LIBRERIA UARTPRINTF

Introduzione

Questa libreria consente di trasmettere e ricevere su seriale stringhe grazie al semplice utilizzo della funzione `uartprintf()`, per ottenere una stampa del tutto simile alla classica printf del C, e dell'implementazione di una callback per la ricezione.

Essa fa uso di un buffer che permette di gestire anche chiamate asincrone alla funzione stessa senza perdita di dati.

Alla prima chiamata della funzione la libreria configura in modo automatico il modulo seriale `USART2` abilitandolo sia alla trasmissione che alla ricezione non bloccanti (facendo uso degli interrupt) e mettendo a disposizione due routine di callback per due diverse modalità di ricezione:

- singolo carattere;
- stringa terminata da sequenze di terminazione di linea (`\r`, `\n`, `\r\n`, `\n\r`).

Inizializzazione

Sono previsti 2 tipi di inizializzazione:

- richiamando la funzione di inizializzazione `uartprintf_init(uint32_t baudRate, uint32_t wordLength, uint32_t stopBits, uint32_t parity)`, alla quale si specificano i parametri di configurazione della porta seriale;
- al primo richiamo della funzione `uartprintf()`. In questo caso la porta seriale verrà inizializzata con valori di default:
  - baudRate: 115200
  - wordLength: 8 bit
  - stopBits: 1
  - parity: pari

In questa fase, oltre a configurare la seriale e i relativi pin (come previsto dalle librerie HAL), vengono inizializzate anche le variabili di stato:

- `_uartprintf_count_wr = 0`: indice del buffer circolare. Utilizzato per inserire le stringhe nel buffer;
- `_uartprintf_count_rd = 0`: indice del buffer circolare. Utilizzato per prelevare le stringhe dal buffer;
- `_uartprintf_toFree = NULL`: puntatore ad una stringa da deallocare al termine della trasmissione;
- `_uartprintf_busy = 0`: flag che indica se una trasmissione è in corso;
- `_uartprintf_rxLine_len = 0`: lunghezza della stringa ricevuta;

- `_uartprintf_initialized = 1`: flag che indica se la seriale è già stata inizializzata.

Trasmissione

La trasmissione avviene tramite la chiamata di `uartprintf()`:

```
#define uartprintf(...) \
{ \
     char* output = malloc(_UARTPRINTF_MAX_STR); \
     if(output!=NULL) { \
          sprintf(output,__VA_ARGS__); \
          _uartprintf_print(output); \
     } \
}
```

Essa può ricevere un numero arbitrario di parametri che verranno passati alla `sprintf`, la quale provvede a generare una stringa formattata sulla base degli argomenti passati. La stringa così ottenuta viene passata alla funzione `_uartprintf_print(output)`.

```
void _uartprintf_print(char* outputdata) {
     if(!_uartprintf_initialized) {
          _uartprintf_default_init();
     }

     int len = strlen(outputdata);
     char* outputdataNorm = malloc(len+1);
     sprintf(outputdataNorm, "%s", outputdata);
     free(outputdata);
     if(_uartprintf_busy) {
          _uartprintf_outputBuffer[_uartprintf_count_wr] =
outputdataNorm;
          _uartprintf_count_wr++;
          _uartprintf_count_wr = _uartprintf_count_wr %
_UARTPRINTF_MAX_BUFF;
          _uartprintf_outputBuffer[_uartprintf_count_wr] =
NULL;
     } else {
          _uartprintf_busy = 1;
          HAL_UART_Transmit_IT(&_uartprintf_UartHandle,
(uint8_t *)outputdataNorm, len);
          _uartprintf_toFree = outputdataNorm;
     }
}
```

Per prima cosa viene ridimensionata la stringa, che inizialmente era allocata con dimensione di 512 byte.

Se è in corso una trasmissione, procedo a salvare la stringa nel buffer nella posizione indicata da `_uartprintf_count_wr`. Dopodiché si calcola il nuovo valore dell'indice (se ne incrementa il valore e se maggiore o uguale della massima dimensione del buffer viene azzerato). Nella nuova posizione si inserisce un puntatore nullo per indicare la fine del buffer.

Se invece non è in corso nessuna trasmissione, si richiama la funzione fornita dalle librerie HAL per la stampa su seriale non bloccante. Viene salvato inoltre il puntatore della stringa da deallocare a fine trasmissione.

```
void _uartprintf_TxCompleted() {
     if(_uartprintf_toFree!=NULL) {
          free(_uartprintf_toFree);
          _uartprintf_toFree = NULL;
     }
     if(_uartprintf_busy) {

     if(_uartprintf_outputBuffer[_uartprintf_count_rd]!=NULL) {
               char* outputdata =
_uartprintf_outputBuffer[_uartprintf_count_rd];
               HAL_UART_Transmit_IT(&_uartprintf_UartHandle,
(uint8_t *)outputdata, strlen(outputdata));
               _uartprintf_toFree = outputdata;
               _uartprintf_count_rd++;
               _uartprintf_count_rd = _uartprintf_count_rd %
_UARTPRINTF_MAX_BUFF;
          } else {
               _uartprintf_busy = 0;
          }
     }
}
```

Questa funzione viene richiamata al verificarsi dell'interrupt che indica la fine della trasmissione.

Per prima cosa si liberano le risorse relative alla stampa appena completata. Si verifica poi la presenza di eventuali stringhe da trasmettere e nel caso si procede alla trasmissione. Viene inoltre salvato il puntatore della stringa da deallocare successivamente e si incrementa l'indice della prossima stringa da prelevare (verificando che non sia arrivato a fine buffer).

Nel caso non ci sia nulla da trasmettere resetto il flag `_uartprintf_busy`.

Ricezione

Per quanto riguarda la ricezione, c'è da sottolineare il fatto che nella funzione `uartprintf_init` viene richiamata la funzione fornita dalle librerie HAL per la ricezione non bloccante. Questo è necessario per predisporre il sistema affinché alla ricezione di un dato venga scatenato l'interrupt.

```
void _uartprintf_RxCompleted() {
    uartprintf_charreceived(_uartprintf_rxByte);

    // uartprintf_linereceived
    char isLineBreak = (_uartprintf_rxByte=='\r' ||
_uartprintf_rxByte=='\n');
    char cond = (isLineBreak && _uartprintf_rxLine_len>0);
    if(cond || _uartprintf_rxLine_len+1 >=
_UARTPRINTF_RXLINE_MAX_STR) {
        uartprintf_linereceived(_uartprintf_rxLineString);
        _uartprintf_rxLine_len = 0;
    } else {
        if(!isLineBreak) {
            _uartprintf_rxLineString[_uartprintf_rxLine_len]
= _uartprintf_rxByte;
            _uartprintf_rxLine_len++;
            _uartprintf_rxLineString[_uartprintf_rxLine_len]
= '\0';
        }
    }

    HAL_UART_Receive_IT(&_uartprintf_UartHandle, (uint8_t
*)&_uartprintf_rxByte, 1);
}
```

Questa funzione viene richiamata al verificarsi dell'interrupt, per ogni carattere ricevuto. Essa provvede a manipolare i dati ricevuti per poi fornirli a due funzioni implementabili a discrezione dell'utente.

A una di queste (`uartprintf_charreceived`) vengono passati i singoli caratteri ricevuti, mentre l'altra (`uartprintf_linereceived`) viene richiamata una volta ricevuti i caratteri terminatori di linea oppure quando viene raggiunta la dimensione massima.

Per quanto riguarda la chiamata di quest'ultima se il carattere ricevuto non è di terminazione di linea viene accodato nel buffer `_uartprintf_rxLineString`. In caso contrario si passa la stringa privata dei caratteri di terminazione di linea alla funzione `uartprintf_linereceived`.

In entrambi i casi viene richiamata la funzione `HAL_UART_Receive_IT` per permettere di ricevere i successivi caratteri.