# Elementos para la simulación de sistemas discretos

# Elementos para la simulación discreta

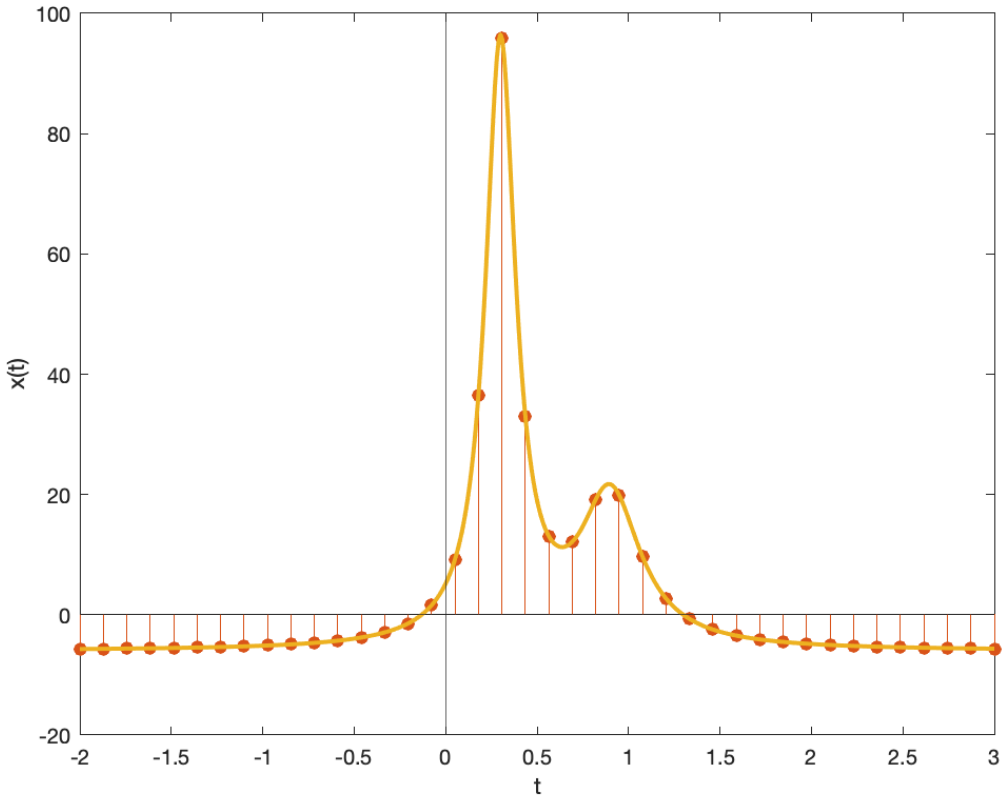


## Introducción

Un sistema discreto es un sistema que puede ser representado utilizando ecuaciones en diferencias y que opera sobre señales discretas. Una señal discreta se puede representar como una sucesión de pulsos. Por ejemplo, la siguiente figura muestra la función $y = \text{humps}(x)$ discretizada:

```
X = linspace(-2,3,40)';
Y = humps(X);
stem(X,Y,'filled')
hold on
t = -2:0.01:3;
```

```
y = humps(t);
plot(t, humps(t), 'linewidth', 2)
xline(0); yline(0); xlabel('t'); ylabel('x(t)')
```



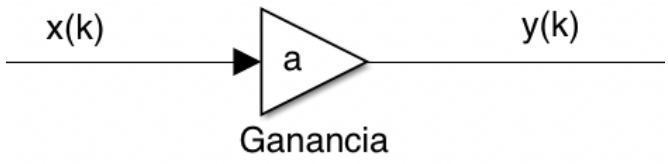

Un sistema discreto toma a la entrada una o más señales discretas y produce en la salida una o más señales discretas. Esto es, las señales tanto a la entrada como a la salida cambian en instantes discretos de tiempo. En muchos de estos sistemas las señales no son inherentemente discretas: son señales continuas que se 'discretizan' por medio de un procedimiento que se denomina 'muestreo'.

Nos referiremos a las señales que varían con el tiempo usando la notación $x\left(t\right)$. La notación para el correspondiente sistema discreto será $x\left(k\right)$, donde $k$ es el correspondiente número de pulso, es decir, $x\left(k\right) = x\left(k\,T\right)$, donde $T$ es el periodo de muestreo.

A continuación se describen los elementos que se utilizarán para simular sistemas discretos. Según vayamos trabajando con sistemas discretos descubriremos nuevos bloques y formas más simples de hacer las cosas. Pero no se trata de hacer modelos óptimos, sino de aprender a modelar y entender los problemas que se nos propongan. Ya habrá tiempo de pasar a la optimización. Pensemos siempre que es mejor presentar modelos lo más sencillos posibles y de fácil seguimiento y comprensión, que modelos que sean muy difíciles de seguir y que desanimen el estudio.

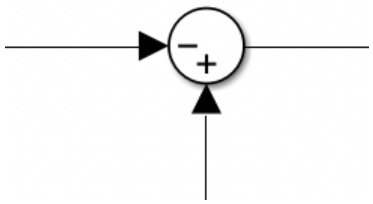
## Ganancia discreta

Representa la ecuación algebraica $y\,(k) = a\;x\,(k)$. Esto es, la salida representa la entrada multiplicada por la ganancia. Se representa con el símbolo:




## Bloque de suma

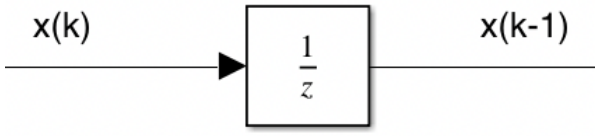
Permite la suma/diferencia de una o más entradas. Se representa por:




## Unidad de retraso (Unit Delay)

Es el bloque fundamental de los sistemas en tiempo discreto. A veces se denomina también 'cambio de registro' o 'elemento de retraso en el tiempo'. La salida de este bloque es la entrada en el instante de tiempo anterior. Representa la ecuación en diferencias.

$y\,(k) = x\,(k-1)$. Su representación es:




## Integrador de tiempo discreto (Discrete-Time Integrator)

(Para todo lo siguiente, conviene repasar el apéndice 'Transformada $z$'). Es un bloque que aproxima la integración en tiempo continuo por todos conocida. Responde a la siguiente aproximación:

$$y\left(k\right) = y\left(k-1\right) + \int_{T(k-1)}^{T\ k} u\left(t\right)\ \mathrm{dt}$$

donde $u\left(t\right)$ es la entrada al integrador, $y\left(k\right)$ es la salida y $T$ es el periodo de muestreo.

Hay tres implementaciones distintas de este componente: *Integración de Euler hacia adelante*, *Integración de Euler hacia atrás* e *Integración trapezoidal*. Las tres implementaciones aproximan el área bajo la curva como la suma de las áreas de un número finito de rectángulos.

**Euler hacia adelante**

Está basada en la aproximación $u\left(t\right) = u\left(T\left(k-1\right)\right)$. Así, Euler hacia adelante aproxima

$$y\left(k\right) = y\left(k-1\right) + T\ u\left(T\left(k-1\right)\right)$$

Calculando la transformada $z$ de la ecuación anterior, obtenemos

$$Y\left(z\right) = z^{-1}Y\left(z\right) + T\ z^{-1}U\left(z\right)$$

Reordenando los términos, obtenemos la función de transferencia del integrador hacia adelante de Euler:

$$\frac{Y\left(z\right)}{U\left(z\right)} = H\left(z\right) = \frac{T}{z-1}$$

**Euler hacia atrás**

La integración de Euler hacia atrás está basada en la aproximación $u\left(t\right) = u\left(T\ k\right)$. De este modo, Euler hacia atrás aproxima

$$y\left(k\right) = y\left(k-1\right) + T\ u\left(\mathrm{Tk}\right)$$

La correspondiente función de transferencia es:

$$\frac{Y\left(z\right)}{U\left(z\right)} = H\left(z\right) = \frac{\mathrm{Tz}}{z-1}$$

**Integración trapezoidad**

Se basa en la aproximación siguiente:

$$u\left(t\right)=\frac{u\left(\mathrm{Tk}\right)+u\left(T\left(k-1\right)\right)}{2}$$

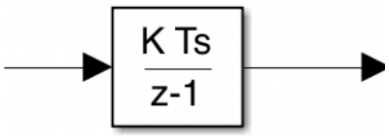
De esta manera, a integración trapezoidal aproxima

$$y\left(k\right)=y\left(k-1\right)+\frac{\mathrm{Tu}\left(\mathrm{Tk}\right)+\mathrm{Tu}\left(T\left(k-1\right)\right)}{2}$$

y su función de transferencia será

$$\frac{Y\left(z\right)}{U\left(z\right)}=H\left(z\right)=\frac{T\left(z+1\right)}{2\left(z-1\right)}$$

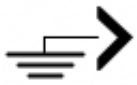
La representación del bloque de integración en tiempo discreto es el siguiente:




## Tierra (Ground Block)

El bloque Tierra puede utilizarse para conectar bloques cuyos puertos de entrada no están conectados a otros bloques. Si ejecutamos una simulación con bloques que tienen puertos de entrada desconectados, Simulink emite mensajes de advertencia. Podemos evitar los mensajes de advertencia utilizando los bloques de tierra. Así, el bloque Tierra emite una señal con valor cero. El tipo de datos de la señal es el mismo que el del puerto al que está conectado. Utilizándolo junto con el bloque Switch, puede ser útil para conectar o desconectar entradas según se cumpla o no una condición.

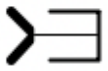
Su representación es:



## Terminador (Terminator)

El bloque Terminador puede utilizarse para tapar bloques cuyos puertos de salida no están conectados a otros bloques. Si ejecutamos una simulación con bloques que tienen puertos de salida no conectados, Simulink emite mensajes de advertencia. Podemos evitar los mensajes de advertencia utilizando los bloques Terminator. Por tanto, podemos decir que el bloque Tierra es a las entradas como el bloque Terminador es a las salidas.

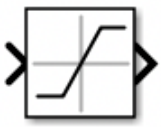
Su representación es:



## Bloque de saturación

Este bloque permite limitar una fuente de entrada entre dos valores (límite superior y límite inferior), de manera que aquellos valores de la entrada por encima o por debajo de dichos límites sean ignorados.

Su representación es la siguiente:



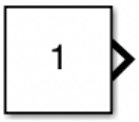
## Constante y Producto (Constant - Product)

El bloque Constante se utiliza para definir un valor constante real o complejo. Este bloque acepta una salida escalar (matriz 1x1 2-D), vectorial (matriz 1-D) o matricial

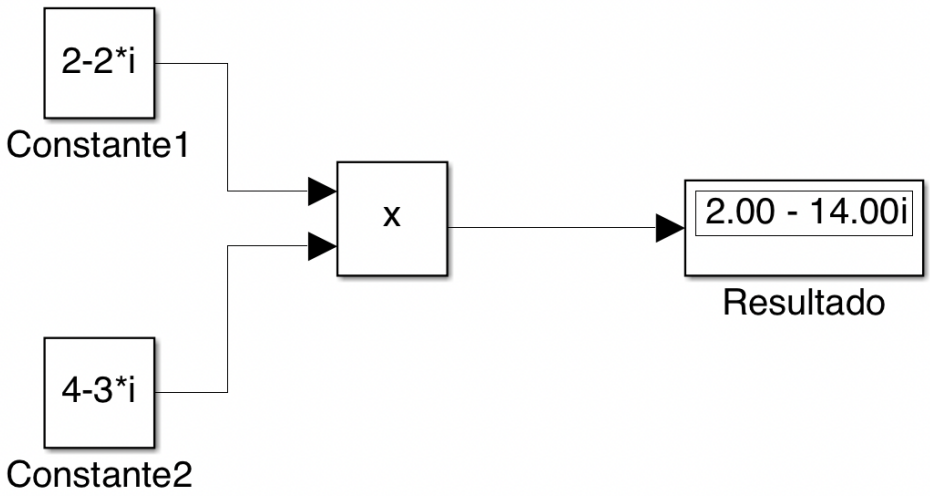
(matriz 2-D), dependiendo de la dimensión del parámetro Valor constante que especifiquemos y de la configuración de los parámetros del vector «Interpret» como parámetro 1-D. La salida del bloque tiene las mismas dimensiones y elementos que el parámetro «Constant Value». Si especificamos un vector para este parámetro, y queremos que el bloque lo interprete como un vector (es decir, un array 1-D), seleccionamos el parámetro «Interpreter» vector como parámetro 1-D; de lo contrario, el bloque trata el parámetro «Constant Value» como una matriz (es decir, un array 2-D).

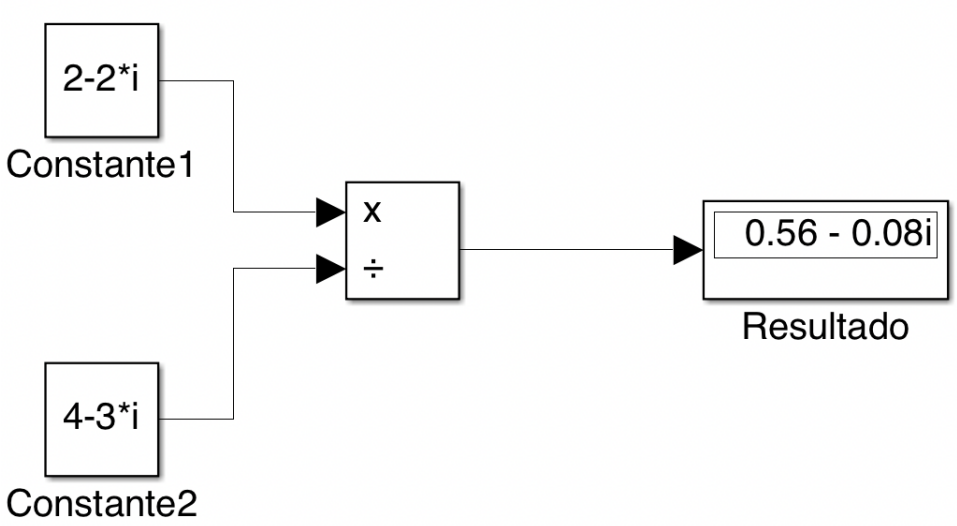
Por defecto, el bloque Constante da salida a una señal cuyo tipo de datos y complejidad son los mismos que los del parámetro de «Constant Value» del bloque. Sin embargo, podemos especificar que la salida sea cualquier tipo de datos soportado por Simulink, incluidos los tipos de datos de punto fijo. Para conocer los tipos de datos soportados por Simulink, consulte Tipos de datos soportados por Simulink en la documentación.

Su representación es:




El bloque Producto realiza la multiplicación o división de sus entradas. Este bloque produce salidas utilizando la multiplicación por elementos o por matrices, dependiendo del valor del parámetro Multiplicación. Las operaciones se especifican con el parámetro Número de entradas. Los caracteres Multiplicar(*) y Dividir(/) indican las operaciones a realizar en las entradas. En la figura se multiplican los números complejos expresados en Constante1 y Constante2. En la segunda, se dividen.

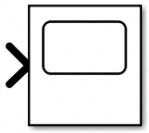





## Bloque «Scope»

El bloque «Scope» muestra las formas de onda como funciones del tiempo de simulación. El bloque «Scope» puede tener múltiples ejes y con un rango de tiempo común. Podemos ajustar la cantidad de tiempo y el rango de valores de entrada mostrados, podemos mover y cambiar el tamaño de la ventana del «Scope». Podemos tambiém modificar los valores de los parámetros de «Scope» durante la simulación. El bloque «Scope» no muestra automáticamente las formas de onda, pero sí escribe datos en los canales de entrada conectados. La señal o señales de entrada de «Scope» se mostrarán si después de terminar una simulación, hacemos doble clic en el bloque.

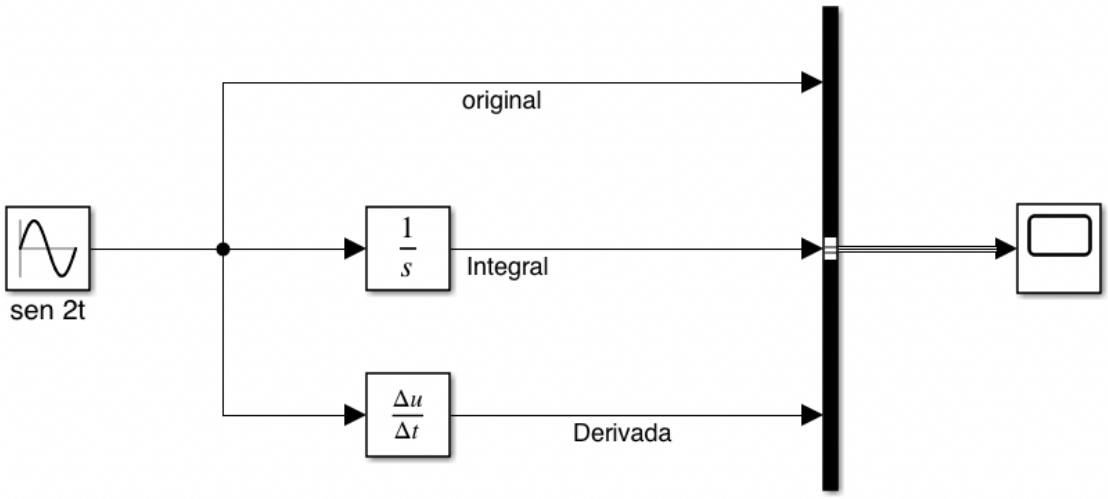
Su representación es:



## Bloque Bus (Creador y Selector)

Para entender los usos de los bloques Creador de Buses y Selector de Buses, repasemos el concepto de un bus de señal que puede ser considerado como un conjunto de varios cables unidos por bridas. Gráficamente, podemos representar un bus como una señal

compuesta formada por un conjunto de señales individuales con un grupo de líneas, cada línea representando una señal individual. Utilizamos los bloques Creador de Buses para crear buses de señales y los bloques Selector de Buses para acceder a los componentes de un bus. Simulink oculta el nombre de los bloques Creador de bus y Selector de bus cuando los copiamos de la biblioteca de Simulink a un modelo. Sin embargo, podemos hacer visibles los nombres haciendo clic en Mostrar nombre en el menú Formato. Se recomienda hacer visibles los nombres porque los bloques Creador de bus, Selector de bus, Mux y Demux están representados por una línea vertical gruesa (los bloques Mux y Demux se describen en la siguiente entrada). Los bloques Creador de Bus y Mux pueden, en la mayoría de los casos, ser utilizados de forma indistinta. Del mismo modo, los bloques Bus Selector y Demux pueden utilizarse indistintamente. Mientras que los bloques Bus Creator y Bus Selector se utilizan con señales de bus, los bloques Mux y Demux se utilizan con vectores. Un ejemplo:




## Operador relacional

El bloque Operador relacional realiza la comparación especificada de sus dos entradas. Seleccionamos el operador relacional que conecta las dos entradas con el parámetro «Operador relacional». El bloque se actualiza para mostrar el operador seleccionado. A continuación se enumeran las operaciones admitidas.

Descripción de la operación:

1: TRUE

0: FALSE

== 1, si la primera entrada es igual a la segunda

~= 1, si la primera entrada no es igual a la segunda

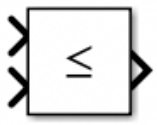
< 1, si la primera entrada es menor que la segunda

<= 1, si la primera entrada es menor o igual que la segunda entrada

>= 1, si la primera entrada es mayor o igual que la segunda entrada

> 1, si la primera entrada es mayor que la segunda entrada

Su representación es la siguiente:




## Operador lógico

El bloque Operador lógico realiza la operación lógica especificada en sus entradas. Un valor de entrada es TRUE (1) si es distinto de cero y FALSE (0) si es cero. La operación booleana que conecta las entradas se selecciona con la lista de parámetros del operador en el cuadro de diálogo «Parámetros» del bloque de funciones. El bloque se actualiza para mostrar el operador seleccionado. Las operaciones lógicas se indican a continuación.

Descripción de la operación:

AND TRUE si todas las entradas son TRUE

OR TRUE si al menos una entrada es TRUE

NAND TRUE si al menos una entrada es FALSE

NOR TRUE cuando ninguna entrada es TRUE

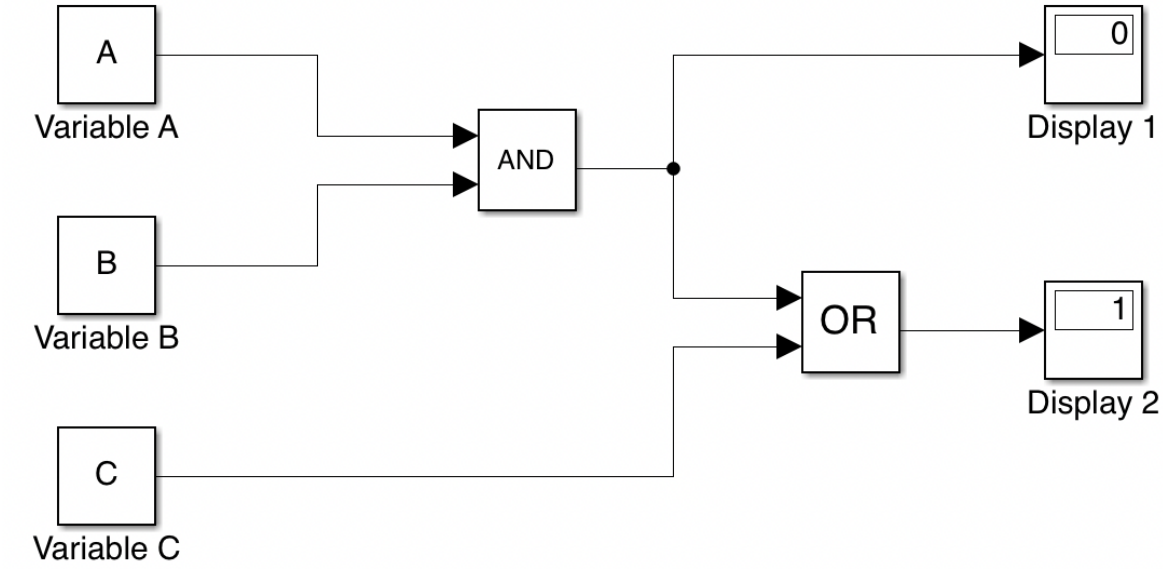
XOR TRUE si un número impar de entradas es TRUE

NOT TRUE si la entrada es FALSE y viceversa

El número de puertos de entrada se especifica con el parámetro «Número de puertos de entrada». El tipo de salida se especifica con los parámetros «Tipo de datos de salida». Un valor de salida es 1 si es TRUE y 0 si es FALSE.

Su representación es:

El modelo mostrado en la figura simula la expresión booleana D = A x B + C donde el signo (x) denota la intersección (AND) de las variables A , B , y el signo (+) denota la unión (OR) de A x B con C. Los bloques indicados como Variable A, Variable B y Variable C son bloques Constantes. Especificamos los valores A = 1, B = 0, y C = 1 en la ventana de comandos de MATLAB, y tras la ejecución, observamos los valores 0 y 1 en los bloques Display 1 y Display 2 respectivamente.




## MATLAB Function

Quizá sea este el bloque más versátil de cuantos se han estudiado. Permite, directamente, incluir en el modelo una función construída con MATLAB, lo que posibilita que podamos incluir casi cualquier cálculo.

Su uso es sencillo: se introduce en el modelo, se hace doble click sobre él y se abre el editor de MATLAB para que escribamos la función que necesitemos. Podemos incluir las entradas y salidas que necesitemos.

Su representación es la siguiente:


No es un bloque que tenga un gran rendimiento, ya que el modelo tendrá que llamar al interprete de MATLAB para evaluar la función, pero, en ocasiones, es la forma más

sencilla de pasar una función a Simulink.

## To Workspace

Este bloque nos permite recoger los resultados de la simulación en una variable que puede ser usada en el entorno de MATLAB, normalmente para obtener gráficos más sofisticados o para utilizarlos como entrada en un procesamiento posterior.

Los resultados se pueden recoger como un array, una serie temporal, una estructura o una estructura a la que se añade el tiempo.

Este bloque utiliza una estructura propia como salida «out.» a la que se añade el nombre de la variable que queramos utilizar. Así deberá ser usada en el entorno de MATLAB.

Su representación es la siguiente:


## From Workspace

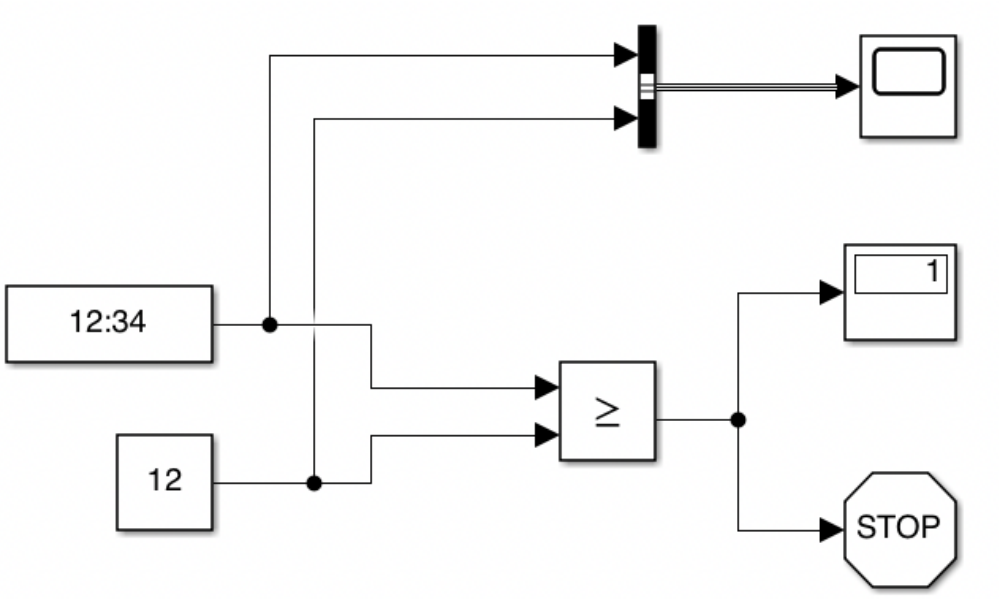
Lo mismo que con el bloque anterior, podemos utilizar una variable que haya sido definida previamente en el entorno de MATLAB de manera que pueda utilizarse para la simulación del modelo. El tipo de dato de la variable de entrada no hay que definirlo ya que se hereda del especificado en entorno de trabajo. Hay que asegurarse de que la variable ha sido previamente cargada antes de correr la simulación; en caso contrario se mostrará un error.

Su representación es la siguiente:


## Stop

Este bloque detiene la simulación cuando la entrada es distinto de cero. Se suele utilizar cuando se busca un resultado a lo largo del tiempo que cumple una determinada condición.

En el siguiente ejemplo, la simulación se para cuando el reloj digital ha generado una secuencia de tiempo mayor o igual de 12:




## Conclusión

En esta sección se han presentado los elementos más comunes a utilizar en la simulación de sistemas discretos. Hay muchos más, pero los presentados son los fundamentales. Se debe tener presente que los procesos de simulación que se están estudiando, aunque son sistemas discretos, no tratan del procesamiento digital de la señal (DSP), sino de situaciones comunes como las finanzas, poblaciones, producción, etc. Para los primeros, hay muchos más bloques interesantes, pero eso sería materia de otra asignatura. Para los segundos, es suficiente con los presentados aquí.