ANNEX P ACP 127 and Character-Oriented Serial Stream (Optional)

The Character-Oriented Serial Stream (COSS) is primarily intended for operation of ACP 127 formal messaging over STANAG 5066 ARQ service. COSS is specified as a generic service, so that it cold be used by other applications with similar transport requirements. The ACP 127 protocol transmits a stream of ITA2 or IA5 characters over a serial line. The COSS service provides a service equivalent to serial line transmission. It provides a replacement for ACP 127 operation directly over an HF modem.

P.1. Changes in This Edition

The text of this annex is taken from Annex F Section F.3 of Edition 3.

The functional differences between this specification and Edition 3 are set out in Section P.8.

There are no significant functional changes.

P.2. CHARACTER-ORIENTED SERIAL STREAM (COSS) CLIENT

This annex defines a character-oriented serial-transport service for the HF subnetwork. This provides a reliable emulation of a serial line type service.

The character-oriented serial stream (COSS) service **may** be used in place of other HF serial transport services, for example a simple modem. To provide high reliability and end-to-end assurance of data delivery, the COSS client uses the STANAG 5066 ARQ mode to provide higher data reliability than simple transmission over a conventional modem might afford.

COSS defines two modes of operation:

1. Integrated Operation
2. Baseband Operation

An implementation of this annex **may** offer one or both modes.

P.3. Integrated Operation

COSS may be integrated directly with ACP 127 or another application as shown in Figure P-1.

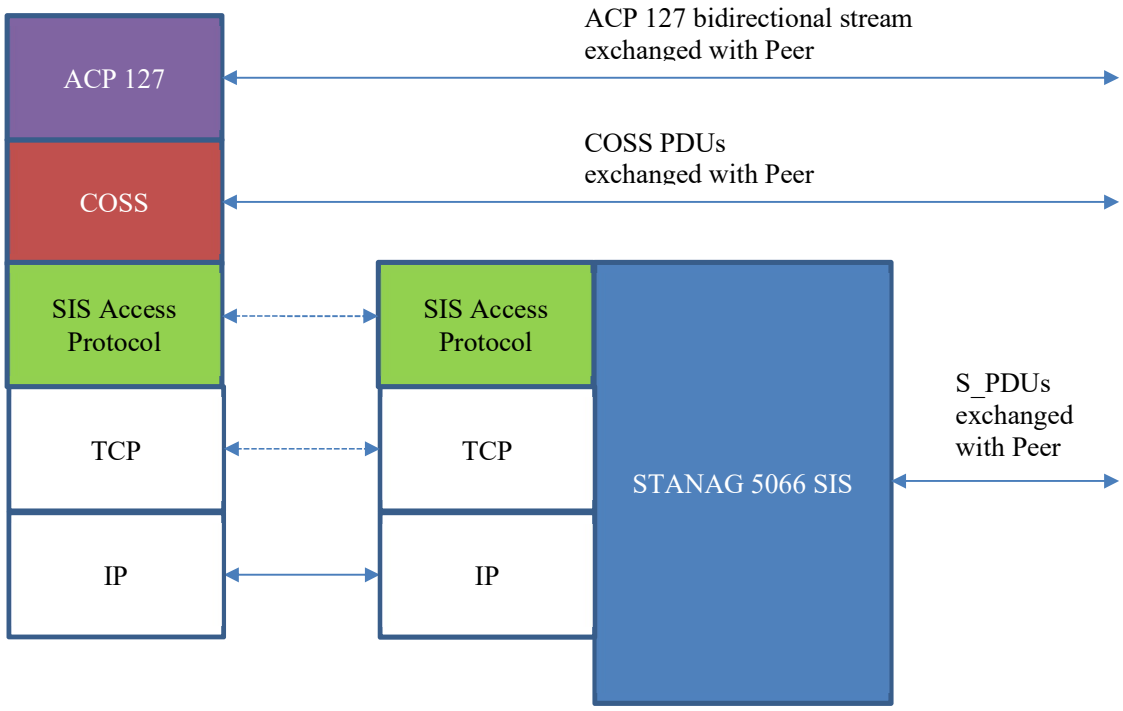

Figure P-1: ACP 127 Integrated Mapping onto COSS

Figure P-1 shows how an application can use a COSS client directly to communicate with a STANAG 5066 server using the SIS Access Protocol. The service provided by the COSS layer enables peer ACP 127 servers to exchange a stream of data in each direction.

P.4. Baseband Operation

COSS can also be configured with a serial interface, so that the COSS module interfaces to ACP 127 or other applications using a serial interface and communicates with a STANAG 5066 server using SIS Access Protocol as specified in Annex S. This mode is termed “baseband operation”, as it can directly replace an application using a baseband serial connection directly to an HF modem.

The interfaces for the Character-Oriented Serial Stream (COSS) Client using Baseband operation are as shown in the Figure below.

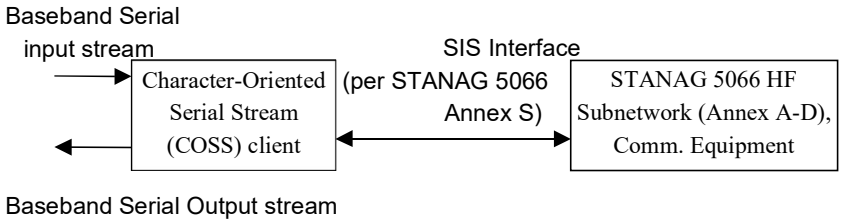

Figure P-2: COSS Client Interfaces

Requirements for the COSS client are placed on its baseband serial interface, its interface to the HF Subnetwork Interface Sublayer (SIS), and its internal processing. Implementations of the COSS client following Baseband operation **shall** use a baseband serial interface, and implementations **shall**(2) be in accordance with the requirements stated herein.

P.4.1. Base-band Serial Interface

A baseband mode COSS client **shall** provide a baseband serial interface meeting the following requirements.

a. *Physical/Electrical*: one of following physical interfaces **shall** be provided:

1. Signalling and connector conforming to EIA/RS-232, EIA//RS-530, configured as Data Communications Equipment (DCE);
2. Signalling and connector conforming to V.35 DCE.

b. *Serial Transmission Mode*: all of the following transmission modes **shall** be supported (as configuration options):

1. Asynchronous: 1 Start Character, selectable 5, 6, 7, or 8 data bits, and 1 or 2 stop bits.
2. Synchronous: provide/accept clock at 1x Data Rate (other rate-multipliers **may** be supported);
   HDLC line control protocol (other synchronous line protocols **may** be supported in addition to HDLC)

c. *Flow-Control*: all of the following flow-control disciplines **shall** be supported (as configuration options):

1. RTS/CTS, DTR/DTS hardware handshaking;
2. XON/XOFF software flow-control;
3. None.

P.5. Character Sets Supported by COSS

The character sets shown in the Table below **shall** be supported by a COSS client:

Table P-1: Character Sets Supported by COSS

Character Set	Comment
ITA-2 / Baudot	5-bit character sets: w/ asynchronous protocol = ITA-2; w/ synchronous protocol =
6-bit codes	6-bit character codes
ITA-5 / ASCII	~ASCII - 7-bit character set [NB: the 7.0-unit 64-ary ITA-2 code defined in STANAG 5030 is encoded in this form, in accordance with section F.3.4.4.1]
Octet data	Any data presented in arbitrary 8-bit formats

P.6. Subnetwork Service Requirements for COSS

COSS clients **shall** bind to the HF Subnetwork at SAP ID 1.

A COSS client **shall** submit its PDUs to the HF subnetwork using the S_UNIDATA_REQUEST Primitives defined in Annex A of this STANAG.

The address in the primitive **shall** be a STANAG 5066 address corresponding to the HF subnetwork address of the host at which the destination COSS client(s) is/are located.

The default service requirements defined when the client binds to the subnetwork **shall** be as follows to support point to point operation:

1. Transmission Mode = ARQ
2. Delivery Confirmation = NONE
3. Deliver in Order = IN-ORDER DELIVERY

COSS **may** also be mapped onto the Non-ARQ transmission mode to support point to multi-point delivery. As ACP 127 is designed to operate over a serial link, Unidata loss from the non-ARQ service may lead to gaps in messages received or to merging

of different messages.

P.7. Data Encapsulation Requirements

The characters from the character stream **shall** be encapsulated within S_PRIMITIVEs using any one of the modes described herein. Implementation of all modes is **mandatory** for a COSS client.

Selection of any given mode for operation **shall** be a configuration parameter in a COSS client, and dependent on the character-set for which the COSS client is configured.

Selection of any given mode **must** be coordinated at the sending and receiving node for use on a given link, through either standard operating procedure or out-of-band coordination channel.

P.7.1. Encapsulation of Arbitrary Octet Data

Octet data in any arbitrary format for the COSS client **shall** be byte-aligned with the octets in each U_PDU encapsulated in the S_UNIDATA_PRIMITIVE.

The least-significant bit (LSB) of each character received on the serial interface **shall** be aligned with the LSB of the octet.

P.7.2. Encapsulation of ITA-5

Characters in ITA-5 format (or other 7-bit character format such as ASCII) for the COSS client **shall** be aligned with the octets in each U_PDU encapsulated in the S_Primitive, one character per octet, as follows.

	MSB				Octet Data			LSB
Octet / Bit	7	6	5	4	3	2	1	0
$Octet_I$	0	msb	-	-	$ITA5_j$	-	-	lsb

The least-significant bit (LSB) of each 7-bit character received on the serial interface **shall** be aligned with the LSB of the octet.

The value of the MSB bit of each octet **shall** be set to zero for ITA5 Encapsulation.

P.7.3. Encapsulation of ITA-2

Two methods of encapsulation of ITA-2 characters are defined:

1. 'Loose-Pack ITA2 Encapsulation' (LPI2E), and
2. 'Dense- Pack ITA2 Encapsulation' (DPI2E).

A COSS client **shall** implement both methods of ITA2 encapsulation.

Selection of either method **must** be coordinated by sending and receiving node

for use on a given link, through either standard operating procedure or out-of-band coordination channel.

P.7.4. **'Loose-Pack Encapsulation of ITA-2 characters**

The Loose-Pack' ITA-2 Encapsulation (LPI2E) algorithm **may** be used for any character set represented as 5-bit symbols. It transports one 5-bit symbol in each octet within the U_PDU field of S_primitives, using the basic packing arrangement defined in the Figure below.

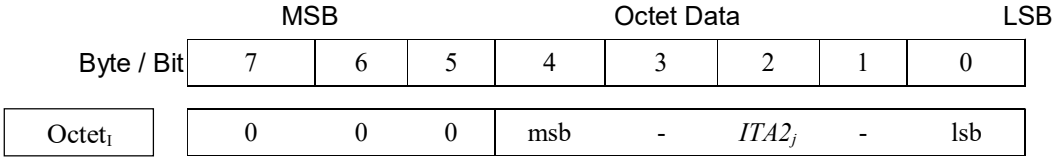

Figure P-3: Loose-Pack Encapsulation of ITA-2 Characters

The least-significant bit (LSB) of each 5-bit character received on the serial interface **shall** be aligned with the LSB of the octet.

The value of the three most-significant bits of each octet **shall** be set to zero for LPI2E.

P.7.5. **'Dense-Pack' Encapsulation of ITA-2 characters**

The 'Dense-Pack' ITA-2 Encapsulation (DPI2E) algorithm **may** be used for any character set represented as 5-bit symbols. It efficiently transports three 5-bit symbols in a pair of octets, called an Encapsulation Pair, within the U_PDU field of S_primitives, using the basic packing arrangement defined in the Figure below.

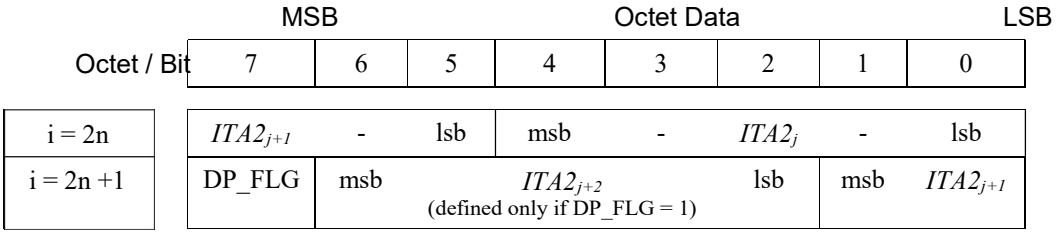

Figure P-4: Nominal Dense-Pack ITA-2 Encapsulation Pair

P.7.5.1.**Encapsulation Pairs**

The first octet of an Encapsulation Pair **shall** be an even-numbered octet in an S_Primitive's U_PDU field (i.e., i = {0,2,4,6, …} , with i = 0 the first octet in the U_PDU).

The second octet of an Encapsulation Pair **shall** be an odd-numbered octet in an S_Primitive's U_PDU field (i.e., i = {1,3,5,7, …} , with i = 0 the first octet in the U_PDU).

The MSB of the second octet of an Encapsulation Pair **shall**(1) be the DP_FLG

('dense-pack flag') that indicates whether the Encapsulation Pair contains three ITA-2 characters (DP_FLG = 1) or two ITA-2 characters (DP_FLG = 0). Encoding of these cases **shall**(2) be performed as follows:

- The first case is defined to be a "Three-into-Two Encapsulation Pair", which **shall** be encoded in accordance with this figure

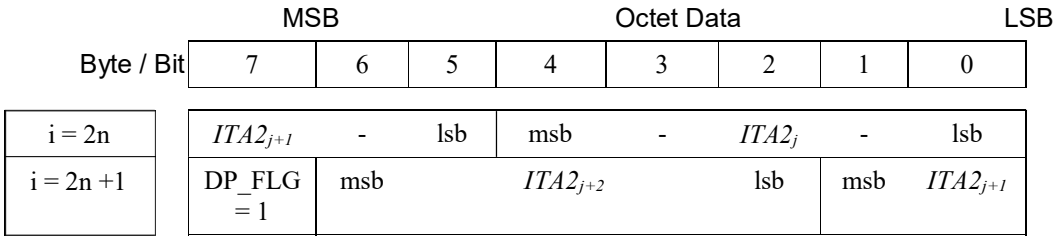

Figure P-5: Three-Into-Two Encapsulation Pair

- the second case is defined to be a "Two-into-Two Encapsulation Pair", which **shall** be encoded in accordance with this figure:

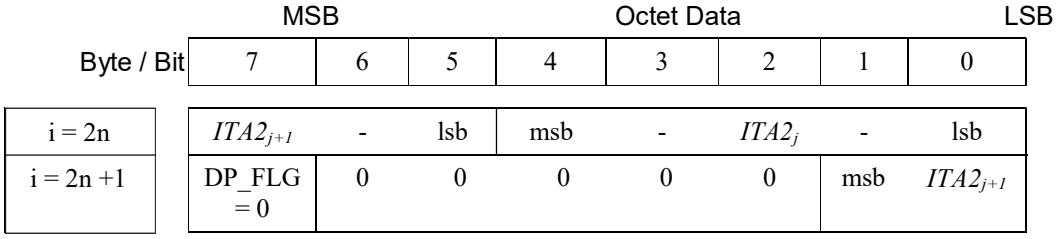

Figure P-6: Two-Into-Two Encapsulation Pair

P.7.5.2. **Maintaining Character-Count Integrity with DPI2E**

The 'Dense-Pack' ITA-2 Encapsulation (DPI2E) algorithm **shall not** add or delete ITA-2 characters from the character stream being transported. This property is denoted character-count integrity.

To maintain character-count integrity, the DPI2E algorithm **shall** depend on the number of characters that remain after all initial characters have been encapsulated as 3-into-2 Encapsulation Pairs.

For a buffer of size B characters encapsulated within a U_PDU field of length L, the DPI2E packing algorithm is defined as follows:

1. **Case R = (B modulo 3) = 0.** In this case, no characters remain after all initial characters are packed in Three-into-Two Encapsulation Pairs:
    - all ITA-2 characters **shall**(1) be densely packed as Three-into-Two Encapsulation Pairs. [i.e.,.: Each Encapsulation Pair will contain three ITA-2 characters, with DP_FLG =1].

- The U_PDU length **shall**(2) be set to the value L = (2 * B/3).

2. **Case R = (B modulo 3) = 1.** In this case, one character remains after all initial characters are densely packed in Three-into-Two Encapsulation Pairs:

   - the first (B-1) ITA-2 characters **shall**(1) be densely packed as 3-into-2 Encapsulation Pairs. [i.e.: each of the Encapsulation Pair will contain three ITA-2 characters, with DP_FLG =1].
   - The last remaining ITA-2 character **shall**(2) be loosely packed in the last octet of the U_PDU in accordance with the 'Loose-Pack Encapsulation of ITA-2 characters specification' of section F.3.4.3.1.
   - The U_PDU length **shall**(3) be set to the value L = (2 * ((B-1)/3) + 1).

3. **Case R = (B modulo 3) = 2.** In this case, two characters remain after all initial characters are densely packed in Three-into-Two Encapsulation Pairs:

   - the first (B-2) ITA-2 characters **shall**(1) be densely packed as 3-into-2 Encapsulation Pairs. [I.E.: Each Encapsulation Pair will contain three ITA-2 characters, with DP_FLG =1]
   - the two remaining ITA-2 characters **shall**(2) be packed as a 2-into-2 Encapsulation Pair. [I.E.: Each Encapsulation Pair will contain two ITA-2 characters, with DP_FLG =0; the bit positions corresponding to the third ITA-2 character in the Encapsulation Pair will be set to zero.].
   - the U_PDU length **shall**(3) be set to the value L = (2 * ((B-2)/3) + 2).

P.7.5.3. Character Unpacking Requirements for DPI2E

In accordance with standard operation procedure or out-of-band coordination circuit, the receiver **must** be configured to perform Dense-Pack ITA-2 Encapsulation.

The receiving client **shall** perform the inverse DPI2E algorithm to unpack the ITA2 characters the U_PDUs contained within an S_UNIDATA_INDICATION primitive:

For a U_PDU of length L,

- If L = 1, the receiving client **shall** unpack the single ITA2 character from the loosely packed octet and send it to the output serial stream;
- If L > 1 and L even, the receiving client **shall**(1) unpack the ITA2 characters in order from each Encapsulation Pair of octets, continuing in order for each successive Encapsulation Pair in the U_PDU. Unpacked ITA2 characters **shall**(2) be sent in the order in which they are unpacked to the

output serial stream. Receiving nodes **may** log a processing error if any Encapsulation Pair except the last has DP_FLG = 0.

- If L > 1 and L odd, the receiving client **shall**(1) unpack the ITA2 characters in order from each Encapsulation Pair of octets, continuing in order for each successive Encapsulation Pair in the U_PDU, and unpacking the last ITA2 character from the single octet (last, and not part of an Encapsulation Pair) in the U_PDU. Unpacked ITA2 characters **shall**(2) be sent in the order in which they are unpacked to the output serial stream. Receiving nodes **may** log a processing error locally if any Encapsulation Pair has DP_FLG = 0. Receiving nodes **may** log a processing error locally if the three most-significant bits of the octet are nonzero.

P.7.6. **Encapsulation of 6-bit Character Codes**

Characters in 6-bit formats for the COSS client **shall** be aligned with the octets in each U_PDU encapsulated in the S_Primitive, one character per octet.

	MSB				Octet Data			LSB
Octet / Bit	7	6	5	4	3	2	1	0
$Octet_I$	0	0	msb	-	$6\_bitChar_j$	-	-	lsb

The least-significant bit (LSB) of each 6-bit character received on the serial interface **shall** be aligned with the LSB of the octet.

The value of the MSB bit of each octet **shall** be set to zero for 6-bit character encapsulation.

The special case of the 64-ary ITA-2 character code defined for STANAG 5030 is specified below.

P.7.6.1.**Encapsulation of 64-ary ITA-2 (i.e., STANAG 5030) character codes**

The 64-ary ITA-2 code as defined in STANAG 5030 for Single and Multichannel VLF/LF Broadcasts actually uses a 7-bit character. It consists of 6-bits for information plus a 7^{th} bit (the Stop Bit) that does not change. This allows the possibility that some applications may choose a simple approach to shortening the STANAG 5030 64-ary ITA-2 code to a true six-bit code (i.e., by deleting the stop bit from the code, relying on other measures to provide character synchonization).

In 7-bit format (i.e, unshortened), the 7.0 unit STANAG 5030 64-ary ITA-2 code **shall** be encapsulated as specified in section F.3.4.2.

As a shortened 6-bit code (i.e, with the 7th/stop-bit removed, the STANAG 5030 64-ary ITA-2 code **shall** be encapsulated as specified in section F.3.4.4.

As the overhead from both approaches is equivalent, there is no reason with respect to STANAG 5066 operation to shorten the code. Consequently, use of the unshortened 7.0 unit STANAG 5030 64-ary ITA-2 code is preferred. Other external considerations may apply however that would favor use of the shortened code.

P.7.7. Character-Flush Requirements

It is assumed that the COSS client will be implemented with an input buffer for temporary storage of characters from the stream prior to their encapsulation in S_PRIMTIVES for transmission over the subnetwork. The events that trigger transfer of characters from this buffer to an S_PRIMITIVE (i.e, that triggers a 'character- flush' operation) need to be specified for the client. Of concern are the performance tradeoffs that exist for various character-flush disciplines. For instance, frequent character-flush operations will reduce end-to-end latency and increase the overhead, while for infrequent character-flush operations, triggered only when the size of the input buffer is the subnetwork's Maximum Transmission Unit Size (MTU), the opposite is true. As this is a largely performance issue and not an interoperability issue, a number of different behaviours could be defined.

The COSS client **shall** have a capability to configure the behaviour of its input-buffer character-flush discipline.

Characters **shall** be flushed from the COSS input buffer and encapsulated in an S_PRIMITIVE for transmission over the subnetwork when one or a combination of the following events occur:

1. The number of characters in the input buffer exceeds a configurable threshold value, COSS_BUF_FLUSH_THRESHOLD [NB: if the specified threshold value is greater than the subnetwork MTU size, then COSS_BUF_FLUSH_THRESHOLD shall equal the MTU value.];
2. A carriage-return/line-feed input character-pair is detected in the input character stream. [NB: this behaviour provides line-by-line transmission of a character stream organised as lines of text.]
3. A configurable timeout interval has occurred following the arrival in the input buffer of the last received character. [NB: this behaviour ensures that characters in the input buffer are eventually transmitted if one of the first two events has not occurred.]

Other behaviours for the character-flush disciplines **may** be defined as additional

and configurable implementation options, e.g., triggering a character-flush operation on detection of a user-specifiable character sequence defined as an End-Of-Message (EOM) sequence. [NB: such operation would be useful to support legacy systems such as the ACP-127 messaging systems, which use a defined character sequence as a message delimiter.]

P.8. Changes Since Edition 3

Edition 4 contains a number of minor corrections and clarifications, but is broadly unchanged.

Direct operation of COSS from an ACP 127 implementation without use of serial line is clarified.