros
// without altering any other bits

TIMER_A1->CTL = ((TIMER_A1->CTL & ~0x1008) | 0x00A0)
```