SERIAL_UART16550

The SERIAL_UART16550 is a driver for the generic ‘16550’ serial UART. The driver supports multiple uart devices, one of which may be used as the polled-mode uart for system debugging. This must be the first uart (*BASE0*) defined in the system.

Process Information

Prototype Name	serial_uart16550
Process Priority	driver-level
Process Name	should correspond with the project definition of the Console serial device if the standard Console process is used.

Module Options

SERIAL_UART16550_ENTER_DEBUG	If this symbol is set to a C character constant, then that character will cause the serial interrupt handler to enter the debugger if it is received as input on the polled-I/O channel. For example, defining the symbol as ‘!’ will make the ! character enter the debugger.
SERIAL_UART16550_NO_POLLEDIO	If this symbol is defined, the polled-mode serial routines will *not* be complied into this module. This is in case multiple serial modules are configured in a system, only one of which should provide the polled-mode routines used by *rome_kprintf*.
SERIAL_UART16550_SKIP_INIT	If this symbol is defined, the polled-mode uart port will *not* be reconfigured during system startup. This may be need during early system debugging if the uart initialisation code fails.

Target File Definitions

SERIAL_UART16550_ADD_HANDLER	The routine to call to add the interrupt handler into the system. This is usually *rome_add_handler* but may need to be changed if the uart is on a different bus, for example.
SERIAL_UART16550_ADD_INCLUDE	An additional header file to be included for this version of the driver, for example to bring in the prototype for a special interrupt handler.

SERIAL_UART16550_BASE*n*	The base address of the *n*th uart register set.
SERIAL_UART16550_BAUD_RATE	The baud rate for the polled-mode uart. The defaul value in 9600 baud.
SERIAL_UART16550_BAUD_SHIFT	The baud rate shift factor to match the rate with the crystal. The default value is 4, for a multiplier of 16.
SERIAL_UART16550_CLEAR_INT*n*	An (optional) routine to be called to clear the *n*th uart interrupt condition.
SERIAL_UART16550_INT*n*	The interrupt pin number for the *n*th uart transmit and receive interrupt.
SERIAL_UART16550_MAXPORTS	The maximum number of uarts supported by this board. If undefined, the default value is 1.
SERIAL_UART16550_MCR	The initial value for the modem control register for the polled-mode serial interface. The default value is (*MCR_RTS* | *MCR_DTR*), but a common alternative is (*MCR_OUT2* | *MCR_DTR*).
SERIAL_UART16550_REG_SPACING	The gap between adjacent registers in the uart register set. The default value is 4 (1 register every 4 bytes), but 1 is another common option.
SERIAL_UART16550_VEC_INT*n*	The interrupt vector number for the *n*th uart transmit and receive interrupt.
SERIAL_UART_XTAL_FREQ	The frequency of the external clock crystal used to drive the polled-mode uart. The default value is 1843200 for the 1.8MHz crystal commonly used to drive uarts.

Process Operation

The initialisation routine installs the interrupt handler and enables the ‘character received’ interrupt for the uart. The module has a queue handler and main process. and accepts the messages defined in the *Standard* messageset for dataflows into and out of the driver with the following processing:

CLOSE messages are handled by the main process. Any outstanding queued requests are returned to the sender with error codes.

FETMBLK messages are added to the uart read queue in the queue handler. The interrupt handler responds to ‘character available’ interrupts by passing input data one-character-at-a-time upwards as replies to FETMBLK (or GETMBLK) messages.

FLUSH messages are added to the tail of the output queue by the main process. They are replied to when they reach the head of the queue in the interrupt handler, thereby ensuring that all preceding output has completed.

GETMBLK messages are turned into single-character reads, using the immediate-parameter area in the message as a buffer, and added to the uart read queue in the queue handler, then treated as FETMBLK messages above.

NEWMBLK messages return a 128-character output buffer to the caller from the queue handler.

OPEN messages are replied to from within the queue handler. It is expected that only one process (usually the Console process) will open a path directly to each uart driver. The open string is of the form "process:*n*" where *n* is the number of the uart (default 0). If there is no uart at index *n* the OPEN message is returned with the *ENODEV* error. Otherwise the index is returned in the *dest_context* field which must be supplied on subsequent messages.

OUTMBLK messages are added to the tail of the output queue in the queue handler, and the 'transmit available' interrupt is enabled if required. The interrupt handler transmits character from the message buffer and handles CRLF translations. The reply is sent when the all characters from the message have been transmitted.

PUTMBLK messages with zero length cause the message's buffer to be returned to the buffer pool in the queue handler, otherwise the queue handler places the messages on the tail of the output queue and message is process as with OUTMBLK above. When all the contents hav been transmitted, the interrupt handler frees the message buffer.

RETMBLK messages are replied to from within the queue handler, with no action being taken, since the driver does not allocate extra space for input buffers.

Shared Library Macros and Routines

The following four routines are used by the ROME core for polled-mode I/O. They are prototyped in *rome.h* to avoid binding the low-level core to a specific serial interface driver.

serial_in

uchar *serial_in*(**void)**

The *serial_in* routine returns the next character from the uart input. It suspends the system until a character is available. To test for input without blocking, use the *serial_poll* routine.

serial_initp

void *serial_initp*(**void**)

The *serial_initp* routine initialises the uart for default polled-mode operation according to the configuration definitions from the hardware file.

serial_out

void *serial_out*(
uchar *c*)

The *serial_out* routine writes the character *c* to the uart, blocking until the uart is able to accept the character. The routine also translates the C 'newline' character into a CR and LF pair suitable for tty-based displays.

serial_poll

int *serial_poll*(**void**)

The *serial_poll* routine returns TRUE if there is a character available in the uart input buffer and FALSE otherwise. The actual character must be read using the *serial_in* routine.

Debug Support

The *serial_uart16550showq* routine is callable from within the debugger , and will print the details of each uart configured in the system including the contexts of its transmit queue (i.e. those messages which have not yet been printed on the uart). This can be useful in the case of a system failure to determine the extent of processing in components that use the standard I/O routines.