```
// SoCDAM UART DEVICE DRIVER example code.

#define IO_BASE 0xFFFC1000 // or whatever

#define U_SEND    0x10
#define U_RECEIVE 0x14
#define U_CONTROL 0x18
#define U_STATUS  0x1C

#define  UART_SEND()      (*((volatile char *)(IO_BASE+U_SEND)))
#define  UART_RECEIVE()   (*((volatile char *)(IO_BASE+U_RECEIVE)))
#define  UART_CONTROL()   (*((volatile char *)(IO_BASE+U_CONTROL)))
#define  UART_STATUS()    (*((volatile char *)(IO_BASE+U_STATUS)))

#define UART_STATUS_RX_EMPTY (0x80)
#define UART_STATUS_TX_EMPTY (0x40)

#define UART_CONTROL_RX_INT_ENABLE (0x20)
#define UART_CONTROL_TX_INT_ENABLE (0x10)




// Polled I/O routines:

  char uart_polled_read()
  {
     while (UART_STATUS() &
        UART_STATUS_RX_EMPTY) continue;
     return UART_RECEIVE();
  }


  uart_polled_write(char d)
  {
     while (!(UART_STATUS() &
       UART_STATUS_TX_EMPTY)) continue;
     UART_SEND() = d;
  }




// Interrupt driven receive routine:
// Circular FIFO buffers.
char rx_buffer[256], tx_buffer[256];
int rx_inptr, rx_outptr, tx_inptr, tx_outptr;

void uart_reset()
{
  rx_inptr = 0;
  rx_output = 0;
  tx_inptr = 0;
  tx_output = 0;
  UART_CONTROL() |= UART_CONTROL_RX_INT_ENABLE;
}


char uart_read()
{
  while (rx_inptr==rx_outptr) wait();
  char r = buffer[rx_outptr];
  rx_outptr = (rx_outptr + 1) & 255;
  return r;
```

```
}

// Interrupt service routine:
// Uart ISR: this is called from a short assembler stub placed at the
// hardware interrupt vector location.  The assembler stub sets up the
// stack pointer and frame pointer and saves any registers that might
// be in use in non-interrupt contexts.
char uart_rx_isr()  // interrupt service routine
{
  while (1)
    {
      if (UART_STATUS() & UART_STATUS_RX_EMPTY) return;
      rx_buffer[rx_inptr] = UART_RECEIVE();
      rx_inptr = (rx_inptr + 1) & 255;
    }
}
// on return from the ISR, the processor context is restored, including
// any interrupt mask flag in the main processor control word.



//
uart_write(char c)
{
  while (((tx_inptr+1) & 255)==tx_outptr) wait();
  buffer[tx_inptr] = c;
  tx_inptr = (tx_inptr + 1) & 255;

  UART_CONTROL() |= UART_CONTROL_TX_INT_ENABLE;
}

// Typically the transmit and receive ISRs are not separate.  Instead,
// all forms of service request are checked by one entry point in the
// C device driver code.
char uart_tx_isr()
{
  while (tx_inptr != tx_outptr) // There may be an output FIFO, so send as many bytes as
possible.
    {
      if (!(UART_STATUS() & UART_STATUS_TX_EMPTY)) return; // Return with tx interrupt
enabled.
      UART_SEND() = tx_buffer[tx_outptr];
      tx_outptr = (tx_outptr + 1) & 255;
    }
  UART_CONTROL() &= ~UART_CONTROL_TX_INT_ENABLE;  // Return with tx interrupt disabled.
}


// END (C) 1995-2011 DJ Greaves
```