REAL-TIME LOW LEVEL SOFTWARE ON THE TAPR TNC

Margaret Morrison, KV7D
4301 E. Holmes Street
Tucson, Arizona 85711
602-325-4775

Introduction

This paper describes the low-level assembly language routines (LLR) of software released with the Tucson Amateur Packet Radio (TAPR) Beta Test terminal node controllers (TNCs). The primary functions performed by these routines are initialization of peripheral devices and data in RAM, maintaining input and output (I/O) buffers, servicing interrupts from peripheral devices, handling nonvolatile RAM data storage and retrieval, and calibration and checkout routines. Entry points are provided which are appropriate to the subroutine calling sequence of the Pascal compiler used for the high-level routines (HLR). In addition, a low-level debug program provides capability for direct access to peripherals, inspection of RAM and ROM locations, and execution of temporary code in RAM. The present LLR code occupies about six kilobytes of ROM.

Initialization

On receipt of a RESTART interrupt by the processor, control passes to the low-level initialization routine. This section first sets up the hardware stack, and then reads the on-board DIP switches which direct the initialization of user-settable parameters. These parameters, which can be set to their default values stored in ROM, or read from the 64 x 4 nonvolatile RAM, are decoded and expanded. The program initializes buffer pointers, counters, timers, and status flags, as well as non-permanent user-settable parameters. The peripheral chips are initialized and checked to verify that they can be commanded. If the UART can not be commanded, the program is aborted and another reset is attempted. Failure of other peripherals results in diagnostic messages. The VIA timer functions are configured so that Timer 1 acts as a pulse generator for the HDLC controller and Timer 2 generates interrupts for software timing functions. The initialization section terminates by enabling interrupts and typing a sign-on message, and passes control to the HLR.

Buffer Management

One of the primary tasks of the LLR is maintaining the I/O buffers. At any time there are four active I/O buffers, input and output buffers for terminal and radio data. An echo buffer, which is configured identically with the terminal output buffer may also be present. Data received from peripherals (terminal and radio interface) are placed into input buffers which are read upon calls from HLR. Data placed into output buffers by calls from HLR are passed to the peripherals under interrupt control. Each buffer has an insertion pointer, which is updated as data are added to the buffer so that it points to the next available cell, and a removal pointer which points to the next cell to be read. All buffers are "circular", meaning that each time a pointer is advanced it is compared with the top of the buffer space, and moved if necessary to the bottom of the buffer space. A buffer is "empty" when the insertion and removal pointers are the same, and "full" when the insertion pointer points to the next cell below the removal pointer. The incoming and outgoing packet buffers contain, as the first bytes of each packet, the byte count of the packet.

In addition to the two basic pointers, input buffers have other markers to facilitate input editing, and output buffers have space-available counters for use by routines writing to these buffers.

A pointer to the beginning of an incoming packet serves two purposes. First, it allows an incoming packet to be purged in case of a reception error (invalid frame-check sequence or insufficient buffer space). Second, it facilitates storing the byte count of the packet upon successful reception.

The terminal input buffer management is rather complex, and operates in three different modes. In "command mode," character and line editing functions are in effect,, and a pointer to the beginning of the current line insures that deletion past this point will not take place in case of rapid terminal input or slow input processing. In *'conversation mode," an additional pointer marks the beginning of the current packet (actually the data portion of the packet), and an additional editing feature allows the user to cancel the current packet. Packets remain in the input buffer until they have been acknowledged, at which time the buffer is updated by a call from HLR. A completed packet is signalled by the receipt of a packet-terminating character, and this character is placed in the buffer as a marker for the end of the packet. In order to prevent commands from interfering with partially typed packets, separate input buffers are maintained, and the pointers for the active buffers are swapped when the mode is changed. This has the effect of moving time-consuming decisions from the interrupt dependent program to subroutines called from HLR.

The third input mode is '*transparent mode," in which all characters received from the terminal are transmitted. Packets are terminated on the basis of number of characters or occurrence of a

timeout. Characters between the removal pointer and the packet marker are divided into maximum-length packets, with the remainder going into a short packet. In order to avoid ambiguity as to the composition of a packet, the packet marker is not updated until all outstanding packets have been acknowledged and cleared from the input buffer.

Interrupt Service

Although the 6809 microprocessor supports two levels of hardware interrupt, a fast interrupt (FIRQ) and a normal interrupt (IRQ), only the IRQ is implemented on the TAPR TNC. All peripheral IRQ lines are wire-ORed together into the IRQ input to the processor. Interrupt outputs from each peripheral can be disabled independently without affecting the processor's response to IRQ. Upon receipt of IRQ, the processor transfers execution to a routine which examines each device in turn and passes control to a routine specified in a dispatch table* For maximum flexibility, the interrupt dispatch table is stored in RAM. The addresses in the table are initialized during the startup procedure, and are changed as the TNC operates in different modes.

The peripheral devices are assigned priorities according to the order in which the dispatch routine checks their status. The priority of service is

1. UART (terminal) input
2. UART output
3. Timer interrupt
4. All HDLC (radio interface) interrupts

Interrupts from the parallel I/O port are not enabled in the initial software release; when they they will be assigned lowest priority. The HDLC is currently assigned lowest priority because the interrupt service is quite complex and the chip will generate spurious interrupts at a high rate under noisy conditions. The UART was placed ahead of the timer, since the 6522 timer mode used provides a mechanism for compensating for delayed interrupt servicing. The UART output ought properly to be assigned a lower priority, but it was placed after the input service for convenience, since input and output status are given in the same register. Parallel port I/O must be assigned the lowest priority to insure that it does not interfere with other interrupt service.

As a diagnostic tool, the interrupt dispatch routine writes signals reflecting interrupt conditions to the parallel port, which is configured as 16 output bits for the initial software release.

Terminal I/O

The terminal input interrupt service normally consists of two parts: an initial routine during which interrupts remain disabled, and a subsequent unprotected routine. The protected routine reads a character from the UART data register and places it in a temporary holding buffer, after which interrupts are enabled. This is because the input editing and echoing function is enabled upon receipt of each character, and this can be a complex procedure in some cases. Depending on the input mode, the character may be tested against an assortment of special characters, and this may result in flow control action or change of input mode. Characters which terminate packets or command lines require that pointers be updated and flags set for other routines. Input editing characters cause immediate update of the input buffer. Finally, an echo routine is called, which places characters in an echo output buffer. Input characters are not echoed directly, since adequate terminal support sometimes requires that more than one character be echoed for each character input. In particular, automatic line feed after carriage return is a common requirement, and some hard-copy devices require many null characters following a carriage return. If the input buffer becomes nearly full, a request may be made to the output routine to transmit an XOFF character. Alternatively, a routine to produce the appropriate hardware flow control signals to the RS-232 interface may be called.

Special input service routines are used when the program operates in "transparent mode." Since there are no special characters, the input character is simply placed in the buffer. A timer may be started, and a flag will be set if a maximum packet count has been reached.

The UART output interrupt is disabled whenever there is no data to be sent, and any routine which requests output enables the interrupt. Any special treatment of characters, such as conversion to upper case or adding line feeds or nulls, is done by the routine which fills the output buffer, which also maintains a screen-width counter and inserts extra carriage returns as necessary. The output service routine checks for tasks to be performed in the following priority and performs the first task found.

1. Request to transmit XOFF character
2. Request to transmit XON character
3. Transmit characters in echo buffer
4. Transmit characters in output buffer

If all tasks are exhausted, the routine disables the output interrupt. This is not necessary, but as long as this interrupt is enabled, the UART generates an interrupt at regular intervals.

Timer Interrupts

The basic function of the timer interrupt is to update the software clocks. The interrupt is set to occur at 10 ms intervals, which provides good resolution for timing associated with the radio interface. For longer times, a "slow" clock is updated at one-second intervals. An additional. 10 ms clock is used as a pseudo-random number generator for determining packet retry times.

In addition to the 'basic clock function, several tasks are performed under timer interrupt control. If a CW ID is in progress, the Morse Code routine is invoked every 60 ms to toggle the tone on or off as necessary. Otherwise, a variety of tasks to be performed after time lapses are checked . Packet transmissions are begun following an appropriate interval following detection of a carrier drop, and the ID is sent at regular intervals. If a CW ID is commanded manually, it is begun in this routine. A special set of routines is entered when the program runs in "transparent

mode," and the appropriate routine is selected by reference to a status table. These routines mark packets for sending upon timeout of clocks which are started on character input, and set up guard times for the escape sequence for exiting this mode.

HDLC Interrupts

The WD-1933 HDLC controller generates interrupts for seven different conditions. Since any combination of interrupt conditions can be present, and most conditions are cleared by reading the status of the chip, all possibilities must be considered at every interrupt. The interrupt conditions are requests for service of input or output registers (DRQI or DRQO), transmission or receipt of end of message, with or without errors (XEOM or REOM), and change of state of carrier detect (DSC).

Transmit interrupts, DRQO, XEOM-ok, and XEOM-err, are handled according to a state table which indicates the progress of the packet in transmission. The appropriate action, as determined by the state table, is taken if any of these interrupts is detected, without reference to which condition was indicated. The only transmit error possible is under-run, or failure to service a DRQO. This condition should never occur in normal operation. Following complete transmission of a packet and while final flags are being sent, the routine checks for further packets to be transmitted before the transmitter is unkeyed. The CW ID may also be started from this routine.

Receive interrupts, DRQI, REOM-ok, and REOM-err, are handled according to the condition indicated by the chip status. In the event that more than one condition is present, a DRQI is assumed to precede an REOM, and the input register is read before closing the packet. If both REOM-ok and REOM-err are present, the error condition is disregarded. Causes of the error condition are aborted frame, invalid frame-check sequence, and over-run (failure to service DRQI). The cause of the error is not investigated and any partially received frame is cancelled. Another possible source of error is insufficient room in the input buffer for the incoming packet. If the input buffer becomes full, a flag is set and the routine continues to read incoming characters as they appear, but an REOM-ok is treated as if it were an REOM-err.

The DSC interrupt does not affect either transmit or receive operation. The carrier detect input of the HDLC controller is the demodulator lock-detect and is used to recognize a busy channel. The service routine maintains software flags which reflect the carrier detect state, and if the carrier is found to have dropped, a timer is started which indicates when the next transmission may be started.

Nonvolatile RAM Interface

A number of user-settable parameters are stored in a semi-permanent state in the Xicor NOVRAM. To make maximum use of the 32 bytes of storage, the data is encoded in a compact form. In order to be useful to the program, it must be translated. The information stored includes terminal attributes such as baud rate and parity; radio-link attributes such as baud rate, packet length, and transmitter keyup time; display features such as case conversion, echo mode, auto line feed, screen width, and nulls required. Special command characters for flow control, exit to command mode, and editing features are also stored, along with the station call sign. These parameters are changed by commands to HLR, which maintains the compressed copy of the parameters by calling LLR whenever a parameter is changed. This copy is overlaid on the nonvolatile storage as volatile RAM data, and becomes permanent when the "STORE" line is toggled. The nonvolatile RAM is controlled through the parallel I/O port of the 6522 VIA.

Calibration Routines

The hardware design of the TAPR TPJC provides for on-board calibration routines. Jumpers are connected which allow the VIA Timer 2 to count the modulator or demodulator frequency to be calibrated. The VIA timers are reconfigured so that Timer 1 acts as a free-running count-down timer, counting at the 921.6 kHz system clock rate, and Timer 2 generates an interrupt after two cycles of the tone frequency being calibrated. By starting the timers simultaneously, Timer 1 can be used to count clock pulses for the duration of two periods of the frequency being calibrated. Two of the LED indicators are controlled in parallel with the microphone audio and HDLC reset lines, and are used in this routine as visual indicators of frequency deviation from the desired values. The timer routines in the interrupt dispatch table are replaced with special calibration service routines.

High-level Interface

Most of the entry points provided for HLR are concerned with buffer management. The I/O functions performed through these routines are: read a string from terminal or packet input buffer to a specified location; write a string from a specified location to terminal or packet output buffer; return character count for an outgoing packet in the terminal input buffer; update terminal input buffer; and return space-available count for output buffers. Other functions provided are: update nonvolatile RAM, temporary or permanent mode; enter calibration routine; force CW ID; and perform a soft reset. A special routine is called by HLR to notify LLR of a change of mode from "command mode" to "converse mode" or "transparent mode." All routines are called with the addresses of the arguments in processor registers D, X, Y, and U as needed.

Low-level Debug Program

A debugging facility is provided in the LLR, primarily as a software development tool. This program is invoked by a special user-settable character. In order to preserve as much information as possible about the system prior to entering the debugger, the active terminal input and output buffers are "frozen" upon entry by swapping the buffer pointers with a set of pointers used exclusively in the debugging program. All terminal input thenceforth is interpreted as commands to the debugger. These commands permit examina-

tion or modification of any addressable location, modification of the processor registers as they were upon entry to the program, or transfer of execution to any location. This allows test routines to be stored in RAM and executed. The user also has direct access to I/O addresses, and can observe the contents of I/O buffers without interfering with them.