in include/common.h

```
#ifdef DEBUG
#define _DEBUG  1
#else
#define _DEBUG  0
#endif

/*
 * Output a debug text when condition "cond" is met. The "cond" should be
 * computed by a preprocessor in the best case, allowing for the best
 * optimization.
 */
#define debug_cond(cond, fmt, args...)          \
        do {                                    \
                if (cond)                       \
                        printf(fmt, ##args);    \
        } while (0)

#define debug(fmt, args...)                     \
        debug_cond(_DEBUG, fmt, ##args)
```

diff --git a/include/common.h b/include/common.h

index 5721cdc..5b34f64 100644

--- a/include/common.h

+++ b/include/common.h

@@ -99,6 +99,9 @@ typedef volatile unsigned char vu_char;

#include <flash.h>

#include <image.h>

+// enable u-boot debug output

+#define DEBUG

+

#ifdef DEBUG

#define _DEBUG 1

#else

在u-boot中使用printf()是无条件输出，而debug()则是在定义DEBUG macro的情况下才输出。

in common/console.c

```
1.  int printf(const char *fmt, ...)
2.  {
3.          va_list args;
4.          uint i;
5.          char printbuffer[CONFIG_SYS_PBSIZE];            ①
6.
7.  #if !defined(CONFIG_SANDBOX) && !defined(CONFIG_PRE_CONSOLE_BUFFER)
8.          if (!gd->have_console)
9.                  return 0;
10. #endif
11.
12.         va_start(args, fmt);
13.
14.         /* For this to work, printbuffer must be larger than
15.          * anything we ever want to print.
16.          */
17.         i = vscnprintf(printbuffer, sizeof(printbuffer), fmt, args);
18.         va_end(args);
19.
20.         /* Print the string */
21.         puts(printbuffer);                                          ②
22.         return i;
23. }
```

①

输出字符串的长度受限于CONFIG_SYS_PBSIZE，且space分配在stack上。

in include/configs/pegmatite.h

```
#define CONFIG_SYS_CBSIZE       512             /* Console I/O Buffer Size */

......

#define CONFIG_SYS_PBSIZE       \
                        (CONFIG_SYS_CBSIZE + sizeof(CONFIG_SYS_PROMPT) + 16)
```

②

in common/console.c

```
void puts(const char *s)
{
#ifdef CONFIG_SANDBOX
        if (!gd) {
                os_puts(s);
                return;
        }
#endif

#ifdef CONFIG_SILENT_CONSOLE
        if (gd->flags & GD_FLG_SILENT)
                return;
#endif

#ifdef CONFIG_DISABLE_CONSOLE
        if (gd->flags & GD_FLG_DISABLE_CONSOLE)
                return;
#endif

        if (!gd->have_console)
                return pre_console_puts(s);

        if (gd->flags & GD_FLG_DEVINIT) {
                /* Send to the standard output */
                fputs(stdout, s);
        } else {
                /* Send directly to the handler */
                serial_puts(s);
        }
}
```

in drivers/serial/serial.c

```
/**
 * serial_puts() - Output string via currently selected serial port
 * @s:  Zero-terminated string to be output from the serial port.
 *
 * This function outputs a zero-terminated string via currently
 * selected serial port. This function behaves as an accelerator
 * in case the hardware can queue multiple characters for transfer.
 * The whole string that is to be output is available to the function
 * implementing the hardware manipulation. Transmitting the whole
 * string may take some time, thus this function may block for some
 * amount of time. This function uses the get_current() call to
 * determine which port is selected.
 */
void serial_puts(const char *s)
{
        get_current()->puts(s);
}
```