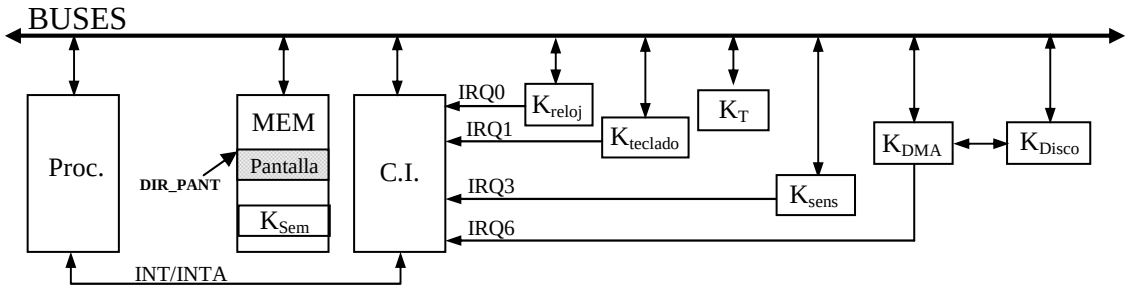
Universidad del País Vasco
Facultad de Informática

Departamento de Arquitectura y Tecnología de Computadores

Arquitectura de Computadores I

Subsistema de entrada/salida 3 (solución): Telepeaje

Queremos diseñar un sistema que controle las cabinas de pago automático (las denominadas de telepago) sitas en los peajes de salida de los tramos de precio fijo de una autopista. Para ello, el sistema dispone, en cada cabina, de los dispositivos indicados en la figura siguiente.



Las características de los controladores de los periféricos son las siguientes:

K_T: Es el controlador de telepago. Tiene tres registros: registro de control (R_CONT_KT), registro de estado (R_EST_KT) y registro de datos (R_DAT_KT). El registro de estado toma el valor 1 cuando se detecta un coche que posee el sistema de telepago. En el registro de datos —de 5 bytes—, estará el número de identificación (*NI*) del coche detectado. En el registro de control se debe realizar una secuencia de *strobe*, gracias a la cual el valor del registro de estado pasará a ser 0 y el controlador quedará listo para detectar el siguiente coche. La sincronización de este periférico se realizará por **encuesta**.

K_Sem: Es el controlador del semáforo que hay encima de la cabina. Tiene un registro de control (R_CONT_KSem) mapeado en memoria; cuando se escribe un 1 en él, el semáforo se pone en rojo, y en verde cuando se escribe un 0.

K_sens: Es el controlador de un sensor situado junto a la barrera de salida de la cabina. Solicita una interrupción cuando detecta que un coche ha pasado completamente frente a él.

K_DMA: Es el controlador de DMA y se utiliza para hacer transferencias de memoria a disco. Posee los siguientes registros:

- **registro de control** (R_CONT_KDMA): cuando se escribe un 1 en él se inicia la transferencia (se pone a 0 automáticamente).
- **registro de dirección** (R_DIR_KDMA): dirección de comienzo del bloque a transferir.
- **registro de longitud** (R_LON_KDMA): contiene la longitud en bytes del bloque a transferir.
- **registro de estado** (R_EST_KDMA): cuando su contenido es un 0 indica que la última operación realizada ha sido errónea.

K_DISCO: Es el controlador de disco. Para realizar la transferencia hay que programar este controlador, para lo que tenemos la rutina *ProgramarKDisco()*.

Los controladores de reloj, teclado e interrupciones son los mismos que los vistos en la asignatura.

En este sistema la sincronización de **teclado** se realiza **por interrupción**. La pantalla está mapeada en memoria a partir de la dirección DIR_PANT.

El **funcionamiento** del sistema es el siguiente.

Cuando a la cabina controlada llega un coche que dispone del sistema de telepago, si el semáforo está verde, lo detectará el controlador K_T. Como el pago se realizará automáticamente (gracias al número de identificación del coche), el sistema deberá levantar la barrera que hay a la salida de la cabina, mediante la rutina ***levantar_barrera()***, para dejar pasar al coche.

La barrera de salida se mantendrá levantada el tiempo necesario para que el coche pase completamente, y 2 segundos más, por razones de seguridad, tras lo cual se bajará, mediante la rutina ***bajar_barrera()***. Para simplificar el sistema, supondremos que es necesario bajar la barrera para detectar el coche siguiente.

Para gestionar el cobro automático del peaje, cada vez que pasa un coche por la cabina, se debe guardar en memoria la información siguiente: *NI* número de identificación —de 5 bytes— y la hora de paso (que está en la variable global *HORA*) —ésta también de 5 bytes—. Para escribir esa información en memoria, disponemos de la rutina ***escribmem (NI, HORA, númerococches)***; la propia rutina se encarga de calcular en qué posición de memoria debe escribir la información en cada momento, teniendo en cuenta el número de coches que han pasado por la cabina y que la dirección de comienzo en la que se escribe la información correspondiente al primer coche que pasa al inicializar el sistema es INFO_DIR. En lo que respecta a la variable *HORA*, disponemos de las dos rutinas siguientes: ***InicializarHora()***, que asigna a la variable *HORA* el valor actual indicado por el sistema operativo, y ***ActualizarHora()***, que actualiza la variable *HORA* en un segundo.

El color del semáforo de la cabina se controla por medio del teclado:

- Al pulsar la tecla R, el semáforo se pondrá en rojo y el sistema no aceptará que pasen coches por esa cabina (para simplificar el sistema, supondremos que la persona que debe pulsar la tecla lo hará cuando no haya coches en la cabina). Además de cambiar el color del semáforo, el sistema debe transferir a disco, mediante DMA, toda la información que se ha ido almacenando en memoria acerca de los coches que han pasado a lo largo de todo el intervalo de tiempo en que el semáforo ha estado verde. Al finalizar la transferencia por DMA, si ha ocurrido algún error, el sistema deberá intentarlo otras dos veces más como máximo, pero si después del tercer intento no se ha conseguido finalizar con éxito la transferencia, entonces de deberá ejecutar la rutina ***error_transferencia()*** para que aparezca un mensaje en pantalla y se deberá finalizar el programa. Si no ocurren errores en la transferencia, el sistema quedará a la espera de que se ponga nuevamente el semáforo en verde. La sincronización con el controlador de DMA se debe hacer por **interrupción**.
- Estando el semáforo en rojo, al pulsar la tecla V, el semáforo se pondrá en verde y el sistema recuperará el funcionamiento indicado anteriormente. La información acerca de los coches que pasen por la cabina en el nuevo intervalo de tiempo en que el semáforo esté verde se escribirá otra vez a partir de la dirección INFO_DIR

Se pide: Escribe en lenguaje algorítmico todas las rutinas de atención que consideres necesarias, así como el programa principal. Comenta cualquier supuesto que hagas para la resolución del problema. Se valorará representar el comportamiento del sistema mediante un autómata de estados y transiciones.

Solución

En primer lugar, intentaremos deducir un autómata de estados que refleje el funcionamiento del sistema. Para ello es fundamental distinguir bien todas las situaciones o estados posibles del funcionamiento del sistema, así como identificar correctamente cuáles son las condiciones que se han de cumplir para que se produzcan las transiciones de unos estados a otros. En general, las transiciones entre estados serán consecuencia de sucesos o eventos externos al propio sistema, que serán detectados por medio de los periféricos presentes en el sistema, sea cual sea el modo en que dichos periféricos se sincronicen con el procesador, tanto por encuesta como por interrupción.

En este ejercicio no se nos dice cuáles son los estados posibles del sistema, por lo que es necesario leer atentamente la información proporcionada acerca del funcionamiento del sistema, para poder deducir correctamente los estados del mismo. Veamos, pues, cuáles son las circunstancias y sucesos o eventos que pueden acontecer en este sistema y cuáles son las consecuencias que de ellos se derivan según el estado del sistema. Para ello, releeremos el enunciado atentamente

Para empezar, supondremos que, cuando el sistema se pone en marcha, el semáforo que está encima de la cabina estará verde y el sistema estará preparado para detectar a los coches que se acerquen, y dejarlos pasar si cumplen las condiciones necesarias. A ese estado inicial del sistema lo denominaremos *verde*.

1. Estado *verde*:

- Características del estado:
  En este estado, el sistema debe realizar de manera continua la encuesta del sensor K_T, para detectar si se acerca algún coche a la cabina. En caso negativo, debe leer una y otra vez el registro de estado del sensor K_T, hasta que detecte un coche que disponga del dispositivo de telepago; cuando eso ocurra, el sistema deberá alzar la barrera de la cabina para dejarle pasar al coche, y además deberá almacenar la información correspondiente a ese coche.
  Por otra parte, es posible que, cuando no haya ningún coche, el operario de la cabina pulse la tecla R, lo que hará que el semáforo pase a rojo y que, a partir de ese momento, no se deban detectar los coches que se acerquen a la cabina, hasta que de nuevo el semáforo se vuelva a poner en verde (lo que ocurrirá cuando el operario pulse la tecla V, es decir, en otro estado).
  Así pues, cuando el sistema está en este estado, pueden ocurrir dos eventos, y cada uno de ellos implica una transición de estado diferente.

- Transiciones posibles:

a) Cuando el sensor K_T detecta un coche que dispone del sistema de telepago, se debe alzar la barrera para dejar pasar el coche, por lo que el sistema pasará a otro estado que denominaremos *pasando_coche*.
Como queremos sicronizar el sensor K_T por encuesta, los eventos correspondientes serán tratados en el programa principal, no en una rutina de servicio a una interrupción, como habría que hacer en el caso de que la sincronización fuera por interrupciones. Así, el programa principal deberá detectar de alguna manera que un coche se ha acercado a la cabina. En la medida en que K_T es un periférico estándar, para realizar la encuesta el procesador deberá leer su registro de estado, hasta que

detecte el evento deseado. Así, cuando el valor del registro de estado sea 1, eso significará que el sensor ha detectado un coche, y el sistema deberá realizar las acciones oportunas. Por el contrario, cuando el valor del registro de estado del periférico sea 0, no habrá que hacer nada especial, únicamente seguir leyendo el registro de estado de manera continua.

- Acciones a realizar en la sección del programa encargada del tratamiento del sensor K_T:

Además de la transición de estado indicada, la sección del programa principal encargada del tratamiento de los eventos correspondientes al sensor K_T deberá realizar otras acciones (quede claro que el código correspondiente que veremos ahora se ejecutará después de haberse detectado en la encuesta que se ha acercado un coche a la cabina; no escribiremos aquí la encuesta, porque es genérica y no depende del estado del autómata; al escribir el código del programa principal comentaremos en detalle cómo realizar la encuesta):

T.1) Se debe levantar la barrera para dejar que pase el coche:

```
levantar_barrera();
```

T.2) Se debe leer el registro de datos del controlador K_T (`R_DAT_KT`), para obtener la información correspondiente al coche (el número de identificación, NI):

```
NI = InPort(R_DAT_KT);
```

T.3) Se debe realizar una secuencia de "strobe" sobre el registro de control (`R_CONT_KT`) del controlador:

```
strobe(R_CONT_KT);
```

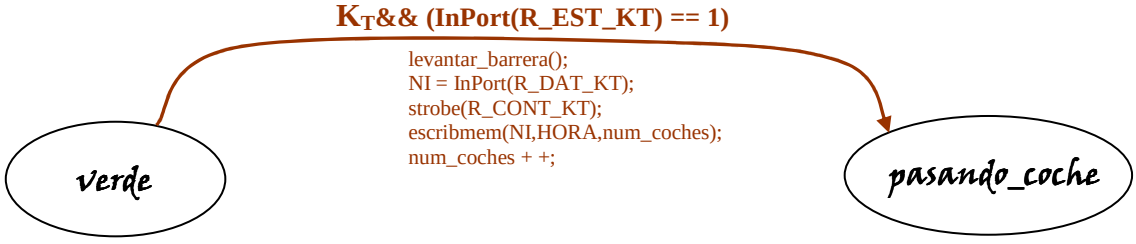
T.4) Se debe almacenar en memoria el número de identificación del coche:

```
escribmem(NI,HORA,num_coches);
```

T.5) Como está pasando otro coche, se debe controlar cuántos coches han ido pasando hasta ahora, para lo que empleamos la variable global `num_coches`, y en este momento hay que actualizarla:

```
num_coches ++;
```

Es conveniente indicar todas esas acciones sobre el autómata, ya que ello resultará de gran ayuda a la hora de escribir el código del programa. Por tanto, bajo la flecha que indica la transición de estado, escribiremos todas esas acciones, así:



b) Mientras el sistema está en el estado ***verde***, si el operario de la cabina pulsa la tecla R, el sistema deberá pasar a otro estado, en el que el semáforo de la cabina estará en rojo y el sistema no deberá detectar los coches que se acerquen a la cabina. Además de eso, se deberá transferir de memoria a disco toda la información que se ha almacenado hasta ese momento sobre todos los coches que han pasado por la cabina. Esa transferencia se debe realizar mediante DMA, por lo que será necesario programar el disco y el controlador de DMA, para que realicen la transferencia. El nuevo estado al que pasa el programa lo denominaremos ***transf_DMA***.
He aquí las acciones que hay que realizar en esta transición de estado:

- Acciones a realizar en la rutina de servicio a la interrupción del teclado:

Tec.1) Hay que leer el registro de datos (`R_DAT_Ktec`) del controlador del teclado, para saber qué tecla ha pulsado el operario:

```
código_tecla = InPort(R_DAT_Ktec);
```

Tec.2) En el caso del teclado hay que hacer una secuencia de "strobe" sobre su registro de control (`R_CONT_Ktec`):

```
strobe(R_CONT_Ktec);
```

Tec.3) Ahora se debe analizar si la tecla ha sido pulsada (MAKE) o liberada (BREAK), y además se debe mirar si ha sido la tecla R, y todo eso sólo en el estado ***verde***, ya que en cualquier otro estado no se le debe hacer caso a la tecla R:

```
if ((MAKE(código_tecla))&&(TABLA_ASCII[código_tecla]== 'R')
&&(estado_automata == verde))
```

Tec.4) En caso de cumplirse las condiciones anteriores, se debe poner en rojo el semáforo. Como el controlador del semáforo está mapeado en memoria, basta con escribir un 1 en la posición de memoria correspondiente (`SEM`), para lo que consideraremos como si esa posición se tratara de una variable global:

```
SEM = 1;
```

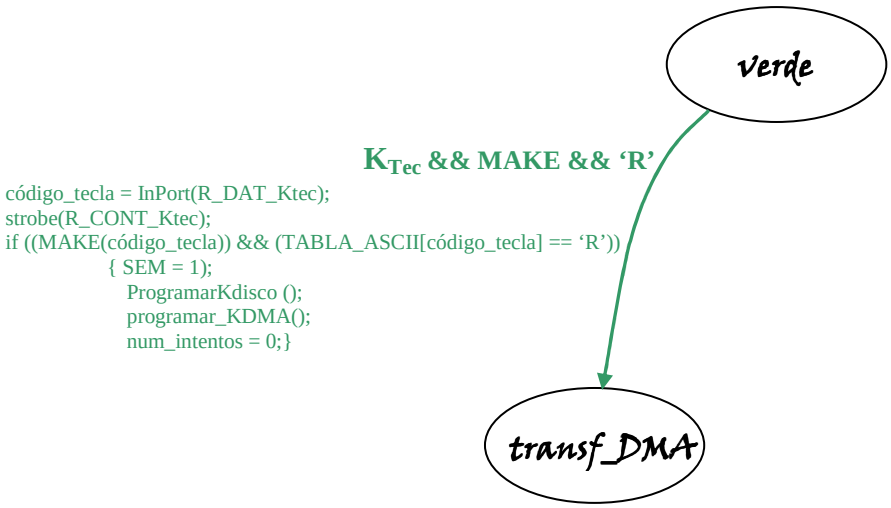
Tec.5) Como hay que realizar una transferencia por DMA, es necesario programar el disco y el controlador de DMA. Además, como sólo están permitido hasta tres intentos en caso de que en la transferencia ocurra algún error, deberemos inicializar una variable que controle el número de intentos realizados:

```
ProgramarKdisco ();
Programar_KDMA();
num_intentos = 0;
```

En donde la función `Programar_KDMA()` tiene el código siguiente:

```
OutPort(R_DIR_KDMA,INFO_DIR);
OutPort(R_LON_KDMA,num_coches × 10);
OutPort(R_CONT_KDMA,1);
```

La dirección es INFO_DIR, ya que es a partir de esa dirección en donde se ha almacenado la información asociada a los coches que han pasado por la cabina, y esos son los datos que hay que transferir a disco; la longitud del bloque a transferir es (`num_coches` × 10), ya que, por cada coche que ha pasado por la cabina, se han almacenado 10 bytes en memoria y el número total de coches que han pasado por la cabina es el indicado por la variable `num_coches`. Finalmente, al escribir un 1 en el registro de control, la transferencia comenzará automáticamente.



2. Estado *pasando_coche*:

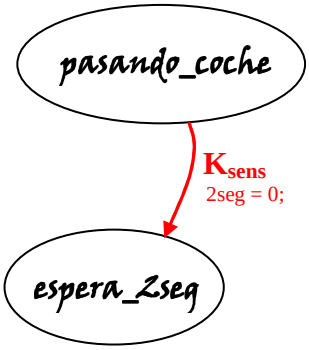
- Características del estado:
  La barrera está levantada, y está pasando un coche por la cabina.
- Transiciones posibles:

a) A la salida de la cabina hay un sensor que detecta si el coche ha pasado completamente. Así, el sistema saldrá del estado ***pasando_coche*** cuando el controlador del sensor, Ksens, genere una petición de interrupción para indicar que, efectivamente, el coche ha pasado completamente por debajo de la barrera. Como medida de seguridad, no obstante, no se deberá bajar la barrera inmediatamente, sino que esperaremos dos segundos antes de bajar la barrera (es decir, necesitaremos la variable global `2seg`). Así, tan pronto como el sensor indique que el coche ha terminado de pasar, pondremos el "cronómetro" en marcha para controlar esos dos segundos. El nuevo estado lo denominaremos ***espera_2seg***.

↳ Acciones a realizar en la rutina de servicio a la interrupción del sensor Sens:

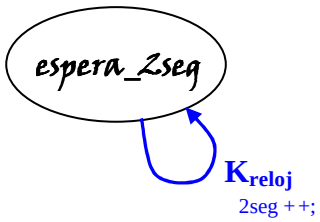
Sens.1) Inicialización de la variable `2seg`:

```
2seg = 0;
```



3. Estado *espera_2seg*:

- Características del estado:
  En este estado, el sistema está a la espera de que transcurran los dos segundos pertinentes antes de proceder a bajar la barrera, es decir, lo único que debe hacer es dejar pasar el tiempo, para lo que la rutina de servicio a la interrupción del reloj debe actualizar adecuadamente la variable `2seg`:



↳ Acciones a realizar en la rutina de servicio a la interrupción del reloj:

Reloj.1) Actualización de la variable `2seg` cada vez que interrumpe el reloj:

```
if (estado_automata == espera_2seg)
      2seg++;
```

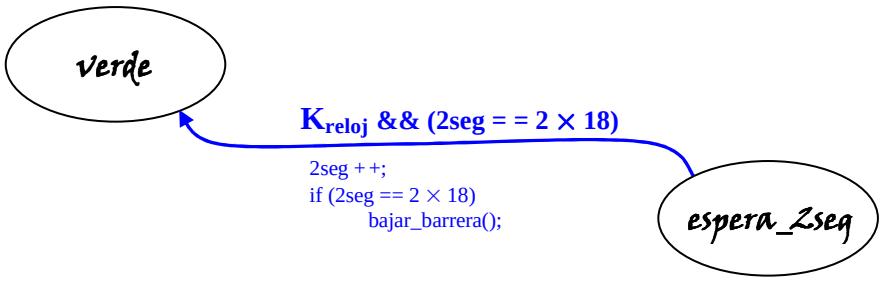
- Transiciones posibles:

a) Una vez transcurridos los dos segundos, se debe bajar la barrera y el sistema deberá volver al estado inicial, a la espera de detectar el siguiente coche. En consecuencia, la rutina de atención al reloj queda como sigue en lo que respecta a esta transición:

Reloj.2) Transición de estado, si es necesario:

```
if (2seg == 2 × 18)
   {
      bajar_barrera();
      estado_automata = verde;
   }
```

Y en el autómata:



4. Estado *transf_DMA*:

- Características del estado:
  El sistema ha llegado a este estado como consecuencia de que el operario ha pulsado la tecla R cuando el semáforo estaba verde, y con ello se ha dado comienzo a la transferencia por DMA. Así pues, cuando el sistema está en este estado el controlador de DMA está realizando la transferencia deseada y el sistema está a la espera de que finalice la misma; este hecho lo indicará el controlador de DMA

realizando una petición de interrupción. Cuando el procesador la acepte, pasará a ejecutar la rutina de servicio del controlador de DMA, en la que habrá que leer el registro de estado del mismo para saber si la transferencia ha finalizado correctamente o si, por el contrario, se ha producido algún error en la misma, dependiendo de lo cual habrá que realizar diferentes acciones.

Así, si ha ocurrido algún error en la transferencia, habrá que tener en cuenta cuántos intentos se llevan realizados: si son menos que 3, entonces hay que intentarlo de nuevo y mantenerse en el mismo estado. Si, por el contrario, se ha intentado ya tres veces realizar la transferencia y aún sigue produciéndose el error, el sistema deberá cambiar de estado, para dar por finalizado el programa (esto lo veremos en el siguiente subapartado, por tratarse de una transición). Ahora bien, si la transferencia finaliza correctamente antes del tercer intento, el sistema pasará a un estado diferente, en el que permanecerá a la espera de que el operario pulse de nuevo la tecla V (esto también lo veremos en el subapartado siguiente).

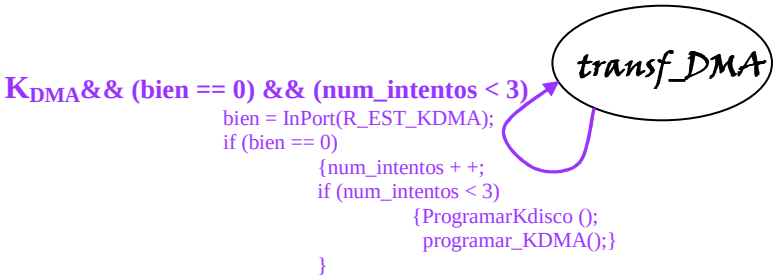
Es decir, en la rutina de servicio a la interrupción del controlador de DMA deberemos distinguir esos casos. Primeramente, veamos qué hacer si la transferencia resulta errónea pero todavía se han realizado menos de tres intentos:

- Acciones a realizar en la rutina de servicio a la interrupción del controlador KDMA:

DMA.1) Hay que leer el registro de estado; si ha ocurrido un error, hay que incrementar el valor de la variable `num_intentos`; si el nuevo valor es menor que 3, hay que programar de nuevo Kdisco y KDMA para realizar un nuevo intento de transferencia:

```
bien = InPort(R_EST_KDMA);
if (estado_automata == transf_DMA)
   {
      if (bien == 0)
          {
             num_intentos + +;
             if (num_intentos < 3)
                {
                   ProgramarKdisco ();
                   Programar_KDMA();
                }
          }
   }
```

En el autómata, queda como sigue:



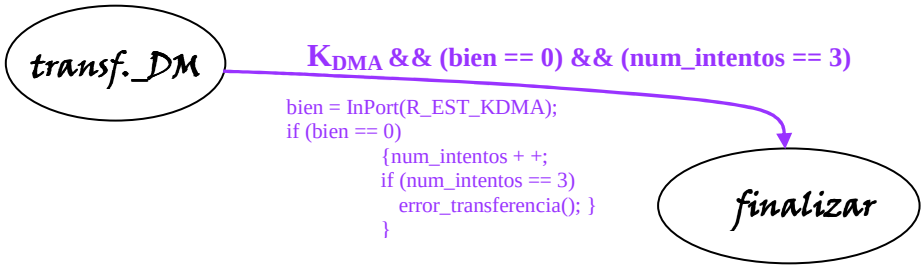
- Transiciones posibles:

a) Si después de haber hecho tres intentos de realizar la transferencia por DMA, ésta sigue siendo errónea, el sistema deberá pasar al estado ***finalizar*** y dar por finalizada la ejecución del programa.

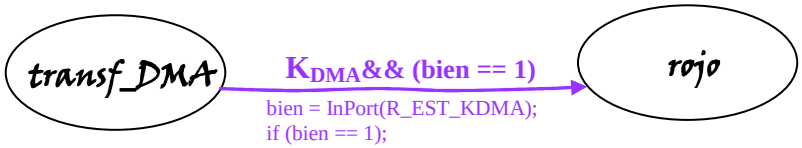
↳ Acciones a realizar en la rutina de servicio a la interrupción del controlador KDMA:

DMA.2) Una vez leído el registro de estado del controlador de DMA, si de nuevo ha ocurrido un error en la transferencia, se ha incrementado en uno el valor de la variable `num_intentos` y si el nuevo valor de ésta es 3, el sistema debe pasar al estado ***finalizar*** y ejecutar la función `error_transferencia()`para avisarle al operario de que la transferencia no se ha realizado:

```
if (num_intentos == 3)
{
     error_transferencia();
     estado_automata = finalizar;
}
```



b) Si la transferencia finaliza correctamente, el sistema debe pasar al estado ***rojo*** y no tiene que realizar ninguna otra acción, hasta que un operario pulse la tecla V, momento en el que el sistema regresará al estado inicial.



5. Estado *rojo*:

- Características del estado:
  El sistema llegará a este estado únicamente si la transferencia por DMA finaliza correctamente, y permanecerá en este estado hasta que un operario pulse la tecla V.

- Transiciones posibles:

a) Cuando el sistema está en el estado ***rojo***, si un operario pulsa la tecla V, entonces el sistema deberá pasar al estado inicial ***verde***, y el semáforo deberá ponerse en verde.

- Acciones a realizar en la rutina de servicio a la interrupción del teclado:

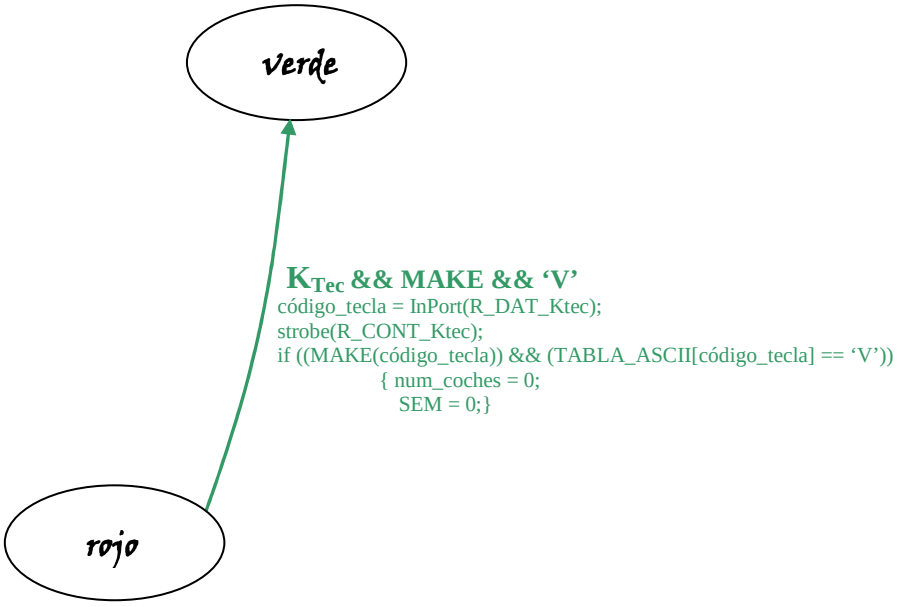
Tec.6) Se debe poner en verde el semáforo:

```
if ((MAKE(código_tecla))&&(TABLA_ASCII[código_tecla]== 'V')
&&(estado_automata == rojo))
            SEM =0;
```

Tec.7) Como el sistema volverá al estado inicial, volviendo al funcionamiento normal, se debe inicializar a 0 la variable que computa el número de coches que pasan por la cabina:

```
num_coches = 0;
```

En el autómata, plasmaremos así esta transición de estado:



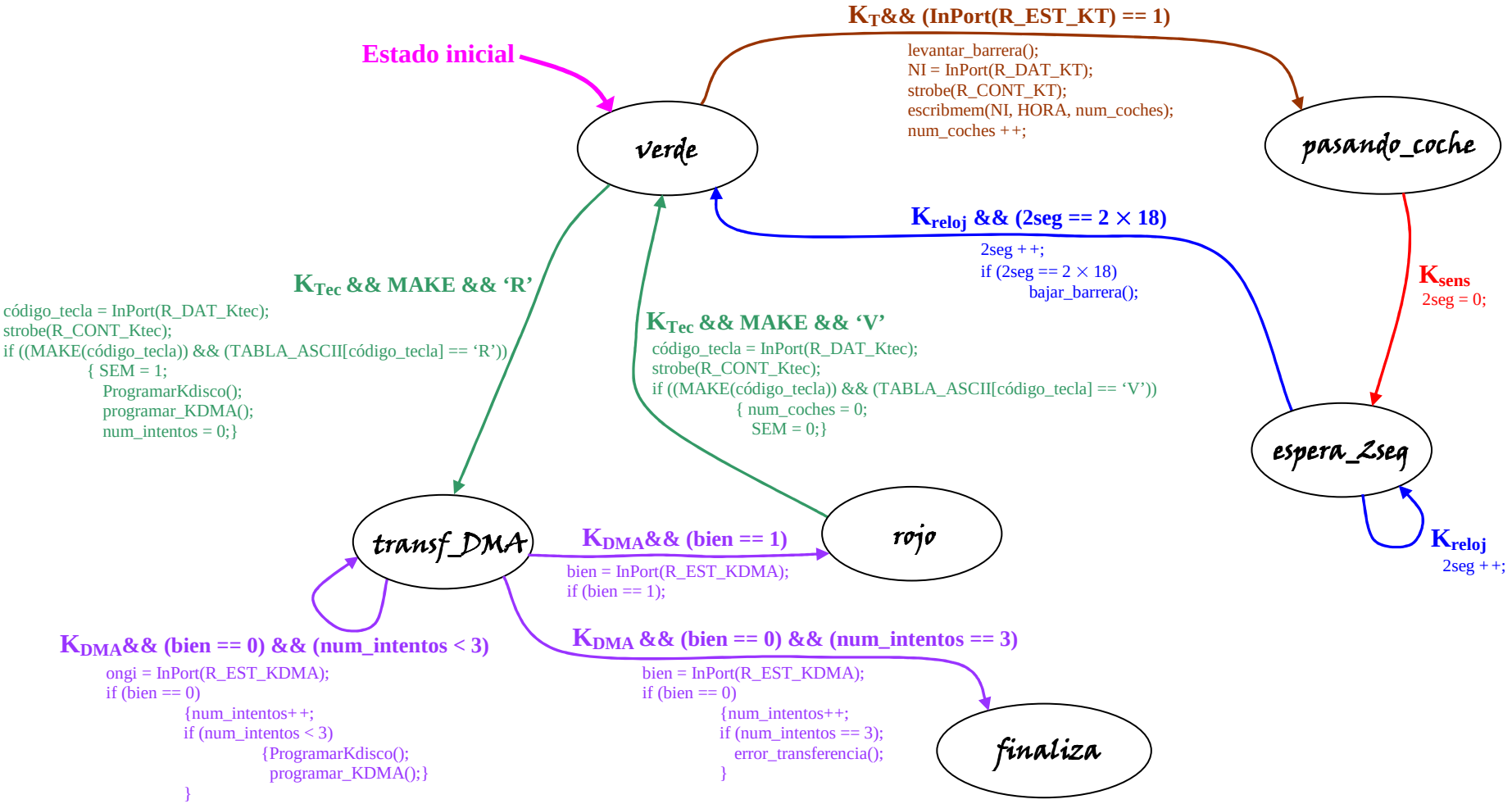
6. Estado *finalizar*:

- Características del estado:

Cuando el sistema llega a este estado, se debe finalizar la ejecución del programa. Para ello, en el programa principal pondremos un bucle que se ejecutará continuamente mientras el sistema no llegue a este estado, y el programa saldrá del bucle cuando el sistema llegue a este estado, para finalizar la ejecución del programa.

```
while (estado_automata != finalizar)
```

Teniendo en cuenta conjuntamente todos los casos posibles vistos, el autómata queda así:



Y ya sólo resta escribir el código. Por una parte, tenemos que escribir el programa principal, en el que se ejecutará un bucle dependiente del estado del sistema, que finalizará cuando el sistema llegue al estado *finalizar*, y dentro del bucle se realizará la encuesta del periférico KT. Además, también tenemos que escribir las rutinas de atención a las interrupciones (RAI) de los periféricos: K_{Sens}, K_{Tec}, K_{reloj} y K_{DMA}.

✍ Programa principal.

En el programa principal se deben realizar determinada acciones, en concreto: inicialización de las variables (cuando sea necesario), modificación de algunos de los componentes del vector de interrupciones (al principio del programa) —para asegurar que se ejecutarán nuestras rutinas de servicio a las interrupciones, y no las de un hipotético sistema operativo— y recuperación de los valores iniciales de los mismos (al final del programa), y realización de un bucle para que nuestro programa se ejecute de manera continua mientras no tenga que finalizar. Dentro de ese bucle se realizará la encuesta del periférico KT, para detectar cuándo se acerca un coche que tiene el sistema de telepago a la cabina.

- Variables que hay que inicializar:
  Para empezar, supondremos que cuando el sistema se pone en marcha estará en el estado *verde*. Debemos tener en cuenta, por tanto, que para asegurar el correcto funcionamiento del sistema es necesario saber en todo momento en qué estado se encuentra, para lo que utilizaremos una variable global, que denominaremos `estado_automata`, cuyo valor inicial será *verde*. Además, deberemos poner en verde el semáforo y también inicializar a 0 la variable `num_coches` que va a computar cuántos coches van pasando por la cabina de telepago mientras el semáforo está en verde.
  Así, estas son las inicializaciones que hay que realizar:
  - ✓ `estado_automata = verde;`
  - ✓ `num_coches = 0;`
  - ✓ `SEM = 0;  //semáforo = verde`

  Además, al principio de nuestro programa se debe ejecutar la función `InicializarHora()` para poder controlar correctamente la hora actual dentro de nuestra aplicación.

  Conviene subrayar que no es necesario inicializar el resto de variables utilizadas, ya que se inicializan con el valor adecuado en el preciso momento en que van a ser utilizadas. Por ejemplo, la variable `2seg` se inicializa con el valor 0 en el instante en que el sistema pasa del estado *pasando_coche* al estado *espera_2seg*, ya que es en ese momento cuando se debe poner en marcha el "cronómetro".

- Componentes del vector de interrupciones que hay que modificar:
  Si se emplea un PC para controlar el sistema, es necesario modificar los componentes del vector de interrupciones correspondientes a los periféricos del sistema, para que el PC realice el control que nosotros deseamos, es decir, para que, cuando interrumpa uno de los periféricos, se ejecuten las rutinas de servicio que escribamos nosotros, no las que tiene programadas el sistema operativo.

Así, en este caso concreto, los componentes que hay que modificar son los siguientes: el del reloj (entrada 0x1C del vector de interrupciones), el del controlador del teclado (entrada 0x09), el del sensor K_sens (entrada 0x0B) y el del controlador de DMA (entrada 0x0E). Para ello, supondremos que disponemos de una función ya escrita, que toma como parámetros las entradas del vector de interrupciones que hay que modificar:

- ✓ `CambiaVI(0x1C, 0x09, 0x0B, 0x0E);`

De la misma manera, cuando finalice la ejecución de nuestro programa, será necesario recuperar los valores originales de esos componentes, para que el sistema operativo tome de nuevo el control del PC:

- ✓ `RecuperaVI(0x1C, 0x09, 0x0B, 0x0E);`

- Control del bucle:

  Debemos asegurarnos de que nuestro programa se ejecuta de manera continua, mientras no le indiquemos que debe finalizar su ejecución. Para ello, pondremos un bucle dependiente del estado del autómata, que se repetirá una y otra vez mientras se cumpla la condición de que el sistema no esté en el estado en que hay que finalizar el programa. En este caso concreto, está previsto que se pueda finalizar la ejecución de nuestro programa si se realizan tres intentos de transferencia por DMA y no se consigue que ninguno finalice correctamente. En ese caso, el sistema pasará al estado *finalizar*, y éste será el estado que utilizaremos como condición del bucle:

  - ✓ `while (estado_automata != finalizar)`

- Encuesta del periférico K_T:

  Como se ha indicado, la sincronización con el periférico K_T se debe realizar por encuesta. Pero el sistema únicamente debe detectar los coches mientras el semáforo está verde, y si está rojo no se debe permitir el paso de los coches por la cabina. En consecuencia, dentro del bucle principal pondremos como condición el estado del sistema: si el autómata está en el estado *verde*, se deberá leer el registro de estado del controlador K_T para ver si detecta algún coche; si el valor leído es 0, se debe leer de nuevo el registro de estado, mientras no cambie el estado del autómata. Si, por el contrario, el valor leído en el registro de estado es 1, estando el autómata en el estado *verde*, eso significa que en la cabina se ha detectado un coche que dispone del sistema de telepago, y se le debe permitir el paso, para lo que el programa principal deberá ejecutar las acciones correspondientes al periférico K_T:

```
while (estado_automata != finalizar)
    {
       if (estado_automata == verde)
          {
             coche = InPort(R_EST_KT);
             if (coche == 1)
                {
                   levantar_barrera();
                   NI= InPort(R_DAT_KT);
```

```
                        strobe(R_CONT_KT);
                        escribmem(NI,HORA,num_coches);
                        num_coches++;
                    }
                }
            }
```

Resumiendo, el código del programa principal queda así:

```
void main()
{
        // inicialización de variables
 estado_automata = verde;
 num_coches = 0;
 SEM = 0 // poner el semáforo en verde
 InicializarHora();

        // modificación de los componentes del vector de interrupciones
 CambiaVI(0x1C, 0x09, 0x0B, 0x0E);

        // bucle principal
 while (estado_automata != finalizar);
   {
        //si el estado del autómata es verde, detectar los coches; si no, no.
    if (estado_automata == verde)
       {
         coche = InPort(R_EST_KT);
            {
               if (coche == 1)
                     //si se detecta un coche:
                  {
                    levantar_barrera();
                    NI = InPort(R_DAT_KT);
                    strobe(R_CONT_KT);
                    escribmem(NI,HORA,num_coches);
                    num_coches++;
                        // transición de estado pertinente
                    estado_automata = pasando_coche;
                  }
            }
        }
   }
        // recuperación de los componentes del vector de interrupciones al
        // finalizar el programa
 RecuperaVI(0x1C, 0x09, 0x0B, 0x0E);
}
```

➢ Rutina de atención a la interrupción (RAI) del sensor K_sens.

Cuando el sensor que está a la salida de la cabina de telepago detecta que un coche ha pasado completamente debajo de la barrera, se debe poner en marcha el "cronómetro" para esperar dos segundos antes de bajar la barrera, por razones de seguridad. En consecuencia, en este rutina de servicio únicamente se debe inicializar la variable `2seg` con el valor 0, y cambiar el estado del autómata, desde luego. No obstante, para obviar las posibles interferencias que pudiera sufrir el funcionamiento del sensor, al comienzo de la rutina de servicio pondremos como condición que el sistema esté en el estado *pasando_coche*, y en cualquier otro caso no se hará nada.

```
void interrupt RAI_Ksens()
   {
    if (estado_automata == pasando_coche)
      {
            // poner el cronómetro en marcha para contar 2 segundos
       2seg = 0;
            // transición de estado pertinente
       estado_automata = espera_2seg;
      }
    Eoi();
    IRET;
   }
```

➢ Rutina de atención a la interrupción (RAI) del controlador K_{reloj}.

En este caso necesitamos la variable `tic`, ya que cada segundo se le debe llamar a la función `ActualizarHora()`, para controlar adecuadamente la variable HORA. Pero en lo que respecta a la variable temporal `2seg`, la actualizaremos independientemente en cada ejecución de la rutina, contando cada una de las interrupciones del reloj, para evitar posibles problemas.

```
void interrupt RAI_Kreloj()
{
 static int tic = 0;
      //control de los segundos
 tic++;
 if (tic == 18)
      {
       tic = 0;
            //actualización de la variable HORA cada segundo
      ActualizarHora();
      }

 if (estado_automata == espera_2seg)
        {
          2seg++;
            // transición de estado pertinente
```

```
            if (2seg == 2 × 18)
               {
                   bajar_barrera();
                   estado_automata = verde;
               }
         }
 IRET;
}
```

➢ Rutina de atención a la interrupción (RAI) del controlador del teclado K_{Tec}.

Al detectar que se ha pulsado/liberado una tecla, tendremos en cuenta el estado del autómata, para no realizar ninguna acción si se pulsa una tecla cualquiera en cualquier estado, sino únicamente si se pulsan las teclas R y V, en los estados correspondientes.

```
void interrupt RAI_Ktec()
{
       //lectura del código de posición de la tecla pulsada/liberada
 código_tecla = InPort(R_DAT_Ktec);
       //secuencia de "strobe" sobre el registro de control
 strobe(R_CONT_Ktec);
       //transiciones de estado pertinentes
 if ((MAKE(código_tecla))&&(TABLA_ASCII[código_tecla]=='R')
    &&(estado_automata == verde))
    {
              //poner el semáforo en rojo
      SEM = 1;
      ProgramarKdisco ();
      Programar_KDMA();
      num_intentos = 0;
      estado_automata = transf_DMA;
    }
 else
 if ((MAKE(código_tecla))&&(TABLA_ASCII[código_tecla]=='V')
    &&(estado_automata == rojo))
    {
              //poner el semáforo en verde
      SEM = 0;
      num_coches = 0;
      estado_automata = verde;
    }
 Eoi();
 IRET;
}
```

En este caso, se lee el registro de datos en todos los estados del sistema, y también se realiza la secuencia de `strobe`, pero sólo se realizan acciones concretas si se pulsa la tecla V estando en el estado ***rojo***, o la tecla R estando en el estado ***verde***.

El código de la función `Programar_KDMA()` que se ha utilizado en la rutina anterior:

```
void Programar_KDMA()
{
        OutPort(R_DIR_KDMA,INFO_DIR);
        OutPort(R_LON_KDMA,num_coches × 10);
        OutPort(R_CONT_KDMA,1);
}
```

- Rutina de atención a la interrupción (RAI) del controlador K_{DMA}.

```
void interrupt RAI_KDMA()
{
 bien = InPort(R_EST_KDMA);
 if (estado_automata == transf_DMA)
    {
      if (bien == 0)
      {
            num_intentos++;
            if (num_intentos < 3)
            {
                 ProgramarKdisco ();
                 Programar_KDMA();
            }
            else
            {
                 error_transferencia();
                 estado_automata = finalizar;
            }
      }
       else
        estado_automata = rojo;
 Eoi();
 IRET;
}
```