```
// C Code to control the AC Relay Wildcard

// Relays are active low (i.e. writing a 0 to the relay turns it on).
#define RELAY_ON                    0
#define RELAY_OFF                   1

void Control_AC_Relay ( uchar module_number, uchar relay_num, uchar
state )
// Sets the relay number to the appropriate state (on or off).
// Valid relay numbers are 0-3.  Valid module numbers are 0-7.
{
  EXTENDED_ADDR module_addr;

  module_addr.page16 = module_number;
  module_addr.addr16 = RELAY_CONTROL_REGISTER;

  if(state) // turn relay off
  {
    state = state << relay_num;
    SetBits( state, module_addr.addr32 );
  }
  else      // turn relay on
  {
    state = 1 << relay_num;
    ClearBits ( state, module_addr.addr32 );
  }
}

uchar Read_AC_Relay_Status ( uchar module_number )
// Reads the current state of the AC Relays.  Valid module numbers are
0-7.
// Returns a character whose least significant nibble represents the
four
// relays.  For example, if 1 is returned (0001 in binary), then Relay
0 is
// off and the other relays are on.  If 12 is returned (1100 in
binary),
// then relays 2 and 3 are off and 0 and 1 are on.  The four most
significant
// bits do not matter.
{
  EXTENDED_ADDR module_addr;
  Char ac_relay_status;

  module_addr.page16 = module_number;
  module_addr.addr16 = RELAY_CONTROL_REGISTER;

  ac_relay_status = FetchChar( module_addr );

  return( ac_relay_status );
}
```