



The  
University  
Of  
Sheffield.

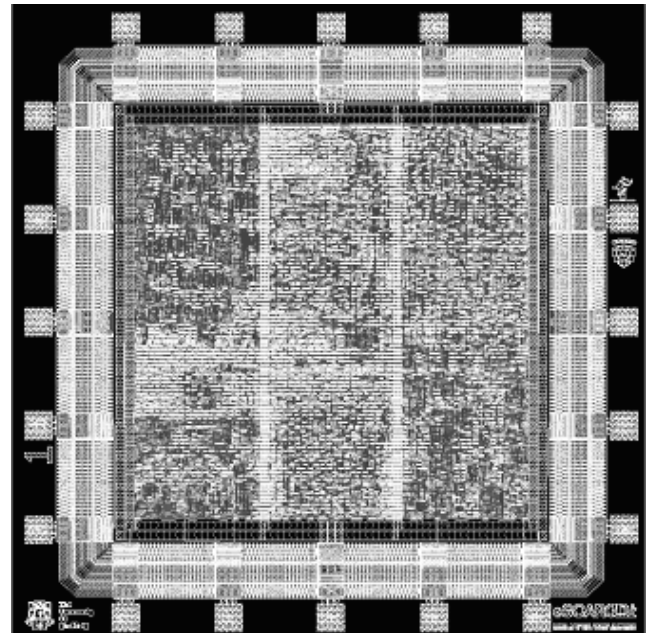
Department of  
Electrical and  
Electronic  
Engineering

# eSCARGO

European Stream Ciphers Are Ready (to) GO

0.18 $\mu$ m ASIC Datasheet

- Technology: UMC 0.18 $\mu$ m CMOS  
overall chip size 1521x1521 $\mu$ m
- Package: SOIC20
- Voltage: 3.3V I/O, 1.8V core
- Synchronous Serial Interface  
with handshaking
- Four bit number to select cipher
- Phase-III hardware profile designs:  
*Moustique, Edon80, Trivium,  
Decim80/128, F-fcsr-h/16,  
Grain80/128, Mickey80/128 &  
Pomaranch80/128*
- Internally  $\times 8$  accelerated designs for  
*Trivium & Grain80.*



## Interface

ready_for_iv ←	1	o	20	— V <sub>IO</sub> 3.3V
ready_for_key ←	2		19	← keyiv
cipher[0] →	3		18	← decrypt
cipher [1] →	4		17	← rst
V <sub>CORE</sub> 1.8V —	5	SOIC-20	16	← clk [Schmitt]
cipher [2] →	6		15	— GND <sub>CORE</sub>
cipher[3] →	7		14	← slew_control
load →	8		13	← drive_strength
din →	9		12	→ output_valid
GND <sub>IO</sub> —	10		11	→ dout

I/O is LVCMOS/LVTTL compatible i.e. 3.3V nominal. It is NOT 5V tolerant. The internal core of the chip operates from a 1.8V nominal supply. Do not exceed the maximum ratings as tabulated below. Power up ordering for V<sub>CORE</sub> and V<sub>IO</sub> is not critical.

<b>clk</b>	operating clock, chip uses schmitt trigger input levels, design maximum 50 MHz. Clock is gated to each cipher, only currently selected cipher is clocked.
<b>rst</b>	reset the current cipher, non-selected ciphers remain unaffected
<b>keyiv</b>	synch. serial key/IV input
<b>din</b>	synch serial plain/cipher text input
<b>dout</b>	synch serial cipher/plain text output
<b>load</b>	handshake input; load/ack I/O
<b>ready_for_key</b>	handshake output
<b>ready_for_iv</b>	handshake output
<b>output_valid</b>	handshake output
<b>decrypt</b>	selects decrypt (Moustique only) also selects alternate design variation for Decim and Pomaranch.
<b>cipher[0..3]</b>	selects the “current cipher”; internally synchronised with next clock so changes become effective on next rising edge of clock. For ciphers code number see following table. Only the currently selected cipher is connected to any of the I/O lines, all other ciphers will remain unaffected.
<b>slew_control</b>	output driver cells (1=fast slew), normally low
<b>drive_strength</b>	output driver cells (1=max strength), normally low
<b>V<sub>IO</sub></b>	3.3V nominal, 3.63V maximum
<b>V<sub>CORE</sub></b>	1.8V nominal 1.98V maximum Typical current consumption 680uA @ 25MHz clock during tests applying random test vectors to all ciphers in sequence.
<b>GND<sub>CORE</sub>, GND<sub>IO</sub></b>	Ground pins for core and I/O respectively. Package body bonded to GND <sub>IO</sub> .

### Cipher selection

Cipher Number	cipher [3][2][1][0]	Name
0	0 0 0 0	“idle”
1	0 0 0 1	Moustique
2	0 0 1 0	Edon80
3	0 0 1 1	Trivium
4	0 1 0 0	Decim80
5	0 1 0 1	Decim 128
6	0 1 1 0	F-FCSR-H
7	0 1 1 1	F-FCSR-16
8	1 0 0 0	Grain80
9	1 0 0 1	Grain128
10	1 0 1 0	Mickey-80 (loads IV first)
11	1 0 1 1	Mickey-128 (loads IV first)
12	1 1 0 0	Pomaranch80
13	1 1 0 1	Pomaranch128
14	1 1 1 0	Grain80 (×8 internally)
15	1 1 1 1	Trivium (×8 internally)

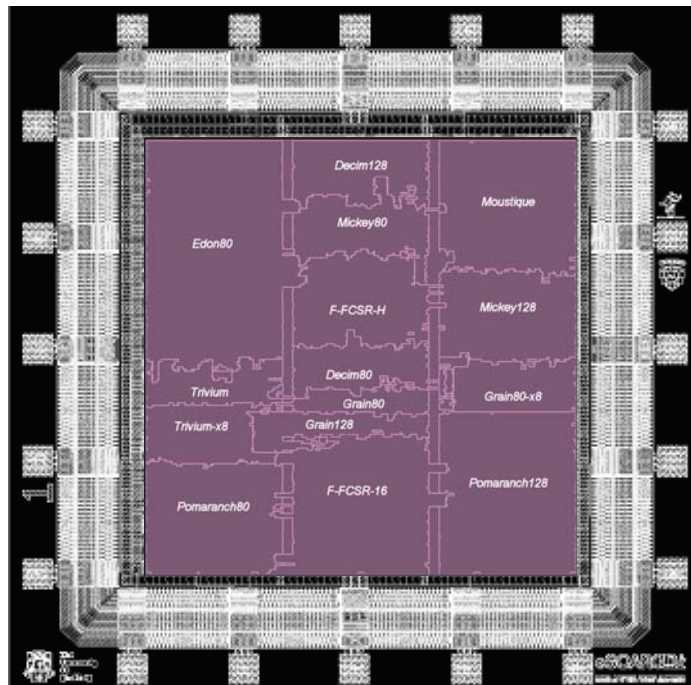
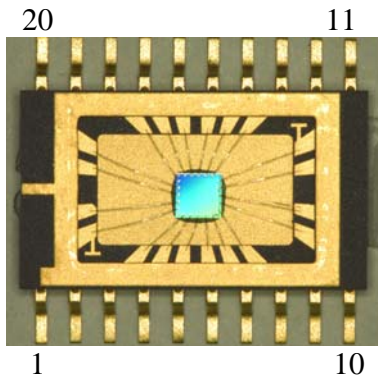
### Normal Operation

1. Supply a monotonic clock signal to the clk pin.
2. Select the desired cipher by applying the 4-bit number to cipher.
3. Drive the reset signal high for at least one complete clock cycle
4. Wait for “ready\_for\_key” or “ready\_for\_iv” to be asserted high.
5. Supply each bit of Key/IV as appropriate to the keyiv pin and signal its presence as valid by driving the load pin high. This should be performed synchronously with the clock.
6. Once both ready\_for\_key and ready\_for\_iv are low the cipher is running.
7. Wait for “output\_valid” to be asserted high, this indicates that the keystream bit is valid and that dout will be equal to din XOR keystream. Prior to this dout is internally driven low. Once the present keystream bit is finished with drive “load” high to acknowledge this. The next keystream bit will be valid when output\_valid is again asserted. Most of the ciphers will support continuous operation with load driven constantly high.
8. When the current key/iv session is finished with assert rst high to load a new key or IV.

## Packaging

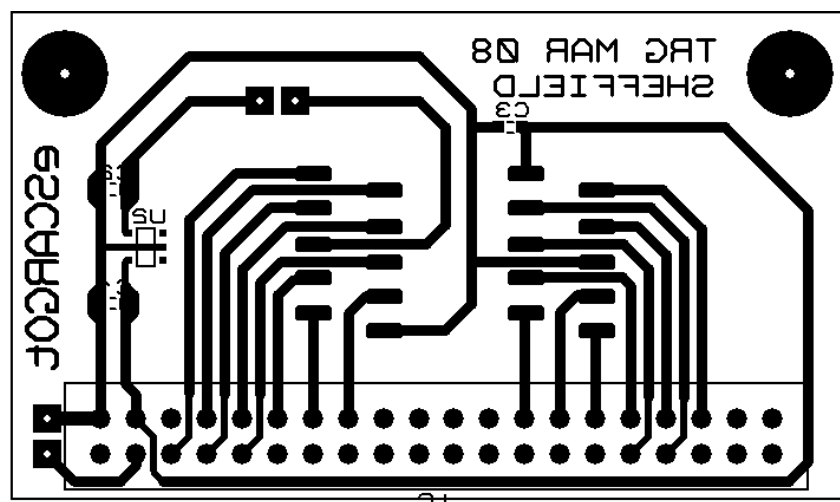
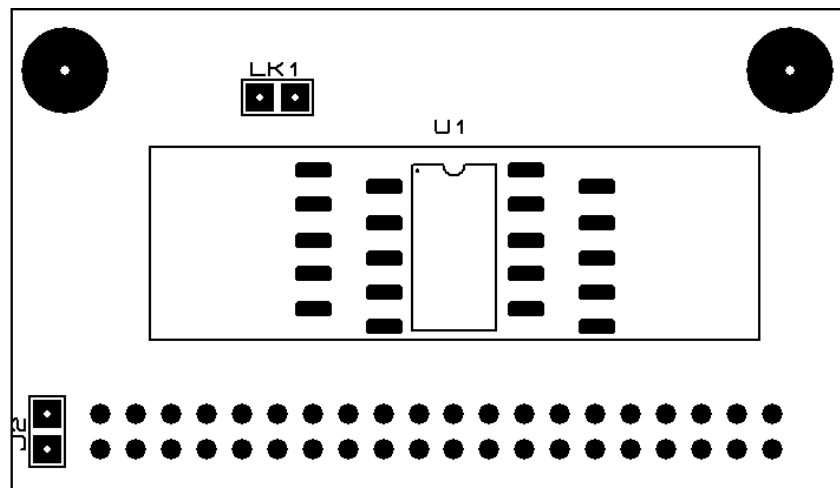
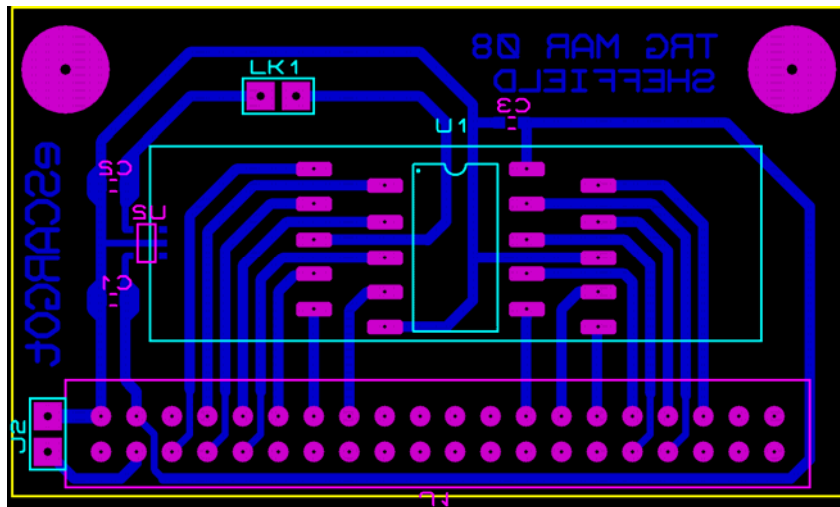
For those wishing to attempt fault injection the following information is provided in line with Kerckhoff's assumptions (the design is known):

- (a) The lids have been taped on rather than sealed to simplify their removal.
- (b) The layout of the design and orientation within the chip is as shown below.
- (c) The high-K dielectric used in the 0.18 process is a relatively opaque green.
- (d) The metal fill has been performed using normal rules: the result is small rectangles of electrically isolated metal over the whole active chip area.
- (e) The general implementation for each cipher is as per the hardware chapter in the LNCS book "The eSTREAM Finalists", volume 4986, Springer, May 2008.





Example PCB as used for testing



## Design Notes

**“idle” cipher:** a 32-bit shift register fed from `key_iv_in` and new bits loaded according to `ld_data`, the output ciphertext is the result of the final bit of the SR being XORed with the plaintext. Included to permit some testing/calibration in (anyone’s) side channel test rig.

**Mickey80/128:** note that the key and IV loading process is reversed. i.e. IV is loaded before the key.

**Decim128:** uses 64-bit output buffer. Occurrence of buffer empty is improbable so the buffer refill mechanism cannot be tested. At time of this design a single tap change had been proposed and was still under consideration; accessed by setting the “decrypt” line.

**Pomaranch80/128:** Reference code to design changed 22/1/08, for designs inclusive of this change, set “decrypt” line high.

**Moustique:** dedicated decrypt input pin allows operation of self-synch decryption, if already in operation selecting this line followed by sending a ‘0’ then IV will result in decryption alternatively if decrypt selected post-reset before key loading is completed, the cipher will not ask for an IV and should proceed with ‘0’ followed by IV.

**slew\_control and drive\_strength:** set these inputs high to operate ciphers at higher I/O speeds. Frequency at which change-over required depends on capacitive loading of outputs, TBD for typical experimental setup post manufacture.

**bit ordering:** many of the test vectors for the designs are in a non-standard order. For some the bit ordering for the key, IV and keystream within the ciphers operating word has not been fully defined. The following table reflects this designers understanding and the ordering expected by this implementation.

<i>cipher</i>	<i>key/iv</i>	<i>keystream</i>
Moustique	normal	normal
Trivium	quad byte swapped	quadbyte swapped
Pomaranch	normal (but 18bit hex values)	normal
Mickey	normal	normal
Grain	bits in 8-bit bytes reversed	bits in 8-bit bytes reversed
F-fcsr-h	bytes reversed	normal
F-fcsr-16	bits in 16-bit words reversed	byte pairs swapped
Edon80	normal	normal
Decim	bits reversed	bits reversed



```

R(X"4AFD9923") & R(X"254AFE72") & R(X"26654519") & R(X"76A955BB") &
R(X"930967FF") & R(X"2A68617A") & R(X"A45A2C34") & R(X"EF4BD60F") ) ) ),

(C =>trivium, LK=>80, LI=>80, LO=>16*32, PT=>(others=>'0'),
K =>N(80, R80(X"0F62B5085BAE0154A7FA")),
IV=>N(80, R80(X"288FF65DC42B92F960C7")),
CT=>NKS(16*32, (R(X"FC9659CB") & R(X"953A37FF") & R(X"E869C13F") & R(X"462FE099") &
R(X"02C2B955") & R(X"2D976A45") & R(X"62EA79F6") & R(X"F9540801") &
R(X"48587926") & R(X"5FA2239D") & R(X"E46CF09B") & R(X"EFD3A0FD") &
R(X"80BC0B78") & R(X"2ED18134") & R(X"F9A0D74D") & R(X"B5A003E5") ) ) ),

(C =>trivium, LK=>80, LI=>80, LO=>16*32, PT=>(others=>'0'),
K =>N(80, R80(X"0A5DB00356A9FC4FA2F5")),
IV=>N(80, R80(X"1F86ED54BB2289F057BE")),
CT=>NKS(16*32, (R(X"DFD142F1") & R(X"65C7364B") & R(X"4A2EBA2C") & R(X"A051EE22") &
R(X"C0571416") & R(X"7EED4D76") & R(X"E6545CED") & R(X"B87C399A") &
R(X"557BD2E0") & R(X"F43A6060") & R(X"BDDDE20C") & R(X"EF50A3B1") &
R(X"0C5B2E18") & R(X"A6C424DA") & R(X"9EDA446B") & R(X"53B0D0F3") ) ) ),

(C =>trivium_x8, LK=>80, LI=>80, LO=>16*32, PT=>(others=>'0'),
K =>N(80, R80(X"0F62B5085BAE0154A7FA")),
IV=>N(80, R80(X"288FF65DC42B92F960C7")),
CT=>NKS(16*32, (R(X"FC9659CB") & R(X"953A37FF") & R(X"E869C13F") & R(X"462FE099") &
R(X"02C2B955") & R(X"2D976A45") & R(X"62EA79F6") & R(X"F9540801") &
R(X"48587926") & R(X"5FA2239D") & R(X"E46CF09B") & R(X"EFD3A0FD") &
R(X"80BC0B78") & R(X"2ED18134") & R(X"F9A0D74D") & R(X"B5A003E5") ) ) ),

(C =>trivium_x8, LK=>80, LI=>80, LO=>16*32, PT=>(others=>'0'),
K =>N(80, R80(X"0A5DB00356A9FC4FA2F5")),
IV=>N(80, R80(X"1F86ED54BB2289F057BE")),
CT=>NKS(16*32, (R(X"DFD142F1") & R(X"65C7364B") & R(X"4A2EBA2C") & R(X"A051EE22") &
R(X"C0571416") & R(X"7EED4D76") & R(X"E6545CED") & R(X"B87C399A") &
R(X"557BD2E0") & R(X"F43A6060") & R(X"BDDDE20C") & R(X"EF50A3B1") &
R(X"0C5B2E18") & R(X"A6C424DA") & R(X"9EDA446B") & R(X"53B0D0F3") ) ) ),

C =>pomaranch128, LK=>128, LI=>162, LO=>512, PT=>(others=>'0'),
K =>N(128, (x"C7C0"&x"BBBA"&x"F93A"&x"00F3"&x"CC71"&x"BBAE"&x"3DAB"&x"00AF")),
IV=>N(162, (B18(x"3C7C0") & B18(x"02BBA") & B18(x"3893A") &
B18(x"250F3") & B18(x"1EC71") & B18(x"0D7C0") &
B18(x"2A93A") & B18(x"06C71") & B18(x"19DAB") ) ),
CT=>NKS(512, "1110111010000011100000001111010101001011100001110001110111111111"&
"101010010000001010010110000100010011101010110011110101000111101010001100000"&
"1010010001100111100110011100111101111011101110000001101111001110111101"&
"101000001011001100101011101100110100000001000000001111100001000"&
"011110111001110100010001000000010000010111101100100100010110001"&
"0001011110001010100101010011101101001100101001010110110010000010"&
"111110010100100110101011101111011000001100000100010111101111"&
"1011010111010011011010010111010010101110101101011010011000000101010"&
"1011010111010011011010010111010010101110101101011010011000000101010" ) ),

(C =>pomaranch80, LK=>80, LI=>108, LO=>512, PT=>(others=>'0'),
K =>N(80, (x"00F3" & x"CC71" & x"BBAE" & x"3DAB" & x"00AF")),
IV=>N(108, (B18(x"250F3") & B18(x"1EC71") & B18(x"0D7C0") &
B18(x"2A93A") & B18(x"06C71") & B18(x"19DAB") ) ),
CT=>NKS(512, "101011111101110001010001011111101001010010110111010111010111011110"&
"1110100000011011011101011101001111110100111110100101000110111100111"&
"00101100111100110000100101110101110110001110001010010001111011"&
"0100000110110110110101001110000000010010101101110111011110111101"&
"1110001101101000011001001101010111011010101101001011001100110111100"&
"0111000100100101010101000100100110010010010010010010010010010101"&
"00011010111101110110101110010001110100110000011010000101000"&
"0101101100011000001101110100011101001110100100001000101011010000" ) ),

(C =>mickey128, LK=>128, LI=>128, LO=>64*8, PT=>(others=>'0'),
IV=>N(128, (X"f11a5627ce43b61f8912299486094486")), -- really key
K =>N(128, (X"9c532f8ac3ea4b2ea0f59640308377cc")), -- really IV
CT=>NKS(64*8, (X"77de5b94186367b2127aa8395e1946771be963d5ec513dc1c52c50e6f1e2442e48a6c09ae
dc3762a074efa7edd54656ae71962785a9c201b8cfc8f41f5a70b4e")) ),

(C =>mickey80, LK=>80, LI=>80, LO=>64*8, PT=>(others=>'0'),
IV=>N(80, (X"f11a5627ce43b61f8912")), -- really key
K =>N(80, (X"9c532f8ac3ea4b2ea0f5")) -- really IV

```

CT=>NKS(64\*8,(X"21a0436619cb9f3f6f1fb303f56a09a9d8a6de29e966f2e08a1058b817dc7640c590681403e187fa3eedf5846aacdd6e036e7f546430563f28c6e14dd8249996" ) ) ,

(C =>grain80, LK=>80, LI=>64, LO=>512, PT=>(others=>'0'),  
K =>N(80, (X"0123456789abcdef1234" ) ) ,  
IV=>N(64, (X"0123456789abcdef" ) ) ,  
CT=>NKS(512,(X"42b567ccc65317680225cd83b21db3e41781ea81f82274c44f8e2ea43c188d92e24105c7c0e7fdacc5557d5309e208ebcf9292482e5ddfa3315245cf4900eaaa" ) ) ,

(C =>grain80x8, LK=>80, LI=>64, LO=>512, PT=>(others=>'0'),  
K =>N(80, (X"0123456789abcdef1234" ) ) ,  
IV=>N(64, (X"0123456789abcdef" ) ) ,  
CT=>NKS(512,(X"42b567ccc65317680225cd83b21db3e41781ea81f82274c44f8e2ea43c188d92e24105c7c0e7fdacc5557d5309e208ebcf9292482e5ddfa3315245cf4900eaaa" ) ) ,

(C =>grain128, LK=>128, LI=>96, LO=>512, PT=>(others=>'0'),  
K =>N(128, (X"0123456789abcdef123456789abcdef0" ) ) ,  
IV=>N( 96, (X"0123456789abcdef12345678" ) ) ,  
CT=>NKS(512,(X"db032aff3788498b57cb894fffb6bb96b1220ce5a70dfcc86eff0c4f733073aa4c6a35f2c79fe3a5930laf7a67d1296b3077c1cbd51404572f76ed742f7c6b2a" ) ) ,

(C =>ffcsr, LK=>80, LI=>80, LO=>9\*8,  
K =>FBR(80, (X"0088639d6bf847ed59c6" ) ) ,  
IV=>FBR(80, (X"00112233445566778899" ) ) ,  
PT=>NKS(9\*8,(X"534f53454d414e554b" ) ) ,  
CT=>NKS(9\*8,(X"f591ceca76ef4ab65e" ) ) ,

(C =>ffcsr, LK=>80, LI=>80, LO=>64\*8, PT=>(others=>'0'),  
K =>FBR(80, (X"0088639d6bf847ed59c6" ) ) ,  
IV=>FBR(80, (X"00112233445566778899" ) ) ,  
CT=>NKS(64\*8,(X"a6de9d8f3bae04e315225532f7b6181ba2e8c6aec274dfe2366e6a79dbe49273778780d273e8f14ebbd9ee5ecac7c0bc403db6df2f3bfc8d53d4ea966fb2ecf" ) ) ,

(C =>ffcsr16, LK=>128, LI=>128, LO=>5\*16,  
K =>FWR(128,(X"0088639d6bf847ed59c621795d3363f1" ) ) ,  
IV=>FWR(128,(X"00112233445566778899aabbccddeeff" ) ) ,  
PT=>BSKS(5\*16,(X"462d464353522d313621" ) ) , -- "F-FCSR-16!"  
CT=>BSKS(5\*16,(X"15fd4a9de00ede722b29" ) ) ,

(C =>ffcsr16, LK=>128, LI=>128, LO=>64\*8, PT=>(others=>'0'),  
K =>FWR(128,(X"0088639d6bf847ed59c621795d3363f1" ) ) ,  
IV=>FWR(128,(X"00112233445566778899aabbccddeeff" ) ) ,  
CT=>BSKS(64\*8,(X"53d00cdeb35cf3431d081ca692e30d8b405fa43e232bfe6f7888ab6d1d1b68a34feb532941f103fb119ab192e328170ab6bfc1ffe8888bc90fa2775da578506e" ) ) ,

(C =>edon80, LK=>80, LI=>64, LO=>40\*8, PT=>(others=>'0'),  
K =>N(80,X"00000000000000000000" ) ,  
IV=>N(64,X"0000000000000000" ) ,  
CT=>NKS(40\*8,X"45F3AD99375000DCDBAD40C455BD6BE41E527C68896E16C8D94673CA4AD46301E2BCEFBFC23390F3" ) ) ,

(C =>edon80, LK=>80, LI=>64, LO=>40\*8, PT=>(others=>'0'),  
K =>N(80,X"00000000000000000000" ) ,  
IV=>N(64,X"0000000000000001" ) ,  
CT=>NKS(40\*8,X"805CF5B84E112CB966F6973FE72588FCBCF50ABFAA3E83E80B1CBF590F683E593A40D90996FA66C1" ) ) ,

(C =>edon80, LK=>80, LI=>64, LO=>40\*8, PT=>(others=>'0'),  
K =>N(80,X"00000000000000000000" ) ,  
IV=>N(64,X"000000000000002" ) ,  
CT=>NKS(40\*8,X"1CB943777F8E4D2B7698002CF7EAF95F4AA5A6CA20AE2A7668CD6975FDCC9C03BAB5B355DD933890" ) ) ,

(C =>edon80, LK=>80, LI=>64, LO=>40\*8, PT=>(others=>'0'),  
K =>N(80,X"00000000000000000000" ) ,  
IV=>N(64,X"800000000000000000" ) ,  
CT=>NKS(40\*8,X"BB333B7A998B6996E7F6DC5E9892B3B26E7F5847CE283DB83D483BC79BFBD7048CD136D29F9982AC" ) ) ,

(C =>edon80, LK=>80, LI=>64, LO=>40\*8, PT=>(others=>'0'),  
K =>N(80,X"00000000000000000001" ) ,

```

IV=>N(64,X"0000000000000000"),
CT=>NKS(40*8,X"5511C2F3104B86E1E8D53C643038C306E61694FB9BA7FCC38C6F93024ACA87123200CDA404
B9135D")),

(C =>edon80, LK=>80, LI=>64, LO=>40*8, PT=>(others=>'0'),
K =>N(80,X"40000000000000000000000000000000"),
IV=>N(64,X"000000000000000000000000"),
CT=>NKS(40*8,X"1FD62FD964EFA672E982FCA4DDE35DF9C5110AAA083FEF730C84B800D7AAD664AF8DAFA3EB
D5F643")),

(C =>edon80, LK=>80, LI=>64, LO=>40*8, PT=>(others=>'0'),
K =>N(80,X"80000000000000000000000000000000"),
IV=>N(64,X"000000000000000000000000"),
CT=>NKS(40*8,X"51586D396D4FB16A77952B53D6A694987248FE1D4C0174DED772DE246A926446521B83A60E
8B80F5")),

-- decim128 TEST VECTOR FOR 64-bit BUFFER (as per paper)
(C =>decim128, LK=>128, LI=>128, LO=>24*8, PT=>(others=>'0'),
K =>BR(128,X"f63db4854f857da955430c548ccd36d5"),
IV=>BR(128,X"0c8edb8801aa95b43684baf411dc47df"),
CT=>BRKS(24*8,X"a7a5822b0e9591a4321999cac109abe55a7da84d587ad2f4") ),

(C =>decim80, LK=>80, LI=>64, LO=>24*8, PT=>(others=>'0'),
K =>BR(80,X"f63db4854f857da95543"),
IV=>BR(64,X"0c8edb8801aa95b4"),
CT=>BRKS(24*8,X"020e060f6a2f008a28e82f355d203f2f443decc627fe068c") )

```

### **Further Information**

T Good or M Benaissa, Department of Electronic and Electrical Engineering,  
University of Sheffield, Mappin Building, Mappin Street, Sheffield S1 3JD, U.K.  
{t.good, m.benaissa} @ sheffield.ac.uk