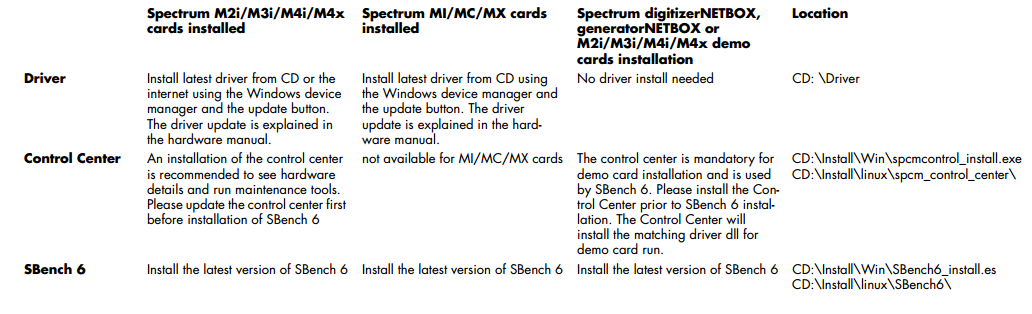
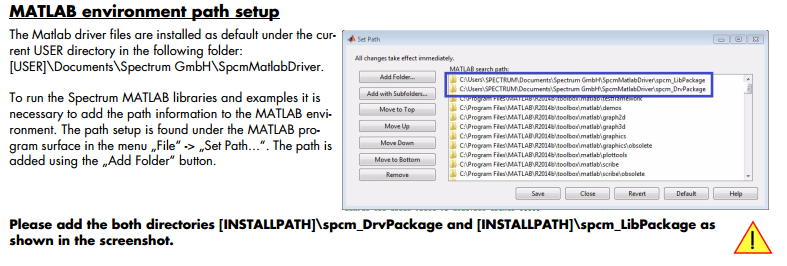
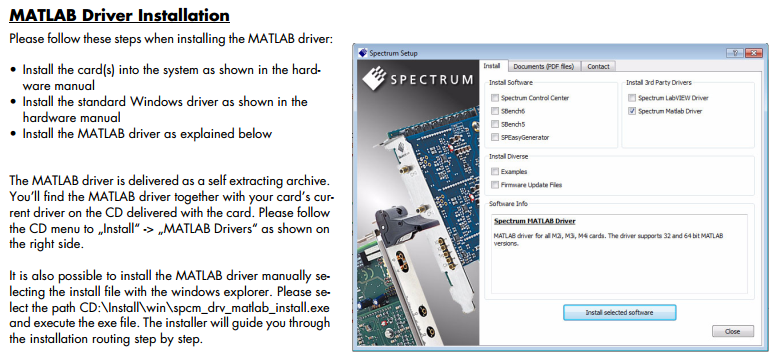
Install：

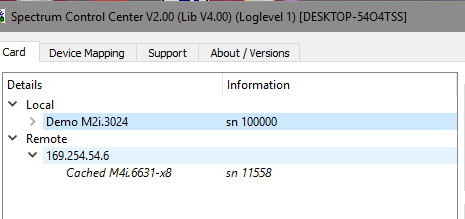


See manual: basically Bench6 and Driver

Install matlab library, and set the environment



First of all, open the control center to connect the hardware



And get the ip 169.254.54.6

Programming and use matlab to control it :

You can write your own program at basis of any one example program

**The example of a program**

1. connect to the ip of hardware

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*connect order \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% helper maps to use label names for registers and errors

mRegs = spcMCreateRegMap ();

mErrors = spcMCreateErrorMap ();

% \*\*\*\*\* use device string to open single card or digitizerNETBOX \*\*\*\*\*

% digitizerNETBOX

deviceString = 'TCPIP::169.254.54.6::inst0'; % XX.XX.XX.XX = IP Address, as an example : 'TCPIP::169.254.119.42::inst0'

[success, cardInfo] = spcMInitDevice (deviceString);

% \*\*\*\*\* init card and store infos in cardInfo struct \*\*\*\*\*

%[success, cardInfo] = spcMInitCardByIdx (0);

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*connect over \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Please Add the paragraph above into your program to connect your PC with netbox. Where devicestring is the ip of netbox.

2. function of spcm hopen

This function initializes and opens an installed card supporting the new SpcM driver interface.



Spcm\_hopen(ip addresses )

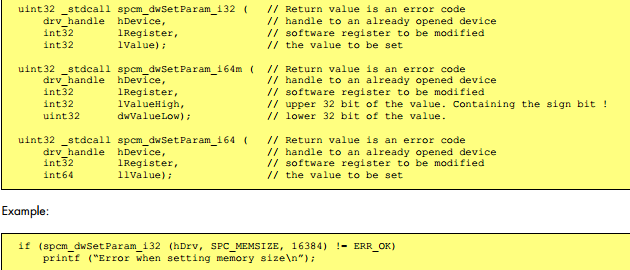
3. function of close

Same as fopen

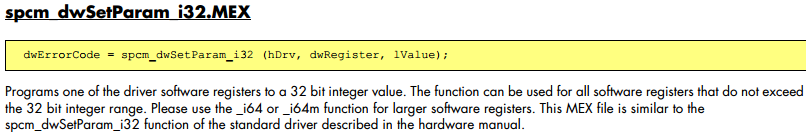


4. setting parameter： functions spcm\_dwSetParam

set the parameter of hardware by software registers



The command in matlab is



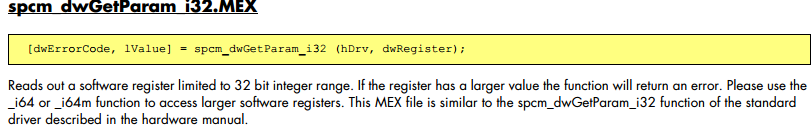
5. get the parameter of hardware: functions spcm\_dwGetParam



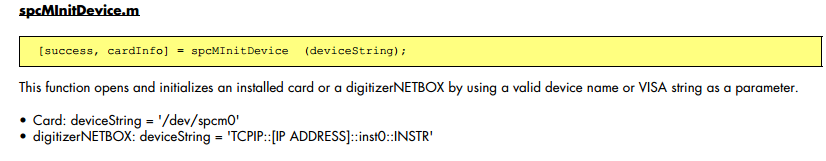
Where hDrv is the error code

Iserialnumber is the code of the parameter you want to get

The command in matlab is



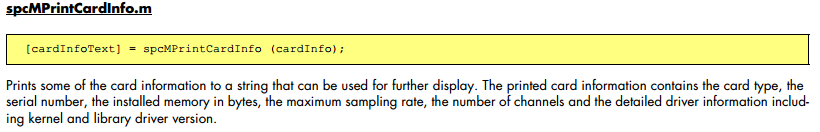
6. The integrated matlab program of initialization, read the parameter from card



See the introduction of spcMInitDevice, used for initialization of hardware

As for more details, see the script of spcMInitDevice-explain.m

6. The integrated matlab program of initialization, witer these information to the card



See more details and explains in spcMPrintCardInfo.m

Basically print the type of the card , vision

For example of print

M4i.6631-Exp sn 11558

Installed memory: 4096 MByte

Max sampling rate: 1250.0 MS/s

Channels: 2

Kernel Version: 1.20 build 13676

Library Version: 4.0 build 13877

7. check whether the hardware you use can support the operation in the next

USING the ((cardInfo.cardFunction ~= mRegs('SPCM\_TYPE\_AO')) & (cardInfo.cardFunction ~= mRegs('SPCM\_TYPE\_DO')) & (cardInfo.cardFunction ~= mRegs('SPCM\_TYPE\_DIO')))

spcMErrorMessageStdOut (cardInfo, 'Error: Card function not supported by this example\n', false);

return;

8. select the operation you want(1,2,3…) , use the parameter replaymode to achieve the selection

For example:

fprintf ('\nPlease select the output mode:\n');

fprintf (' (1) Singleshot\n (2) Continuous\n (3) Single Restart\n (0) Quit\n');

replayMode = input ('Select: ');

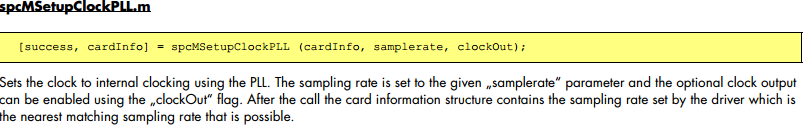
if (replayMode < 1) | (replayMode > 3)

spcMCloseCard (cardInfo);

return;

end

9. set the clock PLL



Use the internal clock using PLL, you can also use the external clock message

See more detail in spcMsetupClockPLL.m

For example:

fprintf ('\n ..... Sampling rate set to %.1f MHz\n', cardInfo.setSamplerate / 1000000);

..... Sampling rate set to 50.0 MHz

10. set the channel

if cardInfo.maxChannels == 64

chMaskH = hex2dec ('FFFFFFFF');

chMaskL = hex2dec ('FFFFFFFF');

else

chMaskH = 0;

chMaskL = bitshift (1, cardInfo.maxChannels) - 1;

end

11. the main part------select the relay mode

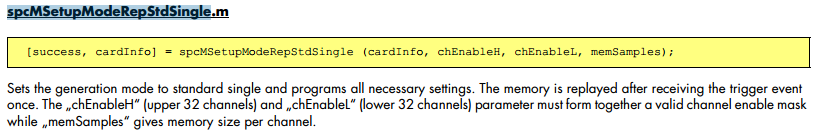
For the example of rep\_std\_single\_work.m

1. Mode 1 single shot

Frist of all,

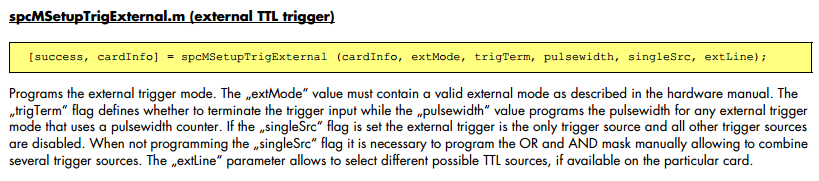
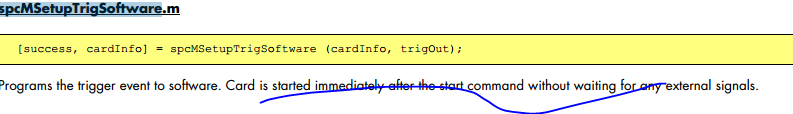
set the parameter: SPC\_LOOPS, SPC\_MEMSIZE，SPC\_CHENABLE

for the mode we pick



Secondly

Set the software trigger (can used internal TTL or External )

If use external TTL single: the parameter will be used: extMode, trigTerm, pulsewidth, singleSrc, extLine. You can write a program to generate a sequence of these five parameter.

extMode :value must contain a valid external mode as described in the hardware manual

trigTerm: flag defines whether to terminate the trigger input

pulsewidth : value programs the pulsewidth for any external trigger mode that uses a pulsewidth counter.

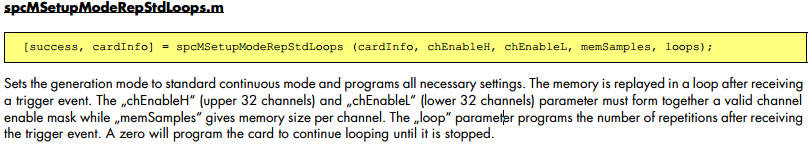
singleSrc: is set the external trigger is the only trigger source and all other trigger sources are disabled

singleSrc: is necessary to program the OR and AND mask manually allowing to combine several trigger sources.

extLine: parameter allows to select different possible TTL sources, if available on the particular card.

1. Mode 2 continuous mode

Fristly



[success, cardInfo] = spcMSetupModeRepStdLoops (cardInfo, chEnableH, chEnableL, memSamples, loops)

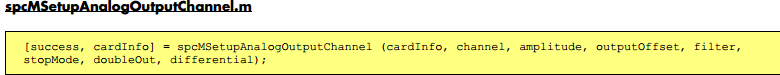
Memsamples is usually 64 \* 1024, the size of memory

Loops is usually 0 continue looping until it is stopped

Secondly, also do a software trigger as same as before.

12. the main part-------output -------generator card setup

When cardfunction is mRegs('SPCM\_TYPE\_AO')



channel：Does a setup for one analog output channel in ith channel. And Channel numbering start from 0!

Amplitude: defines the output amplitude in mV

OutputOffset: defines the output offset of the channel, the filter defines the code of the output filter to be used.

stopMode, doubleOut and differential are special functions of the generator

usually we use the parameter like this:

[success, cardInfo] = spcMSetupAnalogOutputChannel (cardInfo, i, 1000, 0, 0, 16, 0, 0);

% 16 = SPCM\_STOPLVL\_ZERO, doubleOut = disabled, differential = disabled

1. Firstly check all parameter is right
2. Check the hardware is ok
3. Enable the output channel

13.the main part——— digital acquisition card setup

case { mRegs('SPCM\_TYPE\_DO'), mRegs('SPCM\_TYPE\_DIO') }



group: Does a setup for one digital output channel group which must be specified. The difference between 12 is that group of channels can be set. Channel numbering start from 0!

digital input or i/o card can have different setup for a group of channels.

StopMode: defines the behaviour between outputs, details are described in the hardware manual.

LowLevel, and highLevel: define the output low and high levels in mV.

DiffMode: flag enables a special differential output mode which is described

14. The important part-----------calculate waveforms

We use this function to calculate waveforms and output it.

(1)if cardInfo.cardFunction == mRegs('SPCM\_TYPE\_AO')

% \*\*\*\*\* calculate waveforms \*\*\*\*\*

if cardInfo.setChannels >= 1

% ----- ch0 = sine waveform -----

[success, cardInfo, Dat\_Ch0] = spcMCalcSignal (cardInfo, cardInfo.setMemsize, 1, 1, 100);

if cardInfo.setChannels >= 2

% ----- ch1 = rectangle waveform -----

[success, cardInfo, Dat\_Ch1] = spcMCalcSignal (cardInfo, cardInfo.setMemsize, 2, 1, 100);

if cardInfo.setChannels == 4

% ----- ch2 = triangle waveform -----

[success, cardInfo, Dat\_Ch2] = spcMCalcSignal (cardInfo, cardInfo.setMemsize, 3, 1, 100);

]

% ----- ch3 = sawtooth waveform -----

[success, cardInfo, Dat\_Ch3] = spcMCalcSignal (cardInfo, cardInfo.setMemsize, 4, 1, 100);

]

% spcMCalcSignal:

% Calculates waveform data

% shape: 1 : sine

% 2 : rectangel

% 3 : triangel

% 4 : sawtooth

[success, cardInfo, signal] = spcMCalcSignal (cardInfo, len, shape, loops, gainP)

Where singal of ch0-3 is the most important parameter, first pick different shape for different channel.

The core code of waveform calculation:

block = len / loops;

% len is the total length of time during operation, loops is the number of circle

blockHalf = block / 2;

% half time of each loop

sineXScale = 2 \* pi / len \* loops;

% the angle speed omega for periodic function of wave

span = maxFS - minFS;

% the amplitude of each kind waveform

for i=1 : len

posInBlock = mod (i, block);

% the position of this moment in a block(loop )

switch shape

% ----- sine -----

case 1

signal (1, i) = scale \* sin (sineXScale\*i);

% ----- rectangel -----

case 2

if posInBlock < blockHalf % the + - half period

signal (1, i) = maxFS;

else

signal (1, i) = minFS;

end

% ----- triangel -----

case 3

if posInBlock < blockHalf

signal (1, i) = minFS + posInBlock \* span / blockHalf;

else

signal (1, i) = maxFS - (posInBlock - blockHalf) \* span / blockHalf;

end

% ----- sawtooth -----

case 4

signal (1, i) = minFS + posInBlock \* span / block;

end

end

1. Else cardFunction <> mRegs('SPCM\_TYPE\_AO')

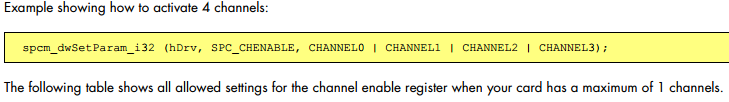
% ----- digital data -----

[success, Data] = spcMCalcDigitalSignal (cardInfo.setMemsize, cardInfo.setChannels);

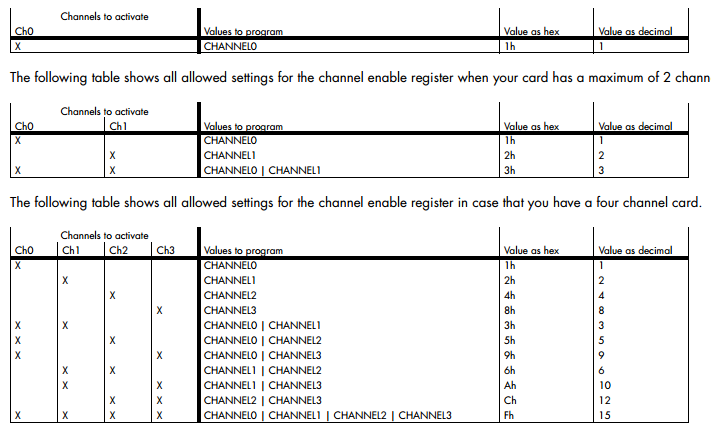
errorCode = spcm\_dwSetRawData (cardInfo.hDrv, 0, length (Data), Data, 1)

refresh the channel

15. wait until it ends



Active the channel by command mask



16. 判断是否end

**库函数说明**