Universidade Estadual de Campinas
FACULDADE DE ENGENHARIA ELÉTRICA E COMPUTAÇÃO
DEPARTAMENTO DE TELEMÁTICA

UNICAMP

Uma Especificação Formal e Funcional de Interconexão de Redes DQDB/ATM para Serviços com Conexão Não Orientada

Tese apresentada à Faculdade de Engenharia Elétrica da Universidade Estadual de Campinas, como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica.

por

Luciano Pinto De Nadai
Engenheiro Eletricista - UFC/CE

em 25 de outubro de 1996 perante a banca examinadora

Prof. Dr. Akebo Yamakami	Orientador
Prof. Dr. Rege Romeu Scarabucci	FEEC/UNICAMP
Prof. Dr. Vitório Bruno Mazzola	UFSC/SC

Este exemplar corresponde à redação final da tese, defendida por Luciano Pinto de Nadai e aprovada pela Comissão Julgadora em 25/10/96

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

N121e

Nadai, Luciano Pinto de
Uma especificação formal e funcional de interconexão de redes DQDB/ATM para serviços com conexão não orientada / Luciano Pinto de Nadai.--Campinas, SP: [s.n.], 1996.

Orientador: Akebo Yamakami.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Redes de computadores - Protocolos. 2. Redes de computação. I. Yamakami, Akebo. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Agradecimentos

Gostaria de expressar os meus mais sinceros agradecimentos a todas as pessoas que me apoiaram e incentivaram, permitindo a conclusão deste trabalho. Especialmente,
Ao meu orientador Prof. Dr. Akebo Yamakami pelo incentivo, dedicação e infinita paciência na revisão deste trabalho.
Aos meus pais, Lino e Valéria, pela priorização da qualidade de nossa educação e formação.

Resumo

Este trabalho apresenta uma implementação formal e funcional da integração de LAN's com protocolos DQDB e redes ATM, baseado nas recomendações encontradas nas normas do CCITT. Concentramos nossa atenção em serviços com conexão não orientada e utilizamos a linguagem SDL para implementação virtual do procedimento proposto e posterior simulação, visando sua validação. Foi adotado o cenário direto para o suporte ao serviço não orientado à conexão em redes DQDB/ATM, implementando o elemento adaptador de protocolos(IWU) como *bridge*. Para isso, foi proposto um formato para o campo de endereços do quadro DQDB. Resultados das simulações de várias situações são apresentados com análises e comentários.

Palavras Chaves: LAN, DQDB, ATM, Interconexão de redes locais.

Abstract

This work present a formal and functional implementation of DQDB-ATM LAN's integration, based on recomendations found in CCITT standards. We will treat connectionless services and the SDL language is used to implement and simulate the proposed procedures. Direct scenary has been adopted to support the connectionless services of the DQDB/ATM LAN'S, using the IWU element as a bridge. A suitable format has been proposed to the addressing field of the DQDB slot. Simulation results of several cases are analised and discussed.

Keywords: LAN, DQDB, ATM, Local Area Network Interconnection.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

As tendências de globalização e necessidades de intercâmbio de informações cada vez mais precisas e rápidas, tornou a questão de redes um assunto de relevância nos dias de hoje. Vemos uma tendência cada vez mais crescente de integração de diversos serviços quanto ao tratamento e transmissão através de meios físicos. Dentro deste aspecto, várias propostas de protocolos de redes, de transmissão, de comutação de pacotes, etc, são encontradas na literatura.

Dentre os procedimentos pertencentes a este cenário que hoje têm atenção significativa da comunidade científica e produtiva, podemos citar ATM(*Asynchronous Transfer Mode*) e DQDB(*Distributed Queued Dual Bus*), os quais são objetos de nosso estudo neste trabalho.

Vamos analisar a integração de LAN's DQDB através de uma rede ATM, seguindo as recomendações encontradas em [8] e [15].

Inicialmente vamos discutir brevemente os protocolo DQDB, principalmente para serviços não orientados à conexão e quanto aos aspectos de interconexão das redes, para em seguida, apresentar nossa proposta de especificação formal. Neste trabalho é utilizada a linguagem SDL(*Specification and Description Language*) [4] na implementação funcional da proposta e simulações de alguns exemplos são apresentados e discutidos.

1.1 Organização do trabalho

O trabalho está dividido em seis capítulos, dos quais o capítulo 1, introdutório, cita a necessidade cada vez maior de integração de diversos serviços e interconexão de redes geograficamente dispersas que motivaram o desenvolvimento deste tema. O capítulo 2 apresenta as características básicas do protocolo DQDB.

O capítulo 3 aborda os cenários previstos para o suporte ao serviço não orientado à conexão, aspectos e opções de implementação dos elementos de rede envolvidos nos cenários e uma proposta de interconexão entre redes DQDB e ATM.

No capítulo 4 é apresentado o modelo e as especificações funcionais dos elementos que compõem o sistema de interconexão DQDB/ATM na linguagem SDL, como também, as metodologias de desenvolvimento utilizadas.

No capítulo 5 são apresentados "traces" de simulação, obtidos da ferramenta SDT(*SDL Design Tool*), que são utilizados para a validação dos elementos que compõem o sistema de interconexão DQDB/ATM.

O último capítulo conclui o trabalho relacionado e ressalta a importância da implementação da interconexão através de uma linguagem de simulação e especificação formal para obtenção de protótipos funcionais.

Três apêndices complementam este trabalho, apresentando:

Apêndice A: os "traces" de depuração de cada caso estudado.

Apêndice B: modelo SDL do sistema de interconexão de redes DQDB/ATM.

Apêndice C: o glossário.

1.2 Motivação

Os principais pontos que influíram decisivamente no desenvolvimento deste trabalho foram: domínio das tecnologias emergentes e promissoras, no caso DQDB e ATM; proposta de uma arquitetura para interconexão entre essas redes, definindo a natureza e formatos de endereçamento; validação da proposta através de simulações de diversas situações.

Esta arquitetura permite interconexão de diversas LAN's DQDB geograficamente distribuídas mas pertencentes a uma mesma classe de endereçamento, através de um *backbone* ATM.

Capítulo 2

Rede DQDB

A necessidade de integrar serviços, tais como: dados, voz e vídeo, até então oferecidos por redes locais, ao longo de uma grande área geográfica, contribuiu para o surgimento de novos padrões para redes de área metropolitana(*Metropolitan Area Network-MAN*).

Embora as redes digitais de serviços integrados de faixa larga (*Broadband Integrated Services Digital Network-BISDN*) que utilizam a técnica de transferência assíncrona(*Asynchronous Transfer Mode-ATM)* representem uma promessa de redes cada vez mais velozes, os altos custos dessas novas tecnologias motivaram o estudo de novas alternativas para prover serviços em uma rede MAN.

Depois de inúmeras reuniões, o grupo de trabalho IEEE 802.6 em 1987 resolveu optar por um padrão de sub-rede denominado de DQDB(*Distributed Queue Dual Bus*) que suportava serviços não orientados à conexão, serviços orientados à conexão e serviços isócronos [1]. O termo DQDB refere-se ao tipo de topologia empregada e a técnica de acesso ao meio.

Apesar do seu ótimo desempenho para transferência de dados, DQDB apresenta limitações para serviços de multimídia e de faixa larga. Portanto, baseado nas sugestões do IEEE 802.6, uma sub-rede ou um conjunto de sub-redes DQDB interconectadas por *bridges* ou *routers*, podem ser usadas como concentradores de tráfego de LAN's, de redes de comutação de pacote, de redes de comutação de circuito, ISDN, etc, formando assim uma rede MAN capaz de suportar diversos serviços (fig 2.1).

A seguir citamos algumas das características do padrão DQDB que foram herdadas das LAN's e ATM [2].

- Altas velocidades: O DQDB oferece uma variedade de velocidades. Inicialmente o IEEE 802.6 especificou a velocidade de 44.7 Mbps, porém novos estudos estão sendo realizados para que velocidades de 1.544 Mbps até 155 Mbps suportadas por LAN's e pela BISDN, sejam possíveis.

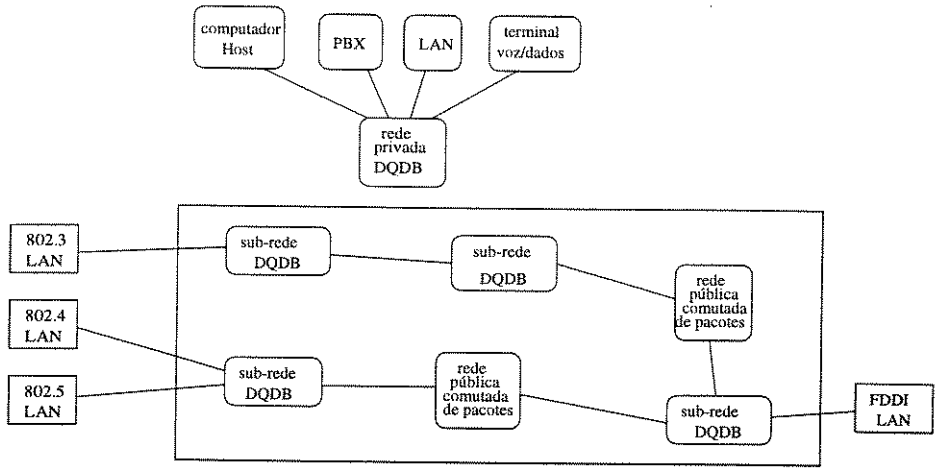

Figura 2.1: MAN formada por sub-redes DQDB

- Meio Compartilhado: Como LAN's, o DQDB utiliza um meio compartilhado com maior capacidade, garantindo um maior desempenho no tráfego de dados em rajada(*bursty data*), tráfego assíncrono e tráfego isócrono.

- Suporte para a sub-camada de controle de enlace lógico(*Logical Link Control-LLC*): O DQDB possui internamente implementado funções específicas para prover serviços semelhantes aos oferecidos pelo nível 2 do modelo OSI, ou pela sub-camada LLC do padrão 802 para redes locais.

- Endereçamento: As estações DQDB são capazes de operar com endereçamento de 48 bits ou 16 bits, idêntico aos das redes locais, como também operar com 64 bits, atualmente padronizado para a BISDN. Portanto, o DQDB pode ser interconectado com qualquer uma dessas redes.

- Tamanho fixo dos pacotes: Diferentemente das redes locais clássicas e semelhante a rede ATM, o DQDB utiliza um tamanho fixo de pacote. Por motivos de compatibilidade com o padrão ATM, o DQDB utiliza o mesmo tamanho de célula ATM, isto é, 53 bytes, com 48 bytes de carga útil. Entretanto, no DQDB, esse pacote foi denominado de slot.

- Barramento duplo: Diferentemente das redes locais clássicas, o DQDB utiliza dois barramentos totalmente independentes.

2.1 Topologia

A rede DQDB [1] consiste de um barramento duplo (fig. 2.2a), onde cada um dos barramentos é unidirecional e transmite informações em direções opostas, obtendo dessa maneira, uma comunicação *full duplex* entre qualquer par de estações.

As transmissões nos dois barramentos são independentes, isto é, nenhum slot pode trafegar por mais de um barramento, pois todos os slots são descartados no final do respectivo barramento. A taxa de transmissão de dados efetiva é o dobro da taxa de cada barramento.

Em cada barramento, a primeira estação na direção do fluxo é a responsável pela geração dos slots (fig. 2.2b). Sob condições normais, existe uma única fonte de temporização para os dois barramentos. Isto é necessário para manter estável a operação do mecanismo de acesso às filas distribuídas e para assegurar o serviço isócrono. A fonte de temporização pode ser externa ou interna e deve operar um relógio de período nominal de 125μs, semelhante ao utilizado pela rede de telefonia para serviços de voz.

Ainda, a rede DQDB possue a capacidade de se auto reconfigurar, pois cada estação é equipada com uma chave comutadora(*bypass switch*) capaz de isolar uma falha na rede, evitando desse modo, a interrupção de algum serviço.

Com redes DQDB, ao contrário de redes locais que utilizam a técnica CSMA_CD para acessar o meio de transmissão, colisões de slots não ocorrerão, pois através da leitura do status de cada slot, as estações não poderão preencher slots que já tenham sido preenchidos por outras estações.

2.1.1 Topologia em dupla barra

Como foi mencionado, nesse tipo de topologia, as estações situadas no início do fluxo de cada barramento serão responsáveis pela geração dos slots vazios. Na ocorrência de uma eventual falha no enlace ou em uma das estações, as estações geradoras de slots vazios informam as estações adjacentes à falha que essas ativem suas chaves comutadoras para que desse modo a falha seja isolada. Porém, a comunicação *full duplex* entre determinadas estações poderá ser rompida devido a segmentação do barramento duplo (fig. 2.2c).

2.1.2 Topologia em dupla barra circular

Na topologia em dupla barra circular (fig. 2.3a), diferentemente da topologia em dupla barra, a função geradora de slots vazios é desempenhada por somente uma estação (fig. 2.3b). Na presença de uma falha, a estação geradora de slots vazios informa as estações adjacentes que essas ativem suas chaves comutadoras e assumam a função geradora de slots vazios. Desse modo, a falha é isolada e a comunicação *full duplex* entre quaisquer estações é mantida (fig. 2.3c). Portanto, essa topologia apresenta uma confiabilidade maior.

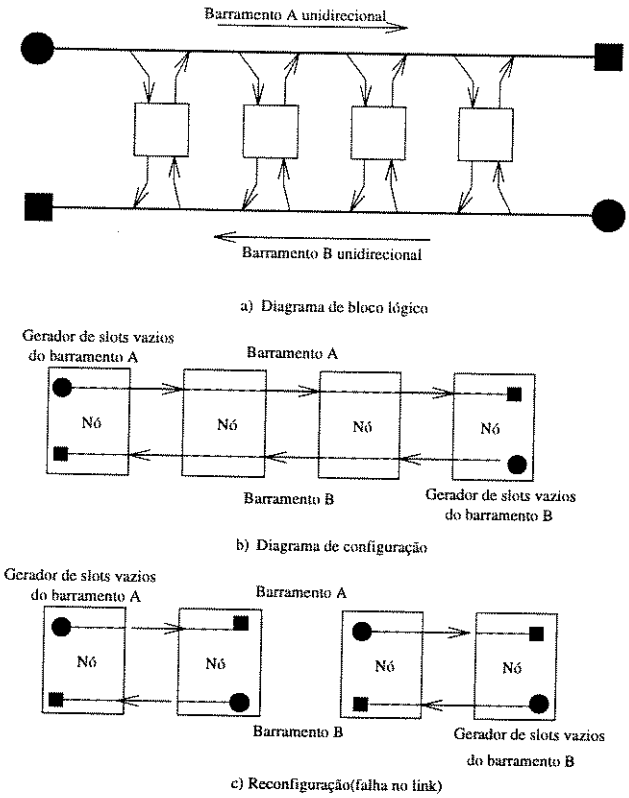

Figura 2.2: Topologia em dupla barra

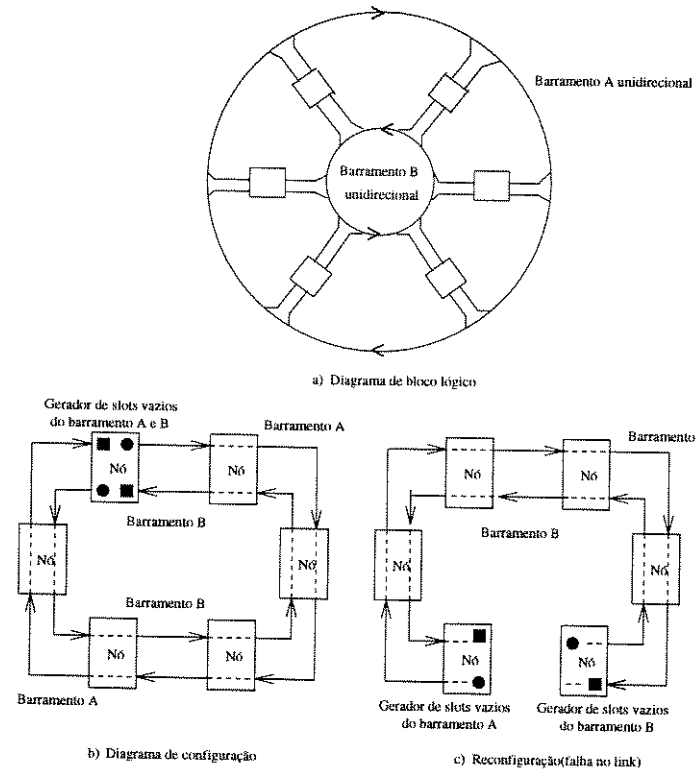

Figura 2.3: Topologia em dupla barra circular

2.2 Arquitetura de Protocolos da Rede DQDB

O padrão DQDB é dividido em três camadas (fig. 2.4):

- Uma camada mais elevada, que corresponde a camada de enlace de dados do padrão OSI(*Open System Interconnection*). No caso de redes locais, essa camada pode ser comparada com a sub-camada de controle de enlace de dados(*Logical Link Control-LLC*). Porém, no caso de redes DQDB, essa camada pode suportar uma variedade enorme de protocolos de acordo com as necessidades dos serviços, por exemplo, retransmissão de pacotes de informação. É importante salientar que os protocolos implementados nessa camada não fazem parte do padrão IEEE 802.6.

- Uma camada intermediária, denominada de camada DQDB. Essa camada é responsável por regular o acesso ao meio de transmissão e corresponde a sub-camada de controle de acesso ao meio(*Medium Access Control-MAC*) do padrão OSI para redes locais.

- Uma camada inferior, que corresponde a camada física.

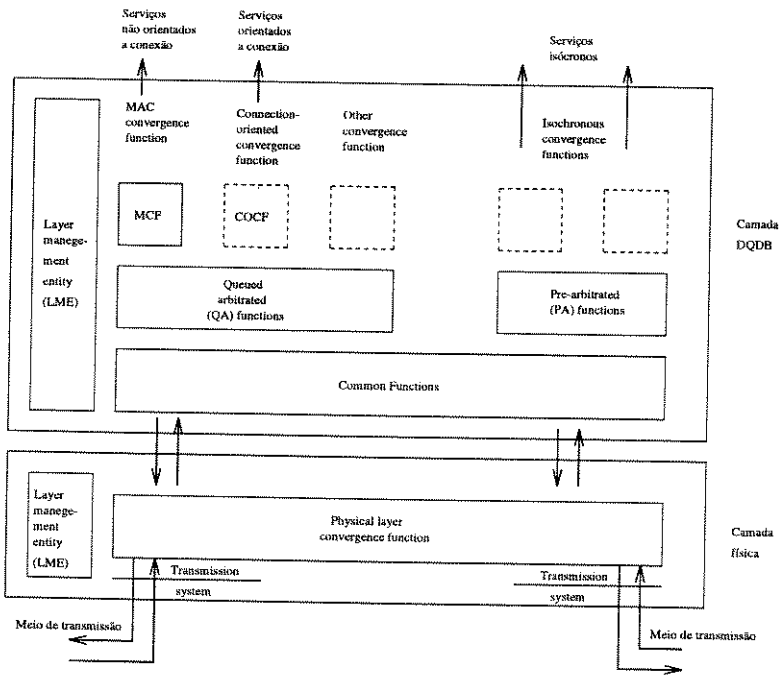

Figura 2.4: Arquitetura funcional do DQDB

2.2.1 Serviços DQDB

Como foi dito anteriormente, a camada acima da camada DQDB não faz parte do padrão IEEE 802.6. Portanto, ela pode ser utilizada para definir os tipos de serviços que a rede DQDB pode suportar. Também, funções de convergência implementadas dentro da camada DQDB poderão definir serviços para os usuários. 0 IEEE 802.6 define três tipos de serviços:

- Serviços com conexão não orientada. Esses serviços suportam a transferência de quadros de informação de até 9188 bytes. A transmissão é realizada através de segmentos de 52 bytes e processos de segmentação e remontagem de pacotes de informação podem ser necessários.

- Serviços com conexão orientada. Esses serviços suportam a transferência de segmentos de 52 bytes entre estações que compartilham um mesmo canal virtual através de uma conexão virtual. Os processos de segmentação e remontagem poderão ser necessários. Porém, os sinais de controle necessários para estabelecer, manter e desfazer a conexão virtual não são definidos pelo padrão IEEE 802.6.

- Serviços isócronos. Esses serviços possibilitam que usuários usufruam de serviços CBR(*Continuous bit rate*). Porém, os sinais de controle necessários para estabelecer, manter e desfazer a conexão síncrona não são definidos pelo padrão IEEE 802.6.

2.2.2 Camada DQDB

A camada DQDB pode ser vista como sendo organizada dentro de três sub-camadas:

- Funções Comuns
- Funções de Arbitragem
- Funções de Convergência

Funções Comuns tratam da retransmissão de slots nas duas direções dos barramentos, providenciam uma plataforma comum para serviços assíncronos e síncronos, e se responsabilizam pela geração de slots, pela função de controle da configuração e pela função de alocação de MID's(*Message Identifier*).

- A função de geração de slots, dependendo da topologia utilizada, é desempenhada por uma ou duas estações para gerar e transmitir slots vazios. Dependendo das solicitações, indicará através do cabeçalho do slot, se esse slot será síncrono ou assíncrono, ou também se esse slot será usado para estabelecer uma conexão através de um canal virtual.

- A função de controle de configuração, é utilizada na inicialização da sub-rede e na reconfiguração após uma falha. Um exemplo é a ativação e desativação da função de geração de slots de determinadas estações(adjacentes à falha) durante o processo de reconfiguração.

- A função de alocação de MID's, é responsável pela alocação de MID's para todas as estações da sub-rede. O MID é usado nos processos de segmentação e remontagem dos quadros dos usuários DQDB. Essa função será melhor compreendida após a explanação detalhada da camada DQDB nas próximas seções.

Funções de Arbitragem são responsáveis pelo controle de acesso ao meio. Existem dois tipos de slots que trafegam no barramento: slots de filas distribuídas(*Queue distributed-QA*) que transportam dados assíncronos e slots previamente arbitrados(*Prearbitrated-PA*) que transportam dados síncronos.

A camada DQDB é responsável pelos vários serviços suportados pela rede DQDB. Para cada serviço, uma **função de convergência** é necessária para mapear um bloco de informação de um usuário DQDB para dentro de slots DQDB. Vale a pena ressaltar que a camada de adaptação da rede ATM, também utiliza funções de convergência para o suporte dos serviços. A seguir apresentamos as seguintes funções de convergência encontradas na camada DQDB:

- Função de Convergência MAC, utilizada na provisão de serviço MAC para a LLC, proporcionando um serviço com conexão não orientada. É importante ressaltar que em nosso trabalho nos restringiremos a análise somente dessa função.
- Função de Convergência Isócrona, utilizada na provisão de serviços isócronos.
- Função de Convergência orientada à conexão, providencia o suporte a serviços orientados à conexão em slots QA.

2.2.3 Camada física

A camada DQDB é independente da camada física. Portanto, uma variedade de redes DQDB pode ser implementada usando o mesmo controle de acesso, porém operando em diferentes taxas através de diferentes sistemas de transmissão. Três sistemas são mencionados pelo padrão IEEE 802.6.

- ANSI DS3 : transmite dados na taxa de 44.736 Mbps através de cabo coaxial e fibra óptica.
- ABNSI SONET(CCITT SDH) : transmite dados na taxa de 155.52 Mbps e acima desta, através de fibras ópticas monomodo.
- CCITT G.703 : transmite dados na taxa de 34.368 Mbps e 139.264 Mbps através de um meio metálico.

2.3 Controle de Acesso ao Meio

Os modos de controle de acesso arbitrado(QA) e pré-arbitrado(PA), utilizam slots QA e PA, respectivamente. Cada slot contém um campo de controle de acesso(ACF) e um campo de segmento, que contém a carga de transmissão(*payload*). O acesso QA é controlado pelo protocolo de fila distribuída e é usado tipicamente para o oferecimento de serviços não isócronos. O acesso PA é usado tipicamente para o oferecimento de serviços isócronos.

2.3.1 Protocolo de Filas Distribuídas

O protocolo de fila distribuída para acesso ao barramento A é ilustrado na figura 2.5 e também se aplica ao barramento B.

O ACF de cada slot contém um bit de ocupação(*busy bit*) e um campo de requisição, com três bits de requisição(*request bit*), um para cada nível de prioridade. O bit de ocupação indica se o slot está sendo usado ou não. Os bits de requisição indicam que um segmento QA foi colocado na fila de espera para transmissão no outro barramento. Quando uma estação tem um segmento para ser transmitido, ela causa uma única requisição no barramento reverso.

Essa requisição se dá com a escrita(1) no bit, da prioridade correspondente, do primeiro slot do barramento reverso que contiver requisição livre (0). Esse bit vai servir para indicar a todas as estações anteriores à estação requisitante, que um segmento QA foi colocado na fila de transmissão e, que para tanto, precisa de um slot desocupado. Para cada estação, o protocolo permite o enfileiramento de no máximo um segmento QA por nível de prioridade, para cada barramento.

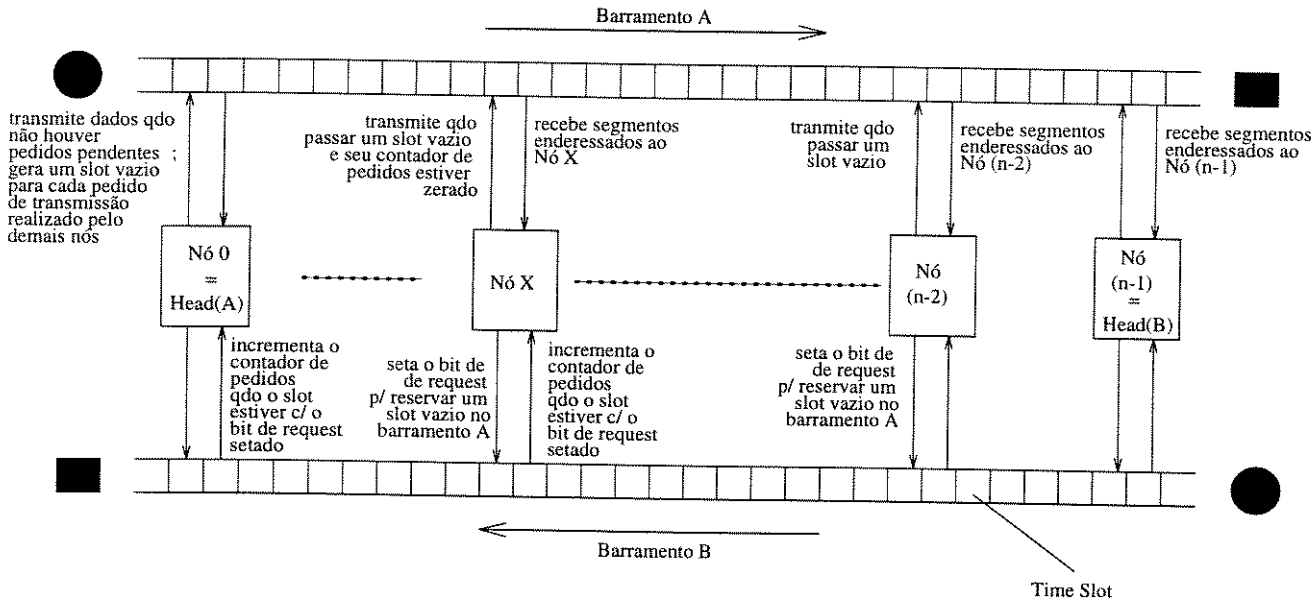

Figura 2.5: Operação básica do protocolo de filas distribuídas

2.3.2 Mecanismo dos Contadores

Cada estação mantém o número de requisições feitas por outras estações a sua frente(no sentido de transmissão do barramento), contando os bits de requisição à medida que passam no barramento reverso. Para uma estação que não tem nada enfileirado para transmitir, uma requisição no contador de requisições(*Request Counter(RQ)*) é cancelada, cada vez que um slot vazio é deixado passar à sua frente, no barramento de transmissão, uma vez que uma requisição à frente será atendida.

Quando uma estação quer transmitir um segmento QA, envia, na primeira oportunidade, o pedido de requisição no barramento reverso, e transfere o valor do RQ para um outro contador(*Countdown Counter(CD)*), que conterá o número de requisições de estações à sua frente(no sentido de transmissão do barramento) que deverão transmitir primeiro. Esse ato põe efetivamente o segmento QA em uma fila distribuída. A cada novo segmento que passa vazio na barra de transmissão, esse contador é decrementado. Uma estação só pode transmitir seu segmento QA quando esse contador chegar a zero. Cada nova requisição, no barramento reverso, incrementa o contador de requisições, como ilustrado na figura 2.6. Observamos que a transmissão de um segmento QA é controlada apenas pelos contadores. Nada impede que haja a transmissão antes do envio da requisição, ou seja, a operação de envio de um segmento QA e a de envio de requisição são independentes.

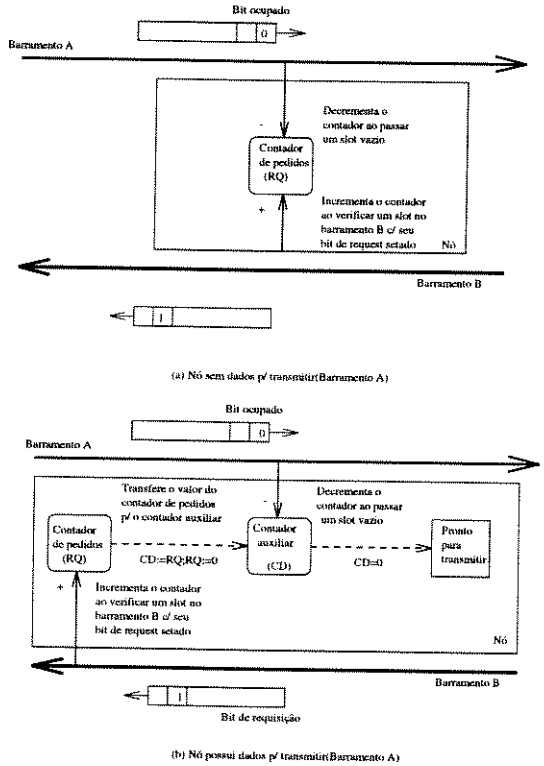

Figura 2.6: Operação dos contadores RQ e CD

2.3.3 Mecanismo de Prioridades das Filas Distribuídas

Como mencionamos, o protocolo das filas distribuídas suporta três níveis de prioridade, embora o padrão IEEE 802.6 não especifique o uso dos níveis de prioridades. O IEEE 802.6 define que o serviço com conexão não orientada opere com a mais baixa prioridade(nível 0). As demais prioridades deverão ser usadas posteriormente pelos serviços isócronos e orientados à conexão.

O suporte à prioridade é dado pelo uso de três bits separados para requisição e contadores RQ(i) e CD(i) separados para cada prioridade i, em cada barramento. Para as estações que não têm segmentos QA de prioridade i enfileirados para transmissão, os contadores RQ(i) devem contar as requisições, do barramento reverso, de prioridade maior ou igual a i. Se uma estação tem algum segmento QA enfileirado para transmissão em um nível de prioridade i, RQ(i) é transferido para CD(i) e partir de então o contador RQ(i) só contará as requisições do barramento reverso deste nível i. O contador CD(i) não só deve ser decrementado na passagem de cada slot vazio, mas também deve ser incrementado para requisições recebidas de nível maior que i. A estação só poderá transmitir um segmento QA de prioridade i quando CD(i) for igual a zero.

2.3.4 Mecanismo de Balanceamento da Largura de Banda Passante

O protocolo das filas distribuídas descrito anteriormente, possui um problema de equidade que se agrava à medida que o retardo de propagação entre duas estações cresce.

Em princípio, na rede DQDB, uma estação só transmite um segmento QA quando seu CD for zero e houver um slot vazio. Entretanto devido ao problema já citado anteriormente, um mecanismo de balanceamento da largura de banda passante foi incorporado ao protocolo.

O mecanismo de balanceamento da largura de banda passante estabelece que uma estação somente pode transmitir β segmentos QA de cada vez. Portanto, a estação é obrigada a deixar passar um slot vazio para as outras estações. Para implementar esse mecanismo em cada barramento, dois novos contadores(*trigger counter*) foram incorporados à estação para cada barramento. Esses contadores são incrementados de 1 a cada segmento QA transmitido. Quando esses contadores forem iguais a β, seus valores serão setados para 0 e seus RQ's incrementados de 1. O parâmetro β pode ser setado em cada uma das estações. O valor varia de 0 a 64, com o valor default de 8. O valor 0 desabilita o mecanismo.

O IEEE 802.6 recomenda que o valor do β seja proporcional ao comprimento do barramento e a carga média de informação de cada estação a ser transmitida a fim de atingir uma ótima eficiência de utilização do meio de transmissão.

2.4 Descrição Funcional de um Nó DQDB para Serviços com Conexão Não Orientada

A seguir é apresentado um detalhamento da funcionalidade da camada DQDB no suporte a serviços com conexão não orientada. Em princípio abordaremos as funções necessárias referentes a transmissão de um quadro LLC PDU(*Protocol Data Unit(PDU)*) de um nó DQDB para um ou vários nós DQDB e em seguida abordaremos no que diz respeito a recepção desses quadros.

Os quadros LLC PDU's são entregues ao nó de origem na forma de unidades de serviço de dados da camada MAC(*MAC Service Data Unit(MSDU)*) através da primitiva *MA-UNITDATA request*. Na figura 2.7 são ilustradas as funções de transmissão e recepção que são exigidas pelo bloco MCF(*MAC Convergence Function*) e pelo bloco QA(*Queue Arbitrated function*) transferirem MSDU's entre os nós. A MSDU é entregue para cada nó de destino através da primitiva *MA-UNITDATA indication*). A seguir comentaremos cada um dos blocos citados.

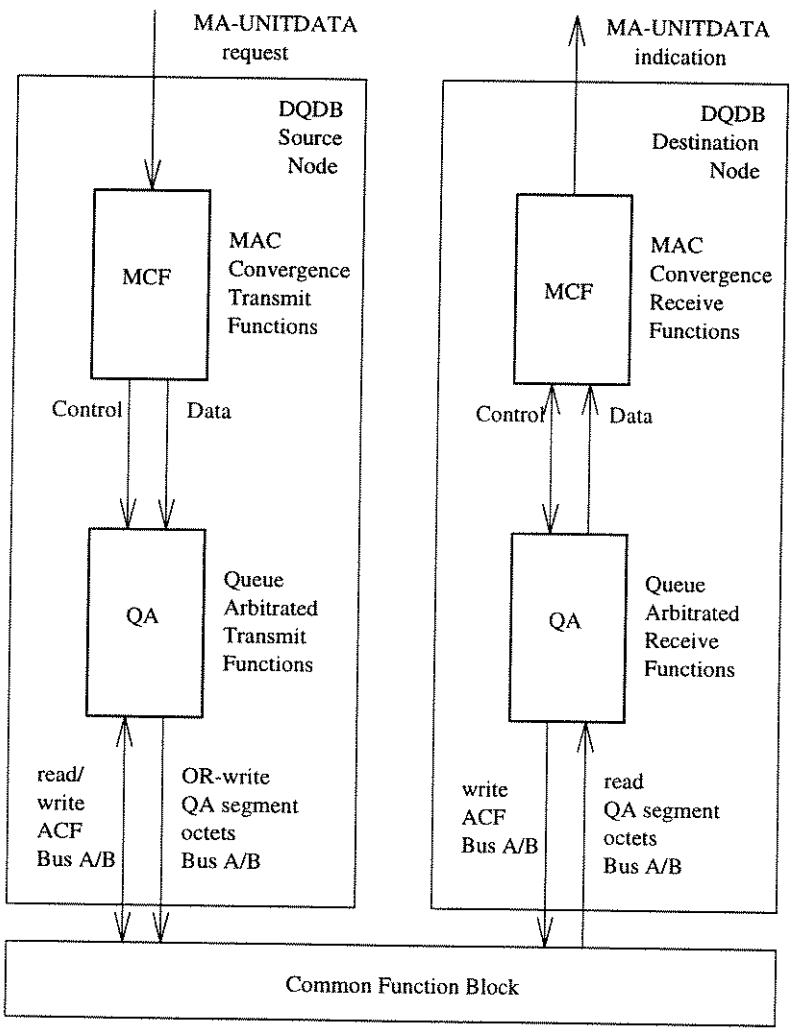

Figura 2.7: Provisão de serviço MAC para LLC pela camada DQDB

2.4.1 Bloco MCF para transmissão

Como é ilustrado na figura 2.8, o bloco MCF é composto por três funções que juntas providenciam um suporte a serviços com conexão não orientada. A seguir são descritas as funções.

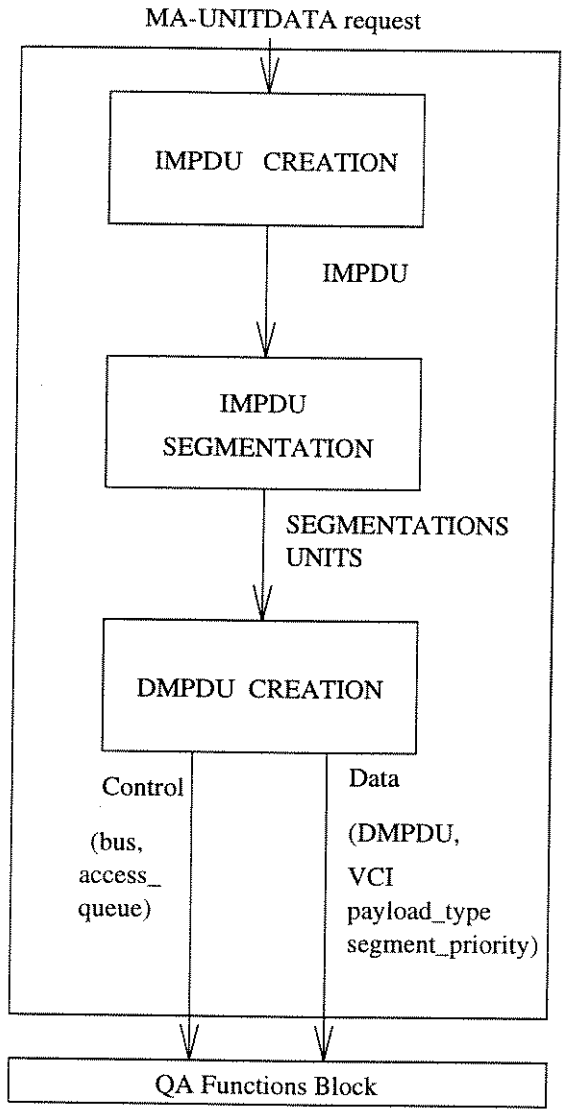

Figura 2.8: Funções de transmissão do bloco MCF

2.4.2 Formação da IMPDU

Ao receber a MSDU proveniente da camada LLC, o bloco MCF acrescenta um protocolo de controle de informação(PCI), formando dessa maneira, a unidade de dados inicial do protocolo MAC(IMPDU). A seguir observamos na figura 2.9, a formação de uma IMPDU e em seguida descrevemos cada um dos campos que fazem parte desse protocolo.

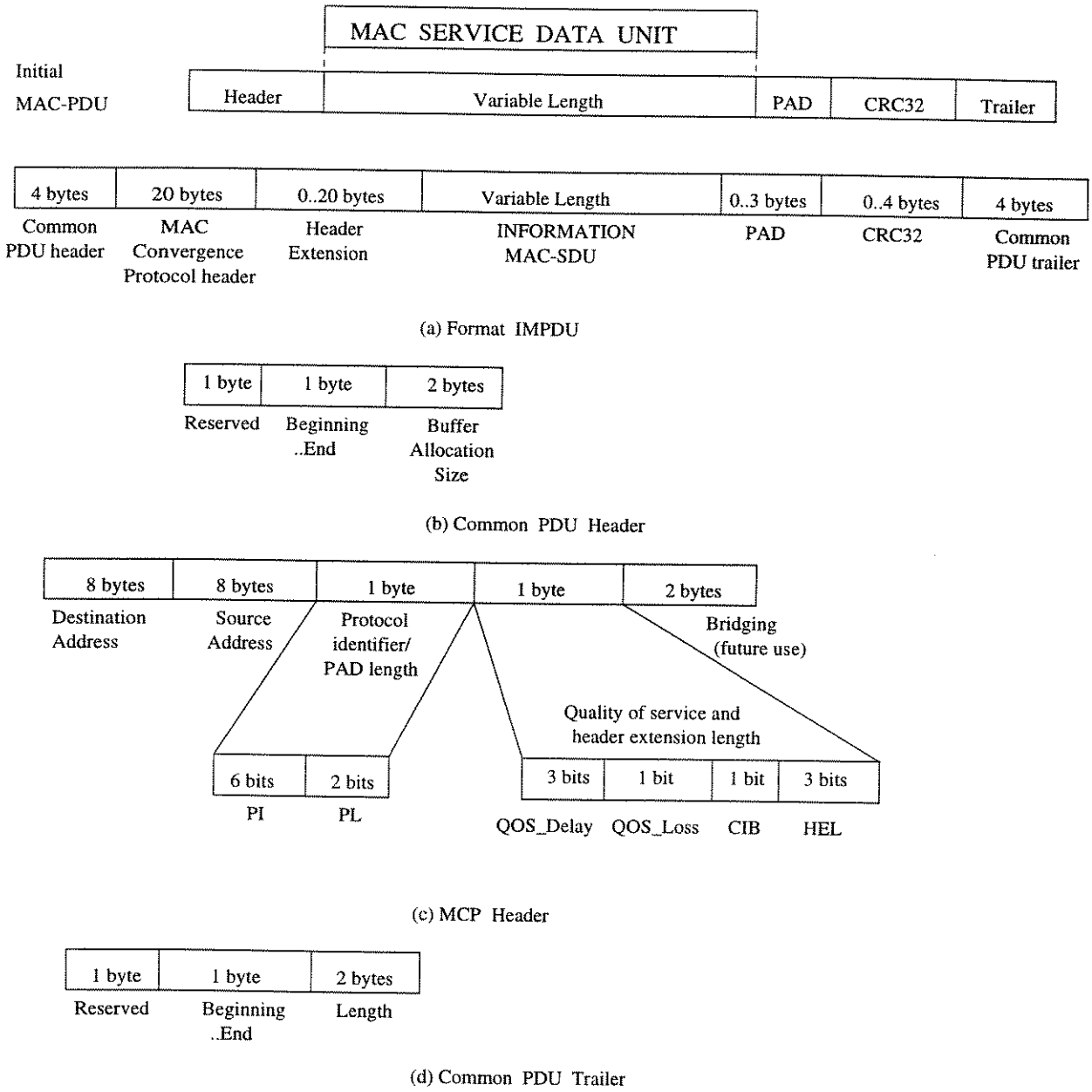

Figura 2.9: Unidade de dados inicial do protocolo MAC(IMPDU)

PCI para formação do campo *Common PDU Header*

1. O sub-campo reservado deverá ser setado para zero pelo bloco MCF durante a criação da IMPDU.

2. O valor do sub-campo BEtag(*Beginning-End tag*) é selecionado para cada IMPDU pelo bloco MCF. Para uma determinada IMPDU, o nó de origem deverá inserir o mesmo valor no sub-campo BEtag tanto no campo *Common PDU Header* quanto no campo *Common PDU Trailer*. Isto é feito independentemente do tamanho da IMPDU, tanto que os sub-campos BEtag em uma SSM DMPDU deverão ser enviados com o mesmo valor. O sub-campo BEtag permite a associação de BOM DMPDU com EOM DMPDU oriunda de uma mesma IMPDU. A associação é utilizada para detectar a perda de alguma(s) DMPDU's através da sequência das IMPDU's.

3. O sub-campo BAsize(*Buffer Allocation size*) deverá ser setado pelo bloco MCF com o valor do tamanho em bytes da soma dos campos: *MCP Header*, *Header extension*, *INFO*, *PAD* e *CRC32*(se houver) durante a criação da IMPDU. O nó origem deverá inserir o mesmo valor no sub-campo *Length* do campo *Common PDU Trailer* da IMPDU. Esse sub-campo informa a estação receptora do tamanho da fila necessária para completar o processo de remontagem.

PCI para formação do campo *MAC Convergence Protocol(MCP) Header*

1. O sub-campo DA(*Destination Address*) do campo *MCP Header* recebe o parâmetro *destination_address* encontrado na primitiva *MA-UNITDATA request.*

2. O sub-campo SA(*Source Address*) do campo *MCP Header* recebe o parâmetro *source_address* encontrado na primitiva *MA-UNITDATA request.*

3. O sub-campo PI(*Protocol Identification*) é usado para identificar para que protocolos o quadro MAC deve ser entregue.

faixa do PI	Entidade do protocolo
1	LLC
48-63	Reservado à administração local
outros valores	Reservado para padronizações futuras

4. O sub-campo PL(*PAD Length*) indica o número de bytes do campo PAD.

PL	n bytes
00	0 byte de PAD
01	1 bytes de PAD
10	2 bytes de PAD
11	3 bytes de PAD

5. Os 3 bits do sub-campo QOS_DELAY indicam a qualidade de serviço requesitada para uma IMPDU no que diz respeito ao atraso de acesso ao meio. O valor desse sub-campo é obtido através do parâmetro *priority* encontrado na primitiva *MA-UNITDATA request.*

Prioridade requisitada	Sub-campo QOS_DELAY	Nível do atraso
7	111	baixo
6	110	
5	101	
4	100	
3	011	
2	010	
1	001	
0	000	alto

6. O sub-campo QOS_LOSS está reservado, porém pode ser utilizado para controle de congestionamento nas *bridges* quando houver interconexão de sub-redes DQDB. Isto é, indicaria a qualidade de serviço requesitada por uma IMPDU no que diz respeito a prioriade de descarte de slots DQDB devido a um congestionamento no bloco MCF. Valor default é QOS_LOSS=0.

7. O sub-campo CIB(*CRC32 Indicator Bit*) indica a presença ou ausência do campo CRC32 na IMPDU. Para o bloco MCF, o valor default é CIB=0, isto é, não há campo CRC32.

8. Os 3 bits do sub-campo HEL(*Header Extension Length*) indicam o tamanho em bytes do campo *Header Extension* na IMPDU. O tamanho em bytes do *Header Extension* é obtido multiplicando-se o valor numérico do sub-campo HEL por 4. O valor decimal do sub-campo HEL varia de 0 a 5. Portanto, o campo *Header Extension* têm um tamanho de 0 a 20 bytes.

9. Os 2 bytes do sub-campo *BRIDGING* são reservados para uso futuro. Todos os bits deverão estar setados em 0. Este sub-campo pode ser utilizado posteriormente para indicar o número de *bridges* que a IMPDU pode trafegar. Portanto, a IMPDU poderia ser descartada quando um certo número fosse alcançado.

PCI para formação do campo HEL(*Header Extension*)

1. Quando o valor do sub-campo HEL for diferente de zero, o campo HEL providenciará uma capacidade adicional de protocolo para o transporte de IMPDU's.

PCI para formação do campo INFO

1. O campo INFO contém a MSDU que é obtida do parâmetro *data* da primitiva *MA-UNITDATA request*. O tamanho do campo INFO será no máximo de 9188 bytes para sub-campo PI=1.

PCI para formação do campo PAD

1. O número de bytes do campo PAD é tal que o campo INFO mais o campo PAD seja múltiplo de 4. O número de bytes do PAD pode variar de 0 a 3 bytes. Todos os bytes do PAD deverão ser setados em zero.

PCI para formação do campo CRC32

1. Se sub-campo CIB for igual a 1, a IMPDU terá o campo CRC32. O campo CRC32 controla erros nos campos: *MCP Header*, *Header Extension*, INFO e PAD.

PCI para formação do campo *Common PDU Trailer*

1. O sub-campo reservado deverá ser setado em zero pelo bloco MCF.
2. O sub-campo BEtag deverá ter o mesmo valor do sub-campo BEtag do *Common PDU Header*. Os dois sub-campos são utilizados em conjunto para detectar a perda de alguma(s) DMPDU's ao longo de uma sequência de IMPDU's.
3. O sub-campo *Length* deverá ser setado pelo bloco MCF com o valor do tamanho em bytes dos campos: *MCP header*, *Header extension*, INFO, PAD e CRC32(se houver). O sub-campo *Length* e o sub-campo BAsize do campo *Common PDU header* deverão ter o mesmo valor.

2.4.3 Segmentação da IMPDU

O bloco MCF após formado a IMPDU, deverá dividí-la em uma ou mais segmentações. Cada segmentação deverá conter no máximo 44 bytes, com exceção da última segmentação que poderá ter menos de 44 bytes. Os bytes não utilizados em uma segmentação deverão ser setados em zero. O número de segmentações dependerá do tamanho da IMPDU. A seguir é ilustrado na figura 2.10 a segmentação de uma IMPDU.

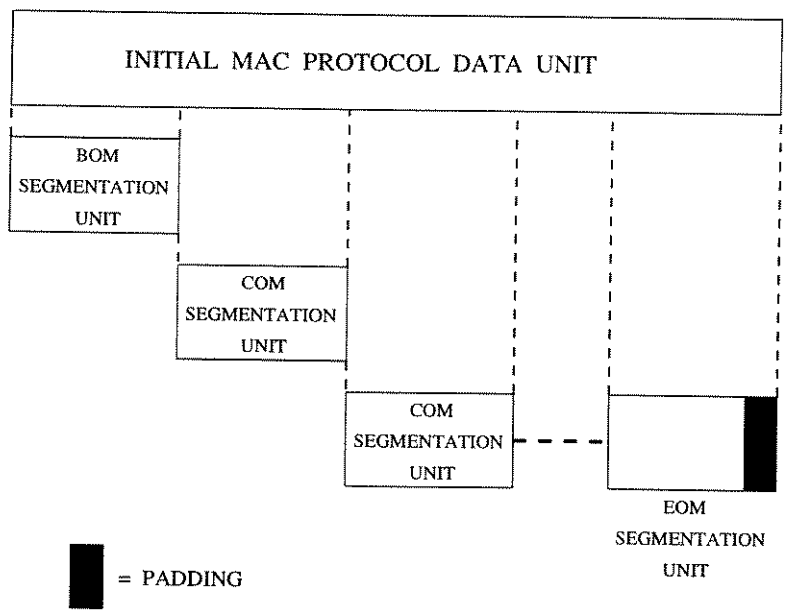

Figura 2.10: Segmentação de uma IMPDU com mais de 88 bytes

2.4.4 Formação das DMPDU's

Novamente um protocolo de controle de informação(PCI) é acrescentado em cada uma das unidades segmentadas da IMPDU para formar a *Derived MAC Protocol Data Unit(DMPDU)*. A seguir observamos na figura 2.11 a formação da DMPDU seguida da descrição de cada um dos campos que fazem parte desse protocolo.

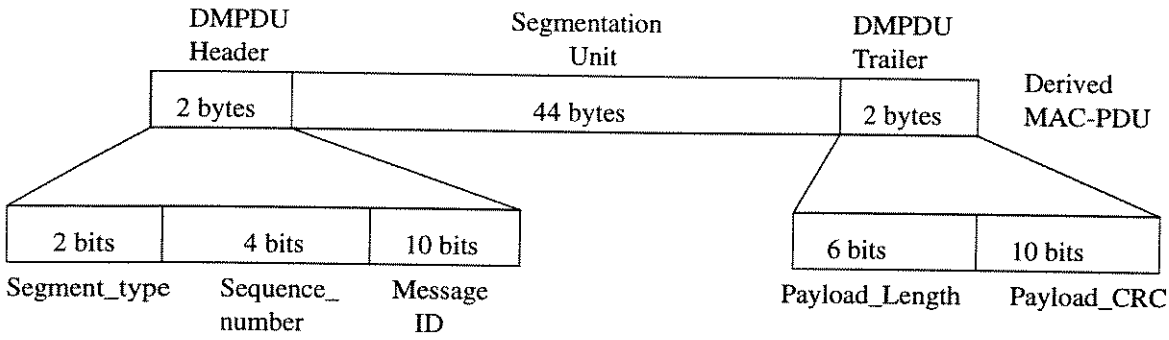

Figura 2.11: Formação de uma DMPDU

PCI para formação do campo *DMPDU header*

1. Os 2 bits do sub-campo ST(*Segment_Type*) indicam como a(s) unidade(s) segmentada(s) deverão ser processadas pelo bloco MCF de recepção. Os códigos para esse sub-campo são:

Tipo do segmento	Tipo da DMPDU
00	Continuação de mensagem(COM)
01	Fim de mensagem(EOM)
10	Começo de mensagem(BOM)
11	Único segmento de mensagem(SSM)

   Os primeiros 24 bytes das unidades segmentadas BOM DMPDU ou SSM DMPDU contêm os campos: *Common PDU header* e *MCP header* da IMPDU. Os próximos 20 bytes da IMPDU preencherão o restante da unidade segmentada.

2. O sub-campo SN(*Sequence Number*) é utilizado pelo processo de remontagem da IMPDU para verificar se todas as DMPDU's oriundas de uma mesma IMPDU foram recebidas. O valor do sub-campo SN deverá ser inicializado na transmissão de uma BOM DMPDU e incrementado a cada recebimento de uma COM DMPDU ou EOM DMPDU de uma mesma IMPDU. Esse sub-campo permite a estação receptora monitorar a sequência de DMPDU's recebidas e avaliar se houve perda de alguma DMPDU. Caso tenha havido perda, a estação receptora poderá decidir por encerrar o processo de remontagem e descartar as DMPDU's oriundas da mesma IMPDU.

3. Os 10 bits do sub-campo MID(*Message Identifier*) são utilizados pelo processo de remontagem das DMPDU's no nó de destino. O nó origem deverá enviar todas as DMPDU's de uma determinada IMPDU com o mesmo valor de MID escolhido pela entidade de gerenciamento da camada DQDB.

Em qualquer instante não poderá existir duas IMPDU's compartilhando um mesmo valor de MID. Todas as DMPDU's do tipo SSM deverão ter os bits do MID setados em zero.

PCI para formação do campo *DMPDU Trailer*

1. Os 6 bits do sub-campo *Payload Length* indicam o número de bytes da IMPDU que ocupam a DMPDU. Esse número é múltiplo de 4 e varia entre 4 e 44.

2. Os 10 bits do sub-campo *Payload CRC* possibilitam a detecção e correção de erros singulares na DMPDU, incluindo o campo *DMPDU header*, a carga útil da DMPDU e o sub-campo *Payload Length* do campo *DMPDU Trailer*.

2.4.5 Interações entre o bloco MCF e o bloco QA para transmissão

É importante ressaltar que as interações entre o bloco MCF e o bloco QA para a transferência de uma DMPDU como carga útil de um segmento QA dependem somente dos parâmetros recebidos na primitiva *MA-UNITDATA request*.

A seguir ilustramos na figura 2.12, os sinais necessários para o suporte a serviços não orientados à conexão.

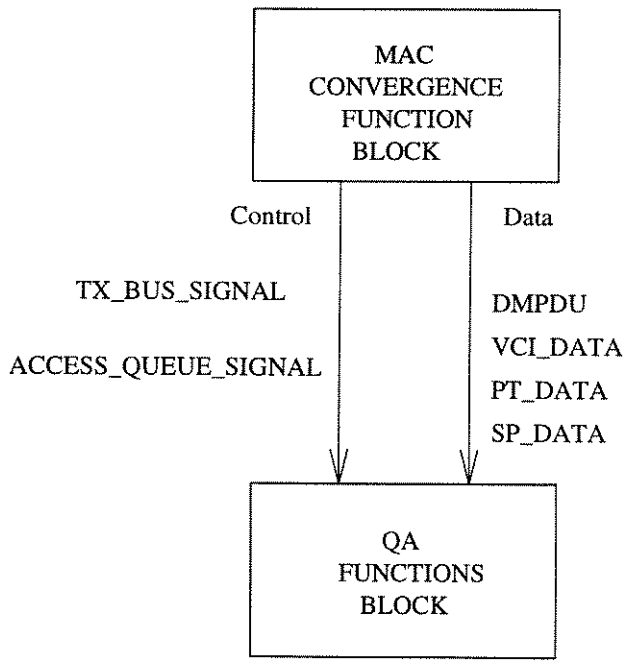

Figura 2.12: Interações entre o bloco MCF e o bloco QA para transmissão

1. O bloco MCF determina o valor apropriado para o sinal TX_BUS_SIGNAL apenas examinando o parâmetro *destination_address* da primitiva *MA-UNITDATA request* que gerou a IMPDU.

2. O bloco MCF determina o valor apropriado para o sinal ACCESS_QUEUE_SIGNAL apenas examinando o parâmetro *priority* da primitiva *MA-UNITDATA request* que gerou a IMPDU. Esse valor será usado pelo bloco QA para acessar o meio na prioridade desejada.

3. O sinal VCI_DATA contém o valor do sub-campo VCI do cabeçalho de cada segmento QA usado para transferir as DMPDU's. O valor do VCI correspondente a todos os bits setados em 1 é o valor default para serviços não orientados à conexão.

4. O sinal PT_DATA contém o valor do sub-campo *Payload_Type* do cabeçalho de cada segmento QA usado para transferir DMPDU's. Esse sinal deverá ser setado em zero quando o valor do VCI for o default.

5. O sinal SP_DATA contém o valor do sub-campo *Segment_Priority* do cabeçalho de cada segmento QA usado para transferir DMPDU's. Esse sinal deverá ser setado em zero quando o valor do VCI for o default.

2.4.6 Bloco QA para transmissão

A principal função do bloco QA é controlar a transferência de DMPDU's geradas pelo bloco MCF em segmentos QA. A figura 2.13 ilustra os diversos processos incluídos nesse bloco.

2.4.7 Formação do segmento QA

Para a formação do segmento QA é acrescentado um PCI na DMPDU, isto é, cada DMPDU é encapsulada com 4 bytes de cabeçalho. O preenchimento dos campos do cabeçalho é obtido através dos sinais enviados pelo bloco MCF como observado na figura 2.13. A seguir descrevemos os passos necessários para formação de um segmento QA.

1. O valor do sinal VCI_DATA é transferido para o sub-campo VCI do cabeçalho do segmento QA.

2. O valor do sinal PT_DATA é transferido para o sub-campo *Payload_Type* do cabeçalho do segmento QA.

3. O valor do sinal SP_DATA é transferido para o sub-campo *Segment_Priority* do cabeçalho do segmento QA.

4. É calculado o valor do sub-campo HCS do cabeçalho do segmento QA.

Após sua formação, o segmento QA(52 bytes) está pronto para ser enviado pelo barramento indicado pelo sinal TX_BUS_SIGNAL com a prioridade de acesso indicada pelo sinal ACCESS_QUEUE_SIGNAL.

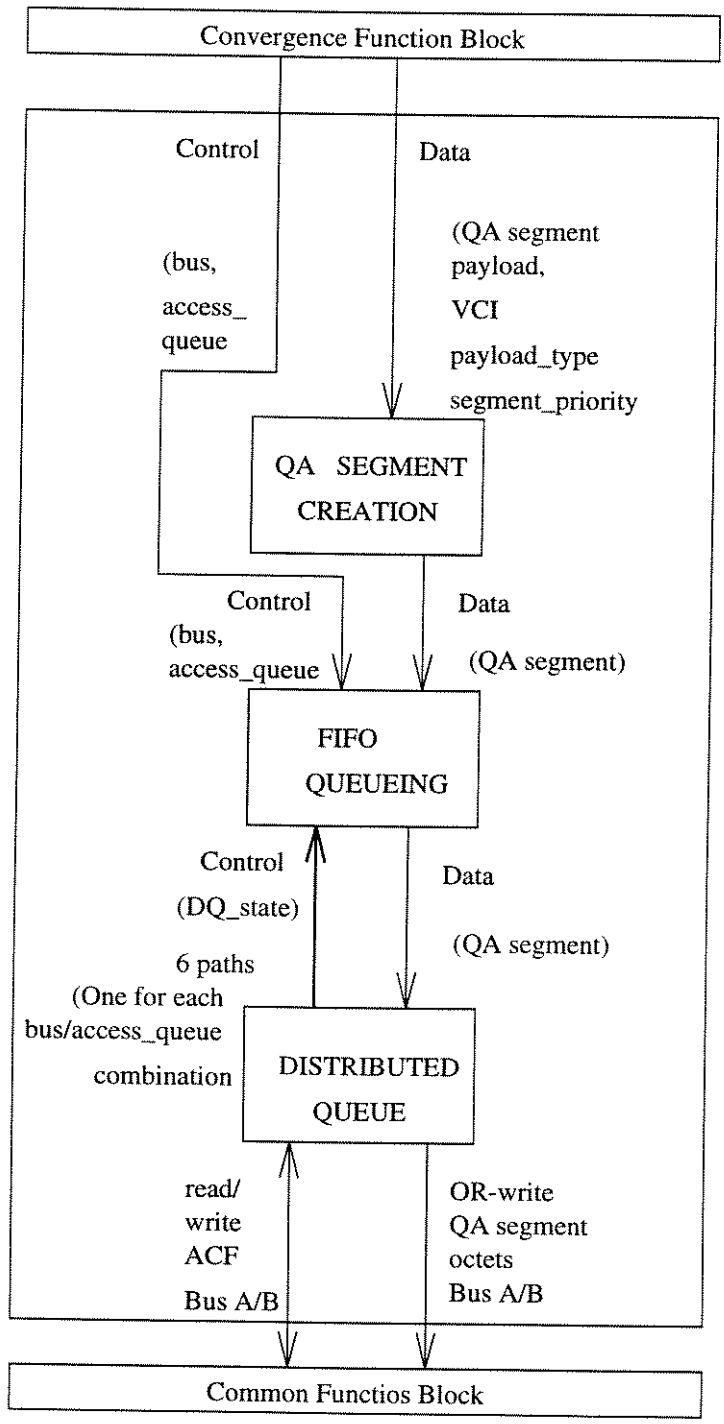

Figura 2.13: Funções de transmissão do bloco QA

O cabeçalho do segmento QA, apresentado na figura 2.14d, contém:

- 20 bits para o campo VCI(*Virtual Channel Identifier*) que identifica o canal virtual, ou seja a conexão lógica a qual pertence os segmentos. Diferentes valores de VCI são usados para os diversos serviços : serviços não orientados à conexão, serviços orientados à conexão e serviços isócronos.
- 2 bits para o campo *Payload_Type*, usado para declarar o tipo de informação que está sendo transmitida: dados, sinalização de rede, ou informação de gerenciamento.
- 2 bits para o campo *Segment_priority*, reservados para uso futuro. Esse campo poderá ser utilizado para definir o nível de prioridade de descarte de slots DQDB quando houver interconexão de sub-redes DQDB.
- 8 bits para o campo CRC que ajudam a detectar e corrigir erros ocorridos dentro do cabeçalho do segmento QA.

A seguir é apresentado na figura 2.14c, o formato do segmento QA seguido do formato do slot DQDB (fig. 2.14a).

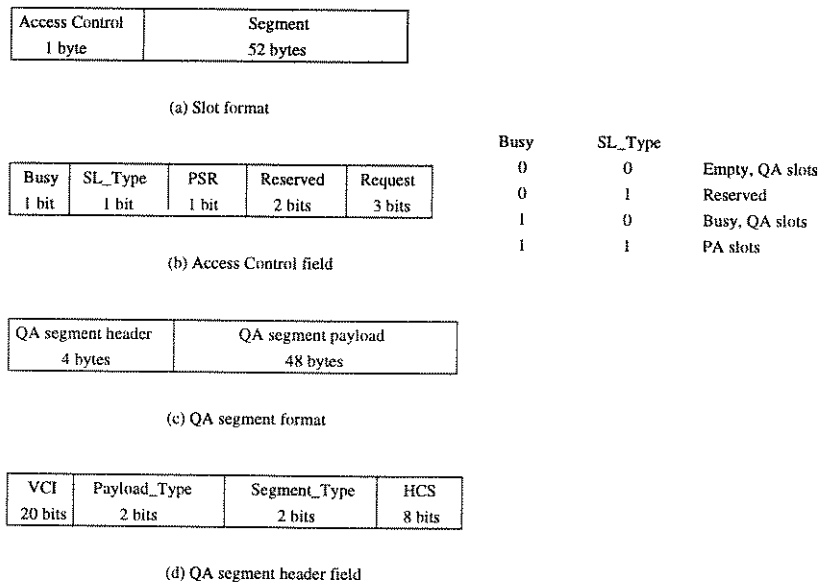

Figura 2.14: Formato do slot DQDB, formato do campo de controle de acesso, formato do segmento QA, formato do cabeçalho do segmento QA

2.4.8 Filas FIFO para os segmentos QA

Após a formação dos segmentos QA, esses segmentos são bufferizados em filas do tipo FIFO associadas com o valor dos sinais TX_BUS_SIGNAL e ACCESS_QUEUE_SIGNAL. Existem no máximo 6 filas dentro de cada nó, uma para cada combinação do TX_BUS_SIGNAL e ACCESS_QUEUE-_SIGNAL. Portanto, o bloco QA pode bufferizar múltiplos segmentos QA aguardando o acesso de um determinado barramento com um certo nível de prioridade.

2.4.9 Filas FIFO distribuídas

Paralelamente à bufferização dos segmentos QA, ocorre o controle do acesso ao meio para escrita dos segmentos QA(52 bytes) dentro de slots(53 bytes) vazios. Esse controle é realizado através dos mecanismos de acesso ao meio, citados anteriormente na seção 2.3.

2.4.10 Interações entre o bloco QA e o bloco de funções comuns para transmissão

As interações entre o bloco QA e o bloco de funções comuns como observados na figura 2.15 se limitam a leitura e escrita do sub-campo *Busy* do campo ACF do slot DQDB e a escrita do segmento QA dentro da carga útil do slot DQDB quando a leitura do sub-campo indica que o slot está vazio.

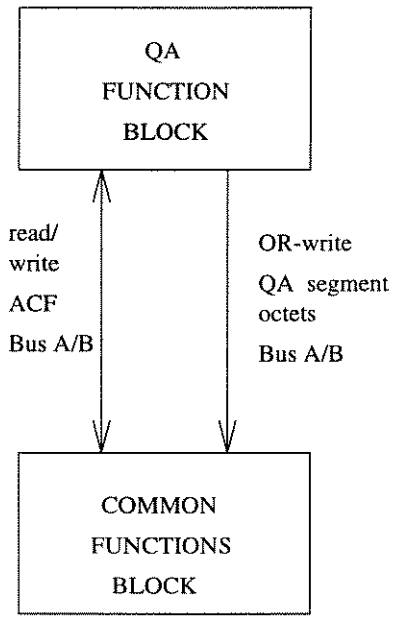

Figura 2.15: Interações entre o bloco QA e o bloco de funções comuns para transmissão

2.4.11 Bloco de funções comuns

Além das funções já citadas na seção 2.3, o bloco de funções comuns se encarrega de providenciar um serviço para o bloco QA de leitura e escrita nos campos do slot DQDB.

2.4.12 Interações entre o bloco de funções comuns e o bloco QA para recepção

Quando o bloco de funções comuns recebe um slot com o sub-campo *Busy* setado em 1 e o sub-campo *SL_Type* setado em zero, esse bloco deve fazer a cópia dos bytes do segmento QA e os enviar para o bloco QA.

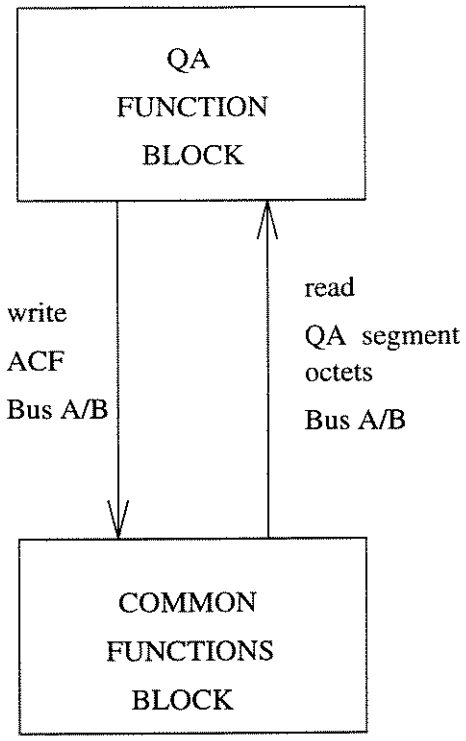

Figura 2.16: Interações entre o bloco de funções comuns e o bloco QA para recepção

Se o bloco QA é notificado pelo bloco MCF que o segmento foi destinado somente para esse nó, o bloco de funções comuns, a pedido do bloco QA, deve imediatamente setar o sub-campo *PSR bit* para 1 no campo ACF do próximo slot que passar por esse nó, independentemente do valor do sub-campo *SL_Type*. Na figura 2.16 é ilustrado as interações entre esses blocos.

2.4.13 Bloco QA para recepção

Como observamos na figura 2.17, o bloco QA é composto por três processos que juntos providenciam um serviço com conexão não orientada. A seguir comentamos cada um desses processos.

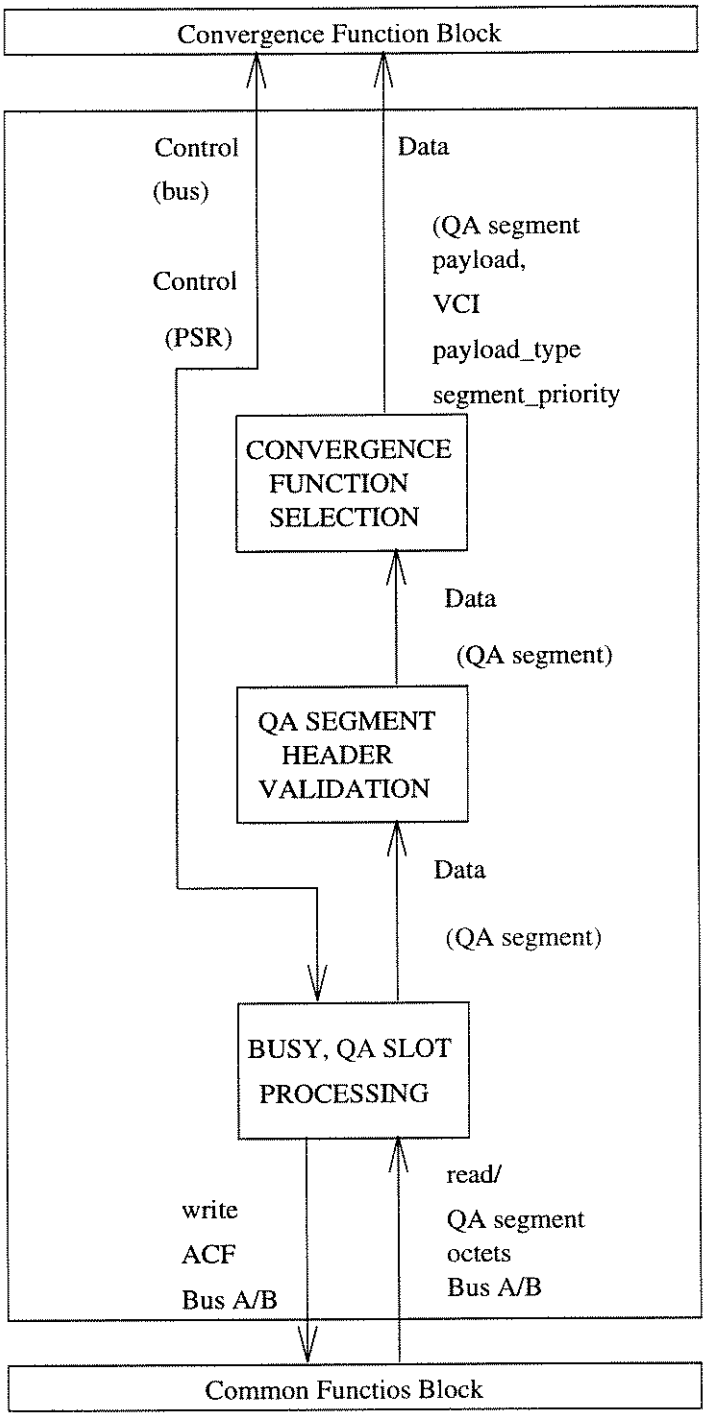

Figura 2.17: Funções de recepção do bloco QA

2.4.14 Processamento dos segmentos QA

Este processo é responsável por bufferizar os segmentos QA recebidos do bloco de funções comuns e os enviar para o processo de validação do cabeçalho do segmento QA. Também, esse processo deve requisitar do bloco de funções comuns a pedido do bloco MCF que o sub-campo PSR do campo ACF do próximo slot que passar por esse nó seja setado em 1. Isso é realizado quando o bloco MCF identifica que o segmento QA recebido foi endereçado somente a esse nó.

2.4.15 Validação do cabeçalho do segmento QA

O bloco QA utiliza o sub-campo HCS do cabeçalho do segmento QA para detectar erros no cabeçalho desse segmento. Caso haja erros, o HCS poderá ser ou não usado para correção; se não for usado, o segmento QA será descartado. Se não houver erros, o segmento QA é enviado imediatamente para o processo de seleção da função de convergência.

2.4.16 Seleção da função de convergência

Após a validação do cabeçalho, o bloco QA extrai o valor do sub-campo VCI do cabeçalho do segmento QA e verifica para qual função de convergência, a carga útil desse segmento QA, isto é, a própria DMPDU deve ser enviada.

2.4.17 Interações entre o bloco QA e o bloco MCF para recepção

As interações entre o bloco QA e o bloco MCF no que diz respeito a recepção das DMPDU's como carga útil de segmentos QA são ilustradas na figura 2.18 e descritas abaixo.

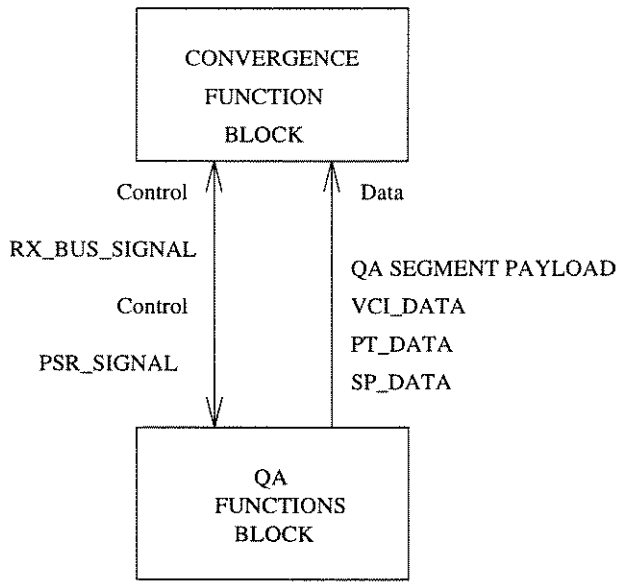

Figura 2.18: Interações entre o bloco QA e o bloco MCF para recepção

1. O sinal RX_BUS_SIGNAL é ativado pelo bloco QA para indicar em que barramento a DMPDU foi recebida.

2. O sinal PSR_SIGNAL é ativado pelo bloco MCF para indicar que a DMPDU recebida no barramento indicado pelo sinal RX_BUS_SIGNAL foi destinado somente para este nó.

3. O sinal VCI_DATA contém o valor do sub-campo VCI do cabeçalho do segmento QA recebido.

4. O sinal PT_DATA contém o valor do sub-campo *Payload_Type* do cabeçalho do segmento QA recebido.

5. O sinal SP_DATA contém o valor do sub-campo *Segment_Priority* do cabeçalho do segmento QA recebido.

2.4.18 Bloco MCF para recepção

A seguir como podemos verificar na figura 2.19, o bloco MCF é responsável pelas seguintes funções: receber as DMPDU's validadas pelo bloco QA, remontá-las, validar a unidade de dados remontada(IMPDU) e por último extrair da IMPDU, a MSDU que deve ser entregue a sub-camada LLC através da primitiva *MA-UNITDATA indication.*

2.4.19 Máquina de estados de remontagem

Como DMPDU's de diferentes IMPDU's podem chegar de maneira aleatória em um determinado nó, o bloco MCF inclue funções que suportam a operação simultânea de múltiplos processos de remontagem, cada qual controlado por uma máquina de estados de remontagem. Essa máquina é unicamente associada ao par VCI/MID.

Quando o bloco MCF recebe uma DMPDU do bloco QA com um valor de VCI para serviços sem conexão, esse bloco verifica a integridade dessa DMPDU através do sub-campo *Payload_CRC* do campo *DMPDU Trailer.*

Caso não seja detectado erros na DMPDU, o valor do MID é analisado, e se confirmado, o sub-campo DA também é examinado. Se confirmado o DA, o bloco MCF extrai a IMPDU dos N bytes(onde N é o valor do sub-campo *Payload_Length* do campo *DMPDU Trailer*) da DMPDU e repassa a IMPDU para o processo de validação. Em qualquer um dos casos acima, se a verificação for negativa, a DMPDU é descartada.

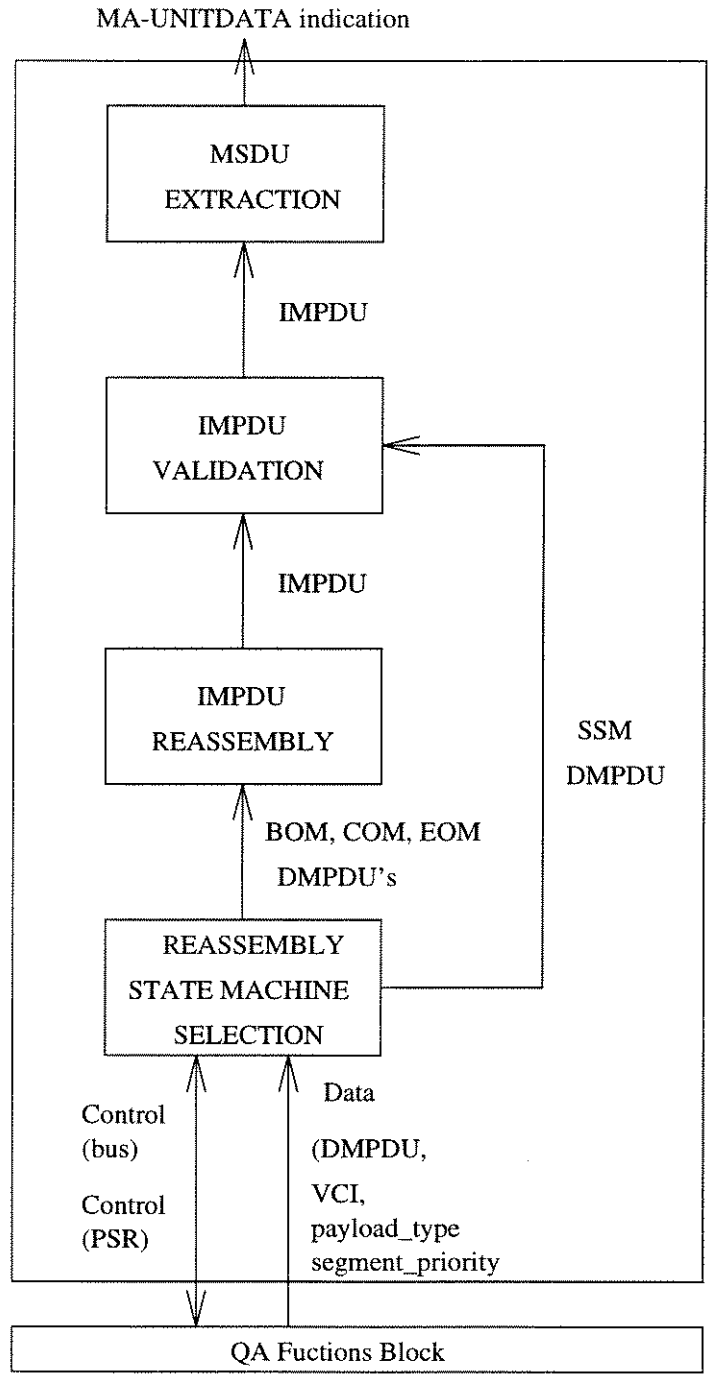

Figura 2.19: Funções de recepção do bloco MCF

2.4.20 Validação da IMPDU

O processo de validação das IMPDU's é realizado em quatro etapas.

1. O valor do sub-campo *Length* do campo *Common PDU Trailer* é comparado com o número de bytes recebidos para a IMPDU.

2. O valor do sub-campo *BEtag* do campo *Common PDU Header* é comparado com o sub-campo BEtag do campo *Common PDU Trailer.*

3. Se o sub-campo CIB se encontra setado em 1 na IMPDU, então, o campo CRC32 é usado para detectar erros nos campos: *MCP header*, *Header extension*, INFO e PAD. O valor recebido no campo CRC32 é comparado com o valor calculado no nó receptor.

4. O valor do sub-campo HEL do campo *MCP Header* é examinado. Um valor fora da faixa de 0 a 5 é considerado como inválido.

Se todas as quatro operações acima forem satisfeitas, a IMPDU é enviada para o processo de recuperação da MSDU. Caso falhe algumas dessas, a IMPDU é descartada. O processo de validação das IMPDU's deve enviar as IMPDU's para o processo de recuperação na mesma ordem que as recebeu.

2.4.21 Recuperação da MSDU

O processo de recuperação da MSDU executa as seguintes operações para criar os parâmetros utilizados na primitiva *MA-UNITDATA indication.*

1. Utiliza o valor do sub-campo DA do campo *MCP header* para criar o parâmetro *destination_address.*

2. Utiliza o valor do sub-campo SA do campo *MCP header* para criar o parâmetro *source_address.*

3. Utiliza o valor do sub-campo *QOS_DELAY* do campo *MCP header* para criar o parâmetro *priority.*

4. Extrai a MSDU contida no campo INFO para criar o parâmetro *data.*

O processo de recuperação da MSDU deve gerar as primitivas *MA-UNITDATA indication* na mesma ordem que recebeu as IMPDU's.

A seguir é apresentado na figura 2.20 e 2.21, a hierarquia das unidades de dados do protocolo DQDB para serviços com conexão não orientada.

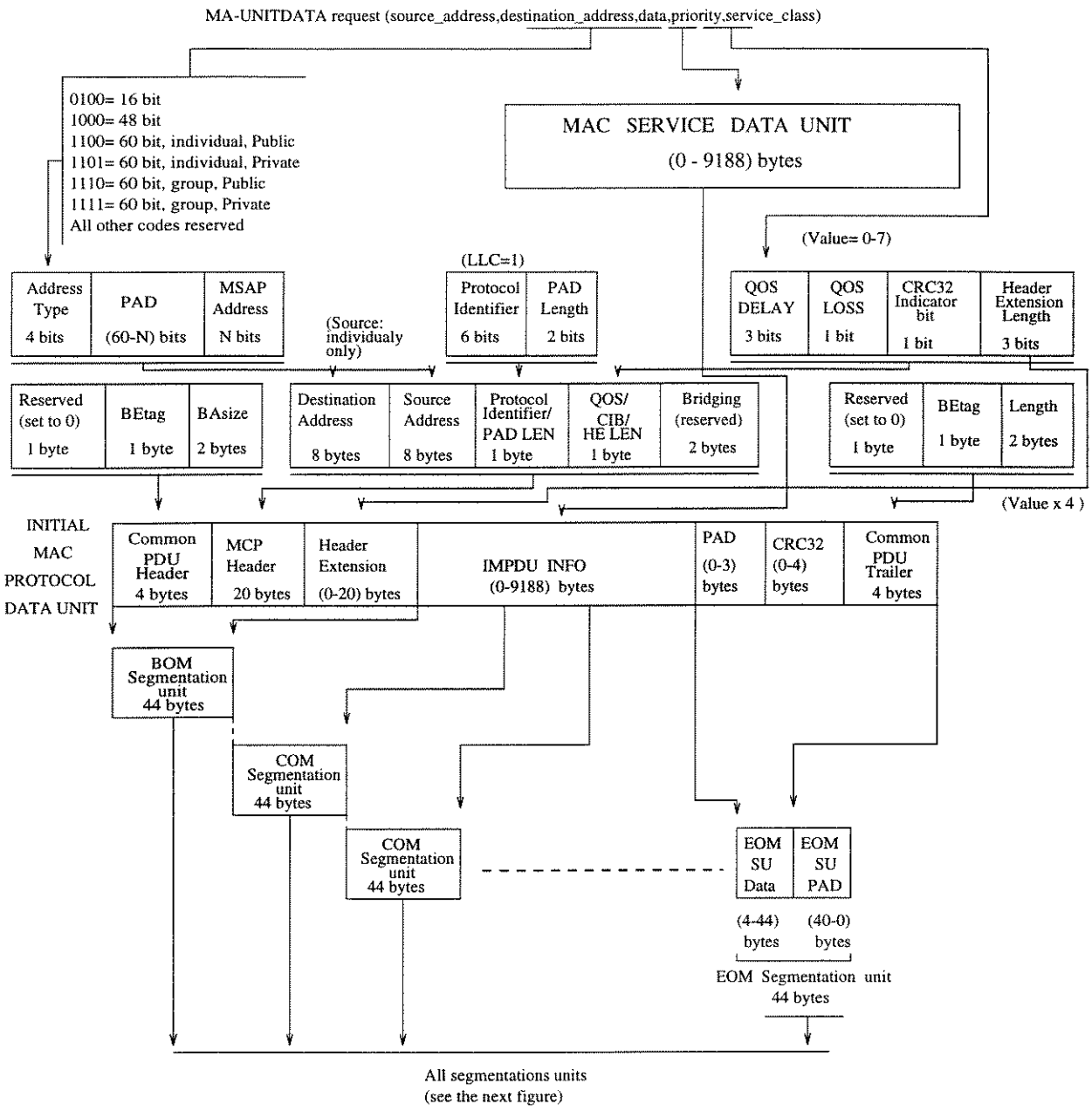

Figura 2.20: Hierarquia das unidades de dados do serviço sem conexão

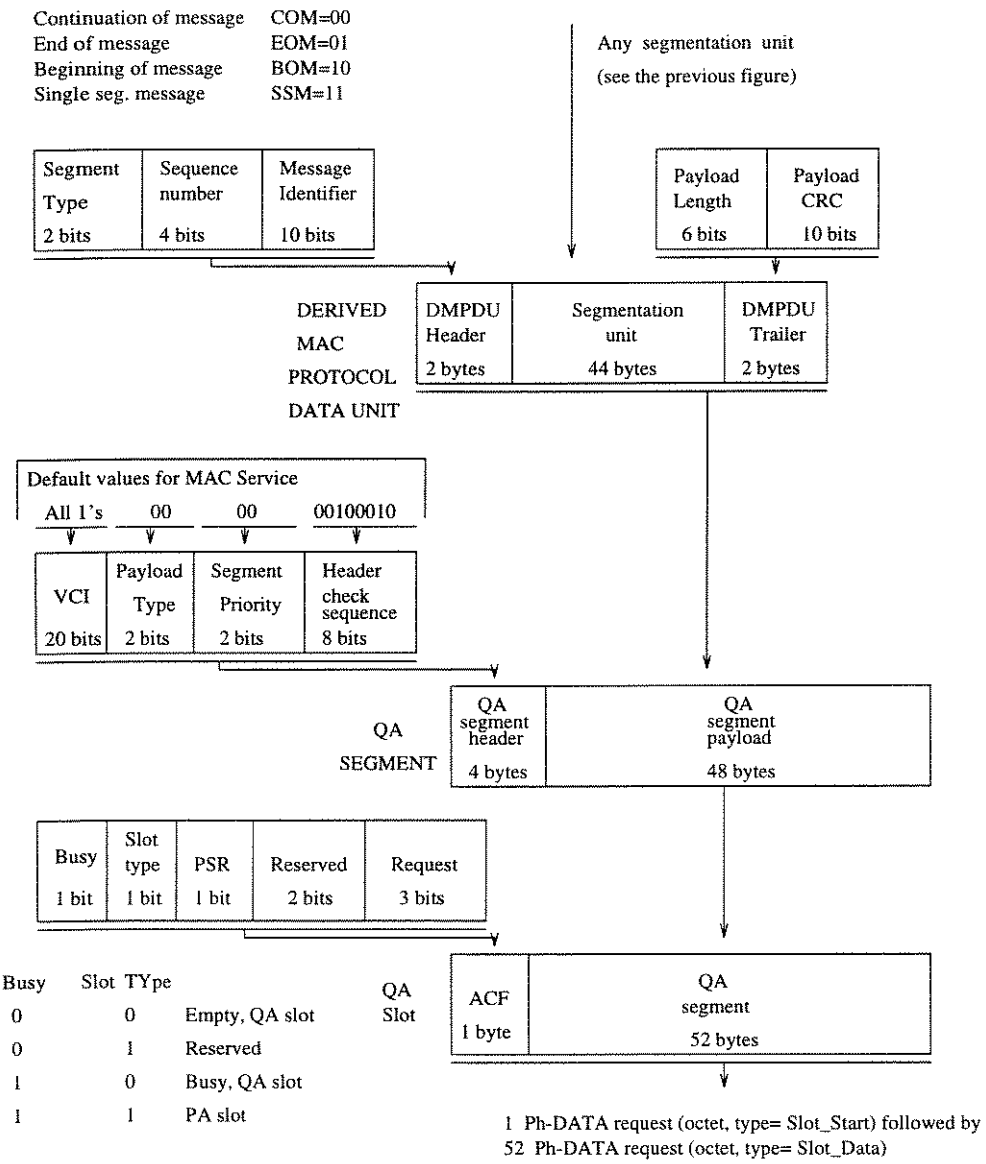

Figura 2.21: Hierarquia das unidades de dados do serviço sem conexão(cont.)

2.5 Conclusão

Nesse capítulo foram apresentadas as características gerais de uma rede DQDB, enfocando aspectos de topologia, controle de acesso ao meio e as características de um nó DQDB quanto à forma de acesso ao meio. Essas informações são fundamentais para os capítulos que seguem, permitindo a simulação da interconexão de redes locais DQDB através de redes ATM.

Capítulo 3

Suporte ao Serviço com Conexão Não Orientada na Interconexão de Redes DQDB/ATM

O fato da Rede Digital de Serviços Integrados de Faixa Larga(B-ISDN) ter sido padronizada pelo CCITT para suportar a demanda dos serviços atuais e futuros basear na técnica de multiplexação e comutação ATM que opera no modo orientado à conexão associado com a necessidade de interconectar LAN's e MAN's geograficamente dispersas que operam no modo não orientado à conexão, resultou na especificação pelo CCITT, do protocolo CLNAP[1] (*Connectionless Network Access Protocol*) [8] que atua , segundo o modelo de referência, em uma camada de nível superior a AAL 4, no plano de usuário. A principal função executada pelo protocolo CLNAP é o roteamento das unidades de dado do usuário de um serviço não orientado à conexão.

3.1 Cenários para o suporte ao serviço não orientado à conexão

Dois cenários são previstos pelo CCITT para o suporte ao serviço não orientado à conexão na B-ISDN [9].

3.1.1 Cenário Indireto

No cenário indireto apresentado na figura 3.1 , a B-ISDN oferece apenas a infraestrutura de conexões ATM semipermanentes, transparente ao serviço demandado, para o transporte das unidades de dados do usuário. Isto é, todo o processamento de dados referentes ao serviço não orientado à conexão é realizado fora da rede ATM. Portanto, os elementos adaptadores de protocolo IWU (*Interworking Unit*) são interconectados diretamente através de conexões semipermanentes, ficando assim responsáveis pela execução das funcionalidades do protocolo CLNAP.

[1] Atualmente, estudos estão sendo realizados para a definição do protocolo CLNIP(*Connectionless Network Interface Protocol*) [17] para a interface de rede. Desta forma, este trabalho pretende generalizar o uso do protocolo CLNAP para as interfaces de usuário e de rede.

Esse cenário traz dois inconvenientes que se agravam a medida que o número "N" de elementos IWU que se deseja comunicar, aumenta. O primeiro é que a rede deve manter e tratar "N*(N-1)/2" conexões semipermanentes, e o segundo é que as tabelas de roteamento estão distribuídas nos elementos IWU, tornando complexo o controle de consistência dessas.

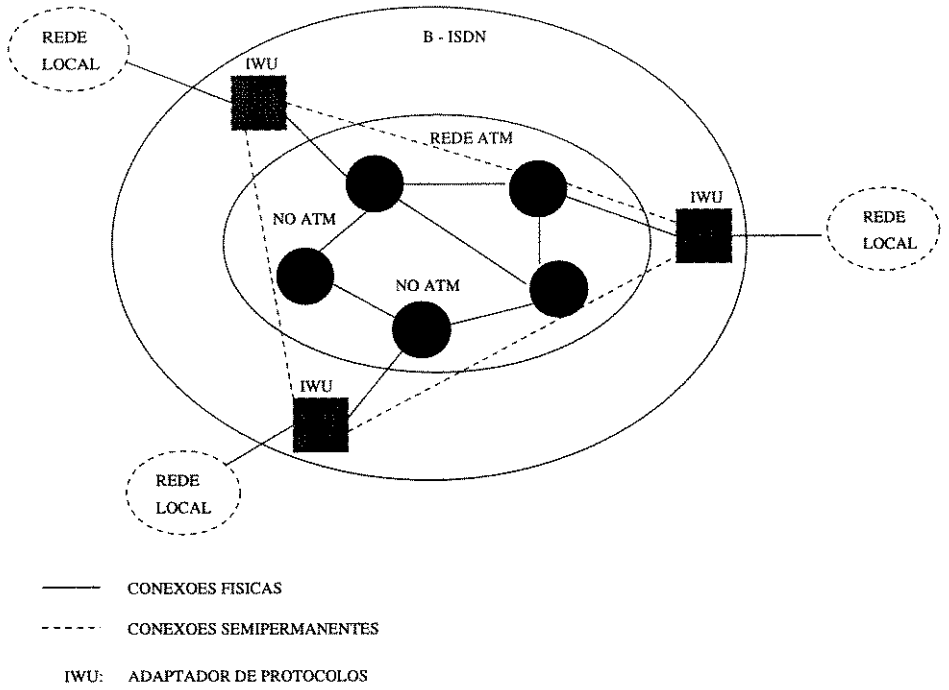

Figura 3.1: Cenário indireto

3.1.2 Cenário Direto

No cenário direto apresentado na figura 3.2, os elementos IWU são interconectados entre si através de um ou mais elementos servidores CLS (*Connectionless Server*), externos a B-ISDN, que executam a função de roteamento das unidades de dado. Portanto, a conexão semipermanente de acesso do elemento IWU à rede ATM fica designada exclusivamente ao serviço não orientado à conexão. Os elementos CLS são implementados adicionando-se as funcionalidades das camadas CLNAP e AAL 4 ao nó ATM.

Esse cenário pode ser estratificado em dois planos, onde um representa a rede virtual não orientado à conexão e o outro a rede de transporte ATM.

O cenário direto possui as seguintes vantagens em relação ao indireto:

- O servidor, pelo qual o elemento IWU tem acesso à rede, pode atuar como filtro de rede, descartando as células que compõem a unidade de dado do usuário, por exemplo, no caso de detecção de célula inválida, fora de sequência, ou com informações de endereçamento de origem e de destino incompatíveis. Portanto, essa função do servidor pode evitar a propagação de tráfego desnecessário ao longo da rede.

- Diminuição do tráfego a ser suportado pela rede ATM quando se deseja encaminhar a mesma unidade de dado do usuário para mais de um destino.

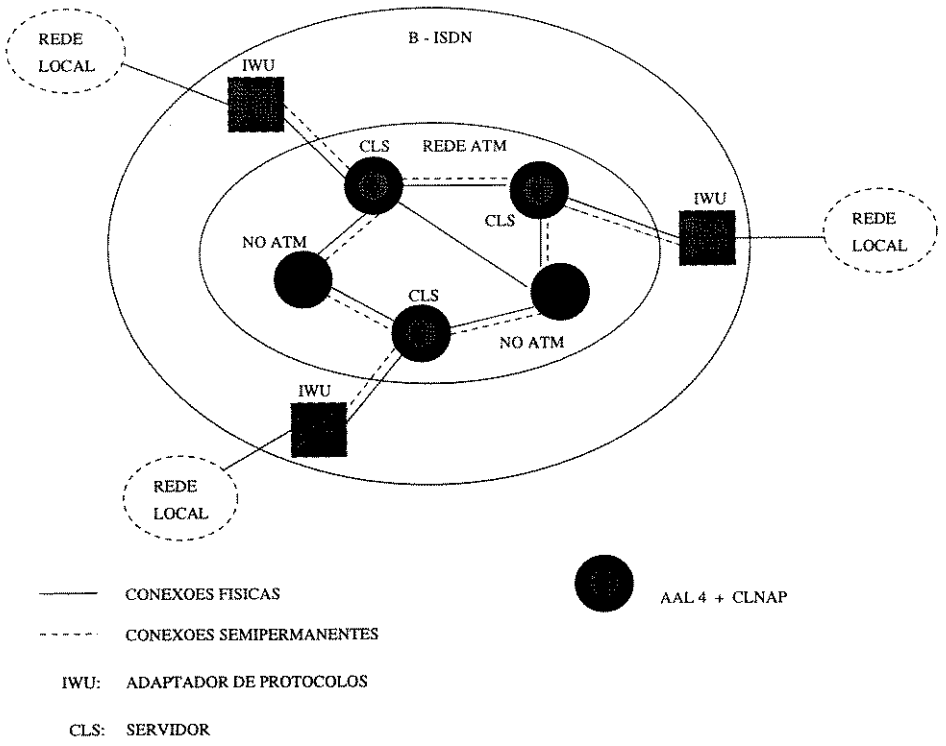

Figura 3.2: Cenário direto

Essa diminuição acontece porque o servidor só replica a unidade de dado em pontos específicos da rede, determinados em função dos endereços dos usuários de destino e da topologia da rede virtual.

- A rede virtual não orientada à conexão fornece intrinsecamente a opção de rotas alternativas, que podem ser devidamente utilizadas em situações de congestionamento, ou de falhas da rede.

3.2 Elementos de rede

A arquitetura funcional dos elementos de rede envolvidos no suporte do serviço não orientado à conexão na B-ISDN é apresentada a seguir:

3.2.1 Elemento Adaptador de Protocolos (IWU)

O elemento IWU pode ser implementado como uma *bridge*, ou como um *router*, dependendo do perfil das redes que se desejam interconectar.

"Bridge" : Esta implementação permite a extensão do serviço de uma rede local a outras remotas, com endereçamento e tamanho de dado compatíveis. Na *bridge* o encaminhamento das unidades de dado entre terminais locais ou remotos, é realizado através do endereço físico dos terminais.

"Router" : Esta implementação se aplica no caso da comunicação entre terminais associados a redes locais não homogêneas, ou seja, que operam com endereçamento e tamanho de unidades de dado não compatíveis. Neste caso, um protocolo de nível de rede(nível 3 do modelo OSI) é adicionado à estrutura do IWU para executar as funções de compatibilização entre os formatos de endereço e do tamanho das unidades de dado entre as redes locais. Notar que processos de segmentação e remontagem de quadros podem ocorrer nesse tipo de implementação.

3.2.2 Elemento Servidor (CLS)

O elemento servidor CLS possui como funcionalidade básica a execução do protocolo CLNAP. Este pode ser implementado integrando-se as funções dos protocolos CLNAP e AAL 4 em um único nó ATM, ou através de um equipamento independente atuando exclusivamente como servidor.

A figura 3.3 apresenta a estrutura de protocolos do elemento servidor[2].

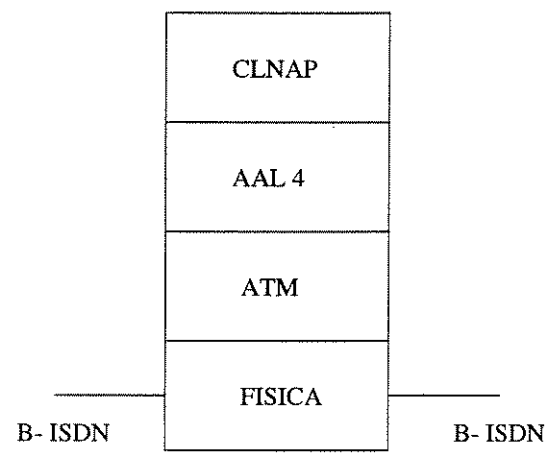

Figura 3.3: Estrutura de protocolos do servidor

3.3 Modelo de interconexão DQDB/ATM

Aproveitando a semelhança entre a estrutura de dado SAR-PDU pertencente a camada de adaptação AAL 4 definida para prover um serviço não orientado à conexão na rede ATM, com a estrutura de dado DMPDU da rede DQDB, utilizamos o modelo de interconexão apresentado na figura 3.4, inspirado no trabalho de Vernieris at al [15].

Considerando o modelo de interconexão apresentado na figura 3.4, implementamos o elemento IWU como *bridge* por duas razões:

- Não há necessidade de processos de segmentação e remontagem pelo fato que as duas redes utilizam estruturas de dados com tamanhos compatíveis.

[2]Maiores detalhes sobre a implementação completa do elemento CLS podem ser encontrados nos trabalhos [17] e [18].

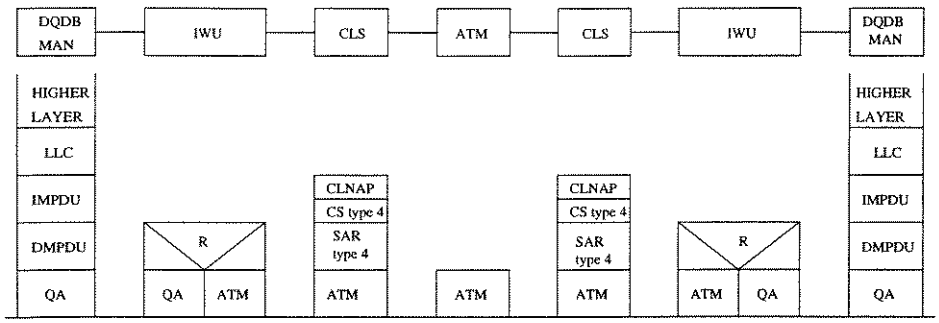

Figura 3.4: Modelo do "stack" para a interconexão DQDB/ATM

- Não há necessidade de translação de endereços pelo fato de a rede DQDB suportar endereçamento de 64 bits [10], utilizado pela rede ATM.

3.4 Formato proposto para o campo de endereços do quadro DQDB

Assumindo o endeçamento de 64 bits e o formato proposto na figura 3.5 do campo de endereço para um quadro de usuário DQDB, obtemos 2^{16} redes DQDB interconectadas através de *bridges* via rede ATM, onde cada rede DQDB pode ter no máximo 2^{48} endereços diferentes para seus terminais, havendo a possibilidade de existir terminais com endereços idênticos, porém em redes DQDB distintas.

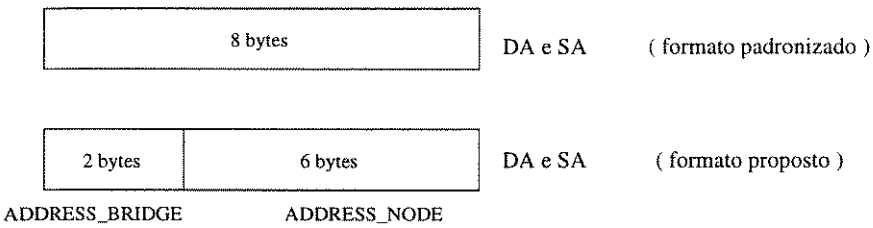

Figura 3.5: Formato do campo de endereço do quadro de usuário DQDB

Um quadro de usuário DQDB que não se destina a uma outra rede DQDB interconectada via uma rede de transporte ATM deve ser gerado com seguinte codificação: o sub-campo ADDRESS_BRIDGE do campo SA(*Source Address*) deve ser preenchido com o mesmo endereço do sub-campo ADDRESS_BRIDGE do campo DA(*Destination Address*). Isto é, ambos devem possuir o endereço da *bridge* de acesso à rede ATM. Quando esse quadro for destinado a uma rede remota DQDB, o sub-campo ADDRESS_BRIDGE do campo SA deve ser preenchido com o endereço da *bridge* de acesso a rede ATM, enquanto que o sub-campo ADDRESS_BRIDGE do campo DA deve ser preenchido com o endereço da *bridge* de acesso a rede remota DQDB.

3.5 Conclusão

Este capítulo apresentou as alternativas de cenários para prover o suporte ao serviço não orientado à conexão, identificando os elementos de rede IWU e CLS envolvidos, o modelo de interconexão DQDB/ATM adotado e o formato proposto para o campo de endereços do quadro DQDB.

O cenário direto, a implementação do elemento IWU como *bridge* e o formato proposto para o campo de endereços do quadro DQDB, são apontados como as opções mais genéricas e adotadas como hipóteses no desenvolvimento dos próximos capítulos.

Capítulo 4

Especificação Formal do Sistema de Interconexão de Redes DQDB/ATM

Este capítulo apresenta a especificação funcional do sistema de interconexão de redes DQDB/-ATM que inclui o elemento de rede IWU e estações DQDB, em uma linguagem de descrição formal, e os procedimentos de simulação adotados, considerando o cenário direto proposto pelo CCITT para o suporte ao serviço não orientado à conexão na B-ISDN. A especificação funcional do elemento de rede CLS [18] citado no cenário não é considerada nesse capítulo.

4.1 Especificação formal

A especificação completa de um sistema envolve um conjunto de informações de maneira informal, semiformal ou formal, dependendo do tipo de informação, da etapa dentro do ciclo de desenvolvimento do sistema e da metodologia de desenvolvimento utilizada. A linguagem informal (ou natural), por exemplo, pode ser utilizada nas etapas iniciais do desenvolvimento do sistema, para definição dos objetivos e requisitos do sistema, onde detalhes de especificação podem ser omitidos. A linguagem formal, por outro lado, aplica-se de forma mais adequada nas etapas onde se deseja expressar informações de forma precisa.

Em geral, uma linguagem de descrição formal possibilita as seguintes operações:

a- Especificações não ambíguas, claras e concisas.

b- Verificação das especificações.

c- Análise funcional das especificações.

d- Implementação das especificações.

e- Teste de conformidade entre a implementação e as especificações.

Para suportar os itens "a","d" e "e", a linguagem de descrição deve possibilitar uma fácil composição, entendimento e expansão das especificações. Isto inclui facilidades, tais como a estruturação do sistema em níveis hierárquicos de abstração, a definição de processos e instâncias e recursividade. Para suportar os itens "b" e "c", a linguagem deve ter uma forte base matemática de descrição para execução das análises sintáticas e semântica.

Desde a década de 70, as universidades e as instituições ISO e CCITT vêm atuando em atividades de desenvolvimento e padronização de linguagens de descrição formal. Na década de 80, três linguagens, padronizadas internacionalmente, se destacaram: Estelle e LOTOS definidos pela ISO e SDL(*Specification and Description Language*) [3] [4] definida pelo CCITT.

Para a especificação funcional do sistema de interconexão de redes DQDB/ATM foi adotada a linguagem SDL, motivada pela disponibilidade da ferramenta SDT(*SDL Design Tool*) [12], com recursos para simulação do sistema.

4.2 Metodologia adotada

A metodologia adotada para a especificação funcional do sistema de interconexão de redes DQDB/ATM considera:

- Especificação dos blocos como sistemas isolados.[3]
- Particionamento visando clareza e flexibilidade.
- Especificação respeitando as restrições da ferramenta SDT.
- Descrição de maneira informal [4] de funções consideradas complementares à especificação.
- Especificação visando eficiência na simulação e localização de erros.

4.3 Especificação funcional do sistema de interconexão de redes DQDB/ATM

A figura 4.1 apresenta o cenário de interconexão adotado para a especificação do elemento IWU e das estações DQDB.

Este cenário considera redes locais DQDB, no exemplo, três redes ("A","B" e "C") acessando, através de seus elementos IWU associados, a rede virtual não orientada à conexão, que é constituída de elementos CLS de acesso interconectados por uma rede de conexões ATM semipermanentes.

[3] O objeto de simulação para a ferramenta SDT é o sistema

[4] As descrições informais, na linguagem SDL, são declaradas entre aspas simples (ex. 'discard DMPDU') ou como comentários.

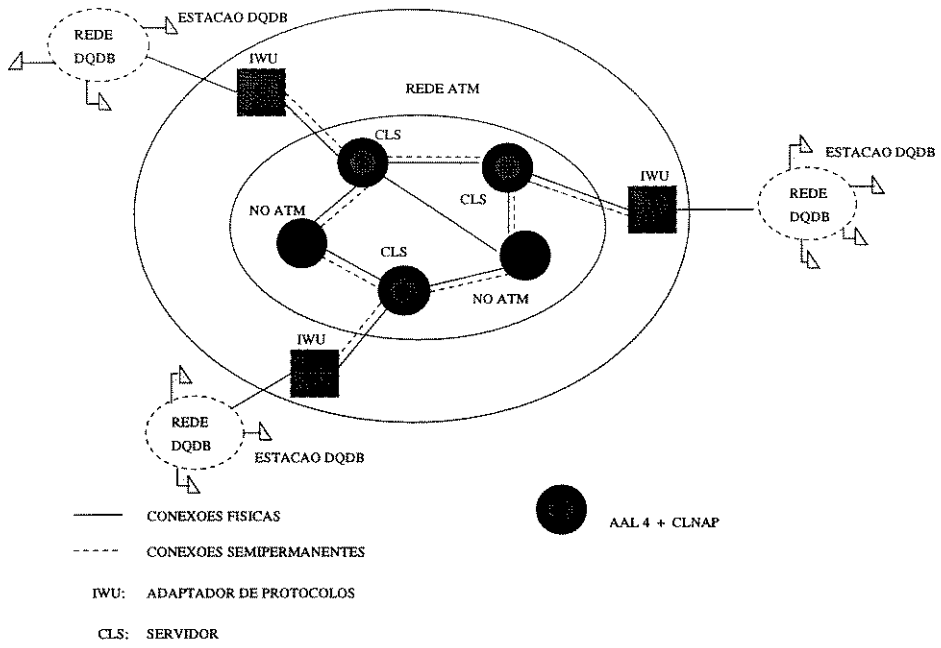

Figura 4.1: Cenário direto adotado

Como primeiro nível de abstração, o cenário de interconexão adotado será representado por um sistema constituído por : uma estação DQDB local geradora de quadros de informação, uma outra estação DQDB local interconectada ao elemento IWU local, o elemento IWU local, o elemento IWU remoto, uma estação DQDB remota interconectada ao elemento IWU remoto e por último, a estação DQDB remota receptora dos quadros gerados pela estação DQDB local. O ambiente do usuário, no caso as redes DQDB, e as gerências de camada são considerados como externos ao sistema. Notar que daqui em diante iremos nos referir ao elemento IWU como uma *bridge*.

A figura 4.2 representa a especificação funcional do sistema "Internetworking" considerado com seus elementos(ou blocos) e as interfaces de interação entre estes e com o ambiente externo.

No sistema "Internetworking" são identificados os seguintes canais:

- C1,C3,C4,C11,C12,C14,C15,C16: representam as interfaces do sistema com as gerências de camada do protocolo DQDB.
- C7: representa a conexão semipermanente que interliga o elememto IWU local ao elemento CLS e esse ao elemento IMU remoto.
- C17: representa a primitiva "MA-UNITDATA_indication" da camada DQDB de acesso as camadas superiores. No modelo ISO, essa primitiva corresponde a primitiva "LLC-DATA_indication".
- C6,C8,C9: representam as interfaces do sistema com as gerências de camada do elemento IWU.
- C2,C5,C10,C13: representam as conexões permanentes que interligam os elementos de rede(estações a estações, ou estações ao elemento IWU).

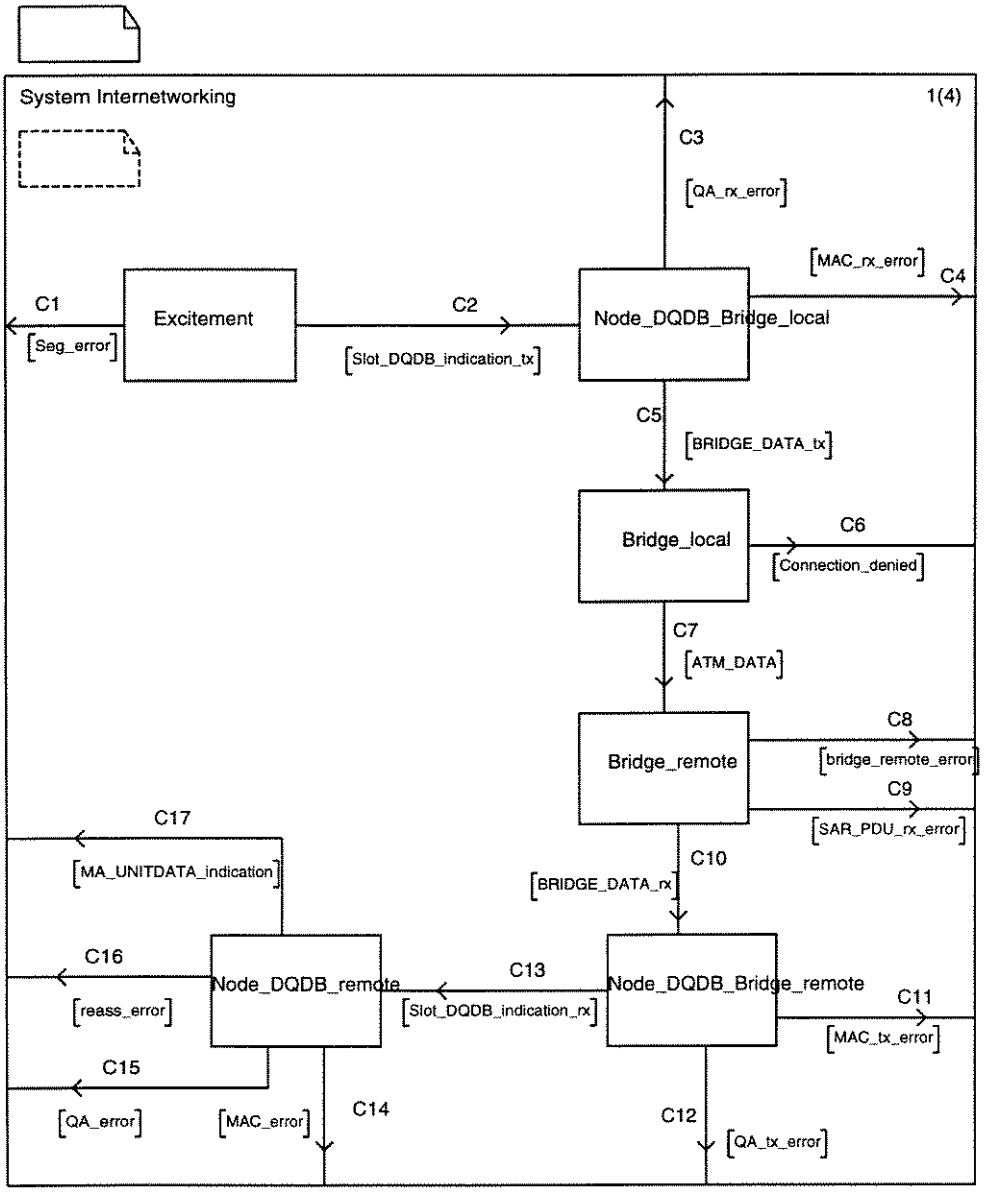

Figura 4.2: Bloco "Internetworking"

O sistema "Internetworking" adotado, apresentado na figura 4.2, é composto por seis blocos: o bloco "Excitement", o bloco "Node_DQDB_Bridge_local", o bloco "Bridge_local", o bloco "Bridge_remote", o bloco "Node_DQDB_Bridge_remote" e o bloco "Node_DQDB_remote".

Esse sistema considera que quadros DQDB gerados em uma rede local DQDB sejam capazes de serem roteados para uma rede remota DQDB via uma rede de transporte ATM.

4.3.1 Descrição dos tipos

A seguir são apresentados os tipos dos parâmetros associados aos sinais do sistema "Internetworking":

- IMPDU_type: representa o formato inicial de um quadro DQDB.

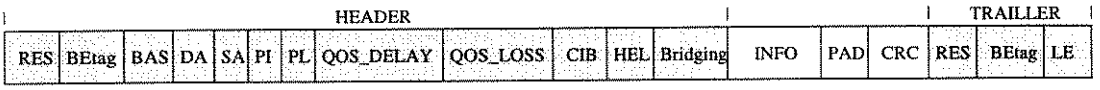

Figura 4.3: IMPDU_type

- DMPDU_type: representa a unidade segmentada do quadro DQDB. O campo INFO corresponde ao tipo IMPDU_type.

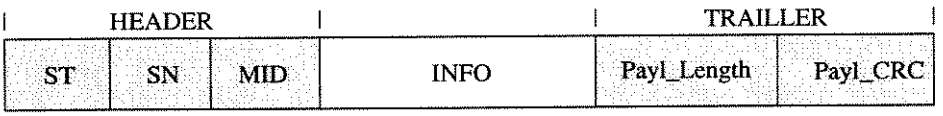

Figura 4.4: DMPDU_type

- QA_segment_type: representa o formato final do quadro DQDB que será inserido no slot. O campo INFO_QA corresponde ao tipo DMPDU_type.

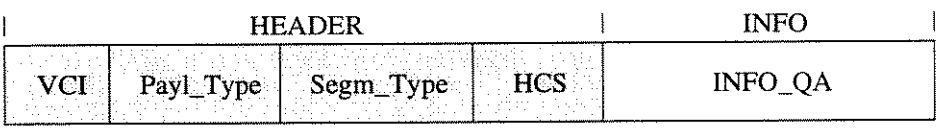

Figura 4.5: QA_segment_type

- Slot_DQDB_type: representa o formato do slot DQDB.

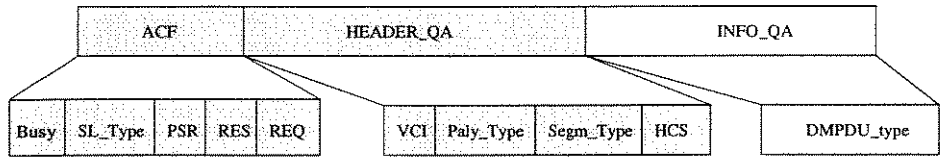

Figura 4.6: Slot_DQDB_type

- SAR_PDU_type: representa a unidade SAR_PDU do protocolo AAL4 da rede ATM. O campo INFO_SAR corresponde ao tipo IMPDU_type.

Figura 4.7: SAR_PDU_type

- ADDRESS_type: representa os endereços de origem e destino da estação e da *bridge*, identificando se esses são individuais ou de grupo.

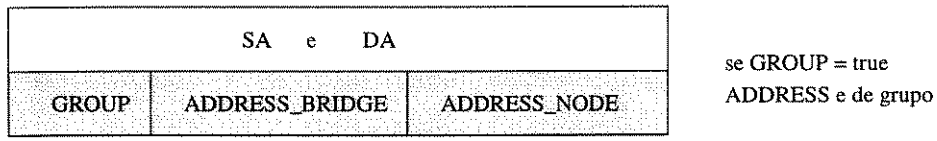

Figura 4.8: ADDRESS_type

- BUS_type: identifica o barramento da rede DQDB: BUS_A, BUS_B, ou BOTH.
- CI_type: identificação da conexão semipermanente que interliga o elemento IWU à rede virtual não orientada à conexão.

4.4 Especificação funcional do bloco "Excitement"

O bloco "Excitement" apresentado na figura 4.9 representa a estação DQDB geradora de quadros DQDB na rede local DQDB. Portanto, esse bloco é responsável pela geração e segmentação(quando necessário) desses quadros, seguido do envio desses através do sinal "Slot_DQDB_indication_tx" para o bloco "Node_DQDB_Bridge_local". Esse bloco não implementa totalmente o protocolo DQDB, pois sua principal função é prover a estimulação do sistema.

O particionamento desse bloco em processos, foi realizado seguindo a ordem de formação das diversas unidades de dados do protocolo DQDB.

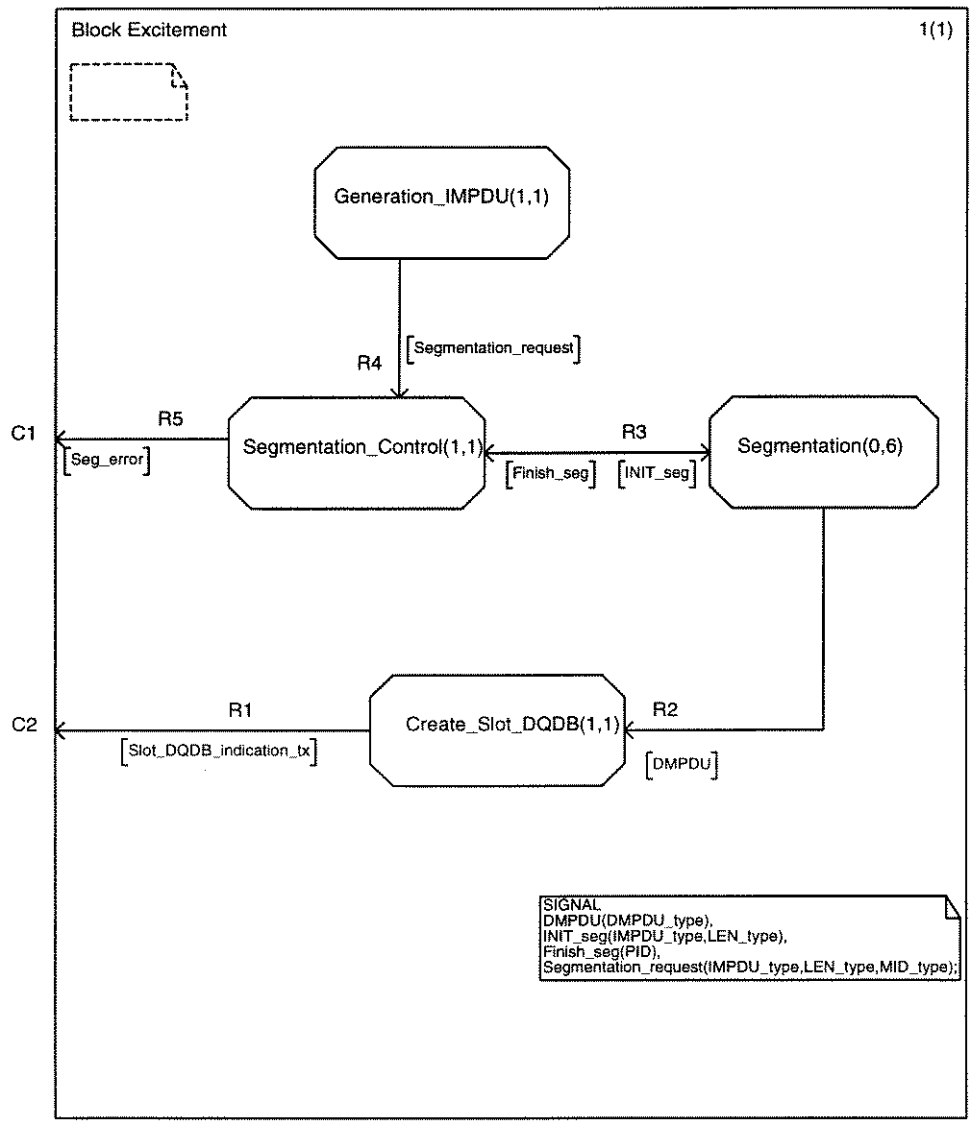

Figura 4.9: Bloco "Excitement"

O processo "Segmentation", por executar as funções mais complexas, foi definido como instanciável, sob controle do processo "Segmentation_Control". Desta forma, por exemplo, o processo "Segmentation_Control" ao receber as unidades IMPDU do processo "Generation_IMPDU", as distribui para diferentes instâncias do processo "Segmentation" para que sejam executadas paralelamente.

Note que a alteração da sequência das unidades de dado, que eventualmente ocorra com este procedimento, é permitida no serviço não orientado à conexão.

4.4.1 Descrição do tipos

- LEN_type: representa o tamanho máximo do quadro DQDB.
- PID: identifica uma instância de um processo.
- MID_type: representa a identificação de um quadro DQDB.

4.4.2 Descrição dos processos

"Generation_IMPDU": Gera quadros DQDB, isto é, unidades IMPDU que estimularam o sistema.

"Segmentation_Control": Gerencia, aloca e libera as instâncias do processo "Segmentation". A alocação compreende a verificação da disponibilidade de uma instância e, no caso positivo, a criação e encaminhamento da unidade IMPDU para o processo "Segmentation". No caso negativo, a unidade IMPDU é descartada. A liberação da instância é efetuada quando o processo recebe o sinal "Finish_seg" gerado, no término da execução do processo "Segmentation".

"Segmentation": Segmenta a unidade IMPDU em unidades DMPDU.

"Create_Slot_DQDB": Gera slots DQDB a partir das unidades segmentadas DMPDU e os encaminha sequencialmente para o barramento definido da rede DQDB. Notar que o procedimento correto, seria formar primeiro o segmento QA a partir das unidades segmentadas e aí então, formar os slots DQDB. Porém, esse passo foi omitido pelo fato que nosso interesse nesse bloco ser somente estimular o sistema.

4.5 Especificação funcional do bloco "Node_DQDB_Bridge_local"

O bloco "Node_DQDB_Bridge_local" apresentado na figura 4.10 corresponde a estação DQDB local que se encontra conectada a uma *bridge*. Esse bloco implementa o protocolo da camada DQDB no que se refere a recepção de slots pelo barramento, porém, não executa remontagem, já que durante a simulação não endereçamos nenhum slot DQDB a essa estação. Portanto, ao receber os slots DQDB gerados pelo bloco "Excitement", essse bloco deve encaminhá-los para o bloco "Bridge_local".

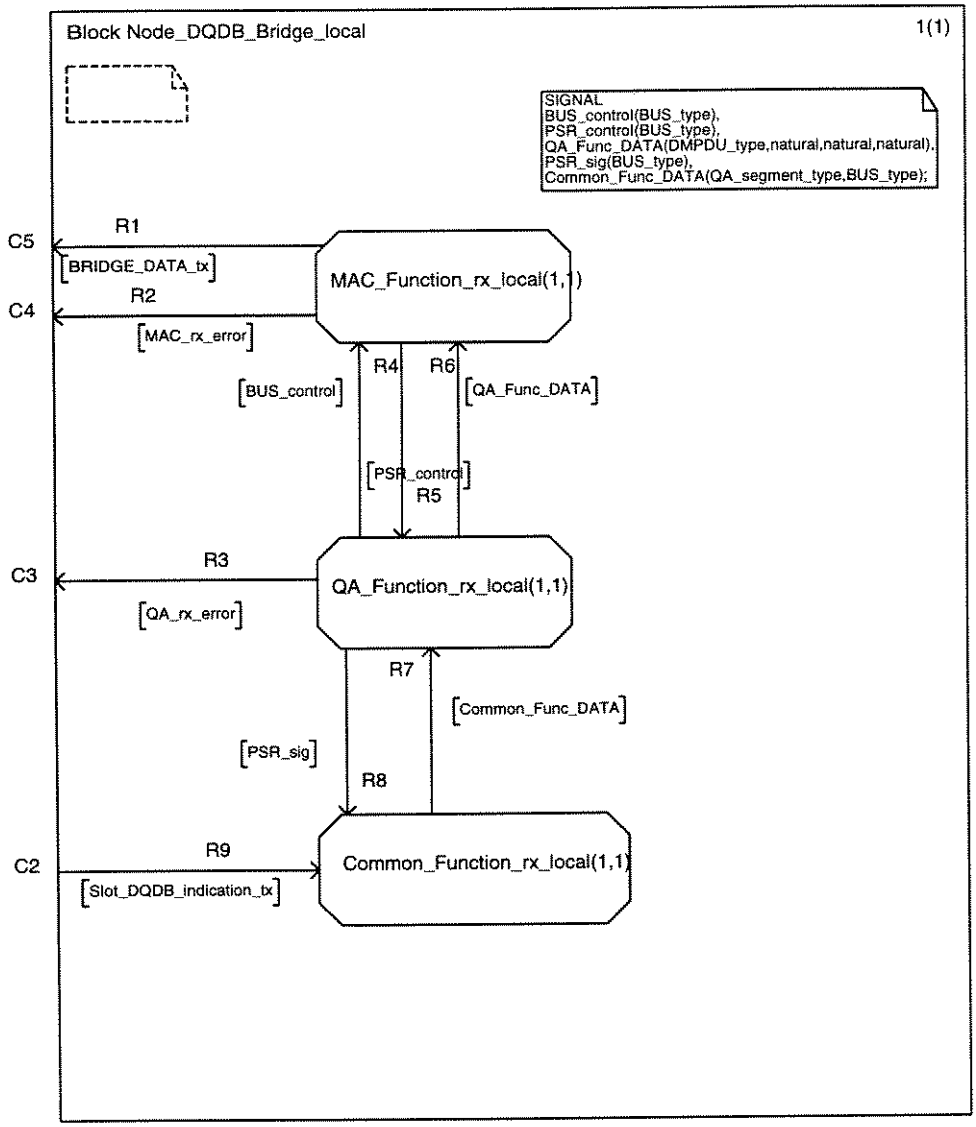

Figura 4.10: Bloco "Node_DQDB_Bridge_local"

O particionamento desse bloco, em processos, foi realizado seguindo o modelo de sub-camadas da camada DQDB, de forma que o processo "Common_Function_rx_local" representa a sub-camada de funções comuns, o processo "QA_Function_rx_local" representa a sub-camada de funções de arbitragem e o processo "MAC_Function_rx_-local" representa a sub-camada de funções de convergência.

4.5.1 Descrição dos tipos

- MID_table_type: tabela dinâmica que armazena o valor do campo MID da primeira unidade de uma IMPDU e o tipo de endereçamento(individual ou de grupo) utilizado por essa unidade.

4.5.2 Descrição dos processos

"Common_Function_rx_local": Monitora cada slot DQDB que trafega pelos barramentos. A monitoração compreende a verificação da disponibilidade do slot e o tipo de segmento. Caso seja um slot DQDB ocupado contendo um segmento QA, imediatamente é realizada uma cópia da carga útil desse slot e encaminhada para o processo "QA_Function_rx_local" através do sinal "Common_Func_DATA". Notar que a carga citada compreende o segmento QA. Esse processo, também, ao receber o sinal "PSR_sig", seta um bit do campo de controle de acesso do próximo slot DQDB que passar por essa estação, no barramento indicado pelo sinal recebido, proveniente da sub-camada imediatamente superior da camada DQDB, representada pelo processo "QA_Function_rx_local". A recepção desse sinal indica que a última unidade DMPDU processada pela sub-camada MCF da camada DQDB, representada pelo processo "MAC_Function_rx_local", apresentava um endereçamento do tipo individual.

"QA_Function_rx_local": Valida o cabeçalho do segmento QA, seleciona e o encaminha para uma função de convergência especicífica, através do sinal "QA_Func_DATA". Notar que esse sinal possue como um dos parâmetros a unidade DMPDU já desencapsulada do segmento QA e o valor do VCI retirado do cabeçalho do segmento QA que indica a função de convergência apropriada. No caso da validação ser negativa ou a estação não suportar uma determinada função de convergência, o segmento QA é descartado e o sinal "QA_rx_error" é enviado com um parâmetro contendo a causa do descarte. Notar que a função de convergência é representada pelo processo "MAC_Function_rx_local" e que esse somente implementa as funções referentes aos serviços não orientados à conexão. Esse processo, também, envia o sinal "BUS_control" para o processo "MAC_Function_rx-_local" contendo o barramento de recepção da DMPDU repassada a esse processo e aguarda o sinal "PSR_control" contendo a informação do tipo de endereçamento da última DMPDU recebida por esse processo.

"MAC_Function_rx_local": Certifica-se da integridade da unidade DMPDU e verifica o endereço de destino dessa unidade. A verificação da integridade consiste na comparação dos valores do campo Payl_CRC recebido e calculado, caso haja discrepâncias, essa unidade é descartada. Partindo do pressuposto que o campo de endereço de destino do quadro DQDB possui o formato apresentado na seção 3.4, a verificação do endereço consiste em investigar na seguinte ordem: o endereço da *bridge* de destino, da estação de destino e o tipo de endereçamento (individual ou de grupo). Caso seja constatado que o endereçamento é do tipo individual, esse processo envia para a sub-camada imediatamente inferior da camada DQDB, representada pelo processo "QA_Function_rx_local", o sinal "PSR_control".

Se o endereço de destino da *bridge* obtido da DMPDU corresponder ao endereço da *bridge* de sua rede local, e o endereço de destino da estação for o endereço dessa estação, essa DMPDU deve ser enviada para o processo de remontagem. No caso de não haver correspondência entre os endereços de destino da *bridge*, essa DMPDU deve ser enviada através do sinal "BRIDGE_DATA_tx" para a *bridge local*, representada pelo bloco "Bridge_local". Notar que a informação sobre endereçamento está contida somente na primeira unidade DMPDU de uma IMPDU. Portanto, foi utilizado a tabela "MID_table" que armazena o valor do campo MID da primeira DMPDU e o tipo de endereçamento encontrado. Dessa maneira, as subsequentes DMPDU's serão processadas de acordo com a tabela. Em todos os casos onde há descarte das unidades, o sinal "MAC_rx_error" é enviado como indicação de erro, cujo parâmetro indica a causa do descarte. A função de remontagem não foi implementada, aparecendo referenciada nesse processo de maneira informal.

4.6 Especificação funcional do bloco "Bridge_local"

O bloco "Bridge_local" apresentado na figura 4.11 representa a *bridge* que se encontra conectada a rede local DQDB e, portanto, deverá executar as conversões necessárias para que um slot DQDB possa trafegar por uma rede ATM e alcançar a *bridge* de entrada da rede remota DQDB.

4.6.1 Descrição dos tipos

- route1_table_type: tabela estática que apresenta as opções de conexões, por endereço de destino da *bridge* e pela classe de serviço.
- route2_table_type: tabela dinâmica que armazena o valor do campo MID e a conexão semipermanente selecionada no roteamento da primeira unidade DMPDU de uma IMPDU.

4.6.2 Descrição dos processos

"DQDB_ATM": Identifica a primeira unidade DMPDU pertencente a unidade IMPDU e, a partir das informações de endereço de destino da *bridge* e da classe de qualidade de serviço no que diz respeito a prioridade de descarte de slots DQDB devido a um congestionamento, seleciona na tabela "route1_table" a conexão semipermanente para qual será encaminhada. Em seguida seta a tabela "route2_table" com o valor do MID da primeira unidade DMPDU e com a conexão selecionada. As demais unidades DMPDU, da mesma IMPDU, serão roteadas de acordo com a tabela "route2_table". Ao identificar a primeira unidade DMPDU, esse processo verifica o valor do campo Bridging. Se constatado o valor zero, a unidade DMPDU e as demais unidades da mesma IMPDU são descartadas, caso contrário, o valor desse campo é decrementado.

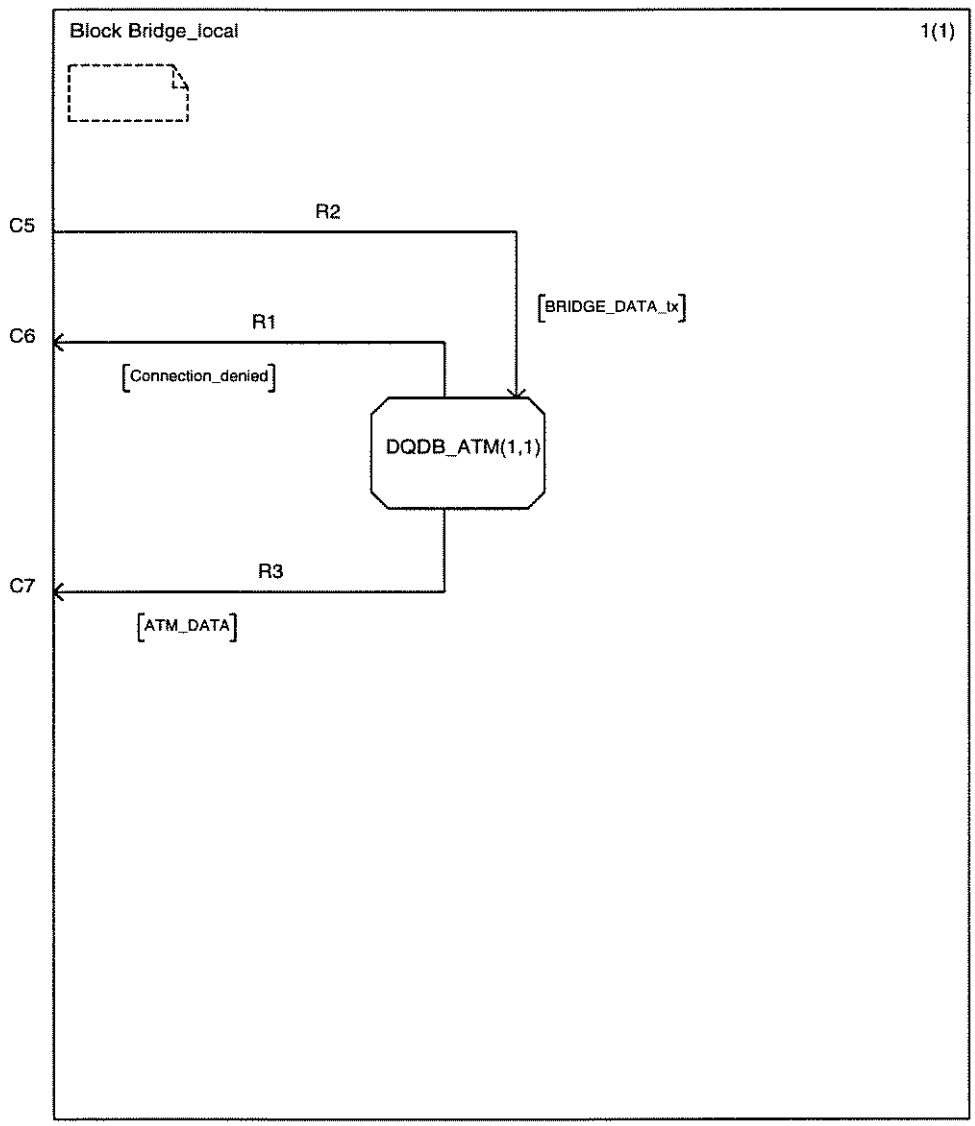

Figura 4.11: Bloco "Bridge_local"

Este procedimento limita um número máximo de *bridges* que uma determinada DMPDU pode trafegar e, portanto, evita, por algum motivo, que uma unidade DMPDU fique em "loop" na rede. No caso do endereço de destino da *bridge* não constar na tabela "route1_table", ou da não disponibilidade de uma conexão com a classe de qualidade de serviço desejada, a unidade DMPDU e as demais unidades consecutivas, da mesma IMPDU, são descartadas. O descarte das unidades é sinalizado pelo sinal "Connection_denied", cujo parâmetro indica a causa do descarte. Esse processo, antes de efetuar o encaminhamento das unidades pela conexão selecionada para o bloco "Bridge_remote" através do sinal "ATM_DATA", realiza o mapeamento dos campos da unidade DMPDU nos campos da unidade SAR_PDU do protocolo AAL 4. Notar que a semelhança entre os campos "header" e "trailler" das estruturas citadas, como também do tamanho do campo de dados, possibilita uma troca transparente entre os campos.

4.7 Especificação funcional do bloco "Bridge_remote"

O bloco "Bridge_remote" apresentado na figura 4.12 representa a *bridge* que se encontra conectada a rede remota DQDB e, portanto, deve executar as conversões necessárias para mapear uma célula ATM em um slot DQDB.

4.7.1 Descrição dos tipos

- route3_table_type: tabela estática que apresenta os endereços das estações DQDB remotas.
- route4_table_type: tabela dinâmica que armazena o valor do campo MID da primeira unidade DMPDU de cada IMPDU.

4.7.2 Descrição dos processos

"ATM_DQDB:" Verifica a integridade das unidades SAR_PDU, descartando as unidades inválidas através do sinal "SAR_PDU_rx_error". Mapeia as unidades SAR_PDU em unidades DMPDU. Certifica-se que o endereço da estação de destino contida na primeira unidade DMPDU de uma IMPDU corresponde a algum dos endereços encontrados na tabela "route3_table" e, no caso positivo, o encaminhamento dessa unidade para o bloco "Node_DQDB_Bridge_remote" através do sinal "BRIDGE_DATA_rx", seguido do armazenamento do valor de seu campo MID na tabela "route4_table". As demais unidades da mesma IMPDU devem ser roteadas de acordo com tabela "route4_table", isto é, as unidades são consideradas roteavéis quando o valor de seu campo MID corresponder a alguns daqueles armazenados nessa tabela. Em qualquer dos casos de não correspondência nas tabelas citadas, as unidades são descartadas através do sinal "bridge_remote_error". Notar que o campo Bridging da primeira unidade DMPDU possue a mesma importância apresentada no bloco "Bridge_local".

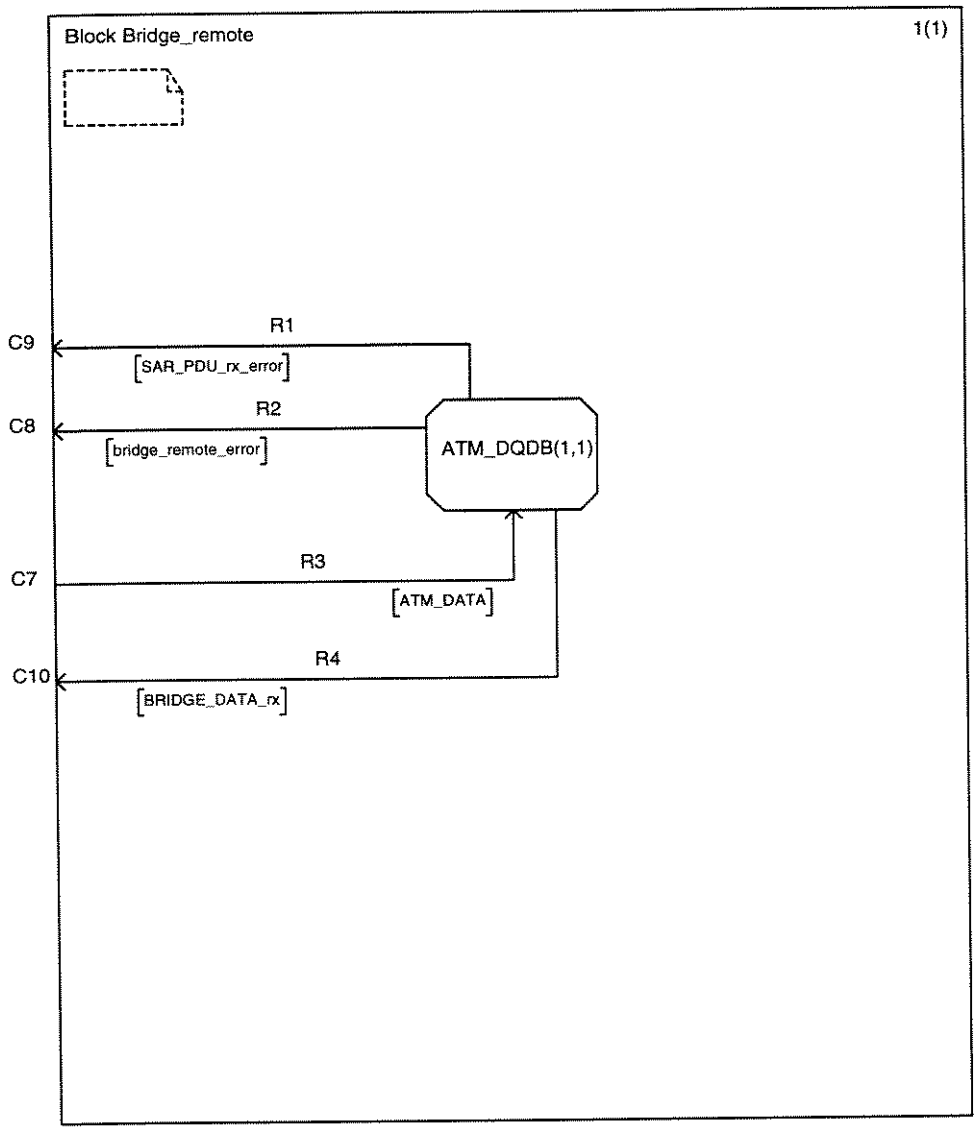

Figura 4.12: Bloco "Bridge_remote"

4.8 Especificação funcional do bloco "Node_DQDB-_Bridge_remote"

O bloco "Node_DQDB_Bridge_remote" apresentado na figura 4.13 corresponde a estação DQDB remota que se encontra conectada a uma *bridge*. Esse bloco implementa o protocolo da camada DQDB no que se refere a transmissão de slots pelo barramento. Por simplicidade, não implementamos os processos de formação da IMPDU, segmentação e formação da DMPDU pelo fato de que em nossa simulação, essa estação não requer o envio de quadros DQDB. Notar que as únicas unidades que trafegam pelo nosso sistema são originadas do bloco "Excitement" pertencente a rede local DQDB e endereçadas ao bloco "Node_DQDB_remote". Portanto, esse bloco deve encaminhar todas as unidades recebidas da *bridge* para o barramento da estação remota de destino.

O particionamento desse bloco em processos, foi realizado seguindo o modelo de sub-camadas da camada DQDB, de forma que o processo "Common_Function_tx_local" representa a sub-camada de funções comuns, o processo "QA_Function_tx_local" representa a sub-camada de funções de arbitragem e o processo "MAC_Function_tx_-local" representa a sub-camada de funções de convergência.

4.8.1 Descrição dos tipos

- translate_MID_table_type: tabela dinâmica que armazena os valores do campo MID das primeiras unidades DMPDU de uma IMPDU recebidas da *bridge* e os novos valores a serem usados para esse campo. Esses novos valores são calculados pela entidade de gerência da camada DQDB.

- access_queue_buffer_type: tabela dinâmica que implementa um buffer no qual armazena os segmentos QA de acordo com a prioridade de acesso ao barramento.

4.8.2 Descrição dos processos

"Common_Function_tx_remote": Insere o segmento QA na carga útil do slot DQDB, e seta o campo ACF desse slot, de acordo com os sinais "write_ACF_bus_B" e "request_ACF_bus_A".

"QA_Function_tx_remote": Encapsula as unidades DMPDU, formando os segmentos QA. Bufferiza os segmentos em filas, no caso implementado através da tabela "Access_Queue_buffer_type", de acordo com o barramento e nível de prioridade no acesso ao meio. Envia um pedido requisitando um slot vazio para cada segmento bufferizado, pelo barramento oposto ao de transmissão, através do sinal "request_ACF_bus_A". Executa o mecanismo de acesso ao meio e, portanto, os segmentos permanecem nas respectivas filas enquanto não houver um slot disponível, isto é, um slot vazio e, no caso positivo, os segmentos são encaminhados para a sub-camada imediatamente inferior da camada DQDB, representada pelo processo "Common_Function_tx_remote" através do sinal "QA_Func_DATA", juntamente com o sinal "write_ACF_bus_B".

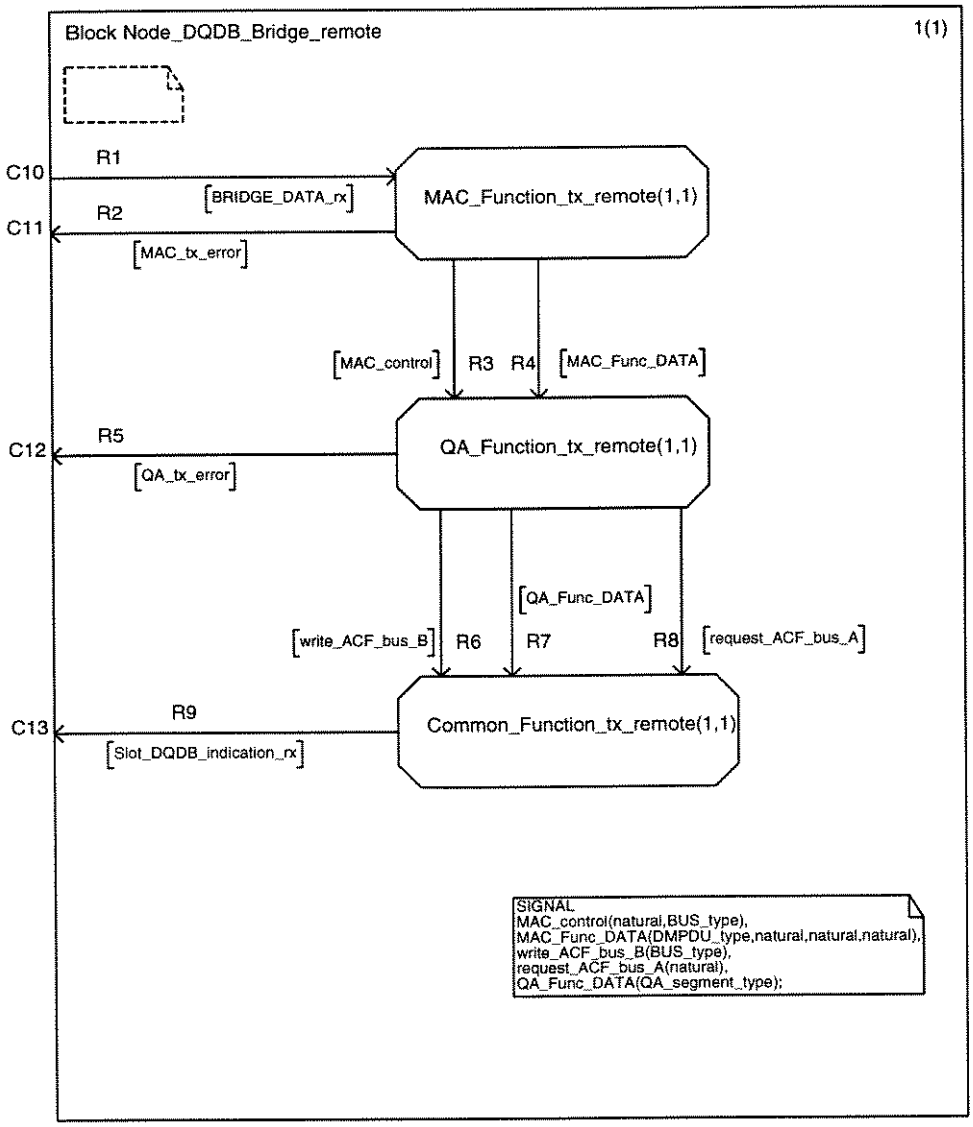

Figura 4.13: Bloco "Node_DQDB_Bridge_remote"

Notar que a verificação da disponibilidade do slot aparece referenciada de maneira informal e que o mecanismo citado acima foi implementado somente no que se refere ao barramento "B" e nível 2 de prioridade.

"MAC_Function_tx_remote": Certifica-se da integridade da unidade DMPDU recebida da *bridge* através do sinal "BRIDGE_DATA_rx" e, no caso de haver erros, será sinalizado através do sinal "MAC_tx_error", contendo a causa do erro. Executa a translação do valor do campo MID. A translação é necessária para garantir a confiabilidade do processo de remontagem, pois valores de campo MID de unidades DMPDU provenientes de uma rede local poderiam coincidir com valores utilizados por essa rede. Portanto, ao identificar a primeira unidade DMPDU de uma IMPDU, o campo MID e o campo QOS_DELAY são armazenados na tabela "translate_MID_table" juntamente com o novo valor para o campo MID. Dessa maneira, as demais unidades da mesma IMPDU serão enviadas pelo barramento para o bloco "Node_DQDB_remote", com o novo valor do campo MID encontrado na tabela. Esse novo valor para o campo MID é calculado pela entidade de gerenciamento da camada DQDB. O valor do campo QOS_DELAY armazenado na tabela "translate_MID_table" é utilizado pelo sinal "MAC_control" para indicar o nível de prioridade de acesso ao meio que a sub-camada imediatamente inferior da camada DQDB, representada pelo processo "QA_Function_tx_remote" deverá obedecer. Notar que a verificação do endereço de destino das unidades DMPDU recebidas por esse processo foi omitida devido ao fato de que todas as unidades existentes no sistema são endereçadas ao bloco "Node_DQDB_remote", como também, as funções de formação da IMPDU e da DMPDU, segmentação dessa primeira, são somente referenciadas. O motivo da não implementação dessas funções, decorre do fato de que em nenhum instante, a estação representada por esse bloco, solicita o envio de algum quadro de dados.

4.9 Especificação funcional do bloco "Node_DQDB_remote"

O bloco "Node_DQDB_remote" apresentado na figura 4.14 corresponde a estação DQDB localizada na rede remota DQDB de destino dos quadros de dados gerados por uma estação DQDB situada na rede local DQDB. Esse bloco foi inserido no sistema com o intuito de prover a validação desse, e implementar por completo o protocolo DQDB no que se refere a recepção de slots, inclusive a remontagem das unidades segmentadas geradas pelo bloco "Excitement".

O particionamento desse bloco em processos, foi novamente realizado seguindo o modelo de sub-camadas da camada DQDB, de forma que o processo "Common_Function" representa a sub-camada de funções comuns, o processo "QA_Function" representa a sub-camada de funções de arbitragem e o processo "MAC_Function" representa a sub-camada de funções de convergência. Para prover a função de remontagem foi inserido o processo "Reassembly".

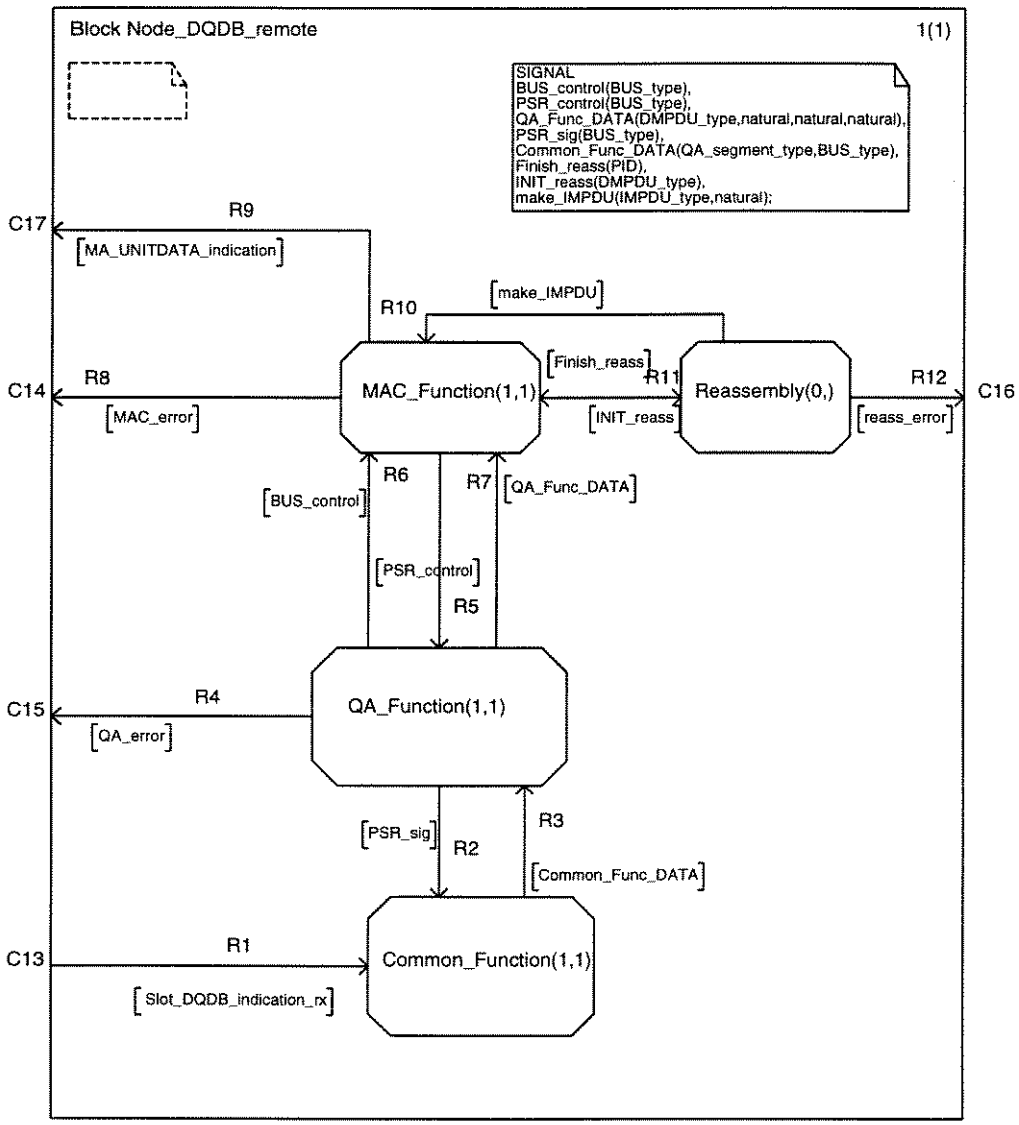

Figura 4.14: Bloco "Node_DQDB_remote"

4.9.1 Descrição dos tipos

- VCI_MID_table_type: tabela dinâmica que armazena o par de valores do campo MID e VCI, e o tipo de endereçamento(individual ou de grupo), das primeiras unidades DMPDU a serem remontadas.

- INST_table_type: tabela dinâmica que armazena o valor do identificador da instância para o par de valores do campo MID e VCI.

4.9.2 Descrição dos processos

"Common_Function": Monitora cada slot DQDB que trafega pelos barramentos. A monitoração compreende a verificação da disponibilidade do slot e o tipo de segmento. Caso seja um slot DQDB ocupado contendo um segmento QA, imediatamente é realizada uma cópia da carga útil desse slot e o encaminhamento dessa carga através do sinal "Common_Func_DATA" para a camada imediatamente superior da camada DQDB, representada pelo processo "QA_Function". Notar que a carga citada trata-se do segmento QA. Esse processo, também, ao receber o sinal "PSR_sig", seta através do sinal "PSR_signal_remote", um bit do campo de controle de acesso do próximo slot DQDB que passar por essa estação, no barramento indicado pelo sinal recebido, proveniente da sub-camada imediatamente superior da camada DQDB, representada pelo processo "QA_Function". A recepção desse sinal indica que a última unidade DMPDU processada pela sub-camada MCF da camada DQDB, representada pelo processo "MAC_Function", apresentava um endereçamento do tipo individual.

"QA_Function": Valida o cabeçalho do segmento QA, seleciona e o encaminha para uma função de convergência especicífica através do sinal "QA_Func_DATA". Notar que esse sinal possue como um dos parâmetros a unidade DMPDU já desencapsulada do segmento QA e o valor do VCI retirado do cabeçalho do segmento QA que indica a função de convergência apropriada. No caso da validação ser negativa ou a estação não suportar uma determinada função de convergência, o segmento QA é descartado e o sinal "QA_error" é enviado com parâmetro contendo a causa do descarte. Notar que a função de convergência é representada pelo processo "MAC_Function" e que esse somente implementa as funções referentes aos serviços não orientados à conexão. Esse processo, também, envia o sinal "BUS_control" para o processo "MAC_Function" contendo o barramento de recepção da DMPDU repassada. Caso haja a recepção do sinal "PSR_control", esse processo imediatamente repassa a informação recebida por esse sinal para a sub-camada imediatamente inferior da camada DQDB, representada pelo processo "Common_Function", através do sinal "PSR_sig".

"MAC_Function": Certifica-se da integridade da unidade DMPDU, verifica o endereço de destino e o tipo de endereçamento dessa unidade. A verificação da integridade consiste na comparação dos valores do campo Payl_CRC recebido e calculado. Caso haja discrepâncias, essa unidade é imediatamente descartada. Notar que a informação sobre endereçamento está contida somente na primeira unidade DMPDU de uma IMPDU.

Portanto, foi utilizado a tabela "VCI_MID_table" para garantir o correto processamento das subsequentes DMPDU da mesma IMPDU. A medida que esse processo recebe as unidades, o sinal "PSR_control" é enviado para a sub-camada imediatamente inferior da camada DQDB, representada pelo processo "QA_Function", caso a unidade possua um endereço do tipo individual. Gerencia, aloca e libera as instâncias do processo "Reassembly". A alocação consiste na verificação da disponibilidade de uma instância e, no caso positivo, a criação e encaminhamento da unidade DMPDU para esta. No caso negativo, a unidade DMPDU é descartada. A liberação da instância é efetuada quando o processo recebe o sinal "Finish_reass", gerado no término da execução do processo "Reassembly". Verifica a integridade da IMPDU remontada pelo processo "Reassembly", descartando as unidades inválidas. Em todos os casos onde há descarte das unidades, o sinal "MAC_error" é enviado como indicação de erro, cujo o parâmetro indica a causa do descarte.

"Reassembly": Efetua a remontagem das unidades IMPDU a partir das unidades DMPDU. Esse processo é temporizado de forma a liberar a instância quando o tempo para remontagem se tornar excessivo, como é o caso quando ocorre a perda de uma unidade DMPDU. O término da remontagem de uma determinada IMPDU é sinalizado através do sinal "Finish_reass".

4.10 Conclusão

Este capítulo apresentou a especificação formal, em linguagem SDL, de cada elemento que compõe o sistema utilizado para simular a interconexão de redes locais DQDB via uma rede de transporte ATM.

Capítulo 5

Simulação dos elementos de interconexão de redes DQDB/ATM

Este capítulo apresenta a simulação do sistema de interconexão de redes DQDB/ATM para serviços não orientados à conexão, e os procedimentos de simulação adotados.

5.1 Configuração para simulação

Para a simulação do sistema "Internetworking" foram considerados o modelo de interconexão apresentado na figura 5.1 e um conjunto de definições complementares, arbitrariamente selecionado para a análise do comportamento dinâmico do sistema.

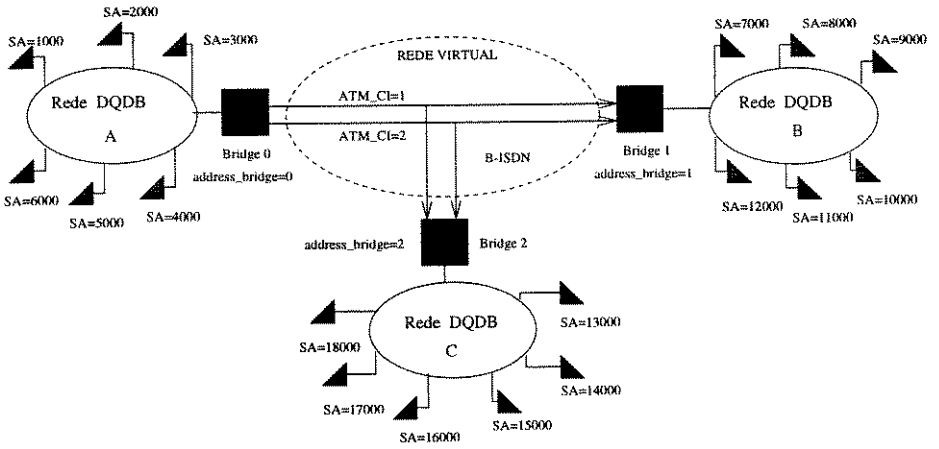

Figura 5.1: Modelo utilizado na simulação

Neste modelo foram considerados três redes locais DQDB, com seis estações cada, sendo que, a rede DQDB "A" é a geradora de unidades de dado e as redes DQDB "B" e"C" são as receptoras. Considera-se, também, a existência de dois circuitos virtuais da rede de transporte ATM dedicados exclusivamente para interligar os elementos IWU e o CLS, e dessa maneira, prover um serviço não orientado à conexão entre a rede DQDB e ATM.

A partir desse modelo, o sistema "Internetworking" foi configurado com as seguintes tabelas:

ADDRESS_BRIDGE	ATM_CI	QOS_LOSS
1	1	TRUE
	2	FALSE
2	1	TRUE
	2	FALSE

Figura 5.2: Tabela "route1_table"

MID_BOM	AAL_CI	BUSY
0	1	false
0	1	false
0	1	false
0	1	false
0	1	false
0	1	false

Figura 5.3: Tabela "route2_table"

ADDRESS_NODE
7000
8000
9000
10000
11000
12000
13000
14000
15000
16000
17000
18000

Figura 5.4: Tabela "route3_table"

MID_BOM	BUSY
0	false
0	false
0	false
0	false
0	false
0	false

Figura 5.5: Tabela "route4_table"

MID	NEW_MID	ACCESS	BUSY
0	0	0	false
0	0	0	false
0	0	0	false
0	0	0	false
0	0	0	false
0	0	0	false

Figura 5.6: Tabela "translate_MID_table"

Access_Queue_0	Access_Queue_1	Access_Queue_2	Busy_0	Busy_1	Busy_2
nothing	nothing	nothing	false	false	false
nothing	nothing	nothing	false	false	false
nothing	nothing	nothing	false	false	false
nothing	nothing	nothing	false	false	false
nothing	nothing	nothing	false	false	false
nothing	nothing	nothing	false	false	false

Figura 5.7: Tabela "Access_Queue_buffer_type"

MID_BOM	VCI	INDIVIDUAL	BUSY
0	0	false	false
0	0	false	false
0	0	false	false
0	0	false	false
0	0	false	false
0	0	false	false

Figura 5.8: Tabela "VCI_MID_table"

MID_BOM	INDIVIDUAL	BUSY
0	false	false
0	false	false
0	false	false
0	false	false
0	false	false
0	false	false

Figura 5.9: Tabela "MID_table"

ID	BUSY
0	false
0	false
0	false
0	false
0	false
0	false

Figura 5.10: Tabela "INST_Seg_table"

VCI	MID	ID	BUSY
0	0	0	false
0	0	0	false
0	0	0	false
0	0	0	false
0	0	0	false
0	0	0	false

Figura 5.11: Tabela "INST_Reass_table"

Essas tabelas podem ser atualizadas automaticamente, por exemplo, em ocasiões de queda do enlace físico, queda da conexão semipermanente ou em situações de congestionamento da rede.

Além dessas tabelas, as seguintes definições foram consideradas:

- Tempo máximo para remontagem do processo "Reassembly" de 10 unidades de tempo [5].
- Número máximo de instâncias para o processo "Segmentation" igual a seis [6].
- O campo INFO da IMPDU corresponde ao conteúdo do quadro de dados de um usuário DQDB [7].
- O parâmetro LEN_INFO corresponde ao tamanho em bytes do quadro de dados de um usuário DQDB [8].
- Valor utilizado para o campo VCI para serviços não orientados à conexão: 1 [9].

5.1.1 Procedimento

A estimulação do sistema é feita através dos sinais "Slot_DQDB_indication_tx", variando o número desses sinais, ou seja, das unidades IMPDU transmitidas e o conteúdo dos campos dessas unidades.

[5] Essa unidade, definida pelo simulador, refere-se ao intervalo de tempo de execução de uma transição entre estados de um processo.

[6] O objetivo desse número é limitar a capacidade de tratamento desse processo.

[7] Para facilitar a análise do sistema, foi adotado a seguinte equação (número de unidades segmentadas * 10) para o valor do campo INFO.

[8] O valor do parâmetro LEN_INFO está estritamente relacionado com o número de unidades segmentadas de uma IMPDU.

[9] O valor do VCI para serviços não orientados à conexão corresponde a todos os bits setados em "1".

Os seguintes estímulos foram gerados:

1. Três unidades IMPDU, onde a primeira é segmentada em quatro unidades DMPDU, a segunda em seis e a terceira dispensa a segmentação. Portanto em nossa simulação, haverá somente onze unidades DMPDU trafegando pelo sistema. As figuras 5.11,5.12,5.13 apresentam os valores utilizados para os campos das unidades segmentadas das três IMPDU.

ST	SN	MID	RES	BEtag	BAS	DA	SA	PI	PL	QOS_DELAY	QOS_LOSS	CIB	HEL	Bridging	INFO	Payl_Length	Payl_CRC
BOM	0	1	0	1	168	FALSE 1 9000	FALSE 0 1000	1	3	2	TRUE	TRUE	0	2	40	44	TRUE

ST	SN	MID	INFO	Payl_Length	Payl_CRC
COM	1	1	40	44	TRUE

ST	SN	MID	INFO	Payl_Length	Payl_CRC
COM	2	1	40	44	TRUE

ST	SN	MID	INFO	PAD	CRC	RES	BEtag	LE	Payl_Length	Payl_CRC
EOM	3	1	40	3	TRUE	0	1	168	36	TRUE

Figura 5.12: DMPDU's referente a primeira IMPDU_PDU

ST	SN	MID	RES	BEtag	BAS	DA	SA	PI	PL	QOS_DELAY	QOS_LOSS	CIB	HEL	Bridging	INFO	Payl_Length	Payl_CRC
BOM	0	2	0	2	224	FALSE 1 9000	FALSE 0 1000	1	0	2	FALSE	TRUE	0	2	60	44	TRUE

ST	SN	MID	INFO	Payl_Length	Payl_CRC
COM	1	2	60	44	TRUE

ST	SN	MID	INFO	Payl_Length	Payl_CRC
COM	2	2	60	44	TRUE

ST	SN	MID	INFO	Payl_Length	Payl_CRC
COM	3	2	60	44	TRUE

ST	SN	MID	INFO	Payl_Length	Payl_CRC
COM	4	2	60	44	TRUE

ST	SN	MID	INFO	PAD	CRC	RES	BEtag	LE	Payl_Length	Payl_CRC
EOM	5	2	60	0	TRUE	0	2	224	4	TRUE

Figura 5.13: DMPDU's referente a segunda IMPDU_PDU

ST	SN	MID	RES	BEtag	BAS	DA	SA	PI	PL	Q_D	Q_L	CIB	HEL	Br	INF	PAD	CRC	R	BEtag	LE	Payl_Length	Payl_CRC
SSM	0	0	0	3	40	FALSE 1 9000	FALSE 0 1000	1	1	2	TRUE	TRUE	0	2	10	1	TRUE	0	3	40	40	TRUE

Figura 5.14: DMPDU referente a terceira IMPDU_PDU

2. Alteração do valor do campo MID [10] da terceira IMPDU para 10 e análise do processo "MAC_Function".

3. Alteração do endereço da estação de destino contido no campo ADDRESS_NODE para 20000 e análise do processo "ATM_DQDB".

4. Alteração do endereço da *bridge* de destino contido no campo ADDRESS_BRIDGE para 3 e análise do processo "DQDB_ATM".

[10] O IEEE 802.6 especifica que todos os bits desse campo deverão ser setados em "0" quando a unidade em questão se tratar de uma SSM DMPDU.

5. Alteração do valor do campo Bridging para 1 e análise do processo "ATM_DQDB".

6. Alteração do valor do campo BEtag do TRAILLER da IMPDU para o valor do campo BEtag do HEADER da IMPDU incrementado de um, e análise do processo "MAC_Function".

7. Descarte da primeira unidade DMPDU da primeira IMPDU e análise dos processos "Reassembly" e "MAC_Function".

8. Descarte da terceira unidade DMPDU da segunda IMPDU e análise dos processos "Reassembly" e "MAC_Function".

9. Descarte da última unidade DMPDU da primeira IMPDU e análise dos processos "Reassembly" e "MAC_Function".

10. Alteração do valor do campo CRC da quarta unidade DMPDU da segunda IMPDU para FALSE e análise dos processos "Reassembly" e "MAC_Function".

5.1.2 Lista dos parâmetros dos sinais de erros gerados pelo sistema

A finalidade dessa lista é auxiliar na compreensão do estudo dos casos simulados [11]. Notar que o envio desses sinais é acompanhado do descarte da unidade em questão.

- Lista para o bloco "Excitement"
  - Sinal "Seg_error"
    - 0: indica que a IMPDU foi descartada devido a indisponibilidade de alocação de uma instância para o processo "Segmentation".

- Lista para o bloco "Node_DQDB_Bridge_local"
  - Sinal "QA_rx_error"
    - 0: indica que o segmento QA foi descartado por apresentar um valor para o campo VCI incompatível com o serviço suportado por essa estação. Um valor diferente de "1" será considerado incompatível.
    - 1: indica que o segmento QA foi descartado por apresentar o cabeçalho inválido.
  - Sinal "MAC_rx_error"
    - 2: indica que a DMPDU foi descartada por apresentar um valor para o campo Payl_CRC diferente do valor calculado para esse campo.
    - 3: indica que a BOM DMPDU foi descartada devido a indisponibilidade de armazenar informações dessa unidade na tabela "MID_table".

[11] Alguns sinais erros foram somente inseridos no sistema para facilitar a compreensão do comportamento de seus respectivos processos e, portanto, não há nenhuma relação com o padrão IEEE 802.6.

- 4: indica que a COM ou EOM DMPDU foi descartada por apresentar um valor para o campo MID desconhecido daqueles armazenados na tabela "MID_table".

- Lista para o bloco "Bridge_local"

  – Sinal "Connection_denied"

  - 0: indica que a BOM DMPDU foi descartada por apresentar um endereço de *bridge* de destino diferente daqueles encontrados na tabela "route1_table".
  - 1: indica que a COM DMPDU foi descartada por apresentar um valor para o campo MID diferente daqueles encontrados na tabela "route2_table".
  - 2: indica que a EOM DMPDU foi descartada por apresentar um valor para o campo MID diferente daqueles encontrados na tabela "route2_table".
  - 3: indica que a SSM DMPDU foi descartada por apresentar um endereço de *bridge* de destino diferente daqueles encontrados na tabela "route1_table".
  - 4: indica que não houve disponibilidade para armazenar a informação do valor do campo MID da BOM DMPDU na tabela "route2_table".

- Lista para o bloco "Bridge_remote"

  – Sinal "SAR_PDU_rx_error"

  - 0: indica que a SAR_PDU recebida pela *bridge* remota foi descartada por apresentar um valor para o campo CRC diferente do valor calculado.

  – Sinal "bridge_remote_error"

  - 0: indica que não houve disponibilidade para armazenar na tabela "route2_table", a informação do valor do campo MID da BOM DMPDU proveniente da rede local DQDB e, portanto, essa unidade foi descartada.

- Lista para o bloco "Node_DQDB_Bridge_remote"

  – Sinal "MAC_tx_error"

  - 0: indica que a BOM DMPDU foi descartada devido a indisponibilidade de se armazenar informação dessa unidade na tabela "translate_MID_table".
  - 1: indica que a COM ou EOM DMPDU foi descartada por apresentar um valor para o campo MID diferente daqueles armazenados na tabela "translate_MID_table".
  - 2: indica que a DMPDU foi descartada por apresentar um valor para o campo Payl_CRC diferente do valor calculado para esse campo.

  – Sinal "QA_tx_error"

  - 0: indica que a fila utilizada para bufferizar um segmento QA com nível 2 de prioridade de acesso ao barramento BUS_B, se encontra vazia.

- 1: indica que um segmento QA com nível 0 de prioridade de acesso ao barramento BUS_A foi descartado devido a indisponibilidade de armazená-lo na fila correspondente.
- 2: indica que um segmento QA com nível 0 de prioridade de acesso ao barramento BUS_B foi descartado devido a indisponibilidade de armazená-lo na fila correspondente.
- 3: indica que um segmento QA com nível 1 de prioridade de acesso ao barramento BUS_A foi descartado devido a indisponibilidade de armazená-lo na fila correspondente.
- 4: indica que um segmento QA com nível 1 de prioridade de acesso ao barramento BUS_B foi descartado devido a indisponibilidade de armazená-lo na fila correspondente.
- 5: indica que um segmento QA com nível 2 de prioridade de acesso ao barramento BUS_A foi descartado devido a indisponibilidade de armazená-lo na fila correspondente.
- 6: indica que um segmento QA com nível 2 de prioridade de acesso ao barramento BUS_B foi descartado devido a indisponibilidade de armazená-lo na fila correspondente.

• Lista para o bloco "Node_DQDB_remote"

– Sinal "QA_error"

- 0: indica que o segmento QA foi descartado por apresentar o cabeçalho inválido.
- 1: indica que o segmento QA foi descartado por apresentar um valor para o campo VCI incompatível com o serviço suportado por essa estação. Um valor diferente de "1" será considerado incompatível.

– Sinal "MAC_error"

- 0: indica que a DMPDU foi descartada por apresentar um valor para o campo Payl_CRC diferente do valor calculado para esse campo.
- 1: indica que a BOM DMPDU foi descartada por apresentar endereços de destino da *bridge* e da estação incompatíveis com o endereço dessa estação.
- 2: indica que a SSM DMPDU foi descartada por apresentar um valor para o campo MID diferente de zero.
- 3: indica que a SSM DMPDU foi descartada por apresentar endereços de destino da *bridge* e da estação incompatíveis com o endereço dessa estação.
- 4: indica que a DMPDU foi descartada devido a indisponibilidade de alocação de uma instância para o processo "Reassembly".
- 5: indica que a IMPDU foi descartada por ter apresentado erro durante o processo de remontagem, ou por ter apresenatado erro durante a validação dessa, após concluído o processo de remontagem.

- 6: indica que não houve disponibilidade para armazenar na tabela "VCI_MID_table", as informações dos valores do campo MID , do campo VCI e o tipo de endereçamento utilizado pela BOM DMPDU e, portanto, essa unidade foi descartada.
- 7: indica que a COM ou EOM DMPDU foi descartada por apresentar um valor para o campo MID diferente daqueles armazenados na tabela "VCI_MID_table".

– Sinal "reass_error"

- 8: indica que houve "timeout" durante a remontagem de uma IMPDU e, portanto, o processo de remontagem foi abortado.
- 9: indica que o processo de remontagem de uma IMPDU foi abortado devido algumas das COM DMPDU apresentarem um valor para o campo SN diferente daquele esperado pelo processo "Reassembly". Isso ocorre quando as unidades chegam ao processo "Reassembly" fora de sequência ou há a perda de algumas das unidades.
- 10: indica que o processo de remontagem de uma IMPDU foi abortado devido a EOM DMPDU apresentar um valor para o campo SN diferente daquele esperado pelo processo "Reassembly". Isso ocorre quando as unidades chegam ao processo "Reassembly" fora de sequência ou há a perda de algumas das unidades.

5.2 Estudo dos casos

A validação do sistema foi realizada por etapas, de forma que envolvesse gradualmente as funções implementadas. Notar que a validação dos casos citados é acompanhado do "trace" de depuração do bloco "Node_DQDB_remote", gerado pelo simulador da ferramenta SDT. O "trace" completo de cada estudo é apresentado no Apêndice A.

5.2.1 Caso 1

A validação desse caso é observada através do envio de três sinais "MA_UNITDATA_indication" para o ambiente externo. O envio desses sinais indicam que todas as unidades DMPDU provenientes da rede local DQDB que trafegaram pela rede de transporte ATM, chegaram intactas à estação de destino, localizada na rede remota DQDB e, portanto, foram remontadas com sucesso.

```
0.0000 MA_UNITDATA_indication from MAC_Function:1 to env:1
Parameter(s) : 1000, 9000, 2, 40
0.0000 MA_UNITDATA_indication from MAC_Function:1 to env:1
Parameter(s) : 1000, 9000, 2, 60
0.0000 MA_UNITDATA_indication from MAC_Function:1 to env:1
Parameter(s) : 1000, 9000, 2, 10
```

Figura 5.15: Parte do trace de simulação do caso 1

5.2.2 Caso 2

Nesse caso podemos validar o processo "MAC_Function" no que se refere aos valores possíveis utilizados para o campo MID pelas unidades DMPDU.

```
0.0000 MA_UNITDATA_indication from MAC_Function:1 to env:1
Parameter(s) : 1000, 9000, 2, 40
0.0000 MA_UNITDATA_indication from MAC_Function:1 to env:1
Parameter(s) : 1000, 9000, 2, 60
0.0000 MAC_error from MAC_Function:1 to env:1
Parameter(s) : 2
```

Figura 5.16: Parte do trace de simulação do caso 2

5.2.3 Caso 3

Nesse caso podemos verificar a correta utilização das tabelas de roteamento "route3_table" e "route4_table" pelo processo "ATM_DQDB". Desse modo, as unidades DMPDU recebidas através dos sinais "ATM_DATA" foram descartadas e, portanto, essas unidades não serão enviadas através do sinal "BRIDGE_DATA_rx" para a estação DQDB mais próxima. O descarte foi devido a um endereço de destino inválido, isto é, um endereço que não consta na tabela "route3_table".

```
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
```

Figura 5.17: Parte do trace de simulação do caso 3

5.2.4 Caso 4

Nesse caso podemos verificar a correta utilização das tabelas de roteamento "route1_table" e "route2_table" pelo processo "DQDB_ATM". Notar que a primeira unidade DMPDU de uma IMPDU é simplesmente descartada ao apresentar um endereço de destino de *bridge* que não consta na tabela de roteamento "route1_table", enquanto que as restantes unidades pertencentes a mesma IMPDU são descartadas e acompanhadas do envio do sinal "Connection_denied".

```
0.0000 Connection_denied from DQDB_ATM:1 to env:1
Parameter(s) : 0
0.0000 Connection_denied from DQDB_ATM:1 to env:1
Parameter(s) : 1
0.0000 Connection_denied from DQDB_ATM:1 to env:1
Parameter(s) : 1
0.0000 Connection_denied from DQDB_ATM:1 to env:1
Parameter(s) : 2
0.0000 Connection_denied from DQDB_ATM:1 to env:1
Parameter(s) : 0
0.0000 Connection_denied from DQDB_ATM:1 to env:1
Parameter(s) : 1
0.0000 Connection_denied from DQDB_ATM:1 to env:1
Parameter(s) : 1
0.0000 Connection_denied from DQDB_ATM:1 to env:1
Parameter(s) : 1
0.0000 Connection_denied from DQDB_ATM:1 to env:1
Parameter(s) : 1
0.0000 Connection_denied from DQDB_ATM:1 to env:1
Parameter(s) : 2
0.0000 Connection_denied from DQDB_ATM:1 to env:1
Parameter(s) : 3
```

Figura 5.18: Parte do trace de simulação do caso 4

5.2.5 Caso 5

Nesse caso podemos verificar que todas as unidades recebidas pela *bridge* remota através do sinal "ATM_DATA" foram descartadas por terem atingido o número máximo de *bridges* que essas poderiam percorrer. Portanto, nenhuma das unidades foram enviadas para o próximo bloco através do sinal "BRIDGE_DATA_rx".

```
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
```

Figura 5.19: Parte do trace de simulação do caso 5

5.2.6 Caso 6

Nesse caso verificamos que o processo de remontagem das três unidades IMPDU foi bem sucedido, porém, essas unidades foram descartadas por apresentarem erro durante o processo de validação das mesmas.

```
0.0000 MAC_error from MAC_Function:1 to env:1
Parameter(s) : 5
0.0000 MAC_error from MAC_Function:1 to env:1
Parameter(s) : 5
0.0000 MAC_error from MAC_Function:1 to env:1
Parameter(s) : 5
```

Figura 5.20: Parte do trace de simulação do caso 6

5.2.7 Caso 7

Nesse caso obtivemos sucesso somente na remontagem da segunda e terceira IMPDU, pois não foi possível iniciar o processo de remontagem da primeira IMPDU pelo fato dessa apresentar a perda da primeira unidade DMPDU.

```
0.0000 MAC_error from MAC_Function:1 to env:1
Parameter(s) : 7
0.0000 MAC_error from MAC_Function:1 to env:1
Parameter(s) : 7
0.0000 MAC_error from MAC_Function:1 to env:1
Parameter(s) : 7
0.0000 MA_UNITDATA_indication from MAC_Function:1 to env:1
Parameter(s) : 1000, 9000, 2, 10
0.0000 MA_UNITDATA_indication from MAC_Function:1 to env:1
Parameter(s) : 1000, 9000, 2, 60
```

Figura 5.21: Parte do trace de simulação do caso 7

5.2.8 Caso 8

Nesse caso obtivemos sucesso somente na remontagem da primeira e terceira IMPDU, pois não foi possível concluir o processo de remontagem da segunda IMPDU pelo fato dessa apresentar a perda da terceira unidade DMPDU.

```
0.0000 MA_UNITDATA_indication from MAC_Function:1 to env:1
Parameter(s) : 1000, 9000, 2, 40
0.0000 reass_error from Reassembly:2 to env:1
Parameter(s) : 9
0.0000 reass_error from Reassembly:2 to env:1
Parameter(s) : 9
0.0000 reass_error from Reassembly:2 to env:1
Parameter(s) : 10
0.0000 MAC_error from MAC_Function:1 to env:1
Parameter(s) : 5
0.0000 MA_UNITDATA_indication from MAC_Function:1 to env:1
Parameter(s) : 1000, 9000, 2, 10
```

Figura 5.22: Parte do trace de simulação do caso 8

5.2.9 Caso 9

Nesse caso obtivemos sucesso somente na remontagem da segunda e terceira IMPDU, pois não foi possível concluir o processo de remontagem da primeira IMPDU pelo fato dessa apresentar a perda da última unidade DMPDU.

```
10.0000 Timeout from Reassembly:1 to Reassembly:1
10.0000 reass_error from Reassembly:1 to env:1
Parameter(s) : 8
10.0000 MAC_error from MAC_Function:1 to env:1
Parameter(s) : 5
0.0000 MA_UNITDATA_indication from MAC_Function:1 to env:1
Parameter(s) : 1000, 9000, 2, 60
0.0000 MA_UNITDATA_indication from MAC_Function:1 to env:1
Parameter(s) : 1000, 9000, 2, 10
```

Figura 5.23: Parte do trace de simulação do caso 9

5.2.10 Caso 10

Nesse caso obtivemos sucesso somente na remontagem da primeira e terceira IMPDU, pois não foi possível concluir o processo de remontagem da segunda IMPDU pelo fato da sua quarta unidade DMPDU apresentar um CRC inválido e, portanto, ser descartada.

```
0.0000 MA_UNITDATA_indication from MAC_Function:1 to env:1
Parameter(s) : 1000, 9000, 2, 40
0.0000 MAC_error from MAC_Function:1 to env:1
Parameter(s) : 0
0.0000 reass_error from Reassembly:2 to env:1
Parameter(s) : 9
0.0000 reass_error from Reassembly:2 to env:1
Parameter(s) : 10
0.0000 MAC_error from MAC_Function:1 to env:1
Parameter(s) : 5
0.0000 MA_UNITDATA_indication from MAC_Function:1 to env:1
Parameter(s) : 1000, 9000, 2, 10
```

Figura 5.24: Parte do trace de simulação do caso 10

5.3 Conclusão

Este capítulo apresentou os procedimentos de simulação adotados e "traces" de depuração para as mais diversas situações. Foi verificado que a linguagem SDL é bastante apropriada para a especificação funcional do sistema de interconexão de redes locais DQDB via uma rede de transporte ATM, devido a facilidade de descrição em diferentes níveis hierárquicos de abstração, a flexibilidade dos tipos abstratos de dados e aos recursos para tratamento de eventos temporais.

Capítulo 6

Conclusão

Neste trabalho, foi apresentado uma proposta de integração de redes DQDB com ATM para serviços não orientados à conexão, utilizando estrutura de *bridge* e definindo um formato adequado de endereçamento. Foram estabelecidas as descrições formais e funcionais para esta proposta e através de uma implentação virtual utilizando a linguagem SDL, foi possível validar os procedimentos apresentados, através de simulações de diversas situações que forneceram resultados esperados.

Em particular, cabe ressaltar a importância da implementação da interconexão através de uma linguagem de simulação e especificação formal de protocolos de redes, considerando aspectos de validação como também de análises de desempenho. Ainda, deve-se considerar a tendência encontrada hoje no processo de engenharia de hardware que faz uso intensivo de linguagens de alto nível na descrição, concepção e testes de análise e validação. Dentro deste último aspecto, uma proposta já formalizada a nível de linguagens pode trazer facilidades no processo de obtenção de protótipos funcionais.

Sem dúvida, procedimentos semelhantes podem ser extendidos para outras propostas, tipos de serviços e interconexões de outras naturezas. Uma extensão natural deste trabalho é a utilização do simulador para efetuar análise de desempenho da proposta aqui apresentada. Além disso, a extensão do procedimento aqui adotado para serviços isócronos e orientados à conexão utilizando o protocolo DQDB surge como um interessante desafio, principalmente levando em consideração que estes ainda não apresentam recomendações.

Finalmente, ainda dentro da idéia de implementação de simuladores, propostas para que interconexões do tipo aqui tratado suportem serviços específicos como "*browse*", "propagação de rotas", fundamentais na *internet*, merecem estudos e análise na direção de validação e análise de desempenho.

Acreditamos que este trabalho alcançou os seus objetivos dentro dos aspectos de formação, apresentação de uma proposta, implementação de um simulador e validação através de inúmeros testes e análises.

Apêndice A

"Traces" de depuração

A.1 "Trace" do Caso 1

```
Signal log for system Internetworking with unit Block Node_DQDB_remote
on file /home/home1/luciano/sdl/teste/simu2.log
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1 to
Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. BOM, 0, 10 .), (. (. 0, 1, 168 .), (. (. false, 1, 9000 .),
(. false, 0, 1000 .), 1, 3, 2, true, true, 0, 2 .), 40, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1 to
Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 1, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 2, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1 to
Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. EOM, 3, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 3, true,
(. 0, 1, 168 .) .), (. 36, true .) .) .), BUS_B
```

```
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1 to
Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. BOM, 0, 10 .), (. (. 0, 2, 224 .), (. (. false, 1, 9000 .),
(. false, 0, 1000 .), 1, 0, 2, false, true, 0, 2 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1 to
Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 1, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 MA_UNITDATA_indication from MAC_Function:1 to env:1
Parameter(s) : 1000, 9000, 2, 40
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1 to
Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 2, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1 to
Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 3, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1 to
Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 4, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1 to
Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. EOM, 5, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, true,
(. 0, 2, 224 .) .), (. 4, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1 to
Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. SSM, 0, 0 .), (. (. 0, 3, 40 .), (. (. false, 1, 9000 .),
(. false, 0, 1000 .), 1, 1, 2, true, true, 0, 2 .), 10, 1, true,
(. 0, 3, 40 .) .), (. 40, true .) .) .), BUS_B
0.0000 MA_UNITDATA_indication from MAC_Function:1 to env:1
Parameter(s) : 1000, 9000, 2, 60
```

```
0.0000 MA_UNITDATA_indication from MAC_Function:1 to env:1
Parameter(s) : 1000, 9000, 2, 10
```

A.2 "Trace" do Caso 2

```
Signal log for system Internetworking with unit Block Node_DQDB_remote
on file /home/home1/luciano/sdl/teste/simu1.log
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1 to
Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. BOM, 0, 10 .), (. (. 0, 1, 168 .), (. (. false, 1, 9000 .),
(. false, 0, 1000 .), 1, 3, 2, true, true, 0, 2 .), 40, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1 to
Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 1, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 2, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1 to
Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. EOM, 3, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 3, true,
(. 0, 1, 168 .) .), (. 36, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1 to
Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. BOM, 0, 10 .), (. (. 0, 2, 224 .), (. (. false, 1, 9000 .),
(. false, 0, 1000 .), 1, 0, 2, false, true, 0, 2 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1 to
Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 1, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
```

```
0.0000 MA_UNITDATA_indication from MAC_Function:1 to env:1
Parameter(s) : 1000, 9000, 2, 40
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1 to
Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 2, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1 to
Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 3, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1 to
Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 4, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1 to
Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. EOM, 5, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, true,
(. 0, 2, 224 .) .), (. 4, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1 to
Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. SSM, 0, 10 .), (. (. 0, 3, 40 .), (. (. false, 1, 9000 .),
(. false, 0, 1000 .), 1, 1, 2, true, true, 0, 2 .), 10, 1, true,
(. 0, 3, 40 .) .), (. 40, true .) .) .), BUS_B
0.0000 MA_UNITDATA_indication from MAC_Function:1 to env:1
Parameter(s) : 1000, 9000, 2, 60
0.0000 MAC_error from MAC_Function:1 to env:1
Parameter(s) : 2
```

A.3 "Trace" do Caso 3

```
Signal log for system Internetworking with unit Block Bridge_remote
on file /home/home1/luciano/sdl/teste/simu3.log
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
Parameter(s) : (. (. BOM, 0, 1 .), (. (. 0, 1, 168 .),
(. (. false, 1, 20000 .), (. false, 0, 1000 .), 1, 3, 2, true, true, 0, 2 .),
40, 0, false, (. 0, 0, 0 .) .), (. 44, true .) .), 1
```

```
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
Parameter(s) : (. (. COM, 1, 1 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .), 1
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
Parameter(s) : (. (. COM, 2, 1 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .), 1
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
Parameter(s) : (. (. EOM, 3, 1 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 3, true,
(. 0, 1, 168 .) .), (. 36, true .) .), 1
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
Parameter(s) : (. (. BOM, 0, 2 .), (. (. 0, 2, 224 .),
(. (. false, 1, 20000 .), (. false, 0, 1000 .), 1, 0, 2, false, true, 0, 2 .),
60, 0, false, (. 0, 0, 0 .) .), (. 44, true .) .), 2
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
Parameter(s) : (. (. COM, 1, 2 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .), 2
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
Parameter(s) : (. (. COM, 2, 2 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .), 2
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
Parameter(s) : (. (. COM, 3, 2 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .), 2
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
Parameter(s) : (. (. COM, 4, 2 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .), 2
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
Parameter(s) : (. (. EOM, 5, 2 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, true,
(. 0, 2, 224 .) .),(. 4, true .) .), 2
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
Parameter(s) : (. (. SSM, 0, 0 .), (. (. 0, 3, 40 .),
(. (. false, 1, 20000 .),(. false, 0, 1000 .), 1, 1, 2, true, true, 0, 2 .),
10, 1, true, (. 0, 3, 40 .) .), (. 40, true .) .), 1
```

A.4 "Trace" do Caso 4

```
Signal log for system Internetworking with unit Block Bridge_local
on file /home/home1/luciano/sdl/teste/simu4.log
0.0000 BRIDGE_DATA_tx from MAC_Function_rx_local:1 to DQDB_ATM:1
Parameter(s) : (. (. BOM, 0, 1 .), (. (. 0, 1, 168 .), (. (. false, 3, 9000 .)
,(. false, 0, 1000 .), 1, 3, 2, true, true, 0, 2 .), 40, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .)
0.0000 Connection_denied from DQDB_ATM:1 to env:1
Parameter(s) : 0
0.0000 BRIDGE_DATA_tx from MAC_Function_rx_local:1 to DQDB_ATM:1
Parameter(s) : (. (. COM, 1, 1 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .)
0.0000 Connection_denied from DQDB_ATM:1 to env:1
Parameter(s) : 1
0.0000 BRIDGE_DATA_tx from MAC_Function_rx_local:1 to DQDB_ATM:1
Parameter(s) : (. (. COM, 2, 1 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .)
0.0000 Connection_denied from DQDB_ATM:1 to env:1
Parameter(s) : 1
0.0000 BRIDGE_DATA_tx from MAC_Function_rx_local:1 to DQDB_ATM:1
Parameter(s) : (. (. EOM, 3, 1 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 3, true,
(. 0, 1, 168 .) .), (. 36, true .) .)
0.0000 Connection_denied from DQDB_ATM:1 to env:1
Parameter(s) : 2
0.0000 BRIDGE_DATA_tx from MAC_Function_rx_local:1 to DQDB_ATM:1
Parameter(s) : (. (. BOM, 0, 2 .), (. (. 0, 2, 224 .), (. (. false, 3, 9000 .)
,(. false, 0, 1000 .), 1, 0, 2, false, true, 0, 2 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .)
0.0000 Connection_denied from DQDB_ATM:1 to env:1
Parameter(s) : 0
0.0000 BRIDGE_DATA_tx from MAC_Function_rx_local:1 to DQDB_ATM:1
Parameter(s) : (. (. COM, 1, 2 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .)
0.0000 Connection_denied from DQDB_ATM:1 to env:1
Parameter(s) : 1
0.0000 BRIDGE_DATA_tx from MAC_Function_rx_local:1 to DQDB_ATM:1
Parameter(s) : (. (. COM, 2, 2 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .)
0.0000 Connection_denied from DQDB_ATM:1 to env:1
Parameter(s) : 1
```

```
0.0000 BRIDGE_DATA_tx from MAC_Function_rx_local:1 to DQDB_ATM:1
Parameter(s) : (. (. COM, 3, 2 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .)
0.0000 Connection_denied from DQDB_ATM:1 to env:1
Parameter(s) : 1
0.0000 BRIDGE_DATA_tx from MAC_Function_rx_local:1 to DQDB_ATM:1
Parameter(s) : (. (. COM, 4, 2 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .)
0.0000 Connection_denied from DQDB_ATM:1 to env:1
Parameter(s) : 1
0.0000 BRIDGE_DATA_tx from MAC_Function_rx_local:1 to DQDB_ATM:1
Parameter(s) : (. (. EOM, 5, 2 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, true,
(. 0, 2, 224 .) .), (. 4, true .) .)
0.0000 Connection_denied from DQDB_ATM:1 to env:1
Parameter(s) : 2
0.0000 BRIDGE_DATA_tx from MAC_Function_rx_local:1 to DQDB_ATM:1
Parameter(s) : (. (. SSM, 0, 0 .), (. (. 0, 3, 40 .), (. (. false, 3, 9000 .)
, (. false, 0, 1000 .), 1, 1, 2, true, true, 0, 2 .), 10, 1, true,
(. 0, 3, 40 .) .), (. 40, true .) .)
0.0000 Connection_denied from DQDB_ATM:1 to env:1
Parameter(s) : 3
```

A.5 "Trace" do Caso 5

```
Signal log for system Internetworking with unit Block Bridge_remote
on file /home/home1/luciano/sdl/teste/simu3.log
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
Parameter(s) : (. (. BOM, 0, 1 .), (. (. 0, 1, 168 .),
(. (. false, 1, 20000 .), (. false, 0, 1000 .), 1, 3, 2, true, true, 0, 1 .),
40, 0, false, (. 0, 0, 0 .) .), (. 44, true .) .), 1
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
Parameter(s) : (. (. COM, 1, 1 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .), 1
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
Parameter(s) : (. (. COM, 2, 1 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .), 1
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
Parameter(s) : (. (. EOM, 3, 1 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 3, true,
(. 0, 1, 168 .) .), (. 36, true .) .), 1
```

```
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
Parameter(s) : (. (. BOM, 0, 2 .), (. (. 0, 2, 224 .),
(. (. false, 1, 20000 .), (. false, 0, 1000 .), 1, 0, 2, false, true, 0, 1 .),
60, 0, false, (. 0, 0, 0 .) .), (. 44, true .) .), 2
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
Parameter(s) : (. (. COM, 1, 2 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .), 2
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
Parameter(s) : (. (. COM, 2, 2 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .), 2
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
Parameter(s) : (. (. COM, 3, 2 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .), 2
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
Parameter(s) : (. (. COM, 4, 2 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .), 2
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
Parameter(s) : (. (. EOM, 5, 2 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, true,
(. 0, 2, 224 .) .),(. 4, true .) .), 2
0.0000 ATM_DATA from DQDB_ATM:1 to ATM_DQDB:1
Parameter(s) : (. (. SSM, 0, 0 .), (. (. 0, 3, 40 .),
(. (. false, 1, 20000 .),(. false, 0, 1000 .), 1, 1, 2, true, true, 0, 1 .),
10, 1, true, (. 0, 3, 40 .) .), (. 40, true .) .), 1
```

A.6 "Trace" do Caso 6

```
Signal log for system Internetworking with unit Block Node_DQDB_remote
on file /home/home1/luciano/sdl/teste/simu7.log
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. BOM, 0, 10 .), (. (. 0, 1, 168 .), (. (. false, 1, 9000 .),
(. false, 0, 1000 .), 1, 3, 2, true, true, 0, 2 .), 40, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 1, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 2, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. EOM, 3, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 3, true,
(. 0, 2, 168 .) .), (. 36, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. BOM, 0, 10 .), (. (. 0, 2, 224 .), (. (. false, 1, 9000 .),
(. false, 0, 1000 .), 1, 0, 2, false, true, 0, 2 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 1, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 MAC_error from MAC_Function:1 to env:1
Parameter(s) : 5
```

```
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 2, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 3, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 4, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .), (. false, 0, 0 .)
, 0, 0, 0, false, false, 0, 0 .), 60, 0, false, (. 0, 0, 0 .) .),
(. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. EOM, 5, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .), (. false, 0, 0 .)
, 0, 0, 0, false, false, 0, 0 .), 60, 0, true, (. 0, 3, 224 .) .),
(. 4, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. SSM, 0, 0 .), (. (. 0, 3, 40 .), (. (. false, 1, 9000 .),
(. false, 0, 1000 .), 1, 1, 2, true, true, 0, 2 .), 10, 1, true,
(. 0, 4, 40 .) .), (. 40, true .) .) .), BUS_B
0.0000 MAC_error from MAC_Function:1 to env:1
Parameter(s) : 5
0.0000 MAC_error from MAC_Function:1 to env:1
Parameter(s) : 5
```

A.7 "Trace" do Caso 7

```
Signal log for system Internetworking with unit Block Node_DQDB_remote
on file /home/home1/luciano/sdl/teste/simu8.log
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
 to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 1, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false,0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
```

```
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 2, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false,0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 0, false,
(. 0, 0, 0 .) ,(. 44, true .) .) .), BUS_B
0.0000 MAC_error from MAC_Function:1 to env:1
Parameter(s) : 7
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. EOM, 3, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 3, true,
(. 0, 1, 168 .).), (. 36, true .) .) .), BUS_B
0.0000 MAC_error from MAC_Function:1 to env:1
Parameter(s) : 7
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. BOM, 0, 10 .), (. (. 0, 2, 224 .), (. (. false, 1, 9000 .),
(. false, 0, 1000 .), 1, 0, 2, false, true, 0, 2 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 MAC_error from MAC_Function:1 to env:1
Parameter(s) : 7
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 1, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 2, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 3, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
```

```
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 4, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. EOM, 5, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, true,
(. 0, 2, 224 .) .), (. 4, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. SSM, 0, 0 .), (. (. 0, 3, 40 .), (. (. false, 1, 9000 .),
(. false, 0, 1000 .), 1, 1, 2, true, true, 0, 2 .), 10, 1, true,
(. 0, 3, 40 .) .), (. 40, true .) .) .), BUS_B
0.0000 Finish_reass from Reassembly:2 to MAC_Function:1
Parameter(s) : Reassembly:2
0.0000 make_IMPDU from Reassembly:2 to MAC_Function:1
Parameter(s) : (. (. 0, 2, 224 .), (. (. false, 1, 9000 .),
(. false, 0, 1000 .), 1, 0, 2, false, true, 0, 2 .), 60, 0, true,
(. 0, 2, 224 .) .), 224
0.0000 MA_UNITDATA_indication from MAC_Function:1 to env:1
Parameter(s) : 1000, 9000, 2, 10
0.0000 MA_UNITDATA_indication from MAC_Function:1 to env:1
Parameter(s) : 1000, 9000, 2, 60
```

A.8 "Trace" do Caso 8

```
Signal log for system Internetworking with unit Block Node_DQDB_remote
on file /home/home1/luciano/sdl/teste/simu9.log
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. BOM, 0, 10 .), (. (. 0, 1, 168 .), (. (. false, 1, 9000 .),
(. false, 0, 1000 .), 1, 3, 2, true, true, 0, 2 .), 40, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
```

```
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 1, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 2, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. EOM, 3, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 3, true,
(. 0, 1, 168 .) .), (. 36, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. BOM, 0, 10 .), (. (. 0, 2, 224 .), (. (. false, 1, 9000 .),
(. false, 0, 1000 .), 1, 0, 2, false, true, 0, 2 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Finish_reass from Reassembly:1 to MAC_Function:1
Parameter(s) : Reassembly:1
0.0000 make_IMPDU from Reassembly:1 to MAC_Function:1
Parameter(s) : (. (. 0, 1, 168 .), (. (. false, 1, 9000 .),
(. false, 0, 1000 .), 1, 3, 2, true, true, 0, 2 .), 40, 3, true,
(. 0, 1, 168 .) .), 168
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 1, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 MA_UNITDATA_indication from MAC_Function:1 to env:1
Parameter(s) : 1000, 9000, 2, 40
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 3, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
```

```
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 4, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 reass_error from Reassembly:2 to env:1
Parameter(s) : 9
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. EOM, 5, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, true,
(. 0, 2, 224 .) .), (. 4, true .) .) .), BUS_B
0.0000 reass_error from Reassembly:2 to env:1
Parameter(s) : 9
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. SSM, 0, 0 .), (. (. 0, 3, 40 .), (. (. false, 1, 9000 .),
(. false,  0, 1000 .), 1, 1, 2, true, true, 0, 2 .), 10, 1, true,
(. 0, 3, 40 .) .), (. 40, true .) .) .), BUS_B
0.0000 reass_error from Reassembly:2 to env:1
Parameter(s) : 10
0.0000 Finish_reass from Reassembly:2 to MAC_Function:1
Parameter(s) : Reassembly:2
0.0000 make_IMPDU from Reassembly:2 to MAC_Function:1
Parameter(s) : (. (. 0, 2, 224 .), (. (. false, 1, 9000 .),
(. false, 0, 1000 .), 1, 0, 2, false, true, 0, 2 .), 60, 0, false,
(. 0, 0, 0 .) .), 88
0.0000 MAC_error from MAC_Function:1 to env:1
Parameter(s) : 5
0.0000 MA_UNITDATA_indication from MAC_Function:1 to env:1
Parameter(s) : 1000, 9000, 2, 10
```

A.9 "Trace" do Caso 9

```
Signal log for system Internetworking with unit Block Node_DQDB_remote
on file /home/home1/luciano/sdl/teste/simu10.log
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. BOM, 0, 10 .), (. (. 0, 1, 168 .), (. (. false, 1, 9000 .),
(. false, 0, 1000,  .), 1, 3, 2, true, true, 0, 2 .), 40, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 1, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 2, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. BOM, 0, 10 .), (. (. 0, 2, 224 .), (. (. false, 1, 9000 .),
(. false, 0, 1000 .), 1, 0, 2, false, true, 0, 2 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
10.0000 Timeout from Reassembly:1 to Reassembly:1
10.0000 reass_error from Reassembly:1 to env:1
Parameter(s) : 8
10.0000 Finish_reass from Reassembly:1 to MAC_Function:1
Parameter(s) : Reassembly:1
0.0000 make_IMPDU from Reassembly:1 to MAC_Function:1
Parameter(s) : (. (. 0, 1, 168 .), (. (. false, 1, 9000 .),
(. false, 0, 1000 .), 1, 3, 2, true, true, 0, 2 .), 40, 0, false,
(. 0, 0, 0 .) .), 132
10.0000 MAC_error from MAC_Function:1 to env:1
Parameter(s) : 5
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 1, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
```

```
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 2, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 3, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 4, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. EOM, 5, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, true,
(. 0, 2, 224 .) .), (. 4, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. SSM, 0, 0 .), (. (. 0, 3, 40 .), (. (. false, 1, 9000 .),
(. false, 0, 1000 .), 1, 1, 2, true, true, 0, 2 .), 10, 1, true,
(. 0, 3, 40 .) .), (. 40, true .) .) .), BUS_B
0.0000 Finish_reass from Reassembly:2 to MAC_Function:1
Parameter(s) : Reassembly:2
10.0000 make_IMPDU from Reassembly:2 to MAC_Function:1
Parameter(s) : (. (. 0, 2, 224 .), (. (. false, 1, 9000 .),
(. false, 0, 1000 .), 1, 0, 2, false, true, 0, 2 .), 60, 0, true,
(. 0, 2, 224 .) .), 224
0.0000 MA_UNITDATA_indication from MAC_Function:1 to env:1
Parameter(s) : 1000, 9000, 2, 60
0.0000 MA_UNITDATA_indication from MAC_Function:1 to env:1
Parameter(s) : 1000, 9000, 2, 10
```

A.10 "Trace" do Caso 10

```
Signal log for system Internetworking with unit Block Node_DQDB_remote
on file /home/home1/luciano/sdl/teste/simu11.log
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. BOM, 0, 10 .), (. (. 0, 1, 168 .), (. (. false, 1, 9000 .),
(. false, 0, 1000 .), 1, 3, 2, true, true, 0, 2 .), 40, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 1, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 2, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. EOM, 3, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 40, 3, true,
(. 0, 1, 168 .) .), (. 36, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. BOM, 0, 10 .), (. (. 0, 2, 224 .), (. (. false, 1, 9000 .),
(. false, 0, 1000 .), 1, 0, 2, false, true, 0, 2 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Finish_reass from Reassembly:1 to MAC_Function:1
Parameter(s) : Reassembly:1
0.0000 make_IMPDU from Reassembly:1 to MAC_Function:1
Parameter(s) : (. (. 0, 1, 168 .), (. (. false, 1, 9000 .),
(. false, 0, 1000 .), 1, 3, 2, true, true, 0, 2 .), 40, 3, true,
(. 0, 1, 168 .) .), 168
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 1, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
```

```
0.0000 MA_UNITDATA_indication from MAC_Function:1 to env:1
Parameter(s) : 1000, 9000, 2, 40
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 2, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 3, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, false .) .) .), BUS_B
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. COM, 4, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, false,
(. 0, 0, 0 .) .), (. 44, true .) .) .), BUS_B
0.0000 MAC_error from MAC_Function:1 to env:1
Parameter(s) : 0
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. EOM, 5, 10 .), (. (. 0, 0, 0 .), (. (. false, 0, 0 .),
(. false, 0, 0 .), 0, 0, 0, false, false, 0, 0 .), 60, 0, true,
(. 0, 2, 224 .) .), (. 4, true .) .) .), BUS_B
0.0000 reass_error from Reassembly:2 to env:1
Parameter(s) : 9
0.0000 Slot_DQDB_indication_rx from Common_Function_tx_remote:1
to Common_Function:1
Parameter(s) : (. (. true, false, false, 0, 0 .), (. 1, 1, 1, true .),
(. (. SSM, 0, 0 .), (. (. 0, 3, 40 .), (. (. false, 1, 9000 .),
(. false, 0, 1000 .), 1, 1, 2, true, true, 0, 2 .), 10, 1, true,
(. 0, 3, 40 .) .), (. 40, true .) .) .), BUS_B
0.0000 reass_error from Reassembly:2 to env:1
Parameter(s) : 10
0.0000 Finish_reass from Reassembly:2 to MAC_Function:1
Parameter(s) : Reassembly:2
0.0000 make_IMPDU from Reassembly:2 to MAC_Function:1
Parameter(s) : (. (. 0, 2, 224 .), (. (. false, 1, 9000 .),
(. false, 0, 1000 .), 1, 0, 2, false, true, 0, 2 .), 60, 0, false,
(. 0, 0, 0 .) .), 132
```

```
0.0000 MAC_error from MAC_Function:1 to env:1
Parameter(s) : 5
0.0000 MA_UNITDATA_indication from MAC_Function:1 to env:1
Parameter(s) : 1000, 9000, 2, 10
```

Apêndice B

Modelo SDL do sistema de interconexão de redes DQDB/ATM

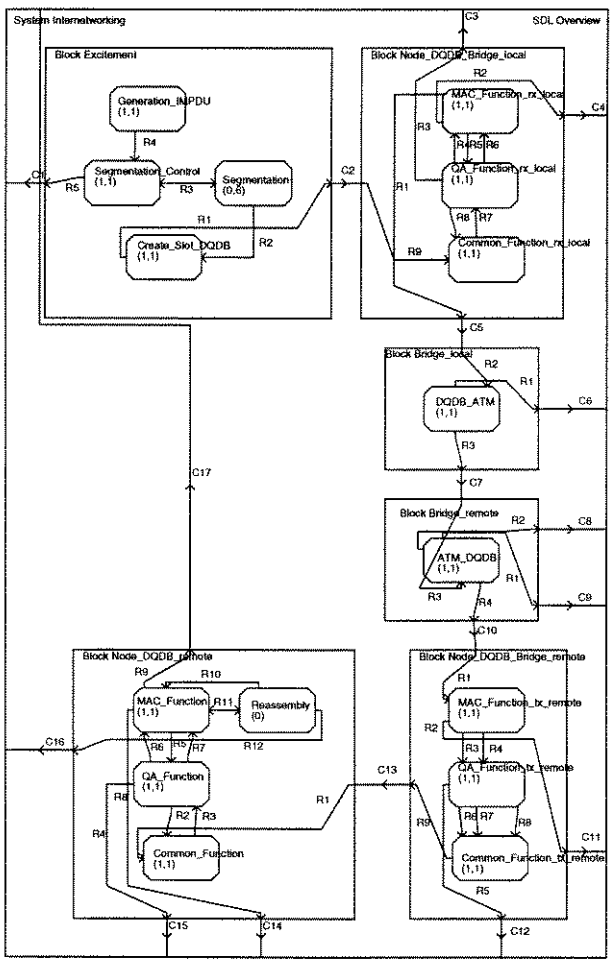

Figura B.1: "Overview SDL"

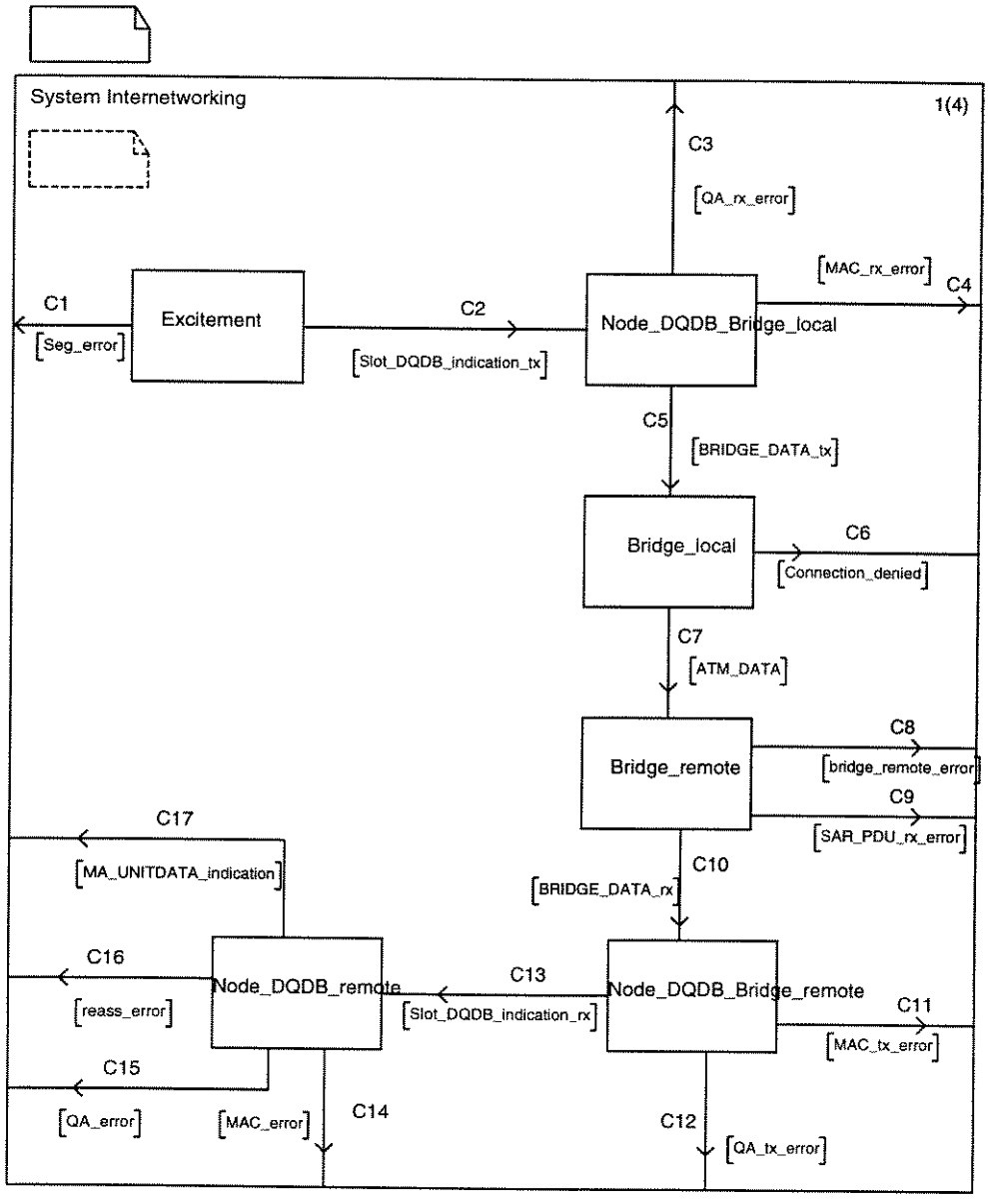

Figura B.2: "System Internetworking"

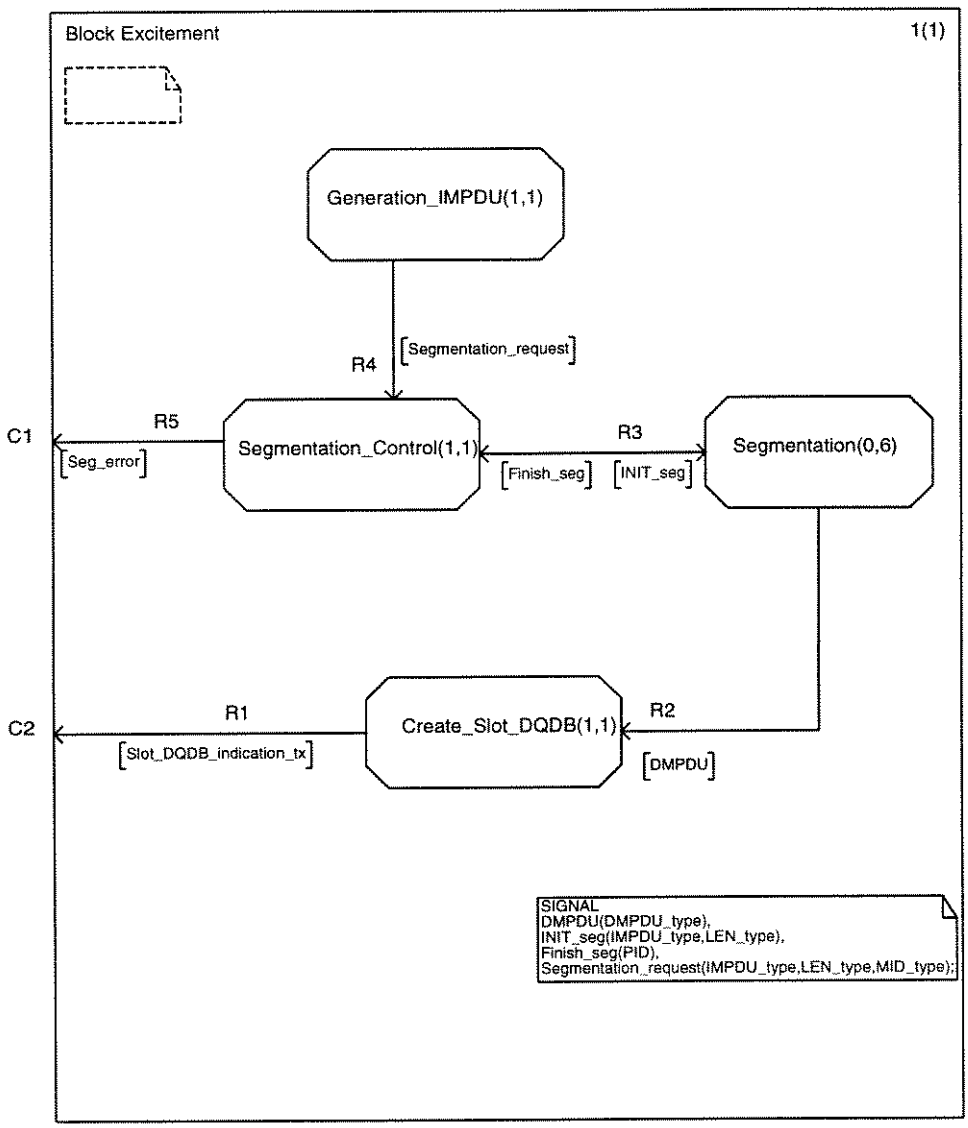

Figura B.3: "Bloco Excitement"

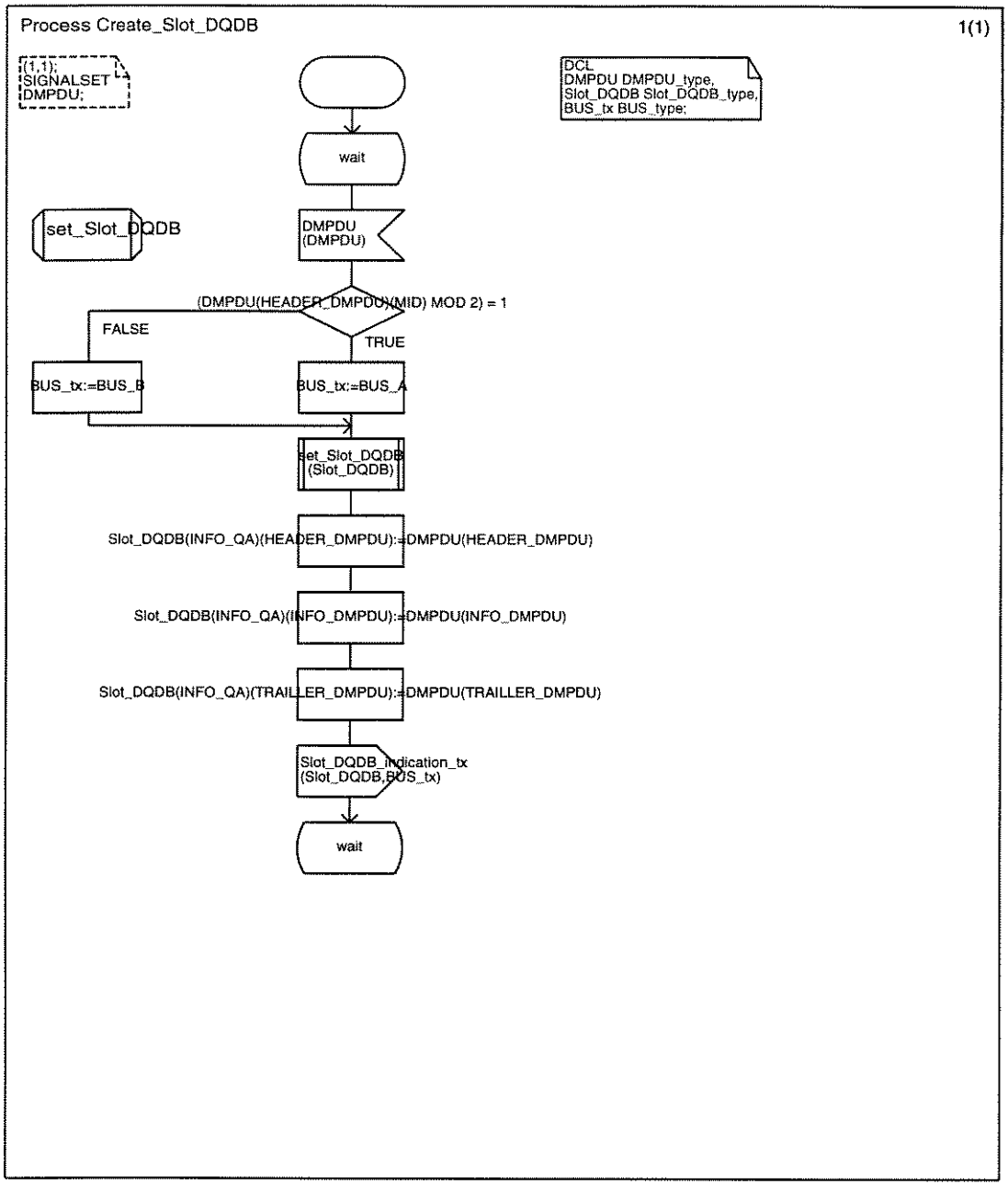

Figura B.4: "Processo Create_Slot_DQDB"

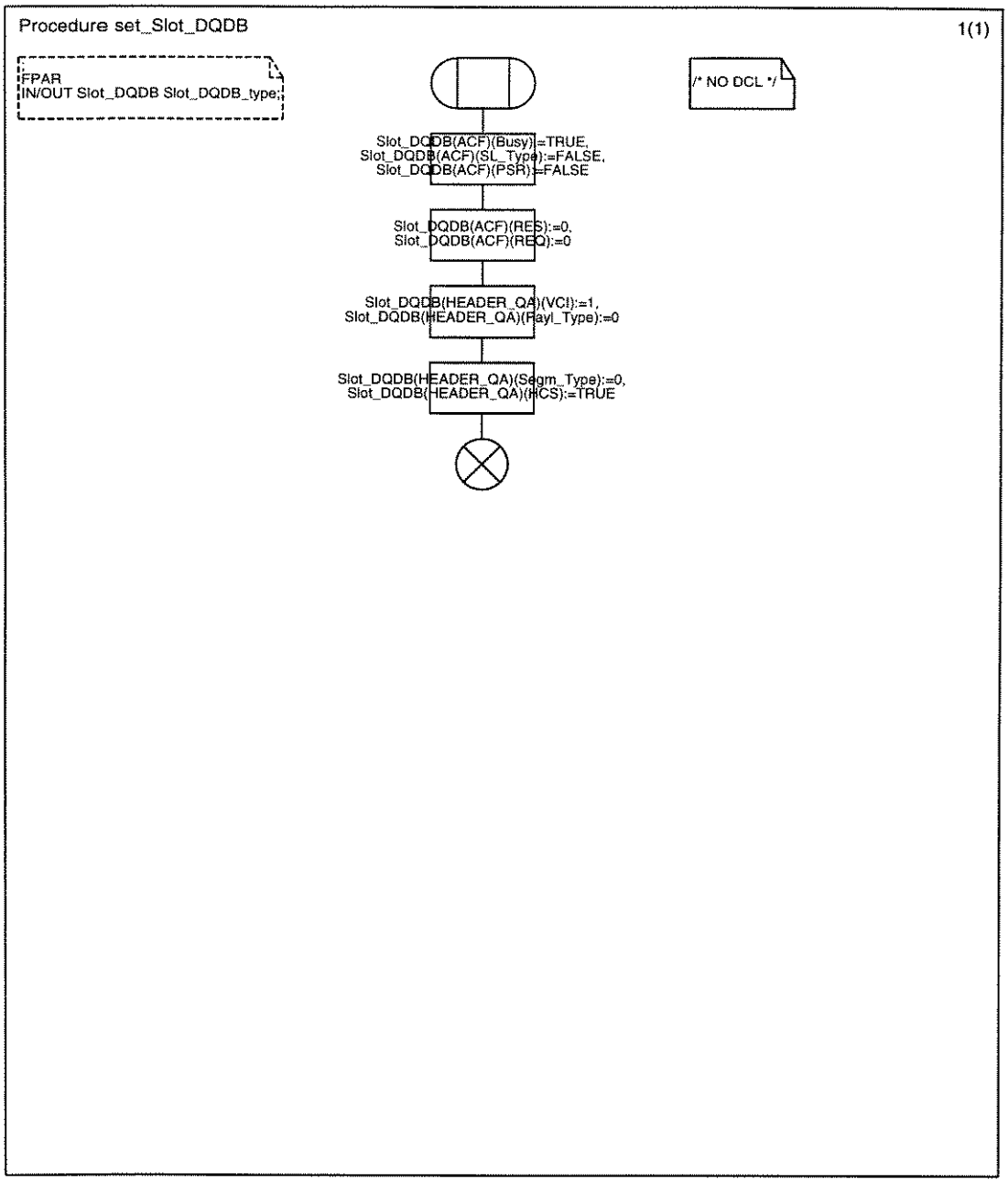

Figura B.5: "Procedimento set_Slot_DQDB"

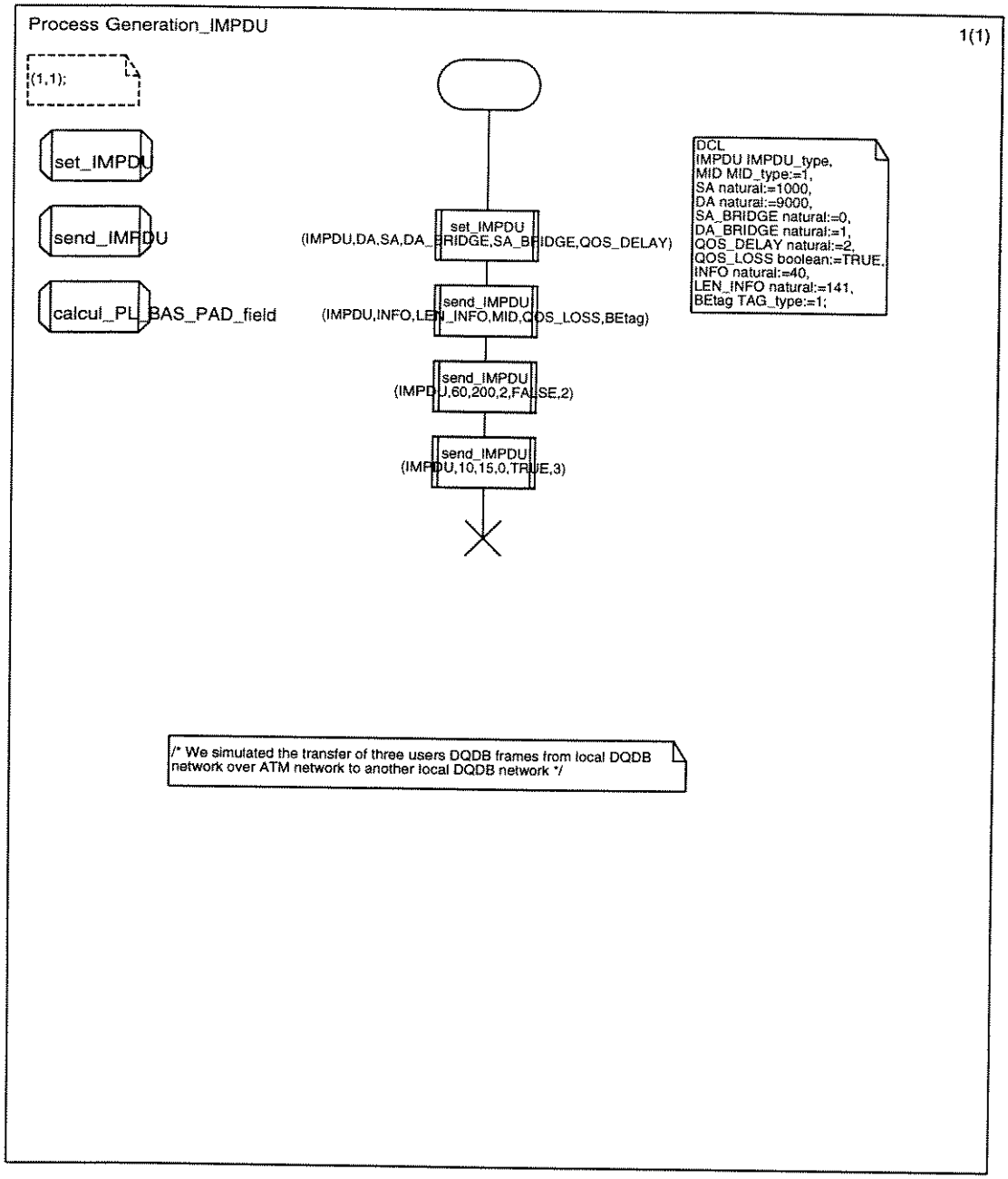

Figura B.6: "Processo Generation_IMPDU"

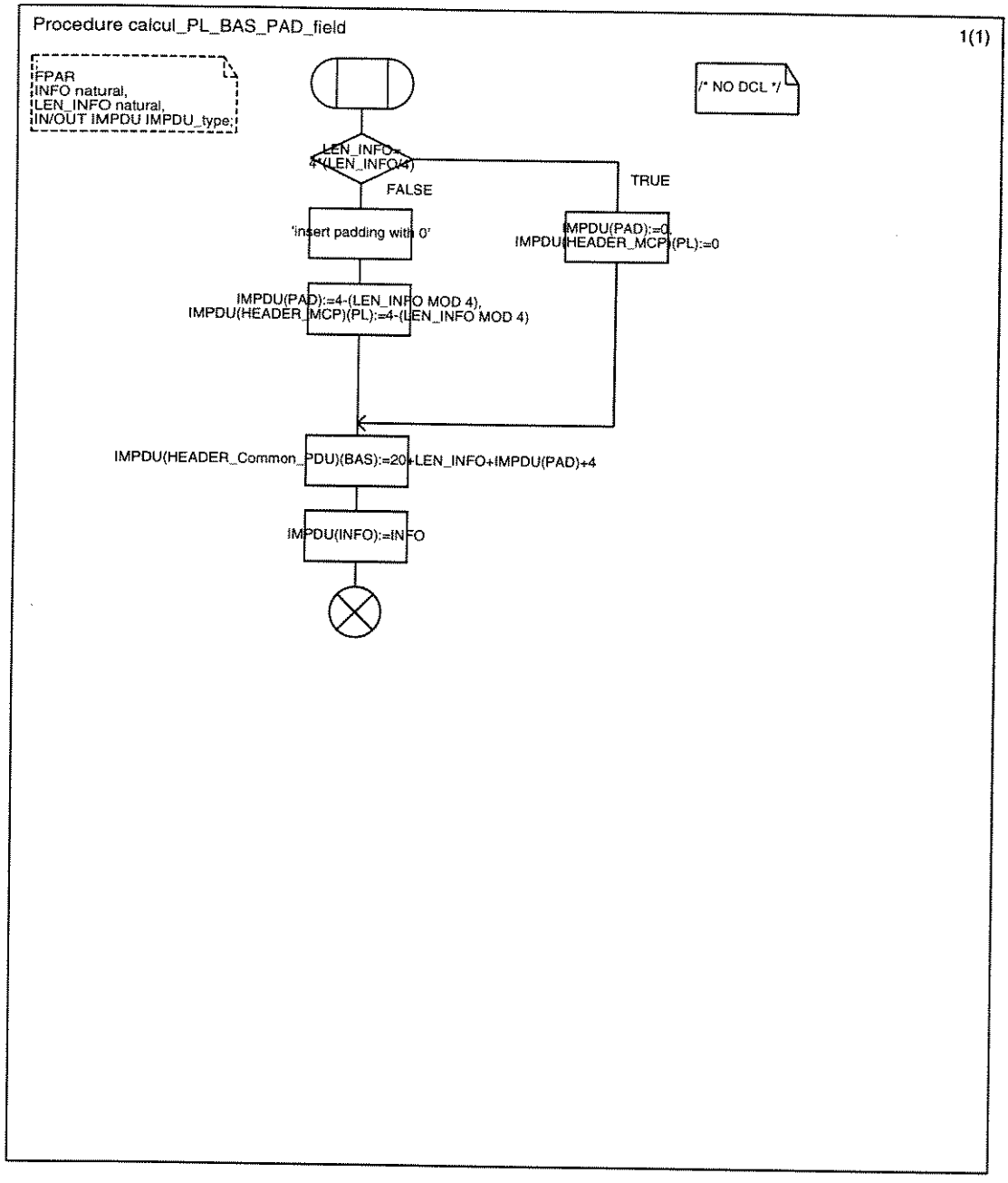

Figura B.7: "Procedimento calcul_PL_BAS_PAD_field"

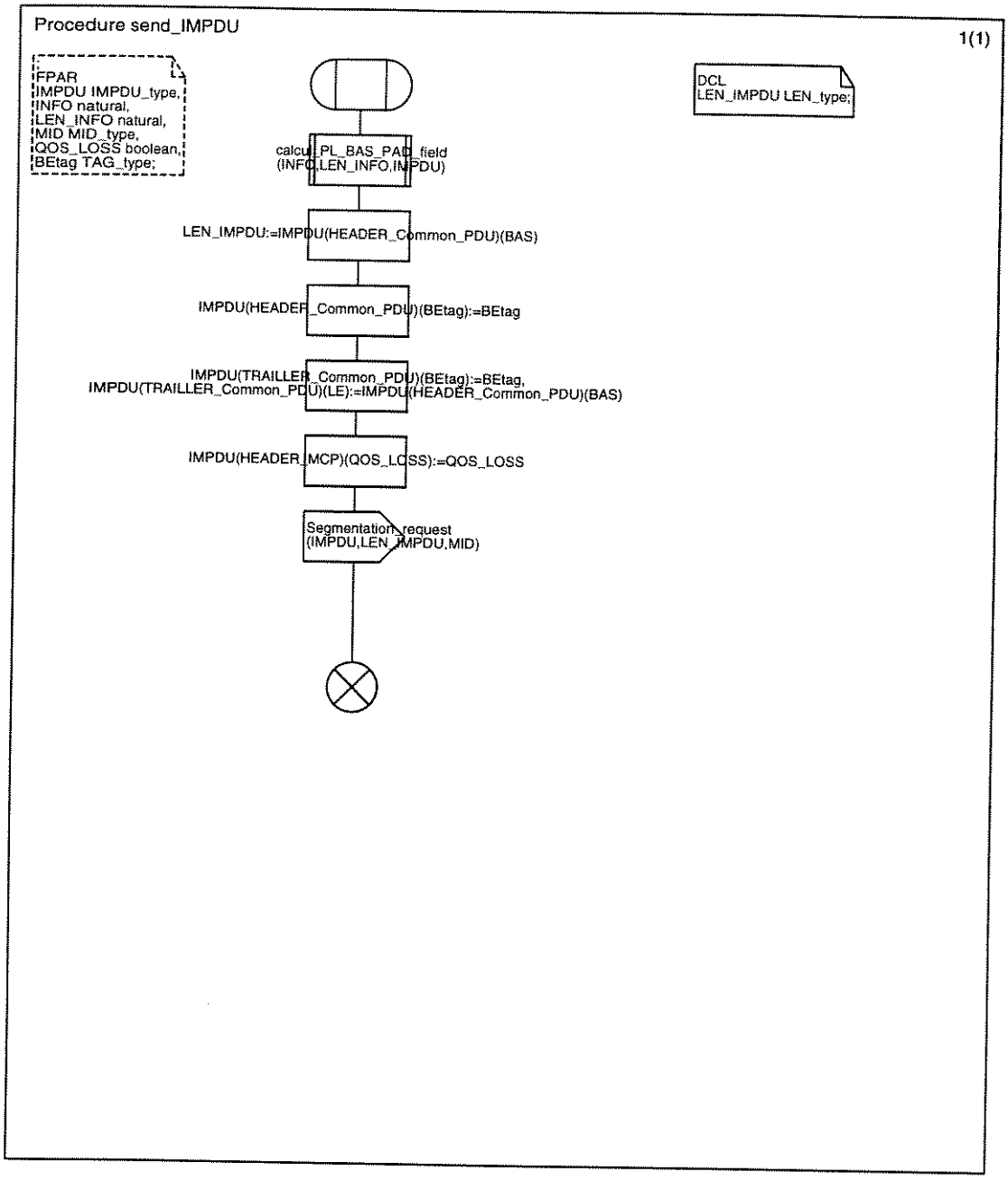

Figura B.8: "Procedimento send_IMPDU"

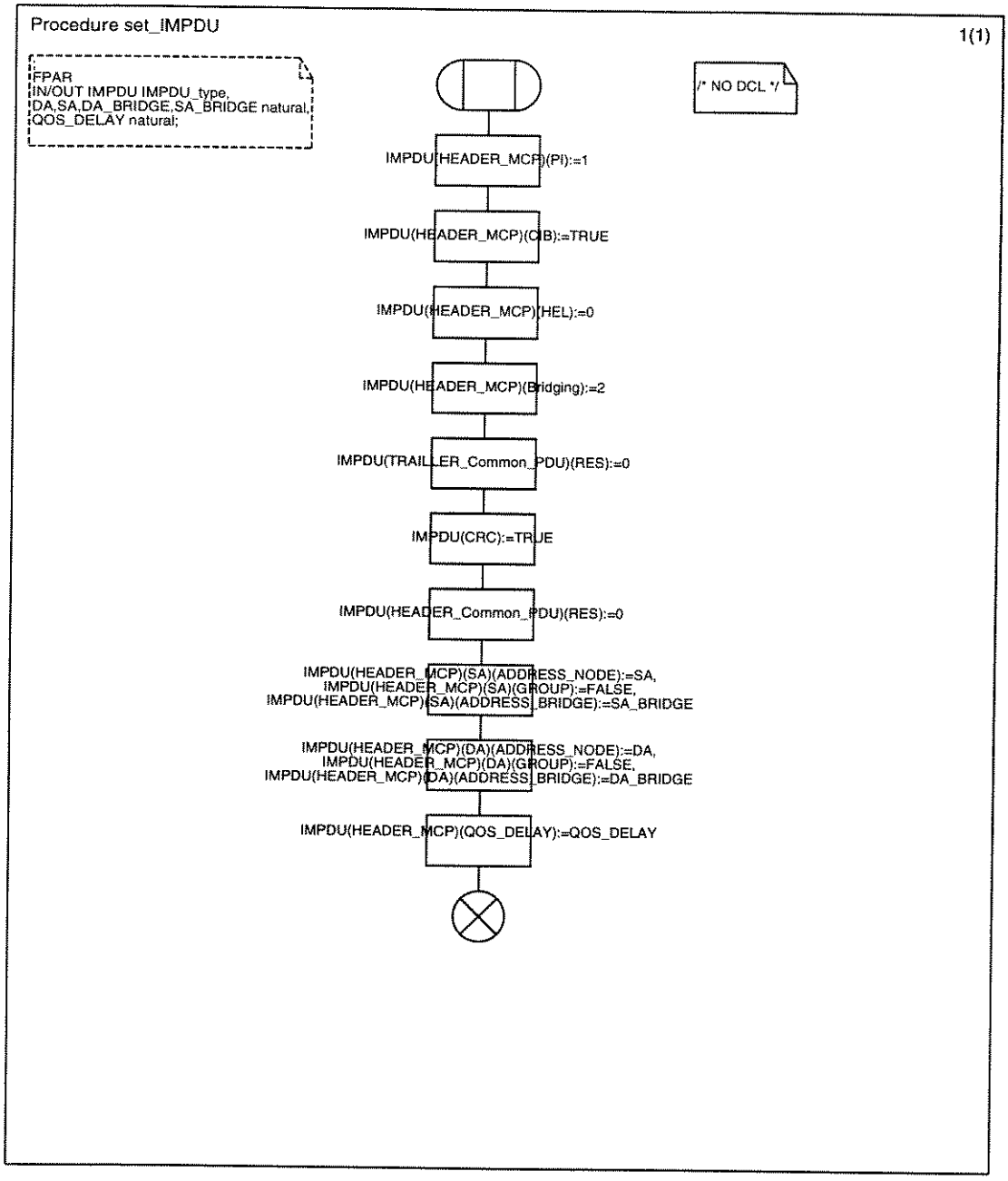

Figura B.9: "Procedimento set_IMPDU"

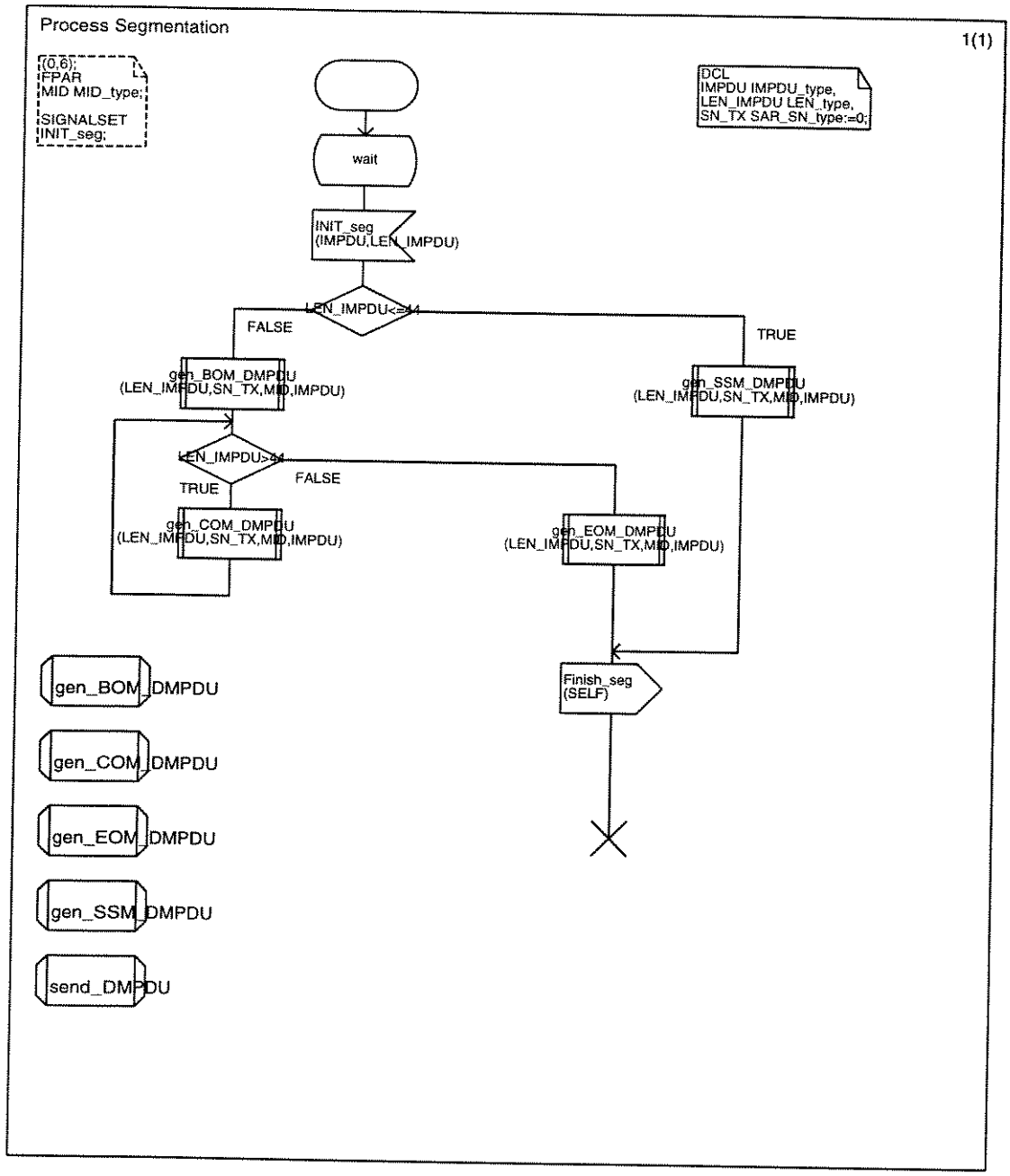

Figura B.10: "Processo Segmentation"

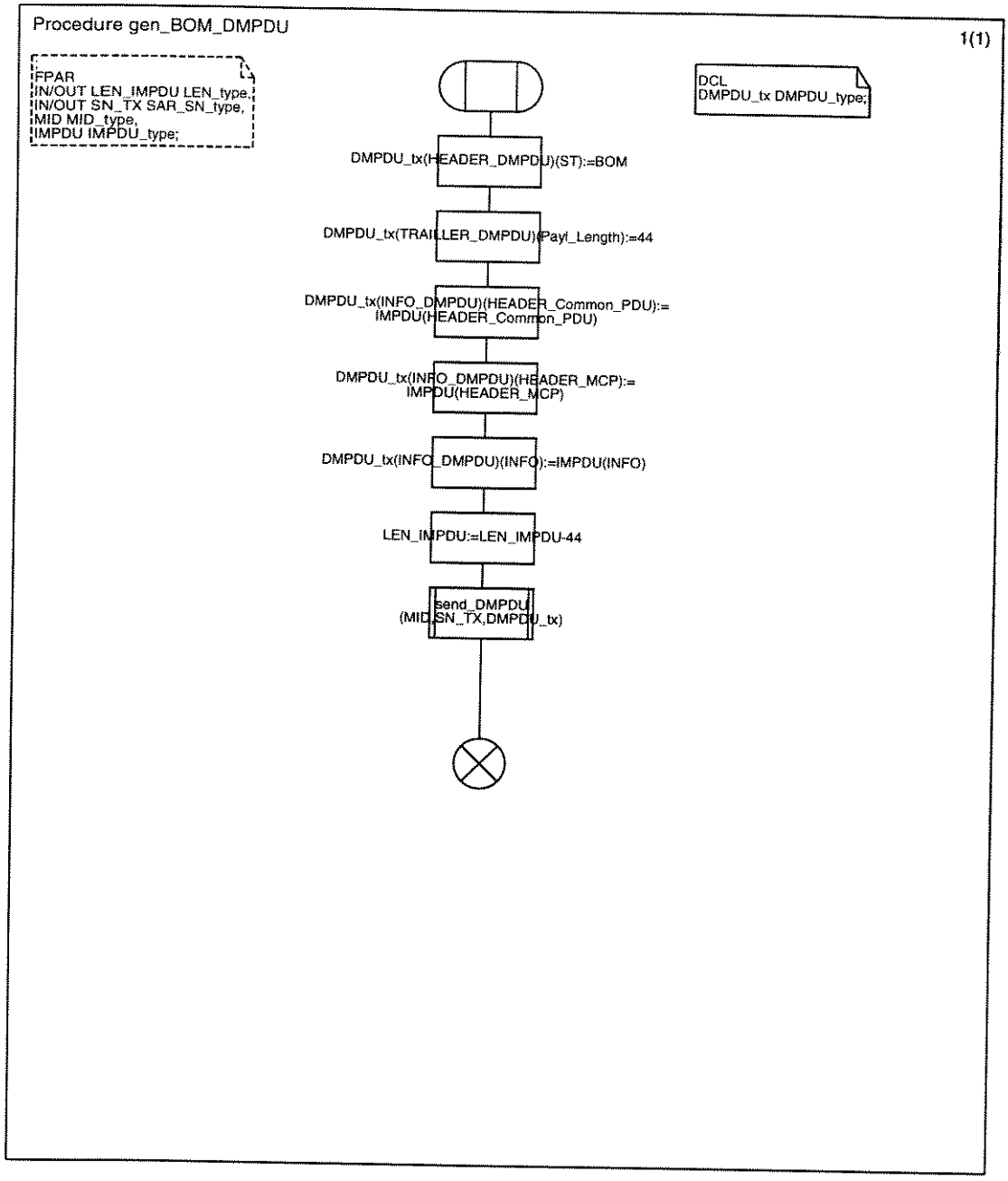

Figura B.11: "Procedimento gen_BOM_DMPDU"

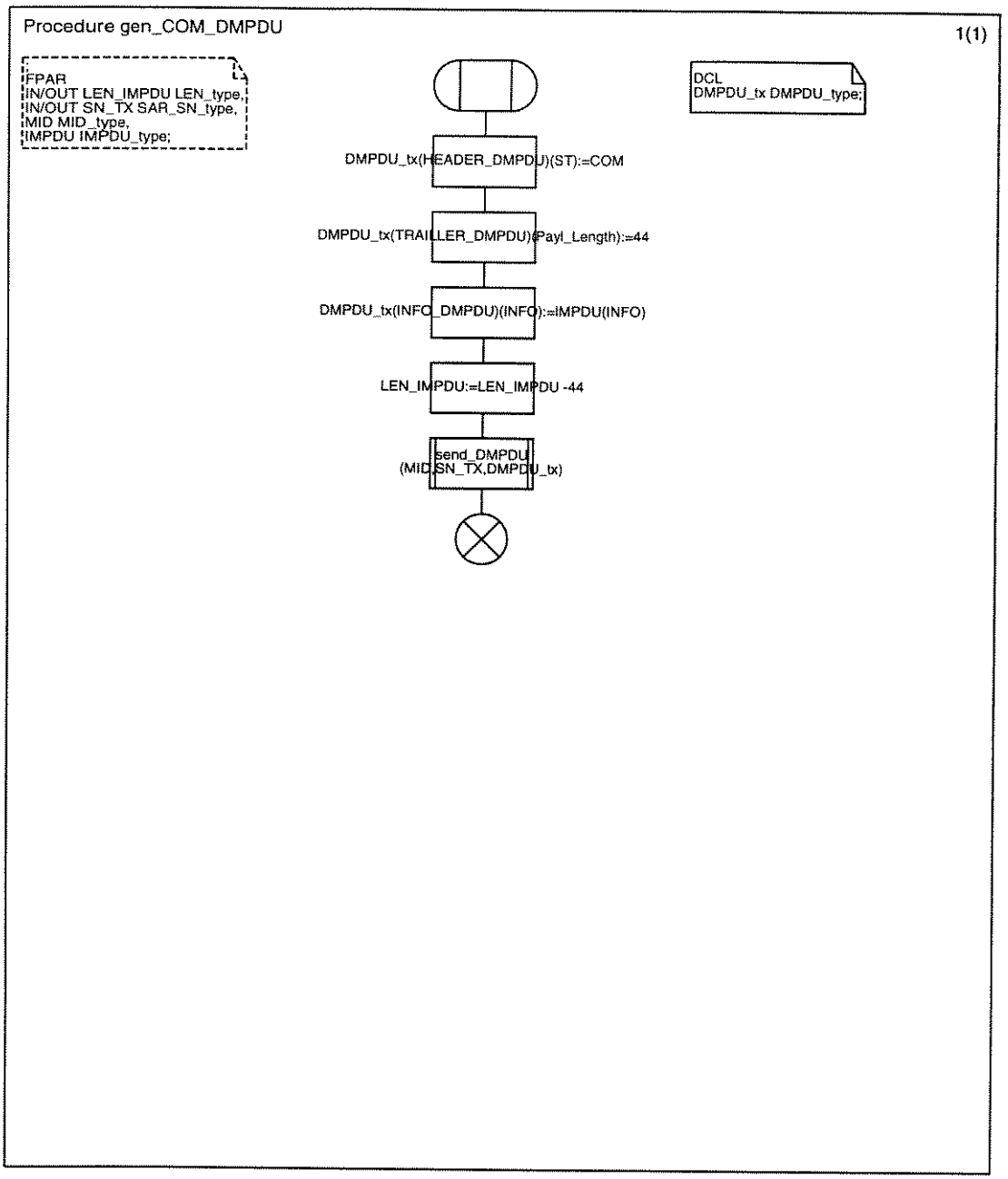

Figura B.12: "Procedimento gen_COM_DMPDU"

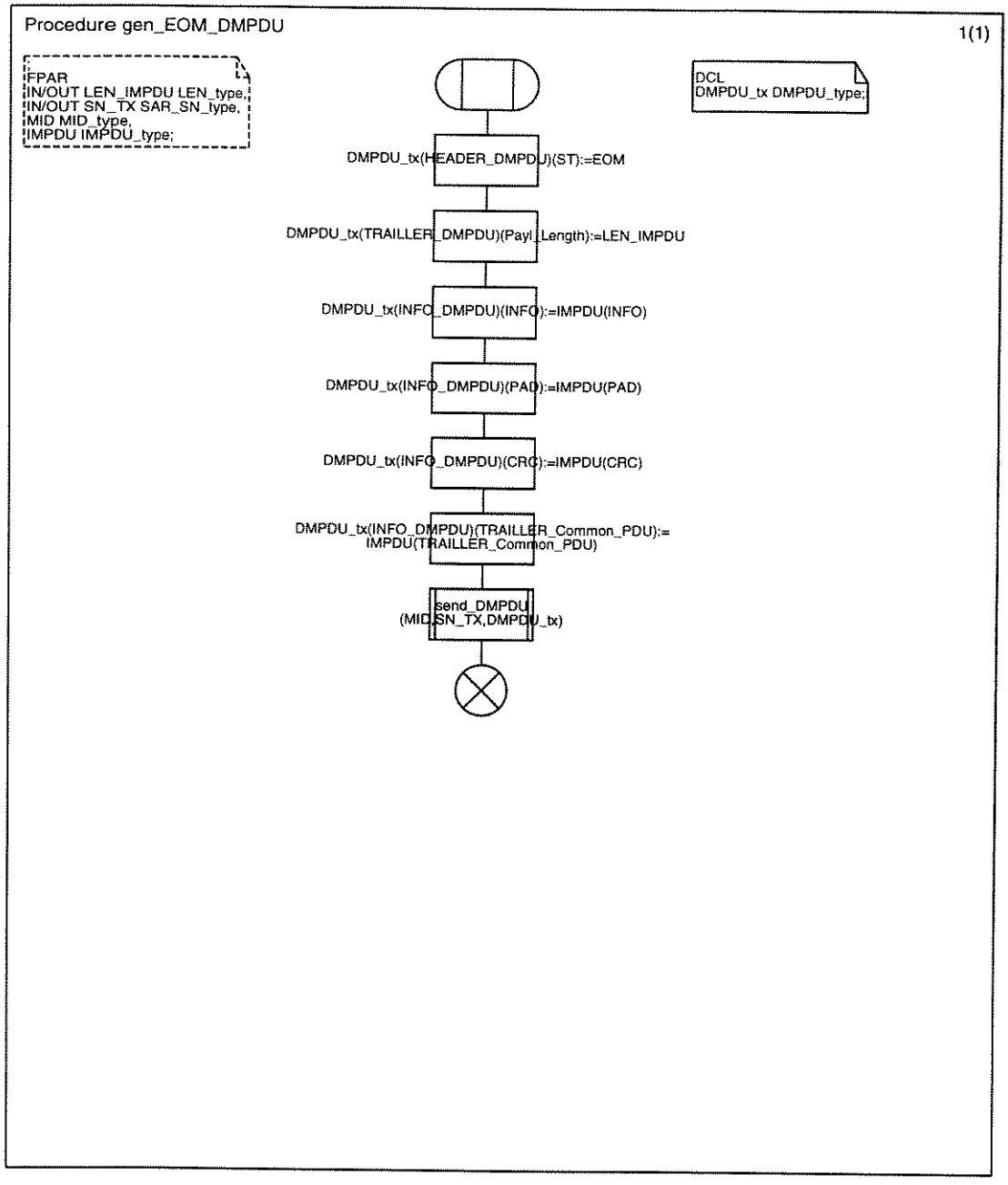

Figura B.13: "Procedimento gen_EOM_DMPDU"

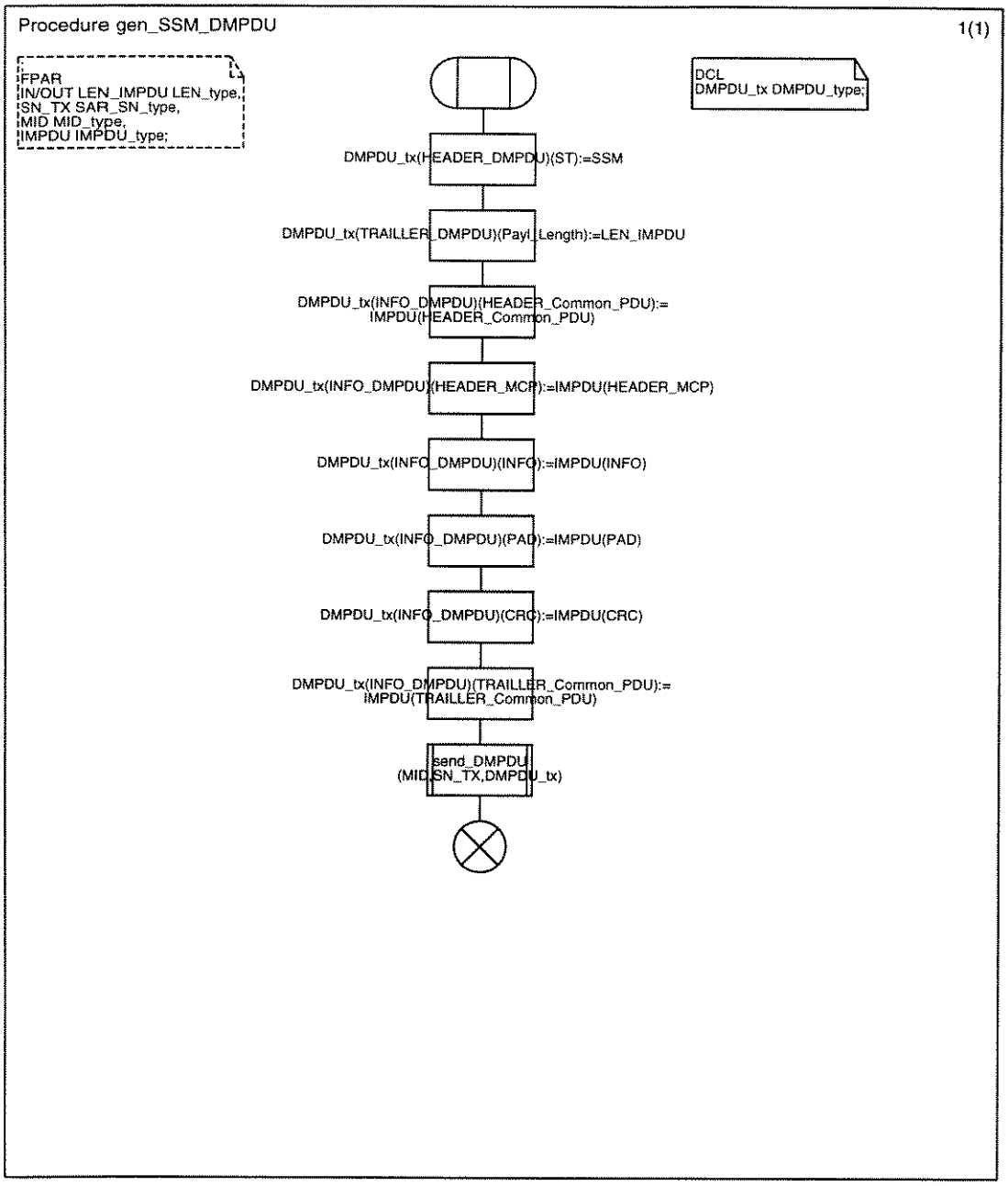

Figura B.14: "Procedimento gen_SSM_DMPDU"

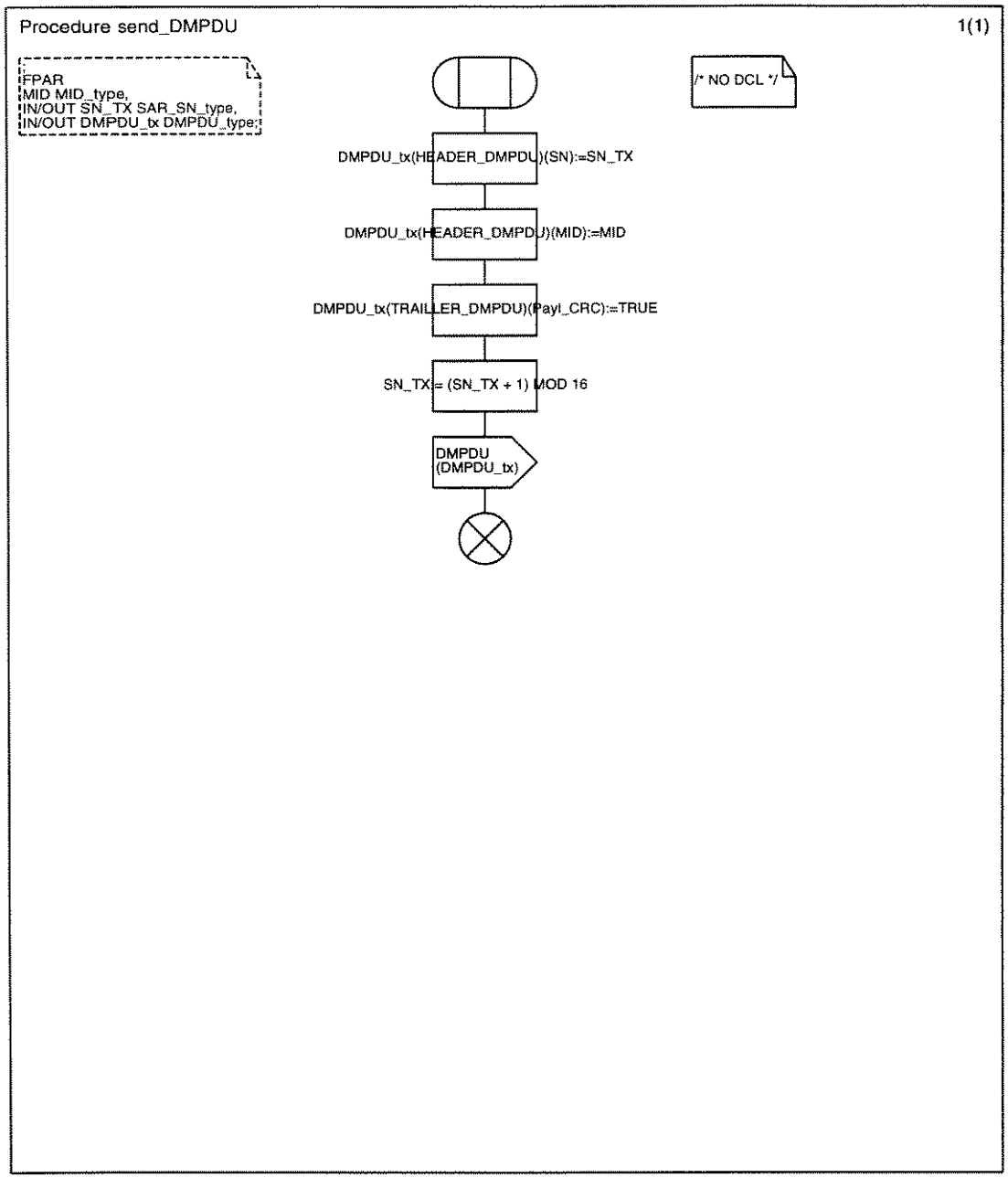

Figura B.15: "Procedimento send_DMPDU"

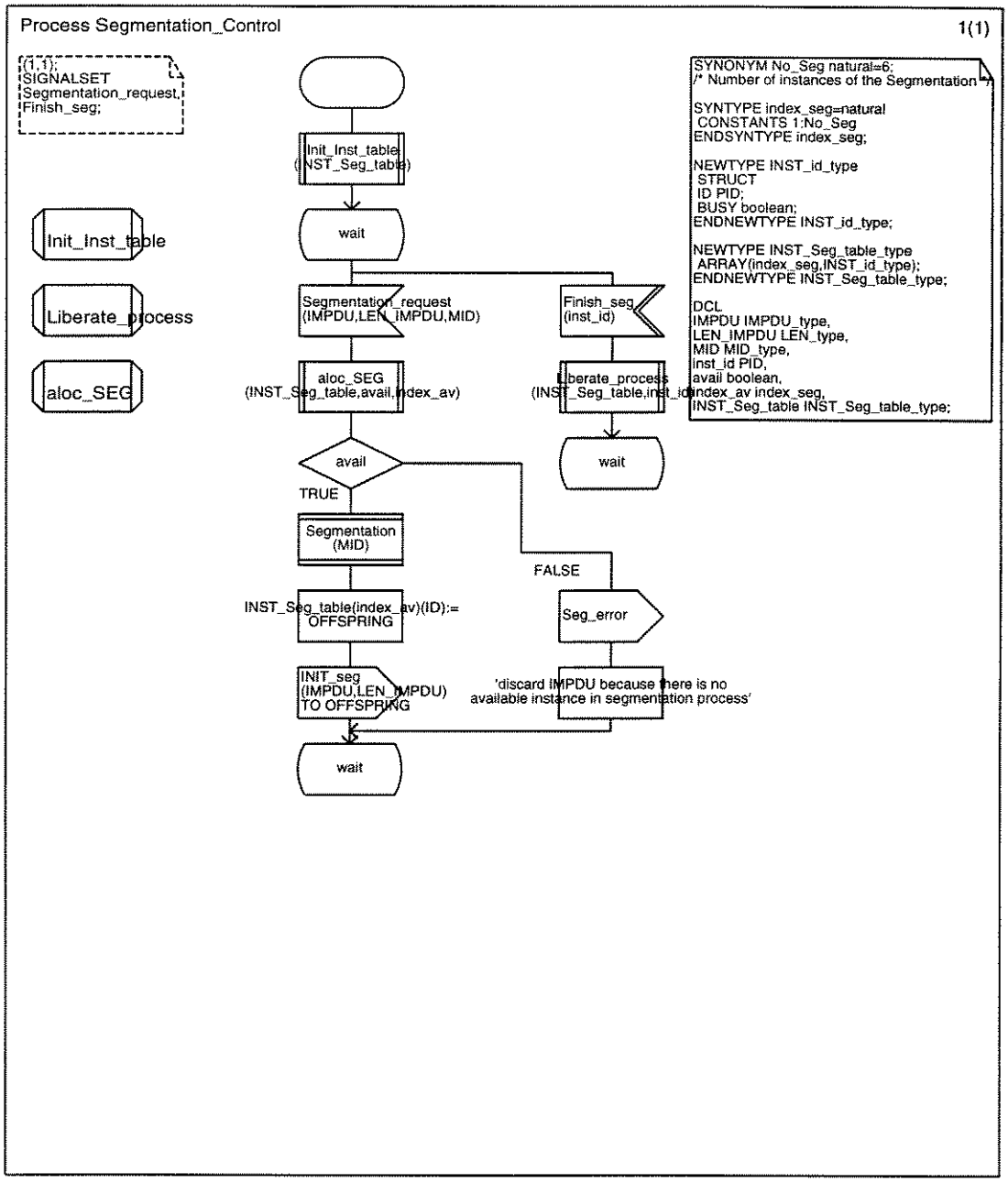

Figura B.16: "Processo Segmentation_Control"

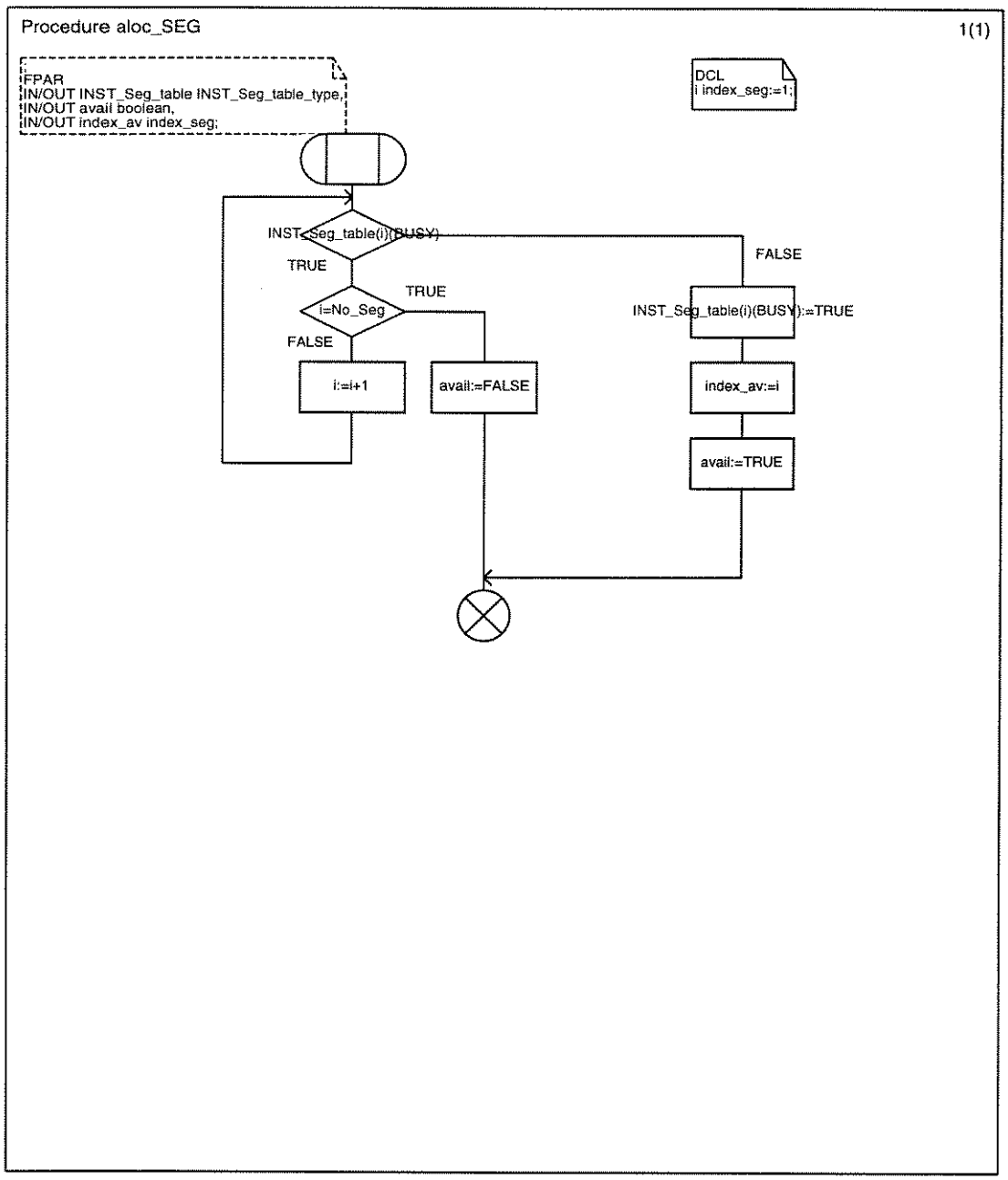

Figura B.17: "Procedimento aloc_SEG"

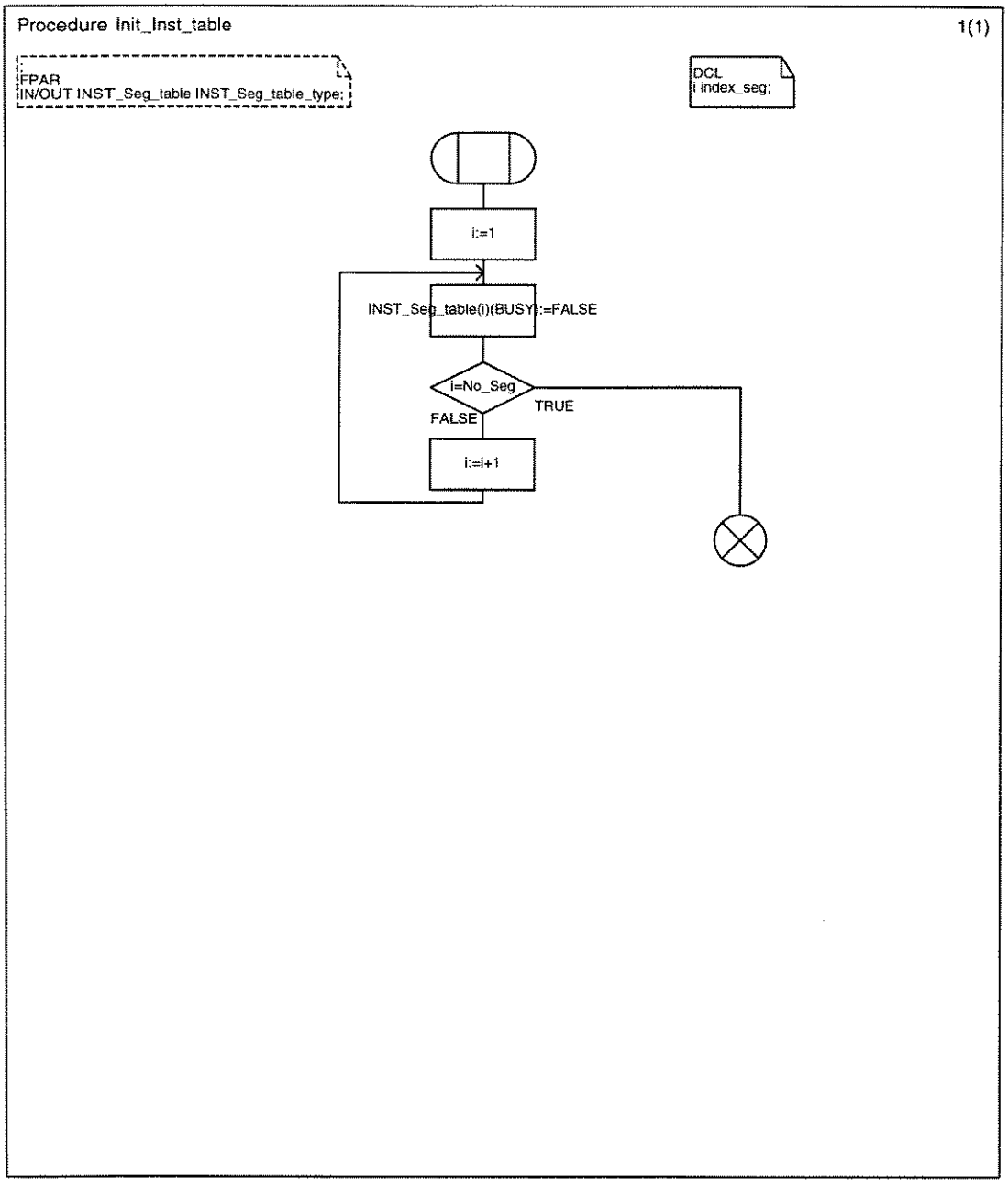

Figura B.18: "Procedimento Init_Inst_table"

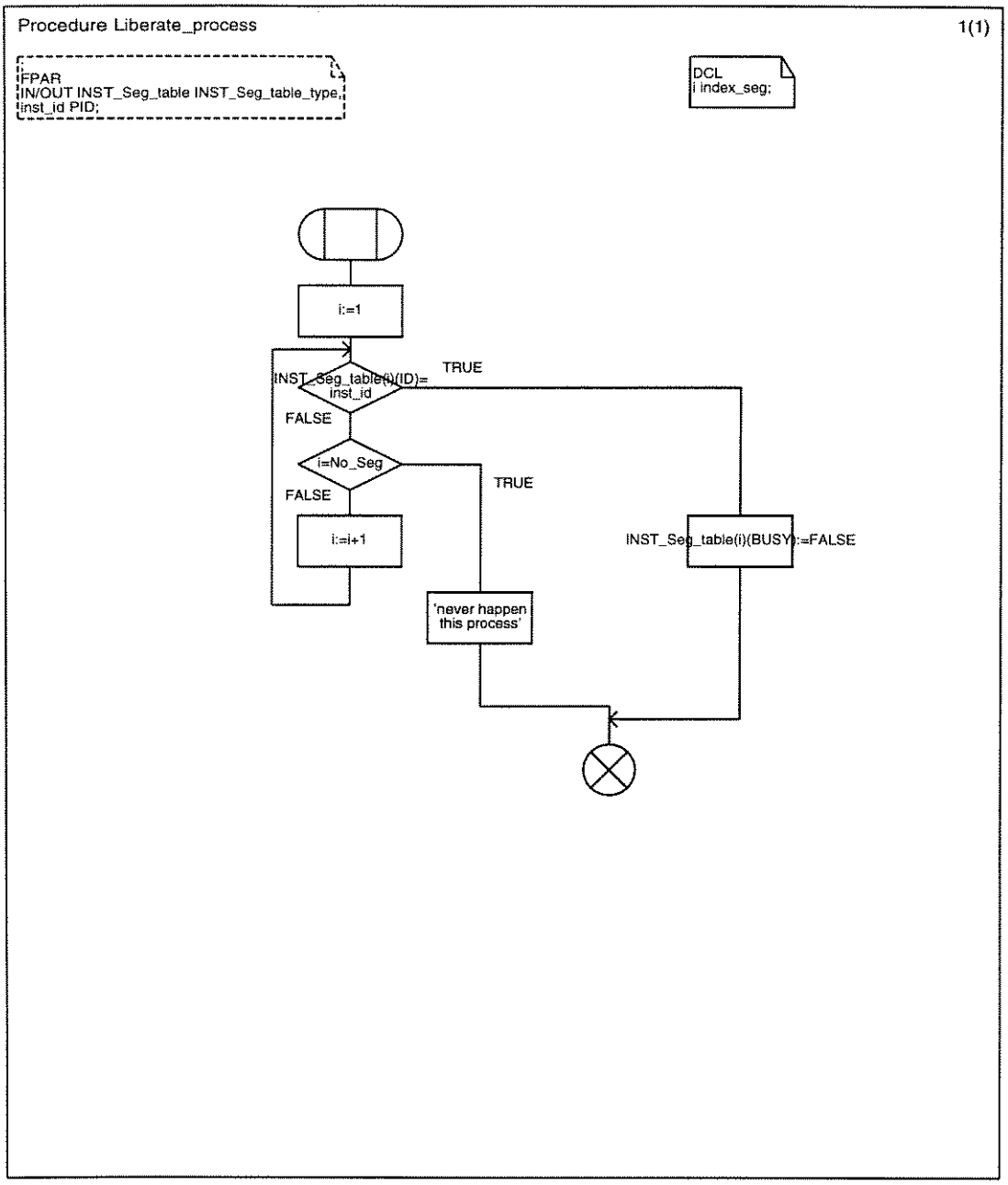

Figura B.19: "Procedimento Liberate_process"

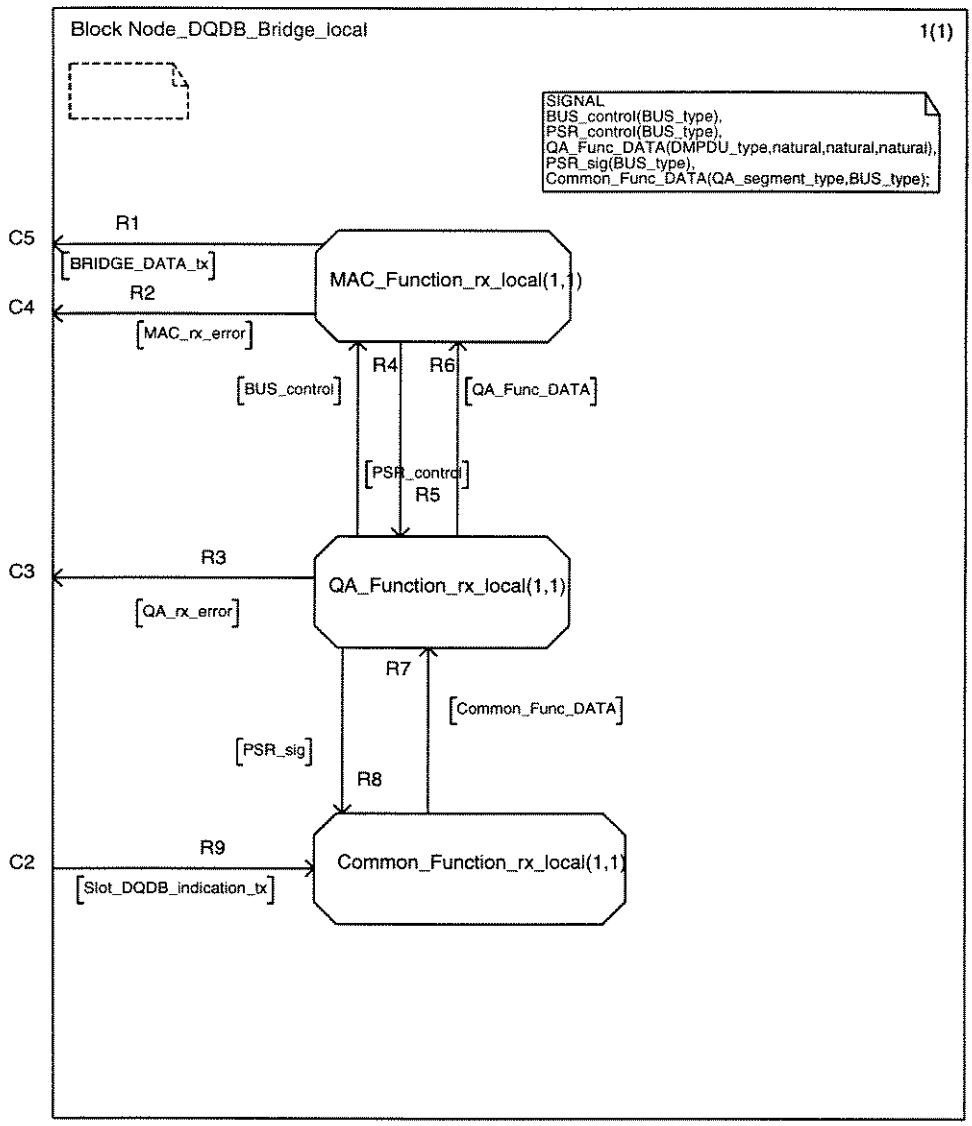

Figura B.20: "Bloco Node_DQDB_Bridge_local"

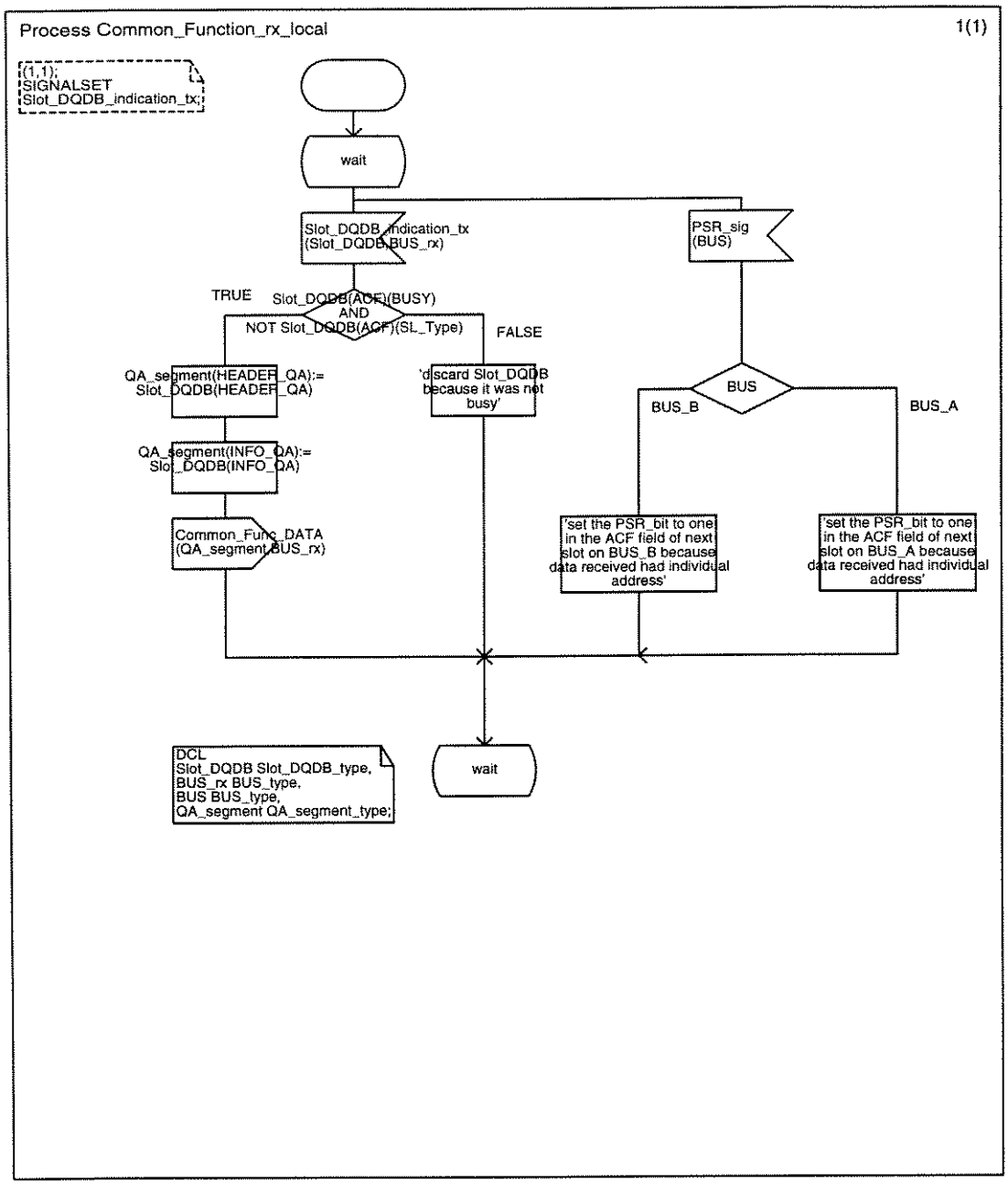

Figura B.21: "Processo Common_Function_rx_local"

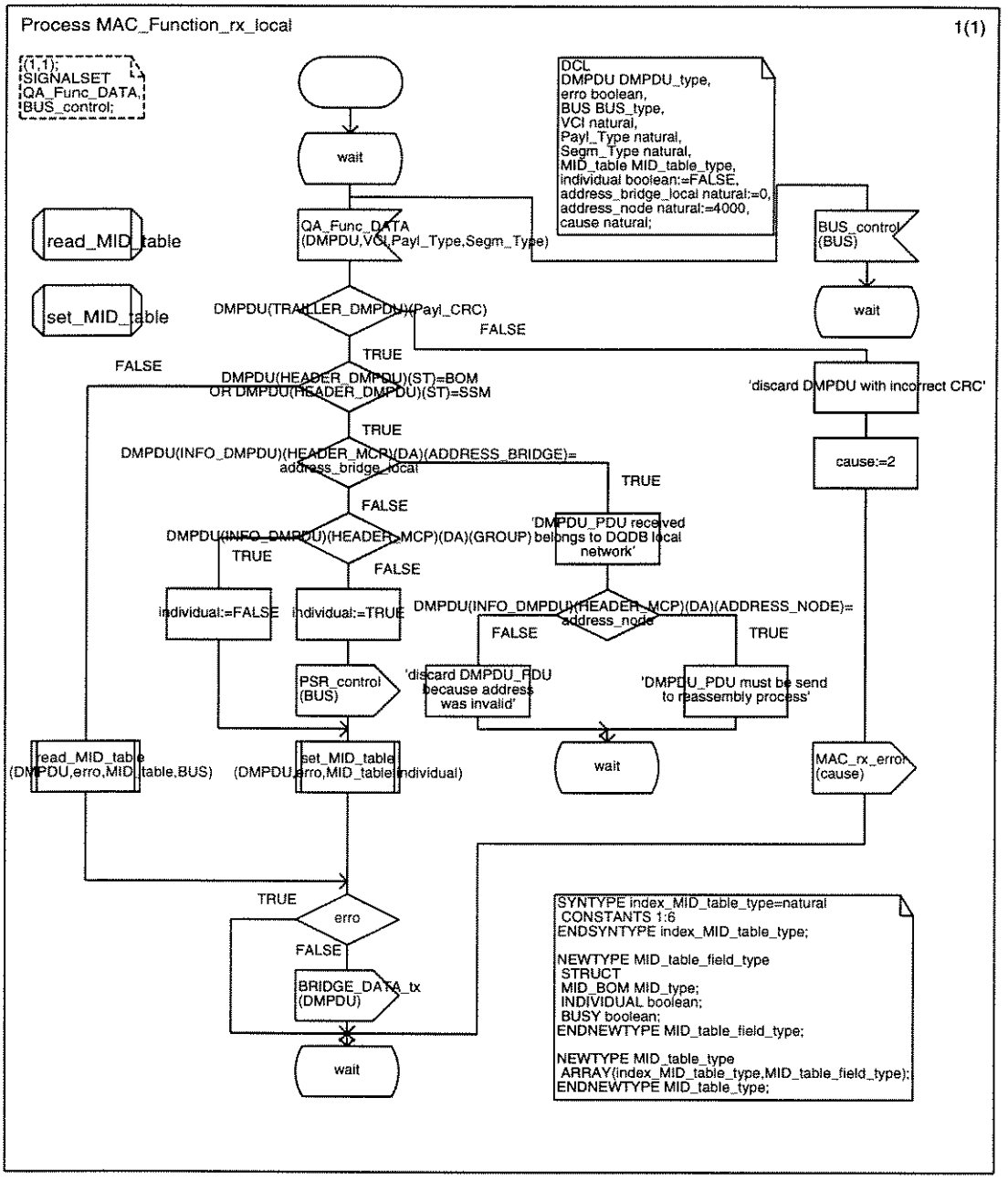

Figura B.22: "Processo MAC_Function_rx_local"

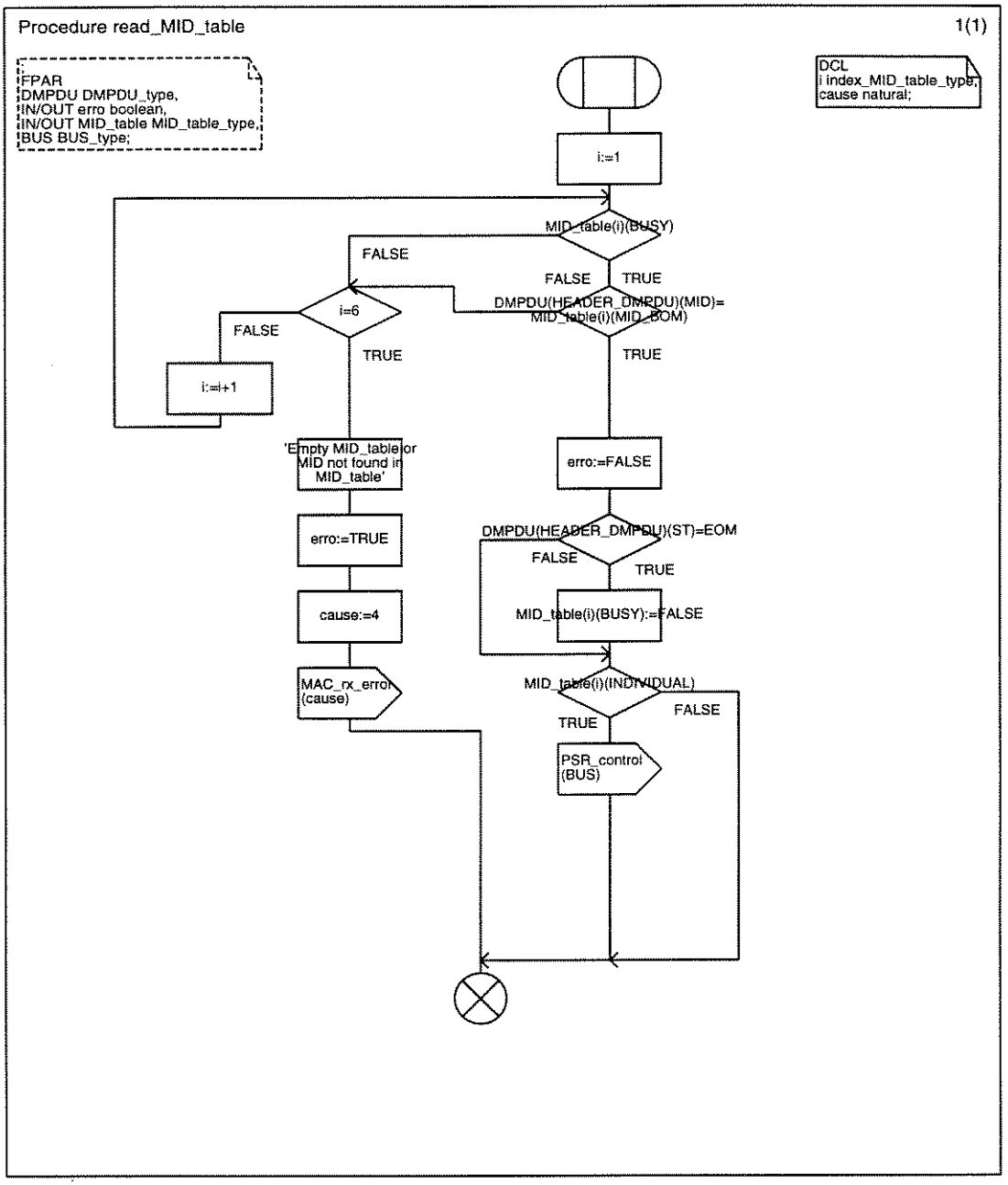

Figura B.23: "Procedimento read_MID_table"

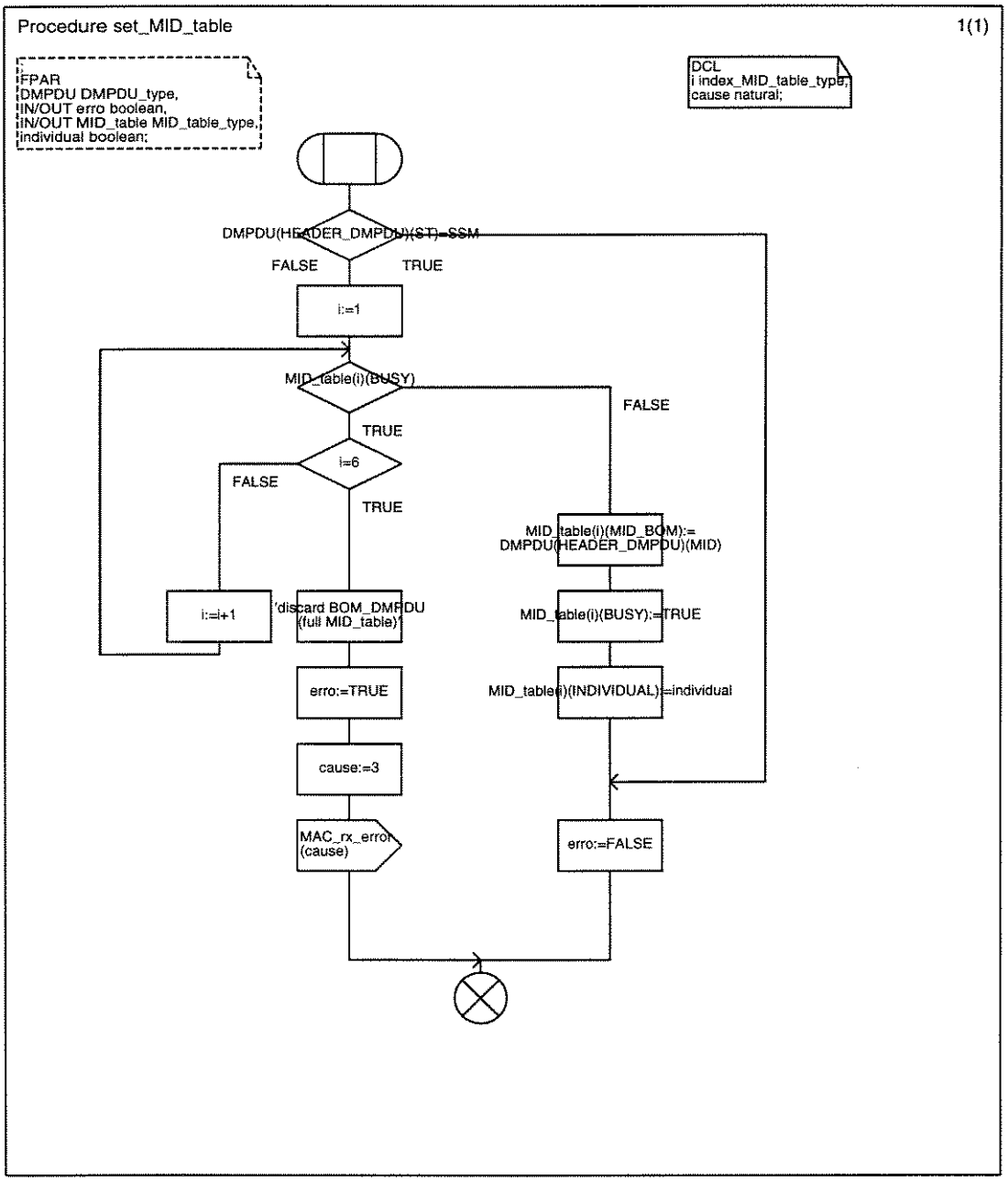

Figura B.24: "Procedimento set_MID_table"

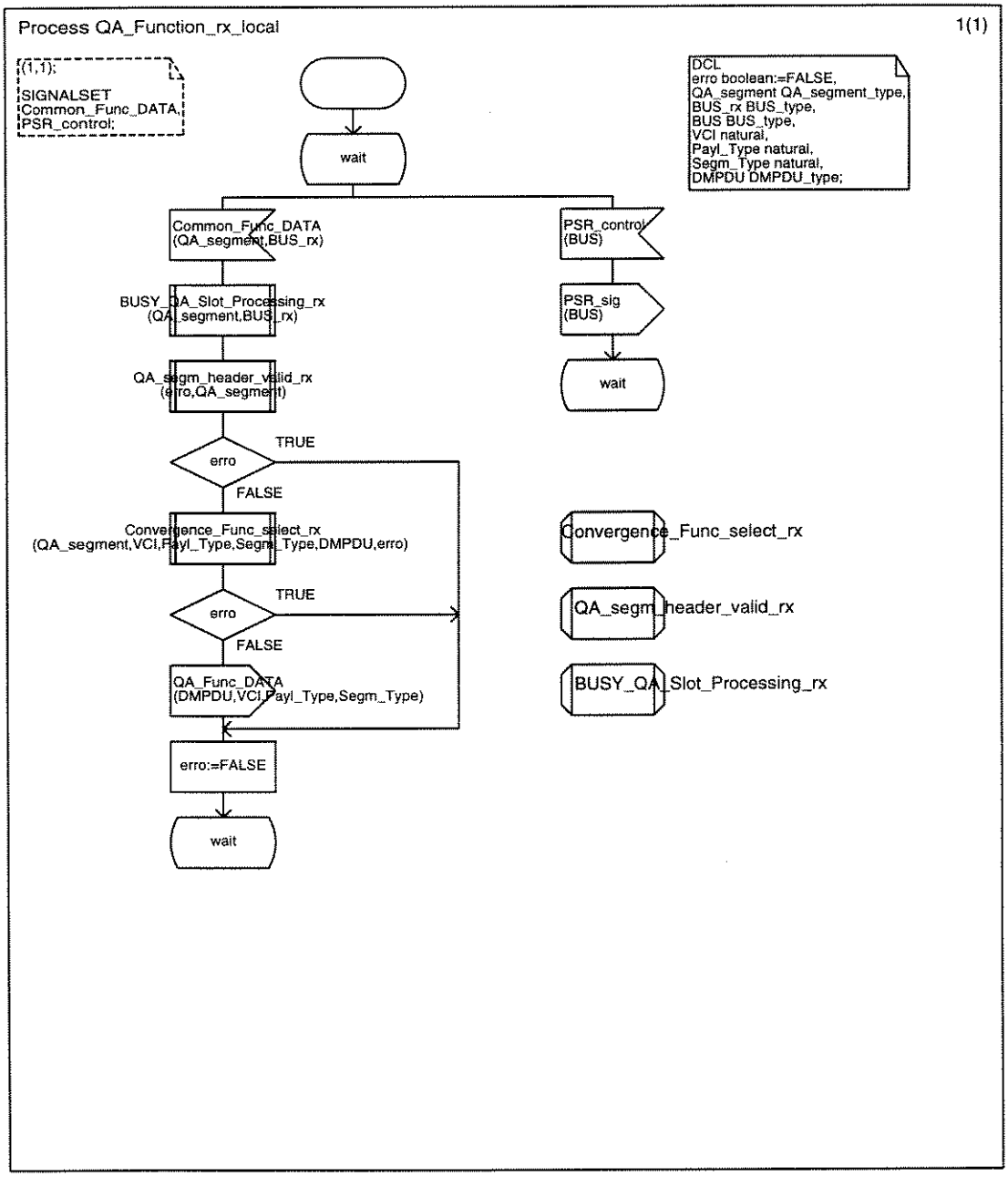

Figura B.25: "Processo QA_Function_rx_local"

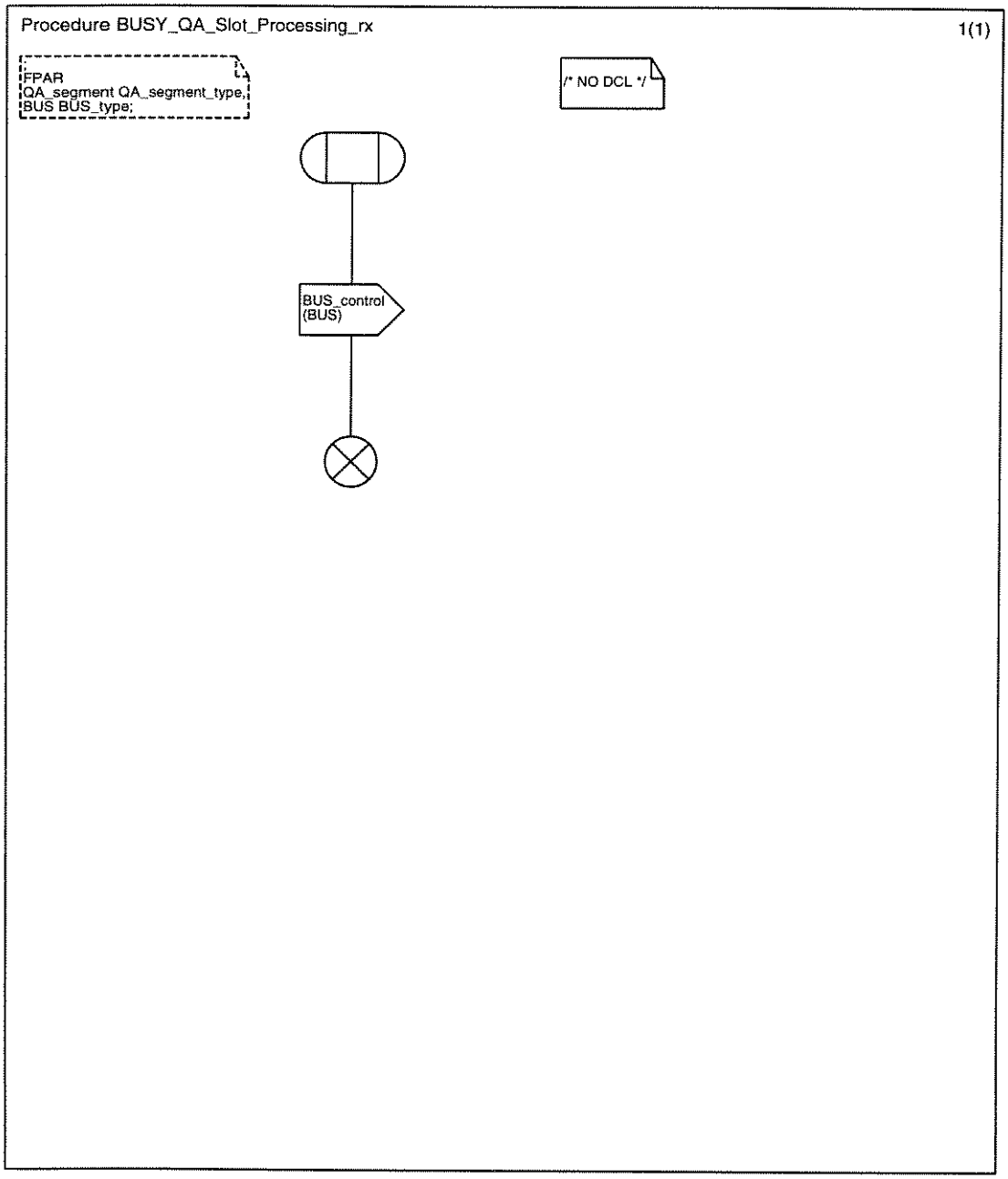

Figura B.26: "Procedimento BUSY_QA_Slot_Processing"

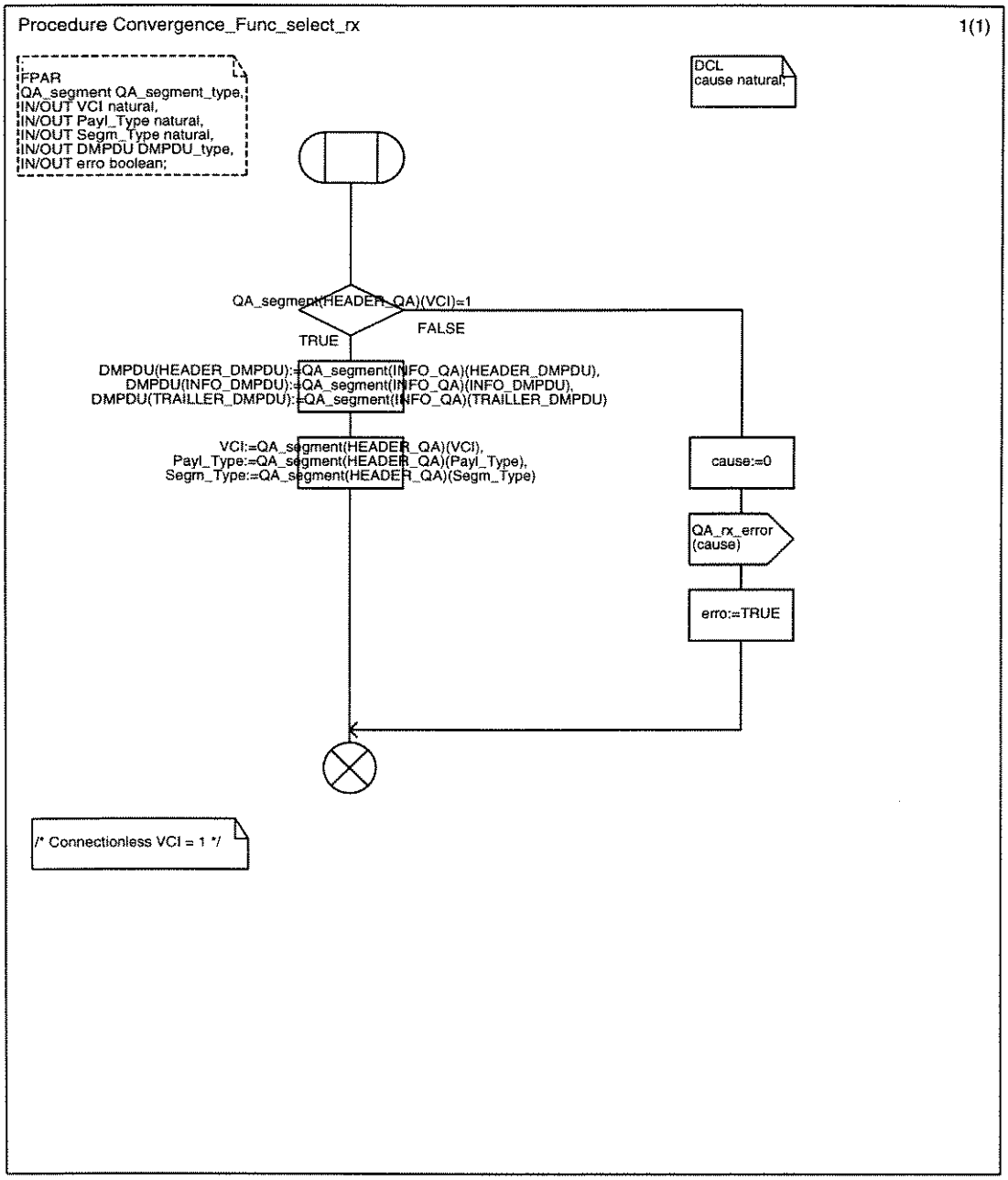

Figura B.27: "Procedimento Convergence_Func_select_rx"

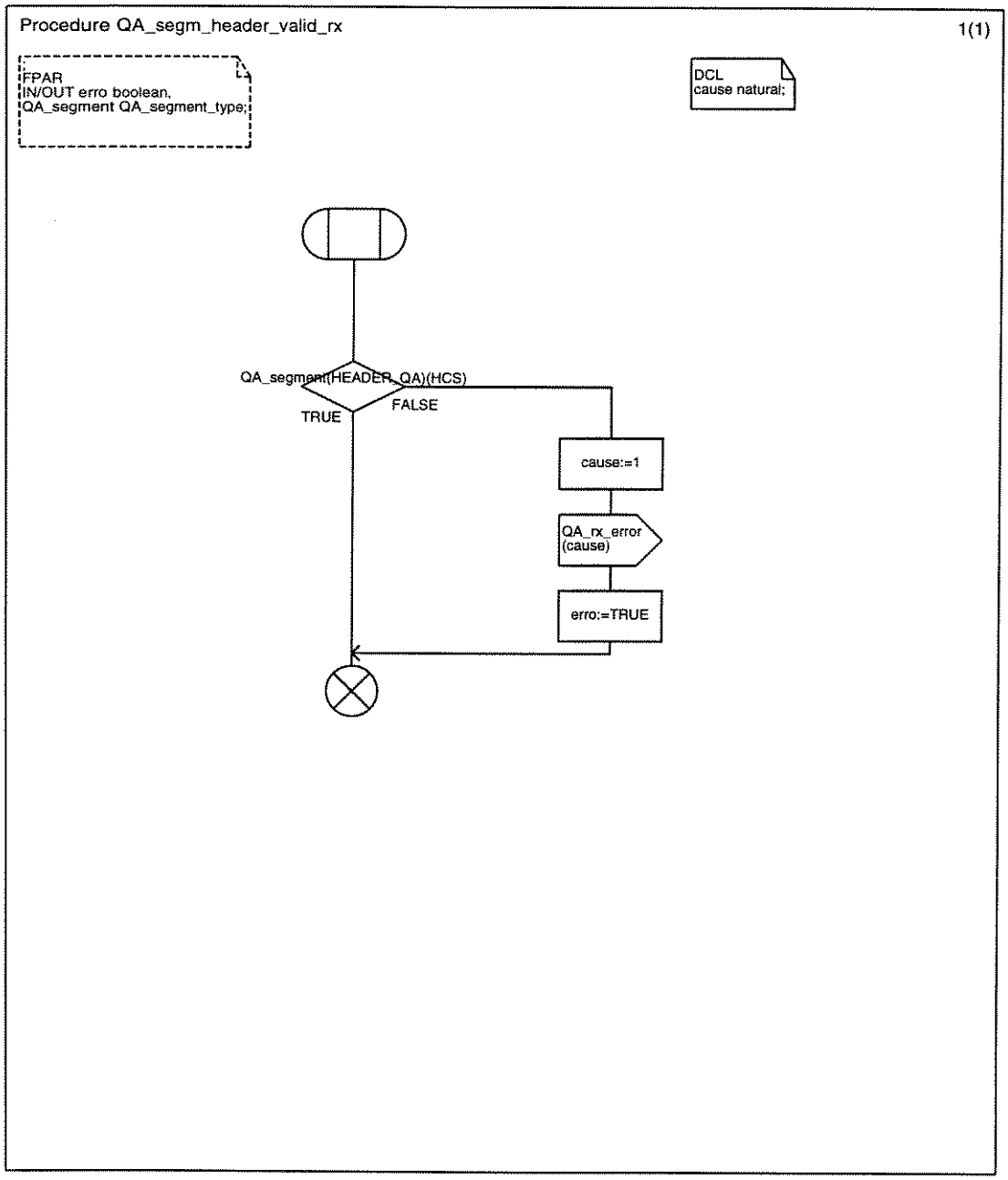

Figura B.28: "Procedimento QA_segm_header_valid_rx"

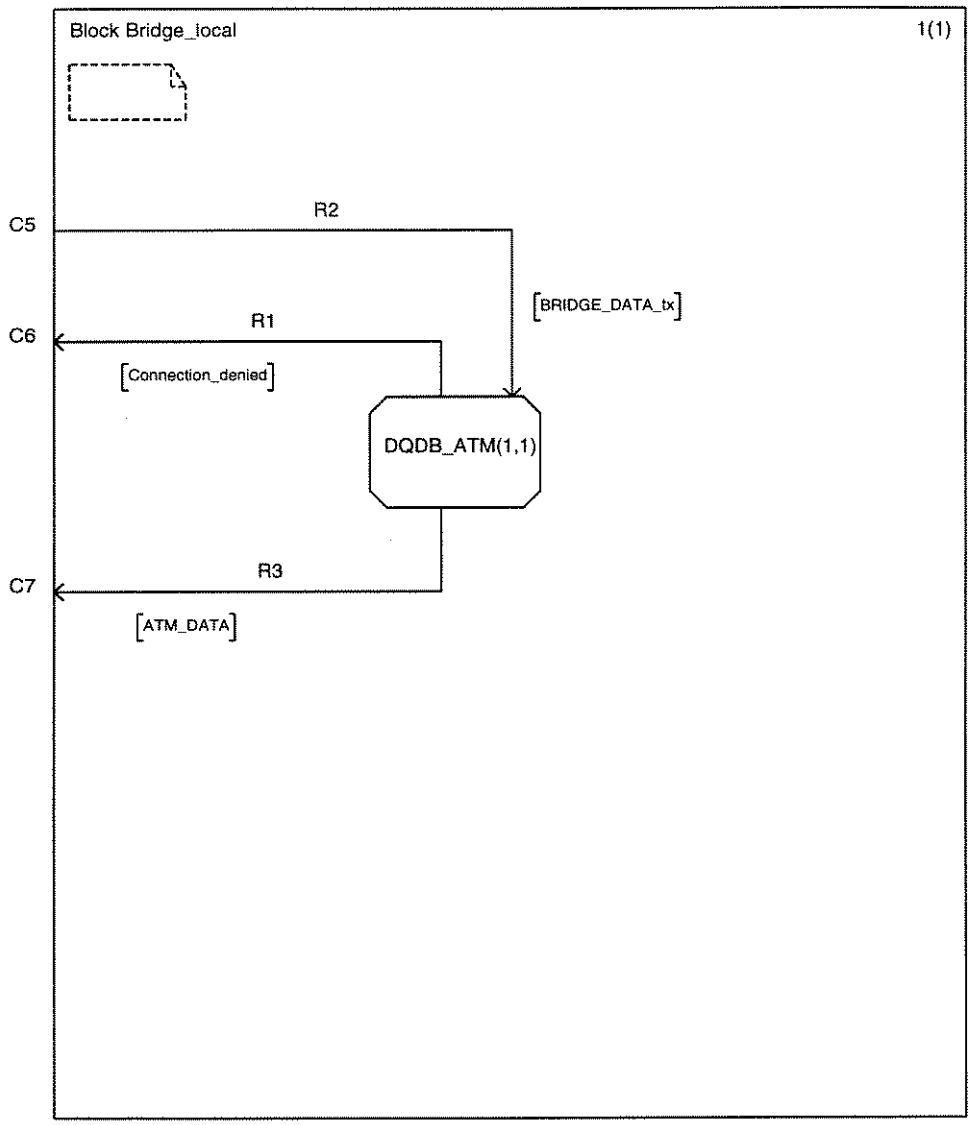

Figura B.29: "Bloco Bridge_local"

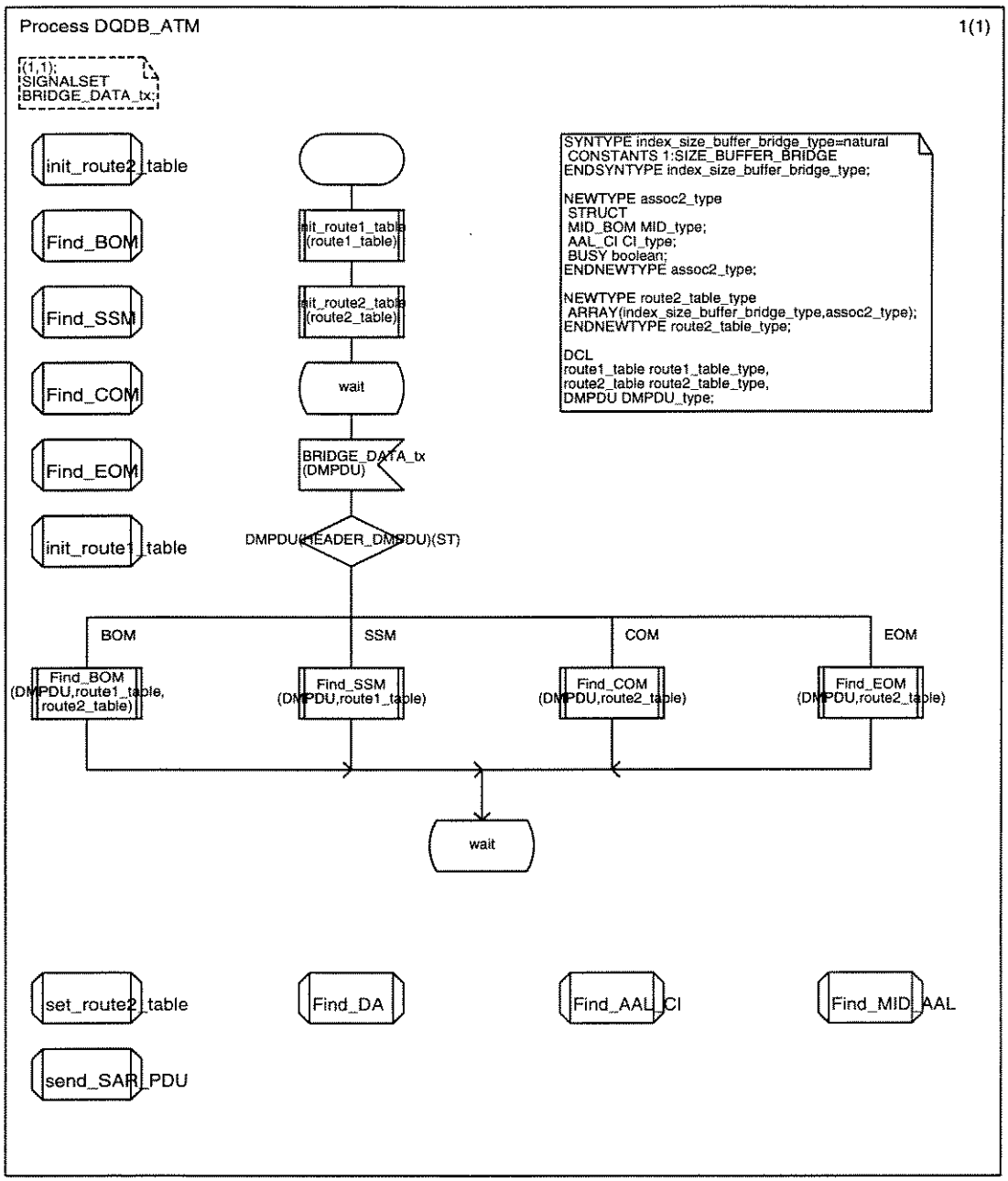

Figura B.30: "Processo DQDB_ATM"

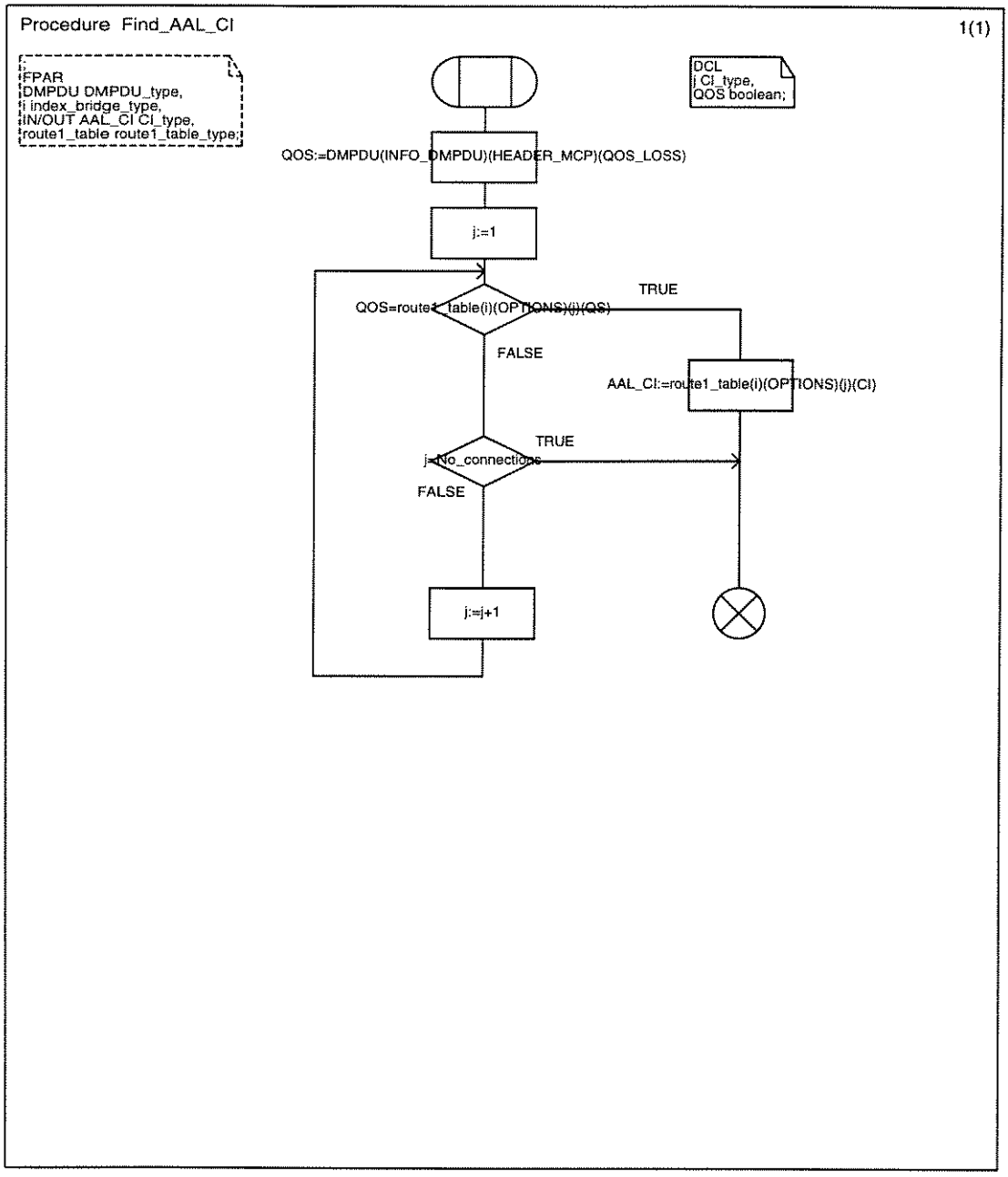

Figura B.31: "Procedimento Find_AAL_CI"

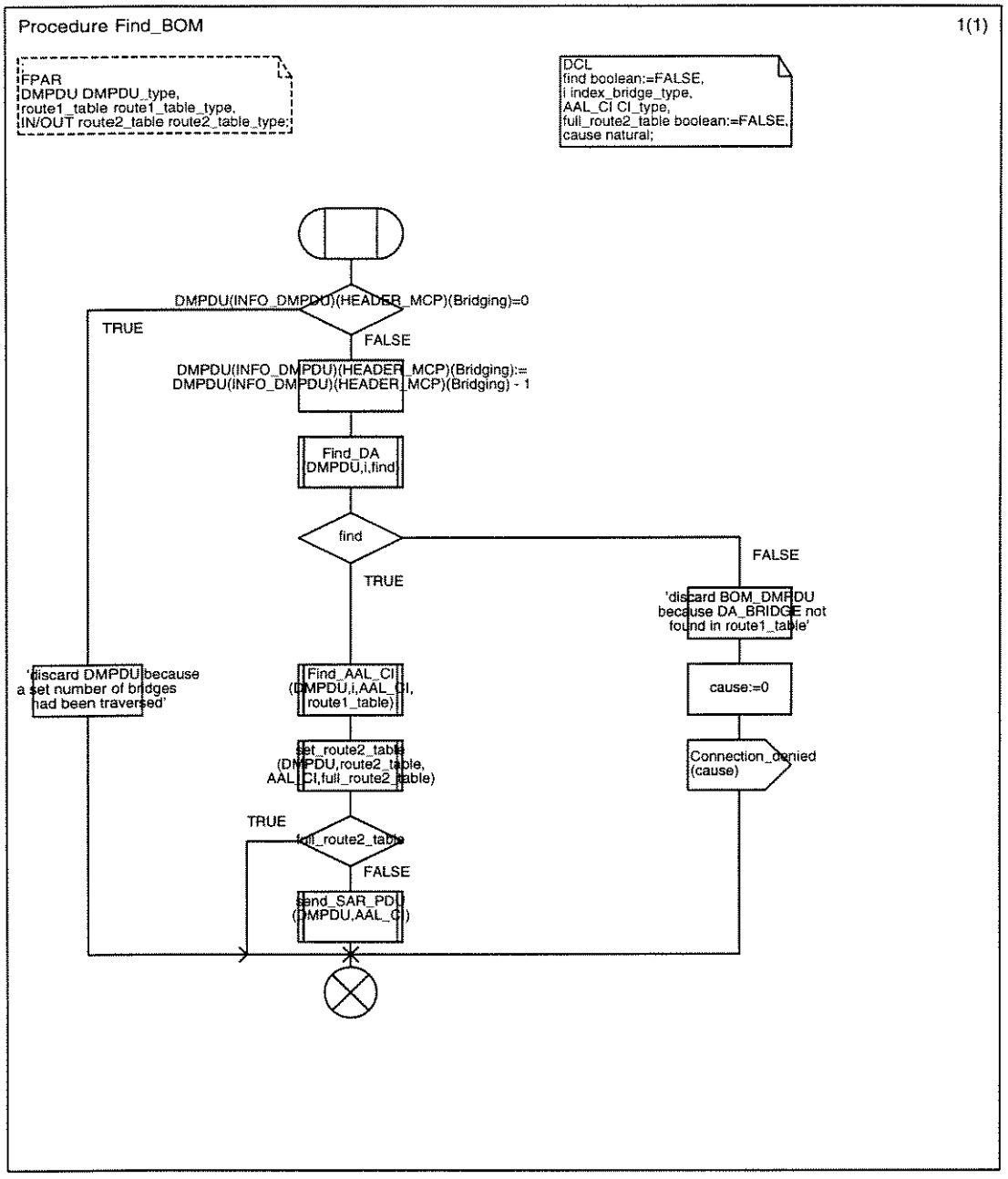

Figura B.32: "Procedimento Find_BOM"

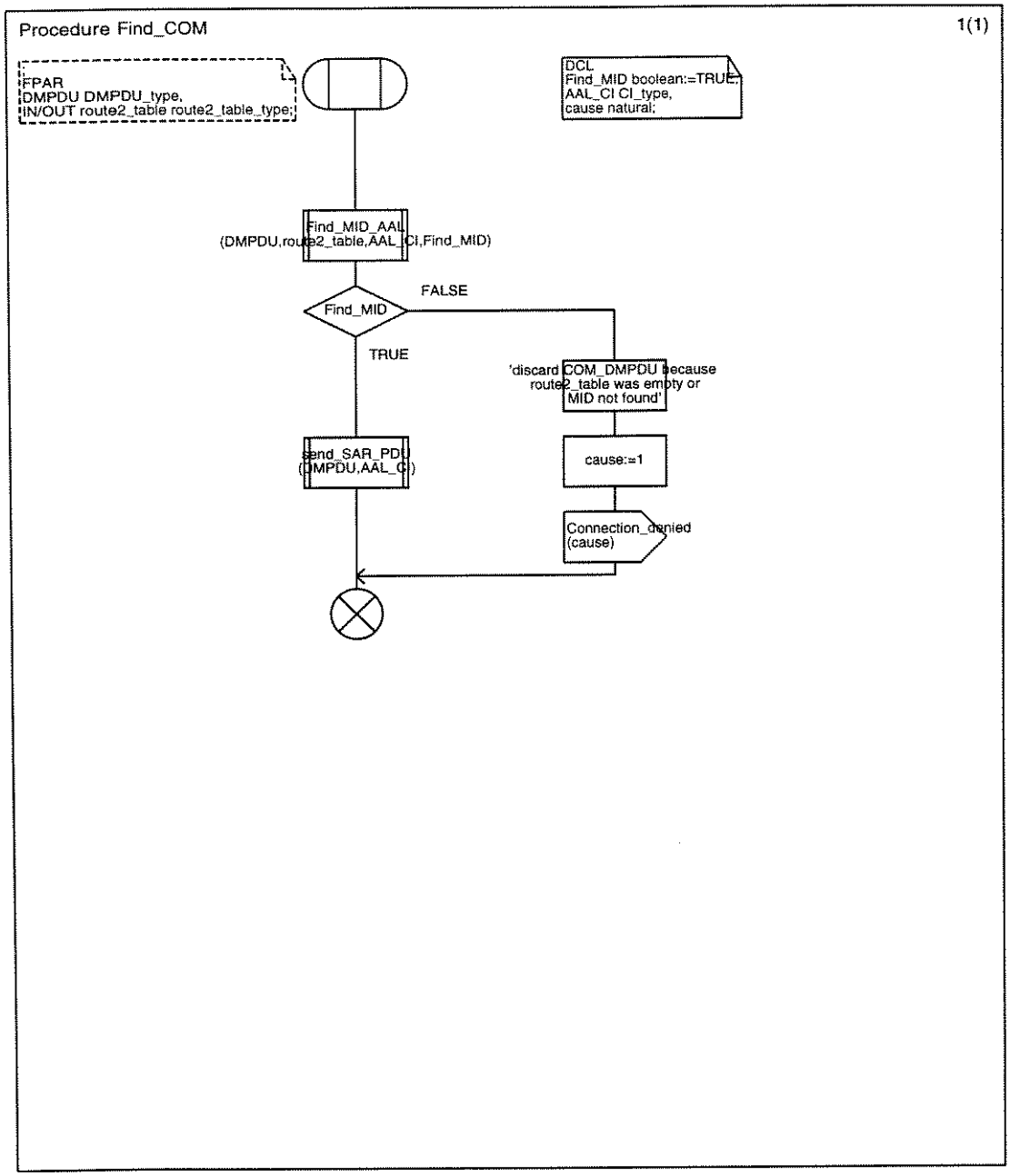

Figura B.33: "Procedimento Find_COM"

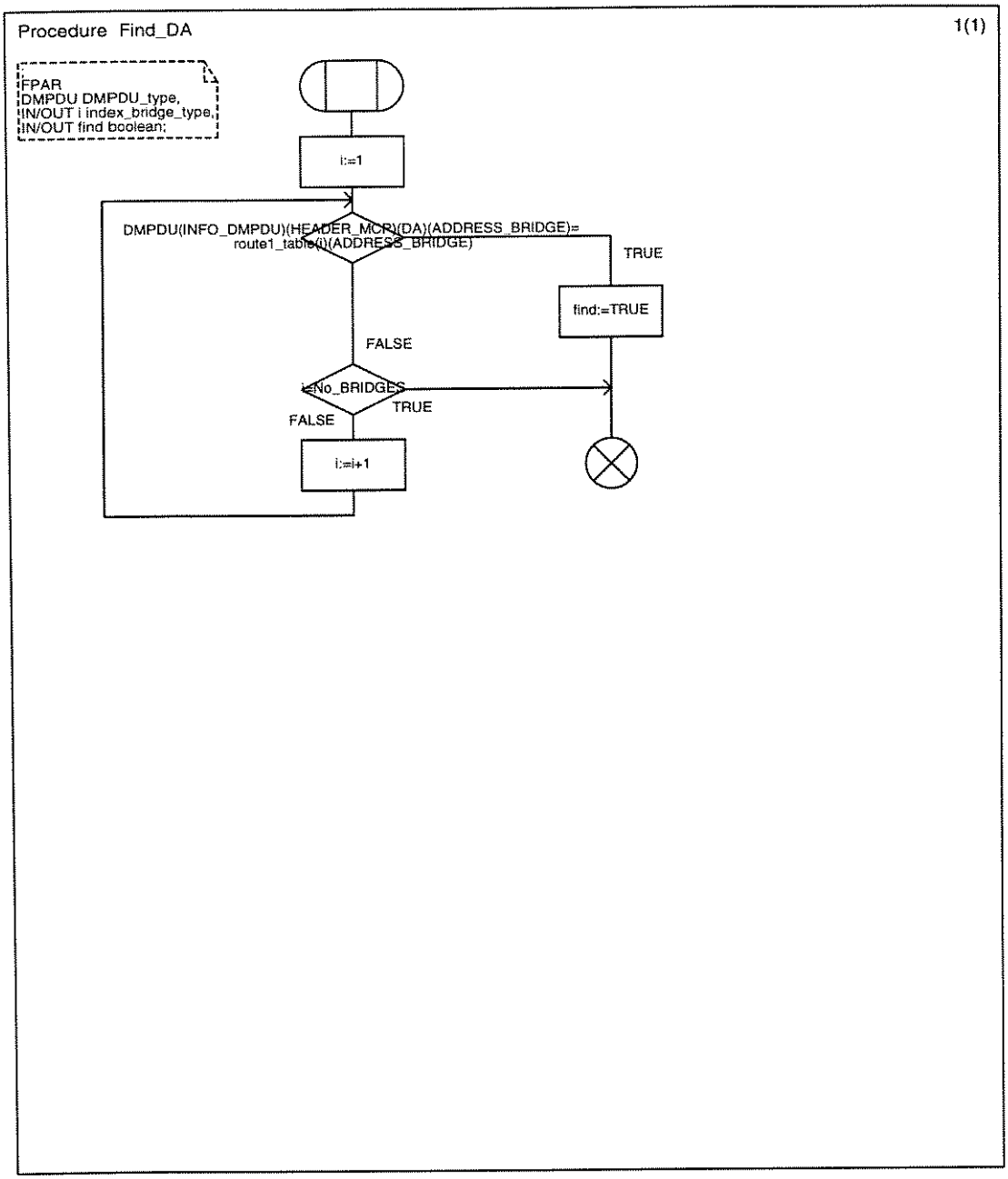

Figura B.34: "Procedimento Find_DA"

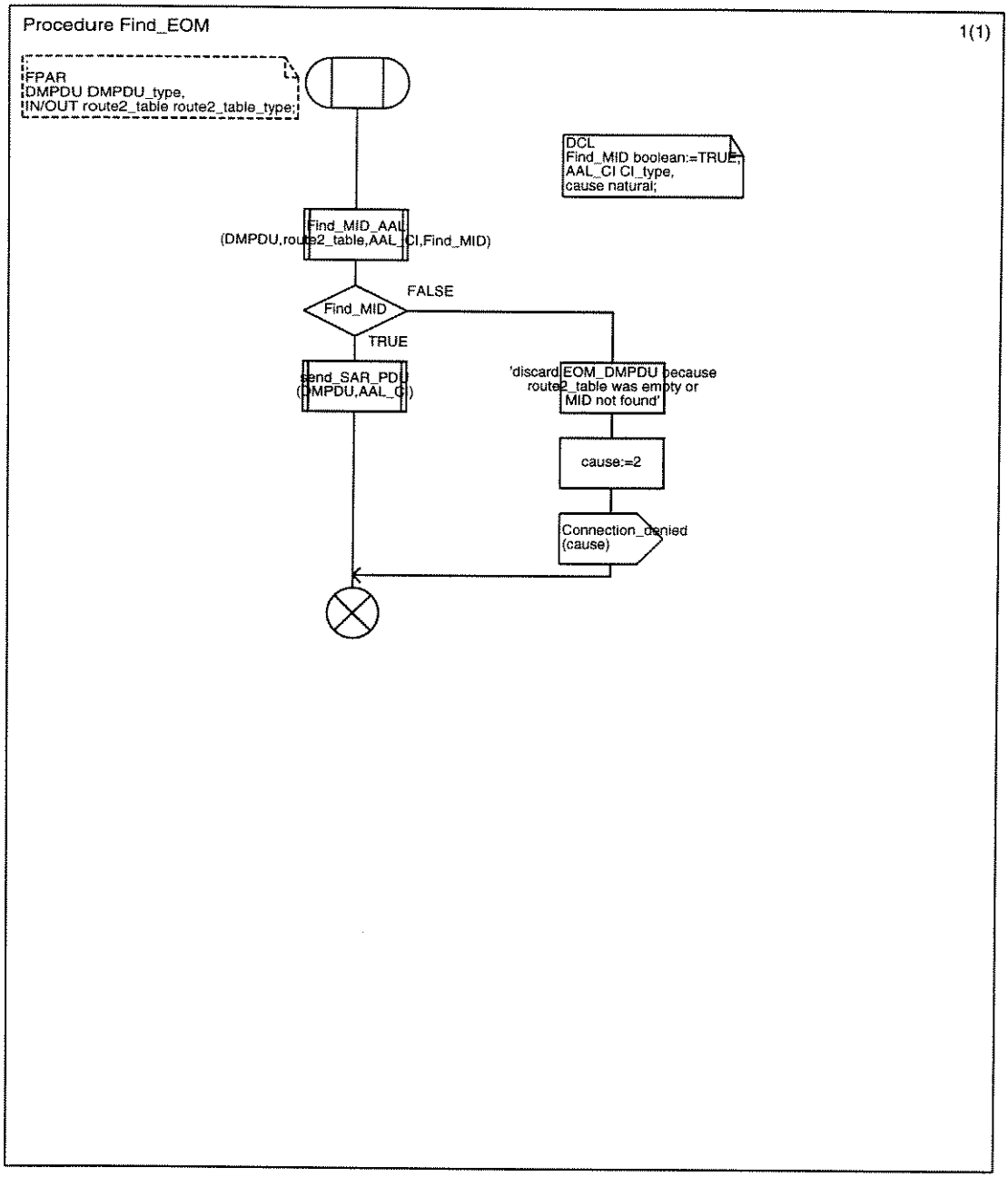

Figura B.35: "Procedimento Find_EOM"

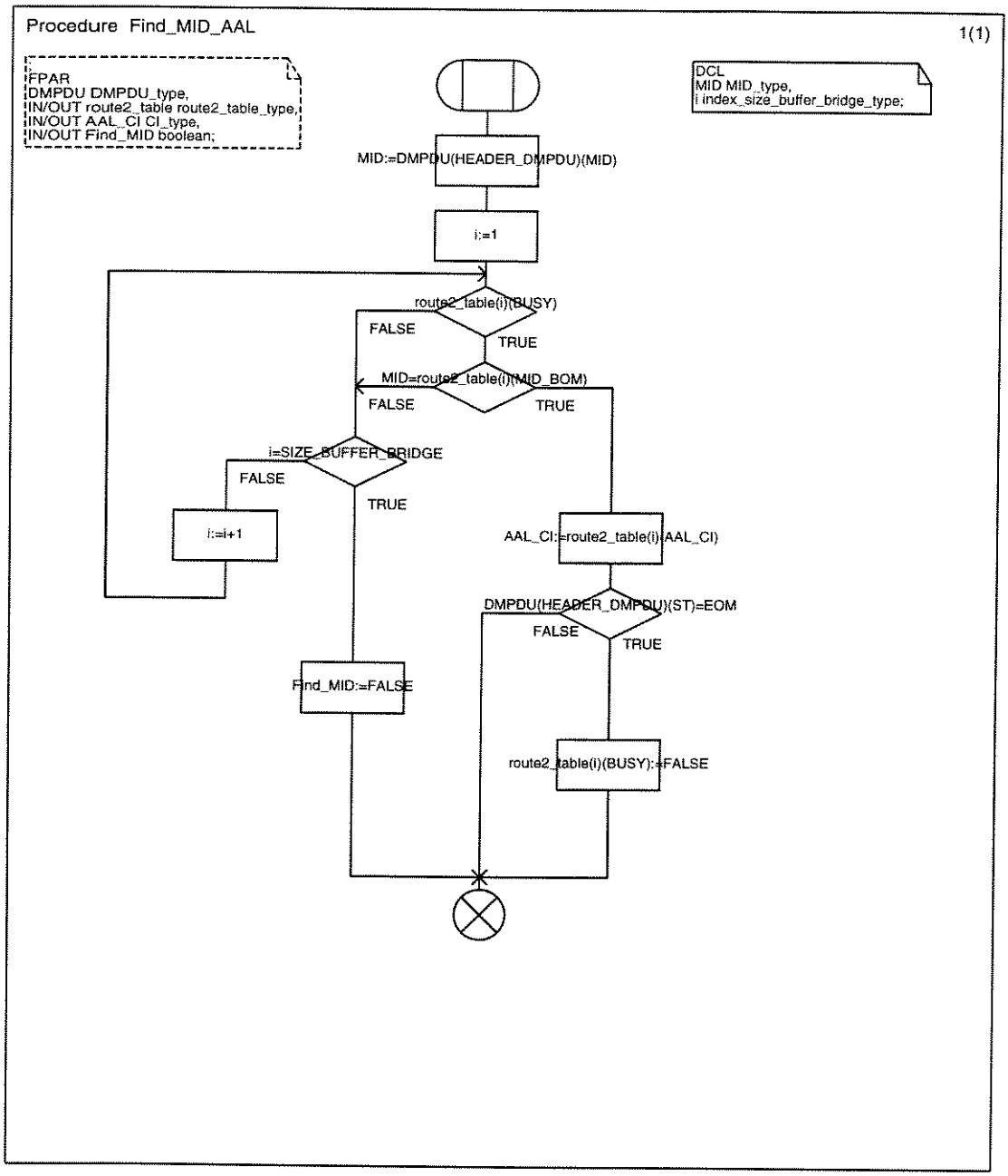

Figura B.36: "Procedimento Find_MID_AAL"

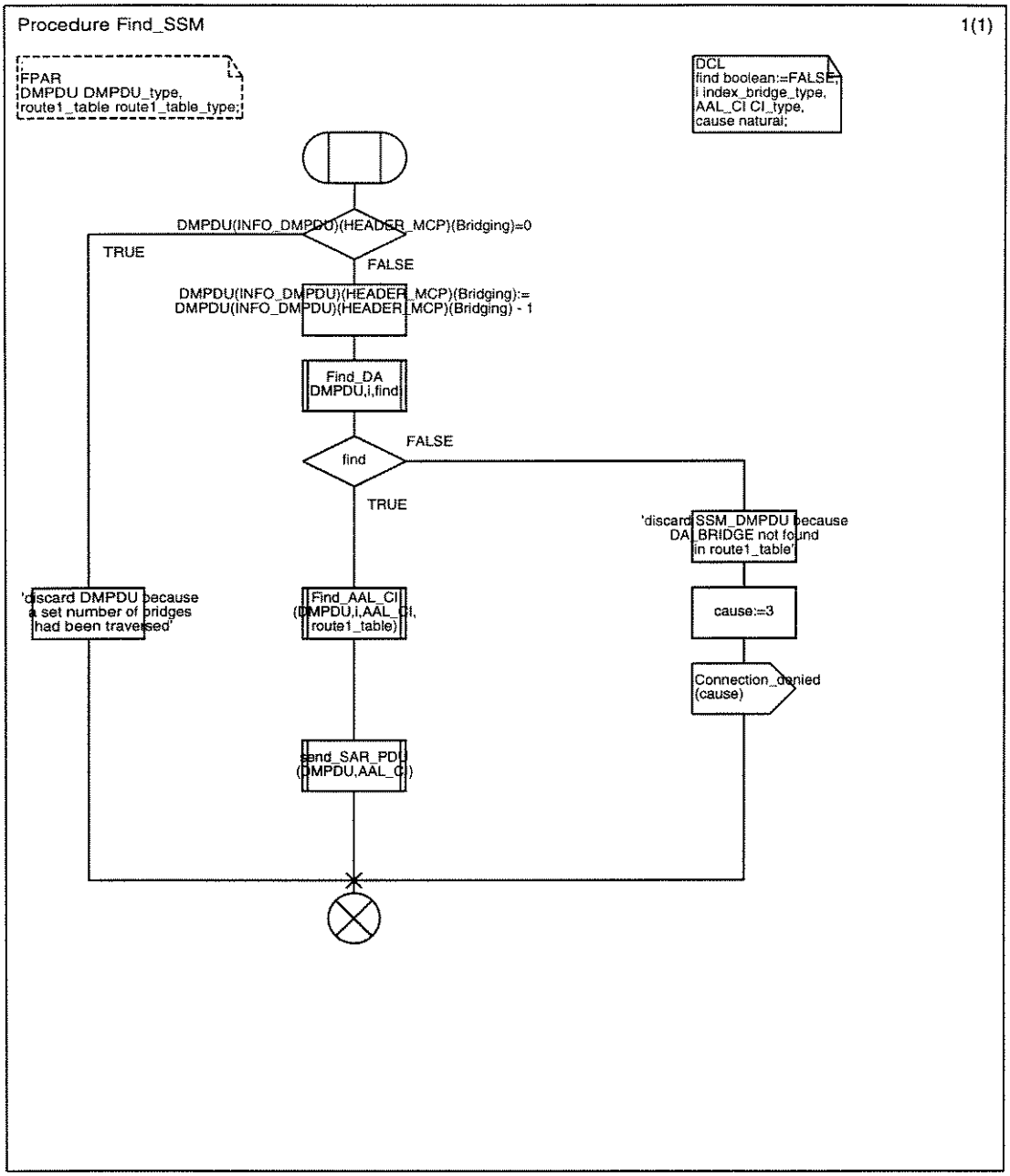

Figura B.37: "Procedimento Find_SSM"

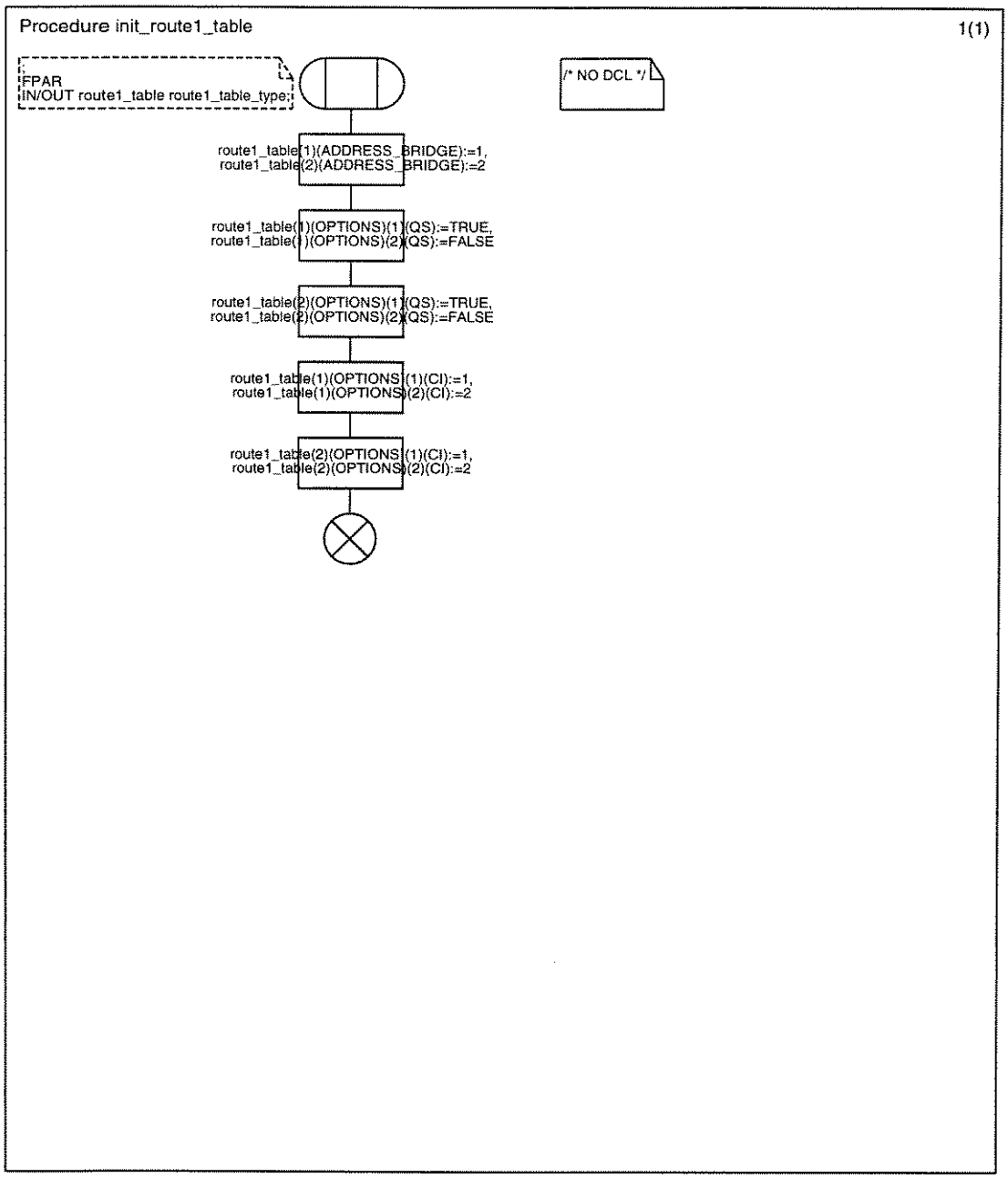

Figura B.38: "Procedimento init_route1_table"

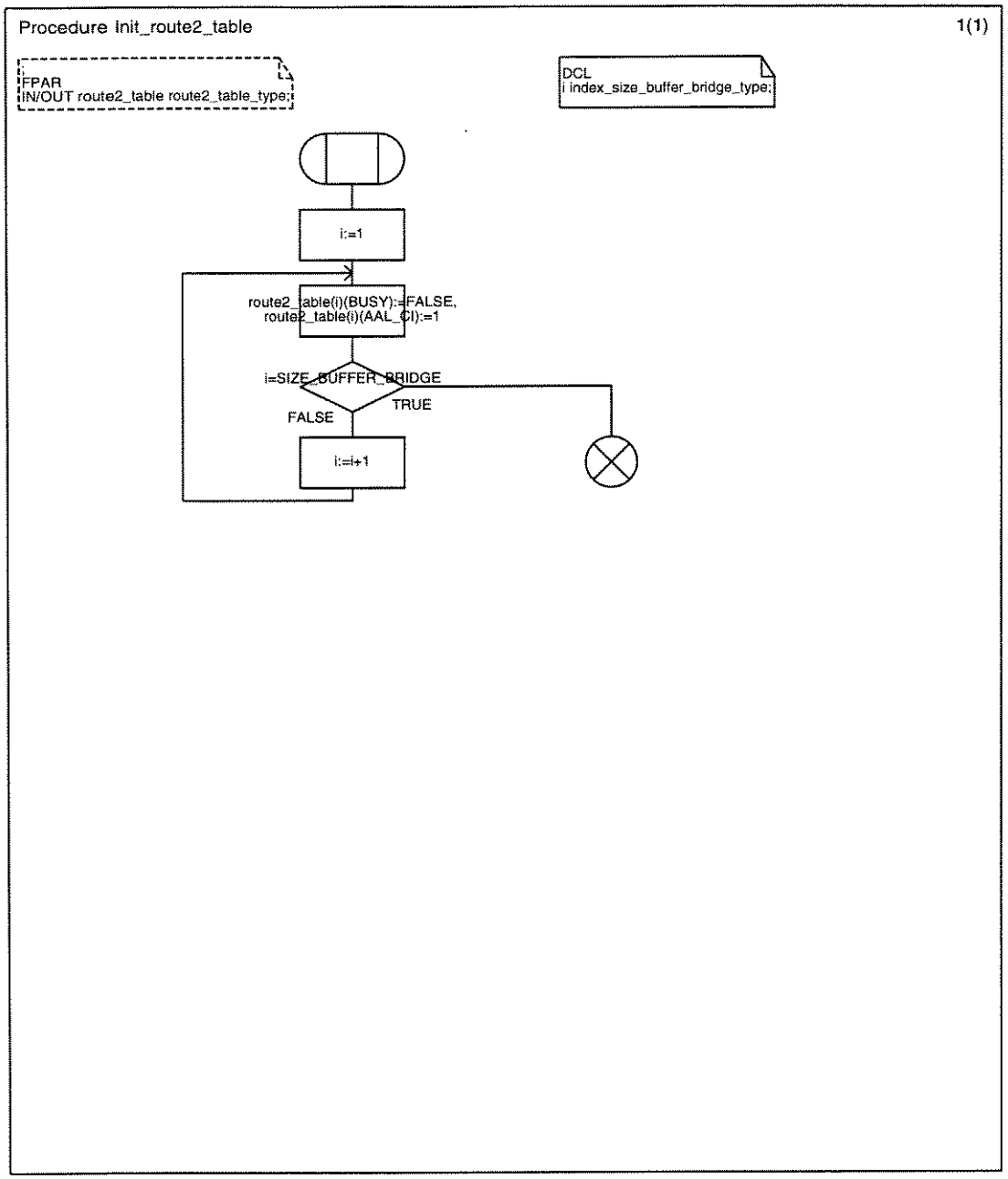

Figura B.39: "Procedimento init_route2_table"

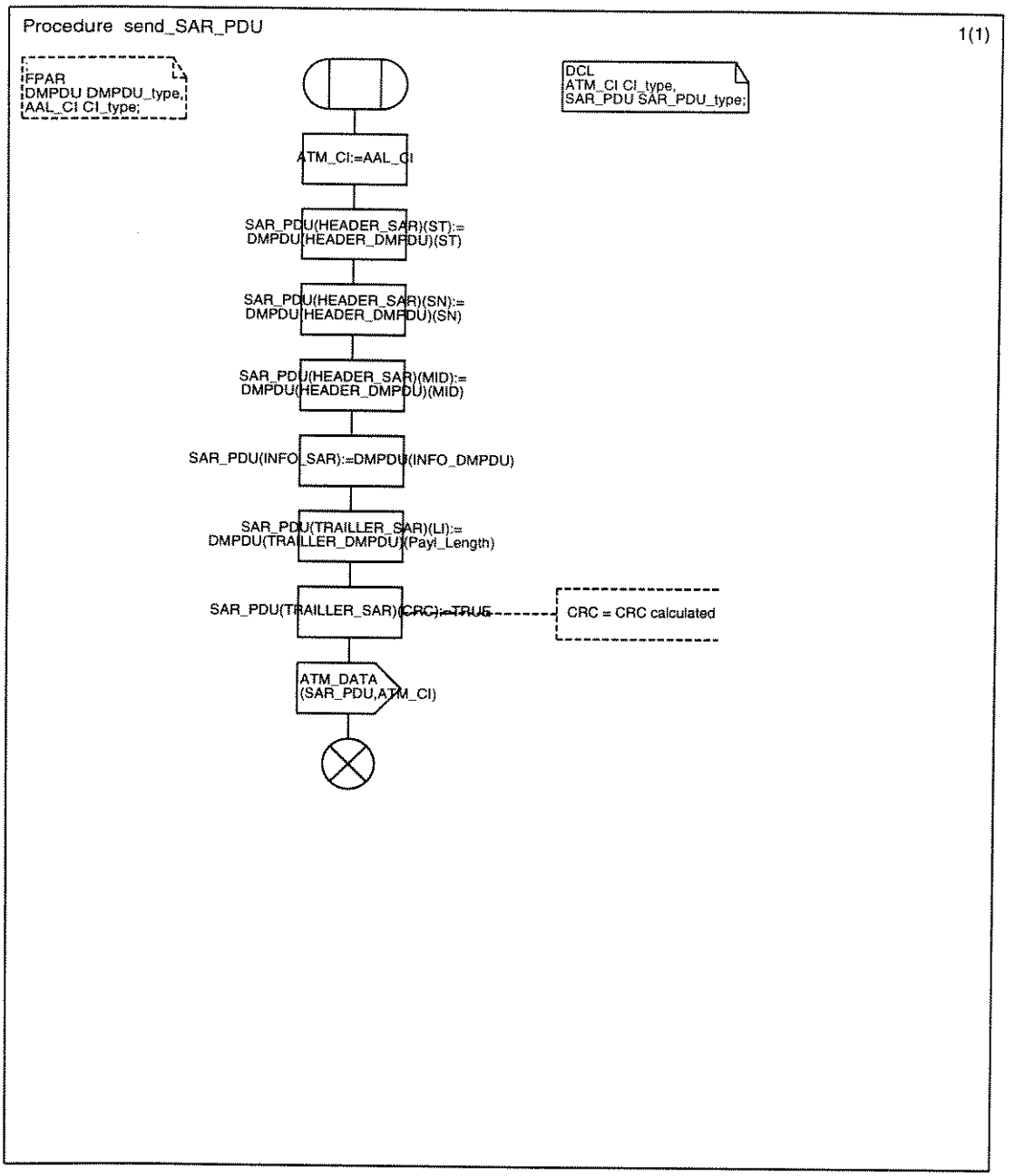

Figura B.40: "Procedimento send_SAR_PDU"

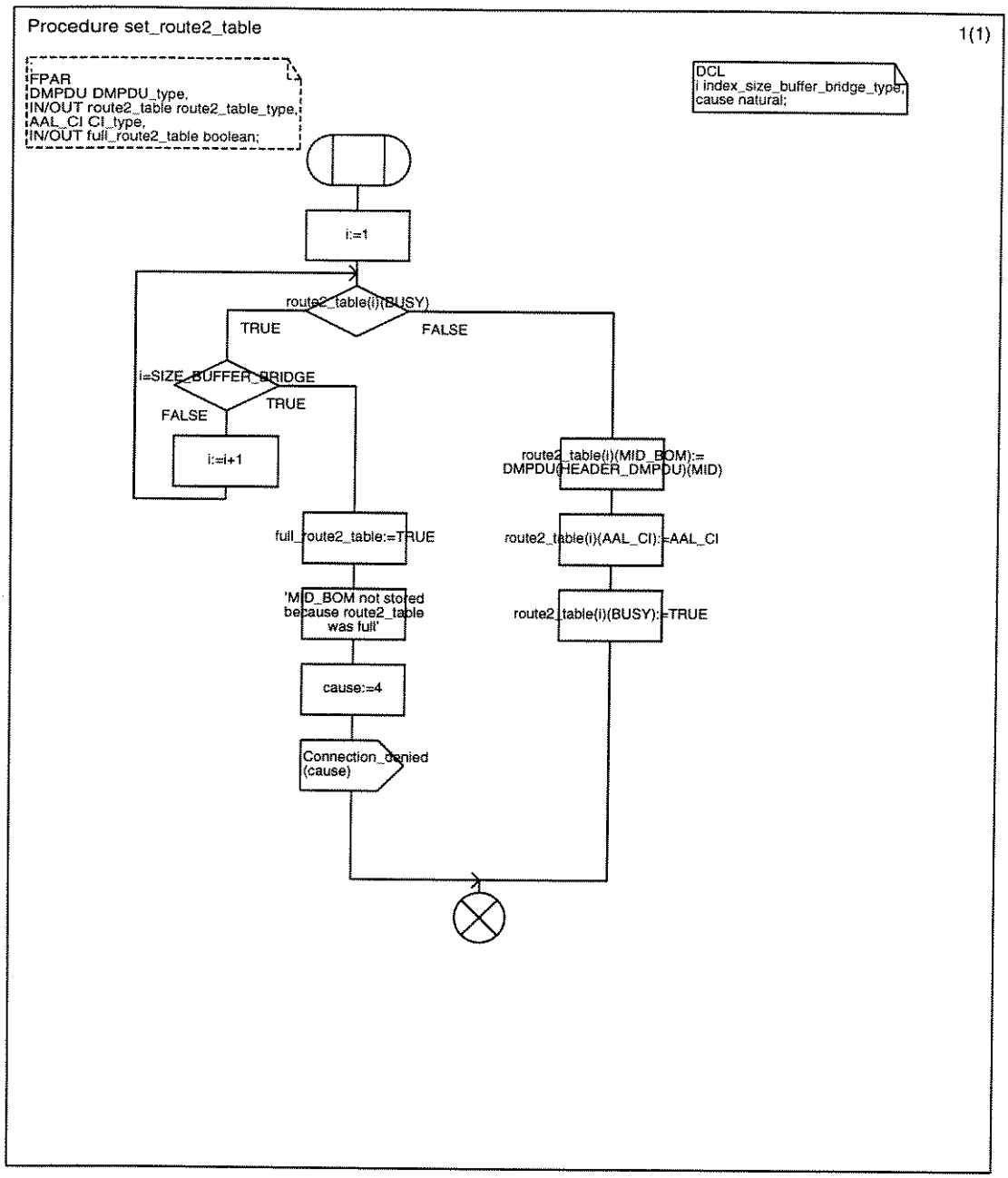

Figura B.41: "Procedimento set_route2_table"

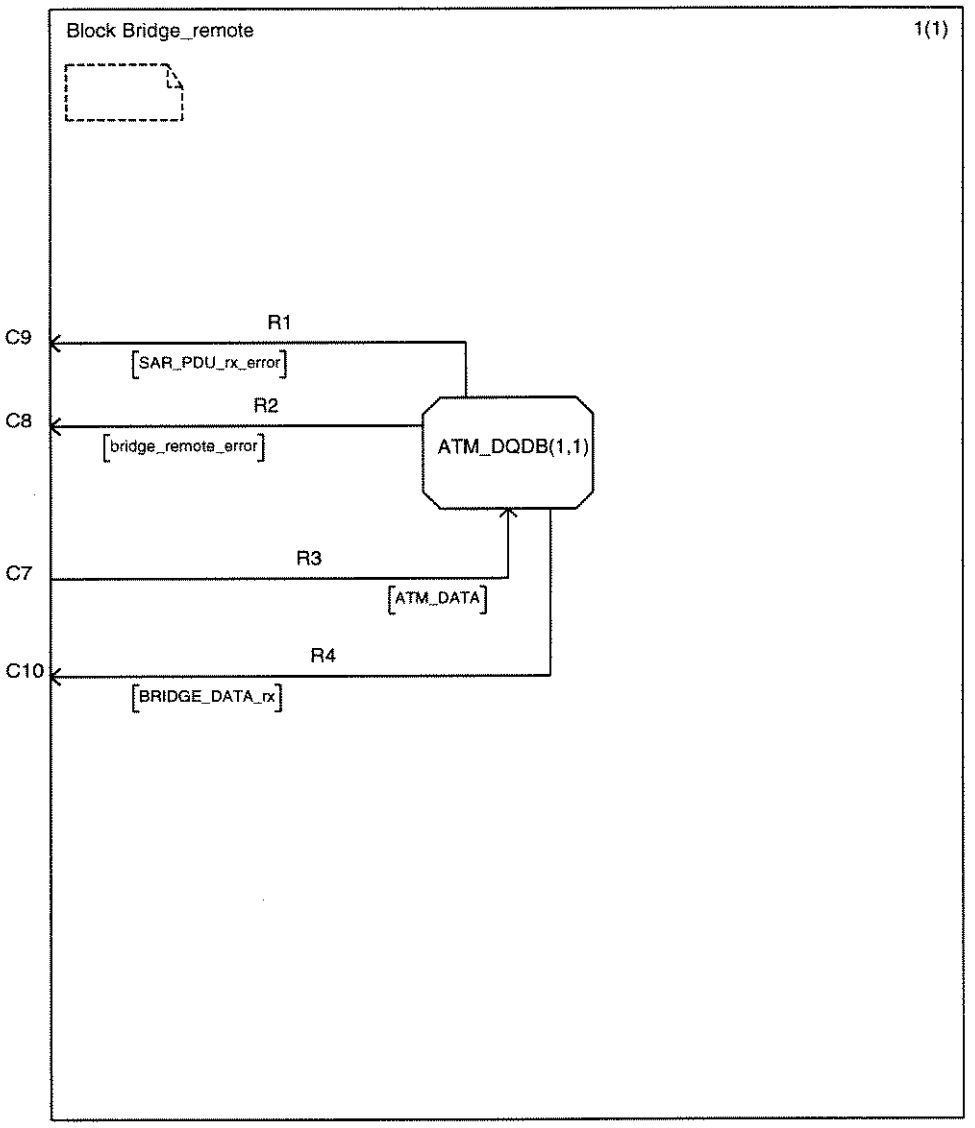

Figura B.42: "Bloco Bridge_remote"

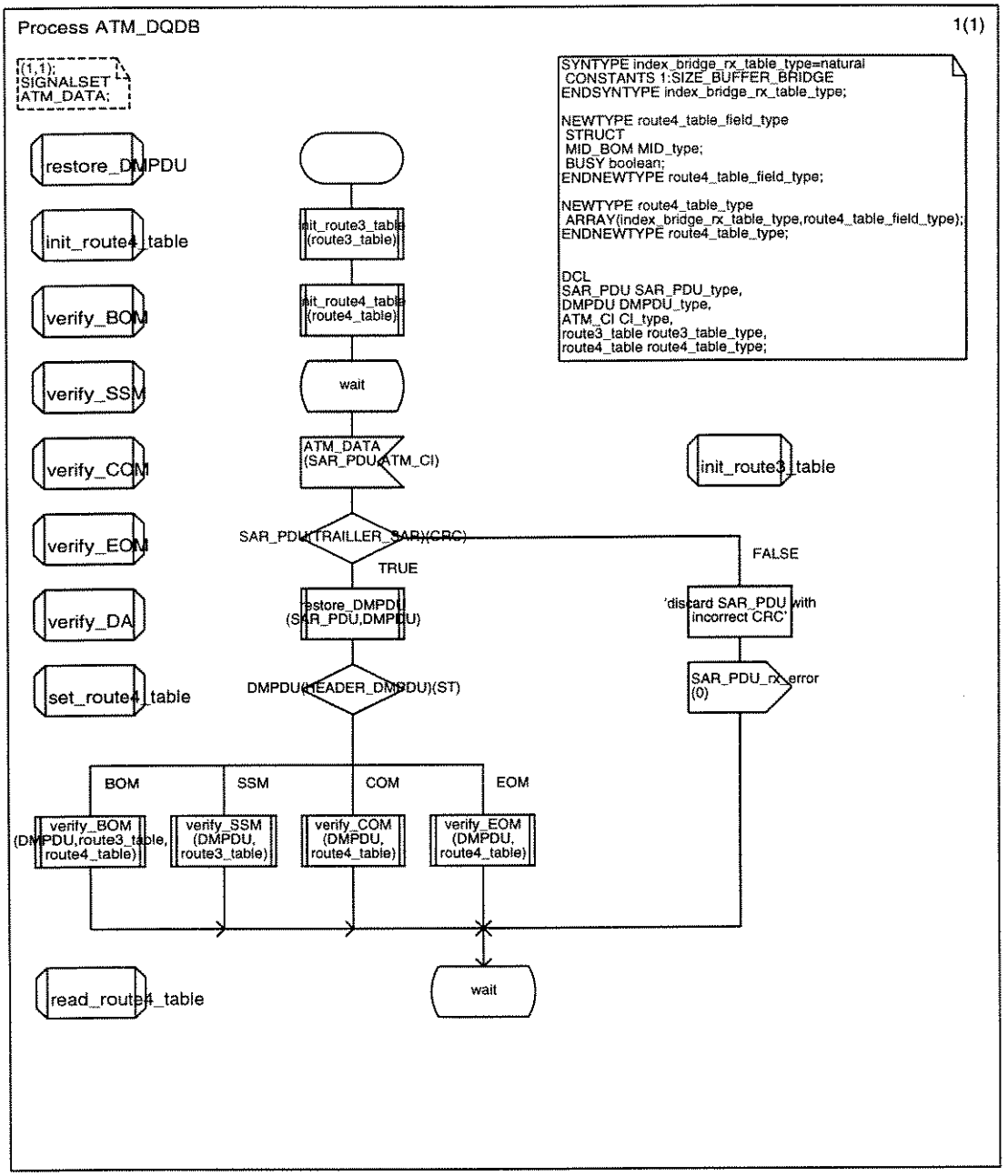

Figura B.43: "Processo ATM_DQDB"

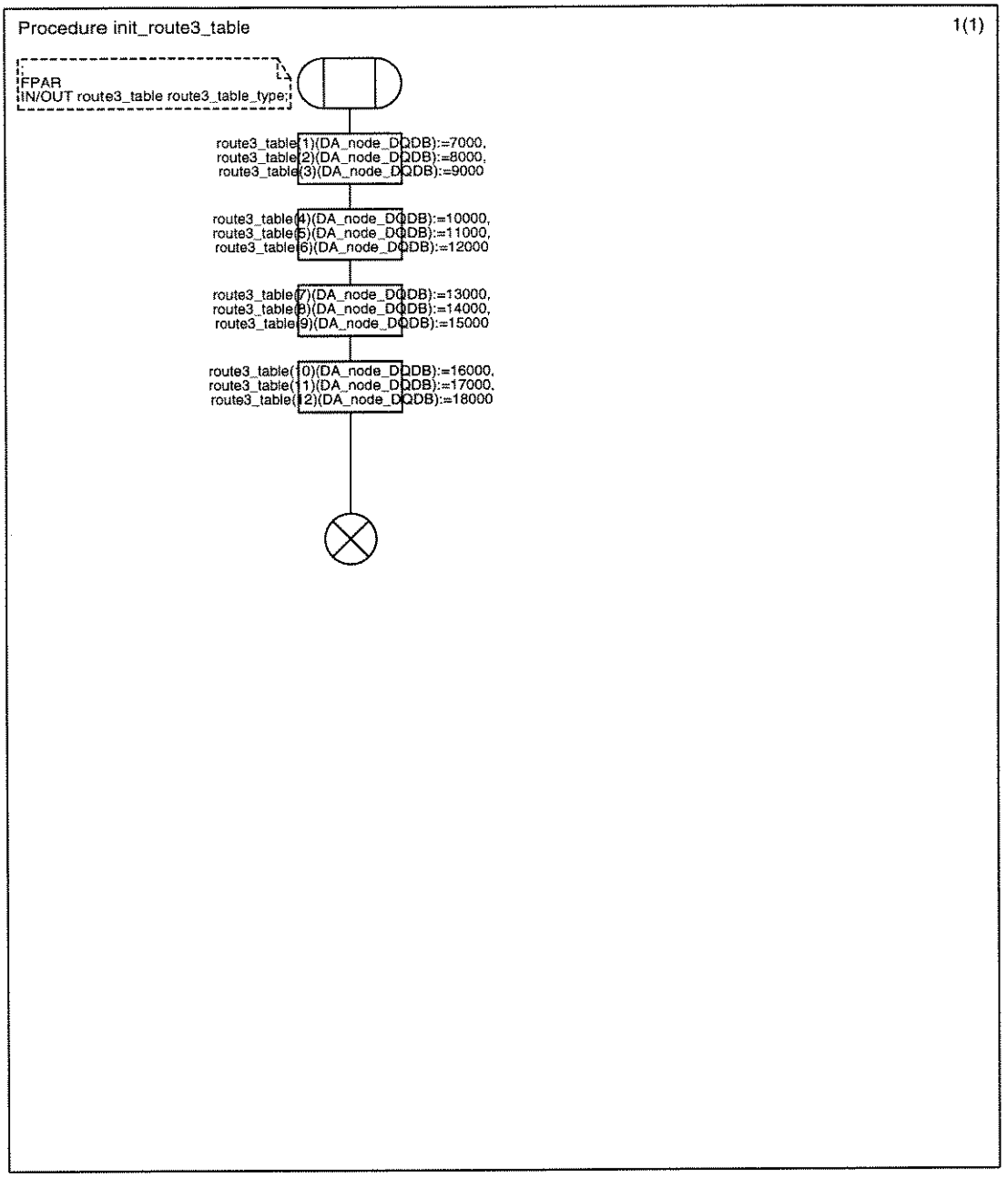

Figura B.44: "Procedimento init_route3_table"

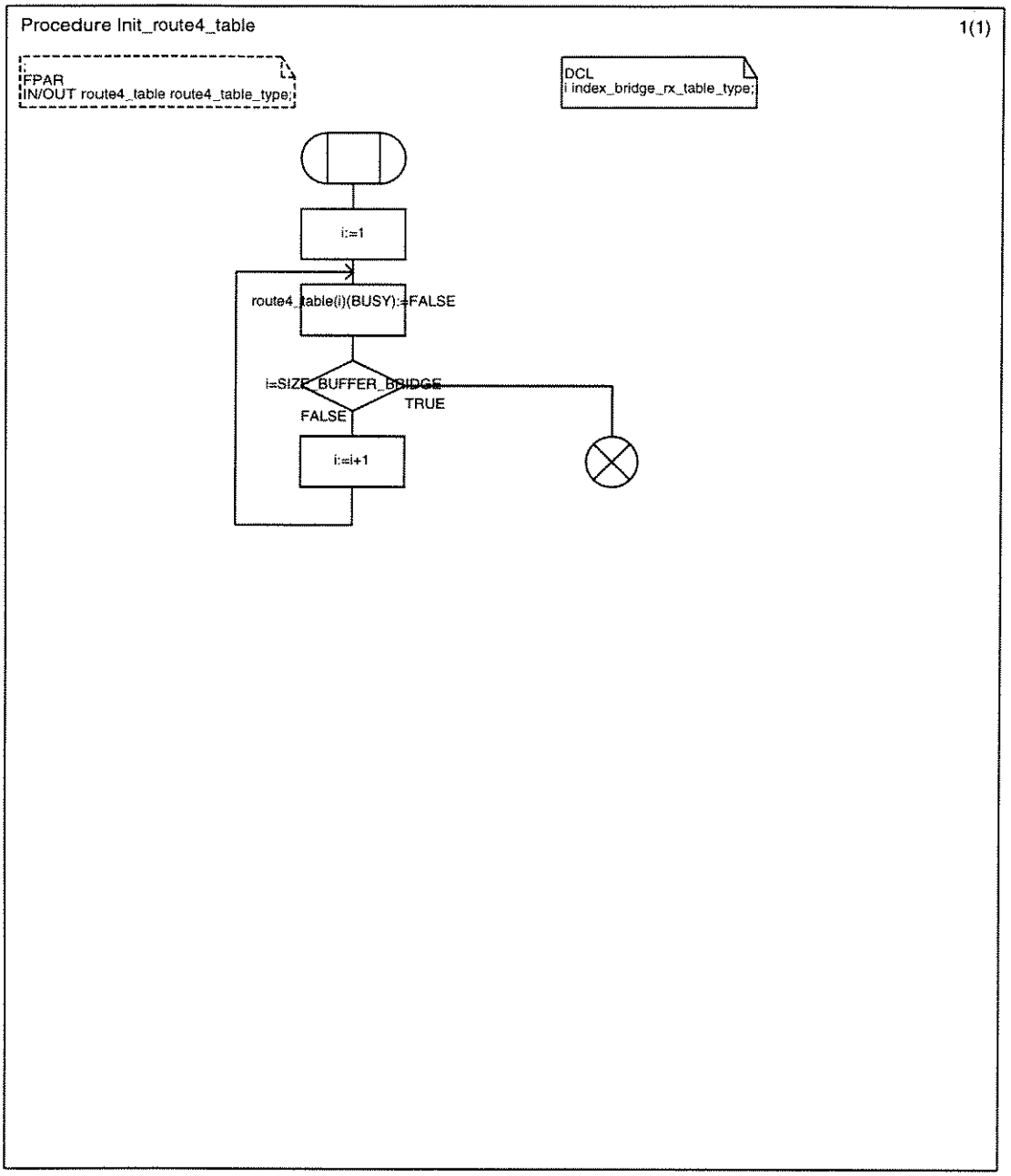

Figura B.45: "Procedimento init_route4_table"

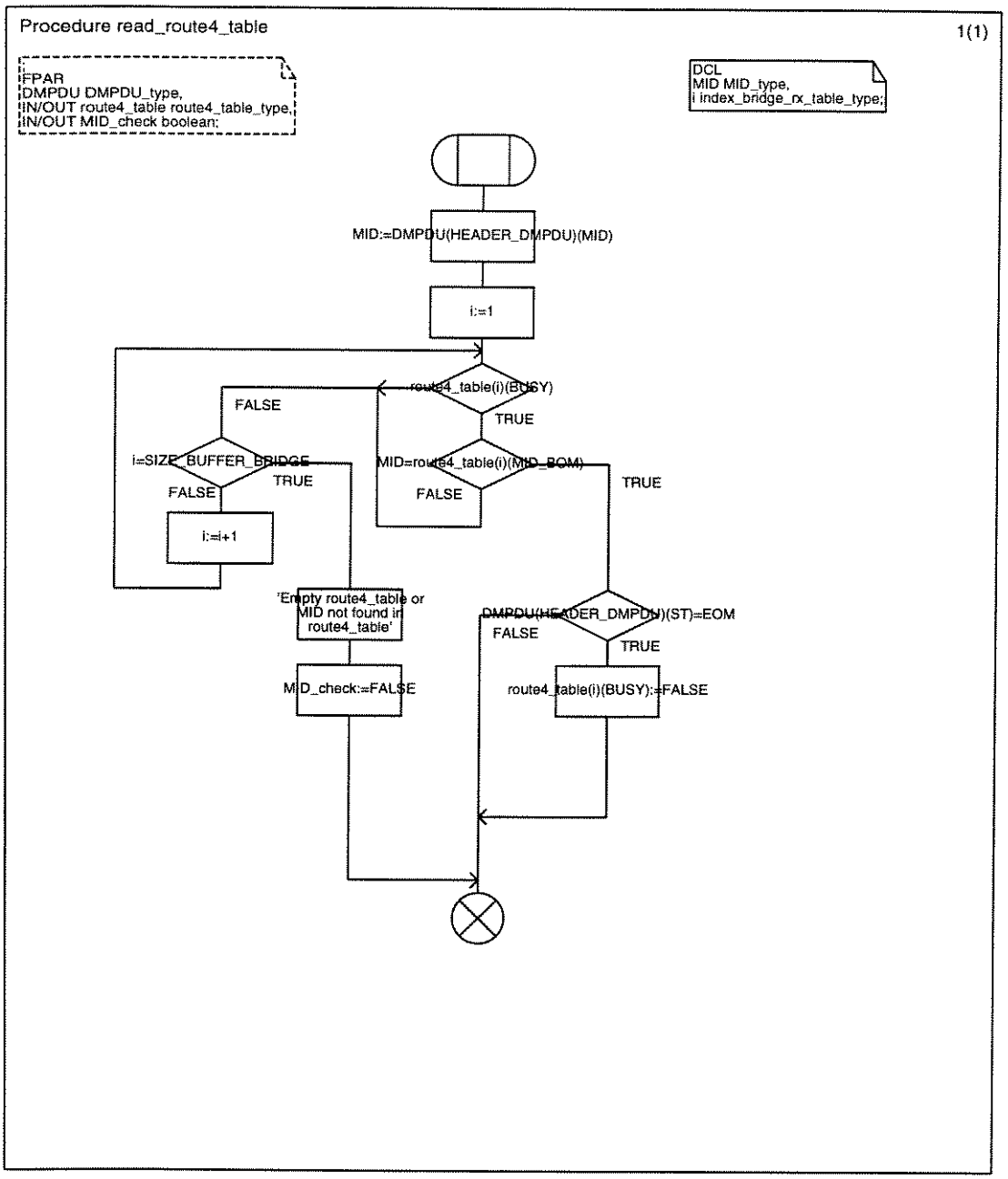

Figura B.46: "Procedimento read_route4_table"

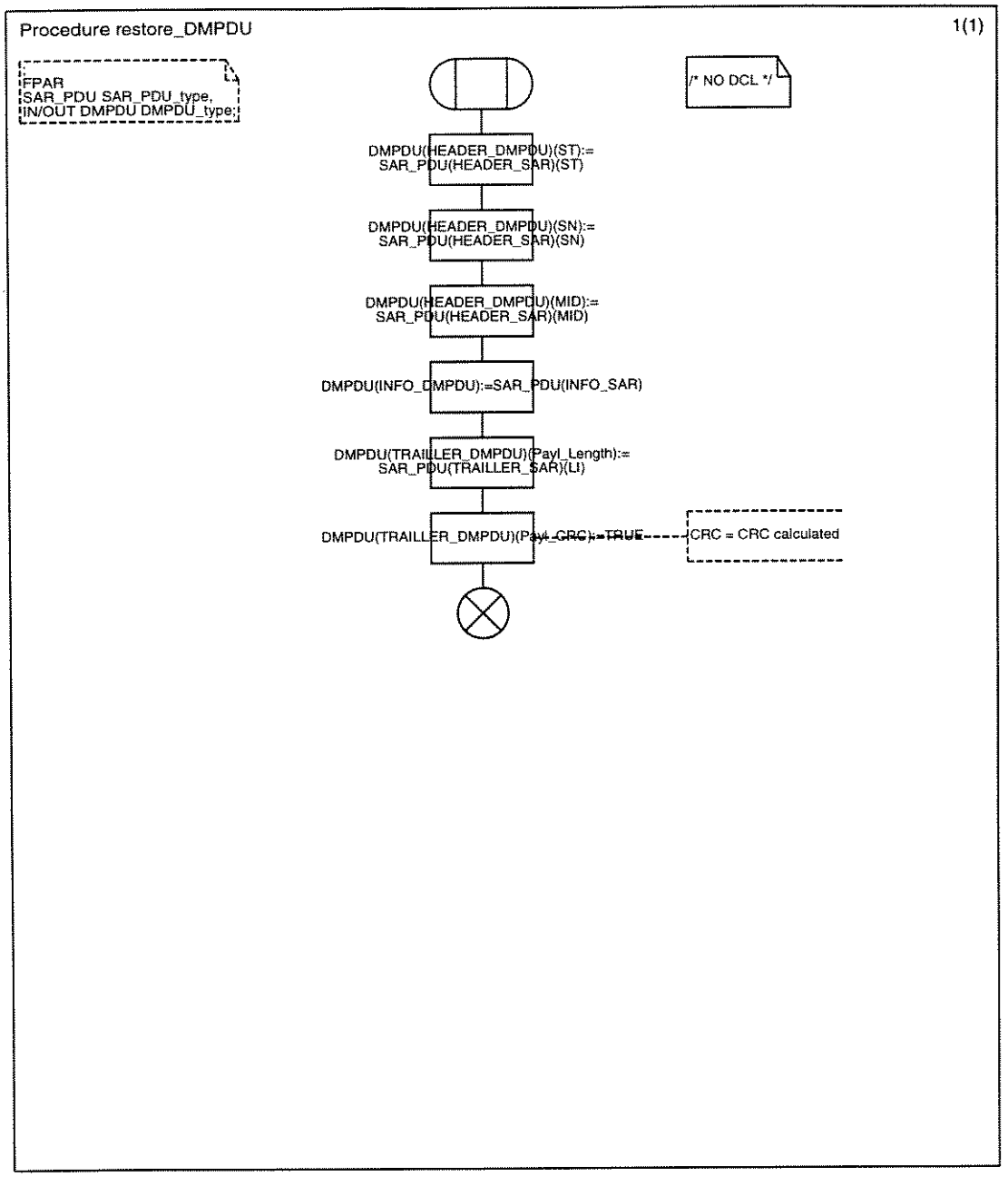

Figura B.47: "Procedimento restore_DMPDU"

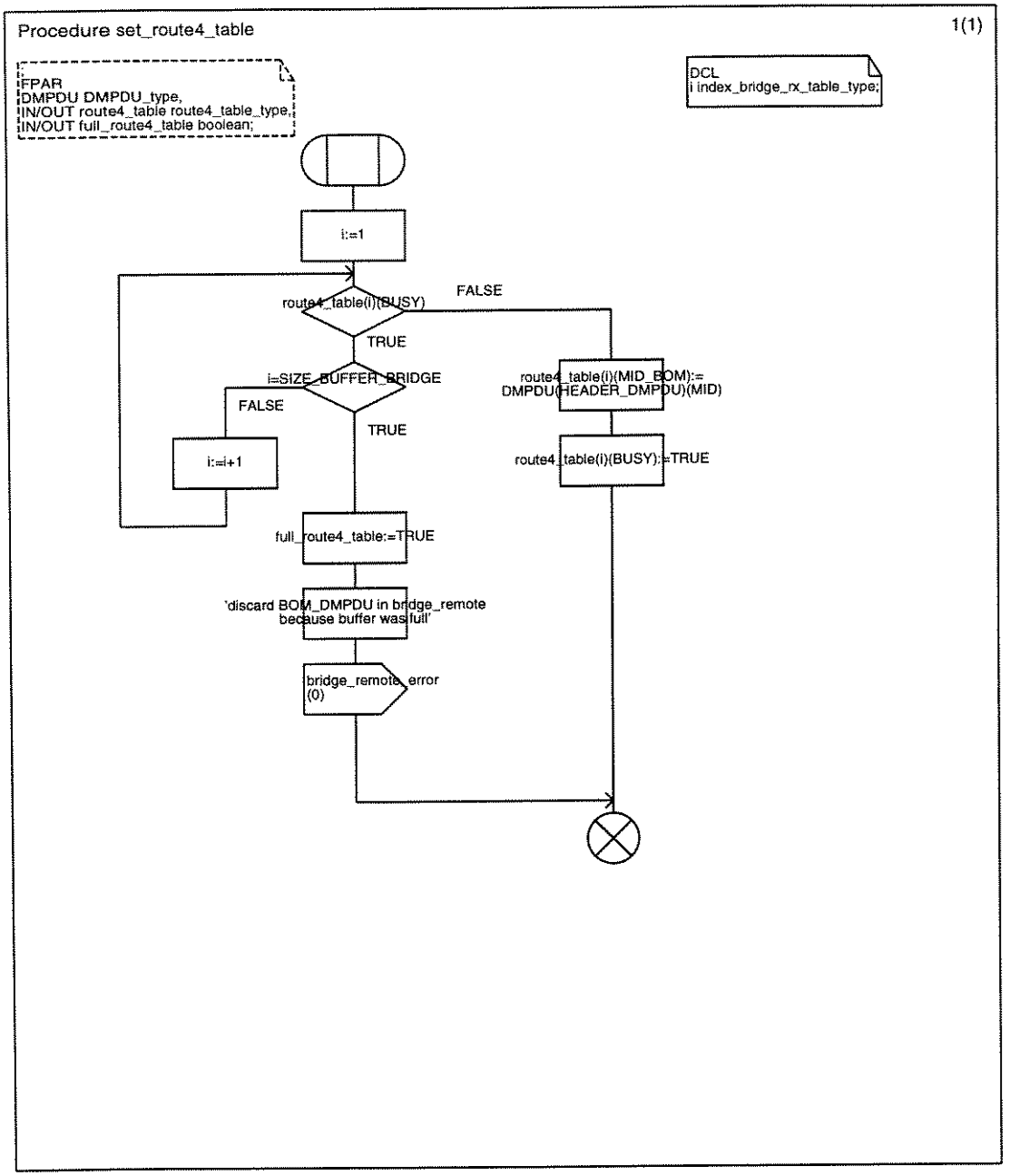

Figura B.48: "Procedimento set_route4_table"

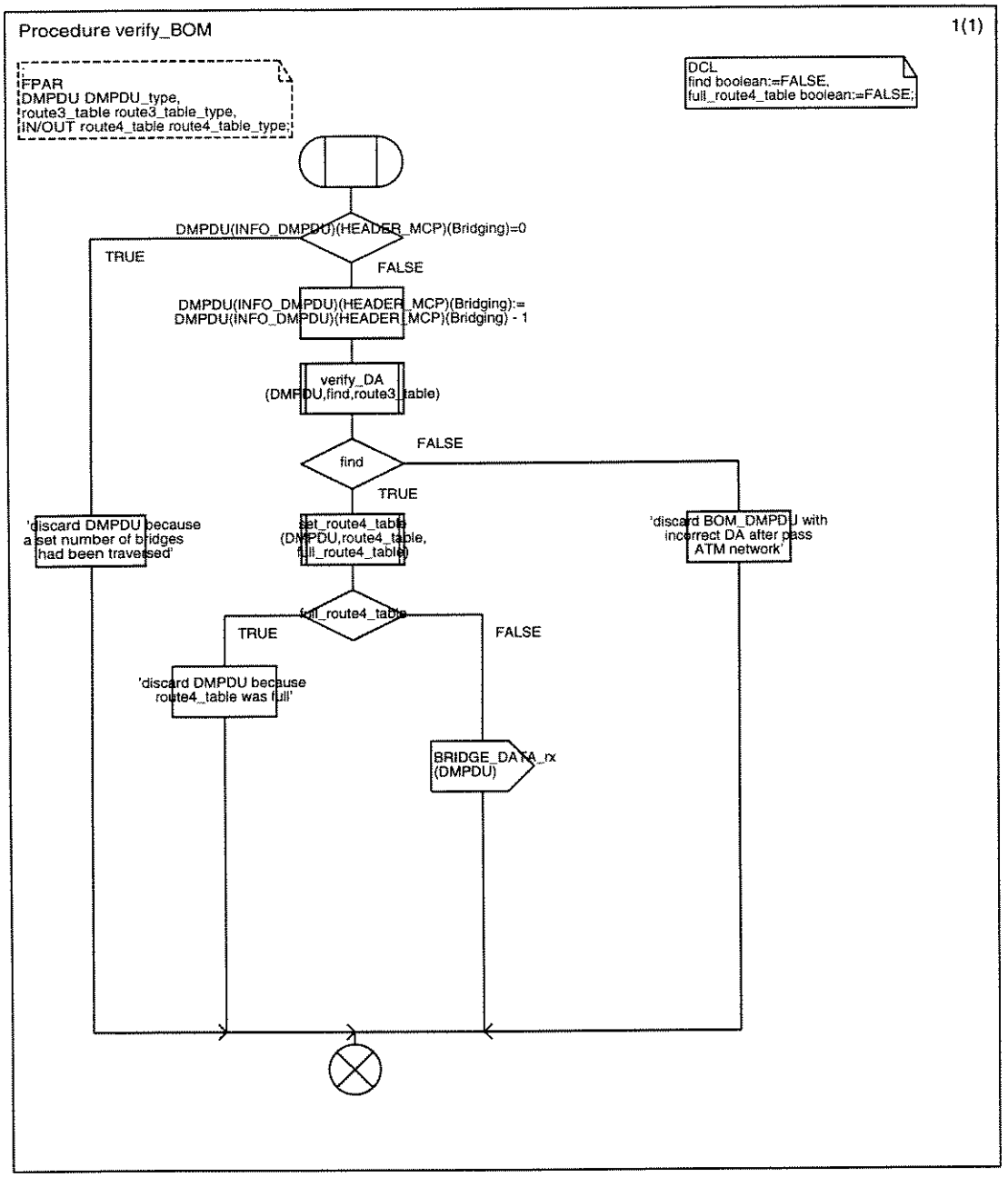

Figura B.49: "Procedimento verify_BOM"

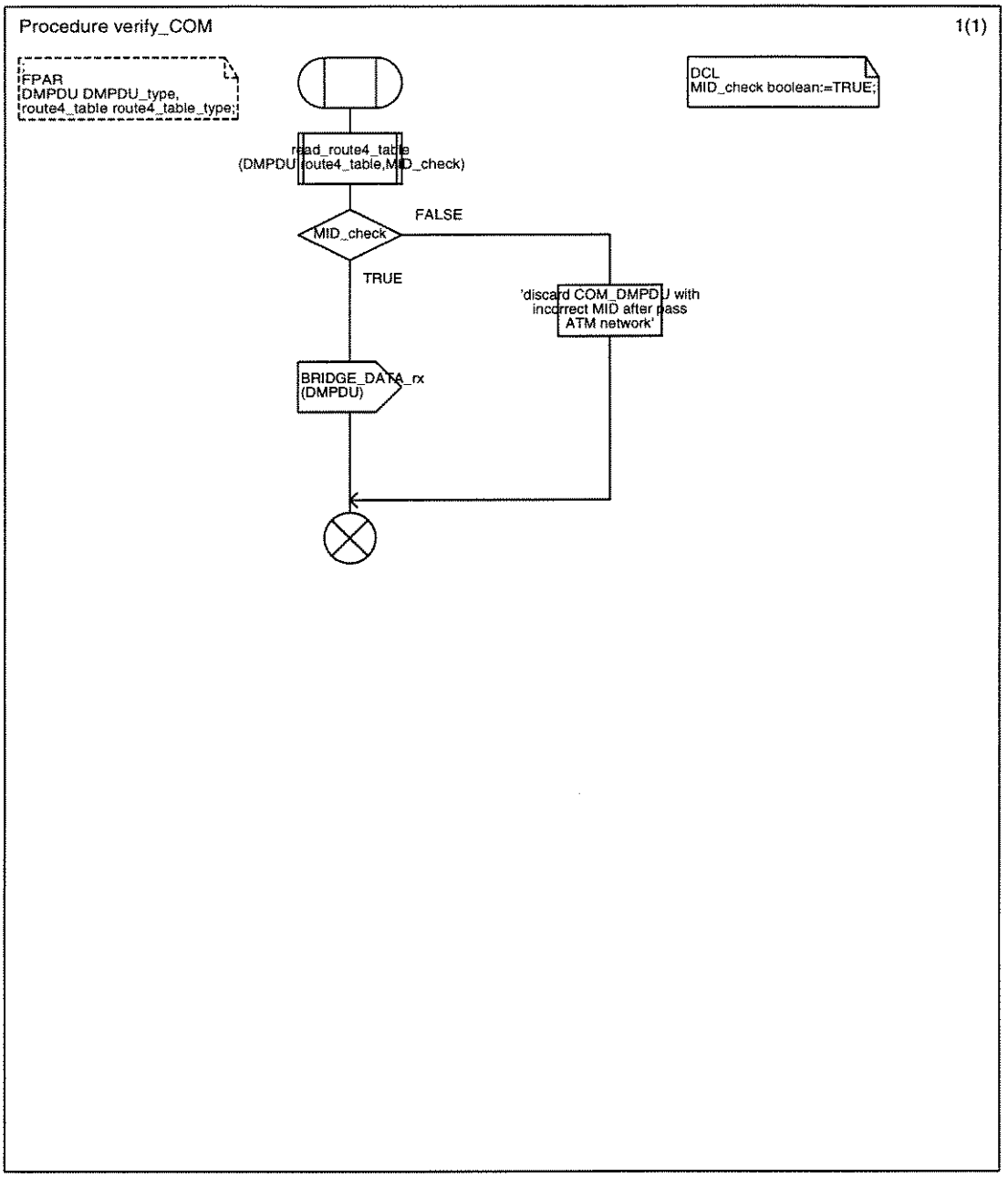

Figura B.50: "Procedimento verify_COM"

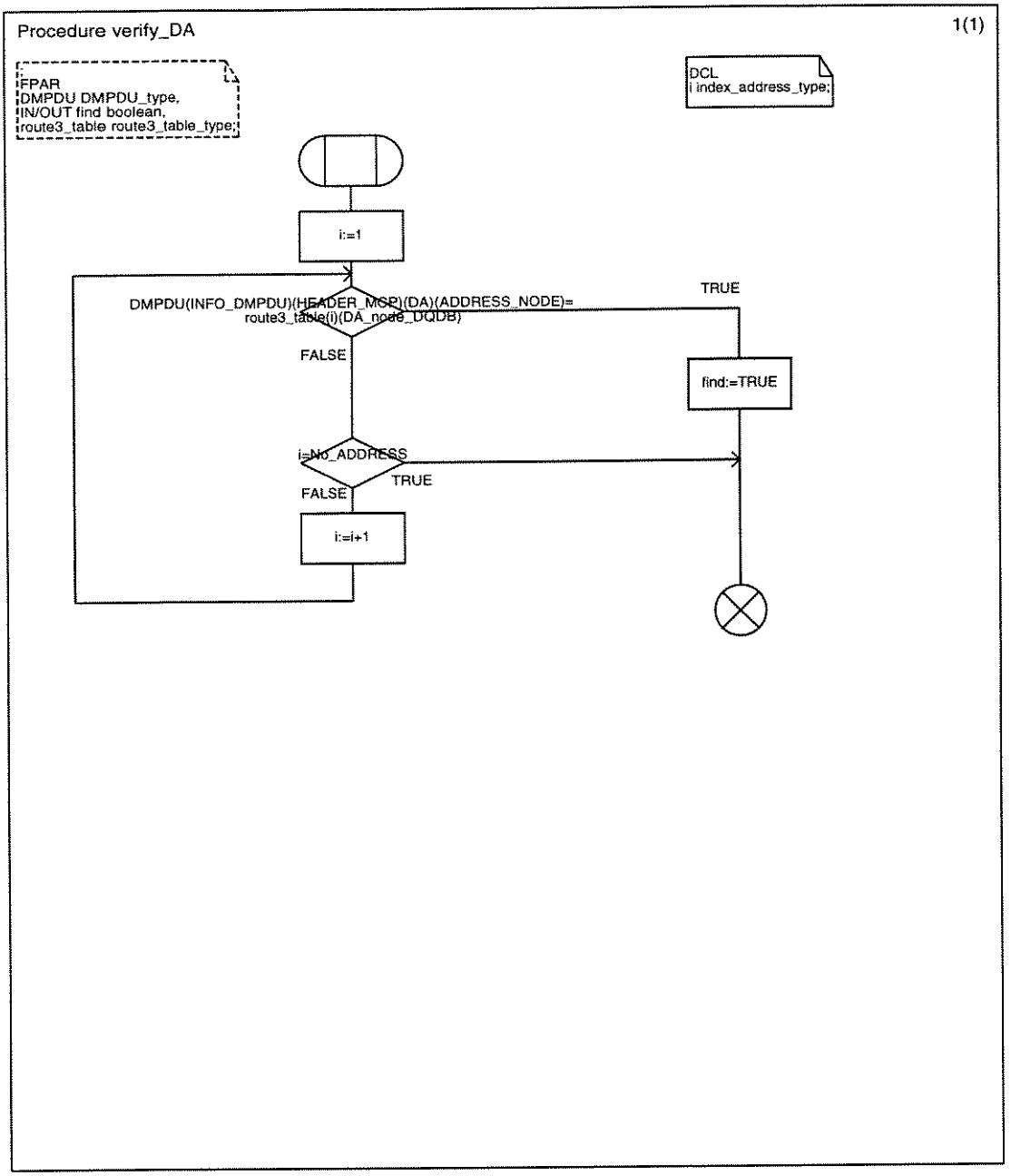

Figura B.51: "Procedimento verify_DA"

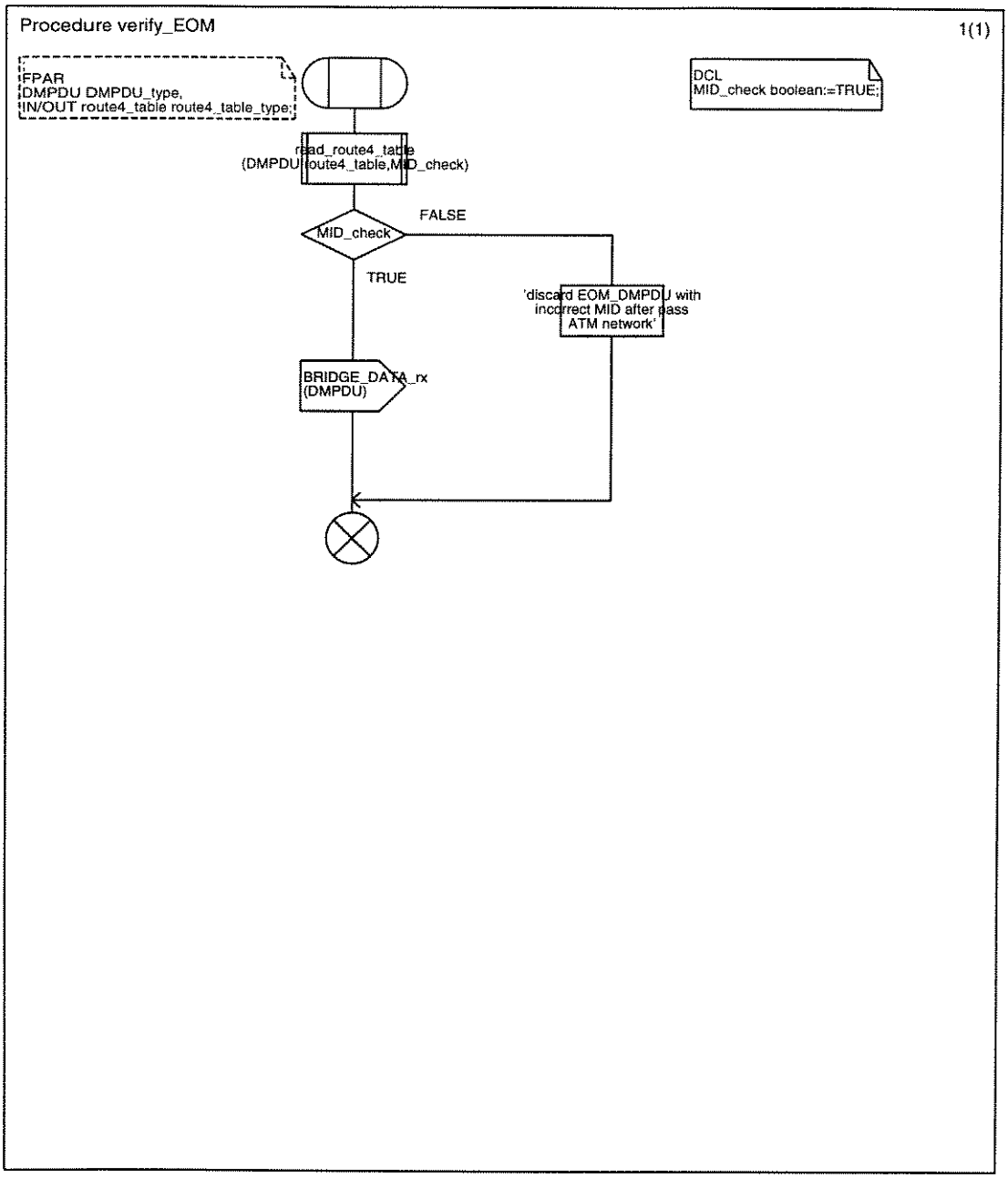

Figura B.52: "Procedimento verify_EOM"

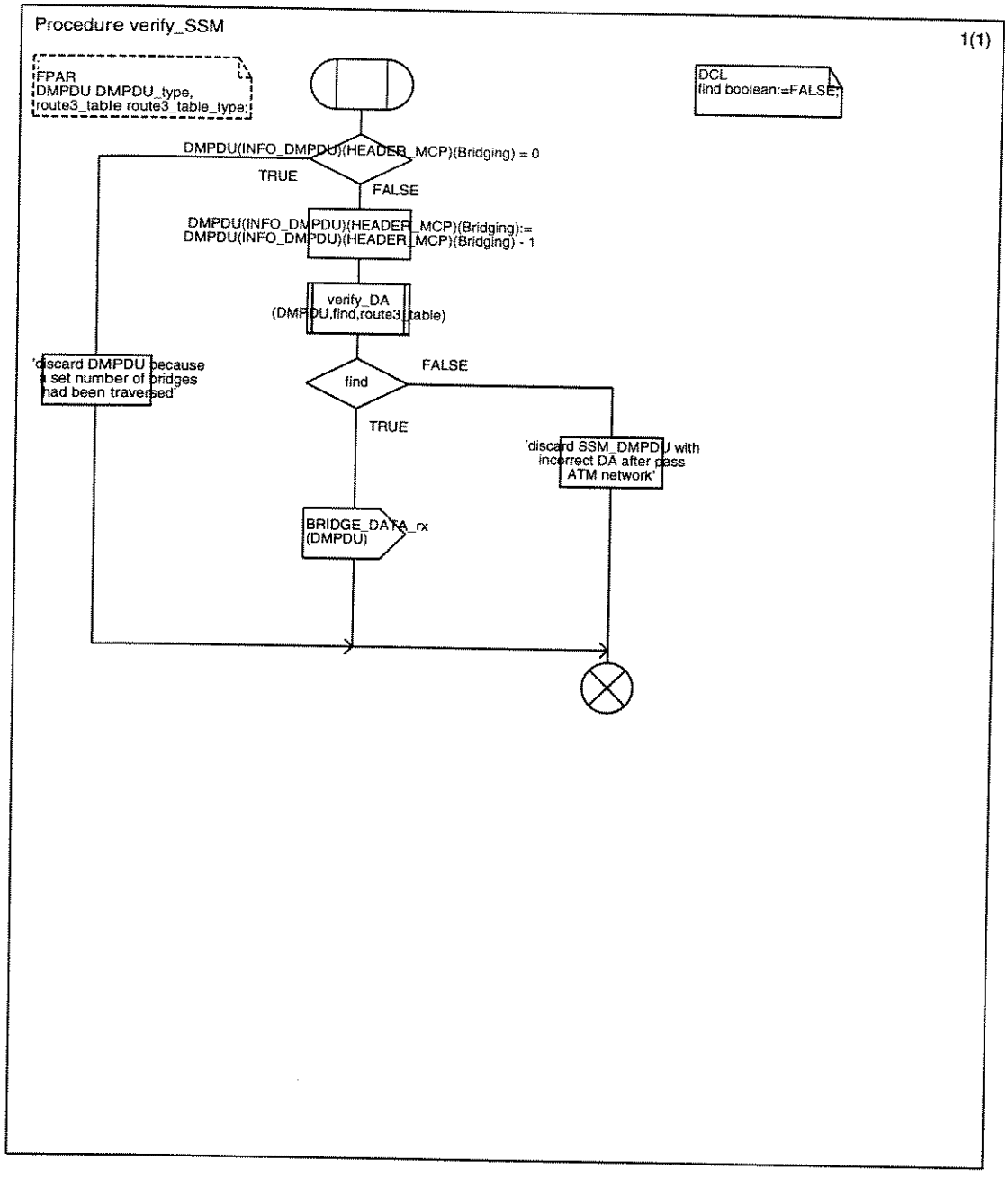

Figura B.53: "Procedimento verify_SSM"

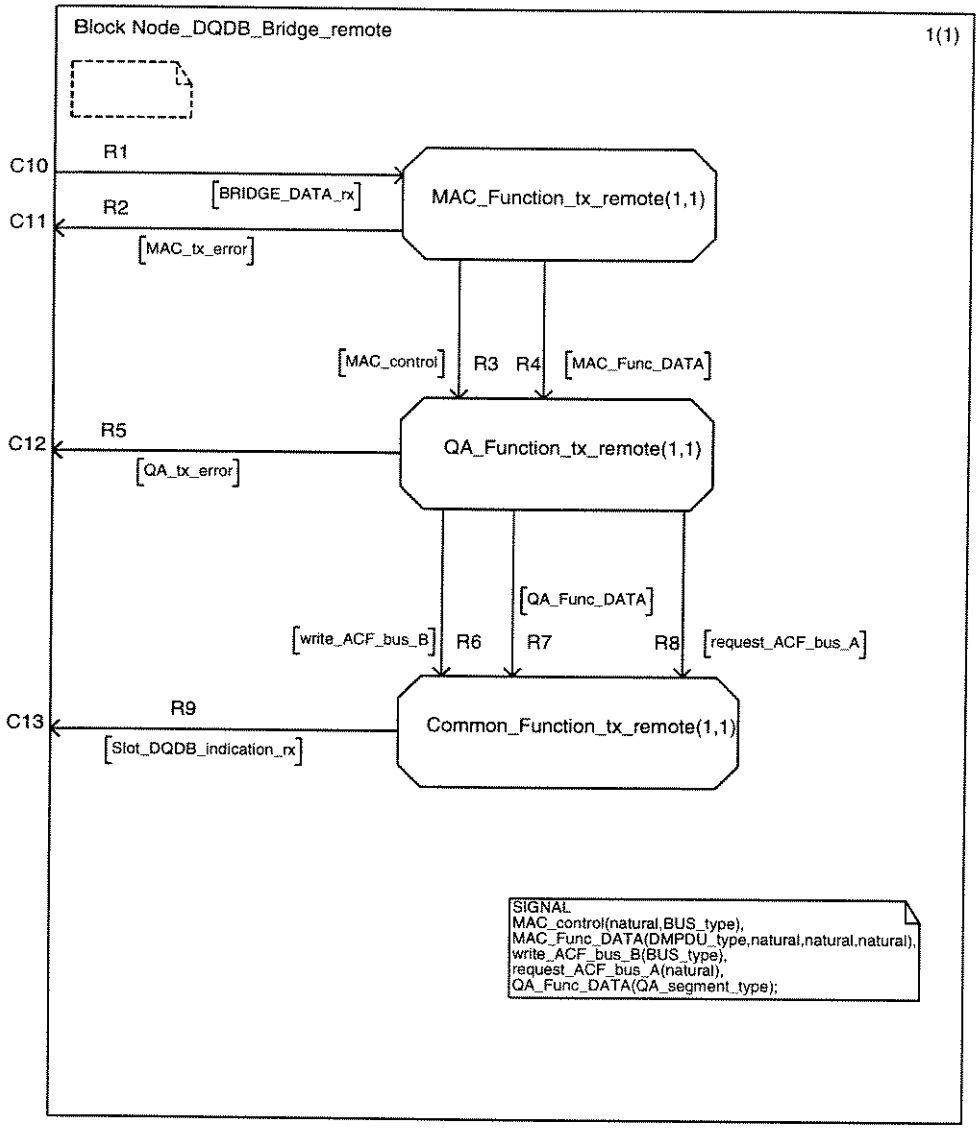

Figura B.54: "Bloco Node_DQDB_Bridge_remote"

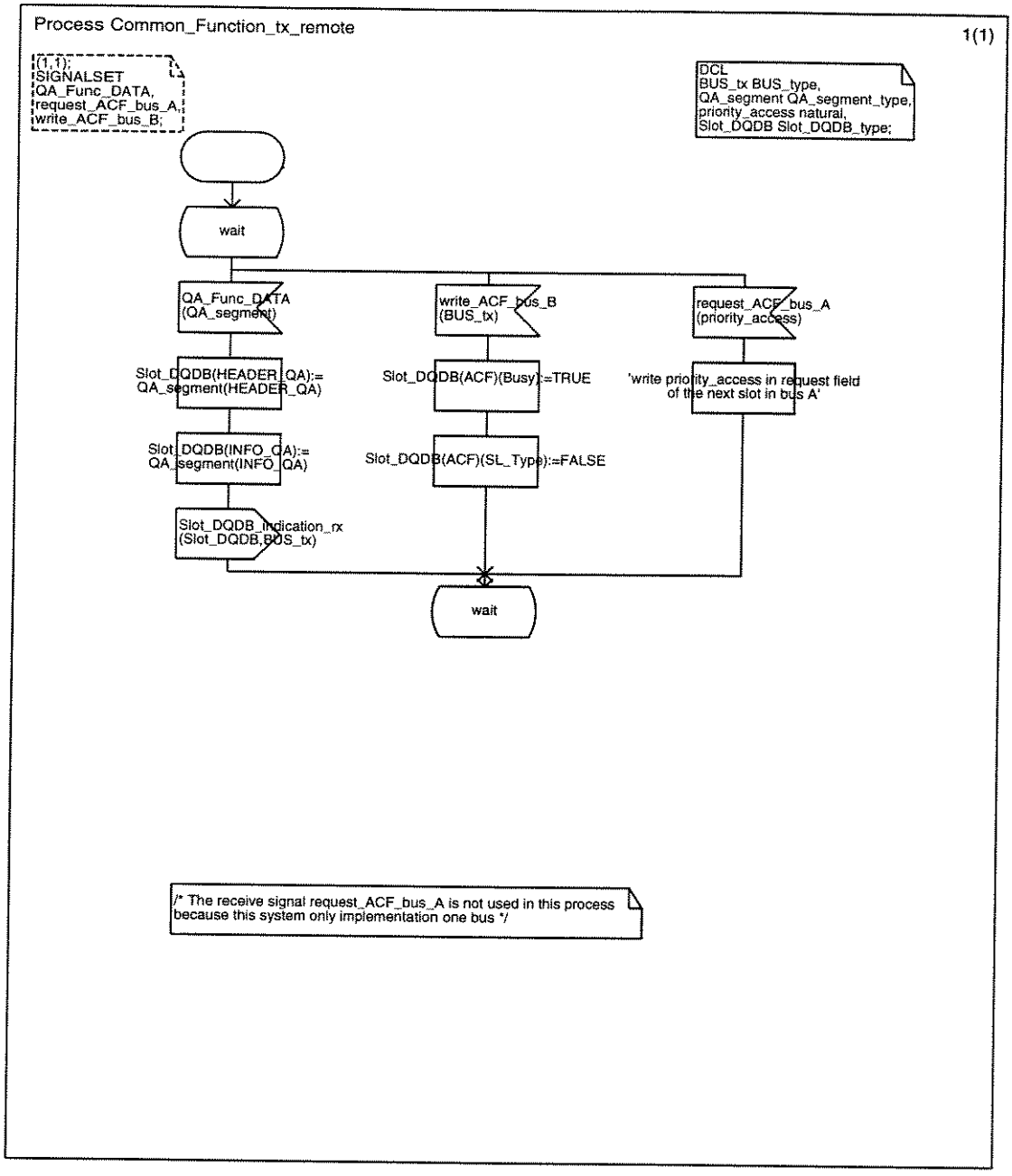

Figura B.55: "Processo Common_Function_tx_remote"

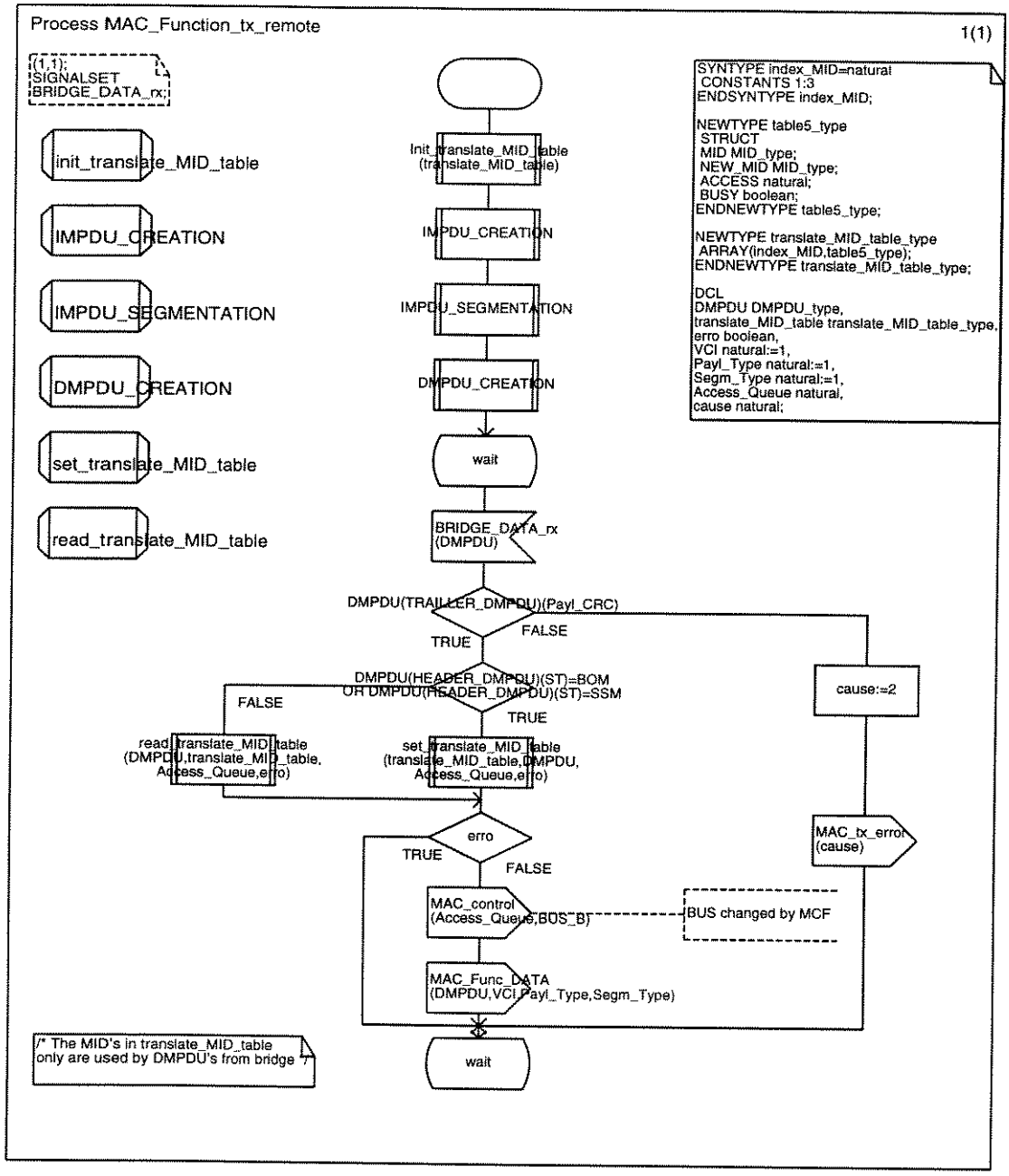

Figura B.56: "Processo MAC_Function_tx_remote"

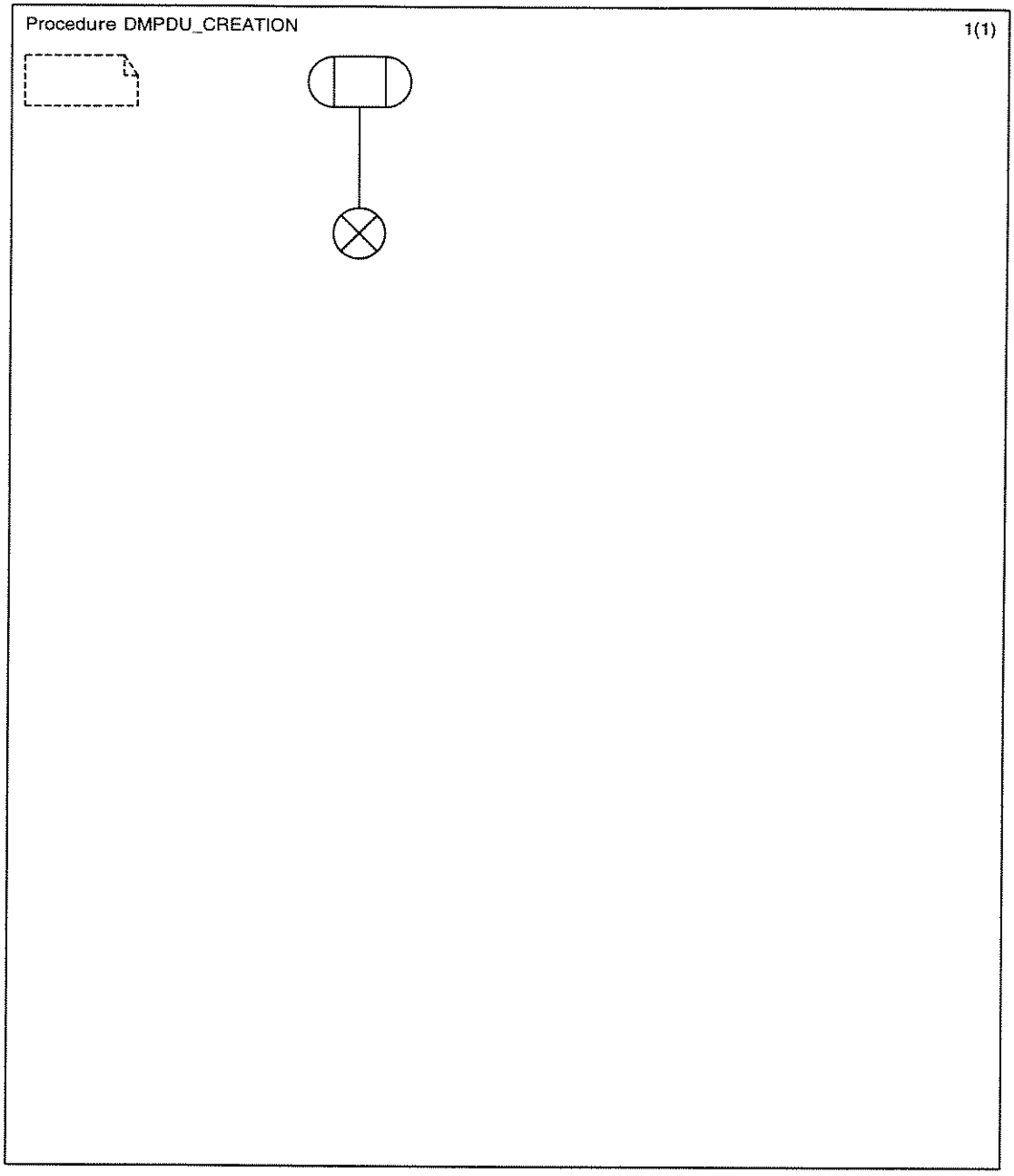

Figura B.57: "Procedimento DMPDU_CREATION"

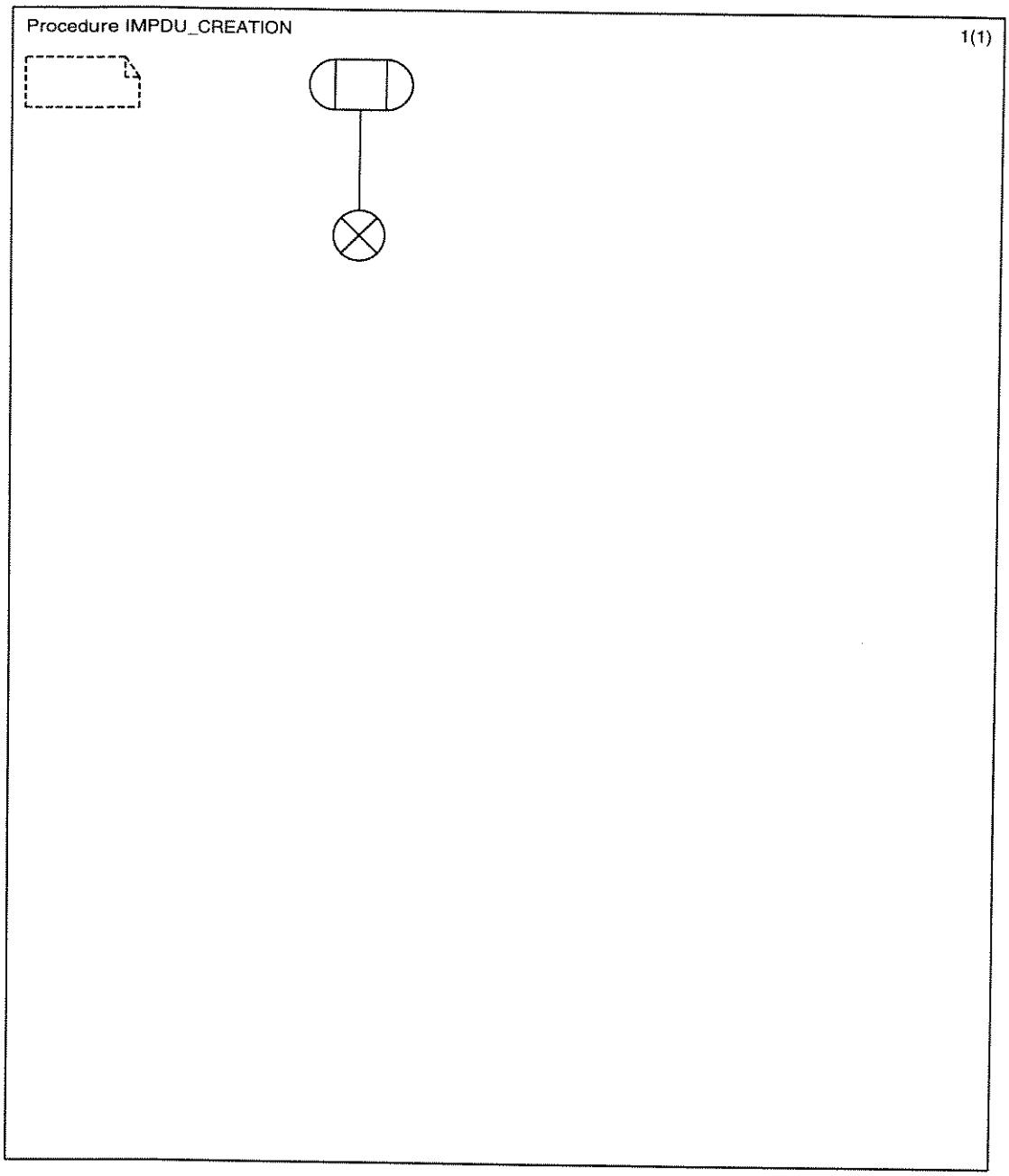

Figura B.58: "Procedimento IMPDU_CREATION"

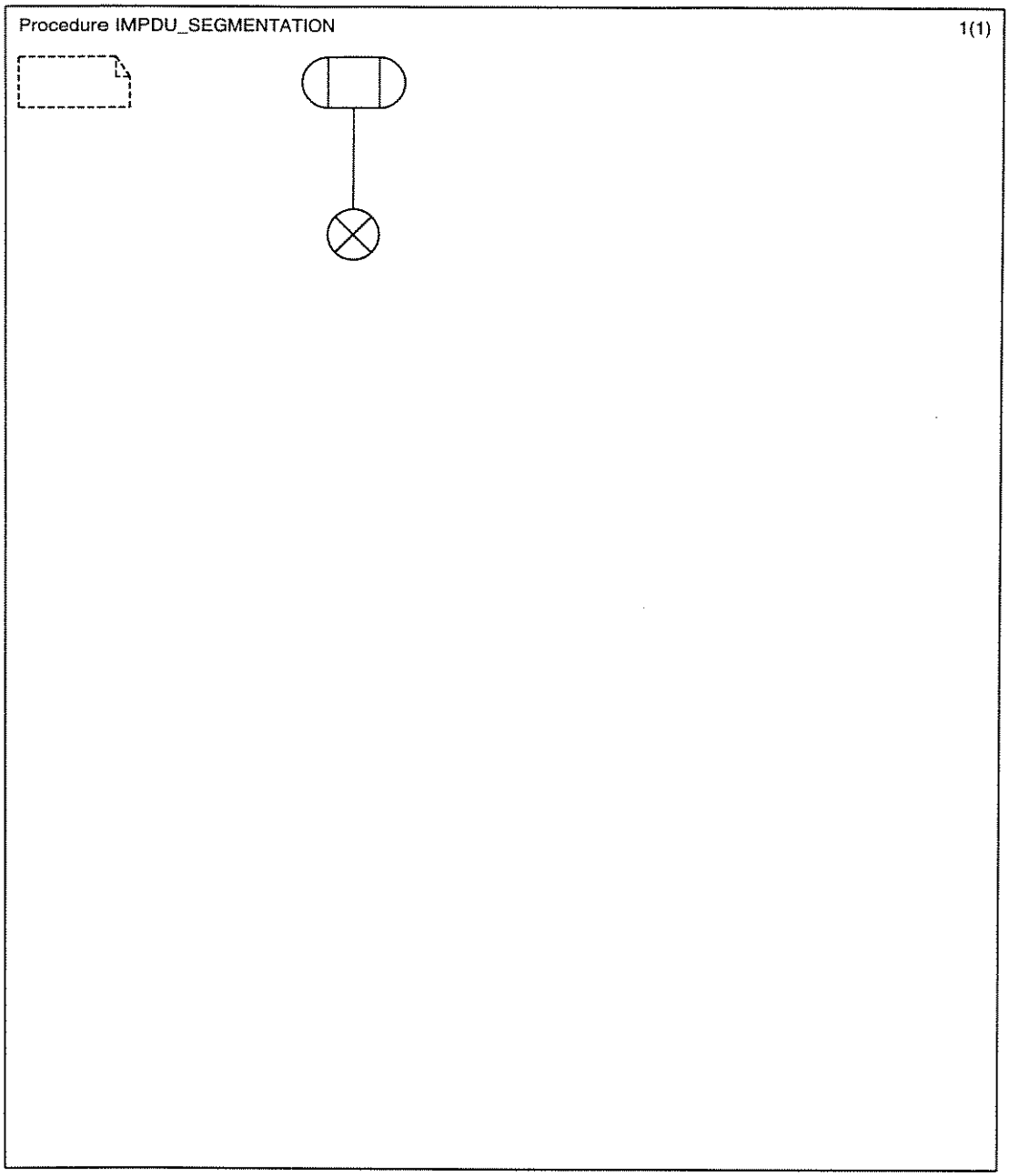

Figura B.59: "Procedimento IMPDU_SEGMENTATION"

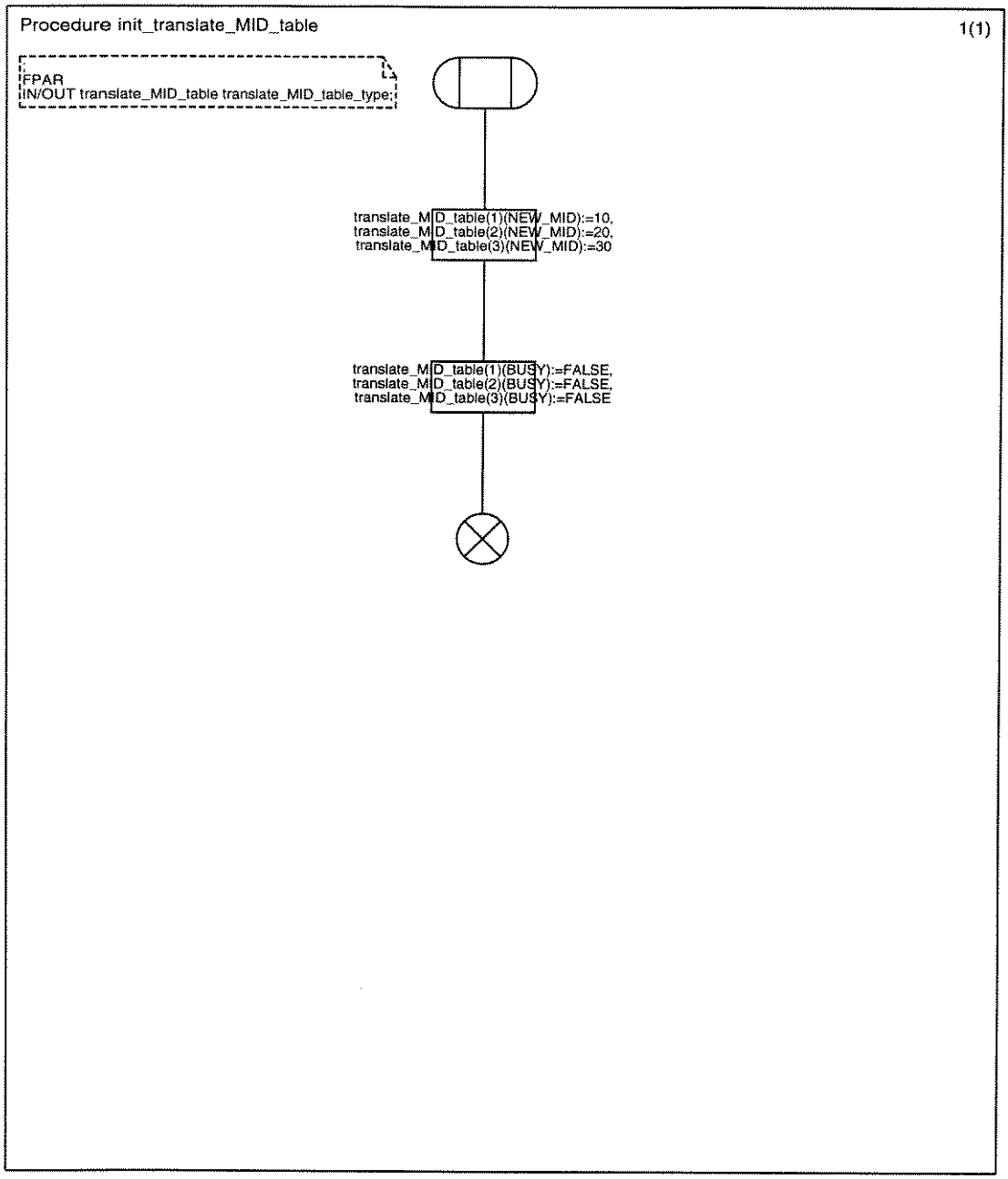

Figura B.60: "Procedimento init_translate_MID_table"

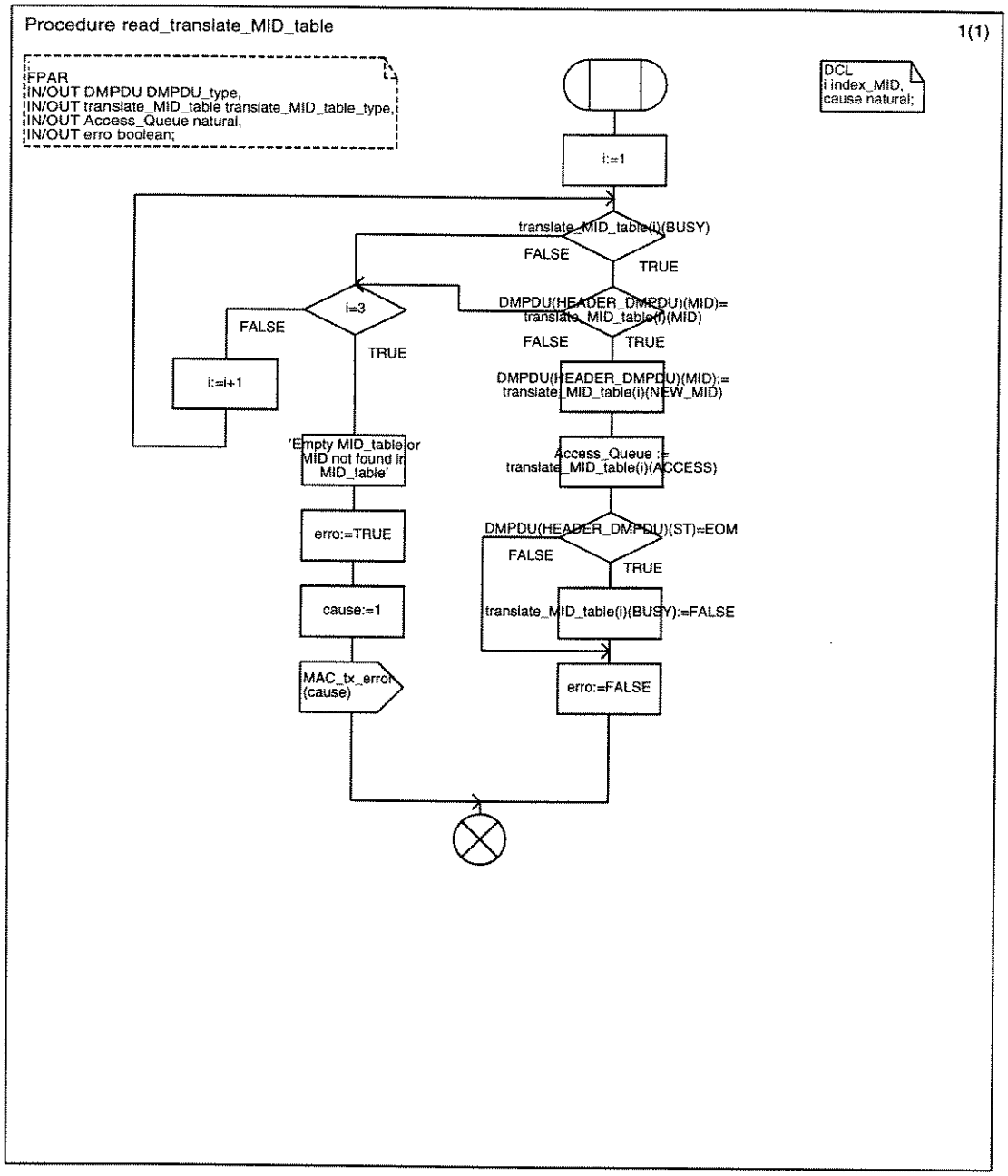

Figura B.61: "Procedimento read_translate_MID_table"

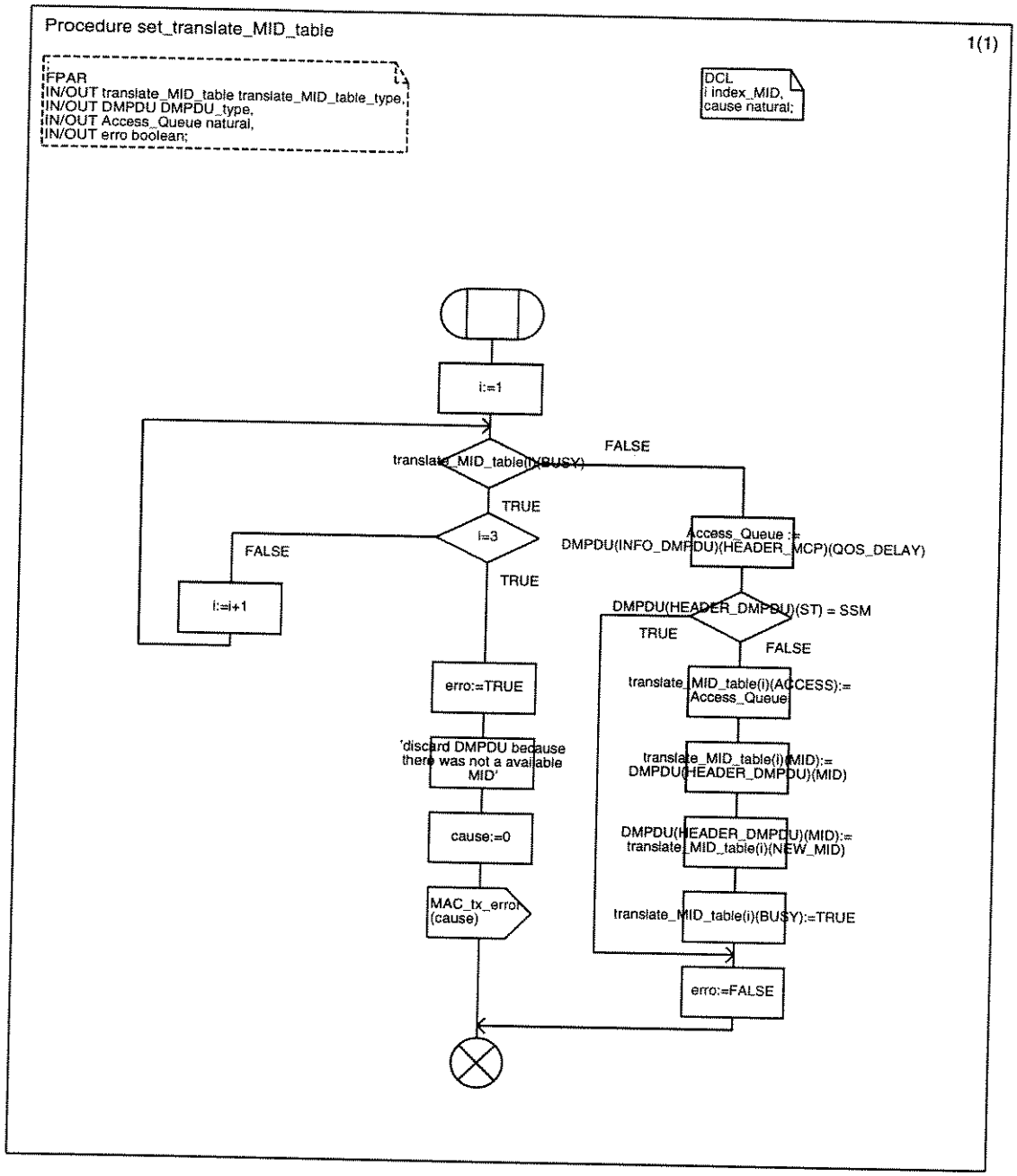

Figura B.62: "Procedimento set_translate_MID_table"

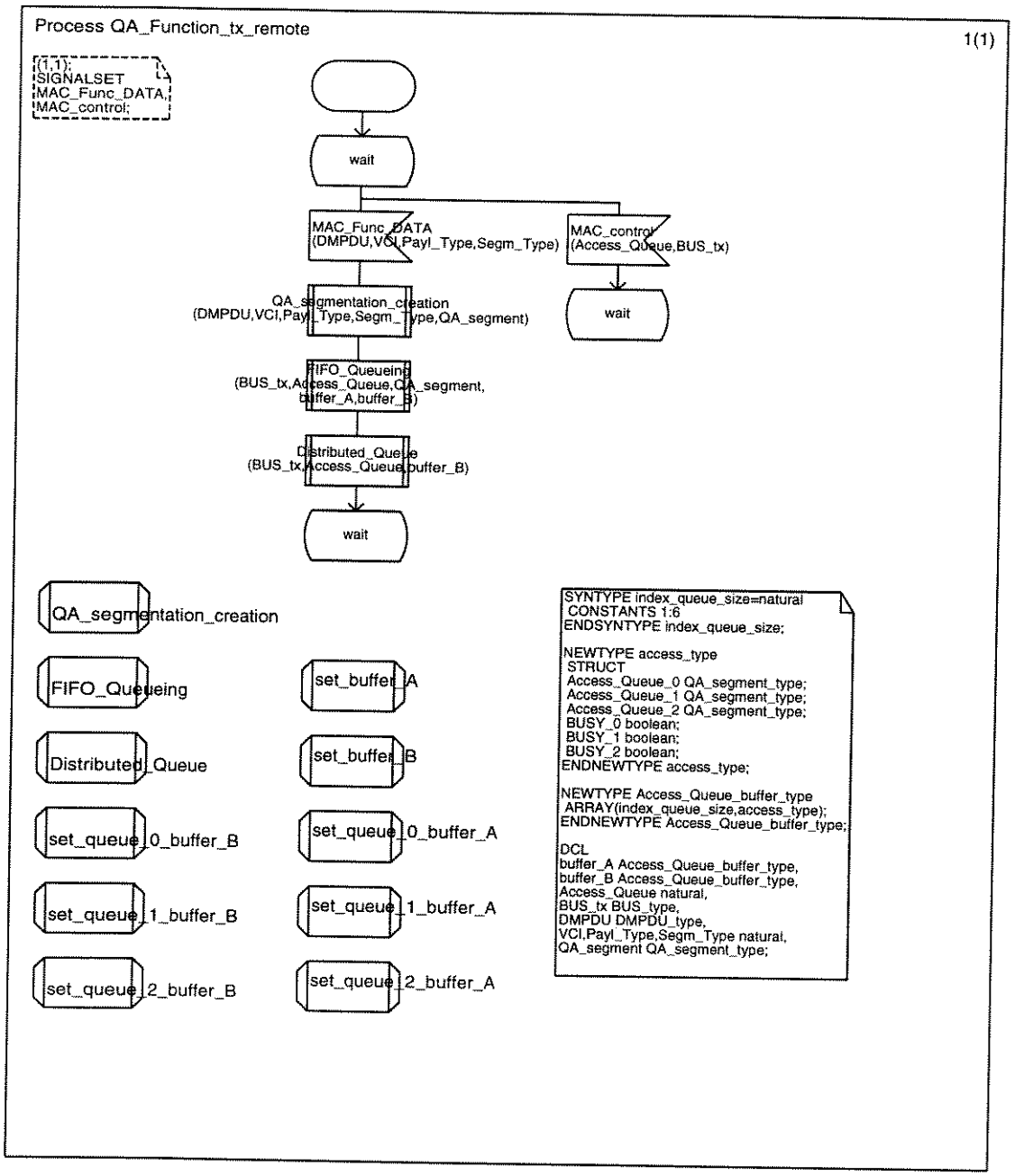

Figura B.63: "Processo QA_Function_tx_remote"

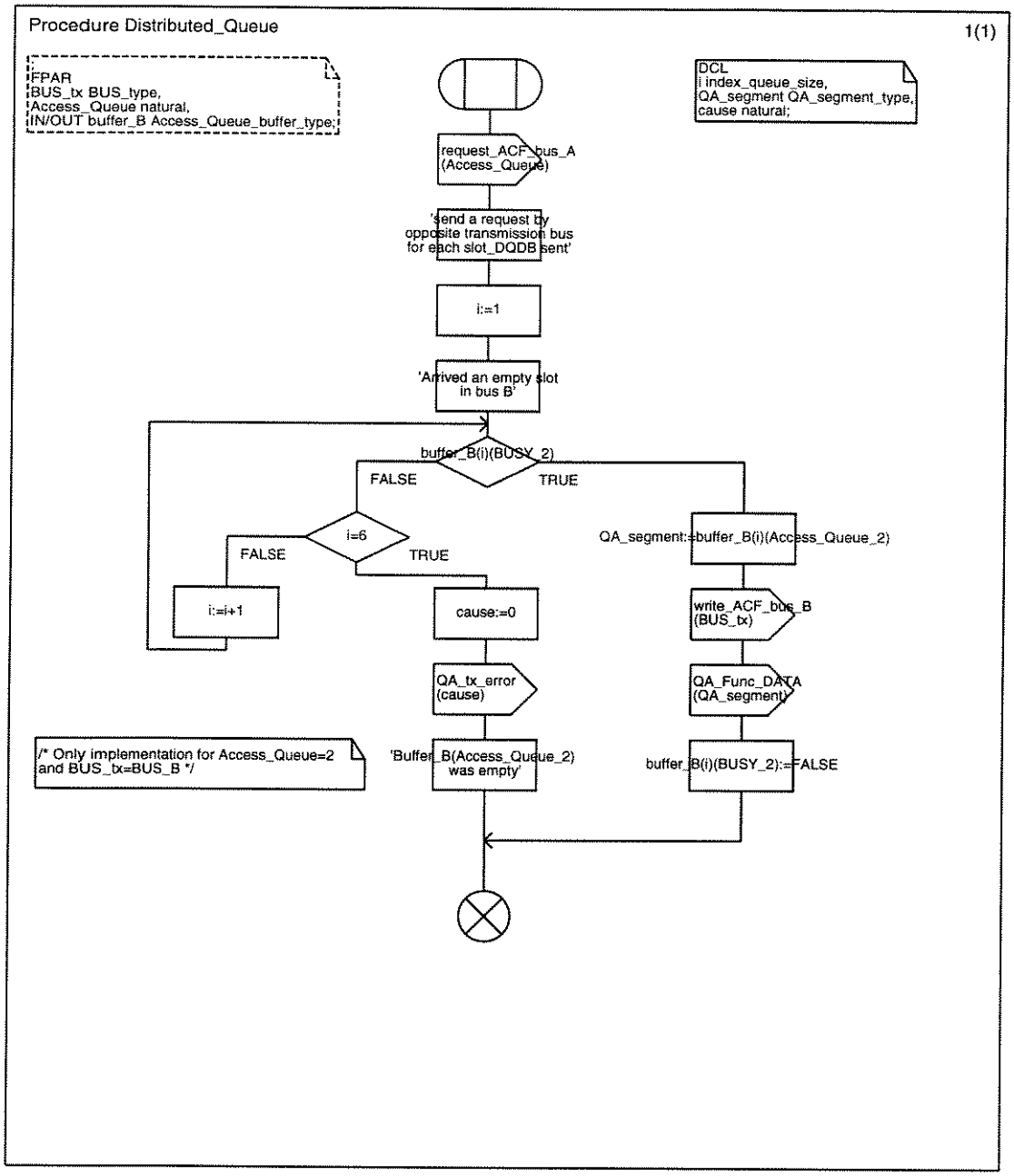

Figura B.64: "Procedimento Distributed_Queue"

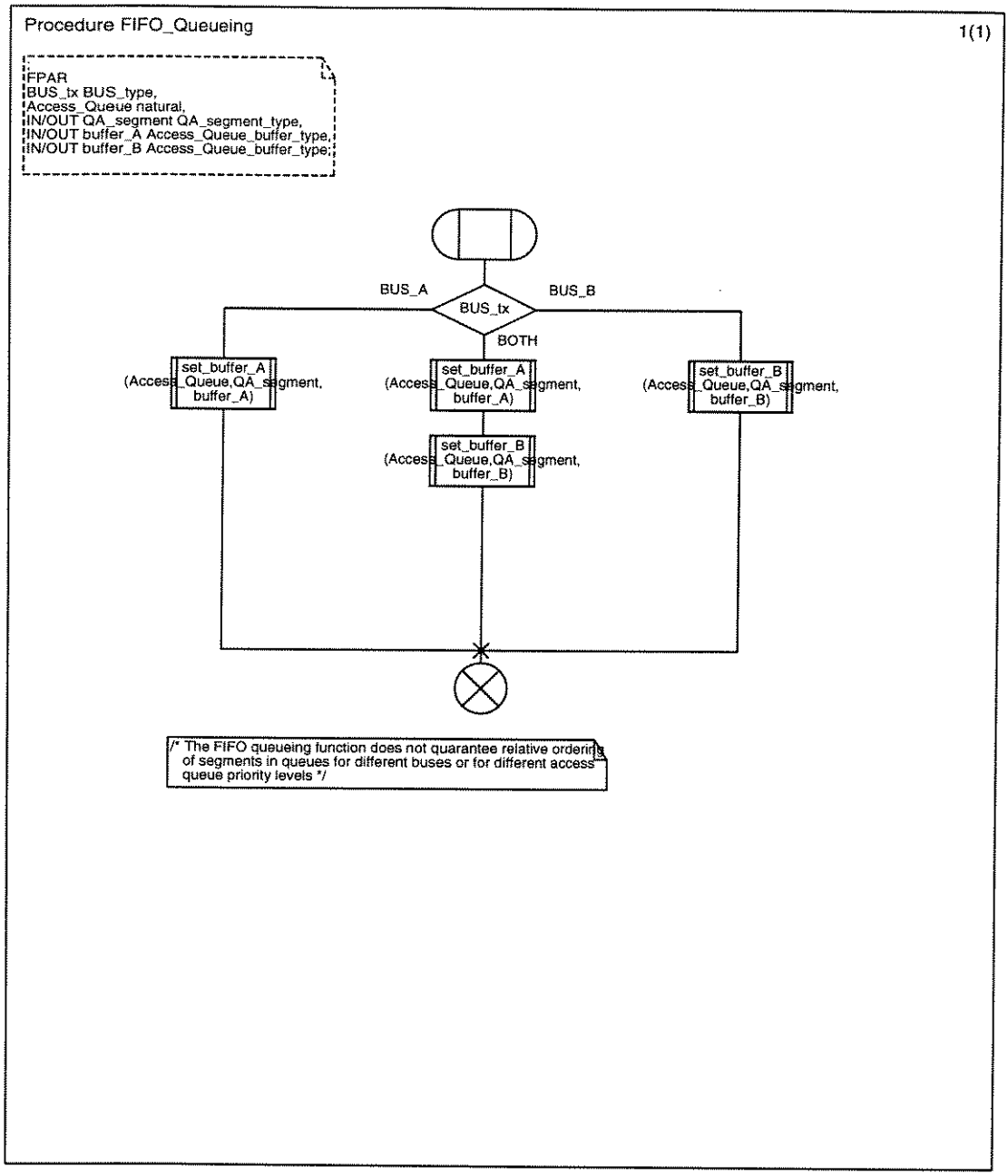

Figura B.65: "Procedimento FIFO_Queueing"

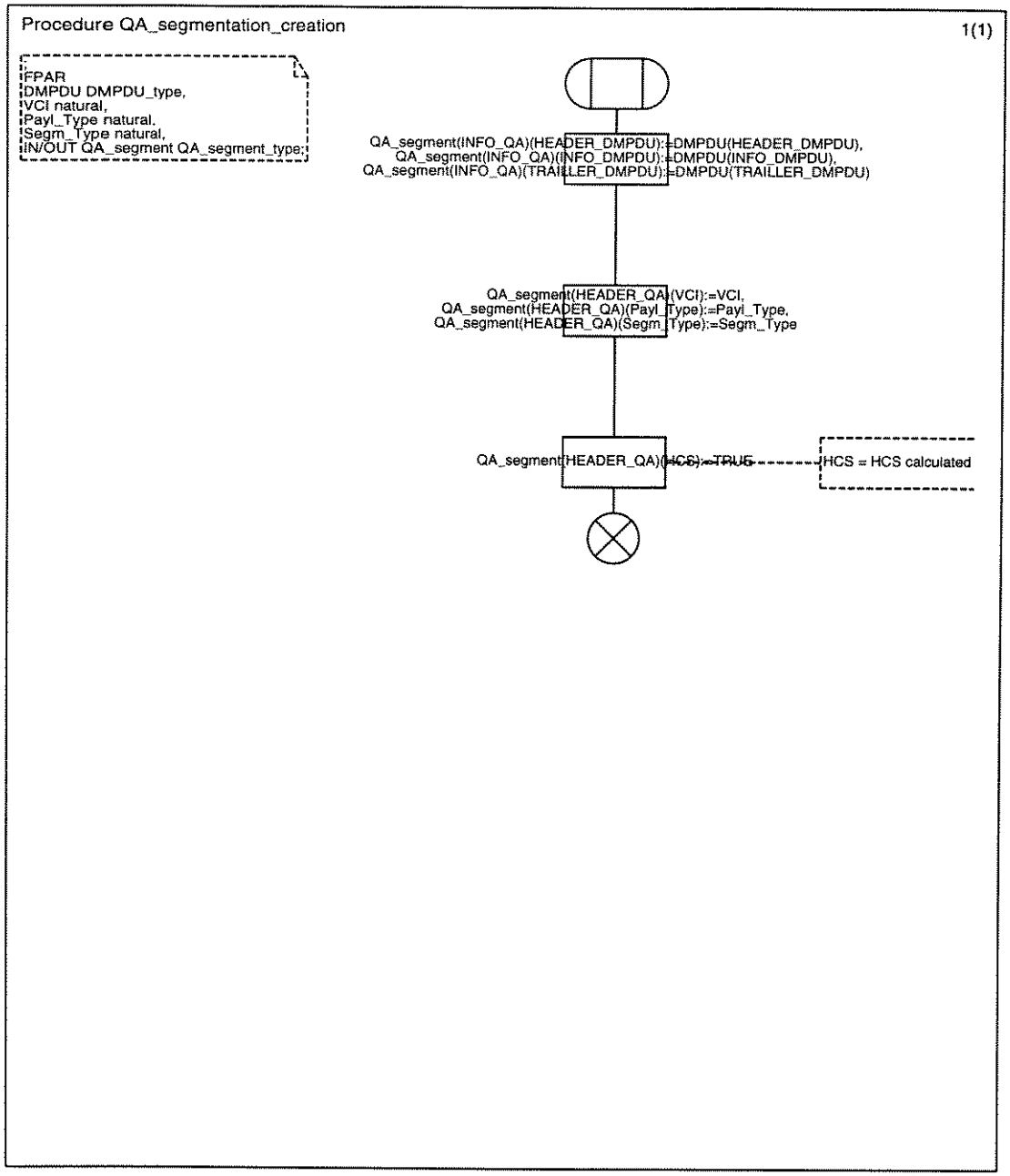

Figura B.66: "Procedimento QA_segmentation_creation"

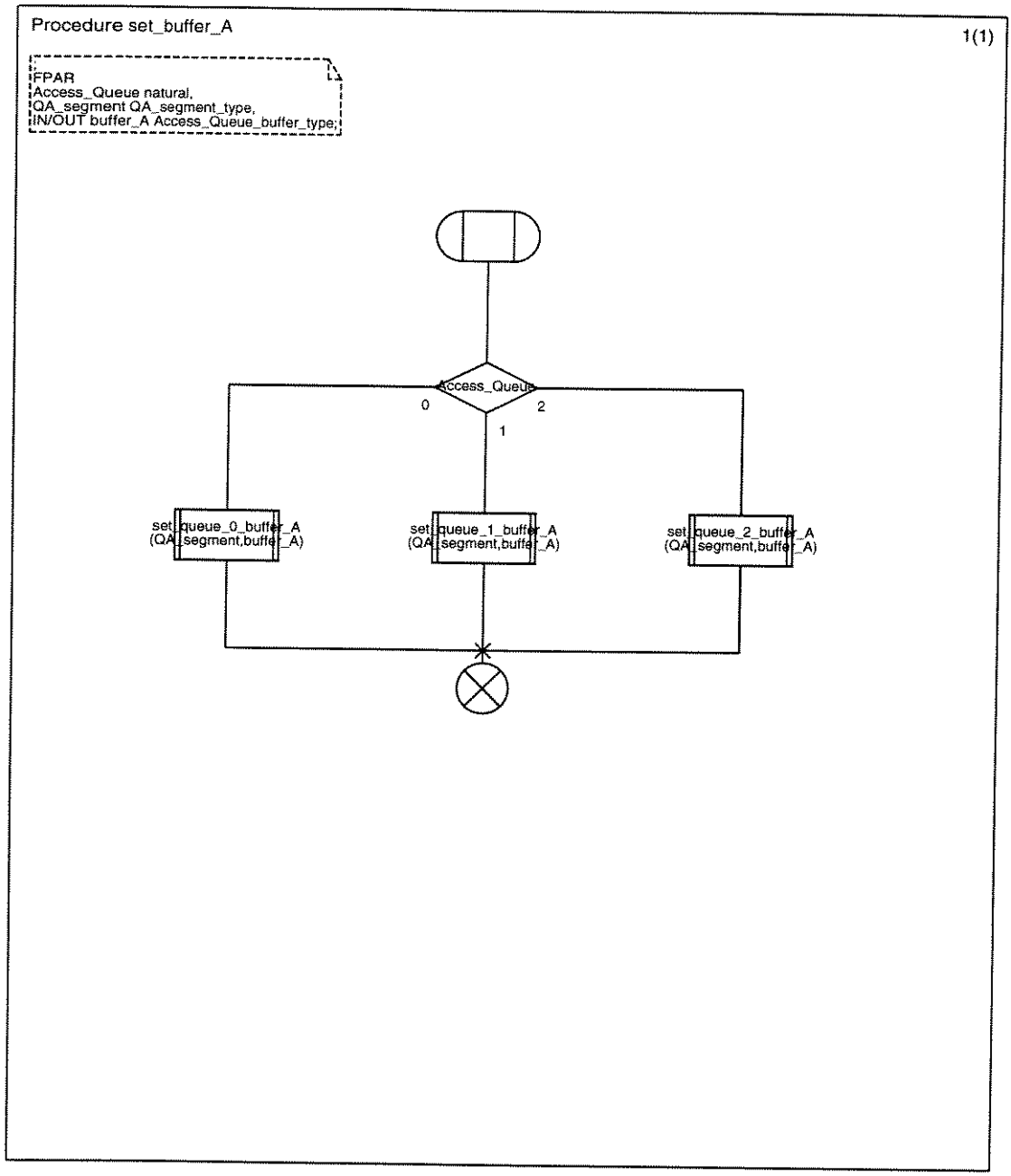

Figura B.67: "Procedimento set_buffer_A"

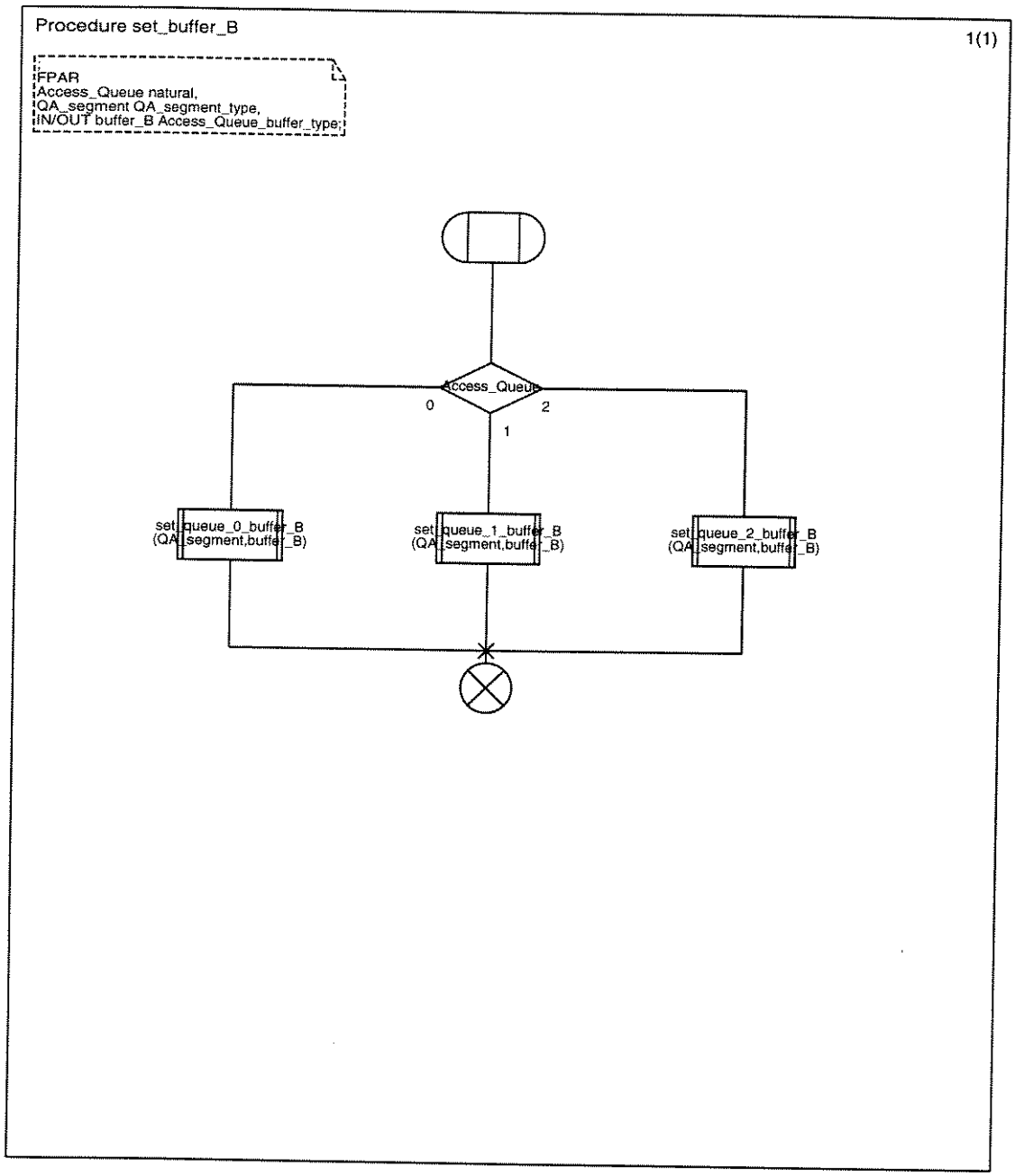

Figura B.68: "Procedimento set_buffer_B"

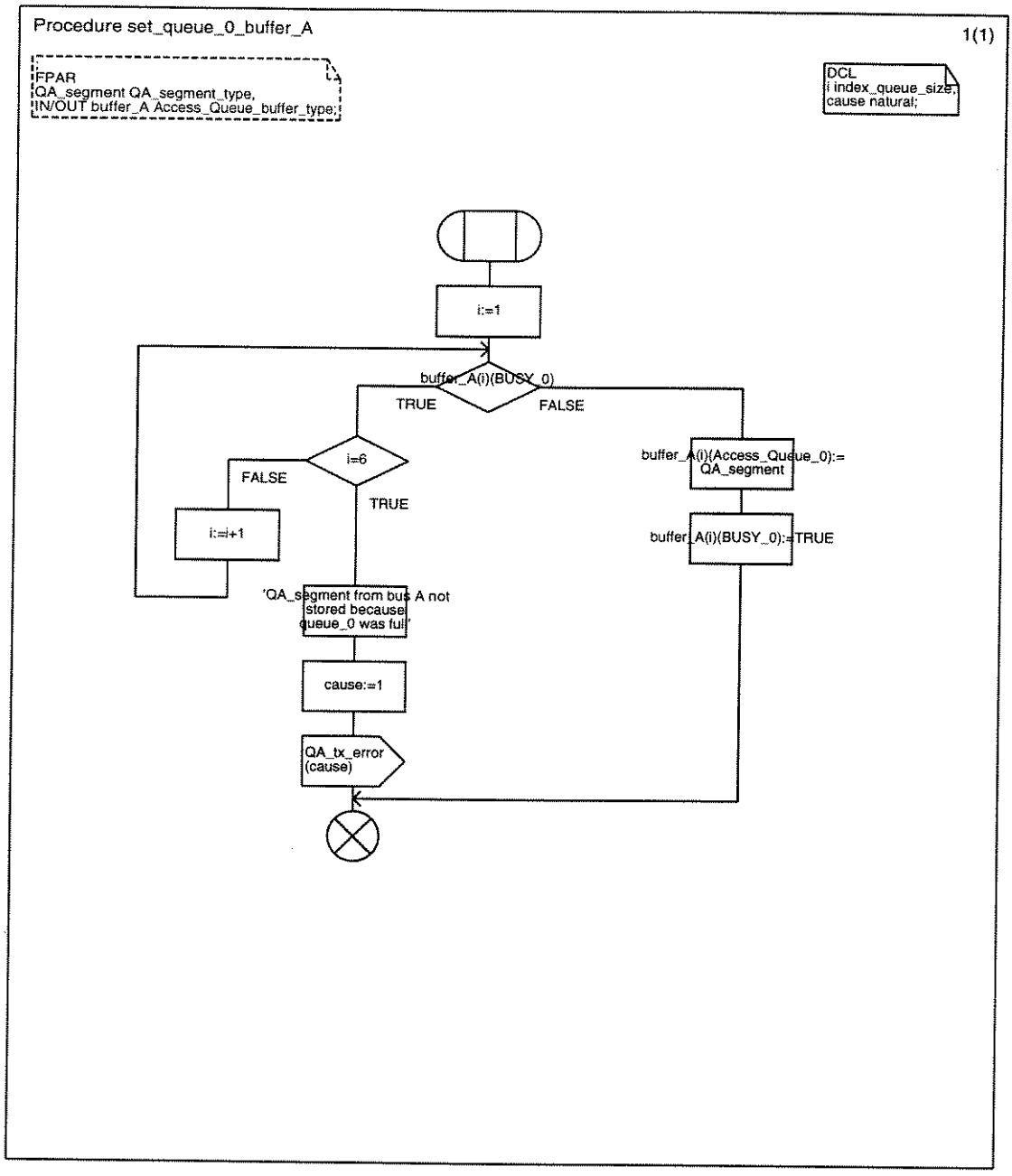

Figura B.69: "Procedimento set_queue_0_buffer_A"

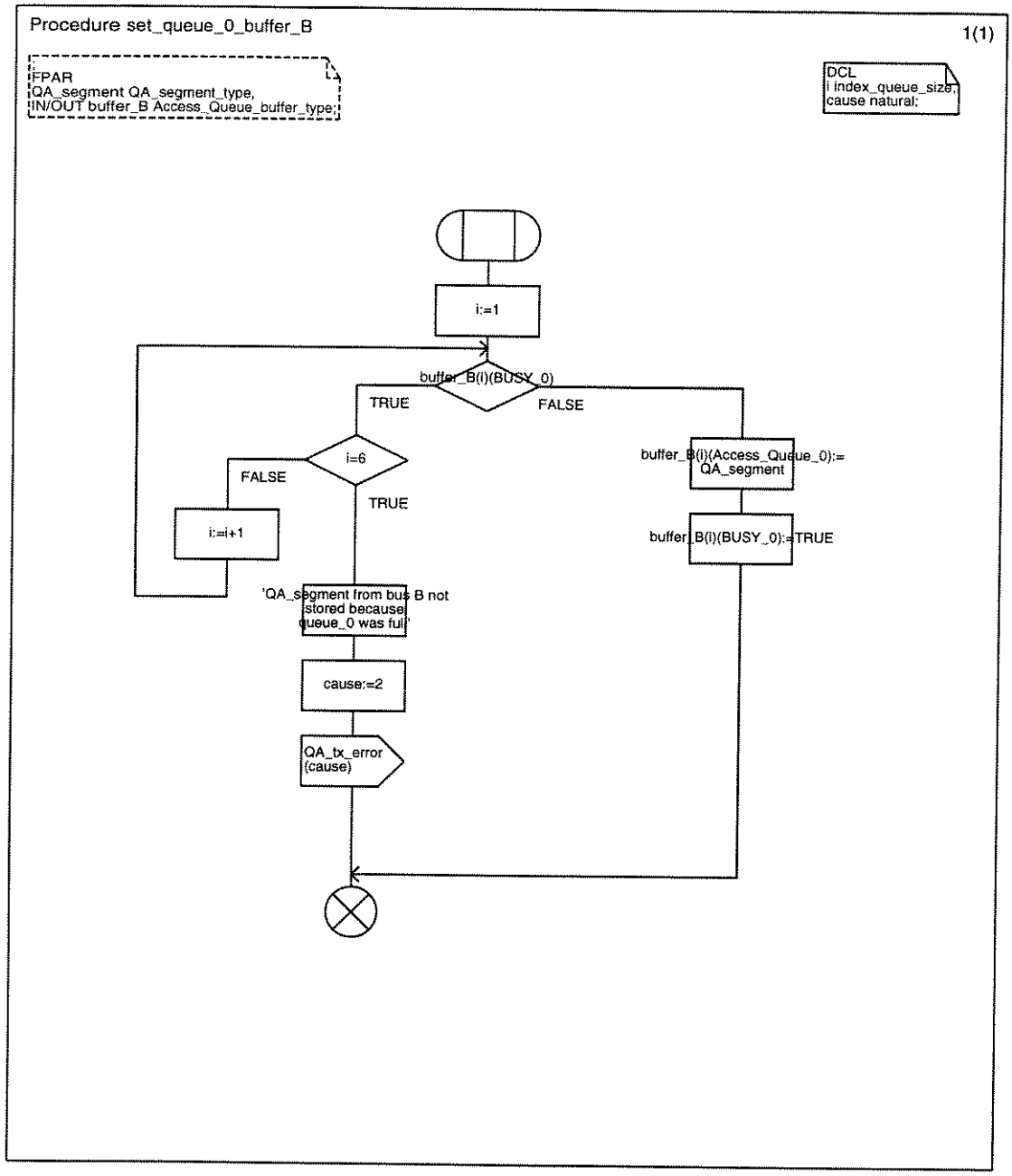

Figura B.70: "Procedimento set_queue_0_buffer_B"

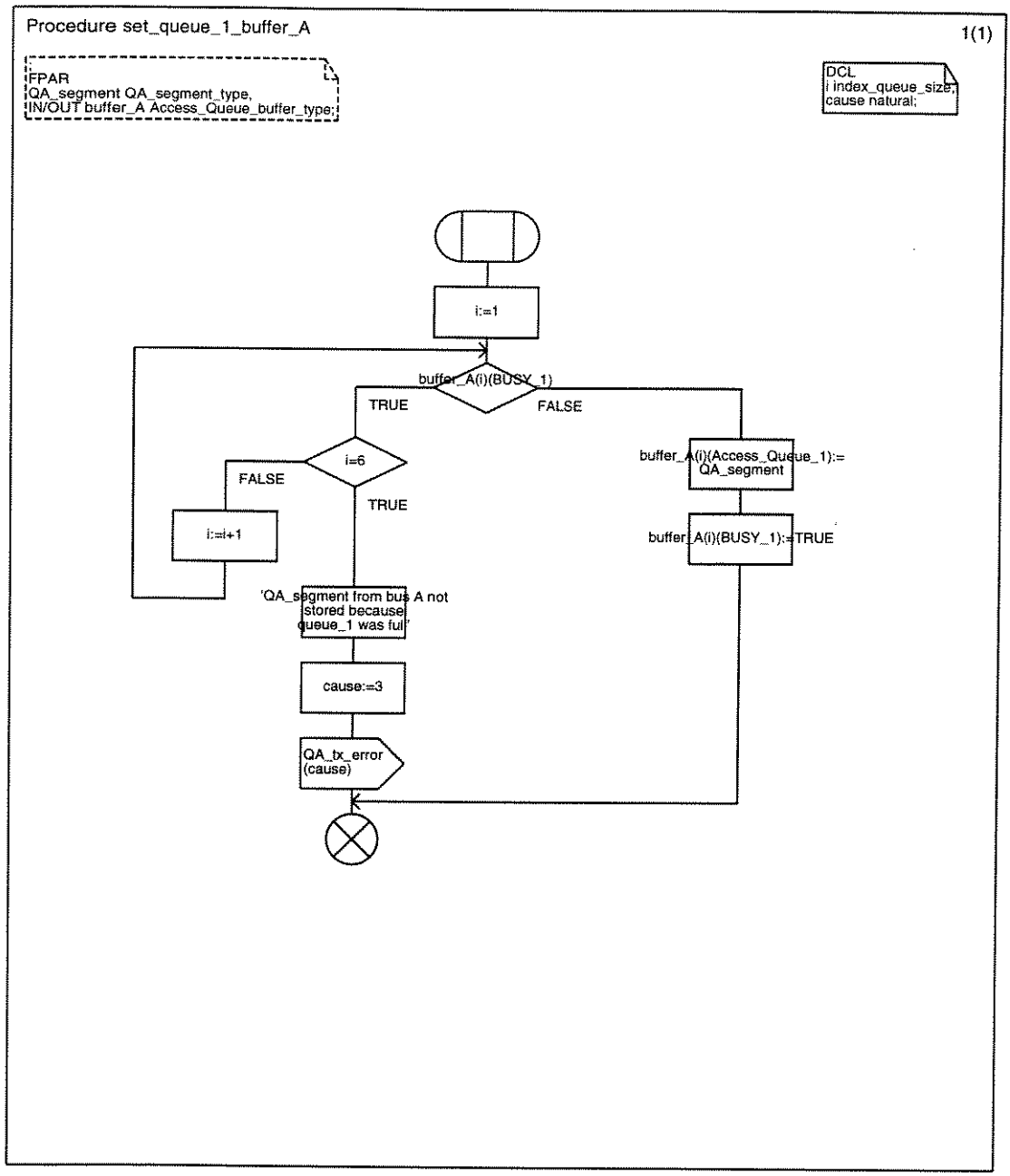

Figura B.71: "Procedimento set_queue_1_buffer_A"

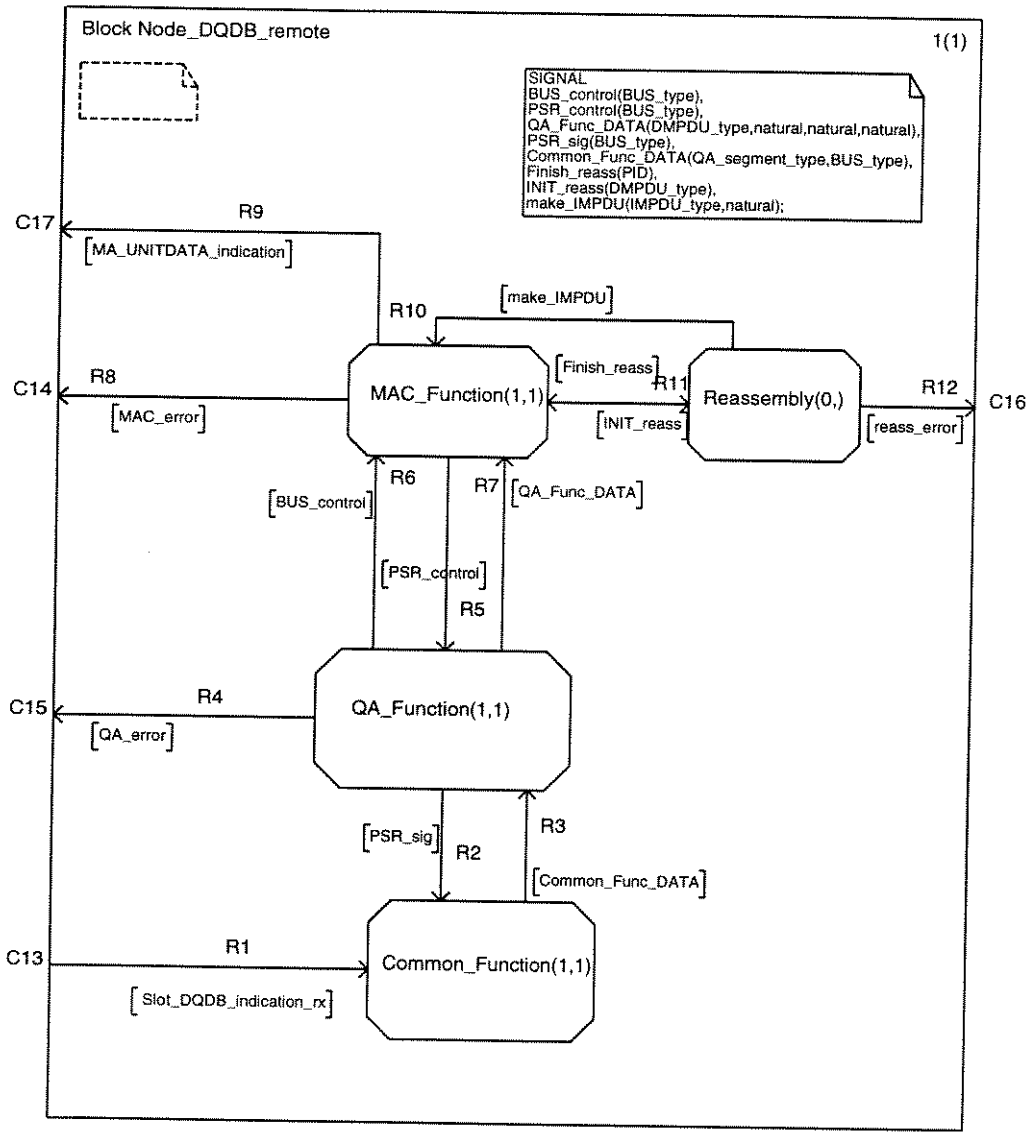

Figura B.72: "Bloco Node_DQDB_remote"

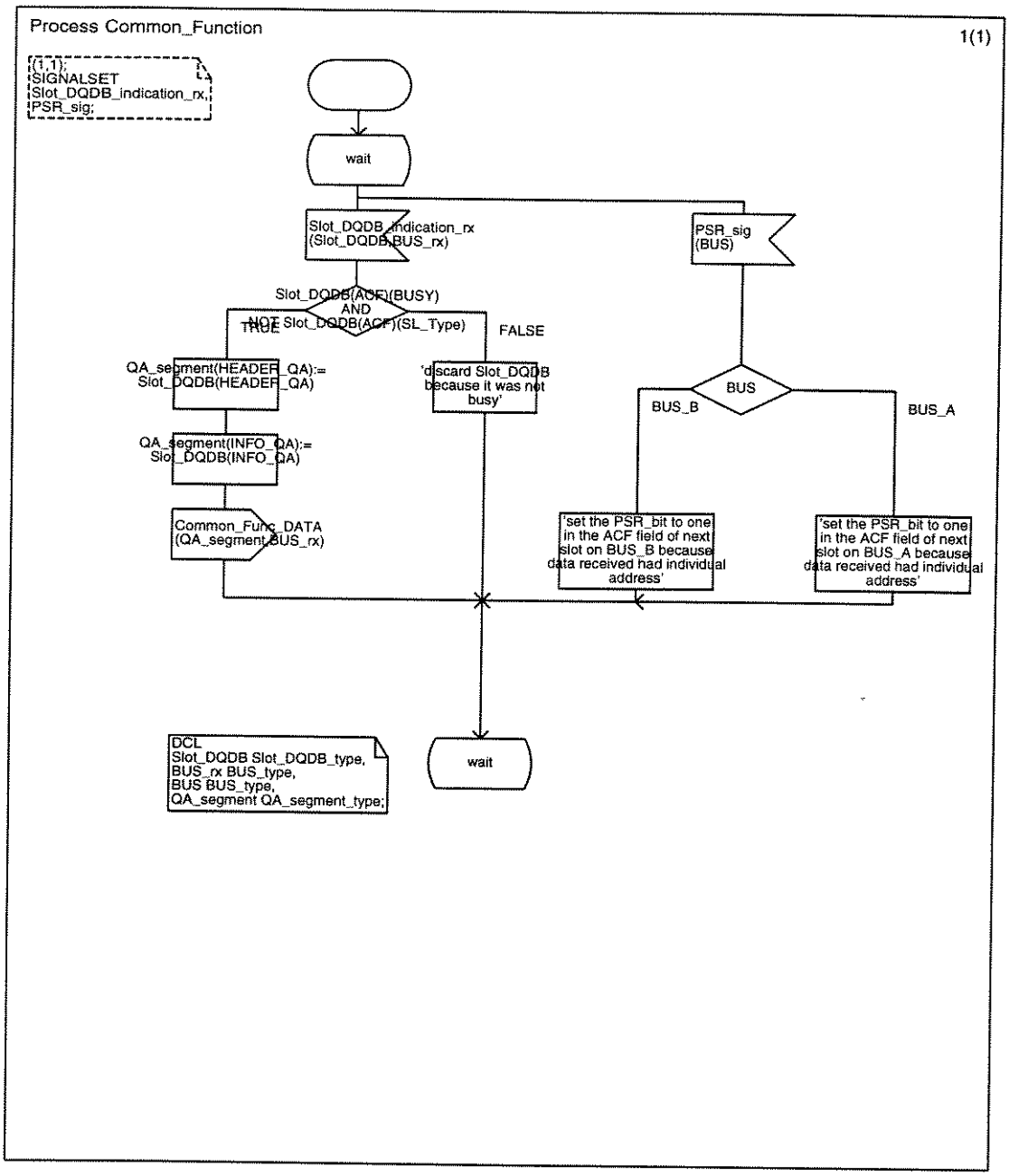

Figura B.73: "Processo Common_Function"

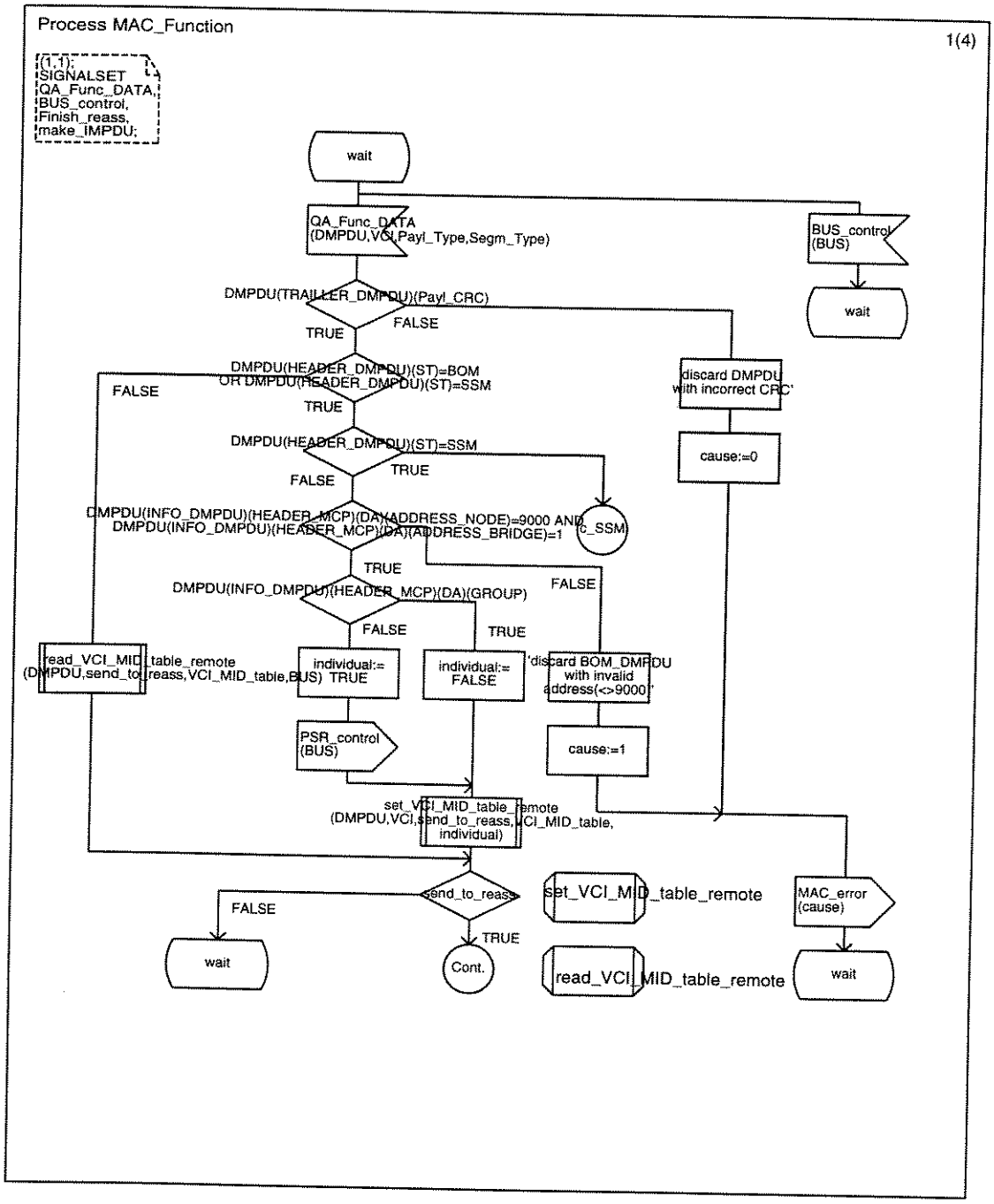

Figura B.74: "Processo MAC_Function"

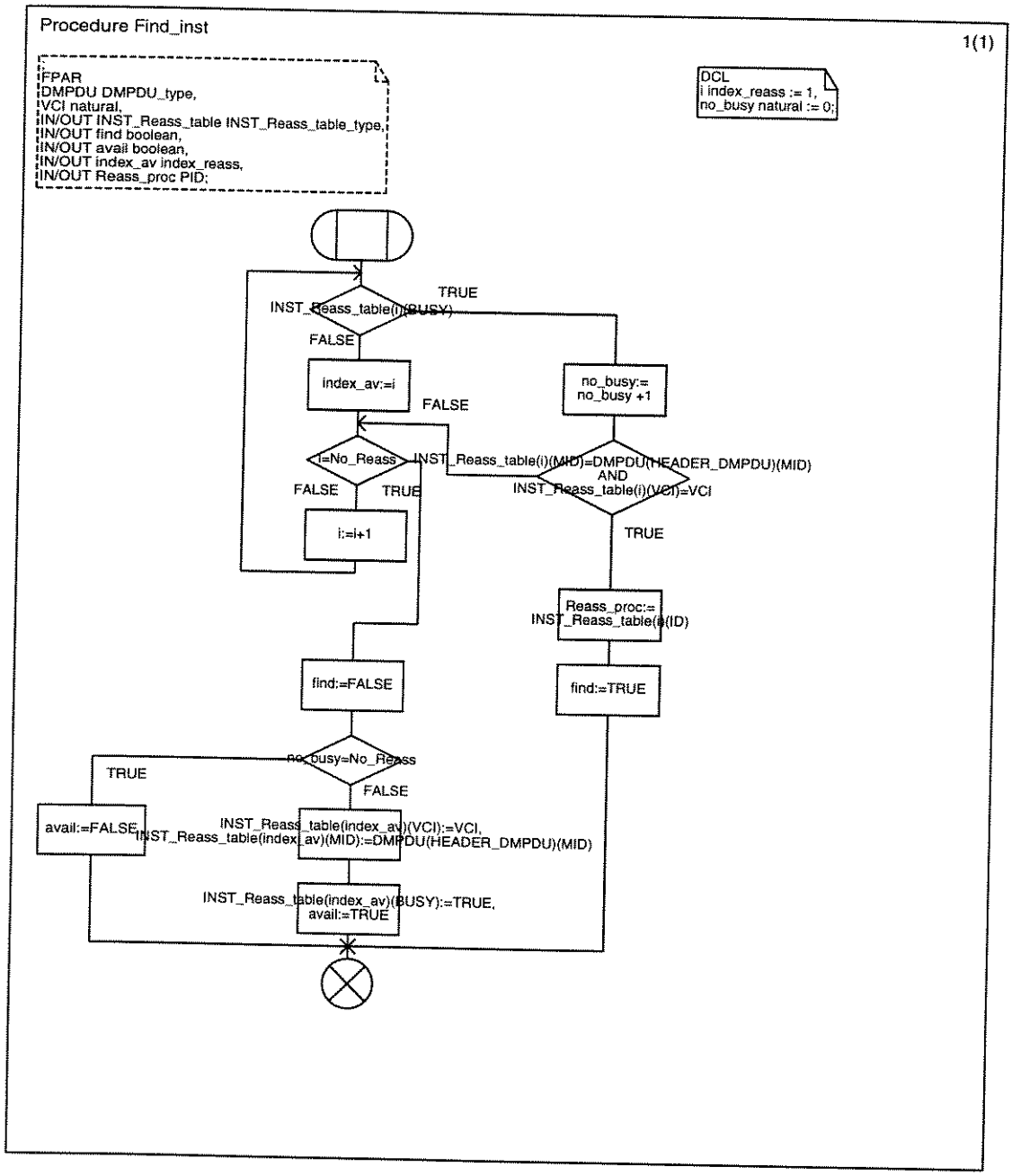

Figura B.75: "Procedimento Find_inst"

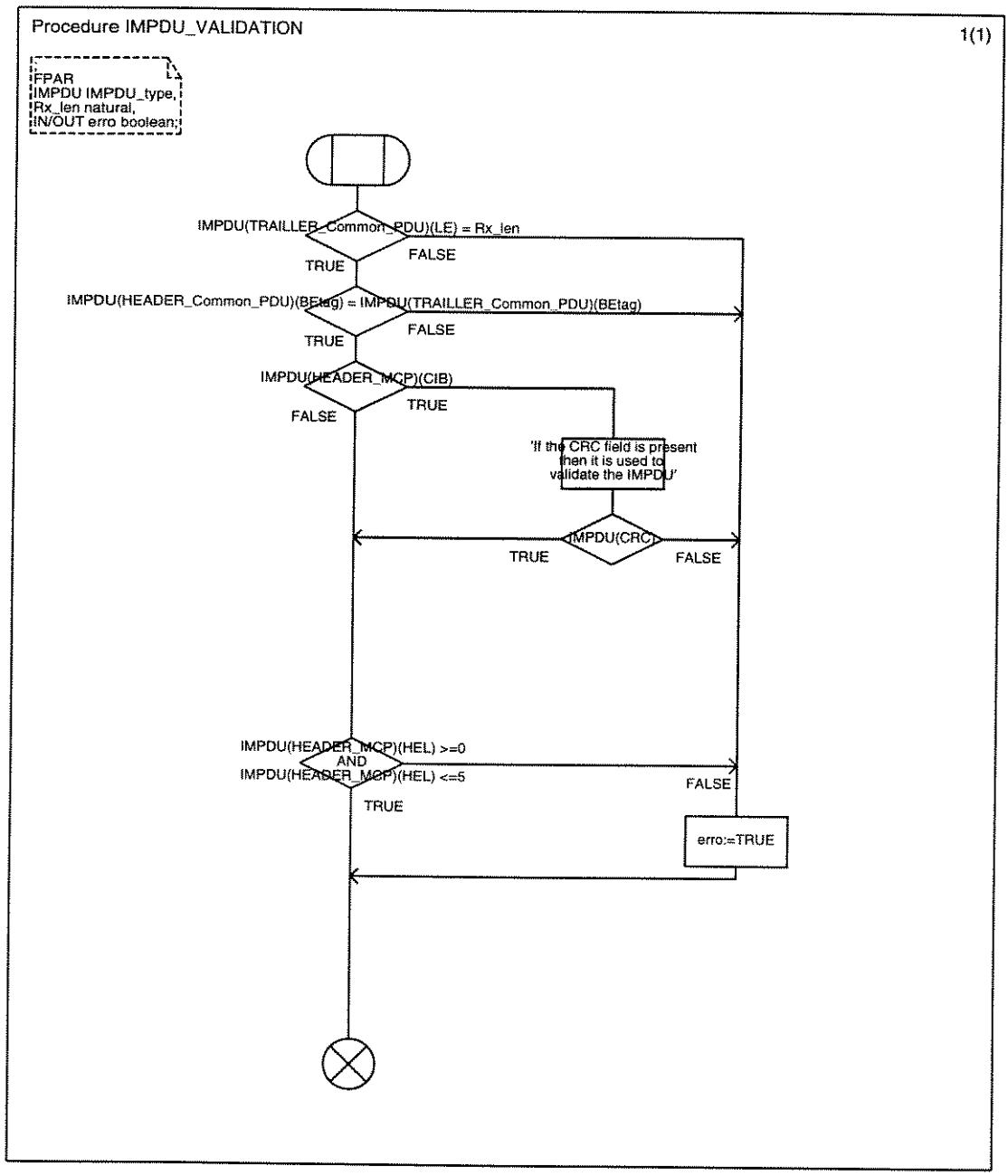

Figura B.76: "Procedimento IMPDU_VALIDATION"

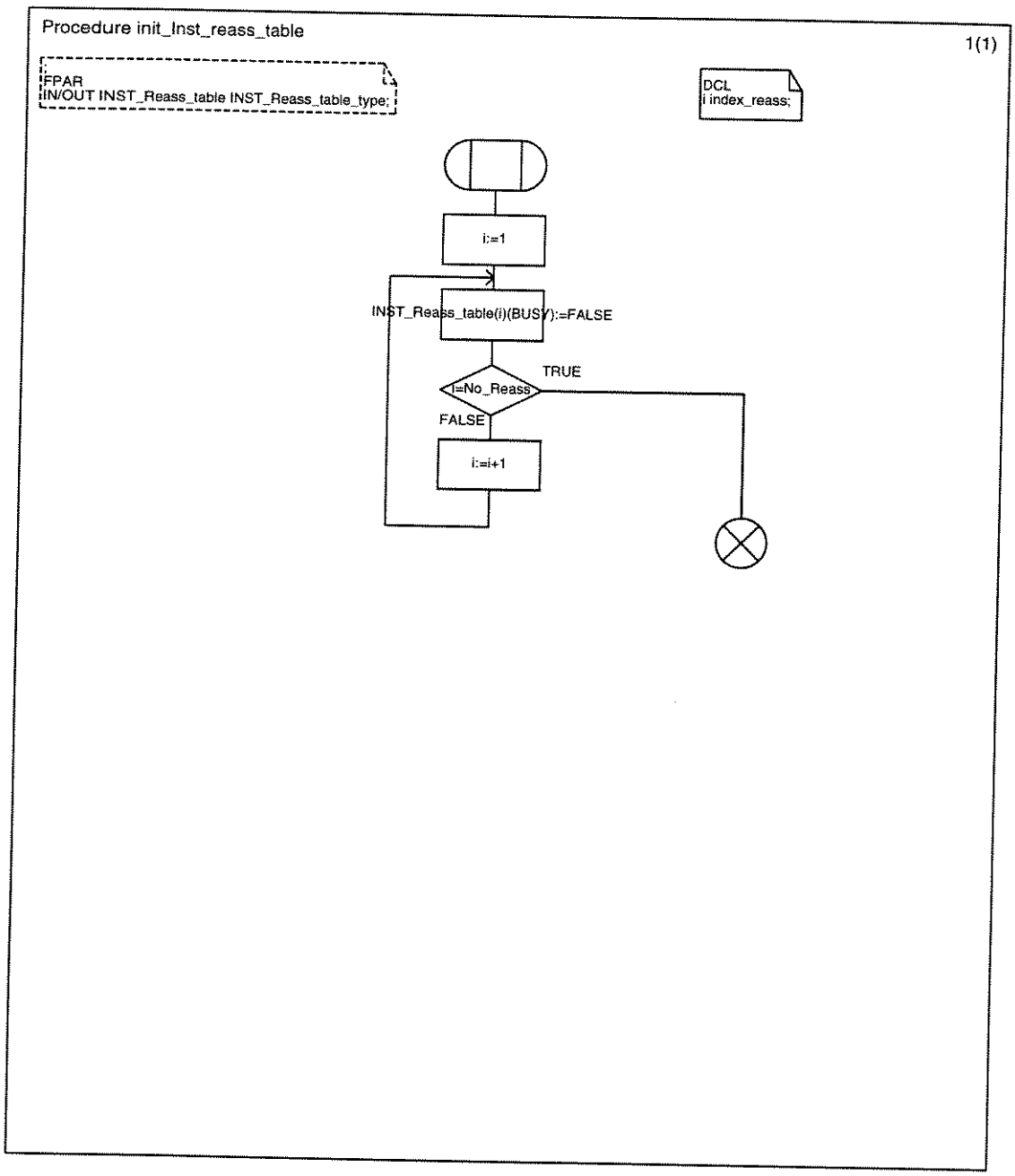

Figura B.77: "Processo init_Inst_reass_table"

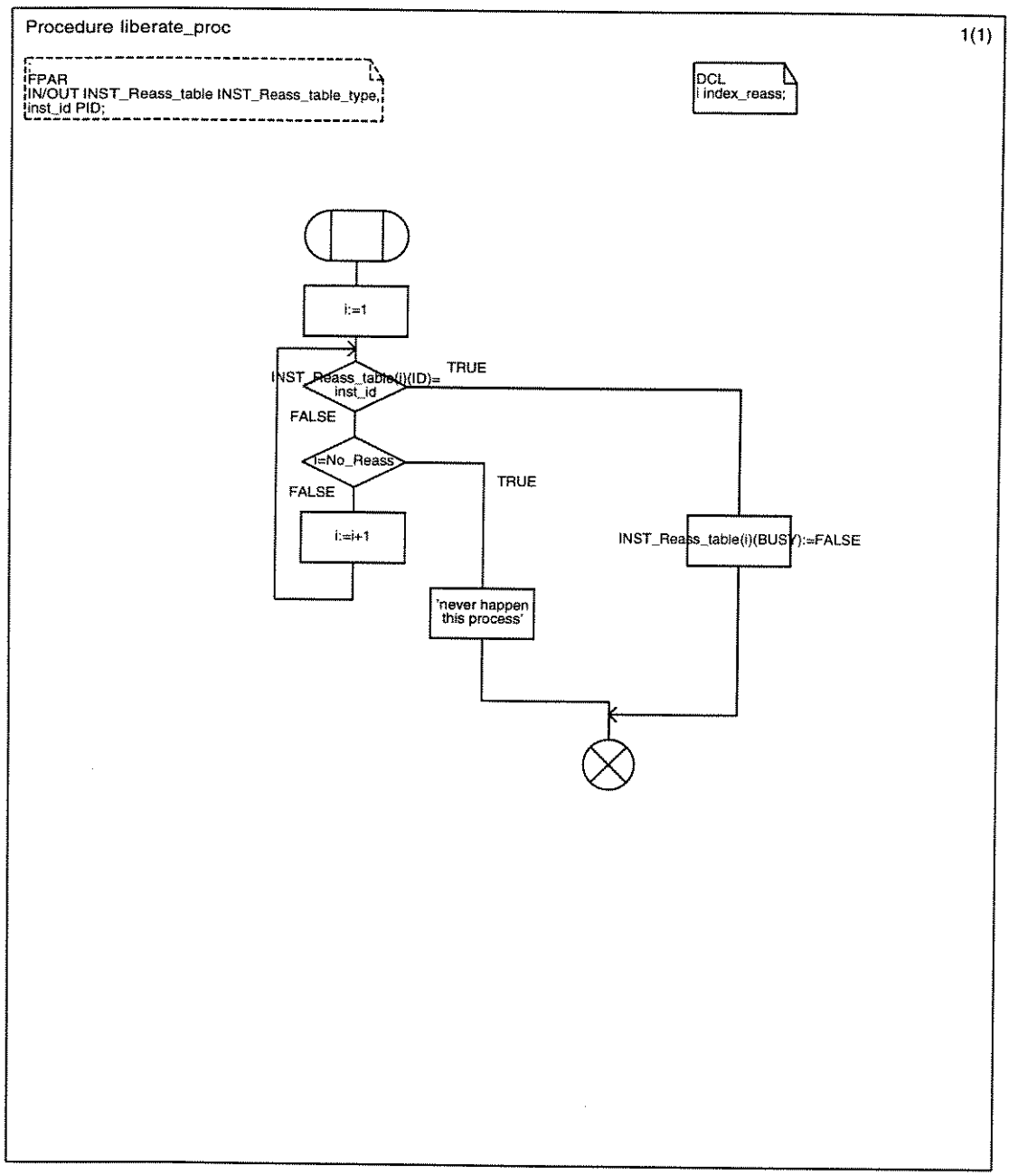

Figura B.78: "Procedimento liberate_proc"

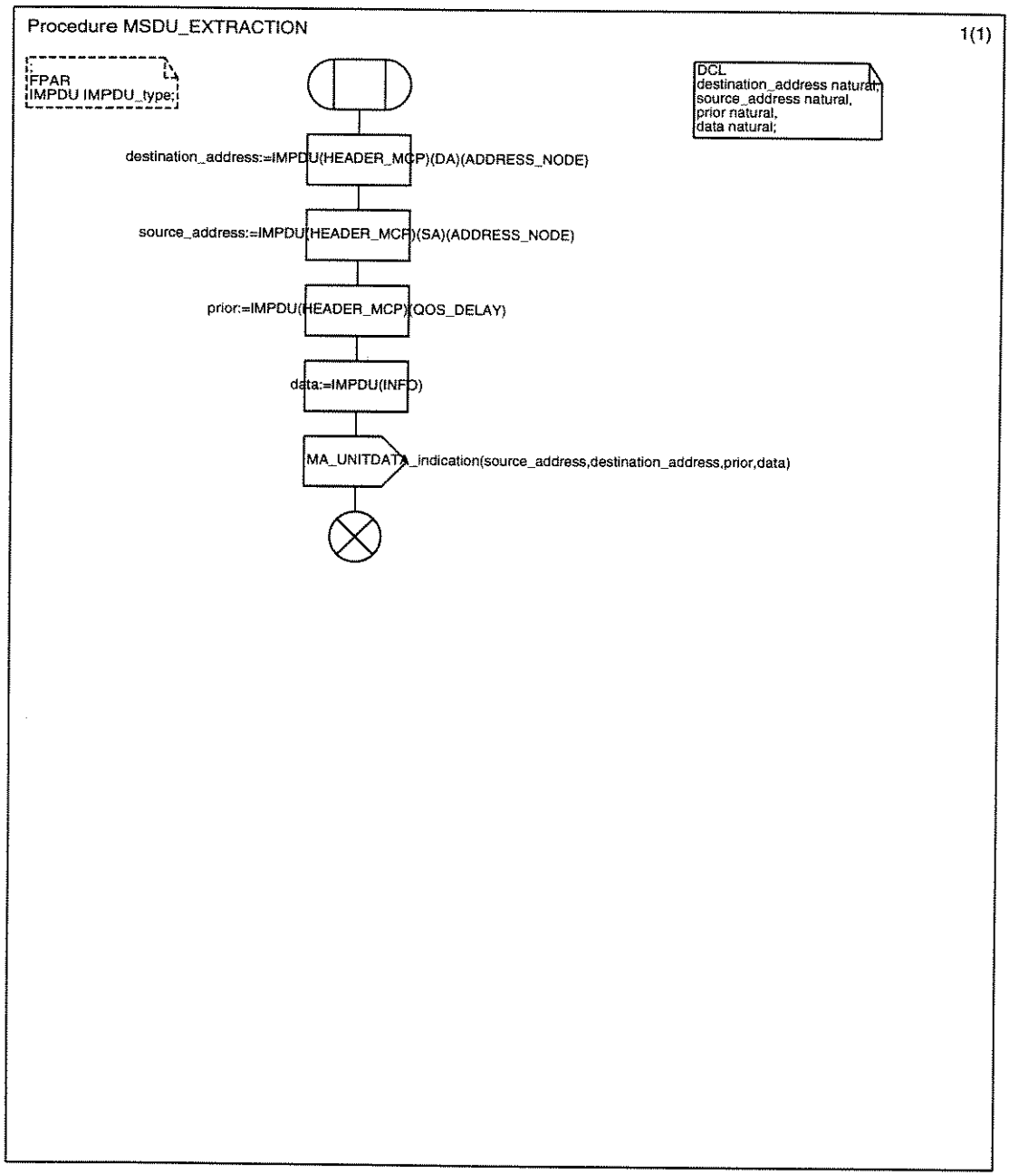

Figura B.79: "Procedimento MSDU_EXTRACTION"

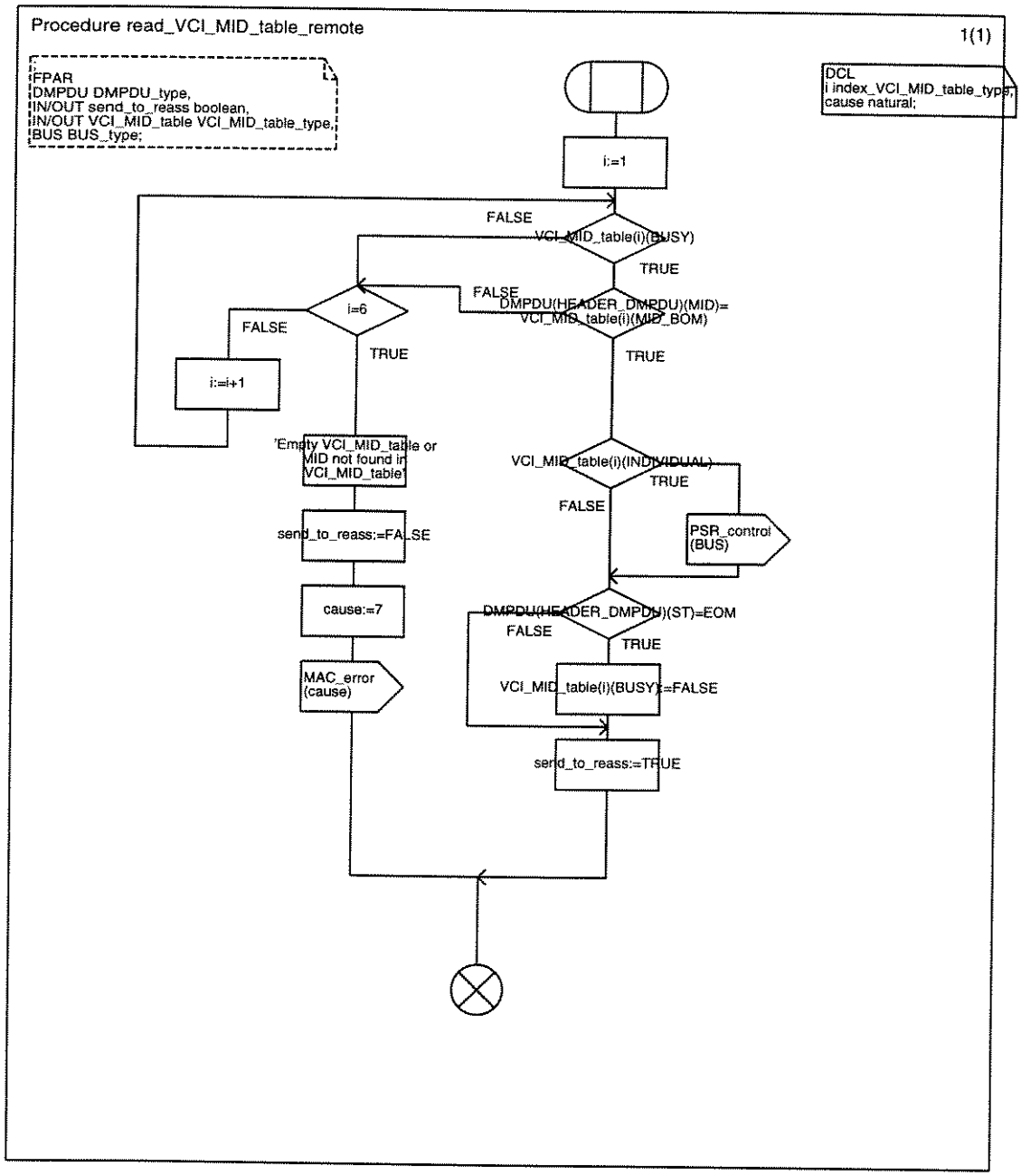

Figura B.80: "Procedimento read_VCI_MID_table_remote"

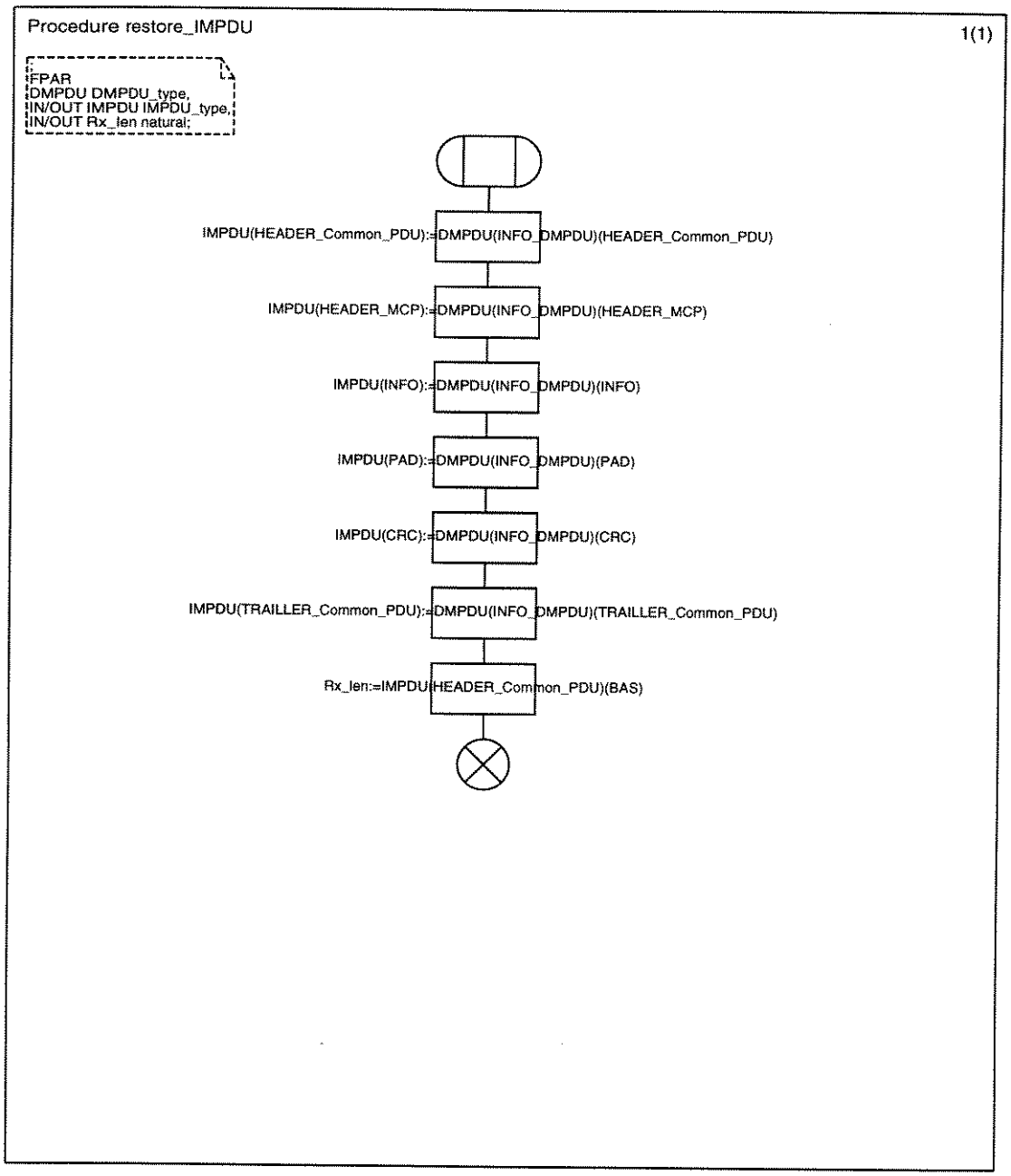

Figura B.81: "Procedimento restore_IMPDU"

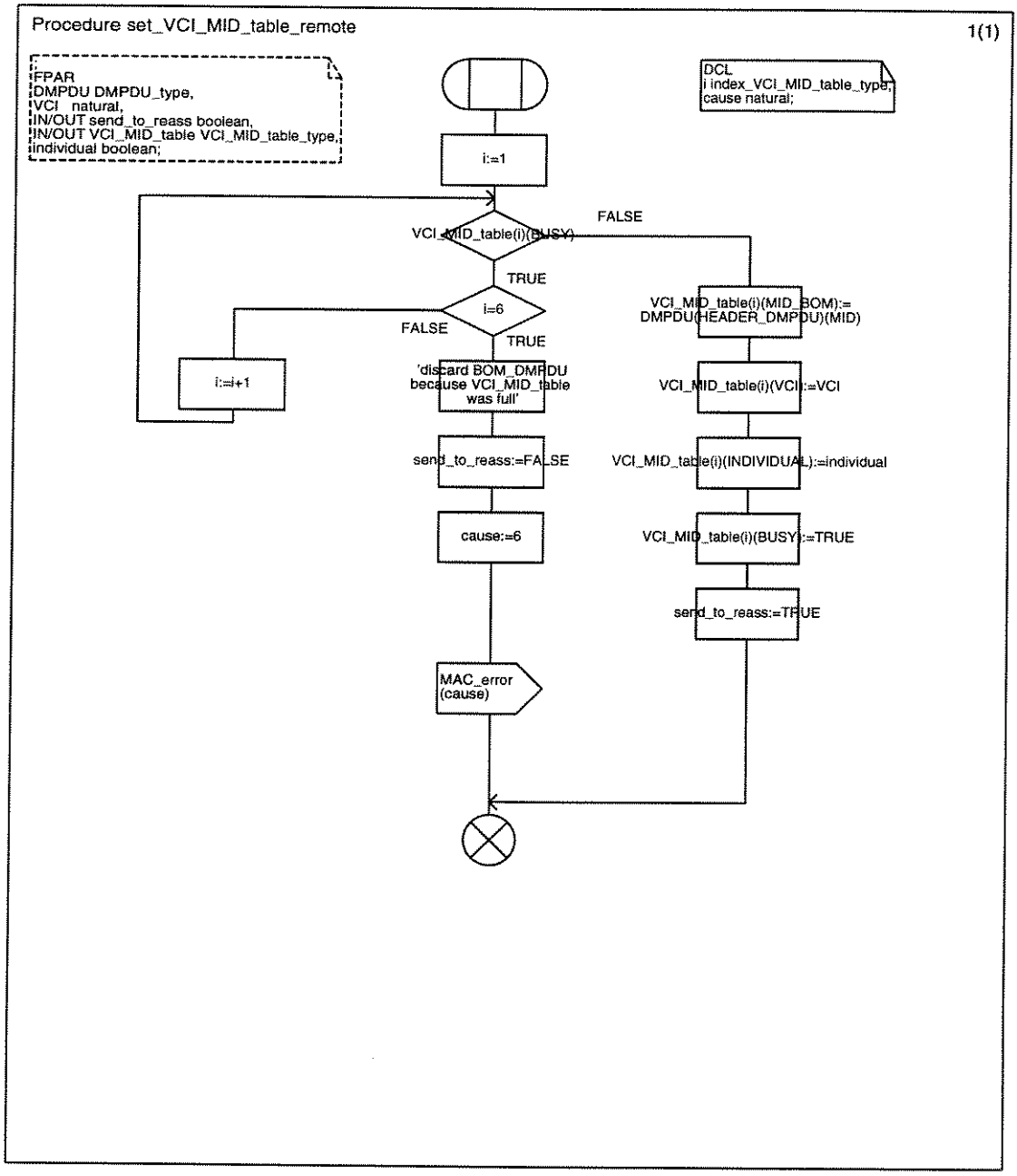

Figura B.82: "Procedimento set_VCI_MID_table_remote"

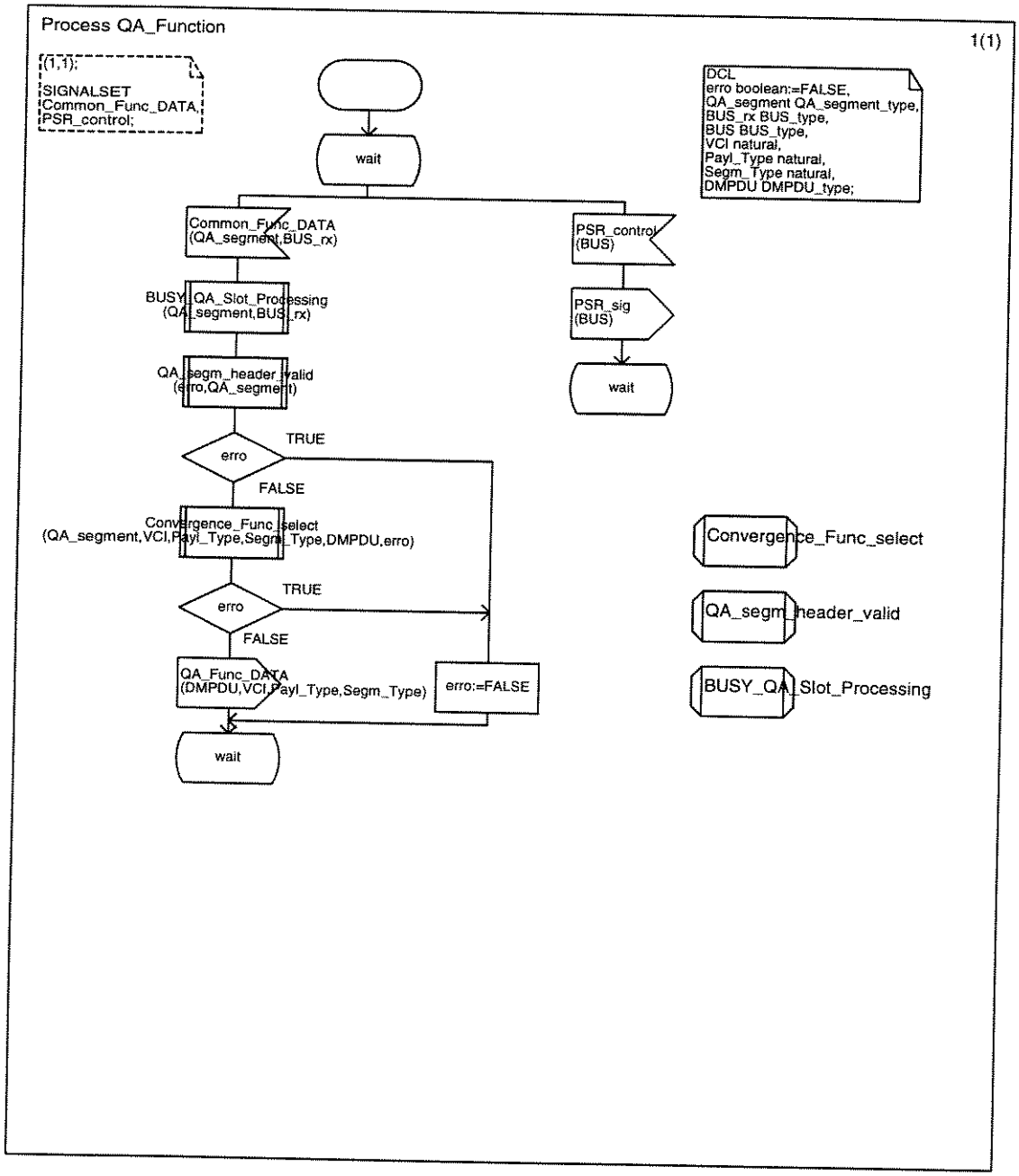

Figura B.83: "Processo QA_Function"

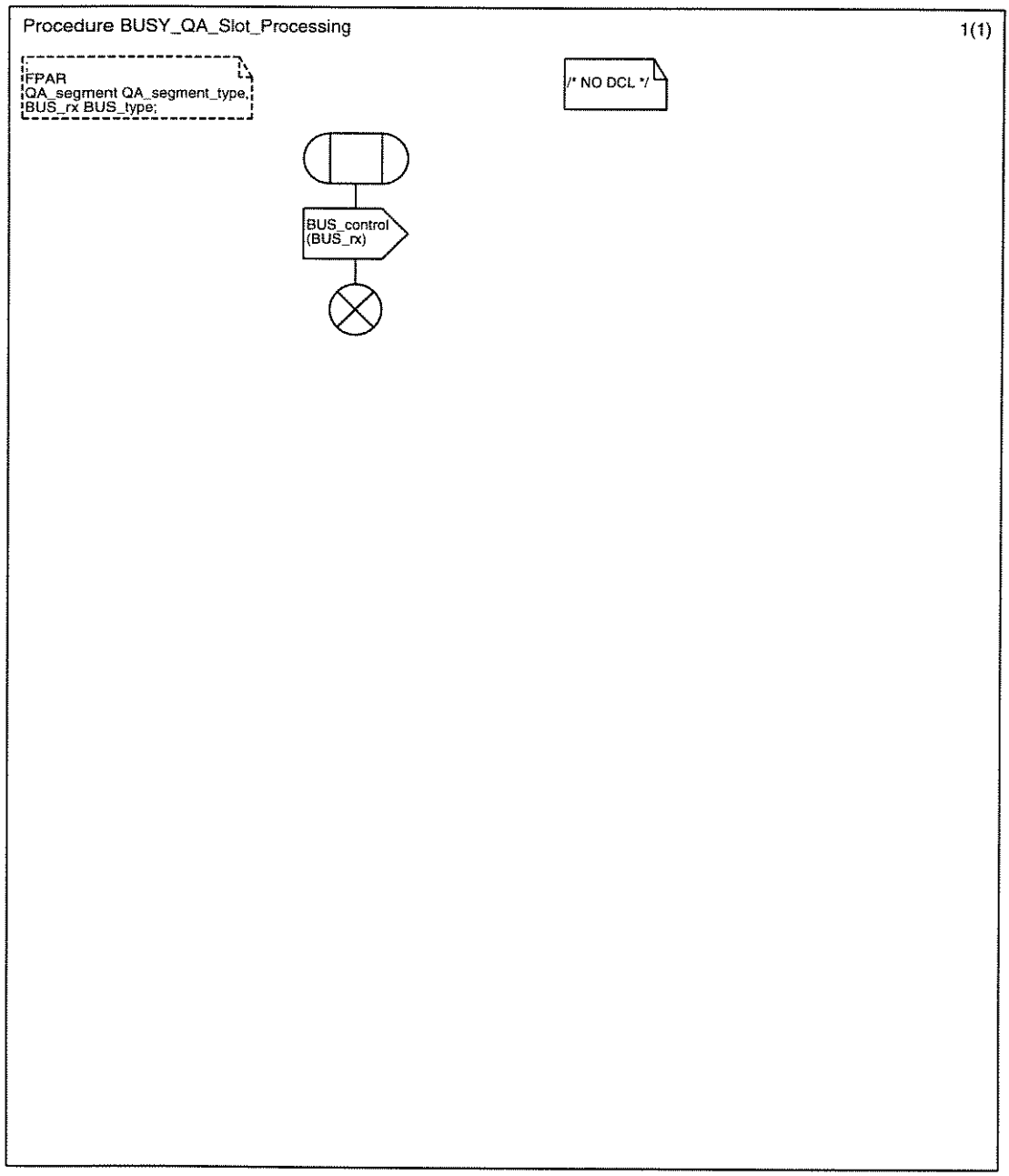

Figura B.84: "Procedimento BUSY_QA_Slot_Processing"

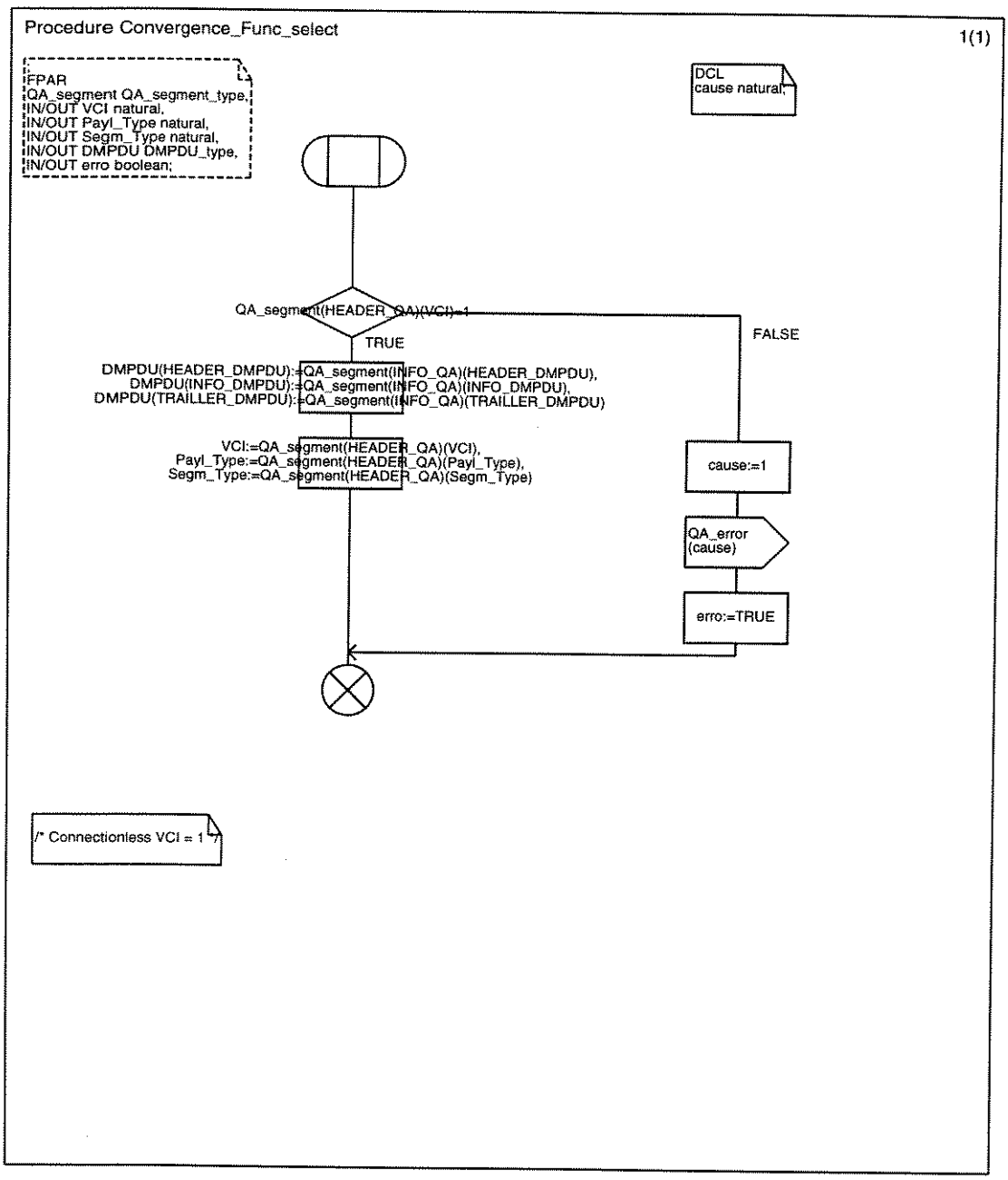

Figura B.85: "Procedimento Convergence_Func_select"

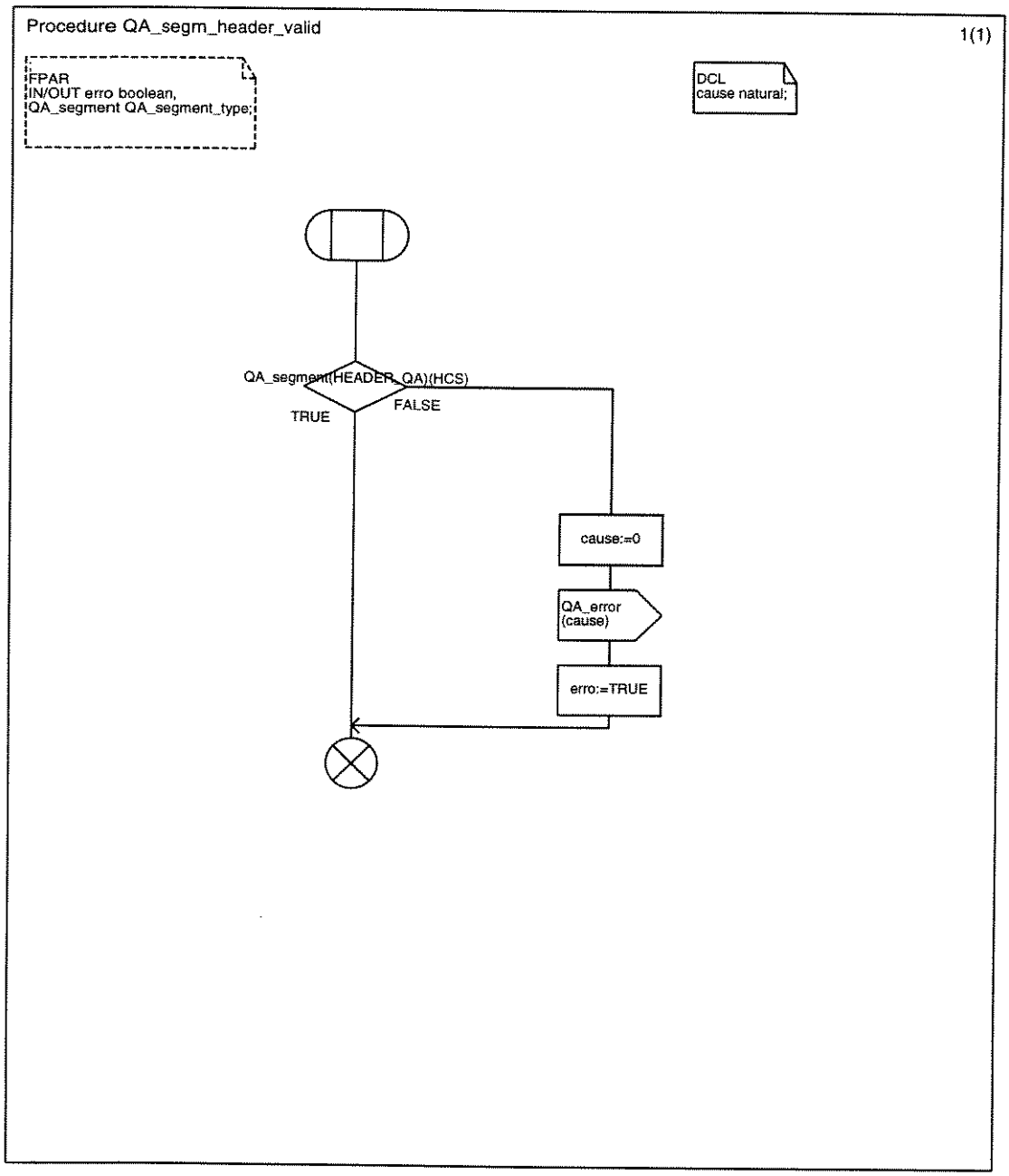

Figura B.86: "Procedimento QA_segm_header_valid"

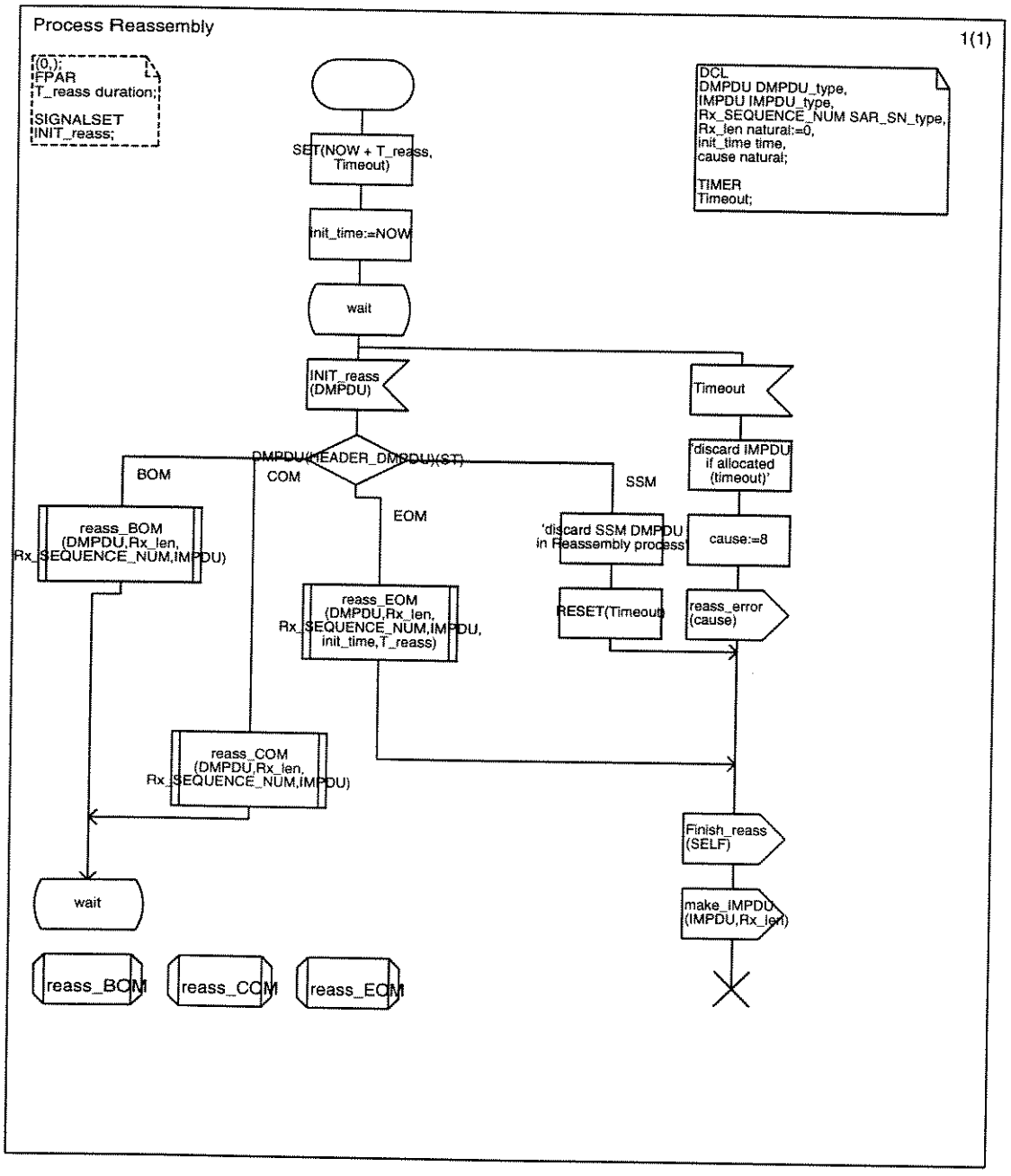

Figura B.87: "Processo Reassembly"

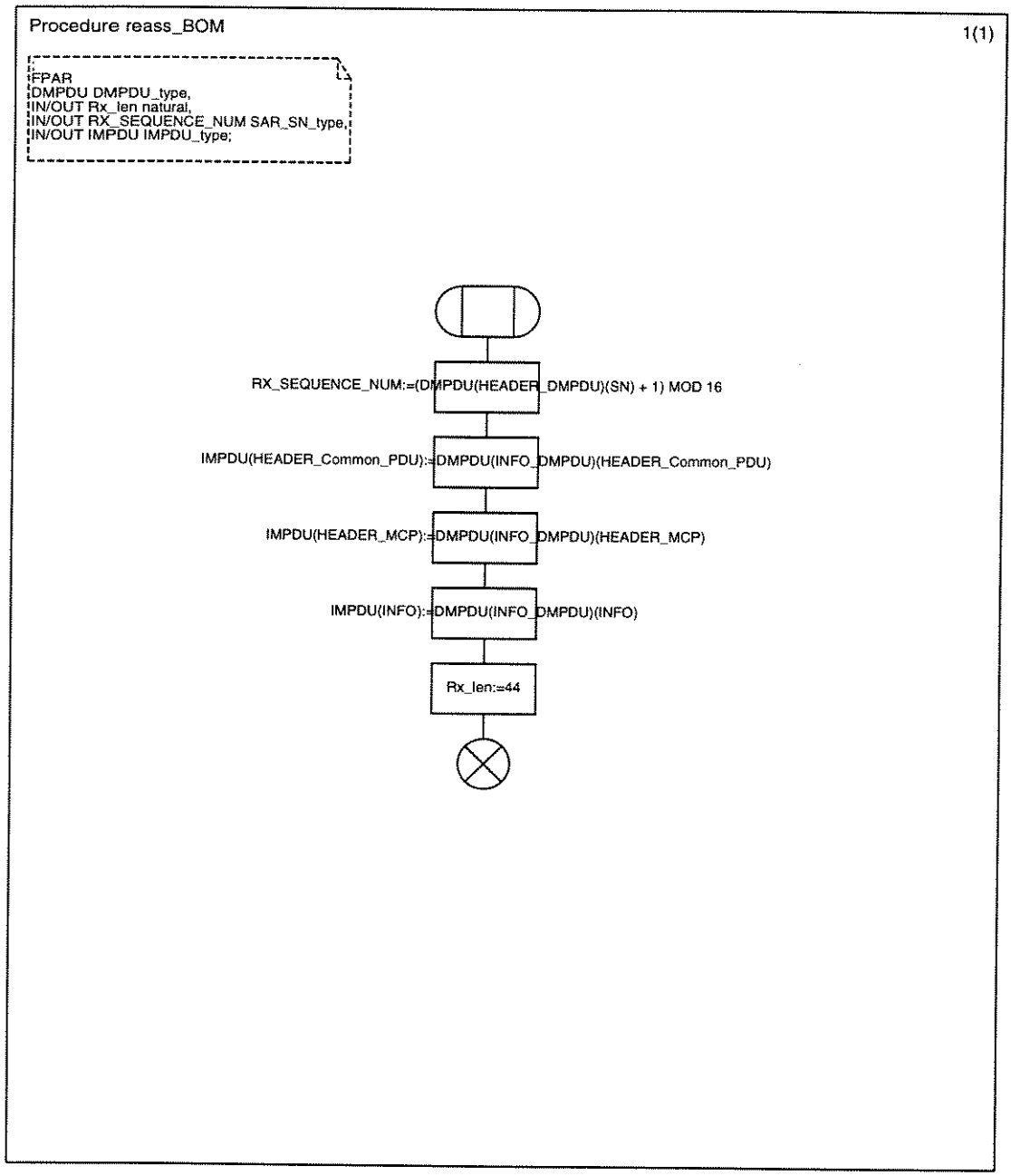

Figura B.88: "Procedimento reass_BOM"

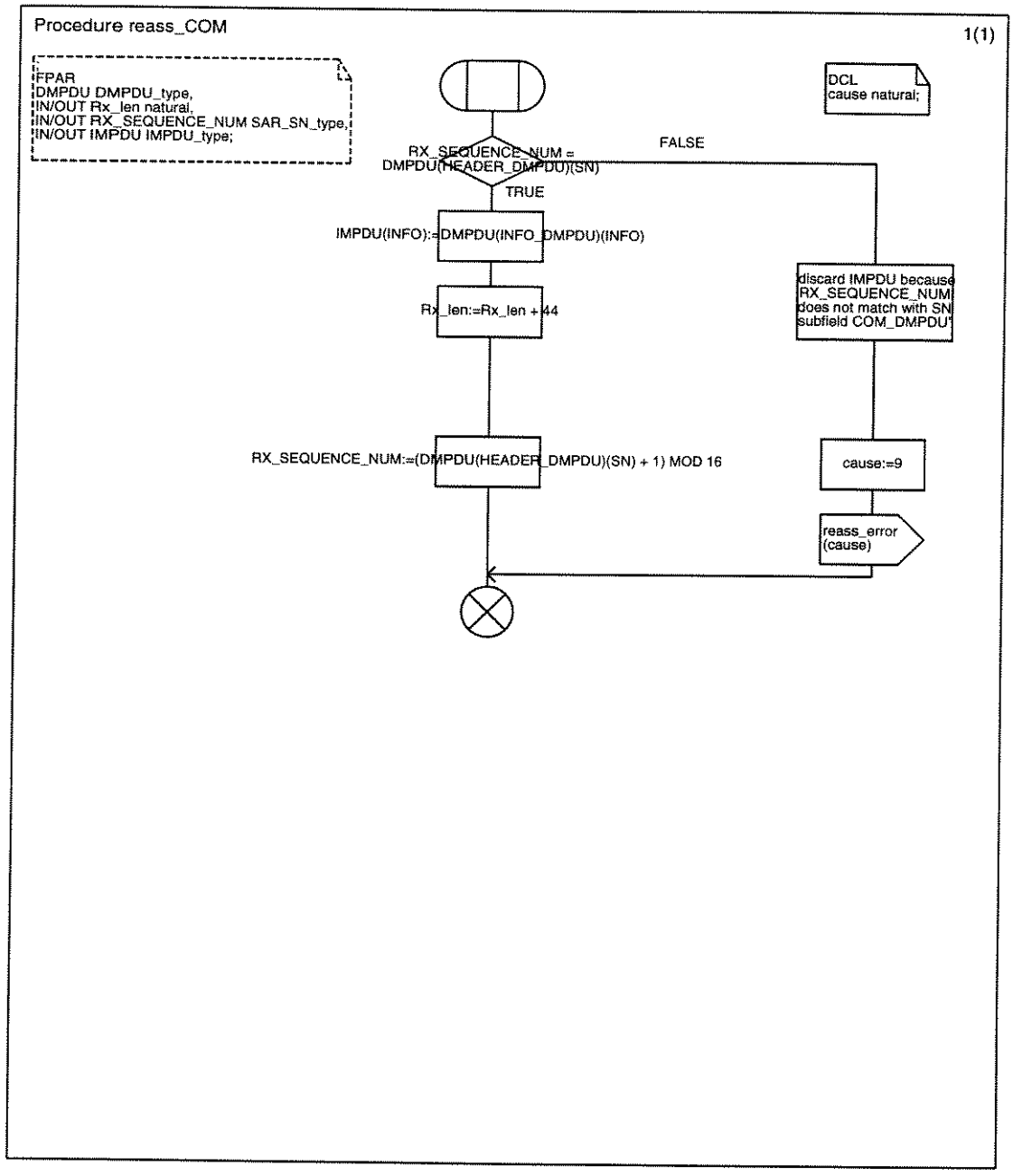

Figura B.89: "Procedimento reass_COM"

Apêndice C

Glossário

AAL	ATM Adaptation Layer
ATM	Asynchronous Transfer Mode
BAS	Buffer Allocation Size
BISDN	Broadband Integrated Service Digital Network
BOM	Beginning of Message
BTAG	Beginning Tag
CCITT	International Telegraph and Telephone Consultative Committee(atualmente designado por ITU-TSS)
CIB	CRC Indicator Bit
COM	Continuation of Message
CRC	Cyclic Redundancy Check
DA	Destination Address
DMPDU	Derived MAC Protocol Data Unit
DQDB	Distributed Queue Dual Bus
EOM	End of Message
ETAG	End Tag
FIFO	First In First Out
HE	Header Extension
HEL	Header Extension Lenght
IMPDU	Initial MAC Protocol Data Unit
INFO	Information
ISO	International Standard Organization
ITU	International Telecommunication Union
LAN	Local Area Network
LLC	Logical Link Control
LME	Layer Management Entity
MAC	Medium Access Control
MAN	Metropolitran Area Network
MCF	MAC Convergence Function
MSDU	MAC Service Data Unit

MID	Multiplexing Identification
MSDU	MAC Service Data Unit
PAD	Padding
PCI	Protocol Control Information
PDU	Protocol Data Unit
QA	Queue Arbitrated
QOS	Quality of Service
SA	Source Address
SAP	Service Access Point
SDH	Synchronous Digital Hierarchy
SDL	Specification and Description Language
SDT	SDL Design Tool
SDU	Service Data Unit
SG_D	Default Slot Generator
SN	Sequence Number
SSM	Single Segment Message
TSS	Telecommunication Standardization Sector
VCI	Virtual Channel Identifier

Bibliografia

[1] Stallings W., "Local and Metropolitan Area Network", New York, 4a.ed, Macmillan Publishing Company, 1993.

[2] Stallings W., "Networking Standards: A Guide to OSI, ISDN, LAN and MAN Standards", New York, 4a.ed, ADDISON-WESLEY PUBLISHING COMPANY, January, 1994.

[3] CCITT Recommendation Z.100: Specification and Description Language SDL, Blue Book, 1988.

[4] CCITT Recommendation Z.100 Annex D: SDL User Guidelines, Blue Book, 1988.

[5] CCITT Recommendation I.361: B-ISDN ATM Layer Specification, March, 1993.

[6] CCITT Recommendation I.362: B-ISDN ATM Adaptation Layer (AAL) Functional Description, March, 1993.

[7] CCITT Recommendation I.363: B-ISDN Adaptation Layer (AAL) Specification, March, 1993.

[8] CCITT Recommendation I.364: Suport of Broadband Connectionless Data Service on B-ISDN, March, 1993.

[9] CCITT Recommendation I.211: B-ISDN Service Aspects, March, 1993.

[10] CCITT Recommendation E.164: Numbering Plan for the ISDN Era, August, 1991.

[11] IEEE Project 802 - Local and Metropolitan Area Networks, Proposed standard P802.6/D15, IEEE, October 90.

[12] SDT 3.1 User's Guide, TeleLOGIC Malmo AB, 1993.

[13] R. Saracco, J.R. Smith, Telecommunications Systems Engineering using SDL, 1989.

[14] Concher S., Lemos G., Soares L.F.G., Redes de Computadores: das LANs, MANs e WANs às redes ATM, Rio de Janeiro, Editora Campus, 1995.

[15] Venieris I.S., Angelopoulos J.D., Stassinopoulos G.I., "Efficient use of protocol stacks for LAN/MAN-ATM interworking", IEEE J. Select. Areas Commun., vol. 11, no. 8, Oct. 1993.

[16] Venieris I.S., Protonotarios E.N., Stassinopoulos G.I., Carli R., "Bridging remote connectionless LAN/MANs through connection oriented ATM networks", Computer Communications, vol. 15, no. 7, Sept. 1992.

[17] Pedro Grael Jr., "Suporte ao Serviço não Orientado à Conexão na Rede Digital de Serviços Integrados de Faixa Larga", Tese de Mestrado, FEE, UNICAMP, Maio 1994.

[18] Pedro Grael Jr., "Especificação funcional de um servidor para o serviço não orientado à conexão na B-ISDN", Nota técnica 50/94, CPqD Telebrás - DDS, Março 1994.

[19] Pedro Grael Jr., "Especificação funcional de um adaptador de protocolo LAN-ATM para o serviço não orientado à conexão na B-ISDN", Nota Técnica 51/94, CPqD Telebrás-DDS, Março 1994.