El Instrument Control Toolbox de Matlab admite dos formas de comunicarse con un instrumento:

-interface objects

-device objects

La que he estado usando hasta ahora es el interface object, que lo que hace es abrir un tipo de comunicación (gpib, visa, etc). Luego es responsabilidad del usuario enviar los comandos adecuados al instrumento conectado ahí. Esto lo he hecho usando instrumentos y comandos gpib para los HP y la electrónica magnicon. Para la PXI-5922 la secuencia de instrucciones funciona, pero no tengo una lista de comandos que acepte la tarjeta y con la que pueda comunicarme!

pxi=visa('ni','PXI9::14::INSTR')

fopen(pxi)

str=sprintf('%s\n','\*TST?')

query(pxi,str)

Esa secuencia supuestamente manda un self-test, pero da error, no reconoce ese comando. Sin embargo, sí que se crea y se abre el instrumento. En realidad existen instrumentos ‘comand’ based e instrumentos ‘register’ based. Estos son por ejemplo las tarjetas digitalizadoras VXI. Así que si esta PXI-5922 es register based tendría que comunicarme con ella de otra forma. En el manual del Instrument control toolbox menciona funciones de lectura/escritura en memoria de alto y bajo nivel. Necesitaría la lista y significado de los registros de la tarjeta. Hay un ejemplo en que usa “vxi.MemorySpace” pero el objeto pxi creado como ‘visa’ no tiene ese miembro.

La otra opción es usar un device object. Ahí no sólo se crea el interface, sino que hay que pasarle el driver del instrumento. En teoría la librería ‘niScope’ incluye la PXI-5922, por lo que estaría soportada. Las primeras pruebas me dieron problemas porque no conseguía generar el driver. Al parecer Matlab necesita un compilador asociado y este es el problema que he venido teniendo desde hace tiempo y que me ha impedido usar también el Simulink. Finalmente, a fecha 23Oct17 consigo resolver este problema siguiendo las instrucciones de una web.

(<https://es.mathworks.com/matlabcentral/answers/95039-why-does-the-sdk-7-1-installation-fail-with-an-installation-failed-message-on-my-windows-system>). Tengo que desinstalar todas las redistributable versions de VS C++ anteriores a 2010. Luego instalo .NET4.7.1 (le cuesta un rato) (fichero: NDP471-KB4033344-Web.exe). Luego el SDK7.1 (fichero: winsdk\_web.exe) y finalmente un patch (fichero: VC-Compiler-KB2519277.exe). Según las instrucciones después debía reinstalar las redistributables, pero me daba error. No obstante, al ejecutar Matlab parece que él solo instaló alguna y al ejecutar el mex –setup ya fue capaz de reconocer un compilador!

Una vez resuelto este problema pruebo a crear un objeto con ese driver. Pruebo a crear un driver de tipo niScope y conectarlo a la dirección de la tarjeta. El niScope es un IVI driver genérico. Matlab requiere crear un Matlab Wrapper driver, que son ficheros .mdd. Luego se usa ese driver para comunicarse con el instrumento.

niscope=makemid('niScope','niscope.mdd')

pxi=icdevice('niscope.mdd','PXI9::14::INSTR')

Sin embargo, la instrucción connect(pxi) da error. Se puede ver el .mdd y crea prototipos de muchas funciones, pero no hay una de init(). No está claro si hay que completar el driver o debería funcionar ya out-of-the-box.

Con la tmtool se puede inspeccionar el hardware y software instalados y se ve que no aparece la tarjeta PXI-5922 en el slot3. Sólo aparece el PXI-backplane. ¿Puede estar ahí el problema? En teoría debería poder listarse todo el hardware disponible vía software:

info=instrhwinfo('visa','ni')

info.ObjectConstructorName

ans =

'visa('ni', 'ASRL1::INSTR');'

'visa('ni', 'ASRL3::INSTR');'

'visa('ni', 'ASRL5::INSTR');'

'visa('ni', 'ASRL10::INSTR');'

'visa('ni', 'GPIB0::4::INSTR');

Se listan los dispositivos disponibles, pero no aparece la PXI.

Hay también una opción para configurar directamente un scope, con la función ‘oscilloscope(), que permite por ejemplo listar availableResources y al hacerlo sale el PXI0:1:Backplane, pero no la tarjeta. Intento conectarme con el backplane pero da error. La clave está por tanto en intentar que reconozca automáticamente la tarjeta. A partir de ahí, esa clase ofrece métodos para configurar los canales y adquirir waveforms, etc.

La tmtool también lista el PXI0::1::BACKPLANE pero no el puente ni la tarjeta.

Instalo matlab2015a pero tampoco aparece la tarjeta en la tmtool.

De NI me contestan que ellos no dan soporte para Matlab y que tienen que ser ellos los que resuelvan el problema. Pruebo a ‘eliminar’(cambiar nombre) a los pxiesys.ini y pxisys.ini. Se listan con el NI-MAX las tarjetas PXI pero no en la estructura del chasis PXI. Si dejo el pxiesys sí que aparece la estructura del chasis con el PXIe-8375 dentro, pero no la 5922. Al añadir el pxisys.ini sí aparece esta tarjeta. La tmtool muestra en todos los casos el PXI0::1::BACKPLANE.

Update 16-01-2018

Tras conversación con Nicolás de electrónica, me comenta la función ‘daqregister()’ del daq toolbox de Matlab, que es necesario ejecutar para instalar una tarjeta de adquisición de datos nueva. Además hay que ejecutarlo como Administrador. Una vez instalada, se puede usar ya por cualquier usuario. Ejecuto daqregister(‘nidaq’) y unos cuantas opciones más, pero no encuentra ningún .dll. Ejecuto daqhwinfo y obtengo un mensaje de error diciendo que la ‘legacy interface’ no está soportada en mi plataforma, sólo la ‘session-based’. Parece ser que hay dos formas diferentes de comunicarse con tarjetas de adquisición de datos, ‘Legacy Interface’ y ‘Session-Based Interface’ y para W10 64-bits sólo está disponible la segunda. Hay instrucciones para configurar una sesión de adquisición. Primero ejecutar daq.getDevices para ver los dispositivos disponibles. Al hacerlo no encuentra ninguno. Hay una lista de los dispositivos PXI soportados y no está la 5922 (<https://es.mathworks.com/hardware-support/nidaqmx.html> ). Parece que no es lo mismo una tarjeta digitalizadora que una tarjeta de adquisición de datos, por ejemplo la PXI-4461, que se usa en adquisición de audio o acelerómetros.

Ejecuto la aplicación ‘VISA Interactive Control’ y sólo reconoce el Backplane PXI, el chasis NI PXIe-1071, igual que la tmtool, aunque el NIMAX reconoce tanto el puente NI PXIe-8375 como la tarjeta PXI-5922 en los slots correspondientes. Sigue sin resolverse por tanto el problema.

28-Mayo-2018

Parece que he encontrado el problema!

>> makemid('niScope','pxi5922.mdd')

>> scope=icdevice('pxi5922.mdd', 'PXI1Slot3\_3')

>>connect(scope)

No tenía que poner el ‘VISA resource name’ sino directamente el nombre! Puedo activar el SoftFrontPannel y ver cómo se actualizan los cambios que voy tecleando en la línea de comandos! Esto lo he conseguido también en Python! Ejecuto get(scope) para tener acceso a todas las opciones y consigo cambiar el rango vertical o la impedancia con set(…). Ahora tengo que indagar en las opciones para ser capaz de configurarlo completamente, lanzar una adquisición y guardarla! Encuentro un ejemplo en (<https://es.mathworks.com/help/instrument/examples/read-waveform-data-from-keysight-dso-x-2002a-oscilloscopes-using-the-ivi-c-driver.html>) para configurar y adquirir, pero no funcionan la definición de Measurement de ese instrumento en nuestra PXI. Este otro ejemplo sí funciona, pero tengo que eliminar el simulation mode en la definición del scope (<https://es.mathworks.com/help/instrument/examples/fetch-waveforms-through-niscope-matlab-instrument-driver-in-simulation-mode.html>). También configuro con:

>> invoke(scope.Configurationfunctionsvertical, 'configurevertical', '1', Range, Offset, Coupling, ProbeAttenuation, false);

Y adquiero la forma de onda con:

>> [waveformArray, waveformInfo] = invoke(scope.Acquisition, 'fetch', channelList,TimeOut, numSamples, waveformArray, waveformInfo)

Donde he definido:

>> channelList=’0’, TimeOut=3, numSamples=1000,waveformArray=zeros(1,1000),waveformInfo=\*\*\*\*\*(el wfi es un struct, se inicializa como en un ejemplo):

for i = 1:1 %numChannels

waveformInfo(i).absoluteInitialX = 0;

waveformInfo(i).relativeInitialX = 0;

waveformInfo(i).xIncrement = 0;

waveformInfo(i).actualSamples = 0;

waveformInfo(i).offset = 0;

waveformInfo(i).gain = 0;

waveformInfo(i).reserved1 = 0;

waveformInfo(i).reserved2 = 0;

end



Fig1. Primera WaveForm de la PXI5922 adquirida con Matlab!!!!!!!!!!!!

Tengo que hacer >> invoke(scope.Acquisition, 'initiateacquisition') cada vez.

Creo un script que adquiere y pinta la f(t) y la FFT(w). Ahora sólo queda aprender a configurarla adecuadamente!

%pxiacqscript

for i=1:100

invoke(scope.Acquisition, 'initiateacquisition')

[waveformArray, waveformInfo] = invoke(scope.Acquisition, 'fetch', channelList,TimeOut, numSamples, waveformArray, waveformInfo);

psd=abs(fft(waveformArray)).^2;

subplot(2,1,1)

plot(waveformArray);

grid on

subplot(2,1,2)

loglog(psd)

grid on

pause(1)

end

Consigo configurar el record Length y sampling rate con el menú ‘hotizontal’. Pero ojo cuando llamo a ‘fetch’ con un nsamples más grande que el tamaño predefinido del waveformarray, peta Matlab. Hay que redefinir waveformarray cada vez.

Para generar espectros hasta 100KHz tengo que usar al menos SR=200KS/S y con ese SR, para llegar a 2Hz de freq minima tengo que poner un record length (nsamples) de 1e5. Con nsamples=2e5 conseguimos una fmin=1Hz. En ese caso el DT=1Hz, pero eso implica que en la primera década hay 10 puntos, en la segunda 100, y así sucesivamente, Se podría subsamplear el espectro para que ocupase mucho menos, como los tomados con el HP.

Voy creando drivers para la adquisición:



Ojo con el Record Length, si se pone uno ‘raro’ que no puede poner la tarjeta, pone el más cercano, pero al leer el campo ‘actual\_record\_length’ devuelve el que se había pasado, por lo que no es sencillo hacer un control de que el RL verdadero es correcto. Pasar valore razonables y testados!

En el ConfigureChannel puedo cambiar todos los parámetros de los canales, pero el Rango no responde bien al valor que se le pasa, y no he encontrado las funciones para leer esos datos de un canal concreto, por lo que tampoco se puede comprobar que se ha configurado bien, aunque este parámetro sólo es útil para ver el SFP, en adquisición pintamos los datos a parte.

Para acceder a parámetros de cada canal hay otro grupo de funciones de más bajo nivel:

>> invoke(scope.Configurationfunctionssetgetcheckattributegetattribute,'getattributevireal64','0',1250001)

El string ‘0’ o ‘1’ es el canal, y el ‘1250001’ es el AttributeID, que mirando en el menu ‘Vertical’ corresponde al Rango. Pero lo que devuelve no es directamente el rango, sino algún código. Falta depurar. Para cambiar el valor se invoca cambiando ‘get’ por ‘set’ y añadiendo el parametervalue adecuado como input parameter extra.

La forma de utilizar el get\_Vertical\_Range es seleccionando antes de qué canal se quiere, y para eso hay que actualizar un miembro de la estructura scope llamado ‘RepCapIdentifier‘

Haciendo:

>> scope.RepcapIdentifier=’0’

>> get(scope.Vertical,'Vertical\_Range')

Ahora ya no da error y devuelve el rango del canal seleccionado, pero sólo aparecen 2 valores, ‘2’ cuando el rango es 0.2V y ‘10’ cuando es 1.2V y no permite cambiar a otros. No está claro cómo funciona esta opción.

Ojo porque cuando luego ejecuto el script, tener activado el RepCapIdentifier da error, hay que volver a poner RepCapIdentifier=’’;

He creado funciones para adquirir y salvar el PSD y pulsos. Uso también la fte magnicon con un SQUID frio, para testar el trigger. Creo función de configuración, pero resulta un tanto oscuro cómo configurar correctamente el Type y source.

El ‘Type’ es un enum con estos posibles valores (double):

|  |
| --- |
| 1004.0 |
| 1001.0 |
| 5.0 |
| 1002.0 |
| 1003.0 |
| 1.0 |
| 6.0 |

El source es un enum con estos posibles valores (string):

|  |  |
| --- | --- |
| NISCOPE\_VAL\_SW\_TRIG\_FUNC | VAL\_SW\_TRIG\_FUNC |
| NISCOPE\_VAL\_PFI\_2 | VAL\_PFI\_2 |
| NISCOPE\_VAL\_RTSI\_6 | VAL\_RTSI\_6 |
| NISCOPE\_VAL\_PFI\_1 | VAL\_PFI\_1 |
| NISCOPE\_VAL\_RTSI\_5 | VAL\_RTSI\_5 |
| NISCOPE\_VAL\_PFI\_0 | VAL\_PFI\_0 |
| NISCOPE\_VAL\_RTSI\_4 | VAL\_RTSI\_4 |
| NISCOPE\_VAL\_RTSI\_3 | VAL\_RTSI\_3 |
| NISCOPE\_VAL\_RTSI\_2 | VAL\_RTSI\_2 |
| NISCOPE\_VAL\_RTSI\_1 | VAL\_RTSI\_1 |
| NISCOPE\_VAL\_RTSI\_0 | VAL\_RTSI\_0 |
| NISCOPE\_VAL\_EXTERNAL | VAL\_EXTERNAL |
| NISCOPE\_VAL\_IMMEDIATE | VAL\_IMMEDIATE |
| NISCOPE\_VAL\_PXI\_STAR | VAL\_PXI\_STAR |

Parece que Source tiene valores diferentes según sea el Type. He identificado:

-1001.0: hysteresis

-1002.0:digital

-1003.0: window

-1.0: edge

-6.0: immediate.

Si type=6 entonces source=’NISCOPE\_VAL\_IMMEDIATE’, pero si type=1, entonces source=’0’o ‘1’. Pero esto es al leerlo, si lo intento escribir así, da error. El type lo pone bien en modo Edge, el slope y el level lo escribe bien también, sólo falla el source. En realidad hay que hacerlo invocando una función ‘configuretriggeredge’ en el bloque ‘Configurationfunctionstrigger’. Hay funciones para configurar el resto de tipos de trigger.

Update Nov24

La adquisición evento a evento es muy lenta y descubro que hay un modo multi-record. Hay una variable en ‘horizontal‘ al igual que el RL y SR que es ‘Number\_of\_records’. En un primer intento, cambio sin más esta variable y parece que la variable data que devuelve el GetWaveForm tiene un tamaño que es RL\*NumRecords. Intento jugar con ello y no consigo reproducir ese comportamiento, sino que peta Matlab. Curiosamente, después de esto, al hacer adquisición de baselines, parece que la acq se ha ralentizado mucho. Antes tomaba casi 10seg (que seguía siendo lento) y después toma 2-3/seg, que es anormalmente lento. Veo que esto no depende del RL, así que parece que he tocado algo internamente que está enlenteciendo la acq.(\* probablemente era simplemente que tenía activado el boolplot. Con boolplot=0 va casi 3 veces más rapido) No obstante, jugando con el NumRecords, debería poder adquirirse mucho más rápido. Otro truco sería tomar una captura muy larga y partirla después en N del tamaño requerido. Para baselines puede funcionar, pero para pulsos, es mas problemático.