16. Infrared (IR) PHY specification

16.1 Overview

The physical layer for the infrared system is specified in this clause. The IR PHY uses near-visible light in the 850 nm to 950 nm range for signaling. This is similar to the spectral usage of both common consumer devices such as infrared remote controls, as well as other data communications equipment, such as Infrared Data Association (IrDA) devices.

Unlike many other infrared devices, however, the IR PHY is not directed. That is, the receiver and transmitter do not have to be aimed at each other and do not need a clear line-of-sight. This permits the construction of a true LAN system, whereas with an aimed system, it would be difficult or impossible to install a LAN because of physical constraints.

A pair of conformant infrared devices would be able to communicate in a typical environment at a range up to about 10 m. This standard allows conformant devices to have more sensitive receivers, and this may increase range up to about 20 m.

The IR PHY relies on both reflected infrared energy as well as line-of-sight infrared energy for communications. Most designs anticipate that *all* of the energy at the receiver is reflected energy. This reliance on reflected infrared energy is called *diffuse infrared* transmission.

This standard specifies the transmitter and receiver in such a way that a conformant design will operate well in most environments where there is no line-of-sight path from the transmitter to the receiver. However, in an environment that has few or no reflecting surfaces, and where there is no line-of-sight, an IR PHY system may suffer reduced range.

The IR PHY will operate only in indoor environments. Infrared radiation does not pass through walls, and is significantly attenuated passing through most exterior windows. This characteristic can be used to "contain" an IR PHY in a single physical room, like a classroom or conference room. Different LANs using the IR PHY can operate in adjacent rooms separated only by a wall without interference, and without the possibility of eavesdropping.

At the time of this standard's preparation, the only known regulatory standards that apply to the use of infrared radiation are safety regulations, such as IEC 60825-1: 1998 [B2] and ANSI Z136.1-1993 [B1]. While a conformant IR PHY device can be designed to also comply with these safety standards, conformance with this standard does not ensure conformance with other standards.

Worldwide, there are currently no frequency allocation or bandwidth allocation regulatory restrictions on infrared emissions.

Emitter (typically LED) and detector (typically PIN diode) devices for infrared communications are relatively inexpensive at the infrared wavelengths specified in the IR PHY, and at the electrical operating frequencies required by this PHY.

While many other devices in common use also use infrared emissions in the same optical band, these devices usually transmit infrared intermittently and do not interfere with the proper operation of a compliant IR PHY. If such a device does interfere, by transmitting continuously and with a very strong signal, it can be physically isolated (placing it in a different room) from the IEEE 802.11 LAN.

16.1.1 Scope

The PHY services provided to the IEEE 802.11 wireless LAN MAC by the IR system are described in this clause. The IR PHY layer consists of two protocol functions as follows:

a) A physical layer convergence function, which adapts the capabilities of the physical medium dependent (PMD) system to the PHY service. This function is supported by the physical layer convergence procedure (PLCP), which defines a method of mapping the IEEE 802.11 MAC sublayer protocol data units (MPDU) into a framing format suitable for sending and receiving user data and management information between two or more STAs using the associated PMD system.
b) A PMD system, whose function defines the characteristics of, and method of transmitting and receiving data through, the wireless medium (WM) between two or more STAs.

16.1.2 IR PHY functions

The IR PHY contains three functional entities: the PMD function, the physical layer convergence function, and the layer management function. Each of these functions is described in detail below.

The IR PHY service is provided to the MAC entity at the STA through a service access point (SAP) as described in Clause 12. For a visual guide to the relationship of the IR PHY to the remainder of the system, refer to Figure 11.

16.1.2.1 PLCP sublayer

To allow the IEEE 802.11 MAC to operate with minimum dependence on the PMD sublayer, a physical layer convergence sublayer is defined. This function simplifies the PHY service interface to the IEEE 802.11 MAC services. The PHY-specific preamble is normally associated with this convergence layer.

16.1.2.2 PMD sublayer

The PMD sublayer provides a clear channel assessment (CCA) mechanism, transmission mechanism, and reception mechanism that are used by the MAC via the PLCP to send or receive data between two or more STAs.

16.1.2.3 PHY management entity (PLME)

The PLME performs management of the local PHY functions in conjunction with the MAC management entity. Subclause 16.4 lists the MIB variables that may be accessed by the PHY sublayer entities and intralayer of higher-layer management entities (LMEs). These variables are accessed via the PLME-GET, PLME-SET, and PLME-RESET primitives defined in Clause 10.

16.1.3 Service specification method and notation

The models represented by figures and state diagrams are intended as illustrations of functions provided. It is important to distinguish between a model and a real implementation. The models are optimized for simplicity and clarity of presentation; the actual method of implementation is left to the discretion of the IEEE 802.11 IR PHY compliant developer. Conformance to this standard is not dependent on following the model, and an implementation that follows the model closely may not be conformant.

Abstract services are specified here by describing the service primitives and parameters that characterize each service. This definition is independent of any particular implementation. In particular, the PHY-SAP operations are defined and described as instantaneous; however, this may be difficult to achieve in an implementation.

16.2 IR PLCP sublayer

While the PLCP sublayer and the PMD sublayer are described separately, the separation and distinction between these sublayers is artificial, and is not meant to imply that the implementation must separate these functions. This distinction is made primarily to provide a point of reference from which to describe certain functional components and aspects of the PMD. The functions of the PLCP can be subsumed by a PMD sublayer; in this case, the PMD will incorporate the PHY-SAP as its interface, and will not offer a PMD-SAP.

16.2.1 Overview

A convergence procedure is provided by which MPDUs are converted to and from PLCPDUs. During transmission, the MPDU (PLCSDU) is prepended with a PLCP Preamble and PLCP Header to create the PLCPDU. At the receiver, the PLCP Preamble is processed and the internal data fields are processed to aid in demodulation and delivery of the MPDU (PSDU).

16.2.2 PLCP frame format

Figure 101 shows the format for the PLCPDU including the PLCP Preamble, the PLCP Header, and the PSDU. The PLCP Preamble contains the following fields: Synchronization (SYNC) and Start Frame Delimiter (SFD). The PLCP Header contains the following fields: Data Rate (DR), DC Level Adjustment (DCLA), Length (LENGTH), and Cyclic Redundancy Check (CRC). Each of these fields is described in detail in 16.2.4.

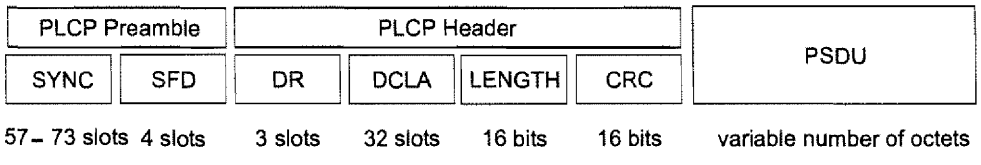


Figure 101—PLCPDU frame format

16.2.3 PLCP modulation and rate change

The PLCP Preamble shall be transmitted using the basic pulse defined in 16.3.3.2. The PLCSDU, LENGTH, and CRC fields shall be transmitted using pulse position modulation (PPM). PPM maps bits in the octet into symbols: 16-PPM maps four bits into a 16-position symbol, and 4-PPM maps two bits into a 4-position symbol. The basic L-PPM time unit is the slot. A slot corresponds to one of the L positions of a symbol and has a 250 ns duration. The PLCSDU, LENGTH, and CRC fields are transmitted at one of two bit rates: 1 Mbit/s or 2 Mbit/s. The Data Rate field indicates the data rate that will be used to transmit the PLCSDU, LENGTH, and CRC fields. The 1 Mbit/s data rate uses 16-PPM (basic access rate), and the 2 Mbit/s data rate uses 4-PPM (enhanced access rate). The transmitter and receiver will initiate the modulation or demodulation indicated by the DR field starting with the first 4 bits (in 16-PPM) or 2 bits (in 4-PPM) of the LENGTH field. The PSDU transmission rate is set by the DATARATE parameter in the PHY-TXSTART.request primitive. Any conformant IR PHY shall be capable of receiving at 1 Mbit/s and 2 Mbit/s. Transmission at 2 Mbit/s is optional.

A PHY-TXSTART.request that specifies a data rate that is not supported by a PHY instance will cause the PHY to indicate an error to its MAC instance. A PHY is not permitted under any circumstance to transmit at a different rate than the requested rate.

16.2.4 PLCP field definitions

16.2.4.1 PLCP Synchronization (SYNC) field

The SYNC field consists of a sequence of alternated presence and absence of a pulse in consecutive slots. The SYNC field has a minimum length of 57 L-PPM slots and a maximum length of 73 L-PPM slots and shall terminate with the absence of a pulse in the last slot. This field is provided so that the receiver can perform clock recovery (slot synchronization), automatic gain control (optional), signal-to-noise ratio estimation (optional), and diversity selection (optional).

The SYNC field is not modulated using L-PPM, but instead consists of transitions in L-PPM slots that would otherwise constitute an illegal symbol. See 16.3.2.1 for legal symbols.

16.2.4.2 PLCP Start Frame Delimiter (SFD) field

The SFD field length is four L-PPM slots and consists of the binary sequence 1001, where 1 indicates a pulse in the L-PPM slot and 0 indicates no pulse in the L-PPM slot. The leftmost bit shall be transmitted first. The SFD field is provided to indicate the start of the PLCP Preamble and to perform bit and symbol synchronization.

The SFD field is not modulated using L-PPM, but instead consists of transitions in L-PPM slots that would otherwise constitute an illegal symbol.

16.2.4.3 PLCP Data Rate (DR) field

The DR field indicates to the PHY the data rate that shall be used for the transmission or reception of the PLCSDU, LENGTH, and CRC fields. The transmitted value shall be provided by the PHY-TXSTART.request primitive as described in Clause 12. The DR field has a length of three L-PPM slots. The leftmost bit, as shown below, shall be transmitted first. The IR PHY currently supports two data rates defined by the slot pattern shown for the three L-PPM slots following the SFD, where 1 indicates a pulse in the L-PPM slot and 0 indicates no pulse in the L-PPM slot:

1 Mbit/s: 000
2 Mbit/s: 001

The DR field is not modulated using L-PPM, but instead consists of transitions in L-PPM slots that would otherwise constitute an illegal symbol.

16.2.4.4 PLCP DC Level Adjustment (DCLA) field

The DCLA field is required to allow the receiver to stabilize the dc level after the SYNC, SFD, and DR fields. The leftmost bit, as shown below, shall be transmitted first. The length of the DCLA field is 32 L-PPM slots and consists of the contents shown, where 1 indicates a pulse in the L-PPM slot and 0 indicates no pulse in the L-PPM slot:

1 Mbit/s: 00000000100000000000000010000000
2 Mbit/s: 00100010001000100010001000100010

The DCLA field is not modulated using L-PPM, but instead consists of transitions in L-PPM slots that would otherwise constitute an illegal symbol.

16.2.4.5 PLCP LENGTH field

The LENGTH field is an unsigned 16-bit integer that indicates the number of octets to be transmitted in the PSDU. The transmitted value shall be provided by the PHYTXSTART.request primitive as described in Clause 12. The lsb shall be transmitted first. This field is modulated and sent in L-PPM format. This field is protected by the CRC described in 16.2.4.6.

16.2.4.6 PLCP CRC field

The LENGTH field shall be protected by a 16-bit CRC-CCITT. The CRC-CCITT is the one's complement of the remainder generated by the modulo 2 division of the LENGTH field by the polynomial:

$$x^{16}+x^{12}+x^{5}+1$$

The protected bits will be processed in transmit order. The msb of the 16-bit CRC-CCITT shall be transmitted first. This field shall be modulated and sent in L-PPM format. All CRC-CCITT calculations shall be made prior to L-PPM encoding on transmission and after L-PPM decoding on reception.

16.2.4.7 PSDU field

This field is composed of a variable number of octets. The minimum is 0 (zero) and the maximum is 2500. The lsb of each octet shall be transmitted first. All the octets of this field shall be modulated and sent in L-PPM format.

16.2.5 PLCP procedures

16.2.5.1 PLCP transmit procedure

All commands issued by the MAC require that a confirmation primitive be issued by the PHY. The confirmation primitives provide flow control between the MAC and the PHY.

The transmit procedure is as follows:

a) Based on the status of CCA, the MAC shall determine whether the channel is clear.
b) If the channel is clear, transmission of the PSDU shall be initiated by a PHY-TXSTART.request with parameters LENGTH and DATARATE.
c) The PHY entity shall immediately initiate transmission of the PLCP Preamble and PLCP Header based on the LENGTH and DATARATE parameters passed in the PHY-TXSTART.request. Once the PLCP Preamble and PLCP Header transmission is completed, the PHY entity shall issue a PHY-TXSTART.confirm.
d) Each octet of the PSDU is passed from the MAC to the PHY by a single PHY-DATA.request primitive. Each PHY-DATA.request shall be confirmed by the PHY with a PHY-DATA.confirm before the next request can be made.
e) At the PHY layer each PSDU octet shall be divided into symbols of 2 bits or 4 bits each. The symbols shall be modulated using L-PPM and transmitted into the medium.
f) Transmission is terminated by the MAC through the primitive PHY-TXEND.request. The PHY shall confirm the resulting end of transmission with a PHY-TXEND.confirm.

16.2.5.2 PLCP receive procedure

The receive procedure is as follows:

a) CCA is provided to the MAC via the PHY-CCA.indicate primitive. When the PHY senses activity on the medium, it shall indicate that the medium is busy with a PHY-CCA.indicate with a value of BUSY. This will normally occur during the SYNC field of the PLCP Preamble.

b) The PHY entity shall begin searching for the SFD field. Once the SFD field is detected, the PHY entity shall attempt to receive the PLCP Header. After receiving the DR and DCLA fields, the PHY shall initiate processing of the received CRC and LENGTH fields. The data rate indicated in the DR field applies to all symbols in the latter part of the received PHYSDU, commencing with the first symbol of the LENGTH field. The CRC-CCITT shall be checked for correctness immediately after its reception.

c) If the CRC-CCITT check fails, or the value received in the DR field is not one supported by the PHY, then a PHY-RXSTART.indicate shall not be issued to the MAC. When the medium is again free, the PHY shall issue a PHY-CCA.indicate with a value of IDLE.

d) If the PLCP Preamble and PLCP Header reception is successful, the PHY shall send a PHY-RXSTART.indicate to the MAC; this includes the parameters DATARATE and LENGTH.

   In the absence of errors, the receiving PHY shall report the same length to its local MAC, in the RXVECTOR parameter of the PHY-RXSTART.indicate primitive, that the peer MAC presented to its local PHY entity in the TXVECTOR parameter of its respective PHY-TXSTART.request.

e) The received PLCSDU L-PPM symbols shall be assembled into octets and presented to the MAC using a series of PHY-DATA.indicate primitives, one per octet.

f) Reception shall be terminated after the reception of the final symbol of the last PLCSDU octet indicated by the PLCP Header's LENGTH field. After the PHY-DATA.indicate for that octet is issued, the PHY shall issue a PHY-RXEND.indicate primitive to its MAC.

g) After issuing the PHY-RXEND.indicate primitive, and when the medium is no longer busy, the PHY shall issue a PHY-CCA.indicate primitive with a value of IDLE.

16.2.5.3 CCA procedure

CCA is provided to the MAC via the PHY-CCA.indicate primitive.

The CCA procedure is as follows:

a) When the PHY senses activity on the medium, a PHY-CCA.indicate primitive with a value of BUSY shall be issued. This will normally occur during reception of the SYNC field of the PLCP Preamble.

b) When the PHY senses that the medium is free, a PHY-CCA.indicate primitive with a value of IDLE shall be issued.

c) At any time, the MAC may issue a PHY-CCARESET.request primitive, which will reset the PHY's internal CCA detection mechanism to the medium not-busy (IDLE) state. This primitive will be acknowledged with a PHY-CCARESET.confirm primitive.

16.2.5.4 PMD_SAP peer-to-peer service primitive parameters

Several service primitives include a parameter vector. This vector shall be a list of parameters that may vary depending on PHY type. Table 68 indicates the parameters required by the MAC or IR PHY in each of the parameter vectors used for peer-to-peer interactions.

Table 68—IR PMD_SAP peer-to-peer service primitives

Parameter	Associated primitive	Value
LENGTH	RXVECTOR, TXVECTOR	4 to $2^{16} - 1$
DATARATE	RXVECTOR, TXVECTOR	PHY dependent

16.3 IR PMD sublayer

The IR PMD sublayer does not define PMD SAPs. The mechanism for communications between the PLCP and PMD sublayers, as well as the distinction between these two sublayers, if any, is left to implementors. In particular, it is possible to design and implement, in a conformant way, a single sublayer that subsumes the functions of both the PLCP and PMD, presenting only the PHY-SAP.

16.3.1 Overview

The PMD functional, electrical, and optical characteristics required for interoperability of implementations conforming to this specification are described in this subclause. The relationship of this specification to the entire IR physical layer is shown in Figure 11.

16.3.2 PMD operating specifications, general

General specifications for the IR PMD sublayer are provided in this subclause. These specifications apply to both the receive and transmit functions and general operation of a compliant IR PHY.

16.3.2.1 Modulation and channel data rates

Two modulation formats and data rates are specified for the IR PHY: a *basic access rate* and an *enhanced access rate*. The basic access rate is based on 1 Mbit/s 16-PPM modulation. The 16-PPM encoding is specified in Table 69. Each group of 4 data bits is mapped to one of the 16-PPM symbols. The enhanced access rate is based on 2 Mbit/s 4-PPM. The 4-PPM encoding is specified in Table 70. Each group of 2 data bits is mapped to one of the 4-PPM symbols. Transmission order of the symbol slots is from left to right, as shown in the table, where a 1 indicates in-band energy in the slot, and a 0 indicates the absence of in-band energy in the slot.

The data in Table 69 and Table 70 have been arranged (gray coded) so that a single out-of-position-by-one error in the medium, caused, for example, by intersymbol interference, results in only a single bit error in the received data, rather than in a multiple bit error.

Table 69—Sixteen-PPM basic rate mapping

Data	16-PPM symbol
0000	0000000000000001
0001	0000000000000010
0011	0000000000000100
0010	0000000000001000
0110	0000000000010000
0111	0000000000100000
0101	0000000001000000
0100	0000000010000000
1100	0000000100000000
1101	0000001000000000
1111	0000010000000000
1110	0000100000000000
1010	0001000000000000
1011	0010000000000000
1001	0100000000000000
1000	1000000000000000

Table 70—Four-PPM enhanced rate mapping

Data	4-PPM symbol
00	0001
01	0010
11	0100
10	1000

16.3.2.2 Octet partition and PPM symbol generation procedure

Since PPM is a block modulation method, with the block size less than a full octet, octets have to be partitioned prior to modulation (mapping into PPM symbols).

Octet partition depends on the PPM order being used.

Assume an octet is formed by eight bits numbered 7 6 5 4 3 2 1 0, where bit 0 is the lsb. Partition the octet as follows:

For 16-PPM, create two PPM symbols:

— The symbol using bits 3 2 1 0 shall be transmitted onto the medium first.
— The symbol using bits 7 6 5 4 shall be transmitted onto the medium last.

For 4-PPM, create four PPM symbols:

— The symbol using bits 1 0 shall be transmitted onto the medium first.
— The symbol using bits 3 2 shall be transmitted onto the medium second.
— The symbol using bits 5 4 shall be transmitted onto the medium third.
— The symbol using bits 7 6 shall be transmitted onto the medium last.

16.3.2.3 Operating environment

The IR PHY will operate only in indoor environments. IR PHY interfaces cannot be exposed to direct sunlight. The IR PHY relies on reflected infrared energy and does not require a line-of-sight between emitter and receiver in order to work properly. The range and bit error rate of the system may vary with the geometry of the environment and with natural and artificial illumination conditions.

16.3.2.4 Operating temperature range

The temperature range for full operation compliance with the IR PHY is specified as 0 °C to 40 °C.

16.3.3 PMD transmit specifications

The following subclauses describe the transmit functions and parameters associated with the PMD sublayer.

16.3.3.1 Transmitted peak optical power

The peak optical power of an emitted pulse shall be as specified in Table 71.

Table 71—Peak optical power as a function of emitter radiation pattern mask

Emitter radiation pattern mask	Peak optical power
Mask 1	2 W ± 20%
Mask 2	0.55 W ± 20%

16.3.3.2 Basic pulse shape and parameters

The basic pulse width, measured between the 50% amplitude points, shall be 250 ± 10 ns. The pulse rise time, measured between the 10% and 90% amplitude points, shall be no more than 40 ns. The pulse fall time, measured between the 10% and 90% amplitude points, shall be no more than 40 ns. The edge jitter, defined as the absolute deviation of the edge from its correct position, shall be no more than 10 ns. The basic pulse shape is shown in Figure 102.

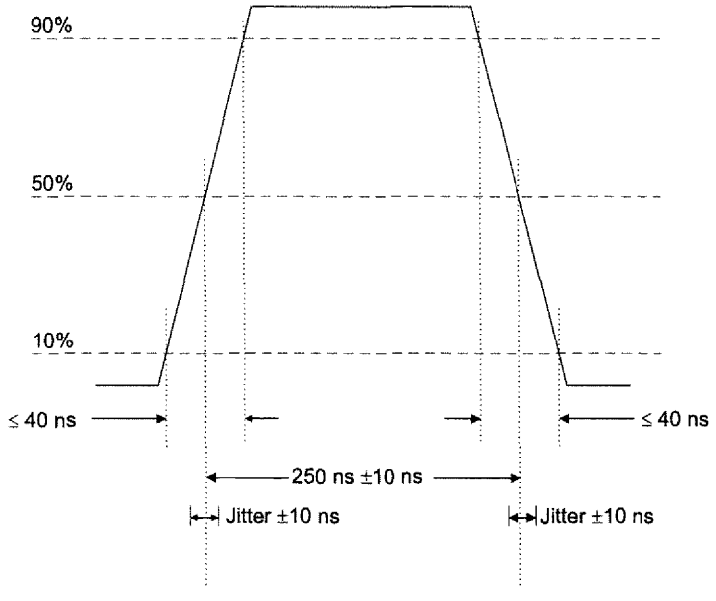


Figure 102—Basic pulse shape

16.3.3.3 Emitter radiation pattern mask

The standard contains two emitter radiation pattern masks. Mask 1 is defined in Table 72 and illustrated in Figure 103. Mask 2 is defined in Table 73 and illustrated in Figure 105.

Table 72—Definition of the emitter radiation pattern mask 1

Declination angle	Normalized irradiance
$\alpha \leq 60^\circ$	$> 3.5 \times 10^{-6}$
$\alpha \leq 29^\circ$	$\leq 2.2 \times 10^{-5}$
$29^\circ < \alpha \leq 43^\circ$	$\leq -1.06 \times 10^{-4} + (0.44 \times 10^{-5})\,\alpha$
$43^\circ < \alpha \leq 57^\circ$	$\leq 1.15 \times 10^{-4} - (7.1 \times 10^{-7})\,\alpha$
$57^\circ < \alpha \leq 74^\circ$	$\leq 2.98 \times 10^{-4} - (3.9 \times 10^{-6})\,\alpha$
$74^\circ < \alpha \leq 90^\circ$	$\leq 4.05 \times 10^{-5} - (4.5 \times 10^{-7})\,\alpha$

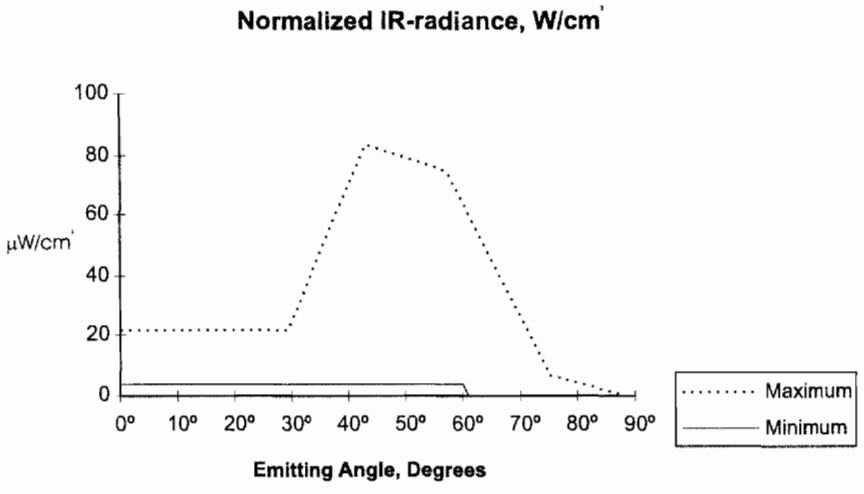


Figure 103—Emitter radiation pattern mask 1

Following is a description of how to interpret the Mask 1 table and figure. Position the conformant Mask 1 device in its recommended attitude. Define the conformant Mask 1 device axis as the axis passing through the emitter center and having the direction perpendicular to the floor. The mask represents the irradiance normalized to the total peak emitted power, as a function of the angle between the conformant Mask 1 device axis and the axis from the emitter center to the test receiver center (declination angle). The distance between emitter and test receiver is 1 m. The test receiver normal is always aimed at the emitter center. The azimuth angle is a rotation angle on the conformant device axis.

A device is conformant if for any azimuth angle its radiation pattern as a function of declination angle falls within the pattern mask.

Figure 104 is a description of how to interpret the Mask 2 table with reference to Figure 105.

Table 73—Definition of emitter radiation pattern mask 2

Declination angle	Pitch angle	Normalized irradiance
$\alpha \leq 60$	$\alpha = 0$	$0.05 \pm 15\%$
$\alpha \leq 90$	$\alpha = 0$	$0.025 \pm 15\%$
$\alpha \geq 100$	$\alpha = 0$	≤ 0.015
$0 \leq \alpha \leq 60$	$0 \leq \alpha \leq 10$	$0.035 \leq I \leq 0.055$
$0 \leq \alpha \leq 60$	$10 \leq \alpha \leq 20$	$0.0225 \leq I \leq 0.05$
$0 \leq \alpha \leq 60$	$\alpha \geq 30$	≤ 0.015

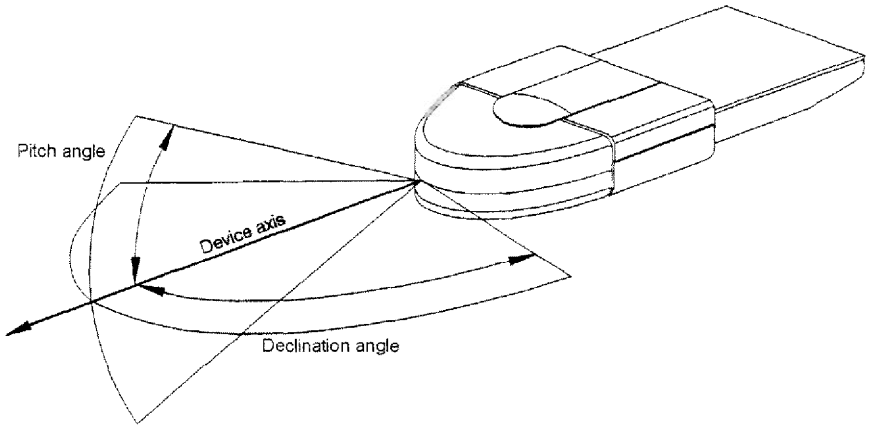


Figure 104—Mask 2 device orientation drawing

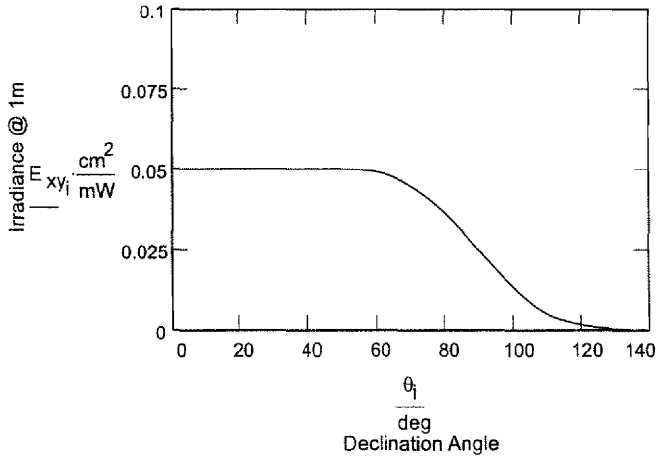


Figure 105—Emitter radiation pattern mask 2

Position the conformant Mask 2 device in its recommended attitude. Define the conformant Mask 2 device axis as passing through the emitter center and having the direction relative to the device as defined by the manufacturer. The declination angle plane is as defined by the manufacturer. The mask represents the irradiance normalized to the peak emitted power on the conformant Mask 2 device axis, as a function of the angle between the conformant device axis and the axis from the emitter center to the test receiver center (declination angle) in the declination plane. The distance between emitter and test receiver is 1 m. The test receiver normal is always aimed at the emitter center. The pitch angle is an angle relative to the conformant device axis which is perpendicular to the declination plane.

The device is conformant if, for a pitch angle of 0 degrees, at any declination angle from 0 to 100 degrees, and if, for any declination angle from 0 to 60 degrees, at any pitch angle from 0 to 20 degrees, its radiation pattern as a function of angle falls within the pattern mask.

Other radiation patterns are for future study.

16.3.3.4 Optical emitter peak wavelength

The optical emitter peak wavelength shall be between 850 nm and 950 nm.

16.3.3.5 Transmit spectrum mask

Define the transmit spectrum of a transmitter as the Fourier Transform, or equivalent, of a voltage (or current) signal whose amplitude, as a function of time, is proportional to the transmitted optical power.

The transmit spectrum of a conformant transmitter shall be 20 dB below its maximum for all frequencies above 15 MHz. The transmit spectrum mask is shown in Figure 106.

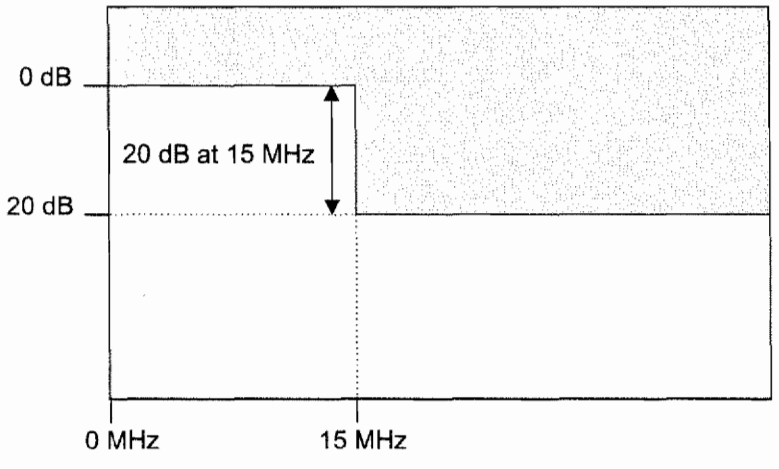


Figure 106—Transmit spectrum mask

16.3.4 PMD receiver specifications

The following subclauses describe the receive functions and parameters associated with the PMD sublayer.

16.3.4.1 Receiver sensitivity

The receiver sensitivity, defined as the minimum irradiance (in mW/cm^2) at the photodetector plane required for a frame error ratio (FER) of 4×10^{-5} with a PLCSDU of 512 octets and with an unmodulated background IR source between 800 nm and 1000 nm with a level of 0.1 mW/cm^2, shall be

1 Mbit/s: $2 \times 10^{-5} mW/cm^2$
2 Mbit/s: $8 \times 10^{-5} mW/cm^2$

16.3.4.2 Receiver dynamic range

The receiver dynamic range, defined as the ratio between the maximum and minimum irradiance at the plane normal to the receiver axis that assures an FER lower than or equal to 4×10^{-5} with a PLCSDU of 512 octets and with an unmodulated background IR source between 800 nm and 1000 nm with a level of 0.1 mW/cm^2, shall be ≥30 dB.

16.3.4.3 Receiver field-of-view (FOV)

The receiver axis is defined as the direction of incidence of the optical signal at which the received optical power is maximum.

The received optical power shall be greater than the values given in Table 74, at the angles indicated, where "angle of incidence" is the angle of the optical signal relative to the receiver axis, and "received power" is the received optical power as a percentage of that measured at the receiver axis.

Table 74—Definition of the receiver field of view

Angle of incidence	Received power
$\alpha \leq 20°$	≥ 65%
$\alpha \leq 40°$	≥ 55%
$\alpha \leq 60°$	≥ 35%
$\alpha \leq 80°$	≥ 10%

16.3.5 Energy Detect, Carrier Sense, and CCA definitions

16.3.5.1 Energy Detect (ED) signal

The ED signal shall be set true when IR energy variations in the band between 1 MHz and 10 MHz exceed 0.001 mW/cm^2.

The ED shall operate independently of the CS. The ED shall not be asserted at the minimum signal level specified in 16.3.4.1, which is below the level specified in this subclause.

This signal is not directly available to the MAC.

16.3.5.2 Carrier Sense (CS) signal

The CS shall be asserted by the PHY when it detects and locks onto an incoming PLCP Preamble signal. Conforming PHYs shall assert this condition within the first 12 μs of signal reception, at the minimum signal

level equal to the receiver sensitivity specified in 16.3.4.1, with a background IR level as specified in 16.3.4.1.

The CS shall be deasserted by the PHY when the receiving conformant device loses carrier lock.

NOTE—The 12 μs specification is somewhat less than the minimum length of the PLCP SYNC interval, which is 14.25 μs.

The CS shall operate independently of the ED and shall not require a prior ED before the acquisition and assertion of CS. This permits reception of signals at the minimum signal level specified in 16.3.4.1, even though these signals fall below the ED level.

This signal is not directly available to the MAC.

16.3.5.3 CCA

CCA shall be asserted "IDLE" by the PHY when the CS and the ED are both false, or when ED has been continuously asserted for a period of time defined by the product of dot11CCAWatchdogTimerMax and dot11CCAWatchdogCountMax without CS becoming active. When either CS or ED go true, CCA is indicated as "BUSY" to the MAC via the primitive PHY-CCA.indicate. CS and DE behavior are defined in 16.3.5.2.

Normally, CCA will be held "BUSY" throughout the period of the PLCP Header. After receiving the last PLCP bit and the first data octet, the PHY shall signal PHY-RXSTART.indicate with the parameters LENGTH and RATE. CCA shall be held "BUSY" until the number of octets specified in the decoded PLCP Header are received. At that time the PHY shall signal PHY-RXEND.indicate. The CCA may remain "BUSY" after the end of data if some form of energy is still being detected. The PHY will signal PHY-CCA.indicate with a value of IDLE only when the CCA goes "CLEAR."

The transition of CCA from "BUSY" to "IDLE" is indicated to the MAC via the primitive PHY-CCA.indicate.

If CS and ED go false before the PHY signals PHY-RXSTART.indicate, CCA is set to "IDLE" and *immediately* signaled to the MAC via PHY-CCA.indicate with a value of IDLE. If CS and ED go false after the PHY has signaled PHY-RXSTART.indicate, implying that the PLCP Header has been properly decoded, then the PHY shall not signal a change in state of CCA until the proper interval has passed for the number of octets indicated by the received PLCP LENGTH. At that time, the PHY shall signal PHY-RXEND.indicate with an RXERROR parameter of CarrierLost followed by PHY-CCA.indicate with a value of IDLE.

The transition of CCA from "CLEAR" to "BUSY" resets the CCA watchdog timer and CCA watchdog counter. dot11CCAWatchdogTimerMax and dot11CCAWatchdogCountMax are parameters available via MIB entries and can be read and set via the LME.

Rise and fall times of CCA relative to the OR'ing of the CS and ED signals shall be less than 30 ns. CS and ED are both internal signals to the PHY and are not available directly to the MAC, nor are they defined at any exposed interface.

16.3.5.4 CHNL_ID

For the IR PHY, CHNL_ID = X'01' is defined as the baseband modulation method. All other values are not defined.

16.4 PHY attributes

PHY attributes have allowed values and default values that are PHY dependent. Table 75 and Table 76 describe those values, and further specify whether they are permitted to vary from implementation to implementation.

Table 75 does not provide the definition of the attributes, but only provides the IR PHY-specific values for the attributes whose definitions are in Clause 13.

Table 75—IR PHY MIB attributes

PHY MIB object	Default value	Operational semantics	Operational behavior
dot11CCAWatchdogTimerMax	Implementation dependent	Dynamic	A conformant PHY may set this via the LME
dot11CCAWatchdogCountMax	Implementation dependent	Dynamic	A conformant PHY may set this via the LME
dot11CCAWatchdogTimerMin	22 μs	Static	Identical for all conformant PHYs
dot11CCAWatchdogCountMin	1	Static	Identical for all conformant PHYs
dot11SupportedDataRatesTx	Implementation dependent	Static	All conformant PHYs must include the value X'02' (1 Mbit/s).
dot11SupportedDataRatesRx	Implementation dependent	Static	All conformant PHYs must include the values X'02' (1 Mbit/s) and X'04' (2 Mbit/s).
dot11PhyType	03	Static	Identical for all conformant PHYs
dot11PhyTempType	X'01'	Static	Identical for all conformant PHYs

The static IR PHY characteristics, provided through the PLME-CHARACTERISTICS service primitive, are shown in Table 76. The definitions of these characteristics are in 10.4.3.

Table 76—IR PHY characteristics

Characteristic	Value
aSlotTime	8 μs
aSIFSTime	10 μs
aCCATime	5 μs
aRxTxTurnaroundTime	0 μs
aTxPLCPDelay	Implementors may choose any value for this delay as long as the requirements of aRxTxTurnaroundTime are met.
aRxPLCPDelay	1 μs
aRxTxSwitchTime	0 μs
aTxRampOnTime	0 μs
aTxRampOffTime	0 μs
aTxRFDelay	Implementors may choose any value for this delay as long as the requirements of aRxTxTurnaroundTime are met.
aRxRFDelay	Implementors may choose any value for this delay as long as the requirements of aSIFSTime and aCCATime are met.
aAirPropagationTime	1 μs
aMACProcessingDelay	2 μs
aPreambleLength	16 μs (1 Mbit/s) 20 μs (2 Mbit/s)
aPLCPHeaderLength	41 μs (1 Mbit/s) 25 μs (2 Mbit/s)
aMPDUDurationFactor	0
aMPDUMaxLength	2500
aCWmin	63
aCWmax	1023

Annex A

(normative)

Protocol Implementation Conformance Statement (PICS) proforma

A.1 Introduction

The supplier of a protocol implementation that is claimed to conform to ISO/IEC 8802.11: 1999 shall complete the following PICS proforma.

A completed PICS proforma is the PICS for the implementation in question. The PICS is a statement of which capabilities and options of the protocol have been implemented. The PICS can have a number of uses, including use

a) By the protocol implementor, as a checklist to reduce the risk of failure to conform to the standard through oversight;
b) By the supplier and acquirer, or potential acquirer, of the implementation, as a detailed indication of the capabilities of the implementation, stated relative to the common basis for understanding provided by the standard PICS proforma;
c) By the user, or potential user, of the implementation, as a basis for initially checking the possibility of interworking with another implementation (note that, while interworking can never be guaranteed, failure to interwork can often be predicted from incompatible PICS proformas);
d) By a protocol tester, as the basis for selecting appropriate tests against which to assess the claim for conformance of the implementation.

A.2 Abbreviations and special symbols

A.2.1 Status symbols

M	mandatory
O	optional
O.<n>	optional, but support of at least one of the group of options labeled by the same numeral <n> is required
pred:	conditional symbol, including predicate identification

A.2.2 General abbreviations

N/A	not applicable
AD	address function capability
CF	implementation under test (IUT) configuration
FR	MAC frame capability
FS	frame sequence capability
PC	protocol capability
PICS	protocol implementation conformance statement

A.3 Instructions for completing the PICS proforma

A.3.1 General structure of the PICS proforma

The first part of the PICS proforma, Implementation identification and Protocol summary, is to be completed as indicated with the information necessary to identify fully both the supplier and the implementation.

The main part of the PICS proforma is a fixed questionnaire, divided into subclauses, each containing a number of individual items. Answers to the questionnaire items are to be provided in the rightmost column, either by simply marking an answer to indicate a restricted choice (usually Yes or No) or by entering a value or a set or a range of values. (Note that there are some items where two or more choices from a set of possible answers may apply. All relevant choices are to be marked in these cases.)

Each item is identified by an item reference in the first column. The second column contains the question to be answered. The third column contains the reference or references to the material that specifies the item in the main body of ISO/IEC 8802-11: 1999. The remaining columns record the status of each item, i.e., whether support is mandatory, optional, or conditional, and provide the space for the answers (see also A.3.4). Marking an item as supported is to be interpreted as a statement that all relevant requirements of the subclauses and normative annexes, cited in the References column for the item, are met by the implementation.

A supplier may also provide, or be required to provide, further information, categorized as either Additional Information or Exception Information. When present, each kind of further information is to be provided in a further subclause of items labeled A<*i*> or X<*i*>, respectively, for cross-referencing purposes, where <*i*> is any unambiguous identification for the item (e.g., simply a numeral). There are no other restrictions on its format or presentation.

The PICS proforma for a station consists of A.4.1 through A.4.4 inclusive, and at least one of A.4.5, A.4.6, or A.4.7 corresponding to the PHY implemented.

A completed PICS proforma, including any Additional Information and Exception Information, is the PICS for the implementation in question.

NOTE—Where an implementation is capable of being configured in more than one way, a single PICS may be able to describe all such configurations. However, the supplier has the choice of providing more than one PICS, each covering some subset of the implementation's capabilities, if this makes for easier and clearer presentation of the information.

A.3.2 Additional information

Items of Additional Information allow a supplier to provide further information intended to assist in the interpretation of the PICS. It is not intended or expected that a large quantity of information will be supplied, and a PICS can be considered complete without any such information. Examples of such Additional Information might be an outline of the ways in which an (single) implementation can be set up to operate in a variety of environments and configurations, or information about aspects of the implementation that are outside the scope of this standard but have a bearing upon the answers to some items.

References to items of Additional Information may be entered next to any answer in the questionnaire, and may be included in items of Exception Information.

A.3.3 Exception information

It may happen occasionally that a supplier will wish to answer an item with mandatory status (after any conditions have been applied) in a way that conflicts with the indicated requirement. No preprinted answer will be found in the Support column for this. Instead, the supplier shall write the missing answer into the Support column, together with an X<*i*> reference to an item of Exception Information, and shall provide the appropriate rationale in the Exception Information item itself.

An implementation for which an Exception Information item is required in this way does not conform to ISO/IEC 8802-11: 1999.

NOTE—A possible reason for the situation described above is that a defect in ISO/IEC 8802-11: 1999 has been reported, a correction for which is expected to change the requirement not met by the implementation.

A.3.4 Conditional status

The PICS proforma contains a number of conditional items. These are items for which both the applicability of the item itself, and its status if it does apply, mandatory or optional, are dependent upon whether or not certain other items are supported.

Where a group of items is subject to the same condition for applicability, a separate preliminary question about the condition appears at the head of the group, with an instruction to skip to a later point in the questionnaire if the Not Applicable (N/A) answer is selected. Otherwise, individual conditional items are indicated by a conditional symbol in the Status column.

A conditional symbol is of the form "<pred>:<S>", where "<pred>" is a predicate as described below, and "<S>" is one of the status symbols M or O.

If the value of the predicate is true, the conditional item is applicable, and its status is given by S: the support column is to be completed in the usual way. Otherwise, the conditional item is not relevant and the N/A answer is to be marked.

A predicate is one of the following:

a) An item-reference for an item in the PICS proforma: the value of the predicate is true if the item is marked as supported, and is false otherwise.
b) A boolean expression constructed by combining item-references using the boolean operator OR: the value of the predicate is true if one or more of the items is marked as supported, and is false otherwise.

Each item referenced in a predicate, or in a preliminary question for grouped conditional items, is indicated by an asterisk in the Item column.

A.4 PICS proforma—ISO/IEC 8802-11: 1999[7]

A.4.1 Implementation identification

Supplier	
Contact point for queries about the PICS	
Implementation Name(s) and Version(s)	
Other information necessary for full identification, e.g., name(s) and version(s) of the machines and/or operating systems(s), system names	

NOTES

1—Only the first three items are required for all implementations. Other information may be completed as appropriate in meeting the requirement for full identification.

2—The terms Name and Version should be interpreted appropriately to correspond with a supplier's terminology (e.g., Type, Series, Model).

A.4.2 Protocol summary, ISO/IEC 8802-11: 1999

Identification of protocol standard	ISO/IEC 8802-11: 1999
Identification of amendments and corrigenda to this PICS proforma that have been completed as part of this PICS	Amd. : Corr. : Amd. : Corr. :
Have any exception items been required? (See A.3.3; the answer Yes means that the implementation does not conform to ISO/IEC 8802-11: 1999.)	Yes ❑ No ❑

Date of statement (dd/mm/yy)	

[7]*Copyright release for PICS proforma:* Users of this standard may freely reproduce the PICS proforma in this annex so that it can be used for its intended purpose and may further publish the completed PICS.

A.4.3 IUT configuration

Item	IUT configuration	References	Status	Support
	What is the configuration of the IUT?			
* CF1	Access Point (AP)	5.2	O.1	Yes ❑ No ❑
* CF2	Independent station (*not* an AP)	5.2	O.1	Yes ❑ No ❑
* CF3	Frequency-Hopping spread spectrum (FHSS) PHY for the 2.4 GHz band		O.2	Yes ❑ No ❑
* CF4	Direct Sequence Spread Spectrum (DSSS) PHY for the 2.4 GHz band		O.2	Yes ❑ No ❑
* CF5	Infrared PHY		O.2	Yes ❑ No ❑

A.4.4 MAC protocol

A.4.4.1 MAC protocol capabilities

Item	Protocol capability	References	Status	Support
	Are the following MAC protocol capabilities supported?			
PC1	Authentication service	5.4.3.1, 5.4.3.2, 5.7.6, 5.7.7, 8.1, Annex C	M	Yes ❑ No ❑
PC1.1	Authentication state	5.5	M	Yes ❑ No ❑
PC1.2	Open System authentication	8.1.1	M	Yes ❑ No ❑
PC1.3	Shared Key authentication	8.1.2, 8.3	PC2:M	Yes ❑ No ❑ N/A ❑
* PC2	WEP algorithm	5.4.3.3, 8.2, Annex C	O	Yes ❑ No ❑
PC2.1	WEP Encryption procedure	8.2.3, 8.2.4, 8.2.5	PC2:M	Yes ❑ No ❑ N/A ❑
PC2.2	WEP Decryption procedure	8.2.3, 8.2.4, 8.2.5	PC2:M	Yes ❑ No ❑ N/A ❑
PC2.3	Security services management	8.3	M	Yes ❑ No ❑
PC3	Distributed Coordination function	9.1, 9.2, Annex C	M	Yes ❑ No ❑
PC3.1	Net Allocation Vector (NAV) function	9.2.1, 9.2.5, 9.3.2.2	M	Yes ❑ No ❑
PC3.2	Interframe space usage and timing	9.2.3, 9.2.5, 9.2.10	M	Yes ❑ No ❑
PC3.3	Random Backoff function	9.2.4	M	Yes ❑ No ❑
PC3.4	DCF Access procedure	9.2.5.1, 9.2.5.5	M	Yes ❑ No ❑
PC3.5	Random Backoff procedure	9.2.5.2	M	Yes ❑ No ❑
PC3.6	Recovery procedures and retransmit limits	9.2.5.3	M	Yes ❑ No ❑
PC3.7	RTS/CTS procedure	9.2.5.4, 9.2.5.6, 9.2.5.7	M	Yes ❑ No ❑

A.4.4.1 MAC protocol capabilities *(continued)*

Item	Protocol capability	References	Status	Support
PC3.8	Directed MPDU transfer	9.2.6	M	Yes ❑ No ❑
PC3.9	Broadcast and multicast MPDU transfer	9.2.7	M	Yes ❑ No ❑
PC3.10	MAC level acknowledgment	9.2.2, 9.2.8	M	Yes ❑ No ❑
PC3.11	Duplicate detection and recovery	9.2.9	M	Yes ❑ No ❑
* PC4	Point coordinator (PC)	9.1, 9.3, Annex C	CF1:O	Yes ❑ No ❑ N/A ❑
PC4.1	Maintenance of CFP structure and timing	9.3.1, 9.3.2	PC4:M	Yes ❑ No ❑ N/A ❑
PC4.2	PCF MPDU transfer from PC	9.3.3	PC4:M	Yes ❑ No ❑ N/A ❑
* PC4.3	PCF MPDU transfer to PC	9.3.3	PC4:O	Yes ❑ No ❑ N/A ❑
PC4.4	Overlapping PC provisions	9.3.3.2	PC4:M	Yes ❑ No ❑ N/A ❑
PC4.5	Polling list maintenance	9.3.4	PC4.3: M	Yes ❑ No ❑ N/A ❑
* PC5	CF-Pollable	9.1, 9.3, Annex C	CF2:O	Yes ❑ No ❑ N/A ❑
PC5.1	Interpretation of CFP structure and timing	9.3.1, 9.3.2	PC5:M	Yes ❑ No ❑ N/A ❑
PC5.2	PCF MPDU transfer to/from and CF-Pollable STA	9.3.3	PC5:M	Yes ❑ No ❑ N/A ❑
PC5.3	Polling list update	9.3.4	PC5:M	Yes ❑ No ❑ N/A ❑
PC6	Fragmentation	9.2, 9.4, Annex C	M	Yes ❑ No ❑
PC7	Defragmentation	9.2, 9.5, Annex C	M	Yes ❑ No ❑
PC8	MAC data service	9.1.5, 9.8, Annex C	M	Yes ❑ No ❑
PC8.1	Reorderable-Multicast service class	9.8	M	Yes ❑ No ❑
PC8.2	StrictlyOrdered service class	9.8	O	Yes ❑ No ❑
PC9	Multirate support	9.6, Annex C	M	Yes ❑ No ❑
* PC10	Multiple outstanding MSDU support	9.8, Annex C	O	Yes ❑ No ❑
PC10.1	Multiple outstanding MSDU transmission restrictions	9.8	PC10:M	Yes ❑ No ❑ N/A ❑
PC11	Timing synchronization	11.1, Annex C	M	Yes ❑ No ❑
PC11.1	Timing in an infrastructure network	11.1.1.1, 11.1.4	CF1:M	Yes ❑ No ❑ N/A ❑
PC11.2	Timing in an Independent BSS (IBSS)	11.1.1.2, 11.1.4	CF2:M	Yes ❑ No ❑ N/A ❑
PC11.3	Beacon Generation function	11.1.2	M	Yes ❑ No ❑ N/A ❑
PC11.5	TSF synchronization and accuracy	11.1.2	M	Yes ❑ No ❑
PC11.5	Infrastructure BSS initialization	11.1.3	CF1:M	Yes ❑ No ❑ N/A ❑

A.4.4.1 MAC protocol capabilities *(continued)*

Item	Protocol capability	References	Status	Support
PC11.6	Independent BSS initialization	11.1.3	CF2:M	Yes ❑ No ❑ N/A ❑
PC11.7	Passive scanning	11.1.3	CF2:M	Yes ❑ No ❑ N/A ❑
PC11.8	Active scanning	11.1.3	CF2:M	Yes ❑ No ❑ N/A ❑
PC11.9	Probe response	11.1.3	M	Yes ❑ No ❑
PC11.10	Hop Synchronization function	11.1.5	CF3:M	Yes ❑ No ❑ N/A ❑
PC12	Infrastructure power management	11.2.1, Annex C	M	Yes ❑ No ❑
PC12.1	Station power management modes	11.2.1.1, 11.2.1.8	CF2:M	Yes ❑ No ❑ N/A ❑
PC12.2	TIM transmission	11.2.1.2, 11.2.1.3	CF1:M	Yes ❑ No ❑ N/A ❑
PC12.3	AP function during CP	11.2.1.4	CF1:M	Yes ❑ No ❑ N/A ❑
PC12.4	AP function during CFP	11.2.1.5	PC4:M	Yes ❑ No ❑ N/A ❑
PC12.5	Receive function during CP	11.2.1.6	CF2:M	Yes ❑ No ❑ N/A ❑
PC12.6	Receive function during CFP	11.2.1.7	PC5:M	Yes ❑ No ❑ N/A ❑
PC12.7	Aging function	11.2.1.9	CF1:M	Yes ❑ No ❑ N/A ❑
PC13	IBSS power management	11.2.2, Annex C	CF2:M	Yes ❑ No ❑ N/A ❑
PC13.1	Initialization of power management	11.2.2.2	CF2:M	Yes ❑ No ❑ N/A ❑
PC13.2	STA power state transitions	11.2.2.3	CF2:M	Yes ❑ No ❑ N/A ❑
PC13.3	ATIM and frame transmission	11.2.2.4	CF2:M	Yes ❑ No ❑ N/A ❑
PC14	Association and reassociation	5.4, 5.7, 11.3, Annex C	M	Yes ❑ No ❑
PC14.1	Association state	5.5	M	Yes ❑ No ❑
PC14.2	STA association procedure	11.3.1	CF2:M	Yes ❑ No ❑ N/A ❑
PC14.3	AP association procedure	11.3.2	CF1:M	Yes ❑ No ❑ N/A ❑
PC14.4	STA reassociation procedure	11.3.3	CF2:M	Yes ❑ No ❑ N/A ❑
PC14.5	AP reassociation procedure	11.3.4	CF1:M	Yes ❑ No ❑ N/A ❑
PC15	Management information base (MIB)	Annex D	M	Yes ❑ No ❑
PC15.1	dot11SMTbase, dot11SmtAuthenticationAlgorithms	Annex D	M	Yes ❑ No ❑
* PC15.2	dot11SMTprivacy	Annex D	PC2:M	Yes ❑ No ❑ N/A ❑
PC15.3	dot11MACbase, dot11CountersGroup, dot11MacGroupAddresses	Annex D	M	Yes ❑ No ❑
* PC15.4	dot11MACStatistics	Annex D	O	Yes ❑ No ❑
PC15.5	dot11ResourceTypeID	Annex D	M	Yes ❑ No ❑

A.4.4.2 MAC frames

Item	MAC frame	References	Status	Support
	Is transmission of the following MAC frames supported?	7, Annex C		
FT1	Association request	7	CF2:M	Yes ❑ No ❑ N/A ❑
FT2	Association response	7	CF1:M	Yes ❑ No ❑ N/A ❑
FT3	Reassociation request	7	CF2:M	Yes ❑ No ❑ N/A ❑
FT4	Reassociation response	7	CF1:M	Yes ❑ No ❑ N/A ❑
FT5	Probe request	7	CF2:M	Yes ❑ No ❑ N/A ❑
FT6	Probe response	7	M	Yes ❑ No ❑
FT7	Beacon	7	M	Yes ❑ No ❑
FT8	ATIM	7	CF2:M	Yes ❑ No ❑ N/A ❑
FT9	Disassociation	7	M	Yes ❑ No ❑
FT10	Authentication	7	M	Yes ❑ No ❑
FT11	Deauthentication	7	M	Yes ❑ No ❑
FT12	PS-Poll	7	CF2:M	Yes ❑ No ❑ N/A ❑
FT13	RTS	7	M	Yes ❑ No ❑
FT14	CTS	7	M	Yes ❑ No ❑
FT15	ACK	7	M	Yes ❑ No ❑
FT16	CF-End	7	PC4:M	Yes ❑ No ❑ N/A ❑
FT17	CF End+CF-Ack	7	PC4:M	Yes ❑ No ❑ N/A ❑
FT18	Data	7	M	Yes ❑ No ❑
FT19	Data + CF-Ack	7	(PC4 OR PC5):M	Yes ❑ No ❑ N/A ❑
FT20	Data + CF-Poll	7	PC4.3:M	Yes ❑ No ❑ N/A ❑
FT21	Data + CF-Ack+CF-Poll	7	PC4.3:M	Yes ❑ No ❑ N/A ❑
FT22	Null	7	M	Yes ❑ No ❑
FT23	CF-Ack (no data)	7	(PC4 OR PC5):M	Yes ❑ No ❑ N/A ❑
FT24	CF-Poll (no data)	7	PC4.3:M	Yes ❑ No ❑ N/A ❑
FT25	CF-Ack+CF-Poll (no data)	7	PC4.3:M	Yes ❑ No ❑ N/A ❑
	Is reception of the following MAC frames supported?	7, Annex C		
FR1	Association request	7	CF1:M	Yes ❑ No ❑ N/A ❑
FR2	Association response	7	CF2:M	Yes ❑ No ❑ N/A ❑
FR3	Reassociation request	7	CF1:M	Yes ❑ No ❑ N/A ❑
FR4	Reassociation response	7	CF2:M	Yes ❑ No ❑ N/A ❑
FR5	Probe request	7	M	Yes ❑ No ❑
FR6	Probe response	7	M	Yes ❑ No ❑
FR7	Beacon	7	M	Yes ❑ No ❑
FR8	ATIM	7	CF2:M	Yes ❑ No ❑ N/A ❑
FR9	Disassociation	7	M	Yes ❑ No ❑
FR10	Authentication	7	M	Yes ❑ No ❑

A.4.4.2 MAC frames *(continued)*

Item	MAC frame	References	Status	Support
FR11	Deauthentication	7	M	Yes ❑ No ❑
FR12	PS-Poll	7	CF1:M	Yes ❑ No ❑ N/A ❑
FR13	RTS	7	M	Yes ❑ No ❑
FR14	CTS	7	M	Yes ❑ No ❑
FR15	ACK	7	M	Yes ❑ No ❑
FR16	CF-End	7	M	Yes ❑ No ❑
FR17	CF End+CF-Ack	7	M	Yes ❑ No ❑
FR18	Data	7	M	Yes ❑ No ❑
FR19	Data + CF-Ack	7	M	Yes ❑ No ❑
FR20	Data + CF-Poll	7	PC5:M	Yes ❑ No ❑ N/A ❑
FR21	Data + CF-Ack+CF-Poll	7	PC5:M	Yes ❑ No ❑ N/A ❑
FR22	Null	7	M	Yes ❑ No ❑
FR23	CF-Ack (no data)	7	(PC4 OR PC5):M	Yes ❑ No ❑ N/A ❑
FR24	CF-Poll (no data)	7	PC5:M	Yes ❑ No ❑ N/A ❑
FR25	CF-Ack+CF-Poll (no data)	7	PC5:M	Yes ❑ No ❑ N/A ❑

A.4.4.3 Frame exchange sequences

Item	Frame exchange sequence	References	Status	Support
	Are the following frame sequences supported?			
FS1	Basic frame sequences	9.7, Annex C	M	Yes ❑ No ❑
FS2	CF-Frame sequences	9.7, Annex C	(PC4 OR PC5):M	Yes ❑ No ❑ N/A ❑

A.4.4.4 MAC addressing functions

Item	MAC Address function	References	Status	Support
	Are the following MAC Addressing functions supported?			
AD1	STA universal individual IEEE 802 address	5.3.3, 7.1.3.3	M	Yes ❑ No ❑
AD2	BSS identifier generation	7.1.3.3, 11.1.3, Annex C	M	Yes ❑ No ❑
AD3	Receive address matching	7.1.3.3, 7.2.2, Annex C	M	Yes ❑ No ❑ N/A ❑

A.4.5 Frequency-Hopping PHY functions

Item	Protocol feature	References	Status	Support
	Which requirements and options does the PHY support?			
FH1	PHY service primitive parameters			
FH1.1	TXVECTOR parameter: LENGTH	14.2.2.1	M	Yes ❑ No ❑
FH1.2	TXVECTOR parameter: PLCPBITRATE	14.2.2.2	M	Yes ❑ No ❑
FH1.2.1	PLCPBITRATE = X'00' (1.0 Mbit/s)	14.2.2.2	M	Yes ❑ No ❑
* FH1.2.2	PLCPBITRATE = X'02' (2.0 Mbit/s)	14.2.2.2	O	Yes ❑ No ❑
FH1.3	RXVECTOR parameter: LENGTH	14.2.3.1	M	Yes ❑ No ❑
FH1.4	RXVECTOR parameter: RSSI	14.2.3.2	O	Yes ❑ No ❑
FH2	PLCP frame format			
FH2.1	PLCP Preamble: Sync	14.3.2.1.1	M	Yes ❑ No ❑
FH2.2	PLCP Preamble: Start Frame Delimiter	14.3.2.1.2	M	Yes ❑ No ❑
FH2.3	PLCP Header: Length Word	14.3.2.2.1	M	Yes ❑ No ❑
FH2.4	PLCP Header: Signaling field	14.3.2.2.2	M	Yes ❑ No ❑
FH2.5	PLCP Header: Header Error Check	14.3.2.2.3	M	Yes ❑ No ❑
FH2.6	PLCP Data Whitener: Scrambling and bias suppression encoding	14.3.2.3, 14.3.3.1.1	M	Yes ❑ No ❑
FH3	PLCP Transmit procedure			
FH3.1	Transmit: transmit on MAC request	14.3.3.1.1	M	Yes ❑ No ❑
FH3.2	Transmit: format and whiten frame	14.3.3.1.1	M	Yes ❑ No ❑
FH3.3	Transmit: Timing	14.3.3.1.1	M	Yes ❑ No ❑
FH4	PLCP CS/CCA procedure			
FH4.1	CS/CCA: perform on a minimum of one antenna	14.3.3.2.1	M	Yes ❑ No ❑
FH4.2.	CS/CCA: Detect preamble starting up to 20 μs after start of slot time	14.3.3.2.1	M	Yes ❑ No ❑
FH4.3	CS/CCA: Detect preamble starting at least 16 μs prior to end of slot time	14.3.3.2.1	M	Yes ❑ No ❑
FH4.4	CS/CCA: Detect random data	14.3.3.2.1	M	Yes ❑ No ❑
FH4.5	CS/CCA: Perform on antenna with essentially same gain and pattern as transmit antenna	14.3.3.2.1	M	Yes ❑ No ❑
FH4.6	CS/CCA: Detect valid SFD and PLCP header	14.3.3.2.1	M	Yes ❑ No ❑
FH4.7	CS/CCA: Maintain BUSY indication until end of length contained in valid PLCP header	14.3.3.2.1	M	Yes ❑ No ❑
FH5	PLCP Receive procedure			
FH5.1	Receive: Receive and dewhiten frame	14.3.3.3.1	M	Yes ❑ No ❑
FH6	PHY LME			
FH6.1	PLME: Support FH sync	14.4.2.2	M	Yes ❑ No ❑
FH6.2	PLME: Support PLME primitives	14.4.3.2	O	Yes ❑ No ❑

A.4.5 Frequency-Hopping PHY functions *(continued)*

Item	Protocol feature	References	Status	Support
FH7	Geographic area specific requirements			
* FH7.1	Geographic areas			
FH7.1.1	North America	14.6.2	O.1	Yes ❑ No ❑
FH7.1.2	Most of Europe	14.6.2	O.1	Yes ❑ No ❑
FH7.1.3	Japan	14.6.2	O.1	Yes ❑ No ❑
FH7.1.4	Spain	14.6.2	O.1	Yes ❑ No ❑
FH7.1.5	France	14.6.2	O.1	Yes ❑ No ❑
FH7.2	Operating frequency range	14.6.3	FH7.1:M	Yes ❑ No ❑
FH7.3	Number of operating channels	14.6.4	FH7.1:M	Yes ❑ No ❑
FH7.4	Operating channel frequencies	14.6.5	FH7.1:M	Yes ❑ No ❑
FH7.5	Occupied channel bandwidth	14.6.6	FH7.1:M	Yes ❑ No ❑
FH7.6	Minimum hop rate	14.6.7	FH7.1:M	Yes ❑ No ❑
FH7.7	Hop sequences	14.6.8	FH7.1:M	Yes ❑ No ❑
FH7.8	Unwanted emissions	14.6.9	FH7.1:M	Yes ❑ No ❑
FH8	1 Mbit/s PMD			
FH8.1	Modulation 2GFSK, BT=0.5, 1=positive frequency deviation, 0=negative frequency deviation	14.6.10	M	Yes ❑ No ❑
FH8.2	Peak frequency deviation	14.6.10	M	Yes ❑ No ❑
FH8.3	Zero-Crossing error	14.6.10	M	Yes ❑ No ❑
FH8.4	Nominal channel data rate	14.6.11	M	Yes ❑ No ❑
FH8.5	Channel switching/settling time	14.6.12	M	Yes ❑ No ❑
FH8.6	Receive to transmit switch time	14.6.13	M	Yes ❑ No ❑
FH8.7	Nominal transmit power	14.6.14.1	M	Yes ❑ No ❑
FH8.8	Transmit power levels	14.6.14.2	M	Yes ❑ No ❑
FH8.9	Transmit power level control to <100 mW	14.6.14.3	M	Yes ❑ No ❑
FH8.10	Transmit spectrum shape	14.6.14.4	M	Yes ❑ No ❑
FH8.11	Transmit center frequency tolerance	14.6.14.5	M	Yes ❑ No ❑
FH8.12	Transmitter ramp periods	14.6.14.6	M	Yes ❑ No ❑
FH8.13	Receiver input dynamic range	14.6.15.1	M	Yes ❑ No ❑
FH8.14	Receiver center frequency acceptance range	14.6.15.2	M	Yes ❑ No ❑
FH8.15	Clear channel assessment power threshold for a probability of detection of 90% (preamble)/70% (random data) for 100 mW units	14.6.15.3	M	Yes ❑ No ❑
FH8.16	Clear channel assessment power threshold for units >100 mW; sensitivity threshold is 1/2 dB lower for every dB above 20 dBm	14.6.15.3	M	Yes ❑ No ❑
FH8.17	Minimum receiver sensitivity at FER=3% with 400 octet frames	14.6.15.4	M	Yes ❑ No ❑
FH8.18	Intermodulation protection	14.6.15.5	M	Yes ❑ No ❑

A.4.5 Frequency-Hopping PHY functions *(continued)*

Item	Protocol feature	References	Status	Support
FH8.19	Desensitization	14.6.15.6	M	Yes ❑ No ❑
FH8.20	Operating temperature range	14.6.16	M	Yes ❑ No ❑
FH8.20.1	Temperature type 1	14.6.16	O	Yes ❑ No ❑
FH8.20.2	Temperature type 2	14.6.16	O	Yes ❑ No ❑
FH8.20.3	Temperature type 3	14.6.16	O	Yes ❑ No ❑
FH9	2 Mbit/s PMD			
FH9.1	All 1M PMD requirements	14.7.1	FH1.2.2:M	Yes ❑ No ❑ N/A ❑
FH9.2	Modulation 4GFSK, BT=0.5	14.7.2	FH1.2.2:M	Yes ❑ No ❑ N/A ❑
FH9.3	Frame structure for 2M PHY	14.7.2.1	FH1.2.2:M	Yes ❑ No ❑ N/A ❑
FH9.4	Nominal channel data rate	14.7.3	FH1.2.2:M	Yes ❑ No ❑ N/A ❑
FH9.5	Input dynamic range	14.7.4	FH1.2.2:M	Yes ❑ No ❑ N/A ❑
FH9.6	Minimum receiver sensitivity at FER=3% with 400 octet frames	14.7.5	FH1.2.2:M	Yes ❑ No ❑ N/A ❑
FH9.7	Intermodulation protection	14.7.6	FH1.2.2:M	Yes ❑ No ❑ N/A ❑
FH9.8	Desensitization	14.7.7	FH1.2.2:M	Yes ❑ No ❑ N/A ❑
FH10	MIB	13.1, 14.8, Annex D	M	Yes ❑ No ❑
FH10.1	dot11PhyFHSSComplianceGroup, dot11PhyRegDomainsSupportGroup, and dot11PhyOperationComplianceGroup	13.1,14.8	M	Yes ❑ No ❑

A.4.6 Direct sequence PHY functions

Item	PHY feature	References	Status	Support
	PLCP sublayer procedures	15.2		
DS1	Preamble prepend on TX	15.2.1	M	Yes ❑ No ❑
DS1.1	PLCP frame format	15.2.2, 15.2.3	M	Yes ❑ No ❑
DS1.2	PLCP integrity check generation	15.2.3, 15.2.3.6	M	Yes ❑ No ❑
DS1.3	TX rate change capability	15.2.3.3, 15.2.5	M	Yes ❑ No ❑
DS1.4	Supported data rates	15.1, 15.2.3.3	M	Yes ❑ No ❑
DS1.5	Data whitener scrambler	15.2.4	M	Yes ❑ No ❑
DS1.6	Scrambler initialization	15.2.4	M	Yes ❑ No ❑
DS2	Preamble process on RX	15.2.1		
DS2.1	PLCP frame format	15.2.2, 15.2.3	M	Yes ❑ No ❑
DS2.2	PLCP integrity check verify	15.2.3, 15.2.3.6	M	Yes ❑ No ❑
DS2.3	RX Rate change capability	15.2.3.3, 15.2.5	M	Yes ❑ No ❑
DS2.4	Data whitener descrambler	15.2.4	M	Yes ❑ No ❑
DS3	PN code sequence	15.4.6.3	M	Yes ❑ No ❑
DS4	Chipping continue on power down	15.2.6	O	Yes ❑ No ❑
*DS5	Operating channel capability	15.2.6, 15.4.6.2		

A.4.6 Direct sequence PHY functions *(continued)*

Item	PHY feature	References	Status	Support
* DS5.1	North America (FCC)	15.2.6, 15.4.6.2	DS5:O.1	Yes ❑ No ❑ N/A ❑
DS5.1.1	channel 1	15.2.6, 15.4.6.2	DS5.1:M	Yes ❑ No ❑ N/A ❑
DS5.1.2	channel 2	15.2.6, 15.4.6.2	DS5.1:M	Yes ❑ No ❑ N/A ❑
DS5.1.3	channel 3	15.2.6, 15.4.6.2	DS5.1:M	Yes ❑ No ❑ N/A ❑
DS5.1.4	channel 4	15.2.6, 15.4.6.2	DS5.1:M	Yes ❑ No ❑ N/A ❑
DS5.1.5	channel 5	15.2.6, 15.4.6.2	DS5.1:M	Yes ❑ No ❑ N/A ❑
DS5.1.6	channel 6	15.2.6, 15.4.6.2	DS5.1:M	Yes ❑ No ❑ N/A ❑
DS5.1.7	channel 7	15.2.6, 15.4.6.2	DS5.1:M	Yes ❑ No ❑ N/A ❑
DS5.1.8	channel 8	15.2.6, 15.4.6.2	DS5.1:M	Yes ❑ No ❑ N/A ❑
DS5.1.9	channel 9	15.2.6, 15.4.6.2	DS5.1:M	Yes ❑ No ❑ N/A ❑
DS5.1.10	channel 10	15.2.6, 15.4.6.2	DS5.1:M	Yes ❑ No ❑ N/A ❑
DS5.1.11	channel 11	15.2.6, 15.4.6.2	DS5.1:M	Yes ❑ No ❑ N/A ❑
* DS5.2	Canada (IC)	15.2.6, 15.4.6.2	DS5:O.1	Yes ❑ No ❑ N/A ❑
DS5.2.1	channel 1	15.2.6, 15.4.6.2	DS5.2:M	Yes ❑ No ❑ N/A ❑
DS5.2.2	channel 2	15.2.6, 15.4.6.2	DS5.2:M	Yes ❑ No ❑ N/A ❑
DS5.2.3	channel 3	15.2.6, 15.4.6.2	DS5.2:M	Yes ❑ No ❑ N/A ❑
DS5.2.4	channel 4	15.2.6, 15.4.6.2	DS5.2:M	Yes ❑ No ❑ N/A ❑
DS5.2.5	channel 5	15.2.6, 15.4.6.2	DS5.2:M	Yes ❑ No ❑ N/A ❑
DS5.2.6	channel 6	15.2.6, 15.4.6.2	DS5.2:M	Yes ❑ No ❑ N/A ❑
DS5.2.7	channel 7	15.2.6, 15.4.6.2	DS5.2:M	Yes ❑ No ❑ N/A ❑
DS5.2.8	channel 8	15.2.6, 15.4.6.2	DS5.2:M	Yes ❑ No ❑ N/A ❑
DS5.2.9	channel 9	15.2.6, 15.4.6.2	DS5.2:M	Yes ❑ No ❑ N/A ❑
DS5.2.10	channel 10	15.2.6, 15.4.6.2	DS5.2:M	Yes ❑ No ❑ N/A ❑
DS5.2.11	channel 11	15.2.6, 15.4.6.2	DS5.2:M	Yes ❑ No ❑ N/A ❑
* DS5.3	Europe (ETSI)	15.2.6, 15.4.6.2	DS5:O.1	Yes ❑ No ❑ N/A ❑
DS5.3.1	channel 1	15.2.6, 15.4.6.2	DS5.3:M	Yes ❑ No ❑ N/A ❑
DS5.3.2	channel 2	15.2.6, 15.4.6.2	DS5.3:M	Yes ❑ No ❑ N/A ❑
DS5.3.3	channel 3	15.2.6, 15.4.6.2	DS5.3:M	Yes ❑ No ❑ N/A ❑
DS5.3.4	channel 4	15.2.6, 15.4.6.2	DS5.3:M	Yes ❑ No ❑ N/A ❑
DS5.3.5	channel 5	15.2.6, 15.4.6.2	DS5.3:M	Yes ❑ No ❑ N/A ❑
DS5.3.6	channel 6	15.2.6, 15.4.6.2	DS5.3:M	Yes ❑ No ❑ N/A ❑
DS5.3.7	channel 7	15.2.6, 15.4.6.2	DS5.3:M	Yes ❑ No ❑ N/A ❑
DS5.3.8	channel 8	15.2.6, 15.4.6.2	DS5.3:M	Yes ❑ No ❑ N/A ❑
DS5.3.9	channel 9	15.2.6, 15.4.6.2	DS5.3:M	Yes ❑ No ❑ N/A ❑
DS5.3.10	channel 10	15.2.6, 15.4.6.2	DS5.3:M	Yes ❑ No ❑ N/A ❑
DS5.3.11	channel 11	15.2.6, 15.4.6.2	DS5.3:M	Yes ❑ No ❑ N/A ❑
DS5.3.12	channel 12	15.2.6, 15.4.6.2	DS5.3:M	Yes ❑ No ❑ N/A ❑
DS5.3.13	channel 13	15.2.6, 15.4.6.2	DS5.3:M	Yes ❑ No ❑ N/A ❑
* DS5.4	France	15.2.6, 15.4.6.2	DS5:O.1	Yes ❑ No ❑ N/A ❑
DS5.4.1	channel 10	15.2.6, 15.4.6.2	DS5.4:M	Yes ❑ No ❑ N/A ❑

A.4.6 Direct sequence PHY functions *(continued)*

Item	PHY feature	References	Status	Support
DS5.4.2	channel 11	15.2.6, 15.4.6.2	DS5.4:M	Yes ❑ No ❑ N/A ❑
DS5.4.3	channel 12	15.2.6, 15.4.6.2	DS5.4:M	Yes ❑ No ❑ N/A ❑
DS5.4.4	channel 13	15.2.6, 15.4.6.2	DS5.4:M	Yes ❑ No ❑ N/A ❑
* DS5.5	Spain	15.2.6, 15.4.6.2	DS5:O.1	Yes ❑ No ❑ N/A ❑
DS5.5.1	channel 10	15.2.6, 15.4.6.2	DS5.5:M	Yes ❑ No ❑ N/A ❑
DS5.5.2	channel 11	15.2.6, 15.4.6.2	DS5.5:M	Yes ❑ No ❑ N/A ❑
* DS5.6	Japan (RCR)	15.2.6, 15.4.6.2	DS5:O.1	Yes ❑ No ❑ N/A ❑
DS6	Bits to symbol mapping	15.4.6.4		
DS6.1	1 Mbit/s	15.4.6.4	M	Yes ❑ No ❑
DS6.2	2 Mbit/s	15.4.6.4	M	Yes ❑ No ❑
*DS7	CCA functionality	15.4.8.4		
DS7.1	Energy Only (RSSI above threshold)	15.4.8.4	DS7:O.2	Yes ❑ No ❑
DS7.2	IEEE 802.11 DSSS correlation	15.4.8.4	DS7:O.2	Yes ❑ No ❑
DS7.3	Both methods	15.4.8.4	DS7:O.2	Yes ❑ No ❑
DS7.4	Hold CCA busy for packet duration of a correctly received PLCP but carrier lost during reception of MPDU	15.2.7	M	Yes ❑ No ❑
DS7.5	Hold CCA busy for packet duration of a correctly received but out of specification PLCP	15.2.7	M	Yes ❑ No ❑
DS8	Transmit antenna selection	15.4.5.5, 15.4.5.6	O	Yes ❑ No ❑
DS9	Receive antenna diversity	15.4.5.5, 15.4.5.6, 15.4.5.7	O	Yes ❑ No ❑
*DS10	Antenna port(s) availability	15.4.6.9	O	Yes ❑ No ❑
DS10.1	50 Ω impedance	15.4.6.9	DS10:M	Yes ❑ No ❑ N/A ❑
*DS11	Transmit power level support	15.4.5.8, 15.4.7.3	O	Yes ❑ No ❑
DS11.1	If greater than 100 mW capability	15.4.7.3	DS11:M	Yes ❑ No ❑ N/A ❑
*DS12	Radio type (temperature range)	15.4.6.10		
DS12.1	Type 1	15.4.6.10	DS12:O.3	Yes ❑ No ❑ N/A ❑
DS12.2	Type 2	15.4.6.10	DS12:O.3	Yes ❑ No ❑ N/A ❑
DS13	Spurious emissions conformance	15.4.6.5	M	Yes ❑ No ❑
DS14	TX-RX turnaround time	15.4.6.6	M	Yes ❑ No ❑
DS15	RX-TX turnaround time	15.4.6.7	M	Yes ❑ No ❑
DS16	Slot time	15.4.6.8	M	Yes ❑ No ❑
DS17	ED reporting time	15.4.6.8, 15.4.8.4	M	Yes ❑ No ❑
DS18	Minimum transmit power level	15.4.7.2	M	Yes ❑ No ❑
DS19	Transmit spectral mask conformance	15.4.7.4	M	Yes ❑ No ❑
DS20	Transmitted center frequency tolerance	15.4.7.5	M	Yes ❑ No ❑

A.4.6 Direct sequence PHY functions *(continued)*

Item	PHY feature	References	Status	Support
DS21	Chip clock frequency tolerance	15.4.7.6	M	Yes ❑ No ❑
DS22	Transmit power on ramp	15.4.7.7	M	Yes ❑ No ❑
DS23	Transmit power down ramp	15.4.7.7	M	Yes ❑ No ❑
DS24	RF carrier suppression	15.4.7.8	M	Yes ❑ No ❑
DS25	Transmit modulation accuracy	15.4.7.9	M	Yes ❑ No ❑
DS26	Receiver minimum input level sensitivity	15.4.8.1	M	Yes ❑ No ❑
DS27	Receiver maximum input level	15.4.8.2	M	Yes ❑ No ❑
DS28	Receiver adjacent channel rejection	15.4.8.3	M	Yes ❑ No ❑
DS29	MIB	13.1, 15.3.2, Annex D	M	Yes ❑ No ❑
DS29.1	dot11PhyDSSSComplianceGroup, dot11PhyRegDomainsSupportGroup, and dot11PhyOperationComplianceGroup	13.1, 15.3.2	M	Yes ❑ No ❑

A.4.7 Infrared baseband PHY functions

Item	Feature	References	Status	Support
IR1	Is the transmitted SYNC field length in the range of required number of PPM slots, with the absence of a pulse in the last slot of the field?	16.2.4.1	M	Yes ❑
IR2	Is the transmitted SYNC field entirely populated by alternating presence and absence of pulses in consecutive PPM slots, with the absence of a pulse in the last slot of the field?	16.2.4.1	M	Yes ❑
IR3	Is the transmitted SFD field the binary sequence 1001, where 1 indicates a pulse in the PPM slot and 0 indicates no pulse in the PPM slot?	16.2.4.2	M	Yes ❑
IR4	Is the transmitted DR field pulse sequence equal to the correct value for the data rate provided by the TXVECTOR parameter PLCP BITRATE, where 1 indicates a pulse in the PPM slot and 0 indicates no pulse in the PPM slot?	16.2.4.3	M	Yes ❑
IR5	Is the transmitted DCLA field 32 PPM slots long with the specified sequence for 1 Mbit/s, where 1 indicates a pulse in the PPM slot and 0 indicates no pulse in the PPM slot? 1 Mbit/s: 00000000100000000000000010000000	16.2.4.4	M	Yes ❑
* IR5a	Does the unit support 2 Mbit/s transmission?	16.2.4.4	O	Yes ❑ No ❑

A.4.7 Infrared baseband PHY functions *(continued)*

Item	Feature	References	Status	Support
IR5b	If the unit supports 2 Mbit/s transmission, is the transmitted DCLA field 32 PPM slots long with the specified sequence for 2 Mbit/s, where 1 indicates a pulse in the PPM slot and 0 indicates no pulse in the PPM slot? 2 Mbit/s: 00100010001000100010001000100010	16.2.4.4	IR5a:M	Yes ❑ No ❑ N/A ❑
IR6	Is the transmitted LENGTH field the correct PPM representation of the unsigned 16-bit binary integer, lsb transmitted first, equal to the correct value provided by the TXVECTOR parameter LENGTH?	16.2.4.5	M	Yes ❑
IR7	Is the transmitted CRC field the correct PPM representation of the CRC value calculated as per reference subclause, transmitted lsb first?	16.2.4.6	M	Yes ❑
IR8	Is the transmitted PSDU field the correct PPM representation of the PSDU, transmitted lsb first?	16.2.4.7	M	Yes ❑
IR9	When the CCA is false does transmission begin based on PHYTXSTART.request?	16.2.5.1	M	Yes ❑
IR10	Does the PHY issue a PHYTXSTART.confirm after the transmission of the PLCP header?	16.2.5.1	M	Yes ❑
IR11	Does the PHY accept each octet of the PSDU in a PHYDATA.request and answer with a PHYDATA.confirm?	16.2.5.1	M	Yes ❑
IR12	Does the PHY cease transmission in response to a PHYTXEND.request and answer with a PHYTXEND.confirm?	16.2.5.1	M	Yes ❑
IR13	Does the PHY of a receiving STA send a PHYCCA.indicate during reception of the SYNC field?	16.2.5.2	M	Yes ❑
IR14	Does the PHY of a receiving STA properly receive a transmission that changes data rate according to the DR field?	16.2.5.2	M	Yes ❑
IR15	Does the PHY of a receiving STA properly reject an incorrect CRC?	16.2.5.2	M	Yes ❑
IR16	Does the PHY of a receiving STA properly reject a DR field other than those specified in reference subclause?	16.2.5.2, 16.2.4.3	M	Yes ❑
IR17	Does the PHY of a receiving STA send PHYRXSTART.indicate with correct RATE and LENGTH parameters after proper reception of PLCP preamble and PLCP header?	16.2.5.2	M	Yes ❑
IR18	Does the PHY of a receiving STA forward receive octets in PHYDATA.indicate primitives?	16.2.5.2	M	Yes ❑
IR19	Does the PHY of a receiving STA send a PHYRXEND.indicate after the final octet indicated by the LENGTH field?	16.2.5.2	M	Yes ❑

A.4.7 Infrared baseband PHY functions *(continued)*

Item	Feature	References	Status	Support
IR20	Does the PHY of a receiving STA send a PHYCCA.indicate with a state value of IDLE after the PHYRXEND.indicate?	16.2.5.2	M	Yes ❑
IR21	Does the PHY reset its CCA detection mechanism upon receiving a PHYCCARST.request, and respond with a PHYCCARST.indicate?	16.2.5.3	M	Yes ❑
IR22	When transmitting at 1 Mbit/s does the PHY transmit PPM symbols according to the 16-PPM Basic Rate Mapping table, transmitting from left to right?	16.3.2.1, 16.3.2.2	M	Yes ❑
IR23	When transmitting at 2 Mbit/s does the PHY transmit PPM symbols according to the 4-PPM Enhanced Rate Mapping table, transmitting from left to right?	16.3.2.1, 16.3.2.2	IR5a:M	Yes ❑
IR24	Does the PHY operate over a temperature range of 0 °C to 40 °C?	16.3.2.4	M	Yes ❑
* IR25	If the unit is conformant to emitter radiation mask 1, is the peak optical power of an emitted pulse within the specification range averaged over the pulse width?	16.3.3.1	O.1	Yes ❑ No ❑ N/A ❑
* IR26	If the unit is conformant to emitter radiation mask 2, is the peak optical power of an emitted pulse within the specification range averaged over the pulse width?	16.3.3.1	O.1	Yes ❑ No ❑ N/A ❑
IR27	Does the transmitted pulse shape conform to the description of the reference subclause?	16.3.3.2	M	Yes ❑
IR28	Does the emitter radiation pattern as a function of angle conform to the requirements of the reference subclause as applicable based on conformance to emitter radiation mask 1?	16.3.3.3	IR25:M	Yes ❑ No ❑ N/A ❑
IR28a	Does the emitter radiation pattern as a function of angle conform to the requirements of the reference subclause as applicable based on conformance to emitter radiation mask 2?	16.3.3.3	IR26:M	Yes ❑ No ❑ N/A ❑
IR29	Is the peak emitter optical output as a function of wavelength in the range specified?	16.3.3.4	M	Yes ❑
IR30	Does the spectrum of the transmit signal amplitude as a voltage or current meet the requirements of the reference subclause?	16.3.3.5	M	Yes ❑
IR31	Does the receiver sensitivity meet the requirements of the reference subclause for receive signals of both 1 Mbit/s and 2 Mbit/s?	16.3.4.1	M	Yes ❑
IR32	Does the receiver exhibit a dynamic range as specified in reference subclause?	16.3.4.2	M	Yes ❑
IR33	Does the receiver field-of-view conform to the requirements of the reference subclause?	16.3.4.3	M	Yes ❑
IR34	When it is known that the conditions are such that the Carrier Detect Signal and the Energy Detect Signal are false is the CCA asserted IDLE?	16.3.5.1	M	Yes ❑

A.4.7 Infrared baseband PHY functions *(continued)*

Item	Feature	References	Status	Support
IR35	When the conditions are such that Energy Detect is true for greater than the time defined in reference subclause, does CCA become IDLE?	16.3.5.1	M	Yes ❑
IR36	When conditions are such that either Carrier Detect or Energy Detect go true, does CCA go BUSY?	16.3.5.1	M	Yes ❑
IR37	Are these compliance groups implemented? dot11PhyIRComplianceGroup, dot11PhyRegDomainsSupportGroup, and dot11PhyOperationComplianceGroup	16.4	M	Yes ❑

Annex B

(informative)

Hopping sequences

The following tables pertain to the hopping sequences for North America and ETSI.

Table B.1—Hopping sequence set 1

index	0	3	6	9	12	15	18	21	24	27	30	33	36
1	2	5	8	11	14	17	20	23	26	29	32	35	38
2	25	28	31	34	37	40	43	46	49	52	55	58	61
3	64	67	70	73	76	79	3	6	9	12	15	18	21
4	10	13	16	19	22	25	28	31	34	37	40	43	46
5	45	48	51	54	57	60	63	66	69	72	75	78	2
6	18	21	24	27	30	33	36	39	42	45	48	51	54
7	73	76	79	3	6	9	12	15	18	21	24	27	30
8	49	52	55	58	61	64	67	70	73	76	79	3	6
9	21	24	27	30	33	36	39	42	45	48	51	54	57
10	63	66	69	72	75	78	2	5	8	11	14	17	20
11	78	2	5	8	11	14	17	20	23	26	29	32	35
12	31	34	37	40	43	46	49	52	55	58	61	64	67
13	61	64	67	70	73	76	79	3	6	9	12	15	18
14	24	27	30	33	36	39	42	45	48	51	54	57	60
15	54	57	60	63	66	69	72	75	78	2	5	8	11
16	65	68	71	74	77	80	4	7	10	13	16	19	22
17	28	31	34	37	40	43	46	49	52	55	58	61	64
18	79	3	6	9	12	15	18	21	24	27	30	33	36
19	33	36	39	42	45	48	51	54	57	60	63	66	69
20	4	7	10	13	16	19	22	25	28	31	34	37	40
21	20	23	26	29	32	35	38	41	44	47	50	53	56
22	13	16	19	22	25	28	31	34	37	40	43	46	49
23	38	41	44	47	50	53	56	59	62	65	68	71	74
24	74	77	80	4	7	10	13	16	19	22	25	28	31
25	56	59	62	65	68	71	74	77	80	4	7	10	13
26	71	74	77	80	4	7	10	13	16	19	22	25	28
27	23	26	29	32	35	38	41	44	47	50	53	56	59
28	5	8	11	14	17	20	23	26	29	32	35	38	41
29	39	42	45	48	51	54	57	60	63	66	69	72	75
30	12	15	18	21	24	27	30	33	36	39	42	45	48
31	36	39	42	45	48	51	54	57	60	63	66	69	72
32	68	71	74	77	80	4	7	10	13	16	19	22	25
33	9	12	15	18	21	24	27	30	33	36	39	42	45
34	70	73	76	79	3	6	9	12	15	18	21	24	27
35	77	80	4	7	10	13	16	19	22	25	28	31	34
36	6	9	12	15	18	21	24	27	30	33	36	39	42
37	62	65	68	71	74	77	80	4	7	10	13	16	19
38	29	32	35	38	41	44	47	50	53	56	59	62	65
39	14	17	20	23	26	29	32	35	38	41	44	47	50

Table B.1—Hopping sequence set 1 *(continued)*

index	0	3	6	9	12	15	18	21	24	27	30	33	36
40	27	30	33	36	39	42	45	48	51	54	57	60	63
41	16	19	22	25	28	31	34	37	40	43	46	49	52
42	59	62	65	68	71	74	77	80	4	7	10	13	16
43	43	46	49	52	55	58	61	64	67	70	73	76	79
44	76	79	3	6	9	12	15	18	21	24	27	30	33
45	34	37	40	43	46	49	52	55	58	61	64	67	70
46	72	75	78	2	5	8	11	14	17	20	23	26	29
47	11	14	17	20	23	26	29	32	35	38	41	44	47
48	60	63	66	69	72	75	78	2	5	8	11	14	17
49	80	4	7	10	13	16	19	22	25	28	31	34	37
50	47	50	53	56	59	62	65	68	71	74	77	80	4
51	22	25	28	31	34	37	40	43	46	49	52	55	58
52	75	78	2	5	8	11	14	17	20	23	26	29	32
53	66	69	72	75	78	2	5	8	11	14	17	20	23
54	41	44	47	50	53	56	59	62	65	68	71	74	77
55	15	18	21	24	27	30	33	36	39	42	45	48	51
56	35	38	41	44	47	50	53	56	59	62	65	68	71
57	67	70	73	76	79	3	6	9	12	15	18	21	24
58	52	55	58	61	64	67	70	73	76	79	3	6	9
59	58	61	64	67	70	73	76	79	3	6	9	12	15
60	44	47	50	53	56	59	62	65	68	71	74	77	80
61	50	53	56	59	62	65	68	71	74	77	80	4	7
62	17	20	23	26	29	32	35	38	41	44	47	50	53
63	7	10	13	16	19	22	25	28	31	34	37	40	43
64	19	22	25	28	31	34	37	40	43	46	49	52	55
65	8	11	14	17	20	23	26	29	32	35	38	41	44
66	69	72	75	78	2	5	8	11	14	17	20	23	26
67	51	54	57	60	63	66	69	72	75	78	2	5	8
68	42	45	48	51	54	57	60	63	66	69	72	75	78
69	3	6	9	12	15	18	21	24	27	30	33	36	39
70	30	33	36	39	42	45	48	51	54	57	60	63	66
71	57	60	63	66	69	72	75	78	2	5	8	11	14
72	37	40	43	46	49	52	55	58	61	64	67	70	73
73	55	58	61	64	67	70	73	76	79	3	6	9	12
74	26	29	32	35	38	41	44	47	50	53	56	59	62
75	46	49	52	55	58	61	64	67	70	73	76	79	3
76	53	56	59	62	65	68	71	74	77	80	4	7	10
77	40	43	46	49	52	55	58	61	64	67	70	73	76
78	32	35	38	41	44	47	50	53	56	59	62	65	68
79	48	51	54	57	60	63	66	69	72	75	78	2	5

Table B.1—Hopping sequence set 1 *(continued)*

index	39	42	45	48	51	54	57	60	63	66	69	72	75
1	41	44	47	50	53	56	59	62	65	68	71	74	77
2	64	67	70	73	76	79	3	6	9	12	15	18	21
3	24	27	30	33	36	39	42	45	48	51	54	57	60
4	49	52	55	58	61	64	67	70	73	76	79	3	6
5	5	8	11	14	17	20	23	26	29	32	35	38	41
6	57	60	63	66	69	72	75	78	2	5	8	11	14
7	33	36	39	42	45	48	51	54	57	60	63	66	69
8	9	12	15	18	21	24	27	30	33	36	39	42	45
9	60	63	66	69	72	75	78	2	5	8	11	14	17
10	23	26	29	32	35	38	41	44	47	50	53	56	59
11	38	41	44	47	50	53	56	59	62	65	68	71	74
12	70	73	76	79	3	6	9	12	15	18	21	24	27
13	21	24	27	30	33	36	39	42	45	48	51	54	57
14	63	66	69	72	75	78	2	5	8	11	14	17	20
15	14	17	20	23	26	29	32	35	38	41	44	47	50
16	25	28	31	34	37	40	43	46	49	52	55	58	61
17	67	70	73	76	79	3	6	9	12	15	18	21	24
18	39	42	45	48	51	54	57	60	63	66	69	72	75
19	72	75	78	2	5	8	11	14	17	20	23	26	29
20	43	46	49	52	55	58	61	64	67	70	73	76	79
21	59	62	65	68	71	74	77	80	4	7	10	13	16
22	52	55	58	61	64	67	70	73	76	79	3	6	9
23	77	80	4	7	10	13	16	19	22	25	28	31	34
24	34	37	40	43	46	49	52	55	58	61	64	67	70
25	16	19	22	25	28	31	34	37	40	43	46	49	52
26	31	34	37	40	43	46	49	52	55	58	61	64	67
27	62	65	68	71	74	77	80	4	7	10	13	16	19
28	44	47	50	53	56	59	62	65	68	71	74	77	80
29	78	2	5	8	11	14	17	20	23	26	29	32	35
30	51	54	57	60	63	66	69	72	75	78	2	5	8
31	75	78	2	5	8	11	14	17	20	23	26	29	32
32	28	31	34	37	40	43	46	49	52	55	58	61	64
33	48	51	54	57	60	63	66	69	72	75	78	2	5
34	30	33	36	39	42	45	48	51	54	57	60	63	66
35	37	40	43	46	49	52	55	58	61	64	67	70	73
36	45	48	51	54	57	60	63	66	69	72	75	78	2
37	22	25	28	31	34	37	40	43	46	49	52	55	58
38	68	71	74	77	80	4	7	10	13	16	19	22	25
39	53	56	59	62	65	68	71	74	77	80	4	7	10

Table B.1—Hopping sequence set 1 *(continued)*

index	39	42	45	48	51	54	57	60	63	66	69	72	75
40	66	69	72	75	78	2	5	8	11	14	17	20	23
41	55	58	61	64	67	70	73	76	79	3	6	9	12
42	19	22	25	28	31	34	37	40	43	46	49	52	55
43	3	6	9	12	15	18	21	24	27	30	33	36	39
44	36	39	42	45	48	51	54	57	60	63	66	69	72
45	73	76	79	3	6	9	12	15	18	21	24	27	30
46	32	35	38	41	44	47	50	53	56	59	62	65	68
47	50	53	56	59	62	65	68	71	74	77	80	4	7
48	20	23	26	29	32	35	38	41	44	47	50	53	56
49	40	43	46	49	52	55	58	61	64	67	70	73	76
50	7	10	13	16	19	22	25	28	31	34	37	40	43
51	61	64	67	70	73	76	79	3	6	9	12	15	18
52	35	38	41	44	47	50	53	56	59	62	65	68	71
53	26	29	32	35	38	41	44	47	50	53	56	59	62
54	80	4	7	10	13	16	19	22	25	28	31	34	37
55	54	57	60	63	66	69	72	75	78	2	5	8	11
56	74	77	80	4	7	10	13	16	19	22	25	28	31
57	27	30	33	36	39	42	45	48	51	54	57	60	63
58	12	15	18	21	24	27	30	33	36	39	42	45	48
59	18	21	24	27	30	33	36	39	42	45	48	51	54
60	4	7	10	13	16	19	22	25	28	31	34	37	40
61	10	13	16	19	22	25	28	31	34	37	40	43	46
62	56	59	62	65	68	71	74	77	80	4	7	10	13
63	46	49	52	55	58	61	64	67	70	73	76	79	3
64	58	61	64	67	70	73	76	79	3	6	9	12	15
65	47	50	53	56	59	62	65	68	71	74	77	80	4
66	29	32	35	38	41	44	47	50	53	56	59	62	65
67	11	14	17	20	23	26	29	32	35	38	41	44	47
68	2	5	8	11	14	17	20	23	26	29	32	35	38
69	42	45	48	51	54	57	60	63	66	69	72	75	78
70	69	72	75	78	2	5	8	11	14	17	20	23	26
71	17	20	23	26	29	32	35	38	41	44	47	50	53
72	76	79	3	6	9	12	15	18	21	24	27	30	33
73	15	18	21	24	27	30	33	36	39	42	45	48	51
74	65	68	71	74	77	80	4	7	10	13	16	19	22
75	6	9	12	15	18	21	24	27	30	33	36	39	42
76	13	16	19	22	25	28	31	34	37	40	43	46	49
77	79	3	6	9	12	15	18	21	24	27	30	33	36
78	71	74	77	80	4	7	10	13	16	19	22	25	28
79	8	11	14	17	20	23	26	29	32	35	38	41	44

Table B.2—Hopping sequence set 2

index	1	4	7	10	13	16	19	22	25	28	31	34	37
1	3	6	9	12	15	18	21	24	27	30	33	36	39
2	26	29	32	35	38	41	44	47	50	53	56	59	62
3	65	68	71	74	77	80	4	7	10	13	16	19	22
4	11	14	17	20	23	26	29	32	35	38	41	44	47
5	46	49	52	55	58	61	64	67	70	73	76	79	3
6	19	22	25	28	31	34	37	40	43	46	49	52	55
7	74	77	80	4	7	10	13	16	19	22	25	28	31
8	50	53	56	59	62	65	68	71	74	77	80	4	7
9	22	25	28	31	34	37	40	43	46	49	52	55	58
10	64	67	70	73	76	79	3	6	9	12	15	18	21
11	79	3	6	9	12	15	18	21	24	27	30	33	36
12	32	35	38	41	44	47	50	53	56	59	62	65	68
13	62	65	68	71	74	77	80	4	7	10	13	16	19
14	25	28	31	34	37	40	43	46	49	52	55	58	61
15	55	58	61	64	67	70	73	76	79	3	6	9	12
16	66	69	72	75	78	2	5	8	11	14	17	20	23
17	29	32	35	38	41	44	47	50	53	56	59	62	65
18	80	4	7	10	13	16	19	22	25	28	31	34	37
19	34	37	40	43	46	49	52	55	58	61	64	67	70
20	5	8	11	14	17	20	23	26	29	32	35	38	41
21	21	24	27	30	33	36	39	42	45	48	51	54	57
22	14	17	20	23	26	29	32	35	38	41	44	47	50
23	39	42	45	48	51	54	57	60	63	66	69	72	75
24	75	78	2	5	8	11	14	17	20	23	26	29	32
25	57	60	63	66	69	72	75	78	2	5	8	11	14
26	72	75	78	2	5	8	11	14	17	20	23	26	29
27	24	27	30	33	36	39	42	45	48	51	54	57	60
28	6	9	12	15	18	21	24	27	30	33	36	39	42
29	40	43	46	49	52	55	58	61	64	67	70	73	76
30	13	16	19	22	25	28	31	34	37	40	43	46	49
31	37	40	43	46	49	52	55	58	61	64	67	70	73
32	69	72	75	78	2	5	8	11	14	17	20	23	26
33	10	13	16	19	22	25	28	31	34	37	40	43	46
34	71	74	77	80	4	7	10	13	16	19	22	25	28
35	78	2	5	8	11	14	17	20	23	26	29	32	35
36	7	10	13	16	19	22	25	28	31	34	37	40	43
37	63	66	69	72	75	78	2	5	8	11	14	17	20
38	30	33	36	39	42	45	48	51	54	57	60	63	66
39	15	18	21	24	27	30	33	36	39	42	45	48	51

Table B.2—Hopping sequence set 2 *(continued)*

index	1	4	7	10	13	16	19	22	25	28	31	34	37
40	28	31	34	37	40	43	46	49	52	55	58	61	64
41	17	20	23	26	29	32	35	38	41	44	47	50	53
42	60	63	66	69	72	75	78	2	5	8	11	14	17
43	44	47	50	53	56	59	62	65	68	71	74	77	80
44	77	80	4	7	10	13	16	19	22	25	28	31	34
45	35	38	41	44	47	50	53	56	59	62	65	68	71
46	73	76	79	3	6	9	12	15	18	21	24	27	30
47	12	15	18	21	24	27	30	33	36	39	42	45	48
48	61	64	67	70	73	76	79	3	6	9	12	15	18
49	2	5	8	11	14	17	20	23	26	29	32	35	38
50	48	51	54	57	60	63	66	69	72	75	78	2	5
51	23	26	29	32	35	38	41	44	47	50	53	56	59
52	76	79	3	6	9	12	15	18	21	24	27	30	33
53	67	70	73	76	79	3	6	9	12	15	18	21	24
54	42	45	48	51	54	57	60	63	66	69	72	75	78
55	16	19	22	25	28	31	34	37	40	43	46	49	52
56	36	39	42	45	48	51	54	57	60	63	66	69	72
57	68	71	74	77	80	4	7	10	13	16	19	22	25
58	53	56	59	62	65	68	71	74	77	80	4	7	10
59	59	62	65	68	71	74	77	80	4	7	10	13	16
60	45	48	51	54	57	60	63	66	69	72	75	78	2
61	51	54	57	60	63	66	69	72	75	78	2	5	8
62	18	21	24	27	30	33	36	39	42	45	48	51	54
63	8	11	14	17	20	23	26	29	32	35	38	41	44
64	20	23	26	29	32	35	38	41	44	47	50	53	56
65	9	12	15	18	21	24	27	30	33	36	39	42	45
66	70	73	76	79	3	6	9	12	15	18	21	24	27
67	52	55	58	61	64	67	70	73	76	79	3	6	9
68	43	46	49	52	55	58	61	64	67	70	73	76	79
69	4	7	10	13	16	19	22	25	28	31	34	37	40
70	31	34	37	40	43	46	49	52	55	58	61	64	67
71	58	61	64	67	70	73	76	79	3	6	9	12	15
72	38	41	44	47	50	53	56	59	62	65	68	71	74
73	56	59	62	65	68	71	74	77	80	4	7	10	13
74	27	30	33	36	39	42	45	48	51	54	57	60	63
75	47	50	53	56	59	62	65	68	71	74	77	80	4
76	54	57	60	63	66	69	72	75	78	2	5	8	11
77	41	44	47	50	53	56	59	62	65	68	71	74	77
78	33	36	39	42	45	48	51	54	57	60	63	66	69
79	49	52	55	58	61	64	67	70	73	76	79	3	6

Table B.2—Hopping sequence set 2 *(continued)*

index	40	43	46	49	52	55	58	61	64	67	70	73	76
1	42	45	48	51	54	57	60	63	66	69	72	75	78
2	65	68	71	74	77	80	4	7	10	13	16	19	22
3	25	28	31	34	37	40	43	46	49	52	55	58	61
4	50	53	56	59	62	65	68	71	74	77	80	4	7
5	6	9	12	15	18	21	24	27	30	33	36	39	42
6	58	61	64	67	70	73	76	79	3	6	9	12	15
7	34	37	40	43	46	49	52	55	58	61	64	67	70
8	10	13	16	19	22	25	28	31	34	37	40	43	46
9	61	64	67	70	73	76	79	3	6	9	12	15	18
10	24	27	30	33	36	39	42	45	48	51	54	57	60
11	39	42	45	48	51	54	57	60	63	66	69	72	75
12	71	74	77	80	4	7	10	13	16	19	22	25	28
13	22	25	28	31	34	37	40	43	46	49	52	55	58
14	64	67	70	73	76	79	3	6	9	12	15	18	21
15	15	18	21	24	27	30	33	36	39	42	45	48	51
16	26	29	32	35	38	41	44	47	50	53	56	59	62
17	68	71	74	77	80	4	7	10	13	16	19	22	25
18	40	43	46	49	52	55	58	61	64	67	70	73	76
19	73	76	79	3	6	9	12	15	18	21	24	27	30
20	44	47	50	53	56	59	62	65	68	71	74	77	80
21	60	63	66	69	72	75	78	2	5	8	11	14	17
22	53	56	59	62	65	68	71	74	77	80	4	7	10
23	78	2	5	8	11	14	17	20	23	26	29	32	35
24	35	38	41	44	47	50	53	56	59	62	65	68	71
25	17	20	23	26	29	32	35	38	41	44	47	50	53
26	32	35	38	41	44	47	50	53	56	59	62	65	68
27	63	66	69	72	75	78	2	5	8	11	14	17	20
28	45	48	51	54	57	60	63	66	69	72	75	78	2
29	79	3	6	9	12	15	18	21	24	27	30	33	36
30	52	55	58	61	64	67	70	73	76	79	3	6	9
31	76	79	3	6	9	12	15	18	21	24	27	30	33
32	29	32	35	38	41	44	47	50	53	56	59	62	65
33	49	52	55	58	61	64	67	70	73	76	79	3	6
34	31	34	37	40	43	46	49	52	55	58	61	64	67
35	38	41	44	47	50	53	56	59	62	65	68	71	74
36	46	49	52	55	58	61	64	67	70	73	76	79	3
37	23	26	29	32	35	38	41	44	47	50	53	56	59
38	69	72	75	78	2	5	8	11	14	17	20	23	26
39	54	57	60	63	66	69	72	75	78	2	5	8	11

Table B.2—Hopping sequence set 2 *(continued)*

index	40	43	46	49	52	55	58	61	64	67	70	73	76
40	67	70	73	76	79	3	6	9	12	15	18	21	24
41	56	59	62	65	68	71	74	77	80	4	7	10	13
42	20	23	26	29	32	35	38	41	44	47	50	53	56
43	4	7	10	13	16	19	22	25	28	31	34	37	40
44	37	40	43	46	49	52	55	58	61	64	67	70	73
45	74	77	80	4	7	10	13	16	19	22	25	28	31
46	33	36	39	42	45	48	51	54	57	60	63	66	69
47	51	54	57	60	63	66	69	72	75	78	2	5	8
48	21	24	27	30	33	36	39	42	45	48	51	54	57
49	41	44	47	50	53	56	59	62	65	68	71	74	77
50	8	11	14	17	20	23	26	29	32	35	38	41	44
51	62	65	68	71	74	77	80	4	7	10	13	16	19
52	36	39	42	45	48	51	54	57	60	63	66	69	72
53	27	30	33	36	39	42	45	48	51	54	57	60	63
54	2	5	8	11	14	17	20	23	26	29	32	35	38
55	55	58	61	64	67	70	73	76	79	3	6	9	12
56	75	78	2	5	8	11	14	17	20	23	26	29	32
57	28	31	34	37	40	43	46	49	52	55	58	61	64
58	13	16	19	22	25	28	31	34	37	40	43	46	49
59	19	22	25	28	31	34	37	40	43	46	49	52	55
60	5	8	11	14	17	20	23	26	29	32	35	38	41
61	11	14	17	20	23	26	29	32	35	38	41	44	47
62	57	60	63	66	69	72	75	78	2	5	8	11	14
63	47	50	53	56	59	62	65	68	71	74	77	80	4
64	59	62	65	68	71	74	77	80	4	7	10	13	16
65	48	51	54	57	60	63	66	69	72	75	78	2	5
66	30	33	36	39	42	45	48	51	54	57	60	63	66
67	12	15	18	21	24	27	30	33	36	39	42	45	48
68	3	6	9	12	15	18	21	24	27	30	33	36	39
69	43	46	49	52	55	58	61	64	67	70	73	76	79
70	70	73	76	79	3	6	9	12	15	18	21	24	27
71	18	21	24	27	30	33	36	39	42	45	48	51	54
72	77	80	4	7	10	13	16	19	22	25	28	31	34
73	16	19	22	25	28	31	34	37	40	43	46	49	52
74	66	69	72	75	78	2	5	8	11	14	17	20	23
75	7	10	13	16	19	22	25	28	31	34	37	40	43
76	14	17	20	23	26	29	32	35	38	41	44	47	50
77	80	4	7	10	13	16	19	22	25	28	31	34	37
78	72	75	78	2	5	8	11	14	17	20	23	26	29
79	9	12	15	18	21	24	27	30	33	36	39	42	45

Table B.3—Hopping sequence set 3

index	2	5	8	11	14	17	20	23	26	29	32	35	38
1	4	7	10	13	16	19	22	25	28	31	34	37	40
2	27	30	33	36	39	42	45	48	51	54	57	60	63
3	66	69	72	75	78	2	5	8	11	14	17	20	23
4	12	15	18	21	24	27	30	33	36	39	42	45	48
5	47	50	53	56	59	62	65	68	71	74	77	80	4
6	20	23	26	29	32	35	38	41	44	47	50	53	56
7	75	78	2	5	8	11	14	17	20	23	26	29	32
8	51	54	57	60	63	66	69	72	75	78	2	5	8
9	23	26	29	32	35	38	41	44	47	50	53	56	59
10	65	68	71	74	77	80	4	7	10	13	16	19	22
11	80	4	7	10	13	16	19	22	25	28	31	34	37
12	33	36	39	42	45	48	51	54	57	60	63	66	69
13	63	66	69	72	75	78	2	5	8	11	14	17	20
14	26	29	32	35	38	41	44	47	50	53	56	59	62
15	56	59	62	65	68	71	74	77	80	4	7	10	13
16	67	70	73	76	79	3	6	9	12	15	18	21	24
17	30	33	36	39	42	45	48	51	54	57	60	63	66
18	2	5	8	11	14	17	20	23	26	29	32	35	38
19	35	38	41	44	47	50	53	56	59	62	65	68	71
20	6	9	12	15	18	21	24	27	30	33	36	39	42
21	22	25	28	31	34	37	40	43	46	49	52	55	58
22	15	18	21	24	27	30	33	36	39	42	45	48	51
23	40	43	46	49	52	55	58	61	64	67	70	73	76
24	76	79	3	6	9	12	15	18	21	24	27	30	33
25	58	61	64	67	70	73	76	79	3	6	9	12	15
26	73	76	79	3	6	9	12	15	18	21	24	27	30
27	25	28	31	34	37	40	43	46	49	52	55	58	61
28	7	10	13	16	19	22	25	28	31	34	37	40	43
29	41	44	47	50	53	56	59	62	65	68	71	74	77
30	14	17	20	23	26	29	32	35	38	41	44	47	50
31	38	41	44	47	50	53	56	59	62	65	68	71	74
32	70	73	76	79	3	6	9	12	15	18	21	24	27
33	11	14	17	20	23	26	29	32	35	38	41	44	47
34	72	75	78	2	5	8	11	14	17	20	23	26	29
35	79	3	6	9	12	15	18	21	24	27	30	33	36
36	8	11	14	17	20	23	26	29	32	35	38	41	44
37	64	67	70	73	76	79	3	6	9	12	15	18	21
38	31	34	37	40	43	46	49	52	55	58	61	64	67
39	16	19	22	25	28	31	34	37	40	43	46	49	52

Table B.3—Hopping sequence set 3 *(continued)*

index	2	5	8	11	14	17	20	23	26	29	32	35	38
40	29	32	35	38	41	44	47	50	53	56	59	62	65
41	18	21	24	27	30	33	36	39	42	45	48	51	54
42	61	64	67	70	73	76	79	3	6	9	12	15	18
43	45	48	51	54	57	60	63	66	69	72	75	78	2
44	78	2	5	8	11	14	17	20	23	26	29	32	35
45	36	39	42	45	48	51	54	57	60	63	66	69	72
46	74	77	80	4	7	10	13	16	19	22	25	28	31
47	13	16	19	22	25	28	31	34	37	40	43	46	49
48	62	65	68	71	74	77	80	4	7	10	13	16	19
49	3	6	9	12	15	18	21	24	27	30	33	36	39
50	49	52	55	58	61	64	67	70	73	76	79	3	6
51	24	27	30	33	36	39	42	45	48	51	54	57	60
52	77	80	4	7	10	13	16	19	22	25	28	31	34
53	68	71	74	77	80	4	7	10	13	16	19	22	25
54	43	46	49	52	55	58	61	64	67	70	73	76	79
55	17	20	23	26	29	32	35	38	41	44	47	50	53
56	37	40	43	46	49	52	55	58	61	64	67	70	73
57	69	72	75	78	2	5	8	11	14	17	20	23	26
58	54	57	60	63	66	69	72	75	78	2	5	8	11
59	60	63	66	69	72	75	78	2	5	8	11	14	17
60	46	49	52	55	58	61	64	67	70	73	76	79	3
61	52	55	58	61	64	67	70	73	76	79	3	6	9
62	19	22	25	28	31	34	37	40	43	46	49	52	55
63	9	12	15	18	21	24	27	30	33	36	39	42	45
64	21	24	27	30	33	36	39	42	45	48	51	54	57
65	10	13	16	19	22	25	28	31	34	37	40	43	46
66	71	74	77	80	4	7	10	13	16	19	22	25	28
67	53	56	59	62	65	68	71	74	77	80	4	7	10
68	44	47	50	53	56	59	62	65	68	71	74	77	80
69	5	8	11	14	17	20	23	26	29	32	35	38	41
70	32	35	38	41	44	47	50	53	56	59	62	65	68
71	59	62	65	68	71	74	77	80	4	7	10	13	16
72	39	42	45	48	51	54	57	60	63	66	69	72	75
73	57	60	63	66	69	72	75	78	2	5	8	11	14
74	28	31	34	37	40	43	46	49	52	55	58	61	64
75	48	51	54	57	60	63	66	69	72	75	78	2	5
76	55	58	61	64	67	70	73	76	79	3	6	9	12
77	42	45	48	51	54	57	60	63	66	69	72	75	78
78	34	37	40	43	46	49	52	55	58	61	64	67	70
79	50	53	56	59	62	65	68	71	74	77	80	4	7

Table B.3—Hopping sequence set 3 *(continued)*

index	41	44	47	50	53	56	59	62	65	68	71	74	77
1	43	46	49	52	55	58	61	64	67	70	73	76	79
2	66	69	72	75	78	2	5	8	11	14	17	20	23
3	26	29	32	35	38	41	44	47	50	53	56	59	62
4	51	54	57	60	63	66	69	72	75	78	2	5	8
5	7	10	13	16	19	22	25	28	31	34	37	40	43
6	59	62	65	68	71	74	77	80	4	7	10	13	16
7	35	38	41	44	47	50	53	56	59	62	65	68	71
8	11	14	17	20	23	26	29	32	35	38	41	44	47
9	62	65	68	71	74	77	80	4	7	10	13	16	19
10	25	28	31	34	37	40	43	46	49	52	55	58	61
11	40	43	46	49	52	55	58	61	64	67	70	73	76
12	72	75	78	2	5	8	11	14	17	20	23	26	29
13	23	26	29	32	35	38	41	44	47	50	53	56	59
14	65	68	71	74	77	80	4	7	10	13	16	19	22
15	16	19	22	25	28	31	34	37	40	43	46	49	52
16	27	30	33	36	39	42	45	48	51	54	57	60	63
17	69	72	75	78	2	5	8	11	14	17	20	23	26
18	41	44	47	50	53	56	59	62	65	68	71	74	77
19	74	77	80	4	7	10	13	16	19	22	25	28	31
20	45	48	51	54	57	60	63	66	69	72	75	78	2
21	61	64	67	70	73	76	79	3	6	9	12	15	18
22	54	57	60	63	66	69	72	75	78	2	5	8	11
23	79	3	6	9	12	15	18	21	24	27	30	33	36
24	36	39	42	45	48	51	54	57	60	63	66	69	72
25	18	21	24	27	30	33	36	39	42	45	48	51	54
26	33	36	39	42	45	48	51	54	57	60	63	66	69
27	64	67	70	73	76	79	3	6	9	12	15	18	21
28	46	49	52	55	58	61	64	67	70	73	76	79	3
29	80	4	7	10	13	16	19	22	25	28	31	34	37
30	53	56	59	62	65	68	71	74	77	80	4	7	10
31	77	80	4	7	10	13	16	19	22	25	28	31	34
32	30	33	36	39	42	45	48	51	54	57	60	63	66
33	50	53	56	59	62	65	68	71	74	77	80	4	7
34	32	35	38	41	44	47	50	53	56	59	62	65	68
35	39	42	45	48	51	54	57	60	63	66	69	72	75
36	47	50	53	56	59	62	65	68	71	74	77	80	4
37	24	27	30	33	36	39	42	45	48	51	54	57	60
38	70	73	76	79	3	6	9	12	15	18	21	24	27
39	55	58	61	64	67	70	73	76	79	3	6	9	12

Table B.3—Hopping sequence set 3 *(continued)*

index	41	44	47	50	53	56	59	62	65	68	71	74	77
40	68	71	74	77	80	4	7	10	13	16	19	22	25
41	57	60	63	66	69	72	75	78	2	5	8	11	14
42	21	24	27	30	33	36	39	42	45	48	51	54	57
43	5	8	11	14	17	20	23	26	29	32	35	38	41
44	38	41	44	47	50	53	56	59	62	65	68	71	74
45	75	78	2	5	8	11	14	17	20	23	26	29	32
46	34	37	40	43	46	49	52	55	58	61	64	67	70
47	52	55	58	61	64	67	70	73	76	79	3	6	9
48	22	25	28	31	34	37	40	43	46	49	52	55	58
49	42	45	48	51	54	57	60	63	66	69	72	75	78
50	9	12	15	18	21	24	27	30	33	36	39	42	45
51	63	66	69	72	75	78	2	5	8	11	14	17	20
52	37	40	43	46	49	52	55	58	61	64	67	70	73
53	28	31	34	37	40	43	46	49	52	55	58	61	64
54	3	6	9	12	15	18	21	24	27	30	33	36	39
55	56	59	62	65	68	71	74	77	80	4	7	10	13
56	76	79	3	6	9	12	15	18	21	24	27	30	33
57	29	32	35	38	41	44	47	50	53	56	59	62	65
58	14	17	20	23	26	29	32	35	38	41	44	47	50
59	20	23	26	29	32	35	38	41	44	47	50	53	56
60	6	9	12	15	18	21	24	27	30	33	36	39	42
61	12	15	18	21	24	27	30	33	36	39	42	45	48
62	58	61	64	67	70	73	76	79	3	6	9	12	15
63	48	51	54	57	60	63	66	69	72	75	78	2	5
64	60	63	66	69	72	75	78	2	5	8	11	14	17
65	49	52	55	58	61	64	67	70	73	76	79	3	6
66	31	34	37	40	43	46	49	52	55	58	61	64	67
67	13	16	19	22	25	28	31	34	37	40	43	46	49
68	4	7	10	13	16	19	22	25	28	31	34	37	40
69	44	47	50	53	56	59	62	65	68	71	74	77	80
70	71	74	77	80	4	7	10	13	16	19	22	25	28
71	19	22	25	28	31	34	37	40	43	46	49	52	55
72	78	2	5	8	11	14	17	20	23	26	29	32	35
73	17	20	23	26	29	32	35	38	41	44	47	50	53
74	67	70	73	76	79	3	6	9	12	15	18	21	24
75	8	11	14	17	20	23	26	29	32	35	38	41	44
76	15	18	21	24	27	30	33	36	39	42	45	48	51
77	2	5	8	11	14	17	20	23	26	29	32	35	38
78	73	76	79	3	6	9	12	15	18	21	24	27	30
79	10	13	16	19	22	25	28	31	34	37	40	43	46

Annex C

(normative)

Formal description of MAC operation

This annex contains formal descriptions of the behavior of MAC station (STA) and access point (AP) entities. These descriptions also describe the frame formats and the generation and interpretation of information encoded in MAC frames, in the parameters of service primitives supported by the MAC, and in MIB attributes used or generated by the MAC. The MAC is described using the 1992 version of the ITU Specification and Description Language (SDL-92). SDL-92 is defined in ITU-T Recommendation Z.100 (03/93). An update to Z.100 was approved in 1996 (SDL-96), but none of the SDL facilities used in this annex were modified. An introduction to the MAC formal description is provided in Clause C.1. Definitions of the data types and operators used by the MAC state machines are provided in Clause C.2. An SDL system describing MAC operation at an IEEE 802.11 station is contained in Clause C.3. Finally, a subset of an SDL system describing the aspects of MAC operation at an IEEE 802.11 AP that differ from operation at a non-AP station is provided in Clause C.4.

In Annex D, the MAC and PHY management information bases are described in Abstract Syntax Notation One (ASN.1), defined in ISO/IEC 8824: 1990 and ISO/IEC 8825: 1990. ITU-T Recommendation Z.105 (03/95) defines the use of SDL in conjunction with ASN.1, allowing system behavior to be defined using SDL and data types to be defined using ASN.1. Incomplete tool support precluded the use of ITU-T Recommendation Z.105 in this annex. However, within the limits of ITU-T Recommendation Z.100 (referred to subsequently as Z.100), the data types in Clause C.2 are defined in a similar manner to ITU-T Recommendation Z.105 (referred to subsequently as Z.105). Annex E contains a listing of available documentation.

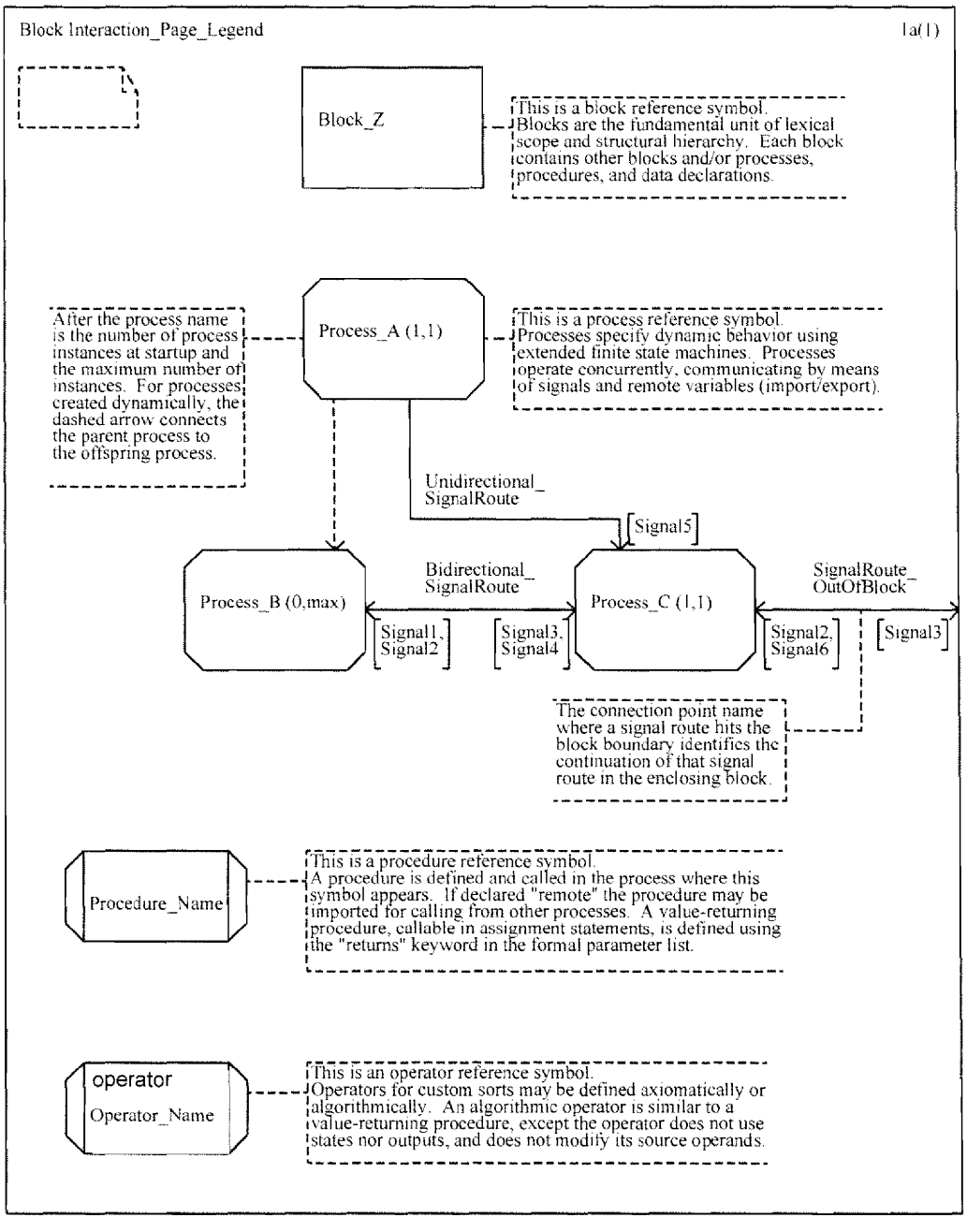
NOTES

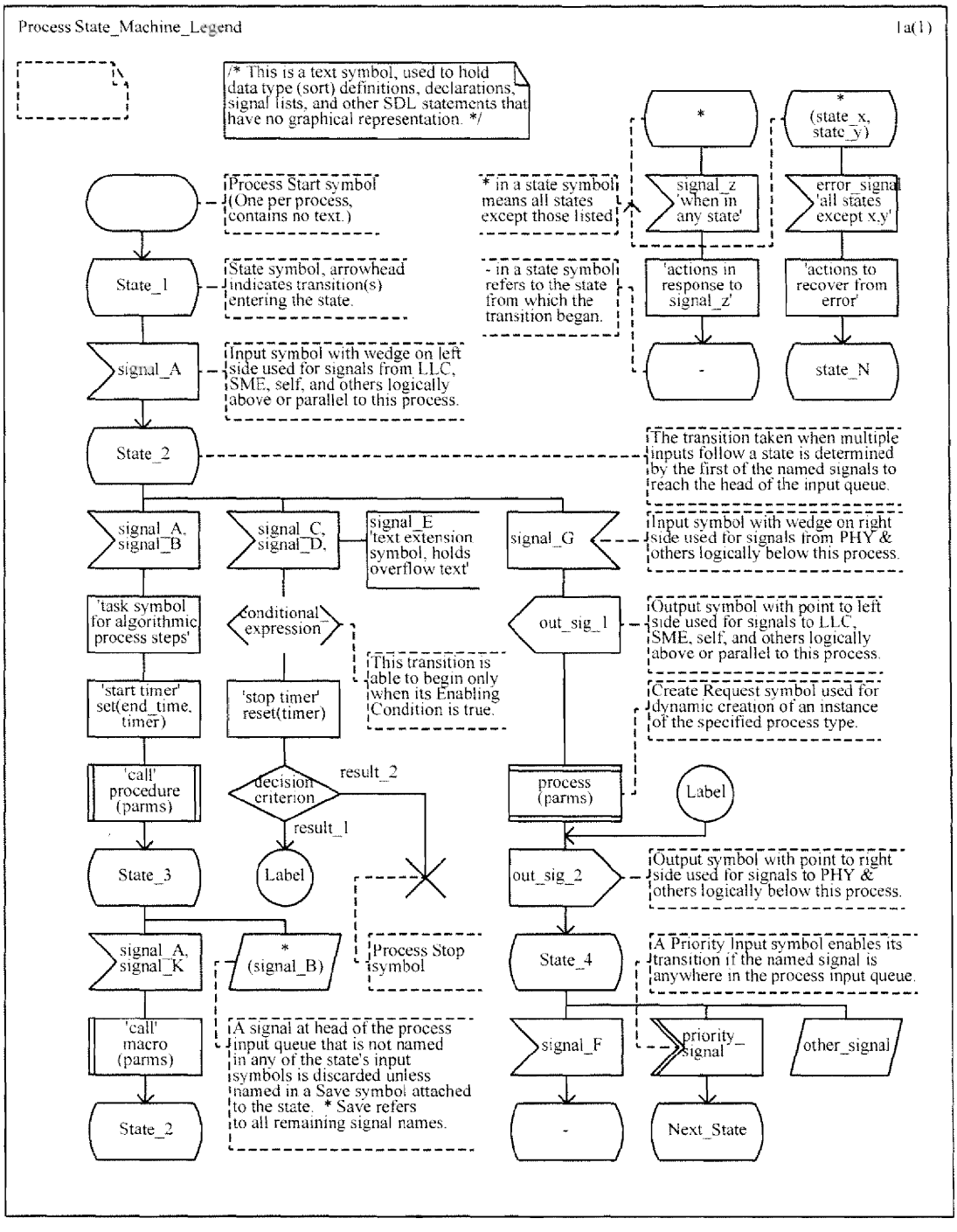
1—The SDL definitions in this annex should be usable with any SDL tool that supports the 1993 version or 1996 update of ITU-T Recommendation Z.100. Software for generating, analyzing, verifying, and simulating SDL system descriptions is available from several sources.

2—The SDL code in this annex was generated using *SDT/PC version 3.02*; from Telelogic AB, Malmo, Sweden (+46-40-174700; internet: telelogic.se); USA office in Princeton, NJ (+1-609-520-1935; internet: telelogic.com). Telelogic offers SDT for several workstation platforms in addition to SDT/PC.

3— The use of Telelogic's product to prepare this annex does not constitute an endorsement of SDT by the IEEE LAN MAN Standards Committee or by the IEEE.

4—The diagrams on the next two pages show most of the symbols of SDL graphical syntax (SDL-GR) used in the MAC formal description. The symbols in these diagrams have labels and comments that explain their meanings. These diagrams are intended to serve as a legend for the SDL-GR symbols that comprise most of the process interaction and state transition diagrams. These diagrams are neither a complete SDL system, nor a complete presentation of SDL-GR symbology. Also, this state machine fragment exists to illustrate the SDL graphical syntax, and does not describe any useful behavior.





C.1 Introduction to the MAC formal description

This formal description defines the behavior of IEEE 802.11 MAC entities. The MAC protocol functional decomposition used herein facilitates explicit description of the reference points and durations of the various timed intervals; the bases for generation and/or validation of header fields, service parameters, and MIB attributes; and the interpretation of each value in cases where enumerated data types are used in service parameters.

C.1.1 Fundamental assumptions

The MAC protocol is described as an SDL system, which is a set of extended finite state machines. Each state machine is a set of independent processes, all of which operate concurrently. All variable data-holding entities and procedures exist solely within the context of a single process. In SDL all interprocess communication is done with signals (there are no global variables). Signals may be sent and received explicitly, using SDL's output and input symbols, or implicitly, using SDL's export/import mechanism (only if the variables or procedures are declared "remote"). By default, signals incur delays when traversing channels between blocks; however, only nondelaying channels and signal routes are used in the MAC state machines, and all remote variables and procedures are declared with the "nodelay" property.

State transitions, procedure calls, and tasks (assignment statements and other algorithmic processing steps) are assumed to require zero time. This permits the time intervals that are part of the normative MAC behavior to be defined explicitly, using SDL timers. One unit of system time (a 1.0 change in the value of "now") is assumed to represent one microsecond of real time. Usec (microsecond) and TU (time unit) data types are defined, with operators to convert Usec and TU values to SDL time or duration when necessary.

The SDL system boundary encloses the MAC entities. The LLC, SME, PHY, and distribution system are part of the environment. SDL generally assumes that entities in the environment operate as specified; however, the MAC state machines that communicate with the various SAPs attempt to validate inputs from the environment, and to handle cases where a pair of communicating entities, one within the system and the other outside the system boundary, have different local views of the medium, station, or service state. All stations in an IEEE 802.11 service set are assumed to exhibit the behaviors described herein. Nevertheless, because of the open nature of the wireless medium, the MAC state machines check for error cases that can arise only when an entity on the wireless medium is transmitting IEEE 802.11 PDUs, but is not obeying the communication protocols specified by this standard.

C.1.2 Notation conventions

When practical, names used in the clauses of this standard are spelled identically in this annex. The principal exceptions are those names that conflict with one of SDL's reserved words (such as power management mode "active," which is renamed "sta_active" in SDL). To help fit the SDL text into the graphic symbols, acronyms with multiple, sequential capital letters are written with only the first letter capitalized (e.g., "MSDU" is written "Msdu" and "MLMEJoin.request" is written "MlmeJoin.request").

SDL reserved words and the names of variables and synonyms (named constants) begin with lowercase letters. The names of sorts (data types), signals, signal routes, channels, blocks, and processes begin with uppercase letters. The names of certain groups of variables and/or synonyms begin with a particular lowercase letter, followed by the remainder of the name, beginning with an uppercase letter. These groups are

"aNameOfAttribute"	PHY operational parameters.
"cNameOfCapability"	Capability bits, also used for internal values exported as MIB counters.
"dNameOfDuration"	Duration (relative time) values, declared as Usec, TU, or Duration.
"dot11NameOfAttribute"	MIB attributes.

"eNameOfElement"	Element ID values.
"mNameOfVariable"	Remote variables used for intra-MAC communication, but not part of the MIB. Most of these variables are exported from the MLME block.
"sNameOfStaticValue"	Synonyms for static data values used within the MAC.
"tNameOfTime"	Time (absolute time) values, declared as Usec, TU, or Time. The names of timers begin with "T."

C.1.3 Modeling techniques

State machines are grouped according to defined function sets that are visible, directly or indirectly, at an exposed interface. The emphasis in the organization of the state machines is explicitly to show initiation of and response to events at the exposed interfaces, and time-related actions, including those dependent on the absence of external events (e.g., response timeouts) and intervals measured in derived units (e.g., backoff "time" in units of slots during which the wireless medium is idle). The operations associated with the various state transitions emphasize communication functions. Most of the details regarding insertion, extraction, and encoding of information in fields of the PDUs is encapsulated with the definitions of those fields. This approach, which relies heavily on SDL's abstract data type and inheritance mechanisms, permits the behavior of the data-holding entities to be precisely defined, without obscuring process flow by adding in-line complexity to the individual state transitions.

The modeling of PDUs and SDUs requires sorts such as octet strings, and operators such as bitwise boolean functions, which are not predefined in SDL. These sorts and operators are defined in Package macsorts, which appears in Clause C.2.

Protocol and service data unit sorts are based on the **Bit** sort. Bit is a subtype of SDL's predefined Boolean sort. As a result, Bit literals "0" and "1" are alternative names for "false" and "true," and have no numeric significance. To use "0" or "1" as integer values requires a conversion operation. Items of the **Bitstring** sort are 0-origin, variable-length strings of Bits. With Bitstring operands, operators "and," "or," "xor," and "not" operate bitwise, with the length of the result equal to the length of the longest (or only) source string. The **Octet** sort is a subtype of Bitstring that adds conversion operators to and from Integer. Each item of the Octet sort has length=8 {by usage convention in Z.100, enforced in Z.105}. Items of the **Octetstring** sort are 0-origin, variable-length strings of Octets. The **Frame** sort is a subtype of Octetstring that adds operators to extract and to modify all MAC header fields and most other MAC frame fields and elements. Most MAC fields and elements that contain named values with specific value assignments or enumerations are defined as subtypes of Frame, Octetstring, or Bitstring with the names added as literals or synonyms, so that the state machines can refer to the names without introducing ambiguity about the value encodings.

Where communication at a SAP or between processes is strictly first in first out (FIFO), the (implicit) input queue of the SDL processes is used. When more sophisticated queue management is needed, a queue whose entries are instances of one, specified sort is created using the **Queue** generator. Entries on Queue sorts may be added and removed at either the tail or the head, and the number of queue entries may be determined. The contents of a Queue may also be searched to locate entries with particular parameter values.

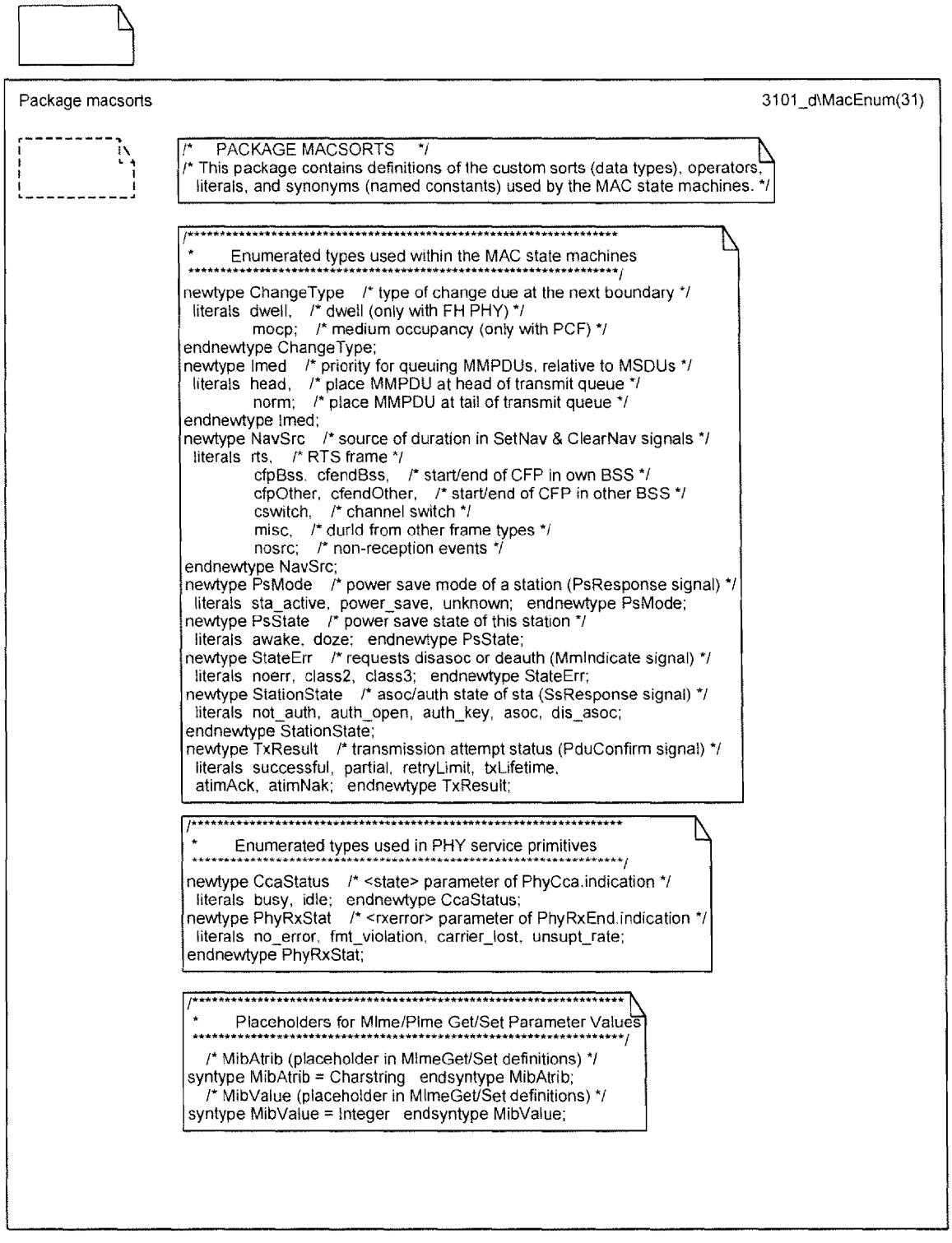
Clause C.2 contains an SDL-92 Package (a named collection of SDL definitions that can be included by reference into an SDL System specification), which is a formal description of the formats and data encodings used in IEEE 802.11 SDUs, PDUs, and the parameters of the service primitives used at each of the SAPs supported by the IEEE 802.11 MAC. This package also contains definitions for some data structures and operators used internally by one or more of the MAC state machines.

The behaviors of many intra-MAC operators are part of the normative description of the MAC protocol because results of the specified operations are visible, directly or indirectly, at exposed interfaces. For example, custom operators are used to define the generation of the CRC-32 value used in the FCS field (operator crc32, page 301), the calculation of frame transmission time used as part of the value in the Duration/ID field

in certain types of frames (operator calcDur, page 316), the comparison of the values of particular fields of a received MAC header with cached data values as part of the procedure for detecting duplicate frames (operator searchTupleCache, page 289), and numerous other aspects of frame formats and information encoding. On the other hand, data structures used solely for intra-MAC storage or for transferring of information between different state machines of a single station or access point, are only normative to the extent that they define items of internal state and the temporal sequence necessary for proper operation of the MAC protocol. The specific structures and encodings used for internal data storage and communication functions in this formal description do *not* constrain MAC implementations, provided those implementations exhibit the specified behaviors at the defined SAPs and, in conjunction with an appropriate PHY, on the wireless medium.

C.2 Data type and operator definitions for the MAC state machines

This clause is in SDL/PR (phrase notation), with the exception of procedural operators, which are defined in SDL/GR (graphic notation). Package macsorts contains the definitions of the sorts (data types with associated operators and literals) and synonyms (named constants) used by the MAC state machines. Package macmib defines data types for attributes in the MAC MIB, and portions of the PHY MIB, accessed by the MAC state machines. Package macmib exists solely to satisfy SDL's strong type checking in the absence of an SDL tool that fully supports Z.105 (the combined use of SDL with ASN.1).

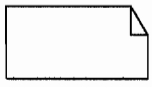


Package macsorts 3102_d\LmeEnum(31)

```
/*************************************************************
*       Enumerated types used in Mac and Mlme service primitives
*************************************************************/
newtype AuthType   /* <authentication type> parm in Mlme primitives */
  inherits Octetstring  operators all;
  adding  literals open_system,  shared_key;
  axioms  open_system == mkOS(0, 2);  shared_key == mkOS(1, 2);
endnewtype AuthType;
newtype AuthTypeSet  powerset( AuthType);   endnewtype AuthTypeSet;
newtype BssType   /* <BSS type> parameter & BSS description element */
  literals infrastructure,  independent,  any_bss;   endnewtype BssType;
newtype BssTypeSet  powerset( BssType);   endnewtype BssTypeSet;
newtype CfPriority   /* <priority> parameter of various requests */
  literals  contention,  contentionFree;   endnewtype CfPriority;
newtype MibStatus   /* <status> parm of Mlme/Plme Get/Set.confirm */
  literals  success,  invalid,  write_only,  read_only;
endnewtype MibStatus;
newtype MlmeStatus   /* <status> parm of Mlme operation confirm */
  literals  success,  invalid,  timeout,  refused,
          tomany_req,  already_bss;   endnewtype MlmeStatus;
newtype PwrSave   /* <power save mode> parameter of MlmePowerMgt */
  literals  sta_active,  power_save;   endnewtype PwrSave;
newtype Routing   /* <routing info> parameter for MAC data service */
  literals  null_rt;   endnewtype Routing;
newtype RxStatus   /* <reception status> parm of MaUnitdata indication */
  literals  rx_success,  rx_failure;   endnewtype RxStatus;
newtype ScanType   /* <scan type> parameter of MlmeScan.request */
  literals  active_scan,  passive_scan;   endnewtype ScanType;
newtype ServiceClass   /* <service class> parameter for MaUnitdata */
  literals  reorderable,  strictlyOrdered;   endnewtype ServiceClass;
newtype TxStatus   /* <transmission status> parm of MaUnitdataStatus */
  literals  successful,  retryLimit,  txLifetime,  noBss,
     excessiveDataLength,  nonNullSourceRouting,
     unsupportedPriority,  unavailablePriority,
     unsupportedServiceClass,  unavailableServiceClass,
     unavailableKeyMapping;  endnewtype TxStatus;
```

Package macsorts 3103_e\IntraMac(31)

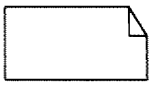
```
/*****************************************************************
 *       Intra-MAC remote variables (names of form mXYZ)
 *****************************************************************/
remote mActingAsAp Boolean nodelay;   /* =true if STA started BSS */
remote mAid  AsocId nodelay;   /* AID assigned to STA by AP */
remote mAssoc  Boolean nodelay;   /* =true if STA associated w/BSS */
remote mAtimW  Boolean nodelay;   /* =true if ATIM window in prog */
remote mBkIP  Boolean nodelay;   /* =true if backoff in prog */
remote mBrates Ratestring nodelay;   /* basic rate set for this sta */
remote mBssId  MacAddr nodelay;   /* identifier of current (I)BSS */
remote mCap  Octetstring nodelay;   /* capability info from MlmeJoin */
remote mCfp  Boolean nodelay;   /* =true if CF period in progress */
remote mDisable  Boolean nodelay;   /* =true if not in any BSS; then */
    /* TX only sends probe_req; RX only accepts beacon, probe_rsp */
remote mDtimCount  Integer nodelay;   /* =0 at Tbtt of Beacon with DTIM */
remote mFxIP  Boolean nodelay;   /* =true during frame exchange seq */
remote mIbss  Boolean nodelay;   /* =true if STA is member of IBSS */
remote mListenInt  Integer nodelay;   /* beacons between wake up @TBTT */
remote mNavEnd  Time nodelay;   /* NAV end Time, <=now when idle */
remote mNextBdry  Time nodelay;   /* next boundary Time; =0 if none */
remote mNextTbtt  Time nodelay;   /* Time next beacon due to occur */
remote mPcAvail  Boolean nodelay;   /* =true if point coord in BSS */
remote mPcDlvr  Boolean nodelay;   /* =true if CF delivery only */
remote mPcPoll  Boolean nodelay;   /* =true if CF delivery & polling */
remote mPdly  Usec nodelay;   /* probe delay from start or join */
remote mPss  PsState nodelay;   /* power save state of STA */
remote mReceiveDTIMs Boolean nodelay;  /* =true if DTIMs received */
remote mRxA  Boolean nodelay;   /* =true if RX indicated by PHY */
remote mSsId  Octetstring nodelay;   /* name of the current (I)BSS */
remote procedure TSF nodelay;   /* read & update 64-bit TSF timer */
   fpar Integer, Boolean;  returns Integer;
```



Package macsorts 3104_d\StaticData(31)

```
/*****************************************************************
 *       Named static data values   (names of form sXYZ)
 *****************************************************************/
synonym sMaxMsduLng Integer = 2304;   /* max octets in an MSDU */
synonym sMacHdrLng  Integer = 24;   /* octets in data header, no WEP */
synonym sWepHdrLng  Integer = 28;   /* octets in data header with WEP */
synonym sWepAddLng  Integer = 8;   /* octets added for WEP */
synonym sWdsAddLng  Integer = 6;   /* octets added for WDS (addr4) */
synonym sCrcLng  Integer = 4;   /* octets for crc32 (FCS, ICV) */
synonym sMaxMpduLng  Integer =       /* max octets in an MPDU */
   (sMaxMsduLng + sMacHdrLng + sWdsAddLng + sWepAddLng + sCrcLng);
syntype FrameIndexRange = Integer   /* index range for octets in MPDU */
   constants 0 : sMaxMpduLng   endsyntype FrameIndexRange;
synonym sTsOctet  Integer = 24;   /* first octet of Timestamp field */
synonym sMinFragLng Integer = 256;   /* min value for aMpduMaxLength */
synonym sMaxFragNum Integer =       /* maximum fragment number */
   (sMaxMsduLng / (sMinFragLng - sMacHdrLng - sCrcLng));
synonym sAckCtsLng Integer = 112;  /* bits in ACK and CTS frames */
```

```
/*****************************************************************
 *       Station configuration flags (static, supplementary to MIB)
 *****************************************************************/
synonym sVersion  Integer = 0;   /* supported Protocol Version */
synonym sCanBeAp  Boolean = false;   /* =true if STA can operate as AP */
synonym sCanBePc  Boolean = false;   /* =true if AP can be Point Coord */
synonym sCfPollable Boolean =true;   /* =true if responds to CF-polls */
```



Package macsorts 3105_d\Usec_TU(31)

```
/*****************************************************************
 *       Discrete microsecond and Time Unit sorts
 *****************************************************************/
/* SDL does not define the relationship between its concept */
/* of Time and physical time in the system being described. */
/* An abstraction is needed to establish this relationship, */
/* because Time in SDL uses the semantics of Real, whereas */
/* time in the MAC protocol is discrete, with the semantics */
/* of Natural and a step size (resolution) of 1 micosecond. */
/* Most MAC times are defined using the subtypes of Integer */
/* Usec and TU.  These have operators for explicit conversion */
/* to SDL Time (tUsec, tTU), SDL Duration (dUsec, dTU), and */
/* from SDL Time (uTime, tuTime) as needed to comply with SDL's */
/* strong type checking.  Where the MAC state machines need to */
/* access the contents of the TSF timer, SDL's 'now' (current */
/* time) is used.  This yields readable time-dependent code, */
/* but the value of 'now' cannot be modified by an SDL program, */
/* so adopting the TSF time from timestamps in received Beacons */
/* or Probe Responses is shown as an informal task symbol. */
/* Microsecond sort -- also has operators tmin and tmax */
newtype Usec  inherits Integer  operators all;
  adding  operators
    dUsec : Usec -> Duration;
    tUsec : Usec -> Time;
    uTime : Time -> Usec;
    tmax  : Usec, Usec -> Usec;
    tmin  : Usec, Usec -> Usec;
  axioms   for all u, w in Usec(
      u >= w ==> tmax(u, w) == u;      u < w ==> tmax(u, w) == w;
      u >= w ==> tmin(u, w) == w;      u < w ==> tmin(u, w) == u;
      for all t in Time(     for all r in Real(
          r = float(u) ==> tUsec(u) == Time!(Duration!(r));
          t = Time!(Duration!(r)) and u = fix(r) ==> u == uTime(t);));
      for all d in Duration(     for all r in Real(
          r = float(u) ==> dUsec(u) == Duration!(r); )));
  constants >= 0 /* constrain value range to be non-negative */
endnewtype Usec;
/* Time Unit sort -- (1 * TU) = (1024 * Usec) */
newtype TU  inherits Integer  operators all;
  adding  operators
    dTU   : TU -> Duration;
    tTU   : TU -> Time;
    tuTime : Time -> TU;
    u2TU   : Usec -> TU;
    tu2U   : TU -> Usec;
  axioms   for all k in TU(   for all t in Time(   for all r in Real(
          r = float(k) ==> tTU(k) == Time!(Duration!(1024 * r));
          t = Time!(Duration!(r)) and k = (fix(r) / 1024) ==> k == tuTime(t);));
      for all d in Duration(   for all r in Real(
          r = float(k) ==> dTU(k) == Duration!(1024 * r);));
      for all u in Usec(   u2TU(u) == u / 1024;   tu2U(k) == k * 1024; ));
  constants >= 0 /* constrain value range to be non-negative */
endnewtype TU;
```

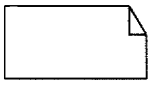
Package macsorts

3106_d\String0(31)

```
/*****************************************************************
 *       Generator for 0-origin String sorts (adapted from Z.105, Annex A)
 *****************************************************************/
/* String0(sort, nullSymbol) can define strings of any sort. */
/* These strings are indexed starting from 0 rather than 1. */
/* Sorts defined by String0 have the normal String operators, plus */
/* Tail (all but first item), Head (all but last item), and */
/* aggregators S2, S3, S4, S6, S8 (make fixed length strings). */
generator String0(type Item, literal Emptystring)
  literals Emptystring;
  operators
    MkString : Item -> String0;    /* make a string from an item */
    Length  : String0 -> Integer;    /* length of string */
    First  : String0 -> Item;    /* first item in string */
    Tail  : String0 -> String0;    /* all but first item in string */
    Last  : String0 -> Item;    /* last item in string */
    head  : String0 -> String0;    /* all but last item in string */
    "//"  : String0, String0 -> String0;    /* concatenation */
    Extract! : String0, Integer -> Item;    /* get item from string */
    Modify!  : String0, Integer, Item -> String0;  /* modify string */
    SubStr  : String0, Integer, Integer -> String0;
       /* SubStr(s,i,j) is string0 of length j starting at string0(i) */
    S2 : Item, Item -> String0;      S3 : Item, Item, Item -> String0;
    S4 : Item, Item, Item, Item -> String0;
    S6 : Item, Item, Item, Item, Item, Item -> String0;
    S8 : Item, Item, Item, Item, Item, Item, Item, Item -> String0;
 /* axioms  continued on next page... */

endgenerator String0;
```

Package macsorts 3107_a\String0(31)

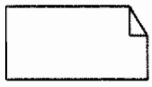
```
/* String0 axioms */
/*  for all item0,item1,item2,item3,item4,item5,item6,item7 in Item(
    for all s, s1, S2, S3 in String0(    for all i, j in Integer(
    constructors are Emptystring, MkString, and "//";
     equalities between constructor terms
       s // Emptystring == s;          Emptystring // s == s;
       (s1 // S2) // S3 == s1 // (S2 // S3);
     definition of Length by applying it to all constructors
        type String Length(Emptystring) == 0;
        type String Length(MkString(item0)) == 1;
        type String Length(s1 // S2) == Length(s1) + Length(S2);
     definition of Extract! by applying it to all constructors,
       Extract!(MkString(item0), 0) == item0;
       i < Length(s1) ==> Extract!(s1 // S2, i) == Extract!(s1, i);
       i >= Length(s1) ==> Extract!(s1 // S2, i) == Extract!(S2, i - Length(s1));
       i < 0 or i >= Length(s) ==> Extract!(s, i) == error!;
     definition of First and Last by other operations
       First(s) == Extract!(s, 0);
       Last(s) == Extract!(s, Length(s) - 1);
     definition of substr(s,i,j) by induction on j,
       i >= 0 and i <= Length(s) ==> SubStr(s, i, 0) == Emptystring;
       i >= 0 and j > 0 and i + j <= Length(s) ==> SubStr(s, i, j) ==
              SubStr(s, i, j - 1) // MkString(Extract!(s, i + j - 1));
       i < 0 or j < 0 or i + j > Length(s) ==> SubStr(s, i, j) == error!;
     definition of Modify!, Head, Tail, Sx by other operations
       Modify!(s, i, item0) == SubStr(s, 0, i) // MkString(item0) //
          SubStr(s, i + 1, Length(s) - i - 1);
       head(s) == SubStr(s, 0, Length(s) - 1);
       Tail(s) == SubStr(s, 1, Length(s) - 1);
       S2(item0, item1) == MkString(item0) // MkString(item1);
       S3(item0, item1, item2) ==
         MkString(item0) // MkString(item1) // MkString(item2);
       S4(item0, item1, item2, item3) ==
         MkString(item0) // MkString(item1) // MkString(item2) //
           MkString(item3);
       S6(item0, item1, item2, item3, item4, item5) ==
         MkString(item0) // MkString(item1) // MkString(item2) //
           MkString(item3) // MkString(item4) // MkString(item5);
       S8(item0, item1, item2, item3, item4, item5, item6, item7) ==
         MkString(item0) // MkString(item1) // MkString(item2) //
           MkString(item3) // MkString(item4) // MkString(item5) //
           MkString(item6) // MkString(item7); ))));
*/
```



Package macsorts 3108_d\Bitstring(31)

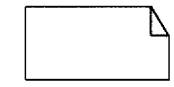
```
/*************************************************************
 *       ASN.1-style BIT sort    (from Z.105, Annex A)
 *************************************************************/
   /* Bit is a subtype of Boolean -- bit values 0 and 1 are
   /* not numerals and cannot be used with Integer operators */
newtype Bit   inherits Boolean
   literals  0 = false,  1 = true;   operators all;   endnewtype Bit;
```

```
/*************************************************************
 *       ASN.1-style BIT STRING sort    (adapted from Z.105, Annex A)
 *************************************************************/
/* Bitstrings are 0-origin strings of Bit.  Z.105 uses ASN.1-style */
/* literals in binary ('1011'B) or hexadecimal ('D3'H), but this */
/* syntax is not accepted for Z.100 string literals.  Therefore, */
/* this version provides only hexadecimal literals 0x00-0xFF. */
/* Bitstring operators '=>', 'not', 'and', 'or', and 'xor' act */
/* bitwise, with the length of the result string equal to the */
/* length of the longest (or only) source string. */
newtype Bitstring String0(Bit, '')
  adding  literals     macro Hex_Literals;
  operators
    "not" : Bitstring -> Bitstring;
    "and" : Bitstring, Bitstring -> Bitstring;
    "or"  : Bitstring, Bitstring -> Bitstring;
    "xor" : Bitstring, Bitstring -> Bitstring;
    "=>"  : Bitstring, Bitstring -> Bitstring;     noequality;
  axioms     macro Hex_Axioms;
    for all s, s1, S2, S3 in Bitstring(
      s = s == true;      s1 = S2 == S2 = s1;
      s1 /= S2 == not (s1 = S2);      s1 = S2 == true ==> s1 == S2;
      ((s1 = S2) and (S2 = S3)) ==> s1 = S3 == true;
      ((s1 = S2) and (S2 /= S3)) ==> s1 = S3 == false;
      for all b, b1, b2 in Bit(
        not (') == ';
        not (MkString(b) // s) == MkString(not (b)) // not (s);
        '' and '' == '';
        Length(s) > 0 ==> '' and s == MkString(0) and s;
        Length(s) > 0 ==> s and '' == s and MkString(0);
        (MkString(b1) // s1) and (MkString(b2) // S2) ==
          MkString(b1 and b2) // (s1 and S2);
        s1 or S2 == not (not s1 and not S2);
        s1 xor S2 == (s1 or S2) and not (s1 and S2);
        s1 => S2 == not (not s1 and S2);));
  map  for all b1, b2 in Bitstring literals(
     for all bs1, bs2 in Charstring literals(
/* connection to the String generator */
       for all b in Bit literals(
          spelling(b1) = "" // bs1 // bs2 // "",
         spelling(b2) = "" // bs2 // "",  spelling(b) = bs1
           ==> b1 == MkString(b) // b2; )));
endnewtype Bitstring;
```



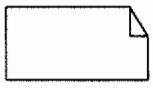
Package macsorts 3109_d\Octetstring(31)

```
/*****************************************************************
 *       OCTET sort    (influenced by Z.105, Annex A)
 *****************************************************************/
/* Octet is a subtype of Bitstring where length always =8. */
/* Z.105 adds a "size" keyword to SDL and defines Octet with */
/* "... constants size (8) ..." to impose this length constraint. */
/* Here Octet relies on proper use maintain lengths as multiples */
/* of 8.  Proper length strings are created by the hexadecimal */
/* Bitstring literals (e.g. 0xD5) and operator mkOctet: */
/*    o:= mkOctet(i)    converts a non-negative Integer (mod 256) */
/*                              to an Octet (exactly 8 bits) */
/*    i:= octetVal(o)    converts an Octet to an Integer (0:255) */
/*    o:= flip(o)    reverses bit order of the Octet */
/*                              (0<-->7, 1<-->6, 2<-->5, 3<-->4) */
newtype Octet   inherits Bitstring   operators all;
  adding  operators
    mkOctet  : Integer -> Octet;
    octetVal : Octet -> Integer;
    flip     : Octet -> Octet;
  axioms
    for all i in Integer(     for all z in Octet(
        i = 0 ==> mkOctet(i) == S8(0, 0, 0, 0, 0, 0, 0, 0);
        i = 1 ==> mkOctet(i) == S8(1, 0, 0, 0, 0, 0, 0, 0);
        i > 1 and i <= 255 ==> mkOctet(i) ==
           SubStr((First(mkOctet(i mod 2)) // mkOctet(i / 2)), 0, 8);
        i > 255 ==> mkOctet(i) == mkOctet(i mod 256);
        i < 0 ==> mkOctet(i) == error!;
        z = MkString(0) ==> octetVal(z) == 0;
        z = MkString(1) ==> octetVal(z) == 1;
        Length(z) > 1 and Length(z) <= 8 ==>
           octetVal(z) == octetVal(First(z)) +
              (2 * (octetVal(SubStr(z, 1, Length(z) - 1))));
        Length(z) > 8 ==> octetVal(z) == error!;
        flip(z) == S8(z(7),z(6),z(5),z(4),z(3),z(2),z(1),z(0)); ));
endnewtype Octet;
```



Package macsorts 3109.1_a\Octetstring(31)

```
/*****************************************************************
 *       OCTET STRING sort    (somewhat influenced by Z.105, Annex A)
 *****************************************************************/
/* Octetstrings are 0-ORIGIN strings of Octet, NOT 1-ORIGIN */
/* strings like Octet_String in Z.105 (hence the name change). */
/* Octetstring has conversion operators to and from Bitstring, */
/* and integer to Octetstring.  Octetstring literals are "null" */
/* and 1-4, 6, 8 item 0x00 strings O1, O2, O3, O4, O6, O8. */
newtype Octetstring String0(Octet, null)
 adding  literals O1, O2, O3, O4, O6, O8;
 operators
  B_S  : Octetstring -> Bitstring;    /* name changed from Z.105 */
  O_S  : Bitstring -> Octetstring;    /* name changed from Z.105 */
  mkOS : Integer,Integer -> Octetstring; /* mkOS(i1,i2) returns */
                /* mkstring(mkOctet(i1)) padded (0x00) to length i2 */
  mk2octets : integer -> Octetstring;    /* 16-bit int to 2-octets */
 axioms
  for all b, b1, b2 in Bitstring(
   for all s in Octetstring(     for all o in Octet(
      B_S(null) == null;          O_S(null) == null;
      B_S(MkString(o) // s) == o // B_S(s);
      Length(b1) > 0, Length(b1) < 8 ==>
       O_S(b1) == MkString(b1 or 0x00);  /* expand b1 to 8 bits */
      b == b1 // b2, Length(b1) = 8 ==>
       O_S(b) == MkString(b1) // O_S(b2);
      for all i, k in Integer(
       k = 1 ==> mkOS(i, k) == MkString(mkOctet(i));
       k > 1 ==> mkOS(i, k) == mkOS(i, k - 1) // MkString(0x00);
       k <= 0 ==> error!;
       mk2octets(i) == MkString(mkOctet(i mod 256)) //
        MkString(mkOctet(i / 256)); );
      O1 == MkString(0x00);     O2 == O1 // O1;
      O3 == O2 // O1;           O4 == O2 // O2;
      O6 == O4 // O2;           O8 == O4 // O4; )));
 map  for all O1, O2 in Octetstring literals(
    for all b1, b2 in Bitstring literals(
      spelling(O1) =  spelling(b1),  spelling(O2) =  spelling(b2)
       ==> O1 = O2 == b1 = b2; ));
endnewtype Octetstring;
```



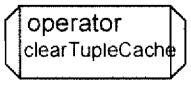
Package macsorts 3110_d\MacAddr(31)

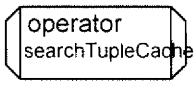
```
/*****************************************************************
 *       MAC Address sorts
 *****************************************************************/
/* MacAddr is a subtype of Octetstring with added operators: */
/*    isGroup(m) =true if given a group address */
/*    isBcst(m)  =true if given the broadcast address */
/*    isLocal(m) =true if given a locally-administered address */
/*    adrOs(m)   converts MacAddr to Octetstring */
/* MAC addresses must be defined to be exactly 6 octets long, */
/* typically using the S6 operator or nullAddr synonym. */
newtype MacAddr  inherits Octetstring  operators all;
 adding  operators
   isGroup : MacAddr -> Boolean;
   isBcst  : MacAddr -> Boolean;
   isLocal : MacAddr -> Boolean;
   adrOs   : MacAddr -> Octetstring;
 axioms
   for all m in MacAddr(
     (Length(m) = 6) and ((Extract!(m,0) and 0x01) = 0x01) ==> isGroup(m) == true;
     (Length(m) = 6) and ((Extract!(m,0) and 0x01) = 0x00) ==> isGroup(m) == false;
     (Length(m) = 6) and (m = S6(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF)) ==> isBcst == true;
     (Length(m) = 6) and (m /= S6(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF)) ==> isBcst == false;
     (Length(m) = 6) and ((Extract!(m,0) and 0x02) = 0x02) ==> isLocal == true;
     (Length(m) = 6) and ((Extract!(m,0) and 0x02) = 0x00) ==> isLocal == false;
     Length(m) /= 6 ==> error! /* common error! term */;
     for all o in Octetstring(m = MacAddr!(o) == adrOs(m) = o; ));
endnewtype MacAddr;
newtype MacAddrSet  powerset( MacAddr)  endnewtype MacAddrSet;
synonym bcstAddr  MacAddr =   /* Broadcast Address */
        <<type MacAddr>>  S6(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
synonym nullAddr  MacAddr =   /* Null Address */
        << type MacAddr>>  S6(0x00,0x00,0x00,0x00,0x00,0x00);
```

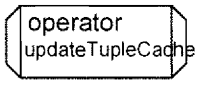
```
/*****************************************************************
 *       BSS description sorts
 *****************************************************************/
   /* BssDscr is used with MlmeScan.confirm and MlmeJoin.request */
newtype BssDscr  struct
   bdBssId   MacAddr;
   bdSsId   Octetstring;   /* 1 <= length <= 32 */
   bdType   BssType;
   bdBcnPer   TU;   /* beacon period in Time Units */
   bdDtimPer   Integer;   /* DTIM period in beacon periods */
   bdTstamp   Octetstring;   /* 8 Octets from ProbeRsp/Beacon */
   bdStartTs   Octetstring;   /* 8 Octets TSF when rx Tstamp */
   bdPhyParms   PhyParms;   /* empty if not needed by PHY */
   bdCfParms   CfParms;   /* empty if not CfPollable/no PCF */
   bdIbssParms   IbssParms;   /* empty if infrastructure BSS */
   bdCap   Capability;   /* capability information */
   bdBrates   Ratestring;   /* BSS basic rate set */
endnewtype BssDscr;
newtype BssDscrSet  powerset( BssDscr)  endnewtype BssDscrSet;
```

Package macsorts 3111_d\TupleCache(31)

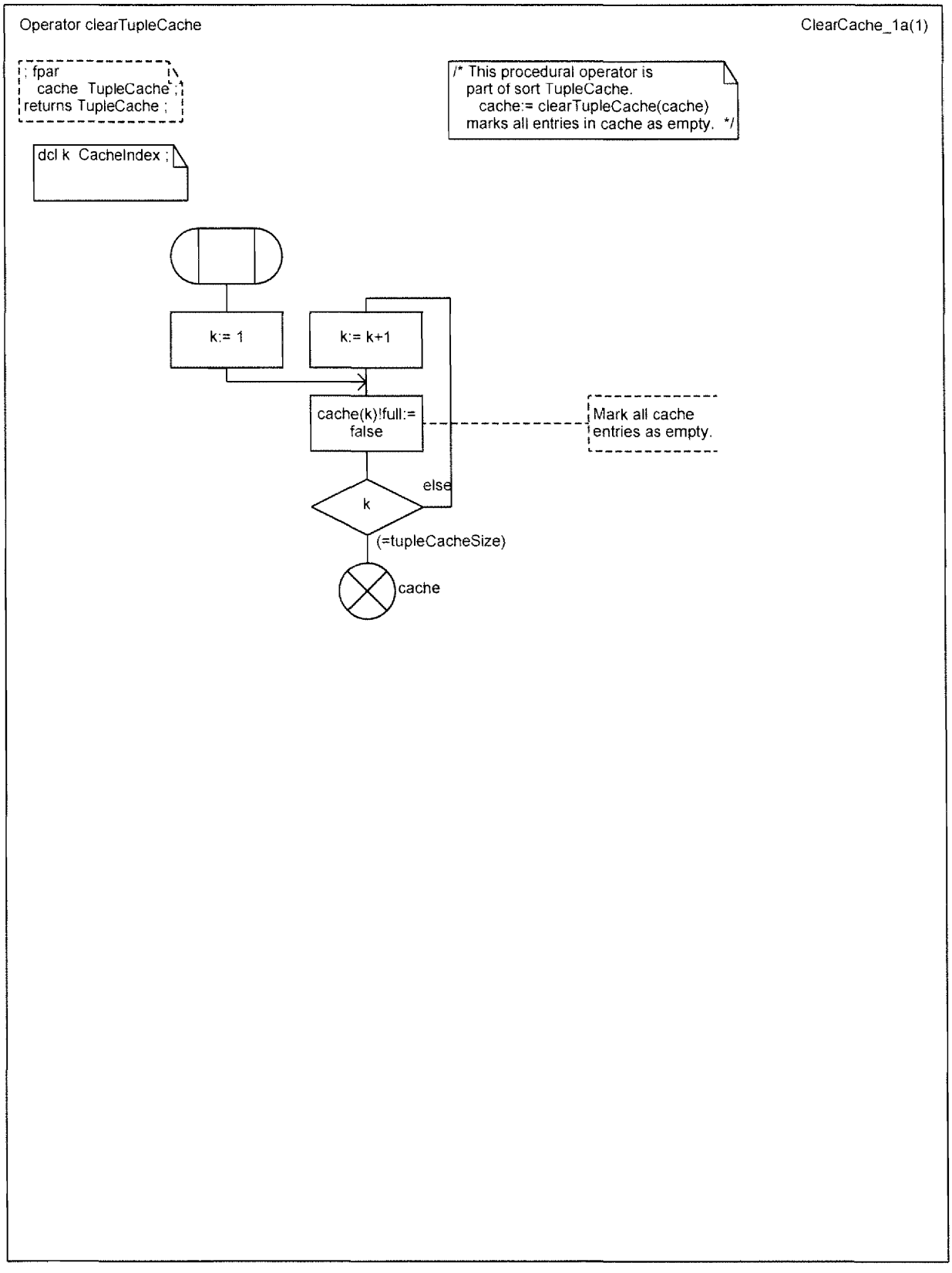
```
/*****************************************************************
 *      Duplicate filtering support sorts
 *****************************************************************/
syntype FragNum = Integer    /* Range of possible fragment numbers */
      constants 0:sMaxFragNum  endsyntype FragNum;
syntype SeqNum = Integer    /* Range of possible sequence numbers */
      constants 0:4095  endsyntype SeqNum;
newtype Tuple  struct    /* for duplicate filtering & defragmentation */
   full    Boolean;    /* =true if Tuple contains valid info */
   ta    MacAddr;    /* transmitting station address (Addr2) */
   sn    SeqNum;    /* Msdu/Mmpdu sequence number */
   fn    FragNum;    /* most recent Mpdu fragment number */
   tRx    Time;    /* reception time (endRx of fragment) */
 default (. false, nullAddr, 0, 0, 0 .);
endnewtype Tuple;
```

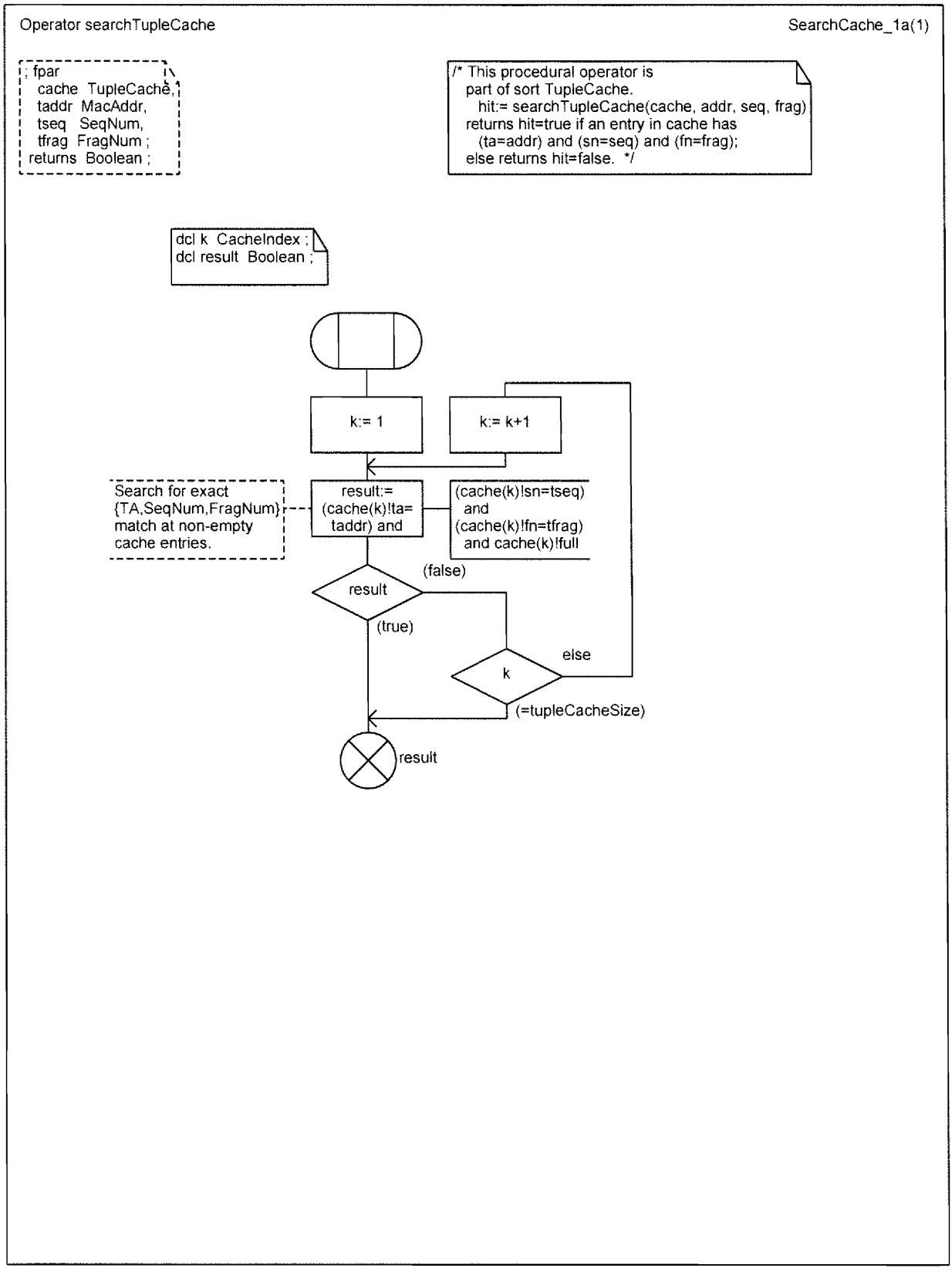


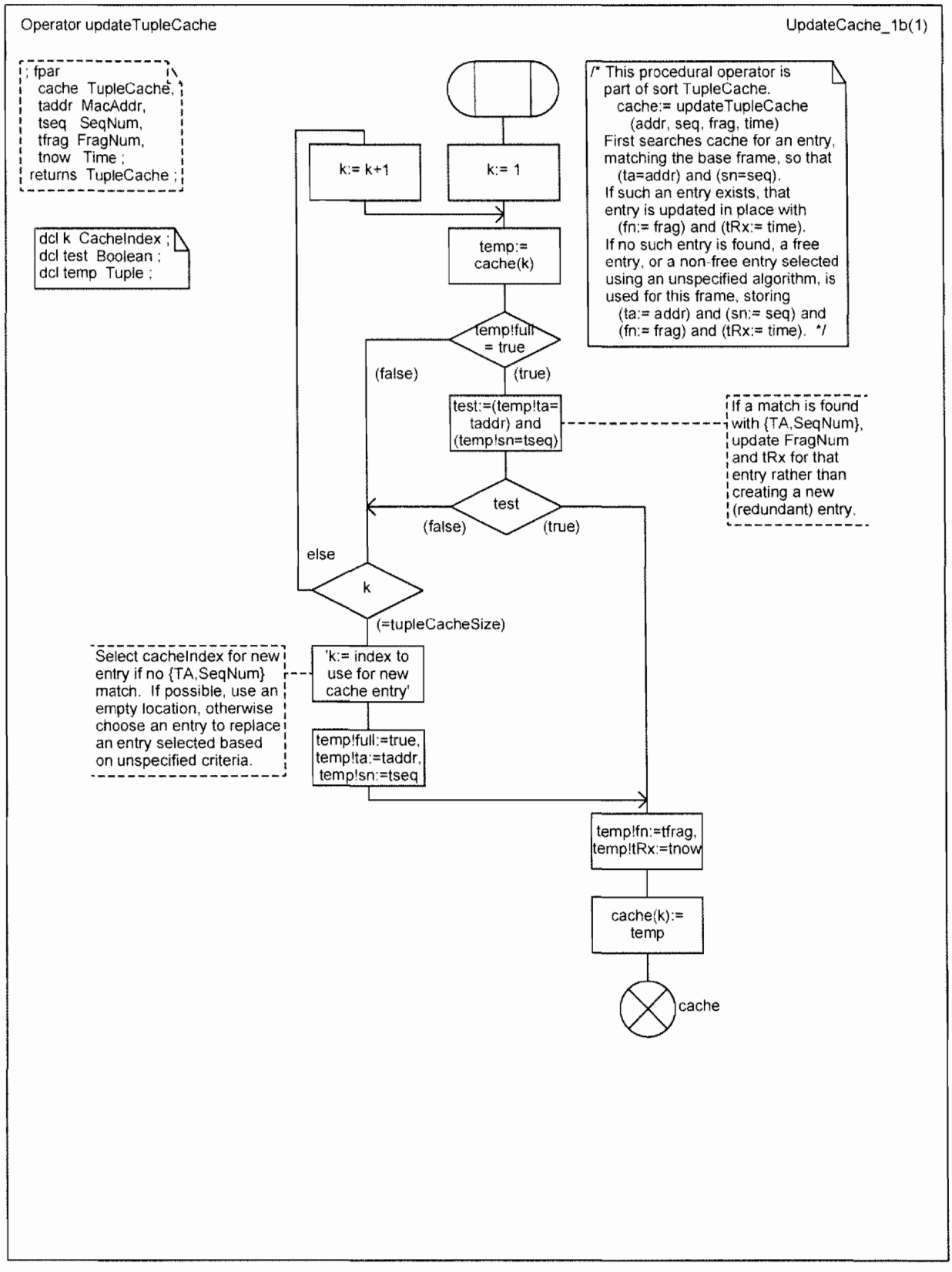





```
/*****************************************************************
 *      TupleCache support sorts
 *****************************************************************/
   /* Number of TupleCache entries and associated index range */
synonym tupleCacheSize Integer = 32;    /* this value is an example,
                  TupleCache size is implementation dependent */
syntype CacheIndex = Integer  constants 1:tupleCacheSize
  endsyntype CacheIndex;
/* TupleCache array */
/*    cache:= ClearTupleCache(cache)  to initialize cache */
/*    cache:= UpdateTupleCache(cache, addr, seq, frag, endRx) */
/*         if <addr,seq> is already cached, updates frag */
/*         if <addr,seq> not cached, fills an empty entry */
/*              or replaces an entry using an unspecified algorithm */
/*    SearchTupleCache(cache, addr, seq, frag) */
/*              returns true if specified <addr,seq,frag> in cache */
newtype TupleCache  Array( CacheIndex, Tuple);
  adding  operators
    ClearTupleCache  : TupleCache -> TupleCache;
    SearchTupleCache : TupleCache, MacAddr, SeqNum, FragNum -> Boolean;
    UpdateTupleCache : TupleCache, MacAddr, SeqNum, FragNum, Time ->
        TupleCache;
  operator ClearTupleCache;
    fpar cache TupleCache;  returns TupleCache;  referenced;
  operator SearchTupleCache;
    fpar cache TupleCache,  taddr MacAddr,  tseq SeqNum,  tfrag FragNum;
    returns Boolean;  referenced;
  operator UpdateTupleCache;
    fpar cache TupleCache,  taddr MacAddr,  tseq SeqNum,  tfrag FragNum,
    tnow Time;  returns TupleCache;  referenced;
endnewtype TupleCache;
```

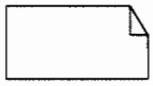






Package macsorts 3112_d\Counter(31)

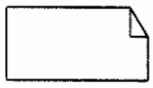
```
/*****************************************************************
 *       32-bit Counter sort and Integer string sort
 *****************************************************************/
/* This sort used for MIB counters, needed because SDL Integers */
/* have no specified maximum value.  inc(counter) increments the */
/* counter value by 1, with wraparound from (2^32)-1 to 0. */

newtype Counter32   inherits Integer   operators all;
  adding  operators
    inc : Counter32 -> Counter32;
  axioms
    for all c in Counter32 (
      c < 4294967295 ==> inc(c) == c + 1;
      c >= 4294967295 ==> inc(c) == 0; );
endnewtype Counter32;
       /* String (1-origin) of Integer */
newtype Intstring   String( Integer, noInt);   endnewtype Intstring;
```



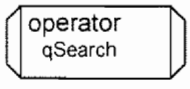
Package macsorts

3113_d\Queue(31)

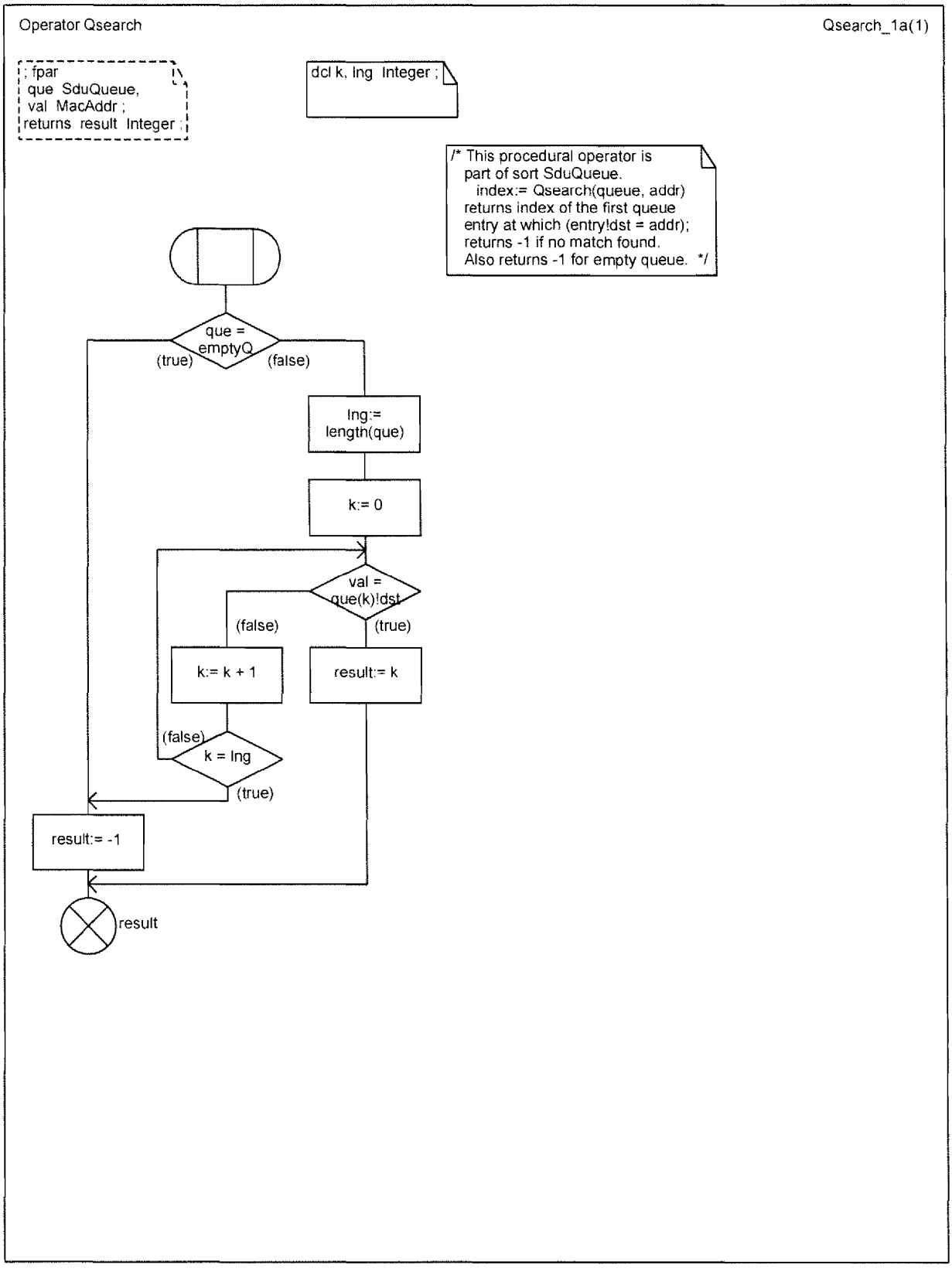
```
/*************************************************************
 *       Generator for Queue sorts
 *************************************************************/
/* The Queue generator is derived from the String0 generator */
/* to create Queues of any sort.  Queues operators are: */
/*    Qfirst(queue,item)  adds item as the first queue element */
/*    Qlast(queue,item)   adds item as the last queue element */
/* and the String0 operators Length, //, First, Last, Head, Tail */
/* Because operators can only return a single value, removing an */
/* element from a queue is a 2-step process: */
/*    dequeue first:  item:=First(queue);  queue:=Tail(queue); */
/*    dequeue last:   item:=Last(queue);   queue:=Head(queue); */
generator Queue(type Item, literal Emptyqueue)
  literals Emptyqueue;
  operators
    MkQ  : Item -> Queue;    /* make a queue from an item */
    Length  : Queue -> Integer;    /* number of items on queue */
    First  : Queue -> Item;    /* first item on queue */
    Qfirst  : Queue,Item -> Queue;    /* add item as first on queue */
    Tail  : Queue -> Queue;    /* all but first item on queue */
    Last  : Queue -> Item;    /* last item on queue */
    Qlast  : Queue,Item -> Queue;    /* add item as last on queue */
    head  : Queue -> Queue;    /* all but last item on queue */
     "//"  : Queue, Queue -> Queue;    /* concatenation */
    Extract! : Queue,Integer -> Item;    /* copy item from queue */
    Modify!  : Queue,Integer,Item -> Queue; /* modify item in queue */
    SubQ  : Queue,Integer,Integer -> Queue;
          /* SubQ(q,i,j) queue of length j starting from queue(i) */
  axioms
    for all item0 in Item(     for all q, q1, q2, q3 in Queue(
       for all i, j in Integer(
    /* constructors are Emptyqueue, MkQueue, and "//"; */
    /* equalities between constructor terms */
        q // Emptyqueue == q;          Emptyqueue // q == q;
        (q1 // q2) // q3 == q1 // (q2 // q3);
    /* definition of Length by applying it to all constructors */
         type Queue Length(Emptyqueue) == 0;
         type Queue Length(MkQueue(item0)) == 1;
         type Queue Length(q1 // q2) == Length(q1) + Length(q2);
    /* definition of Extract! by applying it to all constructors, */
        Extract!(MkQueue(item0), 0) == item0;
        i < Length(q1) ==> Extract!(q1 // q2, i) == Extract!(q1, i);
        i >= Length(q1) ==> Extract!(q1 // q2, i) == Extract!(q2, i - Length(q1));
        i < 0 or i >= Length(q) ==> Extract!(q, i) == error!;
    /* definition of First and Last by other operations */
        First(q) == Extract!(q, 0);     Last(q) == Extract!(q, Length(q) - 1);
    /* definition of SubQ(q,i,j) by induction on j, */
        i >= 0 and i <= Length(q) ==> SubQ(q, i, 0) == Emptyqueue;
        i >= 0 and j > 0 and i + j <= Length(q) ==> SubQ(q, i, j) ==
           SubQ(q, i, j - 1) // MkQueue(Extract!(q, i + j - 1));
        i < 0 or j < 0 or i + j > Length(q) ==> SubQ(q,i,j) == error!;
    /* define Modify!, Head, Tail, Qfirst, Qlast by other ops */
        Modify!(q, i, item0) == SubQ(q, 0, !) //
           MkQueue(item0) // SubQ(q, i + 1, Length(q) - i - 1);
        head(q) == SubQ(q, 0, Length(q) - 1);
        Tail(q) == SubQ(q, 1, Length(q) - 1);
        Qfirst(q, item0) == MkQueue(item0) // q;
        Qlast(q, item0) == q // MkQueue(item0); )));
endgenerator Queue;
```



Package macsorts 3114_d\Fragment(31)




```
/*************************************************************
 *       Fragmentation support sorts
 *************************************************************/
/* Array to hold up to FragNum fragments of an Msdu/Mmpdu */
newtype FragArray  Array(FragNum, Frame);   endnewtype FragArray;
/* FragSdu structure is for OUTGOING MSDUs/MMPDUs (called SDUs) */
/* Each SDU, even if not fragmented, is held in an instance of */
/* this structure awaiting its (re)transmission attempt(s). */
/* Transmit queue(s) are ordered lists of FragSdu instances. */
newtype FragSdu struct
   fTot    FragNum;    /* number of fragments in pdus FragArray */
   fCur    FragNum;    /* next fragment number to send */
   fAnc    FragNum;    /* next fragment to announce in ATIM or TIM
                          when fAnc > fCur, pdus(fCur)+ may be sent */
   eol     Time;    /* set to (now + dUsec(aMaxTxMsduLifetime))
                          when the entry is created */
   sqf     SeqNum;    /* SDU sequence number, set at 1st Tx attempt */
   src     Integer;    /* short retry counter for this SDU */
   lrc     Integer;    /* long retry counter for this SDU */
   dst     MacAddr;    /* destinaton address */
   grpa    Boolean;    /* =true if RA (not DA) is a group address */
   psm     Boolean;    /* =true if RA (not DA) may be in pwr_save */
   resume    Boolean;    /* =true if fragment burst being resumed */
   cnfTo     PId;    /* address to which confirmation is sent */
   txrate    Rate;    /* data rate used for initial fragment */
   cf    CfPriority;    /* requested priority (from LLC) */
   pdus    FragArray;    /* array of Frame to hold fragments */
endnewtype FragSdu;
/* Queue of FragSdu */
/* for power save buffers, etc., searchable with Qsearch operator: */
/*    index:= Qsearch(queue, addr)     where queue is an SduQueue, */
/* index identifies the first queue entry at which */
/* entry!dst = addr; or as -1 if no match (or queue empty). */
newtype SduQueue Queue(FragSdu, emptyQ);
  adding  operators
    qSearch : SduQueue, MacAddr -> Integer;
  operator qSearch;
    fpar que SduQueue,  val MacAddr;  returns Integer; referenced;
endnewtype SduQueue;
```



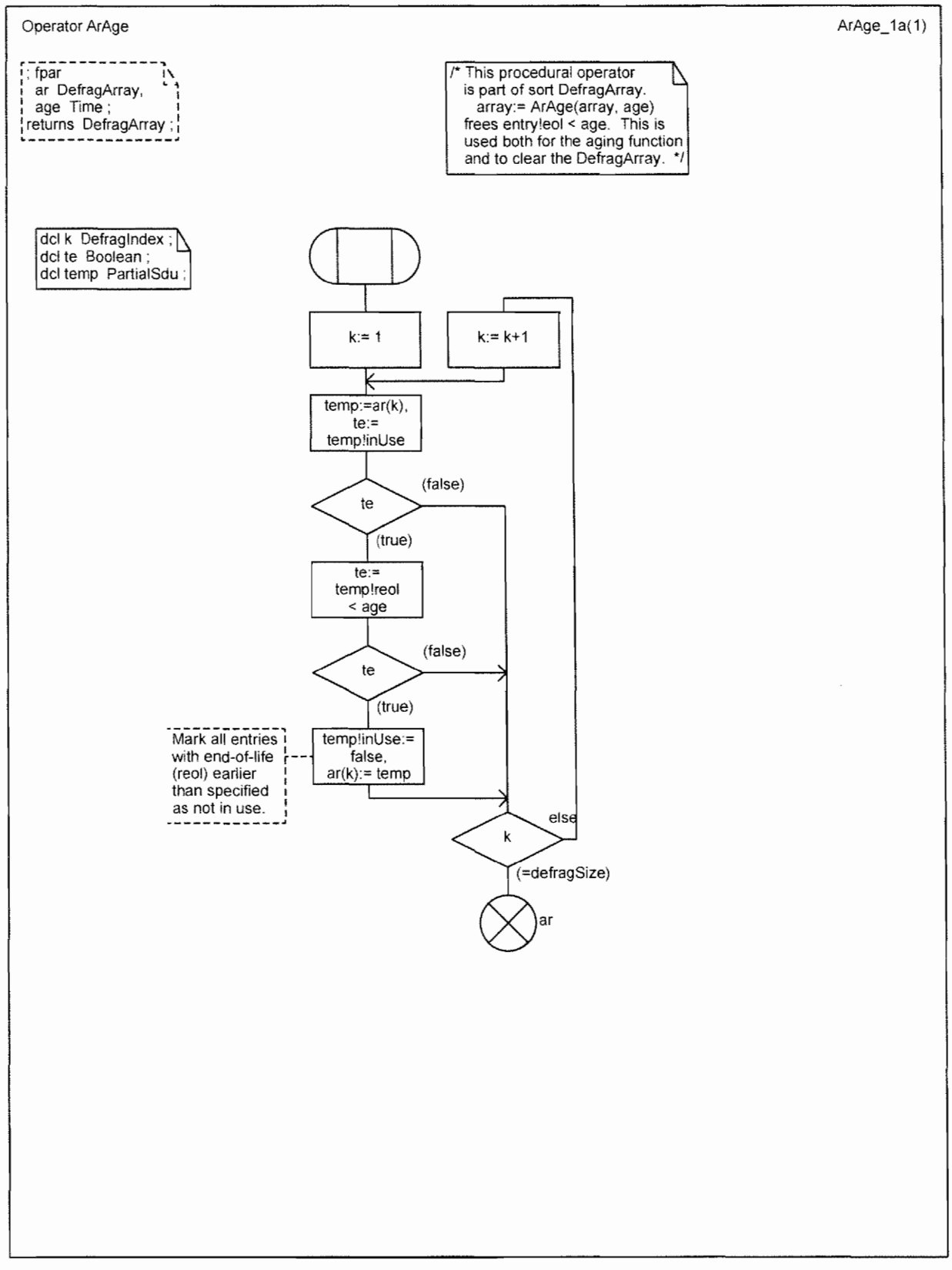
Package macsorts 3115_d\Defragment(31)

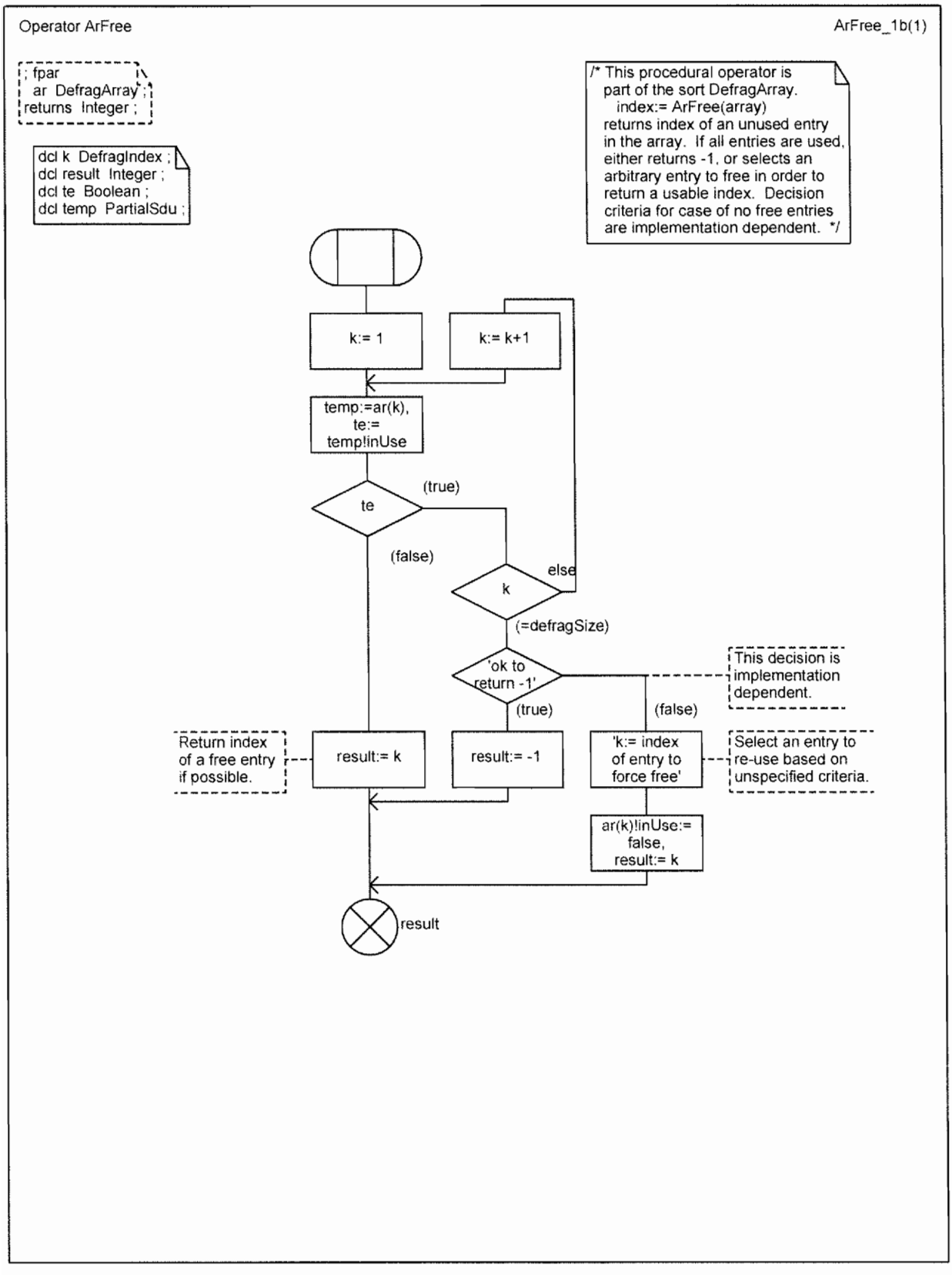
operator ArAge

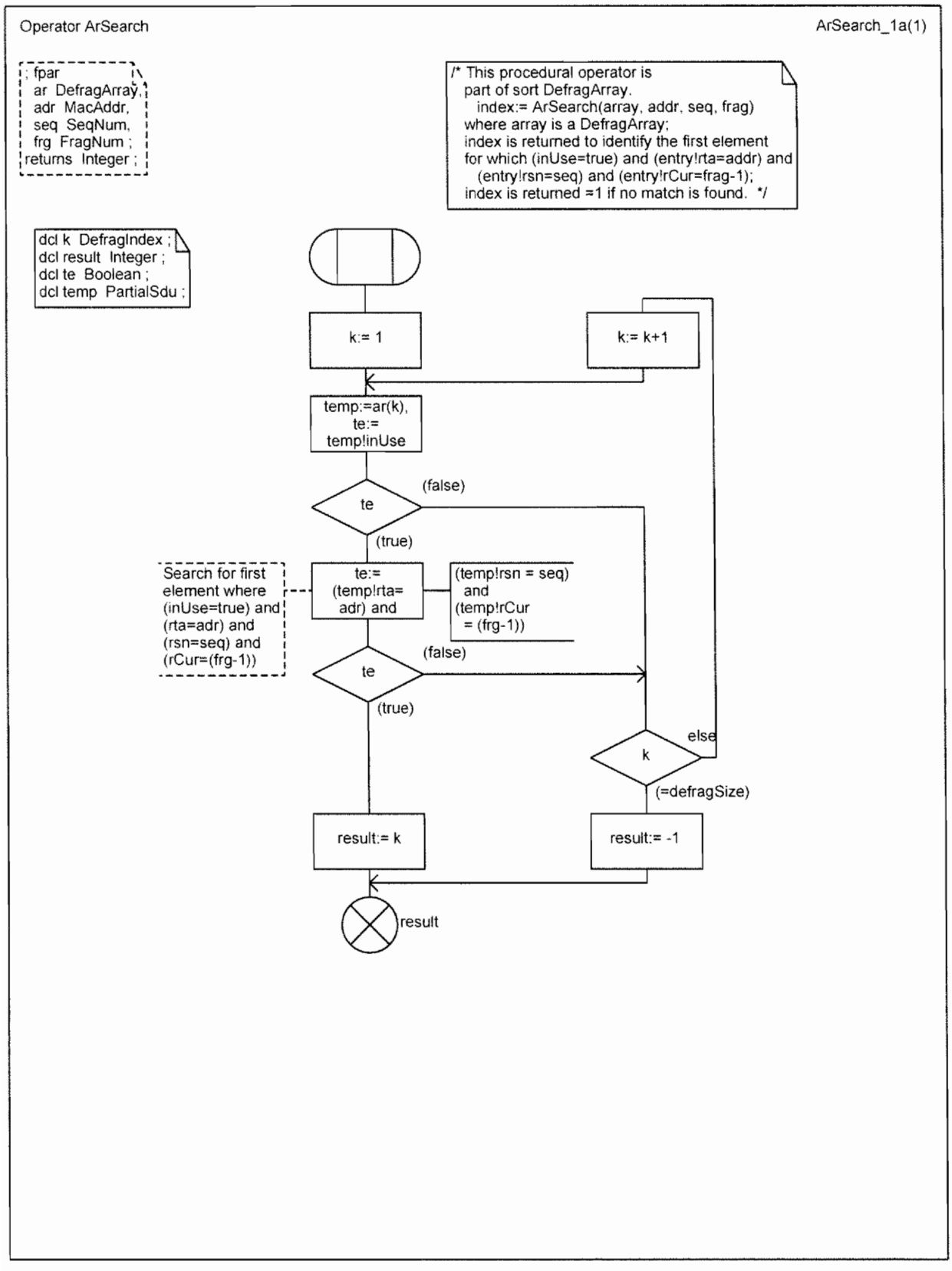
operator ArFree

operator ArSearch

```
/*****************************************************************
*       Defragmentation support sorts
*****************************************************************/
/* The PartialSdu structure is for INCOMPLETE MSDUs/MMPDUs */
/* (generically SDUs) for which at least 1 fragment has been */
/* received. Unfragmented SDUs are reported upward immediately, */
/* and are never stored in instances of this structure. */
newtype PartialSdu struct
   inUse    Boolean;    /* =true if this instance holds any fragments */
   rta    MacAddr;    /* transmitting station (Addr2) */
   rsn    SeqNum;    /* SDU sequence number */
   rCur    FragNum;    /* fragment number of most recent Mpdu */
   reol    Time;    /* (now+dUsec(aMaxReceiveLifetime) @ 1st Mpdu */
   rsdu    Frame;    /* buffer where Mpdus are concatenated */
 default (. false, nullAddr, 0, 0, 0, null .);
endnewtype PartialSdu;
newtype PartialSduKeys struct  /* if aPrivacyOptionImplemented=true */
   wDefKeys    KeyVector;    /* default keys when 1st frag received */
   wKeyMap    KeyMapArray;    /* key mappings when 1st frag received */
   wExclude    Boolean;    /* aExcludeUnencrypted @ 1st frag rx */
endnewtype PartialSduKeys;
/* Number of entries in defragmentation array at this station. */
/* The value is implementation dependent (min=3, see 9.5). */
synonym defragSize Integer = 6;
syntype defragIndex = Integer   constants 1:defragSize
endsyntype defragIndex;
/* Array of PartialSdu for use defragmenting Msdus and Mmpdus. */
/* Searchable using the ArSearch operator */
/*    index:= ArSearch(array, addr, seq, frag) */
/* where index is returned to identify the first element for which */
/* ((inUse = true) and (entry!rta = addr) and (entry!rsn = seq) */
/*  and (entry!rCur = (frag-1));   or as =1 if no match found. */
/*    index:= ArFree(array)    returns the index of a free entry, */
/* or -1 if no entries free.  May free an entry, selected using */
/* an unspecified algorithm, to avoid returning -1. */
/*    array:= ArAge(array, age) */
/* frees where (entry!eol < age), also used to clear array. */
newtype DefragArray  Array( defragIndex, PartialSdu);
  adding  operators
    ArSearch : DefragArray, MacAddr, SeqNum, FragNum -> Integer;
    ArFree   : DefragArray -> Integer;
    ArAge    : DefragArray, Time -> DefragArray;
  operator ArSearch;
    fpar  ar DefragArray,   adr MacAddr,   seq SeqNum,   frg FragNum;
    returns Integer; referenced;
  operator ArFree;  fpar  ar DefragArray;   returns Integer; referenced;
  operator ArAge;  fpar  ar DefragArray,   age Time;
    returns DefragArray; referenced;
endnewtype DefragArray;
newtype DefragKeysArray  Array( defragIndex, PartialSduKeys);
endnewtype DefragKeysArray;
```







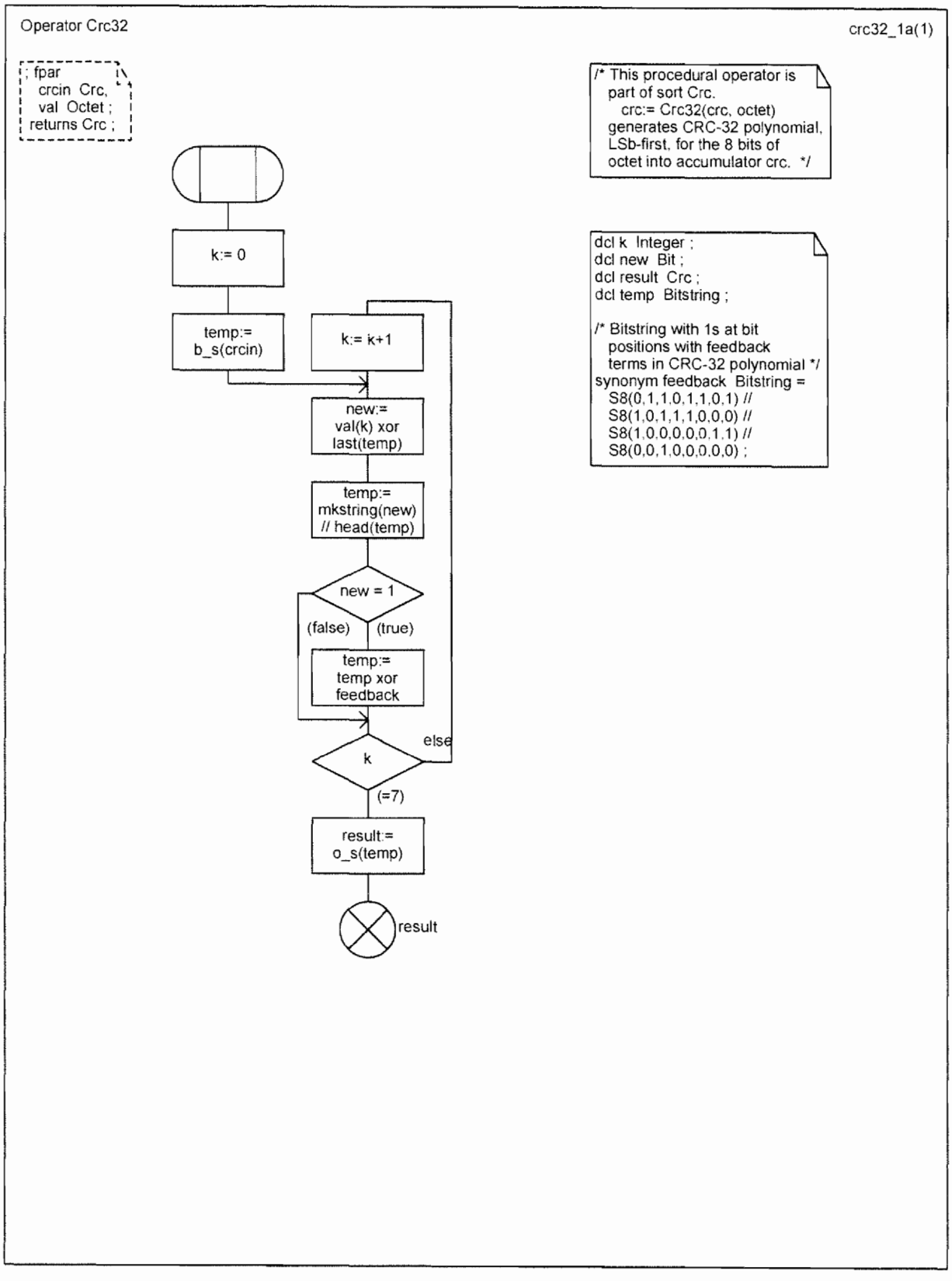
Package macsorts 3116_d\Crc_Wep(31)

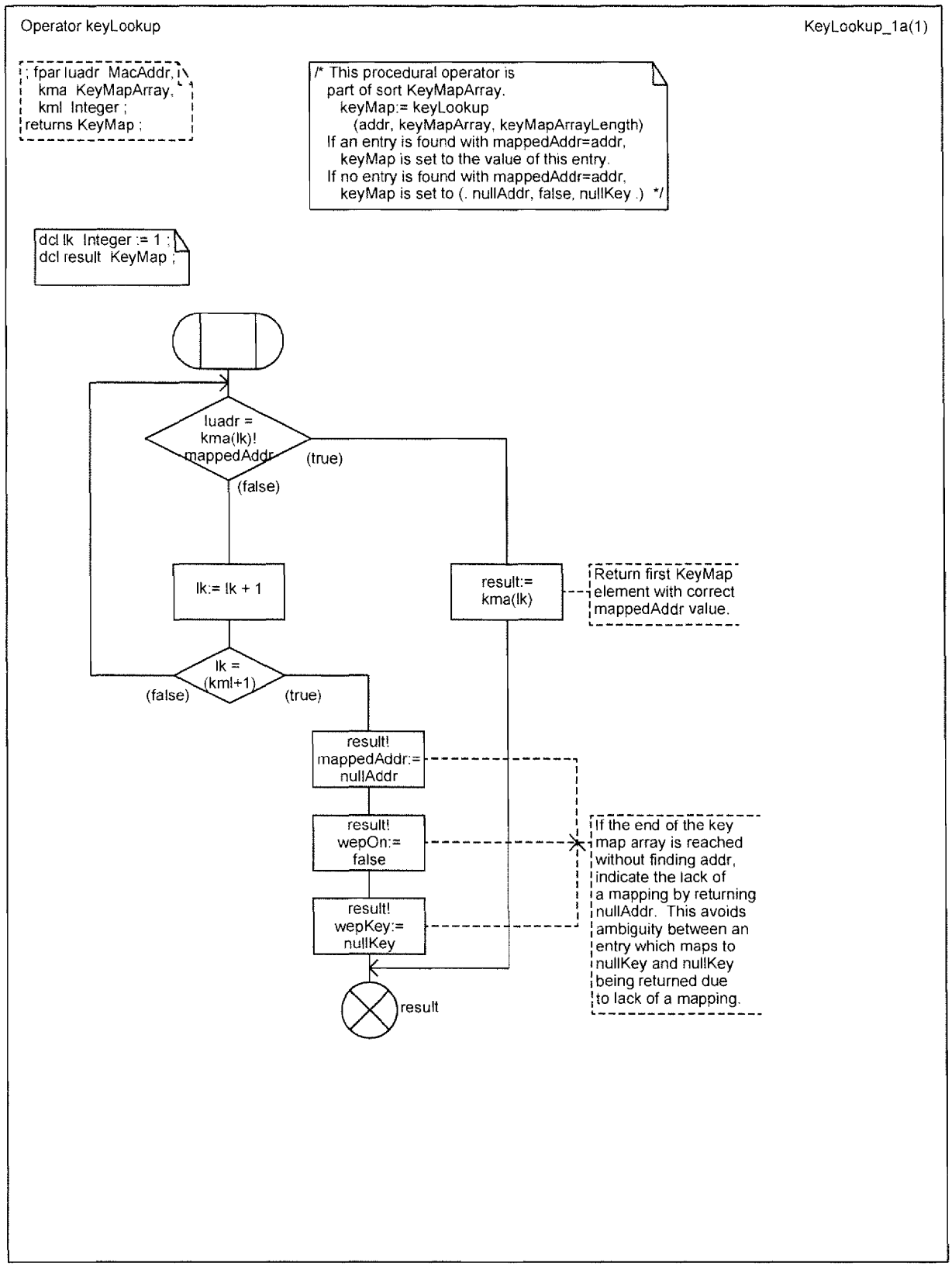
operator
crc32

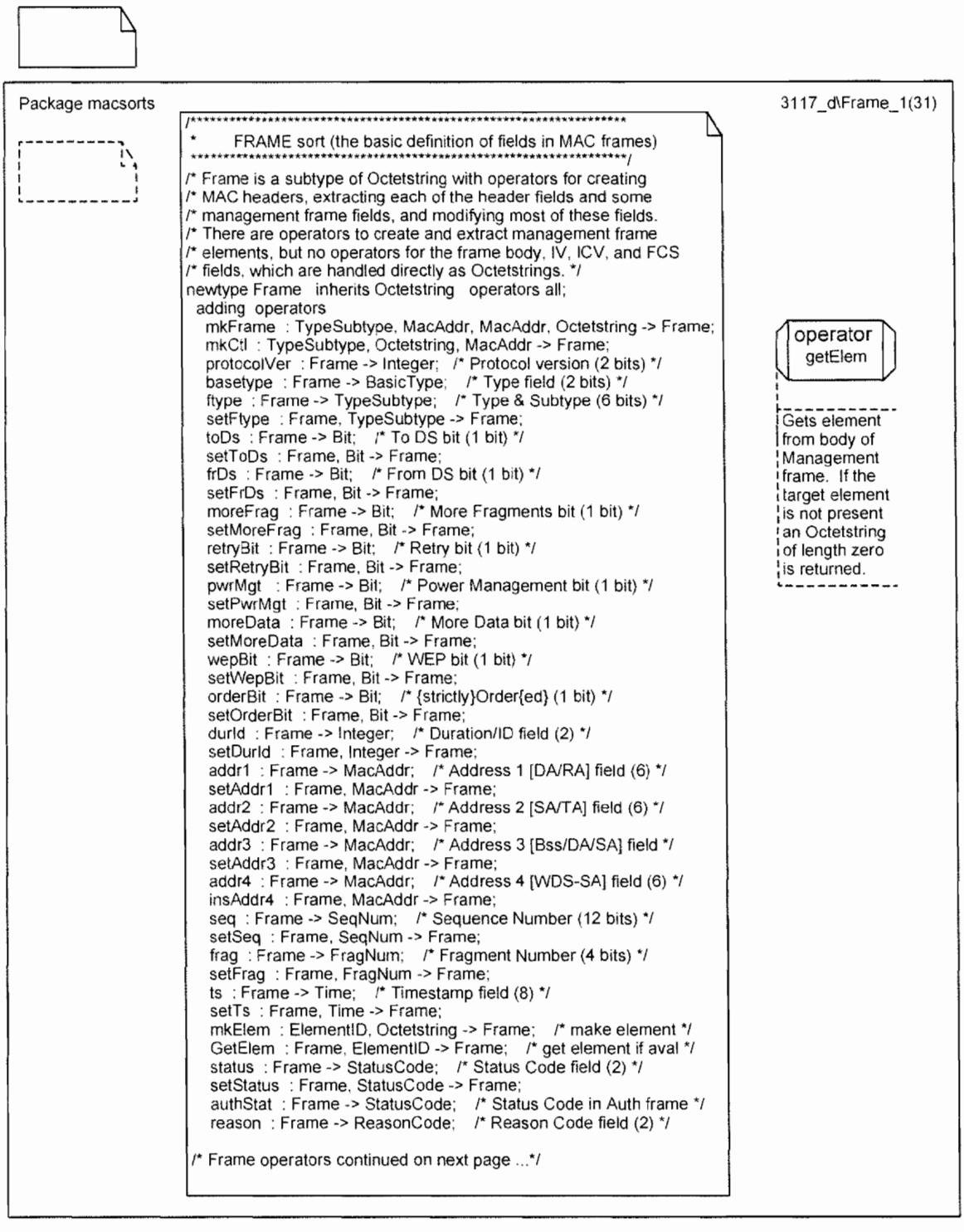
```
/*************************************************************
 *       CRC-32 sorts (for FCS and ICV)
 *************************************************************/
/* Crc is a subtype of Octetstring with added operators: */
/*    crc:= Crc32(crc,octet) */
/* updates the crc value to include the new octet, and */
/*    Mirror(crc), which returns a Crc value with the order */
/* of the octets, and of the bits in each octet, reversed for */
/* MSb-first transmission (see 7.1.1). Crc variables must have */
/* exactly 4 octets, which is done using initCrc or S4. */
newtype Crc  inherits Octetstring  operators all;
  adding  operators
    Crc32  : Crc, Octet -> Crc;
    mirror : Crc -> Octetstring;
  operator Crc32;  fpar crcin Crc, val Octet;  returns Crc;  referenced;
  axioms    for all c in Crc(
     mirror(c) == S4(flip(c(3)),flip(c(2)),flip(c(1)), flip(c(0))); );
endnewtype Crc;
synonym initCrc Crc =    /* Initial Crc value (all 1s) */
          << type Crc>> S4(0xFF,0xFF,0xFF,0xFF);
synonym goodCrc Crc =    /* Unique remainder for valid CRC-32 */
          << type Crc>> S4(0x7B,0xDD,0x04,0xC7);
```

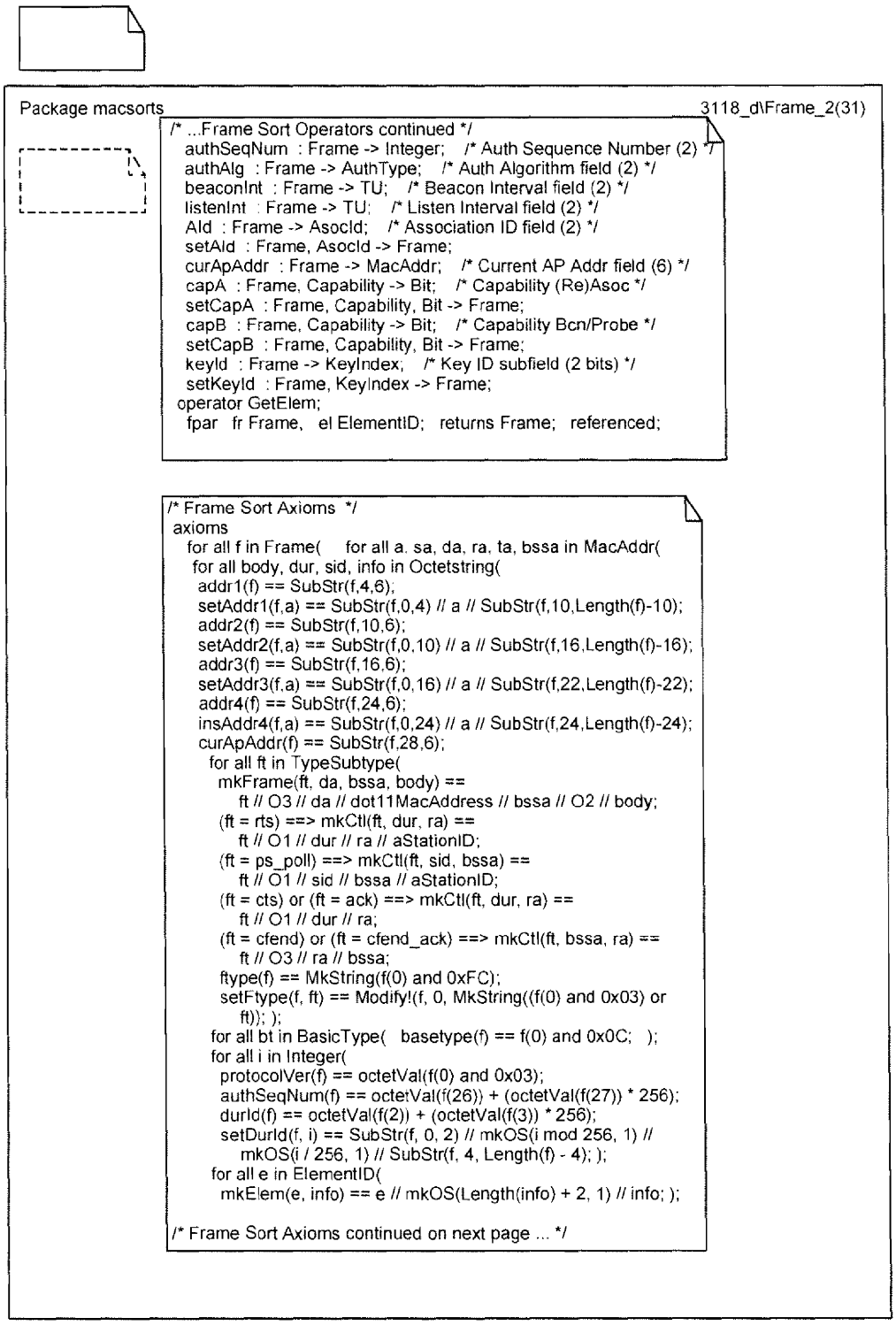
operator
keyLookup

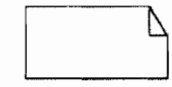
```
/*************************************************************
 *   WEP support sorts
 *************************************************************/
syntype KeyIndex = Integer  constants 0:3  endsyntype KeyIndex;
newtype PrngKey inherits Octetstring  operators all;
  adding literals nullKey;  /* nullKey is not any of 2^40 key values */
  axioms nullKey == null;  default nullKey;  endnewtype PrngKey;
newtype KeyVector  /* vector of default WEP keys */
  Array( KeyIndex, PrngKey);  endnewtype KeyVector;
    /* Number of entries in aWepKeyMappings array at this station.
    /* implementation dependent value, minimum=10 (see 8.3.2). */
synonym sWepKeyMappingLength Integer = 10;
syntype KeyMappingRange = Integer
    constants 1:sWepKeyMappingLength  endsyntype KeyMappingRange;
newtype KeyMap  struct    /* structure used for entries in KeyMapArray */
    mappedAddr   MacAddr;
    wepOn    Boolean;
    wepKey   PrngKey;
endnewtype KeyMap;
/* KeyMapArray -- used for aWepKeyMapping table; */
/* an array of KeyMap indexed by KeyMappingRange, with operator */
/*    KeyMap := keyLookup(addr, keyMapArray, keyMapArrayLength) */
/* returns the KeyMap entry for the specified addr, or */
/* (. nullAddr, false, nullKey .) if no mapping for addr. */
newtype KeyMapArray  Array( KeyMappingRange, KeyMap);
  adding  operators
    keyLookup : MacAddr, KeyMapArray, Integer -> KeyMap;
  operator keyLookup;
    fpar  luadr MacAddr,   kma KeyMapArray,   kml Integer;
    returns KeyMap; referenced;
endnewtype KeyMapArray;
```











Package macsorts 3119_d\Frame_3(31)

```
/*  ... Frame Sort  Axioms continued */
      for all b in Bit(
       toDs(f) == if (f(1) and 0x01) then 1 else 0 fi;
       setToDs(f, b) ==
          Modify!(f, 1, (f(1) and 0xFE) or S8(0,0,0,0,0,0,0,b));
       frDs(f) == if (f(1) and 0x02) then 1 else 0 fi;
       setFrDs(f, b) ==
          Modify!(f, 1, (f(1) and 0xFD) or S8(0,0,0,0,0,0,b,0));
       moreFrag(f) == if (f(1) and 0x04) then 1 else 0 fi;
       setMoreFrag(f, b) ==
          Modify!(f, 1, (f(1) and 0xFB) or S8(0,0,0,0,0,b,0,0));
       retryBit(f) == if (f(1) and 0x08) then 1 else 0 fi;
       setRetryBit(f, b) ==
          Modify!(f, 1, (f(1) and 0xF7) or S8(0,0,0,0,b,0,0,0));
       pwrMgt(f) == if (f(1) and 0x10) then 1 else 0 fi;
       setPwrMgt(f, b) ==
          Modify!(f, 1, (f(1) and 0xFB) or S8(0,0,0,b,0,0,0,0));
       moreData(f) == if (f(1) and 0x20) then 1 else 0 fi;
       setMoreData(f, b) ==
          Modify!(f, 1, (f(1) and 0xFB) or S8(0,0,b,0,0,0,0,0));
       wepBit(f) == if (f(1) and 0x40) then 1 else 0 fi;
       setWepBit(f, b) ==
          Modify!(f, 1, (f(1) and 0xFB) or S8(0,b,0,0,0,0,0,0));
       orderBit(f) == if (f(1) and 0x80) then 1 else 0 fi;
       setOrderBit(f, b) ==
          Modify!(f, 1, (f(1) and 0xFB) or S8(b,0,0,0,0,0,0,0));
       for all c in Capability(
        capA(f,c) == if (B_S(SubStr(f,24,2)) and c) then 1 else 0 fi;
        setCapA(f,c,b) == SubStr(f,0,24) // (B_S(SubStr(f,24,2) and
           (not c)) or (if b then c else O2 fi)) //
           SubStr(f,26,Length(f) - 26);
       capB(f,c) == if (B_S(SubStr(f,34,2)) and c) then 1 else 0 fi;
        setCapB(f,c,b) == SubStr(f,0,34) // (B_S(SubStr(f,34,2) and
           (not c)) or (if b then c else O2 fi)) //
           SubStr(f,36,Length(f) - 36); ));
      for all sq in SeqNum(
       seq(f) == (octetVal(f(22) and 0xF0)/16)+(octetVal(f(23)*16));
       setSeq(f, sq) == SubStr(f, 0, 22) // MkString((f(22) and 0x0F)
          or mkOctet((sq mod 16) * 16)) // mkOS(sq / 16, 1) //
          SubStr(f, 24, Length(f) - 24); );
      for all fr in FragNum(
       frag(f) == octetVal(f(22) and 0x0F);
       setFrag(f, fr) ==
          SubStr(f, 0, 22) // MkString((f(22) and 0xF0) or
          mkOctet(fr)) // SubStr(f, 23, Length(f) - 23); );
      for all tm in Time(
       ts(f) == tUsec( Usec!(octetVal(f(24)) +
          (256 * (octetVal(f(25)) +
           (256 * (octetVal(f(26)) +
            (256 * (octetVal(f(27)) +
             (256 * (octetVal(f(28)) +
              (256 * (octetVal(f(29)) +
               (256 * (octetVal(f(30)) +
                (256 * octetVal(f(31))))))))))))))) ) );

/*  Frame Sort Axioms continued on next page ... */
```

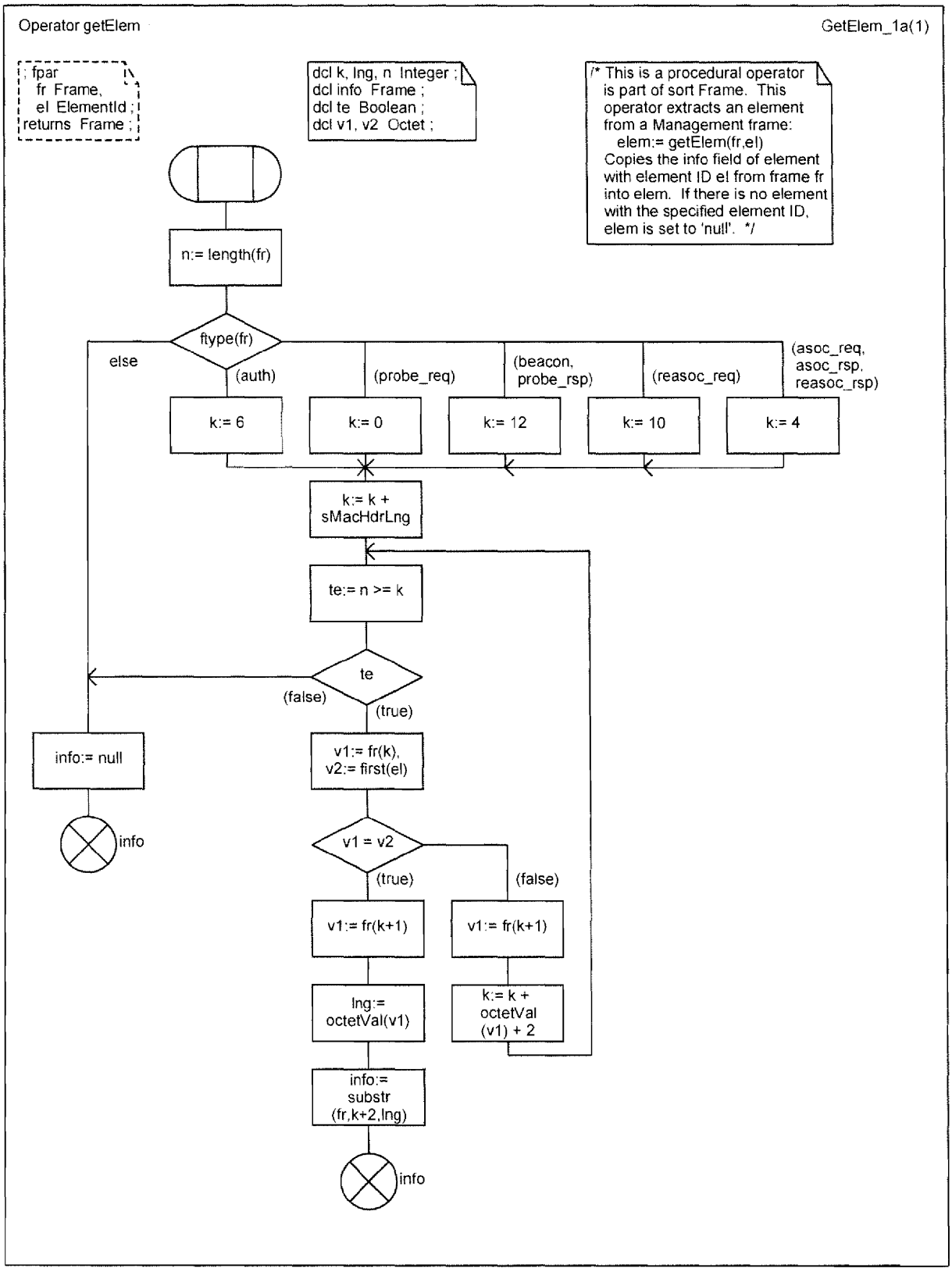
Package macsorts

3120_d\Frame_4(31)

```
/* ... Frame Sort Axioms continued */
      setTs(f, tm) == SubStr(f, 0, 24) // mkOS(fix(tm), 1) //
         mkOS((fix(tm) / 256), 1) // mkOS((fix(tm) / 65536), 1) //
         mkOS((fix(tm) / 16777216), 1) //
         mkOS((fix(tm) / 4294967296), 1) //
         mkOS(((fix(tm) / 4294967296) / 256), 1) //
         mkOS(((fix(tm) / 4294967296) / 65536), 1) //
         mkOS(((fix(tm) / 4294967296) / 16777216), 1) //
         SubStr(f, 32, Length(f) - 32); );
    for all stat in StatusCode(
      status(f) == SubStr(f, 26, 2);
      setStatus(f, stat) ==
         SubStr(f, 0, 26) // stat // SubStr(f, 28, Length(f) - 28);
      authStat(f) == SubStr(f, 28, 2); );
    for all rea in ReasonCode( reason(f) == SubStr(f, 24, 2); );
    for all alg in AuthType( AuthType(f) == SubStr(f, 24, 2); );
    for all u in TU(
      beaconInt(f) == octetVal(f(32)) + (octetVal(f(33)) * 256);
      listenInt(f) == octetVal(f(26)) + (octetVal(f(27)) * 256); );
    for all sta in AssocId(
      AId(f) == octetVal(f(28)) + (octetVal(f(29)) * 256);
      setAId(f, sta) == SubStr(f, 0, 28) // mkOS(sta mod 256, 1) //
         mkOS(sta / 256, 1) // SubStr(f, 30, Length(f) - 30); );
    for all kid in KeyIndexRange(
      keyId(f) == octetVal(f(27)) / 64;
      setKeyId(f, kid) == Modify!(f, 27, mkOS(kid * 64)); ); ));
endnewtype Frame;
```

```
/*************************************************************
 *      ReasonCode sort
 *************************************************************/
newtype ReasonCode inherits Octetstring operators all;
 adding literals unspec_reason, auth_not_valid, deauth_lv_ss,
   inactivity, ap_overload, class2_err, class3_err,
   disas_lv_ss, asoc_not_auth;
 axioms
   unspec_reason == mkOS(1, 2);   auth_not_valid == mkOS(2, 2);
   deauth_lv_ss == mkOS(3, 2);   inactivity == mkOS(4, 2);
   ap_overload == mkOS(5, 2);   class2_err == mkOS(6, 2);
   class3_err == mkOS(7, 2);   disas_lv_ss == mkOS(8, 2);
   asoc_not_auth == mkOS(9, 2);
endnewtype ReasonCode;
```

```
/*************************************************************
 *      StatusCode sort
 *************************************************************/
newtype StatusCode inherits Octetstring operators all;
 adding literals successful, unspec_fail, unsup_cap,
   reasoc_no_asoc, fail_other, unsupt_alg, auth_seq_fail,
   chlng_fail, auth_timeout, ap_full, unsup_rate;
 axioms
   successful == mkOS(0, 2);   unspec_failure == mkOS(1, 2);
   unsup_cap == mkOS(10, 2);   reasoc_no_asoc == mkOS(11, 2);
   fail_other == mkOS(12, 2);   unsupt_alg == mkOS(13, 2);
   auth_seq_fail == mkOS(14, 2);   chlng_fail == mkOS(15, 2);
   auth_timeout == mkOS(16, 2);   ap_full == mkOS(17, 2);
   unsup_rate == mkOS(18, 2);
endnewtype StatusCode;
```



Package macsorts

3121_d\FrameType(31)

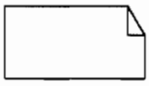
```
/*****************************************************************
*       Frame Type sorts
*****************************************************************/
/* TypeSubtype defines the full, 6-bit frame type identifiers. */
/* These values are useful with ftype operator of Frame sort. */
newtype TypeSubtype  inherits Octetstring  operators all;
 adding literals asoc_req, asoc_rsp, reasoc_req, reasoc_rsp,
   probe_req, probe_rsp, beacon, atim, disasoc, auth, deauth,
   ps_poll, rts, cts, ack, cfend, cfend_ack, data, data_ack,
   data_poll, data_poll_ack, null_frame, cfack, cfpoll, cfpoll_ack;
 axioms
   asoc_req == MkString(S8(0,0,0,0,0,0,0,0));
   asoc_rsp == MkString(S8(0,0,0,0,1,0,0,0));
   reasoc_req == MkString(S8(0,0,0,0,0,1,0,0));
   reasoc_rsp == MkString(S8(0,0,0,0,0,1,1,0,0));
   probe_req == MkString(S8(0,0,0,0,0,0,0,1,0));
   probe_rsp == MkString(S8(0,0,0,0,0,1,0,1,0));
   beacon == MkString(S8(0,0,0,0,0,0,0,0,1));
   atim == MkString(S8(0,0,0,0,0,1,0,0,1));
   disasoc == MkString(S8(0,0,0,0,0,0,1,0,1));
   auth == MkString(S8(0,0,0,0,0,1,1,0,1));
   deauth == MkString(S8(0,0,0,0,0,0,0,1,1));
   ps_poll == MkString(S8(0,0,1,0,0,1,0,1));
   rts == MkString(S8(0,0,1,0,1,1,0,1));
   cts == MkString(S8(0,0,1,0,0,0,1,1));
   ack == MkString(S8(0,0,1,0,1,0,1,1));
   cfend == MkString(S8(0,0,1,0,0,1,1,1));
   cfend_ack == MkString(S8(0,0,1,0,1,1,1,1));
   data == MkString(S8(0,0,0,1,0,0,0,0));
   data_ack == MkString(S8(0,0,0,1,1,0,0,0));
   data_poll == MkString(S8(0,0,0,1,0,1,0,0));
   data_poll_ack == MkString(S8(0,0,0,1,1,1,0,0));
   null_frame == MkString(S8(0,0,0,1,0,0,1,0));
   cfack == MkString(S8(0,0,0,1,1,0,1,0));
   cfpoll == MkString(S8(0,0,0,1,0,1,1,0));
   cfpoll_ack == MkString(S8(0,0,0,1,1,1,1,0));
endnewtype TypeSubtype;
/* BasicTypes defines the 2-bit frame type groups */
newtype BasicType  inherits Bitstring  operators all;
 adding literals control, data, management, reserved;
 axioms
   control == S8(0,0,1,0,0,0,0,0);   data == S8(0,0,0,1,0,0,0,0);
   management == S8(0,0,0,0,0,0,0,0);   reserved == S8(0,0,1,1,0,0,0,0);
endnewtype BasicType;
```

Package macsorts 3122_d\MgmtFields(31)

```
/*************************************************************
 *      ElementID sort
 *************************************************************/
newtype ElementID  inherits Octetstring  operators all;
 adding literals eSsId, eSupRates, eFhParms, eDsParms,
  eCfParms, eTim, eIbParms, eCtext;
 axioms
  eSsId == mkOS(0, 1); /* service set identifier (0:32) */
  eSupRates == mkOS(1, 1); /* supported rates (1:8) */
  eFhParms == mkOS(2, 1); /* FH parameter set (5) */
  eDsParms == mkOS(3, 1); /* DS parameter set (1) */
  eCfParms == mkOS(4, 1); /* CF parameter set (6) */
  eTim == mkOS(5, 1); /* Traffic Information Map (4:254) */
  eIbParms == mkOS(6, 1); /* IBSS parameter set (2) */
  eCtext == mkOS(16, 1); /* challenge text (128, see 8.1.2.2) */
endnewtype ElementID;
```

```
/*************************************************************
 *      Capability field bit assignments sort
 *************************************************************/
newtype Capability  inherits Bitstring  operators all;
 adding literals cEss, cIbss, cPollable, cPollReq, cPrivacy;
 axioms
  cEss == S8(1,0,0,0,0,0,0,0) // 0x00;   /* ESS capability */
  cIbss == S8(0,1,0,0,0,0,0,0) // 0x00;   /* IBSS capability */
  cPollable == S8(0,0,1,0,0,0,0,0) // 0x00;   /* CF-pollable (sta), PC present (ap) */
  cPollReq == S8(0,0,0,1,0,0,0,0) // 0x00;   /* not CF poll req (sta), PC polls (ap) */
  cPrivacy == S8(0,0,0,0,1,0,0,0) // 0x00;   /* WEP required */
endnewtype Capability;
```

```
/*************************************************************
 *      IBSS parameter set sort
 *************************************************************/
newtype IbssParms  inherits Octetstring  operators all;
 adding operators
  atimWin : IbssParms -> TU;
  setAtimWin : IbssParms, TU -> IbssParms;
 axioms
  for all ib in IbssParms(    for all u in TU(
     atimWin(ib) == octetVal(ib(0)) + (octetVal(ib(1)) * 256);
     setAtimWin(ib, u) == mkOS(u mod 256, 1) // mkOS(u / 256, 1); ));
endnewtype IbssParms;
```

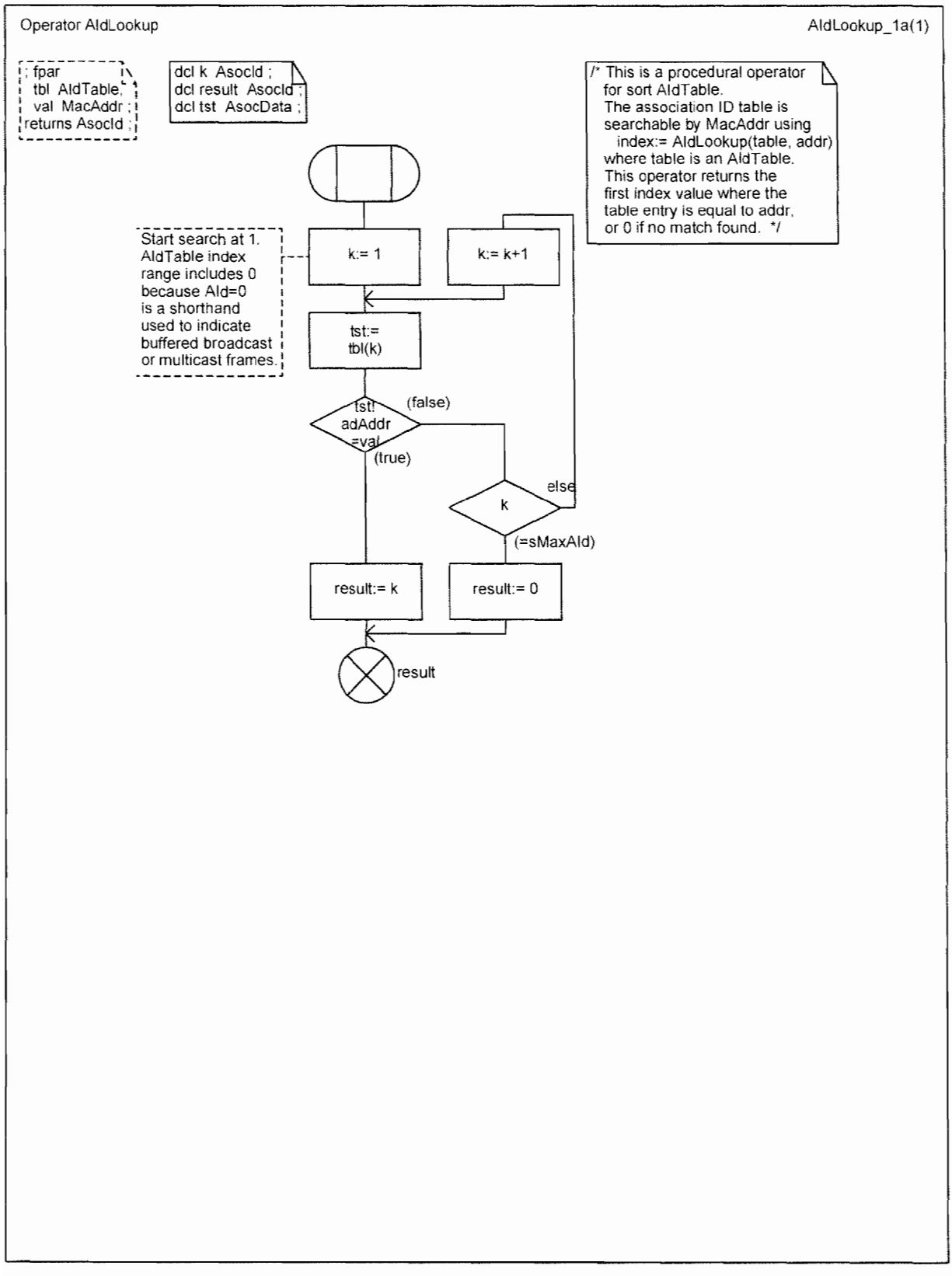


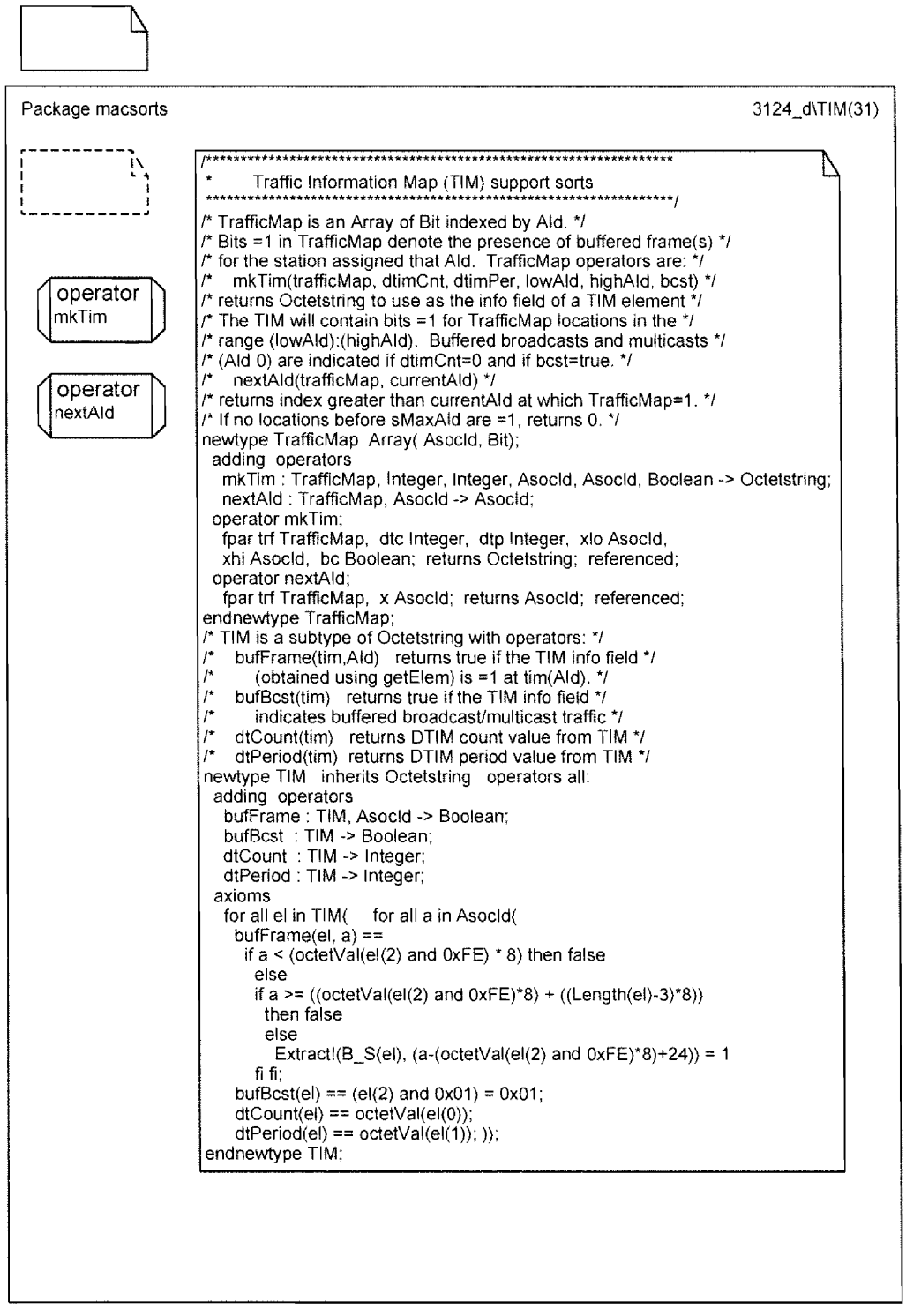
Package macsorts 3123_d\CF_And_AsocParams(31)

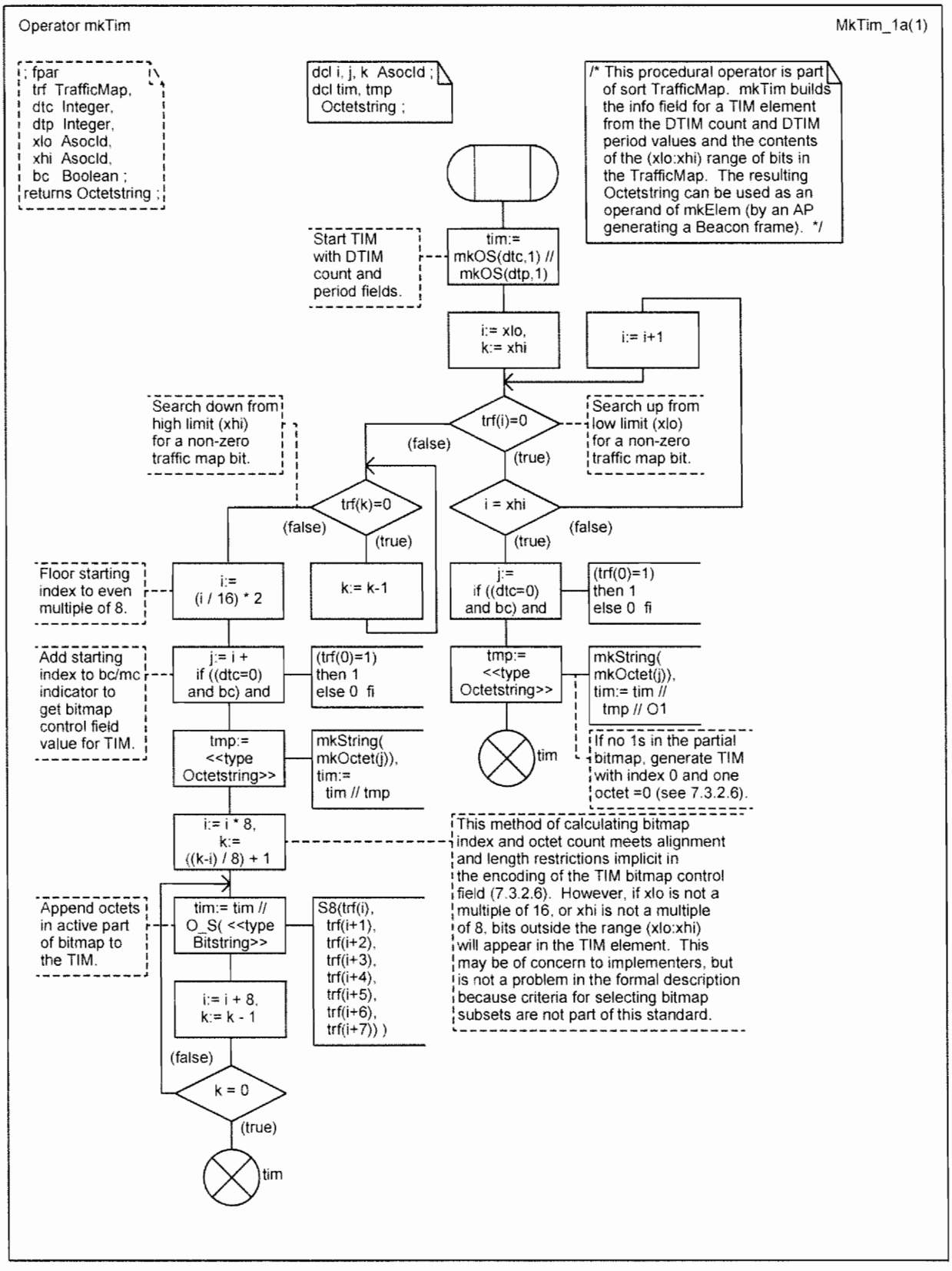
```
/*****************************************************************
 *       CF parameter set sort
 *****************************************************************/
newtype CfParms   inherits Octetstring   operators all;
 adding  operators
   cfpCount  : CfParms -> Integer; /* CfpCount field (1) */
   setCfpCount  : CfParms, Integer -> CfParms;
   cfpPeriod  : CfParms -> Integer; /* CfpPeriod field (1) */
   setCfpPeriod  : CfParms, Integer -> CfParms;
   cfpMaxDur  : CfParms -> TU;    /* CfpMaxDuration field (2) */
   setCfpMaxDur  : CfParms, TU -> CfParms;
   cfpDurRem  : CfParms -> TU;    /* CfpDurRemaining field (2) */
   setCfpDurRem  : CfParms, TU -> CfParms;
 axioms   for all cf in CfParms(   for all i in Integer(   for all u in TU(
       cfpCount(cf) == octetVal(cf(0));
       setCfpCount(cf, i) == mkOS(i, 1) // Tail(cf);
       cfpPeriod(cf) == octetVal(cf(1));
       setCfpPeriod(cf, i) == cf(0) // mkOS(i, 1) // SubStr(cf,2,4);
       cfpMaxDur(cf) == octetVal(cf(2)) + (octetVal(cf(3)) * 256);
       setCfpMaxDur(cf, u) == SubStr(cf, 0, 2) // mkOS(u mod 256, 1)
          // mkOS(u / 256, 1) // SubStr(cf, 4, 2);
       cfpDurRem(cf) == octetVal(cf(4)) + (octetVal(cf(5)) * 256);
       setCfpDurRem(cf, u) == SubStr(cf, 0, 4) // mkOS(u mod 256, 1)
          // mkOS(u / 256, 1); )));
endnewtype CfParms;
```

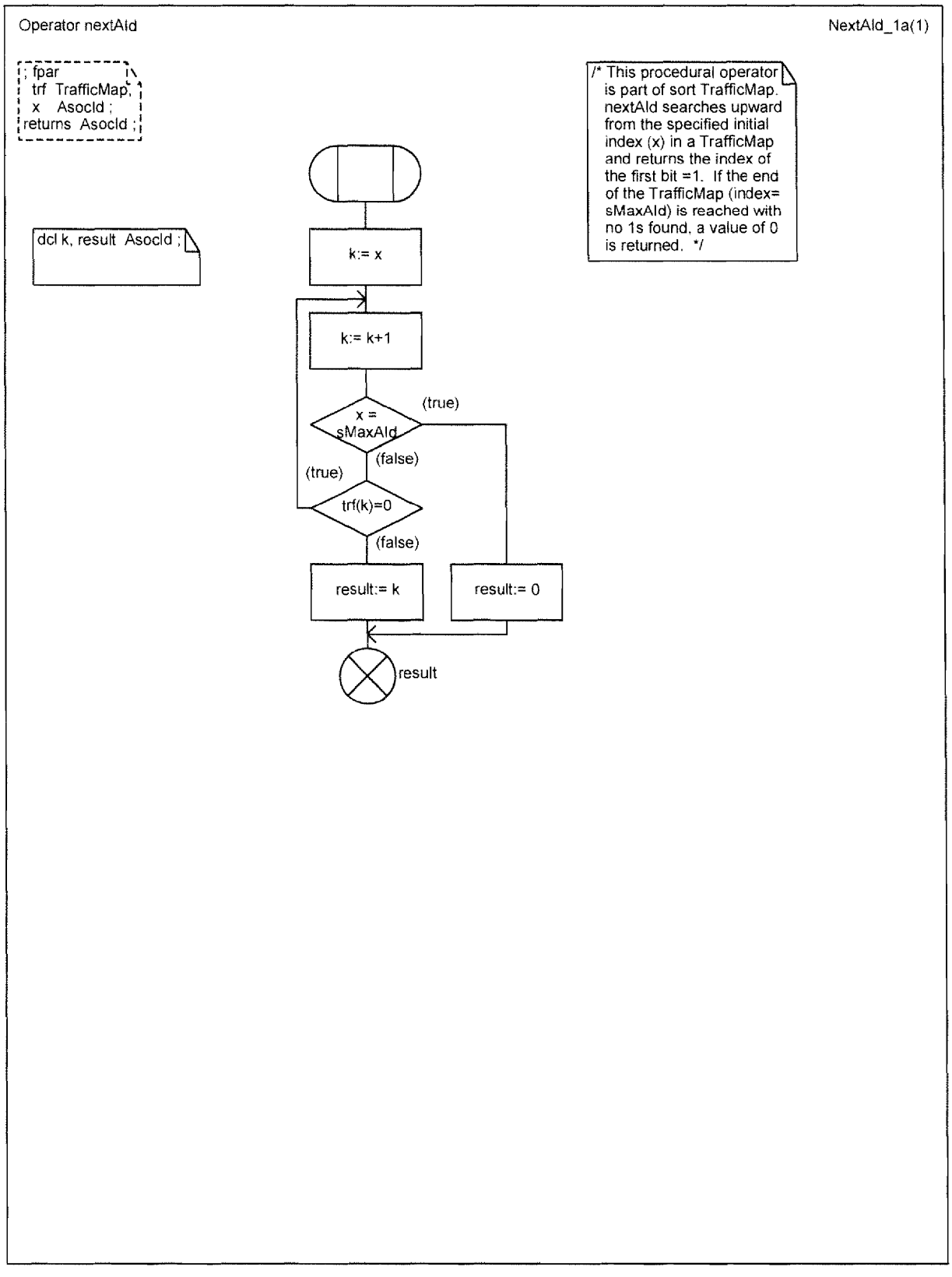
operator
AIdLookup

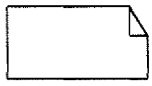
```
/*****************************************************************
 *       Sorts for association management at AP
 *****************************************************************/
synonym sMaxAId Integer = 2007;  /* 2007 is largest allowable value */
                        /* implementation limit may be lower */
syntype AsocId = Integer   constants 0:sMaxAId   endsyntype AsocId;
      /* Station Association Record -- only used at APs */
newtype AsocData struct
   adAddr    MacAddr;    /* address of associated station */
   adPsm     PwrSave;    /* power save mode of the station */
   adCfPoll  Boolean;    /* true if station is CfPollable */
   adPollRq  Boolean;    /* true if station requested polling */
   adNoPoll  Boolean;    /* true if station requested no polling */
   adMsduIP  Boolean;    /* true if partial Msdu outstanding to sta */
   adAuth    AuthType;   /* authentication type used by station */
   adRates   RateSet;    /* supported rates from association request */
   adAge     Time;    /* time of association */
endnewtype AsocData;
/* Association table -- array of AsocData, only used at APs */
/*    index:= AIdLookup(table, addr) */
/* returns the index of location where table(x)!adAddr=addr */
/* or 0 if no such location found. */
newtype AIdTable   Array(AsocId, AsocData);
 adding  operators
   AIdLookup : AIdTable, MacAddr -> AsocId;
 operator AIdLookup;
   fpar tbl AIdTable,   val MacAddr;   returns AsocId;   referenced;
endnewtype AIdTable;
```







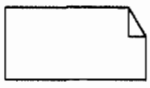




Package macsorts 3125_d\RateAndDurationSorts(31)

```
/*****************************************************************
 *       Multi-rate support sorts
 *****************************************************************/
newtype Rate   inherits Octet   operators all;
 adding  operators
   calcDur : Rate, Integer -> Integer; /* converts (rate,bitCount) to integer usec */
   rateVal : Rate -> Rate; /* clears high-order bit */
   basicRate : Rate -> Rate; /* sets high-order bit */
   isBasic : Rate -> Boolean; /* true if high-order bit set */
 axioms
   for all r in Rate(    for all i in Integer(    for all b in Boolean(
       calcDur(r, i) == ((((10000000 + (octetVal(r and 0x7F) - 1)) /
           (500 * octetVal(r and 0x7F))) * i) + 9999) / 10000;
       rateVal(r) == r and 0x7F;          basicRate(r) == r or 0x80;
       isBasic(r) == (r and 0x80) = 0x80; )));
endnewtype Rate;
syntype RateString = Octetstring   endsyntype RateString;
```

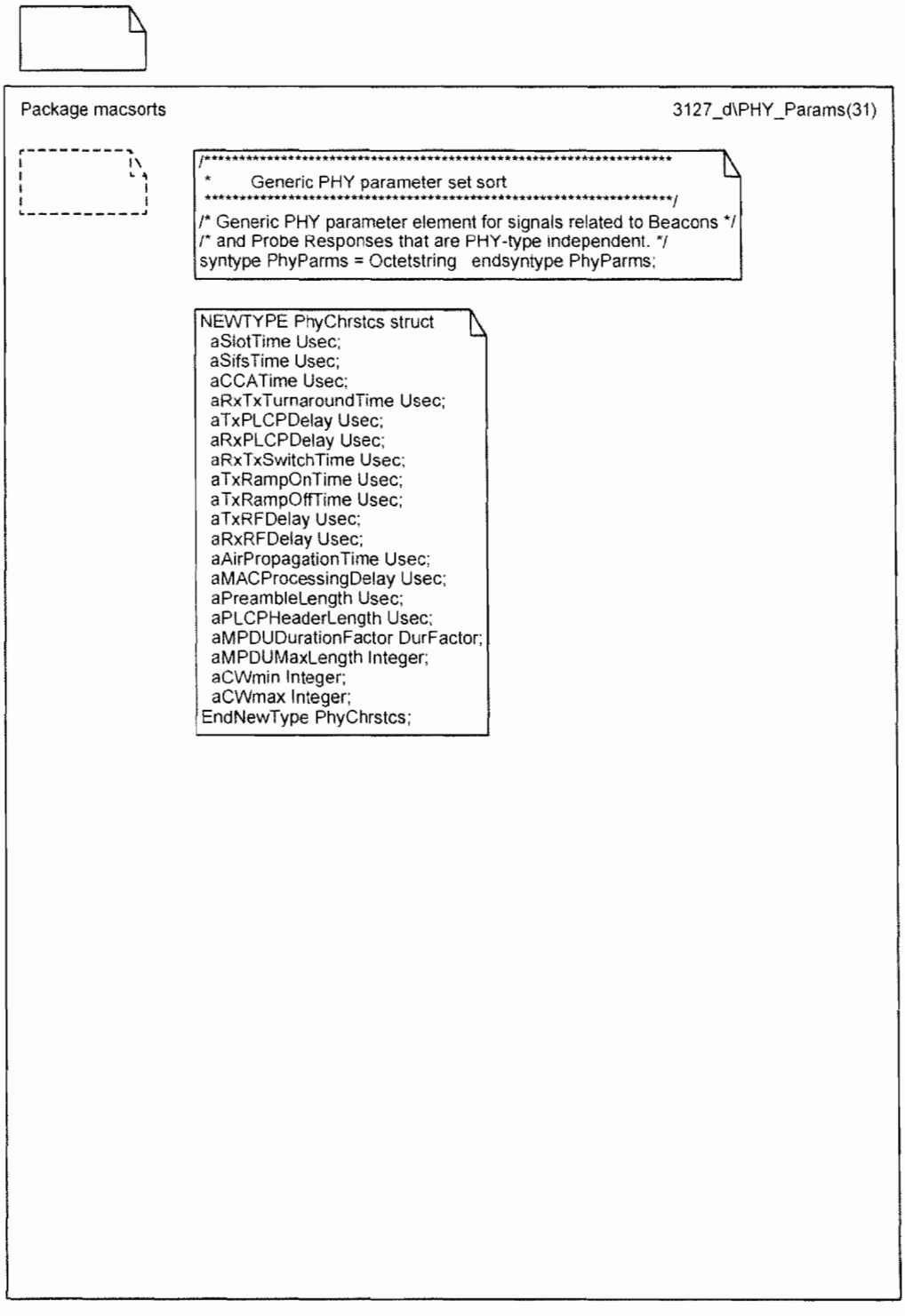
```
/*****************************************************************
 *       MPDU duration factor support sort
 *****************************************************************/
/* These operators support the encoding used to allow */
/* an Integer to represent the value of aMpduDurationFactor. */
/*    calcDF(PlcpBits, MpduBits)  returns an Integer which is */
/* the fractional part of ((PlcpBits/MpduBits)-1)*(1e9). */
/*    stuff(durFactor, MpduBits)  returns the number of PlcpBits */
/* which result from MpduBits at the specified durFactor. */
newtype DurFactor   inherits Integer   operators all;
 adding  operators
   calcDF : Integer, Integer -> DurFactor;
   stuff  : DurFactor, Integer -> Integer;
 axioms
   for all df in DurFactor(     for all mb, pb in Integer(
       calcDF(pb, mb) == ((pb * 1000000000) / mb) - 1000000000;
       stuff(df, mb) == ((mb * df) + (mb - 1)) / 1000000000; ));
endnewtype DurFactor;
```

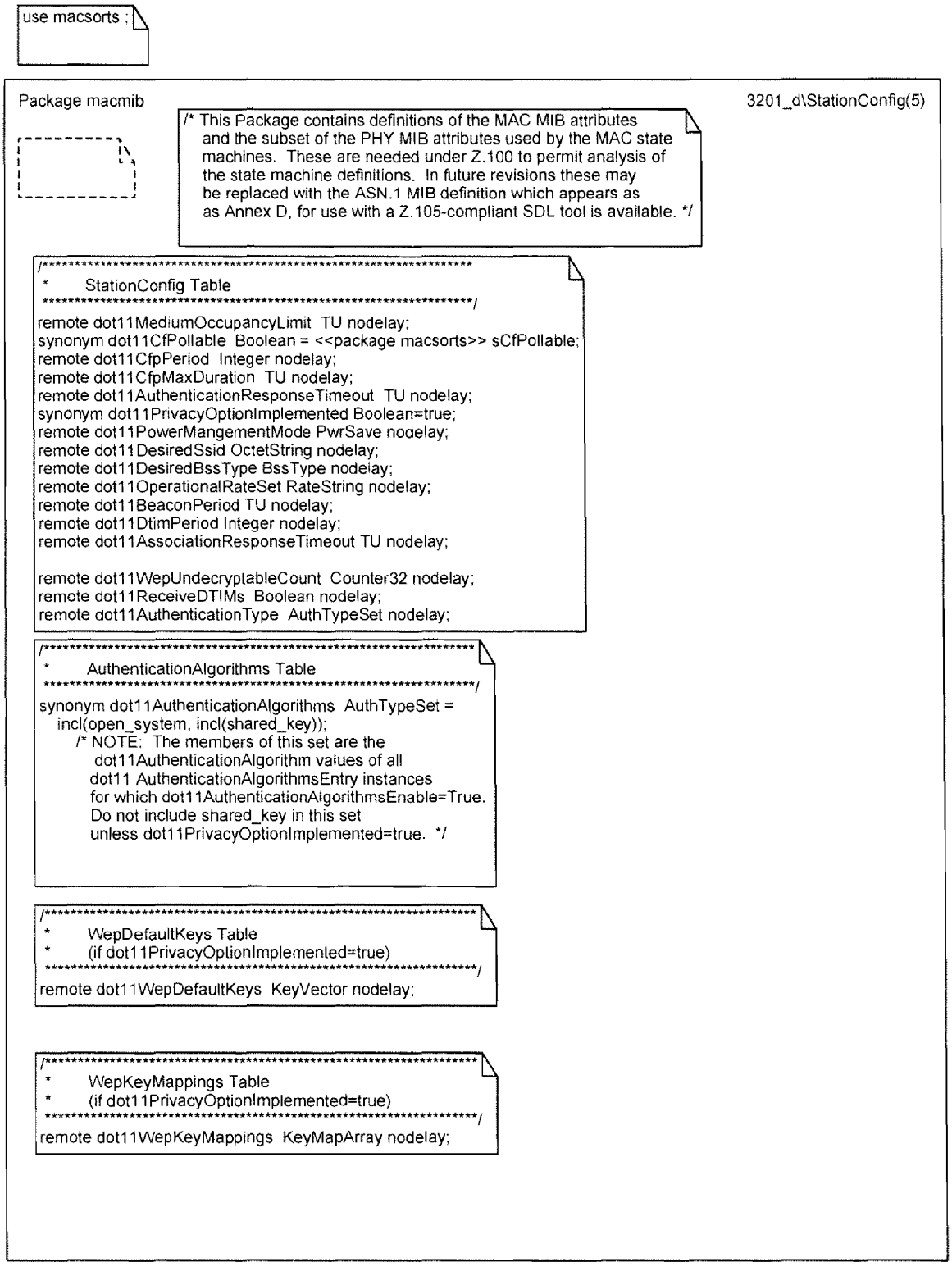


Package macsorts 3126_d\FH_DS_Params(31)

```
/*****************************************************************
 *       FH parameter set sort
 *****************************************************************/
newtype FhParms   inherits Octetstring   operators all;
 adding  operators
   dwellTime  : FhParms -> TU;    /* Dwell Time field (2) */
   setDwellTime  : FhParms, TU -> FhParms;
   hopSet  : FhParms -> Integer; /* Hop Set field (1) */
   setHopSet  : FhParms, Integer -> FhParms;
   hopPattern  : FhParms -> Integer; /* Hop Pattern field (1) */
   setHopPattern  : FhParms, Integer -> FhParms;
   hopIndex  : FhParms -> Integer; /* Hop Index field (1) */
   setHopIndex  : FhParms, Integer -> FhParms;
 axioms
   for all fh in FhParms(   for all i in Integer(   for all u in TU(
     dwellTime(fh) == octetVal(fh(0)) + (octetVal(fh(1)) * 256);
     setDwellTime(fh, u) == mkOS(u mod 256, 1) // mkOS(u / 256, 1) // SubStr(fh, 2, 3);
     hopSet(fh) == octetVal(fh(2));
     setHopSet(fh,i) == SubStr(fh,0,2) // mkOS(i,1) // SubStr(fh,3,2);
     hopPattern(fh) == octetVal(fh(3));
     setHopPattern(fh, i) == SubStr(fh,0,3) // mkOS(i,1) // Last(fh);
     hopIndex(fh) == octetVal(fh(4));
     setHopIndex(fh, i) == SubStr(fh, 0, 4) // mkOS(i, 1);)));
endnewtype FhParms;
```

```
/*****************************************************************
 *       DS parameter set sort
 *****************************************************************/
newtype DsParms   inherits Octetstring   operators all;
 adding  operators
   curChannel  : DsParms -> Integer; /* Current Channel (1) */
   setCurChannel  : DsParms, Integer -> DsParms;
 axioms
   for all ds in DsParms(     for all i in Integer(
      curChannel(ds) == octetVal(ds(0));
      setCurChannel(ds, i) == mkOS(i); ));
endnewtype DsParms;
```





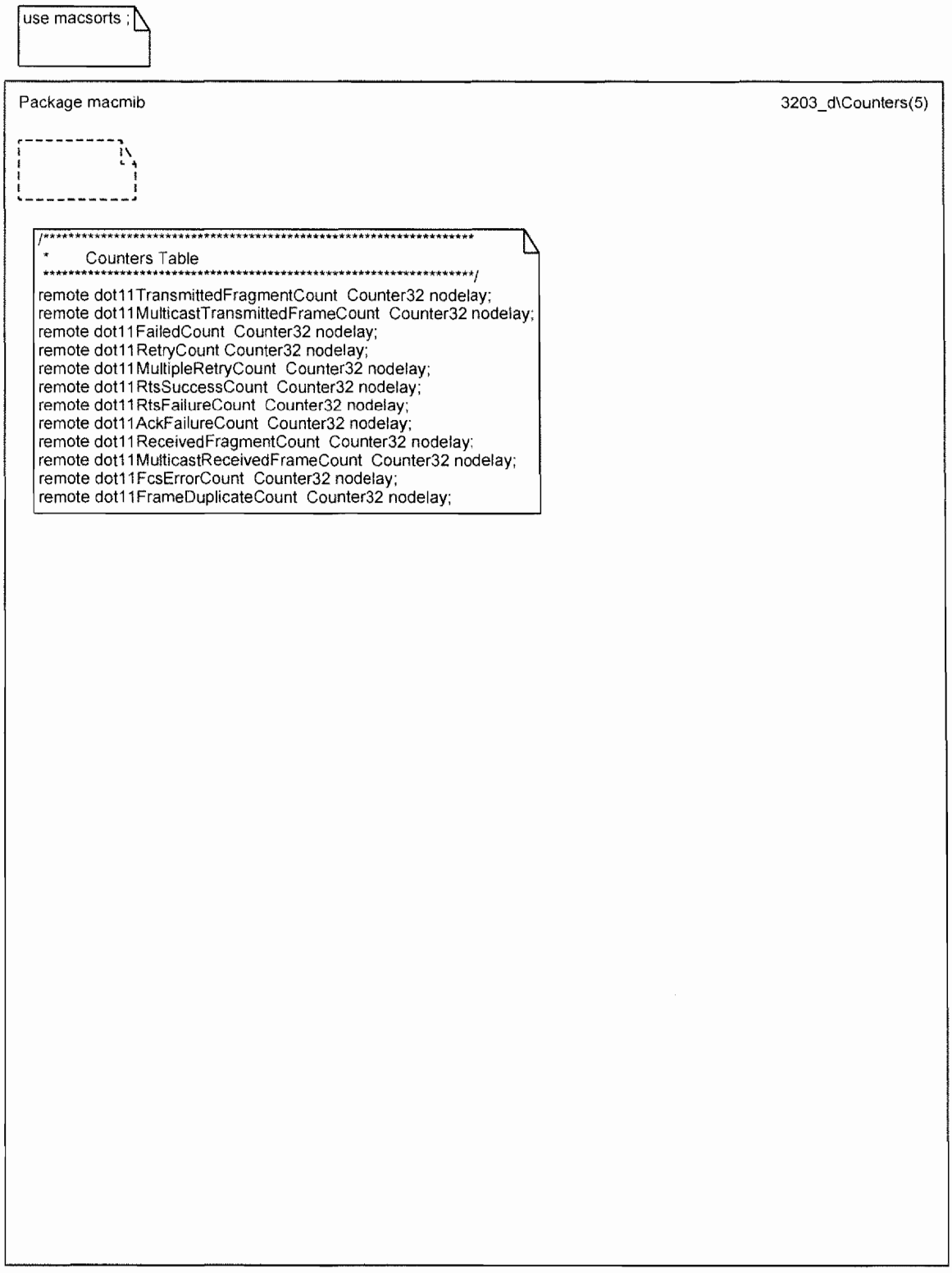
```
use macsorts ;
```

Package macmib 3202_d\PrivOperation(5)

```
/*****************************************************************
 *      Privacy Table
 *      (only if dot11PrivacyOptionImplemented=true)
 *****************************************************************/
remote dot11PrivacyInvoked  Boolean nodelay;
remote dot11WepDefaultKeyId  KeyIndex nodelay;
synonym dot11WepKeyMappingLength  Integer =
   <<package macsorts>> sWepKeyMappingLength;
remote dot11ExcludeUnencrypted  Boolean nodelay;
remote dot11WepIcvErrorCount  Counter32 nodelay;
remote dot11WepExcludedCount  Counter32 nodelay;
```

```
/*****************************************************************
 *      Operation Table
 *****************************************************************/
synonym dot11MacAddress MacAddr =
  <<type MacAddr>> S6(0x00, 0x11, 0x22, 0x33, 0x44, 0x55);
   /* each station has a unique globally administered address */
   /* Value may be overwritten with locally administered address at */
   /* MlmeReset, but is always a static value during MAC operation */
remote dot11RtsThreshold  Integer nodelay;
remote dot11ShortRetryLimit  Integer nodelay;
remote dot11LongRetryLimit  Integer nodelay;
remote dot11FragmentationThreshold  Integer nodelay;
remote dot11MaxTransmitMsduLifetime  TU nodelay;
remote dot11MaxReceiveLifetime  TU nodelay;
synonym dot11ManufacturerId  Charstring = 'name of manufacturer';
synonym dot11ProductId  Charstring = 'identifier unique to manufacturer';
```

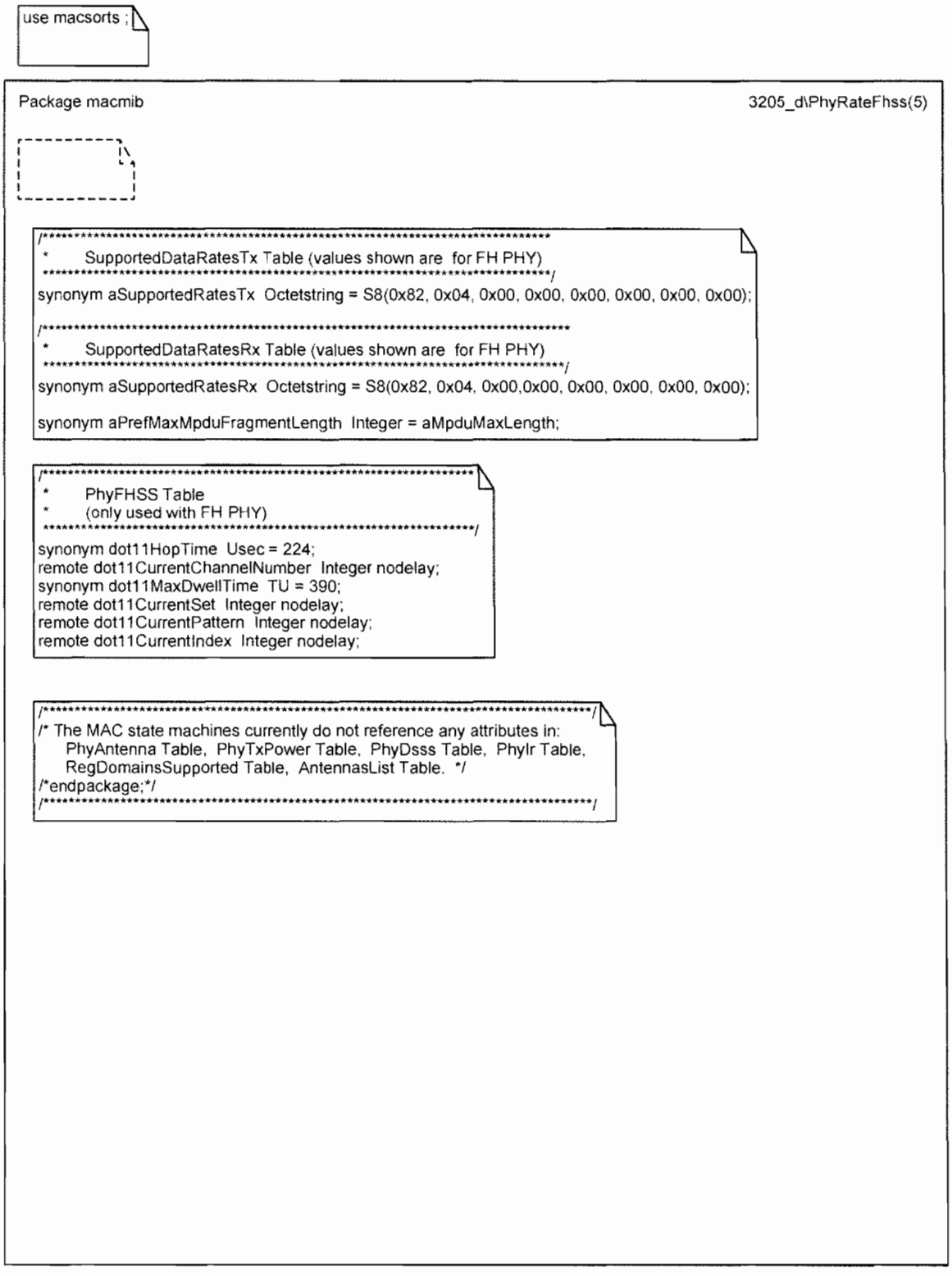
```
/*****************************************************************
 *      GroupAddresses Table
 *****************************************************************/
remote dot11GroupAddresses  MacAddrSet nodelay;
```



use macsorts ;

Package macmib 3204_d\PhyOperation(5)

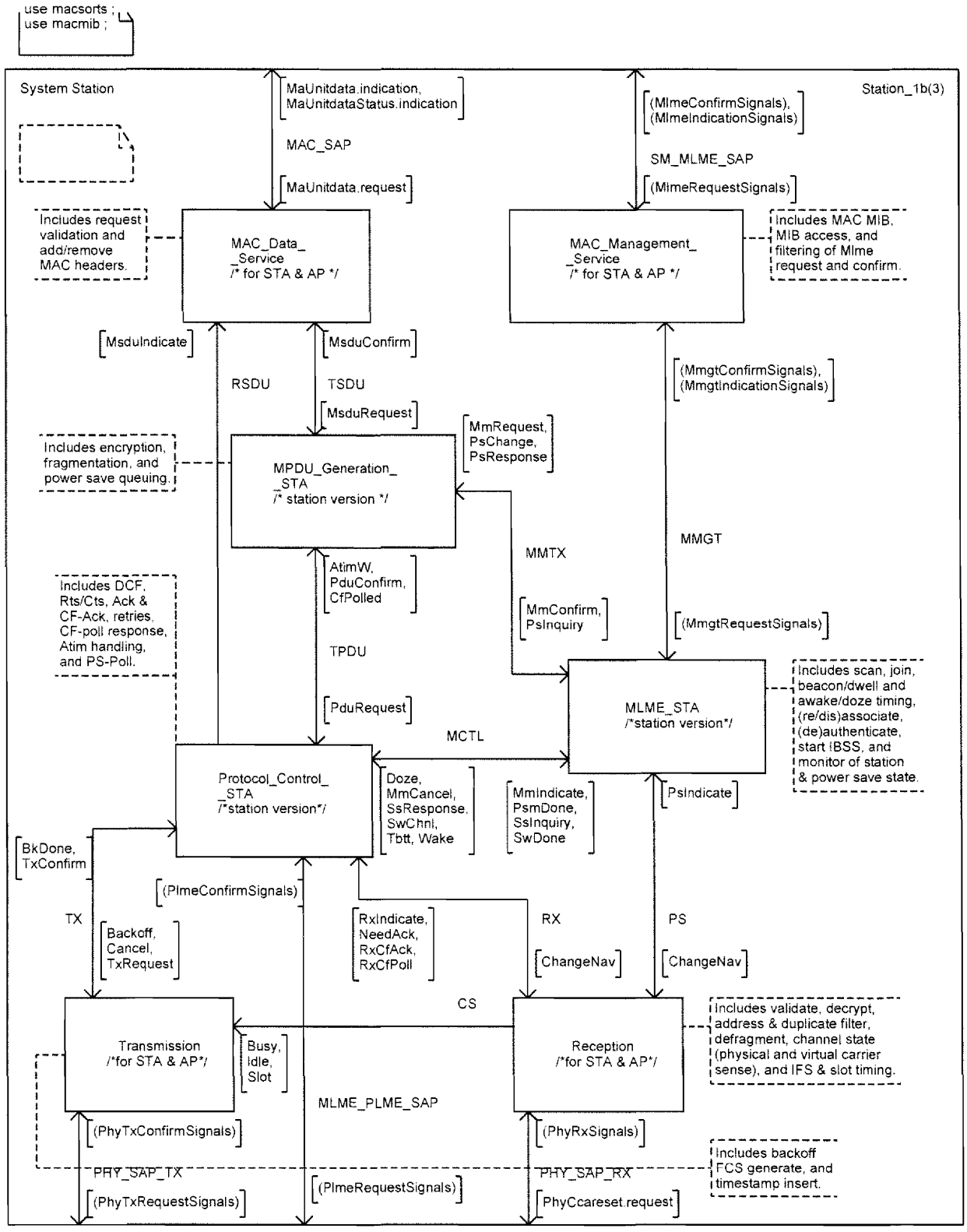
```
/*****************************************************************
 *      PhyOperation Table
 *      (values shown are mostly for FH PHY)
 *****************************************************************/
synonym FHphy Integer = 01; /* enumerated dot11PHYType value */
synonym DSphy Integer = 02; /* enumerated dot11PHYType value */
synonym IRPhy Integer = 03; /* enumerated dot11PHYType value */
synonym dot11PHYType  Integer = FHphy;
remote dot11CurrentRegDomain  Integer nodelay;
synonym dot11TempType Integer = 01;

/*****************************************************************************
* PhyCharacteristic Parameters (values shown are mostly for FH PHY )
*****************************************************************************/
/* NOTE:  The PhyCharacteristics are defined as synonyms because
   their values are static during MAC operation.  It is assumed
   that , during each initialization of MAC operation, current
   values for each of these parameters are obtained from the
   PHY using the PlmeCharacteristics primitive. */
synonym aSlotTime  Usec = (aCcaTime + aRxTxTurnaroundTime +
   aAirPropagationTime + aMacProcessingTime);
synonym aCcaTime  Usec = 27;
synonym aRxTxTurnaroundTime Usec = (aTxPlcpDelay + aRxTxSwitchTime +
   aTxRampOnTime + aTxRfDelay);
synonym aTxPlcpDelay  Usec = 1;
synonym aRxTxSwitchTime  Usec = 10;
synonym aTxRampOnTime  Usec = 8;
synonym aTxRfDelay  Usec = 1;
synonym aSifsTime Usec = (aRxRfDelay + aRxPlcpDelay +
   aMacProcessingTime + aRxTxTurnaroundTime);
synonym aRxRfDelay  Usec = 4;
synonym aRxPlcpDelay  Usec = 2;
synonym aMacProcessingTime  Usec = 2;
synonym aTxRampOffTime  Usec = 8;
synonym aPreambleLength  Usec = 96;
synonym aPlcpHeaderLength  Usec = 32;
synonym aMpduDurationFactor  <<package macsorts>> DurFactor = 31250000;
synonym aMpduMaxLength Integer = 4095;
synonym aAirPropagationTime  Usec = 1;
synonym aCWmax  Integer = 1023;
synonym aCWmin  Integer = 15;
```

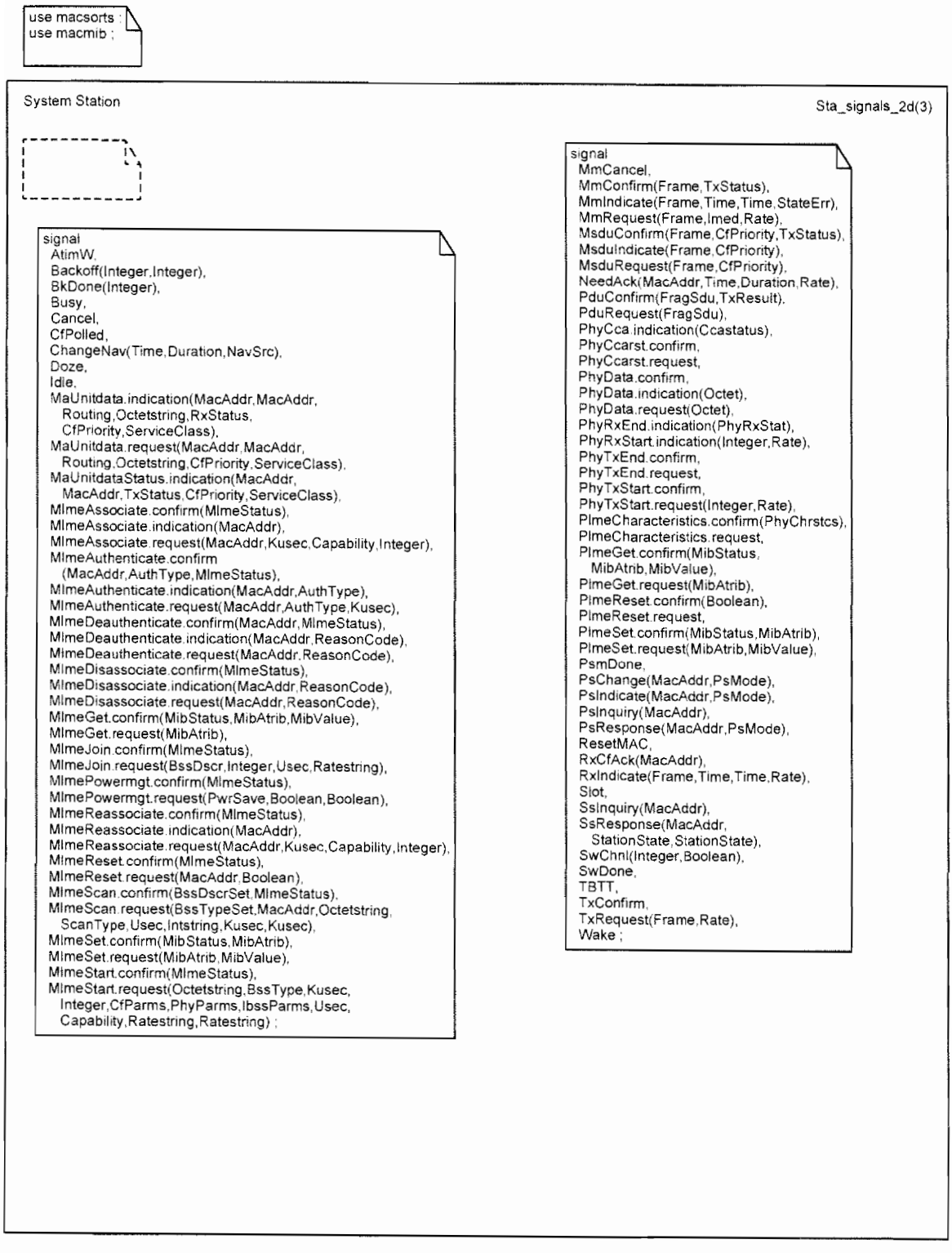


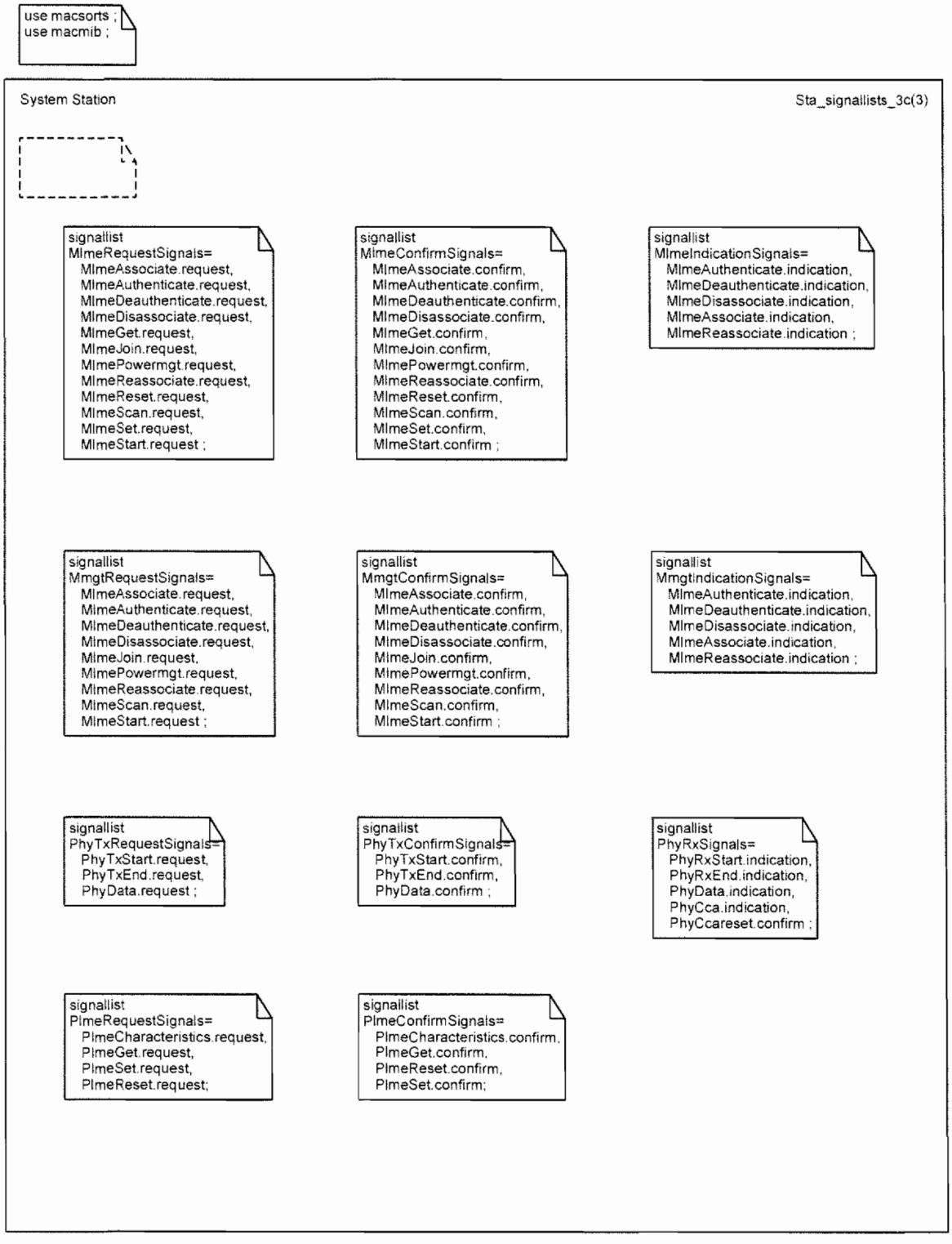
C.3 State machines for MAC stations

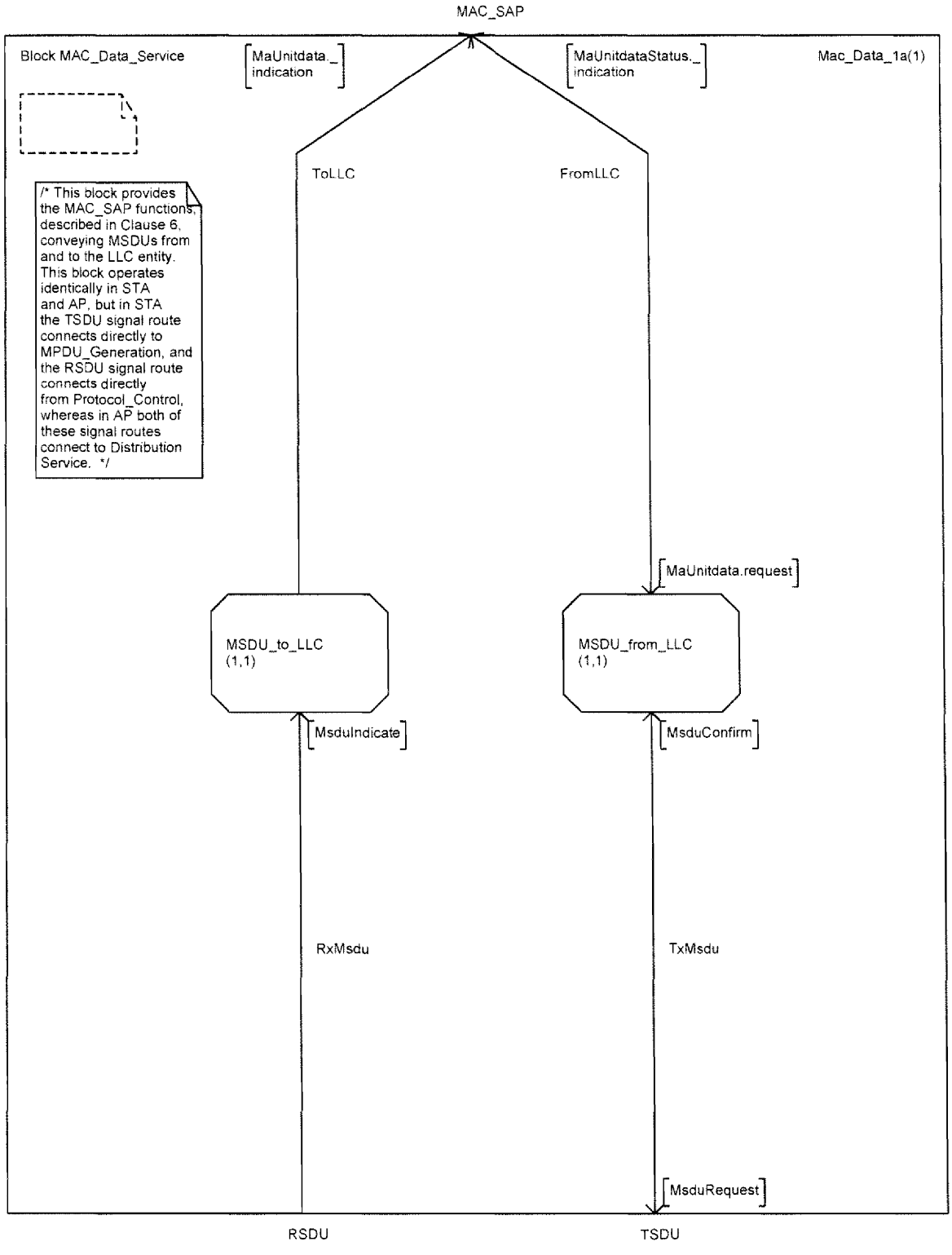
The following SDL-92 system specification defines operation of the MAC protocol at an IEEE 802.11 STA. Many aspects of STA operation also apply to AP operation. These are defined in blocks and processes referenced from both the STA and AP system specifications. Blocks and processes used in both STA and AP are identifiable by the SDL comment /* for STA & AP */ below the block or process name. Blocks and processes specific to STA operation are identifiable by the SDL comment /* station version */ below the block or process name. The definitions of all blocks and processes referenced in the station system specification appear in Clause C.3.

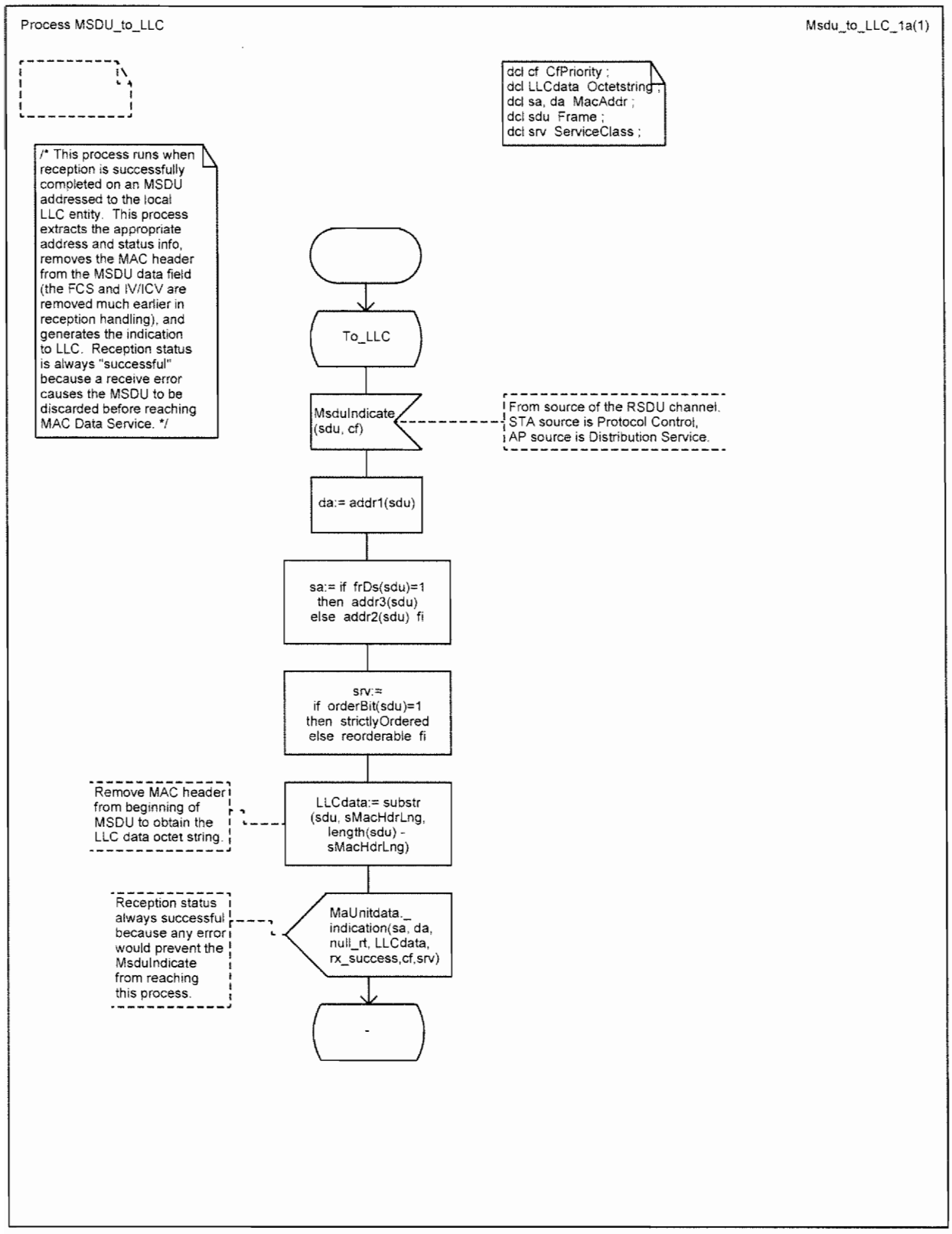
The remainder of Clause C.3 is the formal description, in SDL/GR, of an IEEE 802.11 STA.

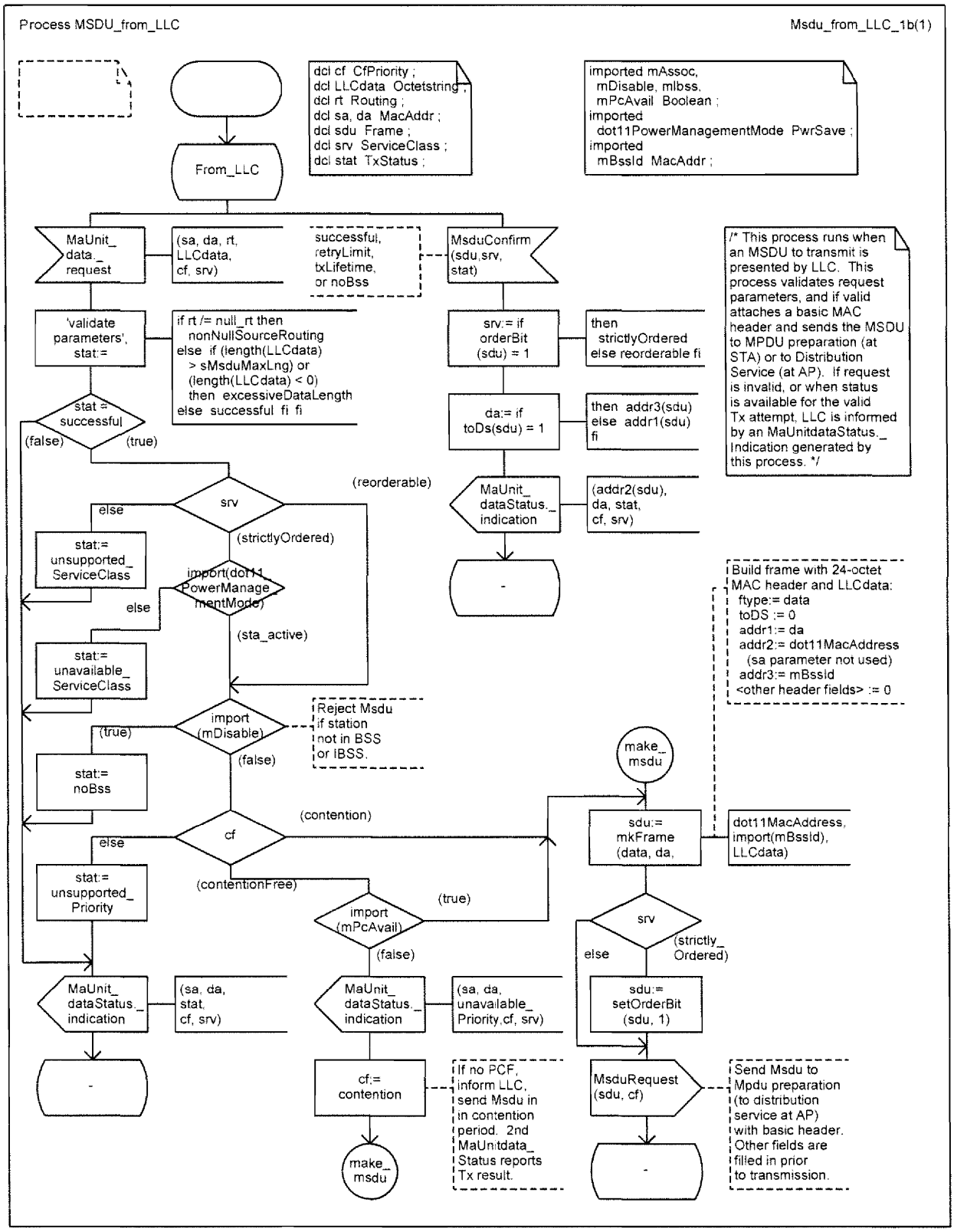


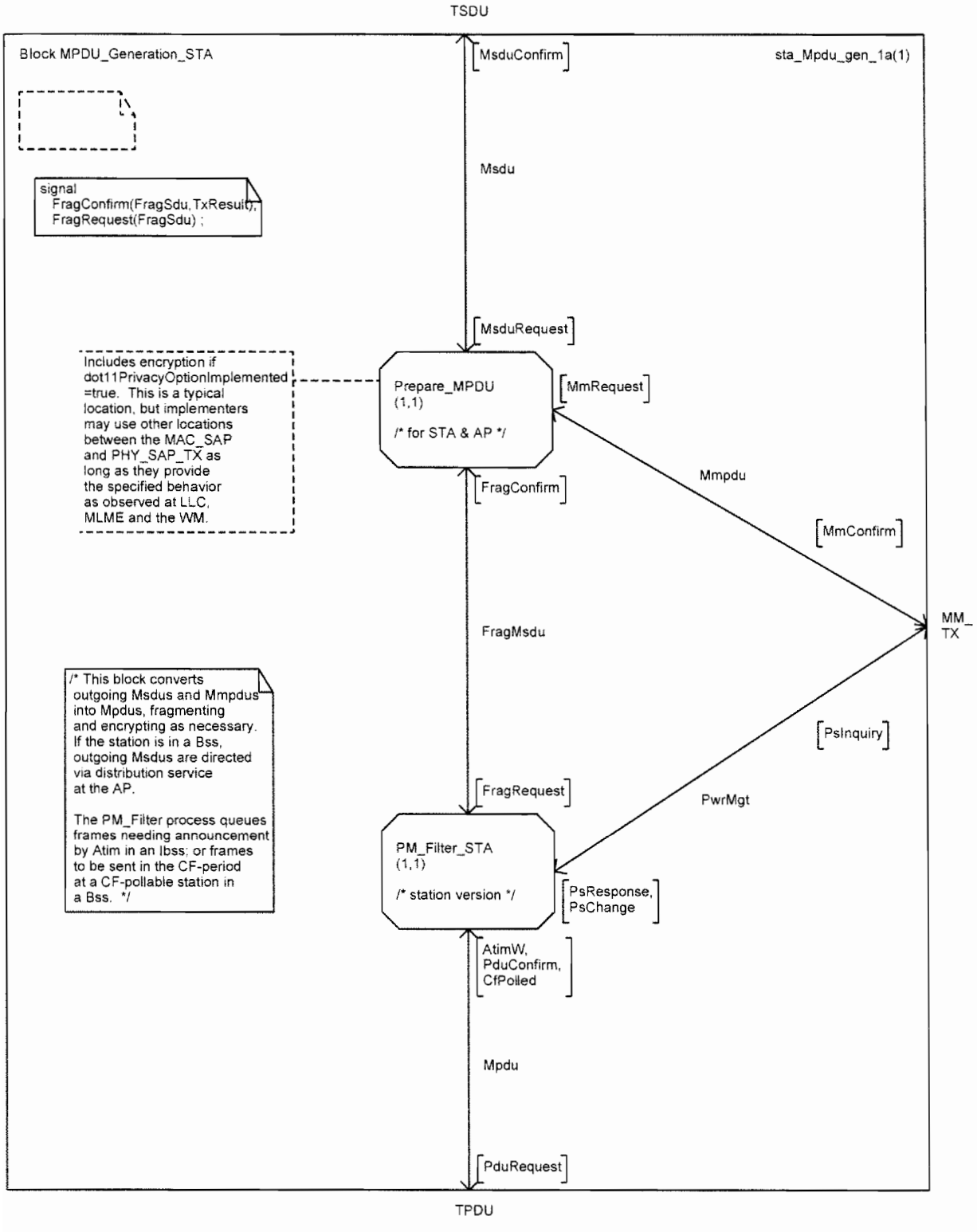


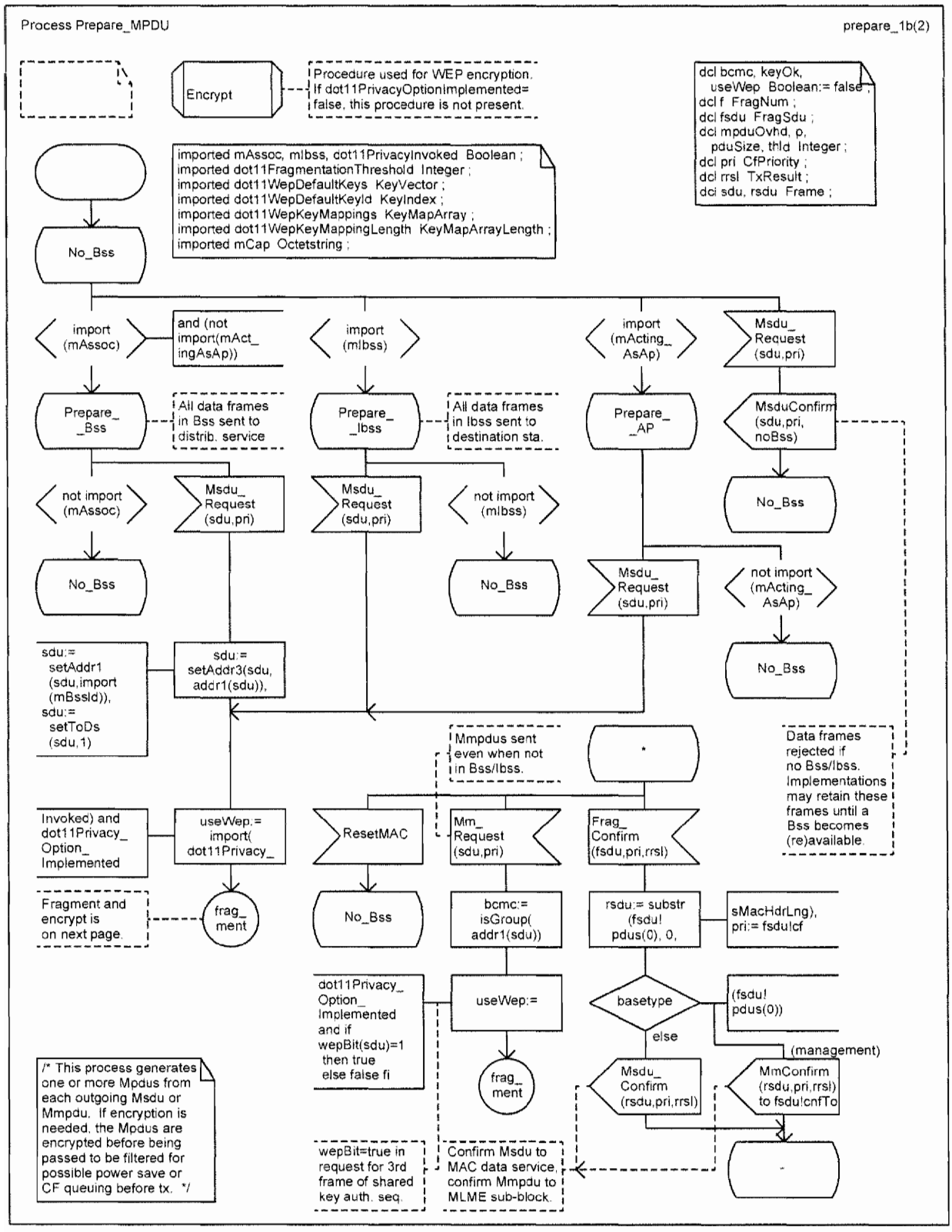


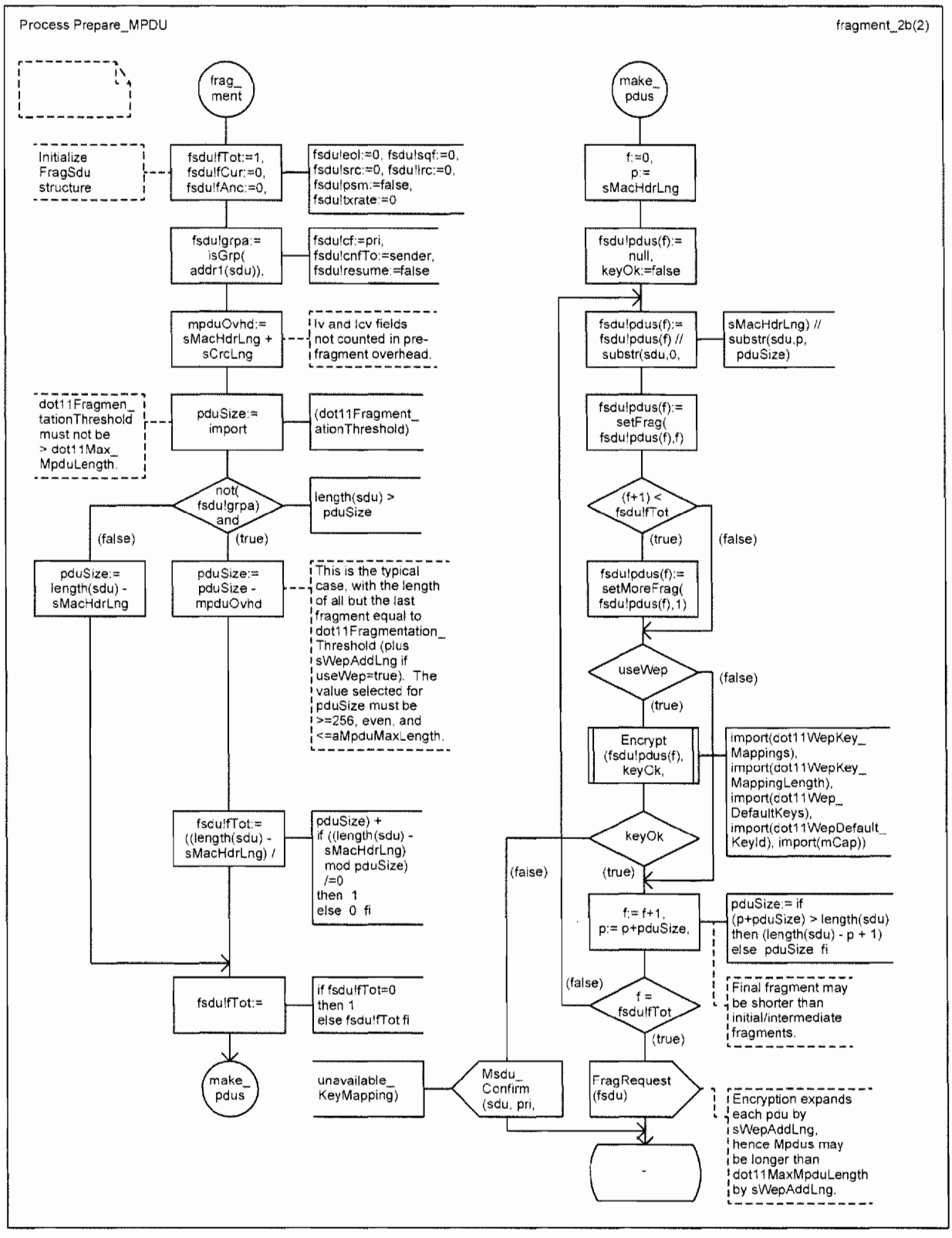


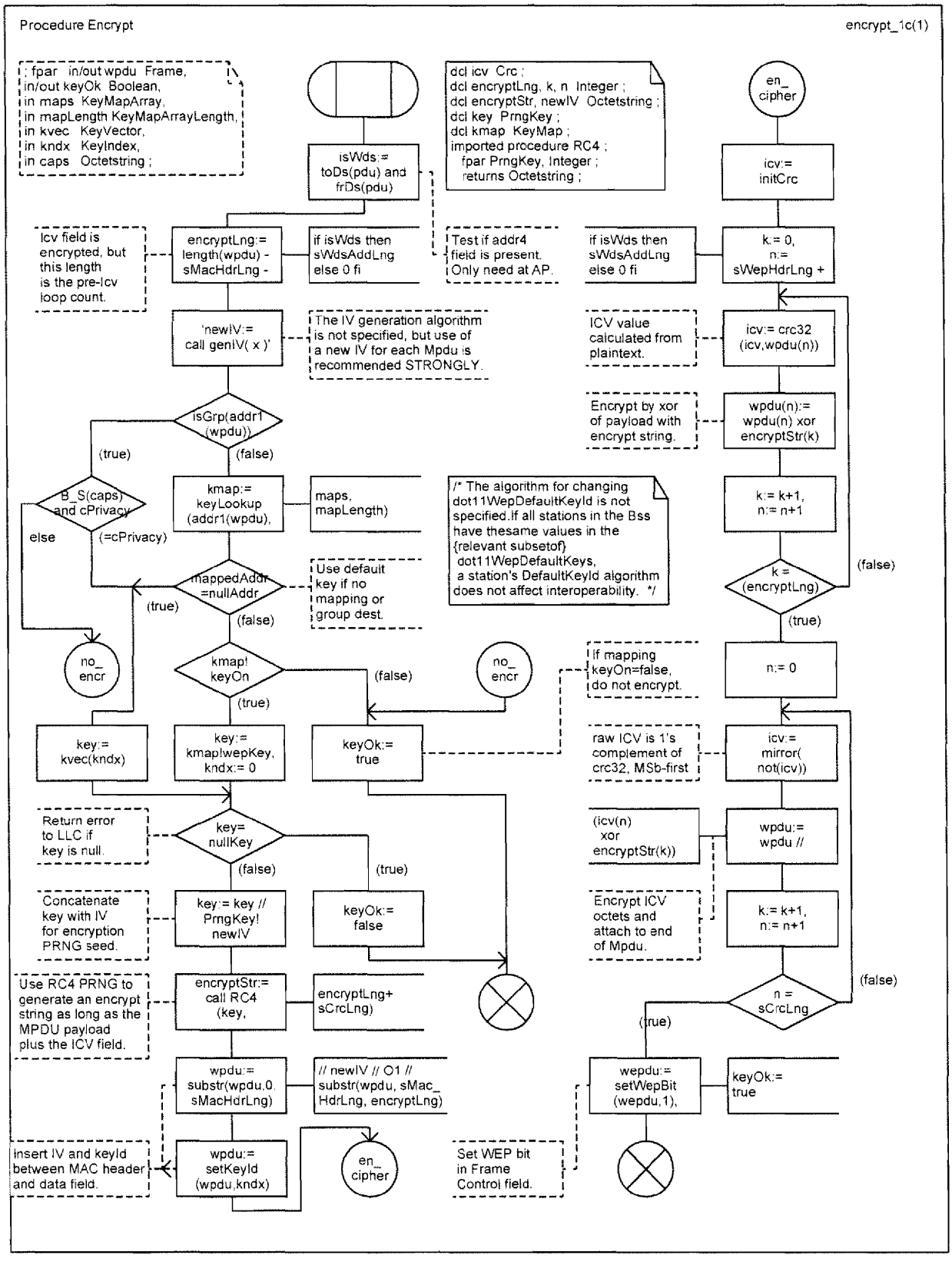


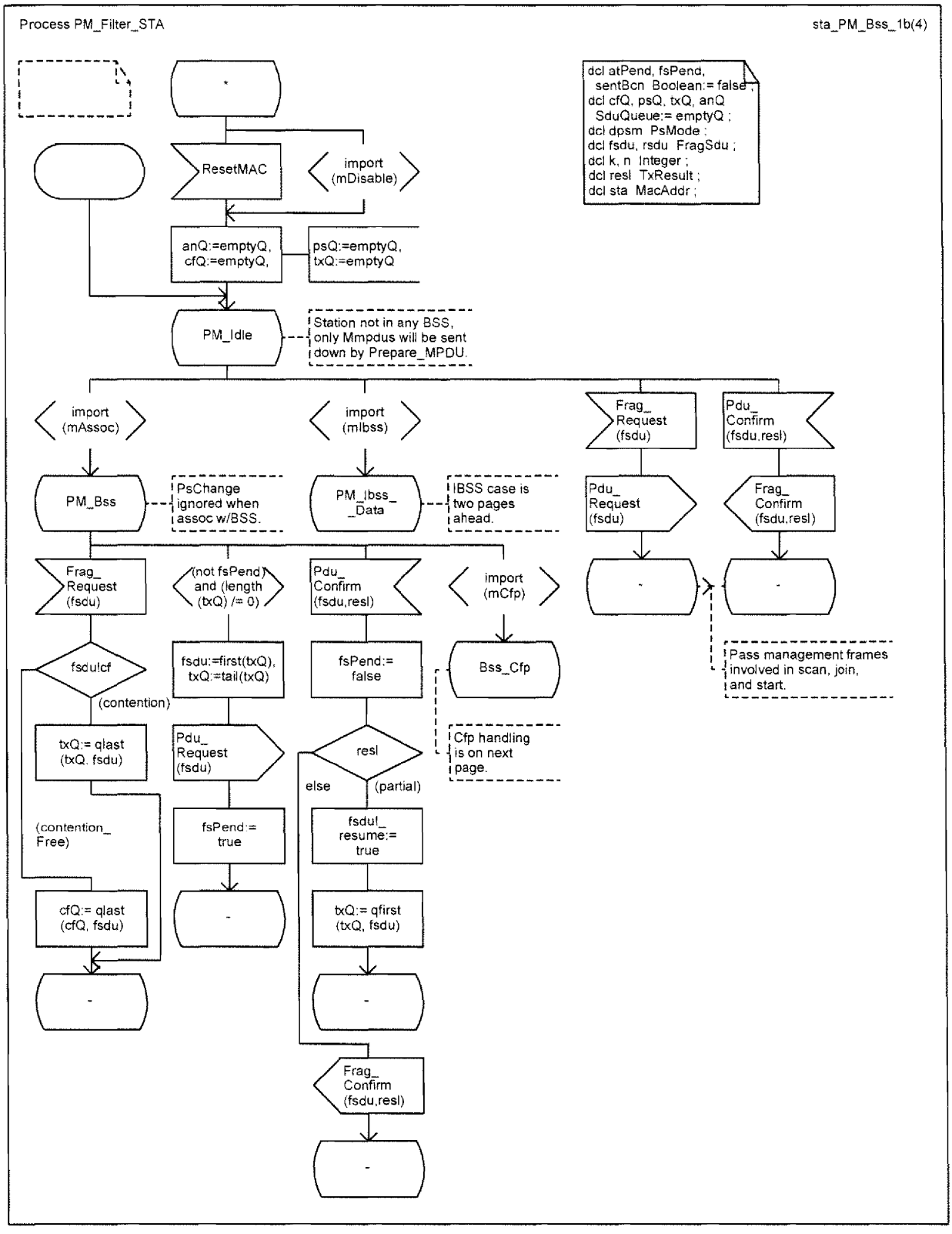


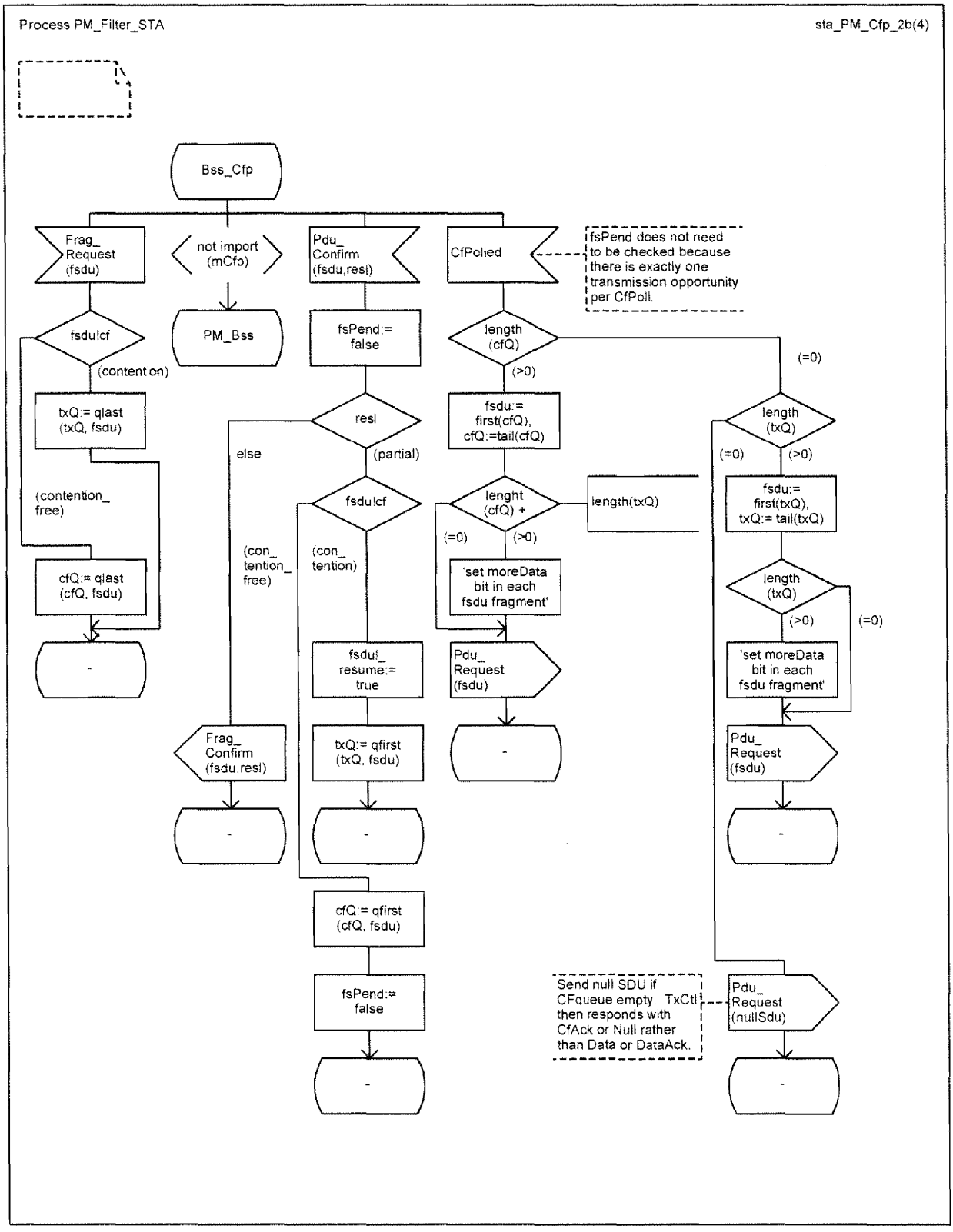


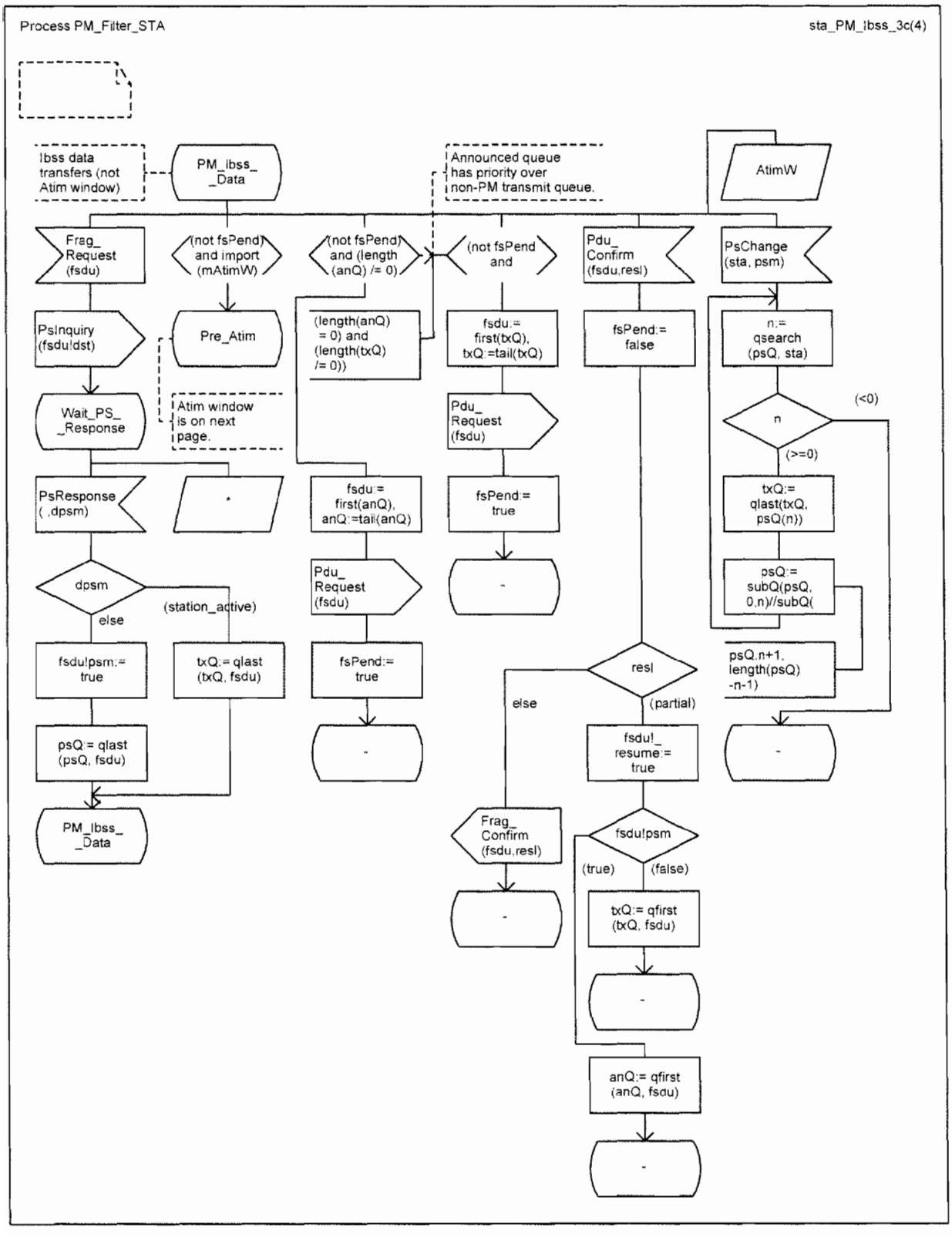


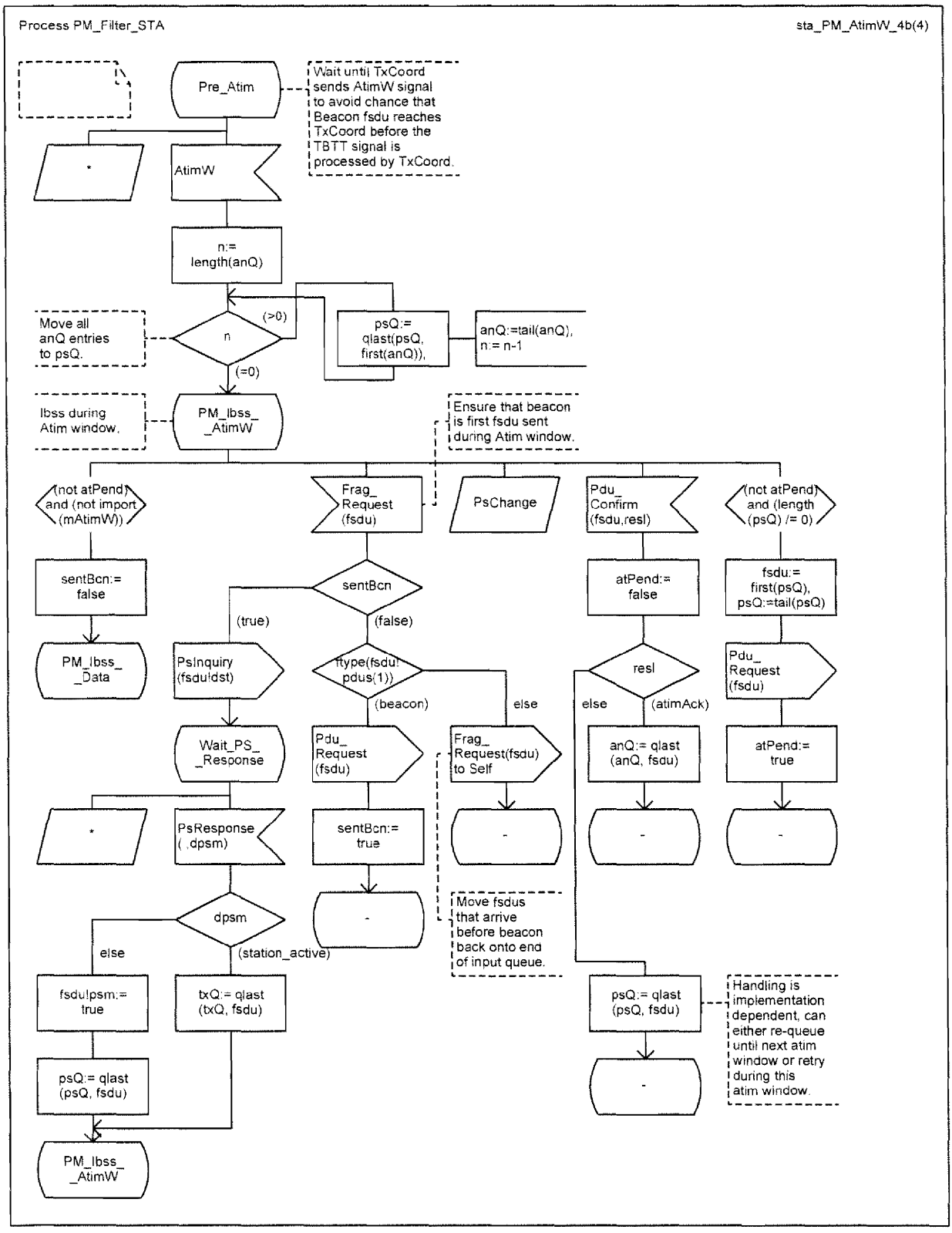


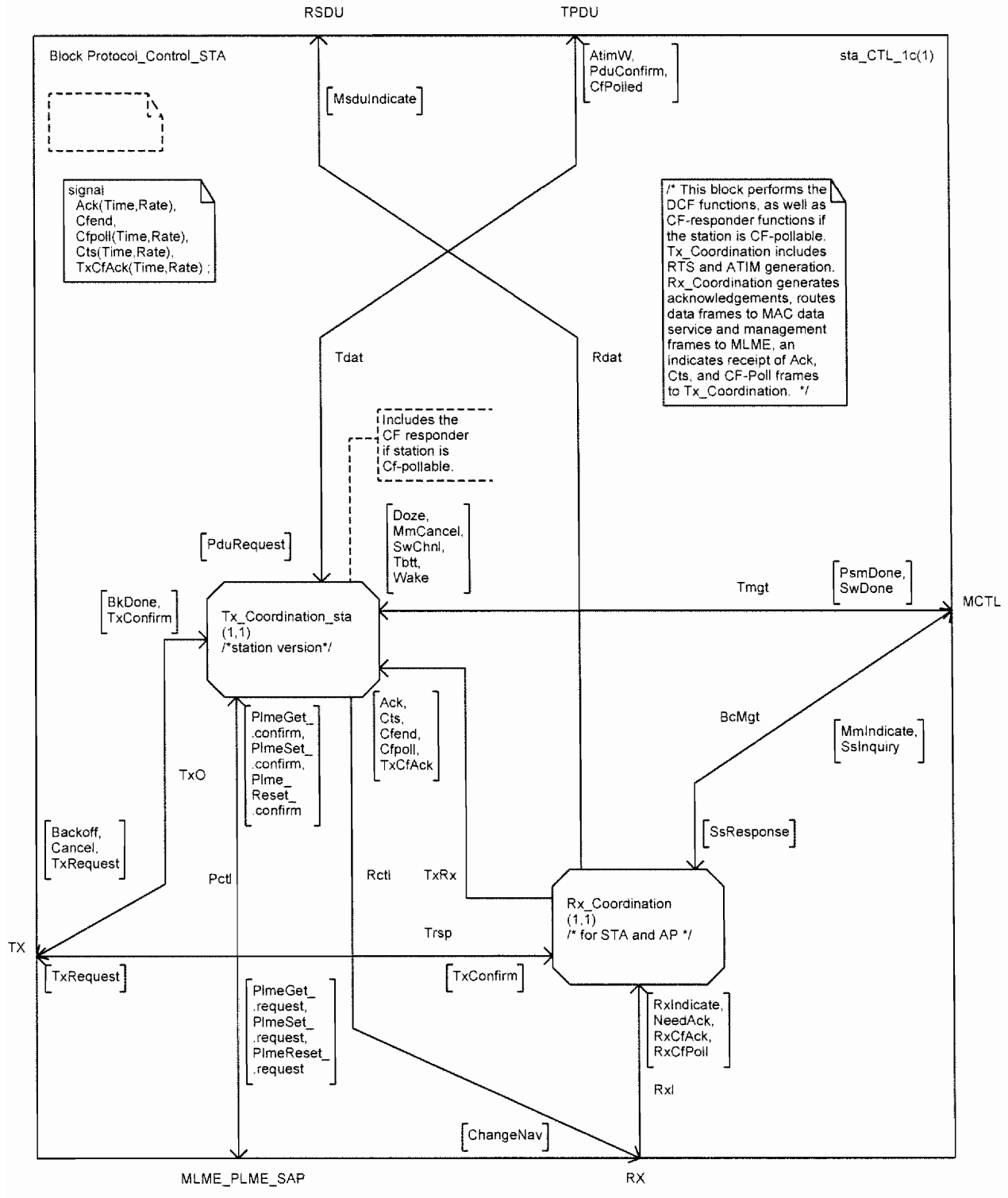


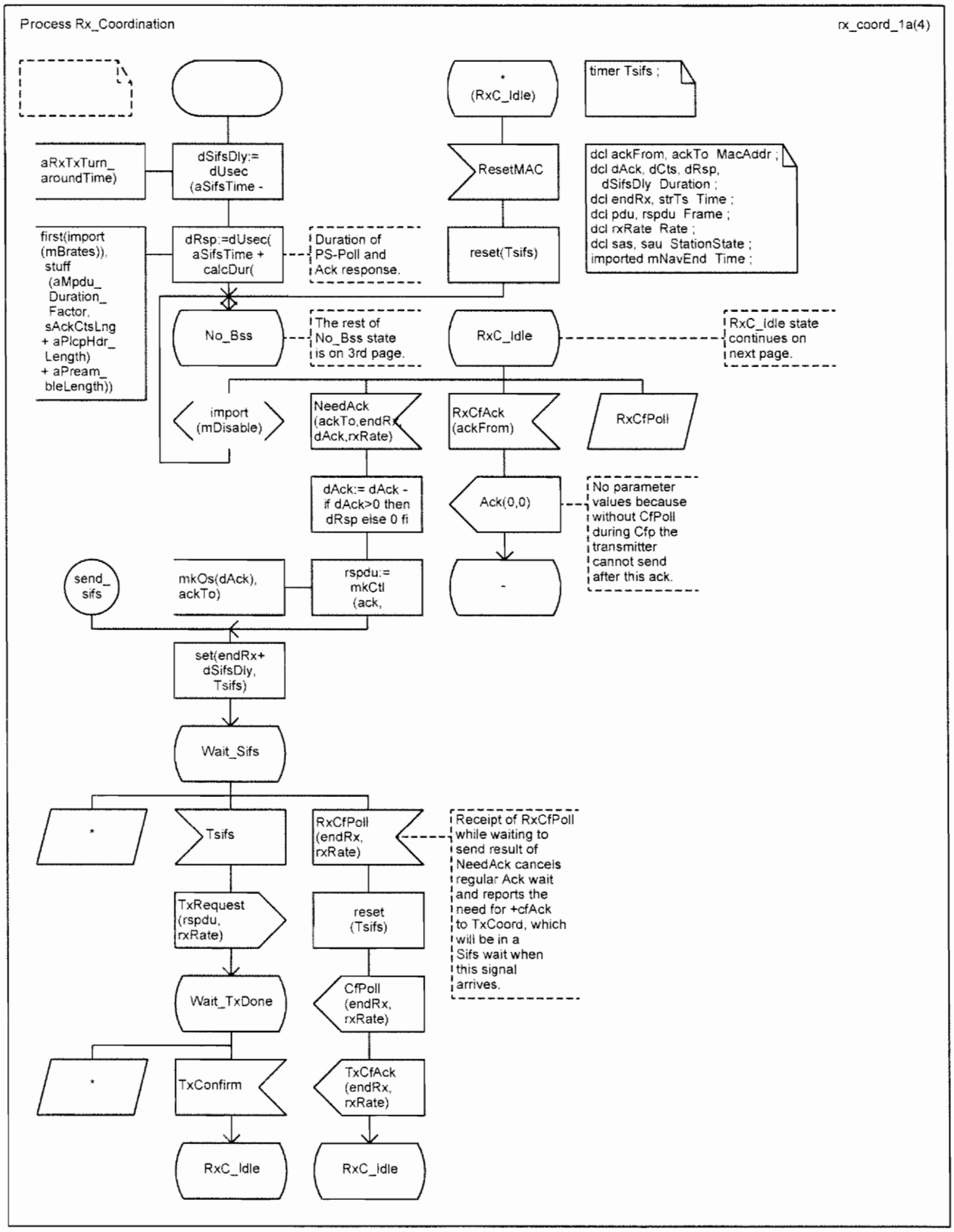


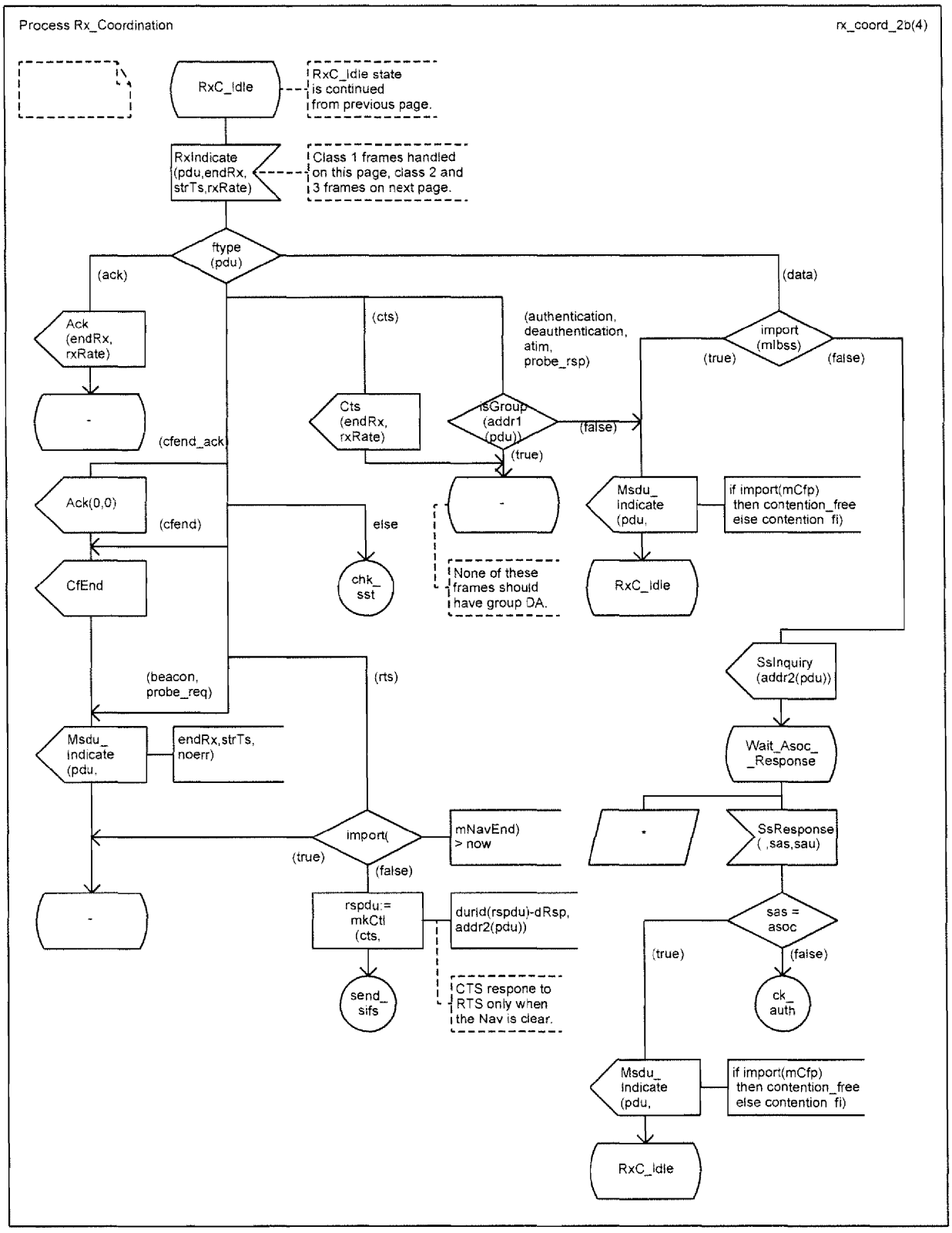


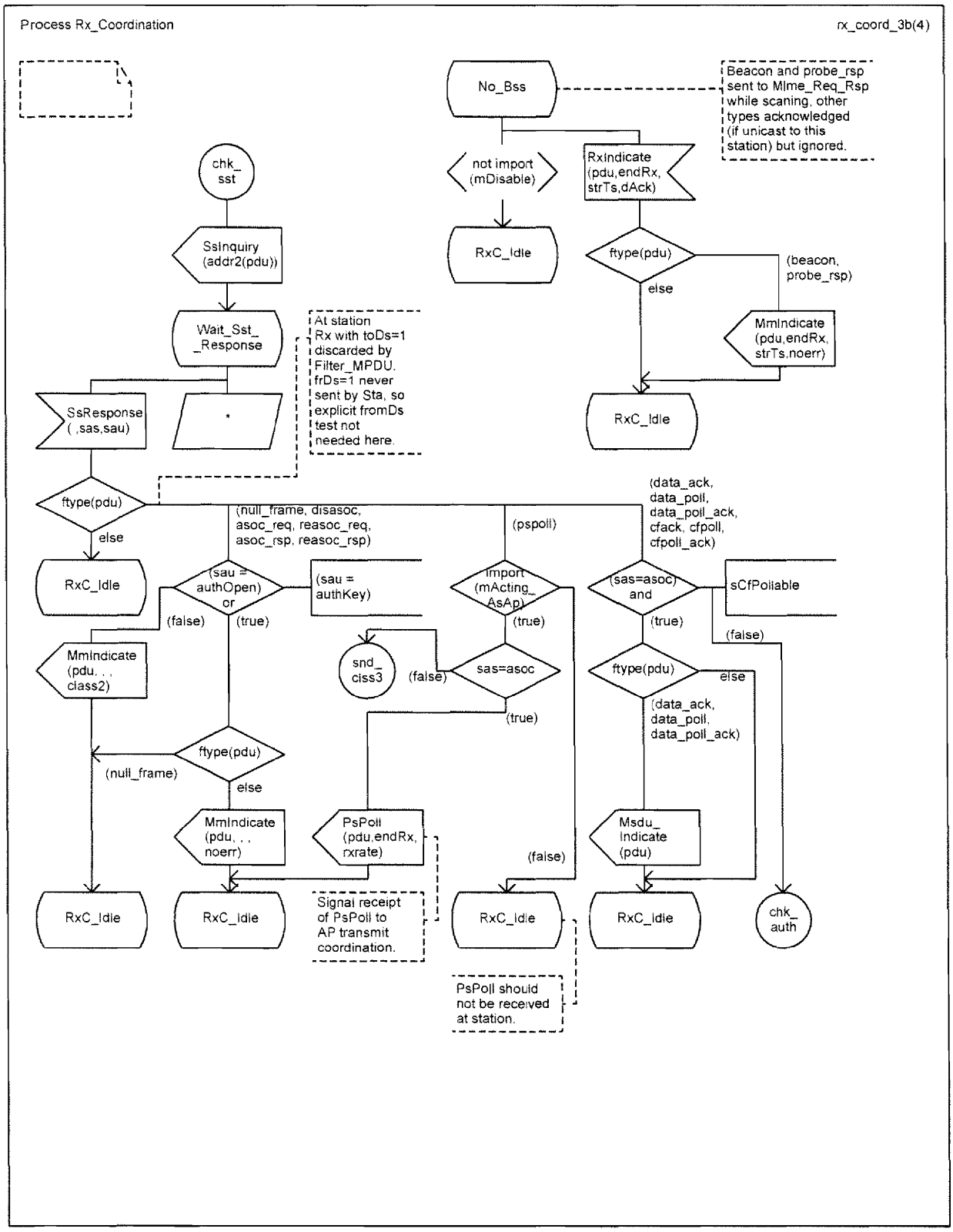


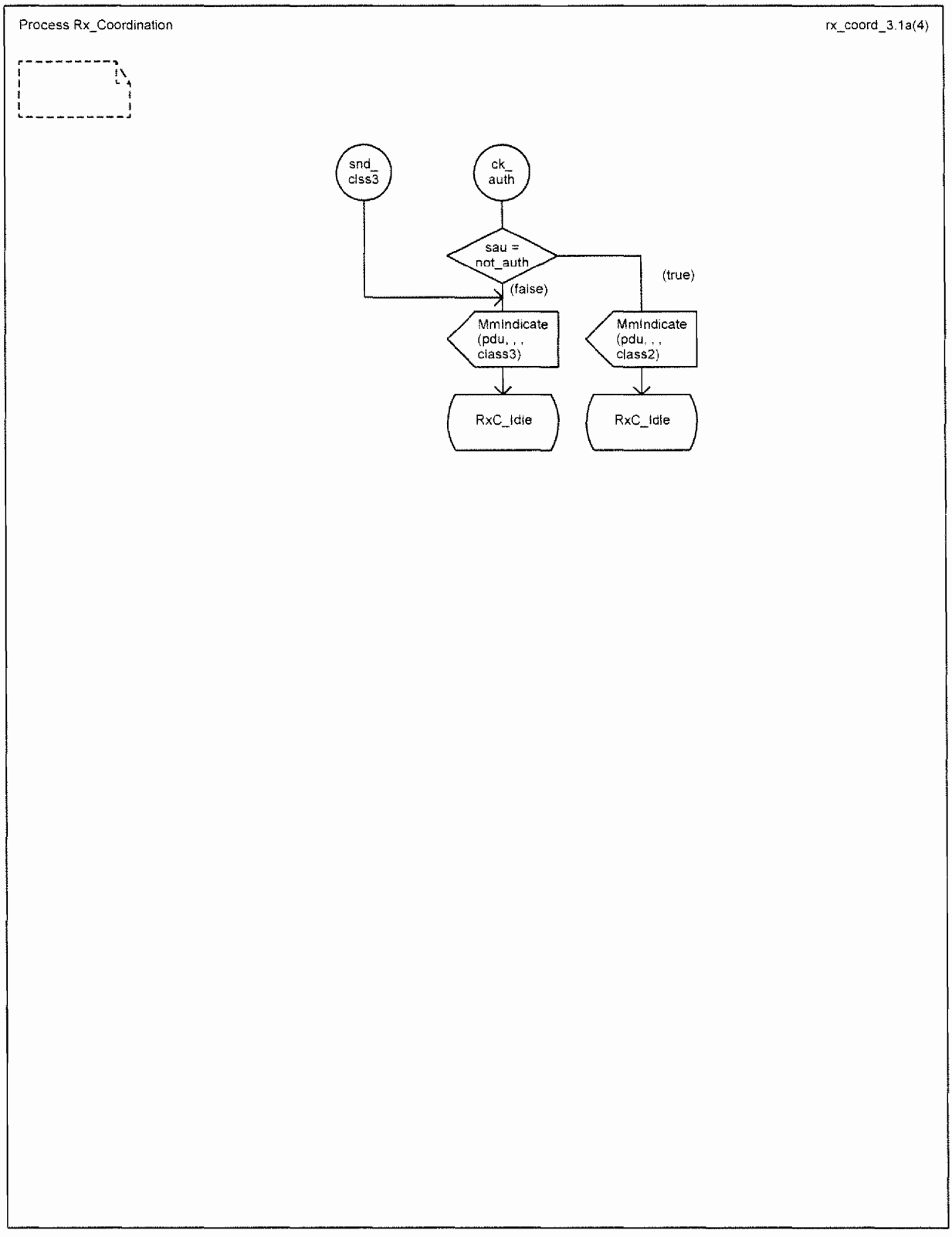


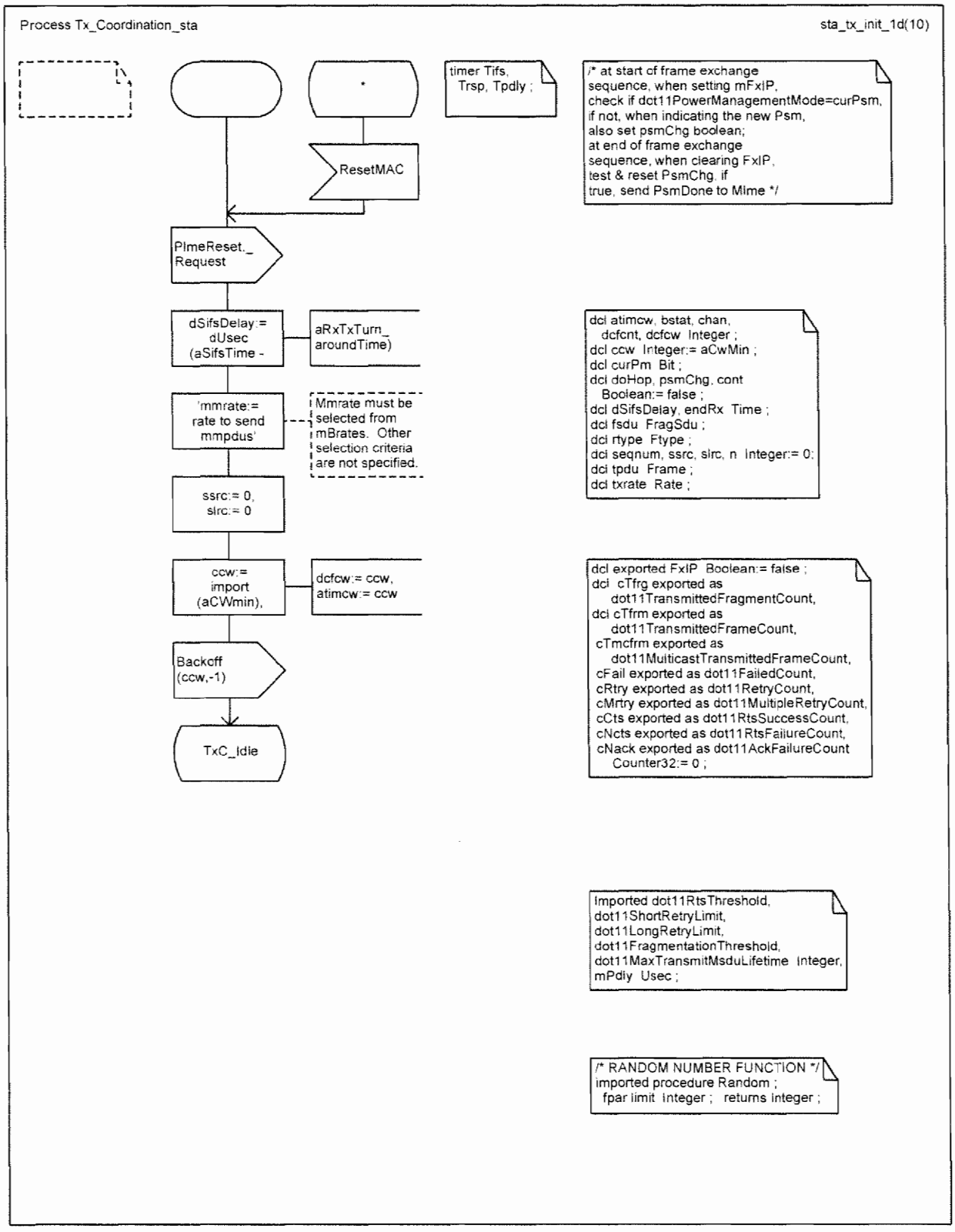


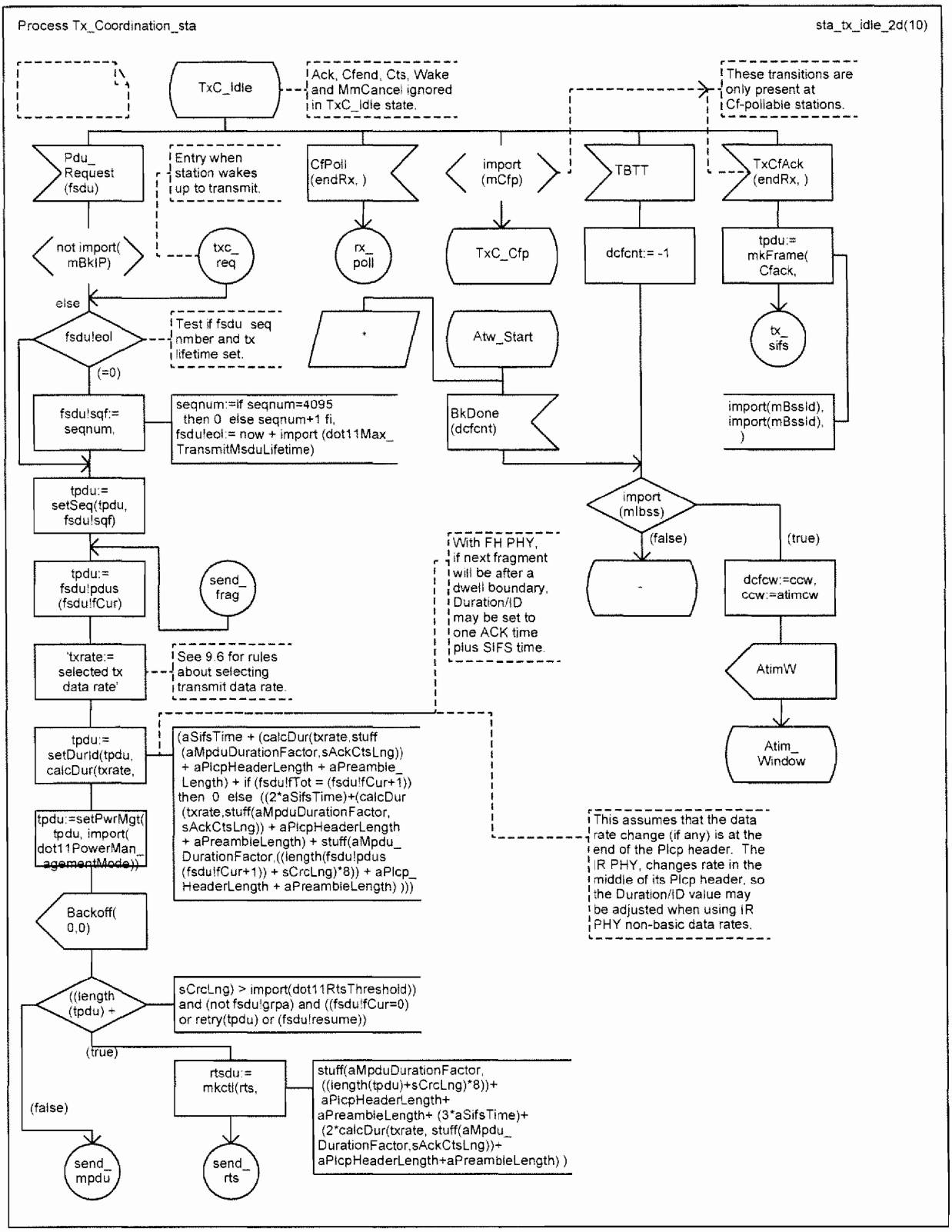


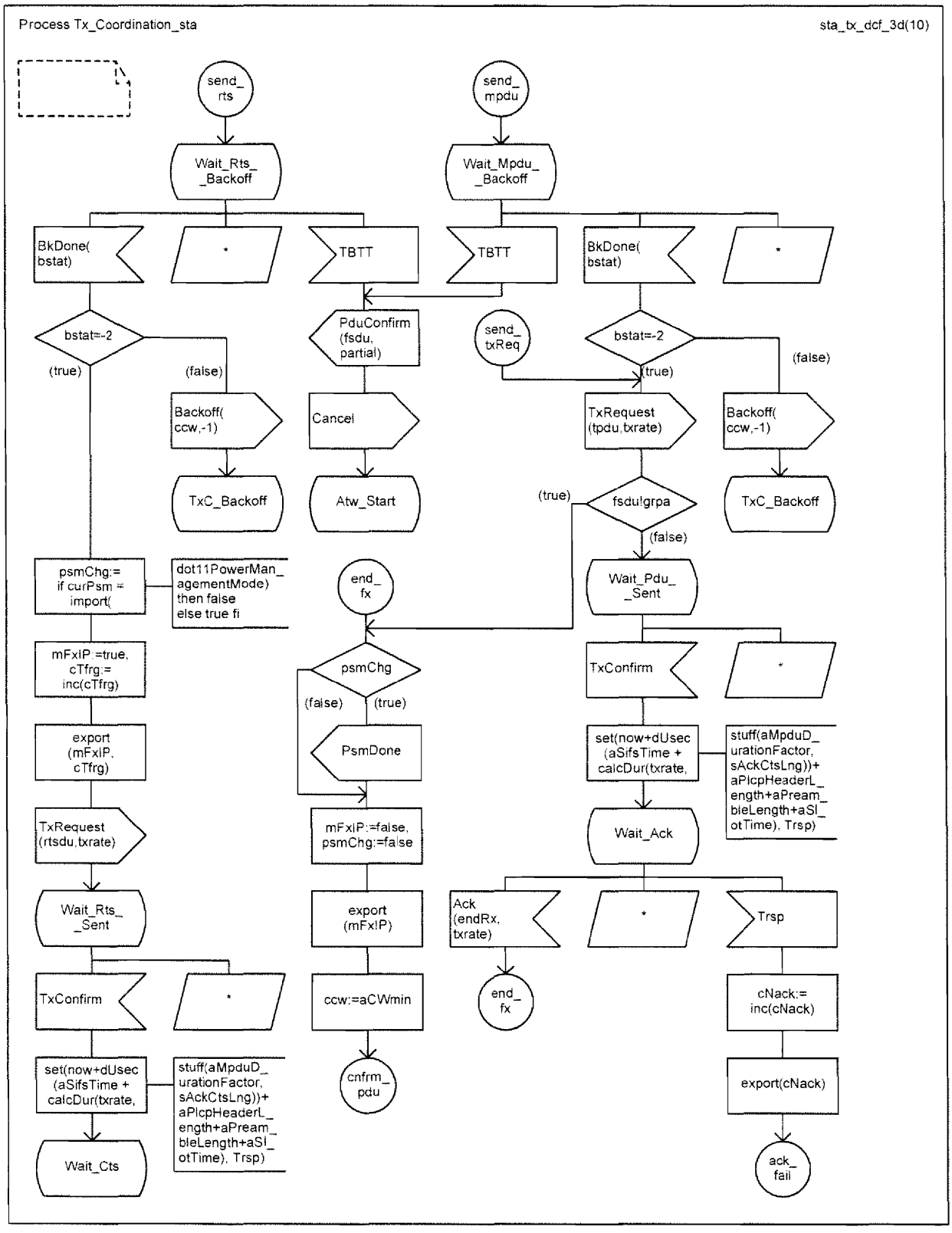


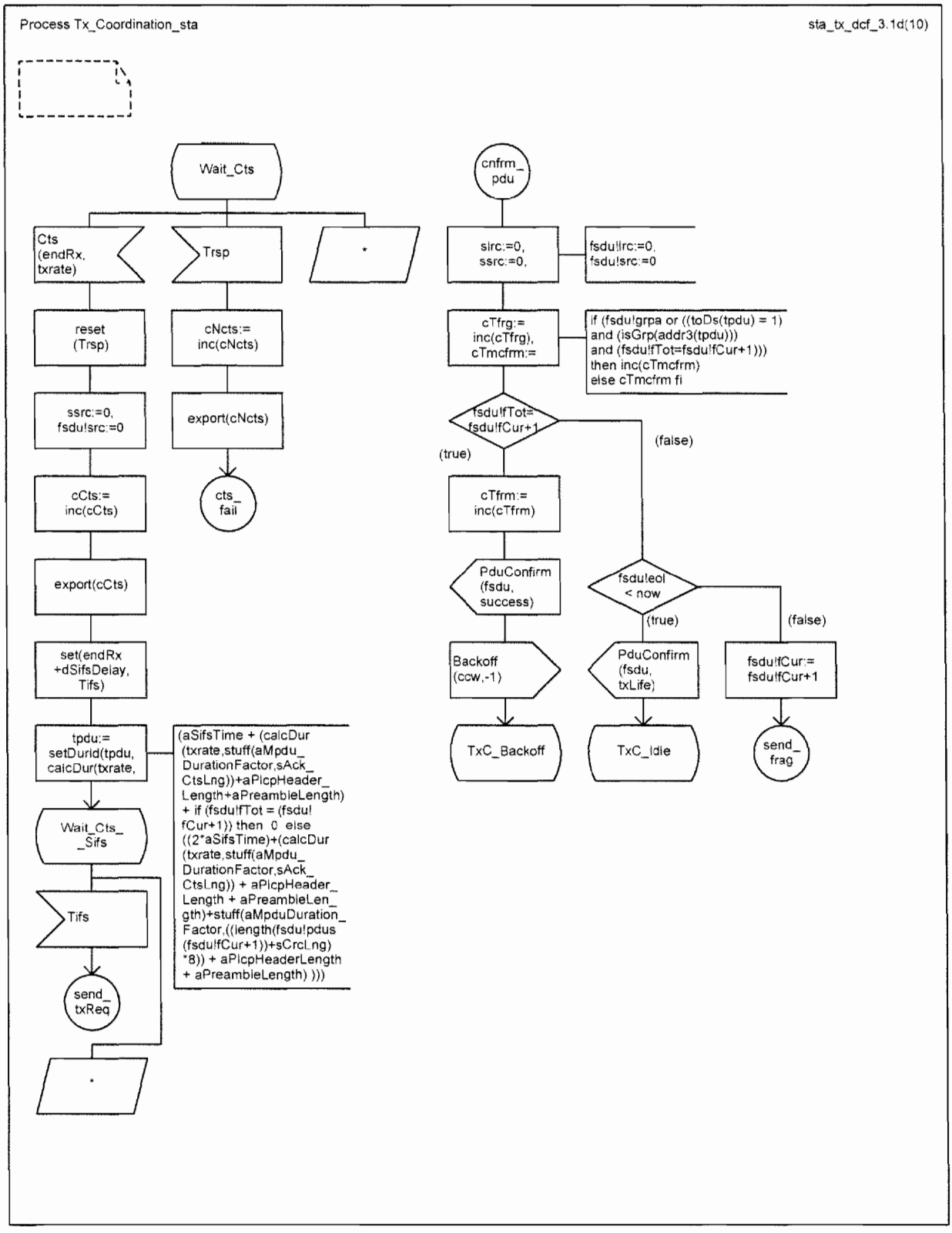


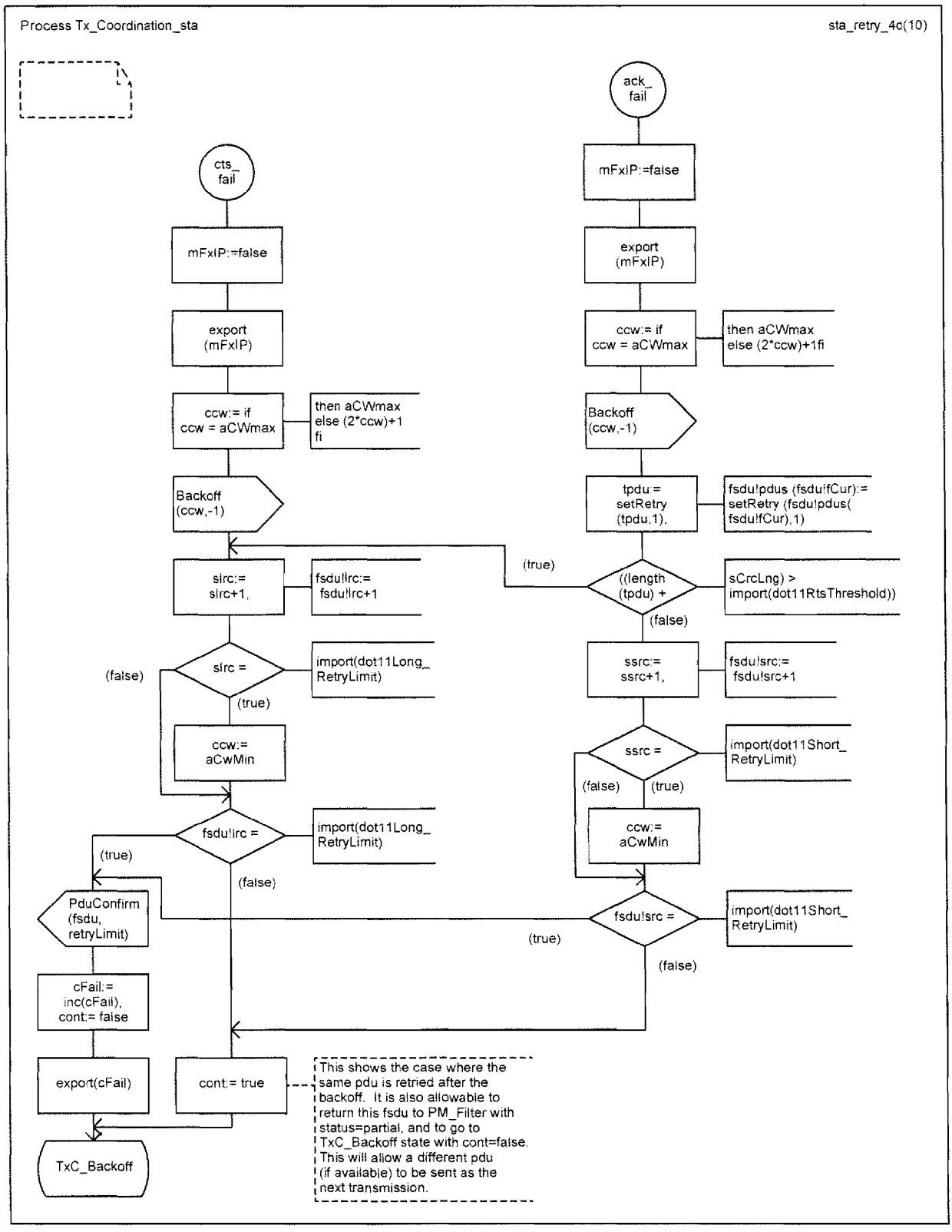


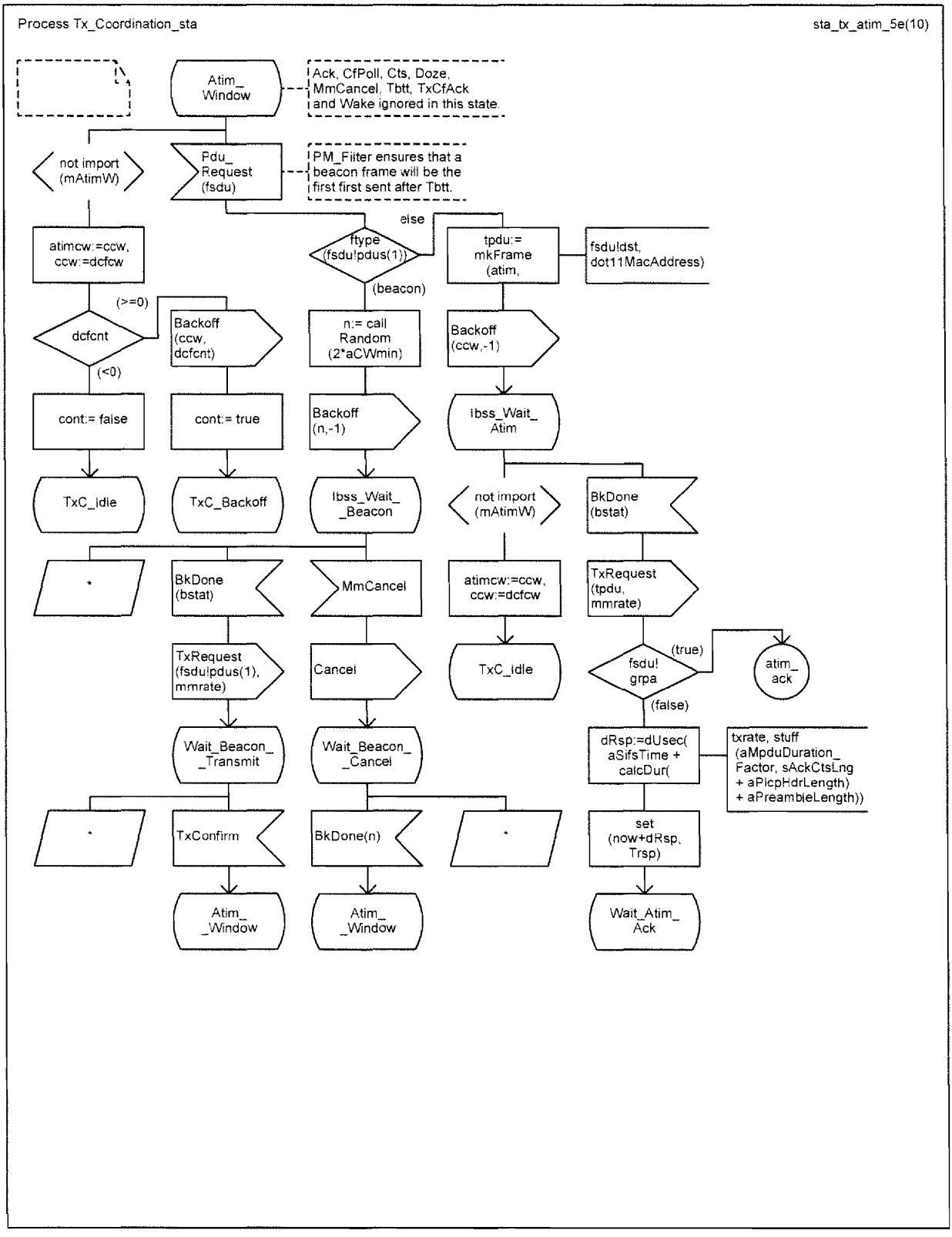


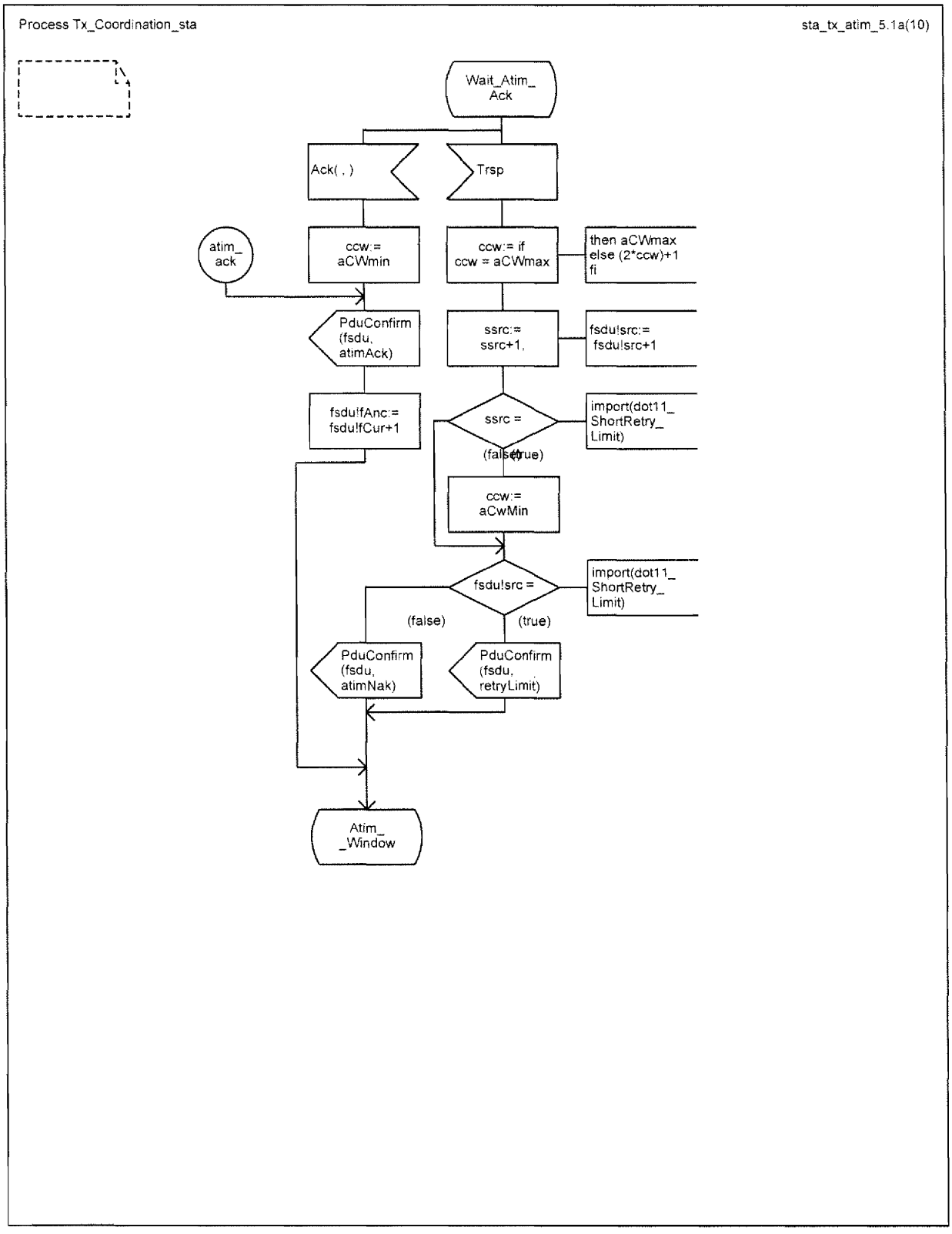


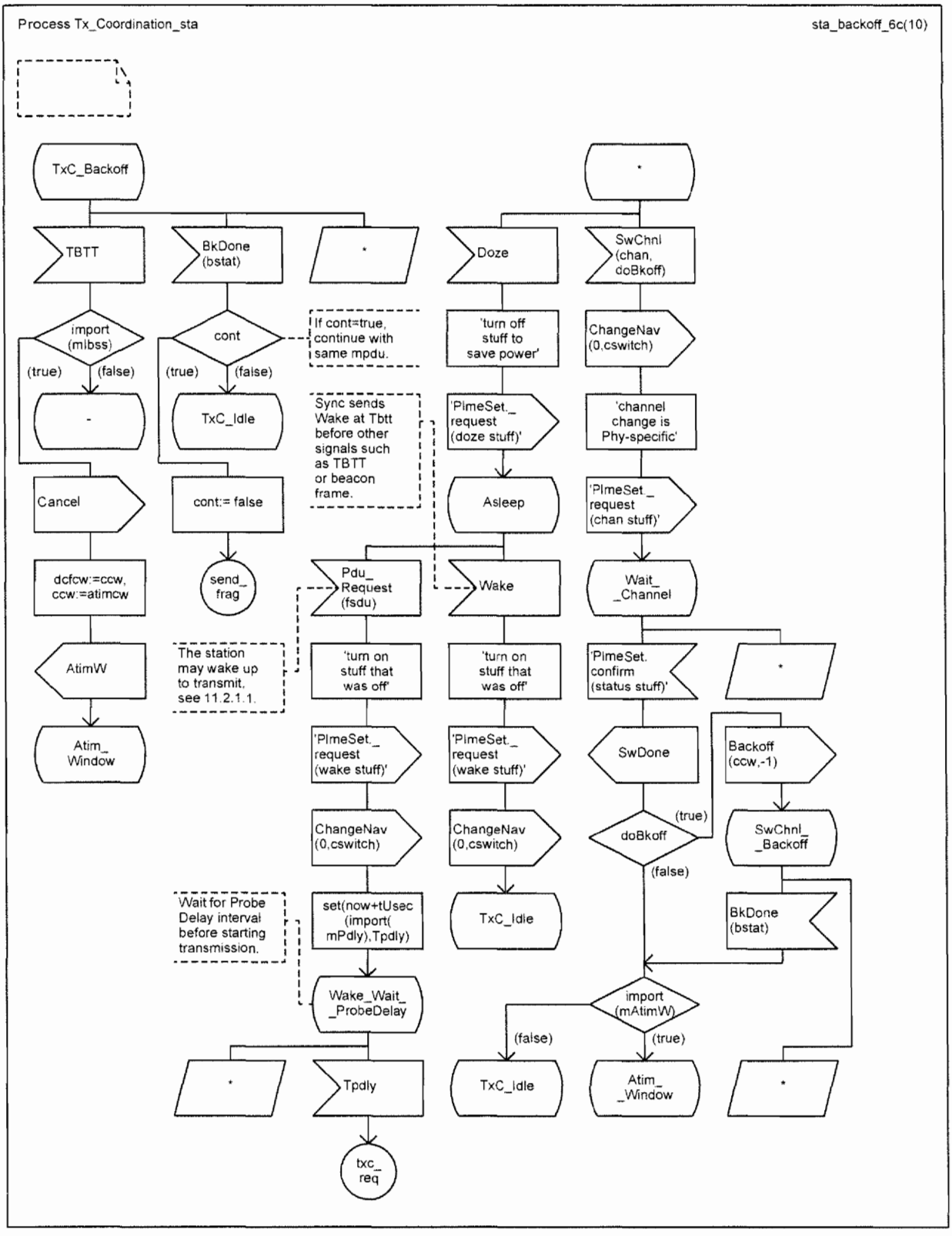


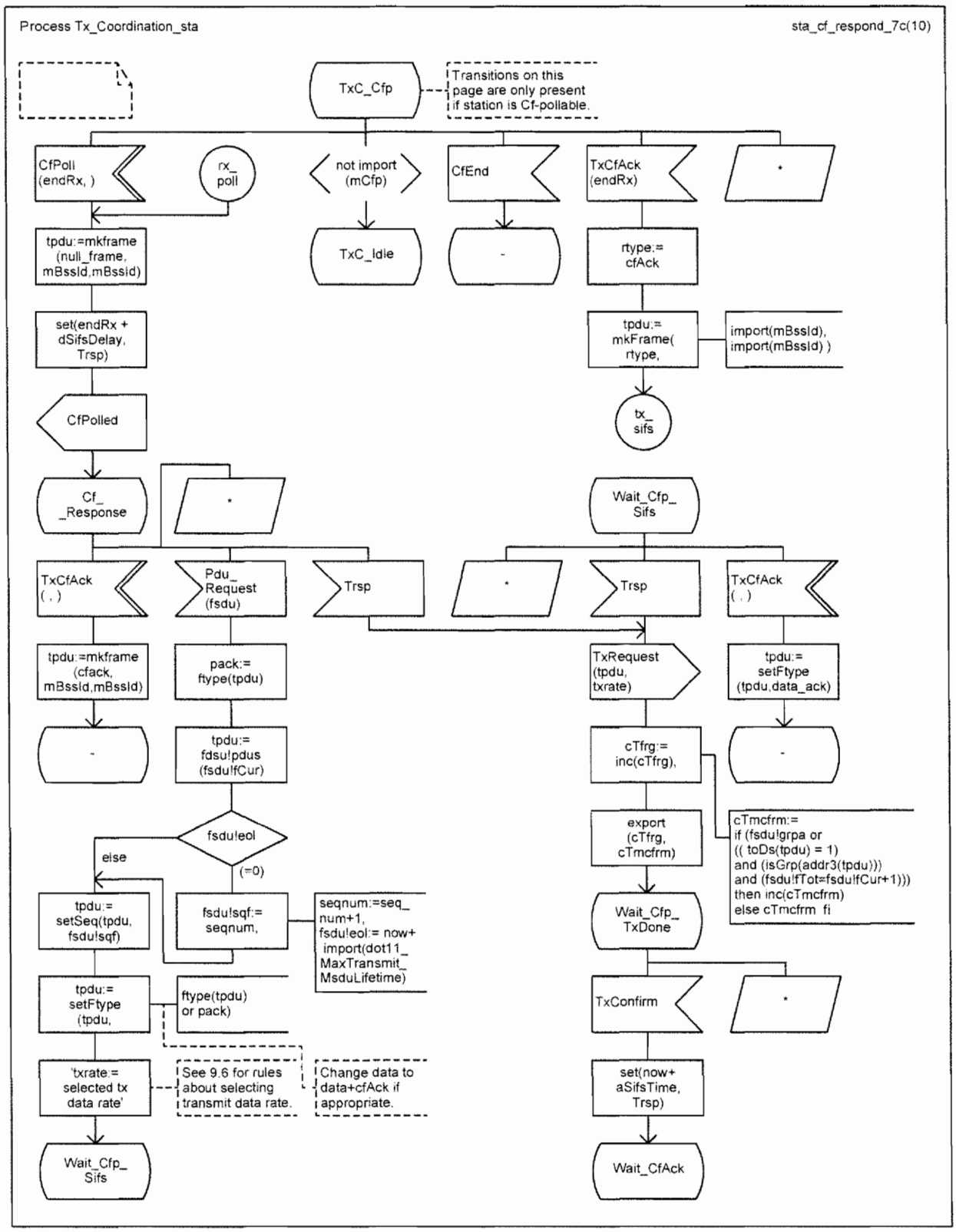


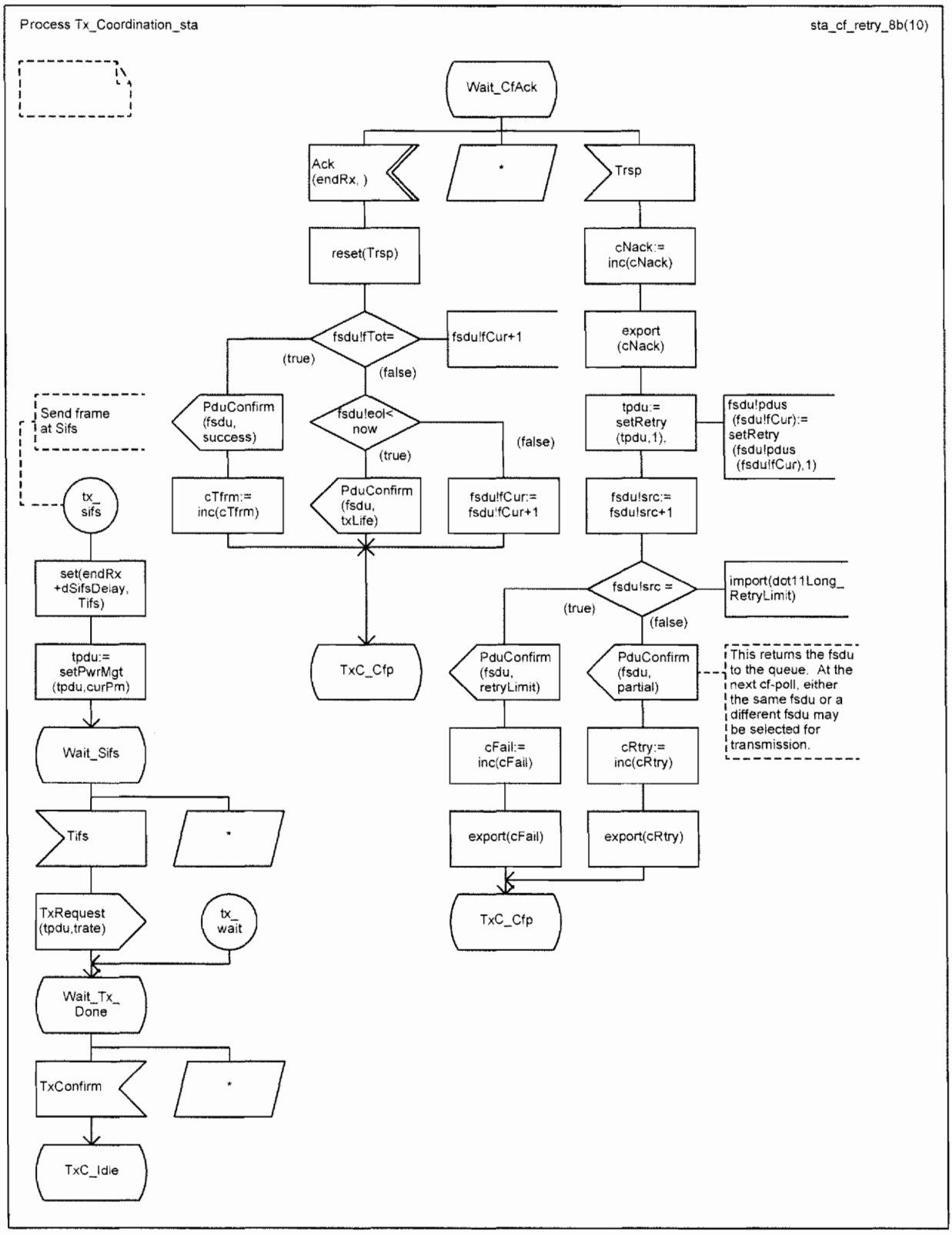


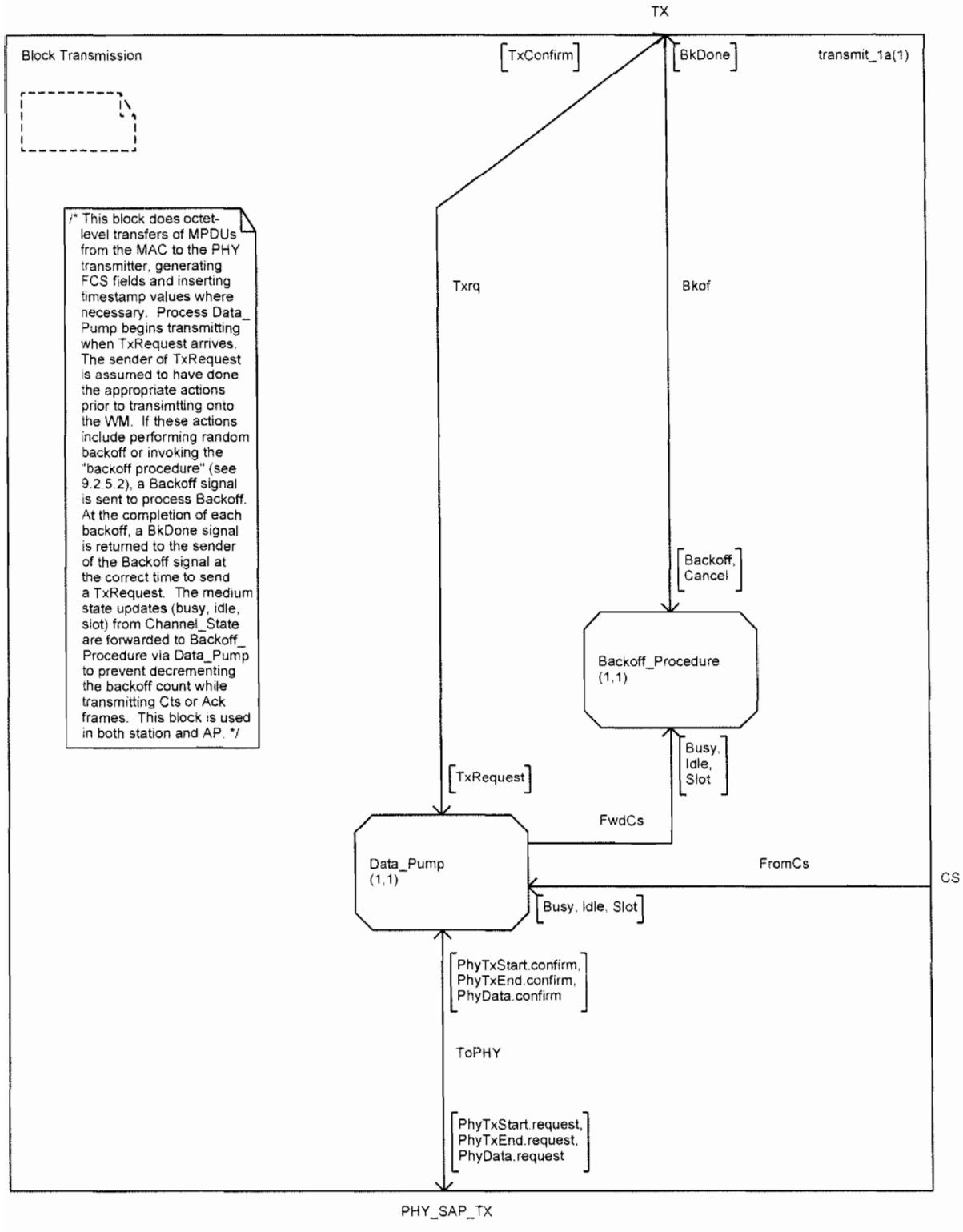


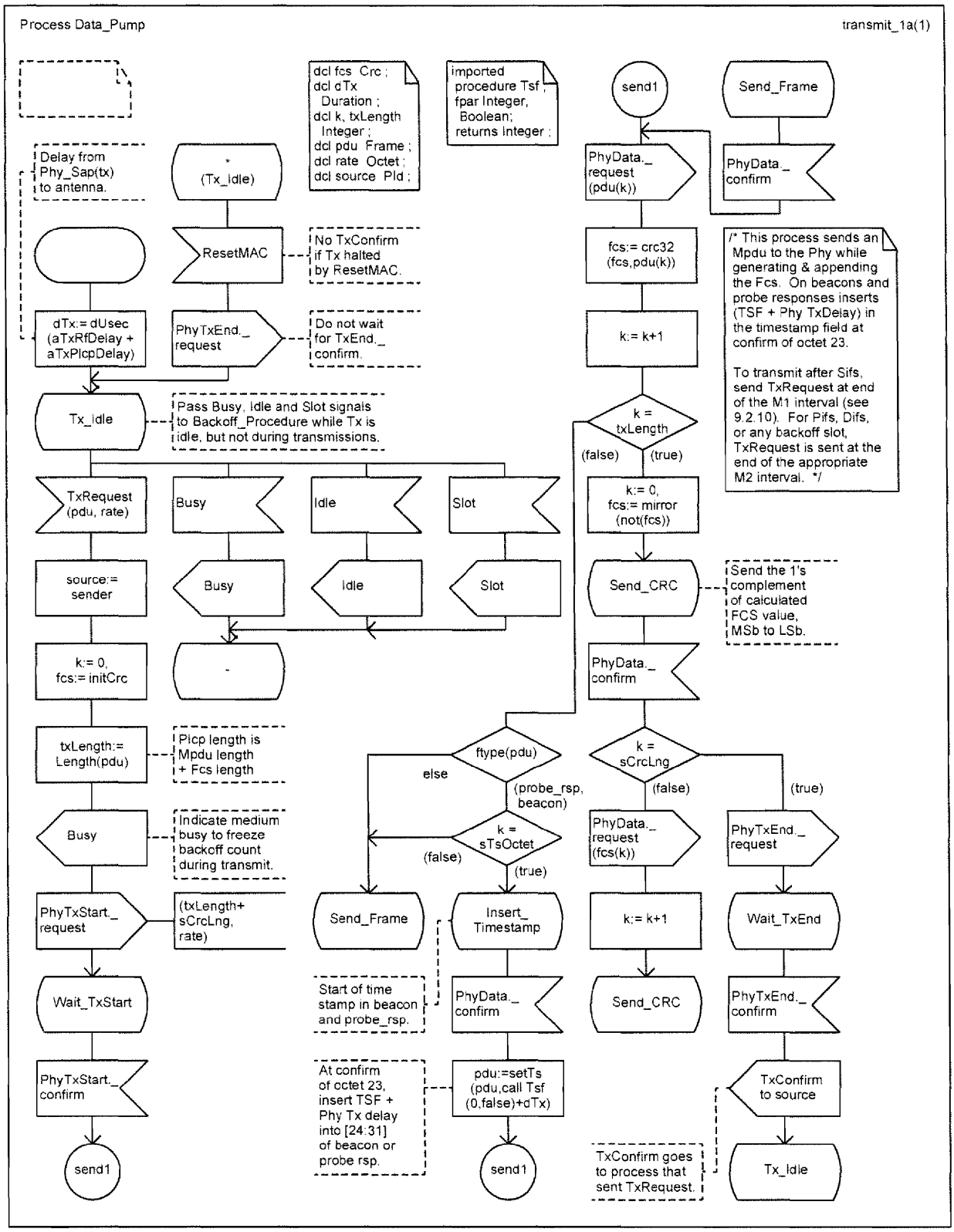


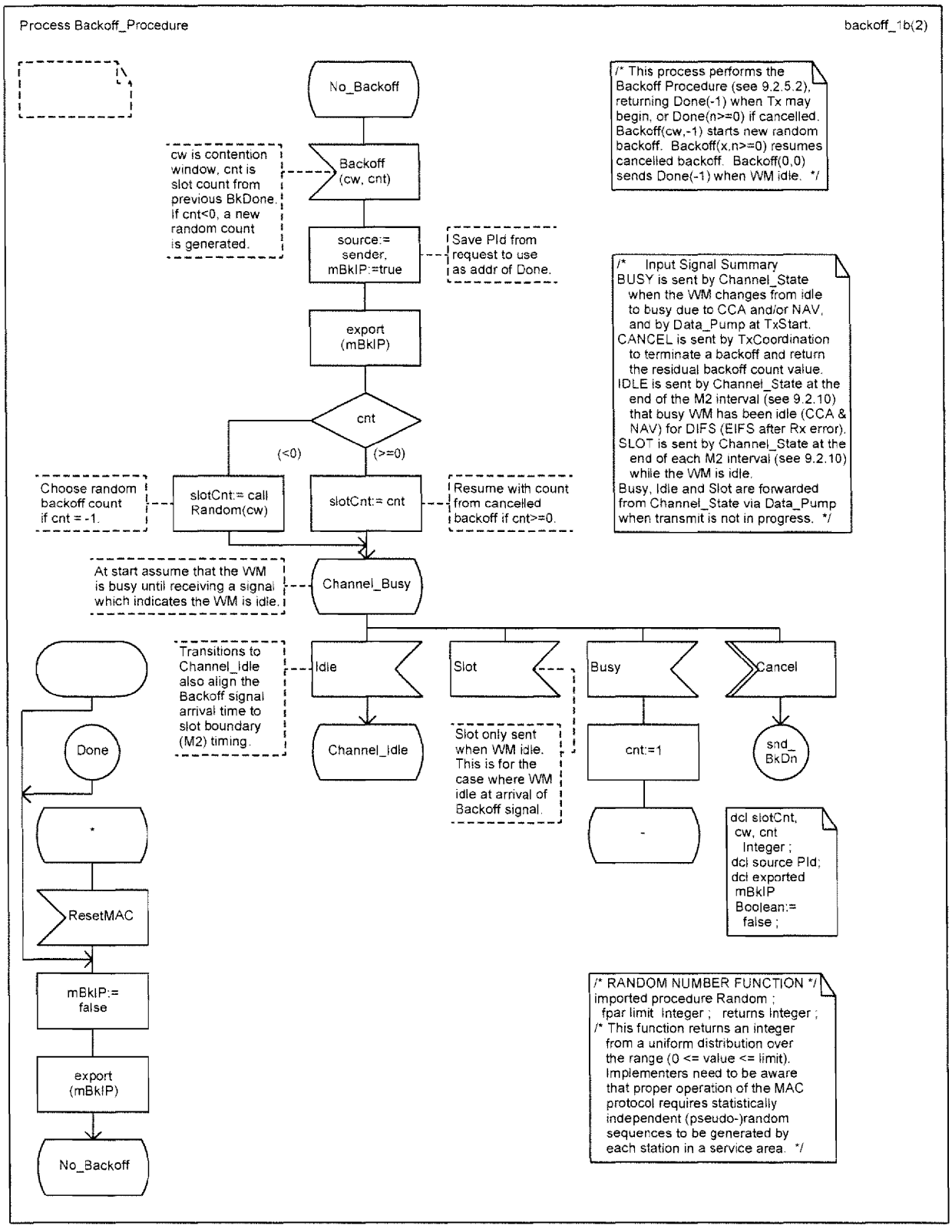


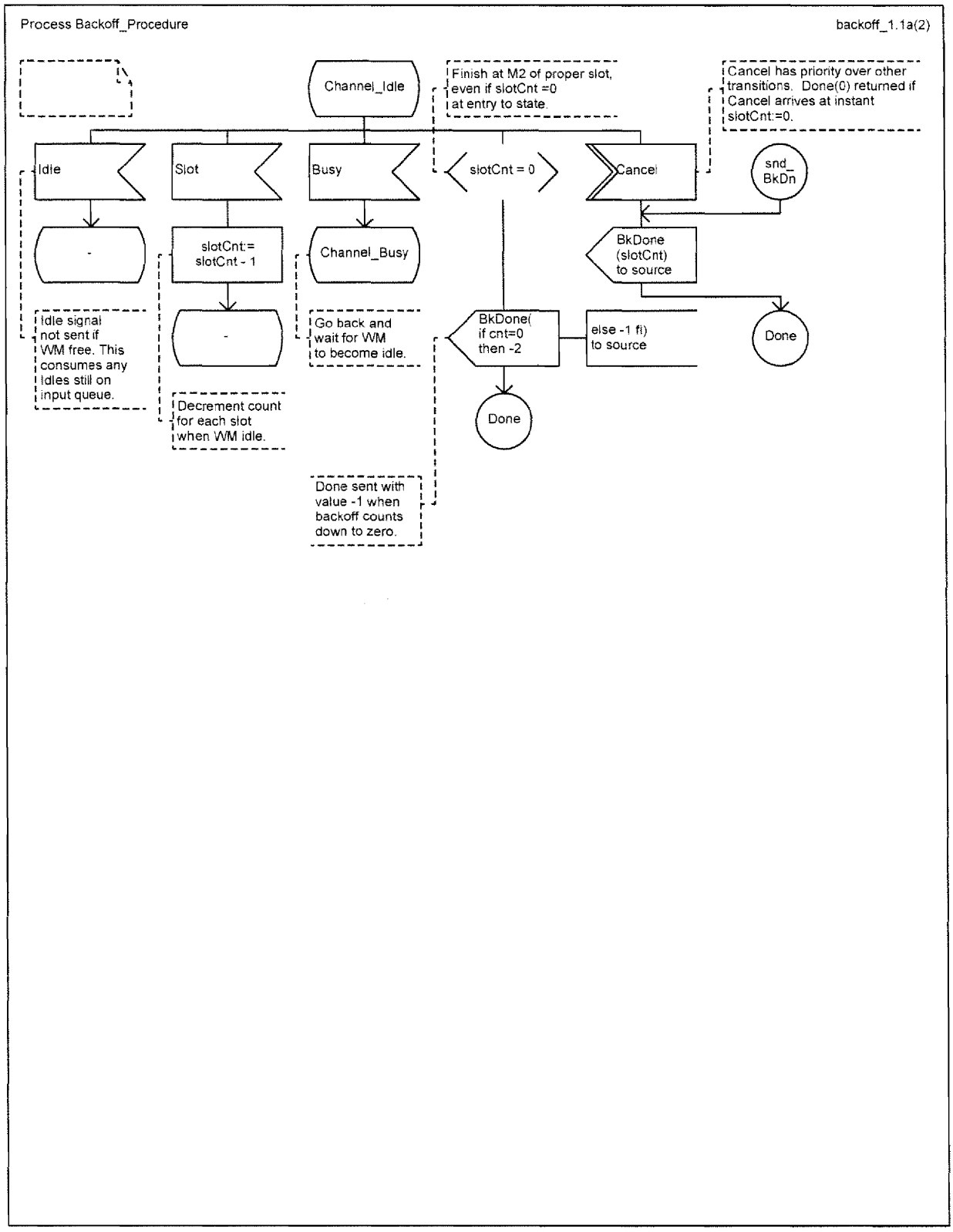


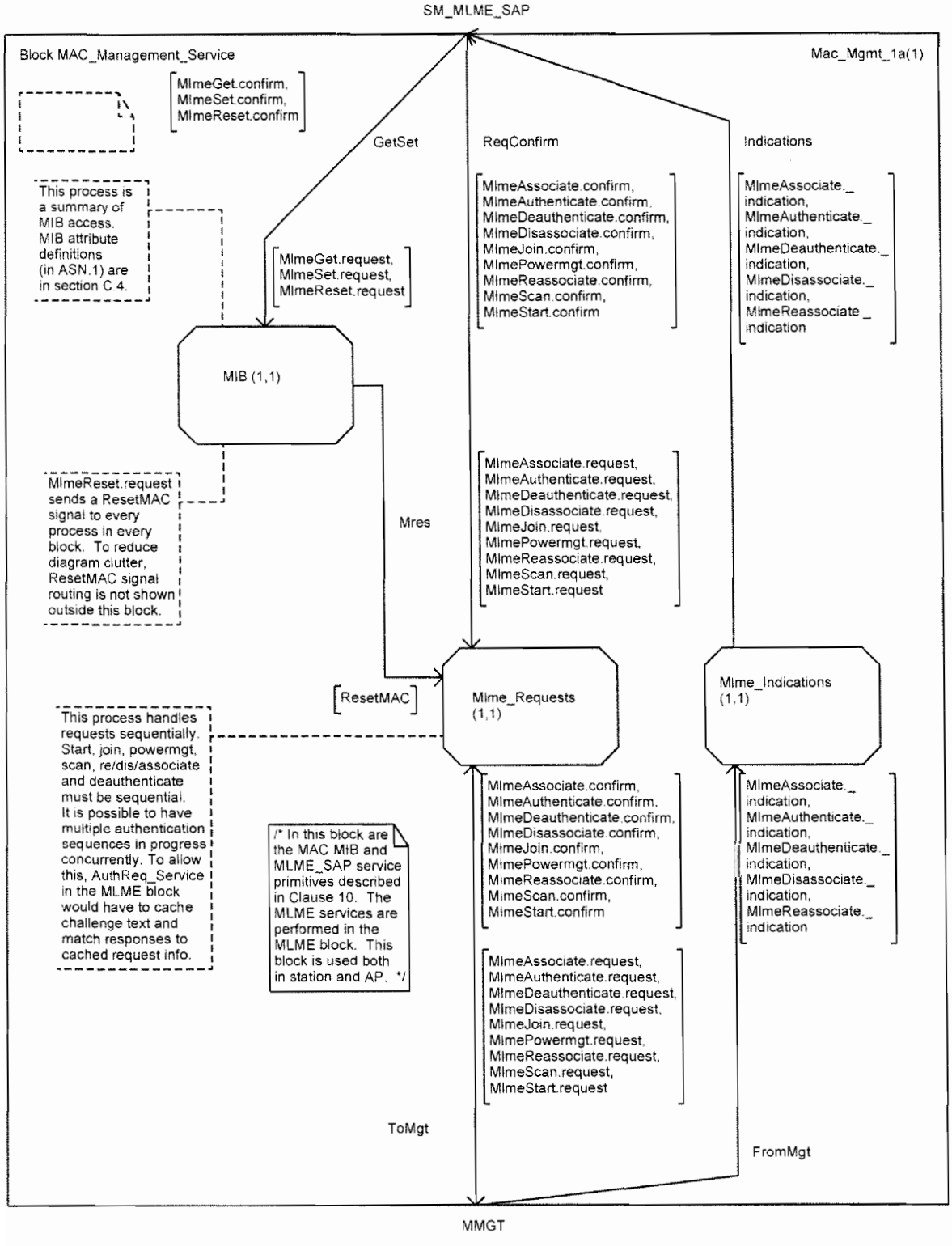


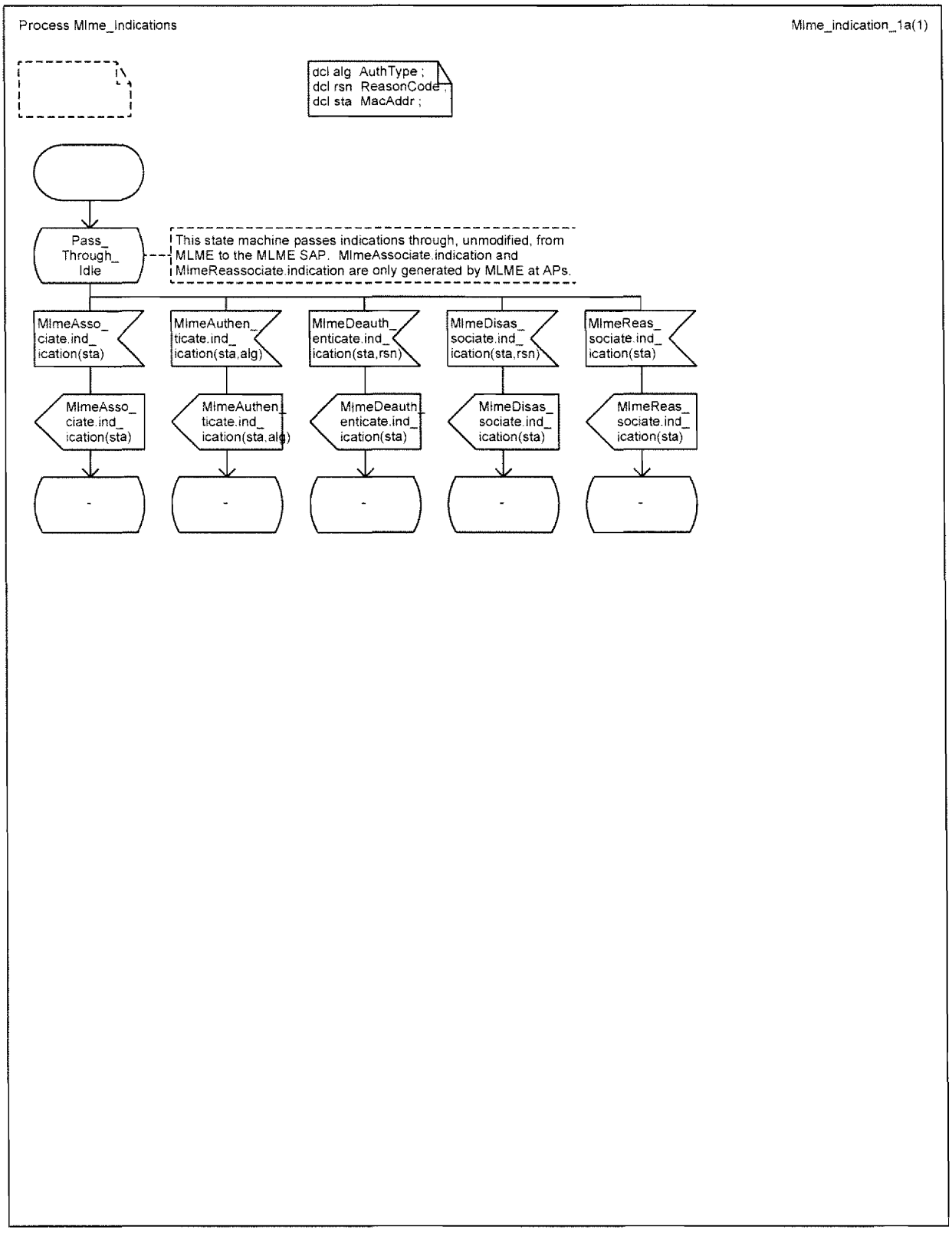


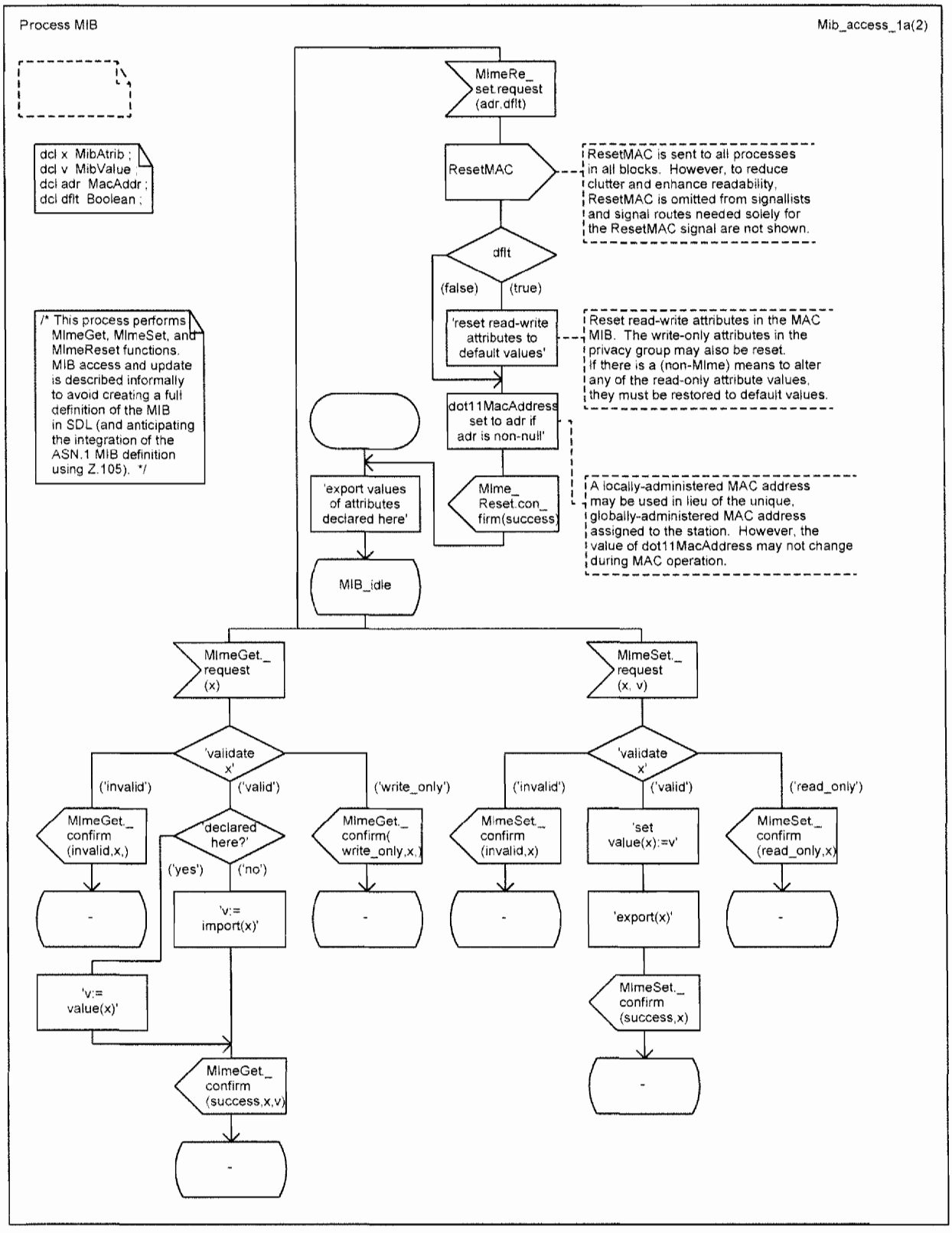












Process MIB

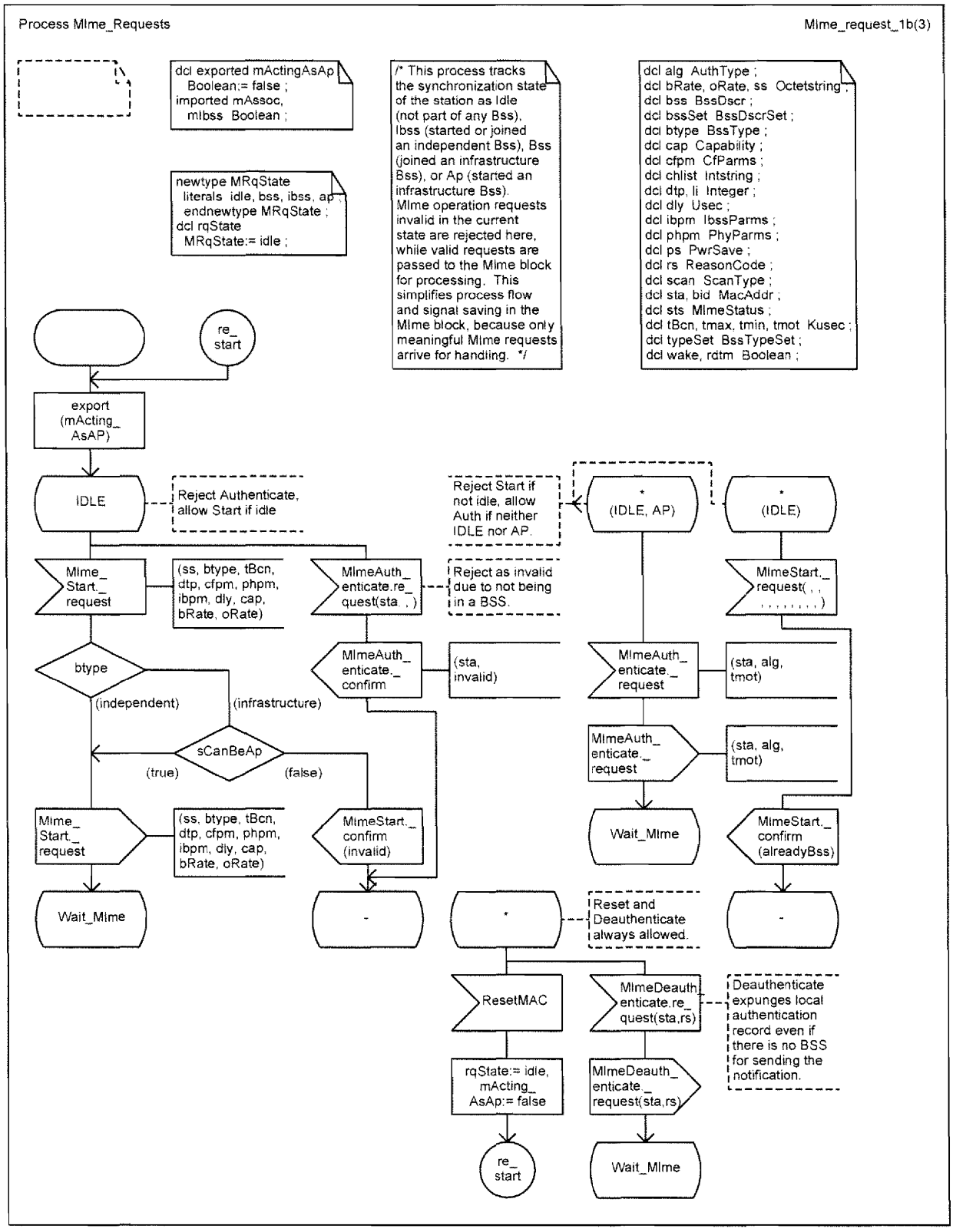
Mib_import_export_2b(2)

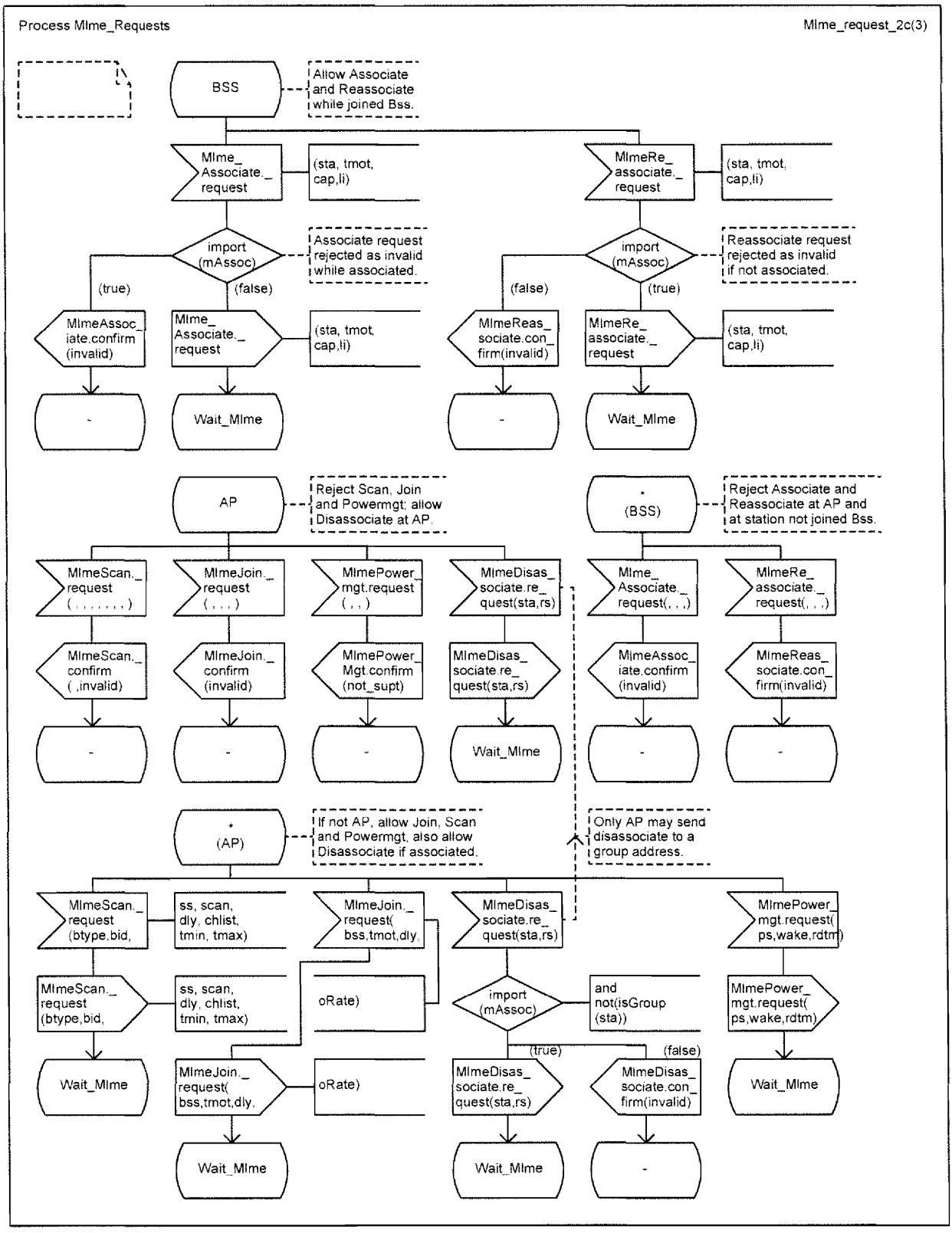
```
/* import of {Read-Only} MIB counter
   values exported from other processes */
imported
  dot11AckFailureCount,
  dot11FailedCount,
  dot11FcsErrorCount,
  dot11FrameDuplicateCount,
  dot11MulticastReceivedFrameCount,
  dot11MulticastTransmittedFrameCount,
  dot11MultipleRetryCount,
  dot11ReceivedFragmentCount,
  dot11RetryCount,
  dot11RtsFailureCount,
  dot11RtsSuccessCount,
  dot11TransmittedFragmentCount,
  dot11WepExcludedCount,
  dot11WepIcvErrorCount,
  dot11WepUndecryptableCount  Counter32 ;
```

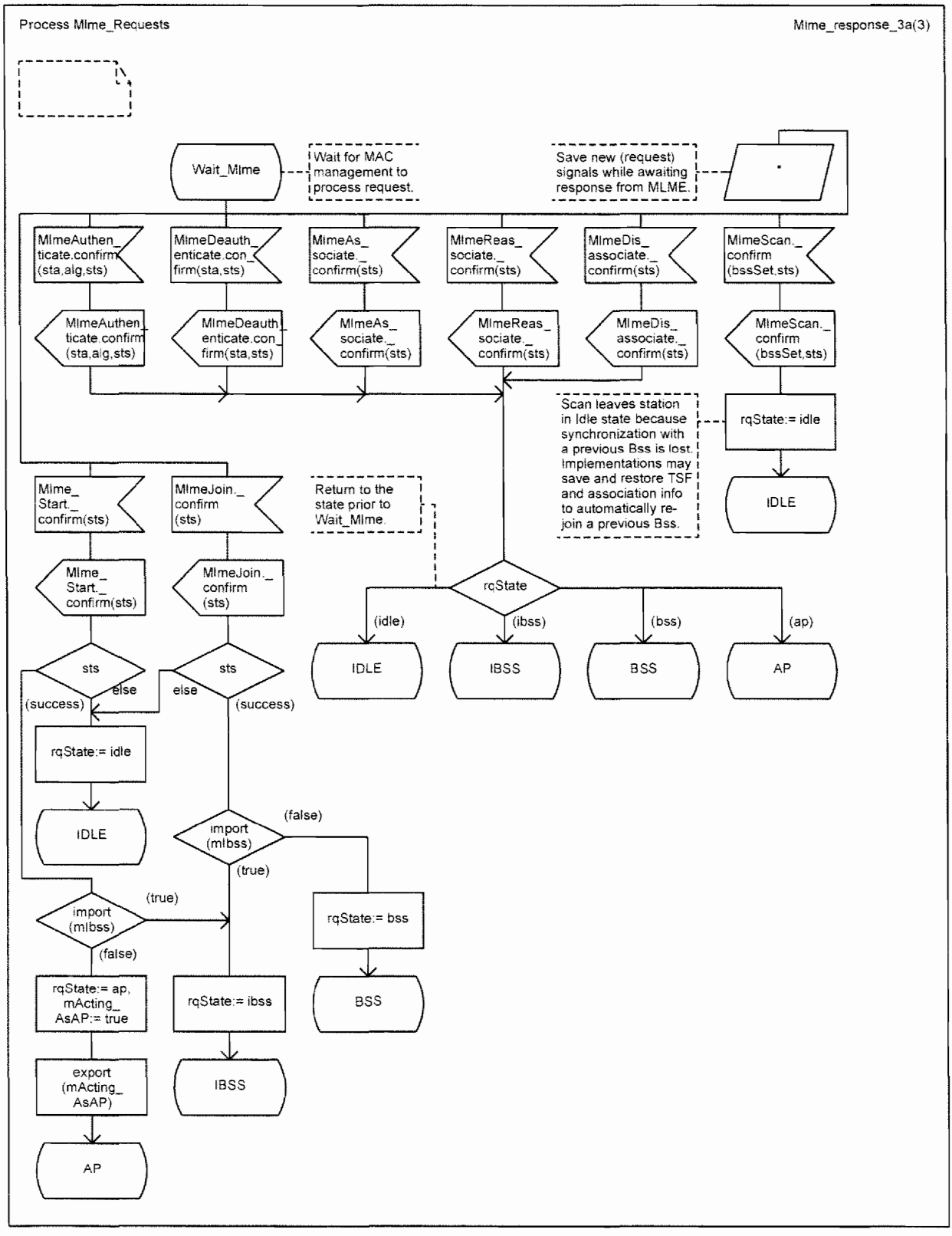
```
/* Declarations of MIB attributes exported from
   this process */

      /* Read-Write attributes */
dcl exported
  dot11AuthenticationAlgorithms  AuthTypeSet:=
    incl(open_system, shared_key),
  dot11ExcludeUnencrypted  Boolean:= false,
  dot11FragmentationThreshold  Integer:= 2346,
  dot11GroupAddresses  MacAddrSet:= empty,
  dot11LongRetryLimit  Integer:= 4,
  dot11MaxReceiveLifetime  Kusec:= 512,
  dot11MaxTransmitMsduLifetime  Kusec:= 512,
  dot11MediumOccupancyLimit  Kusec:= 100,
  dot11PrivacyInvoked  Boolean:= false,
  mReceiveDTIMs  Boolean:= true,
  dot11CfpPeriod  Integer:= 1,
  dot11CfpMaxDuration  Kusec:= 200,
  dot11AuthenticationResponseTimeout  Kusec:= 512,
  dot11RtsThreshold  Integer:= 3000,
  dot11ShortRetryLimit  Integer:= 7,
  dot11WepDefaultKeyId  KeyIndex:= 0,
  dot11CurrentChannelNumber  Integer:= 0,
  dot11CurrentSet  Integer:= 0,
  dot11CurrentPattern  Integer:= 0,
  dot11CurrentIndex  Integer:= 0 ;

      /* Write-Only attributes */
dcl exported
  dot11WepDefaultKeys  KeyVector:= nullKey,
  dot11WepKeyMappings
    KeyMapArray:= (. nullAddr, false, nullKey .) ;
```

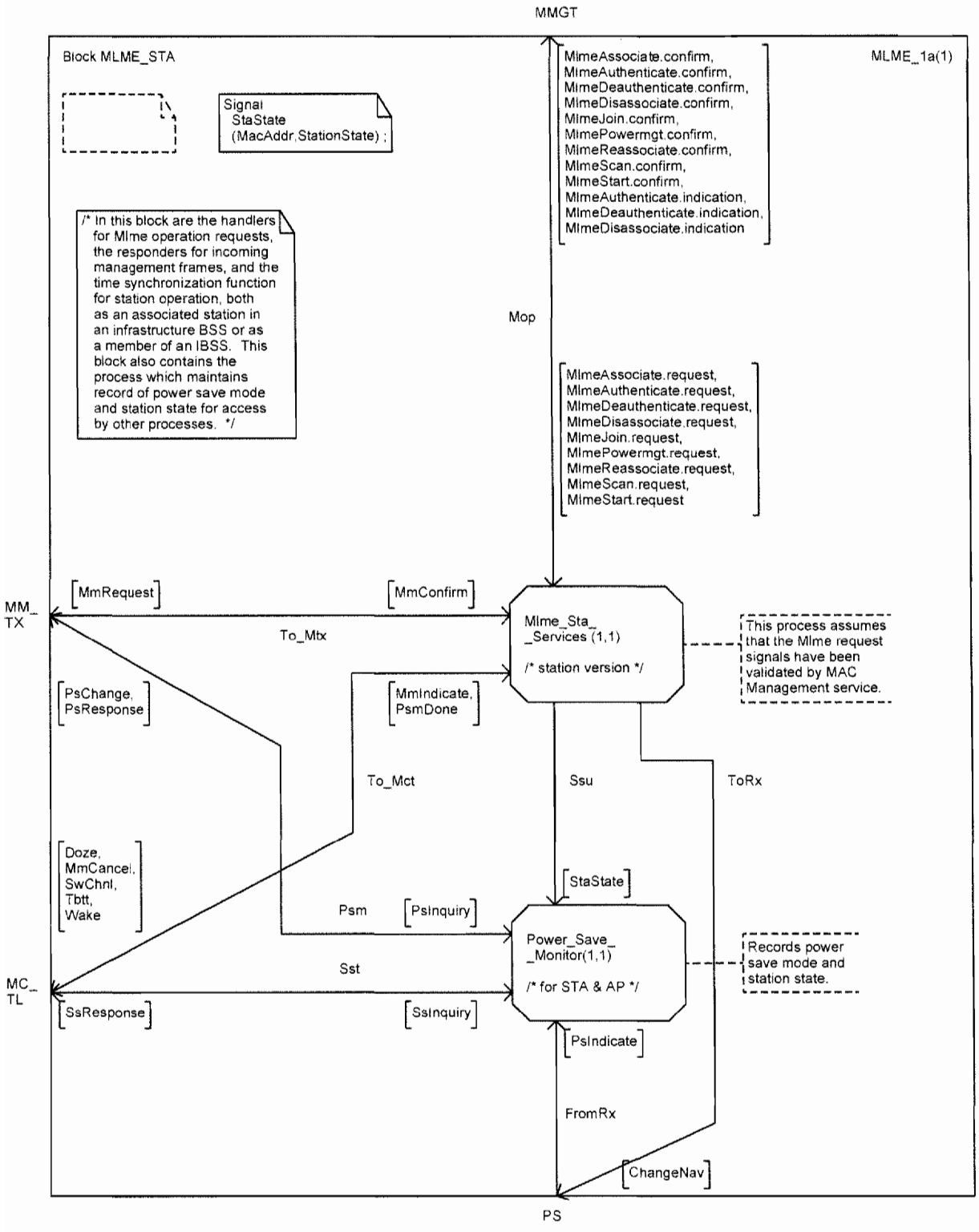
```
/* The following Read-Only attributes in the
   MAC MIB are defined as synonyms (named
   constants) rather than remote variables
   because they describe properties of the
   station which are static, at least during
   any single instance of MAC operation:
     dot11AuthenticationAlgorithms  AuthTypeSet,
     dot11CfPollable  Boolean,
     dot11MacAddress  MacAddr,
     dot11ManufacturerID  Octetstring,
     dot11PrivacyOptionImplemented  Boolean,
     dot11ProductID  Octetstring,
     aStationID  MacAddr,
     dot11WepKeyMappingLength  Integer ;

   In addition, all Read-Only attributes in the
   PHY MIB which are accessed by the MAC
   are defined as synonyms.
*/
```

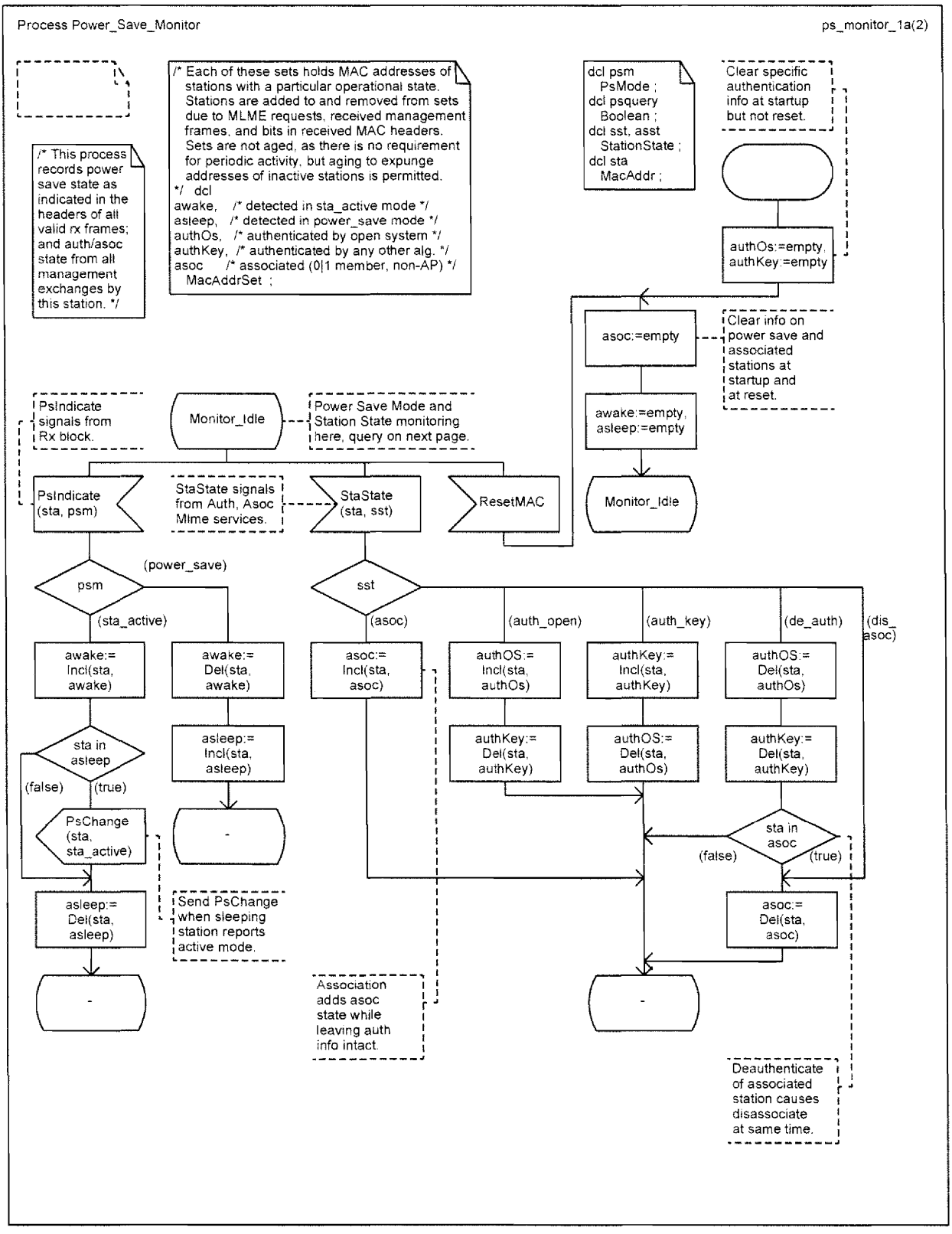
```
/* NOTE:
   The values listed for MAC MIB attributes are the
   specified default values for those attributes.
   The values listed for PHY MIB attributes are either
   the default values for the FH PHY, or arbitrary
   values within the specified range.  The specific
   values for PHY attributes in this SDL description
   of the MAC do not have normative significance.
*/
```

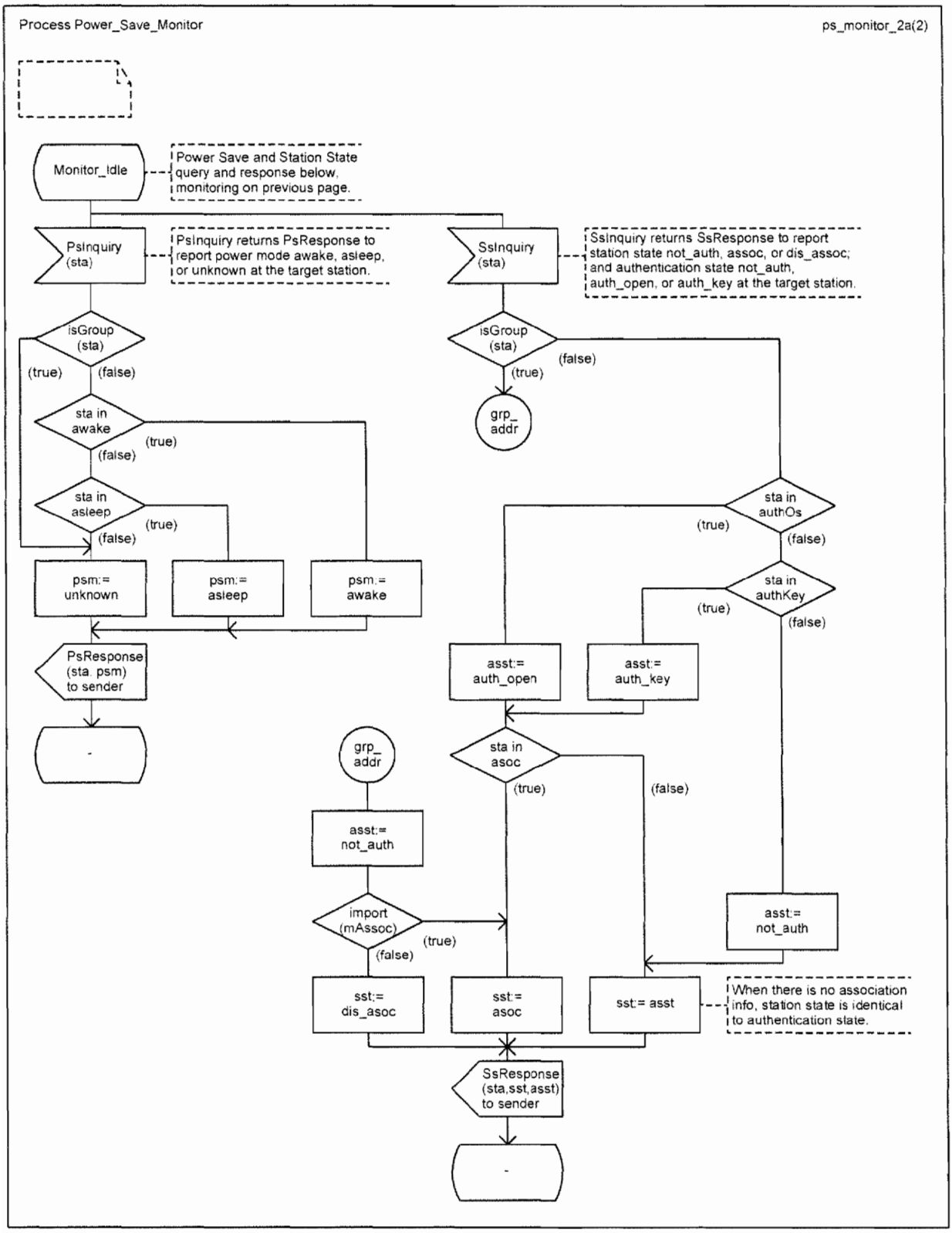


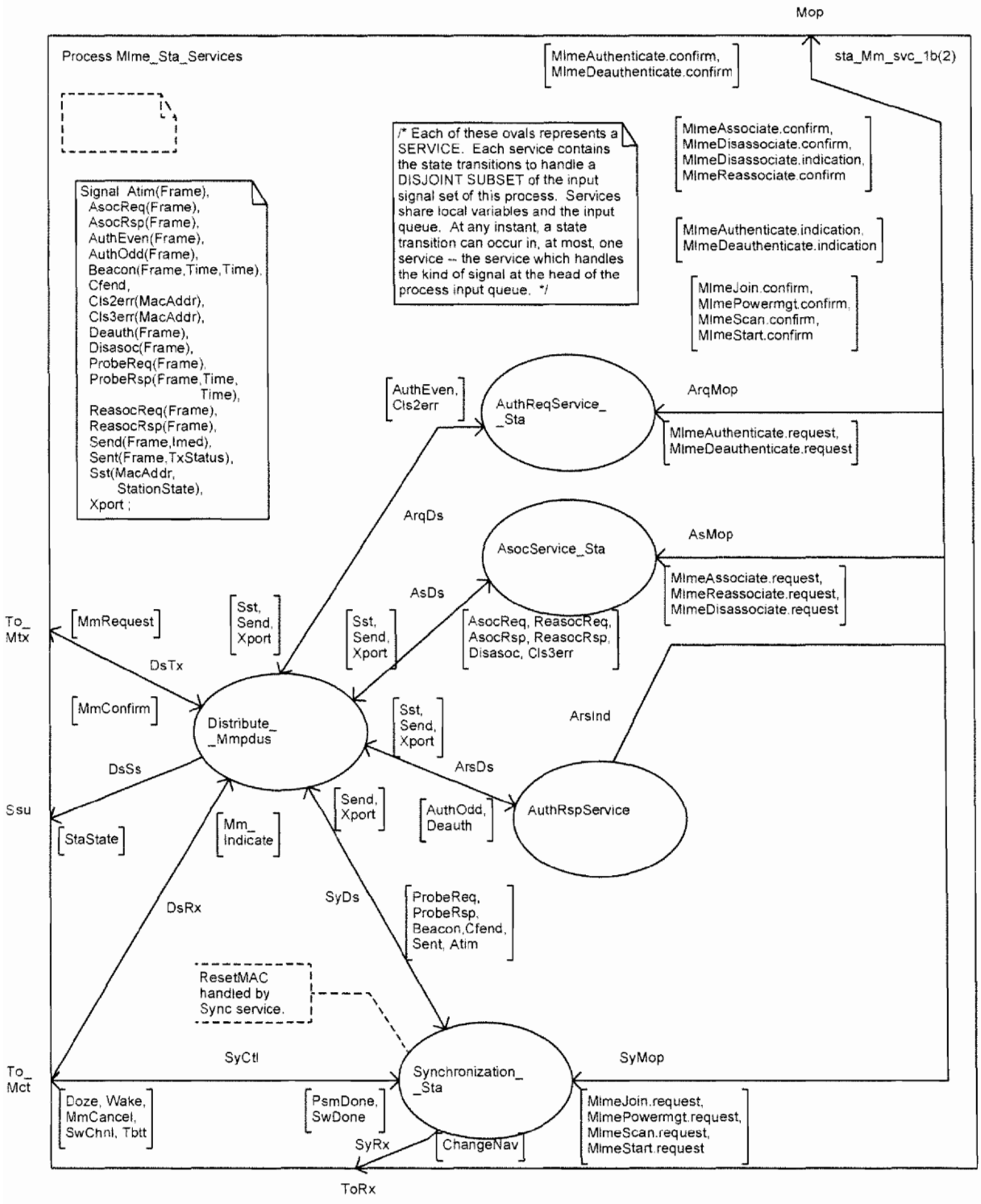


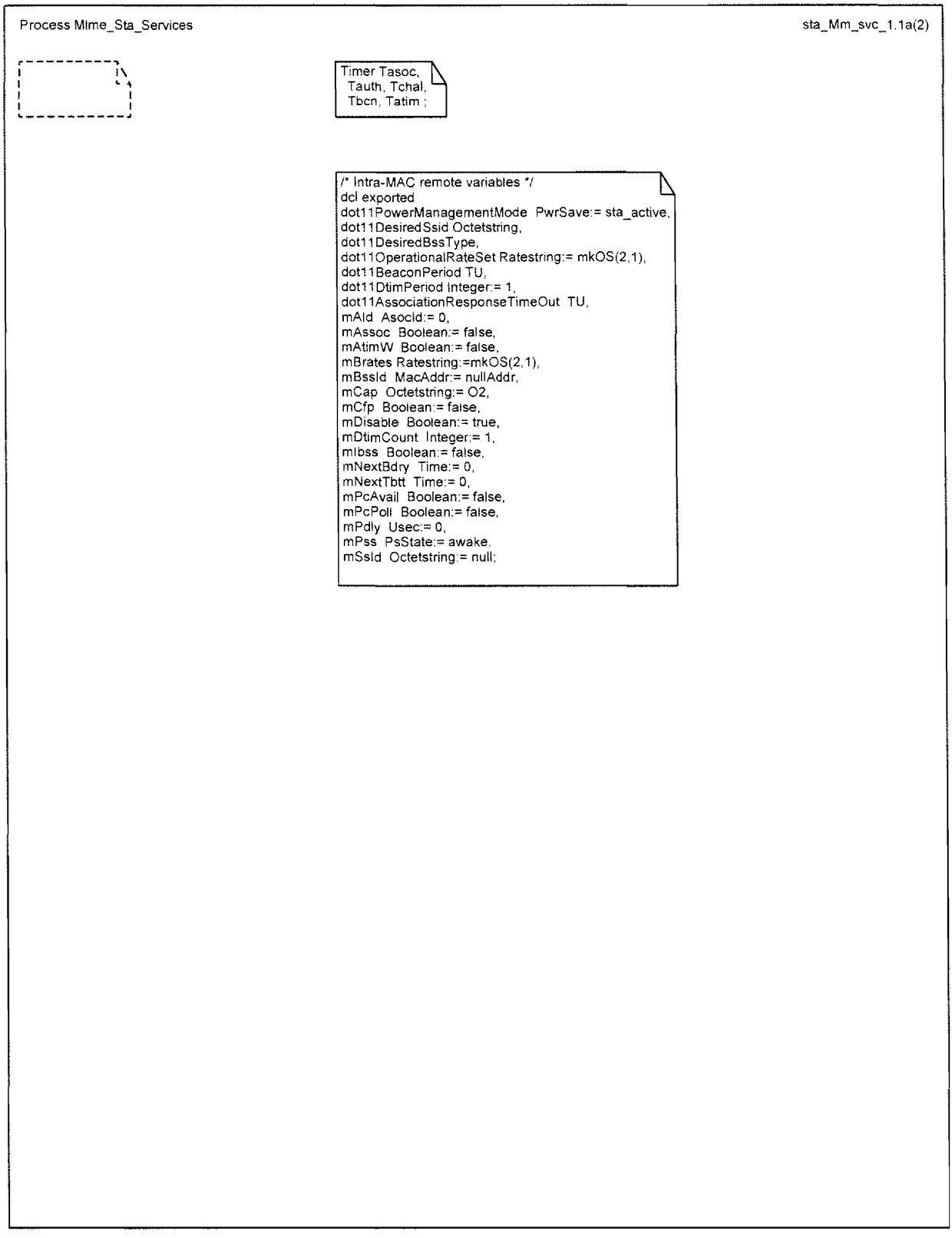


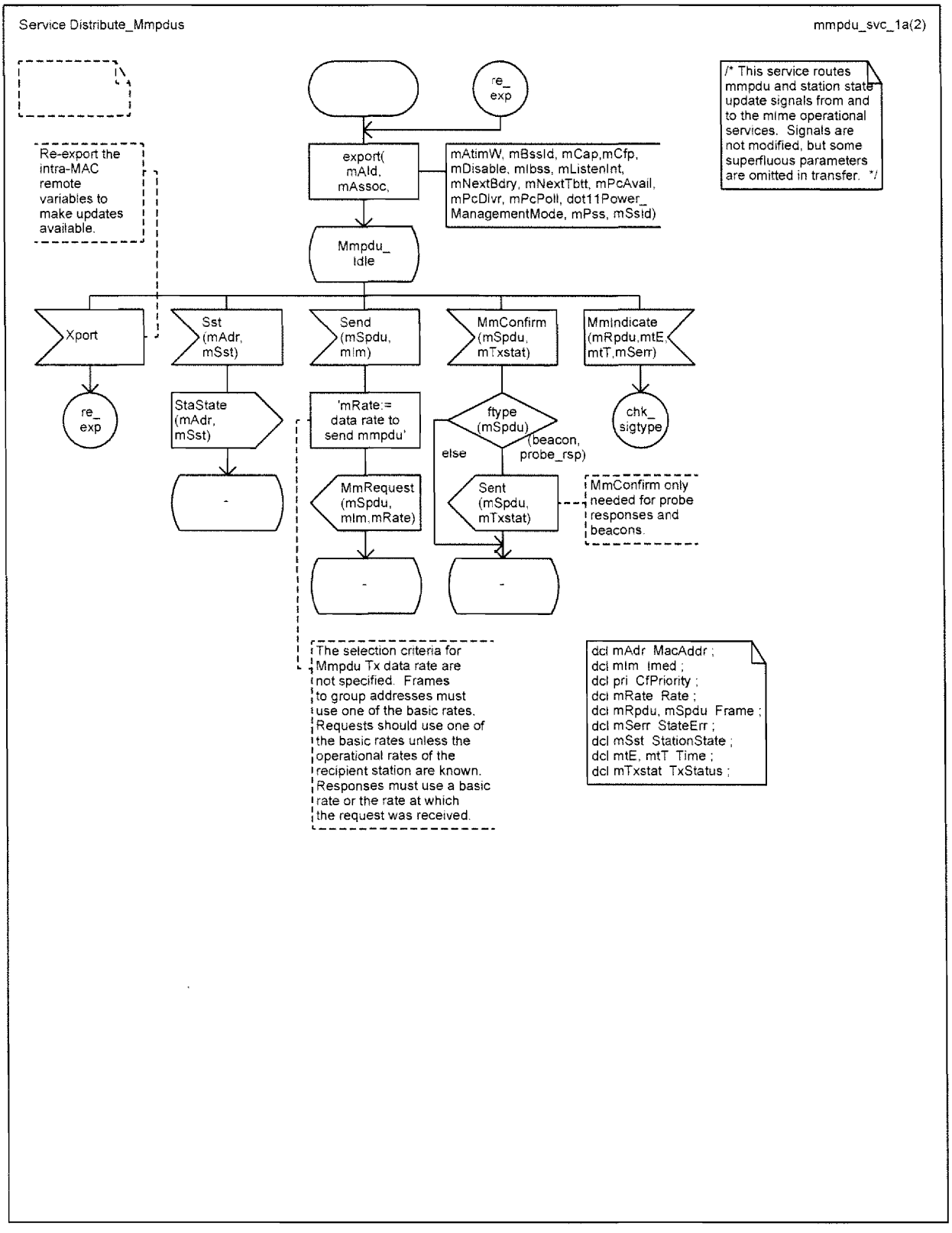


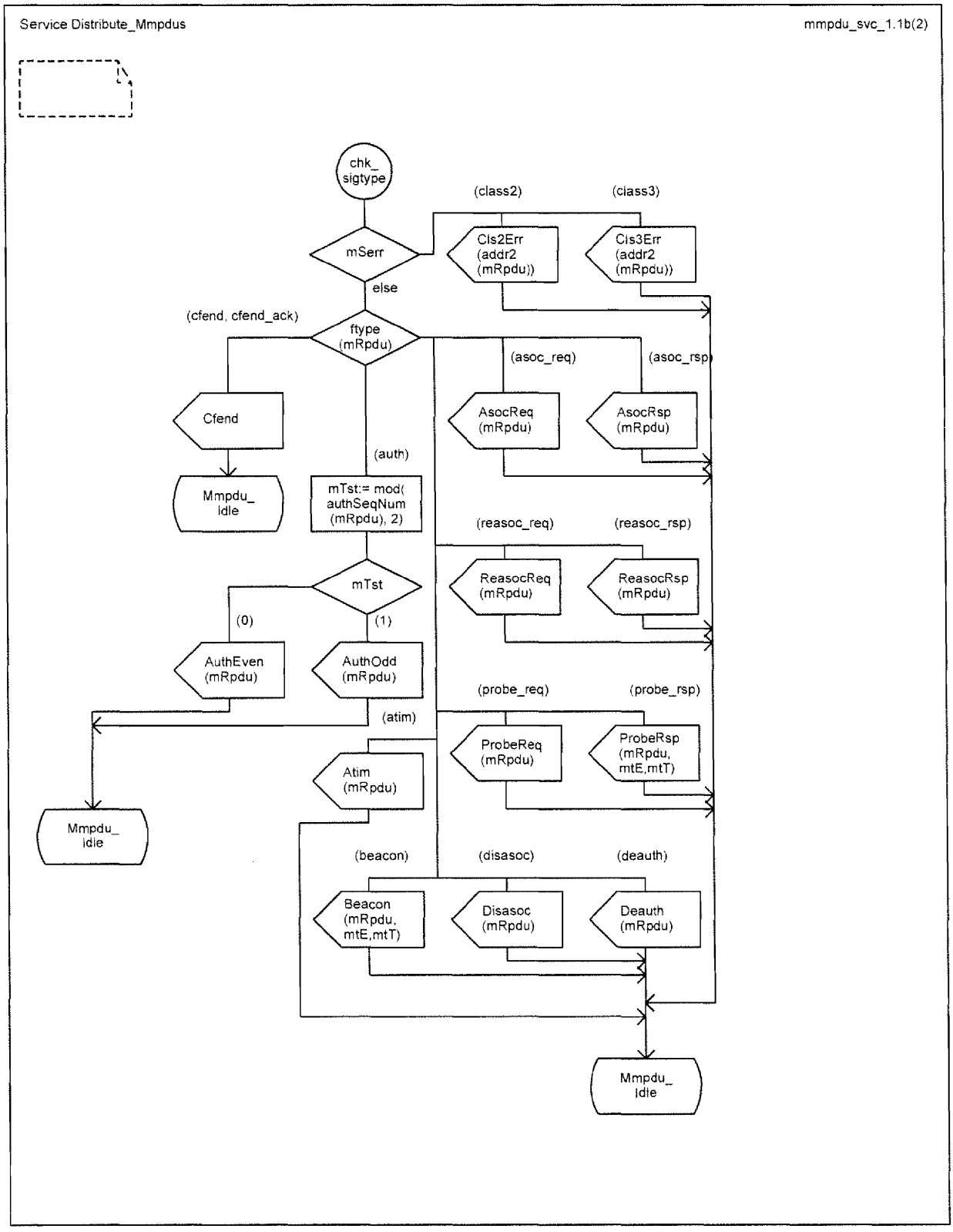


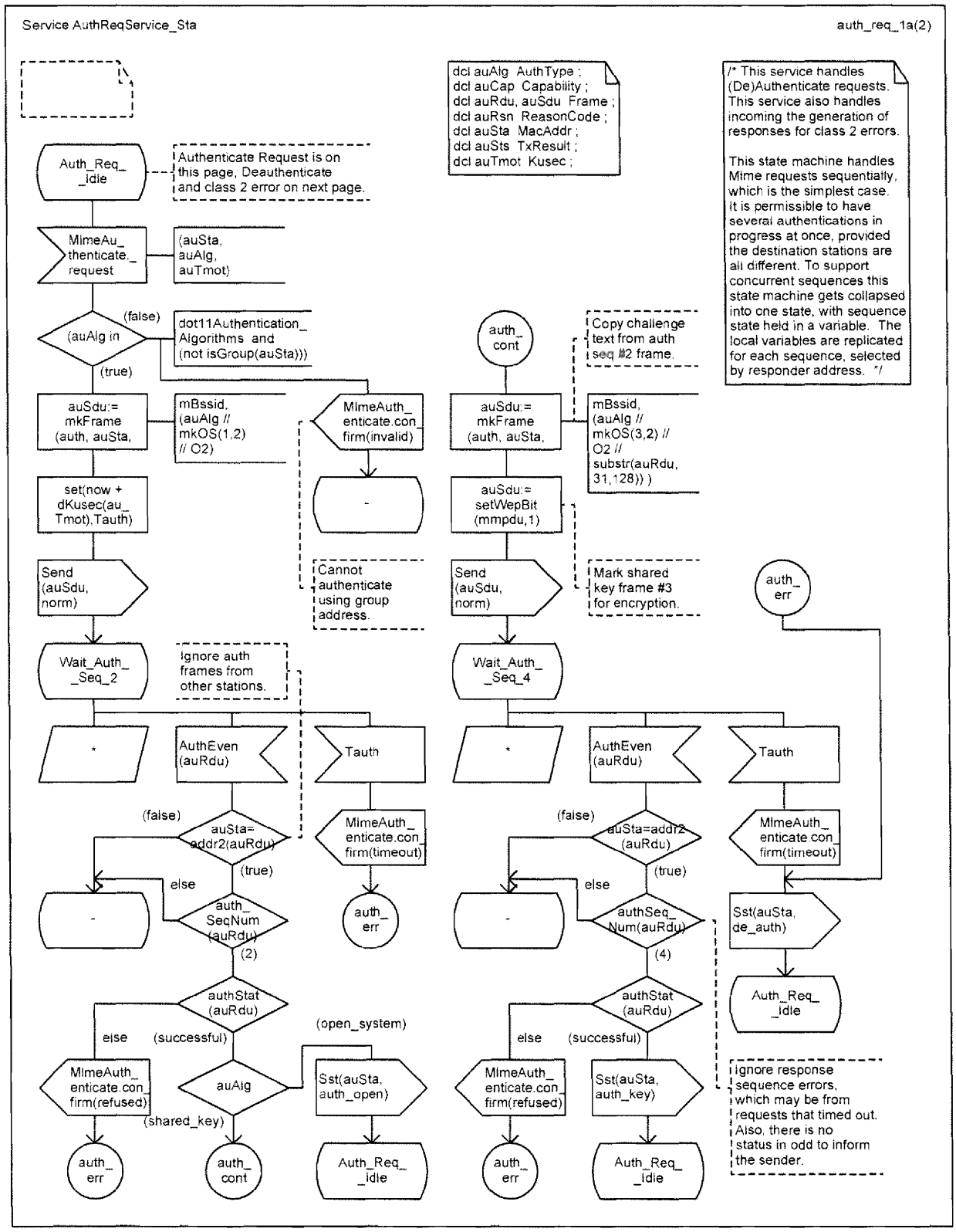


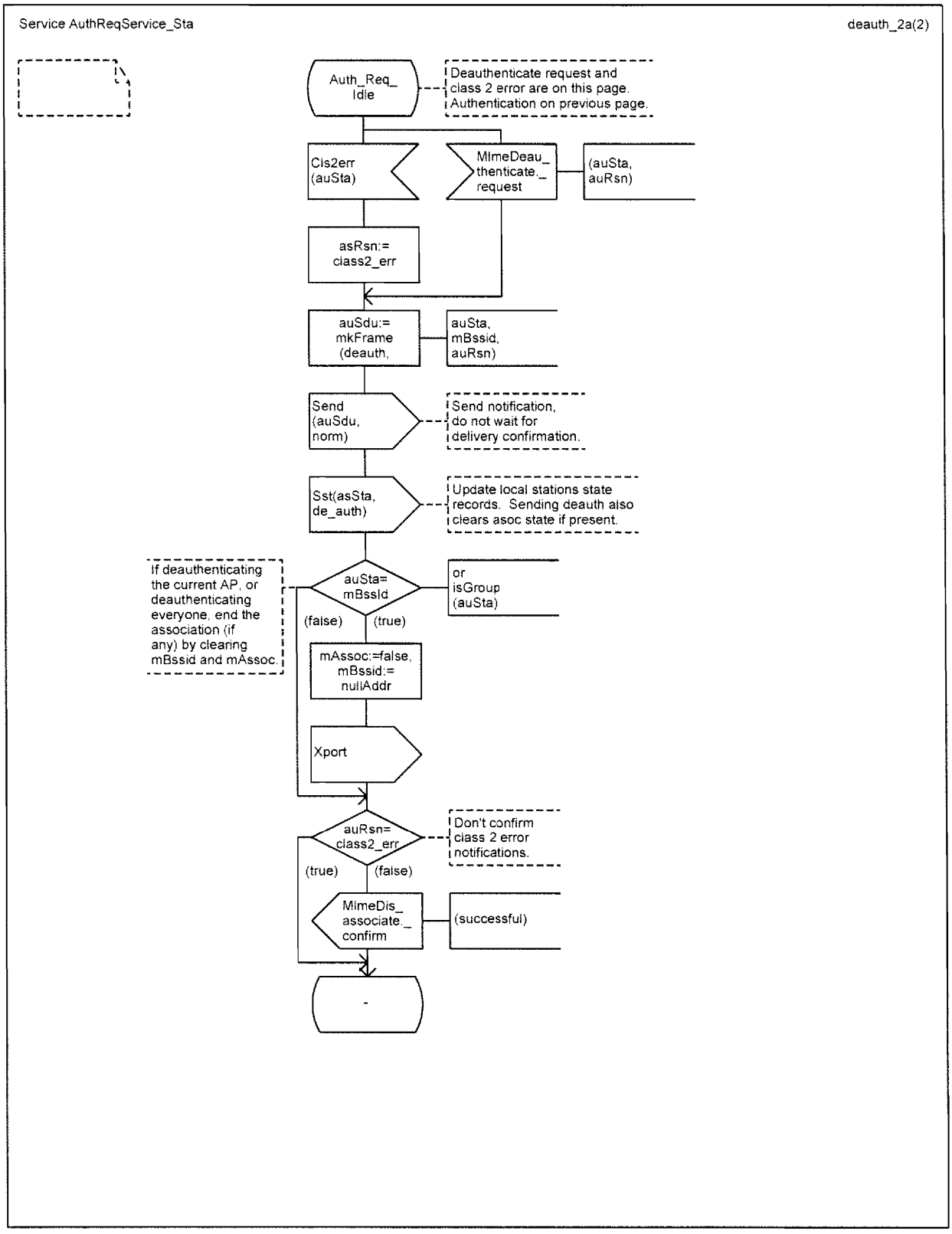


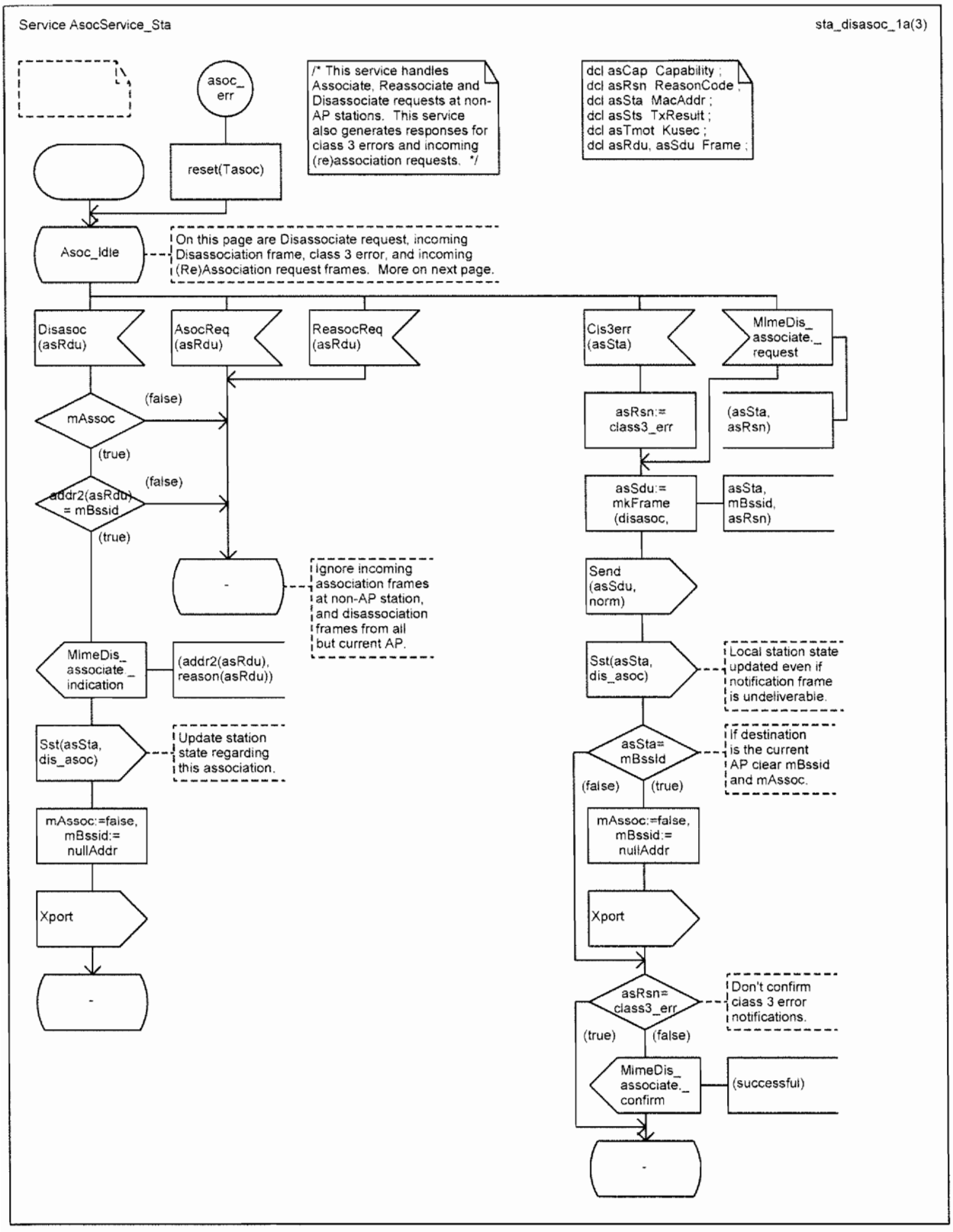


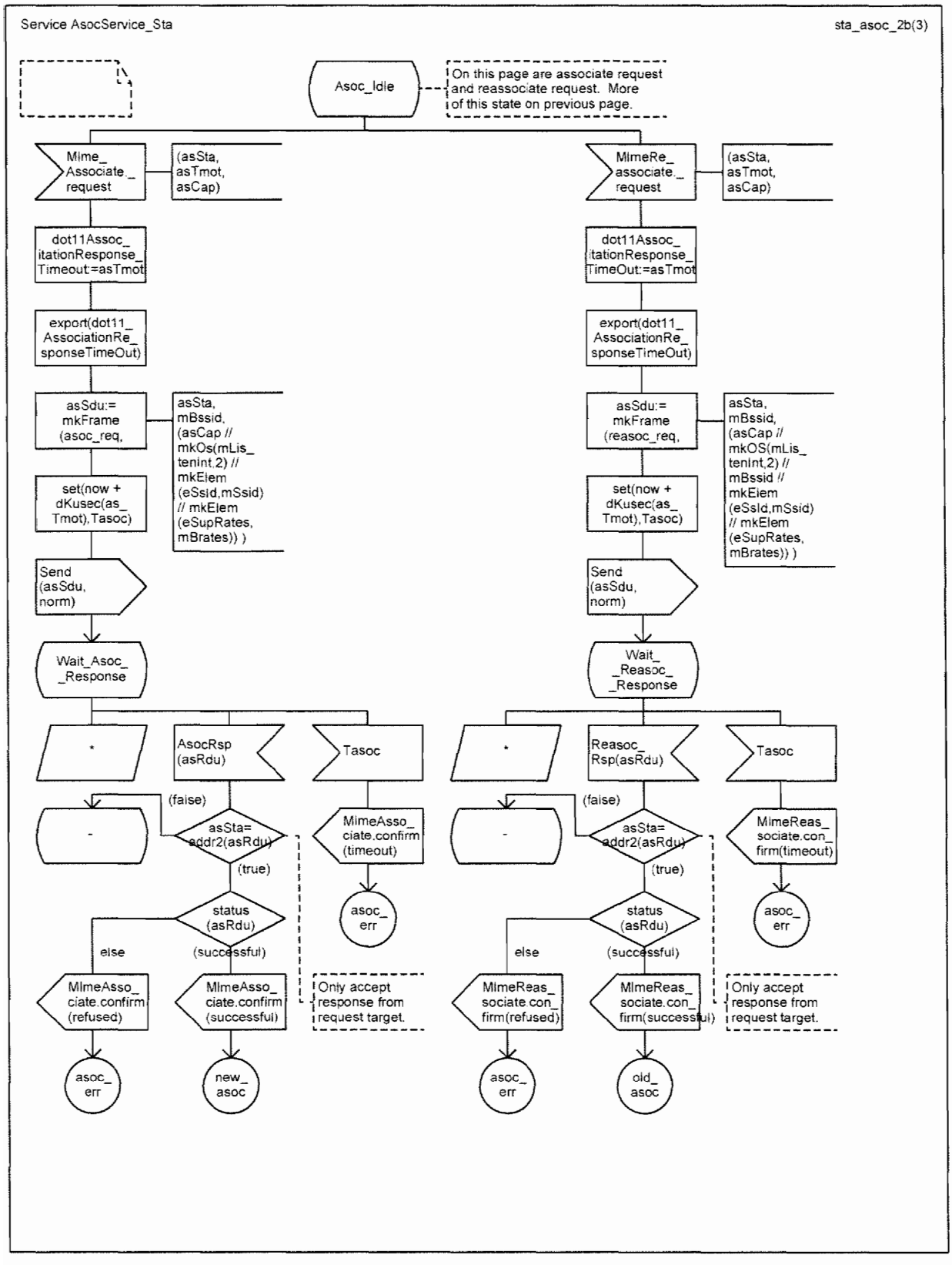


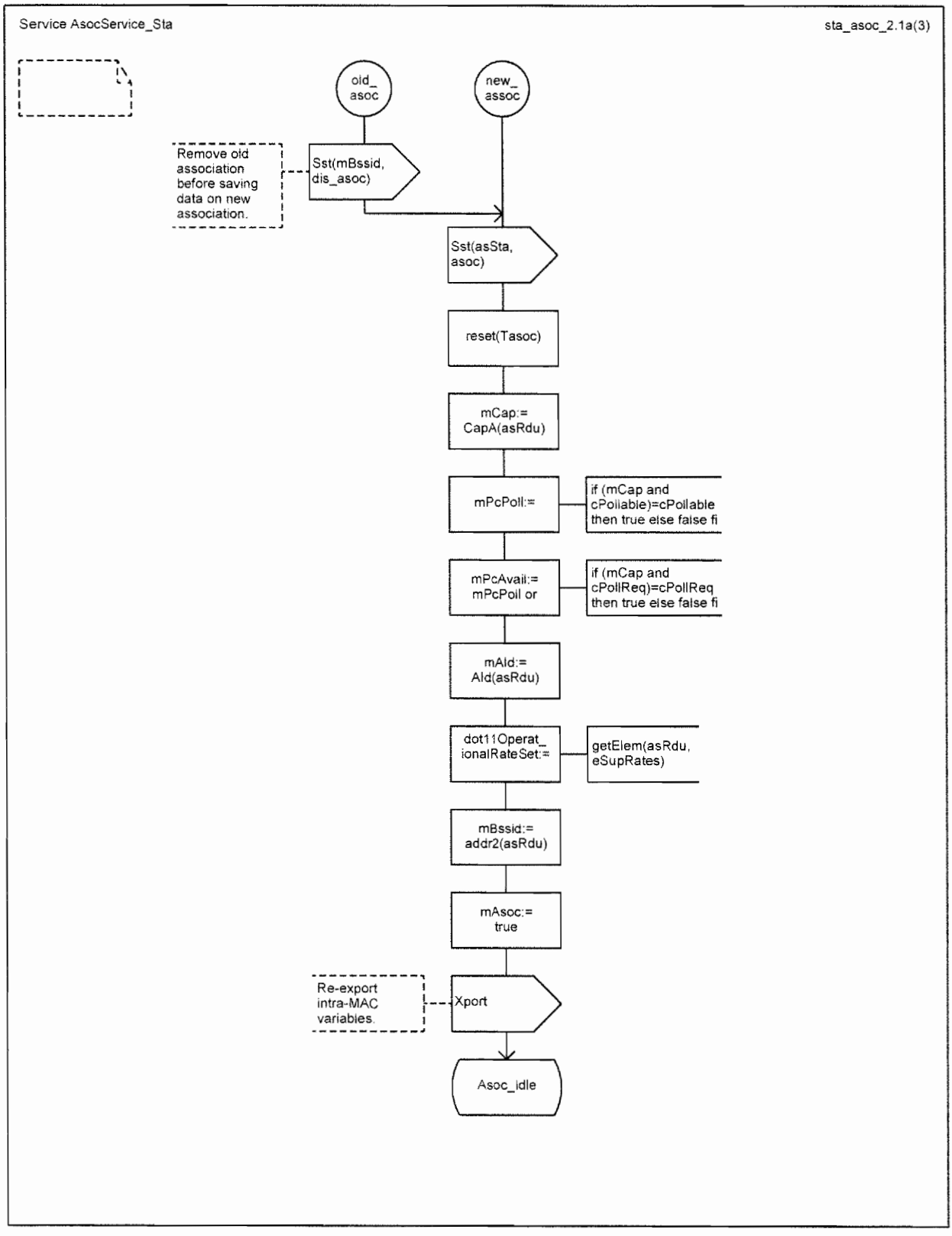


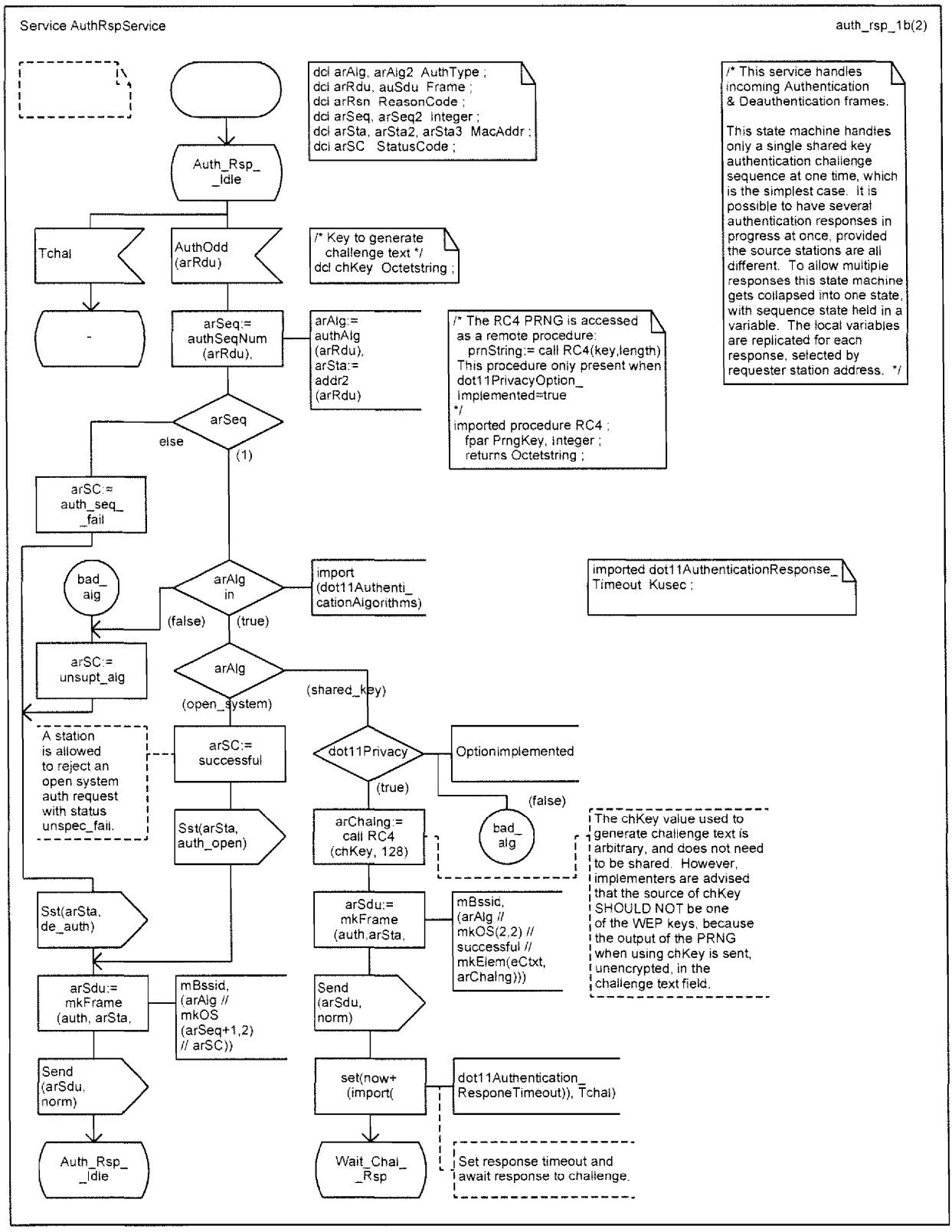


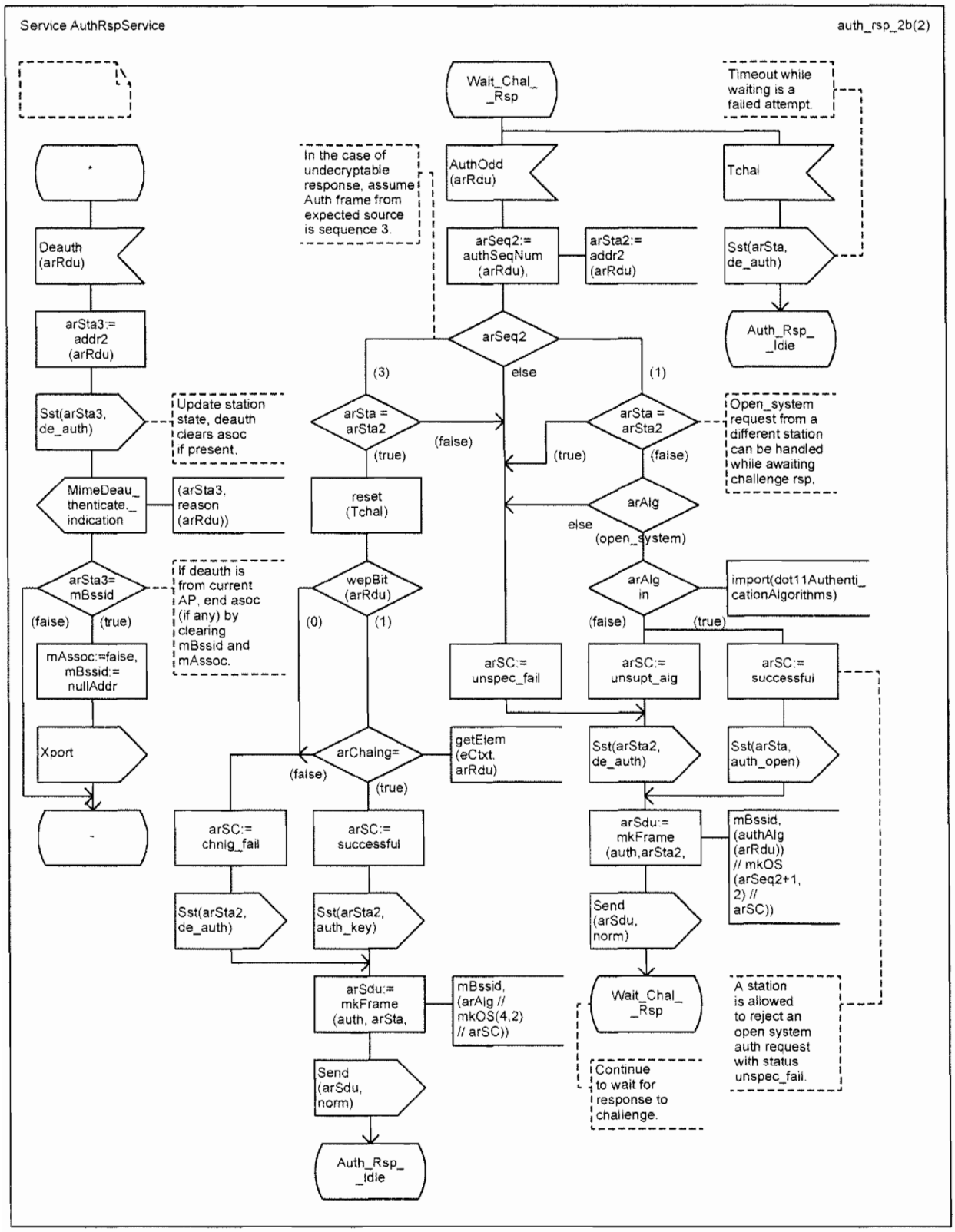


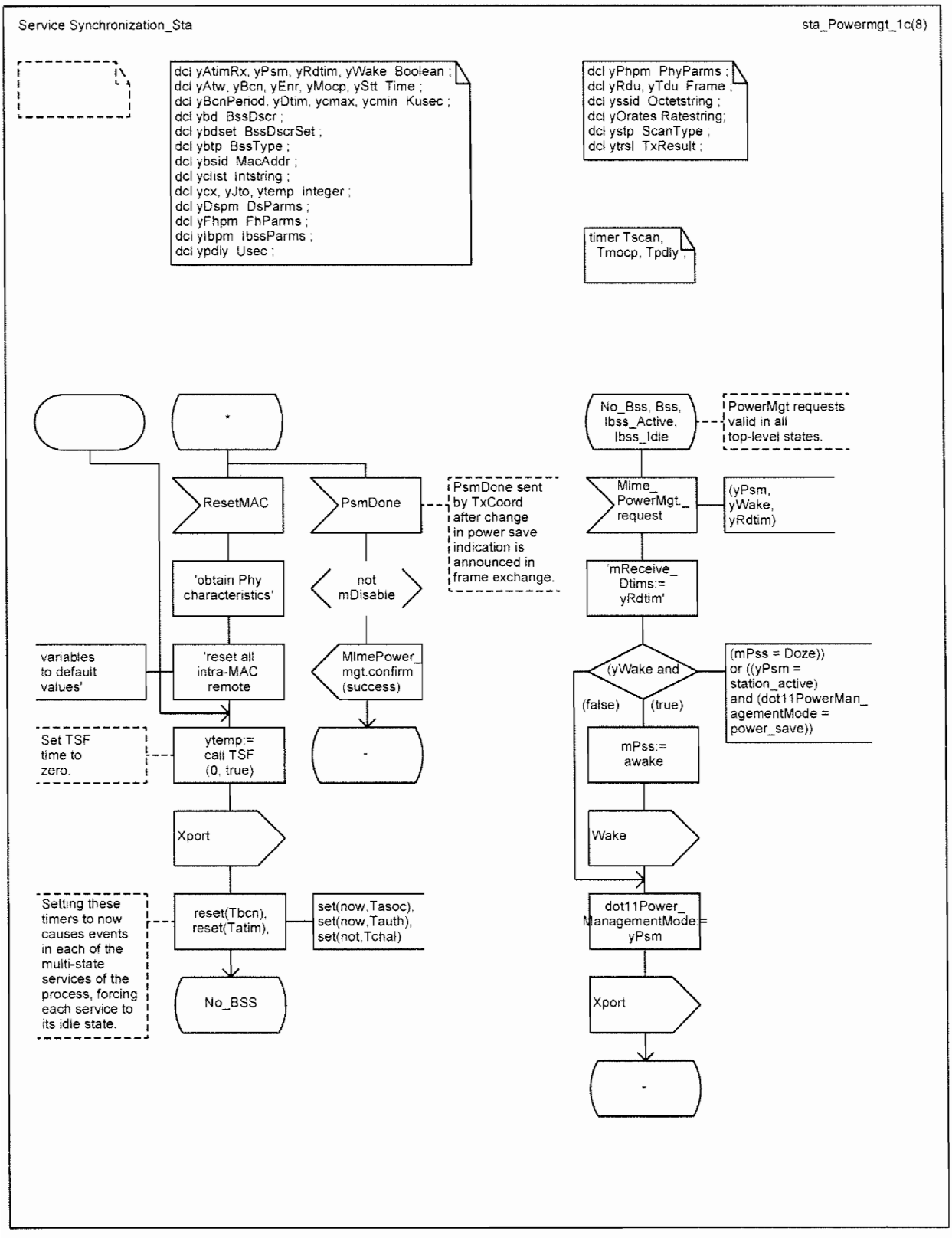


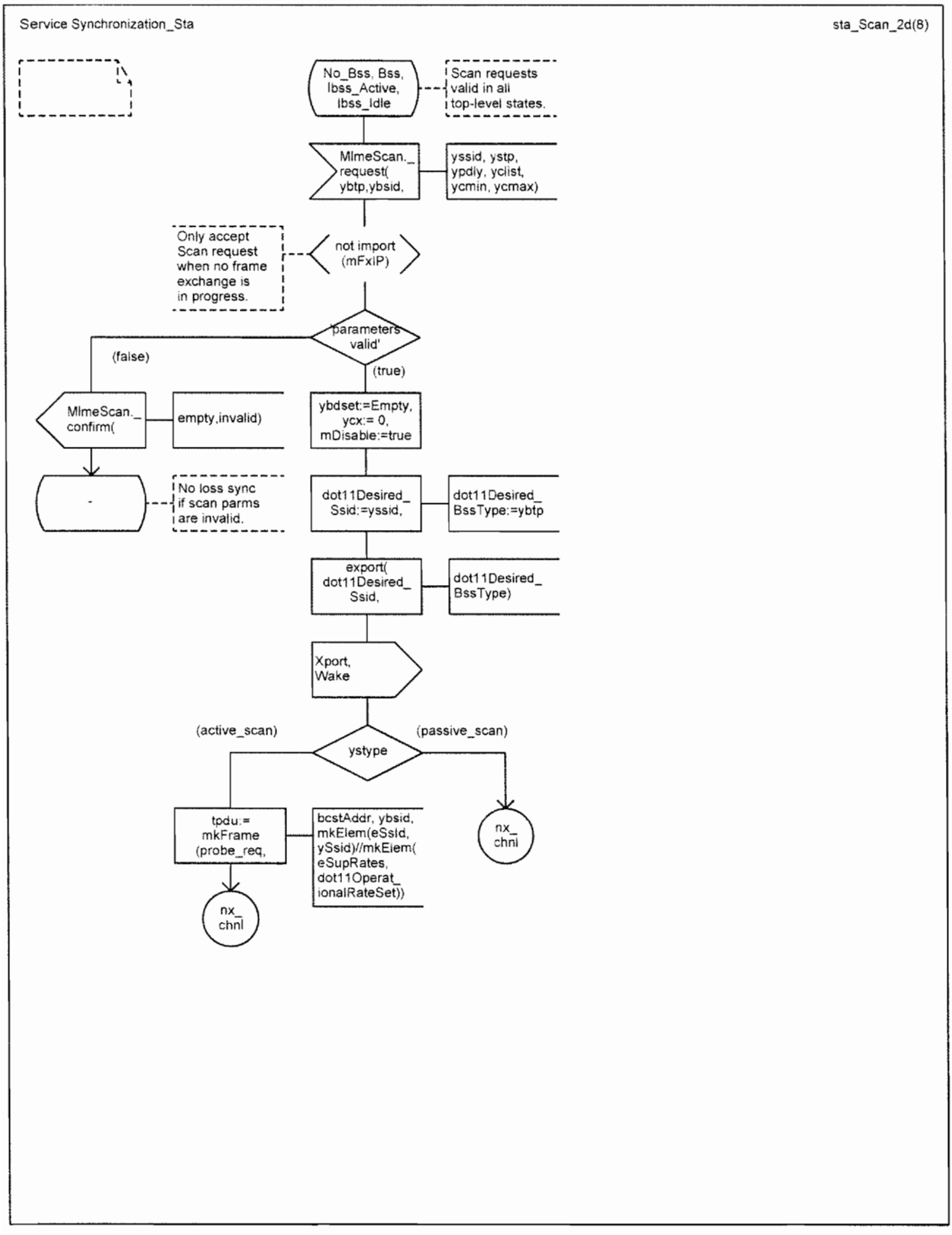


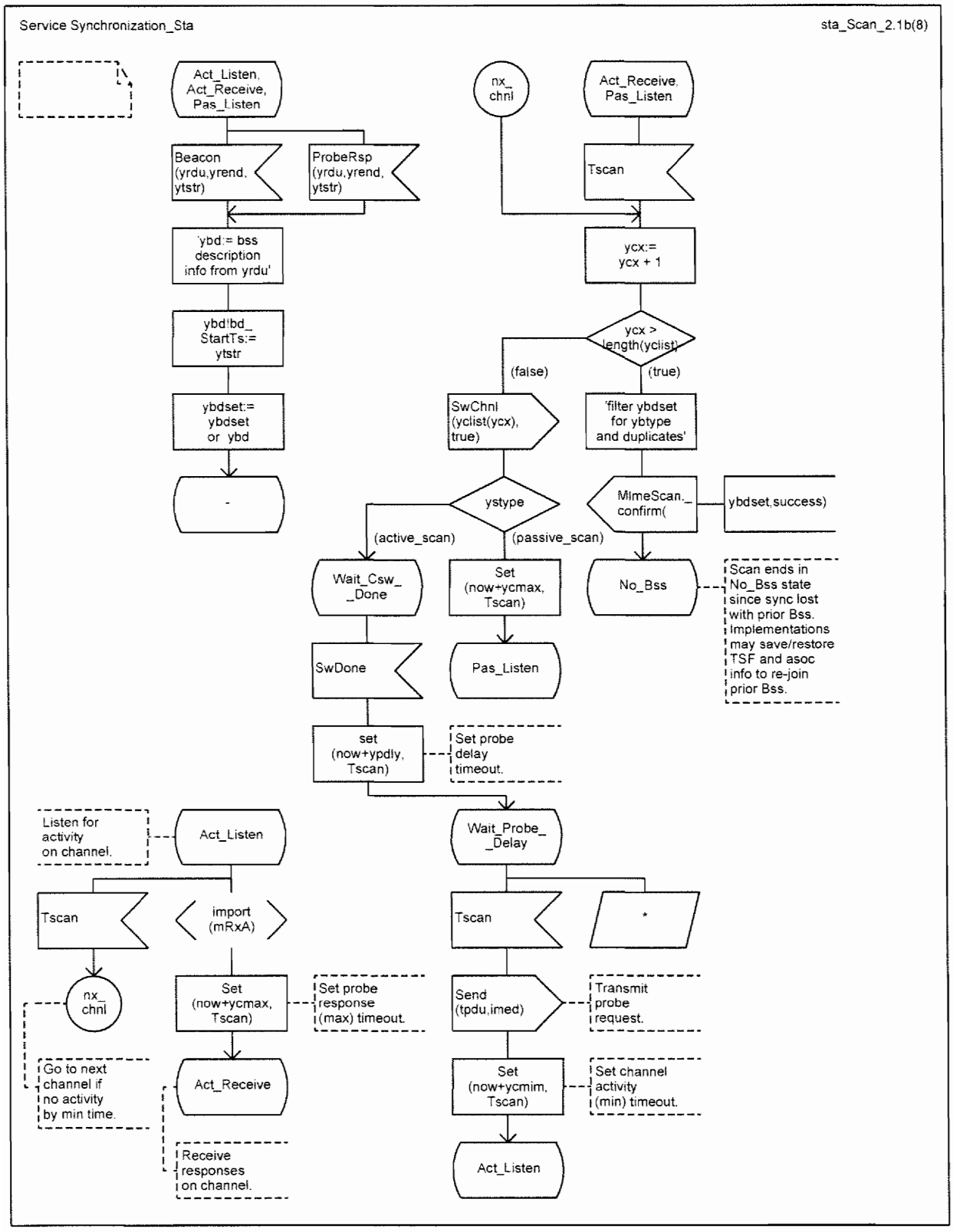


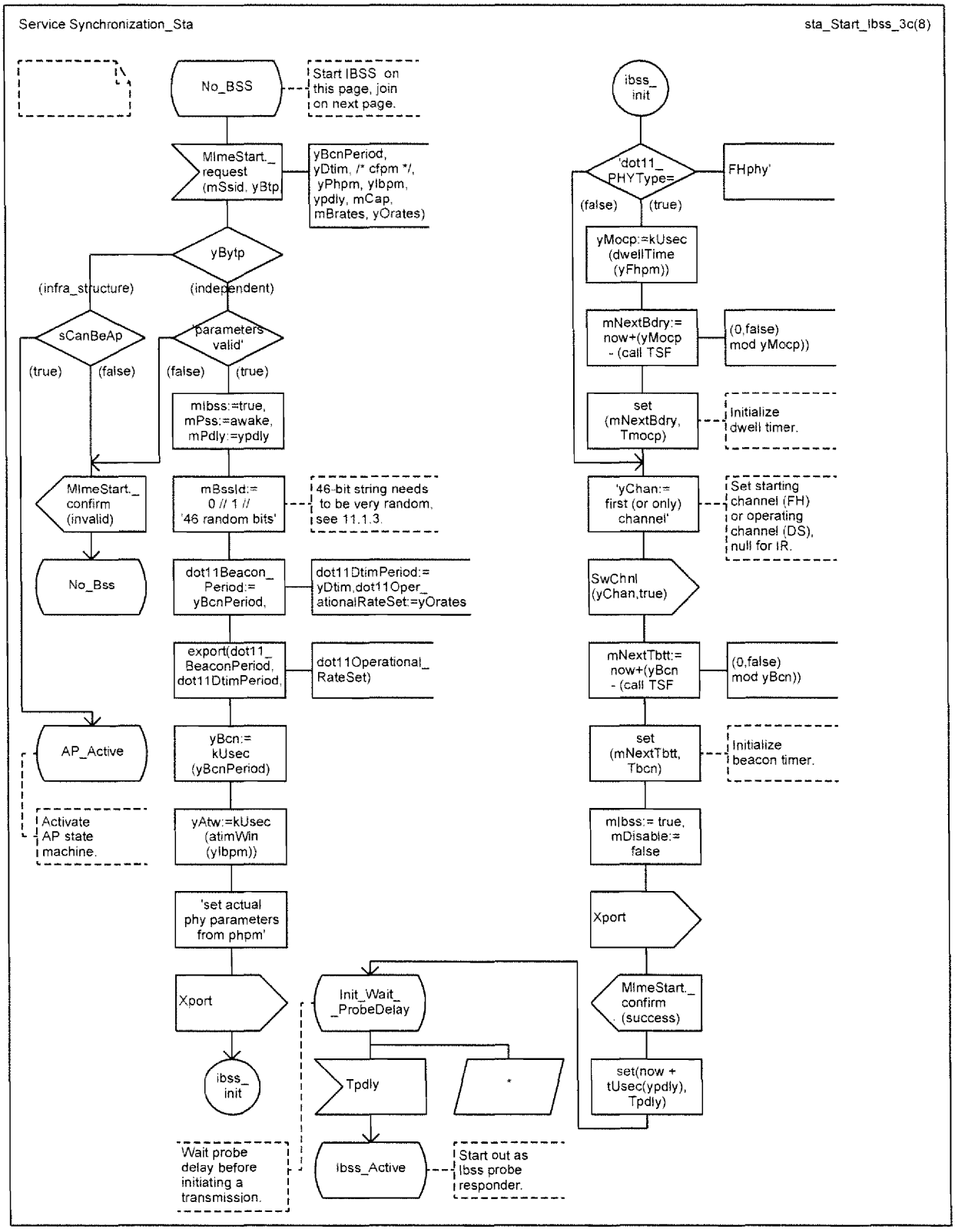


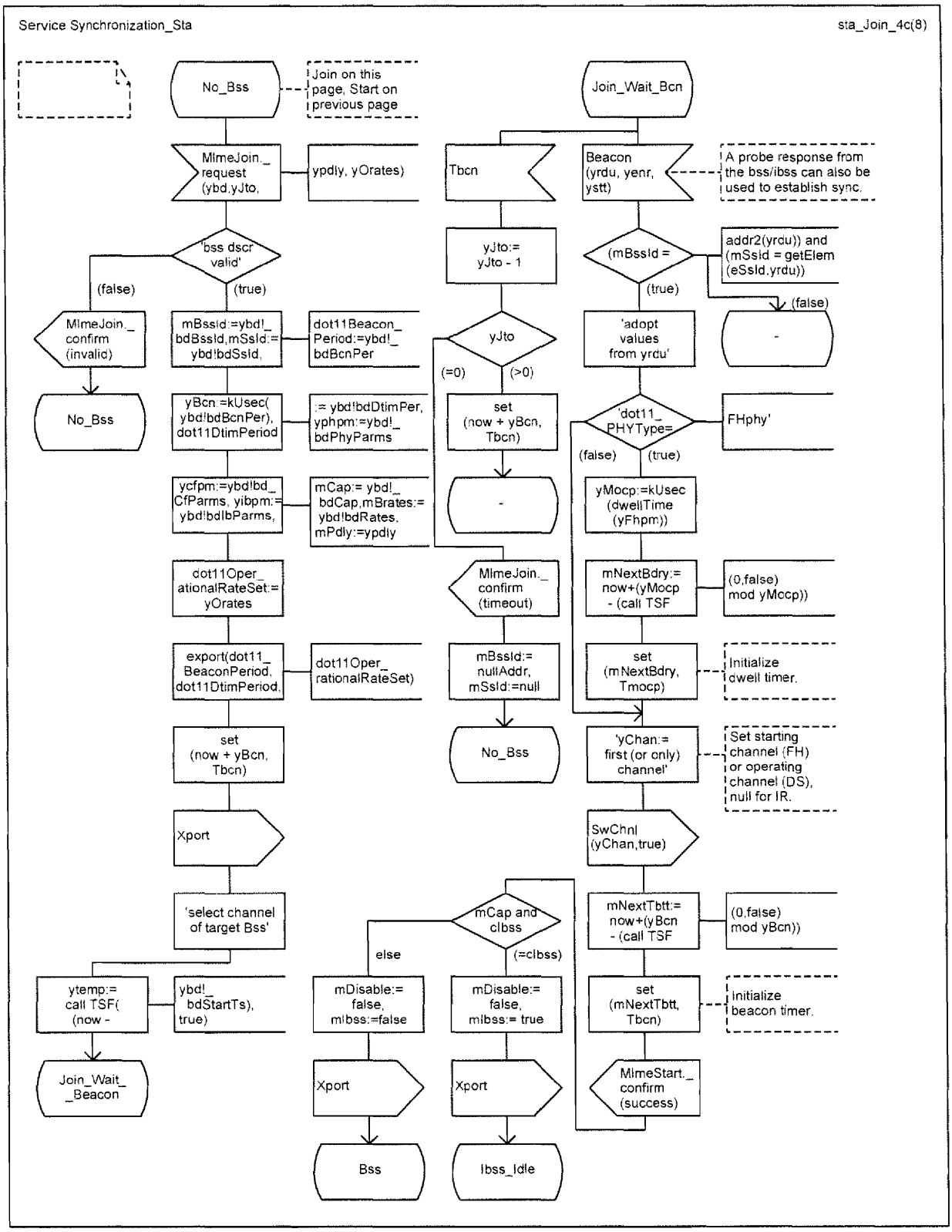


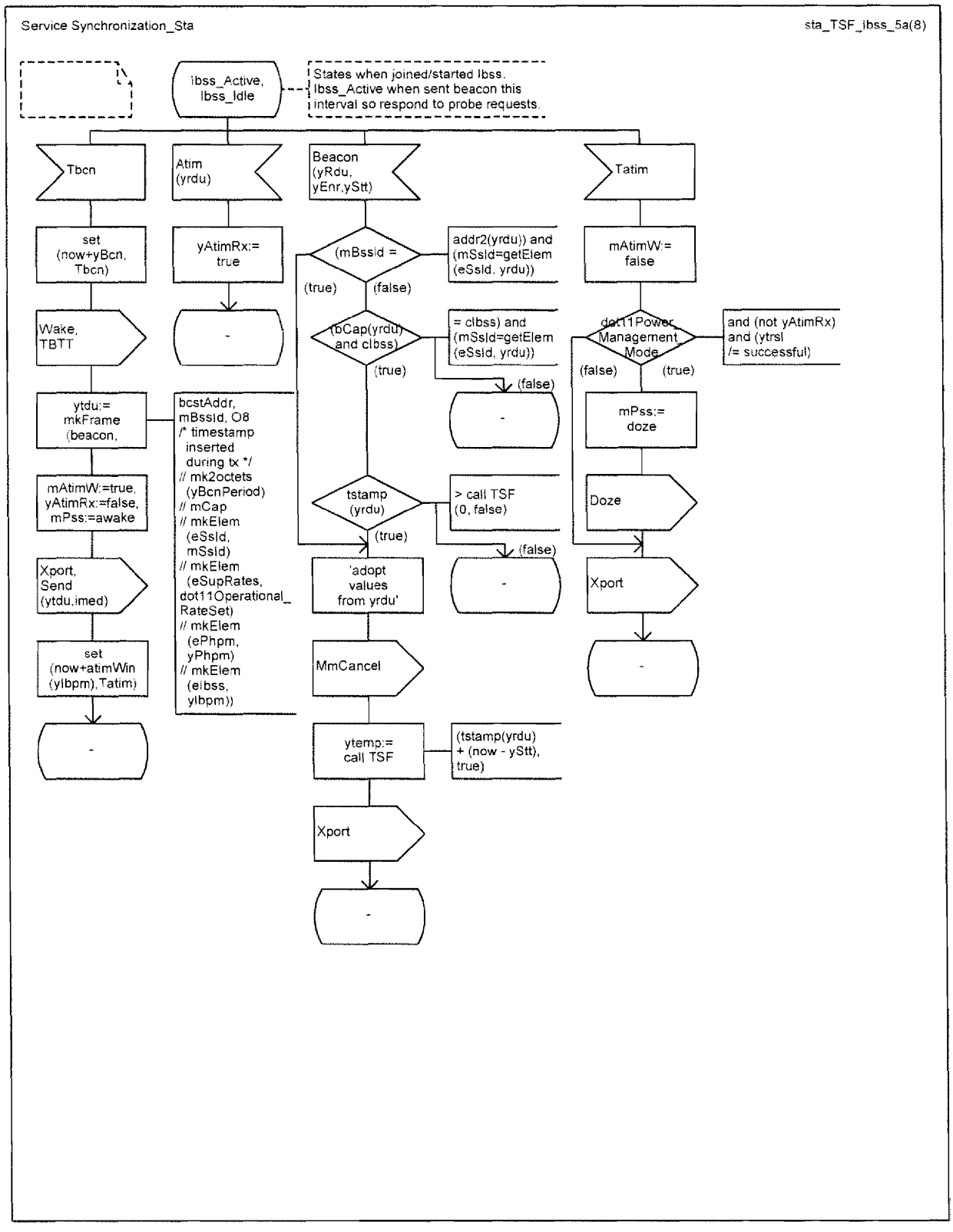


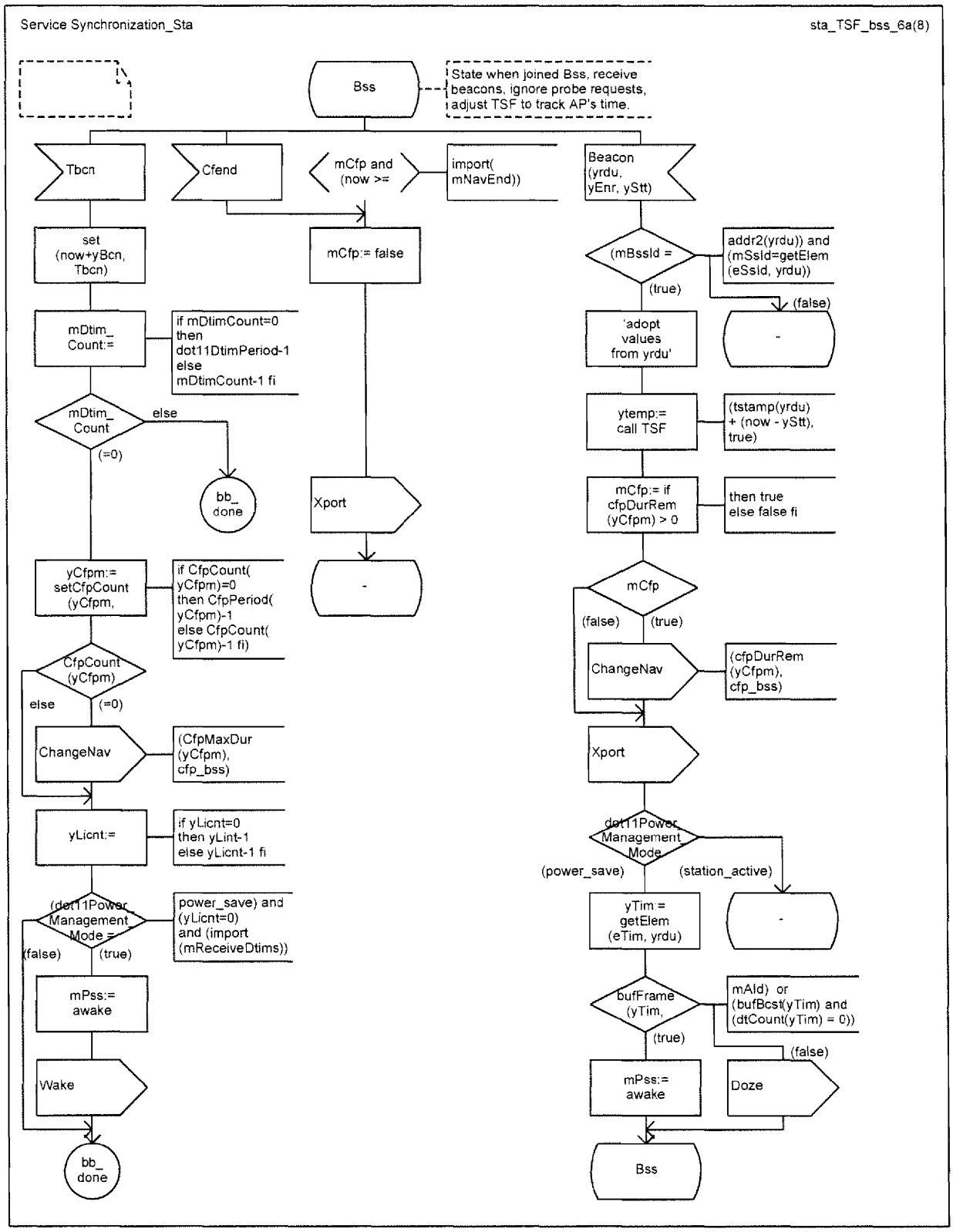


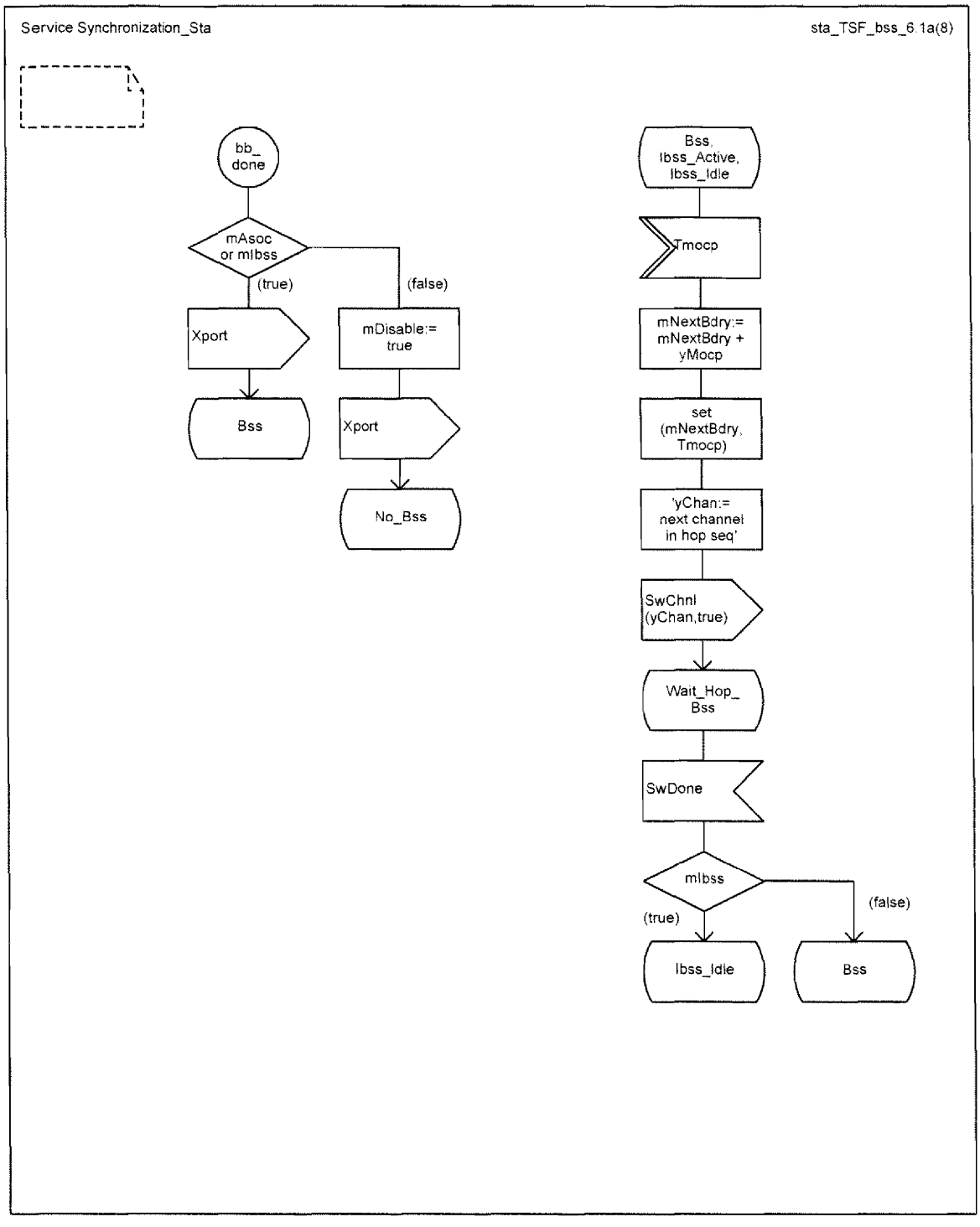


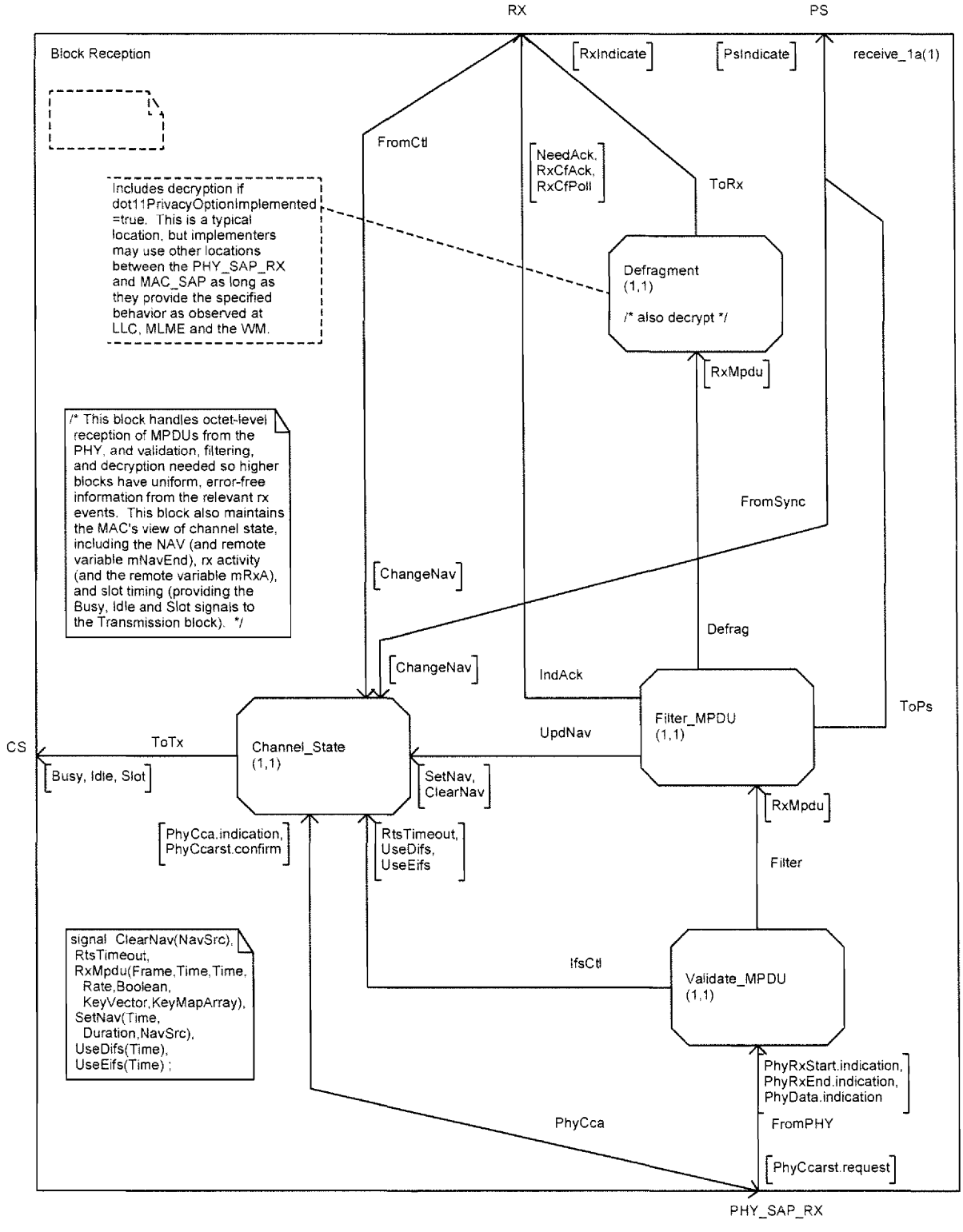


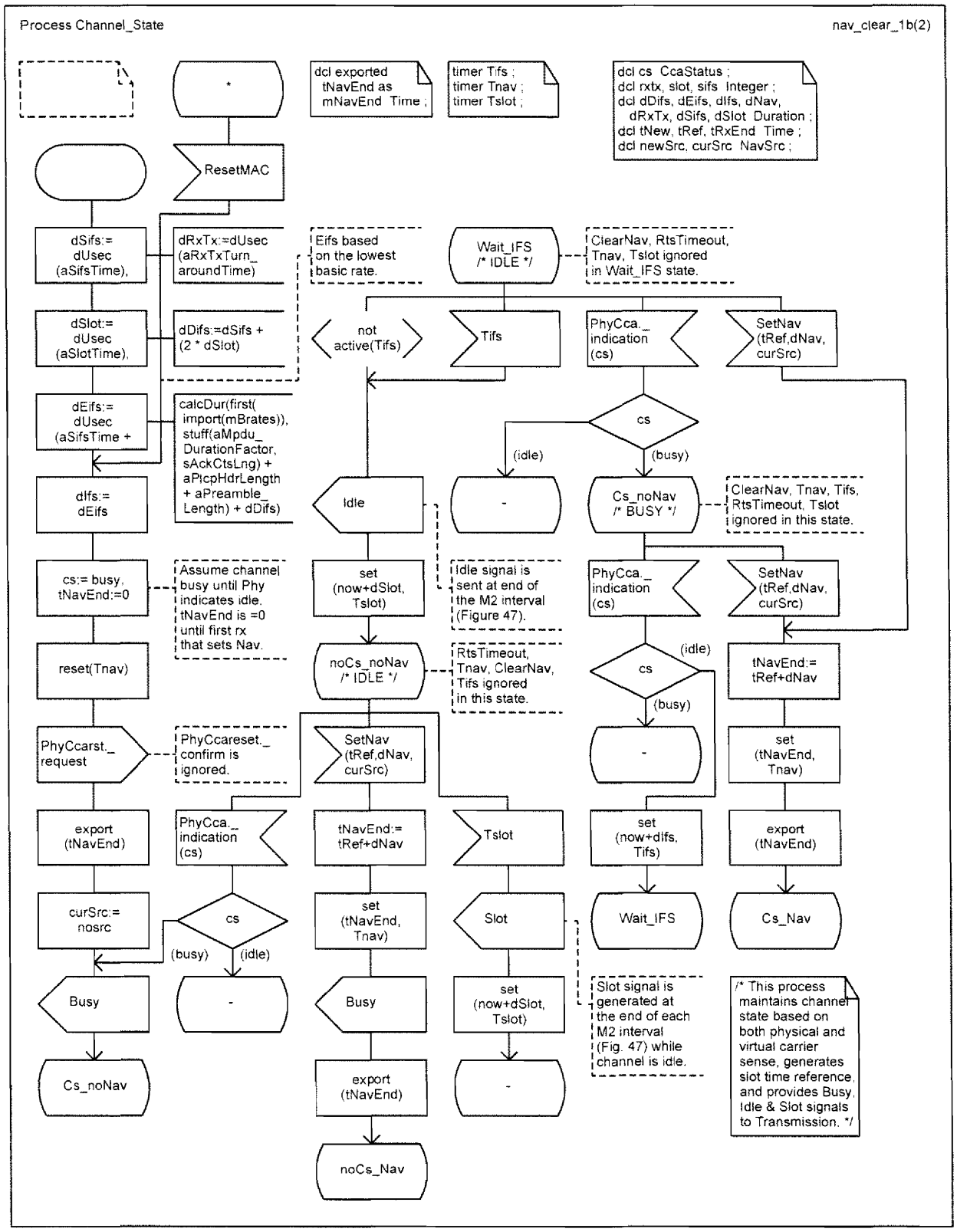


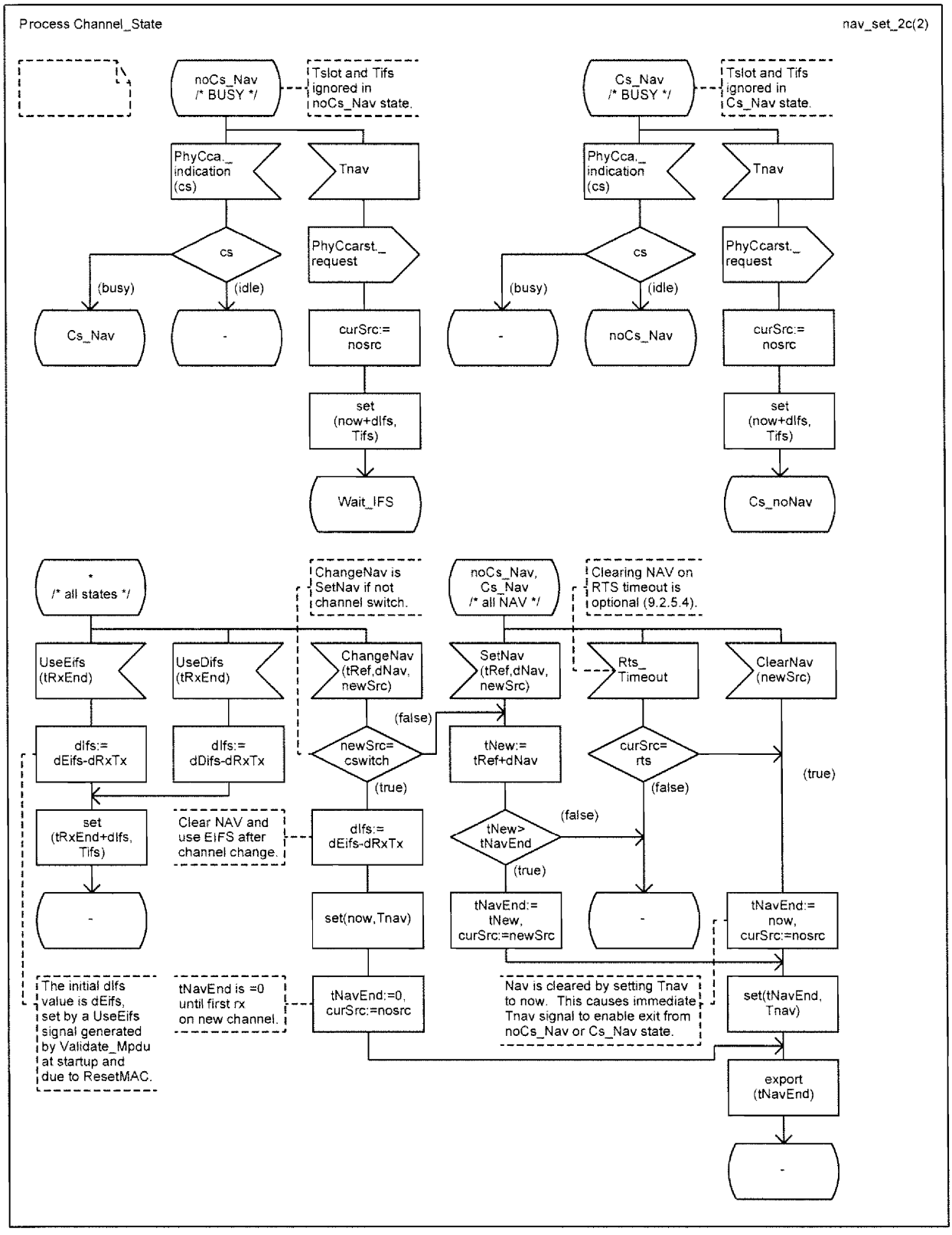


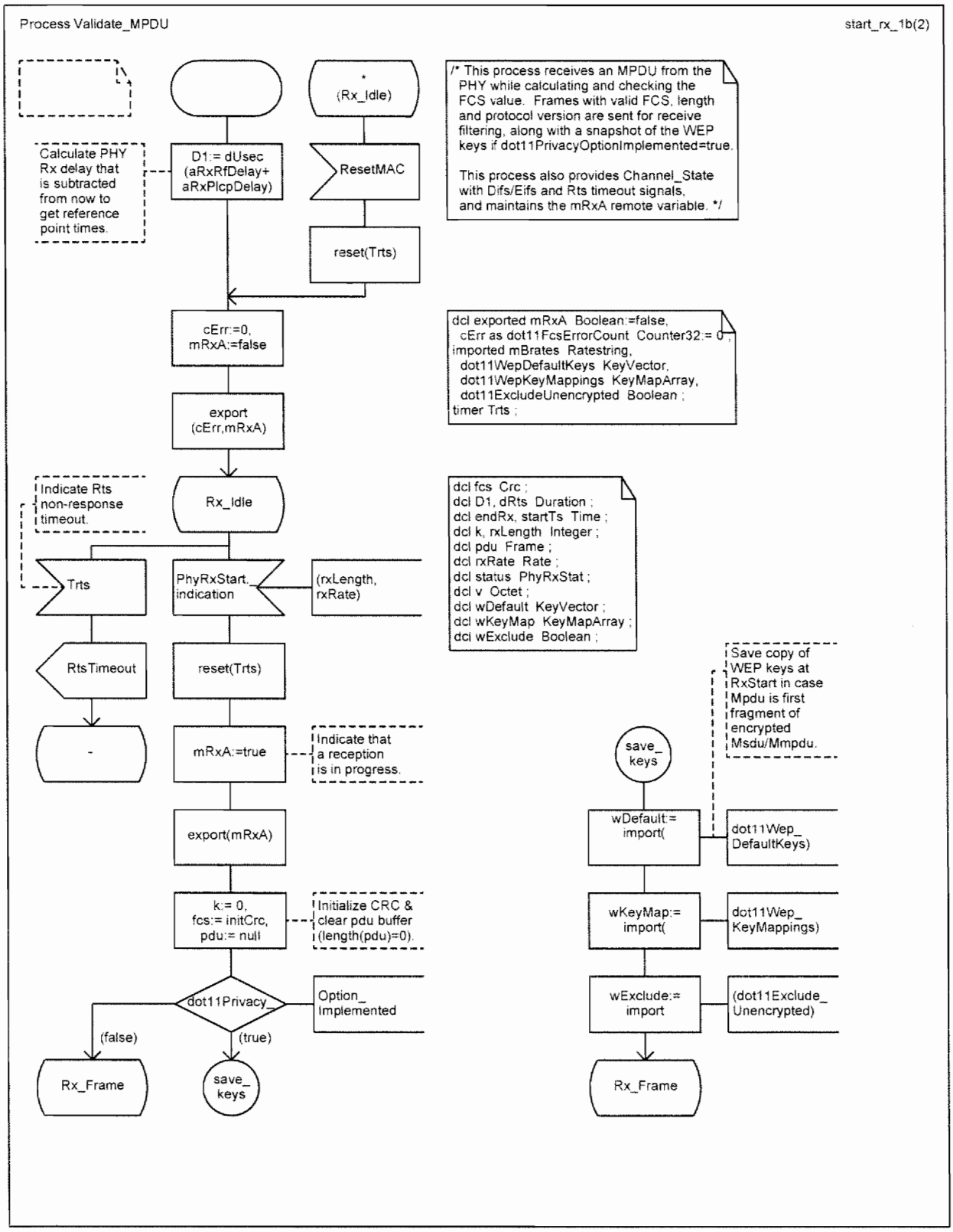


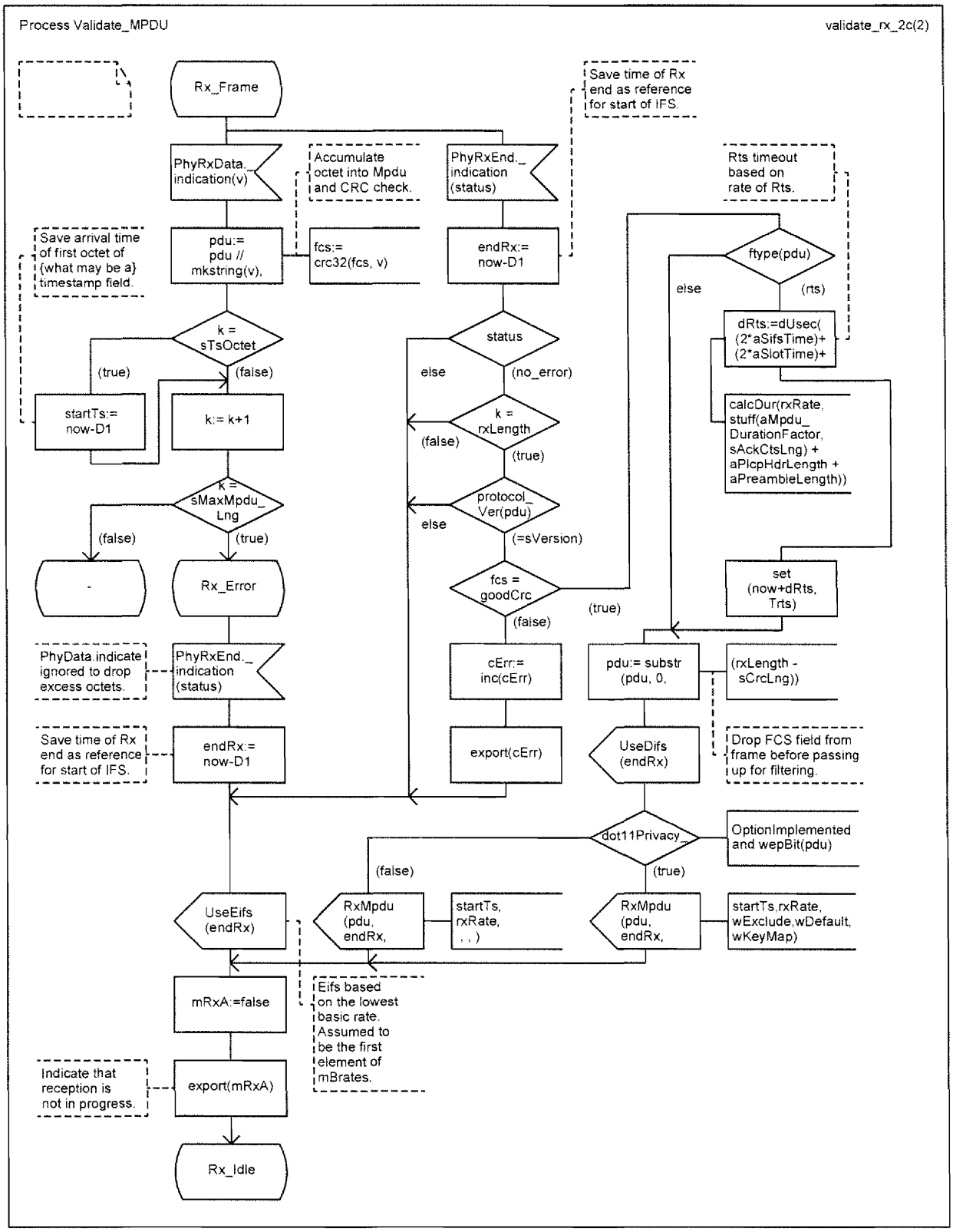


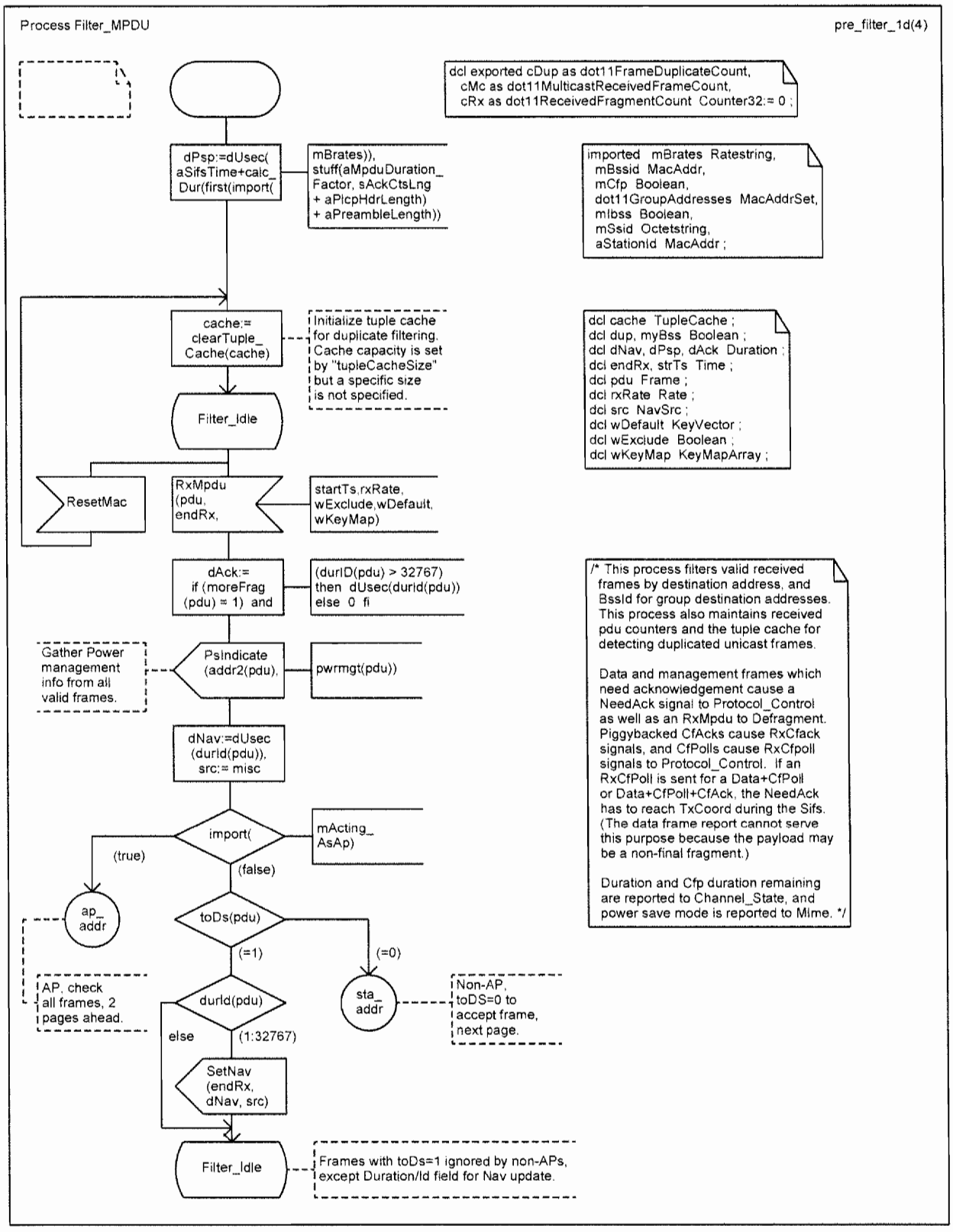


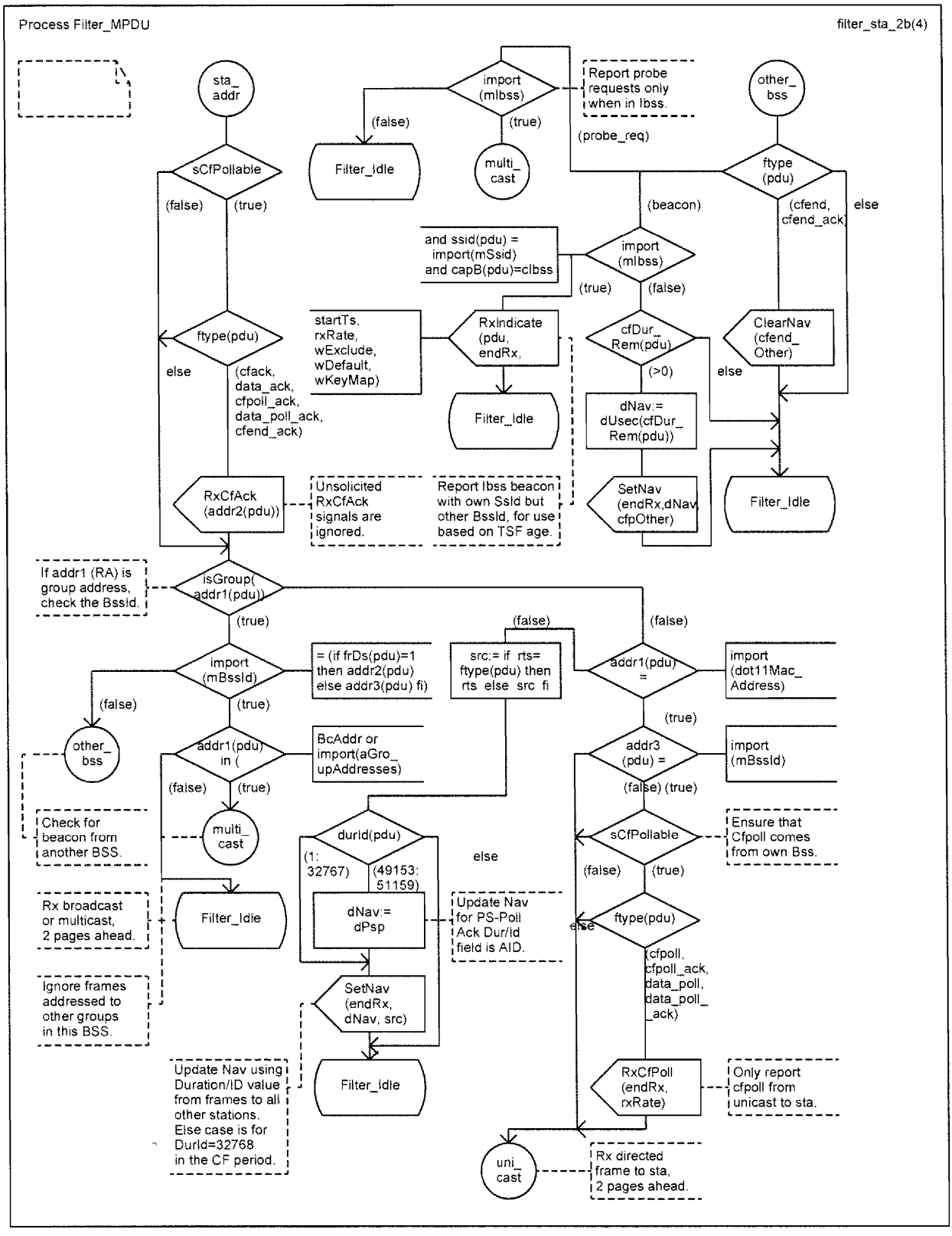


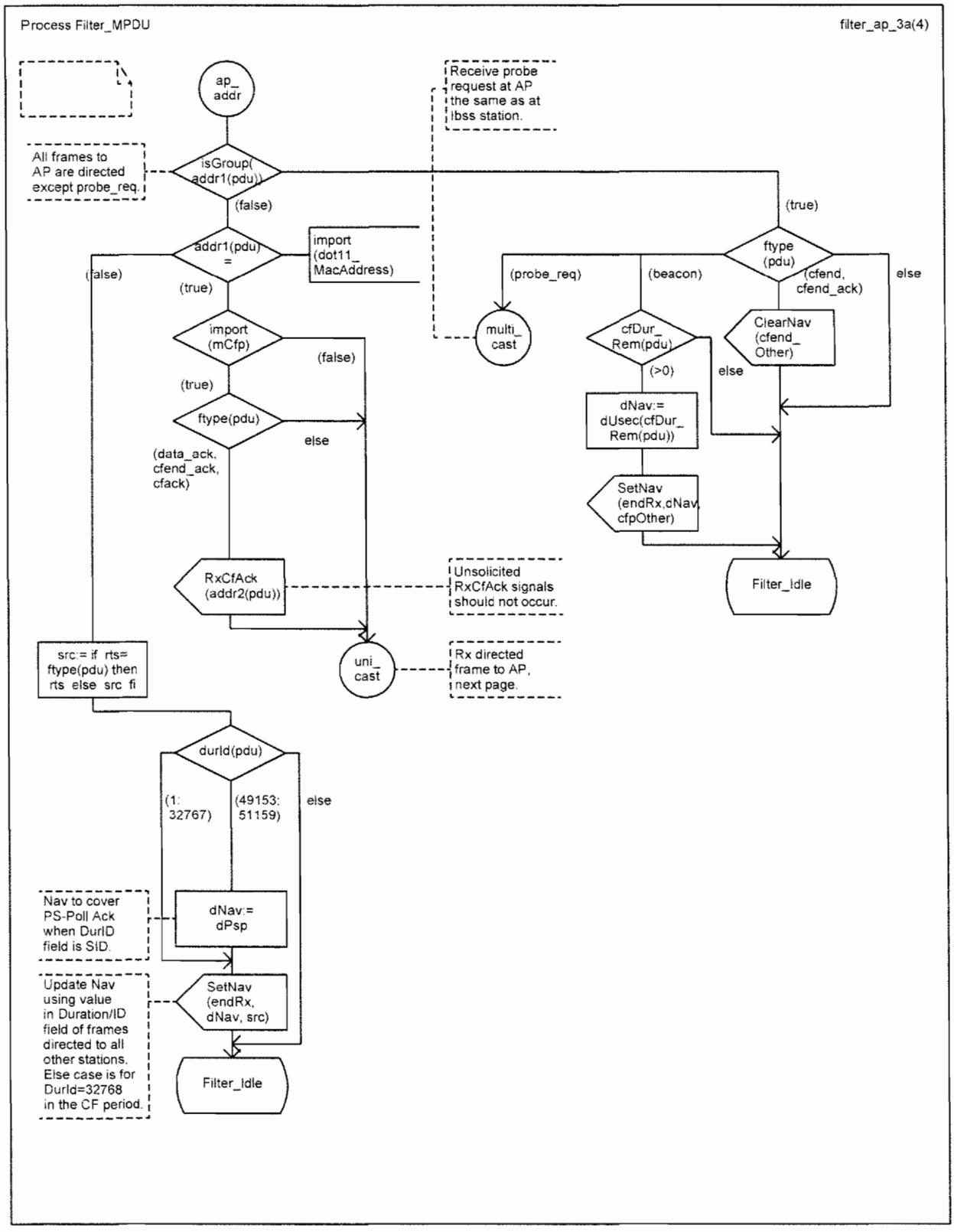


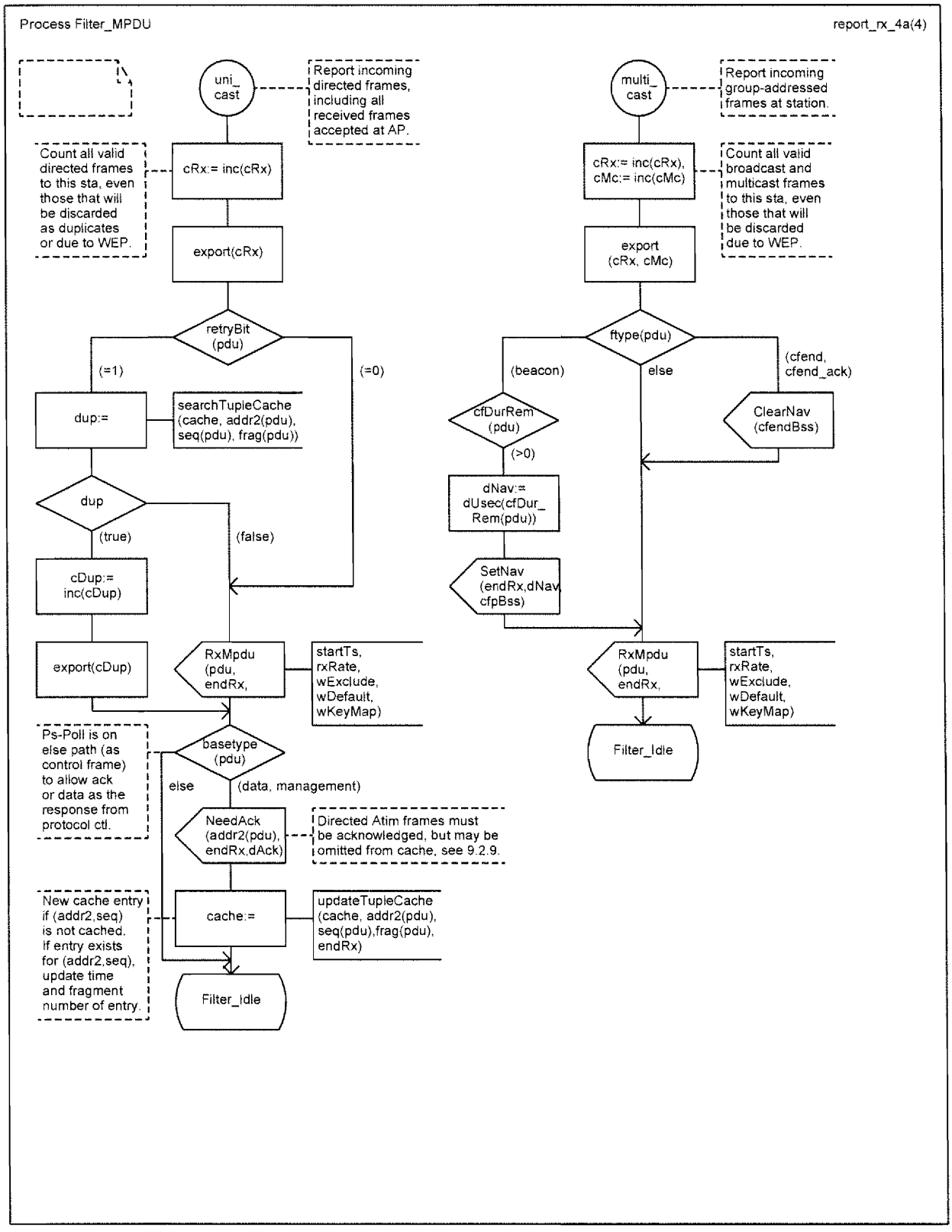


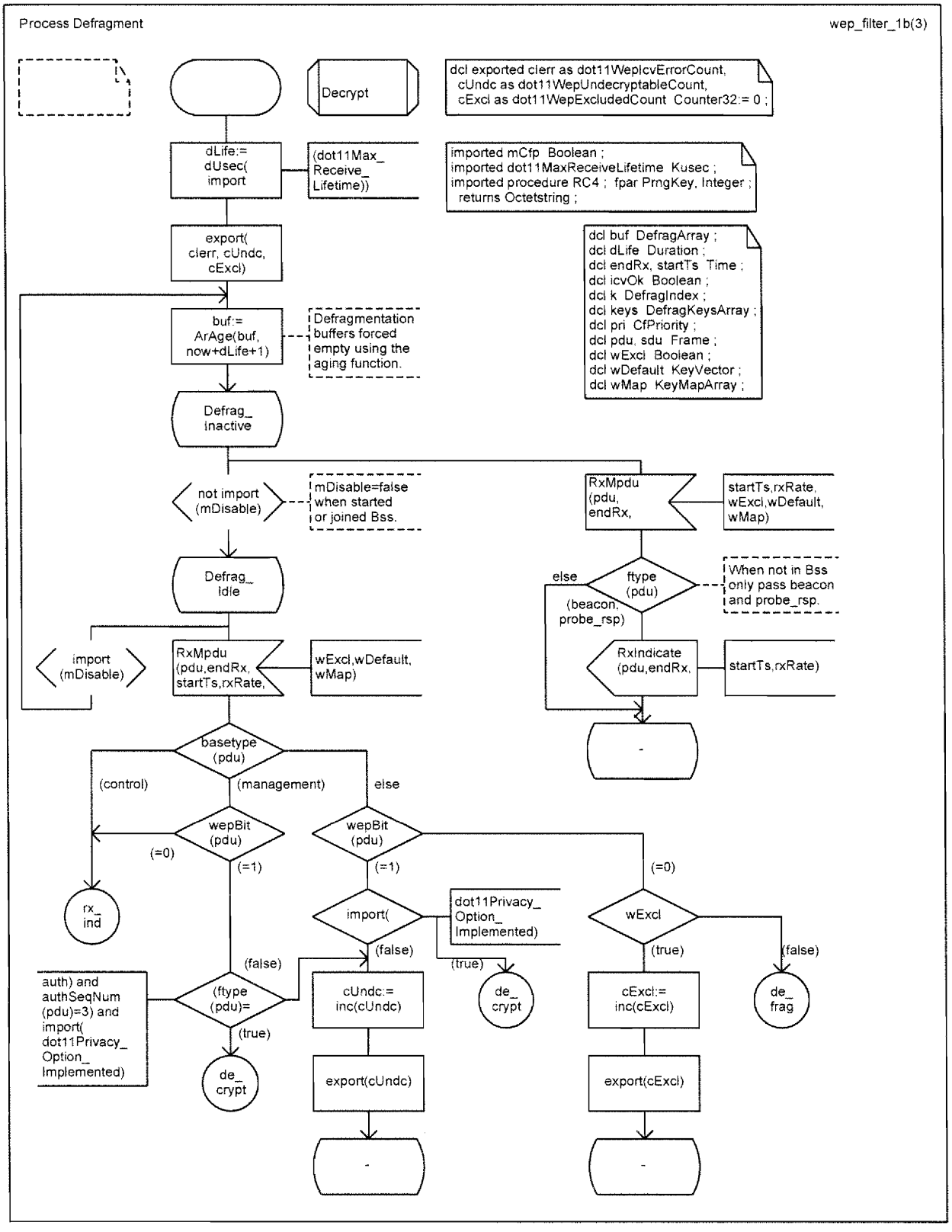


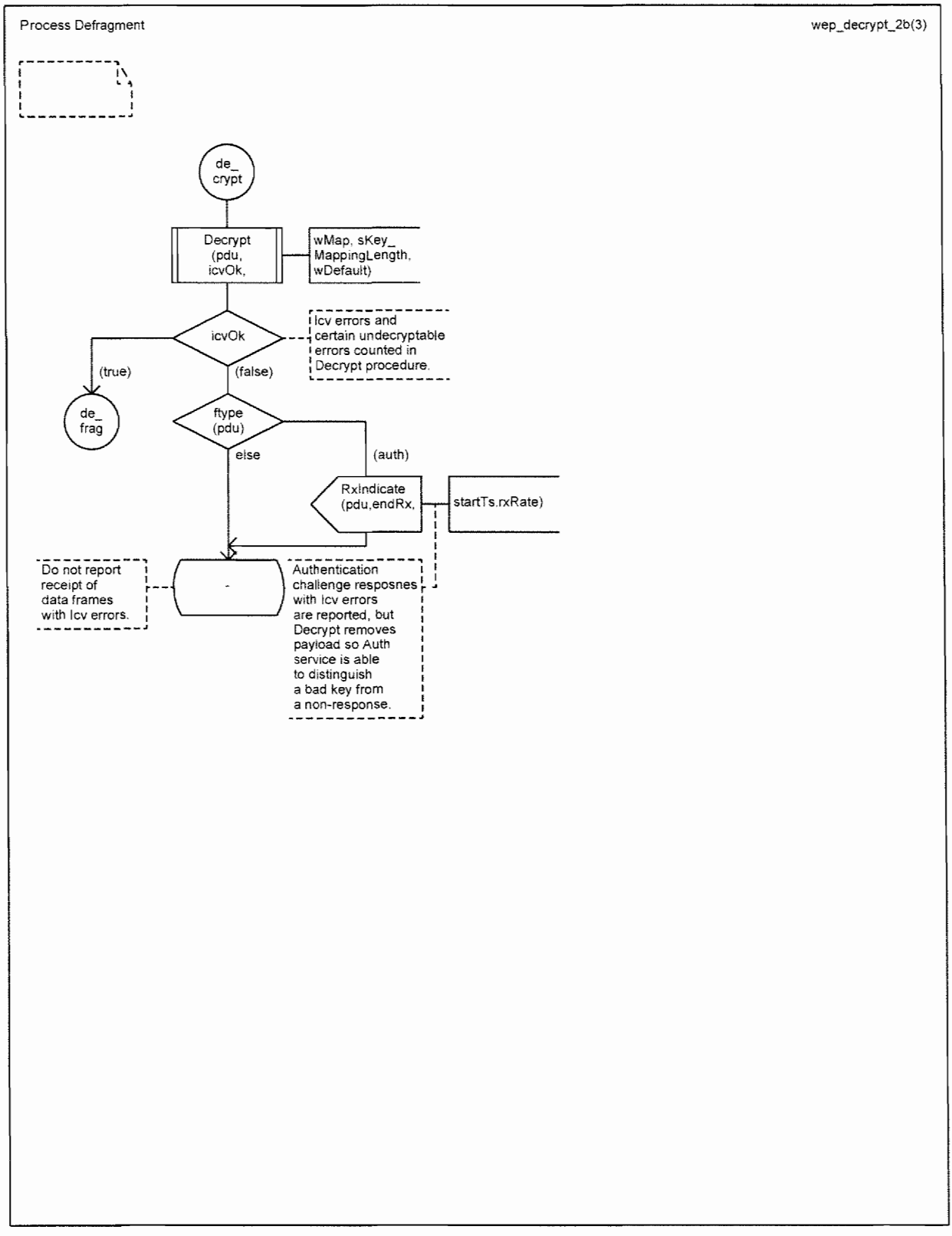


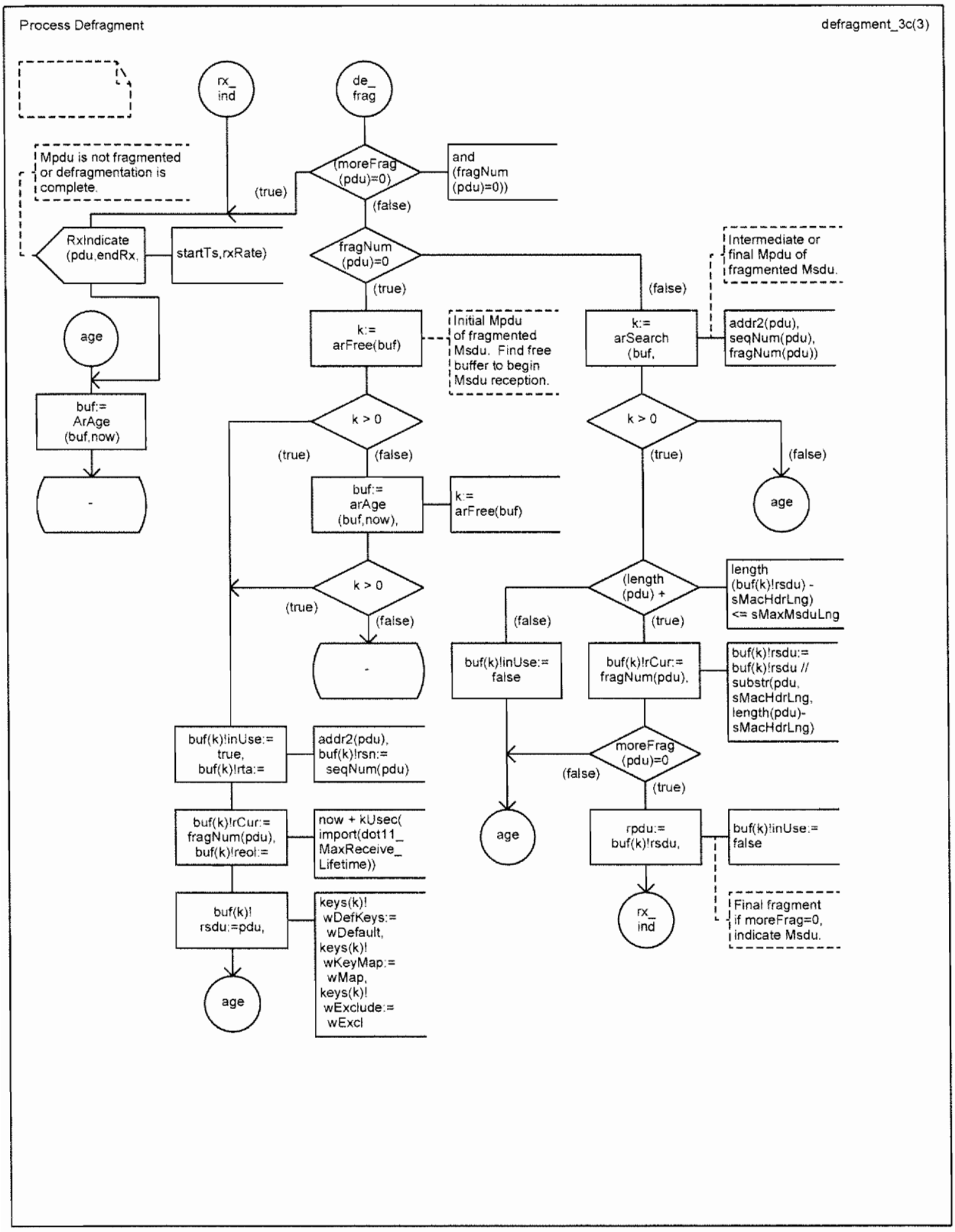


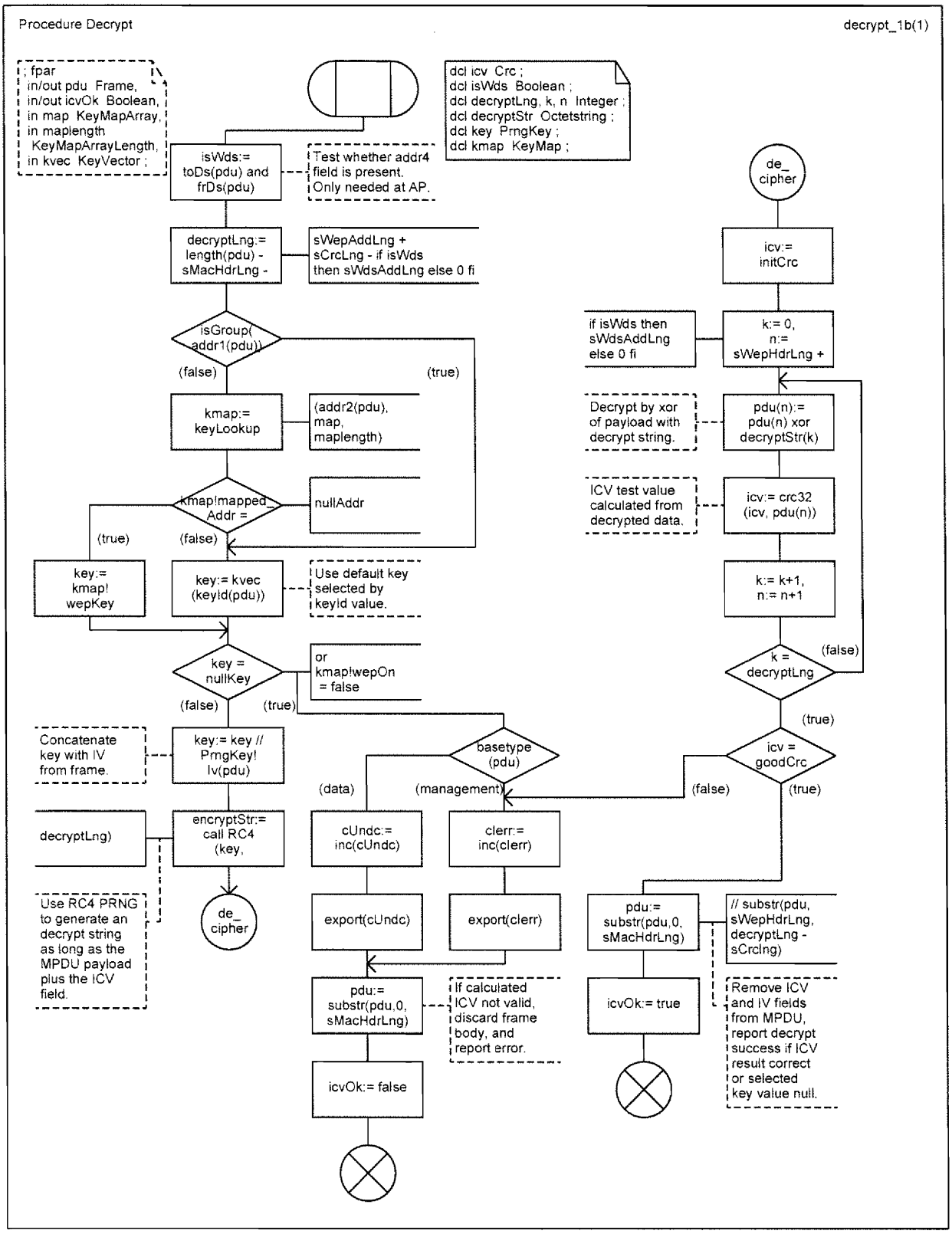








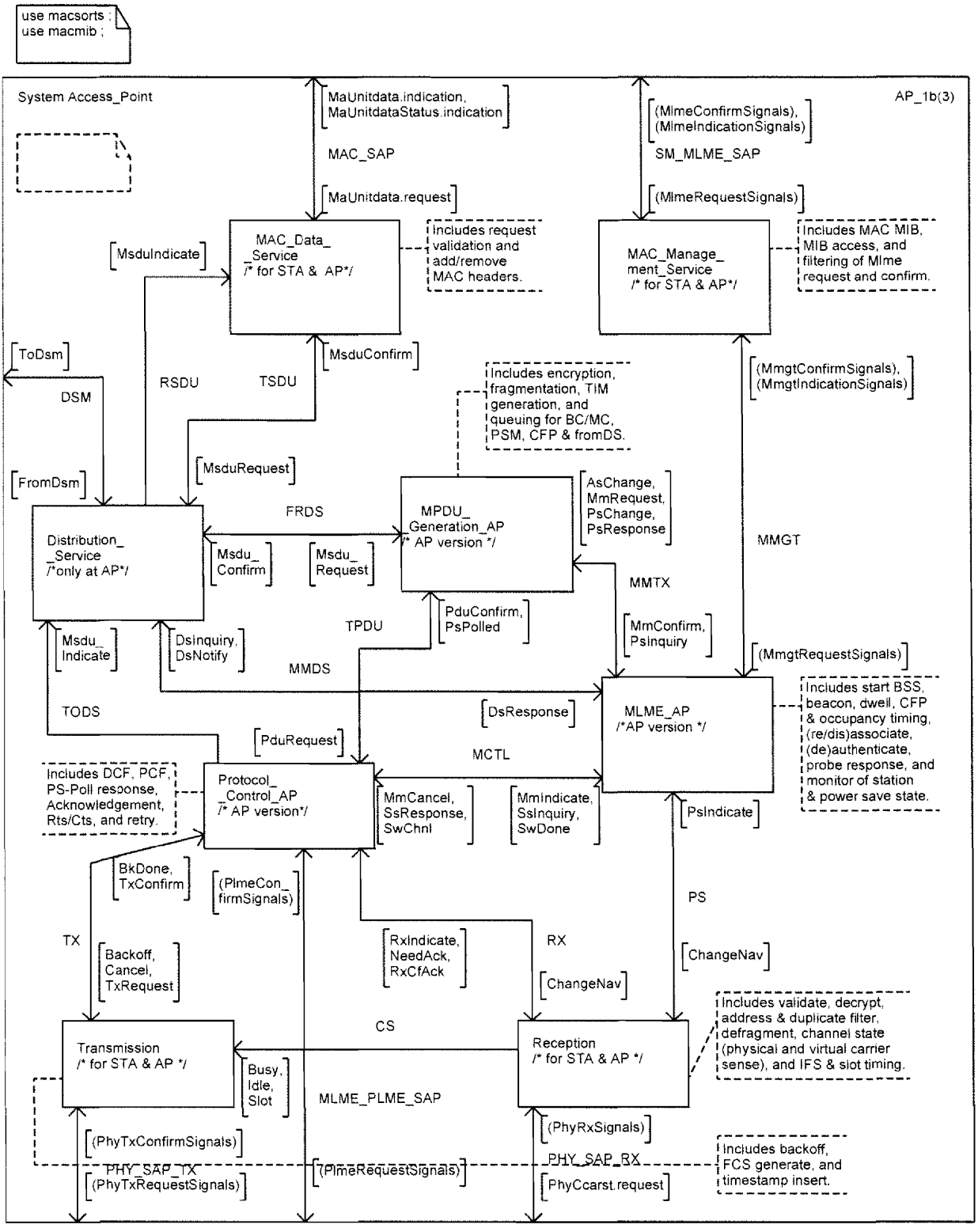


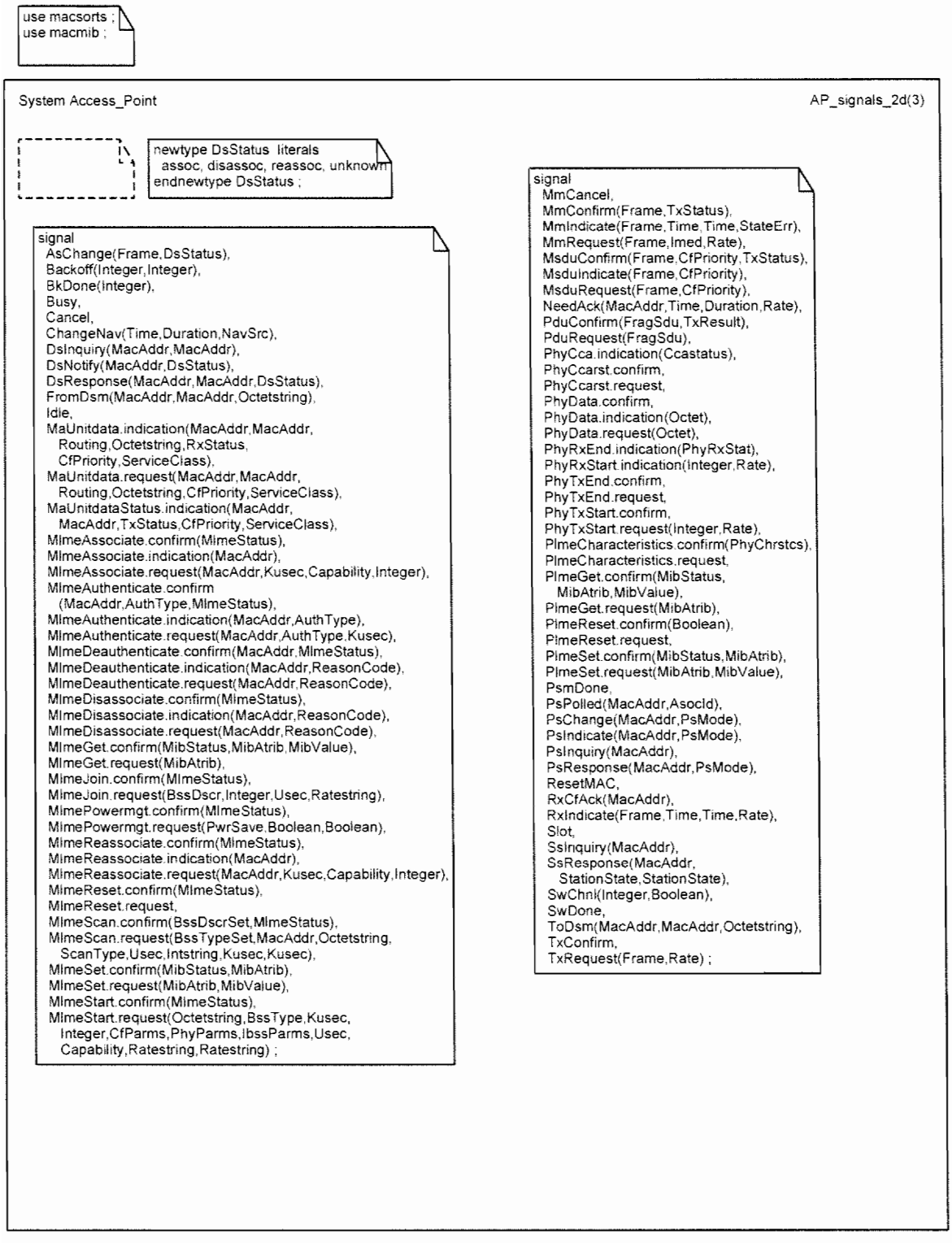


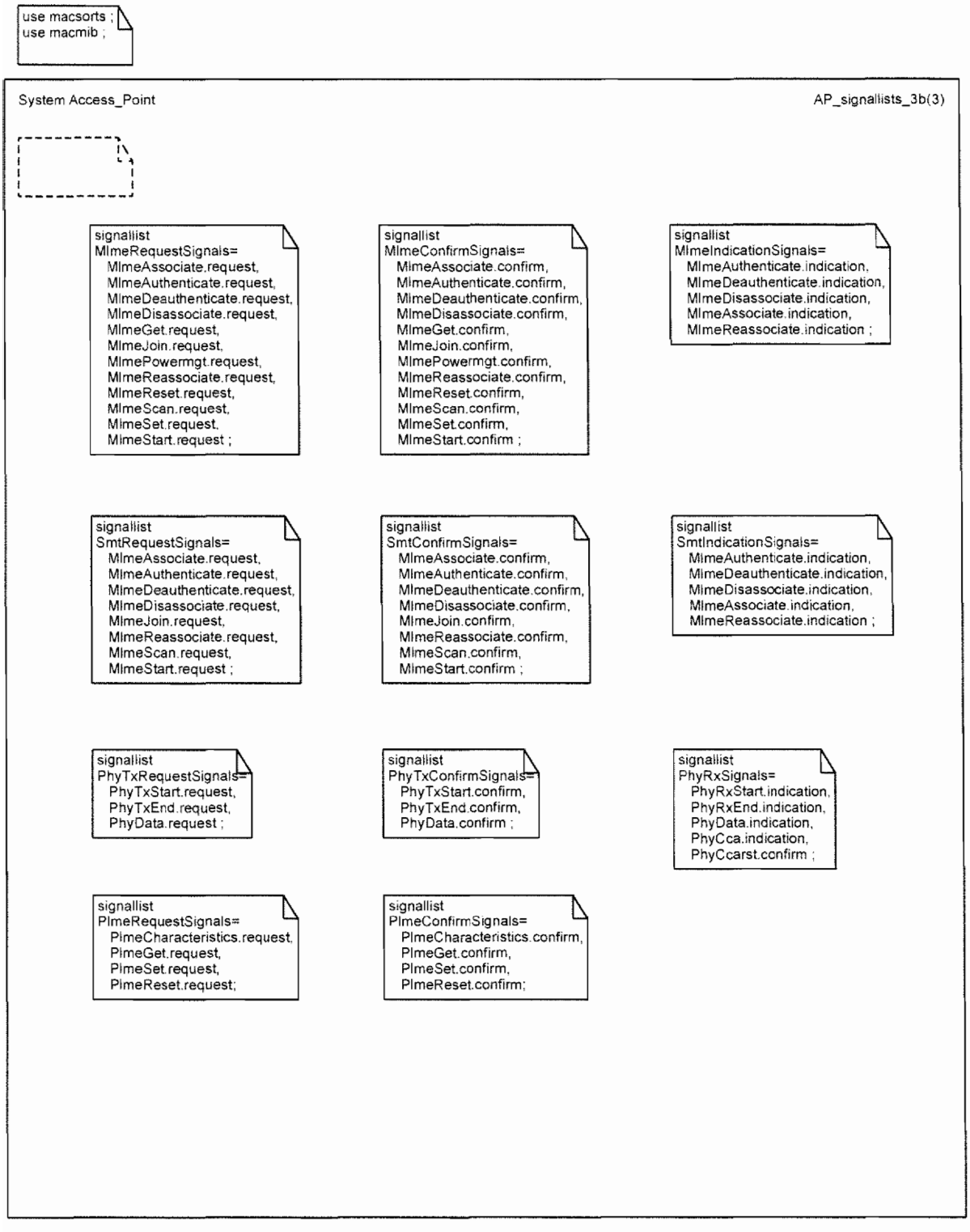
C.4 State machines for MAC access point

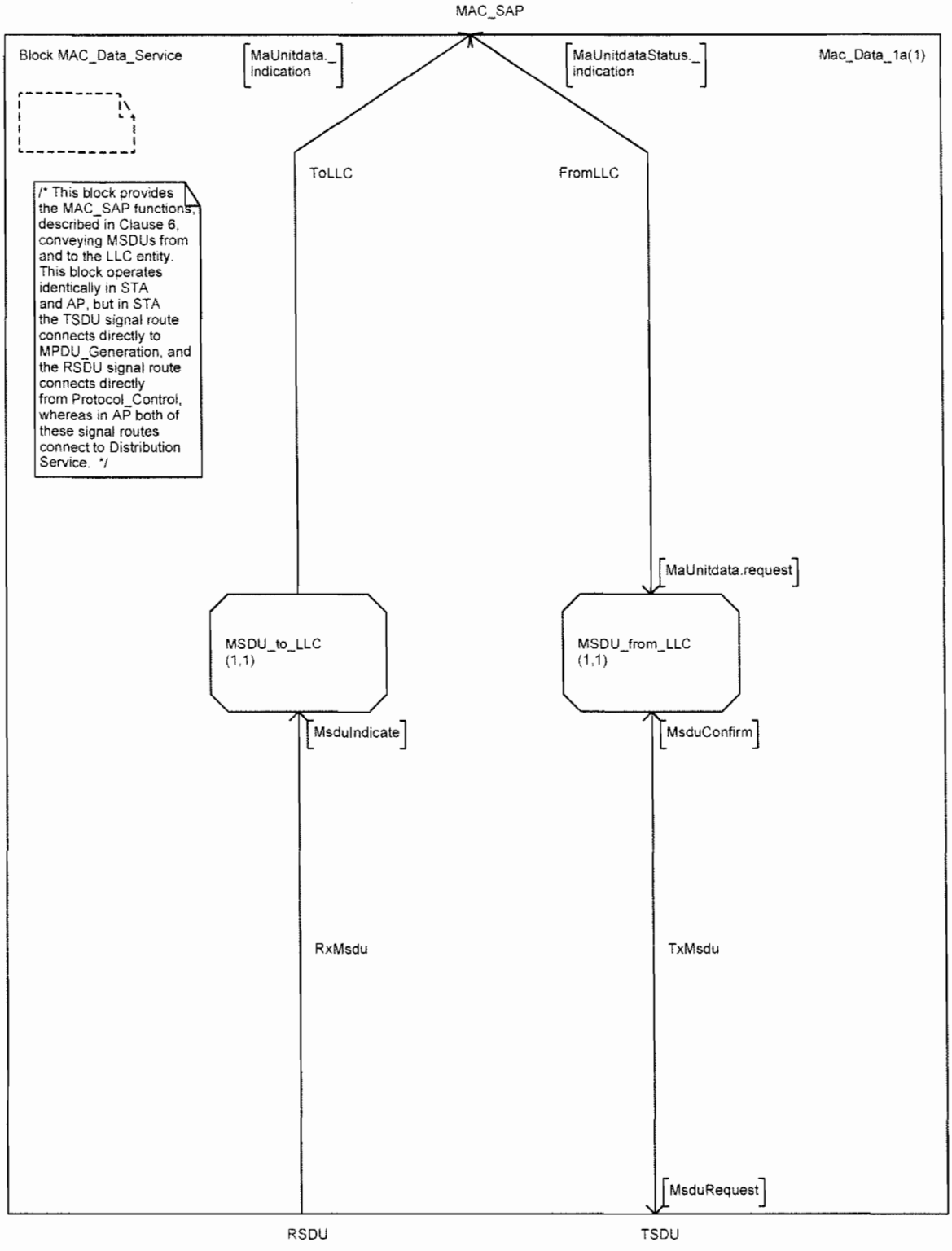
The following SDL-92 system specification defines operation of the MAC protocol at an IEEE 802.11 AP. Many aspects of AP operation are identical to the STA operation. These are defined in blocks and processes referenced from both the STA and AP system specifications. Blocks and processes used in both STA and AP are identifiable by the SDL comment /* for STA & AP */ below the block or process name. Blocks and processes specific to AP operation are identifiable by the SDL comment /* AP version */ below the block or process name. Definitions for the /* AP version */ and the /* STA & AP */ blocks and processes appear in this subclause.

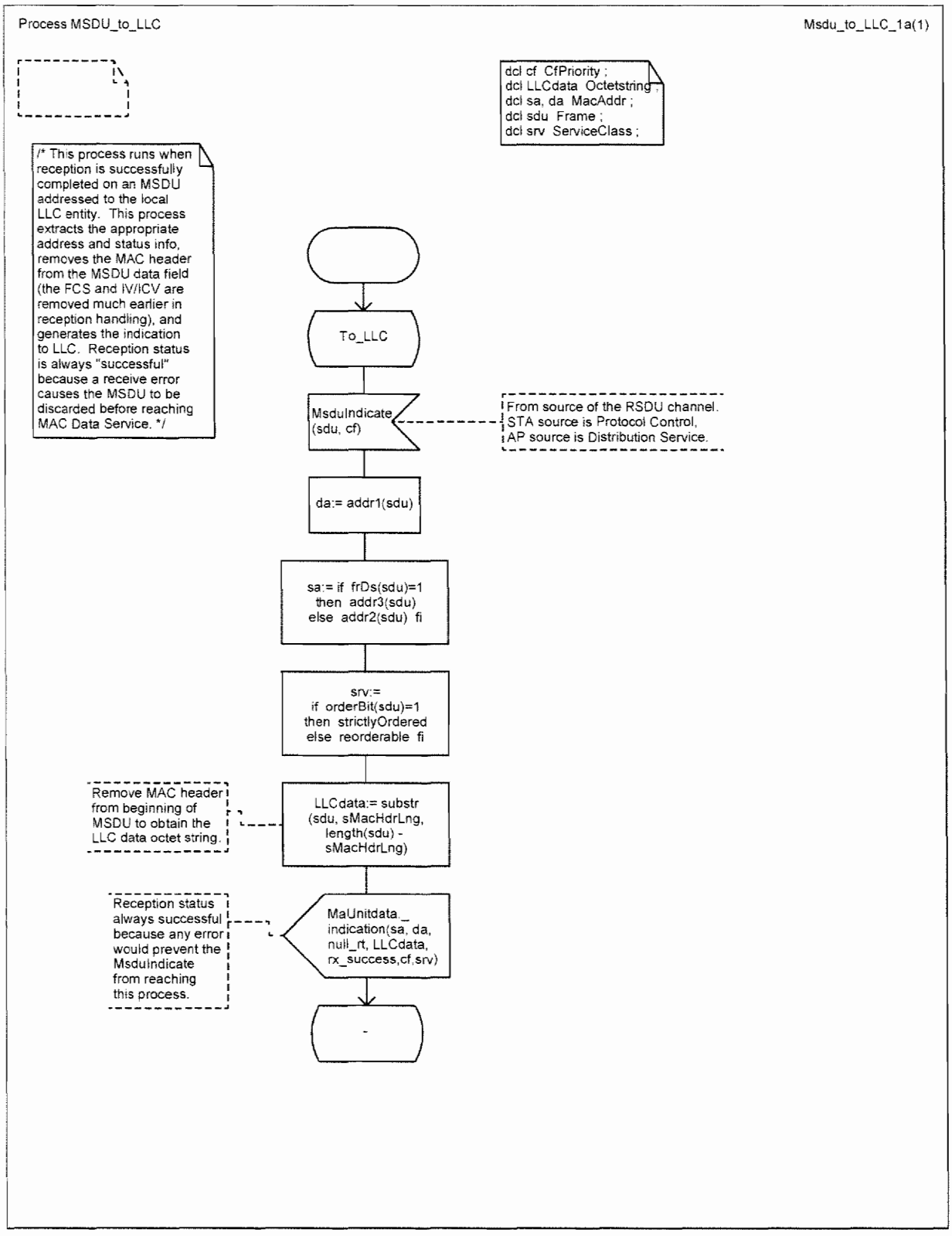
The remainder of this clause is the formal description, in SDL/GR, of an IEEE 802.11 AP.

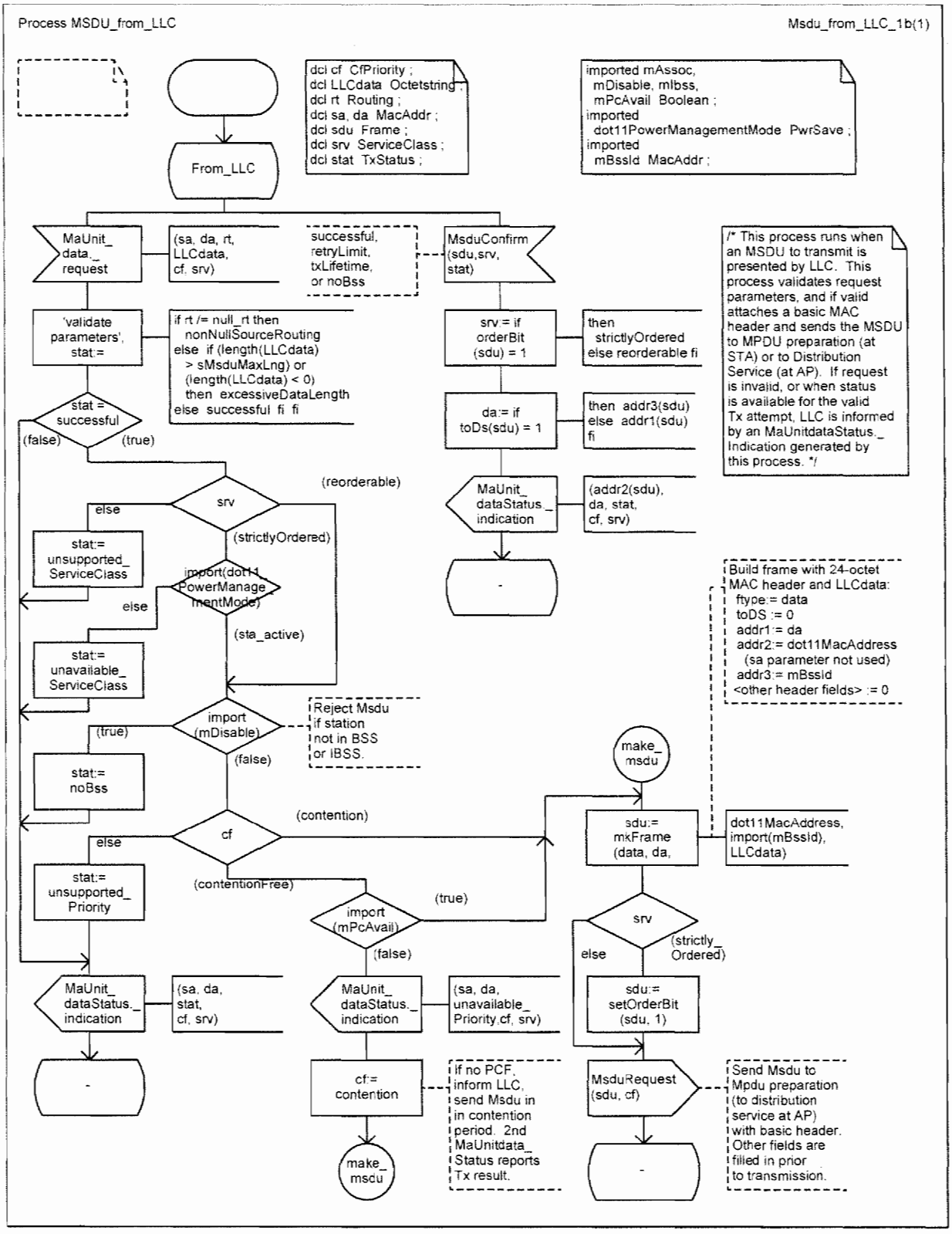


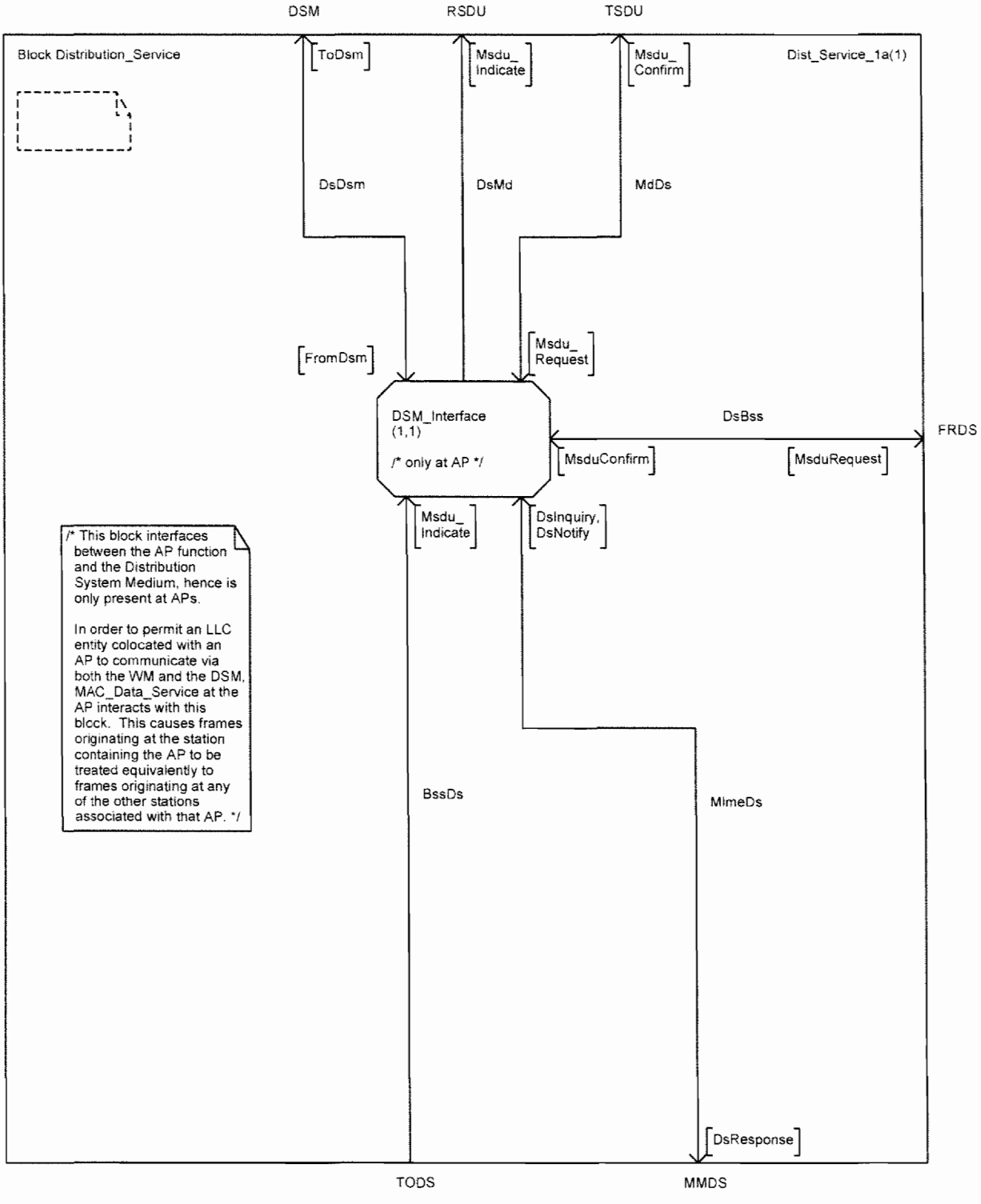


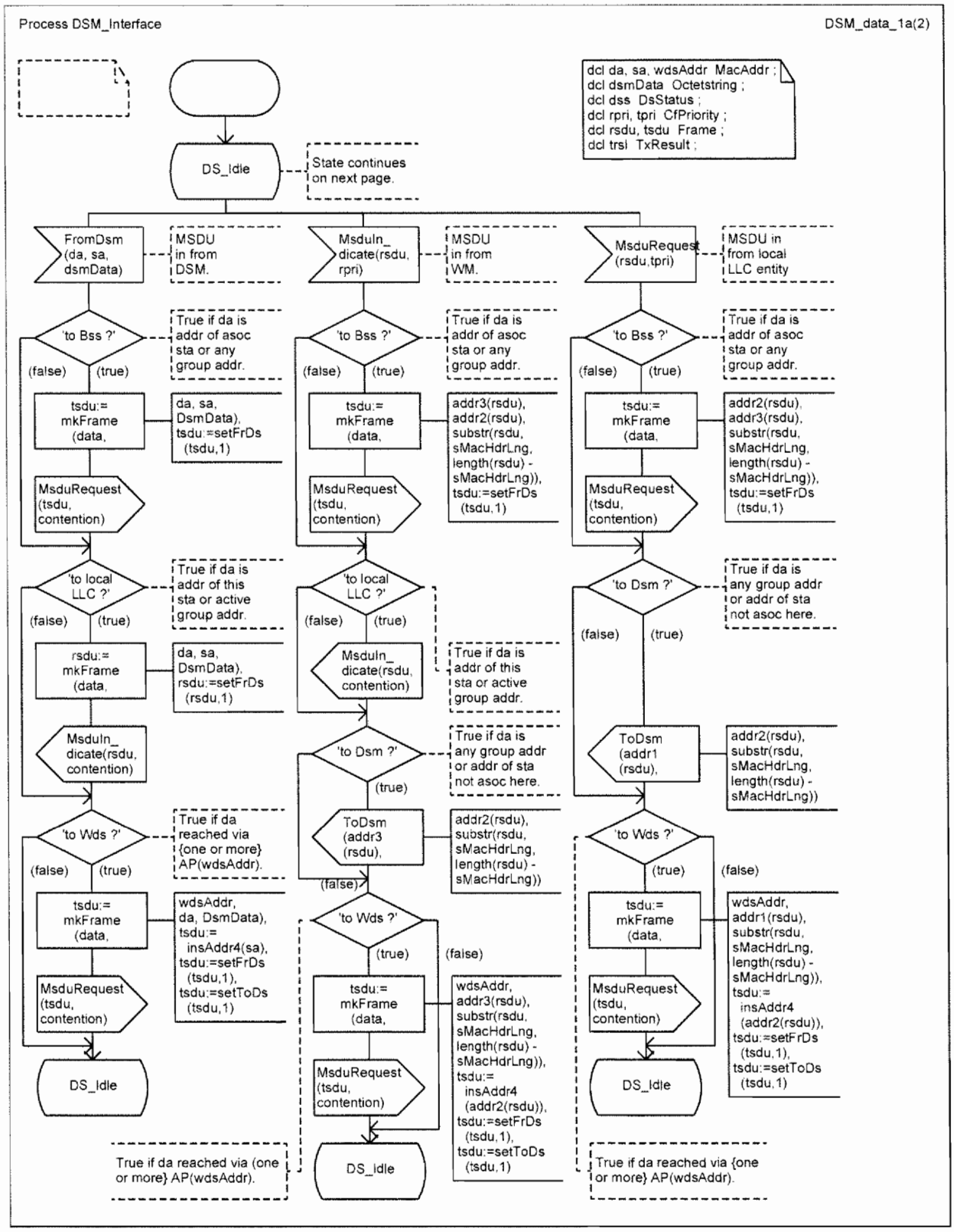


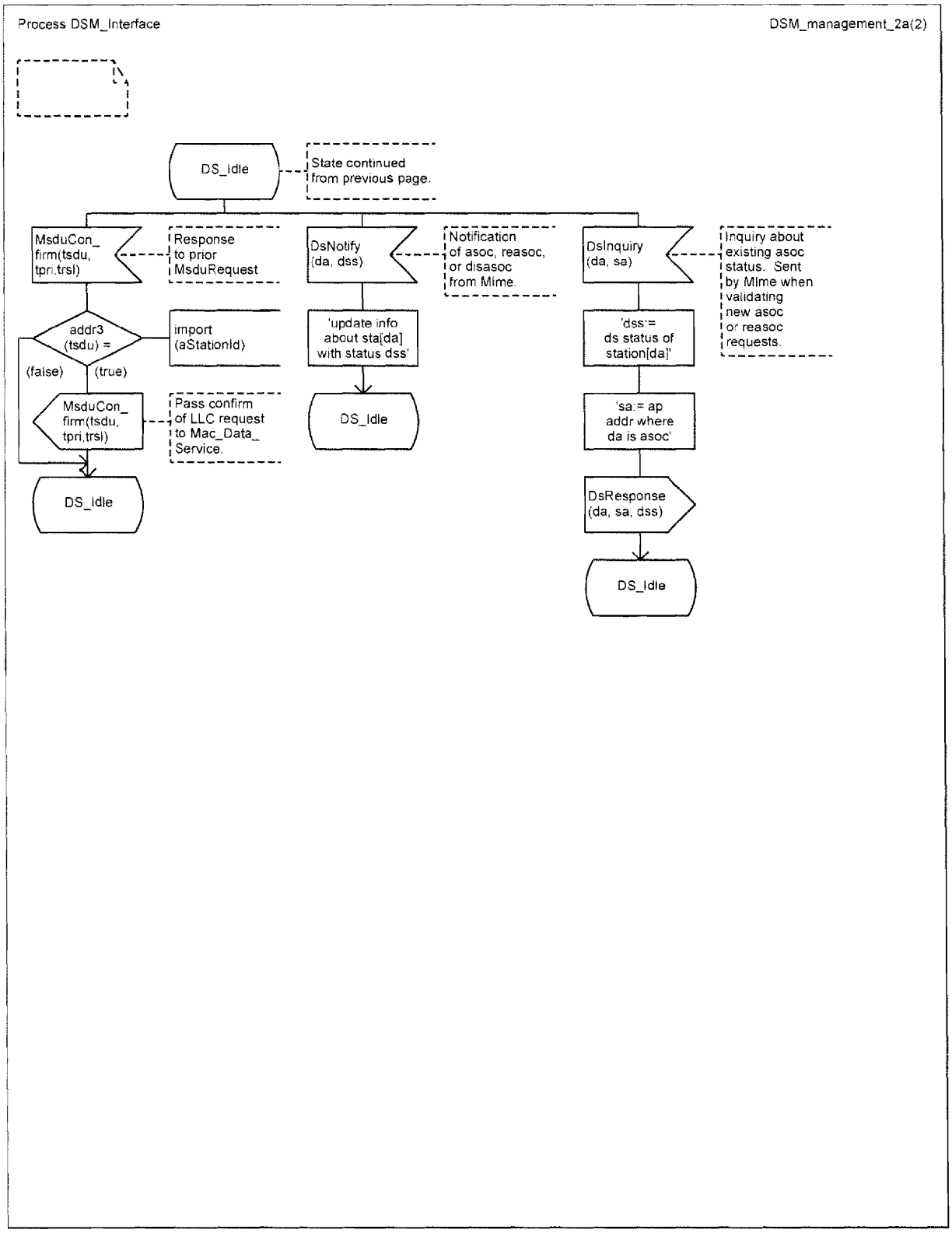


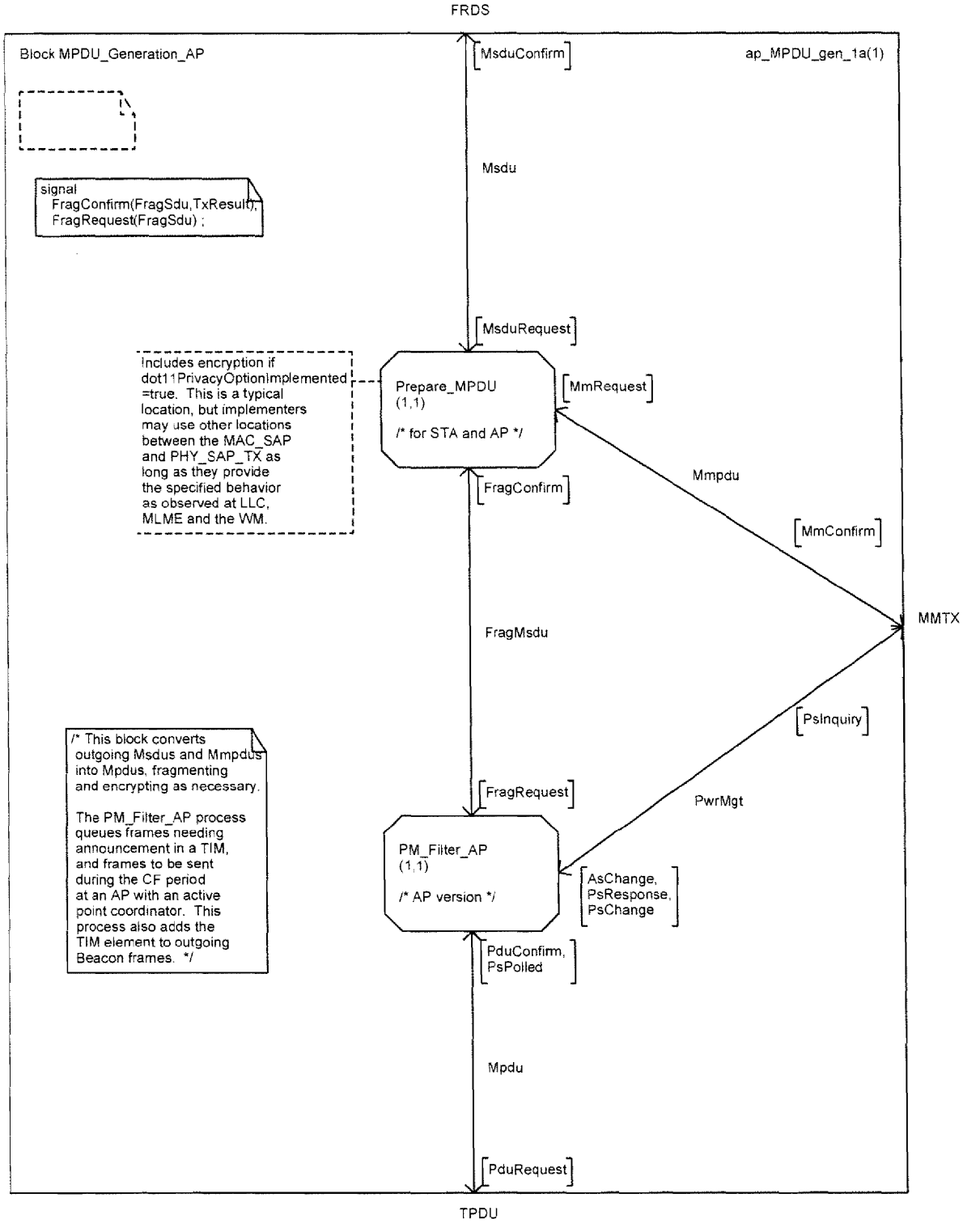


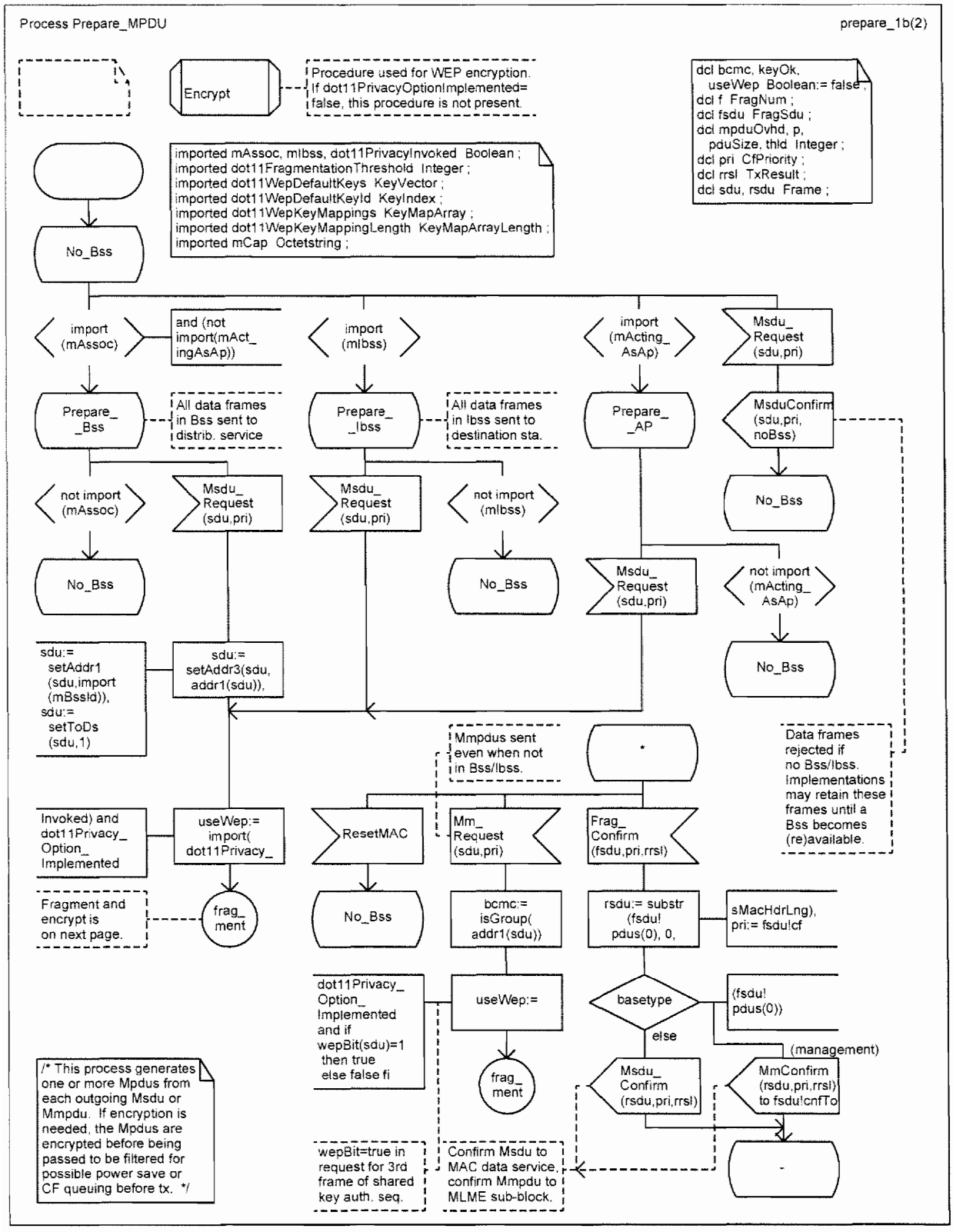


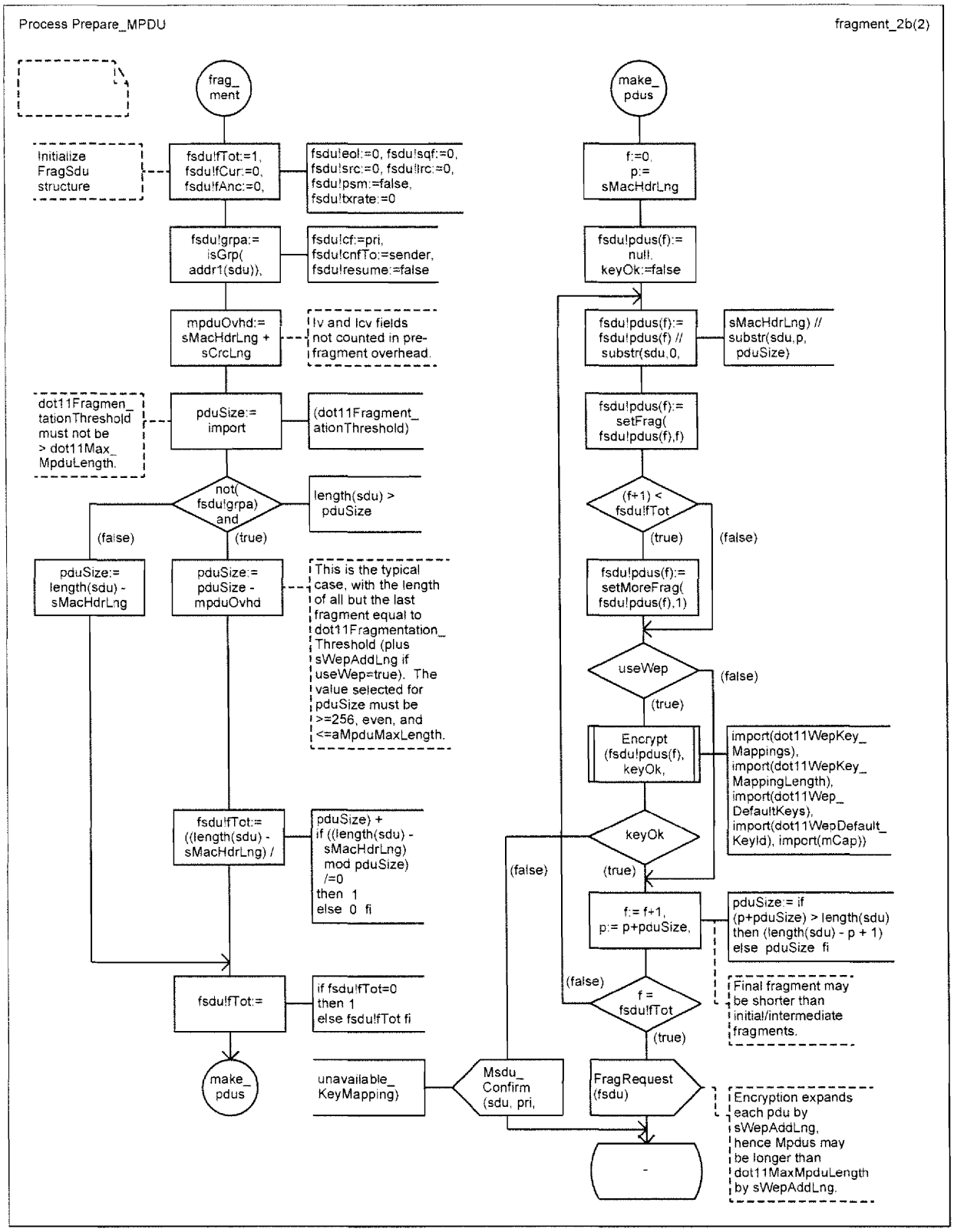


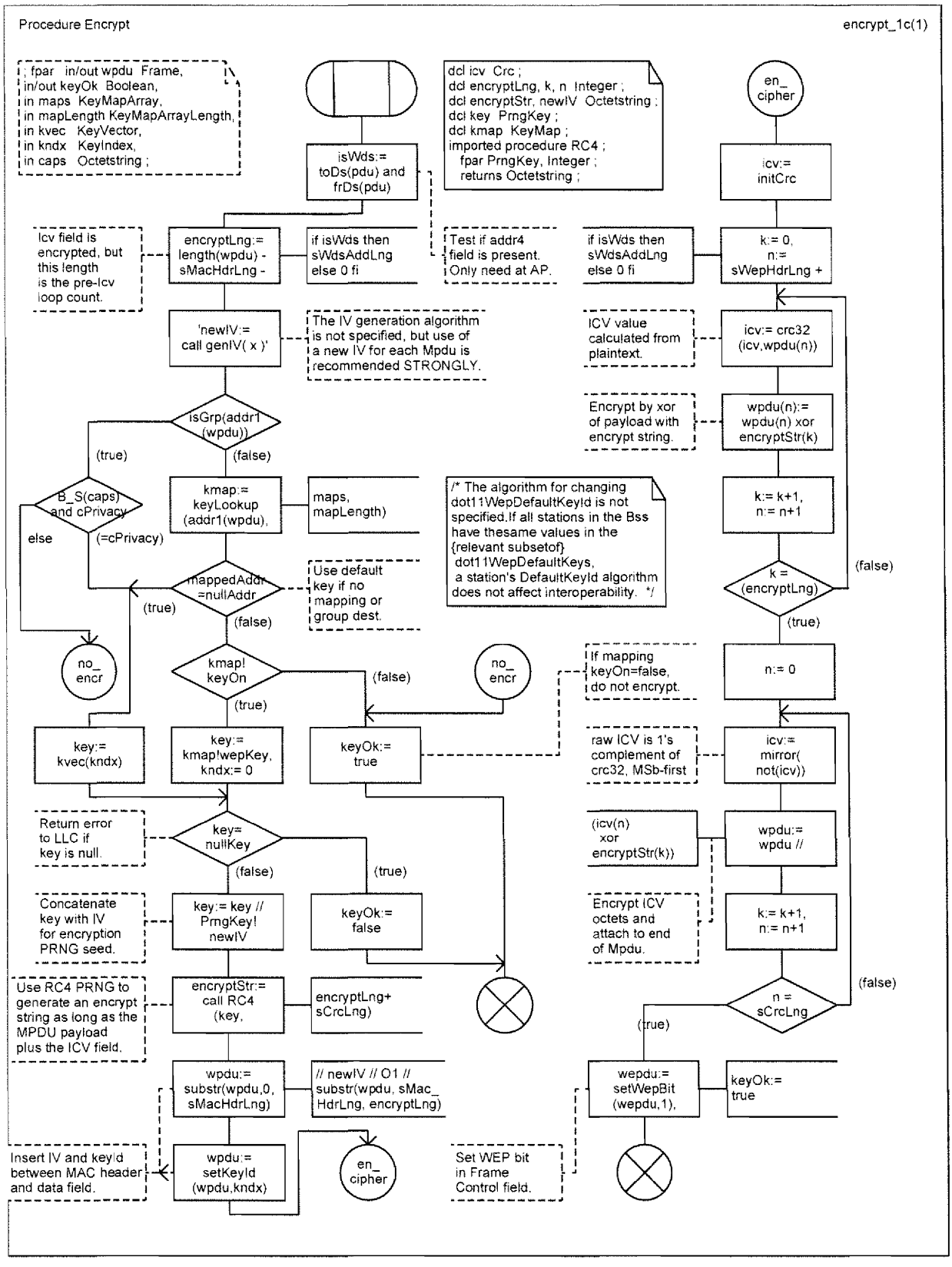


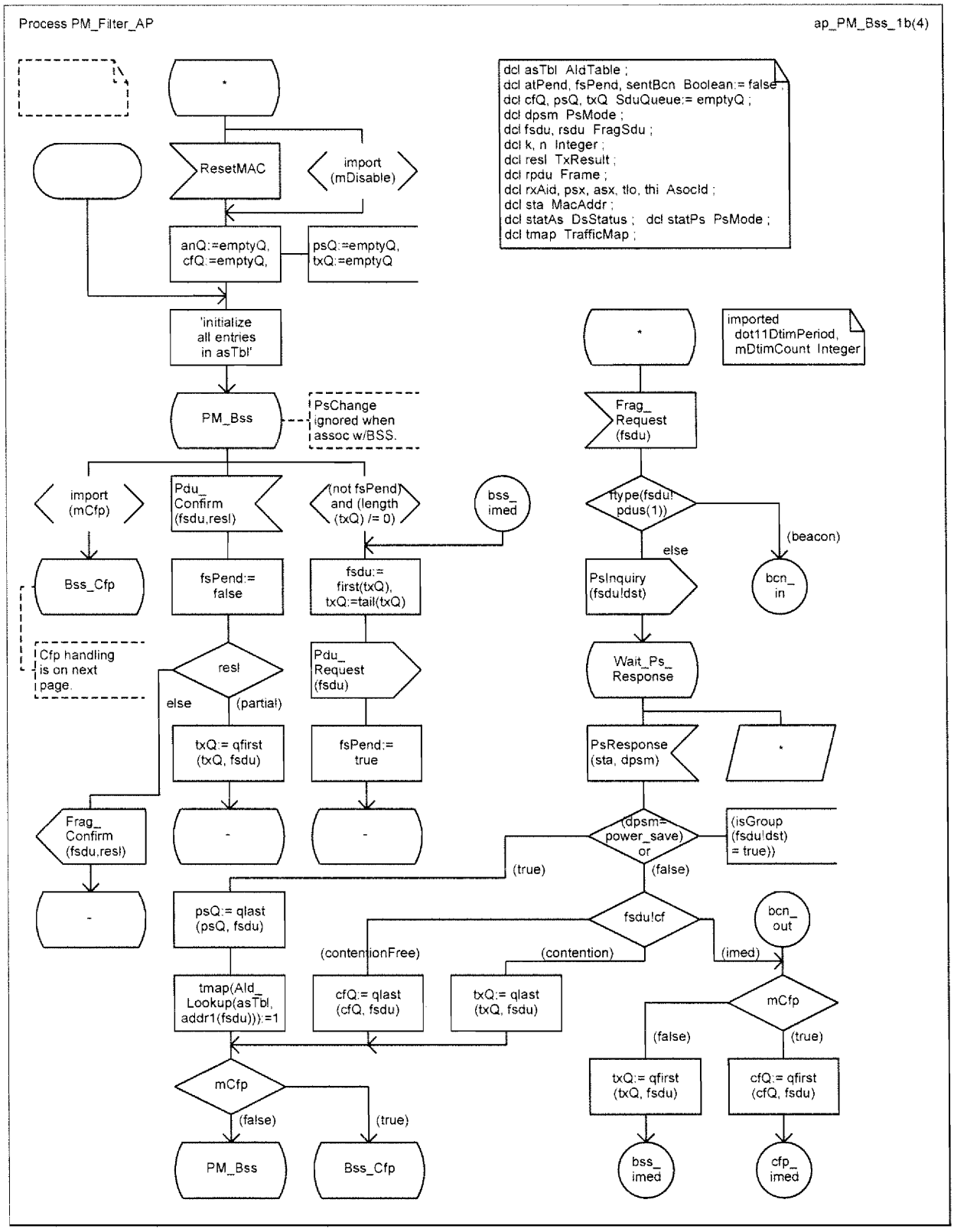


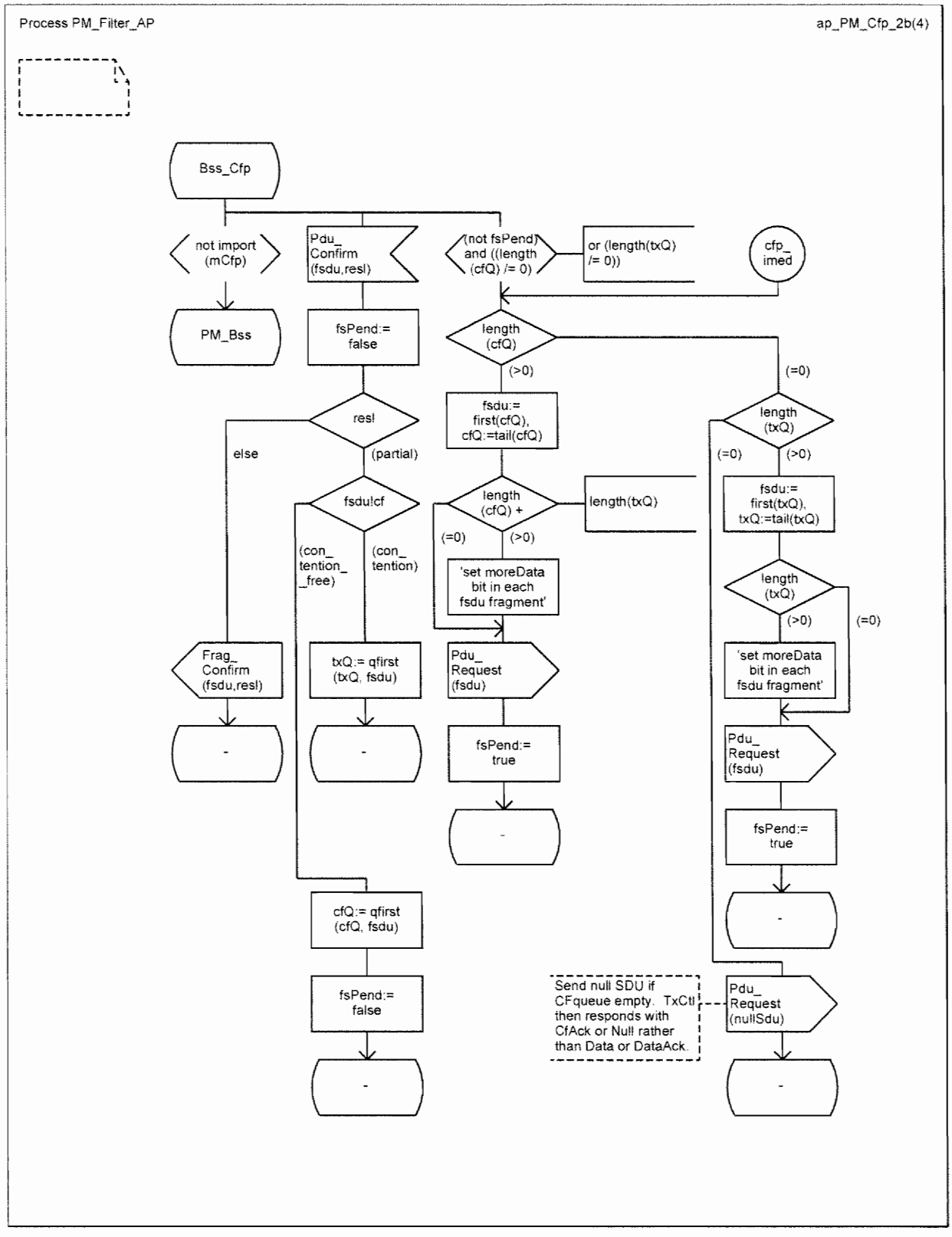


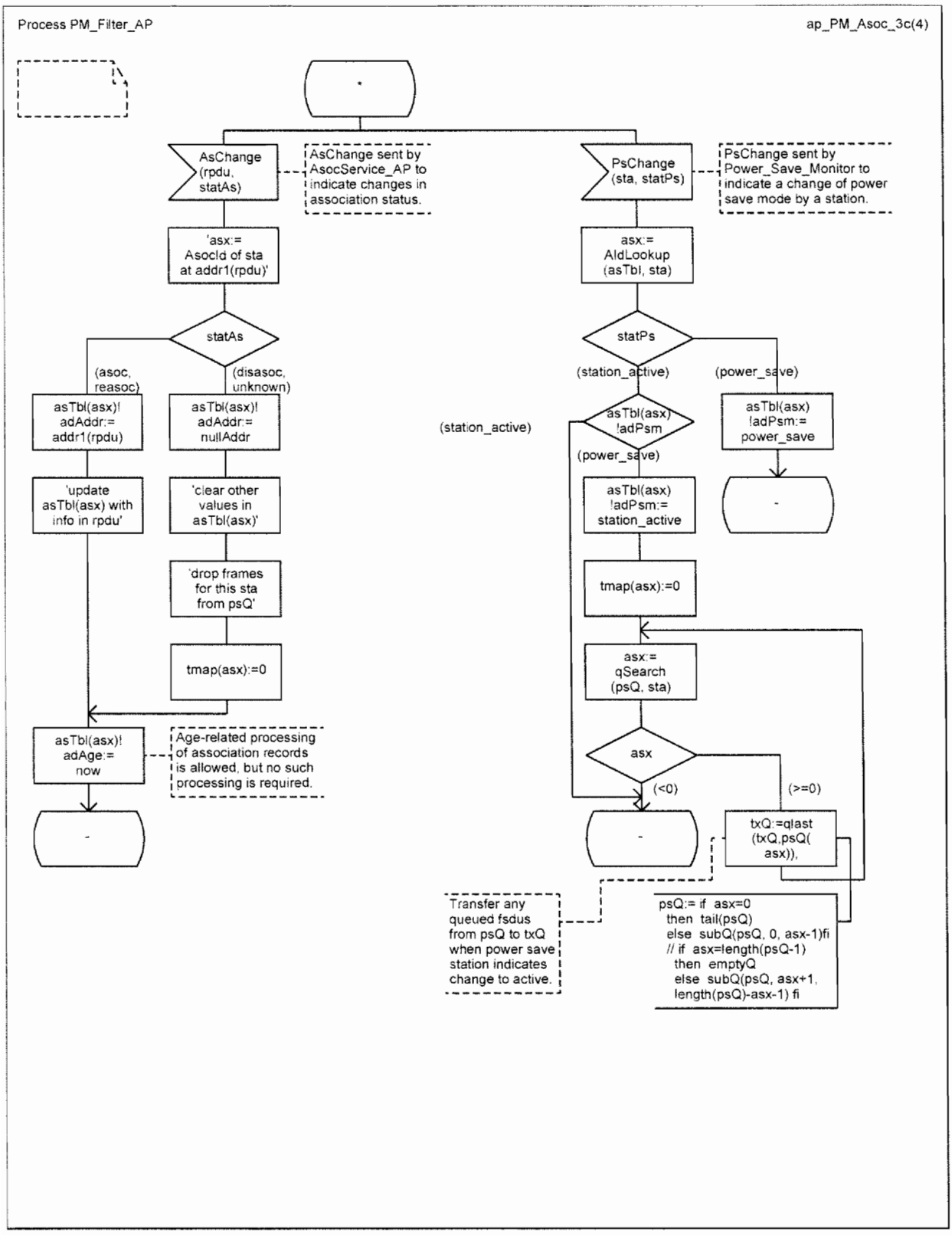


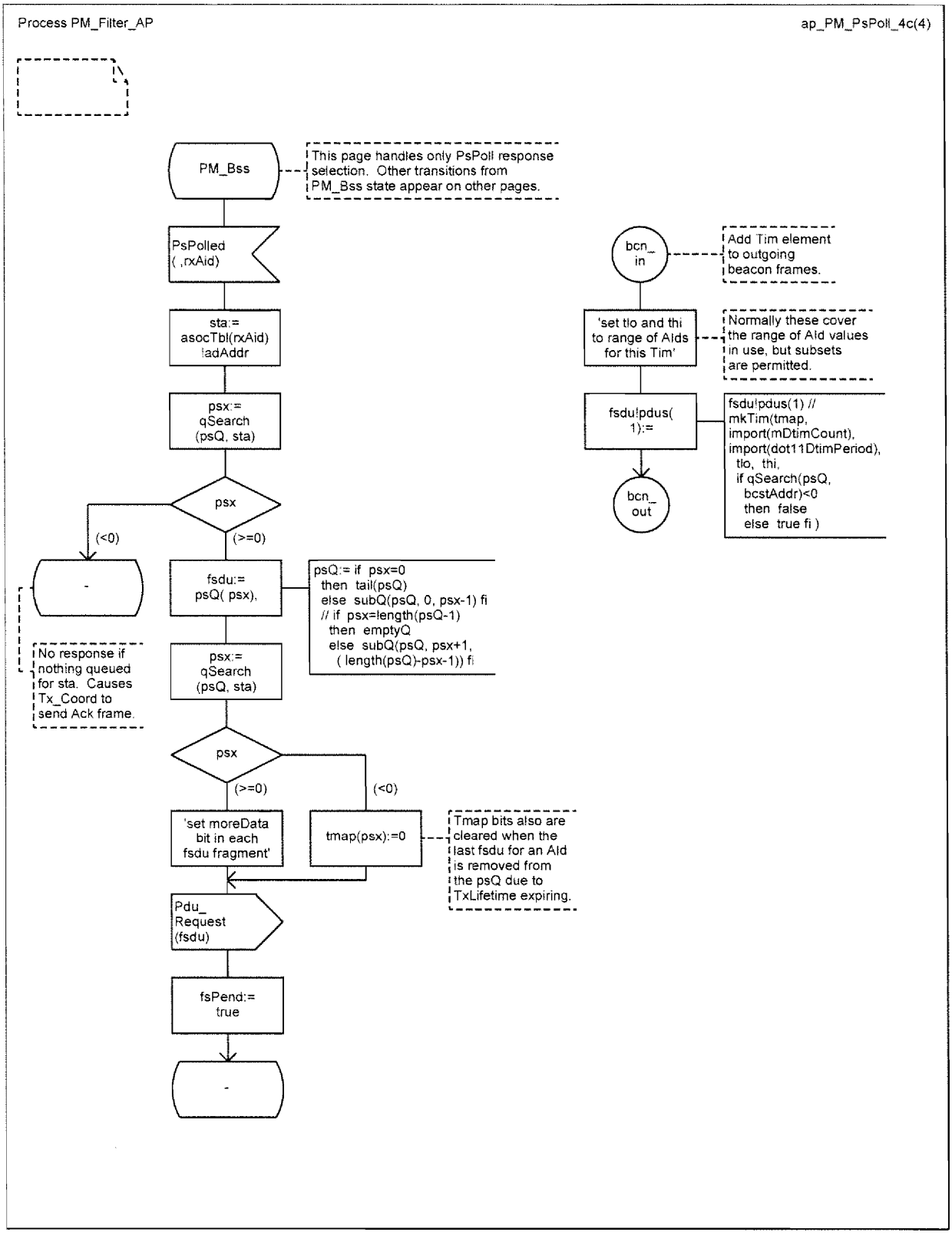


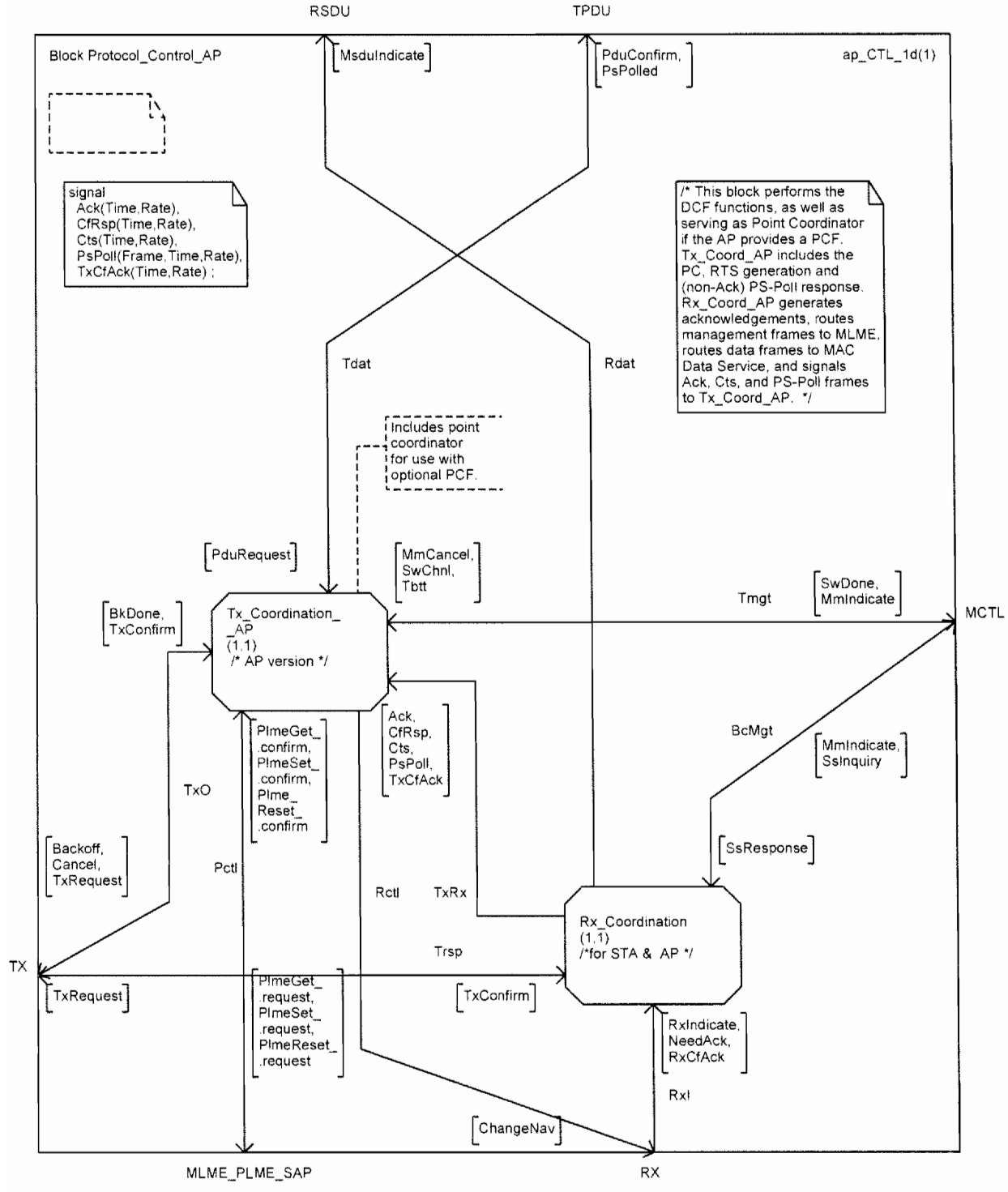


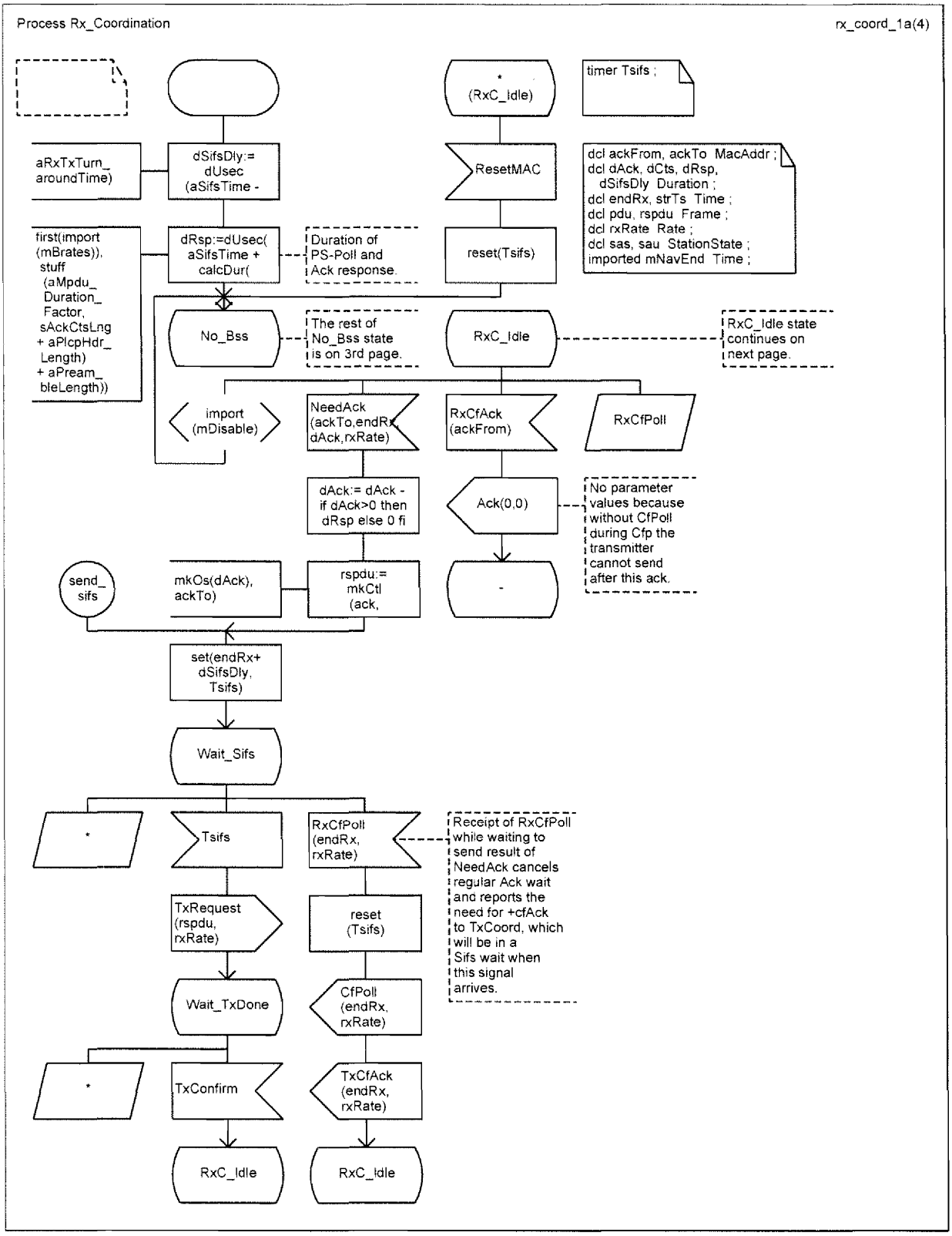


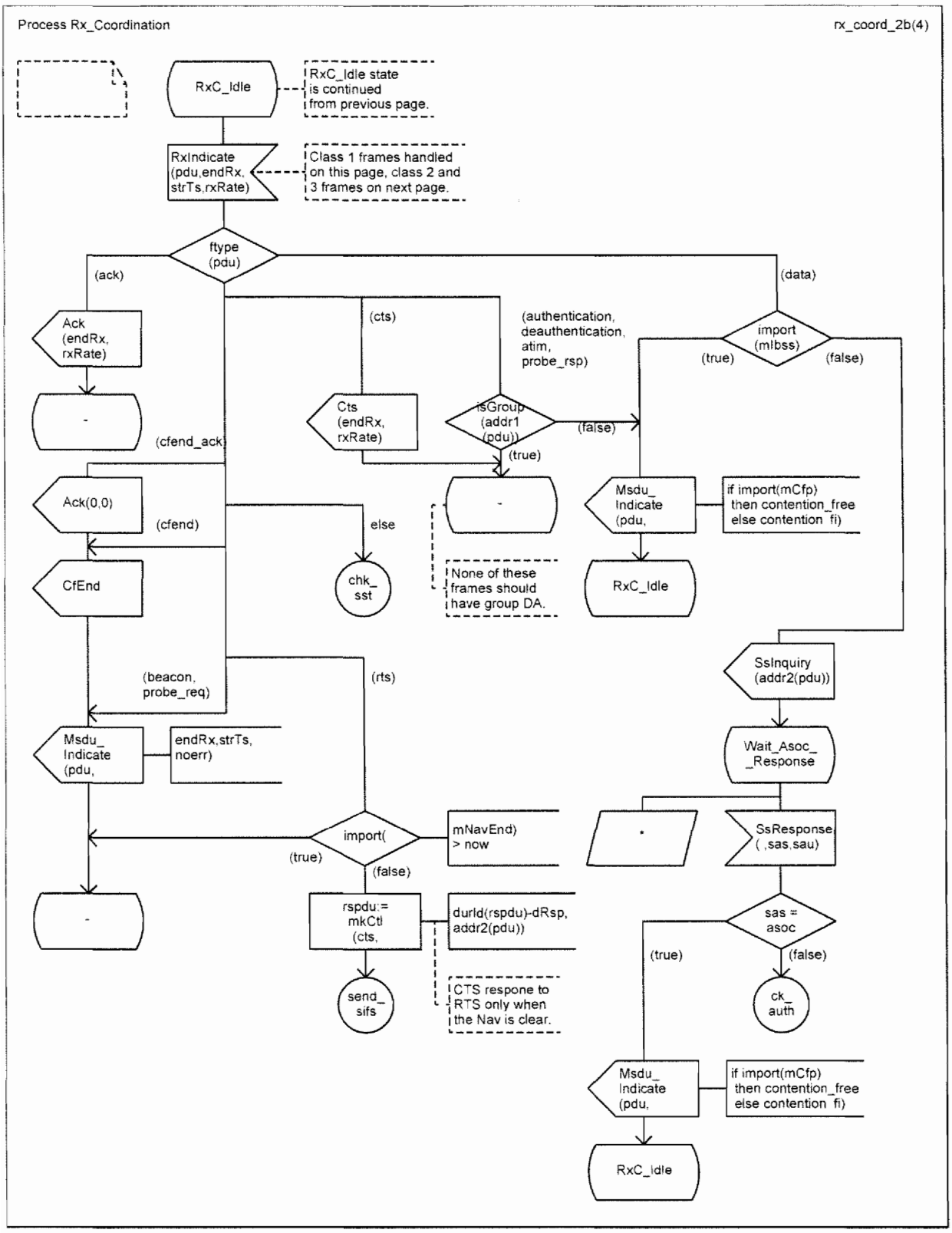


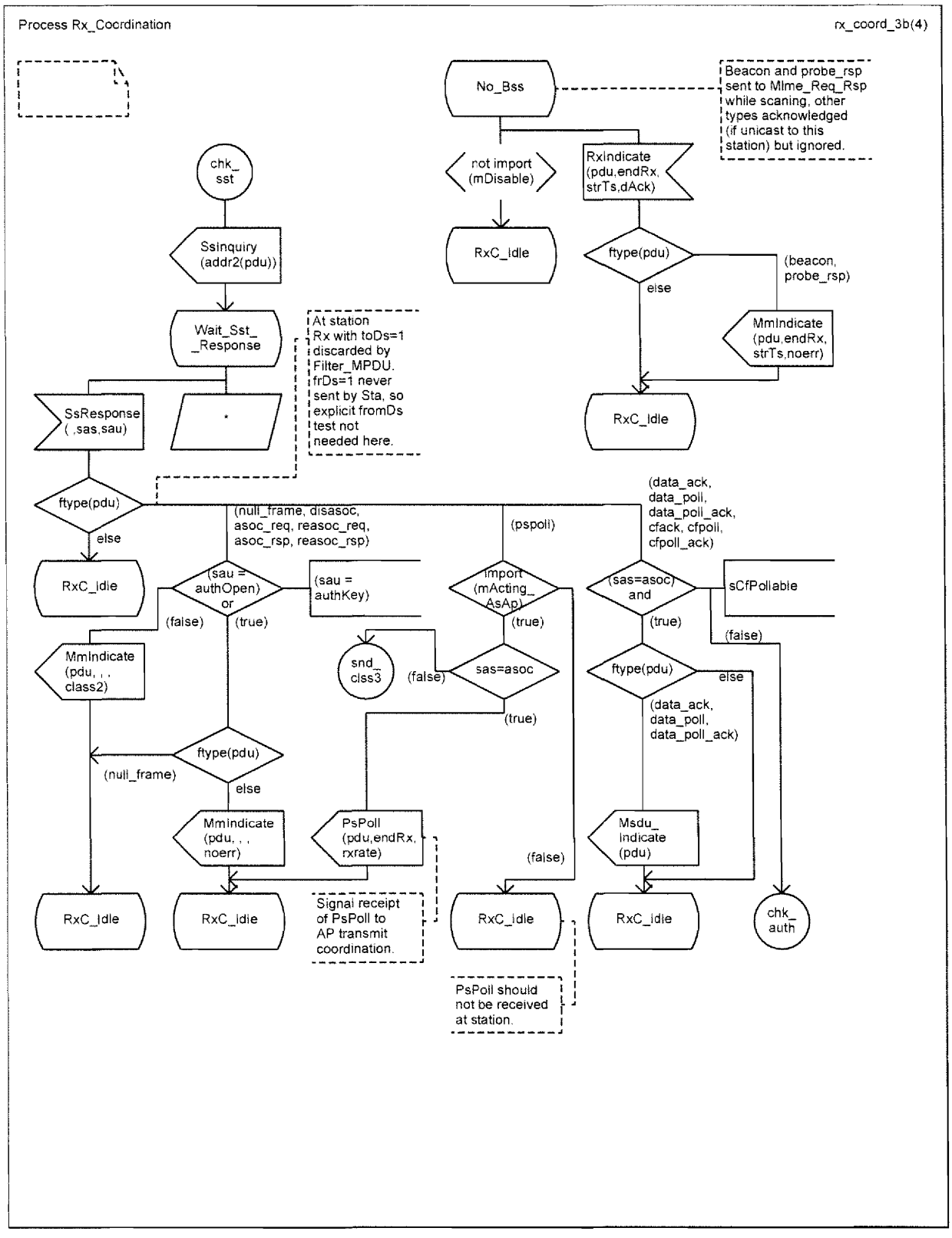


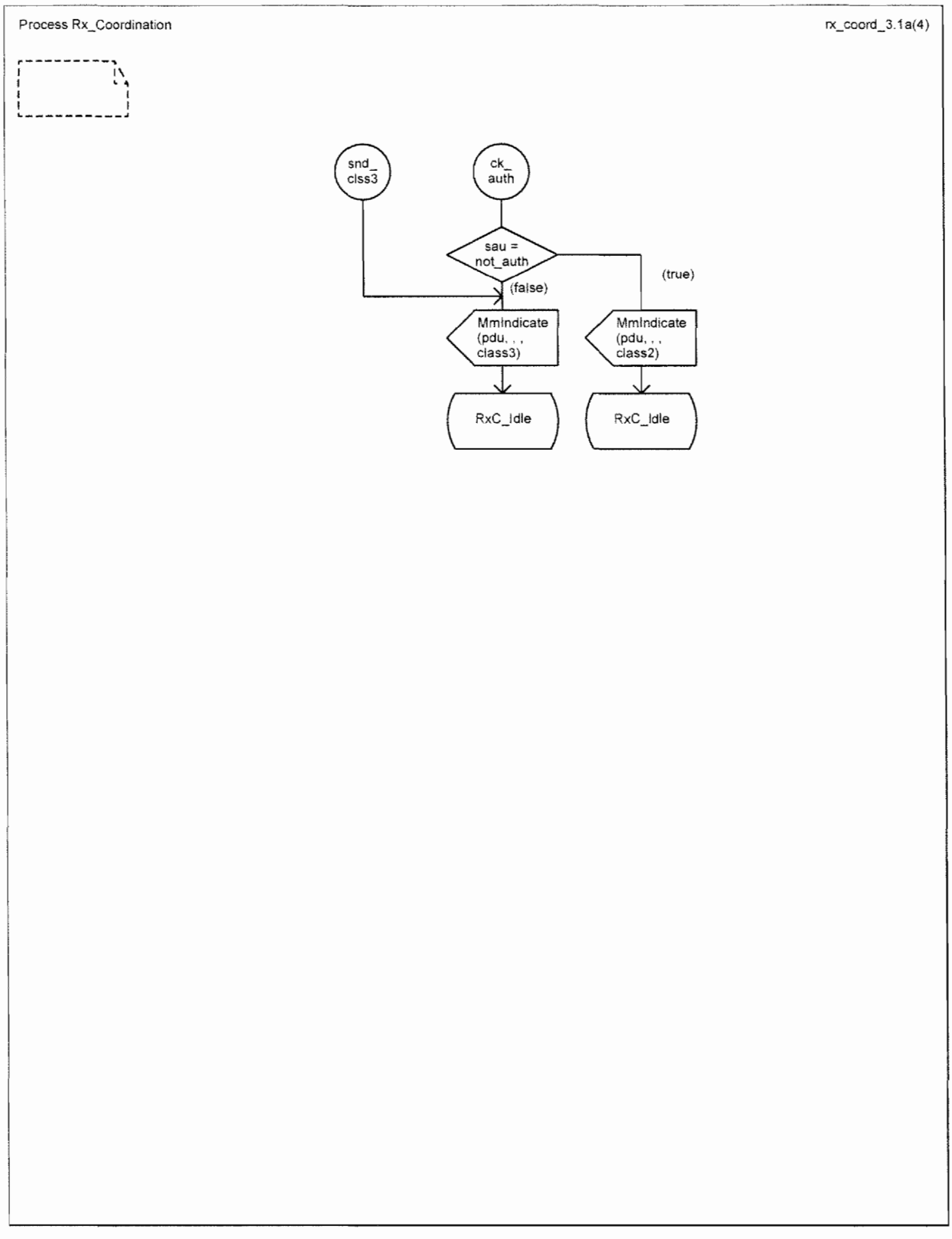


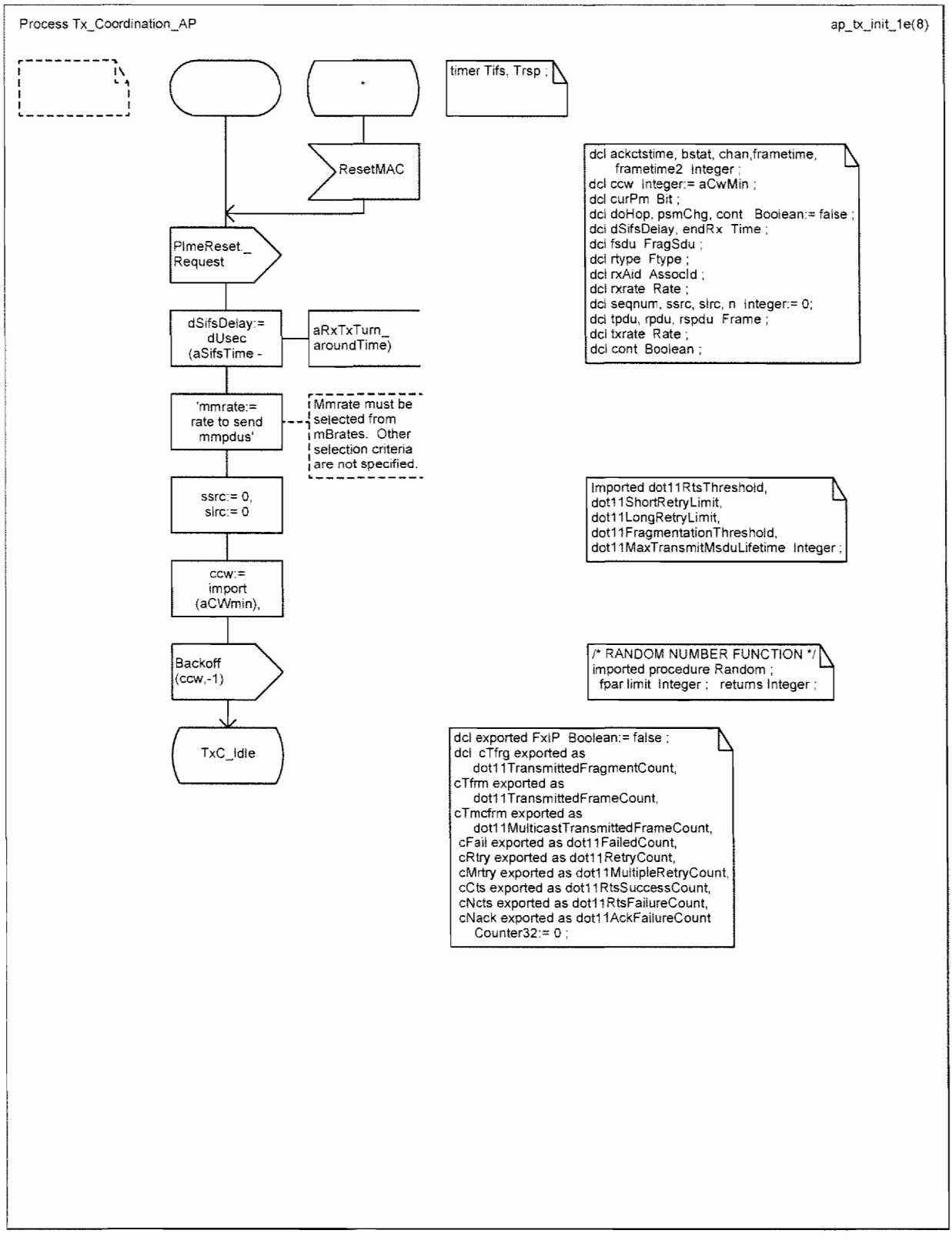


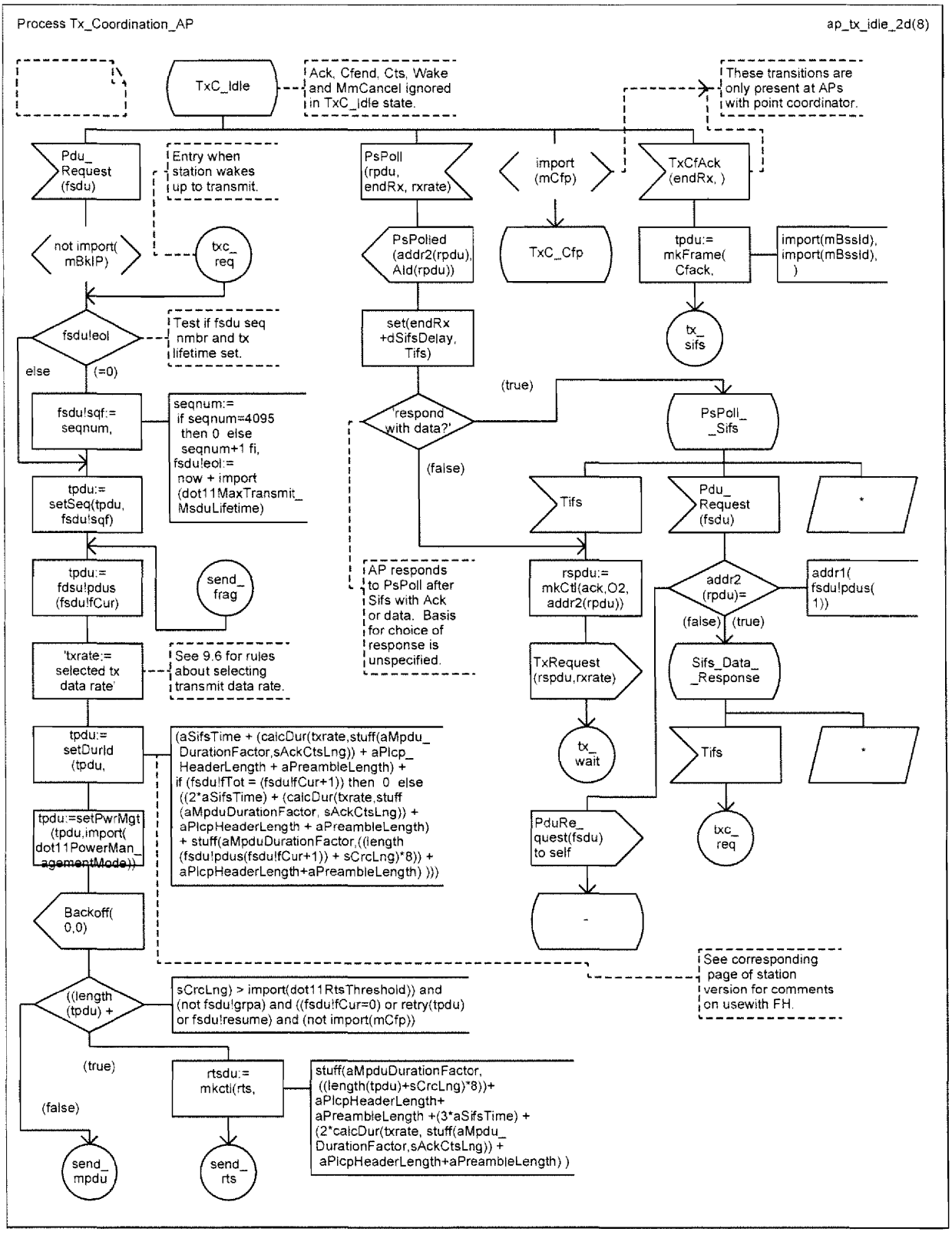


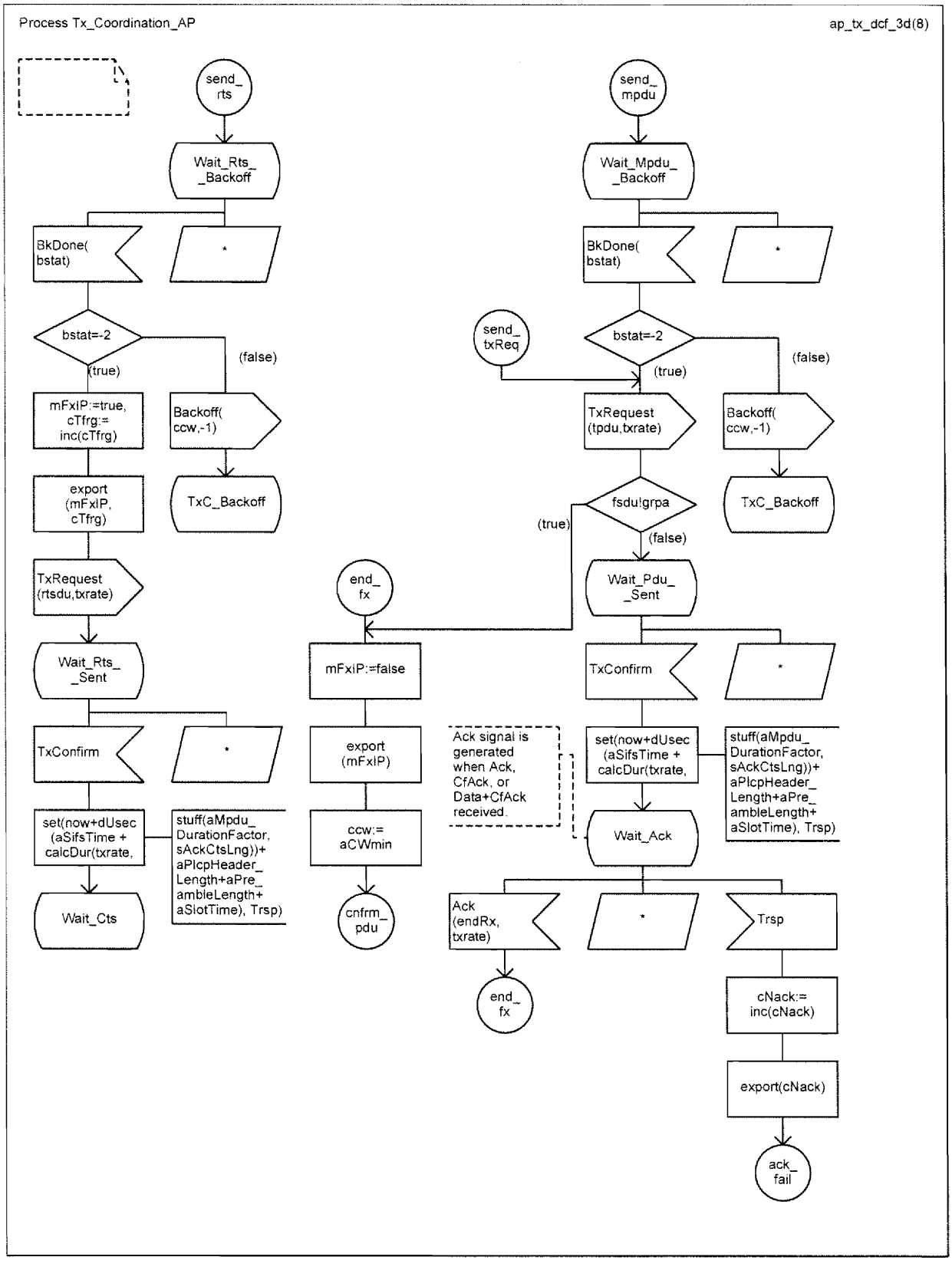


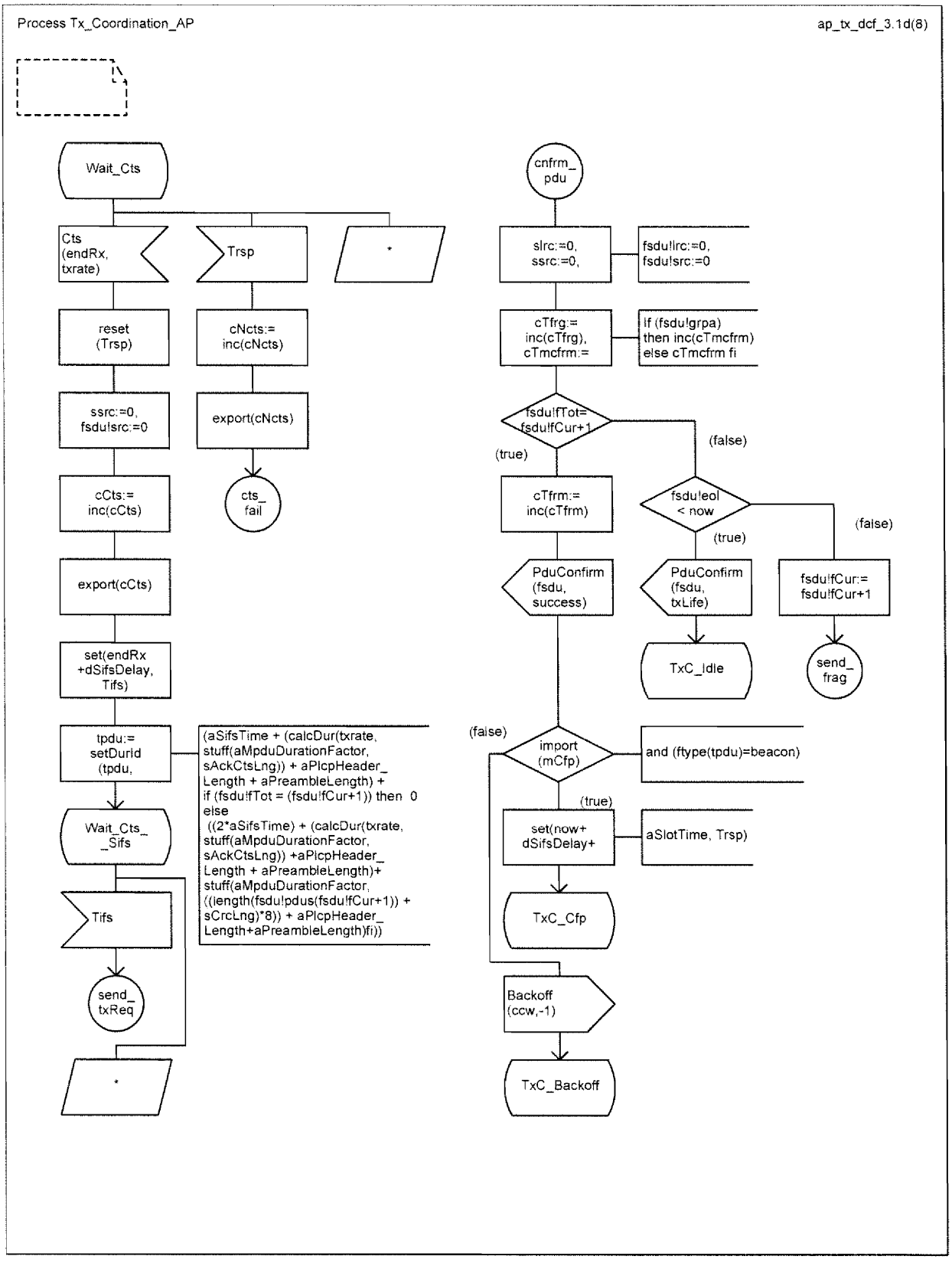


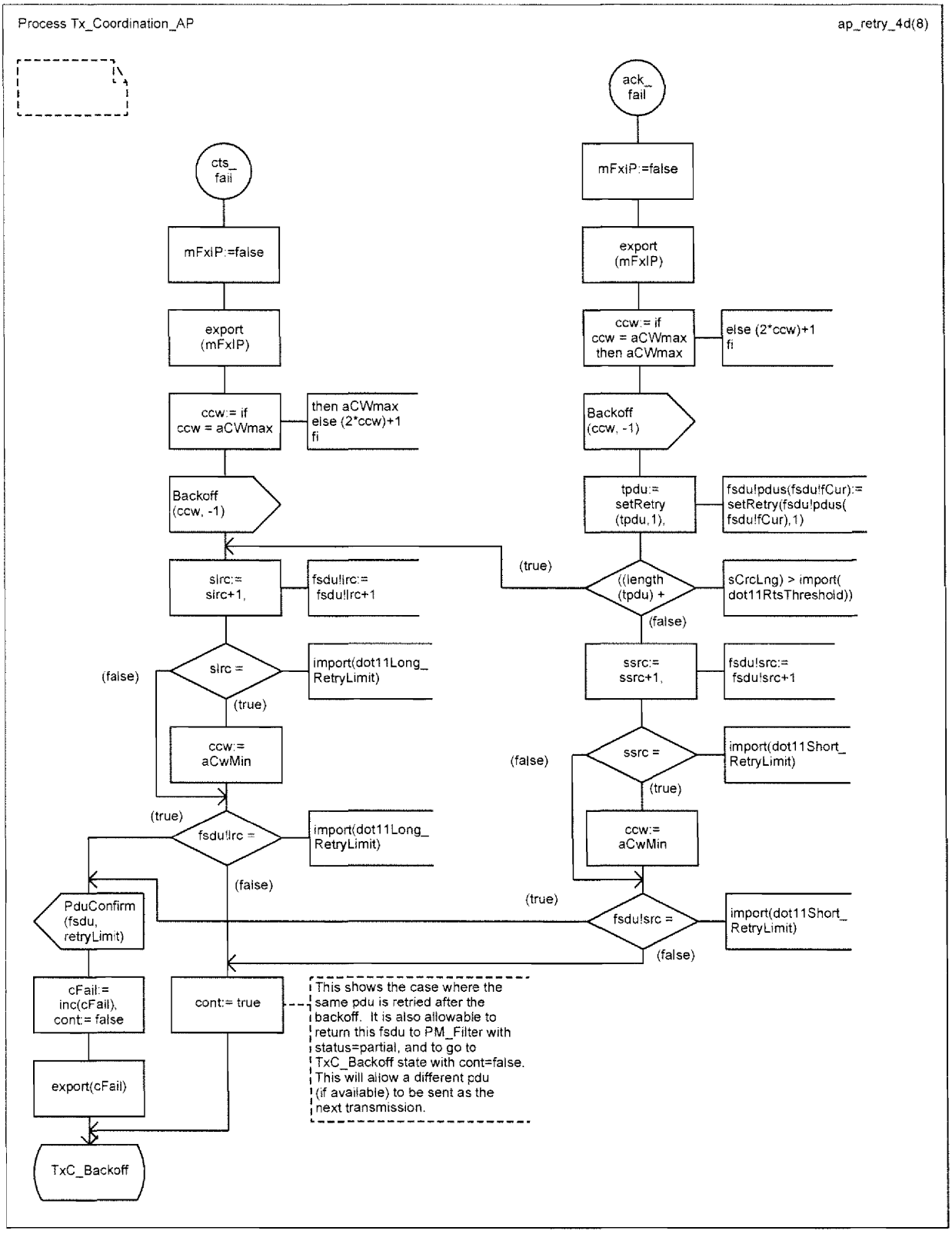


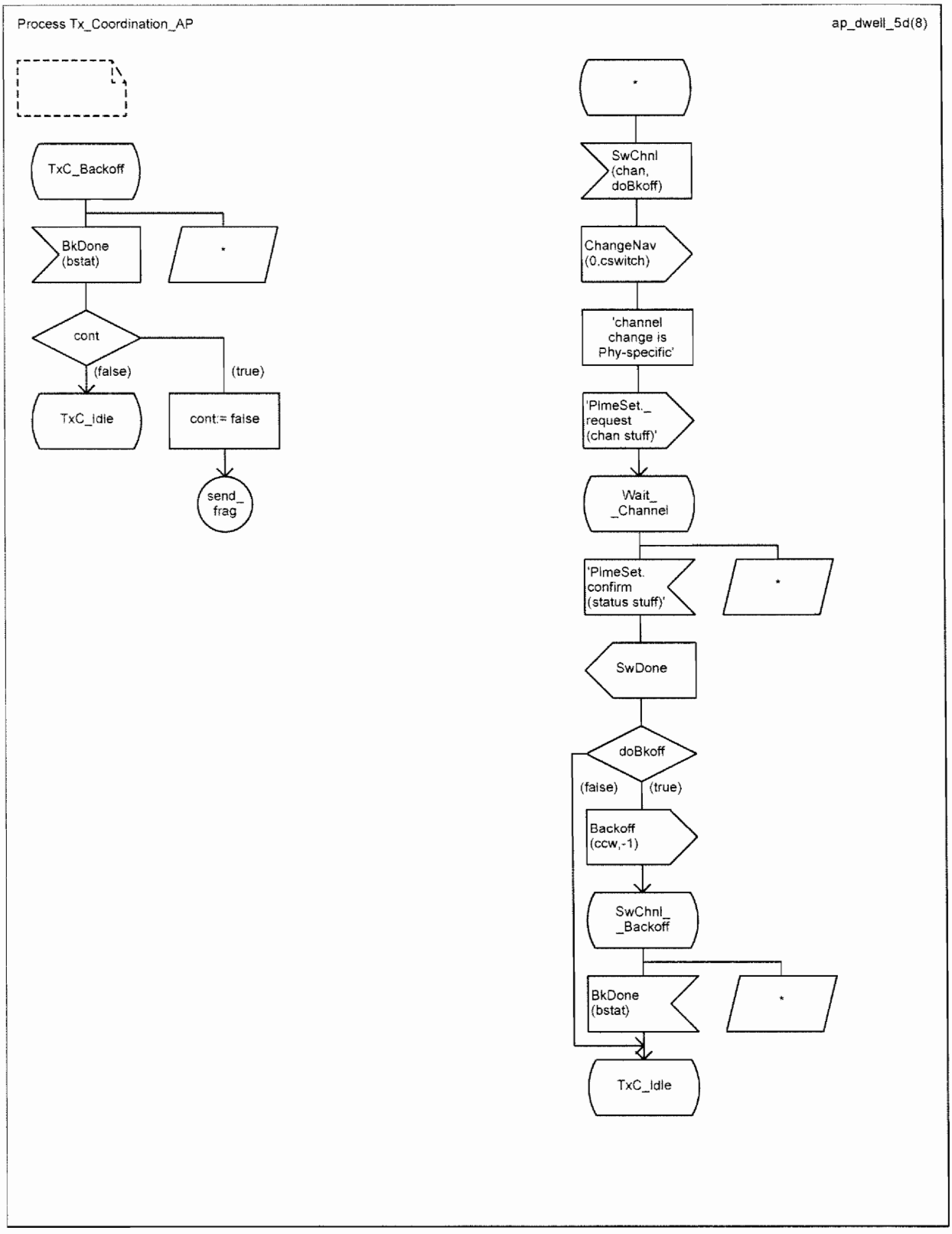


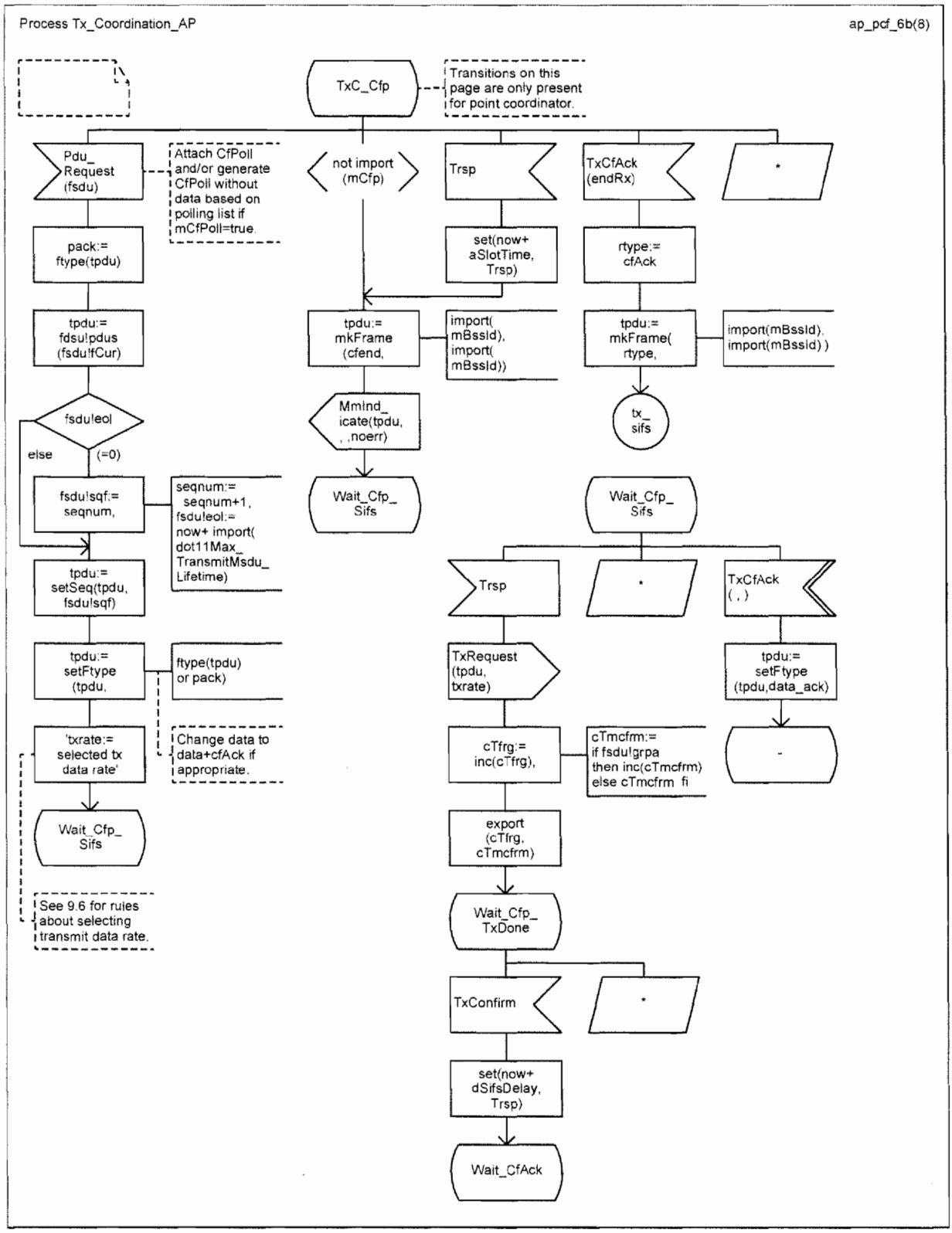


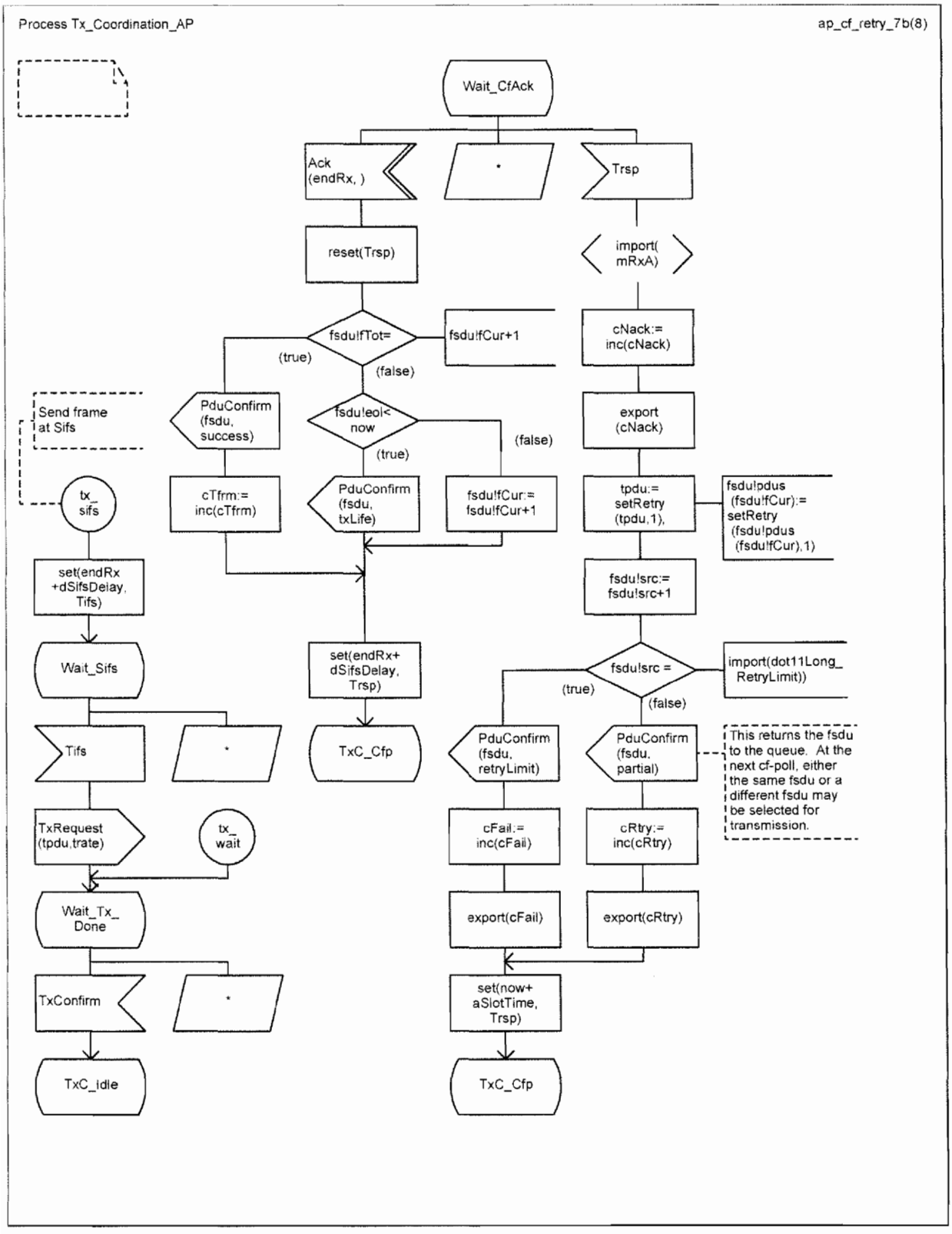


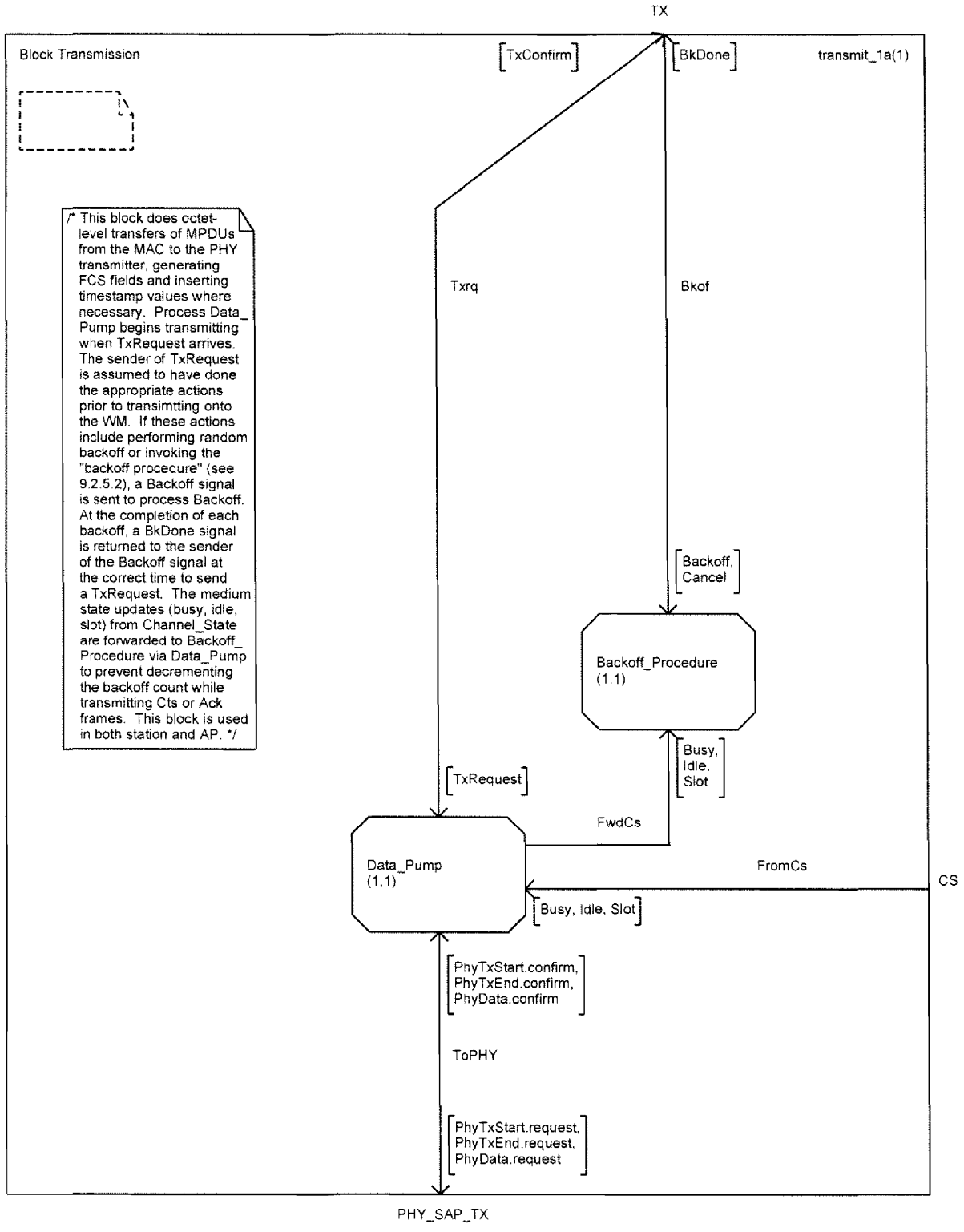


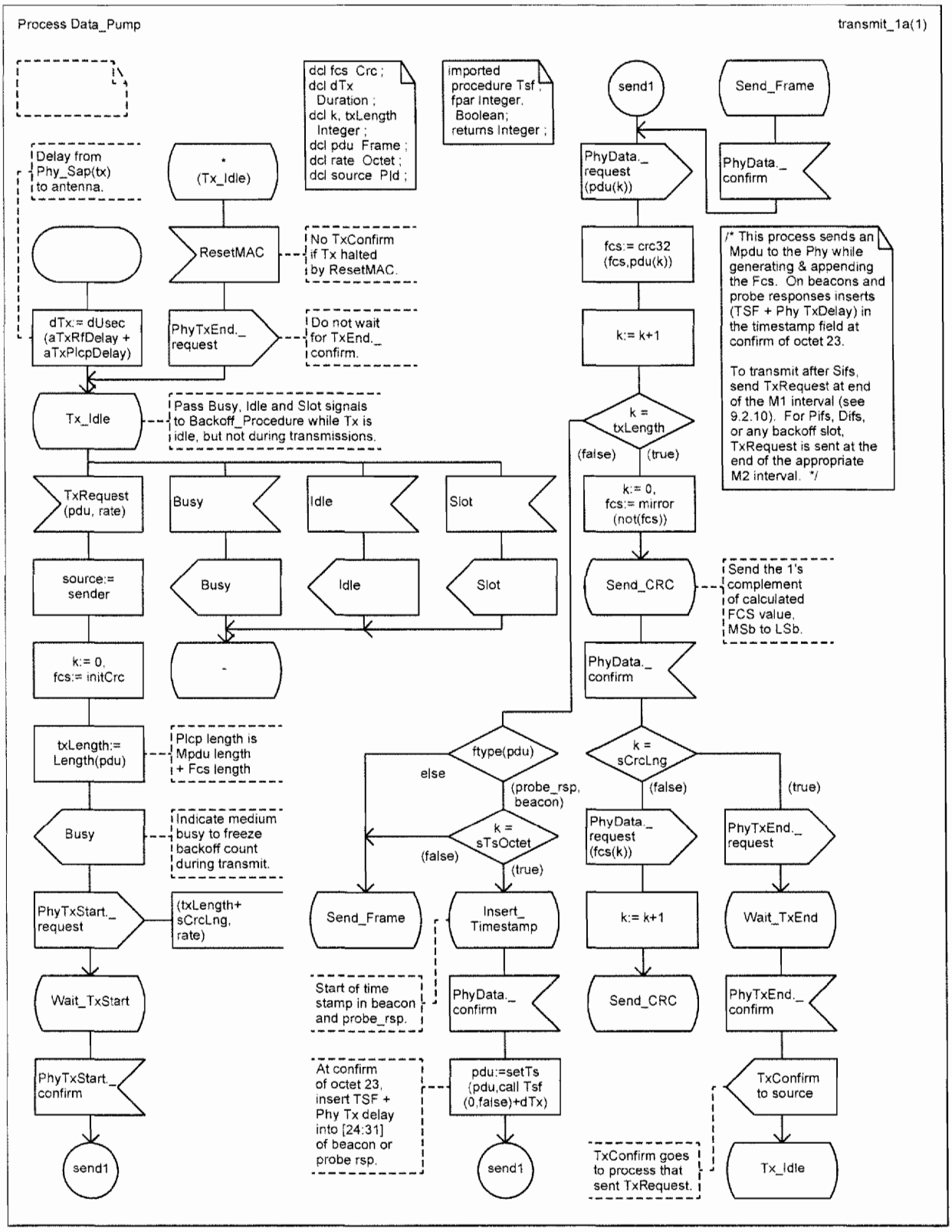


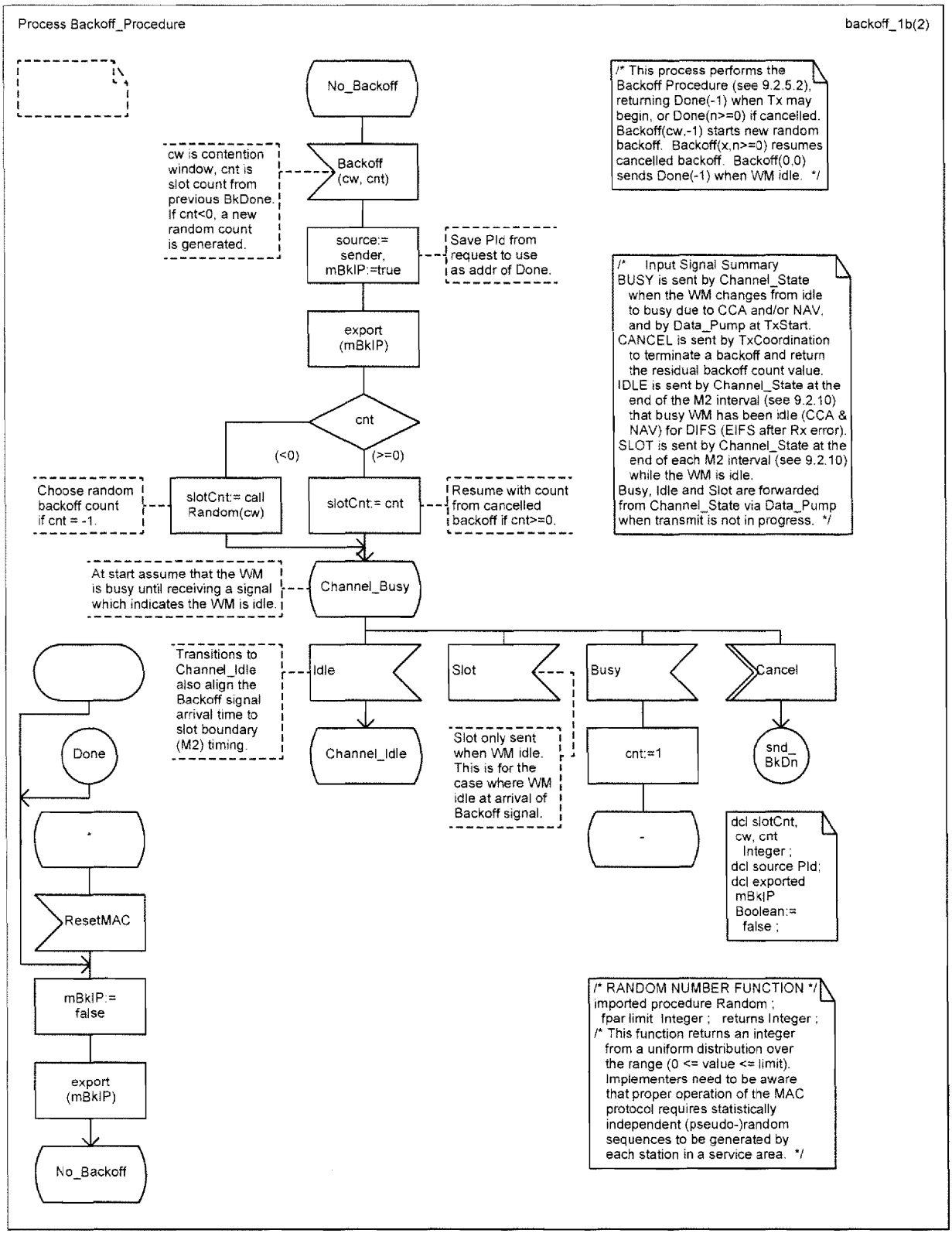


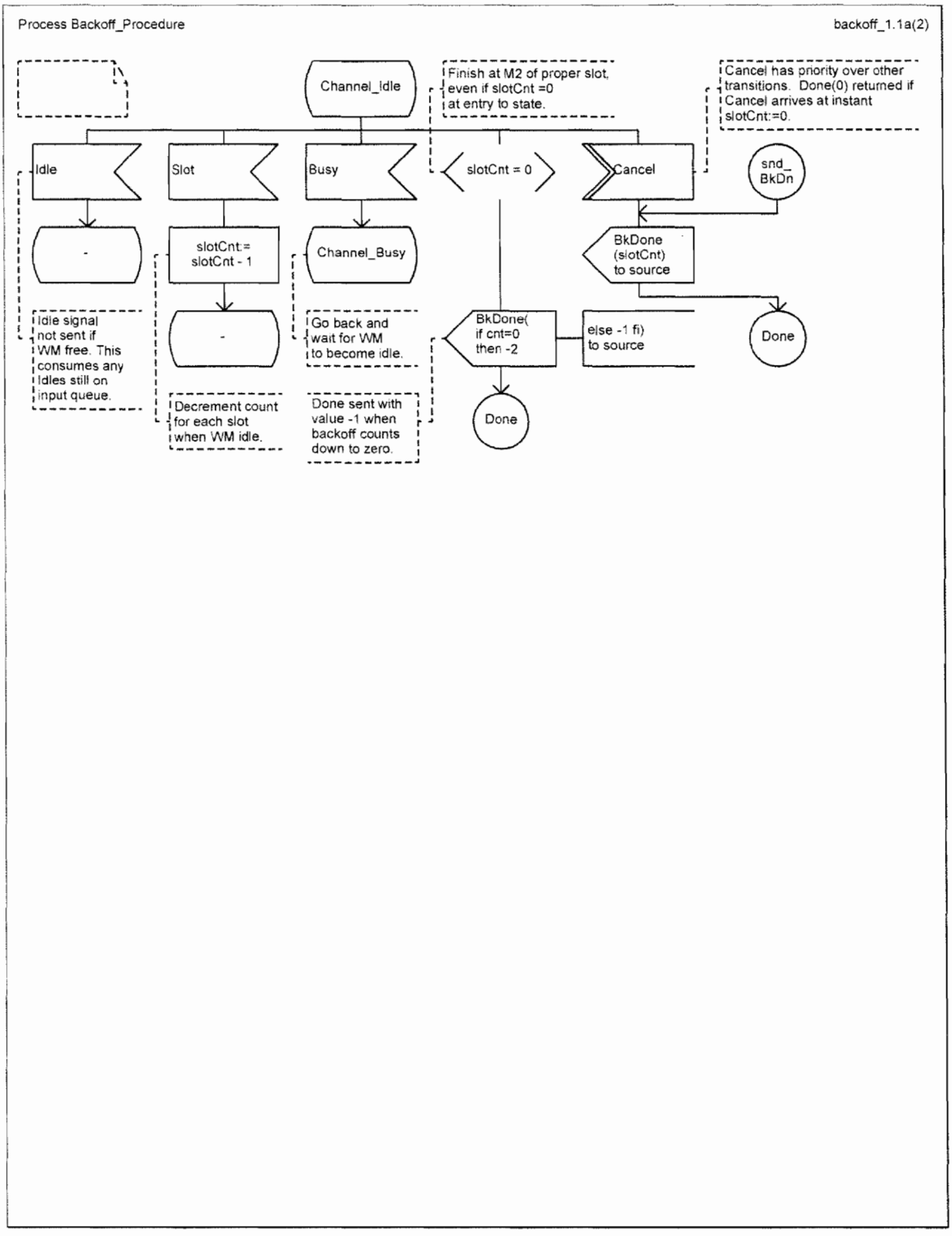


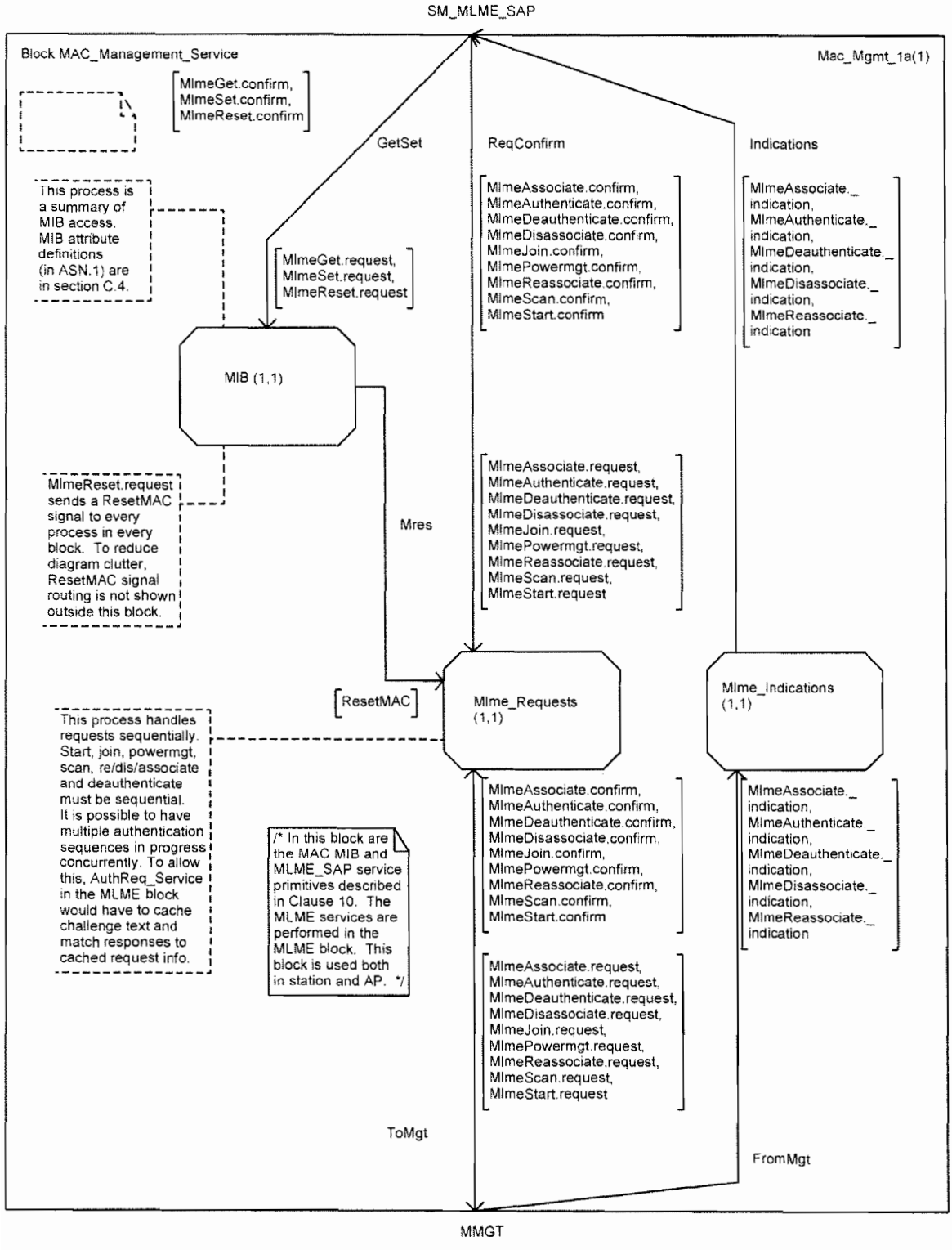


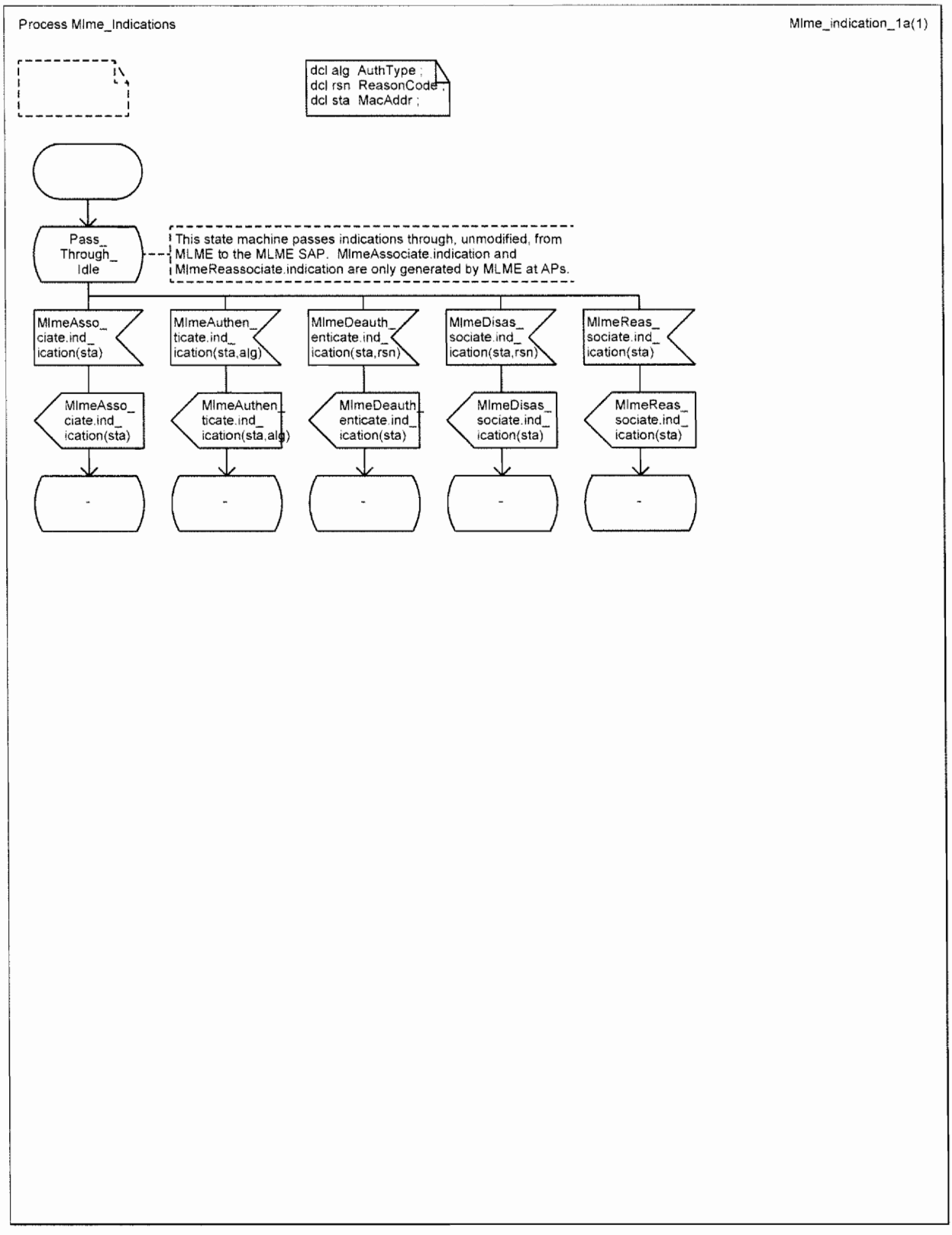


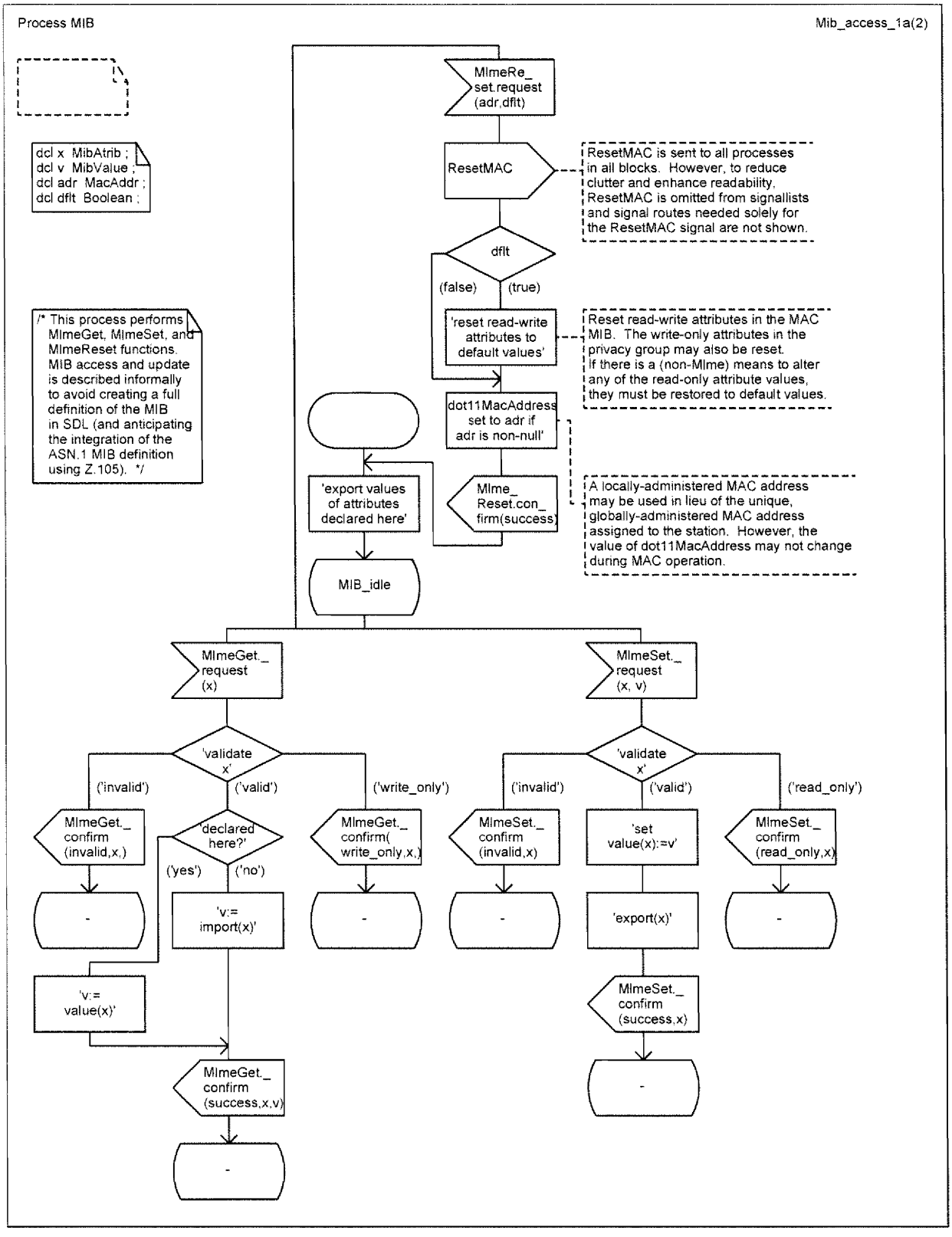












Process MIB

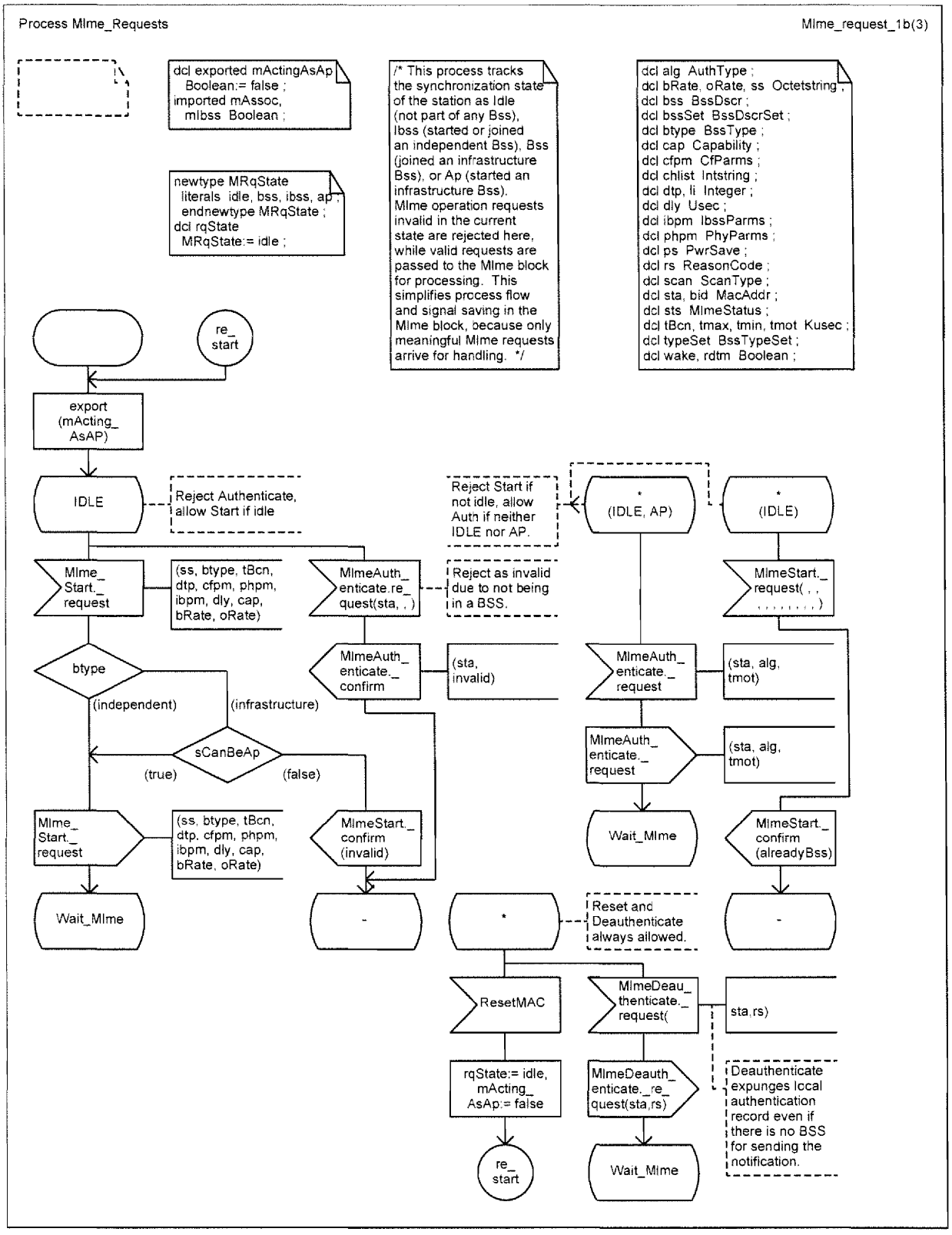
Mib_import_export_2b(2)

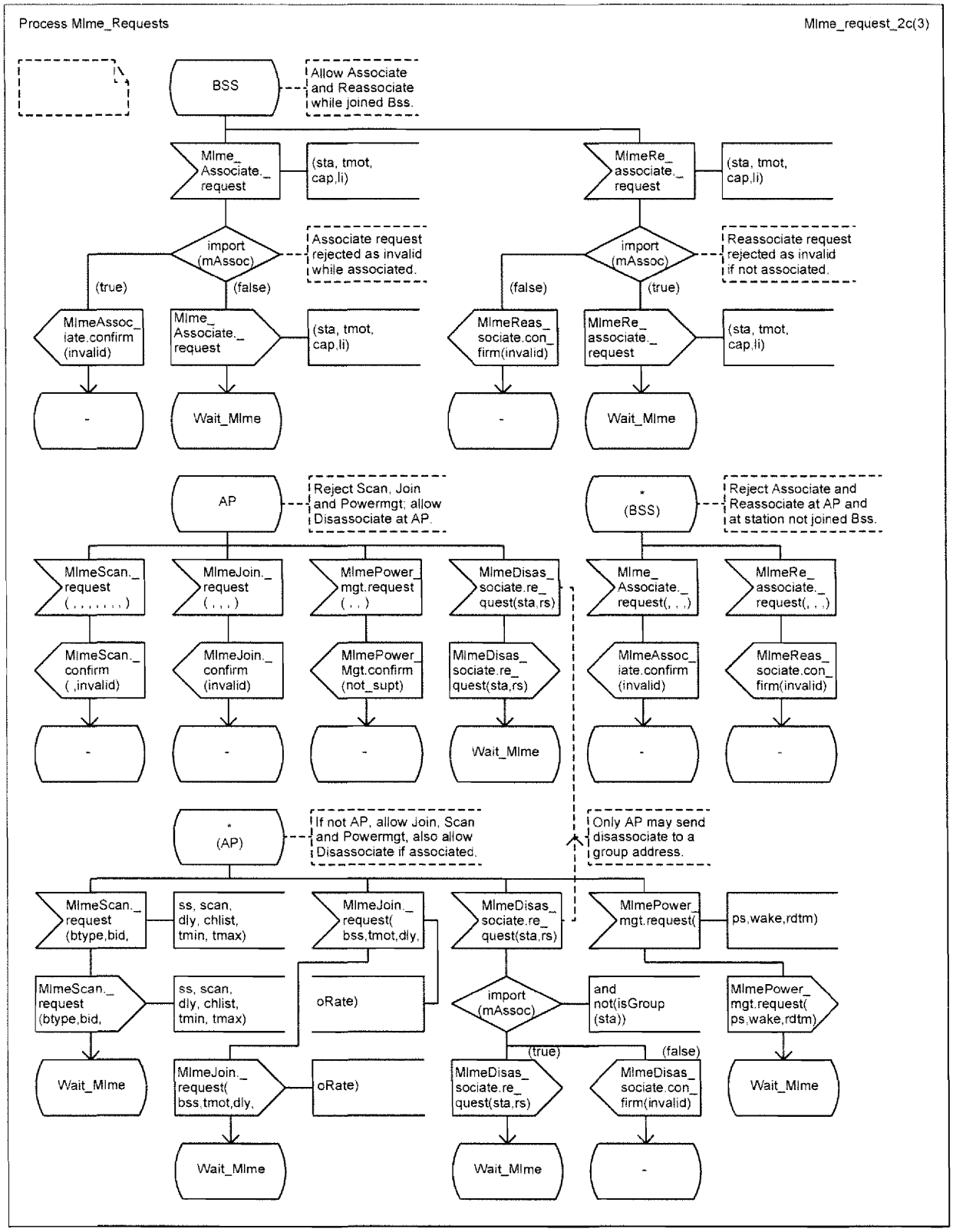
```
/* Import of {Read-Only} MIB counter
   values exported from other processes */
imported
  dot11AckFailureCount,
  dot11FailedCount,
  dot11FcsErrorCount,
  dot11FrameDuplicateCount,
  dot11MulticastReceivedFrameCount,
  dot11MulticastTransmittedFrameCount,
  dot11MultipleRetryCount,
  dot11ReceivedFragmentCount,
  dot11RetryCount,
  dot11RtsFailureCount,
  dot11RtsSuccessCount,
  dot11TransmittedFragmentCount,
  dot11WepExcludedCount,
  dot11WepIcvErrorCount,
  dot11WepUndecryptableCount  Counter32 ;
```

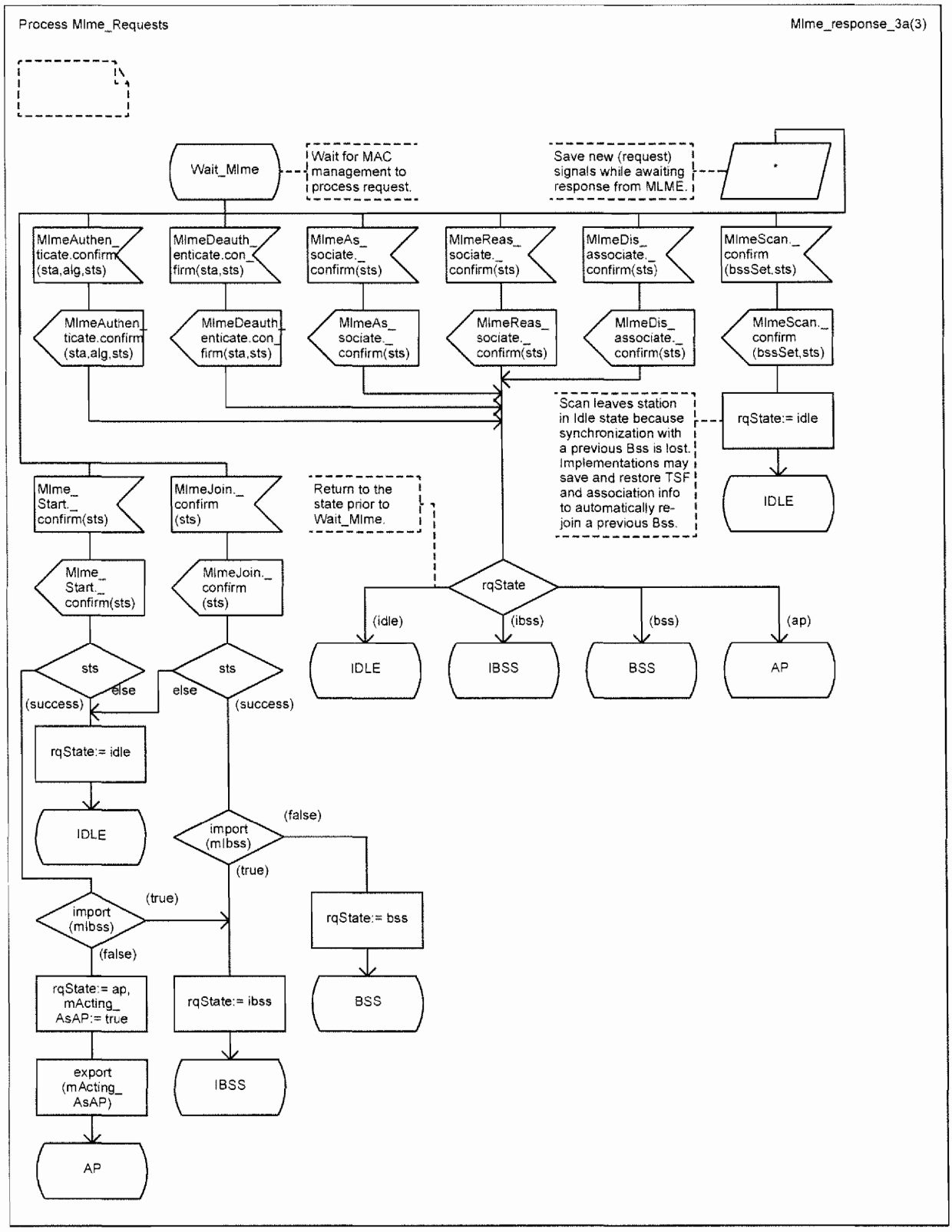
```
/* Declarations of MIB attributes exported from
   this process */

      /* Read-Write attributes */
dcl exported
  dot11AuthenticationAlgorithms  AuthTypeSet:=
    incl(open_system, shared_key),
  dot11ExcludeUnencrypted  Boolean:= false,
  dot11FragmentationThreshold  Integer:= 2346,
  dot11GroupAddresses  MacAddrSet:= empty,
  dot11LongRetryLimit  Integer:= 4,
  dot11MaxReceiveLifetime  Kusec:= 512,
  dot11MaxTransmitMsduLifetime  Kusec:= 512,
  dot11MediumOccupancyLimit  Kusec:= 100,
  dot11PrivacyInvoked  Boolean:= false,
  mReceiveDTIMs  Boolean:= true,
  dot11CfpPeriod  Integer:= 1,
  dot11CfpMaxDuration  Kusec:= 200,
  dot11AuthenticationResponseTimeout  Kusec:= 512,
  dot11RtsThreshold  Integer:= 3000,
  dot11ShortRetryLimit  Integer:= 7,
  dot11WepDefaultKeyId  KeyIndex:= 0,
  dot11CurrentChannelNumber  Integer:= 0,
  dot11CurrentSet  Integer:= 0,
  dot11CurrentPattern  Integer:= 0,
  dot11CurrentIndex  Integer:= 0 ;

      /* Write-Only attributes */
dcl exported
  dot11WepDefaultKeys  KeyVector:= nullKey,
  dot11WepKeyMappings
    KeyMapArray:= (. nullAddr, false, nullKey .) ;
```

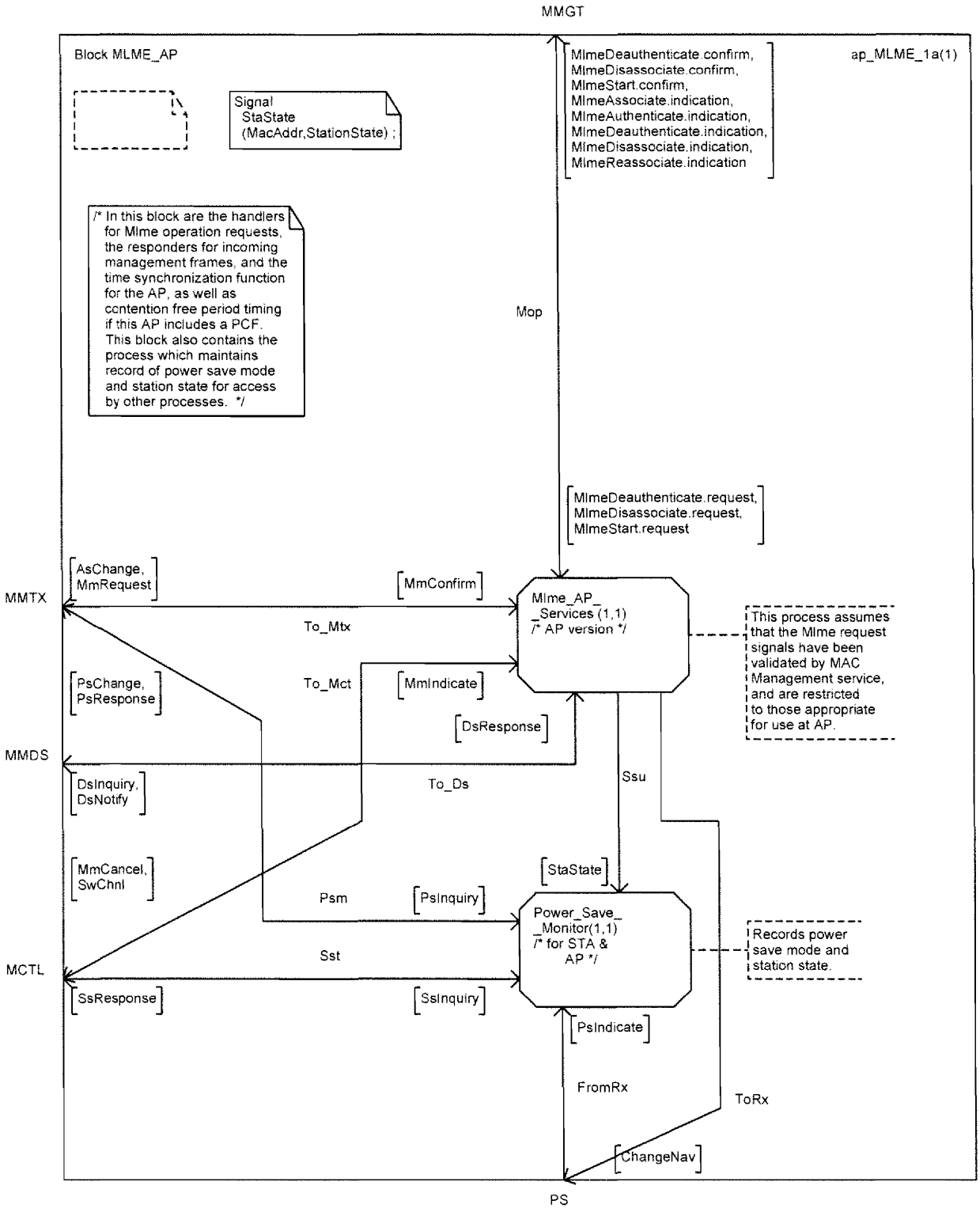
```
/* The following Read-Only attributes in the
   MAC MIB are defined as synonyms (named
   constants) rather than remote variables
   because they describe properties of the
   station which are static, at least during
   any single instance of MAC operation:
     dot11AuthenticationAlgorithms  AuthTypeSet,
     dot11CfPollable  Boolean,
     dot11MacAddress  MacAddr,
     dot11ManufacturerID  Octetstring,
     dot11PrivacyOptionImplemented  Boolean,
     dot11ProductID  Octetstring,
     aStationID  MacAddr,
     dot11WepKeyMappingLength  Integer ;

   In addition, all Read-Only attributes in the
   PHY MIB which are accessed by the MAC
   are defined as synonyms.
*/
```

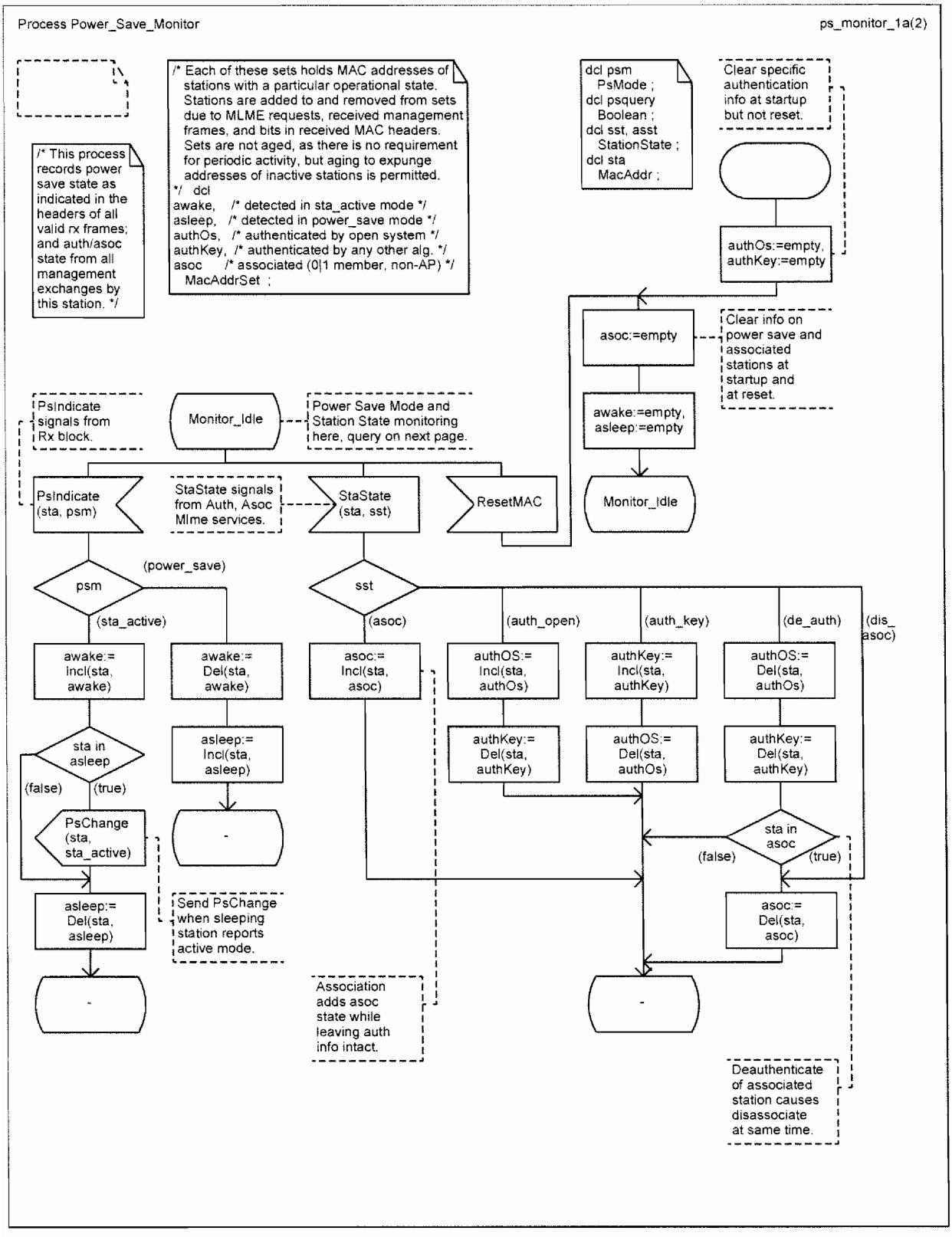
```
/* NOTE:
   The values listed for MAC MIB attributes are the
   specified default values for those attributes.
   The values listed for PHY MIB attributes are either
   the default values for the FH PHY, or arbitrary
   values within the specified range.  The specific
   values for PHY attributes in this SDL description
   of the MAC do not have normative significance.
*/
```

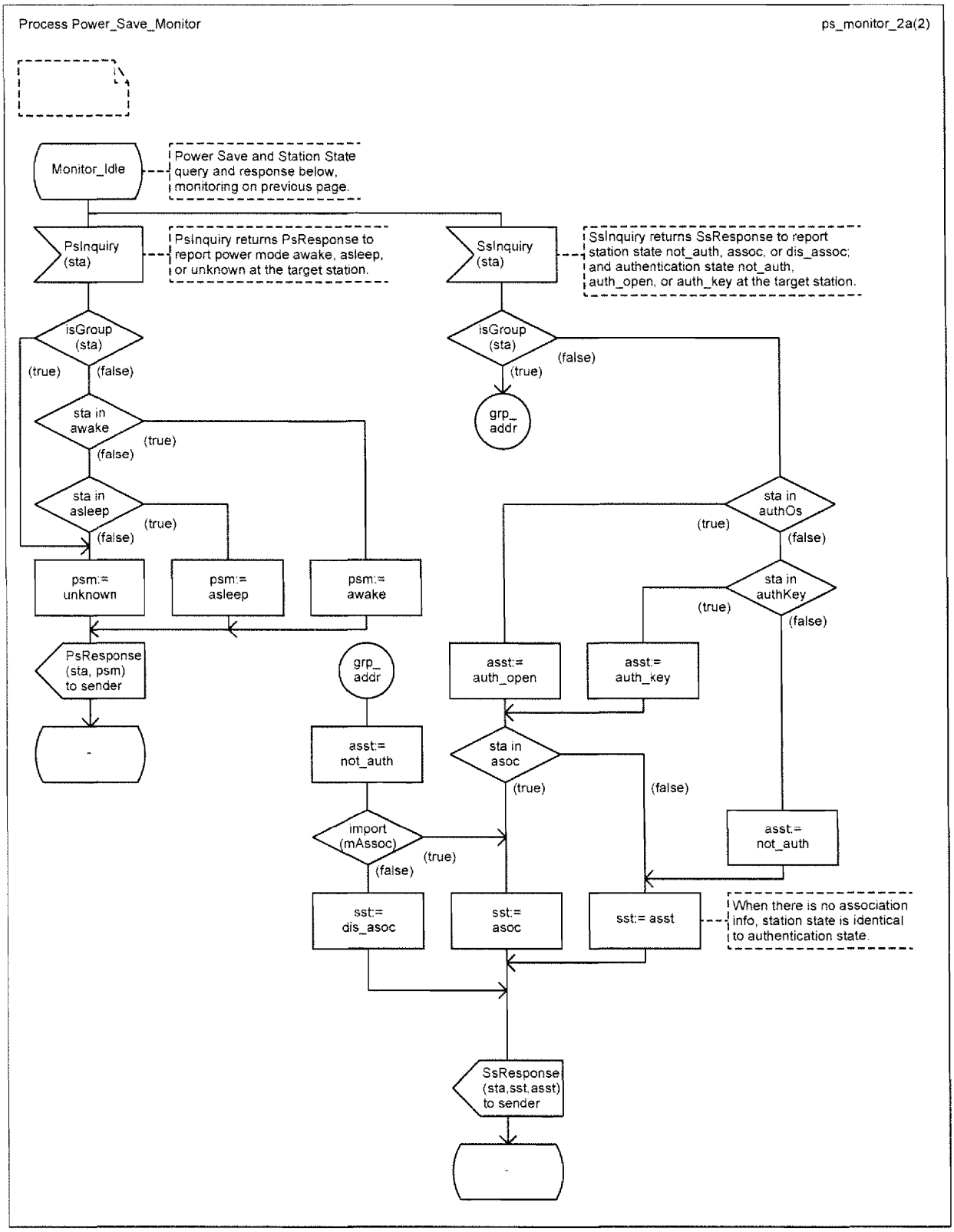


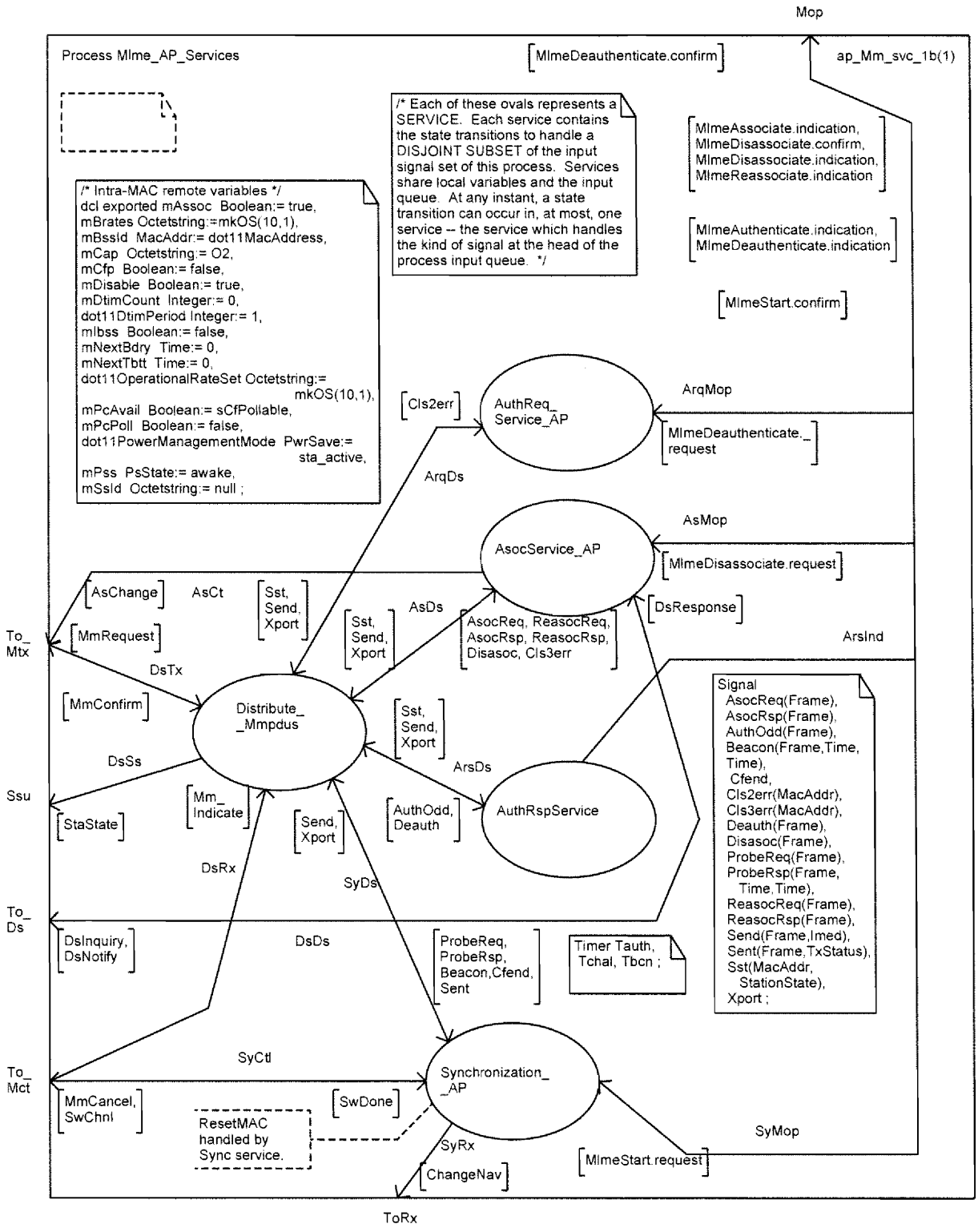


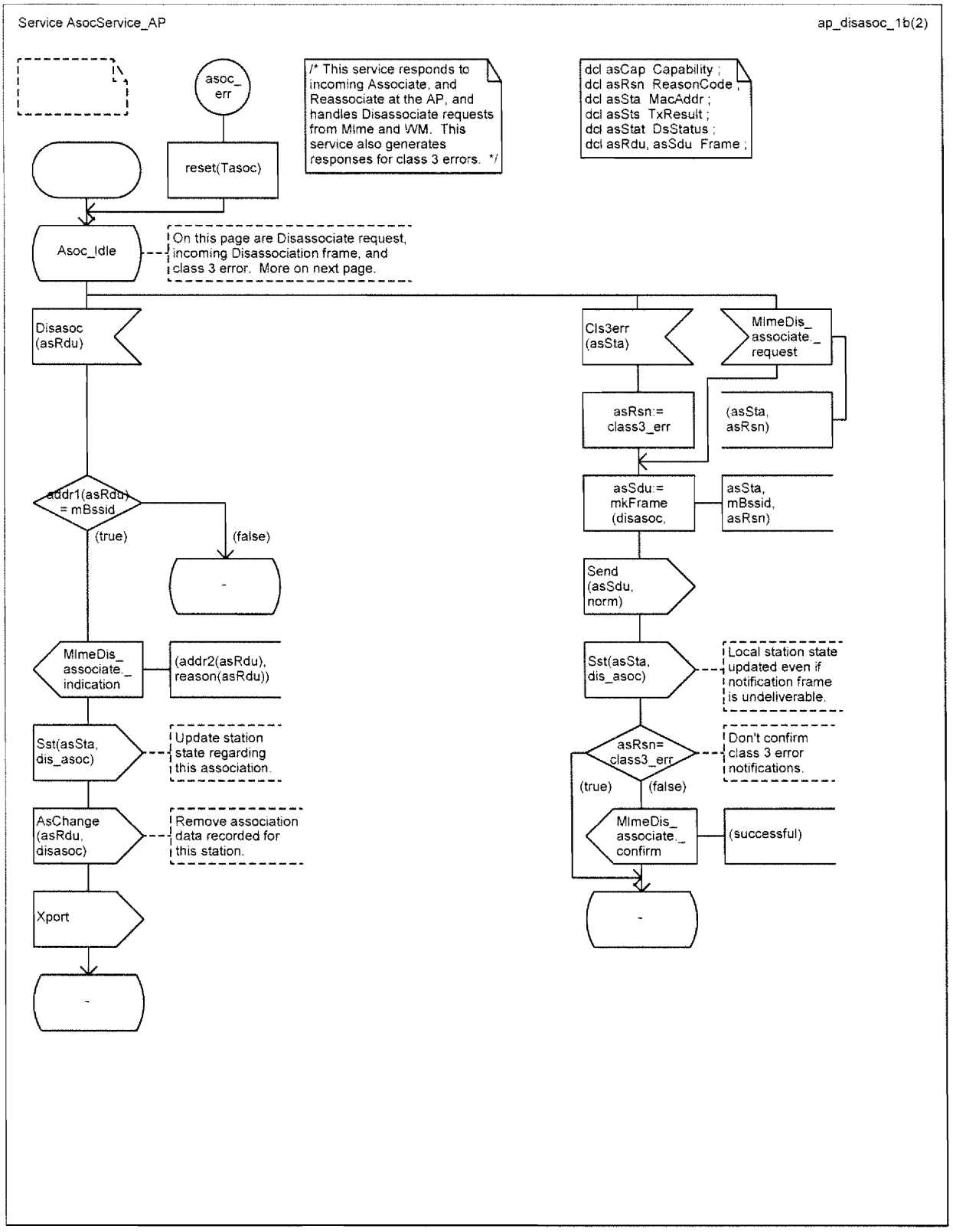


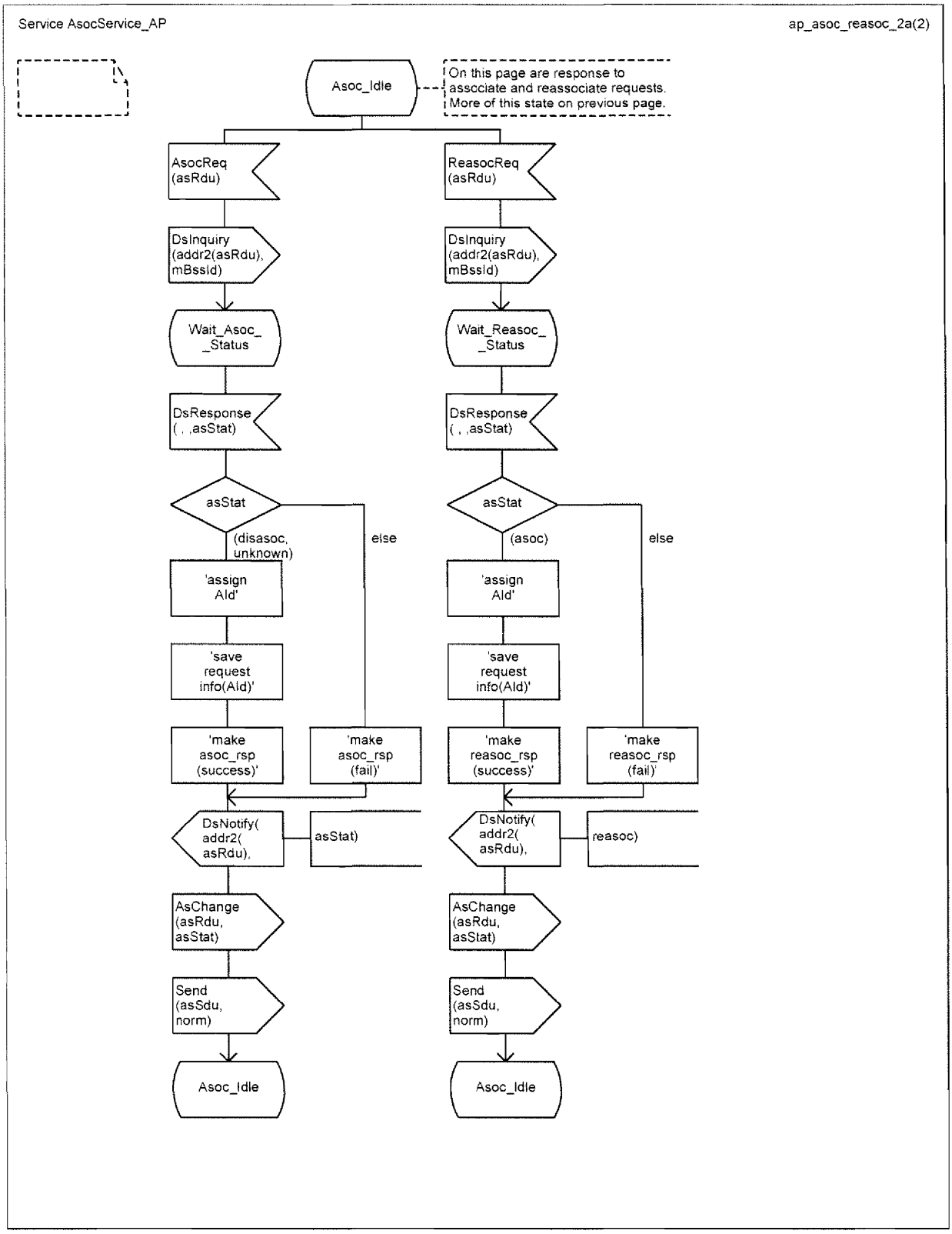


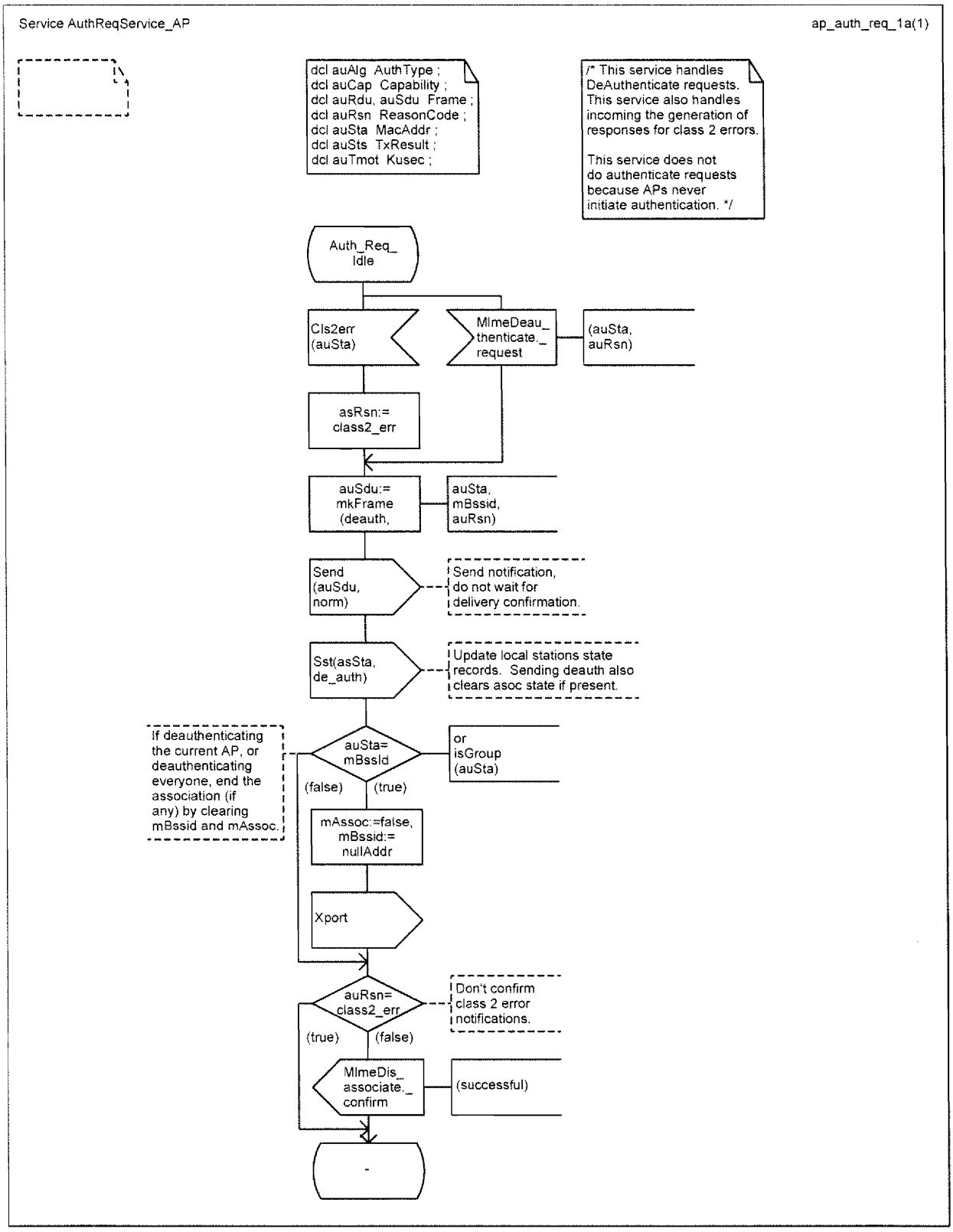


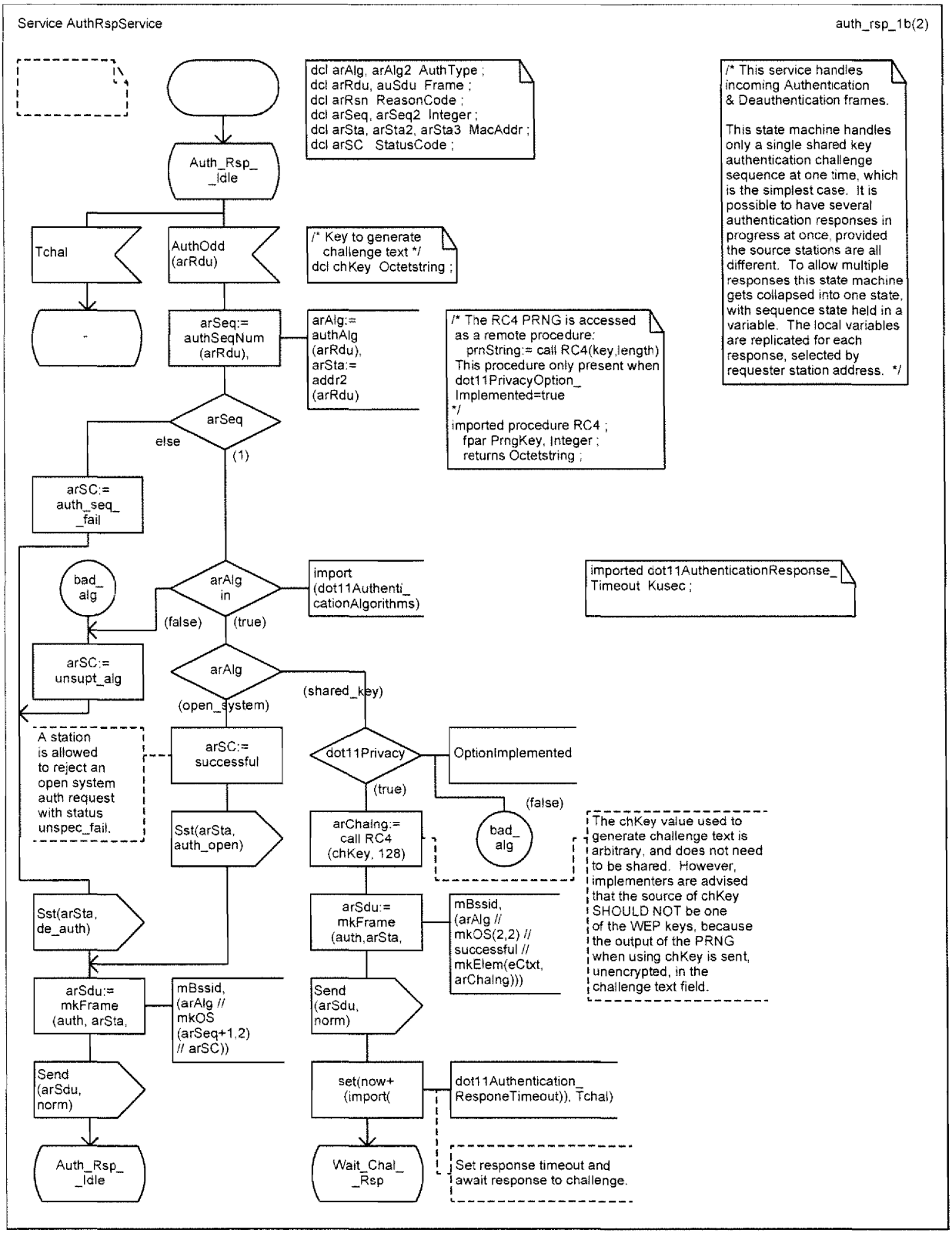


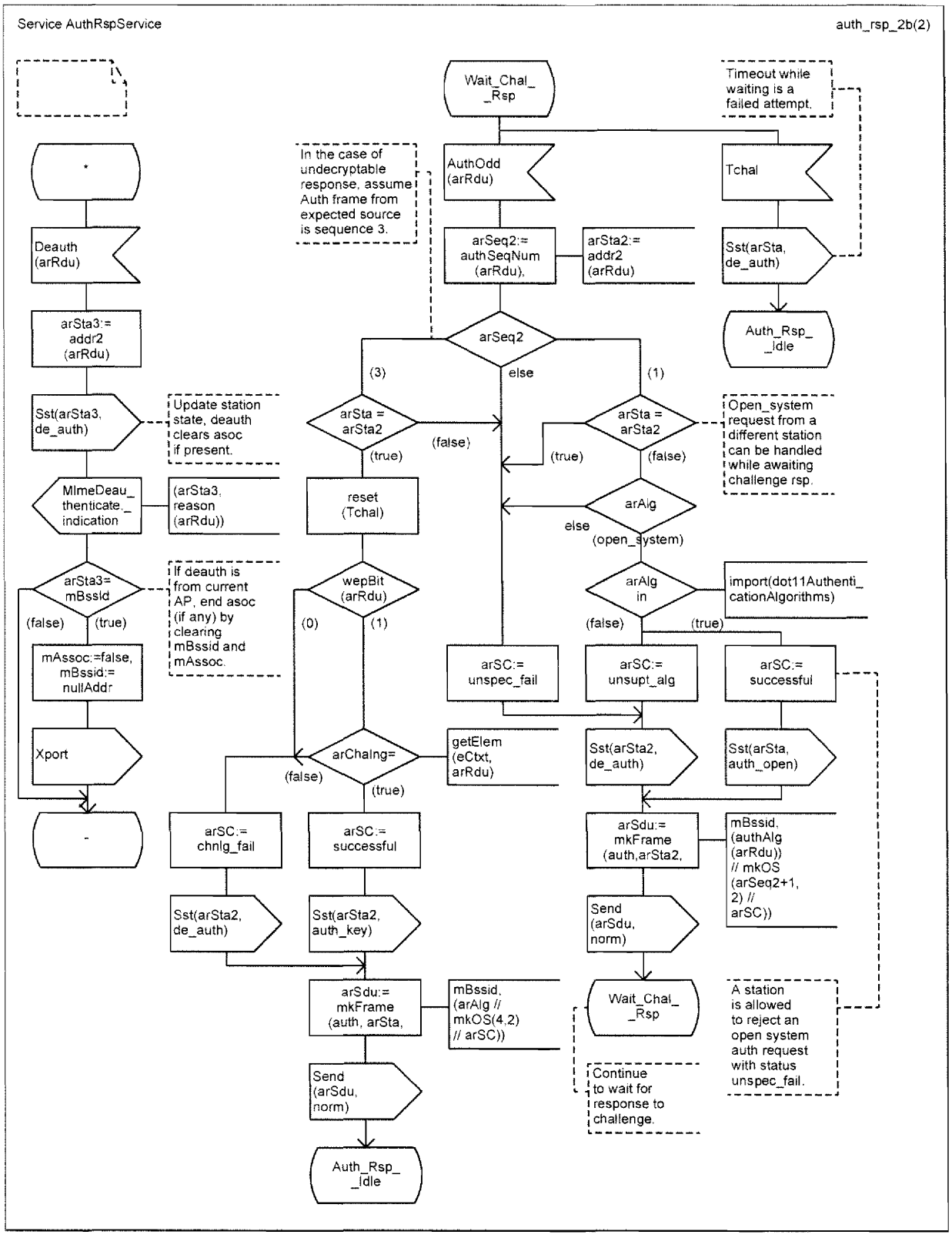


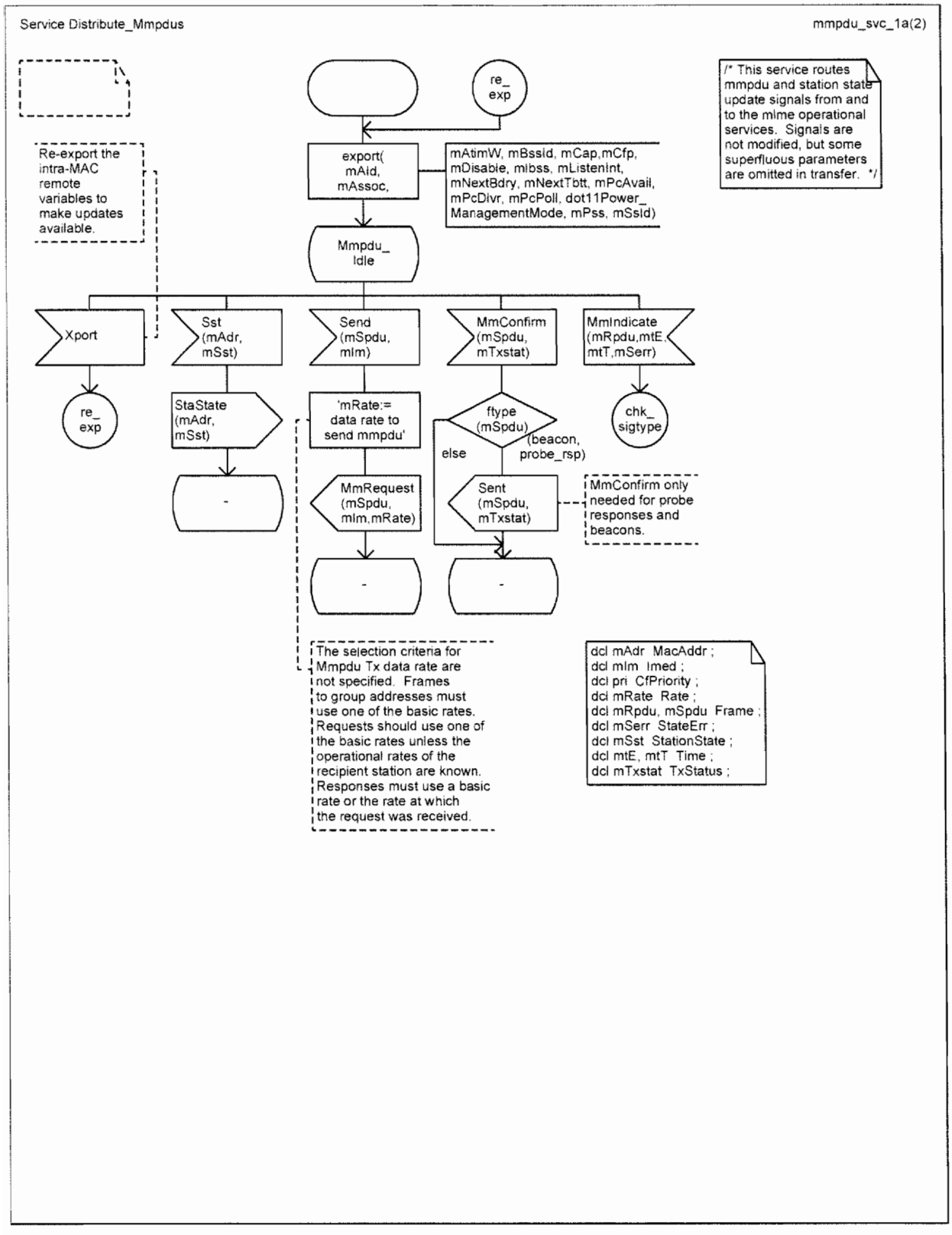


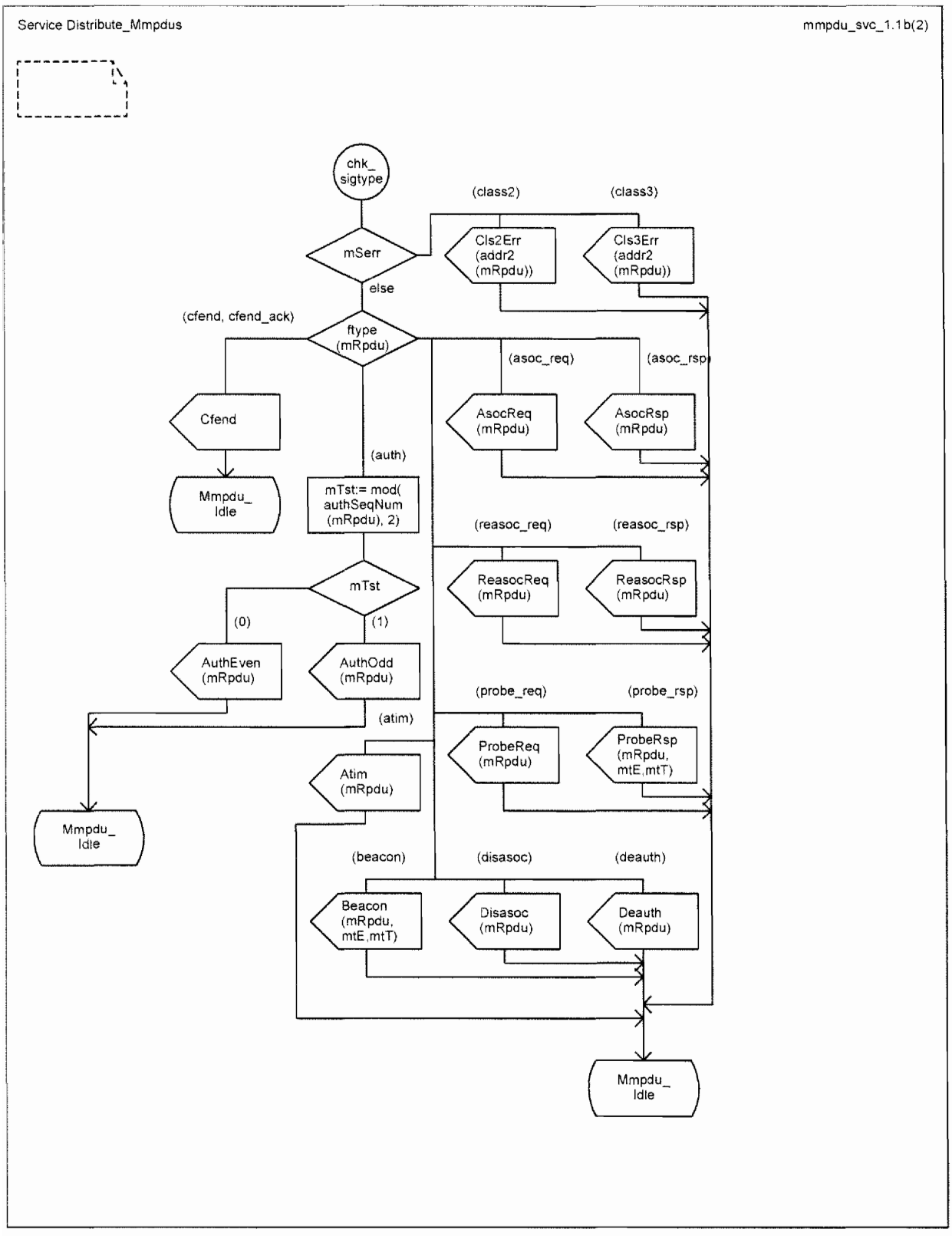


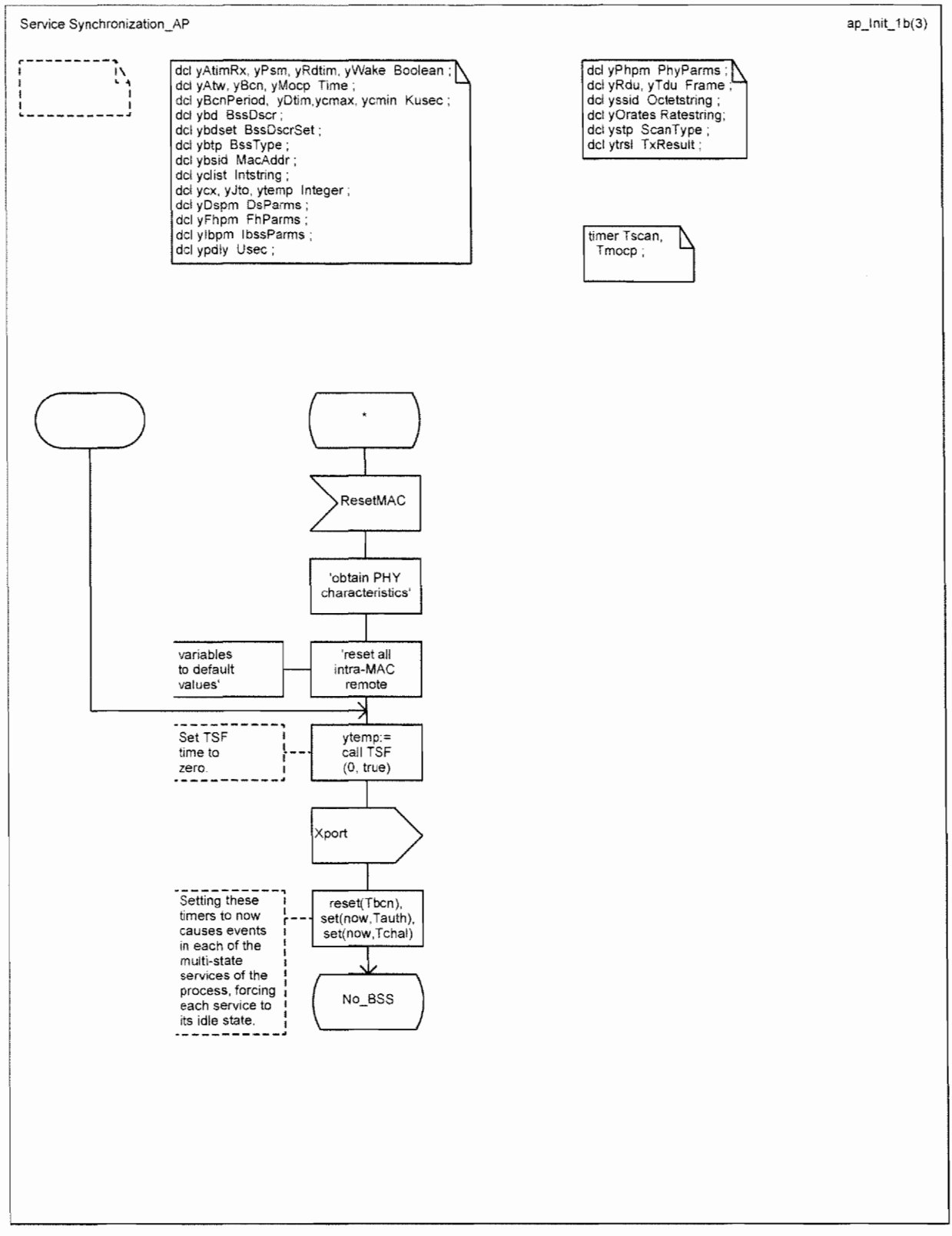


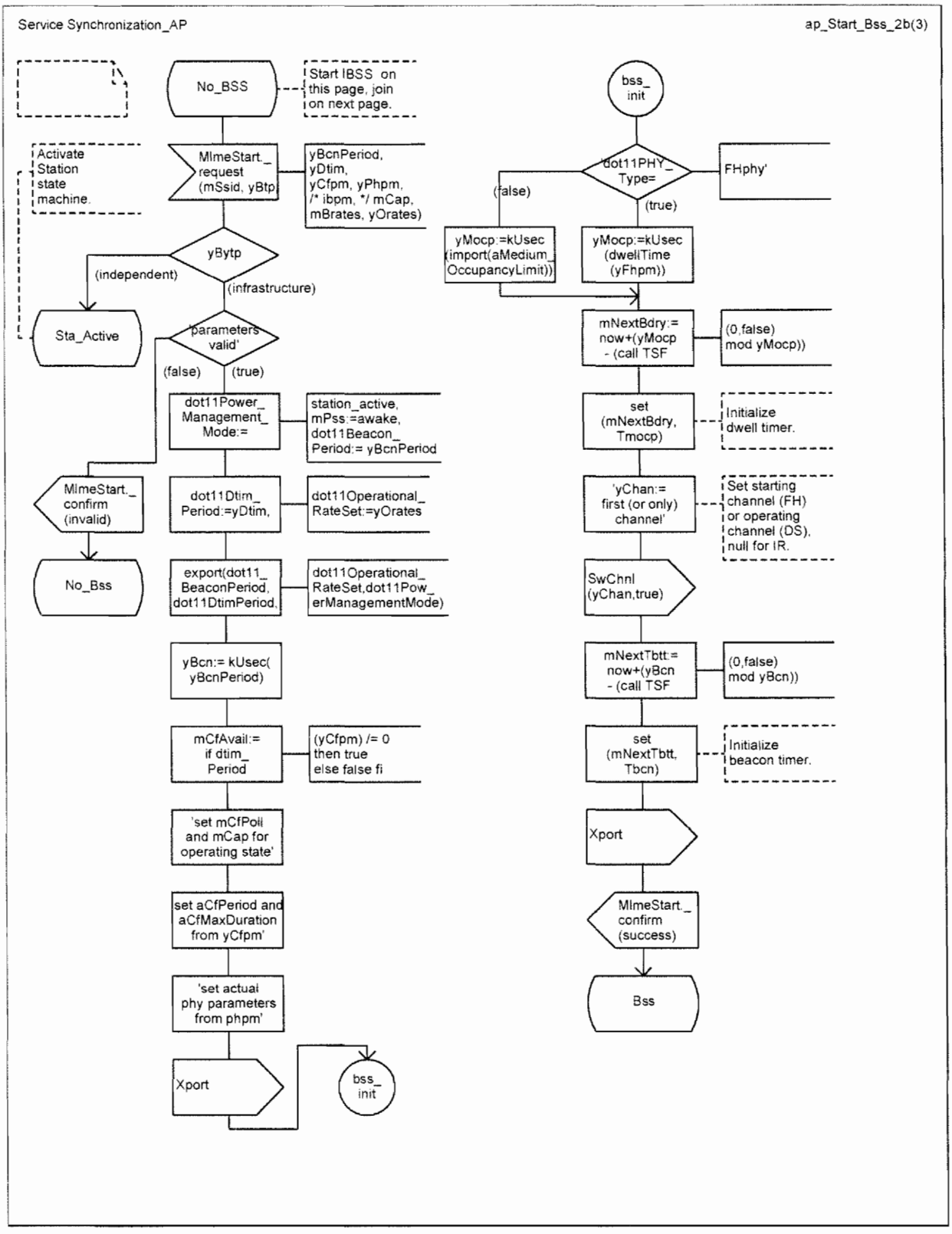


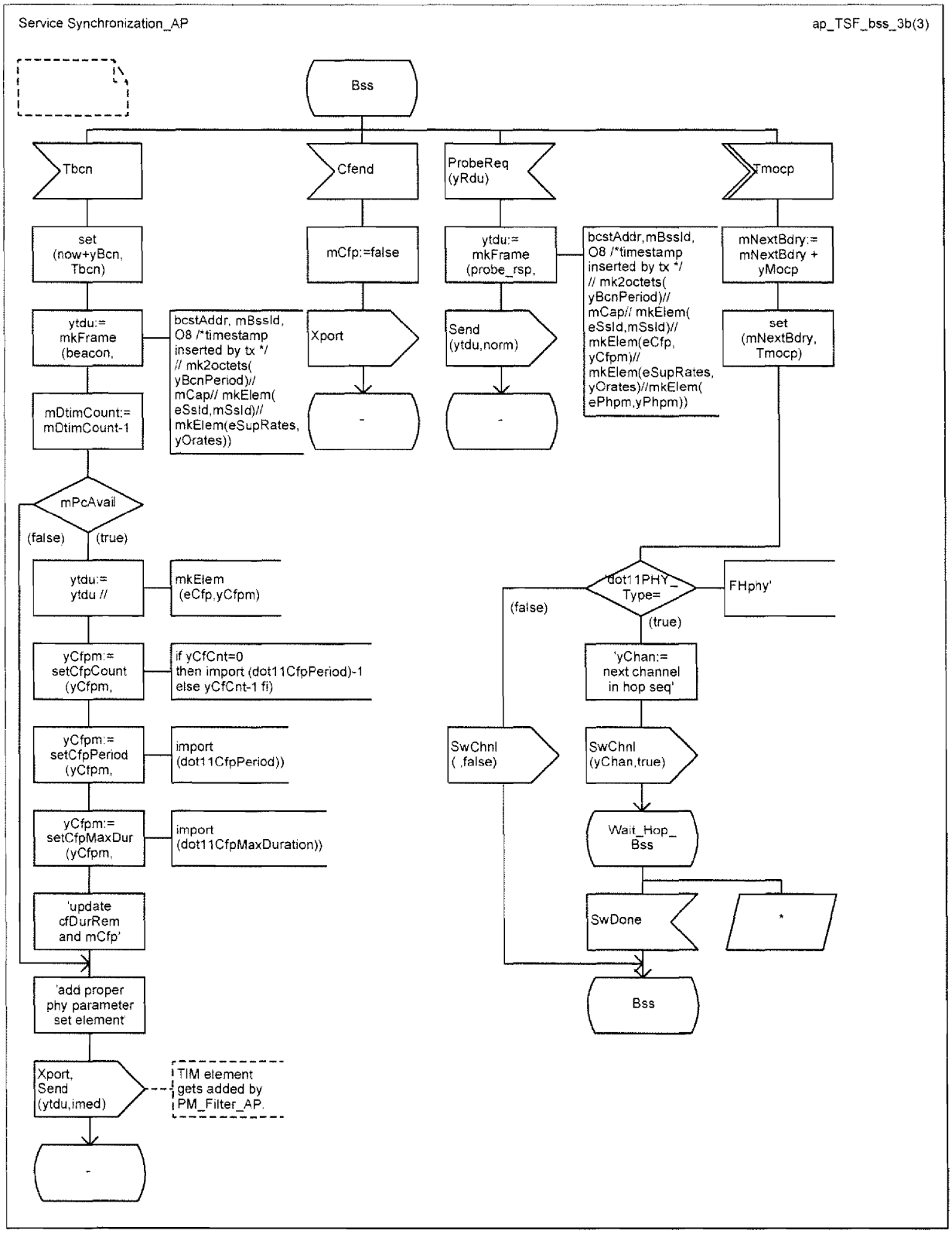


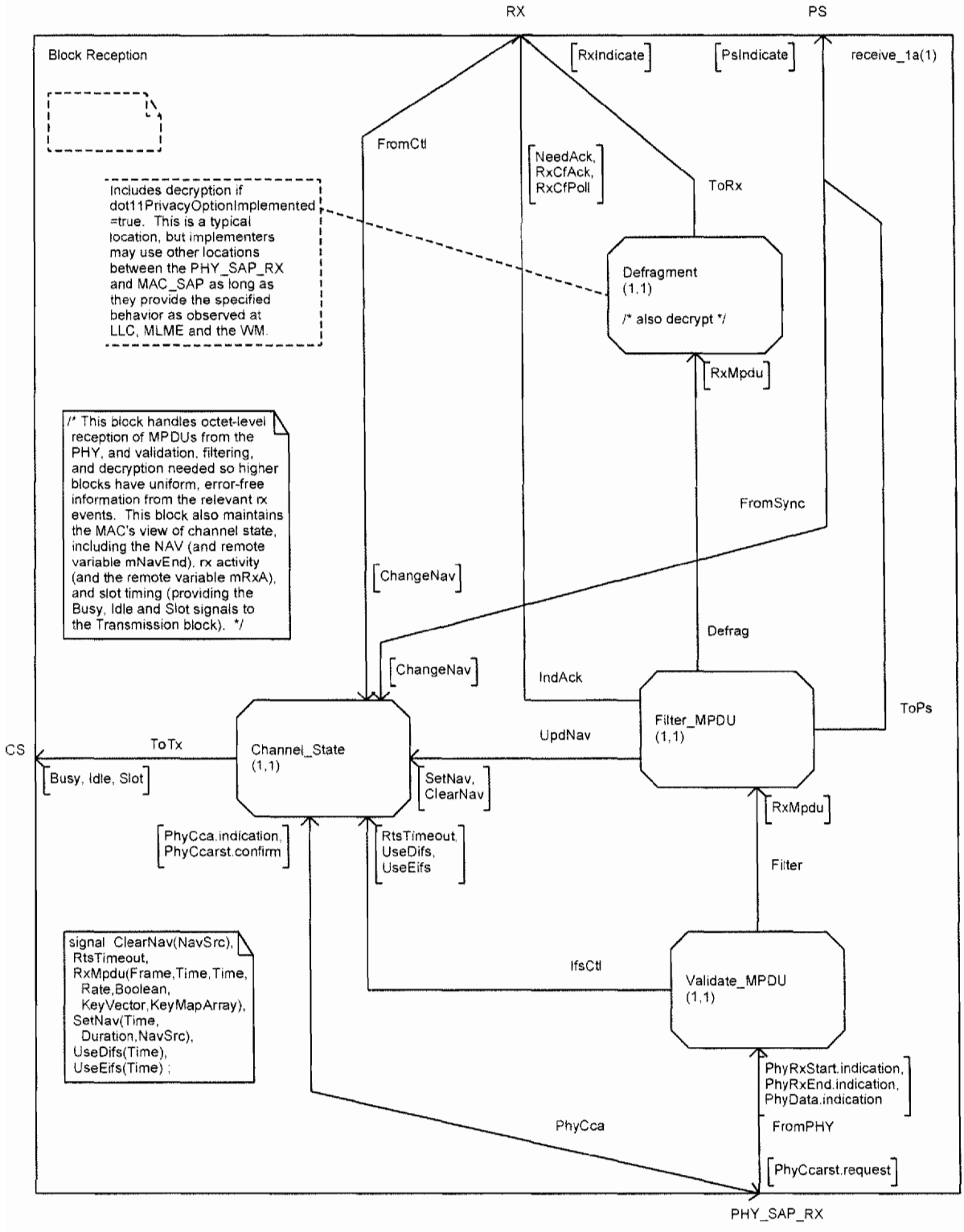


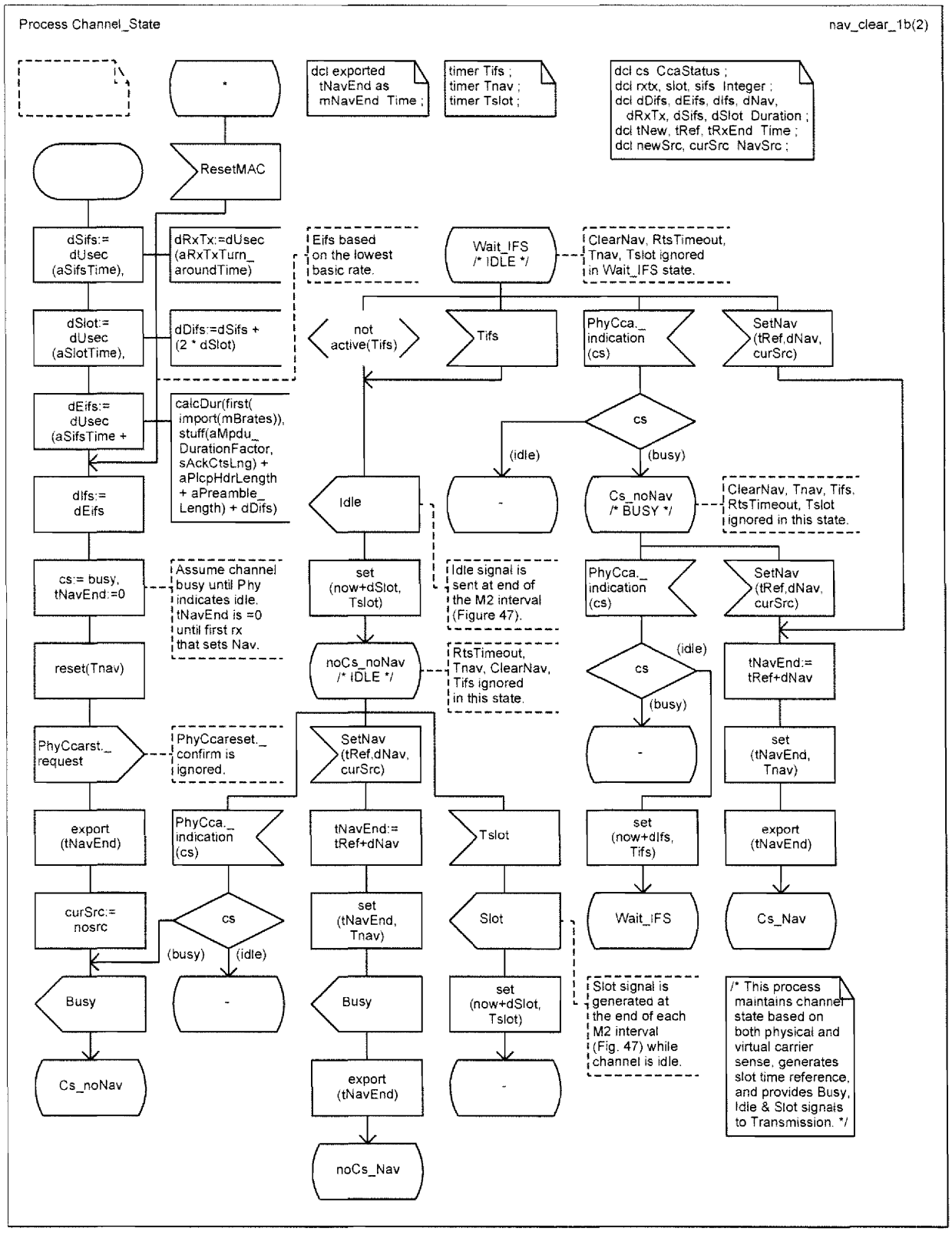


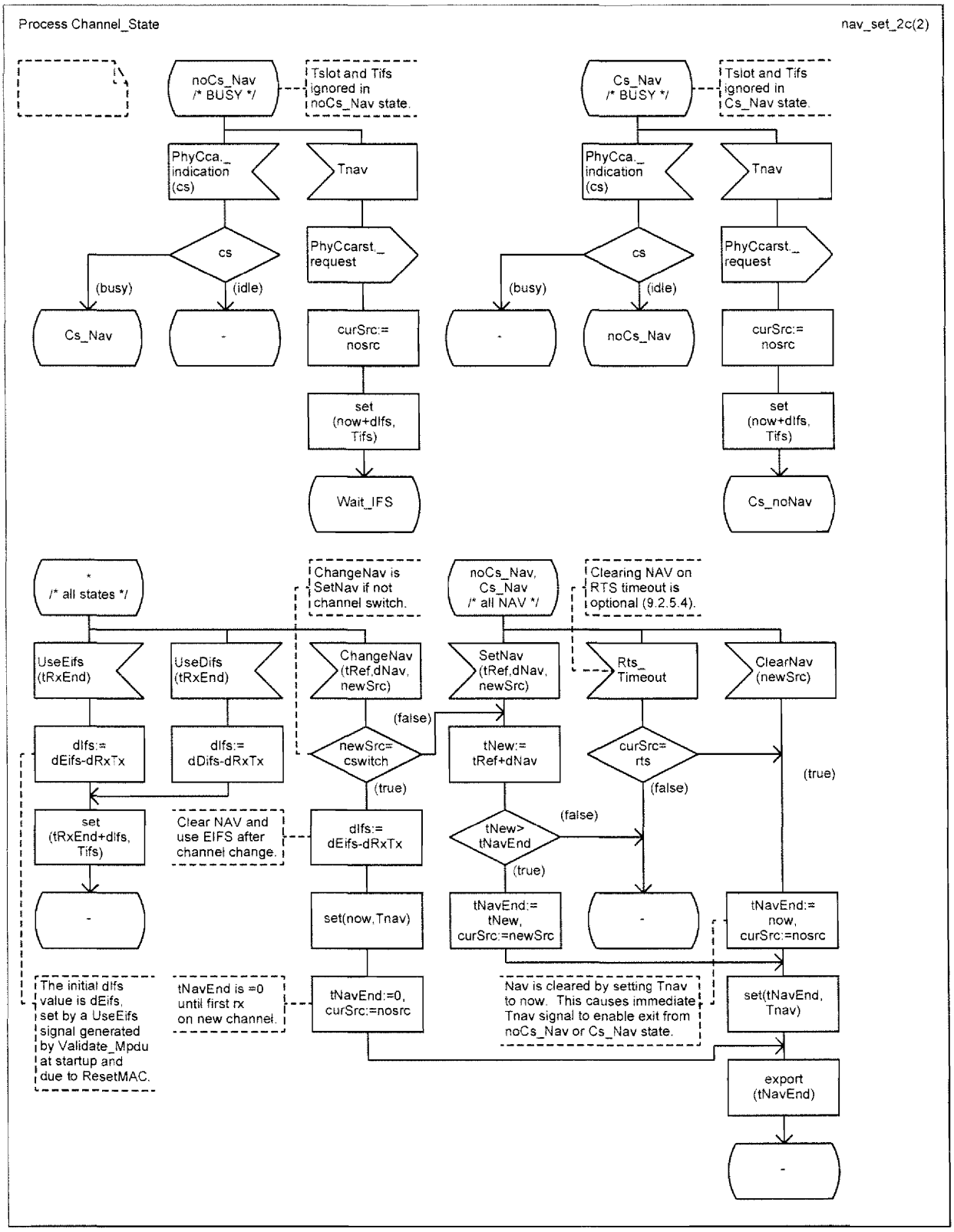


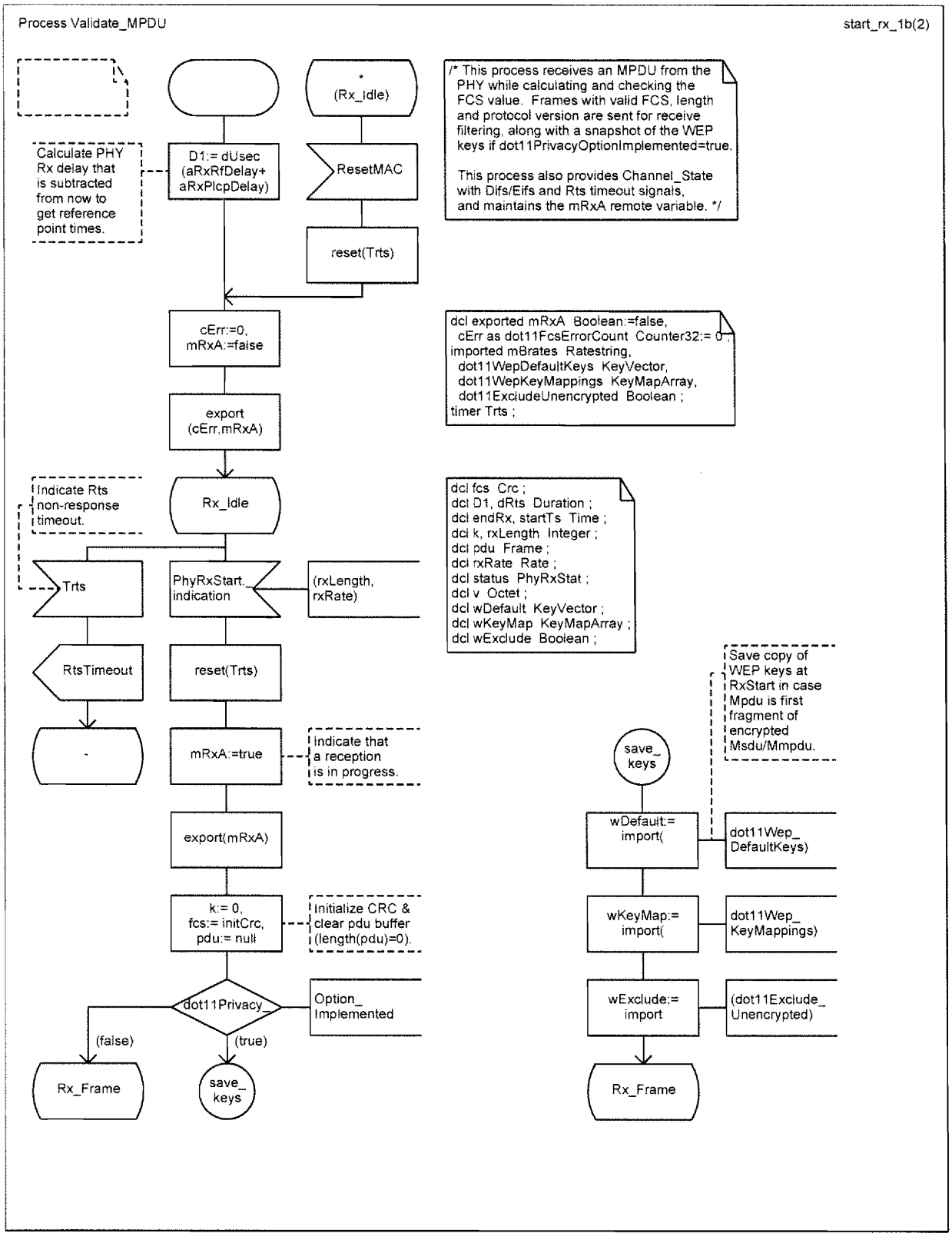


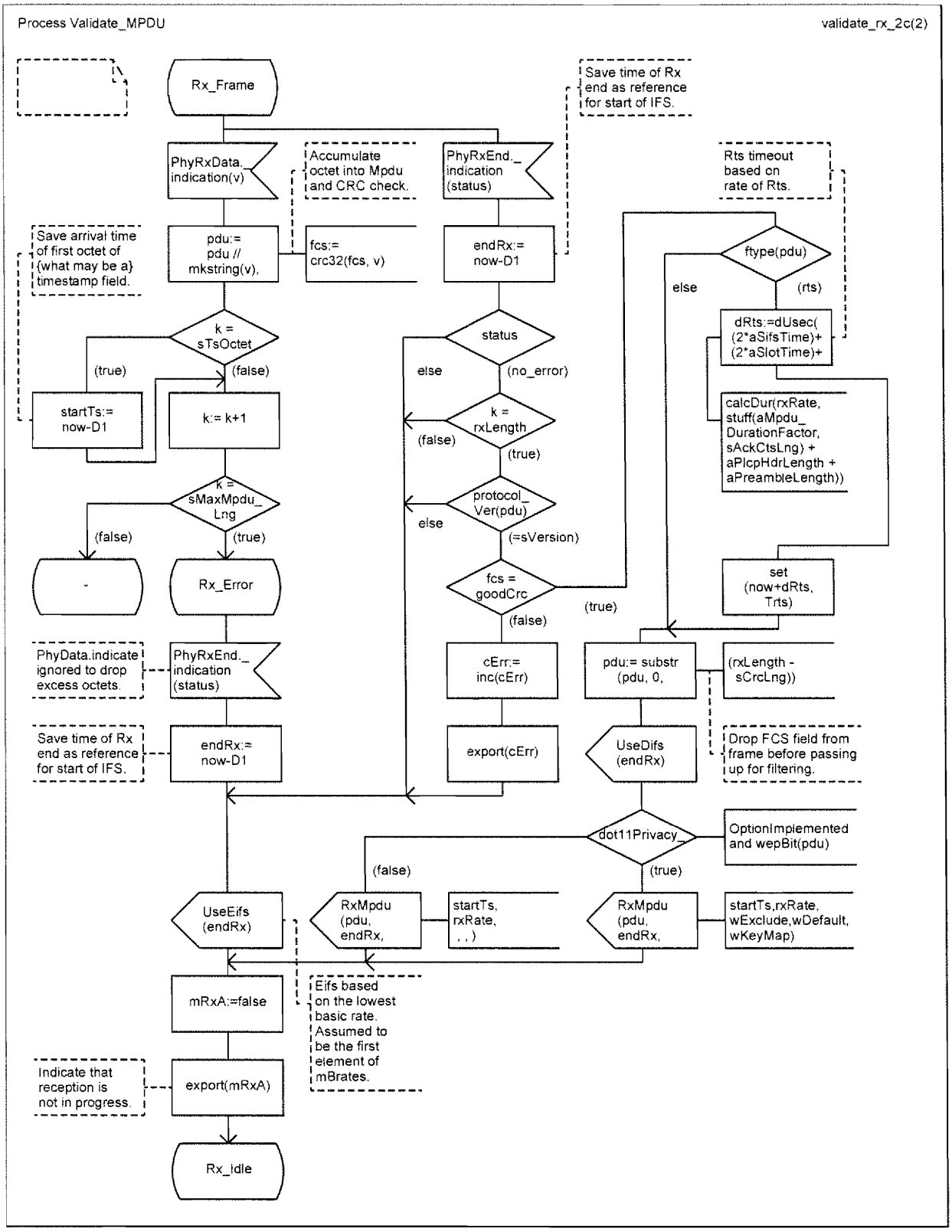


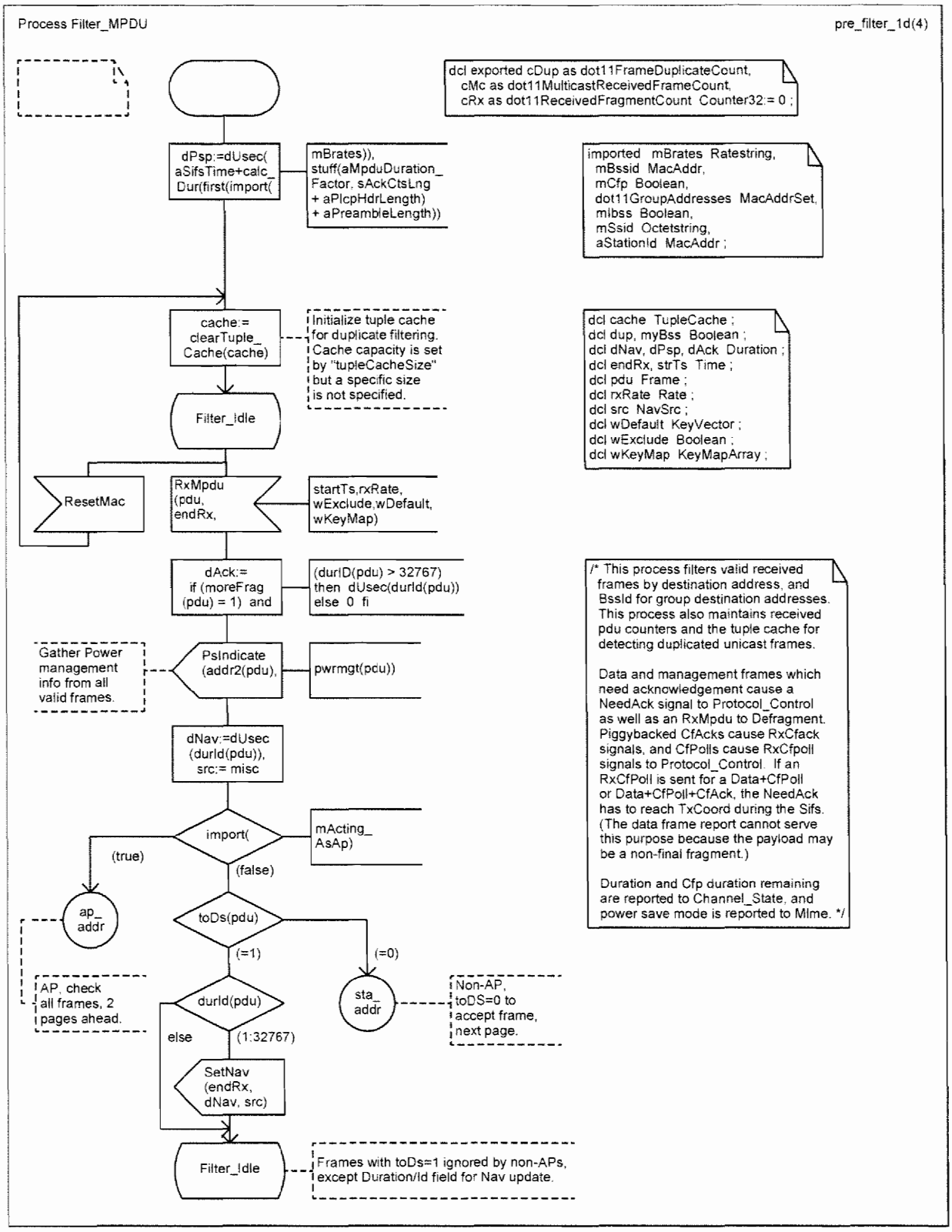


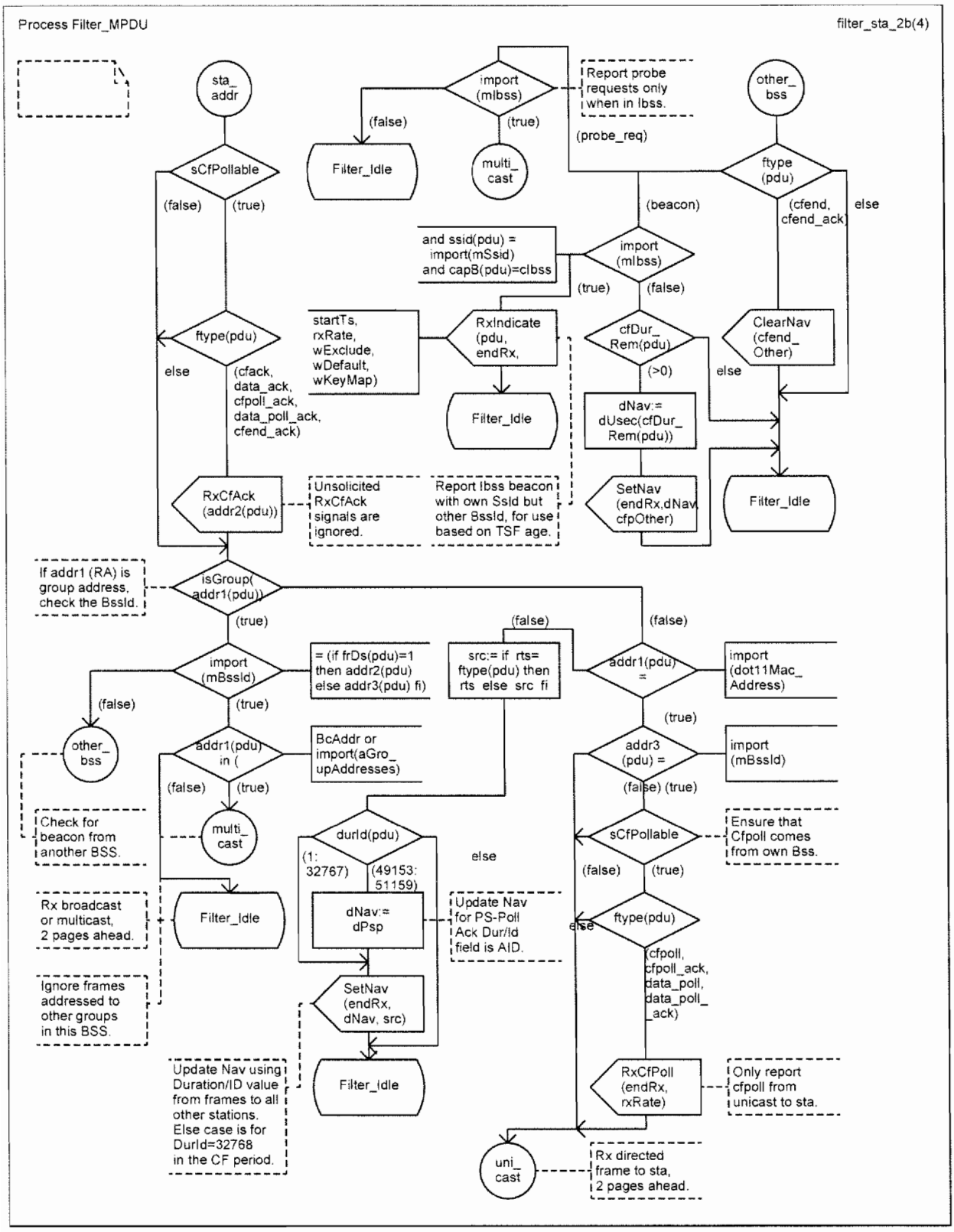


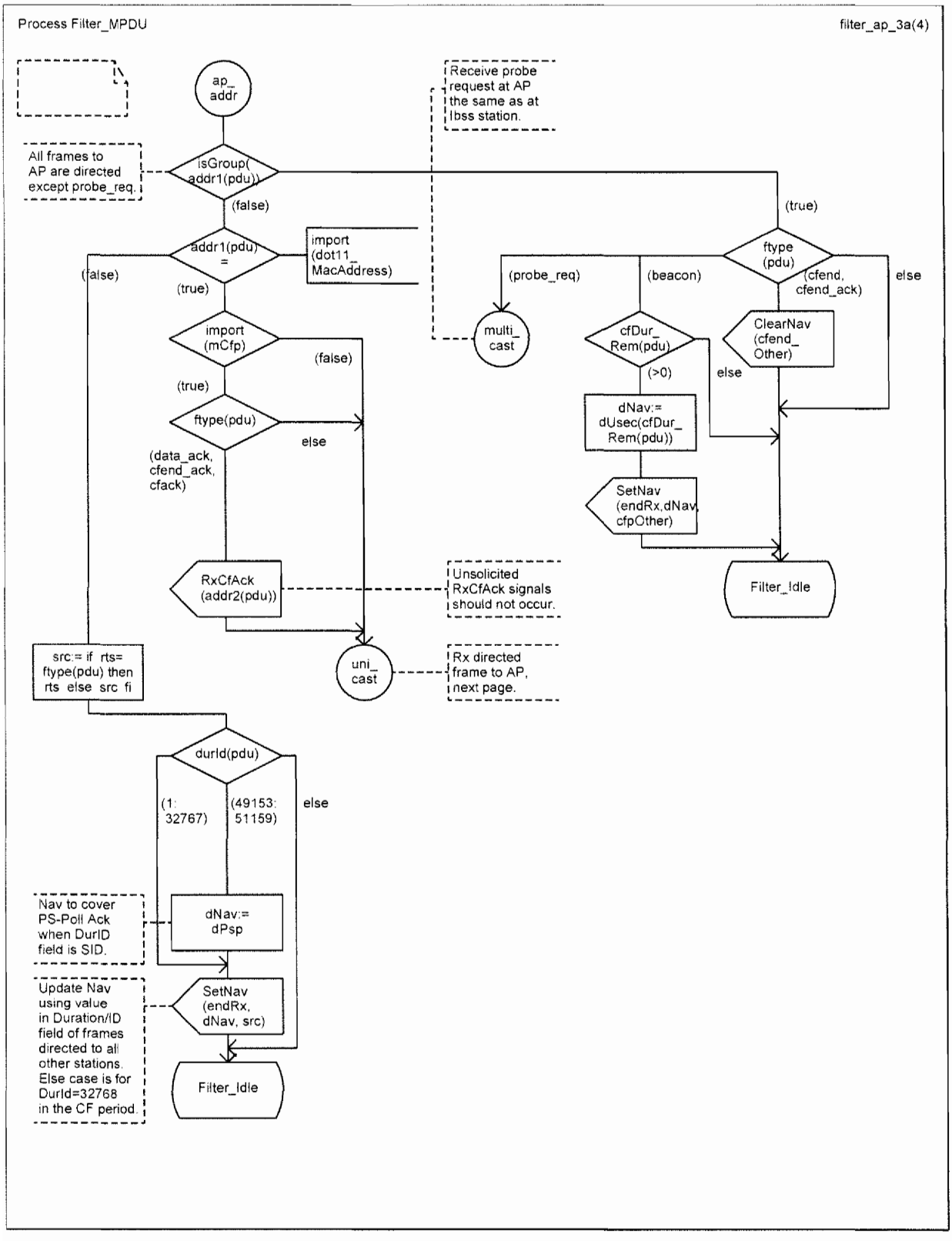


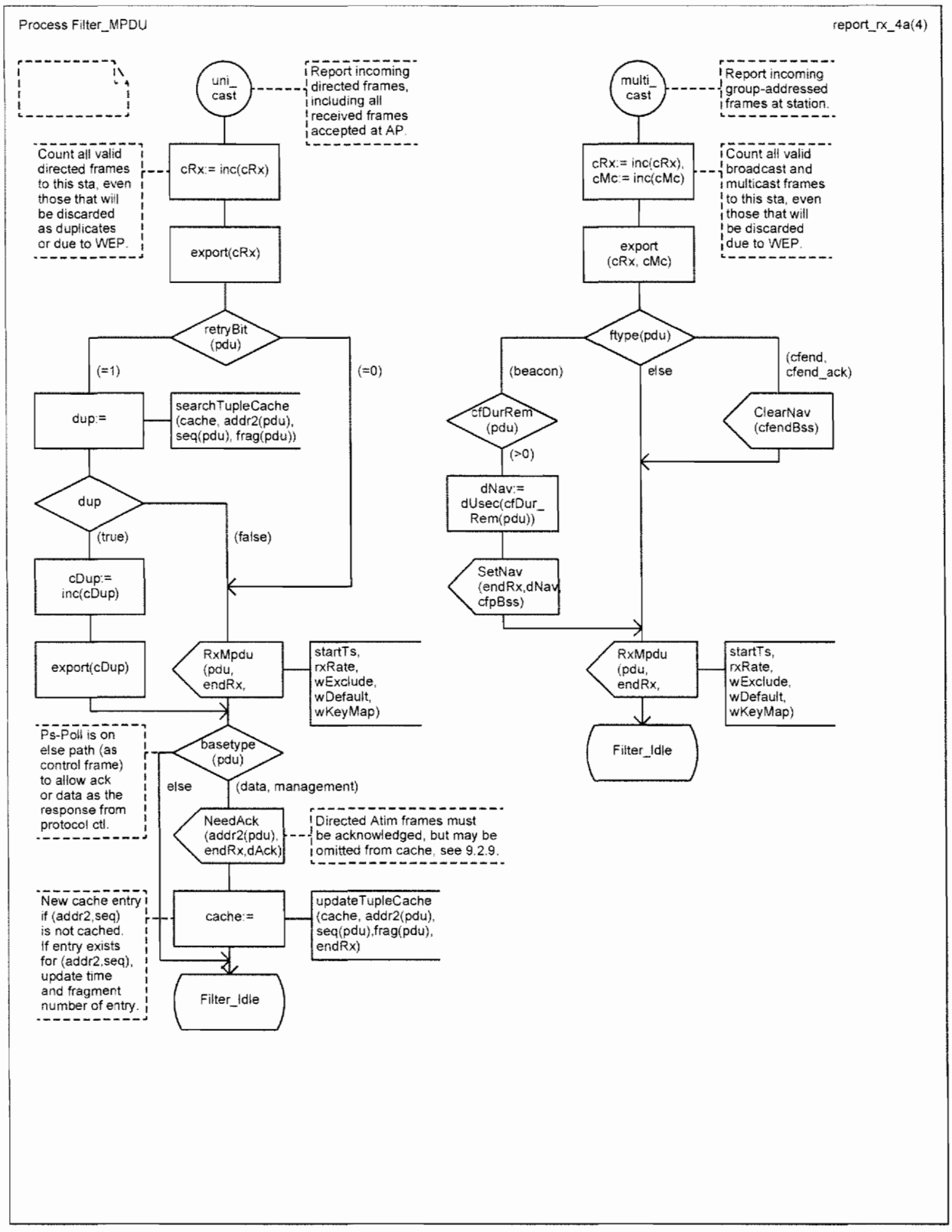


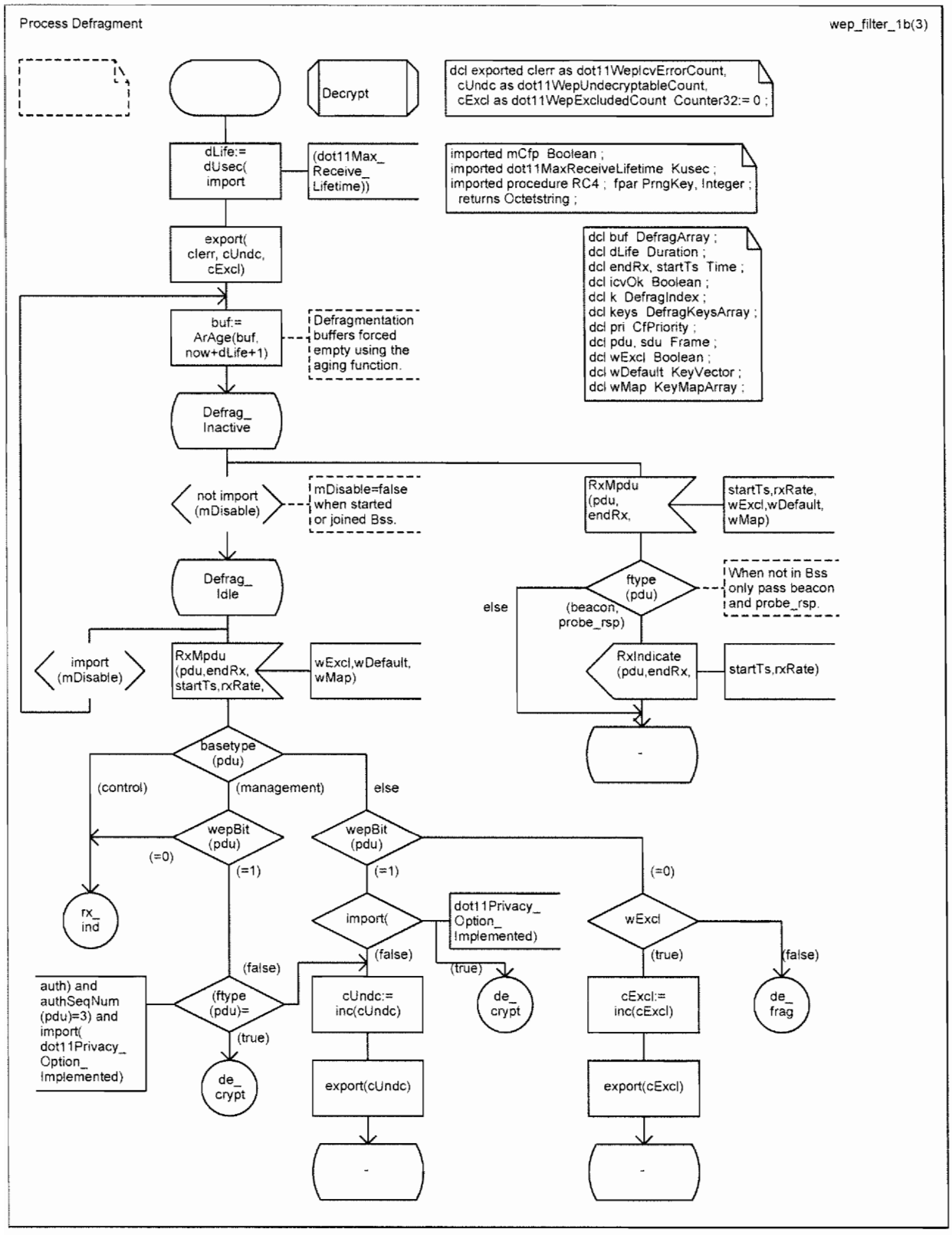


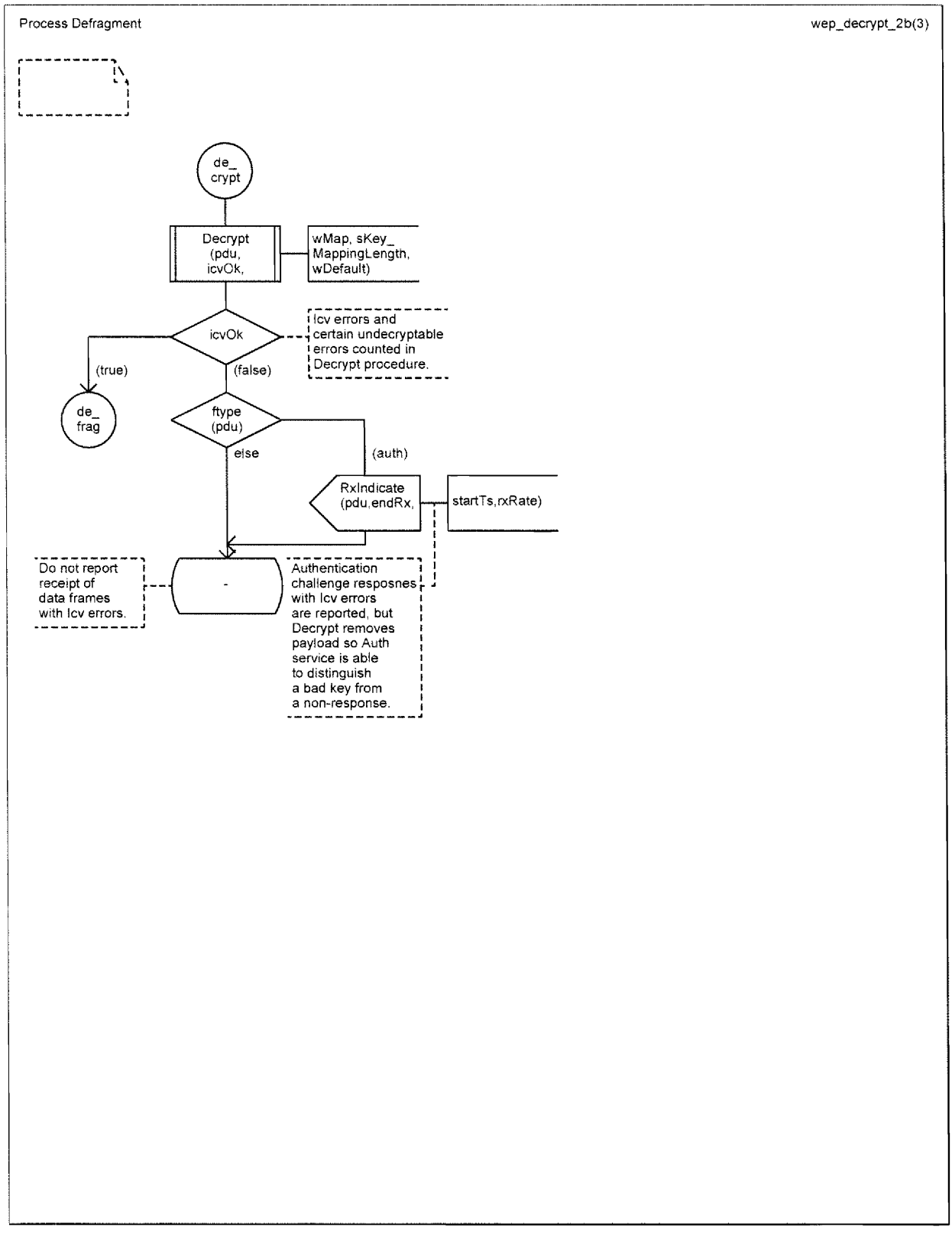


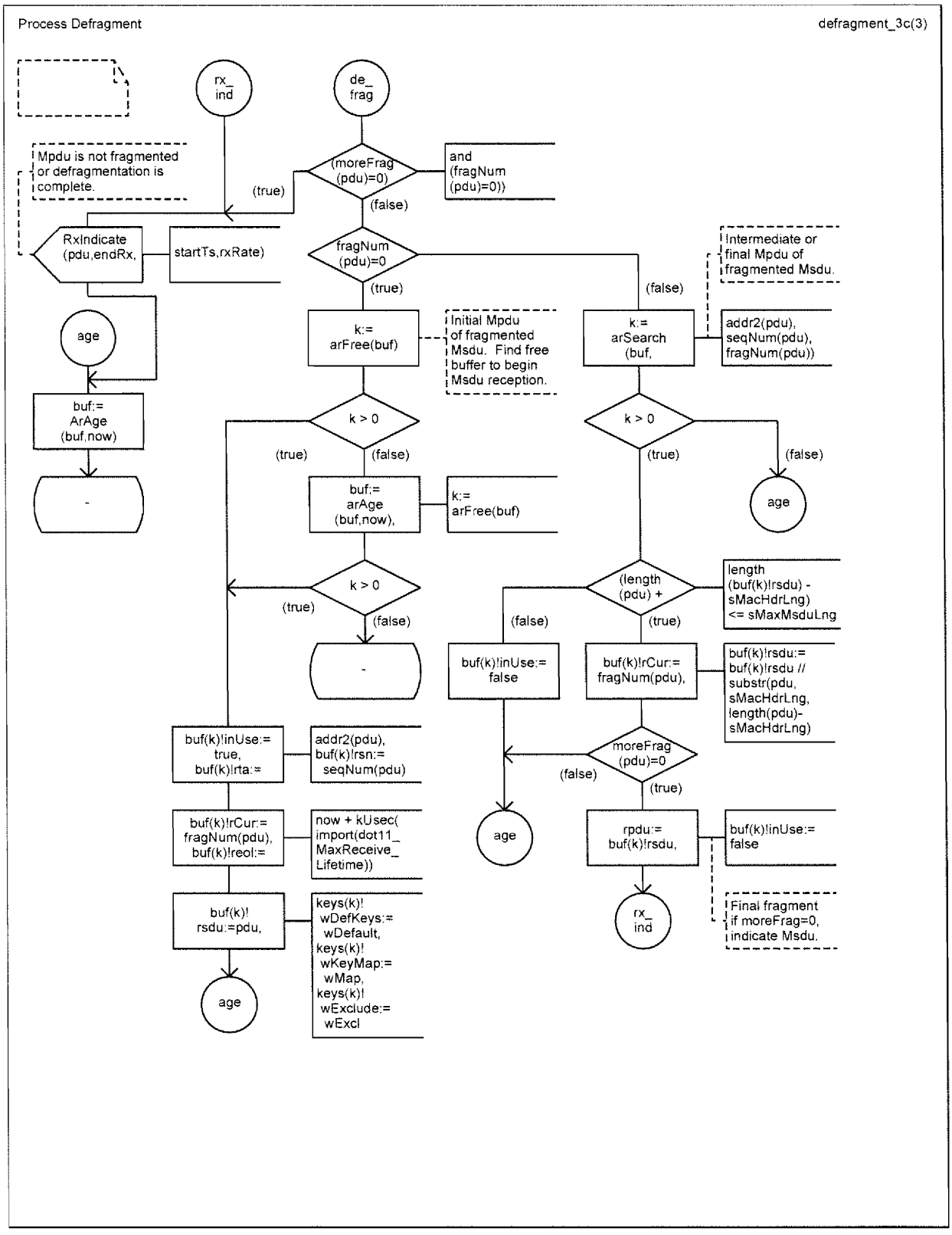


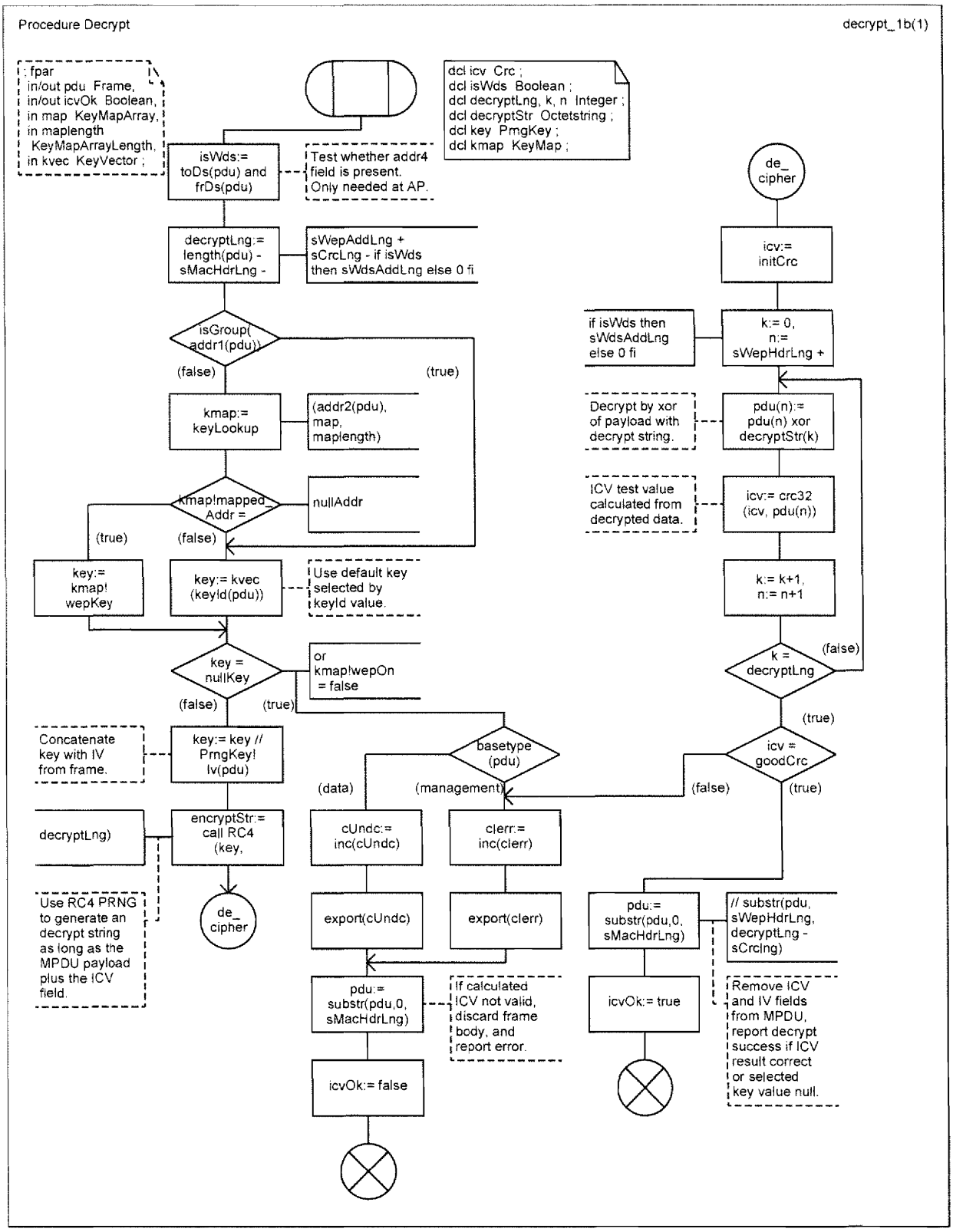












Annex D

(normative)

ASN.1 encoding of the MAC and PHY MIB

```
-- ****************************************************************
-- * IEEE 802.11 Management Information Base
-- ****************************************************************
IEEE802dot11-MIB DEFINITIONS ::= BEGIN
    IMPORTS
        MODULE-IDENTITY, OBJECT-TYPE,
        NOTIFICATION-TYPE,Integer32, Counter32  FROM SNMPv2-SMI
        DisplayString , MacAddress, RowStatus,
        TruthValue                              FROM SNMPv2-TC
        MODULE-COMPLIANCE, OBJECT-GROUP,
        NOTIFICATION-GROUP                      FROM SNMPv2-CONF
        ifIndex                                 FROM RFC1213-MIB;

-- ****************************************************************
-- *  MODULE IDENTITY
-- ****************************************************************
    ieee802dot11 MODULE-IDENTITY
    LAST-UPDATED "9807080000Z"
    ORGANIZATION "IEEE 802.11"
    CONTACT-INFO
           "WG E-mail: stds-802-11@ieee.org

                Chair: Vic Hayes
              Postal: Lucent Technologies, Inc.
                      Zadelstede 1-10
                      Nieuwegein, Netherlands
                      3431 JZ
                 Tel: +31 30 609 7528
                 Fax: +31 30 231 6233
              E-mail: vichayes@lucent.com

              Editor: Bob O'Hara
              Postal: Informed Technology, Inc.
                      1750 Nantucket Circle, Suite 138
                      Santa Clara, CA 95054 USA
                 Tel: +1 408 986 9596
                 Fax: +1 408 727 2654
              E-mail: bob@informed-technology.com"
    DESCRIPTION
        "The MIB module for IEEE 802.11 entities.
        iso(1).member-body(2).us(840).ieee802dot11(10036)"
    ::= { 1 2 840 10036 }

-- ****************************************************************
-- *  Major sections
-- ****************************************************************
--  Station ManagemenT (SMT) Attributes
    --  DEFINED AS "The SMT object class provides the necessary support at the
    --  station to manage the processes in the station such that the
    --  station may work cooperatively as a part of an IEEE 802.11 network.";
```

```
    dot11smt OBJECT IDENTIFIER                    ::= {ieee802dot11 1}

      -- dot11smt GROUPS
      -- dot11StationConfigTable                  ::= {dot11smt 1}
      -- dot11AuthenticationAlgorithmsTable       ::= {dot11smt 2}
      -- dot11WEPDefaultKeysTable                 ::= {dot11smt 3}
      -- dot11WEPKeyMappingsTable                 ::= {dot11smt 4}
      -- dot11PrivacyTable                        ::= {dot11smt 5}
      -- dot11SMTnotification                     ::= {dot11smt 6}

--  MAC Attributes
    -- DEFINED AS "The MAC object class provides the necessary support
    -- for the access control, generation, and verification of frame check
    -- sequences, and proper delivery of valid data to upper layers.";

    dot11mac OBJECT IDENTIFIER                    ::= {ieee802dot11 2}

      -- MAC GROUPS
      -- reference IEEE Std 802.1F-1993
         -- dot11OperationTable                   ::= {dot11mac 1}
         -- dot11CountersTable                    ::= {dot11mac 2}
         -- dot11GroupAddressesTable              ::= {dot11mac 3}

--  Resource Type ID
      dot11res OBJECT IDENTIFIER                  ::= {ieee802dot11 3}
      dot11resAttribute OBJECT IDENTIFIER         ::= {dot11res 1 }

--  PHY Attributes
    -- DEFINED AS "The PHY object class provides the necessary support
    -- for required PHY operational information that may vary from PHY
    -- to PHY and from STA to STA to be communicated to upper layers."

   dot11phy OBJECT IDENTIFIER                     ::= {ieee802dot11 4}

    -- phy GROUPS
    -- dot11PhyOperationTable                     ::= {dot11phy 1}
    -- dot11PhyAntennaTable                       ::= {dot11phy 2}
    -- dot11PhyTxPowerTable                       ::= {dot11phy 3}
    -- dot11PhyFHSSTable                          ::= {dot11phy 4}
    -- dot11PhyDSSSTable                          ::= {dot11phy 5}
    -- dot11PhyIRTable                            ::= {dot11phy 6}
    -- dot11RegDomainsSupportedTable              ::= {dot11phy 7}
    -- dot11AntennasListTable                     ::= {dot11phy 8}
    -- dot11SupportedDataRatesTxTable             ::= {dot11phy 9}
    -- dot11SupportedDataRatesRxTable             ::= {dot11phy 10}


-- ****************************************************************
-- *  Textual conventions from 802 definitions
-- ****************************************************************
    WEPKeytype ::= OCTET STRING (SIZE (5))

-- ****************************************************************
-- *  MIB attribute OBJECT-TYPE definitions follow
-- ****************************************************************

-- ****************************************************************
-- *  SMT Station Config  Table
-- ****************************************************************
dot11StationConfigTable OBJECT-TYPE
```

```
        SYNTAX SEQUENCE OF Dot11StationConfigEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Station Configuration attributes.  In tablular form to
            allow for multiple instances on an agent."
    ::= {  dot11smt 1 }

dot11StationConfigEntry OBJECT-TYPE
        SYNTAX Dot11StationConfigEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "An entry in the dot11StationConfigTable.  It is
            possible for there to be multiple IEEE 802.11 interfaces
            on one agent, each with its unique MAC address. The
            relationship between an IEEE 802.11 interface and an
            interface in the context of the Internet-standard MIB is
            one-to-one.  As such, the value of an ifIndex object
            instance can be directly used to identify corresponding
            instances of the objects defined herein.

            ifIndex - Each 802.11 interface is represented by an
            ifEntry.  Interface tables in this MIB module are indexed
            by ifIndex."


        INDEX {ifIndex}
    ::= {  dot11StationConfigTable 1 }

Dot11StationConfigEntry ::=
        SEQUENCE {
         dot11StationID                                MacAddress,
             dot11MediumOccupancyLimit                 INTEGER,
             dot11CFPollable                           TruthValue,
             dot11CFPPeriod                            INTEGER,
             dot11CFPMaxDuration                       INTEGER,
             dot11AuthenticationResponseTimeOut        INTEGER,
             dot11PrivacyOptionImplemented             TruthValue,
               dot11PowerManagementMode                INTEGER,
               dot11DesiredSSID                        OCTET STRING,
               dot11DesiredBSSType                     INTEGER,
               dot11OperationalRateSet                 OCTET STRING,
               dot11BeaconPeriod                       INTEGER,
               dot11DTIMPeriod                         INTEGER,
               dot11AssociationResponseTimeOut         INTEGER,
             dot11DisassociateReason                   INTEGER,
             dot11DisassociateStation                  MacAddress,
             dot11DeauthenticateReason                 INTEGER,
             dot11DeauthenticateStation                MacAddress,
             dot11AuthenticateFailStatus               INTEGER,
             dot11AuthenticateFailStation              MacAddress }

dot11StationID OBJECT-TYPE
        SYNTAX MacAddress
        MAX-ACCESS read-write
        STATUS deprecated
        DESCRIPTION
            "The purpose of dot11StationID is to allow a manager to identify
```

```
            a station for its own purposes.  This attribute provides
            for that eventuality while keeping the true MAC address
            independent.  Its syntax is MacAddress.  The default value
            is the station's assigned, unique MAC address."

    ::= { dot11StationConfigEntry 1 }

dot11MediumOccupancyLimit OBJECT-TYPE
        SYNTAX INTEGER (0..1000)
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "This attribute shall indicate the maximum amount of time,
            in TU, that a point coordinator may control the usage of
            the wireless medium without relinquishing control for long
            enough to allow at least one instance of DCF access to the
            medium.  The default value of this attribute shall be 100,
            and the maximum value shall be 1000."

    ::= { dot11StationConfigEntry 2 }

dot11CFPollable OBJECT-TYPE
        SYNTAX TruthValue
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
              "When this attribute is true, it shall indicate that the STA
              is able to respond to a CF-Poll with a data frame within a
              SIFS time. This attribute shall be false if the STA is not
              able to respond to a CF-Poll with a data frame within a SIFS
              time."

    ::= { dot11StationConfigEntry 3 }

dot11CFPPeriod OBJECT-TYPE
        SYNTAX INTEGER (0..255)
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "The attribute shall describe the number of DTIM intervals
            between the start of CFPs.  It is modified by
            MLME-START.request primitive."

::= {  dot11StationConfigEntry 4 }

dot11CFPMaxDuration OBJECT-TYPE
        SYNTAX INTEGER (0..65535)
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
              "The attribute shall describe the maximum duration of the CFP
              in TU that may be generated by the PCF. It is modified by
              MLME-START.request primitive."

::= {  dot11StationConfigEntry 5 }

dot11AuthenticationResponseTimeOut OBJECT-TYPE
        SYNTAX INTEGER (1..4294967295)
        MAX-ACCESS read-write
        STATUS current
```

```
        DESCRIPTION
            "This attribute shall specify the number of TUs that a
            responding STA should wait for the next frame in the
            authentication sequence."

    ::= {  dot11StationConfigEntry 6 }

dot11PrivacyOptionImplemented OBJECT-TYPE
        SYNTAX TruthValue
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "This attribute, when true, shall indicate that the IEEE
            802.11 WEP option is implemented.  The default value of
            this attribute shall be false."

    ::= {  dot11StationConfigEntry 7 }

dot11PowerManagementMode OBJECT-TYPE
       SYNTAX INTEGER { active(1), powersave(2) }
       MAX-ACCESS read-write
       STATUS current
       DESCRIPTION
              "This attribute shall specify the power management
              mode of the STA. When set to active, it shall indicate
              that the station is not in power-save mode. When set
              to powersave, it shall indicate that the station is
              in power-save mode. The power management mode is
              transmitted in all frames according to the rules
              in  7.1.3.1.7."
    ::= { dot11StationConfigEntry 8 }

dot11DesiredSSID OBJECT-TYPE
       SYNTAX OCTET STRING (SIZE(0..32))
       MAX-ACCESS read-write
       STATUS current
       DESCRIPTION
              "This attribute reflects the Service Set ID used
              in the DesiredSSID parameter of the most recent
              MLME_Scan.request.  This value may be modified
              by an external management entity and used by the
              local SME to make decisions about the Scanning process."
    ::= { dot11StationConfigEntry 9 }

dot11DesiredBSSType OBJECT-TYPE
       SYNTAX INTEGER { infrastructure(1), independent(2), any(3) }
       MAX-ACCESS read-write
       STATUS current
       DESCRIPTION
              "This attribute shall specify the type of BSS the
              station shall use when scanning for a BSS with which
              to synchronize. This value is used to filter Probe
              Response frames and Beacons. When set to infrastructure,
              the station shall only synchronize with a BSS whose
              Capability Information field has the ESS subfield set
              to 1. When set to independent, the station shall only
              synchronize with a BSS whose Capability Information
              field has the IBSS subfield set to 1. When set to
              any, the station may synchronize to either type of
```

```
                BSS."
        ::= { dot11StationConfigEntry 10 }

dot11OperationalRateSet OBJECT-TYPE
        SYNTAX OCTET STRING (SIZE(1..126))
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
                "This attribute shall specify the set of data rates
                at which the station may transmit data.  Each octet
                contains a value representing a rate.  Each rate
                shall be within the range from 2 to 127,
                corresponding to data rates in increments of
                500 kb/s from 1 Mbit/s to 63.5 Mbit/s, and shall be
                supported (as indicated in the supported rates
                table) for receiving data. This value is reported in
                transmitted Beacon, Probe Request, Probe Response,
                Association Request, Association Response,
                Reassociation Request, and Reassociation Response
                frames, and is used to determine whether a BSS
                with which the station desires to synchronize is
                suitable. It is also used when starting a BSS,
                as specified in  10.3."
        ::= { dot11StationConfigEntry 11 }

dot11BeaconPeriod OBJECT-TYPE
        SYNTAX INTEGER (1..65535)
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
                "This attribute shall specify the number of TUs that a
                station shall use for scheduling Beacon transmissions.
                This value is transmitted in Beacon and Probe Response
                frames."
        ::= { dot11StationConfigEntry 12 }

dot11DTIMPeriod OBJECT-TYPE
        SYNTAX INTEGER(1..255)
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
                "This attribute shall specify the number of beacon
                intervals that shall elapse between transmission of
                Beacons frames containing a TIM element whose DTIM
                Count field is 0. This value is transmitted in
                the DTIM Period field of Beacon frames."
        ::= { dot11StationConfigEntry 13 }

dot11AssociationResponseTimeOut OBJECT-TYPE
        SYNTAX INTEGER(1..4294967295)
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
                "This attribute shall specify the number of TUs that a
                requesting STA should wait for a response to a
                transmitted association-request MMPDU."
        ::= { dot11StationConfigEntry 14 }

dot11DisassociateReason OBJECT-TYPE
```

```
        SYNTAX INTEGER(0..65535)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "This attribute holds the most recently transmitted
                Reason Code in a Disassociation frame.  If no
                Disassociation frame has been transmitted,
                the value of this attribute shall be 0."

REFERENCE "ISO/IEC 8802-11:1999, 7.3.1.7"
     ::= { dot11StationConfigEntry 15 }

dot11DisassociateStation OBJECT-TYPE
        SYNTAX MacAddress
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "This attribute holds the MAC address from the
                Address 1 field of the most recently transmitted
                Disassociation frame.  If no Disassociation frame has
                been transmitted, the value of this attribute
                shall be 0."
     ::= { dot11StationConfigEntry 16 }

dot11DeauthenticateReason OBJECT-TYPE
        SYNTAX INTEGER(0..65535)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "This attribute holds the most recently
                transmitted Reason Code in a Deauthentication frame.
                If no Deauthentication frame has been transmitted, the
                value of this attribute shall be 0."

REFERENCE "ISO/IEC 8802-11:1999, 7.3.1.7"
     ::= { dot11StationConfigEntry 17 }

dot11DeauthenticateStation OBJECT-TYPE
        SYNTAX MacAddress
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "This attribute holds the MAC address from the Address 1
                field of the most recently transmitted Deauthentication frame.
                If no Deauthentication frame has been transmitted,
                the value of this attribute shall be 0."
     ::= { dot11StationConfigEntry 18 }

dot11AuthenticateFailStatus OBJECT-TYPE
        SYNTAX INTEGER(0..65535)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "This attribute holds the most recently
                transmitted Status Code in a failed Authentication frame.
                If no failed Authentication frame has been transmitted, the
                value of this attribute shall be 0."

REFERENCE "ISO/IEC 8802-11:1999, 7.3.1.9"
```

```
    ::= { dot11StationConfigEntry 19 }

dot11AuthenticateFailStation OBJECT-TYPE
       SYNTAX MacAddress
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION
              "This attribute holds the MAC address from the
              Address 1 field of the most recently transmitted
              failed Authentication frame.  If no failed Authentication
              frame has been transmitted, the value of this attribute
              shall be 0."
    ::= { dot11StationConfigEntry 20 }


-- ****************************************************************
-- *    End of dot11StationConfig  TABLE
-- ****************************************************************

-- ****************************************************************
-- *    AuthenticationAlgorithms  TABLE
-- ****************************************************************
dot11AuthenticationAlgorithmsTable OBJECT-TYPE
       SYNTAX SEQUENCE OF Dot11AuthenticationAlgorithmsEntry
       MAX-ACCESS not-accessible
       STATUS current
       DESCRIPTION
           "This (conceptual) table of attributes shall be a set of
           all the authentication algorithms supported by the
           stations.  The following are the default values and the
           associated algorithm:
               Value = 1: Open System
               Value = 2: Shared Key"
       REFERENCE "ISO/IEC 8802-11:1999, 7.3.1.1"
   ::= {  dot11smt 2 }

dot11AuthenticationAlgorithmsEntry OBJECT-TYPE
       SYNTAX Dot11AuthenticationAlgorithmsEntry
       MAX-ACCESS not-accessible
       STATUS current
       DESCRIPTION
           "An Entry (conceptual row) in the Authentication
           Algorithms Table.

           ifIndex - Each 802.11 interface is represented by an
           ifEntry.  Interface tables in this MIB module are indexed
           by ifIndex."

       INDEX { ifIndex,
             dot11AuthenticationAlgorithmsIndex}
   ::= {  dot11AuthenticationAlgorithmsTable  1 }

Dot11AuthenticationAlgorithmsEntry ::= SEQUENCE {
           dot11AuthenticationAlgorithmsIndex            Integer32,
           dot11AuthenticationAlgorithm                  INTEGER,
dot11AuthenticationAlgorithmsEnable                      TruthValue }


dot11AuthenticationAlgorithmsIndex OBJECT-TYPE
       SYNTAX Integer32
```

```
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The auxiliary variable used to identify instances
            of the columnar objects in the Authentication Algorithms Table."
    ::= {  dot11AuthenticationAlgorithmsEntry 1 }

dot11AuthenticationAlgorithm OBJECT-TYPE
        SYNTAX INTEGER {  openSystem (1),   sharedKey (2) }
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
              "This attribute shall be a set of all the authentication
              algorithms supported by the STAs. The following are the
              default values and the associated algorithm.
              Value = 1: Open System
              Value = 2: Shared Key"

     ::= {  dot11AuthenticationAlgorithmsEntry 2 }

dot11AuthenticationAlgorithmsEnable  OBJECT-TYPE
        SYNTAX TruthValue
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
              "This attribute, when true at a station,
              shall enable the acceptance of the authentication
              algorithm described in the corresponding table
              entry in authentication frames received by the
              station that have odd authentication sequence numbers.
              The default value of this attribute shall be 1 for
              the Open System table entry and 2 for all other table entries."

     ::= {  dot11AuthenticationAlgorithmsEntry 3 }

-- ****************************************************************
-- *    End of AuthenticationAlgorithms  TABLE
-- ****************************************************************

-- ****************************************************************
-- *    WEPDefaultKeys  TABLE
-- ****************************************************************
dot11WEPDefaultKeysTable OBJECT-TYPE
        SYNTAX SEQUENCE OF Dot11WEPDefaultKeysEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Conceptual table for WEP default keys.  This table shall
            contain the four WEP default secret key values
            corresponding to the four possible KeyID values.  The WEP
            default secret keys are logically WRITE-ONLY.  Attempts to
            read the entries in this table shall return unsuccessful
            status and values of null or zero.  The default value of
            each WEP default key shall be null."

        REFERENCE "ISO/IEC 8802-11:1999, 8.3.2"
    ::= {  dot11smt 3 }

dot11WEPDefaultKeysEntry OBJECT-TYPE
```

```
        SYNTAX Dot11WEPDefaultKeysEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "An Entry (conceptual row) in the WEP Default Keys Table.

            ifIndex - Each 802.11 interface is represented by an
            ifEntry.  Interface tables in this MIB module are indexed
            by ifIndex."

        INDEX {ifIndex, dot11WEPDefaultKeyIndex}
    ::= {  dot11WEPDefaultKeysTable  1 }

Dot11WEPDefaultKeysEntry ::= SEQUENCE {
            dot11WEPDefaultKeyIndex     INTEGER,
            dot11WEPDefaultKeyValue     WEPKeytype}

dot11WEPDefaultKeyIndex OBJECT-TYPE
        SYNTAX INTEGER (1..4)
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
              "The auxiliary variable used to identify instances
              of the columnar objects in the WEP Default Keys Table.
              The value of this variable is equal to the WEPDefaultKeyID + 1"
    ::= {  dot11WEPDefaultKeysEntry 1 }

dot11WEPDefaultKeyValue OBJECT-TYPE
        SYNTAX WEPKeytype
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "A WEP default secret key value."
    ::= {  dot11WEPDefaultKeysEntry 2 }

-- ****************************************************************
-- *    End of WEPDefaultKeys  TABLE
-- ****************************************************************

-- ****************************************************************
-- *    WEPKeyMappings  TABLE
-- ****************************************************************
dot11WEPKeyMappingsTable OBJECT-TYPE
        SYNTAX SEQUENCE OF Dot11WEPKeyMappingsEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Conceptual table for WEP Key Mappings.  The MIB supports
            the ability to share a separate WEP key for each RA/TA
            pair.  The Key Mappings Table contains zero or one entry
            for each MAC address and contains two fields for each
            entry: WEPOn and the corresponding WEP key.  The WEP key
            mappings are logically WRITE-ONLY.  Attempts to read the
            entries in this table shall return unsuccessful status and
            values of null or zero.  The default value for all WEPOn
            fields is false."
        REFERENCE "ISO/IEC 8802-11:1999, 8.3.2"
    ::= {  dot11smt 4 }
```

```
dot11WEPKeyMappingsEntry OBJECT-TYPE
        SYNTAX Dot11WEPKeyMappingsEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "An Entry (conceptual row) in the WEP Key Mappings Table.

            ifIndex - Each 802.11 interface is represented by an
            ifEntry.  Interface tables in this MIB module are indexed
            by ifIndex."

        INDEX {ifIndex, dot11WEPKeyMappingIndex}
    ::= {  dot11WEPKeyMappingsTable  1 }

Dot11WEPKeyMappingsEntry ::= SEQUENCE {
            dot11WEPKeyMappingIndex              Integer32,
            dot11WEPKeyMappingAddress            MacAddress,
            dot11WEPKeyMappingWEPOn              TruthValue,
            dot11WEPKeyMappingValue              WEPKeytype,
dot11WEPKeyMappingStatus                         RowStatus }

dot11WEPKeyMappingIndex OBJECT-TYPE
        SYNTAX Integer32
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The auxiliary variable used to identify instances
            of the columnar objects in the WEP Key Mappings Table."
    ::= {  dot11WEPKeyMappingsEntry 1 }

dot11WEPKeyMappingAddress OBJECT-TYPE
        SYNTAX MacAddress
        MAX-ACCESS read-create
        STATUS current
        DESCRIPTION
            "The MAC address of the STA for which the values from this
            key mapping entry are to be used."
    ::= {  dot11WEPKeyMappingsEntry 2 }

dot11WEPKeyMappingWEPOn OBJECT-TYPE
        SYNTAX TruthValue
        MAX-ACCESS read-create
        STATUS current
        DESCRIPTION
            "Boolean as to whether WEP is to be used when communicating
            with the dot11WEPKeyMappingAddress STA."
    ::= {  dot11WEPKeyMappingsEntry 3 }

dot11WEPKeyMappingValue OBJECT-TYPE
        SYNTAX WEPKeytype
        MAX-ACCESS read-create
        STATUS current
        DESCRIPTION
            "A WEP secret key value."
    ::= {  dot11WEPKeyMappingsEntry 4 }

dot11WEPKeyMappingStatus OBJECT-TYPE
        SYNTAX RowStatus
        MAX-ACCESS read-create
```

```
        STATUS current
        DESCRIPTION
            "The status column used for creating, modifying, and
            deleting instances of the columnar objects in the WEP key
            mapping Table."

        DEFVAL {active}
    ::= { dot11WEPKeyMappingsEntry 5 }

-- ****************************************************************
-- *    End of WEPKeyMappings  TABLE
-- ****************************************************************

-- ****************************************************************
-- *    dot11PrivacyTable  TABLE
-- ****************************************************************
dot11PrivacyTable OBJECT-TYPE
        SYNTAX SEQUENCE OF Dot11PrivacyEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Group containing attributes concerned with IEEE 802.11
            Privacy.  Created as a table to allow multiple
            instantiations on an agent."

    ::= {  dot11smt 5 }

dot11PrivacyEntry OBJECT-TYPE
        SYNTAX Dot11PrivacyEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "An entry in the dot11PrivacyTable Table.

            ifIndex - Each 802.11 interface is represented by an
            ifEntry.  Interface tables in this MIB module are indexed
            by ifIndex."
        INDEX {ifIndex}
    ::= {  dot11PrivacyTable 1 }

Dot11PrivacyEntry ::= SEQUENCE {
             dot11PrivacyInvoked                TruthValue,
             dot11WEPDefaultKeyID               INTEGER,
             dot11WEPKeyMappingLength           INTEGER,
             dot11ExcludeUnencrypted            TruthValue,
             dot11WEPICVErrorCount              Counter32,
             dot11WEPExcludedCount              Counter32}


dot11PrivacyInvoked OBJECT-TYPE
        SYNTAX TruthValue
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
              "When this attribute is true, it shall indicate that the IEEE
              802.11 WEP mechanism is used for transmitting frames of type
              Data. The default value of this attribute shall be false."
     ::= {  dot11PrivacyEntry 1 }

dot11WEPDefaultKeyID  OBJECT-TYPE
```

```
        SYNTAX INTEGER (0..3)
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "This attribute shall indicate the use of the first,
            second, third, or fourth element of the WEPDefaultKeys
            array when set to values of zero, one, two, or three.  The
            default value of this attribute shall be 0."
        REFERENCE "ISO/IEC 8802-11:1999, 8.3.2"
     ::= {  dot11PrivacyEntry 2 }

dot11WEPKeyMappingLength  OBJECT-TYPE
        SYNTAX INTEGER (10..4294967295)
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "The maximum number of tuples that dot11WEPKeyMappings can hold."
        REFERENCE "ISO/IEC 8802-11:1999, 8.3.2"
     ::= {  dot11PrivacyEntry 3 }

dot11ExcludeUnencrypted  OBJECT-TYPE
        SYNTAX TruthValue
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION

        "When this attribute is true, the STA shall not indicate at
        the MAC service interface received MSDUs that have the WEP
        subfield of the Frame Control field equal to zero. When this
        attribute is false, the STA may accept MSDUs that have the WEP
        subfield of the Frame Control field equal to zero. The default
        value of this attribute shall be false."

      ::= {  dot11PrivacyEntry 4 }

dot11WEPICVErrorCount  OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "This counter shall increment when a frame is received with the
            WEP subfield of the Frame Control field set to one and the value
            of the ICV as received in the frame does not match the ICV value
            that is calculated for the contents of the received frame."
     ::= {  dot11PrivacyEntry 5 }

dot11WEPExcludedCount  OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "This counter shall increment when a frame is received with the
            WEP subfield of the Frame Control field set to zero and the value
            of dot11ExcludeUnencrypted causes that frame to be discarded."
     ::= {  dot11PrivacyEntry 6 }

-- ************************************************************************
-- *    End of dot11Privacy  TABLE
-- ************************************************************************
```

```
-- ****************************************************************
-- *    SMT notification Objects
-- ****************************************************************
    dot11SMTnotification OBJECT IDENTIFIER ::= { dot11smt 6 }

dot11Disassociate NOTIFICATION-TYPE
        OBJECTS { ifIndex, dot11DisassociateReason,
 dot11DisassociateStation }
        STATUS current
        DESCRIPTION
              "The disassociate notification shall be sent when the STA
              sends a Disassociation frame. The value of the notification
              shall include the MAC address of the MAC to which the
              Disassociation frame was sent and the reason for
              the disassociation.

              ifIndex - Each 802.11 interface is represented by an
              ifEntry.  Interface tables in this MIB module are indexed
              by ifIndex."

    ::= { dot11SMTnotification 0 1 }

dot11Deauthenticate NOTIFICATION-TYPE
        OBJECTS { ifIndex, dot11DeauthenticateReason,
 dot11DeauthenticateStation }
        STATUS current
        DESCRIPTION
              "The deauthenticate notification shall be sent when the STA
              sends a Deauthentication frame. The value of the notification
              shall include the MAC address of the MAC to which the
              Deauthentication frame was sent and the reason for the
              deauthentication.

              ifIndex - Each 802.11 interface is represented by an
              ifEntry.  Interface tables in this MIB module are indexed
              by ifIndex."

     ::= { dot11SMTnotification 0 2 }

dot11AuthenticateFail NOTIFICATION-TYPE
        OBJECTS { ifIndex, dot11AuthenticateFailStatus,
 dot11AuthenticateFailStation }
        STATUS current
        DESCRIPTION
              "The authenticate failure notification shall be sent
              when the STA sends an Authentication frame with a
              status code other than 'successful.' The value of
              the notification shall include the MAC address of the
              MAC to which the Authentication frame was sent and the
              reason for the authentication failure.

              ifIndex - Each 802.11 interface is represented by an
              ifEntry.  Interface tables in this MIB module are indexed
              by ifIndex."

     ::= { dot11SMTnotification 0 3 }
```

```
-- ****************************************************************
-- *    End of SMT notification Objects
-- ****************************************************************

-- ****************************************************************
-- *    MAC Attribute Templates
-- ****************************************************************

-- ****************************************************************
-- *    dot11OperationTable  TABLE
-- ****************************************************************
dot11OperationTable OBJECT-TYPE
        SYNTAX SEQUENCE OF Dot11OperationEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Group contains MAC attributes pertaining to the operation
            of the MAC.  This has been implemented as a table in order
            to allow for multiple instantiations on an agent."

    ::= {  dot11mac 1 }

dot11OperationEntry OBJECT-TYPE
        SYNTAX Dot11OperationEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "An entry in the dot11OperationEntry Table.

            ifIndex - Each 802.11 interface is represented by an
            ifEntry.  Interface tables in this MIB module are indexed
            by ifIndex."
        INDEX {ifIndex}
    ::= {  dot11OperationTable 1 }

Dot11OperationEntry ::= SEQUENCE {
            dot11MACAddress                 MacAddress,
            dot11RTSThreshold               INTEGER,
            dot11ShortRetryLimit            INTEGER,
            dot11LongRetryLimit             INTEGER,
            dot11FragmentationThreshold     INTEGER,
            dot11MaxTransmitMSDULifetime    INTEGER,
            dot11MaxReceiveLifetime         INTEGER,
            dot11ManufacturerID             DisplayString,
            dot11ProductID                  DisplayString}


dot11MACAddress OBJECT-TYPE
        SYNTAX MacAddress
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "Unique MAC Address assigned to the STA."
    ::= {  dot11OperationEntry 1 }

dot11RTSThreshold OBJECT-TYPE
        SYNTAX INTEGER (0..2347)
        MAX-ACCESS read-write
        STATUS current
```

```
        DESCRIPTION

        "This attribute shall indicate the number of octets in an MPDU,
        below which an RTS/CTS handshake shall not be performed. An
        RTS/CTS handshake shall be performed at the beginning of any
        frame exchange sequence where the MPDU is of type Data or
        Management, the MPDU has an individual address in the Address1
        field, and the length of the MPDU is greater than
        this threshold. (For additional details, refer to Table 21 in
        9.7.) Setting this attribute to be larger than the maximum
        MSDU size shall have the effect of turning off the RTS/CTS
        handshake for frames of Data or Management type transmitted by
        this STA. Setting this attribute to zero shall have the effect
        of turning on the RTS/CTS handshake for all frames of Data or
        Management type transmitted by this STA. The default value of
        this attribute shall be 2347."

    ::= {  dot11OperationEntry 2 }

dot11ShortRetryLimit OBJECT-TYPE
        SYNTAX INTEGER (1..255)
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION

        "This attribute shall indicate the maximum number of
        transmission attempts of a frame, the length of which is less
        than or equal to dot11RTSThreshold, that shall be made before a
        failure condition is indicated. The default value of this
        attribute shall be 7."

    ::= {  dot11OperationEntry 3 }

dot11LongRetryLimit OBJECT-TYPE
        SYNTAX INTEGER (1..255)
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION

        "This attribute shall indicate the maximum number of
        transmission attempts of a frame, the length of which is
        greater than dot11RTSThreshold, that shall be made before a
        failure condition is indicated. The default value of this
        attribute shall be 4."

    ::= {  dot11OperationEntry 4 }

dot11FragmentationThreshold OBJECT-TYPE
        SYNTAX INTEGER (256..2346)
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION

        "This attribute shall specify the current maximum size, in
        octets, of the MPDU that may be delivered to the PHY. An MSDU
        shall be broken into fragments if its size exceeds the value
        of this attribute after adding MAC headers and trailers. An
        MSDU or MMPDU shall be fragmented when the resulting frame has
        an individual address in the Address1 field, and the length of
        the frame is larger than this threshold. The default value
```

```
        for this attribute shall be the lesser of 2346 or the
        aMPDUMaxLength of the attached PHY and shall never exceed the
        lesser of 2346 or the  aMPDUMaxLength of the attached PHY. The
        value of this attribute shall never be less than 256."

    ::= {  dot11OperationEntry 5 }

dot11MaxTransmitMSDULifetime OBJECT-TYPE
        SYNTAX INTEGER (1..4294967295)
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION

        "The MaxTransmitMSDULifetime shall be the elapsed time in TU,
        after the initial transmission of an MSDU, after which further
        attempts to transmit the MSDU shall be terminated. The default
        value of this attribute shall be 512."

    ::= {  dot11OperationEntry 6 }

dot11MaxReceiveLifetime OBJECT-TYPE
        SYNTAX INTEGER (1..4294967295)
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION

        "The MaxReceiveLifetime shall be the elapsed time in TU,
        after the initial reception of a fragmented MMPDU or MSDU,
        after which further attempts to reassemble the MMPDU or
        MSDU shall be terminated. The default value shall be 512."
    ::= {  dot11OperationEntry 7 }

dot11ManufacturerID OBJECT-TYPE
        SYNTAX DisplayString (SIZE(0..128))
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION

        "The ManufacturerID shall include, at a minimum, the name
        of the manufacturer.  It may include additional
        information at the manufacturer's discretion.  The default
        value of this attribute shall be null."

    ::= {  dot11OperationEntry 8 }

dot11ProductID OBJECT-TYPE
        SYNTAX DisplayString (SIZE(0..128))
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION

        "The ProductID shall include, at a minimum, an identifier
        that is unique to the manufacturer.  It may include
        additional information at the manufacturer's discretion.
        The default value of this attribute shall be null."

    ::= {  dot11OperationEntry 9 }

-- ****************************************************************
-- *    End of dot11OperationEntry  TABLE
```

```
-- ****************************************************************

-- ****************************************************************
-- *    dot11Counters TABLE
-- ****************************************************************
dot11CountersTable OBJECT-TYPE
        SYNTAX SEQUENCE OF Dot11CountersEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION

        "Group containing attributes that are MAC counters.
        Implemented as a table to allow for multiple
        instantiations on an agent."

    ::= {  dot11mac 2 }

dot11CountersEntry OBJECT-TYPE
        SYNTAX Dot11CountersEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION

        "An entry in the dot11CountersEntry Table.

        ifIndex - Each 802.11 interface is represented by an
        ifEntry.  Interface tables in this MIB module are indexed
        by ifIndex."
        INDEX {ifIndex}
    ::= {  dot11CountersTable 1 }

Dot11CountersEntry ::= SEQUENCE {
            dot11TransmittedFragmentCount       Counter32,
            dot11MulticastTransmittedFrameCount Counter32,
            dot11FailedCount                    Counter32,
            dot11RetryCount                     Counter32,
            dot11MultipleRetryCount             Counter32,
            dot11FrameDuplicateCount            Counter32,
            dot11RTSSuccessCount                Counter32,
            dot11RTSFailureCount                Counter32,
            dot11ACKFailureCount                Counter32,
            dot11ReceivedFragmentCount          Counter32,
            dot11MulticastReceivedFrameCount    Counter32,
            dot11FCSErrorCount                  Counter32,
            dot11TransmittedFrameCount          Counter32,
            dot11WEPUndecryptableCount          Counter32 }

dot11TransmittedFragmentCount OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION

        "This counter shall be incremented for an acknowledged
        MPDU with an individual address in the address 1 field
        or an MPDU with a multicast address in the address 1 field
        of type Data or Management."

    ::= {  dot11CountersEntry 1 }
```

```
dot11MulticastTransmittedFrameCount OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION

        "This counter shall increment only when the multicast
        bit is set in the destination MAC address of a successfully
        transmitted MSDU.  When operating as a STA in an ESS, where
        these frames are directed to the AP, this implies having
        received an acknowledgment to all associated MPDUs. "

    ::= { dot11CountersEntry 2 }

dot11FailedCount OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION

        "This counter shall increment when an MSDU is not
        transmitted successfully due to the number of
        transmit attempts exceeding either the
        dot11ShortRetryLimit or dot11LongRetryLimit. "

    ::= { dot11CountersEntry 3 }

dot11RetryCount OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION

        "This counter shall increment when an MSDU is successfully
        transmitted after one or more retransmissions."
    ::= { dot11CountersEntry 4 }

dot11MultipleRetryCount OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION

        "This counter shall increment when an MSDU is successfully
        transmitted after more than one retransmission."

    ::= { dot11CountersEntry 5 }

dot11FrameDuplicateCount  OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION

        "This counter shall increment when a frame is received
        that the Sequence Control field indicates is a
        duplicate."

    ::= { dot11CountersEntry 6 }
```

```
dot11RTSSuccessCount OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION

        "This counter shall increment when a CTS is received in
        response to an RTS."

     ::= { dot11CountersEntry 7 }

dot11RTSFailureCount OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION

        "This counter shall increment when a CTS is not received in
        response to an RTS."

    ::= { dot11CountersEntry 8 }

dot11ACKFailureCount OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION

        "This counter shall increment when an ACK is not received
        when expected."

    ::= { dot11CountersEntry 9 }

dot11ReceivedFragmentCount OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION

        "This counter shall be incremented for each successfully
        received MPDU of type Data or Management."

    ::= { dot11CountersEntry 10 }

dot11MulticastReceivedFrameCount OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION

        "This counter shall increment when a MSDU is received
        with the multicast bit set in the destination
        MAC address."

    ::= { dot11CountersEntry 11 }

dot11FCSErrorCount OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
```

```
        STATUS current
        DESCRIPTION

        "This counter shall increment when an FCS error is
        detected in a received MPDU."

    ::= {  dot11CountersEntry 12 }

dot11TransmittedFrameCount OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION

        "This counter shall increment for each successfully transmitted MSDU."

    ::= {  dot11CountersEntry 13 }

dot11WEPUndecryptableCount OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION

        "This counter shall increment when a frame is received with
        the WEP subfield of the Frame Control field set to one and the
        WEPOn value for the key mapped to the TA's MAC address
        indicates that the frame should not have been encrypted or
        that frame is discarded due to the receiving STA not
        implementing the privacy option."

    ::= { dot11CountersEntry 14 }

-- ****************************************************************
-- *    End of dot11CountersEntry  TABLE
-- ****************************************************************

-- ****************************************************************
-- *    GroupAddresses  TABLE
-- ****************************************************************
dot11GroupAddressesTable OBJECT-TYPE
        SYNTAX SEQUENCE OF Dot11GroupAddressesEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION

        "A conceptual table containing a set of MAC addresses
        identifying the multicast addresses for which this STA
        will receive frames.  The default value of this attribute
        shall be null."

    ::= {  dot11mac 3 }

dot11GroupAddressesEntry OBJECT-TYPE
        SYNTAX Dot11GroupAddressesEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION

        "An Entry (conceptual row) in the Group Addresses Table.
```

```
        ifIndex - Each 802.11 interface is represented by an
        ifEntry.  Interface tables in this MIB module are indexed
        by ifIndex."
      INDEX {ifIndex, dot11GroupAddressesIndex}

     ::= {  dot11GroupAddressesTable  1 }

Dot11GroupAddressesEntry ::= SEQUENCE {
            dot11GroupAddressesIndex    Integer32,
            dot11Address                MacAddress,
            dot11GroupAddressesStatus   RowStatus}

dot11GroupAddressesIndex OBJECT-TYPE
        SYNTAX Integer32
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION

        "The auxiliary variable used to identify instances
        of the columnar objects in the Group Addresses Table."

    ::= {  dot11GroupAddressesEntry 1 }

dot11Address OBJECT-TYPE
        SYNTAX MacAddress
        MAX-ACCESS read-create
        STATUS current
        DESCRIPTION

        "MAC address identifying a multicast addresses
        from which this STA will receive frames."
    ::= {  dot11GroupAddressesEntry 2 }

dot11GroupAddressesStatus OBJECT-TYPE
        SYNTAX RowStatus
        MAX-ACCESS read-create
        STATUS current
        DESCRIPTION

        "The status column used for creating, modifying, and
        deleting instances of the columnar objects in the Group
        Addresses Table."

        DEFVAL {active}
    ::= {  dot11GroupAddressesEntry 3 }
-- ****************************************************************
-- *    End of GroupAddress  TABLE
-- ****************************************************************

-- ****************************************************************
-- *    Resource Type Attribute Templates
-- ****************************************************************
dot11ResourceTypeIDName OBJECT-TYPE
        SYNTAX DisplayString (SIZE(4))
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
```

```
        "Contains the name of the Resource Type ID managed object.
        The attribute is read-only and always contains the value
        RTID.  This attribute value shall not be used as a naming
        attribute for any other managed object class."

        REFERENCE "IEEE Std 802.1F-1993,  A.7"
        DEFVAL {"RTID"}
    ::= {  dot11resAttribute 1 }

-- ****************************************************************
-- *    dot11ResourceInfo  TABLE
-- ****************************************************************
dot11ResourceInfoTable OBJECT-TYPE
        SYNTAX SEQUENCE OF Dot11ResourceInfoEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION

        "Provides a means of indicating, in data readable from a
        managed object, information that identifies the source of
        the implementation."

        REFERENCE "IEEE Std 802.1F-1993,  A.7"
    ::= {  dot11resAttribute 2 }

dot11ResourceInfoEntry OBJECT-TYPE
        SYNTAX Dot11ResourceInfoEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION

        "An entry in the dot11ResourceInfo Table.

        ifIndex - Each 802.11 interface is represented by an
        ifEntry.  Interface tables in this MIB module are indexed
        by ifIndex."
       INDEX {ifIndex}

    ::= {  dot11ResourceInfoTable 1 }

Dot11ResourceInfoEntry ::= SEQUENCE {
            dot11manufacturerOUI             OCTET STRING,
            dot11manufacturerName            DisplayString,
            dot11manufacturerProductName     DisplayString,
            dot11manufacturerProductVersion  DisplayString}


dot11manufacturerOUI OBJECT-TYPE
        SYNTAX OCTET STRING (SIZE(3))
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION

        "Takes the value of an organizationally unique identifier."

    ::= {  dot11ResourceInfoEntry 1 }

dot11manufacturerName OBJECT-TYPE
        SYNTAX DisplayString (SIZE(0..128))
        MAX-ACCESS read-only
```

```
        STATUS current
        DESCRIPTION

        "A printable string used to identify the manufacturer of the
        resource.  Maximum string length is 128 octets."
    ::= {  dot11ResourceInfoEntry 2 }

dot11manufacturerProductName OBJECT-TYPE
        SYNTAX DisplayString (SIZE(0..128))
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION

        "A printable string used to identify the manufacturer's product
        name of the resource.  Maximum string length is 128 octets."
    ::= {  dot11ResourceInfoEntry 3 }

dot11manufacturerProductVersion OBJECT-TYPE
        SYNTAX DisplayString (SIZE(0..128))
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION

        "Printable string used to identify the manufacturer's product
        version of the resource.  Maximum string length is 128 octets."
    ::= {  dot11ResourceInfoEntry 4 }

-- ****************************************************************
-- *    End of dot11ResourceInfo  TABLE
-- ****************************************************************

-- ****************************************************************
-- *   PHY Attribute Templates
-- ****************************************************************

-- ****************************************************************
-- *    dot11PhyOperation  TABLE
-- ****************************************************************
dot11PhyOperationTable OBJECT-TYPE
        SYNTAX SEQUENCE OF Dot11PhyOperationEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION

        "PHY level attributes concerned with
        operation.  Implemented as a table indexed on
        ifIndex to allow for multiple instantiations on an
        Agent."

    ::= {  dot11phy 1 }

dot11PhyOperationEntry OBJECT-TYPE
        SYNTAX Dot11PhyOperationEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION

        "An entry in the dot11PhyOperation Table.
```

```
        ifIndex - Each 802.11 interface is represented by an
        ifEntry.  Interface tables in this MIB module are indexed
        by ifIndex."
      INDEX {ifIndex}
    ::= {  dot11PhyOperationTable 1 }

Dot11PhyOperationEntry ::= SEQUENCE {
            dot11PHYType            INTEGER,
            dot11CurrentRegDomain   Integer32,
            dot11TempType           INTEGER
}

dot11PHYType OBJECT-TYPE
        SYNTAX INTEGER {fhss(1), dsss(2), irbaseband(3)}
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION

        "This is an 8-bit integer value that identifies the PHY type
        supported by the attached PLCP and PMD. Currently defined
        values and their corresponding PHY types are:

        FHSS 2.4 GHz = 01 , DSSS 2.4 GHz = 02, IR Baseband = 03"

    ::= {  dot11PhyOperationEntry 1 }

dot11CurrentRegDomain OBJECT-TYPE
        SYNTAX Integer32
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION

        "The current regulatory domain this instance of the PMD is
        supporting.  This object corresponds to one of the
        RegDomains listed in dot11RegDomainsSupported."

    ::= {   dot11PhyOperationEntry 2 }

dot11TempType OBJECT-TYPE
        SYNTAX INTEGER {tempType1(1), tempType2(2) }
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION

        "There are different operating temperature requirements
        dependent on the anticipated environmental conditions. This
        attribute describes the current PHY's operating temperature
        range capability. Currently defined values and their
        corresponding temperature ranges are:

        Type 1 = X'01'-Commercial range of 0 to 40 degrees C,

        Type 2 = X'02'-Industrial range of -30 to 70 degrees C."

    ::= {   dot11PhyOperationEntry 3 }


-- ****************************************************************
-- *    End of dot11PhyOperation  TABLE
-- ****************************************************************
```

```
-- ****************************************************************
-- *    dot11PhyAntenna  TABLE
-- ****************************************************************
dot11PhyAntennaTable OBJECT-TYPE
        SYNTAX SEQUENCE OF Dot11PhyAntennaEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION

        "Group of attributes for PhyAntenna.  Implemented as a
        table indexed on ifIndex to allow for multiple instances on
        an agent."

    ::= {  dot11phy 2}

dot11PhyAntennaEntry OBJECT-TYPE
        SYNTAX Dot11PhyAntennaEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION

        "An entry in the dot11PhyAntenna Table.

        ifIndex - Each 802.11 interface is represented by an
        ifEntry.  Interface tables in this MIB module are indexed
        by ifIndex."
      INDEX {ifIndex}
    ::= {  dot11PhyAntennaTable 1 }

Dot11PhyAntennaEntry ::= SEQUENCE {
             dot11CurrentTxAntenna          Integer32,
             dot11DiversitySupport          INTEGER,
              dot11CurrentRxAntenna         Integer32 }

dot11CurrentTxAntenna OBJECT-TYPE
        SYNTAX Integer32 (1..255)
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION

        "The current antenna being used to transmit.  This value
        is one of the values appearing in dot11SupportedTxAntenna.
        This may be used by a management agent to control which
        antenna is used for transmission."

    ::= {  dot11PhyAntennaEntry 1 }

dot11DiversitySupport OBJECT-TYPE
        SYNTAX INTEGER {fixedlist(1), notsupported(2), dynamic(3)}
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION

        "This implementation's support for diversity, encoded as:

        X'01'-diversity is available and is performed over the fixed
            list of antennas defined in dot11DiversitySelectionRx.

        X'02'-diversity is not supported.
```

```
        X'03'-diversity is supported and control of diversity is also
            available, in which case the attribute
            dot11DiversitySelectionRx can be dynamically modified by the
            LME."
    ::= {  dot11PhyAntennaEntry 2 }

dot11CurrentRxAntenna OBJECT-TYPE
        SYNTAX Integer32 (1..255)
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION

        "The current antenna being used to receive, if the
        dot11 DiversitySupport indicates that diversity is not
        supported.  The selected antenna shall be one of
        the antennae marked for receive in the dot11AntennasListTable. "

    ::= {  dot11PhyAntennaEntry 3 }

-- ****************************************************************************
-- *    End of dot11PhyAntenna  TABLE
-- ****************************************************************************

-- ****************************************************************************
-- *    dot11PhyTxPower  TABLE
-- ****************************************************************************
dot11PhyTxPowerTable OBJECT-TYPE
        SYNTAX SEQUENCE OF Dot11PhyTxPowerEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION

        "Group of attributes for dot11PhyTxPowerTable.  Implemented
        as a table indexed on STA ID to allow for multiple
        instances on an Agent."

    ::= {  dot11phy 3}

dot11PhyTxPowerEntry OBJECT-TYPE
        SYNTAX Dot11PhyTxPowerEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION

        "An entry in the dot11PhyTxPower Table.

        ifIndex - Each 802.11 interface is represented by an
        ifEntry.  Interface tables in this MIB module are indexed
        by ifIndex."
      INDEX {ifIndex}
    ::= {  dot11PhyTxPowerTable 1 }

Dot11PhyTxPowerEntry ::= SEQUENCE {
            dot11NumberSupportedPowerLevels  INTEGER,
            dot11TxPowerLevel1               INTEGER,
            dot11TxPowerLevel2               INTEGER,
            dot11TxPowerLevel3               INTEGER,
            dot11TxPowerLevel4               INTEGER,
            dot11TxPowerLevel5               INTEGER,
```

```
            dot11TxPowerLevel6              INTEGER,
            dot11TxPowerLevel7              INTEGER,
            dot11TxPowerLevel8              INTEGER,
            dot11CurrentTxPowerLevel        INTEGER}

dot11NumberSupportedPowerLevels OBJECT-TYPE
        SYNTAX INTEGER (1..8)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The number of power levels supported by the PMD.
            This attribute can have a value of 1 to 8."
    ::= {  dot11PhyTxPowerEntry 1 }

dot11TxPowerLevel1 OBJECT-TYPE
        SYNTAX INTEGER (0..10000)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The transmit output power for LEVEL1 in mW.
            This is also the default power level."
    ::= {   dot11PhyTxPowerEntry 2 }

dot11TxPowerLevel2 OBJECT-TYPE
        SYNTAX INTEGER (0..10000)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The transmit output power for LEVEL2 in mW."
    ::= {   dot11PhyTxPowerEntry 3 }

dot11TxPowerLevel3 OBJECT-TYPE
        SYNTAX INTEGER (0..10000)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The transmit output power for LEVEL3 in mW."
    ::= {   dot11PhyTxPowerEntry 4 }

dot11TxPowerLevel4 OBJECT-TYPE
        SYNTAX INTEGER (0..10000)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The transmit output power for LEVEL4 in mW."
    ::= {   dot11PhyTxPowerEntry 5 }

dot11TxPowerLevel5 OBJECT-TYPE
        SYNTAX INTEGER (0..10000)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The transmit output power for LEVEL5 in mW."
    ::= {   dot11PhyTxPowerEntry 6 }

dot11TxPowerLevel6 OBJECT-TYPE
        SYNTAX INTEGER (0..10000)
        MAX-ACCESS read-only
        STATUS current
```

```
        DESCRIPTION
            "The transmit output power for LEVEL6 in mW."
    ::= {   dot11PhyTxPowerEntry 7 }

dot11TxPowerLevel7 OBJECT-TYPE
        SYNTAX INTEGER (0..10000)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The transmit output power for LEVEL7 in mW."
    ::= {   dot11PhyTxPowerEntry 8 }

dot11TxPowerLevel8 OBJECT-TYPE
        SYNTAX INTEGER (0..10000)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The transmit output power for LEVEL8 in mW."
    ::= {   dot11PhyTxPowerEntry 9 }

dot11CurrentTxPowerLevel OBJECT-TYPE
        SYNTAX INTEGER (1..8)
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "The TxPowerLevel N currently being used to transmit data.
            Some PHYs also use this value to determine the receiver
            sensitivity requirements for CCA."

    ::= {   dot11PhyTxPowerEntry 10 }

-- ****************************************************************
-- *    End of dot11PhyTxPower  TABLE
-- ****************************************************************

-- ****************************************************************
-- *    dot11PhyFHSS  TABLE
-- ****************************************************************
dot11PhyFHSSTable OBJECT-TYPE
        SYNTAX SEQUENCE OF Dot11PhyFHSSEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Group of attributes for dot11PhyFHSSTable.  Implemented as a
            table indexed on STA ID to allow for multiple instances on
            an Agent."

    ::= {  dot11phy 4 }

dot11PhyFHSSEntry OBJECT-TYPE
        SYNTAX Dot11PhyFHSSEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "An entry in the dot11PhyFHSS Table.

            ifIndex - Each 802.11 interface is represented by an
            ifEntry.  Interface tables in this MIB module are indexed
            by ifIndex."
```

```
        INDEX {ifIndex}
    ::= {  dot11PhyFHSSTable 1 }

Dot11PhyFHSSEntry ::= SEQUENCE {
             dot11HopTime                    INTEGER,
             dot11CurrentChannelNumber       INTEGER,
             dot11MaxDwellTime               INTEGER,
             dot11CurrentDwellTime           INTEGER,
             dot11CurrentSet                 INTEGER,
             dot11CurrentPattern             INTEGER,
             dot11CurrentIndex               INTEGER}

dot11HopTime OBJECT-TYPE
        SYNTAX INTEGER (224)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The time in microseconds for the PMD to change from
            channel 2 to channel 80"
    ::= {   dot11PhyFHSSEntry 1 }

dot11CurrentChannelNumber OBJECT-TYPE
        SYNTAX INTEGER (0..99)
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
              "The current channel number of the frequency output by the RF
              synthesizer"
    ::= {   dot11PhyFHSSEntry 2 }

dot11MaxDwellTime OBJECT-TYPE
        SYNTAX INTEGER (1..65535)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The maximum time in TU that the transmitter
            is permitted to operate on a single channel."
    ::= {   dot11PhyFHSSEntry 3 }

dot11CurrentDwellTime OBJECT-TYPE
        SYNTAX INTEGER (1..65535)
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "The current time in TU that the transmitter shall operate
            on a single channel, as set by the MAC.  Default is 19 TU."
    ::= {   dot11PhyFHSSEntry 4 }

dot11CurrentSet OBJECT-TYPE
        SYNTAX INTEGER (1..255)
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "The current set of patterns the PHY
            LME is using to determine the hopping sequence. "
    ::= {   dot11PhyFHSSEntry 5 }

dot11CurrentPattern OBJECT-TYPE
        SYNTAX INTEGER (0..255)
```

```
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "The current pattern the PHY LME is
            using to determine the hop sequence."
    ::= {   dot11PhyFHSSEntry 6 }

dot11CurrentIndex OBJECT-TYPE
        SYNTAX INTEGER (1..255)
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "The current index value the PHY LME is using to determine
            the CurrentChannelNumber."

    ::= {   dot11PhyFHSSEntry 7 }

-- ****************************************************************
-- *    End of dot11PhyFHSS  TABLE
-- ****************************************************************

-- ****************************************************************
-- *    dot11PhyDSSSEntry  TABLE
-- ****************************************************************
dot11PhyDSSSTable OBJECT-TYPE
        SYNTAX SEQUENCE OF Dot11PhyDSSSEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Entry of attributes for dot11PhyDSSSEntry.  Implemented as a
            table indexed on ifIndex allow for multiple instances on
            an Agent."

    ::= {  dot11phy 5 }

dot11PhyDSSSEntry OBJECT-TYPE
        SYNTAX Dot11PhyDSSSEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "An entry in the dot11PhyDSSSEntry Table.

            ifIndex - Each 802.11 interface is represented by an
            ifEntry.  Interface tables in this MIB module are indexed
            by ifIndex."
        INDEX {ifIndex}
    ::= {  dot11PhyDSSSTable 1 }

Dot11PhyDSSSEntry ::= SEQUENCE {
             dot11CurrentChannel    INTEGER,
             dot11CCAModeSupported  INTEGER,
             dot11CurrentCCAMode    INTEGER,
             dot11EDThreshold   Integer32}

dot11CurrentChannel OBJECT-TYPE
        SYNTAX INTEGER (1..14)
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
```

```
            "The current operating frequency channel of the DSSS
            PHY. Valid channel numbers are as defined in 15.4.6.2"
    ::= {   dot11PhyDSSSEntry 1 }

dot11CCAModeSupported OBJECT-TYPE
        SYNTAX INTEGER (1..7)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "dot11CCAModeSupported is a bit-significant value, representing all of the
CCA modes supported by the PHY.  Valid values are:

            energy detect only (ED_ONLY) = 01,
              carrier sense only (CS_ONLY) = 02,
              carrier sense and energy detect (ED_and_CS)= 04

or the logical sum of any of these values."
    ::= {   dot11PhyDSSSEntry 2 }

dot11CurrentCCAMode OBJECT-TYPE
        SYNTAX INTEGER {edonly(1), csonly(2), edandcs(4)}
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "The current CCA method in operation.   Valid values are:
                energy detect only (edonly) = 01,
                carrier sense only (csonly) = 02,
                carrier sense and energy detect (edandcs)= 04."
::= {   dot11PhyDSSSEntry 3 }

dot11EDThreshold OBJECT-TYPE
        SYNTAX Integer32
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "The current Energy Detect Threshold being used by the DSSS PHY."
    ::= {   dot11PhyDSSSEntry 4 }

-- ****************************************************************************
-- *    End of dot11PhyDSSSEntry  TABLE
-- ****************************************************************************

-- ****************************************************************************
-- *    dot11PhyIR  TABLE
-- ****************************************************************************
dot11PhyIRTable OBJECT-TYPE
        SYNTAX SEQUENCE OF Dot11PhyIREntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Group of attributes for dot11PhyIRTable.  Implemented as a
            table indexed on ifIndex to allow for multiple instances on
            an Agent."

    ::= {  dot11phy 6 }

dot11PhyIREntry OBJECT-TYPE
        SYNTAX Dot11PhyIREntry
        MAX-ACCESS not-accessible
```

```
        STATUS current
        DESCRIPTION
            "An entry in the dot11PhyIR Table.

            ifIndex - Each 802.11 interface is represented by an
            ifEntry.  Interface tables in this MIB module are indexed
            by ifIndex."
        INDEX {ifIndex}
    ::= {  dot11PhyIRTable 1 }

Dot11PhyIREntry ::= SEQUENCE {
             dot11CCAWatchdogTimerMax       Integer32,
             dot11CCAWatchdogCountMax       Integer32,
             dot11CCAWatchdogTimerMin       Integer32,
             dot11CCAWatchdogCountMin       Integer32}

dot11CCAWatchdogTimerMax OBJECT-TYPE
        SYNTAX Integer32
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "This parameter, together with CCAWatchdogCountMax,
            determines when energy detected in the channel can be
            ignored."

    ::= {   dot11PhyIREntry 1 }

dot11CCAWatchdogCountMax OBJECT-TYPE
        SYNTAX Integer32
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "This parameter, together with CCAWatchdogTimerMax,
            determines when energy detected in the channel can be
            ignored."

    ::= {   dot11PhyIREntry 2 }

dot11CCAWatchdogTimerMin OBJECT-TYPE
        SYNTAX Integer32
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "The minimum value to which CCAWatchdogTimerMax can be
            set."

    ::= {   dot11PhyIREntry 3 }

dot11CCAWatchdogCountMin OBJECT-TYPE
        SYNTAX Integer32
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "The minimum value to which CCAWatchdogCount can be set."
    ::= {   dot11PhyIREntry 4 }

-- ****************************************************************
-- *    End of dot11PhyIR  TABLE
-- ****************************************************************
```

```
-- ****************************************************************
-- *    dot11RegDomainsSupported  TABLE
-- ****************************************************************
dot11RegDomainsSupportedTable OBJECT-TYPE
        SYNTAX SEQUENCE OF Dot11RegDomainsSupportEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "There are different operational requirements dependent on
            the regulatory domain.  This attribute list describes the
            regulatory domains the PLCP and PMD support in this
            implementation.  Currently defined values and their
            corresponding Regulatory Domains are:

            FCC (USA) = X'10', DOC (Canada) = X'20', ETSI (most of
            Europe) = X'30', Spain = X'31', France = X'32', MKK
            (Japan) = X'40' "

    ::= {   dot11phy 7}

dot11RegDomainsSupportEntry OBJECT-TYPE
        SYNTAX Dot11RegDomainsSupportEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "An entry in the dot11RegDomainsSupport Table.

            ifIndex - Each 802.11 interface is represented by an
            ifEntry.  Interface tables in this MIB module are indexed
            by ifIndex."
        INDEX {ifIndex, dot11RegDomainsSupportIndex}
    ::= {  dot11RegDomainsSupportedTable 1 }

Dot11RegDomainsSupportEntry ::= SEQUENCE {
             dot11RegDomainsSupportIndex    Integer32,
             dot11RegDomainsSupportValue    INTEGER}

dot11RegDomainsSupportIndex OBJECT-TYPE
        SYNTAX Integer32
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The auxiliary variable used to identify instances
            of the columnar objects in the RegDomainsSupport Table."
    ::= {  dot11RegDomainsSupportEntry 1 }

dot11RegDomainsSupportValue OBJECT-TYPE

        SYNTAX INTEGER {fcc(16), doc(32), etsi(48), spain (49), france
        (50), mkk (64) }
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "There are different operational requirements dependent on
            the regulatory domain.  This attribute list describes the
            regulatory domains the PLCP and PMD support in this
            implementation.  Currently defined values and their
            corresponding Regulatory Domains are:
```

```
            FCC (USA) = X'10', DOC (Canada) = X'20', ETSI (most of
            Europe) = X'30', Spain = X'31', France = X'32', MKK
            (Japan) = X'40' "
    ::= {  dot11RegDomainsSupportEntry 2 }

-- ****************************************************************
-- *    End of dot11RegDomainsSupported  TABLE
-- ****************************************************************

-- ****************************************************************
-- *    dot11AntennasList  TABLE
-- ****************************************************************
dot11AntennasListTable OBJECT-TYPE
        SYNTAX SEQUENCE OF Dot11AntennasListEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
              "This table represents the list of antennae.  An
              antenna can be marked to be capable of transmitting,
              receiving, and/or for participation in receive diversity.  Each
              entry in this table represents a single antenna with
              its properties.  The maximum number of antennae that can
              be contained in this table is 255."
    ::= {   dot11phy 8 }

dot11AntennasListEntry OBJECT-TYPE
        SYNTAX Dot11AntennasListEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
              "An entry in the dot11AntennasListTable,
              representing the properties of a single antenna.

            ifIndex - Each 802.11 interface is represented by an
            ifEntry.  Interface tables in this MIB module are indexed
            by ifIndex."
        INDEX {ifIndex, dot11AntennaListIndex}
    ::= {  dot11AntennasListTable 1 }

Dot11AntennasListEntry ::= SEQUENCE {
             dot11AntennaListIndex     Integer32,
             dot11SupportedTxAntenna   TruthValue,
             dot11SupportedRxAntenna   TruthValue,
             dot11DiversitySelectionRx TruthValue }

dot11AntennaListIndex OBJECT-TYPE
        SYNTAX Integer32 (1..255)
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
              "The unique index of an antenna which is
              used to identify the columnar objects in
              the dot11AntennasList Table."
    ::= {  dot11AntennasListEntry 1 }

dot11SupportedTxAntenna OBJECT-TYPE
        SYNTAX TruthValue
        MAX-ACCESS read-write
        STATUS current
```

```
        DESCRIPTION
              "When true, this object indicates that the
              antenna represented by dot11AntennaIndex
              can be used as a transmit antenna."

    ::= {  dot11AntennasListEntry 2 }

dot11SupportedRxAntenna OBJECT-TYPE
        SYNTAX TruthValue
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
              "When true, this object indicates that the
              antenna represented by the dot11AntennaIndex
              can be used as a receive antenna."

    ::= {  dot11AntennasListEntry 3 }

dot11DiversitySelectionRx OBJECT-TYPE
        SYNTAX TruthValue
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
              "When true, this object indicates that the
              antenna represented by dot11AntennaIndex can
              be used for receive diversity.  This object
              may only be true if the antenna can be used
              as a receive antenna, as indicated by
              dot11SupportedRxAntenna."

    ::= {  dot11AntennasListEntry 4 }

-- ****************************************************************
-- *    End of dot11AntennasList  TABLE
-- ****************************************************************


-- ****************************************************************
-- *    SupportedDataRatesTx  TABLE
-- ****************************************************************
dot11SupportedDataRatesTxTable OBJECT-TYPE
        SYNTAX SEQUENCE OF Dot11SupportedDataRatesTxEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION

            "The Transmit bit rates supported by the PLCP and PMD,
            represented by a count from X'02-X'7f, corresponding to data
            rates in increments of 500Kb/s from 1 Mbit/s to 63.5 Mbit/s subject
            to limitations of each individual PHY."

    ::= {  dot11phy 9 }

dot11SupportedDataRatesTxEntry OBJECT-TYPE
        SYNTAX Dot11SupportedDataRatesTxEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION

            "An Entry (conceptual row) in the dot11SupportedDataRatesTx
```

```
            Table.

            ifIndex - Each 802.11 interface is represented by an
            ifEntry.  Interface tables in this MIB module are indexed
            by ifIndex."

        INDEX {ifIndex, dot11SupportedDataRatesTxIndex}

    ::= {  dot11SupportedDataRatesTxTable  1 }

Dot11SupportedDataRatesTxEntry ::= SEQUENCE {
            dot11SupportedDataRatesTxIndex  Integer32,
            dot11SupportedDataRatesTxValue  Integer32}

dot11SupportedDataRatesTxIndex OBJECT-TYPE
        SYNTAX Integer32 (1..8)
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION

            "Index object which identifies which data rate to access.
            Range is 1..8."

    ::= {  dot11SupportedDataRatesTxEntry 1 }

dot11SupportedDataRatesTxValue OBJECT-TYPE
        SYNTAX Integer32 (2..127)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION

            "The Transmit bit rates supported by the PLCP and PMD,
            represented by a count from X'02-X'7f, corresponding to data
            rates in increments of 500Kb/s from 1 Mbit/s to 63.5 Mbit/s subject
            to limitations of each individual PHY."

    ::= {  dot11SupportedDataRatesTxEntry 2 }

-- ****************************************************************
-- *    End of dot11SupportedDataRatesTx  TABLE
-- ****************************************************************


-- ****************************************************************
-- *    SupportedDataRatesRx  TABLE
-- ****************************************************************
dot11SupportedDataRatesRxTable OBJECT-TYPE
        SYNTAX SEQUENCE OF Dot11SupportedDataRatesRxEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The receive bit rates supported by the PLCP and PMD,
            represented by a count from X'002-X'7f, corresponding to data
            rates in increments of 500Kb/s from 1 Mbit/s to 63.5 Mbit/s."

    ::= {  dot11phy 10 }

dot11SupportedDataRatesRxEntry OBJECT-TYPE
        SYNTAX Dot11SupportedDataRatesRxEntry
        MAX-ACCESS not-accessible
```

```
        STATUS current
        DESCRIPTION

        "An Entry (conceptual row) in the
        dot11SupportedDataRatesRx Table.

            ifIndex - Each 802.11 interface is represented by an
            ifEntry.  Interface tables in this MIB module are indexed
            by ifIndex."

        INDEX {ifIndex, dot11SupportedDataRatesRxIndex}

    ::= {  dot11SupportedDataRatesRxTable  1 }

Dot11SupportedDataRatesRxEntry ::= SEQUENCE {
            dot11SupportedDataRatesRxIndex  Integer32,
            dot11SupportedDataRatesRxValue  Integer32}

dot11SupportedDataRatesRxIndex OBJECT-TYPE
        SYNTAX Integer32 (1..8)
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Index object which identifies which data rate to access.
            Range is 1..8."
    ::= {  dot11SupportedDataRatesRxEntry 1 }

dot11SupportedDataRatesRxValue OBJECT-TYPE
        SYNTAX Integer32 (2..127)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The receive bit rates supported by the PLCP and PMD,
            represented by a count from X'02-X'7f, corresponding to data
            rates in increments of 500 Kb/s from 1 Mbit/s to 63.5 Mbit/s."
    ::= {  dot11SupportedDataRatesRxEntry 2 }

-- ****************************************************************
-- *    End of dot11SupportedDataRatesRx  TABLE
-- ****************************************************************

-- ****************************************************************
-- *   conformance information
-- ****************************************************************

dot11Conformance  OBJECT IDENTIFIER ::= { ieee802dot11 5 }
dot11Groups  OBJECT IDENTIFIER ::= { dot11Conformance 1 }
dot11Compliances  OBJECT IDENTIFIER ::= { dot11Conformance 2 }

-- ****************************************************************
-- *   compliance statements
-- ****************************************************************
dot11Compliance MODULE-COMPLIANCE
        STATUS  current
        DESCRIPTION
            "The compliance statement for SNMPv2 entities
            that implement the IEEE 802.11 MIB."

  MODULE  -- this module
    MANDATORY-GROUPS {
```

```
        dot11SMTbase2,
        dot11MACbase, dot11CountersGroup,
        dot11SmtAuthenticationAlgorithms,
        dot11ResourceTypeID, dot11PhyOperationComplianceGroup }

   GROUP dot11PhyDSSSComplianceGroup
          DESCRIPTION
              "Implementation of this group is required when object
        dot11PHYType has the value of dsss.  This group is
        mutually exclusive with the groups dot11PhyIRComplianceGroup and
        dot11PhyFHSSComplianceGroup."

    GROUP dot11PhyIRComplianceGroup
          DESCRIPTION
              "Implementation of this group is required when object
        dot11PHYType has the value of irbaseband.  This group is
        mutually exclusive with the groups dot11PhyDSSSComplianceGroup and
        dot11PhyFHSSComplianceGroup."


    GROUP dot11PhyFHSSComplianceGroup
          DESCRIPTION
              "Implementation of this group is required when object
        dot11PHYType has the value of fhss.  This group is
        mutually exclusive with the groups dot11PhyDSSSComplianceGroup and
        dot11PhyIRComplianceGroup."


    -- OPTIONAL-GROUPS { dot11SMTprivacy, dot11MACStatistics,
    --    dot11PhyAntennaComplianceGroup, dot11PhyTxPowerComplianceGroup,
    --    dot11PhyRegDomainsSupportGroup,
    --    dot11PhyAntennasListGroup, dot11PhyRateGroup }
    --

    ::= { dot11Compliances 1 }

-- ****************************************************************************
-- *   Groups - units of conformance
-- ****************************************************************************
dot11SMTbase OBJECT-GROUP
        OBJECTS { dot11StationID, dot11MediumOccupancyLimit,
             dot11CFPollable,
             dot11CFPPeriod,
             dot11CFPMaxDuration,
             dot11AuthenticationResponseTimeOut,
             dot11PrivacyOptionImplemented,
             dot11PowerManagementMode,
             dot11DesiredSSID, dot11DesiredBSSType,
             dot11OperationalRateSet,
             dot11BeaconPeriod, dot11DTIMPeriod,
             dot11AssociationResponseTimeOut
             }
        STATUS deprecated
        DESCRIPTION
         "The SMT object class provides the necessary support at the
         STA to manage the processes in the STA such that the STA may
         work cooperatively as a part of an IEEE 802.11 network."
    ::= {dot11Groups 1    }
```

```
dot11SMTprivacy OBJECT-GROUP
        OBJECTS { dot11PrivacyInvoked,
                dot11WEPKeyMappingLength, dot11ExcludeUnencrypted,
                dot11WEPICVErrorCount , dot11WEPExcludedCount ,
                dot11WEPDefaultKeyID,
                dot11WEPDefaultKeyValue,
                dot11WEPKeyMappingWEPOn,
                dot11WEPKeyMappingValue , dot11WEPKeyMappingAddress,
                dot11WEPKeyMappingStatus }

        STATUS current
        DESCRIPTION
            "The SMTPrivacy package is a set of attributes that shall be
            present if WEP is implemented in the STA."
    ::= {dot11Groups 2 }

dot11MACbase OBJECT-GROUP

            OBJECTS { dot11MACAddress, dot11Address,
                    dot11GroupAddressesStatus,
                    dot11RTSThreshold, dot11ShortRetryLimit,
                    dot11LongRetryLimit, dot11FragmentationThreshold,
                    dot11MaxTransmitMSDULifetime,
                    dot11MaxReceiveLifetime, dot11ManufacturerID,
                    dot11ProductID

                    }
        STATUS current
        DESCRIPTION
         "The MAC object class provides the necessary support for the
         access control, generation, and verification of frame check
         sequences, and proper delivery of valid data to upper
         layers."
    ::= {dot11Groups 3 }


dot11MACStatistics OBJECT-GROUP

        OBJECTS { dot11RetryCount, dot11MultipleRetryCount,
            dot11RTSSuccessCount, dot11RTSFailureCount,
            dot11ACKFailureCount, dot11FrameDuplicateCount }

        STATUS current
        DESCRIPTION
            "The MACStatistics package provides extended statistical
            information on the operation of the MAC.  This
            package is completely optional."
    ::= {dot11Groups 4 }

dot11ResourceTypeID OBJECT-GROUP
        OBJECTS { dot11ResourceTypeIDName, dot11manufacturerOUI,
            dot11manufacturerName, dot11manufacturerProductName,
            dot11manufacturerProductVersion }
        STATUS current
        DESCRIPTION

        "Attributes used to identify a STA, its manufacturer,
        and various product names and versions."

    ::= {dot11Groups 5  }
```

```
dot11SmtAuthenticationAlgorithms OBJECT-GROUP
        OBJECTS { dot11AuthenticationAlgorithm,
                  dot11AuthenticationAlgorithmsEnable }
        STATUS current
        DESCRIPTION
            "Authentication Algorithm Table."
    ::= {dot11Groups 6 }


dot11PhyOperationComplianceGroup OBJECT-GROUP
        OBJECTS { dot11PHYType, dot11CurrentRegDomain, dot11TempType }
        STATUS current
        DESCRIPTION
            "PHY layer operations attributes."
    ::= { dot11Groups 7 }


dot11PhyAntennaComplianceGroup OBJECT-GROUP
        OBJECTS {dot11CurrentTxAntenna, dot11DiversitySupport,
                 dot11CurrentRxAntenna }
        STATUS current
        DESCRIPTION
            "Attributes for Data Rates for IEEE 802.11."
    ::= { dot11Groups 8 }

dot11PhyTxPowerComplianceGroup OBJECT-GROUP
        OBJECTS {dot11NumberSupportedPowerLevels, dot11TxPowerLevel1,
            dot11TxPowerLevel2, dot11TxPowerLevel3, dot11TxPowerLevel4,
            dot11TxPowerLevel5, dot11TxPowerLevel6, dot11TxPowerLevel7,
            dot11TxPowerLevel8, dot11CurrentTxPowerLevel }
        STATUS current
        DESCRIPTION
            "Attributes for Control and Management of transmit power."
    ::= { dot11Groups 9 }

dot11PhyFHSSComplianceGroup OBJECT-GROUP

        OBJECTS {dot11HopTime, dot11CurrentChannelNumber, dot11MaxDwellTime,
                 dot11CurrentDwellTime, dot11CurrentSet, dot11CurrentPattern,
                 dot11CurrentIndex}
        STATUS current
        DESCRIPTION
            "Attributes that configure the Frequency Hopping for IEEE
            802.11."

    ::= { dot11Groups 10 }

dot11PhyDSSSComplianceGroup OBJECT-GROUP

        OBJECTS {dot11CurrentChannel, dot11CCAModeSupported,
                dot11CurrentCCAMode, dot11EDThreshold}
        STATUS current
        DESCRIPTION
            "Attributes that configure the DSSS for IEEE 802.11."
    ::= { dot11Groups 11 }

dot11PhyIRComplianceGroup OBJECT-GROUP
        OBJECTS {dot11CCAWatchdogTimerMax, dot11CCAWatchdogCountMax,
                 dot11CCAWatchdogTimerMin, dot11CCAWatchdogCountMin}
```

```
        STATUS current
        DESCRIPTION
            "Attributes that configure the baseband IR for IEEE 802.11."
    ::= { dot11Groups 12 }

dot11PhyRegDomainsSupportGroup OBJECT-GROUP
        OBJECTS { dot11RegDomainsSupportValue}
        STATUS current
        DESCRIPTION
            "Attributes that specify the supported Regulation Domains."
    ::= { dot11Groups 13}

dot11PhyAntennasListGroup OBJECT-GROUP
        OBJECTS { dot11SupportedTxAntenna,
                  dot11SupportedRxAntenna, dot11DiversitySelectionRx }
        STATUS current
        DESCRIPTION
            "Attributes that specify the supported Regulation Domains."
    ::= { dot11Groups 14 }

dot11PhyRateGroup OBJECT-GROUP
        OBJECTS {dot11SupportedDataRatesTxValue,
                dot11SupportedDataRatesRxValue
 }
        STATUS current
        DESCRIPTION
            "Attributes for Data Rates for IEEE 802.11."
    ::= { dot11Groups 15 }


dot11CountersGroup OBJECT-GROUP
        OBJECTS {
                    dot11TransmittedFragmentCount   ,
                    dot11MulticastTransmittedFrameCount ,
                    dot11FailedCount, dot11ReceivedFragmentCount,
                    dot11MulticastReceivedFrameCount    ,
                    dot11FCSErrorCount,
  dot11WEPUndecryptableCount,
  dot11TransmittedFrameCount }

        STATUS current
        DESCRIPTION
            "Attributes from the dot11CountersGroup that are not described
            in the dot11MACStatistics group.  These objects are
            mandatory."
    ::= {dot11Groups 16 }

dot11NotificationGroup NOTIFICATION-GROUP
      NOTIFICATIONS { dot11Disassociate,
                      dot11Deauthenticate,
                      dot11AuthenticateFail }
       STATUS current
       DESCRIPTION
              "IEEE 802.11 notifications"
       ::= { dot11Groups 17 }



dot11SMTbase2 OBJECT-GROUP
```

```
        OBJECTS { dot11MediumOccupancyLimit,
                  dot11CFPollable,
                  dot11CFPPeriod,
                  dot11CFPMaxDuration,
                  dot11AuthenticationResponseTimeOut,
                  dot11PrivacyOptionImplemented,
                  dot11PowerManagementMode,
                  dot11DesiredSSID, dot11DesiredBSSType,
                  dot11OperationalRateSet,
                  dot11BeaconPeriod, dot11DTIMPeriod,
                  dot11AssociationResponseTimeOut,
                  dot11DisassociateReason,
                  dot11DisassociateStation,
                  dot11DeauthenticateReason,
                  dot11DeauthenticateStation,
                  dot11AuthenticateFailStatus,
                  dot11AuthenticateFailStation
                  }
        STATUS current
        DESCRIPTION
            "The SMTbase2 object class provides the necessary support at the
            STA to manage the processes in the STA such that the STA may
            work cooperatively as a part of an IEEE 802.11 network."
    ::= {dot11Groups 18 }

-- ****************************************************************************
-- *   End of 80211 MIB
-- ****************************************************************************
END
```

Annex E

(informative)

Bibliography

E.1 General

[B1] ANSI Z136.1-1993, American National Standard for the Safe Use of Lasers.

[B2] IEC 60825-1 (1993), Safety of laser products—Part 1: Equipment classification, requirements and user's guide.

[B3] IEEE Std 802.10-1998, IEEE Standards for Local and Metropolitan Area Networks: Interoperable LAN/MAN Security (SILS).

[B4] Schneier, Bruce, *Applied Cryptography, Protocols, Algorithms and Source Code in C.* New York: Wiley, 1994.

E.2 Specification and description language (SDL) documentation

[B5] Belina, Ferenc, Dieter Hogrefe, and Amardeo Sarma, *SDL with Applications from Protocol Specification.* UK: Prentice Hall Europe, Hertfordshire, 1991.[8]

[B6] Ellsberger, Jan, Dieter Hogrefe, and Amardeo Sarma, *SDL, Formal Object-Oriented Language for Communicating Systems.* Hertfordshire, UK: Prentice Hall Europe, 1997.[9]

[B7] Faergemand, Ove and Anders Olsen, "New Features in SDL-92," *SDL Newsletter* (ISSN 1023-7151), no. 16 (May 1993), pp. 10–29. Also available online at http://www.tdr.dk/public/SDL/SDL.html.[10]

[B8] Olsen, Anders, Ove Faergemand, Birger Moller-Pedersen, Rick Reed, and T. R. W. Smith, *Systems Engineering Using SDL-92.* Amsterdam, the Netherlands: Elsevier Science B.V., 1994.[11]

[8] An introductory text on SDL, also useful as a language reference (for SDL-88).

[9] A recently published book, which appears to be the most comprehensive single-volume introduction and reference for SDL-92, including its object-oriented extensions.

[10] This provides a summary of the changes from SDL-88 to SDL-92.

[11] A detailed guide to using SDL-92, including a thorough explanation of abstract data type mechanism and SDL combined with ASN.1 (ITU-T Recommendation Z.105).