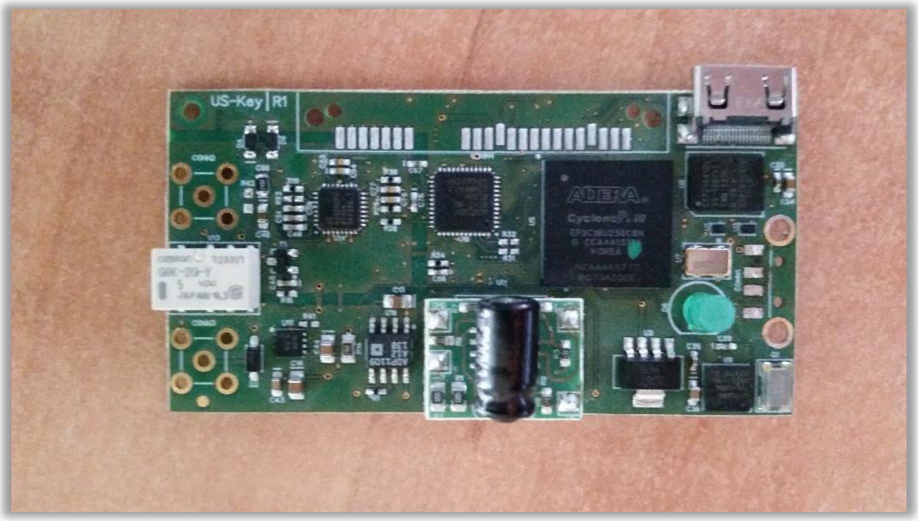
LECOEUR ELECTRONIQUE
19, rue de Courtenay
45220 CHUELLES - FRANCE -
Tel : +33 (0)2 38 94 28 30
Fax : +33 (0)2 38 94 29 67

mailto:info@lecoeur-electronique.com
http://www.lecoeur-electronique.com

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

US-SPI

Révision 3.1 le 28 avril 2016



1 ***DESCRIPTION***

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

<u>All functions are available through a SPI bus at 1MHz</u>. This makes 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 allows having 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.

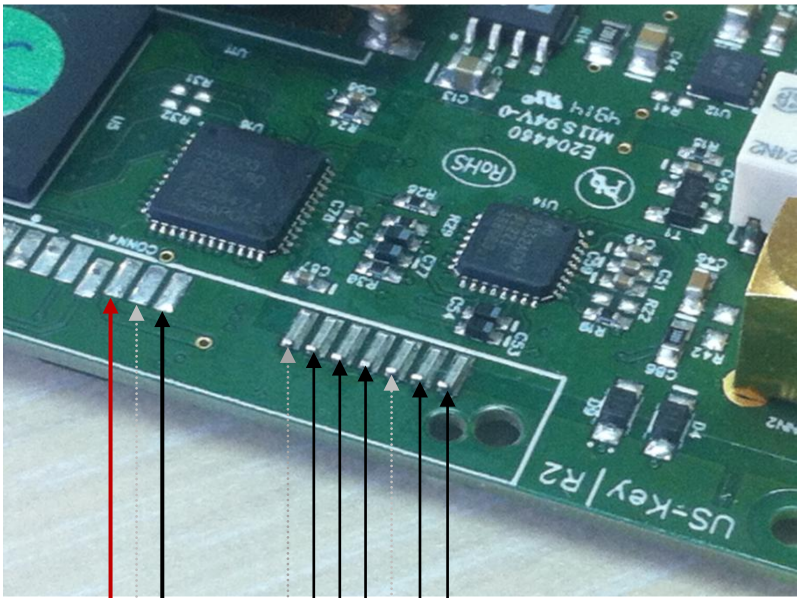
A low noise preamplifier combined to a VGA gives a gain range between 0 and 80 dB, a DAC curve is also available.

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

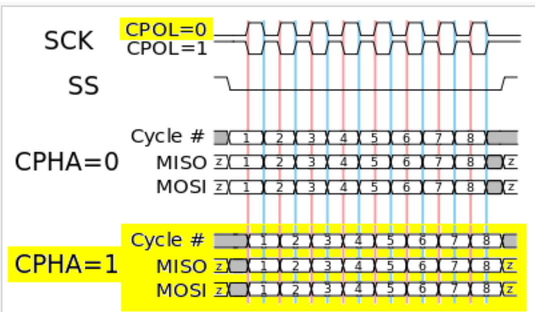


10	+5VDC 500mA (IN)
9	+3.3 (out)
8	GND
7	X (not wired)
6	SCLK
5	Data MOSI
4	Data MISO
3	X (not wired)
2	/Enable MISO (SS1)
1	/Enable MOSI (SS0)

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=ON /1=OFF!!**	-	-	-
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[2..0]	-	-	-
15	PosWidth G1	Position[7..0]	Position[15..8]	Width[7..0]	Width[15..8]
16	PosWidth G2	Position[7..0]	Position[15..8]	Width[7..0]	Width[15..8]
17	PosWidth G3	Position[7..0]	Position[15..8]	Width[7..0]	Width[15..8]
18	ThresholdAlFilter G1	AlFilter[7..0]	Threshold[7..0]	-	-
19	ThresholdAlFilter G2	AlFilter[7..0]	Threshold[7..0]	-	-
20	ThresholdAlFilter G3	AlFilter[7..0]	Threshold[7..0]	-	-
21	AlDelay	Delay[7..0]	Delay[15..8]	AlarmSet[2..0]	-
22	AnalogOutput	Analog1Set[1..0]	Analog2Set[1..0]	Analog3Set[1..0]	Polarity[1..0]
23	ReadingPortFunction	Port[2..0]	-	-	-
24	Sampling Freq	Sampling[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 25ns
(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
(13) Echo-start polarity
(14) Filter → 4:NoFilter, 3:10MHz, 2:5MHz, 1:2.5MHz, 0:1.25MHz

(15) Hardware Gate1 measurement (step of 25ns) : Position[15..0]/Width[15..0]
(16) Gate2
(17) Gate3
(18) Hardware Gate1 measurement : Threshold on 8bit=0..255=0..100%
(19) Gate2
(20) Gate3
(21) Hardware GateX measurement : Delay[15..0] step of 800ns for Alarm and Analog output duration / AlarmSet[0]=AlG1, AlarmSet[1]=AlG2, AlarmSet[2]=AlG3 : 0=Alarm on appearance 1=Alarm on Disappearence
(22) Hardware GateX measurement : AnalogXSet 0=OFF 1=TotalAmplitude 2=AmplitudeOverThreshold / Polarity[1..0] 0=Positive&Negative 1=Negative 2=Positive
(23) ReadingPortFucntion[2..0] : 0=Ascan 1=HardwareGateMeasurement 2=AscanLSB
HardwareGateMeasurement is a 10byte frame : AmplG1, AmplG2, AmplG3, DistG1LSB, DistG1MSB, DistG2LSB, DistG2MSB, DistG3LSB, DistG3MSB, Alarm [2..0]

Alarm[0]=G1, Alarm[1]=G2, Alarm[2]=G3

(24) Sampling Freq : 0=160MHz / 1=80MHz / 2=40MHz

Attention :

- Sampling Frequencies are available only if Filter=4=NoFilter AND compression=0
- Compression is available only if Filter=4=NoFilter AND SamplingFreq=1=80MHz
- Filters are available only if SamplingFreq=1=80MHz and compression=0

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:

Slave Select 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.

wBuffer is the buffer which will be written in US-SPI (bytes)

nb of Byte to write is the number of bytes to write from the wBuffer

number of Byte in transaction is the addition of read and write number of bytes.

rBuffer is the buffer which will be filled by the bytes red in US-SPI

number of Byte to read 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)
```

<u>NOTES :</u>

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.

<u>NOTES :</u>

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)
```

<u>NOTES :</u>

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)
```

<u>NOTES :</u>

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)
```

<u>NOTES :</u>

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
```

<u>NOTES :</u>

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.

<u>NOTES :</u>

11) Hardware Gate Measurement

There are 3 hardware gates which can measure amplitude in real time (PRF)
Each gate can be set by: position, width, threshold, AlarmFilter, AlarmSet and AnalogSet

Polarity, Alarm and Amplitude can be set with a common parameter

Ex:
Gate1: position=1µs → 40 step of 25ns
Gate1: width= 500ns → 20 step of 25ns
Gate1: threshold (8bit)=50% → 128
Gate1: AlarmFilter = 5 strikes before alarm
Gate1: AlarmSet1 = Alarm on Appearance=0
Gate1: AnalogSet1 = Total Amplitude=1

Gate2: position=3µs → 120 step of 25ns
Gate2: width= 800ns → 32 step of 25ns
Gate2: threshold (8bit)=20% → 51
Gate2: AlarmFilter = 0 strikes before alarm
Gate2: AlarmSet2 = Alarm on Appearance=0
Gate2: AnalogSet2 = Total Amplitude=1

Gate3: position=6µs → 240 step of 25ns
Gate3: width= 3µs → 120 step of 25ns
Gate3: threshold (8bit)=75% → 191
Gate3: AlarmFilter = 1 strike before alarm
Gate3: AlarmSet3 = Alarm on DisAppearance=1
Gate3: AnalogSet3 = AmplitudeOverTheshold=2

Common parameter : Polarity:Pos&Neg=0 Alarm&Analog duration 10ms

Programming :

```
‘ Gate 1
   Width1=500ns/25ns=20
   pos1=1µs=1000ns/25ns=40
   threshold1=50%=50*255/100=128
   AlFilter1=5

   wBuffera(0)= Floor(widht1/256)              'msb=0
   wBuffera(1)= widht1-Floor(widht1/256)*256 'lsb=20
   wBuffera(2)= Floor(pos1/256)                'msb=0
   wBuffera(3)= pos1-Floor(pos1/256)*256       'lsb=40
   wBuffera(4)= 15                             'PosWidth Gate1
   spi.WriteRead(0,wBuffera,5,5,rBufferl,0)

   wBuffera(0)= threshold1                     '=128
   wBuffera(1)= AlFilter1                      '=5 strikes before alarm
   wBuffera(2)= 18                             'Threshold AlFilter Gate1
   spi.WriteRead(0,wBuffera,3,3,rBufferl,0)
```

‘ Gate 2

```
Width2=800ns/25ns=32
Pos2=3µs=3000ns/25ns=120
Threshold2=20%=20*255/100=51
AlFilter2=0

wBuffera(0)= Floor(widht2/256)             'msb=0
wBuffera(1)= widht2-Floor(widht2/256)*256 'lsb=32
wBuffera(2)= Floor(pos2/256)               'msb=0
wBuffera(3)= pos2-Floor(pos2/256)*256      'lsb=120
wBuffera(4)= 16                            'PosWidth Gate2
spi.WriteRead(0,wBuffera,5,5,rBufferl,0)

wBuffera(0)= threshold2                    '=51
wBuffera(1)= AlFilter2                     '=0 strikes before alarm
wBuffera(2)= 19                            'Threshold AlFilter Gate2
spi.WriteRead(0,wBuffera,3,3,rBufferl,0)
```

‘ Gate 3

```
Width3=3µs=3000ns/25ns=120
Pos3=6µs=6000ns/25ns=240
Threshold3=75%=75*255/100=191
AlFilter3=1

wBuffera(0)= Floor(widht3/256)             'msb=0
wBuffera(1)= widht3-Floor(widht3/256)*256 'lsb=120
wBuffera(2)= Floor(pos3/256)               'msb=0
wBuffera(3)= pos3-Floor(pos3/256)*256      'lsb=240
wBuffera(4)= 17                            'PosWidth Gate2
spi.WriteRead(0,wBuffera,5,5,rBufferl,0)

wBuffera(0)= threshold3                    '=191
wBuffera(1)= AlFilter3                     '=1 strikes before alarm
wBuffera(2)= 20                            'Threshold AlFilter Gate3
spi.WriteRead(0,wBuffera,3,3,rBufferl,0)
```

‘ Common Parameters

```
Polarity=0
Tempo=10ms=10000000ns/800ns=12500
AlarmSetX=AlarmSet1+(AlarmSet2*2)+(AlarmSet3*4)=4

wBuffera(0)= Polarity                      '0=positive AND negative
wBuffera(1)= 2                             '2=Over Threshold Gate3
wBuffera(2)= 1                             '1=Total Amplitude Gate2
wBuffera(3)= 1                             '1=Total Amplitude Gate1
wBuffera(4)= 22                            'AnalogSetX Polarity
spi.WriteRead(0,wBuffera,5,5,rBufferl,0)

wBuffera(0)= AlarmSetX                     '=4
wBuffera(1)= Floor(Tempo/256)              '48
wBuffera(2)= Tempo-Floor(Tempo/256)*256    '212
wBuffera(3)= 21                            'AlarmSetX Duration
spi.WriteRead(0,wBuffera,4,4,rBufferl,0)
```

NOTES :

12) Reading Port Function

‘A-scan 8Bit reading

```
wBuffera(0)=0                              '0=A-scan
wBuffera(1)=23                             'ReadingPortFunction
spi.WriteRead(0,wBuffera,2,2,rBufferl,0)


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)
```

‘Hardware Gate Measures reading

```
wBuffera(0)=1                              '0=Measures
wBuffera(1)=23                             'ReadingPortFunction
spi.WriteRead(0,wBuffera,2,2,rBufferl,0)

spi.WriteRead(0,wBuffera,0,10,rBufferl,10)

Ampl1=(Bit.AND(rBufferl(0),0xff))*100/255 '%
Ampl2=(Bit.AND(rBufferl(1),0xff))*100/255 '%
Ampl3=(Bit.AND(rBufferl(2),0xff))*100/255 '%
Dist1=(Bit.AND(rBufferl(3),0xff)+(256*Bit.AND(rBufferl(4),0xff)))*12.5/1000'µs
Dist2=(Bit.AND(rBufferl(5),0xff)+(256*Bit.AND(rBufferl(6),0xff)))*12.5/1000'µs
Dist3=(Bit.AND(rBufferl(7),0xff)+(256*Bit.AND(rBufferl(8),0xff)))*12.5/1000'µs
Alarme=(Bit.AND(rBufferl(9),0xff))
If (Bit.AND(Alarme,1)=1) then Alarme1=True Else Alarme1=False
If (Bit.AND(Alarme,2)=2) then Alarme2=True Else Alarme2=False
If (Bit.AND(Alarme,4)=4) then Alarme3=True Else Alarme3=False
```

NOTES :

'A-scan 12Bit reading

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

wBuffera(0)=2                                '2=A-scan LSB  [3..0]
wBuffera(1)=23                               'ReadingPortFunction
spi.WriteRead(0,wBuffera,2,2,rBufferl,0)

spi.WriteReadasync(1,wBuffer,0,your nb of samples,rBuffer1, your nb of samples)

wBuffera(0)=0                                '2=A-scan 8bit  [11..4]
wBuffera(1)=23                               'ReadingPortFunction
spi.WriteRead(0,wBuffera,2,2,rBufferl,0)

spi.WriteReadasync(1,wBuffer,0,your nb of samples,rBuffer2, your nb of samples)
```

For i=0 to "Your nb of samples"-1
    Tab(i)=(rBuffer1 and 0x0f) + 16*(rBuffer2 and 0xff)
Next i

NOTES :