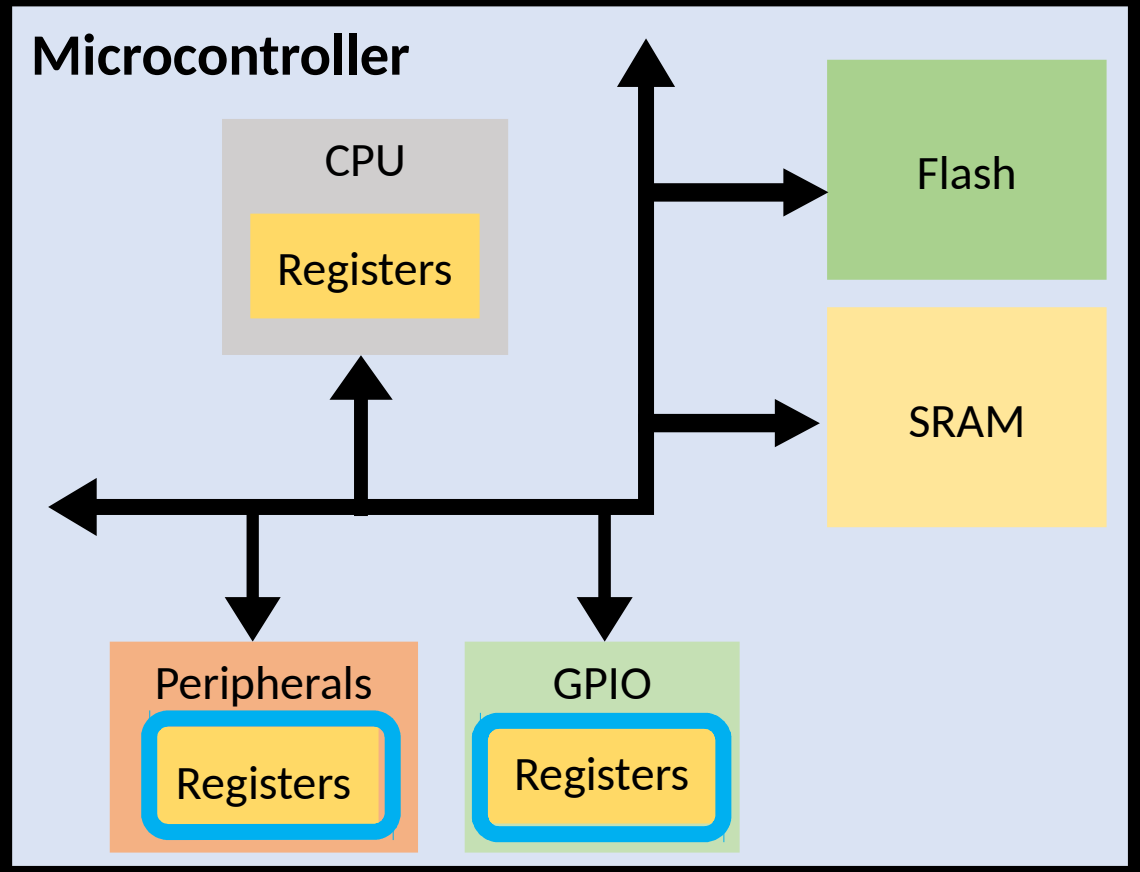
Register Definition Files

Embedded Software Essentials

Embedded System Memories [S1]

- Memories of an Embedded System
  - Code Memory (Flash)
  - Data Memory (SRAM)
  - Register Memory (internal to chip)
  - External Memory (if applicable)

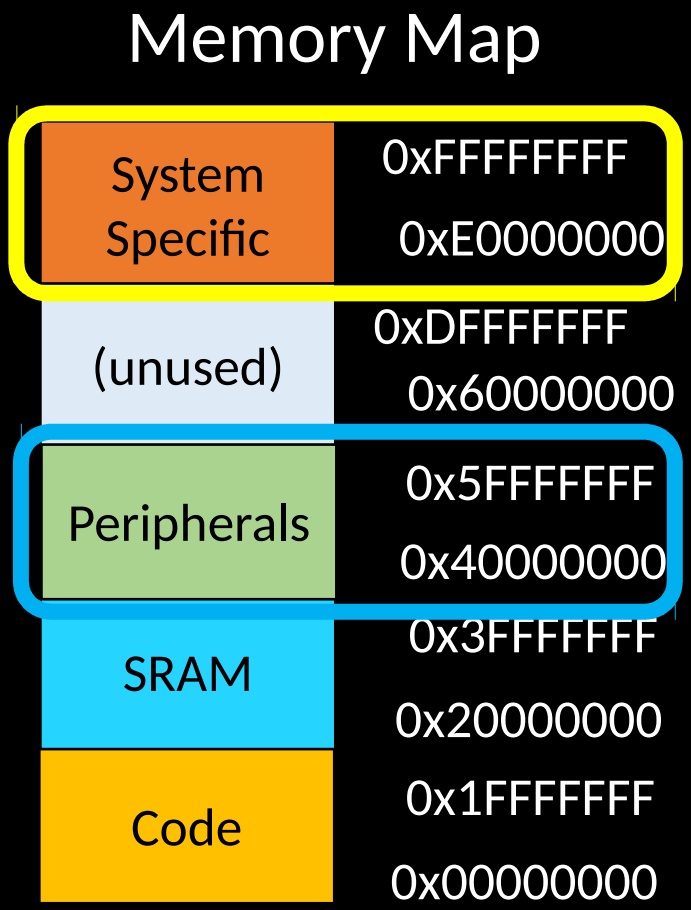
- Peripherals have special functionality each with a reserved spot in the memory map



Addresses specified in a Register Definition File

Peripheral Memory [S2]

- Microcontrollers contain multiple peripherals
  - Private
    - System Control Block (SCB)
    - Nested Vector Interrupt Controller (NVIC)
  - General
    - UART / SPI / I2C
    - Timers
    - ADC

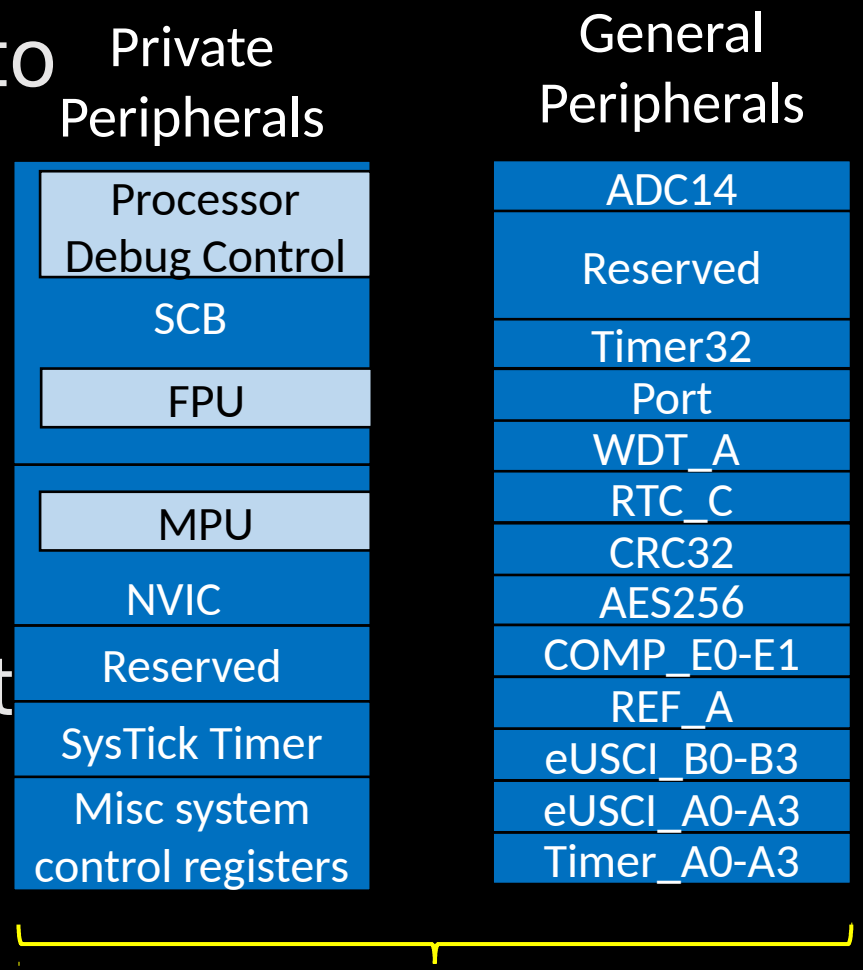


Access Peripherals with Pointers

```
volatile uint16_t * ta0_ctrl = (uint16_t*)0x40000000;
*ta0_ctrl = 0x0202;
```

Register Definition File [S3]

- Platform File that provides interface to peripheral memory by specifying
  - Address List for Peripherals
  - Access Methods
  - Defines for Bit Field and Bit Masks
- Peripheral Access Methods are used t read/write data
  - Direct Dereferencing of Memory
  - Structure Overlays



Directly Dereference Memory [S4]

- Do not need a pointer variable to read/write to memory

Timer_A0 Control Register, Address = 0x40000000

15:10	9	8	7	6	5	4	3	2	1	0
Reserved	TASSEL		ID		MC		Reserved	TACLR	TAIE	TAIFG

```
(*(volatile uint16_t*)0x40000000)) = 0x0202;
```

Dereference!

Equivalent to Defining a Pointer!

```
volatile uint16_t * ta0_ctrl = (uint16_t*)0x40000000;
*ta0_ctrl = 0x0202;
```

Not Very Readable or Maintainable…

Memory Access Macros [S5]

- Use preprocessor to define an access method without using hardcoded values

```
/ * 8, 16, & 32 Bit Register Access Macros */
#define HWREG8(x)      (*((volatile   uint8_t *)(x)))
#define HWREG16(x)     (*((volatile uint16_t *)(x)))
#define HWREG32(x)     (*((volatile uint32_t *)(x)))
```

Pass in any address to access a register

```
#define TA0CTL          (HWREG16(0x40000000))
```

Example Use of Access Macro:

```
TA0CTL = 0x0202;
```

```
volatile uint16_t * ta0_ctrl = (uint16_t*)0x40000000;
*ta0_ctrl = 0x0202;
```

Timer A0 Registers [S6]

```
#define HWREG16(x)     (*((volatile uint16_t *)(x)))

/ * Example Use of Access Macros */
#define TA0CTL          (HWREG16(0x40000000))
#define TA0CCTL0        (HWREG16(0x40000002))
#define TA0CCTL1        (HWREG16(0x40000004))
#define TA0CCTL2        (HWREG16(0x40000006))
#define TA0CCTL3        (HWREG16(0x40000008))
#define TA0CCTL4        (HWREG16(0x4000000A))
#define TA0R            (HWREG16(0x40000010))
#define TA0CCR0         (HWREG16(0x40000012))
#define TA0CCCR1        (HWREG16(0x40000014))
#define TA0CCCR2        (HWREG16(0x40000016))
#define TA0CCCR3        (HWREG16(0x40000018))
#define TA0CCCR4        (HWREG16(0x4000001A))
#define TA0EX0          (HWREG16(0x40000020))
#define TA0IV           (HWREG16(0x4000002E))
```

Timer_A0 Registers

Timer_A0 Interrupt Vector	0x4000002E
Timer_A0 Expansion 0	0x40000020
Timer_A0 CAPCOM 4	0x4000001A
Timer_A0 CAPCOM 3	0x40000018
Timer_A0 CAPCOM 2	0x40000016
Timer_A0 CAPCOM 1	0x40000014
Timer_A0 CAPCOM 0	0x40000012
Timer_A0 Counter	0x40000010
Timer_A0 CAPCOM Control 4	0x4000000A
Timer_A0 CAPCOM Control 3	0x40000008
Timer_A0 CAPCOM Control 2	0x40000006
Timer_A0 CAPCOM Control 1	0x40000004
Timer_A0 CAPCOM Control 0	0x40000002
Timer_A0 Control	0x40000000

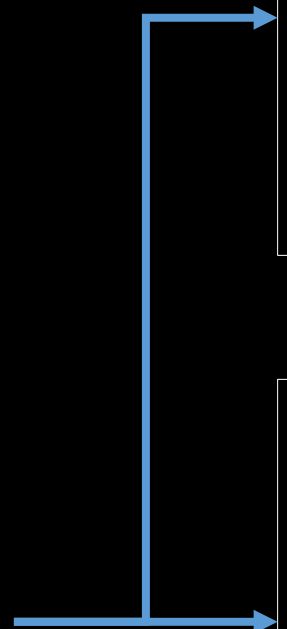
Timer A0 Registers [S6b]

Use Compile Time Switches to include correct Register Definition File

$ make build PLATFORM=MSP432_VER1

MSP432.h

```
#ifdef MSP432_VER1
#include “MSP432_Version1.h”
#else
#include “MSP432_Version2.h”
#endif
```



MSP432_Version1.h

```
#define TA0CTL          (HWREG16(0x40000000))
#define TA0CCTL0        (HWREG16(0x40000002))
#define TA0CCTL1        (HWREG16(0x40000004))
#define TA0CCTL2        (HWREG16(0x40000006))
#define TA0CCTL3        (HWREG16(0x40000008))
#define TA0CCTL4        (HWREG16(0x4000000A))
#define TA0R            (HWREG16(0x40000010))
/* More Register Macros Below */
```

MSP432_Version2.h

```
#define TA0CTL          (HWREG16(0x60000000))
#define TA0CCTL0        (HWREG16(0x60000002))
#define TA0CCTL1        (HWREG16(0x60000004))
#define TA0CCTL2        (HWREG16(0x60000006))
#define TA0CCTL3        (HWREG16(0x60000008))
#define TA0CCTL4        (HWREG16(0x6000000A))
#define TA0R            (HWREG16(0x60000010))
/* More Register Macros Below */
```

Volatile Keyword [S7]

- Volatile tells compiler NOT to optimize this code
  - Volatile variable needs to be directly read and written when specified

```
/ * 8, 16, & 32 Bit Register Access Macros */
#define HWREG8(x)      (*((volatile   uint8_t *)(x)))
#define HWREG16(x)     (*((volatile uint16_t *)(x)))
#define HWREG32(x)     (*((volatile uint32_t *)(x)))
```

- Peripherals should be configured as soon as code executes, not moved to a later point in time

Structure Overlay [S8a]

- Define a Structure to directly match peripheral region registers

```
typedef struct {
  __IO uint16_t CTL;
  __IO uint16_t CCTL[7];
  __IO uint16_t R;
  __IO uint16_t CCR[7];
  __IO uint16_t EX0;
  uint16_t RESERVED0[6];
  __I  uint16_t IV;
} Timer_A_Type;


#define __IO (volatile)
#define __I    (volatile const)
```

Structure Overlay [S8b]

- Define a Structure to directly match peripheral region registers

```
typedef struct {
  __IO uint16_t CTL;
  __IO uint16_t CCTL[7];
  __IO uint16_t R;
  __IO uint16_t CCR[7];
  __IO uint16_t EX0;
  uint16_t RESERVED0[6];
  __I  uint16_t IV;
} Timer_A_Type;
```

```
#define __IO (volatile)
#define __I    (volatile const)
```

```
/* Define the Base Address of Peripheral Regions */
#define PERIPH_BASE          ((uint32_t) 0x40000000)
#define TIMER_A0_BASE        (PERIPH_BASE + 0x00000000)
#define TIMER_A1_BASE        (PERIPH_BASE + 0x00000400)
#define TIMER_A2_BASE        (PERIPH_BASE + 0x00000800)
```

Structure Overlay [S8c]

- Define a Structure to directly match peripheral region registers

```
typedef struct {
  __IO uint16_t CTL;
  __IO uint16_t CCTL[7];
  __IO uint16_t R;
  __IO uint16_t CCR[7];
  __IO uint16_t EX0;
  uint16_t RESERVED0[6];
  __I  uint16_t IV;
} Timer_A_Type;
```

```
#define __IO (volatile)
#define __I  (volatile const)
```

```
/* Define the Base Address of Peripheral Regions */
#define PERIPH_BASE          ((uint32_t) 0x40000000)
#define TIMER_A0_BASE        (PERIPH_BASE + 0x00000000)
#define TIMER_A1_BASE        (PERIPH_BASE + 0x00000400)
#define TIMER_A2_BASE        (PERIPH_BASE + 0x00000800)

/ * Multiple Timer Modules, different Addresses */
#define TIMER_A0       ((Timer_A_Type *) TIMER_A0_BASE)
#define TIMER_A1       ((Timer_A_Type *) TIMER_A1_BASE)
#define TIMER_A2       ((Timer_A_Type *) TIMER_A2_BASE)
```

Example Use of Structure Overlay:

```
TIMER_A0->CTL = 0x0202;
```

Structure Overlay [S9]

- Structure Overlays require exact replica of peripheral region
  - Size registers to equivalent standard types
  - Order matters
  - Leave space for reserved bytes
  - Read-Only Registers = Const!
  - All registers are volatile

Unused memory in the peripheral region →

```
#define __IO (volatile)
#define __I  (volatile const)

typedef struct {
  __IO uint16_t CTL;
  __IO uint16_t CCTL[7];
  __IO uint16_t R;
  __IO uint16_t CCR[7];
  __IO uint16_t EX0;
  uint16_t RESERVED0[6];
  __I  uint16_t IV;
} Timer_A_Type;
```

Structure Overlay Example [S10]

- Use Structure pointer and deference it to access the individual registers

```
typedef struct {
  __IO uint16_t CTL;
  __IO uint16_t CCTL[7];
  __IO uint16_t R;
  __IO uint16_t CCR[7];
  __IO uint16_t EX0;
  uint16_t RESERVED0[6];
  __I  uint16_t IV;
} Timer_A_Type;
```

```
#define PERIPH_BASE        ((uint32_t) 0x40000000)
#define TIMER_A0_BASE      (PERIPH_BASE + 0x00000000)
#define TIMER_A0           ((Timer_A_Type *) TIMER_A0_BASE)
#define __IO (volatile)
#define __I    (volatile const)
```

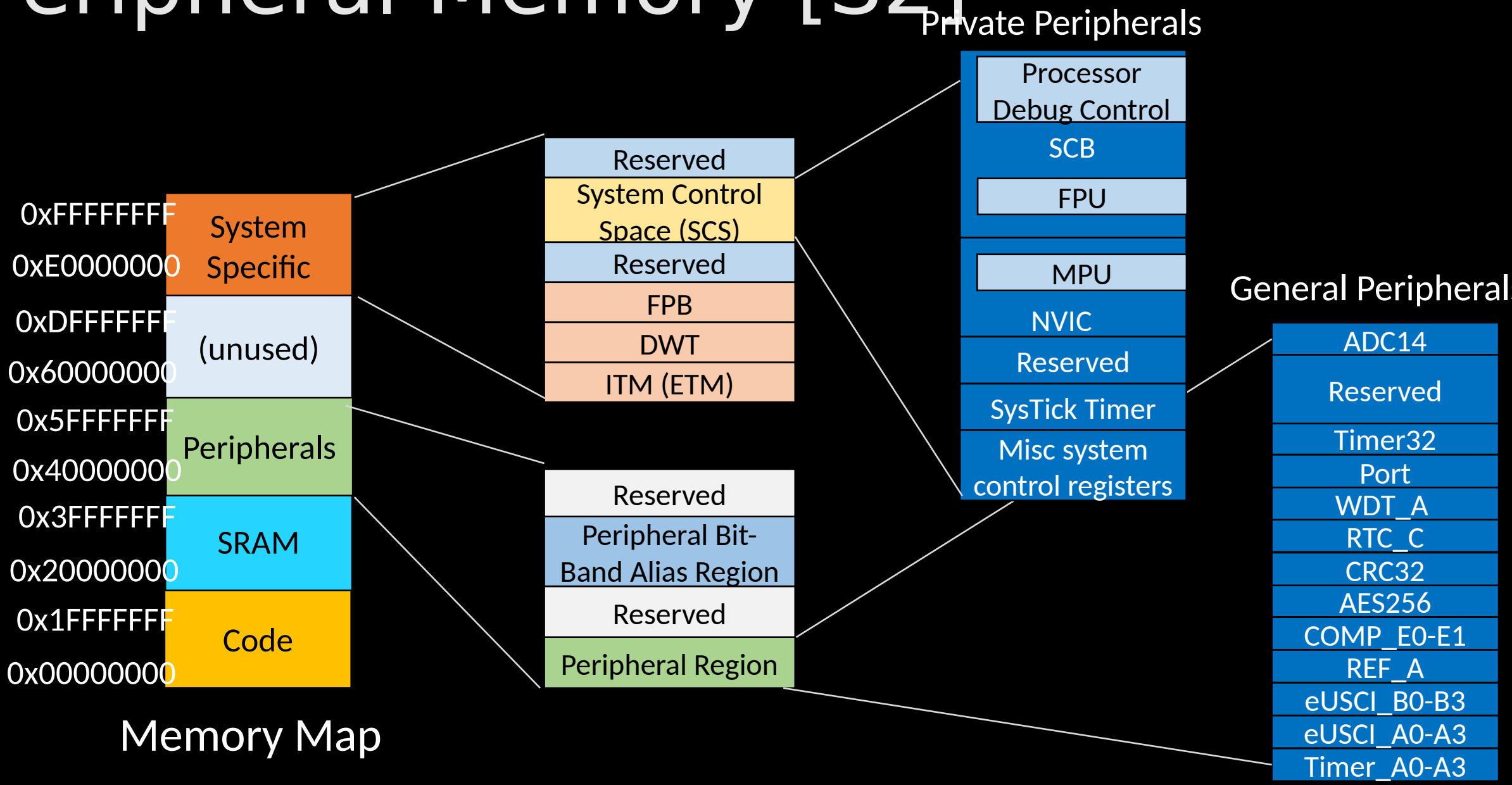
Example Use of Structure Overlay:

```
TIMER_A0->CTL = 0x0202;
```

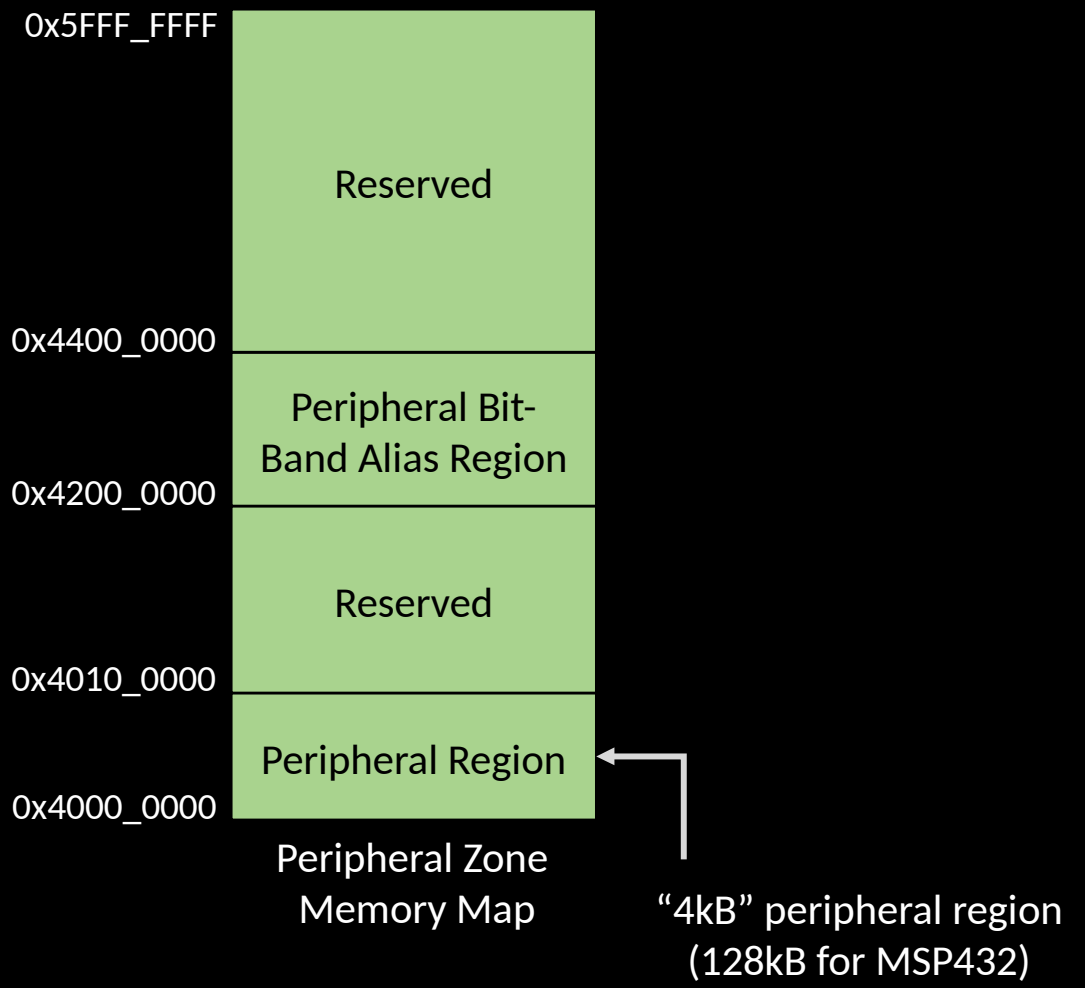
Unused Slides

Ignore all slides below this

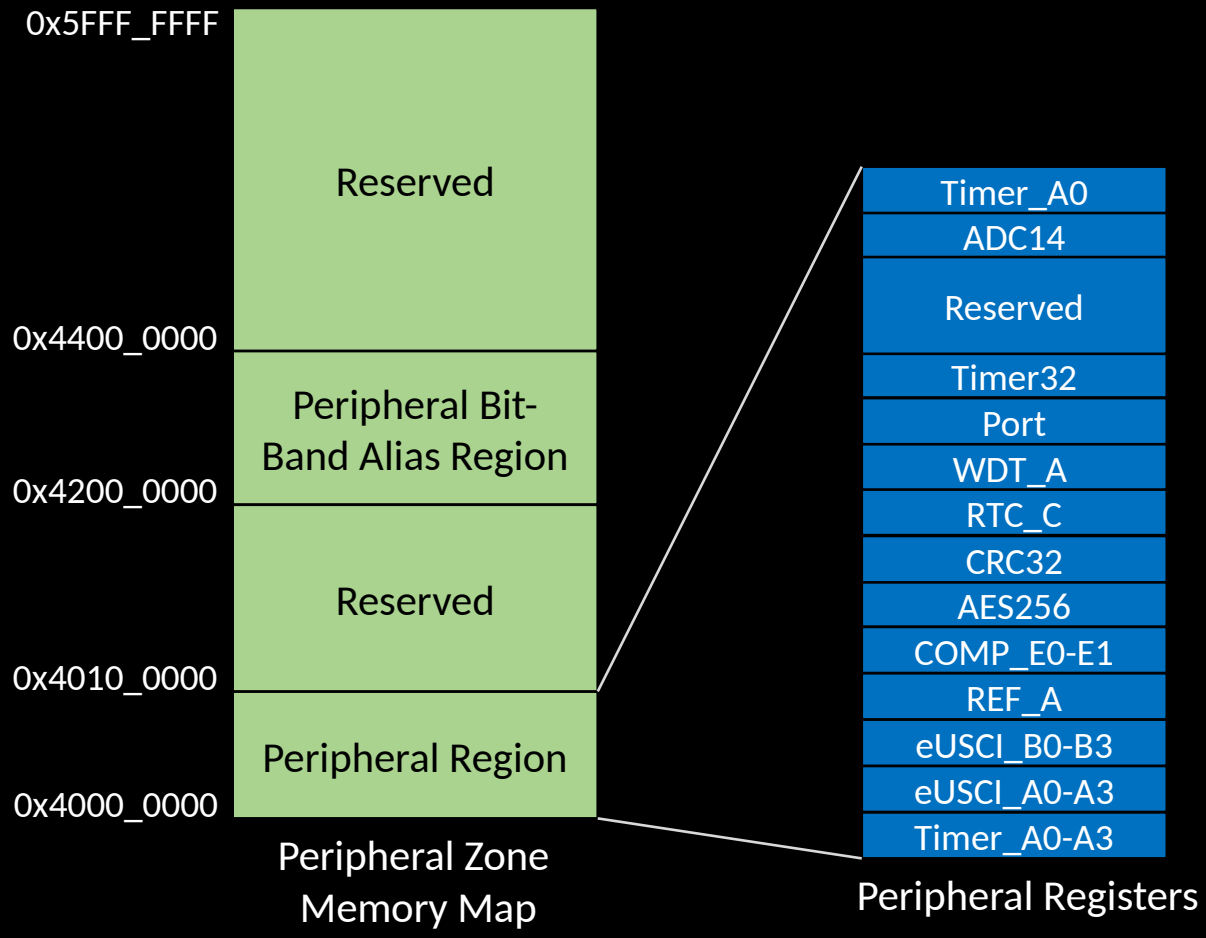
Peripheral Memory [S2]



Peripheral Registers in MSP432 [S1.3.6.c]



Peripheral Registers in MSP432p401r [S1.3.6.c]



Peripheral Registers in MSP432p401r [S1.3.6.c]