```
message_t* packetPtr = &packet;
```

3.3.4 Generic components

By default, components in TinyOS are *singletons*: only one exists. Every configuration that names a singleton component names the same component. For example, if two configurations wire to LedsC, they are wiring to the same code that accesses the same variables. A singleton component introduces a component name that any configuration can use into the global namespace.

In addition to singleton components, nesC has *generic components*. Unlike singletons, a generic component can have multiple instances. For example, while a low-level software abstraction of a hardware resource is inherently a singleton – there is only one copy of a hardware register – software data structures are instantiable. Being instantiable makes them reusable across many different parts of an application. For example, the module BitVectorC provides the abstraction of a bit vector; rather than define macros or functions to manipulate a bit vector a module can just use the interface BitVector and assume that a corresponding configuration connects it to a BitVectorC of the proper width.

Earlier versions of nesC (1.0 and 1.1) did not support generic components. Whenever a component requires a common data structure, a programmer had to make a copy of the data structure component and give it a new name, or separate functionality and allocation by locally allocating data structures and using library routines. For example, network protocols typically all implemented their own queue data structures, rather than relying on a standard implementation. This code copying prevented code reuse, forcing programmers to continually revisit common bugs and problems, rather than building on well-tested libraries.

Generic components have the keyword **generic** before their signature:

```
generic module SineSensorC() {
  provides interface Init;
  provides interface Read<uint16_t>;
}
```

```
generic configuration TimerMilliC() {
  provides interface Timer<TMilli>;
}
```

Listing 3.21 Generic module SineSensorC and generic configuration TimerMilliC

To use a generic component, a configuration must instantiate it with the **new** keyword. This is the beginning of the code for the configuration BlinkAppC, the top-level configuration for the Blink application, which displays a 3-bit counter on a mote's LEDs using three timers: