Generación y Envío de 2000 Números Pseudoaleatorios con USART1 y DMA en STM32F103

Cabrera Aguilar Fabian Andani

Universidad Nacional Autónoma de México, Facultad de Estudios Superiores Cuatitlán

17 de noviembre de 2025

1. Resumen

En esta práctica se desarrolló un sistema modular para el microcontrolador STM32F103 cuyo objetivo fue **generar 2,000 números pseudoaleatorios**, almacenarlos en **SRAM**, y transmitirlos por **USART1 vía Bluetooth** hacia un smartphone.
Se implementaron **dos soluciones**:

1. **Método por encuesta (polling)** usando únicamente el módulo USART.
2. **Método por acceso directo a memoria (DMA)** para transmitir los datos sin carga para el procesador.

Se comprobó visual y funcionalmente que, con DMA, el CPU permanece libre mientras la transferencia continúa en segundo plano.

---

2. Introducción

La comunicación serial asíncrona (USART) es ampliamente utilizada para transmitir datos hacia módulos Bluetooth, terminales, y sistemas externos. Sin embargo, cuando se envían grandes cantidades de información, el método tradicional basado en **polling** bloquea al procesador en cada byte transmitido.

El módulo **DMA (Direct Memory Access)** del STM32F103 permite mover bloques completos desde memoria hacia periféricos sin intervención continua de la CPU. Esto mejora el rendimiento general del sistema y reduce consumo de energía.

Este trabajo compara ambas técnicas en el envío de datos generados en tiempo real.

---

3. Módulos y recursos del procesador utilizados

3.1 USART1

- Configurado a 9 600 bps.
- Pines PA9 (TX) y PA10 (RX).
- Empleado para transmisión hacia el módulo Bluetooth.

- Rutinas esenciales:
    - usart1_init(): configuración del periférico.
    - usart1_sendbyte(): envío directo de un byte (polling).

3.2 DMA1 Canal 4

- Canal asociado al transmisor USART1_TX.
- Configurado para transferencias:
    - *memoria → periférico*
    - modo *incrementar memoria*
    - tamaño de dato 8 bits
    - activado por evento “USART1 TX empty”.

3.3 SRAM interna del microcontrolador

- Lugar donde se almacenaron los **2,000 bytes generados**.
- Administrada mediante una librería (“sram_pool”).

3.4 PRNG (generador pseudoaleatorio)

- Librería dedicada para generar números tipo ASCII.
- Utiliza fórmula lineal congruencial.

3.5 GPIO (LED indicador)

- LED en PC13 para indicar actividad:
    - Parpadeo en método polling.
    - Encendido fijo o libre durante DMA, mostrando menor carga del CPU.

---

4. Solución propuesta (algoritmos usados)

4.1 Generación de 2,000 números pseudoaleatorios

La librería prng2000.c contiene un generador tipo LCG:

x = (a * x + c)

Se utiliza una semilla modificada en cada llamada, generando números de 0–9 en formato ASCII. Los números se almacenan secuencialmente en un arreglo de 2000 posiciones en SRAM.

Partes esenciales señaladas del programa:

- Definición del pool de memoria:

static char pool_sram[2000];

- Función principal del PRNG:

pool_sram[i] = '0' + (x % 10);

Estas líneas ilustran la técnica sin incluir el archivo completo.

4.2 Envío mediante método por encuesta (polling)

Implementado en mode_poll.c.

Algoritmo:

1. Mostrar mensaje “modo por encuesta”.
2. Encender LED.
3. Recorrer los 2000 bytes.
4. En cada byte:
    - Llamar a usart1_sendbyte().
    - Esperar bandera TX completa.
    - Parpadear LED periódicamente.

Partes señaladas del programa:

- Envío directo:

usart1_sendbyte(buffer[i]);

- Parpadeo controlado:

led_toggle();

delay_ms(50);

Este método **bloquea completamente al CPU**, pues cada envío requiere espera activa.

4.3 Envío mediante DMA (acceso directo a memoria)

Implementado en mode_dma.c.

Algoritmo:

1. Preparar dirección origen → SRAM.
2. Preparar dirección destino → USART1->DR.
3. Configurar canal 4 del DMA:
    - Memoria incrementable
    - Periférico fijo
    - Longitud = 2000 bytes
4. Limpiar banderas.
5. Activar canal DMA.
6. El USART activa automáticamente la transferencia.

Partes necesarias del programa:

- Configuración esencial del canal:

```
DMA1_Channel4->CPAR = (uint32_t)&USART1->DR;
DMA1_Channel4->CMAR = (uint32_t)pool_sram;
DMA1_Channel4->CNDTR = 2000;
```

- Activación del canal DMA:

```
DMA1_Channel4->CCR |= DMA_CCR1_EN;
```

Durante la transferencia, el LED permanece sin parpadeo forzado, ya que el CPU **no interviene**.

---

5. Análisis de resultados

5.1 Método por encuesta (polling)

- El CPU permanece **ocupado el 100% del tiempo** enviando byte por byte.
- El LED refleja actividad constante y bloqueante.
- El tiempo del envío es notoriamente mayor debido a espera activa.

5.2 Método mediante DMA

- La CPU queda **libre inmediatamente** después de activar el canal DMA.
- El LED puede indicar libremente el estado sin afectar la transmisión.

- El envío es más rápido y eficiente.
- El simulador y el depurador muestran que el CPU ejecuta otras tareas mientras el DMA continúa en segundo plano.

Comparación visual:

Característica	**Polling**	**DMA**
Carga del CPU	Muy alta	Casi nula
Tiempo de CPU bloqueado	Completo	Prácticamente 0
Complejidad	Baja	Media
Eficiencia energética	Baja	Alta
Velocidad total	Menor	Mayor

Se concluye que el uso de DMA es claramente superior para transmisiones masivas.

---

6. Conclusión

La implementación modular del sistema permitió observar claramente las diferencias entre el método tradicional por encuesta y el envío por DMA. El enfoque DMA demostró liberar completamente al CPU, permitiendo que este continúe ejecutando otras tareas mientras la transferencia se realiza de forma automática.

El uso de librerías separadas (USART, DMA, PRNG, SRAM, indicadores LED, modos de operación) permitió un diseño limpio y escalable, adecuado para aplicaciones reales donde se procesan y transmiten grandes cantidades de datos.

Este proyecto demuestra la importancia del DMA en microcontroladores, especialmente en sistemas embebidos con requisitos de eficiencia, bajo consumo y transmisiones intensivas.

---

7. Referencias bibliográficas

1. STMicroelectronics. *RM0008 Reference Manual for STM32F103*.
2. STMicroelectronics. *STM32F10x Standard Peripheral Library*.
3. ARM. *Cortex-M3 Technical Reference Manual*.
4. Valvano, J. *Embedded Systems: Real-Time Interfacing to ARM Cortex-M Microcontrollers*.
5. Documentación oficial Keil uVision5.