IAR Application Note G - 001
Generic Software Uart written in C

SUMMARY
This application note details a generic software UART written in C which can be implemented on any microprocessor with a C compiler. It requires a timer interrupt to be set to 3 times the baud rate, and two software-controlled pins for the receive and transmit functions.

KEYWORDS
UART, C code

The Problem to be solved
Software UARTs are generally coded in assembly for speed purposes, but with the increasing speed of processors, a software UART written in C is now easily implemented and highly portable.

The Difficulties Involved
The implementation has three fundamental requirements:

1. A compiler must exist for the microprocessor.
2. A timer interrupt must be set to interrupt at three times the required baud rate. Most microprocessors have a timer interrupt that can be used for this, or an external clock interrupt could be used to achieve the same goal.
3. There must be a transmit pin which can be set from the software, and a receive pin which can be read by the software.

The Solution

The C source file must be linked into the user's application with the necessary interface routines listed as follows:

1. void get_rx_pin_status(void)
   Returns 0 or 1 dependent on whether the receive pin is high or low.
2. void set_tx_pin_high(void)
   Sets the transmit pin to the high state.
3. void set_tx_pin_low(void)
   Sets the transmit pin to the low state.
4. void idle(void)
   Background functions to execute while waiting for input.
5. void timer_set(int BAUD_RATE)
   Sets the timer to 3 times the baud rate.

6. void set_timer_interrupt(timer_isr)
   Enables the timer interrupt.

The baud rate is selectable by changing the BAUD_RATE macro at the top of the source file, and can be set to as high a value as the timer can support.

#define BAUD_RATE 19200.0

The user's initialization routine must initially call the init_uart() function before any other UART function.

The following standard functions are provided:

a. void flush_input_buffer(void)
   Clears the contents of the input buffer.
b. char kbhit(void)
   Tests whether an input character has been received.
c. char getchar(void)
   Reads a character from the input buffer, waiting if necessary.
d. void putchar(char)
   Writes a character to the serial port.
e. void turn_rx_on(void)
   Turns on the receive function.
f. void turn_rx_off(void)
   Turns off the receive function.

Note that received characters are buffered so that there is no loss of data for a continuous sequence of characters. The idle() function provides the user with the capability to perform background processing while the getchar() function is called waiting on keystroke input.

User Benefits

Clearly, the low-level adaptations that require many lines of assembly code can be done in C using the IAR C language extensions, thus increasing readability and portability.

Conclusions

The solution can also be modified to similar serial protocols to RS232, such as the 1553 bus.

References

See the attached listing of the C source code UART.C.