文档编号：AN2032

上海东软载波微电子有限公司

用户手册

ES-DAP-Viewer

修订历史

版本	修改日期	更改概要
V1.0	2021-08-28	初版发布
V1.1	2022-06-22	1. SWDPrint 支持浮点数及多通道模式； 2. 全局变量监视支持浮点数、数组及结构体； 3. elf 文件默认路径使用上次加载的路径； 4. SWDPrint 和全局变量监视支持读取速度可调；

地　址：中国上海市徐汇区古美路 1515 号凤凰园 12 号楼 3 楼
E-mail：support@essemi.com
电　话：+86-21-60910333
传　真：+86-21-60914991
网　址：http://www.essemi.com/

目 录

内容目录

第 1 章 简介 4
第 2 章 UART-Print 模式 5
2. 1 PC 界面操作说明 5
2. 2 通信协议和程序示例 6
第 3 章 SWD-Print 模式 10
3. 1 使用说明 10
3. 2 程序示例 11
第 4 章 全局变量监视模式 13
4. 1 使用说明 13
4. 2 程序示例 14

图目录

图 1-1 ES-DAP-Viewer 使用示意图 4
图 1-2 ES-DAP-Viewer 启动图标 4
图 2-1 UART-Print 波形显示界面 5
图 3-1 SWD-Print 波形显示界面 10
图 4-1 全局变量监视调试界面 13

表目录

表 2-1 UART-Print 通信协议 6

第1章 简介

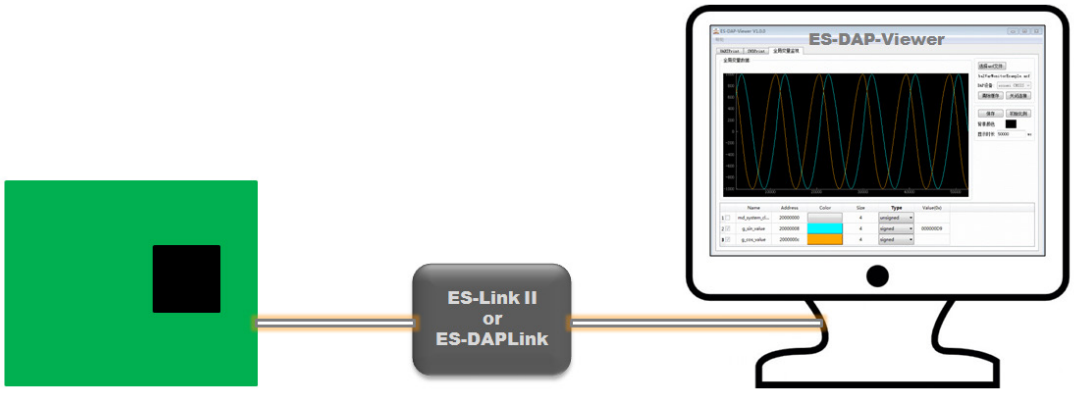

图 1-1 ES-DAP-Viewer 使用示意图

MCU 向调试终端输出信息的方法有很多。ES-DAP-Viewer 是一个更炫更酷、可以图形化显示数据的调试软件，它可以在目标 MCU 运行时，实时分析数据并图形化显示的 PC 端软件。用户在简单的将目标 ES32 芯片连接到 ES-Link II 调试器或 ES-DAPLink 调试器后，打开上位机软件 ES-DAP-Viewer 经过简单地参数配置后就可以像示波器一样显示多个变量的值。它支持通过如下三种模式获取数据：

1. **UART-Print 模式**：用户通过串口将数据按照固定格式发送到 PC 端，在上位机软件用户界面配置通信参数后将收到的数据可视化。

2. **SWD-Print 模式**：这种模式不会占用 UART 引脚， ES-DAP-Viewer 通过 SWD 调试接口获取数据，用户只需要使用 essemi_swd_printf 等函数将数据写入目标缓冲区即可。

3. **全局变量监视模式**：这种模式也是通过 SWD 调试接口获取数据，通过读取 elf 文件，允许选择一定数量的全局变量可视化。

双击安装文件 ES-DAP-Viewer.msi，同意安装协议，选择安装路径，点击确定后等待软件安装，安装完成后即可在桌面看到 ES-DAP-Viewer 快捷方式。

图 1-2 ES-DAP-Viewer 启动图标

注 1：本应用手册中涉及的例程可在 ES32_SDK/Projects/Book2_Example/ES-DAP-Viewer 下查看源码
注 2：“SWD-Print 模式”和“全局变量监视模式”目前无法和 IDE 调试同时使用。

第2章 UART-Print 模式

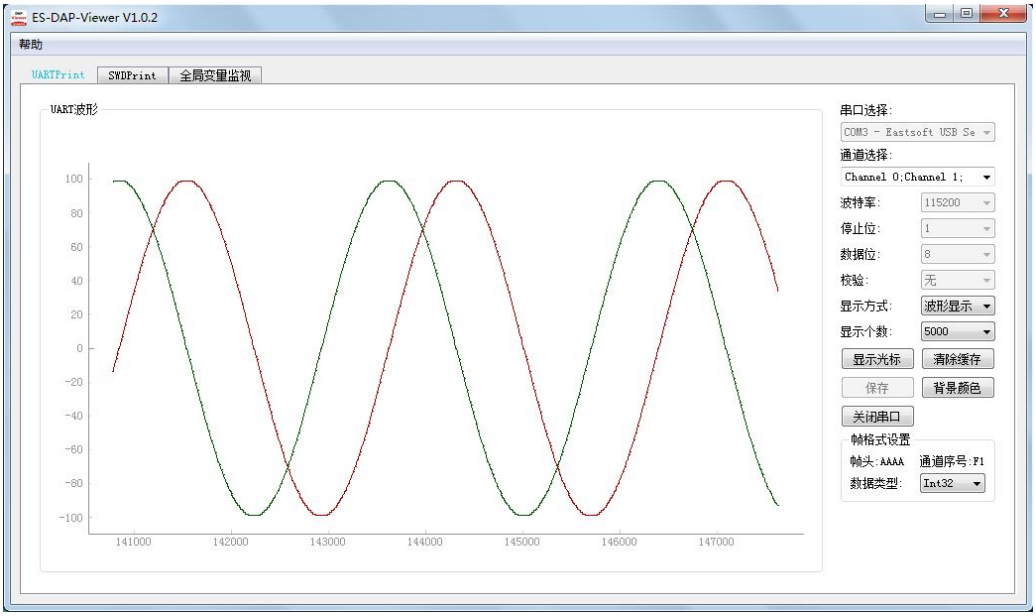

图 2-1 UART-Print 波形显示界面

左边为绘图窗口；右边为配置窗口，涵盖了串口选择、通信参数设置、数据显示方式、显示个数等，右下方为帧格式设置窗口，用于设置数据解析类型。

2.1 PC 界面操作说明

UART-Print 绘图调试的大致流程如下所述：

1. 打开 ES-DAP-Viewer 后选择 `UART-Print` 选项卡，进入 UART-Print 界面。

2. 在配置窗口中选择串口，并根据 MCU 固件中采用的 UART 通信参数配置波特率，停止位，数据位和奇偶校验位。

3. 根据 MCU 固件中发送的帧中的数据段，通过点击下拉框“通道选择”可勾选需要显示的具体通道。需要注意，该功能仅在“波形显示”模式下有效。

4. 根据具体的数据类型，在帧格式设置窗口中进行相应设置，可选 Uint8、Int8、Uint16、Int16 以及 Uint32 和 Int32。

5. 点击“显示方式”下拉框，选择“波形显示”或“数据显示”来决定数据的可视化方式，点击“打开串口”便可在绘图窗口或数据显示窗口看到当前的数据。此外，当选择“数据显示”时，在数据显示窗口中显示的是用户程序发送的原始数据（不包含帧头、功能码、数据长度及校验和），每行显示一帧数据。

6. 点击“保存”可保存已缓存到缓冲区的数据。点击“清除缓存”将清空缓冲区的所有数据，同时，ES-DAP-Viewer 绘图窗口或数据显示窗口也将被清空。

7. 点击“背景颜色”可在弹出的对话框中配置当前视图的背景色。需要注意的是，该按钮仅在“波形显示”模式下有效。

8. 点击“显示个数”，可调整“波形显示”模式下绘图控件所采样的数据个数，因此，可以通过修改该值实现横坐标的缩放。点击“显示光标”按钮，移动鼠标到绘图窗口后，鼠标箭头处将出现光标，同时绘图窗口右上角将出现当前鼠标箭头所处位置的坐标值。同样，这两个按钮仅在“波形显示”模式下有效。

9. 在绘图窗口中上下滑动鼠标滚轮可实现图形的缩放，当 ES-DAP-Viewer 在持续绘图时，滑动滚轮只能实现纵坐标的缩放，而当点击“关闭串口”绘图停止时，滑动滚轮可实现横纵坐标的缩放，绘图窗口缩放后，绘图窗口左下角会出现图标“A”，点击后将恢复默认比例。

2.2 通信协议和程序示例

UART-Print 使用的通信协议如下表所示：

帧头 (2 字节)	功能码 (1 字节)	数据长度 (1 字节)	用户数据 (若干字节)	累加校验和 (1 字节)	备注
0xAAAA	0xF1	data_len	user_data	check_sum	绘图调试功能

表 2-1 UART-Print 通信协议

关于 ES-DAP-Viewer 的 UART-Print 通信协议，详细说明如下：

1. UART-Print 的数据帧由帧头、功能码、数据长度、用户数据以及累加校验和组成，其中，帧头固定为两字节大小的 0xAAAA，功能码固定为 0xF1。

2. data_len 表示该数据帧内包含的用户数据的字节总长度，不包括帧头、功能码、数据长度和累加校验和。

3. check_sum 表示从该数据帧第一个字节开始，也就是帧头开始，到用户数据最后一个字节的累加校验和，高位舍去，只保留低八位。

4. ES-DAP-Viewer 上位机按照上述的协议解析收到的数据并进行校验以显示可靠的数据，这意味着用户在使用 UART-Print 功能时，必须按照该该协议进行发送数据，否则上位机会因校验不通过而丢弃数据。

5. UART-Print 最多支持 10 个通道，每个通道均支持显示 Uint8、Int8、Uint16、Int16 以及 Uint32 和 Int32 格式的数据。

使用如下程序可实现图 2-1 所示效果。

```
1.  int main()
2.  {
3.      uint32_t i, j, length;
4.      int32_t data2send[2];
5.      md_uart_init_t g_uart_init;
6.
7.      /* 配置系统时钟 */
8.      md_cmu_pll1_config(MD_CMU_PLL1_INPUT_HOSC_3, MD_CMU_PLL1_OUTPUT_72M);
9.      md_cmu_clock_config(MD_CMU_CLOCK_PLL1, 72000000);
```

```
10.     /* 初始化 SysTick 中断 */
11.     md_init_1ms_tick();
12.
13.     /* 使能所有外设时钟 */
14.     SYSCFG_UNLOCK();
15.     md_cmu_enable_perh_all();
16.     SYSCFG_LOCK();
17.
18.     /* 初始化 UART 引脚 */
19.     uart_pin_init();
20.
21.     /* 配置 UART 通信参数 */
22.     memset(&g_uart_init, 0x0, sizeof(md_uart_init_t));
23.     g_uart_init.baud        = 115200;
24.     g_uart_init.word_length = MD_UART_WORD_LENGTH_8B;
25.     g_uart_init.stop_bits   = MD_UART_STOP_BITS_1;
26.     g_uart_init.parity      = MD_UART_PARITY_NONE;
27.     g_uart_init.fctl        = MD_UART_FLOW_CTL_DISABLE;
28.     md_uart_init(UART0, &g_uart_init);
29.
30.     i = 0;
31.
32.     while (1)
33.     {
34.         data2send[0] = (int32_t)(100 * sin((2 * 3.1415926 / 2047) * i)); /* 正弦数据 */
35.         data2send[1] = (int32_t)(100 * cos((2 * 3.1415926 / 2047) * i)); /* 余弦数据 */
36.
37.         /* 按照 ES-DAP-Viewer 的 UARTPrint 传输协议要求，格式化传输数据，获取待发送数据的总
38.            长度格式化后的数据暂存在全局变量 g_tx_buf[20] 里，注意数组长度不要越界 */
39.         length = data_initialize(g_tx_buf, (uint8_t *)data2send, 2 * 4);
40.
41.         /* 发送格式化后的数据 */
42.         for (j = 0; j < length; j++)
43.         {
44.             md_uart_set_send_data8(UART0, g_tx_buf[j]);
45.             while (RESET == md_uart_is_active_it_tbc(UART0));
46.             md_uart_clear_it_tbc(UART0);
47.         }
48.
49.         i++;
50.     }
51. }
52.
53. /* Private Function ------------------------------------------------------- */
```

```
54. /**
55.   * @brief  Initializate UART pin
56.   * @param  None
57.   * @retval None
58.   */
59. void uart_pin_init(void)
60. {
61.     md_gpio_init_t x;
62.
63.     /* Initialize tx pin: PB10 */
64.     x.mode  = MD_GPIO_MODE_OUTPUT;
65.     x.odos  = MD_GPIO_PUSH_PULL;
66.     x.pupd  = MD_GPIO_PUSH_UP;
67.     x.podrv = MD_GPIO_OUT_DRIVE_1;
68.     x.nodrv = MD_GPIO_OUT_DRIVE_1;
69.     x.flt   = MD_GPIO_FILTER_DISABLE;
70.     x.type  = MD_GPIO_TYPE_TTL;
71.     x.func  = MD_GPIO_FUNC_3;
72.     md_gpio_init(GPIOB, MD_GPIO_PIN_10, &x);
73.
74.     /* Initialize rx pin: PB11 */
75.     x.mode  = MD_GPIO_MODE_INPUT;
76.     x.odos  = MD_GPIO_PUSH_PULL;
77.     x.pupd  = MD_GPIO_PUSH_UP;
78.     x.podrv = MD_GPIO_OUT_DRIVE_1;
79.     x.nodrv = MD_GPIO_OUT_DRIVE_1;
80.     x.flt   = MD_GPIO_FILTER_DISABLE;
81.     x.type  = MD_GPIO_TYPE_TTL;
82.     x.func  = MD_GPIO_FUNC_3;
83.     md_gpio_init(GPIOB, MD_GPIO_PIN_11, &x);
84.
85.     return;
86. }
87.
88. /**
89.   * @brief  格式化用户数据，使之符合传输协议
90.   * @param  srcdata-保存用户数据的首地址；
91.   *         desdata-保存处理好的数据的内存首地址(其内存大小至少大于前者 5 字节)；
92.   *         len-用户数据长度(单位:字节)
93.   * @retval 返回数据帧的总长度
94.   */
95. uint32_t data_initialize(uint8_t *desdata, uint8_t *srcdata, uint32_t len)
96. {
97.     uint32_t i;
```

```
98.
99.     /* 帧头 */
100.    desdata[0] = 0xAA;
101.    desdata[1] = 0xAA;
102.
103.    /* 功能码 */
104.    desdata[2] = 0xF1;
105.
106.    /* 数据长度 */
107.    desdata[3] = len;
108.
109.    /* 用户数据 */
110.    memcpy(desdata + 4, srcdata, len);
111.
112.    /* 累加校验和 */
113.    desdata[len + 4] = 0U;
114.
115.    for (i = 0; i < len + 4; i++)
116.    {
117.        /* 累加校验和，高位舍去，只保留低 8 位 */
118.        desdata[len + 4] += desdata[i];
119.    }
120.
121.    return (len + 5);
122. }
```

第3章 SWD-Print模式

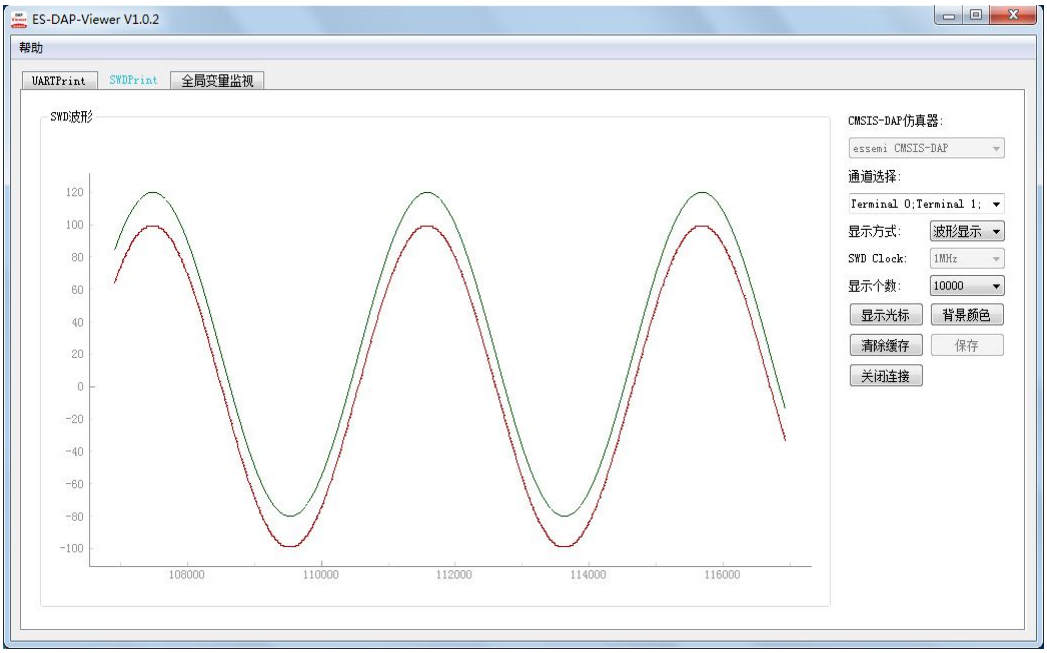

图 3-1 SWD-Print 波形显示界面

SWD-Print 由于不需要配置通信参数，因此其使用界面相比较于 UART-Print 更为简洁。

3.1 使用说明

使用 SWD-Print 时，需要注意以下几点：

1. ES-DAP-Viewer 中“SWD Clock”决定了调试器的 SWD 时钟频率，影响 ES-DAP-Viewer 上位机软件从目标缓冲区中获取数据的速度，“SWD Clock”的数值设定建议不低于 50KHz。

2. 当选用“波形显示”模式时，在勾选相应通道后，打印到不同的终端的数据将统一展示在同一绘图窗口中，而在“数据显示”模式下，勾选相应通道后，将出现对应的子标签页，不同终端的数据除了会打印到对应的子标签页中中，还会打印到名为“All Terminals”的子标签页中。

3. 无论使用“数据显示”还是“波形显示”，在点击“打开连接”按钮前，均需先对 MCU 进行复位操作。

4. 其他操作可参考 UART-Print。

用户程序可调用 ESSEMI_SWD_SetTerminal 选定目标终端，通过调用 essemi_swd_printf 将数据写入目标缓冲区。SWD-Print 功能的实现需要用到以下四个配置文件：

essemi_swd_print.h
essemi_swd_print_conf.h
essemi_swd_print.c
essemi_swd_print_printf.c

用户在工程中成功添加上述四个配置文件后，并在需要用到 SWD-Print 功能的文件中包含如

下头文件#include“essemi_swd_print.h”后，即可调用 essemi_swd_printf()函数将需要打印的数据放到 SWD 专属内存区域中，然后上位机到该内存区域取出数据，显示到绘图窗口。

配置文件放在 ES-DAP-Viewer 如下安装目录：...\configure\essemi_swd_print。

需要注意以下几点：

在使用 SWD-Print 功能时，必须先调用缓存区初始化函数 essemi_swd_configupbuffer 以及 essemi_swd_configdownbuffer 来初始化上行（MCU 端到 PC 端）和下行（PC 端到 MCU 端）缓冲区，其中，对于第一个参数 BufferIndex = 0 的时候，SWD 组件已为其配置了缓冲和默认大小，其默认配置是通过头文件 essemi_swd_print_conf.h 中的宏定义#define BUFFER_SIZE_UP(1024) 以及 #define BUFFER_SIZE_DOWN(16)实现的，因此，在使用缓冲区 0 时，配置比较简单，按照该格式即可：

```
1. essemi_swd_configupbuffer(0, "SWDUP", NULL, 0, ESSEMI_SWD_MODE_BLOCK_IF_FIFO_FULL);
2. essemi_swd_configupbuffer(0, "SWDDOWN", NULL, 0,
3.                                                  ESSEMI_SWD_MODE_BLOCK_IF_FIFO_FULL);
```

上述两个函数的最后一个参数可选择如下宏定义：

```
1. /* Skip(Default): 如果缓冲区不够存储要发送的数据，将放弃写入缓冲区. */
2. #define ESSEMI_SWD_MODE_NO_BLOCK_SKIP       (0)
3. /* Trim: 如果缓冲区不够存储这些数据，则将无法写入的部分丢弃. */
4. #define ESSEMI_SWD_MODE_NO_BLOCK_TRIM       (1)
5. /* Block: 如果缓冲区满，将阻塞等待有空间可用. */
6. #define ESSEMI_SWD_MODE_BLOCK_IF_FIFO_FULL (2)
```

使用 SWD-Print 时，必须要注意的一点是，SWD-Print 通过识别空格来进行数据分割，因此输出之间要有一个空格，即第 N 次输出时要在输出内容的最后额外加一个空格或在输出内容前加一个空格。

3.2 程序示例

使用如下程序可产生图 3-1 所示的效果。

```
1.  int main(void)
2.  {
3.      uint32_t i = 0;
4.      int32_t data2send[2];
5.      float   f_data2send;
6.
7.      /* 配置系统时钟 */
8.      md_cmu_pll1_config(MD_CMU_PLL1_INPUT_HOSC_3, MD_CMU_PLL1_OUTPUT_96M);
9.      md_cmu_clock_config(MD_CMU_CLOCK_PLL1, 96000000);
10.
11.     /* 使能外设时钟 */
12.     SYSCFG_UNLOCK();
13.     md_cmu_enable_perh_all();
14.     SYSCFG_LOCK();
15.
16.     /* 初始化缓冲区 */
```

```
17.     essemi_swd_configupbuffer(0, "SWDUP", NULL, 0,
18.                                                 ESSEMI_SWD_MODE_BLOCK_IF_FIFO_FULL);
19.     essemi_swd_configdownbuffer(0, "SWDDOWN", NULL, 0,
20.                                                 ESSEMI_SWD_MODE_BLOCK_IF_FIFO_FULL);
21.
22. #if (PRINT_STR)
23.     ESSEMI_SWD_SetTerminal(0);
24.     essemi_swd_printf(0, "%s", "SWD print demo\r\n");
25. #endif
26.
27.     while (1)
28.     {
29.         /* 数组赋值 */
30.         data2send[0] = (int32_t)(100 * sin((2 * 3.1415926 / 4095) * i));
31.         f_data2send = 100 * sin((2 * 3.1415926 / 4095) * i) + 20;
32.
33.         /* 设置终端号 */
34.         ESSEMI_SWD_SetTerminal(0);
35.         /* 向 SWD 端口缓冲区输出数据，注意输出的第一个字符必须为空格，并根据数据位宽将数据格
36.            式输出 */
37.         essemi_swd_printf(0, " %d", data2send[0]);
38.         ESSEMI_SWD_SetTerminal(1);
39.         essemi_swd_printf(0, " %8.4f", f_data2send);
40.
41.         i++;
42.     }
43. }
```

第4章　全局变量监视模式

图 4-1　全局变量监视调试界面

4.1　使用说明

全局变量监视功能与前述的两个功能存在的最大区别在于，使用全局变量监视时，必须提前加载待调试程序的 elf 文件，并勾选所需监视的全局变量。

全局变量监视功能可参考下述步骤：

1. 打开 ES-DAP-Viewer 后选择全局变量监视选项卡，进入全局变量监视界面。

2. 点击“选择 elf 文件”按钮，载入由编译器生成的 elf 文件（如 keil 生成的.axf 文件，iar 生成的.out 文件等等）。

3. ES-DAP-Viewer 在完成解析 elf 文件后，将弹出一个对话框，并在该对话框中列出解析到的各文件中的全局变量，勾选想要监视的全局变量后，点击“OK”按钮推出对话框，此时在全局变量监视标签页的最下方的“参数”窗口可观察到所勾选的全局变量，通过点击右下角的“全局变量”按钮可再次弹出对话框来勾选或取消勾选解析到的全局变量。需要特别注意的一点是，如要监视的全局变量为数组，则应在数组变量后的文本框中输入相应的索引值并回车后，方可勾选数组中待监视的元素。

4. 点击“SWD Clock”下拉框，可设定调试器的 SWD 时钟频率。

5. 点击下拉框“CMSIS-DAP 仿真器”，选择上位机识别到的 DAP 设备，点击“打开连接”与该设备建立连接。

6. 点击“打开连接”按钮后便可在绘图窗口看到所监视的全局变量的实时变化波形，在最下方的“参数”窗口中可查看所监视的全局变量的名称，地址、数据类型及当前值。

7. 其余操作可参考 UART-Print 界面操作说明。

4. 2 程序示例

如下程序可产生图 4-1 所示的效果。

```
1.  struct
2.  {
3.      int32_t   sin_value;
4.      int32_t   cos_value;
5.      float     f_sin_value;
6.      float     f_cos_value;
7.      int32_t   array[10];
8.  } monitor_var;
9.
10. int main(void)
11. {
12.     uint16_t i = 0;
13.     md_cmu_pll1_config(MD_CMU_PLL1_INPUT_HOSC_3, MD_CMU_PLL1_OUTPUT_96M);
14.     md_cmu_clock_config(MD_CMU_CLOCK_PLL1, 96000000);
15.
16.     /* Enable ALL peripheral */
17.     SYSCFG_UNLOCK();
18.     md_cmu_enable_perh_all();
19.     SYSCFG_LOCK();
20.
21.     while (1)
22.     {
23.         /* 产生正弦数据 */
24.         monitor_var.sin_value = (int32_t)(1000 * sin((2 * 3.1415926 / 65535) * i));
25.         /* 产生余弦数据 */
26.         monitor_var.cos_value = (int32_t)(1000 * cos((2 * 3.1415926 / 65535) * i));
27.
28.         /* 产生正弦数据 */
29.         monitor_var.f_sin_value = 1000 * sin((2 * 3.1415926 / 65535) * i) + 100;
30.         /* 产生余弦数据 */
31.         monitor_var.f_cos_value = 1000 * cos((2 * 3.1415926 / 65535) * i) + 100;
32.
33.         monitor_var.array[1] = monitor_var.f_sin_value;
34.         monitor_var.array[3] = monitor_var.f_cos_value;
35.
36.         i++;
37.     }
38. }
```