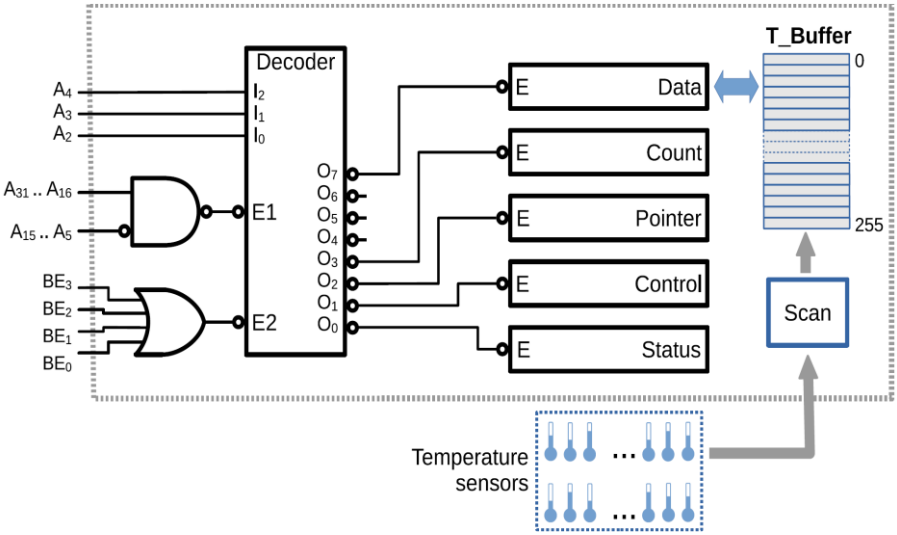
1 La figura muestra los detalles del adaptador de un dispositivo de adquisición de temperaturas conectado a un procesador MIPS R2000. Dicho dispositivo puede ser configurado para registrar las temperaturas de hasta 256 sensores ubicados en las estancias de un edificio. Los valores de temperatura se hallan codificados como enteros con signo de 32 bits. Todos los registros del interfaz son de tamaño 32 bits.



La operación del dispositivo se inicia mediante una orden de *UPDATE*, escrita en el registro de Control. En respuesta a dicha orden, el sistema escaneará las medidas de temperatura de hasta 256 sensores y las almacenará en un buffer interno (*T_Buffer*), en posiciones consecutivas. Una vez *T-Buffer* se actualice con las medidas de temperatura, existen dos posibles modos de transferir su contenido al espacio de memoria de usuario: modos *PIO* y *ADM*. En modo *PIO*, el dispositivo se encontrará *Ready* tan pronto *T_Buffer* se actualice, de modo que la transferencia de datos desde *T_Buffer* a memoria de usuario debe hacerse por programa. En modo *ADM*, una vez *T_Buffer* se actualiza, los datos se transfieren por *ADM* a un buffer en memoria de usuario, de modo que el dispositivo se encontrará *Ready* al término de la transferencia por *ADM*. El adaptador contiene los siguientes registros:

STATUS (lectura y escritura)

- **RDY** (bit 0). Bit *Ready*. Es puesto a 1 por el hardware dependiendo del modo de transferencia
  - Modo *PIO*: un vez que *T_Buffer* ha sido actualizado con las medidas de temperatura
  - Modo *ADM*: cuando el contenido de *T_Buffer* ha sido totalmente transferido a la memoria de usuario por *ADM*.

  Para hacer RDY = 0, basta con escribir un 0 sobre dicho bit (obsérvese que no hay un bit específico de cancelación en este interfaz).
- **IEN** (bit 1). Bit *Interrupt Enable*. Si IEN = 1, la puesta a 1 del bit RDY causa la activación de INT3*.

CONTROL (solo escritura)

- **UPD** (bit 0). Bit *UPDATE*. Cuando se pone a 1 por software, el sistema inicia el registro de temperaturas de los distintos sensores.
- **MOD** (bit 1). Bit *MODO*. A 0 para *PIO* y a 1 para *ADM*. Se ignora cuando UPD = 0.

POINTER (solo escritura)

En modo *ADM*, este registro debe actualizarse con la dirección de inicio del buffer en memoria de usuario, donde el contenido de *T_Buffer* es transferido por *ADM*. No tiene ningún uso en modo *PIO*.

COUNT (solo escritura)

Debe actualizarse con el número de sensores de temperatura a registrar por el sistema con cada orden *UPDATE*.

DATA (lectura y escritura)

Este registro (contiene un valor de temperatura cada vez) se usa solo en modo *PIO*, durante la transferencia por programa del contenido de *T_Buffer*. El registro almacena temporalmente el siguiente valor de temperatura a ser transferido desde *T_Buffer*. Cada vez que se lee el registro, éste se actualiza automáticamente con el siguiente valor de temperatura almacenado en *T_Buffer*.

a) Calcule la dirección base del adaptador y los desplazamientos de cada uno de los registros (en hex).

Dirección Base (DB):	0xFFFF0000

Registro	**Desplazamiento**
Status	BA + 0x00
Control	BA + 0x04
Pointer	BA + 0x08

Registro	**Desplazamiento**
Count	BA + 0x0C
Data	BA + 0x1C

b) De acuerdo a su descripción, se trata de un dispositivo de *bloques* o de *caracteres*? Justifique la respuesta.

This is a character device because there are no block identifiers.

c) Escriba el código de la función del sistema encargada de dar una orden *UPDATE* para escanear *N* sensores y, una vez actualizado *T_Buffer*, almacenar su contenido en la dirección de memoria proporcionada por el usuario **empleando modo *PIO***. Asuma que *N* se pasa en el registro $a0 y el que el puntero al buffer de memoria de usuario lo hace en $a1. La transferencia se debe **sincronizar mediante *consulta de estado***.

```
PIO_Update:
     la $t0, 0xFFFF0000 # BA of Temperature Logger
     sw $a0, 0x0C($t0)  # Count
     sw $zero, 0($t0)   # Status: Disable interrupts
     li $t1, 1          # CLR=0, MOD=0 (PIO), UPD=1
     sw $t1, 0x04($t0)  # Control: send UPD command
Poll:
     lw $t1, 0($t0)     # Read Status
     andi $t1, $t1, 1   # Isolate RDY bit
     beqz $t1, Poll     # Retake polling loop
     sw $zero, 0($t0)   # Clear RDY bit
Transfer:
     lw $t1, 0x1C($t0)  # Read temperature from Data register
     sw $t1, 0($a1)     # Store in user buffer
     addi $a1, $a1, 4   # Update buffer address
     addi $a0, $a0, -1  # Decrease number of transfers
     bnez $a0, Transfer # Retake transfer loop for next temperature









b retexc
```

d) Escriba el código de la función del sistema encargada de dar una orden *UPDATE* para escanear *N* sensores y, una vez actualizado *T_Buffer*, almacenar su contenido en la dirección de memoria proporcionada por el usuario **empleando modo *ADM***. Asuma que *N* se pasa en el registro $a0 y el que el puntero al buffer de memoria lo hace en $a1. La espera a que la tranferencia por *ADM* se complete, se hará asumiendo un **entorno multitarea**.

```
DMA_Update:
     la $t0, 0xFFFF0000 # BA of Temperature Logger
     sw $a0, 0x0C($t0)  # Count
     sw $a1, 0x08($t0)  # Puntoer
     li $t1, 2          # IEN=1
     sw $t1, 0x00($t0)  # Status: enable interrupts
     li $t1, 3          # MOD=1 (DMA), UPD=1
     sw $t1, 0x04($t0)  # Control: send UPD command
     jal suspende_este_proceso




b retexc
```

a) Escriba el código de la rutina de servicio de la interrupción Int3.

```
Int3:
     la $t0, 0xFFFF0000   # BA of Temperature Logger
     sw $zero, 0x00($t0)  # Cancela Int3 (RDY=0) e IEN=0
     jal activa_proceso

     b retexc
```