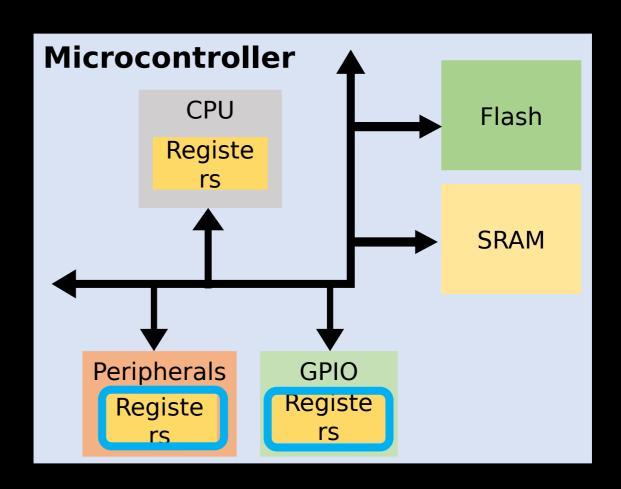
# Register Definition Files

**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 A the with a presime wed spot in the memory meanister

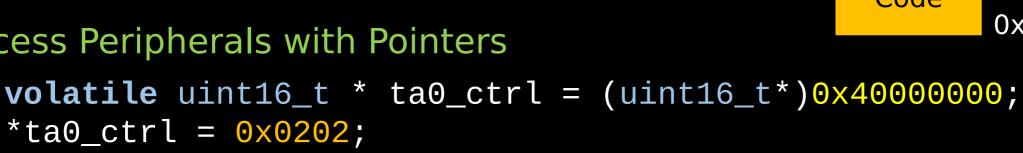
  Definition File



## 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



#### Memory Map

```
0xFFFFFF
System
         0xE000550
Specific
        XDFFFFFF
(unused)
          0x60000000
          0x5FFFFFF
Periphera
         0x4000000
   IS
          UX3FFFFFF
 SRAM
         0x20000000
          0x1FFFFFFF
 Code
         0x0000000
```

#### Register Definition File [S3]

Platform File that provides Interface to Private Peripherals
 Peripherals

- 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

Peripherals Processor Debug Control **FPU MPU** NVIC Reserved SysTick Mis<sup>Ei</sup>system control registers

General Peripherals ADC14 Reserved Timer32 Port WDT A

Timer A0-A3

Could Contain 1000's of Registers!

### Directly Dereference Memory [S4]

 Do not need a pointer variable to read/write to memory
 Timer A0 Control Register, Address =

```
15:10 9 8 7 6 5 4 3 2 1 0

Reserved TASSEL ID MC Reserved Reserved TAGE TAIE
```

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

```
Dereferen ce!
```

```
Equivalent toolatile uint16_t * ta0_ctrl = (uint16_t*)0x40000000

Defining a *ta0_ctrl = 0x0202;

Pointer!
```

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 any address to access a Register
  uint16 t*)(x)
  #define HWREG32(x)
                           (*((volatile
  uint32 t*)(x))
Example Use of Access Macro:
                            volatile uint16 t * ta0 ctrl = (uint16 t*)0x4000000
TA0CTL = 0x0202; ◆ #define TA0CTL
                            *ta0 ctrl = 0x0202;
  (HWREG16(0x40000000))
```

#### Timer A0 Registers [S6]

```
#define HWREG16(x) (*((volatile uint16 t *)
(x)))
/ * Example Use of Access Macros */
#define TAOCTL
(HWREG16(0\times40000000))
#define TA0CCTL0
(HWREG16(0\times40000002))
#define TA0CCTL1
(HWREG16(0\times40000004))
#define TA0CCTL2
(HWREG16(0\times40000006))
#define TA0CCTL3
(HWREG16(0\times40000008))
#define TA0CCTL4
(HWREG16(0\times4000000A))
#define TAOR
(HWREG16(0\times40000010))
```

#### Timer A0

Tim Re gostbersupt	0x400000
Vector Timer_A0 Expansion 0	₽₹400000
Timer_A0 CAPCOM 4	<b>@</b> \$400000
Timer_A0 CAPCOM 3	<b>0</b> ×400000
Timer_A0 CAPCOM 2	<b>₿</b> 8400000
Timer_A0 CAPCOM 1	<b>0</b> £400000
Timer_A0 CAPCOM 0	<b>0</b> 4400000
Timer_A0 Counter Timer_A0 CAPCOM	<b>D</b> 2400000
_	<b>Q</b> 2400000
IImeFantrelation	<b>0</b> &400000
IImeF9Ntrel3	<b>0</b> &400000
TimeFantrel 1	<b>0</b> \&400000
TimeFantreAlcom	0 <b>4</b> 400000
Control 0 Timer_A0 Control	<b>02</b> 400000
	00

## 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(0\times40000000))
#define TA0CCTL0
(HWREG16(0\times40000002))
#define TA0CCTL1
(HWREG16(0x40000004))
#define TA0CCTL2
(HWREG16(0\times40000006))
#define TA0CCTL3
(HWREG1662P499008) sion2.h
(HWREG16(0\times60000004))
#define TA0CCTL2
(HWREG16(0x60000006))
#define TA0CCTL3
/UN/DEC16/0v60000000\\
```

#### 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,

#### 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)
                           (PERIPH BASE +
#define TIMER A0 BASE
0x0000000)
#define TIMER A1 BASE
                           (PERIPH BASE +
0x00000400)
#define TIMER A2 BASE
                           (PERIPH BASE +
(0080000x0
```

#### Structure Overlay [S8c]

 Define a Structure to directly match peripheral region registers
 /\* Define the Base Address of Peripheral Regions

```
typedef struct {
 IO uint16 t CTL;
                           #define PERIPH BASE
                                                        ((uint32 t)
 IO uint16 t CCTL[7];
                           0x40000000)
 IO uint16 t R;
                           #define TIMER A0 BASE
                                                        (PERIPH BASE +
 IO uint16 t CCR[7];
                           0x0000000)
 IO uint16 t EX0;
                           #define TIMER A1 BASE
                                                        (PERIPH BASE +
 uint16 t
                           0x00000400)
RESERVED0[6];
                           #define TIMER A2 BASE
                                                        (PERIPH BASE +
 I uint16 t IV;
                           (0080000x0
} Timer A Type;
#define IO (volatile)
                        Example tipse of near unto the overflerent Addresses */
                        TIME TRANS TO TIME AND 1202; ((Timer A Type *)
#define I (volatile
                           TIMER A0 BASE)
const)
```

### Structure Overlay [S9]

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

Unused memory in the peripheral regi<del>on</del>

```
#define I (volatile
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];
    uint16 t IV;
} Timer A Type;
```

### Structure Overlay Example [S10]

 Use Structure pointer and deference it to access the individual registers

```
#define PERIPH BASE
                                                  ((uint32 t)
typedef struct {
                          0x4000000)
 IO uint16 t CTL;
                          #define TIMER A0 BASE
                                                  (PERIPH BASE +
  IO uint16 t CCTL[7];
                          0x0000000)
  IO uint16 t R;
                          #define TIMER A0
                                                   ((Timer A Type *)
 IO uint16 t CCR[7];
                          TIMER A0 BASE)
 IO uint16 t EXO;
                          #define IO (volatile)
 uint16 t
                          ## Stribe Uset of Stribe Coverlay:
RESERVED0[6];
                           TIMER A0->CTL = 0\times0202;
  I uint16 t IV;
} Timer A Type;
```

# Unused Slides

Ignore all slides below this

# Peripheral Memory [S2] ate Peripherals

0xFFFFFFFF<sub>System</sub> 0xE00000005pecific 0xDFFFFFI (unused) 0x6000000 0x5FFFFFF Periphera 0x400000 0x3FFFFFFF SRAM 0x2000000 0x1FFFFFFF Code 0x0000000

Reserved **Control Space** Reserved **FPB DWT** ITM (ETM)

Reserved renpheral bit-**Band Alias** Region Reserved Peripheral Region

Debug Control **FPU** 

**MPU** 

**NVIC** 

Reserved SysTick Mis&isystem control

registers

General Peripherals

ADC14 Reserved

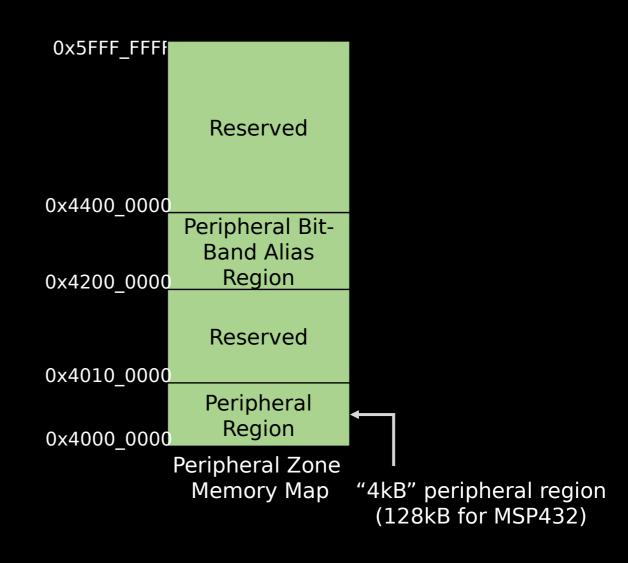
Timer32 Port WDT A RTC C CRC32 **AFS256** F0-F1 REF A

eUSCI A0-A3

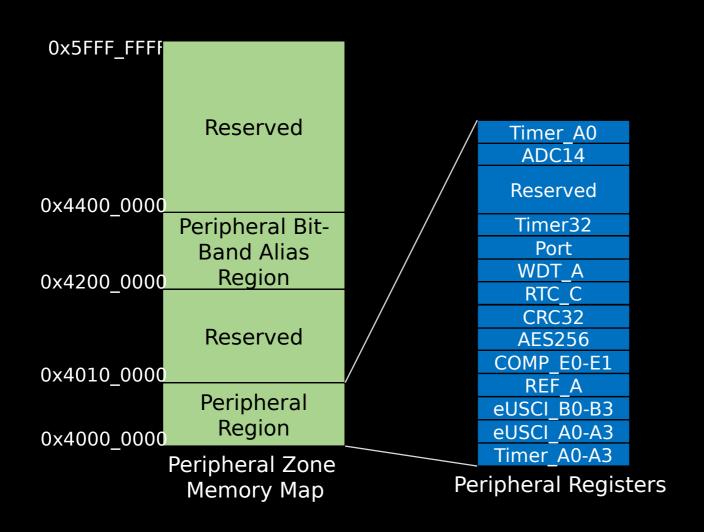
Timer A0-A3

Memory Map

#### Peripheral Registers in MSP432 [S1.3.6.c]



#### Peripheral Registers in MSP432p401r [S1.3.6.c]



#### Peripheral Registers in MSP432p401r [S1.3.6.c]

