Technical Specification Group Services and System Aspects
Meeting #11, Palm Springs, USA, 19-22 March 2001

TSGS#11(01)0100

Source: **TSG-SA WG4**

Title: **CRs to TS 06.73 and TS 26.073 on Corrections to AMR codec (R98, R99, Release 4)**

Document for: **Approval**

Agenda Item: **7.4.3**

The following CRs were agreed at the TSG-SA WG4 meetings #16 and are presented to TSG SA #11 for approval.

Spec	CR	Rev	Phase	Subject	Cat	Ver	WG	Meeting	S4 doc
06.73	A023		R98	Correction of potential bug in AMR decoder due to usage of standard C abs() function	F	7.4.1	S4	TSG-SA WG4#16	S4-010214
26.073	003		R99	Correction of potential bug in AMR decoder due to usage of standard C abs() function	A	3.1.0	S4	TSG-SA WG4#16	S4-010198
26.073	004		Rel-4	Correction of potential bug in AMR decoder due to usage of standard C abs() function	A	3.1.0	S4	TSG-SA WG4#16	S4-010199
06.73	A027		R98	Correction of potential bug in AMR decoder due to the usage of standard C abs() function (VAD option_2)	F	7.4.1	S4	TSG-SA WG4#16	S4-010220
26.073	011		R99	Correction of potential bug in AMR decoder due to the usage of standard C abs() function (VAD option_2)	A	3.1.0	S4	TSG-SA WG4#16	S4-010221
26.073	012		Rel-4	Correction of potential bug in AMR decoder due to the usage of standard C abs() function (VAD option_2)	A	3.1.0	S4	TSG-SA WG4#16	S4-010262
06.73	A024		R98	Correction of comfort noise parameter interpolation bug of AMR decoder	F	7.4.1	S4	TSG-SA WG4#16	S4-010215
26.073	005		R99	Correction of comfort noise parameter interpolation bug of AMR decoder	A	3.1.0	S4	TSG-SA WG4#16	S4-010200
26.073	006		Rel-4	Correction of comfort noise parameter interpolation bug of AMR decoder	A	3.1.0	S4	TSG-SA WG4#16	S4-010201
06.73	A025		R98	Correction of mode state bug in AMR decoder	F	7.4.1	S4	TSG-SA WG4#16	S4-010216
26.073	007		R99	Correction of mode state bug in AMR decoder	A	3.1.0	S4	TSG-SA WG4#16	S4-010202
26.073	008		Rel-4	Correction of mode state bug in AMR decoder	A	3.1.0	S4	TSG-SA WG4#16	S4-010203
06.73	A026	1	R98	Correction of TX_TYPE and RX_TYPE identifiers	F	7.4.1	S4	TSG-SA WG4#16	S4-010282
26.073	009	1	R99	Correction of TX_TYPE and RX_TYPE identifiers	A	3.1.0	S4	TSG-SA WG4#16	S4-010283
26.073	010	1	Rel-4	Correction of TX_TYPE and RX_TYPE identifiers	A	3.1.0	S4	TSG-SA WG4#16	S4-010284

3GPP TSG-SA WG4 Meeting #16
February 26 – March 2, 2001; Sophia Antipolis, France

Tdoc S4 (01)0214

CR-Form-v3

CHANGE REQUEST

⌘ **06.73** **CR** **A023** ⌘ rev **-** ⌘ Current version: **7.4.1** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE **X** Radio Access Network **X** Core Network ☐

Title:	⌘	CR 023 to TS 06.73 (R98) on Correction of potential bug in AMR decoder due to usage of standard C abs() function
Source:	⌘	TSG-SA WG4
Work item code:	⌘	AMR
Date:	⌘	21.02.01
Category:	⌘	**F**
Release:	⌘	R98

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	• In file dtx_dec.c the standard C library function abs() is used instead of the corresponding functions abs_s() of the set of ETSI basic operators. The abs() function does not return the correct result for –32768 as input value.
Summary of change:	⌘	• The file dtx_dec.c has to be changed in one line
Consequences if not approved:	⌘	In rare cases, the decoder might produce incorrect results causing unpleasant sounds.

Clauses affected:	⌘			
Other specs affected:	⌘	☐ Other core specifications ☐ Test specifications ☐ O&M Specifications	⌘	
Other comments:	⌘	It should be noted that the change does not affect the test sequences since they do not trigger the described bug.		

1. How the code is changed

1.1 File `dtx_dec.c`

1.1.1 Before the change (line 354)

```
st->lsf_hist_mean[i+j*M] = abs(st->lsf_hist_mean[i+j*M]);
```

1.1.2 After the change

```
st->lsf_hist_mean[i+j*M] = abs_s(st->lsf_hist_mean[i+j*M]);
```

3GPP TSG-SA WG4 Meeting #16
February 26 – March 2, 2001; Sophia Antipolis, France

Tdoc S4 (01)0198

CR-Form-v3

CHANGE REQUEST

⌘ **26.073** **CR** **A003** ⌘ rev **-** ⌘ Current version: **3.1.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE **X** Radio Access Network **X** Core Network ☐

Title:	⌘	CR 003 to TS 26.073 (R99) on Correction of potential bug in AMR decoder due to usage of standard C abs() function
Source:	⌘	TSG-SA WG4
Work item code:	⌘	AMR
Date:	⌘	21.02.01
Category:	⌘	**A**
Release:	⌘	R99

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	• In file dtx_dec.c the standard C library function abs() is used instead of the corresponding functions abs_s() of the set of ETSI basic operators. The abs() function does not return the correct result for –32768 as input value.
Summary of change:	⌘	• The file dtx_dec.c has to be changed in one line
Consequences if not approved:	⌘	In rare cases, the decoder might produce incorrect results causing unpleasant sounds.

Clauses affected:	⌘			
Other specs affected:	⌘	☐ Other core specifications ☐ Test specifications ☐ O&M Specifications	⌘	
Other comments:	⌘	It should be noted that the change does not affect the test sequences since they do not trigger the described bug.		

1. How the code is changed

1.1 File `dtx_dec.c`

1.1.1 Before the change (line 354)

```
st->lsf_hist_mean[i+j*M] = abs(st->lsf_hist_mean[i+j*M]);
```

1.1.2 After the change

```
st->lsf_hist_mean[i+j*M] = abs_s(st->lsf_hist_mean[i+j*M]);
```

3GPP TSG-SA WG4 Meeting #16
February 26 – March 2, 2001; Sophia Antipolis, France

Tdoc S4 (01)0199

CR-Form-v3

CHANGE REQUEST

⌘ **26.073** **CR** **A004** ⌘ rev **-** ⌘ Current version: **3.1.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE **X** Radio Access Network **X** Core Network ☐

Title:	⌘	CR 004 to TS 26.073 (Rel-4) on Correction of potential bug in AMR decoder due to usage of standard C abs() function
Source:	⌘	TSG-SA WG4
Work item code:	⌘	AMR
Date:	⌘	21.02.01
Category:	⌘	**A**
Release:	⌘	Rel-4

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	• In file dtx_dec.c the standard C library function abs() is used instead of the corresponding functions abs_s() of the set of ETSI basic operators. The abs() function does not return the correct result for –32768 as input value.
Summary of change:	⌘	• The file dtx_dec.c has to be changed in one line
Consequences if not approved:	⌘	In rare cases, the decoder might produce incorrect results causing unpleasant sounds.

Clauses affected:	⌘	
Other specs affected:	⌘	☐ Other core specifications ⌘ ☐ Test specifications ☐ O&M Specifications
Other comments:	⌘	It should be noted that the change does not affect the test sequences since they do not trigger the described bug.

1. How the code is changed

1.1 File `dtx_dec.c`

1.1.1 Before the change (line 354)

```
st->lsf_hist_mean[i+j*M] = abs(st->lsf_hist_mean[i+j*M]);
```

1.1.2 After the change

```
st->lsf_hist_mean[i+j*M] = abs_s(st->lsf_hist_mean[i+j*M]);
```

3GPP TSG-SA WG4 Meeting #16
February 26 – March 2, 2001; Sophia Antipolis, France

Tdoc S4 (01)0220

CR-Form-v3

CHANGE REQUEST

⌘ **06.73** **CR** **A027** ⌘ rev **-** ⌘ Current version: **7.3.1** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☒ Radio Access Network ☒ Core Network ☐

Title:	⌘	CR 027 to TS 06.73 (R98) on Correction of potential bug in AMR decoder due to usage of standard C abs() function (VAD2)
Source:	⌘	TSG-SA WG4
Work item code:	⌘	AMR
Date:	⌘	1.03.01
Category:	⌘	**F**
Release:	⌘	R98

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	• In file vad2.c the standard C library function abs() is used instead of the corresponding functions abs_s() of the set of ETSI basic operators..
Summary of change:	⌘	• The file vad2.c has to be changed in two lines
Consequences if not approved:	⌘	In rare cases, the decoder might produce incorrect results causing unpleasant sounds.

Clauses affected:	⌘	
Other specs affected:	⌘	☐ Other core specifications ⌘ ☐ Test specifications ☐ O&M Specifications
Other comments:	⌘	It should be noted that the change does not affect the test sequences since they do not trigger the described bug.

1. How the code is changed

1.1 File `vad2.c`

1.1.1 Before the change (Lines 169 & 172)

```
169: max = abs(in[0]);
172:   adata = abs(in[i]);         test();
```

1.1.2 After the change

```
169: max = abs_s(in[0]);
172:   adata = abs_s(in[i]);       test();
```

CR-Form-v3

CHANGE REQUEST

⌘ **26.073** **CR** **A011** ⌘ rev **-** ⌘ Current version: **3.1.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☒ Radio Access Network ☒ Core Network ☐

Title:	⌘	CR 011 to TS 26.073 (R99) on Correction of potential bug in AMR decoder due to usage of standard C abs() function (VAD2)
Source:	⌘	TSG-SA WG4
Work item code:	⌘	AMR
Date:	⌘	1.03.01
Category:	⌘	**A**
Release:	⌘	R99

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	• In file vad2.c the standard C library function abs() is used instead of the corresponding functions abs_s() of the set of ETSI basic operators.
Summary of change:	⌘	• The file vad2.c has to be changed in two lines
Consequences if not approved:	⌘	In rare cases, the decoder might produce incorrect results causing unpleasant sounds.

Clauses affected:	⌘	
Other specs affected:	⌘	☐ Other core specifications ⌘ ☐ Test specifications ☐ O&M Specifications
Other comments:	⌘	It should be noted that the change does not affect the test sequences since they do not trigger the described bug.

1. How the code is changed

1.1 File `vad2.c`

1.1.1 Before the change (Lines 169 & 172)

```
169: max = abs(in[0]);
172:   adata = abs(in[i]);         test();
```

1.1.2 After the change

```
169: max = abs_s(in[0]);
172:   adata = abs_s(in[i]);       test();
```

3GPP TSG-SA WG4 Meeting #16
February 26 – March 2, 2001; Sophia Antipolis, France

Tdoc S4 (01)0262

CR-Form-v3

CHANGE REQUEST

⌘ **26.073** **CR** **A012** ⌘ rev **-** ⌘ Current version: **3.1.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE **X** Radio Access Network **X** Core Network ☐

Title:	⌘	CR 012 to TS 26.073 (Rel-4) on Correction of potential bug in AMR decoder due to usage of standard C abs() function (VAD2)
Source:	⌘	TSG_SA WG4
Work item code:	⌘	AMR
Date:	⌘	1.03.01
Category:	⌘	**A**
Release:	⌘	Rel-4

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	• In file vad2.c the standard C library function abs() is used instead of the corresponding functions abs_s() of the set of ETSI basic operators.
Summary of change:	⌘	• The file vad2.c has to be changed in two lines.
Consequences if not approved:	⌘	In rare cases, the decoder might produce incorrect results causing unpleasant sounds.

Clauses affected:	⌘			
Other specs Affected:	⌘	☐ Other core specifications ☐ Test specifications ☐ O&M Specifications	⌘	
Other comments:	⌘	It should be noted that the change does not affect the test sequences since they do not trigger the described bug.		

1. How the code is changed

1.1 File `vad2.c`

1.1.1 Before the change (Lines 169 & 172)

```
169: max = abs(in[0]);
172:   adata = abs(in[i]);         test();
```

1.1.2 After the change

```
169: max = abs_s(in[0]);
172:   adata = abs_s(in[i]);       test();
```

CR-Form-v3

CHANGE REQUEST

⌘ **06.73** **CR** **A024** ⌘ rev **-** ⌘ Current version: **7.4.1** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☒ Radio Access Network ☒ Core Network ☐

Title:	⌘	CR 024 to TS 06.73 (R98) on Correction of comfort noise parameter interpolation bug of AMR decoder
Source:	⌘	TSG-SA WG4
Work item code:	⌘	AMR
Date:	⌘	21.02.01
Category:	⌘	**F**
Release:	⌘	R98

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	• Comfort noise parameters of consecutive SID_UPDATE frames are interpolated incorrectly
Summary of change:	⌘	• The file dtx_dec.c has to be changed in two lines
Consequences if not approved:	⌘	The comfort noise performance for certain types of noise is significantly worse than possible.

Clauses affected:	⌘			
Other specs affected:	⌘	☐ Other core specifications ☒ Test specifications ☐ O&M Specifications	⌘	06.74/26.074 (Test sequences)
Other comments:	⌘	A detailed motivation for the CR is given in Tdoc S4-010218: “Correction of comfort noise parameter interpolation bug of AMR decoder”		

1. How the code is changed

1.1 File `dtx_dec.c`

1.1.1 Before the change (lines 391…400)

```
/* Compute interpolation factor, since the division only works *
 * for values of since_last_sid < 32 we have to limit the      *
 * interpolation to 32 frames                                  */

tmp_int_length = st->since_last_sid;                     move16();

test();
if (sub(tmp_int_length, 32) > 0)
  {
  tmp_int_length = 32;                                   move16();
  }
```

1.1.2 After the change

```
/* Compute interpolation factor, since the division only works *
 * for values of since_last_sid < 32 we have to limit the      *
 * interpolation to 32 frames                                  */

tmp_int_length = st->since_last_sid;                     move16();
st->since_last_sid = 0;                                  move16();

test();
if (sub(tmp_int_length, 32) > 0)
  {
  tmp_int_length = 32;                                   move16();
  }
```

1.2.1 Before the change (lines 477..479)

```
/* Interpolate SID info */
int_fac = shl(st->since_last_sid, 10); /* Q10 */                 move16();
int_fac = mult(int_fac, st->true_sid_period_inv); /* Q10 * Q15 -> Q10 */
```

1.2.2 After the change

```
/* Interpolate SID info */
int_fac = shl(add(1,st->since_last_sid), 10); /* Q10 */
move16();
int_fac = mult(int_fac, st->true_sid_period_inv); /* Q10 * Q15 -> Q10 */
```

3GPP TSG-SA WG4 Meeting #16
February 26 – March 2, 2001; Sophia Antipolis, France

Tdoc S4 (01)0200

CR-Form-v3

CHANGE REQUEST

⌘ **26.073** **CR** **A005** ⌘ rev **-** ⌘ Current version: **3.1.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE **X** Radio Access Network **X** Core Network ☐

Title:	⌘	CR 005 to TS 26.073 (R99) on Correction of comfort noise parameter interpolation bug of AMR decoder
Source:	⌘	TSG-SA WG4
Work item code:	⌘	AMR
Date:	⌘	21.02.01
Category:	⌘	**A**
Release:	⌘	R99

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	• Comfort noise parameters of consecutive SID_UPDATE frames are interpolated incorrectly
Summary of change:	⌘	• The file dtx_dec.c has to be changed in two lines
Consequences if not approved:	⌘	The comfort noise performance for certain types of noise is significantly worse than possible.

Clauses affected:	⌘			
Other specs affected:	⌘	☐ Other core specifications **X** Test specifications ☐ O&M Specifications	⌘	06.74/26.074 (Test sequences)
Other comments:	⌘	A detailed motivation for the CR is given in Tdoc S4-010218: “Correction of comfort noise parameter interpolation bug of AMR decoder”		

1. How the code is changed

1.1 File `dtx_dec.c`

1.1.1 Before the change (lines 391…400)

```
/* Compute interpolation factor, since the division only works *
 * for values of since_last_sid < 32 we have to limit the      *
 * interpolation to 32 frames                                  */

tmp_int_length = st->since_last_sid;                     move16();

test();
if (sub(tmp_int_length, 32) > 0)
  {
  tmp_int_length = 32;                                   move16();
  }
```

1.1.2 After the change

```
/* Compute interpolation factor, since the division only works *
 * for values of since_last_sid < 32 we have to limit the      *
 * interpolation to 32 frames                                  */

tmp_int_length = st->since_last_sid;                     move16();
st->since_last_sid = 0;                                  move16();

test();
if (sub(tmp_int_length, 32) > 0)
  {
  tmp_int_length = 32;                                   move16();
  }
```

1.2.1 Before the change (lines 477..479)

```
/* Interpolate SID info */
int_fac = shl(st->since_last_sid, 10); /* Q10 */                 move16();
int_fac = mult(int_fac, st->true_sid_period_inv); /* Q10 * Q15 -> Q10 */
```

1.2.2 After the change

```
/* Interpolate SID info */
int_fac = shl(add(1,st->since_last_sid), 10); /* Q10 */
move16();
int_fac = mult(int_fac, st->true_sid_period_inv); /* Q10 * Q15 -> Q10 */
```

3GPP TSG-SA WG4 Meeting #16
February 26 – March 2, 2001; Sophia Antipolis, France

Tdoc S4 (01)0201

CR-Form-v3

CHANGE REQUEST

⌘ **26.073** **CR** **A006** ⌘ rev **-** ⌘ Current version: **3.1.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE [X] Radio Access Network [X] Core Network ☐

Title:	⌘	CR 006 to TS 26.073 (Rel-4) on Correction of comfort noise parameter interpolation bug of AMR decoder
Source:	⌘	TSG-SA WG4
Work item code:	⌘	AMR
Date:	⌘	21.02.01
Category:	⌘	**A**
Release:	⌘	Rel-4

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	• Comfort noise parameters of consecutive SID_UPDATE frames are interpolated incorrectly
Summary of change:	⌘	• The file dtx_dec.c has to be changed in two lines
Consequences if not approved:	⌘	The comfort noise performance for certain types of noise is significantly worse than possible.

Clauses affected:	⌘			
Other specs affected:	⌘	☐ Other core specifications [X] Test specifications ☐ O&M Specifications	⌘	06.74/26.074 (Test sequences)
Other comments:	⌘	A detailed motivation for the CR is given in Tdoc S4-010218: “Correction of comfort noise parameter interpolation bug of AMR decoder”		

1. How the code is changed

1.1 File `dtx_dec.c`

1.1.1 Before the change (lines 391…400)

```
/* Compute interpolation factor, since the division only works *
 * for values of since_last_sid < 32 we have to limit the      *
 * interpolation to 32 frames                                  */

tmp_int_length = st->since_last_sid;                     move16();

test();
if (sub(tmp_int_length, 32) > 0)
  {
  tmp_int_length = 32;                                   move16();
  }
```

1.1.2 After the change

```
/* Compute interpolation factor, since the division only works *
 * for values of since_last_sid < 32 we have to limit the      *
 * interpolation to 32 frames                                  */

tmp_int_length = st->since_last_sid;                     move16();
st->since_last_sid = 0;                                  move16();

test();
if (sub(tmp_int_length, 32) > 0)
  {
  tmp_int_length = 32;                                   move16();
  }
```

1.2.1 Before the change (lines 477..479)

```
/* Interpolate SID info */
int_fac = shl(st->since_last_sid, 10); /* Q10 */                 move16();
int_fac = mult(int_fac, st->true_sid_period_inv); /* Q10 * Q15 -> Q10 */
```

1.2.2 After the change

```
/* Interpolate SID info */
int_fac = shl(add(1,st->since_last_sid), 10); /* Q10 */
move16();
int_fac = mult(int_fac, st->true_sid_period_inv); /* Q10 * Q15 -> Q10 */
```

CR-Form-v3

CHANGE REQUEST

⌘ **06.73** **CR** **A025** ⌘ rev **-** ⌘ Current version: **7.4.1** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE **X** Radio Access Network **X** Core Network ☐

Title:	⌘	CR 025 to TS 06. 73 (R98) on Correction of mode state bug in AMR decoder
Source:	⌘	TSG-SA WG4
Work item code:	⌘	AMR
Date:	⌘	21.02.01
Category:	⌘	**F**
Release:	⌘	R98

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	• The AMR decoder does not have a state variable of the codec mode applied in the preceding frame. This state is needed whenever NO_DATA frames are signaled to the decoder since these frames do not convey valid mode information. When checking with the decoder test sequences, the bug is not noticeable since the decoder input sequences "*.COD" do contain valid mode information even for NO_DATA frames.
Summary of change:	⌘	• The bug is fixed by adding a state variable to the Speech_Decode_FrameState struct defined in sp_dec.h. Further code need to be changed at one location in decoder.c and one location in sp_dec.c. • For formal reasons, the top-level speech encoder program file coder.c has to be changed such that no longer for NO_DATA frames valid mode information is inserted into the encoder output sequence files.
Consequences if not approved:	⌘	The AMR decoder crashes when, during NO_DATA frames, it is supplied with an invalid mode outside the allowed range of 0…7. Supplying the decoder with an incorrect mode within the allowed range may decrease quality of the reconstructed speech/comfort noise signal.

Clauses affected:	⌘			
Other specs Affected:	⌘	☐ Other core specifications **X** Test specifications ☐ O&M Specifications	⌘	GSM 06.74, 3GPP 26.074: CR to the test sequences
Other comments:	⌘			

1. How the code is changed

1.1 File decoder.c

1.1.1 Before the change (lines 171…178)

```
/* get frame type and mode information from frame */
if (rxframetypeMode) {
  rx_type = (enum RXFrameType)serial[0];
} else {
  tx_type = (enum TXFrameType)serial[0];
  rx_type = tx_to_rx (tx_type);
}
mode = (enum Mode) serial[1+MAX_SERIAL_SIZE];
```

1.1.2 After the change

```
/* get frame type and mode information from frame */
if (rxframetypeMode) {
  rx_type = (enum RXFrameType)serial[0];
} else {
  tx_type = (enum TXFrameType)serial[0];
  rx_type = tx_to_rx (tx_type);
}
mode = (enum Mode) serial[1+MAX_SERIAL_SIZE];
if (rx_type == RX_NO_DATA) {
  mode = speech_decoder_state->prev_mode;
}
else {
  speech_decoder_state->prev_mode = mode;
}
```

2.1 File sp_dec.c

2.1.1 Before the change (lines 120…126)

```
Decoder_amr_reset(state->decoder_amrState, (enum Mode)0);
Post_Filter_reset(state->post_state);
Post_Process_reset(state->postHP_state);

setCounter(state->complexityCounter);
Init_WMOPS_counter();
setCounter(0); /* set counter to global counter */
```

2.1.2 After the change

```
Decoder_amr_reset(state->decoder_amrState, (enum Mode)0);
Post_Filter_reset(state->post_state);
Post_Process_reset(state->postHP_state);

state->prev_mode = (enum Mode)0;

setCounter(state->complexityCounter);
Init_WMOPS_counter();
setCounter(0); /* set counter to global counter */
```

3.1 File sp_dec.h

3.1.1 Before the change (lines 33…39)

```
typedef struct{
```

```
  Decoder_amrState* decoder_amrState;
  Post_FilterState*  post_state;
  Post_ProcessState* postHP_state;

  int complexityCounter;   /* Only for complexity computation            */
} Speech_Decode_FrameState;
```

3.1.2 After the change

```
typedef struct{
  Decoder_amrState* decoder_amrState;
  Post_FilterState*  post_state;
  Post_ProcessState* postHP_state;
  enum Mode prev_mode;

  int complexityCounter;   /* Only for complexity computation            */
} Speech_Decode_FrameState;
```

4.1 File `coder.c`

4.1.1 Before the change (lines 262…267)

```
/* include frame type and mode information in serial bitstream */
sid_sync (sid_state, used_mode, &tx_type);
serial[0] = tx_type;
serial[1+MAX_SERIAL_SIZE] = mode;

/* write bitstream to output file */
```

4.1.2 After the change

```
/* include frame type and mode information in serial bitstream */
sid_sync (sid_state, used_mode, &tx_type);
serial[0] = tx_type;
if (tx_type != TX_NO_DATA) {
  serial[1+MAX_SERIAL_SIZE] = mode;
}
else {
  serial[1+MAX_SERIAL_SIZE] = -1;
}

/* write bitstream to output file */
```

3GPP TSG-SA WG4 Meeting #16
February 26 – March 2, 2001; Sophia Antipolis, France

Tdoc S4 (01)0202

CR-Form-v3

CHANGE REQUEST

⌘ **26.073** **CR** **A007** ⌘ rev **-** ⌘ Current version: **3.1.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☒ Radio Access Network ☒ Core Network ☐

Title:	⌘	CR 007 to TS 26.073 (R99) on Correction of mode state bug in AMR decoder
Source:	⌘	TSG-SA WG4
Work item code:	⌘	AMR
Date:	⌘	21.02.01
Category:	⌘	**A**
Release:	⌘	R99

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	• The AMR decoder does not have a state variable of the codec mode applied in the preceding frame. This state is needed whenever NO_DATA frames are signaled to the decoder since these frames do not convey valid mode information. When checking with the decoder test sequences, the bug is not noticeable since the decoder input sequences "*.COD" do contain valid mode information even for NO_DATA frames.
Summary of change:	⌘	• The bug is fixed by adding a state variable to the Speech_Decode_FrameState struct defined in sp_dec.h. Further code need to be changed at one location in decoder.c and one location in sp_dec.c. • For formal reasons, the top-level speech encoder program file coder.c has to be changed such that no longer for NO_DATA frames valid mode information is inserted into the encoder output sequence files.
Consequences if not approved:	⌘	The AMR decoder crashes when, during NO_DATA frames, it is supplied with an invalid mode outside the allowed range of 0…7. Supplying the decoder with an incorrect mode within the allowed range may decrease quality of the reconstructed speech/comfort noise signal.

Clauses affected:	⌘			
Other specs Affected:	⌘	☐ Other core specifications ☒ Test specifications ☐ O&M Specifications	⌘	GSM 06.74, 3GPP 26.074: CR to the test sequences
Other comments:	⌘			

1. How the code is changed

1.1 File decoder.c

1.1.1 Before the change (lines 171…178)

```
/* get frame type and mode information from frame */
if (rxframetypeMode) {
  rx_type = (enum RXFrameType)serial[0];
} else {
  tx_type = (enum TXFrameType)serial[0];
  rx_type = tx_to_rx (tx_type);
}
mode = (enum Mode) serial[1+MAX_SERIAL_SIZE];
```

1.1.2 After the change

```
/* get frame type and mode information from frame */
if (rxframetypeMode) {
  rx_type = (enum RXFrameType)serial[0];
} else {
  tx_type = (enum TXFrameType)serial[0];
  rx_type = tx_to_rx (tx_type);
}
mode = (enum Mode) serial[1+MAX_SERIAL_SIZE];
if (rx_type == RX_NO_DATA) {
  mode = speech_decoder_state->prev_mode;
}
else {
  speech_decoder_state->prev_mode = mode;
}
```

2.1 File sp_dec.c

2.1.1 Before the change (lines 120…126)

```
Decoder_amr_reset(state->decoder_amrState, (enum Mode)0);
Post_Filter_reset(state->post_state);
Post_Process_reset(state->postHP_state);

setCounter(state->complexityCounter);
Init_WMOPS_counter();
setCounter(0); /* set counter to global counter */
```

2.1.2 After the change

```
Decoder_amr_reset(state->decoder_amrState, (enum Mode)0);
Post_Filter_reset(state->post_state);
Post_Process_reset(state->postHP_state);

state->prev_mode = (enum Mode)0;

setCounter(state->complexityCounter);
Init_WMOPS_counter();
setCounter(0); /* set counter to global counter */
```

3.1 File sp_dec.h

3.1.1 Before the change (lines 33…39)

```
typedef struct{
```

```
  Decoder_amrState* decoder_amrState;
  Post_FilterState*  post_state;
  Post_ProcessState* postHP_state;

  int complexityCounter;   /* Only for complexity computation            */
} Speech_Decode_FrameState;
```

3.1.2 After the change

```
typedef struct{
  Decoder_amrState* decoder_amrState;
  Post_FilterState*  post_state;
  Post_ProcessState* postHP_state;
  enum Mode prev_mode;

  int complexityCounter;   /* Only for complexity computation            */
} Speech_Decode_FrameState;
```

4.1 File `coder.c`

4.1.1 Before the change (lines 262…267)

```
/* include frame type and mode information in serial bitstream */
sid_sync (sid_state, used_mode, &tx_type);
serial[0] = tx_type;
serial[1+MAX_SERIAL_SIZE] = mode;

/* write bitstream to output file */
```

4.1.2 After the change

```
/* include frame type and mode information in serial bitstream */
sid_sync (sid_state, used_mode, &tx_type);
serial[0] = tx_type;
if (tx_type != TX_NO_DATA) {
  serial[1+MAX_SERIAL_SIZE] = mode;
}
else {
  serial[1+MAX_SERIAL_SIZE] = -1;
}

/* write bitstream to output file */
```

3GPP TSG-SA WG4 Meeting #16
February 26 – March 2, 2001; Sophia Antipolis, France

Tdoc S4 (01)0203

CR-Form-v3

CHANGE REQUEST

⌘ **26.073** **CR** **A008** ⌘ rev **-** ⌘ Current version: **3.1.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☒ Radio Access Network ☒ Core Network ☐

Title:	⌘	CR 008 to TS 26.073 (Rel-4) on Correction of mode state bug in AMR decoder
Source:	⌘	TSG-SA WG4
Work item code:	⌘	AMR
Date:	⌘	21.02.01
Category:	⌘	**A**
Release:	⌘	REL-4

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	• The AMR decoder does not have a state variable of the codec mode applied in the preceding frame. This state is needed whenever NO_DATA frames are signaled to the decoder since these frames do not convey valid mode information. When checking with the decoder test sequences, the bug is not noticeable since the decoder input sequences "*.COD" do contain valid mode information even for NO_DATA frames.
Summary of change:	⌘	• The bug is fixed by adding a state variable to the Speech_Decode_FrameState struct defined in sp_dec.h. Further code need to be changed at one location in decoder.c and one location in sp_dec.c. • For formal reasons, the top-level speech encoder program file coder.c has to be changed such that no longer for NO_DATA frames valid mode information is inserted into the encoder output sequence files.
Consequences if not approved:	⌘	The AMR decoder crashes when, during NO_DATA frames, it is supplied with an invalid mode outside the allowed range of 0…7. Supplying the decoder with an incorrect mode within the allowed range may decrease quality of the reconstructed speech/comfort noise signal.

Clauses affected:	⌘			
Other specs Affected:	⌘	☐ Other core specifications ☒ Test specifications ☐ O&M Specifications	⌘	GSM 06.74, 3GPP 26.074: CR to the test sequences
Other comments:	⌘			

1. How the code is changed

1.1 File decoder.c

1.1.1 Before the change (lines 171…178)

```
/* get frame type and mode information from frame */
if (rxframetypeMode) {
  rx_type = (enum RXFrameType)serial[0];
} else {
  tx_type = (enum TXFrameType)serial[0];
  rx_type = tx_to_rx (tx_type);
}
mode = (enum Mode) serial[1+MAX_SERIAL_SIZE];
```

1.1.2 After the change

```
/* get frame type and mode information from frame */
if (rxframetypeMode) {
  rx_type = (enum RXFrameType)serial[0];
} else {
  tx_type = (enum TXFrameType)serial[0];
  rx_type = tx_to_rx (tx_type);
}
mode = (enum Mode) serial[1+MAX_SERIAL_SIZE];
if (rx_type == RX_NO_DATA) {
  mode = speech_decoder_state->prev_mode;
}
else {
  speech_decoder_state->prev_mode = mode;
}
```

2.1 File sp_dec.c

2.1.1 Before the change (lines 120…126)

```
Decoder_amr_reset(state->decoder_amrState, (enum Mode)0);
Post_Filter_reset(state->post_state);
Post_Process_reset(state->postHP_state);

setCounter(state->complexityCounter);
Init_WMOPS_counter();
setCounter(0); /* set counter to global counter */
```

2.1.2 After the change

```
Decoder_amr_reset(state->decoder_amrState, (enum Mode)0);
Post_Filter_reset(state->post_state);
Post_Process_reset(state->postHP_state);

state->prev_mode = (enum Mode)0;

setCounter(state->complexityCounter);
Init_WMOPS_counter();
setCounter(0); /* set counter to global counter */
```

3.1 File sp_dec.h

3.1.1 Before the change (lines 33…39)

```
typedef struct{
```

```
  Decoder_amrState* decoder_amrState;
  Post_FilterState*  post_state;
  Post_ProcessState* postHP_state;

  int complexityCounter;   /* Only for complexity computation            */
} Speech_Decode_FrameState;
```

3.1.2 After the change

```
typedef struct{
  Decoder_amrState* decoder_amrState;
  Post_FilterState*  post_state;
  Post_ProcessState* postHP_state;
  enum Mode prev_mode;

  int complexityCounter;   /* Only for complexity computation            */
} Speech_Decode_FrameState;
```

4.1 File `coder.c`

4.1.1 Before the change (lines 262…267)

```
/* include frame type and mode information in serial bitstream */
sid_sync (sid_state, used_mode, &tx_type);
serial[0] = tx_type;
serial[1+MAX_SERIAL_SIZE] = mode;

/* write bitstream to output file */
```

4.1.2 After the change

```
/* include frame type and mode information in serial bitstream */
sid_sync (sid_state, used_mode, &tx_type);
serial[0] = tx_type;
if (tx_type != TX_NO_DATA) {
  serial[1+MAX_SERIAL_SIZE] = mode;
}
else {
  serial[1+MAX_SERIAL_SIZE] = -1;
}

/* write bitstream to output file */
```

3GPP TSG-SA WG4 Meeting #16 ***Tdoc S4 (01)0282***
Sophia Antipolis, France, 26 Feb – 2 Mar 2001

CR-Form-v3

CHANGE REQUEST

⌘ **06.73** **CR** **A026** ⌘ rev **1** ⌘ Current version: **7.4.1** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE **X** Radio Access Network **X** Core Network ☐

Title:	⌘	Correction of the TX_TYPE and RX_TYPE identifiers
Source:	⌘	TSG-SA WG4
Work item code:	⌘	AMR
Date:	⌘	01-Mar-2001
Category:	⌘	**F**
Release:	⌘	R98

Use one of the following categories:
- ***F*** *(essential correction)*
- ***A*** *(corresponds to a correction in an earlier release)*
- ***B*** *(Addition of feature),*
- ***C*** *(Functional modification of feature)*
- ***D*** *(Editorial modification)*

Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	TX_TYPE and RX_TYPE identifiers are defined in two different ways in 3G TS 06.73 and 3G TS 06.93
Summary of change:	⌘	TX_TYPE and RX_TYPE identifiers are modified to be consistent with 3G TS 06.93, major changes are: • RX_SPARE removed, RX_ONSET added • RX_SPEECH_PROBABLY_DEGRADED -> RX_SPEECH_DEGRADED • TX_SPEECH ->TX_SPEECH_GOOD
Consequences if not approved:	⌘	Inconsistency between 3G TS 06.73 and 3G TS 06.93 and 3G TS 06.73 and 3G TS 08.60/08.61

Clauses affected:	⌘	Files: frame.h, sid_sync.c, decoder.c, dec_amr.c, dtx_dec.c, strfunc.c		
Other specs affected:	⌘	**X** Other core specifications ☐ Test specifications ☐ O&M Specifications	⌘	~~3G TS 06.74 AMR-NB test vectors for DTX~~
Other comments:	⌘	Sections 1 and 2 of this CR have been copied from S4 Tdoc (00)0123		

How to create CRs using this form:
Comprehensive information and tips about how to create CRs can be found at:
http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://www.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2000-09 contains the specifications resulting from the September 2000 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

1. Background

3G TS 06.93 specifies the TX- and RX-Type Identifiers by the two following tables:

Table 1: TX_TYPE identifiers

TX_TYPE Legend	Information Bits	Mode Indication
SPEECH_GOOD	speech frame, size 95..244 bits depending on codec mode; no errors known.	current code mode
SPEECH_DEGRADED (only in downlink in TFO)	Speech frame, size 95..244 bits, depending on codec mode; there might be errors in class 2 bits.	current codec mode
SPEECH_BAD (only in downlink in TFO)	Speech frame, size 95..244 bits, depending on codec mode; there are errors in class 1 bits.	current codec mode
SID_FIRST	marks the end of a talkspurt, respectively the beginning of a speech pause; does not contain information bits.	the codec mode that would have been used if TX_TYPE had been SPEECH
SID_UPDATE	comfort noise, 35 bits; no errors known	the codec mode that would have been used if TX_TYPE had been SPEECH
SID_BAD (only in downlink in TFO)	comfort noise, 35 bits; errors detected, parameters unusable	the codec mode that would have been used if TX_TYPE had been SPEECH
ONSET (only in downlink in TFO)	announces the beginning of a speech burst; does not contain information bits	the codec mode of the following speech frame
NO_DATA	no useful information	no useful information

Table 2: RX_TYPE identifiers

RX_TYPE Legend	Description
SPEECH_GOOD	Speech frame with CRC OK, Channel Decoder soft values also OK
SPEECH_DEGRADED	Speech frame with CRC OK, but 1B bits and class2 bits may be corrupted
SPEECH_BAD	(likely) speech frame, bad CRC (or very bad Channel Decoder measures)
SID_FIRST	first SID marks the beginning of a comfort noise period
SID_UPDATE	SID update frame (with correct CRC)
SID_BAD	Corrupt SID update frame (bad CRC; applicable only for SID_UPDATE frames)
ONSET	ONSET frames precede the first speech frame of a speech burst
NO_DATA	Nothing useable (for the speech decoder) was received. This applies for the cases of no received frames (DTX) or received FACCH or RATSCCH or SID_FILLER signalling frames.

In order to have full correspondence between the tables and the C-code, the CR proposes a change in the C-code. The change has no impact on the functionality of the C-code.

2. How the code is changed in file *frame.h*

2.1 Before the change (lines 6...42)

```
enum RXFrameType { RX_SPEECH_GOOD = 0,
                   RX_SPEECH_PROBABLY_DEGRADED,
                   RX_SPARE,
                   RX_SPEECH_BAD,
                   RX_SID_FIRST,
                   RX_SID_UPDATE,
                   RX_SID_BAD,
                   RX_NO_DATA,
                   RX_N_FRAMETYPES     /* number of frame types */
};

enum TXFrameType { TX_SPEECH = 0,
                   TX_SID_FIRST,
                   TX_SID_UPDATE,
                   TX_NO_DATA,
                   TX_N_FRAMETYPES     /* number of frame types */
};
```

2.2 After the change

```
/* Note: The order of the TX and RX Type identifiers has been chosen in  */
/*       the way below to be compatible to an earlier version of the     */
/*       AMR-NB C reference program.                                     */

enum RXFrameType { RX_SPEECH_GOOD = 0,
                   RX_SPEECH_DEGRADED,
                   RX_ONSET,
                   RX_SPEECH_BAD,
                   RX_SID_FIRST,
                   RX_SID_UPDATE,
                   RX_SID_BAD,
                   RX_ONSET,
                   RX_NO_DATA,
                   RX_N_FRAMETYPES     /* number of frame types */
};

enum TXFrameType { TX_SPEECH_GOOD = 0,
                   TX_SPEECH_DEGRADED,
                   TX_SPEECH_BAD,
                   TX_SID_FIRST,
                   TX_SID_UPDATE,
                   TX_SID_BAD,
                   TX_ONSET,
                   TX_NO_DATA,
                   TX_SPEECH_DEGRADED,
                   TX_SPEECH_BAD,
                   TX_SID_BAD,
                   TX_ONSET,
                   TX_N_FRAMETYPES     /* number of frame types */
};
```

3. How the code is changed in file *sid_sync.c*

3.1 Before the change (lines 66...72)

```
int sid_sync_reset (sid_syncState *st)
{
    st->sid_update_counter = 3;
    st->sid_handover_debt = 0;
    st->prev_ft = TX_SPEECH;
    return 0;
}
```

3.2 After the change

```
int sid_sync_reset (sid_syncState *st)
{
    st->sid_update_counter = 3;
    st->sid_handover_debt = 0;
    st->prev_ft = TX_SPEECH_GOOD;
    return 0;
}
```

3.3 Before the change (lines 105...109)

```
        if (st->prev_ft == TX_SPEECH)
        {
           *tx_frame_type = TX_SID_FIRST;
           st->sid_update_counter = 3;
        }
```

3.4 After the change

```
        if (st->prev_ft == TX_SPEECH_GOOD)
        {
           *tx_frame_type = TX_SID_FIRST;
           st->sid_update_counter = 3;
        }
```

3.5 Before the change (lines 133...137)

```
    else
    {
       st->sid_update_counter = st->sid_update_rate ;
       *tx_frame_type = TX_SPEECH;
    }
```

3.6 After the change

```
    else
    {
       st->sid_update_counter = st->sid_update_rate ;
       *tx_frame_type = TX_SPEECH_GOOD;
    }
```

4. How the code is changed in file *decoder.c*

4.1 Before the change (lines 61...72)

```
static enum RXFrameType tx_to_rx (enum TXFrameType tx_type)
{    switch (tx_type) {
      case TX_SPEECH:      return RX_SPEECH_GOOD;
      case TX_SID_FIRST:   return RX_SID_FIRST;
      case TX_SID_UPDATE:  return RX_SID_UPDATE;
      case TX_NO_DATA:     return RX_NO_DATA;
      default:
        fprintf(stderr, "tx_to_rx: unknown TX frame type %d\n", tx_type);
        exit(1);
    }
}
```

4.2 After the change

```
static enum RXFrameType tx_to_rx (enum TXFrameType tx_type)
{
    switch (tx_type) {
      case TX_SPEECH_GOOD:      return RX_SPEECH_GOOD;
      case TX_SPEECH_DEGRADED:  return RX_SPEECH_DEGRADED;
      case TX_SPEECH_BAD:       return RX_SPEECH_BAD;
      case TX_SID_FIRST:        return RX_SID_FIRST;
      case TX_SID_UPDATE:       return RX_SID_UPDATE;
      case TX_SID_BAD:          return RX_SID_BAD;
      case TX_ONSET:            return RX_ONSET;
      case TX_NO_DATA:          return RX_NO_DATA;
      default:
        fprintf(stderr, "tx_to_rx: unknown TX frame type %d\n", tx_type);
        exit(1);
    }
}
```

5. How the code is changed in file *dec_amr.c*

5.1 Before the change (lines 332...348)

```
    /* SPEECH action state machine  */
    test (); test (); test ();
    if ((sub(frame_type, RX_SPEECH_BAD) == 0) ||
        (sub(frame_type, RX_NO_DATA) == 0) ||
        (sub(frame_type, RX_SPARE) == 0))
    {
       bfi = 1;                                          move16 ();
       test(); test();
       if ((sub(frame_type, RX_NO_DATA) == 0) ||
           (sub(frame_type, RX_SPARE) == 0))
       {
          build_CN_param(&st->nodataSeed,
                         prmno[mode],
                         bitno[mode],
                         parm);
       }
    }
```

5.2 After the change

```
    /* SPEECH action state machine  */
    test (); test (); test ();
    if ((sub(frame_type, RX_SPEECH_BAD) == 0) ||
        (sub(frame_type, RX_NO_DATA) == 0) ||
        (sub(frame_type, RX_ONSET) == 0))
    {
       bfi = 1;                                          move16 ();
       test(); test();
       if ((sub(frame_type, RX_NO_DATA) == 0) ||
           (sub(frame_type, RX_ONSET) == 0))
       {
          build_CN_param(&st->nodataSeed,
                         prmno[mode],
                         bitno[mode],
                         parm);
       }
    }
```

5.3 Before the change (lines 349...352)

```
    else if (sub(frame_type, RX_SPEECH_PROBABLY_DEGRADED) == 0)
    {
       pdfi = 1;                                         move16 ();
    }
```

5.4 After the change

```
    else if (sub(frame_type, RX_SPEECH_DEGRADED) == 0)
    {
       pdfi = 1;                                         move16 ();
    }
```

6. How the code is changed in file *dtx_dec.c*

6.1 Before the change (lines 720...723)

```
-----------------------------------------------------------------
RX_NO_DATA,           | SPEECH       | DTX/(DTX_MUTE)| DTX_MUTE
RX_SPARE              |(class2 garb.)|               |
-----------------------------------------------------------------
```

6.2 After the change

```
-----------------------------------------------------------------
RX_NO_DATA,           | SPEECH       | DTX/(DTX_MUTE)| DTX_MUTE
RX_ONSET              |(class2 garb.)|               |
-----------------------------------------------------------------
```

6.3 Before the change (lines 734...758)

```
/* DTX if SID frame or previously in DTX{_MUTE} and (NO_RX OR
   BAD_SPEECH) */
test(); test(); test();
test(); test(); test();
test(); test();
if ((sub(frame_type, RX_SID_FIRST) == 0)   ||
    (sub(frame_type, RX_SID_UPDATE) == 0)  ||
    (sub(frame_type, RX_SID_BAD) == 0)     ||
    (((sub(st->dtxGlobalState, DTX) == 0) ||
      (sub(st->dtxGlobalState, DTX_MUTE) == 0)) &&
     ((sub(frame_type, RX_NO_DATA) == 0) ||
      (sub(frame_type, RX_SPEECH_BAD) == 0) ||
      (sub(frame_type, RX_SPARE) == 0))))
{
   newState = DTX;                                          move16();

   /* stay in mute for these input types */
   test(); test(); test(); test(); test();
   if ((sub(st->dtxGlobalState, DTX_MUTE) == 0) &&
       ((sub(frame_type, RX_SID_BAD) == 0) ||
        (sub(frame_type, RX_SID_FIRST) ==  0) ||
        (sub(frame_type, RX_SPARE) ==  0) ||
        (sub(frame_type, RX_NO_DATA) == 0)))
   {
      newState = DTX_MUTE;                                  move16();
   }
```

6.4 After the change

```
/* DTX if SID frame or previously in DTX{_MUTE} and (NO_RX OR
   BAD_SPEECH) */
test(); test(); test();
test(); test(); test();
test(); test();
   if ((sub(frame_type, RX_SID_FIRST) == 0)   ||
    (sub(frame_type, RX_SID_UPDATE) == 0)  ||
    (sub(frame_type, RX_SID_BAD) == 0)     ||
    (((sub(st->dtxGlobalState, DTX) == 0) ||
      (sub(st->dtxGlobalState, DTX_MUTE) == 0)) &&
     ((sub(frame_type, RX_NO_DATA) == 0) ||
      (sub(frame_type, RX_SPEECH_BAD) == 0) ||
      (sub(frame_type, RX_ONSET) == 0))))
{
   newState = DTX;                                          move16();

   /* stay in mute for these input types */
   test(); test(); test(); test(); test();
   if ((sub(st->dtxGlobalState, DTX_MUTE) == 0) &&
```

```
          ((sub(frame_type, RX_SID_BAD) == 0) ||
           (sub(frame_type, RX_SID_FIRST) ==  0) ||
           (sub(frame_type, RX_ONSET) ==  0) ||
           (sub(frame_type, RX_NO_DATA) == 0)))
      {
         newState = DTX_MUTE;                                    move16();
      }
```

6.5 Before the change (lines 7945...801)

```
   test(); test(); test(); test();
   if ((sub(frame_type, RX_SID_FIRST) == 0)  ||
       (sub(frame_type, RX_SID_UPDATE) == 0) ||
       (sub(frame_type, RX_SID_BAD) == 0)    ||
       (sub(frame_type, RX_NO_DATA) == 0))
   {
      encState = DTX;                                            move16();
   }
   else
   {
      encState = SPEECH;                                         move16();
   }
```

6.6 After the change

```
   test(); test(); test(); test(); test();
   if ((sub(frame_type, RX_SID_FIRST) == 0)  ||
       (sub(frame_type, RX_SID_UPDATE) == 0) ||
       (sub(frame_type, RX_SID_BAD) == 0)    ||
       (sub(frame_type, RX_ONSET) == 0)      ||
       (sub(frame_type, RX_NO_DATA) == 0))
   {
      encState = DTX;                                             move16();
   }
   else
   {
      encState = SPEECH;                                          move16();
   }
```

7. How the code is changed in file *strfunc.c*

7.1 Before the change (lines 66...88)

```
/*
 * frame type name <-> frame type id conversion table
 */
static const conv_table rxframetable[] = {
  {"RX_SPEECH_GOOD", RX_SPEECH_GOOD},
  {"RX_SPEECH_PROBABLY_DEGRADED", RX_SPEECH_PROBABLY_DEGRADED},
  {"RX_SPEECH_BAD", RX_SPEECH_BAD},
  {"RX_SID_FIRST", RX_SID_FIRST},
  {"RX_SID_UPDATE", RX_SID_UPDATE},
  {"RX_SID_BAD", RX_SID_BAD},
  {"RX_NO_DATA" , RX_NO_DATA},
  {NULL, -1}
};
/*
 * frame type name <-> frame type id conversion table
 */
static const conv_table txframetable[] = {
  {"TX_SPEECH", TX_SPEECH},
  {"TX_SID_FIRST", TX_SID_FIRST},
  {"TX_SID_UPDATE", TX_SID_UPDATE},
  {"TX_NO_DATA" , TX_NO_DATA},
  {NULL, -1}
};
```

7.2 After the change

```
/*
 * frame type name <-> frame type id conversion table
 */
static const conv_table rxframetable[] = {
  {"RX_SPEECH_GOOD", RX_SPEECH_GOOD},
  {"RX_SPEECH_DEGRADED", RX_SPEECH_DEGRADED},
  {"RX_SPEECH_BAD", RX_SPEECH_BAD},
  {"RX_SID_FIRST", RX_SID_FIRST},
  {"RX_SID_UPDATE", RX_SID_UPDATE},
  {"RX_SID_BAD", RX_SID_BAD},
  {"RX_ONSET", RX_ONSET},
  {"RX_NO_DATA", RX_NO_DATA},
  {NULL, -1}
};
/*
 * frame type name <-> frame type id conversion table
 */
static const conv_table txframetable[] = {
  {"TX_SPEECH_GOOD", TX_SPEECH_GOOD},
  {"TX_SPEECH_DEGRADED", TX_SPEECH_DEGRADED},
  {"TX_SPEECH_BAD", TX_SPEECH_BAD},
  {"TX_SID_FIRST", TX_SID_FIRST},
  {"TX_SID_UPDATE", TX_SID_UPDATE},
  {"TX_SID_BAD", TX_SID_BAD},
  {"TX_ONSET", TX_ONSET},
  {"TX_NO_DATA", TX_NO_DATA},
  {NULL, -1}
};
```

3GPP TSG-SA WG4 Meeting #16 ***Tdoc S4 (01)0283***
Sophia Antipolis, France, 26 Feb – 2 Mar 2001

CR-Form-v3

CHANGE REQUEST

⌘ **26.073** **CR** **009** ⌘ rev **1** ⌘ Current version: **3.1.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM [ ] ME/UE [X] Radio Access Network [X] Core Network [ ]

Title:	⌘	Correction of the TX_TYPE and RX_TYPE identifiers
Source:	⌘	TSG-SA WG4
Work item code:	⌘	AMR
Date:	⌘	01-Mar-2001
Category:	⌘	**A**
Release:	⌘	R99

Use one of the following categories:
- ***F*** *(essential correction)*
- ***A*** *(corresponds to a correction in an earlier release)*
- ***B*** *(Addition of feature),*
- ***C*** *(Functional modification of feature)*
- ***D*** *(Editorial modification)*

Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	TX_TYPE and RX_TYPE identifiers are defined in two different ways in 3G TS 26.073 and 3G TS 26.093
Summary of change:	⌘	TX_TYPE and RX_TYPE identifiers are modified to be consistent with 3G TS 26.093, major changes are: • RX_SPARE removed, RX_ONSET added • RX_SPEECH_PROBABLY_DEGRADED -> RX_SPEECH_DEGRADED • TX_SPEECH ->TX_SPEECH_GOOD
Consequences if not approved:	⌘	Inconsistency between 3G TS 26.073 and 3G TS 26.093 and 3G TS 26.073 and 3G TS 08.60/08.61

Clauses affected:	⌘	Files: frame.h, sid_sync.c, decoder.c, dec_amr.c, dtx_dec.c, strfunc.c		
Other specs affected:	⌘	[X] Other core specifications [ ] Test specifications [ ] O&M Specifications	⌘	~~3G TS 26.074 AMR-NB test vectors for DTX~~
Other comments:	⌘	Sections 1 and 2 of this CR have been copied from S4 Tdoc (00)0123		

How to create CRs using this form:
Comprehensive information and tips about how to create CRs can be found at:
http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://www.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2000-09 contains the specifications resulting from the September 2000 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

1. Background

3G TS 26.093 specifies the TX- and RX-Type Identifiers by the two following tables:

Table 1: TX_TYPE identifiers

TX_TYPE Legend	Information Bits	Mode Indication
SPEECH_GOOD	speech frame, size 95..244 bits depending on codec mode; no errors known.	current code mode
SPEECH_DEGRADED (only in downlink in TFO)	Speech frame, size 95..244 bits, depending on codec mode; there might be errors in class 2 bits.	current codec mode
SPEECH_BAD (only in downlink in TFO)	Speech frame, size 95..244 bits, depending on codec mode; there are errors in class 1 bits.	current codec mode
SID_FIRST	marks the end of a talkspurt, respectively the beginning of a speech pause; does not contain information bits.	the codec mode that would have been used if TX_TYPE had been SPEECH
SID_UPDATE	comfort noise, 35 bits; no errors known	the codec mode that would have been used if TX_TYPE had been SPEECH
SID_BAD (only in downlink in TFO)	comfort noise, 35 bits; errors detected, parameters unusable	the codec mode that would have been used if TX_TYPE had been SPEECH
ONSET (only in downlink in TFO)	announces the beginning of a speech burst; does not contain information bits	the codec mode of the following speech frame
NO_DATA	no useful information	no useful information

Table 2: RX_TYPE identifiers

RX_TYPE Legend	Description
SPEECH_GOOD	Speech frame with CRC OK, Channel Decoder soft values also OK
SPEECH_DEGRADED	Speech frame with CRC OK, but 1B bits and class2 bits may be corrupted
SPEECH_BAD	(likely) speech frame, bad CRC (or very bad Channel Decoder measures)
SID_FIRST	first SID marks the beginning of a comfort noise period
SID_UPDATE	SID update frame (with correct CRC)
SID_BAD	Corrupt SID update frame (bad CRC; applicable only for SID_UPDATE frames)
ONSET	ONSET frames precede the first speech frame of a speech burst
NO_DATA	Nothing useable (for the speech decoder) was received. This applies for the cases of no received frames (DTX) or received FACCH or RATSCCH or SID_FILLER signalling frames.

In order to have full correspondence between the tables and the C-code, the CR proposes a change in the C-code. The change has no impact on the functionality of the C-code.

2. How the code is changed in file *frame.h*

2.1 Before the change (lines 6...42)

```
enum RXFrameType { RX_SPEECH_GOOD = 0,
                   RX_SPEECH_PROBABLY_DEGRADED,
                   RX_SPARE,
                   RX_SPEECH_BAD,
                   RX_SID_FIRST,
                   RX_SID_UPDATE,
                   RX_SID_BAD,
                   RX_NO_DATA,
                   RX_N_FRAMETYPES     /* number of frame types */
};

enum TXFrameType { TX_SPEECH = 0,
                   TX_SID_FIRST,
                   TX_SID_UPDATE,
                   TX_NO_DATA,
                   TX_N_FRAMETYPES     /* number of frame types */
};
```

2.2 After the change

```
/* Note: The order of the TX and RX Type identifiers has been chosen in  */
/*       the way below to be compatible to an earlier version of the     */
/*       AMR-NB C reference program.                                     */

enum RXFrameType { RX_SPEECH_GOOD = 0,
                   RX_SPEECH_DEGRADED,
                   RX_ONSET,
                   RX_SPEECH_BAD,
                   RX_SID_FIRST,
                   RX_SID_UPDATE,
                   RX_SID_BAD,
                   RX_ONSET,
                   RX_NO_DATA,
                   RX_N_FRAMETYPES     /* number of frame types */
};

enum TXFrameType { TX_SPEECH_GOOD = 0,
                   TX_SPEECH_DEGRADED,
                   TX_SPEECH_BAD,
                   TX_SID_FIRST,
                   TX_SID_UPDATE,
                   TX_SID_BAD,
                   TX_ONSET,
                   TX_NO_DATA,
                   TX_SPEECH_DEGRADED,
                   TX_SPEECH_BAD,
                   TX_SID_BAD,
                   TX_ONSET,
                   TX_N_FRAMETYPES     /* number of frame types */
};
```

3. How the code is changed in file *sid_sync.c*

3.1 Before the change (lines 66...72)

```
int sid_sync_reset (sid_syncState *st)
{
    st->sid_update_counter = 3;
    st->sid_handover_debt = 0;
    st->prev_ft = TX_SPEECH;
    return 0;
}
```

3.2 After the change

```
int sid_sync_reset (sid_syncState *st)
{
    st->sid_update_counter = 3;
    st->sid_handover_debt = 0;
    st->prev_ft = TX_SPEECH_GOOD;
    return 0;
}
```

3.3 Before the change (lines 105...109)

```
        if (st->prev_ft == TX_SPEECH)
        {
           *tx_frame_type = TX_SID_FIRST;
           st->sid_update_counter = 3;
        }
```

3.4 After the change

```
        if (st->prev_ft == TX_SPEECH_GOOD)
        {
           *tx_frame_type = TX_SID_FIRST;
           st->sid_update_counter = 3;
        }
```

3.5 Before the change (lines 133...137)

```
    else
    {
       st->sid_update_counter = st->sid_update_rate ;
       *tx_frame_type = TX_SPEECH;
    }
```

3.6 After the change

```
    else
    {
       st->sid_update_counter = st->sid_update_rate ;
       *tx_frame_type = TX_SPEECH_GOOD;
    }
```

4. How the code is changed in file *decoder.c*

4.1 Before the change (lines 61...72)

```
static enum RXFrameType tx_to_rx (enum TXFrameType tx_type)
{    switch (tx_type) {
      case TX_SPEECH:      return RX_SPEECH_GOOD;
      case TX_SID_FIRST:   return RX_SID_FIRST;
      case TX_SID_UPDATE:  return RX_SID_UPDATE;
      case TX_NO_DATA:     return RX_NO_DATA;
      default:
        fprintf(stderr, "tx_to_rx: unknown TX frame type %d\n", tx_type);
        exit(1);
    }
}
```

4.2 After the change

```
static enum RXFrameType tx_to_rx (enum TXFrameType tx_type)
{
    switch (tx_type) {
      case TX_SPEECH_GOOD:      return RX_SPEECH_GOOD;
      case TX_SPEECH_DEGRADED:  return RX_SPEECH_DEGRADED;
      case TX_SPEECH_BAD:       return RX_SPEECH_BAD;
      case TX_SID_FIRST:        return RX_SID_FIRST;
      case TX_SID_UPDATE:       return RX_SID_UPDATE;
      case TX_SID_BAD:          return RX_SID_BAD;
      case TX_ONSET:            return RX_ONSET;
      case TX_NO_DATA:          return RX_NO_DATA;
      default:
        fprintf(stderr, "tx_to_rx: unknown TX frame type %d\n", tx_type);
        exit(1);
    }
}
```

5. How the code is changed in file *dec_amr.c*

5.1 Before the change (lines 332...348)

```
    /* SPEECH action state machine  */
    test (); test (); test ();
    if ((sub(frame_type, RX_SPEECH_BAD) == 0) ||
        (sub(frame_type, RX_NO_DATA) == 0) ||
        (sub(frame_type, RX_SPARE) == 0))
    {
       bfi = 1;                                          move16 ();
       test(); test();
       if ((sub(frame_type, RX_NO_DATA) == 0) ||
           (sub(frame_type, RX_SPARE) == 0))
       {
          build_CN_param(&st->nodataSeed,
                         prmno[mode],
                         bitno[mode],
                         parm);
       }
    }
```

5.2 After the change

```
    /* SPEECH action state machine  */
    test (); test (); test ();
    if ((sub(frame_type, RX_SPEECH_BAD) == 0) ||
        (sub(frame_type, RX_NO_DATA) == 0) ||
        (sub(frame_type, RX_ONSET) == 0))
    {
       bfi = 1;                                          move16 ();
       test(); test();
       if ((sub(frame_type, RX_NO_DATA) == 0) ||
           (sub(frame_type, RX_ONSET) == 0))
       {
          build_CN_param(&st->nodataSeed,
                         prmno[mode],
                         bitno[mode],
                         parm);
       }
    }
```

5.3 Before the change (lines 349...352)

```
    else if (sub(frame_type, RX_SPEECH_PROBABLY_DEGRADED) == 0)
    {
       pdfi = 1;                                         move16 ();
    }
```

5.4 After the change

```
    else if (sub(frame_type, RX_SPEECH_DEGRADED) == 0)
    {
       pdfi = 1;                                         move16 ();
    }
```

6. How the code is changed in file *dtx_dec.c*

6.1 Before the change (lines 720...723)

```
-----------------------------------------------------------------
RX_NO_DATA,           | SPEECH       | DTX/(DTX_MUTE)| DTX_MUTE
RX_SPARE              |(class2 garb.)|               |
-----------------------------------------------------------------
```

6.2 After the change

```
-----------------------------------------------------------------
RX_NO_DATA,           | SPEECH       | DTX/(DTX_MUTE)| DTX_MUTE
RX_ONSET              |(class2 garb.)|               |
-----------------------------------------------------------------
```

6.3 Before the change (lines 734...758)

```
/* DTX if SID frame or previously in DTX{_MUTE} and (NO_RX OR
   BAD_SPEECH) */
test(); test(); test();
test(); test(); test();
test(); test();
if ((sub(frame_type, RX_SID_FIRST) == 0)   ||
    (sub(frame_type, RX_SID_UPDATE) == 0)  ||
    (sub(frame_type, RX_SID_BAD) == 0)     ||
    (((sub(st->dtxGlobalState, DTX) == 0) ||
      (sub(st->dtxGlobalState, DTX_MUTE) == 0)) &&
     ((sub(frame_type, RX_NO_DATA) == 0) ||
      (sub(frame_type, RX_SPEECH_BAD) == 0) ||
      (sub(frame_type, RX_SPARE) == 0))))
{
   newState = DTX;                                          move16();

   /* stay in mute for these input types */
   test(); test(); test(); test(); test();
   if ((sub(st->dtxGlobalState, DTX_MUTE) == 0) &&
       ((sub(frame_type, RX_SID_BAD) == 0) ||
        (sub(frame_type, RX_SID_FIRST) ==  0) ||
        (sub(frame_type, RX_SPARE) ==  0) ||
        (sub(frame_type, RX_NO_DATA) == 0)))
   {
      newState = DTX_MUTE;                                  move16();
   }
```

6.4 After the change

```
/* DTX if SID frame or previously in DTX{_MUTE} and (NO_RX OR
   BAD_SPEECH) */
test(); test(); test();
test(); test(); test();
test(); test();
   if ((sub(frame_type, RX_SID_FIRST) == 0)   ||
    (sub(frame_type, RX_SID_UPDATE) == 0)  ||
    (sub(frame_type, RX_SID_BAD) == 0)     ||
    (((sub(st->dtxGlobalState, DTX) == 0) ||
      (sub(st->dtxGlobalState, DTX_MUTE) == 0)) &&
     ((sub(frame_type, RX_NO_DATA) == 0) ||
      (sub(frame_type, RX_SPEECH_BAD) == 0) ||
      (sub(frame_type, RX_ONSET) == 0))))
{
   newState = DTX;                                          move16();

   /* stay in mute for these input types */
   test(); test(); test(); test(); test();
   if ((sub(st->dtxGlobalState, DTX_MUTE) == 0) &&
```

```
          ((sub(frame_type, RX_SID_BAD) == 0) ||
           (sub(frame_type, RX_SID_FIRST) ==  0) ||
           (sub(frame_type, RX_ONSET) ==  0) ||
           (sub(frame_type, RX_NO_DATA) == 0)))
      {
         newState = DTX_MUTE;                                    move16();
      }
```

6.5 Before the change (lines 7945...801)

```
   test(); test(); test(); test();
   if ((sub(frame_type, RX_SID_FIRST) == 0)  ||
       (sub(frame_type, RX_SID_UPDATE) == 0) ||
       (sub(frame_type, RX_SID_BAD) == 0)    ||
       (sub(frame_type, RX_NO_DATA) == 0))
   {
      encState = DTX;                                            move16();
   }
   else
   {
      encState = SPEECH;                                         move16();
   }
```

6.6 After the change

```
   test(); test(); test(); test(); test();
   if ((sub(frame_type, RX_SID_FIRST) == 0)  ||
       (sub(frame_type, RX_SID_UPDATE) == 0) ||
       (sub(frame_type, RX_SID_BAD) == 0)    ||
       (sub(frame_type, RX_ONSET) == 0)      ||
       (sub(frame_type, RX_NO_DATA) == 0))
   {
      encState = DTX;                                             move16();
   }
   else
   {
      encState = SPEECH;                                          move16();
   }
```

7. How the code is changed in file *strfunc.c*

7.1 Before the change (lines 66...88)

```
/*
 * frame type name <-> frame type id conversion table
 */
static const conv_table rxframetable[] = {
  {"RX_SPEECH_GOOD", RX_SPEECH_GOOD},
  {"RX_SPEECH_PROBABLY_DEGRADED", RX_SPEECH_PROBABLY_DEGRADED},
  {"RX_SPEECH_BAD", RX_SPEECH_BAD},
  {"RX_SID_FIRST", RX_SID_FIRST},
  {"RX_SID_UPDATE", RX_SID_UPDATE},
  {"RX_SID_BAD", RX_SID_BAD},
  {"RX_NO_DATA" , RX_NO_DATA},
  {NULL, -1}
};
/*
 * frame type name <-> frame type id conversion table
 */
static const conv_table txframetable[] = {
  {"TX_SPEECH", TX_SPEECH},
  {"TX_SID_FIRST", TX_SID_FIRST},
  {"TX_SID_UPDATE", TX_SID_UPDATE},
  {"TX_NO_DATA" , TX_NO_DATA},
  {NULL, -1}
};
```

7.2 After the change

```
/*
 * frame type name <-> frame type id conversion table
 */
static const conv_table rxframetable[] = {
  {"RX_SPEECH_GOOD", RX_SPEECH_GOOD},
  {"RX_SPEECH_DEGRADED", RX_SPEECH_DEGRADED},
  {"RX_SPEECH_BAD", RX_SPEECH_BAD},
  {"RX_SID_FIRST", RX_SID_FIRST},
  {"RX_SID_UPDATE", RX_SID_UPDATE},
  {"RX_SID_BAD", RX_SID_BAD},
  {"RX_ONSET", RX_ONSET},
  {"RX_NO_DATA", RX_NO_DATA},
  {NULL, -1}
};
/*
 * frame type name <-> frame type id conversion table
 */
static const conv_table txframetable[] = {
  {"TX_SPEECH_GOOD", TX_SPEECH_GOOD},
  {"TX_SPEECH_DEGRADED", TX_SPEECH_DEGRADED},
  {"TX_SPEECH_BAD", TX_SPEECH_BAD},
  {"TX_SID_FIRST", TX_SID_FIRST},
  {"TX_SID_UPDATE", TX_SID_UPDATE},
  {"TX_SID_BAD", TX_SID_BAD},
  {"TX_ONSET", TX_ONSET},
  {"TX_NO_DATA", TX_NO_DATA},
  {NULL, -1}
};
```

3GPP TSG-SA WG4 Meeting #16 ***Tdoc S4 (01)0284***
Sophia Antipolis, France, 26 Feb – 2 Mar 2001

CR-Form-v3

CHANGE REQUEST

⌘ **26.073** **CR** **010** ⌘ rev **1** ⌘ Current version: **3.1.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM [ ] ME/UE [X] Radio Access Network [X] Core Network [ ]

Title:	⌘	Correction of the TX_TYPE and RX_TYPE identifiers
Source:	⌘	TSG-SA WG4
Work item code:	⌘	AMR
Date:	⌘	01-Mar-2001
Category:	⌘	**A**
Release:	⌘	REL-4

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:
2 (GSM Phase 2)
R96 (Release 1996)
R97 (Release 1997)
R98 (Release 1998)
R99 (Release 1999)
REL-4 (Release 4)
REL-5 (Release 5)

Reason for change:	⌘	TX_TYPE and RX_TYPE identifiers are defined in two different ways in 3G TS 26.073 and 3G TS 26.093
Summary of change:	⌘	TX_TYPE and RX_TYPE identifiers are modified to be consistent with 3G TS 26.093, major changes are: • RX_SPARE removed, RX_ONSET added • RX_SPEECH_PROBABLY_DEGRADED -> RX_SPEECH_DEGRADED • TX_SPEECH ->TX_SPEECH_GOOD
Consequences if not approved:	⌘	Inconsistency between 3G TS 26.073 and 3G TS 26.093 and 3G TS 26.073 and 3G TS 08.60/08.61

Clauses affected:	⌘	Files: frame.h, sid_sync.c, decoder.c, dec_amr.c, dtx_dec.c, strfunc.c		
Other specs affected:	⌘	[X] Other core specifications [ ] Test specifications [ ] O&M Specifications	⌘	~~3G TS 26.074 AMR-NB test vectors for DTX~~
Other comments:	⌘	Sections 1 and 2 of this CR have been copied from S4 Tdoc (00)0123		

How to create CRs using this form:
Comprehensive information and tips about how to create CRs can be found at:
http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://www.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2000-09 contains the specifications resulting from the September 2000 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

1. Background

3G TS 26.093 specifies the TX- and RX-Type Identifiers by the two following tables:

Table 1: TX_TYPE identifiers

TX_TYPE Legend	Information Bits	Mode Indication
SPEECH_GOOD	speech frame, size 95..244 bits depending on codec mode; no errors known.	current code mode
SPEECH_DEGRADED (only in downlink in TFO)	Speech frame, size 95..244 bits, depending on codec mode; there might be errors in class 2 bits.	current codec mode
SPEECH_BAD (only in downlink in TFO)	Speech frame, size 95..244 bits, depending on codec mode; there are errors in class 1 bits.	current codec mode
SID_FIRST	marks the end of a talkspurt, respectively the beginning of a speech pause; does not contain information bits.	the codec mode that would have been used if TX_TYPE had been SPEECH
SID_UPDATE	comfort noise, 35 bits; no errors known	the codec mode that would have been used if TX_TYPE had been SPEECH
SID_BAD (only in downlink in TFO)	comfort noise, 35 bits; errors detected, parameters unusable	the codec mode that would have been used if TX_TYPE had been SPEECH
ONSET (only in downlink in TFO)	announces the beginning of a speech burst; does not contain information bits	the codec mode of the following speech frame
NO_DATA	no useful information	no useful information

Table 2: RX_TYPE identifiers

RX_TYPE Legend	Description
SPEECH_GOOD	Speech frame with CRC OK, Channel Decoder soft values also OK
SPEECH_DEGRADED	Speech frame with CRC OK, but 1B bits and class2 bits may be corrupted
SPEECH_BAD	(likely) speech frame, bad CRC (or very bad Channel Decoder measures)
SID_FIRST	first SID marks the beginning of a comfort noise period
SID_UPDATE	SID update frame (with correct CRC)
SID_BAD	Corrupt SID update frame (bad CRC; applicable only for SID_UPDATE frames)
ONSET	ONSET frames precede the first speech frame of a speech burst
NO_DATA	Nothing useable (for the speech decoder) was received. This applies for the cases of no received frames (DTX) or received FACCH or RATSCCH or SID_FILLER signalling frames.

In order to have full correspondence between the tables and the C-code, the CR proposes a change in the C-code. The change has no impact on the functionality of the C-code.

2. How the code is changed in file *frame.h*

2.1 Before the change (lines 6...42)

```
enum RXFrameType { RX_SPEECH_GOOD = 0,
                   RX_SPEECH_PROBABLY_DEGRADED,
                   RX_SPARE,
                   RX_SPEECH_BAD,
                   RX_SID_FIRST,
                   RX_SID_UPDATE,
                   RX_SID_BAD,
                   RX_NO_DATA,
                   RX_N_FRAMETYPES     /* number of frame types */
};

enum TXFrameType { TX_SPEECH = 0,
                   TX_SID_FIRST,
                   TX_SID_UPDATE,
                   TX_NO_DATA,
                   TX_N_FRAMETYPES     /* number of frame types */
};
```

2.2 After the change

```
/* Note: The order of the TX and RX Type identifiers has been chosen in  */
/*       the way below to be compatible to an earlier version of the     */
/*       AMR-NB C reference program.                                     */

enum RXFrameType { RX_SPEECH_GOOD = 0,
                   RX_SPEECH_DEGRADED,
                   RX_ONSET,
                   RX_SPEECH_BAD,
                   RX_SID_FIRST,
                   RX_SID_UPDATE,
                   RX_SID_BAD,
                   RX_ONSET,
                   RX_NO_DATA,
                   RX_N_FRAMETYPES     /* number of frame types */
};

enum TXFrameType { TX_SPEECH_GOOD = 0,
                   TX_SPEECH_DEGRADED,
                   TX_SPEECH_BAD,
                   TX_SID_FIRST,
                   TX_SID_UPDATE,
                   TX_SID_BAD,
                   TX_ONSET,
                   TX_NO_DATA,
                   TX_SPEECH_DEGRADED,
                   TX_SPEECH_BAD,
                   TX_SID_BAD,
                   TX_ONSET,
                   TX_N_FRAMETYPES     /* number of frame types */
};
```

3. How the code is changed in file *sid_sync.c*

3.1 Before the change (lines 66...72)

```
int sid_sync_reset (sid_syncState *st)
{
    st->sid_update_counter = 3;
    st->sid_handover_debt = 0;
    st->prev_ft = TX_SPEECH;
    return 0;
}
```

3.2 After the change

```
int sid_sync_reset (sid_syncState *st)
{
    st->sid_update_counter = 3;
    st->sid_handover_debt = 0;
    st->prev_ft = TX_SPEECH_GOOD;
    return 0;
}
```

3.3 Before the change (lines 105...109)

```
        if (st->prev_ft == TX_SPEECH)
        {
           *tx_frame_type = TX_SID_FIRST;
           st->sid_update_counter = 3;
        }
```

3.4 After the change

```
        if (st->prev_ft == TX_SPEECH_GOOD)
        {
           *tx_frame_type = TX_SID_FIRST;
           st->sid_update_counter = 3;
        }
```

3.5 Before the change (lines 133...137)

```
    else
    {
       st->sid_update_counter = st->sid_update_rate ;
       *tx_frame_type = TX_SPEECH;
    }
```

3.6 After the change

```
    else
    {
       st->sid_update_counter = st->sid_update_rate ;
       *tx_frame_type = TX_SPEECH_GOOD;
    }
```

4. How the code is changed in file *decoder.c*

4.1 Before the change (lines 61...72)

```
static enum RXFrameType tx_to_rx (enum TXFrameType tx_type)
{    switch (tx_type) {
      case TX_SPEECH:      return RX_SPEECH_GOOD;
      case TX_SID_FIRST:   return RX_SID_FIRST;
      case TX_SID_UPDATE:  return RX_SID_UPDATE;
      case TX_NO_DATA:     return RX_NO_DATA;
      default:
        fprintf(stderr, "tx_to_rx: unknown TX frame type %d\n", tx_type);
        exit(1);
    }
}
```

4.2 After the change

```
static enum RXFrameType tx_to_rx (enum TXFrameType tx_type)
{
    switch (tx_type) {
      case TX_SPEECH_GOOD:      return RX_SPEECH_GOOD;
      case TX_SPEECH_DEGRADED:  return RX_SPEECH_DEGRADED;
      case TX_SPEECH_BAD:       return RX_SPEECH_BAD;
      case TX_SID_FIRST:        return RX_SID_FIRST;
      case TX_SID_UPDATE:       return RX_SID_UPDATE;
      case TX_SID_BAD:          return RX_SID_BAD;
      case TX_ONSET:            return RX_ONSET;
      case TX_NO_DATA:          return RX_NO_DATA;
      default:
        fprintf(stderr, "tx_to_rx: unknown TX frame type %d\n", tx_type);
        exit(1);
    }
}
```

5. How the code is changed in file *dec_amr.c*

5.1 Before the change (lines 332...348)

```
    /* SPEECH action state machine  */
    test (); test (); test ();
    if ((sub(frame_type, RX_SPEECH_BAD) == 0) ||
        (sub(frame_type, RX_NO_DATA) == 0) ||
        (sub(frame_type, RX_SPARE) == 0))
    {
       bfi = 1;                                          move16 ();
       test(); test();
       if ((sub(frame_type, RX_NO_DATA) == 0) ||
           (sub(frame_type, RX_SPARE) == 0))
       {
          build_CN_param(&st->nodataSeed,
                         prmno[mode],
                         bitno[mode],
                         parm);
       }
    }
```

5.2 After the change

```
    /* SPEECH action state machine  */
    test (); test (); test ();
    if ((sub(frame_type, RX_SPEECH_BAD) == 0) ||
        (sub(frame_type, RX_NO_DATA) == 0) ||
        (sub(frame_type, RX_ONSET) == 0))
    {
       bfi = 1;                                          move16 ();
       test(); test();
       if ((sub(frame_type, RX_NO_DATA) == 0) ||
           (sub(frame_type, RX_ONSET) == 0))
       {
          build_CN_param(&st->nodataSeed,
                         prmno[mode],
                         bitno[mode],
                         parm);
       }
    }
```

5.3 Before the change (lines 349...352)

```
    else if (sub(frame_type, RX_SPEECH_PROBABLY_DEGRADED) == 0)
    {
       pdfi = 1;                                         move16 ();
    }
```

5.4 After the change

```
    else if (sub(frame_type, RX_SPEECH_DEGRADED) == 0)
    {
       pdfi = 1;                                         move16 ();
    }
```

6. How the code is changed in file *dtx_dec.c*

6.1 Before the change (lines 720...723)

```
-----------------------------------------------------------------
RX_NO_DATA,           | SPEECH       | DTX/(DTX_MUTE)| DTX_MUTE
RX_SPARE              |(class2 garb.)|               |
-----------------------------------------------------------------
```

6.2 After the change

```
-----------------------------------------------------------------
RX_NO_DATA,           | SPEECH       | DTX/(DTX_MUTE)| DTX_MUTE
RX_ONSET              |(class2 garb.)|               |
-----------------------------------------------------------------
```

6.3 Before the change (lines 734...758)

```
/* DTX if SID frame or previously in DTX{_MUTE} and (NO_RX OR
   BAD_SPEECH) */
test(); test(); test();
test(); test(); test();
test(); test();
if ((sub(frame_type, RX_SID_FIRST) == 0)   ||
    (sub(frame_type, RX_SID_UPDATE) == 0)  ||
    (sub(frame_type, RX_SID_BAD) == 0)     ||
    (((sub(st->dtxGlobalState, DTX) == 0) ||
      (sub(st->dtxGlobalState, DTX_MUTE) == 0)) &&
     ((sub(frame_type, RX_NO_DATA) == 0) ||
      (sub(frame_type, RX_SPEECH_BAD) == 0) ||
      (sub(frame_type, RX_SPARE) == 0))))
{
   newState = DTX;                                          move16();

   /* stay in mute for these input types */
   test(); test(); test(); test(); test();
   if ((sub(st->dtxGlobalState, DTX_MUTE) == 0) &&
       ((sub(frame_type, RX_SID_BAD) == 0) ||
        (sub(frame_type, RX_SID_FIRST) ==  0) ||
        (sub(frame_type, RX_SPARE) ==  0) ||
        (sub(frame_type, RX_NO_DATA) == 0)))
   {
      newState = DTX_MUTE;                                  move16();
   }
```

6.4 After the change

```
/* DTX if SID frame or previously in DTX{_MUTE} and (NO_RX OR
   BAD_SPEECH) */
test(); test(); test();
test(); test(); test();
test(); test();
   if ((sub(frame_type, RX_SID_FIRST) == 0)   ||
    (sub(frame_type, RX_SID_UPDATE) == 0)  ||
    (sub(frame_type, RX_SID_BAD) == 0)     ||
    (((sub(st->dtxGlobalState, DTX) == 0) ||
      (sub(st->dtxGlobalState, DTX_MUTE) == 0)) &&
     ((sub(frame_type, RX_NO_DATA) == 0) ||
      (sub(frame_type, RX_SPEECH_BAD) == 0) ||
      (sub(frame_type, RX_ONSET) == 0))))
{
   newState = DTX;                                          move16();

   /* stay in mute for these input types */
   test(); test(); test(); test(); test();
   if ((sub(st->dtxGlobalState, DTX_MUTE) == 0) &&
```

```
          ((sub(frame_type, RX_SID_BAD) == 0) ||
           (sub(frame_type, RX_SID_FIRST) ==  0) ||
           (sub(frame_type, RX_ONSET) ==  0) ||
           (sub(frame_type, RX_NO_DATA) == 0)))
      {
         newState = DTX_MUTE;                                    move16();
      }
```

6.5 Before the change (lines 7945...801)

```
   test(); test(); test(); test();
   if ((sub(frame_type, RX_SID_FIRST) == 0)  ||
       (sub(frame_type, RX_SID_UPDATE) == 0) ||
       (sub(frame_type, RX_SID_BAD) == 0)    ||
       (sub(frame_type, RX_NO_DATA) == 0))
   {
      encState = DTX;                                            move16();
   }
   else
   {
      encState = SPEECH;                                         move16();
   }
```

6.6 After the change

```
   test(); test(); test(); test(); test();
   if ((sub(frame_type, RX_SID_FIRST) == 0)  ||
       (sub(frame_type, RX_SID_UPDATE) == 0) ||
       (sub(frame_type, RX_SID_BAD) == 0)    ||
       (sub(frame_type, RX_ONSET) == 0)      ||
       (sub(frame_type, RX_NO_DATA) == 0))
   {
      encState = DTX;                                             move16();
   }
   else
   {
      encState = SPEECH;                                          move16();
   }
```

7. How the code is changed in file *strfunc.c*

7.1 Before the change (lines 66...88)

```
/*
 * frame type name <-> frame type id conversion table
 */
static const conv_table rxframetable[] = {
  {"RX_SPEECH_GOOD", RX_SPEECH_GOOD},
  {"RX_SPEECH_PROBABLY_DEGRADED", RX_SPEECH_PROBABLY_DEGRADED},
  {"RX_SPEECH_BAD", RX_SPEECH_BAD},
  {"RX_SID_FIRST", RX_SID_FIRST},
  {"RX_SID_UPDATE", RX_SID_UPDATE},
  {"RX_SID_BAD", RX_SID_BAD},
  {"RX_NO_DATA" , RX_NO_DATA},
  {NULL, -1}
};
/*
 * frame type name <-> frame type id conversion table
 */
static const conv_table txframetable[] = {
  {"TX_SPEECH", TX_SPEECH},
  {"TX_SID_FIRST", TX_SID_FIRST},
  {"TX_SID_UPDATE", TX_SID_UPDATE},
  {"TX_NO_DATA" , TX_NO_DATA},
  {NULL, -1}
};
```

7.2 After the change

```
/*
 * frame type name <-> frame type id conversion table
 */
static const conv_table rxframetable[] = {
  {"RX_SPEECH_GOOD", RX_SPEECH_GOOD},
  {"RX_SPEECH_DEGRADED", RX_SPEECH_DEGRADED},
  {"RX_SPEECH_BAD", RX_SPEECH_BAD},
  {"RX_SID_FIRST", RX_SID_FIRST},
  {"RX_SID_UPDATE", RX_SID_UPDATE},
  {"RX_SID_BAD", RX_SID_BAD},
  {"RX_ONSET", RX_ONSET},
  {"RX_NO_DATA", RX_NO_DATA},
  {NULL, -1}
};
/*
 * frame type name <-> frame type id conversion table
 */
static const conv_table txframetable[] = {
  {"TX_SPEECH_GOOD", TX_SPEECH_GOOD},
  {"TX_SPEECH_DEGRADED", TX_SPEECH_DEGRADED},
  {"TX_SPEECH_BAD", TX_SPEECH_BAD},
  {"TX_SID_FIRST", TX_SID_FIRST},
  {"TX_SID_UPDATE", TX_SID_UPDATE},
  {"TX_SID_BAD", TX_SID_BAD},
  {"TX_ONSET", TX_ONSET},
  {"TX_NO_DATA", TX_NO_DATA},
  {NULL, -1}
};
```