TMS320 DSP
Number 30
DESIGNER'S NOTEBOOK
TEXAS INSTRUMENTS

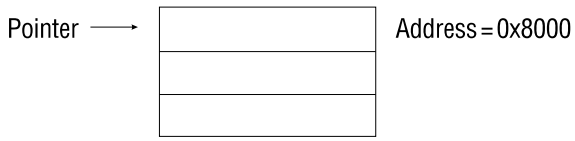
Addressing Peripherals as Data Structures in C

Contributed by Nat Seshan and Mark Utter

Design Problem How can I manipulate a DSP's device peripheral-specific registers in C?

Solution A data structure is usually assigned to .bss by the C compiler. A peripheral such as a serial port has control registers with an address different from .bss. The problem is to connect the two.

Method 1: Use a pointer to the peripheral.



Peripheral as memory locations

1.1 First declare a structure that logically represents the memory locations of the peripheral.

```
struct controller {
                    unsigned int status;
                    ...
                  };
```

1.2 Declare a pointer to the structure and initialize it to the peripheral's address.

```
struct controller *IFperipheral = (struct controller *) 0x8000;
```

1.3 In your code, access the peripheral's memory values indirectly.

```
IFperipheral -> status = 0;
```

Method 2: Placing the structure in its own section.

2.1 Declare a peripheral instead of a pointer.

```
struct controller IFperiph;
```

2.2 Use inline assembly to give the structure its own section.

```
asm("_IFperiph .usect  \"periph\", 128); /* 128 is size of
                                           struct */
```

This creates a user-defined section that can be linked to any address.

2.3 Use your linker command file to map the section to memory.

```
periph:  load = 0x8000
```

2.4 Address the structure elements directly.

```
IFperiph.status = 0;
```

Both methods work. Sometimes the pointer method is most efficient. Other times, the second method is best. Method 1 is very useful for addressing peripheral or memory buffers which are device specific. Method 2 is preferred for addressing peripherals or memory buffers which are not device specific (i.e., peripherals are user specified). This method ensures the task of mapping and aligning user-specific peripherals and/or memory buffers to the linker. The choice depends on your individual application. For more information, read the *TMS320C30 Peripheral Runtime Support Library Users Guide*. Also see: *DSP Applications Using the 'C30 EVM*, "C Coding Tips for Application-Specific Processors."