LECOEUR ELECTRONIQUE

19, rue de Courtenay
45220 CHUELLES   - FRANCE -
Tel : +33 (0)2 38 94 28 30
Fax : +33 (0)2 38 94 29 67
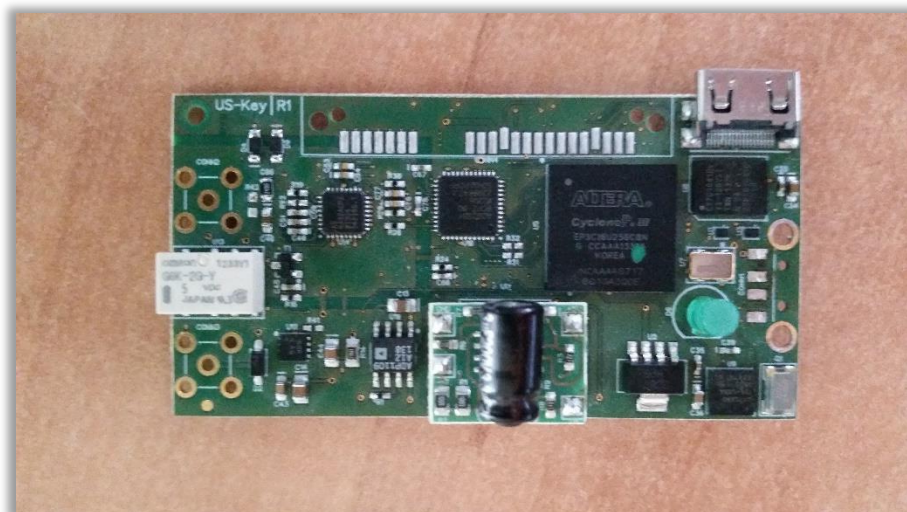
@   *mailto:info@lecoeur-electronique.com*

🌐 *http://www.lecoeur-electronique.com*

Rédacteur	S. BERTOLOTTO / JM.LECOEUR
Date	05/04/2015

US-SPI

Révision 1 le 5 Avril 2015

1   *DESCRIPTION*

US-SPI is our new generation ultrasound devices with a single channel to transmit and receive ultrasonic waves.

All functions are available through a SPI bus at 1MHz. This make the device compatible with any OS or embedded processor.

A single 5 Volts power supply is necessary.

It's very small size and its advanced technology allow to have a unique product for more applications like medical ultrasound imaging, the NDT and also for the research and university.

The transmitter can generate pulses with a voltage level and a width programmed by the user.

A low noise preamplifier combined to a VGA gives a gain range between 0 and 80 dB, a DAC curve is also available.

The bandwidth of the receiver is 0.5 MHZ to 18 MHZ

A 12 bits analog digital converter with a sampling frequency of 80 MHz is used to digitize ultrasound signals (8 bits are only available in the first version).

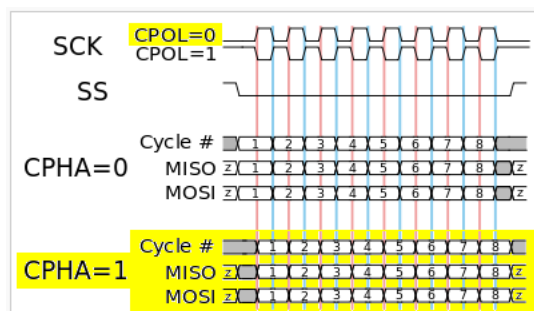The device has 2 working modes: Transmission or Reflection.

2   HARDWARE
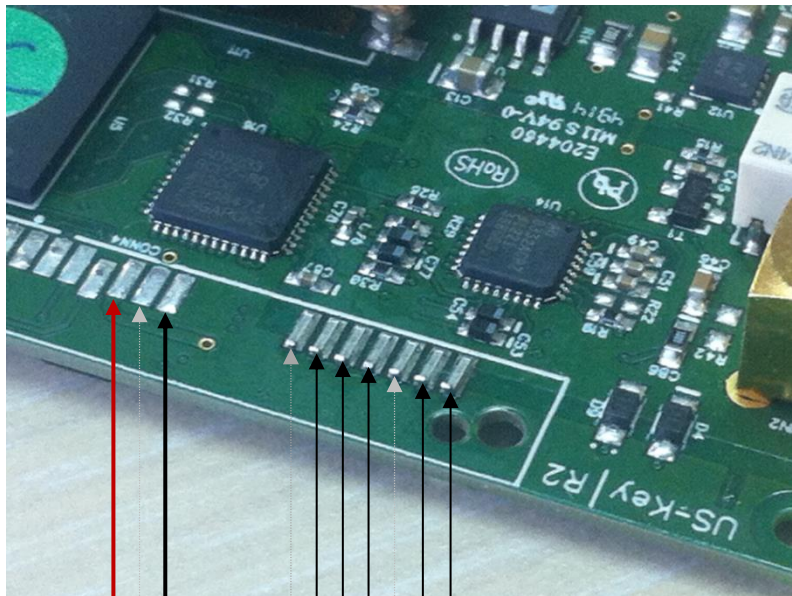
10	+5VDC 500mA (IN)
9	+3.3 (out)
8	GND
7	X (not wired)
6	SCLK
5	Data MOSI
4	Data MISO
3	X (not wired)
2	/Enable MISO (SS1)
1	/Enable MOSI (SS0)

ATTENTION : Maximum input voltage for all logic signals is 3.3V

3   SPI CONFIGURATION

SPI Mode	Clock Polarity (CPOL/CKP)	Clock Edge (CKE/NCPHA)
0	0	1
1	0	0
2	1	1
3	1	0

A timing diagram showing clock polarity and phase. The red vertical line represents CPHA=0 and the blue vertical line represents CPHA=1

4 SOFTWARE

To program US-Key SPI it's necessary to send up to 5 byte.
1 Byte (the last one to send) is a function number that you want to program (CMDSPI).
Depends of this function, between 1 to 4 other byte should be sent.

CMDSPI		Byte 1	Byte 2	Byte 3	Byte 4
Value	Function				
0	Gain (1)	Gain [7..0]	Gain[9..8]	Address[7..0]	-
1	-	-	-	-	-
2	Sampling request (2)	-	-	-	-
3	Compression	Factor[7..0]	-	-	-
4	Sampling request #1 (3)	X LSB	X MSB	-	-
5	Scale delay (4)	LSB	MSB	-	-
6	High voltage (5)	Voltage [7..0]	-	-	-
7	Pulse width (6)	Width[7..0]	-	-	-
8	PRF (7)	PRF[7..0]	PRF[15..8]	PRF[19..16]	-
9	Mode (8)	0/1	-	-	-
10	Scale (9)	Scale [7..0]	Scale[15..8]	-	-
11	ONOFF DAC	0/1	-	-	-
12	Echo-start (10)	Position[7..0]	Position[15..8]	Width[7..0]	Width[15..8]
13	Echo-start (11)	Polarity [7..0]	-	-	-
14	Filter	Filter[1..0]	-	-	-

(1)   Gain [9..0] → 0 dB = 65 and then +10/dB up to 80dB=865
        Address [7..0] → Only when DAC curve is on to program 256 gain values by step of 650ns
(2)   When you send CMDSPI=2, the US-Key will store the current A-scan inside its FIFO
(3)   When you will read the Xth samples, the US-Key will store AUTOMATICALLY the current A-scan inside its FIFO
(4)   Scale delay in step of 25ns
(5)   Voltage[7..0]=(98/180)*X+81.777   230V<X<10V
(6)   Pulse width by step of 6.25ns. If value=0 then width=18ns   up to 18+(255*6.25)=1600ns
(7)   PRF=Pulse Repetition Frequency → Period in step of 1.6µs
(8)   Mode 0=Pulse Echo / 1 = pitch & catch
(9)   Scale of the A-scan in step of 25ns
(10)   Echo-start=0 !!!!
(11)   Echo-start=0 !!!!
(12)   Filter → 0=1.25MHz,   1=2.5MHz,   2=5MHz,   3=10MHz,   4=No filter

5 *SOFTWARE PROGRAMMING*

PROGRAMMING US-SPI

This example is written in Basic language and given as a generic code to start development in other languages

In this language the access to SPI is done using this instruction:

spi.WriteRead(slave Select,wBuffer,nb of Byte To write,total number of Byte In transaction,rBuffer,number of Byte To read)

Where:

<u>Slave Select</u> is the number of the slave (0,1 is this example SS0 is the slave that write into US-SPI and SS1 is the slave that read US-SPI.

<u>wBuffer</u> is the buffer which will be written in US-SPI (bytes)

<u>nb of Byte to write</u> is the number of bytes to write from the wBuffer

<u>number of Byte in transaction</u> is the addition of read and write number of bytes.

<u>rBuffer</u> is the buffer which will be filled by the bytes red in US-SPI

<u>number of Byte to read</u> is the number of bytes to read from US-SPI and store in rBuffer

Example :

```
spi.WriteRead(0,wBuffercmd,3,3,rBuffercmd,0)
```

Write 3 bytes in US-SPI from wBuffercmd

```
spi.WriteReadasync(1,wBuffer,0,64,rBuffer,64)
```

Read 64 bytes from US-SPI and write them in rBuffer

1) Gain programming

"Gainc" contains the value of gain to be sent to US-SPI (0 to 80 db)

```
calculation_gain = gainc*(875-65)/80+65
```

```
wBuffercmd(0)=Floor(calculation_gain/256) 'msb
wBuffercmd(1)=calculation_gain-Floor(calculation_gain/256)*256 'lsb
wBuffercmd(2)=0 'selection gain
spi.WriteRead(0,wBuffercmd,3,3,rBuffercmd,0)
```

NOTES :

2) Delay programming

"Delayc" contains the value of the delay (beginning of sampling windows the width of the sampling windows is fix at 200 microseconds)

```
calculation_delay = delayc / 0.025 ' 25 nS step

wBuffercmd(0)=Floor(calculation_delay/256) 'msb
wBuffercmd(1)=calculation_delay-Floor(calculation_delay/256)*256 'lsb
wBuffercmd(2)=5 ' delay selection
spi.WriteRead(0,wBuffercmd,3,3,rBuffercmd,0)
```

<u>NOTES :</u>

3) Compression factor programming

"Consfactorcomp" contains the compression factor

```
wBuffera(0)=consfactcomp
wBuffera(1)=3
spi.WriteRead(0,wBuffera,2,2,rBufferl,0)
```

The compression factor is used to modify the sampling frequency:

Compression factor =0 -> sampling frequency 80 MHz
Compression factor =1 -> sampling frequency 40 MHz
Compression factor =2 -> sampling frequency 20 MHz
……………..

The compression factor does not work as a simple sampling frequency divider. It returns the maximum amplitude of the echo in the sampling period.

NOTES :

4) Transmitter voltage

Tensionc contains the voltage of the transmitter pulse (between 10 and 250 Volts)

```
wBuffera(0)=tensionc*(98/180)+81.7 'scalling
wBuffera(1)=6
spi.WriteRead(0,wBuffera,2,2,rBufferl,0)
```

NOTES :

5) Transmitter pulse width

Conslargeur contains the frequency of the transmitter pulse in MHZ (between 1MHz and 20 MHz)

```
nb= (1000/(2*conslargeur)-27)/6.5 'convert the pulse frequency in width and
scale it

wBuffera(0)=nb
wBuffera(1)=7
spi.WriteRead(0,wBuffera,2,2,rBufferl,0)
```

NOTES :

6) Repetition frequency (PRF)

Consfreqrec contains the frequency of the pulse repetition (between 100Hz and 2 KHz)

```
consfreqrec = (1000000/consfreqrec)/0.025

wBuffera(0)=Floor(consfreqrec/65536)
wBuffera(1)=Floor((consfreqrec-Floor(consfreqrec/65536)*65536)/256)
wBuffera(2)=consfreqrec-Floor((consfreqrec Floor(consfreqrec/65536)*65536)/256)*256
wBuffera(3)=8
spi.WriteRead(0,wBuffera,4,4,rBufferl,0)
```

NOTES :

7) Receiver Filter

f contains the frequency of the filter (0=1.25MHz, 1=2.5MHz, 2=5MHz, 3=10MHz, 4=No filter)

```
wBuffercmd(0)=f
wBuffercmd(1)=14
spi.WriteRead(0,wBuffercmd,2,2,rBuffercmd,0)
```

NOTES :

8) Transmission/ reflexion (single or dual crystals)

```
If consreftrans =True Then

    wBuffera(0)= 0
    wBuffera(1)=9
    spi.WriteRead(0,wBuffera,2,2,rBufferl,0)'single crystal
    Else

    wBuffera(0)= 1
    wBuffera(1)=9
    spi.WriteRead(0,wBuffera,2,2,rBufferl,0)'dual crystals

End If
```

NOTES :

9) Some initializations that must be done (not documented) for futures functions

Consechascan = 4000

```
wBuffercmd(0)=Floor(consechascan/256) 'msb
wBuffercmd(1)=consechascan-Floor(consechascan/256)*256 'lsb
wBuffercmd(2)=10 'selectionechascan
spi.WriteRead(0,wBuffercmd,3,3,rBuffercmd,0)

    pos=0,duree=10000,polarite=50

    wBuffera(0)=Floor(duree/256)
    wBuffera(1)=duree-Floor(duree/256)*256
    wBuffera(2)=Floor(pos/256)  'msb
    wBuffera(3)=pos-Floor(pos/256)*256
    wBuffera(4)=12
    spi.WriteRead(0,wBuffera,5,5,rBufferl,0)

    wBuffera(0)=polarite
    wBuffera(1)=13
    spi.WriteRead(0,wBuffera,2,2,rBufferl,0)'polarite


    wBuffera(0)=0
    wBuffera(1)=11
    spi.WriteRead(0,wBuffera,2,2,rBufferl,0)

    nbpts=4000

    wBuffercmd(0)=Floor(nbpts/256)
    wBuffercmd(1)=nbpts-Floor(nbpts/256)*256
    wBuffercmd(2)=4
    spi.WriteRead(0,wBuffercmd,3,3,rBuffercmd,0)
```

NOTES :

10) Signal (RF) acquisition

The more rapid method to acquire HF raw of samples is to set parameters described at points 1 to 9. Then you ask for a digitalization using:

```
wBuffera(0)=2
spi.WriteRead(0,wBuffera,1,1,rBufferl,0)'Initialise a new digitalisation
```

Then after a time value of PRF period the raw is available in a FIFO inside US-SPI, you just have to read them using:

```
spi.WriteReadasync(1,wBuffer,0,your nb of samples,rBuffer, your nb of samples)
```

rBuffer is filled with the RF raw :

header 1 : 101
header 1 : 101
Sample 1 : xx
Sample 2 : xx
Sample 3 : xx
Sample 4 : xx
…

Samples values are binary 0 to 255.

NOTES :