Technical Specification Group Services and System Aspects Meeting #21, Frankfurt, Germany, 22-25 September 2003

TSGS#21(03)0363

INTERNATIONAL TELECOMMUNICATION UNION

TELECOMMUNICATION STANDARDIZATION SECTOR

STUDY PERIOD 2001-2004

COM 16 – LS 16 – E

English only

Original: English

Question(s): 10/16 Geneva, 20-30 May 2003

LIAISON STATEMENT

Source: ITU-T SG 16

Title: LS to 3GPP and 3GPP2 on basic operators library

LIAISON STATEMENT

To: **3GPP and 3GPP2**

Approval: **Agreed to at ITU-T SG 16 meeting (Geneva, 20-30 May 2003)**

For: **Information/Action**

Deadline: **1 January 2004**

Contact: Ms Claude Lamblin
France Telecom
France
Tel: +33 2 96 05 13 03
Fax: +33 2 96 05 3530
Email: claude.lamblin@rd.francetelecom.com

Mr Eyal Shlomot
Conexant Systems
4000 MacArthur Blvd.
Newport Beach, CA 92660
USA
Tel: +1-949-579-4988
Fax: +1-949-579-6361
Email: shlomot@mindspeed.com

ITU-T Q.10/16 wishes to inform 3GPP SA4 and 3GPP2 TSG-C1.1 of our intention to update the set of basic operators.

The objectives of this work are to review the current 32-bit basic operators set (especially their weights) as well as to consider the recently proposed 40-bit basic operators to be included in addition to the existing one.

The work will mainly progress email correspondence using the WP3 audio email reflector wp3audio@yahoogroups.com.

As the 32-bit basic operators were defined some years ago, their review and especially the revision of their associated weights used in deriving the WMOPS figures may be justified. The work on the proposed extension with 40-bit basic operators aims to check its completeness and its correctness, in particular while considering various DSP platforms. The 40-bit extension might be considered for alternative implementation (i.e. annex) of speech coding standards in the ITU-T.

ITU-T Q.10/16 would like to take this opportunity to invite 3GPP SA4 and 3GPP2 TSG-C1.1 to review the current set of basic operators and the proposed 40-bit extension, and possibly to provide some feedback. Since 3GPP mandates fixed-point implementation of its standards, and 3GPP2 provides suggested fixed-point implementations for several of its newest standards, the harmonisation of tools for DSP simulation between these standardisation bodies can be beneficial. It is also recognised that experts on DSP implementation and DSP manufacturing serve in these different standardisation bodies, and can provide significant information on the best way to simulate various DSP platforms by simulation software.

The current basic operators set can be found in STL2000 (ITU-T Rec. G.191). http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-G.191

Attached is the proposal for some 40-bit basic operators. This set of operators includes addition (L_add40), subtraction (L_sub40), multiply-and-add (L_mac40), multiple-and-subtract (L_msu40), normalisation (norm32_l40), shifting (L_shr40, L_shl40), saturation (L_sat32_40), and bit-fields deposit and extraction (extract_l40, L_deposit_l40).

Attachment: draft set of 40-bit basic operators

[Attachment in electronic version only]

```
//***************************************************************************

// local functions for 40 bit accumulator

/*_____________________________________________________________________________

|                                                                             |

|   Function Name : L_shift_left40                                            |

|                                                                             |

|   Purpose :                                                                 |

|                                                                             |

|   Perform an integer arithmetic shift on the double format value.          |

|   If nbits > 0, arithmetically shift the value left nbits positions.       |

|   Zero fill the nbits LSB of the value.                                     |

|   If nbits < 0, arithmetically shift the value right by -nbits positions   |

|   with sign extension.                                                      |

|   No overflow control and saturation.                                       |

|_____________________________________________________________________________|
```

```
*/



static Word40 L_shift_left40(Word40 value, Word16 nbits)

{

  if (nbits > 40) nbits = 40;       /* to avoid needless loop after overflow */

  if (nbits < -40) nbits = -40;

  if (nbits < 0)

  {

    for(; nbits<0; nbits++)

    {

      value = value * 0.5;          /* remove fraction bits */

    }

    value = floor(value);

  }

  else
```

```
 {

   value = floor(value);          /* remove fraction bits */

   for(; nbits>0; nbits--)

   {

     value = value * 2;

   }

 }
 return(value);

}




/*_____________________________________________________________________

 |                                                     |

 |   Function Name : L_saturate40                                    |

 |                                                     |

 |   Purpose :                                               |

 |                                                     |
```

```
|    Limit the value to the range of a 40 bit word.                    |

|                                                                      |

|   Return Value :                                                     |

|                                                                      |

|    var_out  40 bit short signed integer (Word40) whose value falls in the |

|             range : MIN_40 <= var_out <= MAX_40.                     |

|______________________________________________________________________|

*/



static Word40 L_saturate40(Word40 value)

{

  Word40 L_var_out;



  L_var_out = value;
```

```
  if (L_var_out > MAX_40)

  {

    L_var_out = MAX_40;

    giOverflow = 1;

  }

  else if (L_var_out < MIN_40)

  {

    L_var_out = MIN_40;

    giOverflow = 1;

  }

  return(L_var_out);

}
```

```
//*****************************************************************************
```

```
//***************************************************************************

//***************************************************************************

//***************************************************************************



// public functions for 40 bit accumulator

//



/*_____________________________________________________________________________

|                                                             |

|   Function Name : L_add40                                   |

|                                                             |

|   Purpose :                                                 |

|                                                             |

|   Add L_var1 by L_var2. Return a 40 bit result without saturation:        |

|       L_add40 (L_var1, L_var2).                             |
```

| |

| Complexity weight : 1 |

| |

| Inputs : |

| |

| L_var1   40 bit long signed integer (Word40) whose value falls in the |

| range : MIN_40 <= L_var1 <= MAX_40. |

| |

| |

| L_var2   32 bit long signed integer (Word32) whose value falls in |

| the : MIN_32 <= L_var2 <= MAX_32. |

| |

| Return Value : |

| |

| L_var_out |

```
|            40 bit long signed integer (Word40) whose value falls in the  |
|            range : MIN_40 <= L_var_out <= MAX_40.                        |
|_____________________________________________________________________|
*/

Word40 L_add40(Word40 L_var1, Word32 L_var2)
{
  Word40 L_var_out;

  L_var_out = L_var1 + (Word40)L_var2;
  L_var_out = L_saturate40(L_var_out);
  return(L_var_out);
}

/*_____________________________________________________________________
```

| |

| Function Name : L_sub40 |

| |

| Purpose : |

| |

| Substract L_var1 by L_var2. Return a 40 bit result without saturation: |

| L_sub40 (L_var1, L_var2). |

| |

| Complexity weight : 1 |

| |

| Inputs : |

| |

| L_var1   40 bit long signed integer (Word40) whose value falls in the |

| range : MIN_40 <= L_var1 <= MAX_40. |

| |

```
|                                                           |

|   L_var2   32 bit long signed integer (Word32) whose value falls in      |

|            the : MIN_32 <= L_var2 <= MAX_32.                             |

|                                                           |

|   Return Value :                                                         |

|                                                           |

|   L_var_out                                                 |

|            40 bit long signed integer (Word40) whose value falls in the  |

|            range : MIN_40 <= L_var_out <= MAX_40.                        |

|___________________________________________________________________________|

*/



Word40 L_sub40(Word40 L_var1, Word32 L_var2)

{

  Word40 L_var_out;
```

```
  L_var_out = L_var1 - (Word40)L_var2;

  L_var_out = L_saturate40(L_var_out);

  return(L_var_out);

}




/*_____________________________________________________________________

 |                                                       |

 |   Function Name : L_mac40                                     |

 |                                                       |

 |   Purpose :                                               |

 |                                                       |

 |   Multiply var1 by var2 and shift the result left by 1. Add the 40 bit    |

 |   result to L_var3 with saturation, return a 40 bit result:               |

 |        L_mac(L_var3,var1,var2) = L_add(L_var3,(L_mult(var1,var2)).        |
```

| |

| Complexity weight : 1 |

| |

| Inputs : |

| |

| L_var3   40 bit long signed integer (Word40) whose value falls in the |

| range : MIN_40 <= L_var3 <= MAX_40. |

| |

| var1   16 bit short signed integer (Word16) whose value falls in |

| the : MIN_16 <= var1 <= MAX_16. |

| |

| var2   16 bit short signed integer (Word16) whose value falls in |

| the : MIN_16 <= var1 <= MAX_16. |

| |

| Return Value : |

```
|                                                                |

|   L_var_out                                                    |

|            40 bit long signed integer (Word40) whose value falls in the  |

|            range : MIN_40 <= L_var_out <= MAX_40.                  |

|________________________________________________________________|

*/



Word40 L_mac40(Word40 L_var3, Word16 var1, Word16 var2)

{

  Word40 L_var_out;



  L_var_out = L_var3 + ((Word40)var1 * (Word40)var2 * 2.0);

  L_var_out = L_saturate40(L_var_out);

  return(L_var_out);
```

```
}
```

```
/*____________________________________________________________________________
 |                                                                            |
 |   Function Name : L_msu40                                                  |
 |                                                                            |
 |   Purpose :                                                                |
 |                                                                            |
 |   Multiply var1 by var2 and shift the result left by 1. Subtract the 40    |
 |   bit result to L_var3 with saturation, return a 40 bit result:            |
 |       L_msu(L_var3,var1,var2) = L_sub(L_var3,(L_mult(var1,var2)).          |
 |                                                                            |
 |   Complexity weight : 1                                                    |
 |                                                                            |
```

| Inputs : |

| |

| L_var3 40 bit long signed integer (Word40) whose value falls in the |

| range : MIN_40 <= L_var3 <= MAX_40. |

| |

| var1 16 bit short signed integer (Word16) whose value falls in |

| the range : MIN_16 <= var1 <= MAX_16. |

| |

| var2 16 bit short signed integer (Word16) whose value falls in |

| therange : MIN_16 <= var1 <= MAX_16. |

| |

| Return Value : |

| |

| L_var_out |

| 40 bit long signed integer (Word40) whose value falls in the |

```
 |            range : MIN_40 <= L_var_out <= MAX_40.                  |

 |____________________________________________________________________|

*/



Word40 L_msu40(Word40 L_var3, Word16 var1, Word16 var2)

{

  Word40 L_var_out;



  L_var_out = L_var3 - ((Word40)var1 * (Word40)var2 * 2.0);

  L_var_out = L_saturate40(L_var_out);

  return(L_var_out);

}



/*____________________________________________________________________
```

| |

| Function Name : L_shr40 |

| |

| Purpose : |

| |

| Arithmetically shift the 40 bit input L_var1 right var2 positions with |

| sign extension. If var2 is negative, arithmetically shift L_var1 left |

| by -var2 and zero fill the var2 LSB of the result. Saturate the result |

| in case of underflows or overflows. |

| |

| Complexity weight : 2 |

| |

| Inputs : |

| |

| L_var1 40 bit long signed integer (Word40) whose value falls in the |

```
|            range : MIN_40 <= L_var3 <= MAX_40.                        |
|                                                                       |
|   var2     16 bit short signed integer (Word16) whose value falls in  |
|            therange : MIN_16 <= var1 <= MAX_16.                       |
|                                                                       |
|  Return Value :                                                       |
|                                                                       |
|   L_var_out                                                           |
|            40 bit long signed integer (Word40) whose value falls in the  |
|            range : MIN_40 <= L_var_out <= MAX_40.                     |
|_______________________________________________________________________|
*/


Word40 L_shr40(Word40 L_var1, Word16 var2)
{
```

```
  Word40 L_var_out;

  Word40 L_shl40(Word40 L_var1, Word16 var2);



  if (var2 < 0)

  {

    L_var_out = L_shl40(L_var1,(Word16)(-var2));

  }

  else

  {

    L_var_out = L_shift_left40(L_var1, (Word16)(-var2));

  }

  return(L_var_out);

}



/*_____________________________________________________________________
```

| |

| Function Name : L_shl40 |

| |

| Purpose : |

| |

| Arithmetically shift the 40 bit input L_var1 left var2 positions. Zero |

| fill the var2 LSB of the result. If var2 is negative, L_var1 right by |

| -var2 arithmetically shift with sign extension. Saturate the result in |

| case of underflows or overflows. |

| |

| Complexity weight : 2 |

| |

| Inputs : |

| |

| L_var1 40 bit long signed integer (Word40) whose value falls in the |

```
|            range : MIN_40 <= L_var3 <= MAX_40.                        |
|                                                                       |
|   var2     16 bit short signed integer (Word16) whose value falls in  |
|            therange : MIN_16 <= var1 <= MAX_16.                       |
|                                                                       |
|  Return Value :                                                       |
|                                                                       |
|   L_var_out                                                           |
|            40 bit long signed integer (Word40) whose value falls in the  |
|            range : MIN_40 <= L_var_out <= MAX_40.                     |
|_______________________________________________________________________|
*/


Word40 L_shl40(Word40 L_var1, Word16 var2)
{
```

```
  Word40 L_var_out;




  if (var2 <= 0)

  {

    L_var_out = L_shr40(L_var1,(Word16)(-var2));



  }

  else

  {

    L_var_out = L_shift_left40(L_var1, var2);




    L_var_out = L_saturate40(L_var_out);

  }

  return(L_var_out);

}
```

```
/*_____________________________________________________________________
|                                                                     |
|   Function Name : L_sat32_40                                        |
|                                                                     |
|   Purpose :                                                         |
|                                                                     |
|    40 bit L_var1 is limited to the range MAX_32..MIN_32.  L_var1 is set   |
|    to MAX_32 on overflow or MIN_32 on underflow.                    |
|                                                                     |
|   Complexity weight : 1                                             |
|                                                                     |
|   Inputs :                                                          |
|                                                                     |
|    L_var1   40 bit long signed integer (Word40) whose value falls in the  |
```

```
|            range : MIN_40 <= var1 <= MAX_40.                          |
|                                                                       |
|  Return Value :                                                       |
|                                                                       |
|   L_var_out                                                           |
|            40 bit long signed integer (Word40) whose value falls in the  |
|            range : MIN_32 <= var_out <= MAX_32.                       |
|_______________________________________________________________________|
*/


Word40 L_sat32_40(Word40 L_var1)
{
  Word40 L_var_out;

  if (L_var1 > MAX_32)
```

```
  {

    L_var_out = MAX_32;

    giOverflow = 1;

  }

  else if (L_var1 < MIN_32)

  {

    L_var_out = MIN_32;

    giOverflow = 1;

  }

  else L_var_out = L_var1;

  return (L_var_out);

}









/*_____________________________________________________________________
```

| |

| Function Name : norm32_l40 |

| |

| Purpose : |

| |

| Produces the number of left shift needed to normalize in 32 bits format |

| the 40 bit variable l_var1 for positive values on the interval with |

| minimum of (MAX_32+1)/2 and maximum of MAX_32, and for negative values |

| on the interval with minimum of MIN_32 and maximum of (MIN_32/2)+1; |

| in order to normalize the result, the following operation must be done: |

| |

| norm_L_var1 = L_shl40(L_var1,norm32_l(L_var1)). |

| |

| Complexity weight : 3 |

| |

```
|   Inputs :                                                    |

|                                                               |

|    L_var1   40 bit long signed integer (Word40) whose value falls in the  |

|             range : MIN_40 <= var1 <= MAX_40.                 |

|                                                               |

|   Return Value :                                              |

|                                                               |

|    var_out  16 bit short signed integer (Word16) whose value falls in  |

|             therange : -8 <= var_out <= 31.                   |

|_______________________________________________________________|

*/



Word16 norm32_l40(Word40 L_var1)

{

  Word16 var_out;
```

```
L_var1 = floor(L_var1);

var_out = 0;

if (L_var1 != 0)

{

  while ((L_var1 > MIN_32) && (L_var1 < MAX_32))

  {

    L_var1 = L_shift_left40(L_var1, 1);

    var_out++;

  }

  while ((L_var1 < MIN_32) || (L_var1 > MAX_32))

  {

    L_var1 = L_shift_left40(L_var1, -1);

    var_out--;

  }
```

```
 }

 return(var_out);

}

/*____________________________________________________________________________

|                                                            |

|   Function Name : extract_l40                              |

|                                                            |

|   Purpose :                                                |

|                                                            |

|   Get the lower 16 bits of a 40-bit variable into a 16-bit variable       |

|                                                            |

|   Complexity weight : 1                                    |

|                                                            |
```

```
|   Inputs :                                                    |
|                                                               |
|    L_var1   40 bit long signed integer (Word40) whose value falls in the  |
|             range : MIN_40 <= var1 <= MAX_40.                 |
|                                                               |
|   Return Value :                                              |
|                                                               |
|    var_out  16 bit short signed integer (Word16) which may take any value |
|_______________________________________________________________|
*/


Word16 extract_l40(Word40 L_var1)
{
  Word16 out;
```

```
  out = (Word16)(L_var1 - floor(L_var1/65536.)*65536.);

  return (out);
}

/*____________________________________________________________________________
 |                                                            |
 |   Function Name : L_deposit_l40                            |
 |                                                            |
 |   Purpose :                                                |
 |                                                            |
 |   Replace lower 16 bits of a 40-bit variable by those of 16-bit input     |
 |   variable var_in                                          |
 |                                                            |
 |   Complexity weight : 1                                    |
 |                                                            |
```

```
|   Inputs :                                                    |
|                                                               |
|    L_var1   40 bit long signed integer (Word40) whose value falls in the  |
|             range : MIN_40 <= var1 <= MAX_40.                 |
|    var_in   16 bit short signed integer (Word16) which may take any value |
|                                                               |
|   Return Value :                                              |
|                                                               |
|             40 bit long signed integer (Word40) whose value falls in the  |
|             range : MIN_40 <= var1 <= MAX_40.                 |
|_______________________________________________________________|
*/


Word40 L_deposit_l40(Word40 L_var1, Word16 var_in)
{
```

```
  Word40 out;

  Word32 tmp32;



  tmp32 = 0x0000ffff & (Word32)var_in;  /* no sign extend  */

  out = L_add40(floor(L_var1/65536.)*65536., tmp32);

  return (out);

}
```