AMT Mini Project - RS232 Communication Documentation

What you should really care about

Dependency: amt.h, uart.c
Header file: RS232.h
C file: RS232.c

*Modify the global variable `char* ports[8]` in RS232.h to select which port to display
*Look under section **Header file: RS232.h** for detailed explanations on all global variables and functions used in this library

Functions

`void RS232_init(void);` Call this to initialise the RS232 library. ui_init() will be called in this function

`void RS232_update(void);` This function will be called in an interrupt to do a port information update on the terminal screen, it should be called with High Interrupt in this format, the code below tries to do an update every 1ms and then resets the variable counter

```
    if(counter >= 1){

        if(!sending)
            RS232_update();

        counter = 0;
    }
```

`void outputLog(char log[]);` Use this to output log messages to the terminal.

Header file: RS232.h

Global Variables

```
char* ports[8] = {0, &PORTB, &PORTC, &PORTD, 0, &PORTF, &PORH, &PORTJ};
```

- The char pointer array is used by the RS232 library determined which PORT needs to be displayed. The library can display up to 8 ports hence the array size. Modify it to select which ports will be display on the UART display, assign a value of 0 at the slot that you don't want to display
- The library will display the information of the ports in the following fashion:

```
          [SCREEN]
======================
| ports[0]  ports[4] |  LOG   |
| ports[1]  ports[5] |        |
| ports[2]  ports[6] |        |
| ports[3]  ports[7] |        |
======================
```

```
char cache[8] = {};
```

- An array that is used to store values of ports since the last check, it will be compared against during the next check to determined if update is needed

```
char cache_t[8] = {};
```

- An array that is used to store TRIS values of ports since the last check, it will be compared against during the next check to determined if update is needed

```
int logStatus
```

- This global variable is needed for the outputLog() function. It is used to keep track of the number of logs being outputted to the HyperTerminal. It is initialised to '0' when ui_init() is called.

  This variable allows the terminal to output the correct log, which you see it being displayed as "[1]: <Your log>". As logStatus increases, the number ("[1]: ") increases.

```
int previousLog
```

- previousLog is used to keep track of how many rows has been used. This variable will increase as more logs are added to the HyperTerminal. When writing the first log, there is no previous log and so previousLog is initialised to '1'.

  Why '1'? Because the first row is used for the title "LOG". This is needed for the outputLog() function to monitor how many rows are available for printing. More details on how it's used will be mentioned in the function explanation of outputLog().

```
int sending
```

- This global variable is used to ensure that the uart will finish to receive the full string of message or command before handling the interrupt routine. It will only skip the RS232_update function when sending == '1'. Other interrupt routines would not be affected.

Function Prototypes

```
void uart_send_str(char str[]);
```

- A function that accept a string (char[]) and send them via RS232 communication character by character until it reaches '\0'

```
void set_position(int col, int line);
```

- A function that accept integer value of col and line and set the cursor to that position on the screen

```
void print_port(char* address, int col, int line);
```

- A function that is used internally to print port information on the predefined position

```
void RS232_update(void);
```

- A function will do a comparison with cache[8] and cache_t[8] and update the information on the screen if there is any changes

```
void RS232_init(void);
```

- A function that initialise the display on the screen

```
void uart_sendCmd(char cmd[]);
```

- Sends a command sequence to the uart for the HyperTerminal to execute. For references of all possible commands, look up for the ANSI escape sequences which the HyperTerminal follows.

```
void ui_init(void);
```

- Used to initialise the global variables needed for the outputLog() function and calls the setLayout() function. It has been called in the RS232_init() function

```
void setDarkMode(void);
```

- Calling this function would set the background to black and the text color to white, giving the terminal a dark look.

```
void uart_send(char str[]);
```

- Similar to the uart_send_str() function, it sends a string to the uart but instead of checking for the terminating character, it uses the length of the character array.

```
void setLayout(void);
```

- This is used as part of the ui_init() function. This function prints out the vertical lines to separate the PORT status and log output, as well as print out the "LOG" title.

```
void clearLog(void);
```

- This will clear the log section line by line.

```
void outputLog(char log[]);
```

- This function handles the output to the log. It has one parameter, a character array. You can either directly give it the message you want, or create a variable first before giving it to this function. (i.e., outputLog("Washing Machine turned on.")). It first checks how many rows the message would require.

  Using the previousLog global variable, it calculates the total amount of rows it would take and if it is more than 23 rows, it would clear the whole log section and output the log at the starting position. If not, it will print the message with 1 row spacing between log messages.

  The 'sending' global variable will be initialised to '1' at the start of the function and set to '0' at the end.