PM6344

EQUAD

MEETING ETS 300 011 REQUIREMENTS WITH THE EQUAD

Issue 2: 1998

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 SCOPE

This document is the second issue of the recommendations for configuring the latest revision of the PM6344 EQUAD for compliance with ETS 300 011 requirements. This revision of the EQUAD is that with ID[0] bit =1.This new issue's software initializations and service routines are simplified versions of the recommendations of the first issue of this document. For recommendations for EQUAD versions with ID[0] bit = 0 please consult the PMC Applications Customer Support.

2 REFERENCES

- ETSI, ETS 300 011 (April 1992), “ISDN; Primary Rate User-Network Interface Layer 1 Specification and Test Principles”
- ETSI, ETS 300 011 Amendment 1 (December 1994), “ISDN; Primary Rate User-Network Interface Layer 1 Specification and Test Principles”
- ETSI, TBR 004 (November 1995); "ISDN; Attachment requirements for terminal equipment to connect to an ISDN using ISDN primary rate access"
- ITU-T Recommendation G.703, Study Group XVIII, Resolution No. 2 (April 1991), “Physical/Electrical Characteristics of Hierarchical Digital Interfaces”
- ITU-T Recommendation G.704, Study Group XVIII, Resolution No. 2 (April 1991), “Synchronous Frame Structures Used at Primary and Secondary Hierarchical Levels”
- ITU-T Recommendation G.706, Study Group XVIII, Resolution No. 2 (April 1991), “Frame Alignment and Cyclic Redundancy Check (CRC) Procedures Relating to Basic Frame Structures Defined in Recommendation G.704”
- ITU-T Recommendation I.431 (03/93), "Primary Rate User-Network Interface — Layer 1 Specification"
- PMC-Sierra, Inc. ,"EQUAD With QDSX Reference Design", PMC-960911, Issue 1

3 DEFINITIONS AND TERM GLOSSARY

AIS — Alarm Indication Signal. This is a standardized (ITU-T G.704) signal consisting of an unframed all-ones signal. It is transmitted when the data source for the transmit data is unavailable. The EQUAD provides the ability to transmit and detect AIS.

B-Channel — The B-Channel is the "building block" of the ISDN user channel. Different numbers of B-Channels are grouped with a D-Channel to produce H0, H11 and H12 ISDN channels.

BER — Bit Error Ratio. This is the ratio of errored bits received to the total bits received.

CAS — Channel-Associated Signalling. This is a standardized (ITU-T G.704) use of TS16 of the E1 frame. CAS consists of sending a 16-frame signalling multiframe in TS16. CAS is not used for ISDN.

CCS — Common Channel Signalling. This is a standardized (ITU-T G.704) use of TS16 of the E1 frame. CCS consists of sending HDLC formatted packets over TS16. CCS is the same as the D-Channel for ISDN primary rate applications.

CDRC — Clock and Data Recovery Unit. Used by PMC-Sierra to denote the functional block within the EQUAD which contains the circuitry for recovering the timing from the received E1 signal to which the received data is re-timed. It is implemented with a DPLL.

CMF — CRC-4 Multiframe. This is a standardized (ITU-T G.704) use of the International Use Bits, Si, in Timeslot 0 of the E1 frame. The Si bits are used to provide a MFAS, a CRC-4 check calculated over the E1 payload, and a far end error indication via E-bits.

CRC — Cyclic Redundancy Check. A type of parity check used to detect bit errors across a transmission link.

D-Channel — The D-Channel is the ISDN datalink channel associated with some number of ISDN user channels. The D-Channel carries LAPD formatted messages. It is carried in TS16 of the E1 frame for ISDN primary rate applications.

DPLL — Digital Phase-Locked Loop. This term is used for phase-locked loops which make phase corrections in discrete (quantized) corrections.

DJAT Digital Jitter Attenuator. This term is used by PMC-Sierra to denote the functional block within the EQUAD which contains the circuitry to attenuate the phase jitter on the transmit timing reference. It is implemented with a DPLL.

E1 European First Order transmission format. This is the standardized (ITU-T G.704) base format of the European Pleisosynchronous Digital Hierarchy. It operates at 2048 kbit/s.

The E1 format consists of frames consisting of 32 octets, or timeslots (numbered 0 to 31). Timeslot 0 alternates between containing an FAS and containing the National Use bits (Sn[8:4]) and an A-bit for RAI. All Timeslot 0s contain an International Use Bit (Si) which can be used to provide a CMF format.

EQUAD Quadruple E1 Framer. This is the mnemonic label PMC-Sierra gives to its PM6344 device.

ETSI European Telecommunication Standards Institute. This is a standards body that has the primary objective of end-to-end compatibility for pan-European telecommunications connections.

F0 State zero in the I.431 states table for the user side of the UNI. This state is "Loss of Power on User Side". In this state, the TE can neither transmit nor receive signals.

F1 State one in the I.431 states table for the user side of the UNI. This state is the "Operational State". In this state, network timing and Layer 1 service is available, CRC multiframing is transmitted and received.

F2 State two in the I.431 states table for the user side of the UNI. This state is the "Fault Condition No. 1". This state is the same as F1 except RAI is received. The differences between this state and State F5 depend on the I.604 option used in the network.

F3 State three in the I.431 states table for the user side of the UNI. This state is the "Fault Condition No. 2". In this state, network timing is not available, the user side detects loss of signal (and corresponding loss of frame alignment), and the user side transmits NOFs, but with RAI.

F4 State four in the I.431 states table for the user side of the UNI. This state is the "Fault Condition No. 3". In this state, network timing is not available, the user side detects AIS (and corresponding loss of frame alignment), and the user side transmits NOFs, but with RAI.

F5 State five in the I.431 states table for the user side of the UNI. This state is the "Fault Condition No. 4". This state is the same as F1 except RAI is received. The differences between this state and State F2 depend on the I.604 option used in the network.

F6 State six in the I.431 states table for the user side of the UNI. This state is the "Power On State". This is the only transient state for the user side defined in I.431. The user side may change to another state after detection of a received signal.

FAS Frame Alignment Signal. This term is used both for the frame alignment signal itself (a 0011011 pattern) and for E1 frames that contain the frame alignment signal in Timeslot 0.

FEBE Far End Block Error. In E1, this term refers to the E-Bits in the CMF structure. Two E-Bits are provided, one for each sub-multiframe. They are set to logic zero in the output signal when the input signal contains a CRC error.

FIFO First In, First Out. This term refers to a kind of digital buffer which outputs the data serially in the same order in which it was input.

FRMR Framer. Used by PMC-Sierra to denote the functional block within the EQUAD that contains the framing circuitry required to find and maintain FAS and MFAS alignment to the received E1 signal. It also contains circuitry for the detection of RAI and AIS.

FTP Feature Test Plan. The term used by PMC-Sierra to describe the procedures used by the PMC-Sierra Product Verification group to verify the functionality of a PMC-Sierra product. Generally, the FTP is performed on an evaluation PCB designed for that purpose, using industry standard test equipment. By PMC-Sierra's ISO-9001 procedures, the FTP must be passed successfully before releasing a product as a Production device.

G0 State zero in the I.431 states table for the network side of the UNI. This state is "Loss of Power in the NT1". In this state, the NT1 can neither transmit nor receive signals.

G1 State one in the I.431 states table for the network side of the UNI. This state is the "Operational State". In this state, network timing and Layer 1 service is available, CRC multiframing is transmitted and received.

G2 State two in the I.431 states table for the network side of the UNI. This state is the "Fault Condition No. 1". This state is the same as G1 except RAI is received. The differences between this state and State G5 depend on the I.604 option used in the network.

G3 State three in the I.431 states table for the network side of the UNI. This state is the "Fault Condition No. 2". In this state, network timing is not provided to the user side, and the network side transmits NOFs.

G4 State four in the I.431 states table for the network side of the UNI. This state is the "Fault Condition No. 3". In this state, network timing is not provided to the user side, the network side transmits AIS, and the network side receives NOFs, but with RAI.

G5 State five in the I.431 states table for the network side of the UNI. This state is the "Fault Condition No. 4". The network side detects LOS or OOF, and the network side transmits NOF with RAI.

G6 State six in the I.431 states table for the network side of the UNI. This state is the "Power On State". This is the only transient state for the network side defined in I.431. The network side may change to another state after detection of a received signal.

H0 This term refers to a ISDN user channel consisting of six B-Channels. A group of five H0s share a D-Channel for ISDN 2048 kbit/s primary rate applications.

H11 This term refers to the ISDN user channel consisting of 23 B-Channels. The H11 channel is associated with a single D-Channel for ISDN 1544 kbit/s primary rate applications.

H12 This term refers to the ISDN user channel consisting of 30 B-Channels. The H12 channel is associated with a single D-Channel for ISDN 2048 kbit/s primary rate applications.

HDB3 High-Density Bipolar encoding of order 3. This is a standardized (ITU-T G.703) line coding scheme used for zero suppression, while maintaining clear channel capability. It is used for E1 transmission.

HDLC High-Level Data Link Control. This protocol is commonly specified for data link communications. A subset of HDLC, LAPD, is used over the ISDN D-Channel.

I_a Interface A. This refers to the user side of the UNI.

I_b	Interface B. This refers to the network side of the UNI.
IPATS	This is a test system from Hewlett-Packard used by many test houses to test for ETS 300 011 compliance.
ISDN	Integrated Services Digital Network. This is a world-wide public telecommunications network that implemented as a set of digital switches and paths supporting a broad range of services.
ISO	International Organization for Standardization. ISO is an international, voluntary, non-treaty organization whose members are designated standards bodies from participating nations.
ITU-T	International Telecommunication Union - Telephony. This is a committee within a United Nations treaty organization. The charter is “to study and issue recommendations on technical, operating, and tariff questions relating to telegraphy and telephony.” Its primary objective is end-to-end compatibility of international telecommunications connections.
IUT	Implementation Under Test. This refers to the equipment to which the test stimulus is being applied, and which is being monitored for correct responses.
LAPD	Link Access Protocol for D-Channel. This is a messaging protocol based on HDLC. It is used for the D-Channel messaging in ISDN user channels.
Layer 1	Layer 1 of the ISDN protocol stack. This layer is analogous to the OSI Physical Layer. It is standardized in ITU-T specifications I.430 and I.431.
LOS	Loss of Signal. This is the state a clock and data recovery unit is in when it is impossible to recover the line timing. For E1, LOS is defined as a number of consecutive zeros received.
MFAS	Multiframe Alignment Signal. This is a pattern (001011) carried in International Use Bits of six of the eight NFAS frames composing the CMF format.
NFAS	No Frame Alignment Signal. This term is used for E1 frames that do not contain the frame alignment signal in TS0. Instead TS0 contains the National Use Bits and the RAI.
NOF	Normal Operating Frames. This term denotes frames in which the FAS and MFAS are correct, the line coding (HDB3) is correct, and no RAI is present (the A-bit in TS0 of the NFAS frames is cleared to logic zero).

NT Network Termination. Equipment on the user premises which terminates the network connection.

OOCMF Out-Of-CMF alignment. This is the state an E1 framer is in if it cannot find the MFAS within the International Use bits of TS0. An E1 framer is always OOCMF when it is OOF.

OOF Out-Of-Frame alignment. This is the state an E1 framer is in if it cannot find the FAS within the received serial 2048 kbit/s data.

PMON Performance Monitor. This term is used by PMC-Sierra to denote the functional block within the EQUAD which contains counters which accumulate performance statistics such as framing bit errors, FEBEs, CRC-4 errors, and line code violations. The accumulation interval is under the control of the microprocessor.

RAI Remote Alarm Indication. The A-bit in TS0 of NFAS frames is used to signal to the remote equipment that a receiver alarm condition (such as OOF) is present.

RED Red Alarm. This is an alarm state that is entered due to persistent OOF conditions. The EQUAD automatically integrates OOF and signals a Red Alarm via the REDI and RED bits in Registers 25H and 27H respectively.

T3 Timer 3. This is the timer used in the transient states matrix defined in this document. It is used to implement the recommendations in G.706 Section 4.2 regarding the time-outs in the search for MFAS alignment.

TE Terminal Equipment. Subscriber interface equipment on the user premises.

TS0 Time Slot 0. This refers to the first timeslot (octet) in the E1 frame. TS0 alternates between the FAS and NFAS frames. The International Use Bit, Si, in TS0 of both FAS and NFAS frames may carry the MFAS. The TS0 of NFAS frames carries the A-Bit used for RAI.

TS16 Time Slot 16. This refers to the seventeenth timeslot (octet) in the E1 frame. TS16 can be used generally for either CAS or CCS. For ISDN, TS16 is used for CCS, also called the D-Channel.

UI Unit Interval. This is a unit used to measure phase jitter. The unit is the time deviation of phase normalized to the bit period.

UNI User-Network Interface.

4 BACKGROUND AND OVERVIEW

PMC-Sierra's PM6344 EQUAD Quadruple E1 Framer is a full-featured device which is fully compatible with European Primary Rate ISDN requirements. In relation to the ISDN protocol stack, the E1 transceiver functionality is part of Layer 1. The primary source of European ISDN specifications is ETSI. For Layer 1, the primary ETSI specifications are ETS 300 011 and TBR 004.

ETS 300 011 and TBR 004 are based on the ITU-T suite of requirements for interfaces operating at 2048 kbit/s. However, ETSI expands on the ITU-T standards by specifying the test principles. Annex C of ETS 300 011 is devoted to "Conformance test principles for the user and the network side of the interface."

The PM6344 EQUAD has a parallel microprocessor port that provides access to the EQUAD'S internal register set. Via these registers, all relevant Layer 1 status conditions can be monitored, and all maintenance responses can be controlled such that the E1 interface complies with ETS 300 011. However, it is left to the software to appropriately control the EQUAD's responses to various status conditions.

This document makes general recommendations for software and hardware development, then describes each test in ETS 300 011 Annex C. It explains how following PMC-Sierra's recommendations for designing with the PM6344 EQUAD will allow the implementation to meet the ETS 300 011 requirements.

It is important to note that the recommendations provided in this document are valid only for the PM6344 EQUAD, or EQUAD with ID[0] bit =1. For EQUADs with ID[0] bit =0, the recommendations in the document “Meeting ETS 300 011 Requirements with the EQUAD”, PMC-970239 Issue #1, should be used.

Note: It is assumed that the reader is familiar with the documents cited in the References section. They should be read in conjunction with this document.

5 SOFTWARE RECOMMENDATIONS

The EQUAD has a full-featured register set which provides access to all the status monitoring and control functions necessary to comply with ETS 300 011 tests. However, it is up to the software to use the EQUAD registers properly. This section details the considerations that must be taken while developing software for an EQUAD design.

The software has four main functions:

- initializing,
- performance monitoring,
- implementing of the I.431 states matrix for static states,
- handling transient states.

This section explains how the software development should be approached for each of these four functions.

Interrupt-driven routines are recommended because they are much more efficient than polling routines when responding to low frequency events such as Layer 1 defects. In general, changes in Layer 1 status can be enabled to generate an interrupt indication on the EQUAD'S INTB output pin.

5.1 Initialization

After EQUAD power up, a software reset should be performed on the EQUAD to put it in a default state. Setting the RESET bit (register 0DH) then clearing it performs the software reset. Each quadrant of the EQUAD can be reset independently. However, additional configuration is required. The recommended initial configuration is given in Table 1.

Registers that are not included in Table 1 should be left in their default state. Also, bits that are not included in Table 1 should be left in their default state. Table 1 shows the offset registers from the base address of each quadrant of the EQUAD. All four quadrants should be initialized for the same configuration.

The base addresses are as follows:

- Quadrant 0 base address is 000H
- Quadrant 1 base address is 080H
- Quadrant 2 base address is 100H
- Quadrant 3 base address is 180H

Table 1. Recommended Register Initialization

Offset Register Address (hex)	Recommended Configuration	
06	TXSA4EN=1	Default value.
09	RXSA4EN=0	This allows TS16 to be extracted for D-Channel processing.
11	LOSE=1	This enables changes in the status of the LOS detection circuit to generate interrupt indications on the EQUAD'S INTB output pin.
10	AMI=X ALGSEL=1	Setting this bit disables the HDB3 decoding. This selects a clock recovery algorithm with the best tolerance of high frequency jitter.
19	N1[7:0]=FF	The value in this register must be the same as in Register 1AH when the transmit timing reference is at the line rate.
1A	N2[7:0]=FF	This sets the DJAT transfer function for maximum jitter attenuation.
1B	CENT=1 SYNC=0 LIMIT=0	This allows the DJAT FIFO to centre itself thereby providing maximum room to absorb phase differences between the input and output transmit clocks. This bit should be cleared whenever Register 1AH does not contain its default value (2FH). This function should be disabled so that the DJAT FIFO does disrupt the DJAT PLL operation. With the hardware connections recommended in this document, the FIFO should never reach a condition where LIMIT would be useful.
20	CRCEN=1 CASDIS=1 REFCRCE=1	This enables the MFAS alignment circuitry in the FRMR. This disables the CAS multiframe alignment circuitry in the FRMR. This enables the CRC error monitor to force a reframe if an excessive CRC error condition is detected.
21	BIT2C=1	This enables the EQUAD to declare OOF if Bit 2 of TS0 of NFAS frames is received incorrectly for three consecutive frames.

Offset Register Address (hex)	Recommended Configuration	
22	OOFE=1	This enables changes in the status of the FAS alignment circuit to generate interrupt indications on the EQUAD'S INTB output pin.
	OOCMFE=1	This enables changes in the status of the MFAS alignment circuit to generate interrupt indications on the EQUAD'S INTB output pin.
23	RRAE=1	This enables changes in the status of the RAI detection circuit to generate interrupt indications on the EQUAD'S INTB output pin.
	REDE=1	This enables changes in the status of the RED detection circuit to generate interrupt indications on the EQUAD'S INTB output pin.
	AISE=1	This enables changes in the status of the AIS circuit to generate interrupt indications on the EQUAD'S INTB output pin.
26	MFASFIX=1	This disables a re-search for BFA after the receipt of 4 consecutive bad MFASs.
30	IND=1	This enables indirect accessing of the Transmit Per-Channel Serial Controller (TPSC) registers within the EQUAD. The indirect registers within the TPSC should be initialized as explained below.
44	SIGEN=0	This disables the transmission of Channel-Associated Signalling in TS16.
	DLEN=0	This disables the transmission of Channel-Associated Signalling in TS16.
	GENCRC=1	This enables the generation of the CRC multiframe.

Idle timeslots within the E1 frame must contain at least three binary ones. To implement insertion of compliant Idle codes, the Transmit Per-Channel Serial Controller (TPSC) in the EQUAD must be properly initialized.

The TPSC contains a number of indirect registers that are used to control the transmitted timeslots on a per-channel basis. Accessing these registers is described in the section of the EQUAD data book entitled "Using the Per-Channel Serial Controllers".

The indirect registers in the TPSC should be initialized so that each timeslot's IDLE code byte contains a value with at least three binary ONEs. Additionally, each timeslot's Data Control byte should be programmed so that the DS[1:0] bits are both cleared to logic zero, and the SUBS bit is set according to whether that timeslot is idle or not. Refer to the EQUAD data book for further information on the use of these bits.

Once the TPSC is initialized, the PCCE bit in Register 30H should be set to logic one, enabling the per-channel functions as programmed in the TPSC indirect registers.

The variables and timers for implementing the I.431 states matrix and the transient states matrix (see section 4.4.4) should all be reset. Note that the most robust implementation would be to initialize the variables with a " best guess " of the current state.

For D-channel processing, either the internal or external HDLC controller can be used. This selection should be set up at initialization by controlling the RXDMASIG and TXDMASIG bits in register 02H. If the internal HDLC controller is used, it should be used in accordance with the " Using the Internal HDLC Receiver " and " Using the Internal HDCL Transmitter " in the EQUAD databook.

5.2 Performance Monitoring

In I.431, performance monitoring of the 2048 kbit/s primary rate interface is optional.

The PMON functional block of the EQUAD provides counters for accumulating performance monitoring statistics, with the accumulation interval under microprocessor control. Refer to the description of Registers 49H to 4FH in the EQUAD data book for an explanation of the operation of these registers.

The PMON counters are large enough such that the probability of saturating them during a one second interval is less than 0.001% under a BER of $1:10^3$. The values in these counters can be used to extrapolate the actual BER (see the section entitled "Using the Performance Monitor Counter Values" in the EQUAD data book).

Whether or not the performance monitoring statistics are collected by the system, the FEBE count must be monitored to detect the I.431 F5 state. The F5 state is distinguished from the F2 state by the continuous reception of FEBEs.

Continuous reception of FEBEs corresponds to FEBEs received at a rate of 1000 per second. Therefore, the PMON FEBE counter (Registers 4AH and 4BH) must be polled to check the rate of FEBE reception.

I.431 says that the reception of continuous FEBEs should cause transitions within the states matrix. It is suggested that the PMON FEBE counters be polled every 10 ms and the value returned compared with a threshold to determine if continuous FEBEs have been received over the last 10 ms interval.

Therefore, if 10 FEBEs are received in 10 ms, continuous FEBEs are being received. However, the tolerance of the timer as well as robustness against bit errors should also be taken into account. Therefore, the threshold corresponding to continuous FEBEs received is given by:

$$threshold = FEBE\_rate_{\max} \times T \times \left(1 - BER_{\max} - \frac{tolerance}{10^6}\right) \quad (1)$$

where *threshold* is the minimum number of FEBEs which corresponds to continuous FEBEs being received during the polling interval, $FEBE\_rate_{\max}$ is the maximum rate that FEBEs can be indicated (two per multiframe), T is the polling time interval, $BER_{\max}$ is the maximum BER under which continuous received FEBEs is to be detected, and *tolerance* is the sum of the tolerances of the polling timer and the incoming line rate (in ppm).

Note: *threshold* should be rounded down to the nearest integer.

For example, using the recommended values of *T*=10ms and $BER_{max}=10^{-3}$, and assuming *tolerance*=1000ppm, the *threshold* value would be:

$$threshold = 1000\text{bps} \times 0.010\text{s} \times \left(1 - 10^{-3} - \frac{1000\text{ppm}}{10^6}\right)$$
$$= 9.98 \approx 9$$

Note: The time it takes for a microprocessor to service the timer interrupts will be variable. This variability should be included in the *tolerance* term.

5.3 Implementing the I.431 States Matrix

ITU-T I.431 defines a states matrix for each side of the ISDN primary rate UNI. The purpose of those matrices is to standardize the way in which each side of the interface informs the other of the Layer 1 status related to the different defects that may be present.

States F0 to F6 (defined in the Glossary and Term Definitions section) are states at the user side, while states G0 to G6 (also defined in the Glossary and Term Definitions section) are the states at the network side of the UNI.

Corresponding to each state, each side of the interface must exchange information on the Layer 1 status, as well as passing primitives between Layer 1 and Layer 2, and between Layer 1 and the management entity.

The microprocessor monitoring and controlling the PM6344 EQUAD, must be able to correctly implement the I.431 states matrix. Typically, this requires detecting a change in Layer 1 status and responding appropriately. The response usually requires a steady-state signal transmitted toward the interface as well as the passing of primitives to the Layer 2 and management entities.

Since the frequency of the state transitions is generally low, it is best to handle them using interrupt-driven routines. In the EQUAD, many different conditions can be enabled to generate interrupt indications on the INTB output.

In order to implement the states matrix, it is sufficient to enable the interrupts due to RAI detection, RED alarm assertion, and AIS detection. The detection circuits for all three of these defects are contained in the FRMR functional block of the EQUAD. To enable these interrupts, the following bits should be set to logic one (as specified in the Initialization section):

- RRAE bit in Register 23H
- REDE in Register 23H
- AISE in Register 23H

When an interrupt is indicated on the INTB output of the EQUAD, the microprocessor should first read Register 08H. Register 08H indicates which functional blocks within the EQUAD are asserting the INTB signal. If the FRMR bit is set to logic one, then one of the above interrupts may have occurred (note that other FRMR interrupt sources will likely be enabled for purposes other than the states matrix — see section entitled "Transient States").

Once it is determined that a FRMR interrupt has occurred, the microprocessor should read Register 25H. For each interrupt source, there is a corresponding interrupt indication bit whose mnemonic label ends with an "I". These bits are set when the corresponding status bit changes state. Therefore, in Register 25H, the RRAI, REDI, and AISI bits should be examined.

Note: The interrupt indication bits are cleared upon read. This also clears the assertion of the INTB signal (unless other interrupts are pending). Therefore, it is important to save the value of the interrupt indication bits until they are fully processed.

If any of these bits is logic one, then the corresponding state has changed since the last time that register was read, generating the interrupt. If one or more of the RRAI, REDI, and AISI bits are set to logic one, then the microprocessor should read Register 27H. In that register, the RRA, RED, and AIS bits should be examined to determine the current state of the FRMR. Depending on these values, the appropriate state transition should be performed, as explained below.

ITU-T I.431 defines the states matrices for the user and network side of the UNI. It contains two tables which show how new events should move the equipment through the state matrix, including what signal to transmit towards the interface and what primitives to pass to the Layer 2 and management entities.

The difficulty with implementing the matrices in I.431 is that they do not explicitly explain which events are mutually exclusive, and which events take precedence over others. For example, when AIS is received a framer will declare RED Alarm; however, AIS and RED Alarm are considered different events that move the equipment into different states. Therefore, a decision has to be made as to the precedence of these events.

In the following two tables, Tables 2 and 3, the I.431 states matrices are re-written for actual interrupt events within the EQUAD. These tables look different than those in I.431 because interrupt events in the EQUAD do not have a one-to-one correspondence with the events defined in I.431.

In Tables 2 and 3, the following acronyms and symbols are used:

/	Impossible event.
—	No response required.
AIS	There are two bits in the EQUAD with this name: Bit 2 in Register 27H, and Bit 0 in Register 45H. Respectively, these bits indicate the current status of the AIS detection circuit and control the transmission of AIS.
AISI	Bit 2 in Register 25H of the EQUAD. This bit indicates that a change of status of the AIS detection circuit has caused an interrupt.
FEBE	This variable contains the number of FEBEs received over the last performance monitoring polling interval. This value is read from the PMON FEBE counter (Registers 4AH and 4BH) in the EQUAD.
F*z*	Go to State F*z*.
G*z*	Go to State G*z*.
MPH-*y*	Issue management primitive *y* as defined in I.431.
RCLKO	The clock recovered from the received signal. It is available on the EQUAD's RCLKO output pin as well as internally as the transmit timing reference. For more information, refer to the Timing Connections section.
RED	Bit 3 in Register 27H of the EQUAD. This bit indicates the current state of the RED alarm integration circuit. RED Alarm is an integrated version of OOF.
REDI	Bit 3 in Register 25H of the EQUAD. This bit indicates that a change of status of the RED alarm integration circuit has caused an interrupt.
REMAIS	Bit 3 in Register 45H of the EQUAD. This bit controls the transmission of the RAI in outgoing signal.
PH-*x*	Issue primitive *x* as defined in I.431.
RRAI	Bit 7 in Register 25H of the EQUAD. This bit indicates that a change of status of the RAI detection circuit has caused an interrupt.
RRA	Bit 7 in Register 27H of the EQUAD. This bit indicates the current status of the RAI detection circuit.
TCLKI	An independent timing signal applied to the TCLKI input of the EQUAD. For more information, refer to the Timing Connections section.

Table 2. States Matrix at User Side

Initial State			F1	F2	F3	F4	F5
Remote Alarm Transmission			REMAIS=0	REMAIS=0	REMAIS=1	REMAIS=1	REMAIS=0
Transmit Timing Reference			RCLKO	RCLKO	TCLKI	RCLKO	RCLKO
New Event	RRAI=1	RRA=0	/	PH-AI MPH-AI F1	/	/	PH-AI MPH-AI F1
		RRA=1	PH-DI MPH-E11 F2	/	/	/	/
	REDI=1	RED=0	/	/	PH-AI MPH-AI F1	PH-AI MPH-AI F1	/
		RED=1	PH-DI MPH-E12 F3	MPH-E12 F3	/	—	MPH-E12 F3
	AISI=1	AIS=0	—	/	—	MPH-E12 F3	/
		AIS=1	PH-DI MPH-E13 F4	MPH-E13 F4	MPH-E13 F4	/	MPH-E13 F4
	FEBE<threshold[1]		—	—	—	—	MPH-E11 F2
	FEBE•threshold[1]		—	MPH-E14 F5	—	—	—

1 The threshold for continuous FEBE detection is calculated using Equation (1) in the Performance Monitoring section.

Table 2 does not contain information on States F0 and F6 since the events related to these states (loss and return of power) cannot be detected within the EQUAD. These events must be detected externally as part of the power supply design.

Table 3. States Matrix at Network Side

Initial State			G1	G3	G5
Transmitter Configuration			REMAIS=0 AIS=0	REMAIS=0 AIS=0	REMAIS=1 AIS=0
New Event	RRAI=1	RRA=0	/	PH-AI MPH-AI G1	/
		RRA=1	PH-DI MPH-E12 G3	/	/
	REDI=1	RED=0	/	/	PH-AI MPH-AI G1
		RED=1	PH-DI MPH-E14 G5	MPH-E14 G5	/

Table 3 does not contain information on States G0, G2, G4 and G6 since the events related to these states (loss of power, FC1, FC3, and power on) cannot be detected within the EQUAD.

5.4 Transient States

In addition to the static states defined in the I.431 states matrices, there are transient states that must be handled by the EQUAD.

Some important transient states that must be handled properly are those related to framing. This section explains how the microprocessor should use the status and control bits within the EQUAD to properly handle transient states related to framing procedures.

Tests C.4.3 (Frame Alignment), C.4.4 (CRC Multiframe Alignment) and C.4.5 (CRC Processing) in ETS 300 011 stress the IUT's response to transient framing conditions. Because the stimulus for C.4.4 was poorly specified in the 1992 issue of the specification, Amendment 1 (A1) to ETS 300 011 was issued (in 1994) containing a new C.4.4 stimulus which supersedes the previous specification. Equipment based on the EQUAD should be tested to Amendment 1 for C.4.4.

The EQUAD can interrupt the microprocessor in response to transient states via its INTB output. It is recommended that interrupts be used for all the maintenance functions since these events usually have a low frequency of occurrence, but require a quick response.

5.4.1 Frame Alignment

The ETS 300 011 C.4.3 test stresses the terminal equipment's ability to properly assert RAI in response to loss of FAS alignment.

The correct response is described in ITU-T G.706 Section 4.1. This standard specifies several options. However, inspection of the expected response in ETS 300 011 C.4.3 implies that:

- Frame alignment is considered found if the FAS is found followed in the next frame with Bit 2 set to logic one, followed by a FAS in the next frame.
- Frame alignment is considered lost if three consecutive incorrect FASs have been received. Additionally, frame alignment is considered lost if Bit 2 in TS0 in NFAS frames has been received in error on three consecutive occasions.

The EQUAD provides configuration bits to support all the framing options given in G.706. If the registers are programmed as indicated in the Initialization section, the EQUAD will conform to the ETS 300 011 expectations.

The EQUAD provides the OOF bit in Register 26H to indicate the state of its FAS alignment circuit. Additionally, changes of state of the OOF bit are indicated in the OOFI bit in Register 24H. The OOFI bit can be enabled to indicate interrupts on the INTB signal by setting the OOFE bit in Register 22H to logic one.

Generally, the equipment should send an RAI in response to loss of FAS alignment. This is accomplished by setting the REMAIS bit to logic one when the EQUAD interrupts indicating OOFI=1 and OOF=1. The REMAIS bit should be cleared to logic zero when the EQUAD interrupts indicating OOFI=1 and OOF=0.

However, as explained in the next subsection, the EQUAD can be forced out of FAS alignment by the circuit searching for MFAS alignment. Therefore, the OOFI interrupt should be handled as part of a more comprehensive FRMR interrupt-handling routine. A transient states matrix to handle this is detailed in Section 4.4.4. An explanation of how that matrix responds to the C.4.3 test stimulus is given in Appendix B.

5.4.2 CRC Multiframe Alignment

The ETS 300 011 (A1) C.4.4 test stresses the terminal equipment's ability to properly assert RAI in response to a loss of MFAS alignment.

The correct response is described in ITU-T G.706 Section 4.2 that specifies three main requirements concerning responses to MFAS alignment status:

- MFAS alignment is considered found if at least two valid MFASs are located within 8 ms (64 basic frames), with the time separating the two MFASs being 2 ms or a multiple of 2 ms.

- If MFAS alignment cannot be achieved in 8 ms, a re-search for FAS alignment should be initiated.

- If MFAS alignment cannot be achieved within a time limit in the range of 100 ms to 500 ms, consequent actions should be taken equivalent to those specified for loss of frame alignment.

It is important to understand the intent of this specification. If the terminal equipment is configured to expect a CRC-4 multiframe formatted signal, then it searches for MFAS alignment after finding FAS alignment. If it cannot find the MFAS alignment in a reasonable time, then the equipment should "assume" that it has framed to a mimic basic FAS. It therefore forces a re-search for the FAS alignment. After doing this a number of times without ever finding MFAS alignment, then the equipment makes another "assumption" — it assumes that the far end equipment is not sending a CMF formatted signal and gives up disrupting the FAS alignment. However, if the equipment does find MFAS alignment at least once before giving up then it assumes that the far end is sending a CMF-formatted signal but there is another problem (e.g. high BER).

The ETS 300 011 (A1) C.4.4 test expects RAIs to be transmitted to indicate the conformance to the specification. An RAI may be sent every time a reframe is forced by the MFAS alignment circuitry. Additionally, if the equipment decides that the far end is not sending a CMF-formatted signal, it should continuously assert RAI, set and keep set the E bits of the transmitted MFAS alignment signal. If the equipment decides that the far end is sending a CMF-formatted signal but that another problem is present, then it stops asserting RAI.

The EQUAD provides a number of register bits to indicate the state of the FRMR functional block:

- the OOF and OOCMF bits in Register 26H. These bits indicate the current status of the basic and CMF frame find circuits.

- the OOFI and OOCMFI bits in Register 24H. These bits indicate that a change of state has occurred in the OOF and OOCMF bits, respectively, at least once since the last time Register 24H was read.

- the OOFE and OOCMFE bits in Register 22H. These bits enable the OOFI and OOCMFI bits to generate interrupt indications on the INTB output pin.

- the CMFACT bit in Register 21H. This bit indicates that a re-search for FAS alignment has been forced at least once since the last time Register 21H was read.

It is expected that the MFAS alignment transient states will be processed using interrupts.

A comprehensive interrupt-handling routine expects interrupts from a variety of sources within the PM6344 EQUAD. Therefore, it should first determine which quadrant interrupted then whether the FRMR was the source of the interrupt before processing the OOF and OOCMF status bits. To do this, the microprocessor should read Register 08H to check if the FRMR bit is set to logic one. If so, the software should call a routine which implements the transient framing states matrix detailed in the next section.

Once a FRMR interrupt is detected, a software routine which implements the transient states matrix in Section 4.4.4 should be called. The operation of that routine in response to the C.4.4 (A1) test stimulus is explained in Appendix C.

5.4.3 CRC Processing

The ETS 300 011 C.4.5 test stresses the terminal equipment's ability to properly send FEBE's and assert RAI in response to received CRC-4 errors.

The correct response is described in ITU-T G.706 Section 4.3.2. That standard says:

- To achieve the probability bounds for detection of false frame alignment, a preferred threshold count is 915 errored CRC blocks out of 1000, with the understanding that a count of $\geq$ 915 errored CRC blocks indicates false frame alignment.

The EQUAD provides configuration bits to support various framing options. If the registers are programmed as indicated in the Initialization section, the EQUAD will respond to stimulus as expected by the ETS 300 011 tests.

The EQUAD provides the REFCRCE bit in Register 20H to enable reframing due to excessive CRC-4 errors. Also, It provides the OOF bit in Register 26H to indicate the state of its FAS alignment circuit. Additionally, changes of state of the OOF bit are indicated in the OOFI bit in Register 24H. The OOFI bit can be enabled to indicate interrupts on the INTB signal by setting the OOFE bit in Register 22H to logic one.

Generally, the equipment should send an RAI in response to loss of FAS alignment. This is accomplished by setting the REMAIS bit to logic one when the EQUAD interrupts indicating OOFI=1 and OOF=1. The REMAIS bit should be cleared to logic zero when the EQUAD interrupts indicating OOFI=1 and OOF=0. An explanation of how the matrix in Section 4.4.4 responds to the C.4.5 test stimulus is given in Appendix D.

Annex B of ITU-T G.706 recommends that for CRC-4 to Non-CRC-4 equipment interworkings the equipment having the CRC-4 capability should continue to transmit CRC-4 data to the distant end with both “E” bits set to one. To ensure that this takes place a few extra bits will be set when the transient code’s T3 timer expires and multiframing is assumed to not being received. Setting and clearing the following bits will force the “E” bits to one.

- the GENCRC and FEBEDIS bits in Register 44H should be set.
- the Si[0] and Si[1] bits in Register 46H should be set.

5.4.4 Transient Framing States Matrix

Figure 1 shows a decision tree to determine which event has occurred based on status bits and software variables.

Figure 1. Event Decision Tree

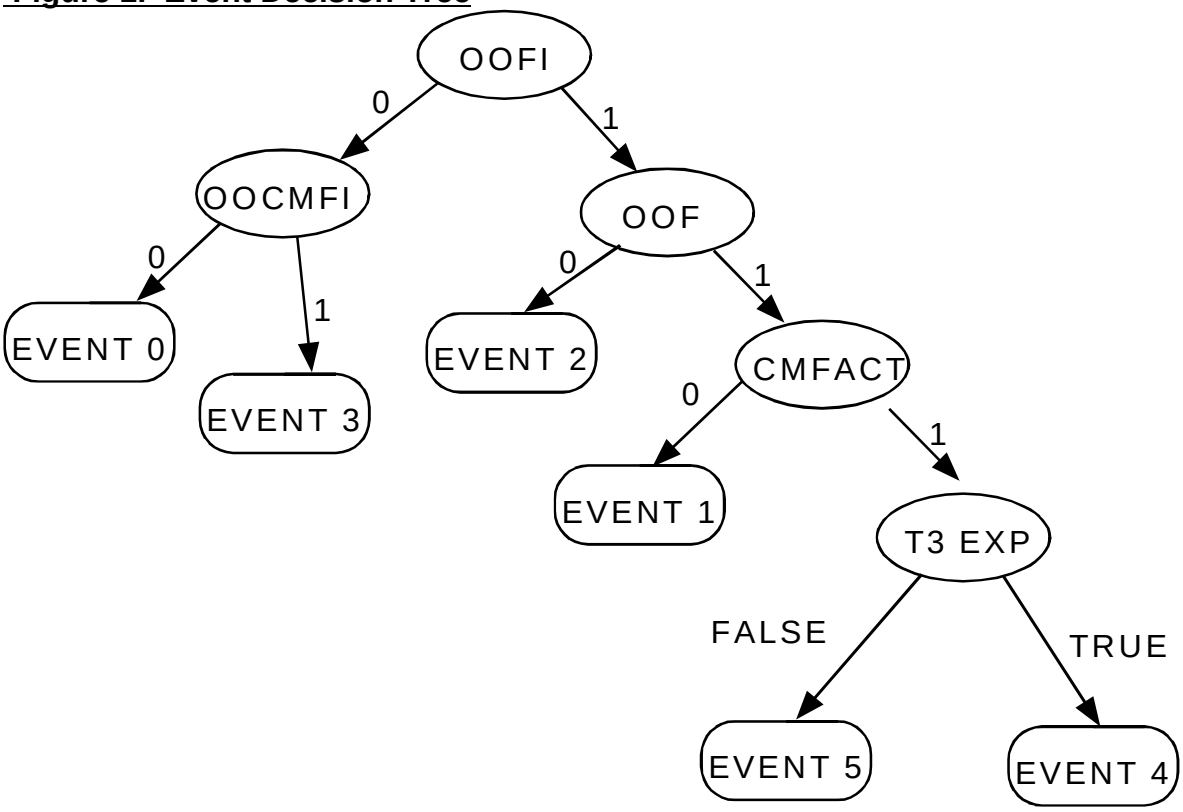

Table 4 depicts a transient state matrix for framing events. The table associates interrupt events with movement about the matrix. The following acronyms and symbols are used:

/ Impossible event. No response is required.

— No response required.

SET_E_BITS Routine in which GENCRC and FEBEDIS of Register 44H are set and Si[0] and Si[1] of Register 46H are set to ensure that the transmitted 'E' bits are set when no MFAS is being received.

CMFACT Bit 1 in Register 21H of the EQUAD. This bit indicates that the current OOF=1 state was caused by the MFAS alignment circuitry which forced a reframe because it could not find MFAS alignment within 8 ms.

MSDIS8 Bit 1 in register 26H of the EQUAD. This bit disables 8 ms reframes due to not finding multiframe alignment.

E0 Event 0. Non transient event. This is indicated by OOFI=0 and OOCMFI-0.

E1 Event 1. FAS alignment lost due to FAS errors. This is indicated by OOFI=1, OOF=1, and CMFACT=0.

E2 Event 2. FAS alignment found. This is indicated by OOFI=1 and OOF=0.

E3 Event 3. MFAS alignment found. This is indicated by OOFI=0 and OOCMFI=1.

E4 Event 4. FAS reframe forced and T3 is expired and no MFAS alignment was found before T3 expired. This is indicated by OOFI=1, OOF=1, CMFACT=1, and T3>300 ms.

E5 Event 5. FAS reframe forced while timer, T3, is not yet expired and MFAS alignment has not been found. This is indicated by OOFI=1, OOF=1, CMFACT=1 and T3<300 ms.

E_BITS Routine that disables the forcing of the "E" bits to one.

FALSE This is the Boolean value FALSE.NOF This is a routine which, when called, clears the RAI indication (A-Bit in TS0) in the transmitted frame. This is accomplished by clearing the REMAIS bit in Register 45H of the EQUAD to logic zero.

OOCMF Bit 4 in Register 26H of the EQUAD. This bit indicates the current state of the MFAS alignment circuit.

OOCMFI Bit 4 in Register 24H of the EQUAD. This bit indicates that a change of status of the MFAS alignment circuit has caused an interrupt.

OOF Bit 6 in Register 26H of the EQUAD. This bit indicates the current state of the FAS alignment circuit.

OOFI Bit 6 in Register 24H of the EQUAD. This bit indicates that a change of status of the FAS alignment circuit has caused an interrupt.

RAI This is a routine which, when called, sets the RAI indication (A-Bit in TS0) in the transmitted frame. This is accomplished by setting the REMAIS bit in Register 45H of the EQUAD to logic one.

RESET_T3 This is a reference to reset timer T3.

S1 Transient State 1. In this state, the IUT is searching for FAS alignment because it was lost due to FAS errors. RAI should be transmitted during this state.

S2 Transient State 2. In this state, the IUT has acquired FAS alignment and is searching for MFAS alignment. RAI should not be transmitted during this state.

S3 Transient State 3. In this state, the IUT has acquired both FAS alignment and MFAS alignment. This is the normal operational state. RAI should not be transmitted during this state.

S4 Transient State 4. In this state, the IUT has acquired FAS alignment, but not MFAS alignment while the timer, T3, has expired. RAI should be transmitted in this state.

S5 Transient State 5. In this state, the IUT is searching for FAS alignment because the MFAS alignment circuit forced the FAS alignment circuit to reframe while the timer, T3, has not expired. RAI should be transmitted during this state.

S*z* Go to State S*z*.

T3 Timer 3. This is a resettable timer which implements the time limit recommended in ITU-T G.706 Section 4.2. A 300 ms expiry limit is recommended since it falls in the middle of the 100 to 500 ms range specified in G.706.

TRUE This is the Boolean value TRUE.

The transient states matrix in Table 4 assumes that the EQUAD has been initialized as recommended in the Initialization section.

Also, the bits in Register 24H and the CMFACT bit in Register 21H are cleared upon read. Therefore, whenever these registers are read all the bits must be fully processed or else stored for future processing.

Table 4 should be interpreted as follows:

a) The interrupt event should be determined based on Figure 1.

b) The entry in Table 4 should be located, the one which corresponds to the current state and the interrupt event.

c) If the table entry has a " / " or " - " then no action need be taken. Else the following actions need to be taken:

i) The next state (indicated in the entry) should be entered.

ii) The software variables should be updated as indicated.
iii) The Timer T3 should be controlled as indicated.
iv) The transmitted RAI should be controlled as indicated.

Table 4. Transient States Matrix

Event	Initial State				
	S1	**S2**	**S3**	**S4**	**S5**
E1	/	S1 RAI MSDIS8=0 E_BITS			/
E2	S2 RESET_T3 NOF	/		—	S2 NOF
E3	/	S3 NOF	/	S3 NOF MSDIS8=0 E_BITS	/
E4	/	S4 RAI MSDIS8=1 SET_E_BITS	/	—	/
E5	/	S5 RAI	/		

6 HARDWARE RECOMMENDATIONS

For details of the hardware necessary for an EQUAD-based design to comply with the ETS 300 011 electrical requirements, refer to the EQUAD with QDSX Reference design (PMC-960911).

7 TEST DETAILS

This section reviews every test described in ETS 300 011's Annex C. For each test, a brief description is given, then PMC-Sierra's recommendations for designing with the EQUAD such that the implementation passes that test. Additionally, information is given which proves the EQUAD'S conformance or compatibility with each requirement.

The following subsections are numbered according to their numeration in ETS 300 011.

7.C.2 Electrical Characteristics

These tests are designed to check that the interface conforms to the electrical requirements of I.431.

7.C.2.1 Bit-Rate When Unsynchronized

This test applies to both the I_a and I_b interfaces. It measures the bit-rate of the IUT output signal when in State F3 or G4.

The stimulus consists of interrupting the signal at the T reference point and measuring to see that the output signal from the IUT is 2048 kbit/s ± 50ppm (or 32ppm depending on the intended application).

To meet this specification, an IUT for the I_a interface should switch to an independent transmit clock reference when it enters State F3. Provided the hardware connections of Figure 1 and the state matrix of Table 2 are implemented, the IUT will pass this test.

Of course, the independent timing source applied to the TCLKI[x] input, or BTCLK[x] of the EQUAD must have the specified frequency tolerance or better.

7.C.2.2 Received and Transmitted Line Code

These tests check that the IUT can decode and encode the HDB3 line code.

7.C.2.2.1 RECEIVED LINE CODE

This test is covered by test C.4.5.

7.C.2.2.2 TRANSMITTED LINE CODE

This test is covered by test C.3.1.1.

7.C.2.3 Specifications At The Output Ports

These tests check that the characteristics of the output pulses meet the G.703 specifications.

7.C.2.3.1 PULSE SHAPE AND AMPLITUDE OF A PULSE

This test applies for both I_a and I_b interfaces. It checks the conformance of the shape of all mark pulses, irrespective of polarity, transmitted by the IUT.

The characteristics of the EQUAD do not directly affect compliance to this test. Rather the LIU characteristics are relevant.

7.C.2.3.2 PEAK VOLTAGE OF A SPACE

This test applies for both I_a and I_b interfaces. It checks that the amplitude of transmitted spaces (logic zeros) from the output port do not exceed 0.30V.

The characteristics of the EQUAD do not directly affect compliance to this test. Rather the LIU characteristics are relevant.

7.C.2.3.3 RATIO OF THE AMPLITUDES OF POSITIVE AND NEGATIVE PULSES

This test applies for both I_a and I_b interfaces. It checks the balance between the amplitude of positive and negative pulses transmitted by the IUT.

The characteristics of the EQUAD do not directly affect compliance to this test. Rather the LIU characteristics are relevant.

7.C.2.3.4 RATIO OF THE WIDTHS OF POSITIVE AND NEGATIVE PULSES

This test applies for both I_a and I_b interfaces. It checks the balance between the width of positive and negative pulses transmitted by the IUT.

The characteristics of the EQUAD do not directly affect compliance to this test. Rather the LIU characteristics are relevant.

7.C.2.4 Specifications At The Input Ports

These tests check that the input port meets G.703 requirements.

7.C.2.4.1 RETURN LOSS

This test applies to both the I_a and I_b interfaces. It checks the return loss of the input port.

The characteristics of the EQUAD do not directly affect compliance to this test. Rather the LIU characteristics are relevant.

7.C.2.4.2 IMMUNITY AGAINST REFLECTIONS

This test applies to both the I_a and I_b interfaces. It checks the input port's immunity against an interfering signal combined with a cable attenuation of maximum 6dB (of cable loss).

The characteristics of the EQUAD do not directly affect compliance to this test. Rather the LIU characteristics are relevant.

7.C.2.5 Frame Structure

These tests check that the frame structure of the 2048 kbit/s signal transmitted from the IUT conforms with the requirements in ITU-T G.704.

7.C.2.5.1 NUMBER OF BITS PER TIMESLOT

This test cannot be verified with Layer 1 tests.

7.C.2.5.2 NUMBER OF TIMESLOTS PER FRAME

This test cannot be verified with Layer 1 tests.

7.C.2.5.3 ASSIGNMENT OF BITS IN TIMESLOT 0

These tests check that the IUT generates the TS0 as specified in ITU-T G.704.

7.C.2.5.3 Generation Of Frame Alignment Word

This test is applicable to both I_a and I_b interfaces. It checks that the FAS and MFAS are correctly generated, and the CRC-4 correctly calculated in the Timeslot 0 of the frames transmitted by the IUT.

In the PM6344 EQUAD, this feature is guaranteed by design. Additionally, FAS and MFAS generation was verified by PMC-Sierra's Product Verification group during feature testing — accomplished by connecting the EQUAD output to the receiver of a standard E1 test unit, while monitoring the framing errors received by the test unit.

For the initial FTP, a TTC (Telecommunications Techniques) FireBERd 6000 with the G.704 Interface was used. The TTC reported no FAS, MFAS or CRC errors during measurement intervals of no less than one minute. Since the initial FTP, the PMC-Sierra Applications Support Group has tested the FAS, MFAS and CRC generation with the

Wandel&Goltermann PA-20 E1/CEPT test unit, as well as the Adtech AX4000 with G.703 (E1) interface.

7.C.2.5.3.2 Sa Bits

Since no specific use is defined for the Sa bits, no test is prescribed.

7.C.2.6 Timeslot Assignment

This requirement cannot be verified via Layer 1 procedures.

7.C.2.7 Timing Considerations

These tests check the IUT's ability to synchronize its timing and to minimize output jitter.

7.C.2.7.1 AIS RECOGNITION

This test applies to the I_a interface. It checks to see that the IUT can detect received AIS. The expected response is that the IUT transmits a continuous RAI.

In the PM6344 EQUAD, the conformance of the AIS detection is guaranteed by design.

PMC-Sierra's Product Verification group verified the EQUAD'S AIS detection during feature testing. The Wandel&Goltermann PA-20 E1/CEPT test unit was used to transmit AIS, with and without bit errors, to stress the EQUAD'S AIS detection algorithm.

In the EQUAD, the AISD bit in Register 27H indicates AIS detection. Another bit, the AIS bit in Register 27H, indicates an integrated version of the AISD status. The integration algorithm provides a 99.1% probability of declaring AIS within 104ms in the presence of an $1:10^{-3}$ mean BER. Of course there is a 100% probability of declaring AIS within 104ms when no bit errors are present. The 104ms integration interval is compatible with the ITU-T Q.251 recommendation.

The C.2.7.1 test monitors to see that a continuous RAI is sent in response to the AIS stimulus. Therefore, the IUT must set the EQUAD'S REMAIS bit in Register 45H when AIS is detected (either by polling the AIS bit or by enabling the AISI interrupt).

An IUT properly implementing the states matrix in Table 2 will pass this test.

7.C.2.7.2 SYNCHRONIZATION

This test applies to the I_a interface. It checks the IUT's ability to synchronize its transmit timing to the timing of the signal received at the input port.

This test tests two things: first, that the line timing can be properly recovered; second, that the recovered timing is looped back to the transmit (loop-timed mode).

In the EQUAD, the CDRC functional block recovers the line timing. The transmit timing is locked to the recovered timing by setting the PLLREF[1:0] bits in Register 07H to 10B (refer to the Timing Connections section). In that configuration, the recovered timing is further passed through the DJAT functional block of the EQUAD.

The ability to lock the transmit timing to the receive timing is guaranteed by design in the EQUAD, provided the recommendations in the Timing Connections section are followed.

The EQUAD'S ability to synchronize its transmit timing to its recovered timing was verified by PMC-Sierra's Product Verification group during feature testing. The Wandel&Goltermann PA-20 E1/CEPT test unit was used with the frequency sourced from a Marconi 2022D Signal Generator. The linear tracking range was verified at greater than ±624ppm.

7.C.2.8 Jitter

These tests check that the IUT handles phase jitter as specified in I.431.

7.C.2.8.1 MINIMUM TOLERANCE TO JITTER AND WANDER

This test applies to both the I_a and I_b interfaces. It checks the IUT's tolerance to sinusoidal phase jitter on the incoming 2048 kbit/s signal. The jitter tolerance of the IUT's input port must exceed the mask given in Figure 10 of I.431 (which is based on G.823).

The characteristics of the EQUAD do not directly affect compliance to this test. Rather the LIU characteristics are relevant.

7.C.2.8.2 OUTPUT JITTER

These tests check that the characteristics of the jitter from the IUT's output port complies with I.431 requirements.

7.C.2.8.2.1 Output Jitter With Jitter At The Input Port Supplying Timing

This test applies only to the I_a interface. This test measures the jitter generated from the IUT in the presence of input jitter when the IUT is recovering timing from the input port.

The characteristics of the EQUAD do not directly affect compliance to this test. Rather the LIU characteristics are relevant.

7.C.2.8.2.2 Output Jitter At Network Side

This test applies only to the I_b interface. This test measures the jitter generated from the IUT, when referenced to a jitter-less timing source.

The characteristics of the EQUAD do not directly affect compliance to this test. Rather the LIU characteristics are relevant.

7.C.2.9 Tolerable Longitudinal Voltage

This test applies to both the I_a and I_b interface. It tests the minimum tolerance to longitudinal (common mode) voltages at the input ports.

The characteristics of the EQUAD do not directly affect compliance to this test. Rather, the line interface circuitry, especially the receive transformer, must have the required characteristics. Contact the transformer manufacturer for information on how to meet this requirement.

7.C.2.10 Output Signal Balance

This test is covered by specifications covering EMC requirements, and is not specified in ETS 300 011.

7.C.2.11 Impedance Towards Ground

These tests check that the impedance measured toward ground of the input and output ports of the IUT conform to ITU-T I.431.

7.C.2.11.1 IMPEDANCE TOWARDS GROUND OF THE RECEIVER

This test applies to both the I_a and I_b interface. It tests the impedance to ground of the IUT's input port.

The characteristics of the EQUAD do not directly affect compliance to this test. Rather, the line interface circuitry, especially, the receive transformer, must have the required characteristics. Contact your transformer manufacturer for more information on how to meet this requirement.

7.C.2.11.2 IMPEDANCE TOWARDS GROUND OF THE TRANSMITTER

This test applies to both the I_a and I_b interface. It tests the impedance to ground of the IUT's output port.

The characteristics of the EQUAD do not directly affect compliance to this test. Rather, the line interface circuitry, especially, the transmit transformer, must have the required

characteristics. Contact your transformer manufacturer for more information on how to meet this requirement.

7.C.3 Functional Characteristics Tests

These tests check that the IUT meets the I.431 specifications for functional characteristics of the interface.

7.C.3.1 Test Of Signals Sent By IUT

These tests check that the signals sent by the IUT conform with I.431.

7.C.3.1.1 HDB3 CODING AND NORMAL OPERATIONAL FRAME

This test applies to both the I_a and I_b interface. It checks the coding, decoding and binary organization of the NOF.

NOFs are sent from the simulator. A Payload Loopback is activated in the IUT, and the returned NOFs are monitored for correct HDB3 coding (as defined in G.703), RAI, FEBE, and CRC-4 block errors.

In the PM6344 EQUAD, HDB3 encoding and decoding is guaranteed by design.

PMC-Sierra's Product Verification group verified the EQUAD'S HDB3 encoding/decoding algorithm during feature testing. A TTC (Telecommunications Techniques) FireBERd 6000 with the G.704 Interface was used. With the EQUAD in Payload Loopback, the FireBERd reported no line code violations during measurement intervals of no less than one minute. The returned NOFs were monitored by the FireBERd that reported no errors.

7.C.3.1.2 REMOTE ALARM INDICATION

This test is combined with C.3.2. An IUT that properly implements I.431 states matrix based on Tables 2 and 3 will pass this test.

7.C.3.1.3 ALARM INDICATION SIGNAL

This test is combined with C.3.2. An IUT that properly implements I.431 states matrix based on Tables 2 and 3 will pass this test.

7.C.3.1.4 CRC ERROR INFORMATION

This test is combined with C.4.5.

7.C.3.1.5 RAI AND CONTINUOUS CRC ERROR INDICATION

This test is combined with C.3.2. An IUT that properly implements I.431 states matrix based on Tables 2 and 3 will pass this test.

7.C.3.2 States-Matrix At The IUT

These tests check that the IUT properly implements the I.431 states matrix appropriate to the interface.

7.C.3.2.1 STATES-MATRIX AT THE IUT NETWORK SIDE

This test applies to the I_b interface. It tests that the IUT properly transitions between the stable states, G0 to G5, defined in I.431.

For state transitions that pertain directly to the functionality of the EQUAD, all the necessary status and control registers are provided for implementing the I.431 states matrix. Table 3 explains how the EQUAD registers should be used to implement the states matrix for the network side. Provided those recommendations are followed, the IUT should have no difficulty passing this test.

7.C.3.2.2 STATES-MATRIX AT THE IUT USER SIDE

This test applies to the I_a interface. It tests that the IUT properly transitions between the stable states, F0 to F5, defined in I.431.

The EQUAD provides all the status and control registers necessary for implementing the I.431 states matrix. Table 2 explains how the EQUAD registers should be used to implement the states matrix for the user side. Provided those recommendations are followed, the IUT should have no difficulty passing this test.

7.C.4 Interface Procedures Tests

These tests check that the IUT properly implements the interface procedures defined in I.431 and G.706.

7.C.4.1 Codes For Idle Channels And Idle Slots

This test applies to both the I_a and I_b interfaces. It checks the pattern in timeslots that are not assigned to a channel.

The test monitors each timeslot to ensure that at least three binary ONEs are present in each timeslot.

The EQUAD provides a serial controller in the transmit path called the TPSC. This functional block contains registers (accessed indirectly via Registers 30H to 33H) which provide per-channel control of the timeslot data. Each timeslot can either be passed transparently from the backplane, or overwritten with an "Idle" code (the Idle codes for each timeslot are independent from each other).

The TPSC should be initialized so that each channel's Idle code meets the requirement of at least three binary ONEs. Whether the EQUAD overwrites the timeslot with the Idle code is controlled by the SUBS and DS[1:0] bits in each channel's Data Control byte within the TPSC.

The use of the TPSC functional block is described in the EQUAD data book, in the section entitled "Using the Per-Channel Serial Controllers" (pp. 183-4 in Issue 6).

7.C.4.2 Interframe (Layer 2) Time Fill

This test applies to both the I_a and I_b interfaces. It checks that the HDLC Idle signal on the D-Channel consists of continuous Flags (01111110).

In the PM6344 EQUAD, HDLC Idle generation is guaranteed by design when using the internal HDLC serial controller. When using an external HDLC serial controller, it is the responsibility of that device to comply with this test.

The ability of the EQUAD'S internal HDLC controller to transmit Idle code was verified by PMC-Sierra's Product Verification group during feature testing. A logic analyzer was used to monitor the output of the HDLC controller to check that it properly transmitted the HDLC Idle signal.

To process the D-Channel, the EQUAD must be properly configured. There are two options for processing the D-Channel in the EQUAD: using the internal HDLC controllers (RFDL and XFDL), or extracting/inserting the D-Channel to/from external serial pins (RDLSIG, RDLCLK, TDLSIG, and TDLCLK).

Note: When using the internal HDLC controllers, the packet payloads can be processed either via the microprocessor, or by an external DMA (Direct Memory Access) controller connected to the datalink interrupt pins (RDLINT, RDLEOM, TDLINT, and TDLUDR).

The EQUAD can either process the D-Channel (TS16) or any combination of the National Use Bits in TS0 as an HDLC datalink. To select the D-Channel, the SIGEN bit and DLEN bit in Register 44H must be set to logic zero and one, respectively. Additionally, the RXSAEN[8:4] bits in Register 09H must all be cleared to logic zero.

To select whether the datalink is processed internally or externally to the EQUAD, the RXDMASIG and TXDMASIG bits in Register 02H must be set accordingly.

7.C.4.3 Frame Alignment (Without the Test Of CRC Procedure)

This test applies to both the I_a and I_b interface. It checks that the IUT correctly executes the FAS alignment procedures.

This test consists of a lengthy sequence of framing stimuli that stress the framing algorithm of the IUT. The signal from the IUT is monitored for presence or absence of RAI as appropriate.

The EQUAD can be configured to automatically find and lose frame in accordance with G.706. To do this, the FASC bit in Register 21H should be a logic zero, and the BIT2C bit in the same register should be set to logic one. This setting allows OOF to be declared if either three consecutive FASs are received in error, or if bit 2 in the TS0 of three consecutive NFAS frames is received in error.

The EQUAD does not automatically transmit RAI in response to OOF. Therefore, the control of RAI transmission must be controlled by the microprocessor, which would set or clear the REMAIS bit in Register 45H as required.

The software implementation recommended in Appendix A will meet this test.

7.C.4.4 CRC Multiframe Alignment

This test applies to both the I_a and I_b interface. It checks that the IUT correctly responds to the ETS 300 011 test, section C.4.4.

Note: The original version of the test stimulus for this test was poorly conceived. Therefore, ESTI issued Amendment 1 to EST 300 011 containing a revised stimulus. Equipment designed using the EQUAD should be tested to the stimulus contained in Amendment 1.

The software controlling the EQUAD must be carefully crafted to meet this specification, which calls for a timer with 100 millisecond resolution. Appendix A of this document details an example interrupt handling routine which responds to the EQUAD'S framer interrupts such that the IUT complies with this test.

7.C.4.5 CRC Processing

This test applies to both the I_a and I_b interface. It checks that the IUT correctly executes the CRC calculation and generation of the FEBE indications (in the E-Bits of the CMF structure) in response to CRC errors.

In the EQUAD, the FEBEs are automatically sent in response to detected CRC errors. The proper operation of this feature is guaranteed by design.

The ability of the EQUAD to transmit FEBE indications was verified by PMC-Sierra's Product Verification group during feature testing. The return of FEBEs was monitored in response to both deterministic and probabilistic CRC errors.

7.C.5 Power Feeding

These tests check that the power supply of the IUT meets the I.431 requirements.

7.C.5.1 Provision Of Power And Feeding Voltage

This test applies only to the I_a interface. It tests the provision of power by the IUT.

The characteristics of the EQUAD do not directly affect compliance with this test. Contact the manufacturer of the power supply components for information on meeting this requirement.

7.C.5.2 Protection Against Short Circuit

This test applies only to the I_a interface. It tests the ability of the power source to withstand a short circuit condition.

The characteristics of the EQUAD do not directly affect compliance with this test. Contact the manufacturer of the power supply components for information on meeting this requirement.

7.C.5.3 Protection Against Overload

This test applies only to the I_a interface. It tests the ability of the power source to withstand an overload condition.

The characteristics of the EQUAD do not directly affect compliance with this test. Contact the manufacturer of the power supply components for information on meeting this requirement.

7.C.5.4 Power Consumption And Interchange Of Wires

This test applies only to the I_b interface. It tests the consumption of power by the IUT and that checks that no damage occurs due to the interchange of power feeding wires.

The characteristics of the EQUAD do not directly affect compliance with this test. Contact the manufacturer of the power supply components for information on meeting this requirement.

APPENDIX A RECOMMENDED FRAMING SOFTWARE

This section details the PMC-Sierra's suggestions for implementing an interrupt-handling routine to handle the transient states related to FAS and MFAS alignment procedures.

The suggested interrupt-handling routine is given in Section A.2. This routine or one similar must be implemented to pass the ETS 300 011 tests.

The routine (called EQUAD_sr_2_0.c) is written in ANSI C. EQUAD_sr_2_0.c is complete with header file EQUAD.h. This header file is designed to be generic for any future interrupt subroutines for the EQUAD. The routine was originally written in 6811 assembler code. The ANSI C version was derived from this code. The 6811 assembler version follows the C version.

A.1 Implementation Assumptions

The EQUAD_SR routine given in the next subsection assumes the following:

- the EQUAD has been initialized as recommended in the Initialization section.
- there is a resettable timer with millisecond resolution available. The value of time elapsed since the last timer reset is passed through the global variable timer_count. This timer is independent of timers T1 and T2 used for the I.431 states matrix.
- the status register 24H should only be read *once* upon entering the EQUAD_sr_2_0.c routine. This is because this register contains bits that are cleared upon read. In order to process the status information contained in this register, the value it contains should be stored in a local variable.
- the status register 21H should only be read where indicated in the routine. This register contains bits that are cleared upon read. Therefore, the value it contains should be stored in a local variable for processing.
- the EQUAD_sr_2_0.c calls upon three other subsections which are not explicitly detailed here because their implementation consists of single register writes to the EQUAD:

  NOF is a constant that clears the RAI indication in the A-bit in Timeslot 0 of the E1 basic frame. This is accomplished by clearing the REMAIS bit in Register 45H to logic zero.

RAI is a constant that sets the RAI indication in the A bit in Timeslot 0 of the E1 basic frame. This is accomplished by setting the REMAIS bit in Register 45H to logic one.

The timer reset subsection clears the timer variable timer_count used to measure the 400ms interval compatible with G.706.

- three separate functions are called in EQUAD_sr_2_0.c:

  COMPARE() is a function that is passed the value of a specific register and a mask value. COMPARE() then returns TRUE if the specific mask bit in the register is set or FALSE if the bit is cleared.

  CKECK() is a function which is passed the value of the event or state variables and also the event or state number which is being checked for. The function returns TRUE if the event or state is valid, FALSE if it is not.

  WHAT_STATE() is the function in which the determined event and the initial state variables are passed. These variables are then checked and the proper parameters are set. The resultant state is then returned to the main subroutine and updated.

This appendix is only concerned with the software for handling the transient framing states. For an implementation compliant to ETS 300 011, additional software as recommended in the section entitled "Implementing the I.431 States Matrix" must also be active.

A.2 Example Interrupt-Handling Routine

```
/*
*****************************************************************************
* Copyright (c) 1998 PMC Sierra, Inc.
* All rights reserved
*
* Name: equadd.h Version: 1.0 Date: 25 May 1998
* Product: equad  Subsystem: framer
*
* Author: Korby Mraze
*
* Description: Header file for frame/multiframe interrupt service routine used
* to examine states and events to decide which state to exit in.
* Based on parameters as defined in the Application Note for the
* PM6344 EQUAD, Document Number PM970239 Issue #2.
*
*****************************************************************************
*/
#define        TRUE           1              /* Constant logic 1 */
#define        FALSE          0              /* Constant logic 0 */

#define        EVENT1         0x01           /* Event 1 */
#define        EVENT2         0x02           /* Event 2 */
#define        EVENT3         0x03           /* Event 3 */
#define        EVENT4         0x04           /* Event 4 */
#define        EVENT5         0x05           /* Event 5 */

#define        RAI            0x08           /* Set RAI */
#define        NOF            0x00           /* No RAI */
#define        TIMER_LIMIT 0x25              /* Maximum timer limit */
#define        CLEAR          0x00           /* Clears all bits */

#define        OOFI           0x40           /* Mask for bit 6 of register 024H */
#define        OOF            0x40           /* Mask for bit 6 of register 026H*/
#define        OOCMFI         0x10           /* Mask for bit 4 of register 024H*/
#define        OOCMF          0x10           /* Mask for bit 4 of register 026H*/
#define        CMFAC          0x02           /* Mask for bit 1 of register 021H*/
#define        FRMR_SA        0x20           /* Mask for bit 5 of register 008H*/

#define        STATE1         0x01           /* State 1 */
#define        STATE2         0x02           /* State 2 */
#define        STATE3         0x03           /* State 3 */
#define        STATE4         0x04           /* State 4 */
#define        STATE5         0x05           /* State 5 */

typedef unsigned char  REG_OFFSET;           /* 1 byte for register offset */
typedef unsigned char  REG_DATA;             /* 1 byte for register data */
typedef unsigned char  MAN_VAR;              /* 1 byte for management variables */
typedef unsigned char  MASK;                 /* 1 byte for mask variable */
typedef unsigned char  BOOLEAN;              /* 1 byte for boolean variable */
typedef unsigned char  EVENT;                /* 1 byte for event number */

typedef struct EQUADa_QUAD {                 /* Stucture definition for a quadrant of */
                                             /* the Equad device */

       REG_DATA       RX_OPT;               /* Offset register 000H of the Equad */
       REG_DATA       RX_BP_OPT;            /* Offset register 001H of the Equad */
       REG_DATA       DATALINK_OPT;         /* Offset register 002H of the Equad */
       REG_DATA       RX_IF_CF;             /* Offset register 003H of the Equad */
       REG_DATA       TX_IF_CF;             /* Offset register 004H of the Equad */
       REG_DATA       TX_BP_OPT;            /* Offset register 005H of the Equad */
       REG_DATA       TX_FRM_OPT;           /* Offset register 006H of the Equad */
       REG_DATA       TX_TIM_OPT;           /* Offset register 007H of the Equad */
       REG_DATA       INT_SOURCE;           /* Offset register 008H of the Equad */
       REG_DATA       RX_DATALINK_EN;       /* Offset register 009H of the Equad */
       REG_DATA       DIAG;                 /* Offset register 00AH of the Equad */
       REG_DATA       TEST_RESV0;           /* Offset register 00BH of the Equad */
       REG_DATA       PMON_UPDATE_RESV1;    /* Offset register 00CH of the Equad */
       REG_DATA       RESET;                /* Offset register 00DH of the Equad */
       REG_DATA       PHASE_STAT_LSB;       /* Offset register 00EH of the Equad */
       REG_DATA       PHASE_STAT_MSB;       /* Offset register 00FH of the Equad */
       REG_DATA       CDRC_CF;              /* Offset register 010H of the Equad */
```

```
        REG_DATA        CDRC_INT_EN;            /* Offset register 011H of the Equad */
        REG_DATA        CDRC_INT_STAT;          /* Offset register 012H of the Equad */
        REG_DATA        ALT_LOS;                /* Offset register 013H of the Equad */
        REG_DATA        CHAN_SEL_0;             /* Offset register 014H of the Equad */
        REG_DATA        CHAN_SEL_1;             /* Offset register 015H of the Equad */
        REG_DATA        CHAN_SEL_2;             /* Offset register 016H of the Equad */
        REG_DATA        CHAN_SEL_3;             /* Offset register 017H of the Equad */
        REG_DATA        DJAT_INT_STAT;          /* Offset register 018H of the Equad */
        REG_DATA        DJAT_N1;                /* Offset register 019H of the Equad */
        REG_DATA        DJAT_N2;                /* Offset register 01AH of the Equad */
        REG_DATA        DJAT_CF;                /* Offset register 01BH of the Equad */
        REG_DATA        ELST_CF;                /* Offset register 01CH of the Equad */
        REG_DATA        ELST_INT_STAT;          /* Offset register 01DH of the Equad */
        REG_DATA        ELST_IDLE;              /* Offset register 01EH of the Equad */
        REG_DATA        RESERVED2;              /* Offset register 01FH of the Equad */
        REG_DATA        FRMR_ALIGN_OPT;         /* Offset register 020H of the Equad */
        REG_DATA        FRMR_MAINT_OPT;         /* Offset register 021H of the Equad */
        REG_DATA        FRMR_FRM_INT_EN;        /* Offset register 022H of the Equad */
        REG_DATA        FRMR_MAINT_INT_EN;      /* Offset register 023H of the Equad */
        REG_DATA        FRMR_FRM_INT;           /* Offset register 024H of the Equad */
        REG_DATA        FRMR_MAINT_INT;         /* Offset register 025H of the Equad */
        REG_DATA        FRMR_FRM_STAT;          /* Offset register 026H of the Equad */
        REG_DATA        FRMR_MAINT_STAT;        /* Offset register 027H of the Equad */
        REG_DATA        FRMR_NATIONAL;          /* Offset register 028H of the Equad */
        REG_DATA        FRMR_EXTRA;             /* Offset register 029H of the Equad */
        REG_DATA        FRMR_CRC_ERR_LSB;       /* Offset register 02AH of the Equad */
        REG_DATA        FRMR_CRC_ERR_MSB;       /* Offset register 02BH of the Equad */
        REG_DATA        TS16_AIS_STAT;          /* Offset register 02CH of the Equad */
        REG_DATA        RESERVED3;              /* Offset register 02DH of the Equad */
        REG_DATA        RESERVED4;              /* Offset register 02EH of the Equad */
        REG_DATA        RESERVED5;              /* Offset register 02FH of the Equad */
        REG_DATA        TPSC_CF;                /* Offset register 030H of the Equad */
        REG_DATA        TPSC_ACCESS_STAT;       /* Offset register 031H of the Equad */
        REG_DATA        TPSC_IND_ADDR;          /* Offset register 032H of the Equad */
        REG_DATA        TPSC_IND_DATA;          /* Offset register 033H of the Equad */
        REG_DATA        XFDL_CF;                /* Offset register 034H of the Equad */
        REG_DATA        XFDL_INT_STAT;          /* Offset register 035H of the Equad */
        REG_DATA        XFDL_TX_DATA;           /* Offset register 036H of the Equad */
        REG_DATA        RESERVED6;              /* Offset register 037H of the Equad */
        REG_DATA        RFDL_CF;                /* Offset register 038H of the Equad */
        REG_DATA        RFDL_INT_STAT;          /* Offset register 039H of the Equad */
        REG_DATA        RFDL_STAT;              /* Offset register 03AH of the Equad */
        REG_DATA        RFDL_RX_DATA;           /* Offset register 03BH of the Equad */
        REG_DATA        INT_ID;                 /* Offset register 03CH of the Equad */
        REG_DATA        BP_PARITY;              /* Offset register 03DH of the Equad */
        REG_DATA        RESERVED7;              /* Offset register 03EH of the Equad */
        REG_DATA        RESERVED8;              /* Offset register 03FH of the Equad */
        REG_DATA        SIGX_CF;                /* Offset register 040H of the Equad */
        REG_DATA        SIGX_ACCESS_STAT;       /* Offset register 041H of the Equad */
        REG_DATA        SIGX_IND_ADDR;          /* Offset register 042H of the Equad */
        REG_DATA        SIGX_IND_DATA;          /* Offset register 043H of the Equad */
        REG_DATA        TRAN_CF;                /* Offset register 044H of the Equad */
        REG_DATA        TRAN_DIAG;              /* Offset register 045H of the Equad */
        REG_DATA        TRAN_NATIONAL;          /* Offset register 046H of the Equad */
        REG_DATA        TRAN_EXTRA;             /* Offset register 047H of the Equad */
        REG_DATA        PMON_STAT;              /* Offset register 048H of the Equad */
        REG_DATA        PMON_FER;               /* Offset register 049H of the Equad */
        REG_DATA        PMON_FEBE_LSB;          /* Offset register 04AH of the Equad */
        REG_DATA        PMON_FEBE_MSB;          /* Offset register 04BH of the Equad */
        REG_DATA        PMON_CRC_LSB;           /* Offset register 04CH of the Equad */
        REG_DATA        PMON_CRC_MSB;           /* Offset register 04DH of the Equad */
        REG_DATA        PMON_LCV_LSB;           /* Offset register 04EH of the Equad */
        REG_DATA        PMON_LCV_MSB;           /* Offset register 04FH of the Equad */

}REGISTER;

typedef struct EQUADa_MEM {

        REG_DATA        FRAME;                  /* local copy of equad reg 24H */
        REG_DATA        STATUS;                 /* local copy of equad reg 26H */
        REG_DATA        CMFACT;                 /* local copy of equad reg 21H */
        REG_DATA        STATE;                  /* current transient state */
        REG_DATA        TIMERX;                 /* Boolean -> 1 if timer has expired */
        REG_DATA        T3;                     /* T3 as defined in PM970239 */
        REG_DATA        FSTATE;                 /* Current FSTATE */
        REG_DATA        UPH;                    /* User PH */
```

```
        REG_DATA        UMPH;                   /* User MPH */
        REG_DATA        INT_STAT;               /* local copy of equad reg 25H */
        REG_DATA        ALARM_STAT;             /* local copy of equad reg 27H */
        REG_DATA        LOS;                    /* local copy of equad reg 12H */

}MEMORY;

BOOLEAN                                         /* Prototype for function COMPARE */
COMPARE(REG_DATA, MASK);

BOOLEAN                                         /* Prototype for function CHECK */
CHECK (REG_DATA, MASK);

REG_DATA                                        /* Prototype for function WHAT_STATE */
WHAT_STATE(MAN_VAR , REG_DATA ); /*
```

```
/*
 ******************************************************************************
 *        Copyright (c) 1998  PMC Sierra, Inc.
 *                         All rights reserved
 *
 * Name:    equadd_sr.c       Version: 2.0   Date: 25 May 1998
 * Product: equad    Subsystem: framer
 *
 * Author:      Korby Mraze
 *
 * Description: Frame/multiframe interrupt service routine used to examine
 *              states and events to decide which state to exit in.
 *              Based on parameters defined in the Applicaton Note for the
 *              PM6344 EQUAD, Document Number PM970239 Issue #2.
 *
 ******************************************************************************
 */


#include "equadd.h"     /* Header file with typedefs and function prototypes */

/*
 ******************************************************************************
 *
 * Global memory initializations from Fourth setup
 *
 *
 ******************************************************************************
*/
REGISTER *quad[4];                    /* 4 equada device structures */
MEMORY  *mem[4];                      /* 4 equada memory structures */
REG_DATA state, cmfact, tadc, timer_expired, timer_count, align;
REG_DATA frame_status;           /* framer framing status, offset reg 026H */
REG_DATA tran_cfg;               /* EQUADa TRAN configuration offset register 044H */
REG_DATA tran_intl;              /* EQUADa TRAN Internation/National bits offset register 46H */
int quad_num;                         /* variable used to determine which quandrant to use */

/*
 ******************************************************************************
 *
 * Main function call
 *
 ******************************************************************************
*/

void main(void){

/*
 ******************************************************************************
 *
 *  Program variable declarations and initializations
 *
 ******************************************************************************
*/

REG_DATA interrupt_id;           /* Interrupt id, offset reg 03CH */
REG_DATA int_source;             /* Interrupt source, offset reg 008H */
REG_DATA frm_main_opt;           /* framer maintenance options, offset reg 021H */
REG_DATA frame_int;              /* framer framing status interrup indication, offset reg 024H */
REG_DATA los_int;                /* EQUADa LOS interrupt indication offset register 12H */

MAN_VAR event;                   /* variable used to pass the event to the WHAT_STATE func */

quad[0] = ( struct EQUADa_QUAD* ) 0xc000;             /* Base address for quadrant 0 */
quad[1] = ( struct EQUADa_QUAD* ) 0xc080;             /* Base address for quadrant 1 */
quad[2] = ( struct EQUADa_QUAD* ) 0xc100;             /* Base address for quadrant 2 */
quad[3] = ( struct EQUADa_QUAD* ) 0xc180;             /* Base address for quadrant 3 */

mem[0] = ( struct EQUADa_MEM* ) 0x1200;             /* Base address for quadrant 0 mem */
mem[1] = ( struct EQUADa_MEM* ) 0x1210;             /* Base address for quadrant 1 mem */
mem[2] = ( struct EQUADa_MEM* ) 0x1220;             /* Base address for quadrant 2 mem */
mem[3] = ( struct EQUADa_MEM* ) 0x1230;             /* Base address for quadrant 3 mem */

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

```
 * Start main program
 *
 ******************************************************************************
*/


interrupt_id = (*quad[0]).INT_ID;                 /* Load interrupt ID register */

/*
 ******************************************************************************
 *
 * Determine which quadrant has triggered an interrupt
 *
 ******************************************************************************
*/

if(COMPARE(interrupt_id, 0x10)){
  quad_num = 0;
}                                        /* Find the first quadrant with an interrupt */
else if(COMPARE(interrupt_id, 0x20)){
  quad_num = 1;
}
else if(COMPARE(interrupt_id, 0x40)){
  quad_num = 2;
}
else
  quad_num = 3;

/*
 ******************************************************************************
 *
 * Check for LOS interrupt
 *
 ******************************************************************************
*/
los_int = (*quad[quad_num]).CDRC_INT_STAT;      /* load LOS interrupt register */

if(!COMPARE(los_int, 0x40)){                    /* Check for LOSI */
                                                /* LOSI was zero, start code */
/*
 ******************************************************************************
 *
 * Check for FRMR interrupt
 *
 ******************************************************************************
*/

  int_source = (*quad[quad_num]).INT_SOURCE;          /* Load Interrupt Source register */

  if(COMPARE(int_source, FRMR_SA)){                   /* If a FRMR interrupt */
    frame_int = (*quad[quad_num]).FRMR_FRM_INT;       /* Get rest of register values */
    frame_status = (*quad[quad_num]).FRMR_FRM_STAT;   /*from quadrant in question*/
    frm_main_opt = (*quad[quad_num]).FRMR_MAINT_OPT;
    state = (*mem[quad_num]).STATE;
    cmfact = (*mem[quad_num]).CMFACT;
    timer_expired = (*mem[quad_num]).TIMERX;
    timer_count = (*mem[quad_num]).T3;

/*
 ******************************************************************************
 *
 * Determine the event according to the status registers, then call WHAT_STATE
 * function to determine the next state based on the current state and the event.
 *
 ******************************************************************************
*/

    /* Handles events 1, 2, and 3 */

    if (!COMPARE(frame_int, OOFI)){         /* OOFI was 0, check OOCMFI */
      if (!COMPARE(frame_int, OOCMFI)){     /* OOCMFI was 0, goto EVENT0 */
        ;                                   /* Do nothing */
      }
      else {                                /* OOCMFI was 1, this is EVENT3 */
        event = EVENT3;
        state = WHAT_STATE(event, state);/* Determine state and set up new state */
```

```
      }
    }
    else {                                  /* OOFI was 1, check OOF */
      if (!COMPARE(frame_status, OOF)){     /* OOF was 0, goto EVENT2 */
        event = EVENT2;
        state = WHAT_STATE(event, state);   /* Determine state and set up new state */
      }
      else {                                /* OOF was 1, check CMFACT */
        if (!COMPARE(frm_main_opt, cmfact)){/* CMFACT was 0, goto EVENT1 */
          event = EVENT1;
          state = WHAT_STATE(event, state); /* Determine state and set up new state */
        }
        else {                              /* CMFACT was 1, check if timer has expired */
          if(timer_expired){                /* If timer has expired goto EVENT4 */
            event = EVENT4;
        state = WHAT_STATE(event, state);
          }
          else {                            /* Timer has not expired so increment and see */
            ++(timer_count);                    /* if it is about to expire */
            if ( timer_count <= TIMER_LIMIT ) { /* it hasn't, goto EVENT5 */
              event = EVENT5;
          state = WHAT_STATE(event, state);
            }
            else {                              /* it has, set timerx to 1 */
              timer_expired = TRUE;
            }
          }
        }
      }
    }

    (*quad[quad_num]).TRAN_DIAG = tadc;        /* Update the TADC register */
    (*quad[quad_num]).FRMR_FRM_STAT = frame_status;
    (*quad[quad_num]).TRAN_CF = tran_cfg;
    (*quad[quad_num]).TRAN_NATIONAL = tran_intl;
    (*mem[quad_num]).STATE = state;
    (*mem[quad_num]).CMFACT = cmfact;
    (*mem[quad_num]).TIMERX = timer_expired;
    (*mem[quad_num]).T3 = timer_count;


  }
  else                    /* Not a valid framer interrupt, exit routine */
  ;
}
else if(!COMPARE(los_int, 0x01)){           /* Check if LOS is set */
  ;                                         /* LOS was 0, end routine */
  }
  else {                                      /* LOS was 1, do following then leave */
    (*mem[quad_num]).STATE = STATE1;          /* Set state to 1 */
    (*quad[quad_num]).TRAN_DIAG = RAI;        /* Assert RAI */
  }
} /*end of program */


/*
 ****************************************************************************
 *
 * Function definitions section
 *
 ****************************************************************************
*/


BOOLEAN                                 /* Function to determine if a specific bit is set */
COMPARE (REG_DATA reg, MASK mask) {

   return ((reg & mask)!= FALSE);

}

BOOLEAN                                 /* Function to check the value of the variable */
CHECK (REG_DATA reg, MASK mask) {

  return ( reg == mask );
```

```
}

                /* Function to determine the state and event then proceeding */
REG_DATA                          /* to set the appropriate parameters */
WHAT_STATE(MAN_VAR fevent, REG_DATA fstate){


  if(!CHECK(fstate, STATE1)){       /* Check if in state 1 */
    if(!CHECK(fstate, STATE2)){     /* Check if in state 2 */
      if(!CHECK(fstate, STATE3)){   /* Check if in state 3 */
        if(!CHECK(fstate, STATE4)){ /* Check if in state 4 */
          if(!CHECK(fevent, EVENT2)){/* Must be state 5 */
            return fstate;          /* EVENT2 has not occurred, leave */
          }
          else {
            tadc = NOF;              /* No RAI */
            return STATE2;           /* Set state to 2 */
          }
        }
             /* STATE 4 */
        else if (!CHECK(fevent, EVENT3)){       /* EVENT3 has not occurred, leave */
            tadc = RAI;                         /* RAI */
            frame_status = frame_status & 0xFD; /* Clear MSDIS8 bit */
            tran_cfg = tran_cfg & 0xFB;         /* Clear E bit force to one */
        }
          else {                                /* EVENT3 has occurred */
            tadc = NOF;                         /* No RAI */
            frame_status = frame_status & 0xFD; /* Clear MSDIS8 bit */
            tran_cfg = tran_cfg & 0xFB;         /* Clear E bit force to one */
            return STATE3;                      /* Set state to 3 */
          }

      }
             /* STATE 3 */
      else if (!CHECK(fevent, EVENT1)){         /*EVENT1 has not occurred */
          return fstate;                        /* do nothing */
      }
        else {                                  /* EVENT1 has occurred */
          tadc = RAI;                           /* RAI */
          frame_status = frame_status & 0xFD;   /* Clear MSDIS8 bit */
          tran_cfg = tran_cfg & 0xFB;           /* Clear E bit force to one */
          return STATE1;                        /* Set to state 1 */
        }
    }
           /* STATE 2 HAS OCCURRED */
    else if (!CHECK(fevent, EVENT1)){           /* EVENT1 has not occurred */
      if (!CHECK(fevent, EVENT3)){              /* EVENT3 has not occurred */
        if (!CHECK(fevent, EVENT4)){            /* EVENT4 has not occurred */
          if (!CHECK(fevent, EVENT5)){          /* EVENT5 has not occurred */
            return fstate;                      /* leave */
          }
          else {                                /* EVENT5 has occurred */
            tadc = RAI;                         /* RAI */
            return STATE5;                      /* Set to state 5 */
          }
        }
        else {                                  /* EVENT4 has occurred */
          tadc = RAI;                           /* RAI */
          frame_status = frame_status | 0x02;   /* Set MSDIS8 bit */
          tran_cfg = tran_cfg | 0x04;           /* Set E bit force to one */
          tran_intl = tran_intl | 0xFF;
          return STATE4;                            /* Set state to 4 */
        }
      }
      else {                                    /* EVENT3 has occurred */
        tadc = NOF;                             /* No RAI */
        return STATE3;                          /* Set to state 3 */
      }
    }
    else {                                      /* EVENT1 has occurred */
      tadc = RAI;                               /* RAI */
      frame_status = frame_status & 0xFD;       /* Clear MSDIS8 bit */
      tran_cfg = tran_cfg & 0xFB;               /* Clear E bit force to one */
      return STATE1;                            /* Set state to 1 */
    }
  }
```

```
          /* STATE 1 HAS OCCURRED */
  else if (!CHECK(fevent, EVENT2)){          /* EVENT2 has not occurred */
    return fstate;                           /* Leave */
  }
  else {                                     /* EVENT2 has occurred */
    timer_count = CLEAR;                     /* Clear t3 */
    timer_expired = FALSE;                   /* Timer has not expired */
    tadc = NOF;                              /* No RAI */
    return STATE2;                           /* Set state to 2 */
  }
return fstate;
}
```

```
; Interrupt service routine for EQUAD
; This routine handles all state conditions
; As defined in PM970239 Issue 2
; Copyright 1998 PMC-Sierra, Inc
; Author: Korby Mraze

; *** Equates ***
; CONSTANTS
F1              EQU     $01
F2              EQU     $02
F3              EQU     $03
F4              EQU     $04
F5              EQU     $05
RRAI_M          EQU     $80
REDI_M          EQU     $08
AISI_M          EQU     $04

; *** 68HC11 MEMORY CONSTANTS ***

FRAME           EQU     $00     ; Local copy of offset reg 24
STATUS          EQU     $01     ; Local copy of offset reg 26
CMFACT          EQU     $02     ; Local copy of offset reg 21
STATE           EQU     $04     ; Current trans. state
TIMERX          EQU     $05     ; Boolean - 1 if timer has expired
T3              EQU     $06     ; T3 as defined in PM951034
FSTATE          EQU     $07     ; Current FSTATE
UPH             EQU     $08     ; USER PH
UMPH            EQU     $09     ; USER MPH
INT_STAT        EQU     $0A     ; Local copy of offset reg 25H
ALARM_STAT      EQU     $0B     ; Local copy of offset reg 27H
LOS             EQU     $0C     ; Local copy of offset reg 12H

; *** EQUAD Constants ***

QUADI           EQU     $C03C   ; Register with info on quadrant that interrupted
PORTA           EQU     $B000   ; PORT A DATA
INT             EQU     $08     ; EQUAD interrupt indication offset
FSII            EQU     $24     ; FRMR framing status interrupt indication offset
FSTAT           EQU     $26     ; FRMR framing status offset
FMMOD           EQU     $21     ; FRMR maintenance mode options offset
TADC            EQU     $45     ; TRAN Transmit alarm/diag control offset
FRMR_ALARM_INT  EQU     $25     ; FRMR alarm interrupt source offset
FRMR_ALARM_STAT EQU     $27     ; FRMR alarm status offset
CDRC            EQU     $12     ; CDRC interrupt status offset
TRANC          EQU     $44     ; TRAN Configuration offset

; MEMORY CONSTANTS
               ORG     $88
EQUAD_OFFSET    RMB     2
MEM_OFFSET      RMB     2
QUAD            RMB     1


; *** The code starts here

        ORG     $0000   ; set start of code (relative to ROM addr space)

; *** GET QUADRANT ***

        LDAA    QUADI   ; load the quadrant interrupt register
        STAA    QUAD    ; save it to ram

        LDAA    QUAD    ; load the quadrant interrupt status
        ANDA    #$10    ; check if quadrant 0 interrupted
        BNE     QUAD0   ; go to quadrant 0 initializations
        LDAA    QUAD    ; load the quadrant interrupt status
        ANDA    #$20    ; check if quadrant 1 interrupted
        BNE     QUAD1   ; go to quadrant 1 initializations
        LDAA    QUAD    ; load the quadrant interrupt status
        ANDA    #$40    ; check if quadrant 2 interrupted
        BNE     QUAD2   ; go to quadrant 2 initializations
        LDAA    QUAD    ; load the quadrant interrupt status
        ANDA    #$80    ; check if quadrant 3 interrupted
        BNE     QUAD3   ; go to quadrant 3 initializations
        RTS
```

```
QUAD0   LDD     #$C000
        STD     EQUAD_OFFSET
        LDD     #$1B00          ; set up offsets for quadrant 0
        STD     MEM_OFFSET
        BRA     CHKLOS

QUAD1   LDD     #$C080
        STD     EQUAD_OFFSET
        LDD     #$1B10          ; set up offsets for quadrant 1
        STD     MEM_OFFSET
        BRA     CHKLOS

QUAD2   LDD     #$C100
        STD     EQUAD_OFFSET
        LDD     #$1B20          ; set up offsets for quadrant 2
        STD     MEM_OFFSET
        BRA     CHKLOS

QUAD3   LDD     #$C180
        STD     EQUAD_OFFSET
        LDD     #$1B30          ; set up offsets for quadrant 3
        STD     MEM_OFFSET

; *** CHECK FOR LOS INTERRUPT ***

CHKLOS  LDX     EQUAD_OFFSET
        LDAA    CDRC,X      ; get reg 12
        LDX     MEM_OFFSET
        STAA    LOS,X       ; save it
        LDAA    LOS,X
        ANDA    #$40
        EORA    #$40
        BNE     FRMR        ; LOSI was 0, start code
        LDAA    LOS,X       ; LOSI was 1, check LOS
        ANDA    #$01
        EORA    #$01
        BNE     EXIT        ; LOS was 0, leave
        LDAA    #$01        ; LOS was 1
        STAA    STATE,X     ; set STATE to 1
        LDX     EQUAD_OFFSET
        LDAA    #$08
        STAA    TADC,X      ; RAI
        CLRA
        STAA    PORTA       ; LED ON
EXIT    RTS


; *** CHECK FOR FRMR INTERRUPT ***

FRMR    LDX     EQUAD_OFFSET
        LDAA    INT,X   ; load the interrupt indication status
        ANDA    #$20    ; check for FRMR interrupts
        BNE     MAIN    ; start code if valid
        RTS             ; if not leave


; *** START MAIN CODE HERE ***
; *** make local copies of equad int register ***

; *** PART 1 TRANSIENT STATES MATRIX ***

MAIN    LDX     EQUAD_OFFSET
        LDAA    FMMOD,X         ; get OFFSET reg 21
        ANDA    #$02            ; mask off unused bits
        LDX     MEM_OFFSET
        STAA    CMFACT,X        ; save it
        LDX     EQUAD_OFFSET
        LDAA    FSTAT,X         ; get OFFSET reg 26
        LDX     MEM_OFFSET
        STAA    STATUS,X        ; save it
        LDX     EQUAD_OFFSET
        LDAA    FSII,X          ; get OFFSET reg 24
        LDX     MEM_OFFSET
        STAA    FRAME,X         ; save it
        ANDA    #$40
```

```
        EORA    #$40
        BNE     OOCMFI    ; OOFI was 0, check OOCMFI
        LDAA    STATUS,X  ; OOFI was 1, check OOF
        ANDA    #$40
        EORA    #$40
        BNE     EVENT2    ; OOF was 0, this is event 2
        LDAA    CMFACT,X  ; OOF was 1, check CMFACT
        BEQ     EVENT1    ; CMFACT was 0, this is event 1
        LDAA    TIMERX,X  ; CMFACT was 1
         BNE     EVENT4    ; if timer has expired go to event 4
        LDAA    T3,X      ; otherwise increment
        INCA
        STAA    T3,X      ; and store
        CMPA    #$25      ; compare to 25H
         BLS     EVENT5    ; if timer has not reached 25H, go to event 5
        LDAA    #$01
         STAA    TIMERX,X  ; if it has, set TIMERX and move on
EVENT4   LDAB    #$04    ; continue with event 4 stuff
        BRA     WHST    ; find out which state

OOCMFI   LDAA    FRAME,X ; another look at reg 24
         ANDA    #$10
         EORA    #$10
         BNE     EVENT0  ; OOCMFI is 0 therefore no interrupt
         BRA     EVENT3  ; OOCMFI is 1 go to event 3

EVENT1   LDAB    #$01
        BRA     WHST    ; find out which state

EVENT2   LDAB    #$02
        BRA     WHST    ; find out which state

EVENT5   LDAB    #$05
        BRA     WHST    ; find out which state

EVENT0   CLRB            ; OOCMF is 1, this is event 0 (impossible event)
NOINT    BRA     STATES1 ; no interrupt, so go to STATES

EVENT3   LDAB    #$03    ;
WHST     LDAA    STATE,X ; get the current state
        EORA    #$01
        BEQ     S1      ; if nonzero, then this must be 1
        LDAA    STATE,X ; reload current state
        EORA    #$02
        BEQ     S2      ; if nonzero, then this must be 2
        LDAA    STATE,X ; reload current state
        EORA    #$03
        BEQ     S3      ; if nonzero, then this must be 3
        LDAA    STATE,X ; reload current state
        EORA    #$04
        BEQ     S4      ; if nonzero, then this must be 4
                        ; must be state 5
S5       TBA             ; copy event to acca
        EORA    #$02
        BNE     NOINT   ; if event 2 has not occurred, take no action
        LDX     EQUAD_OFFSET
        CLRA
        STAA    TADC,X     ; NO RAI
        LDX     MEM_OFFSET
        LDAA    #$80
        STAA    PORTA      ; LED OFF
        LDAA    #$02
        STAA    STATE,X    ; set state to 2
        BRA     STATES1    ; go to STATES

S1       TBA               ; copy event to acca
        EORA    #$02       ; if event 2 has not occurred, go to STATES
        BNE     NOINT
        CLRA
        STAA    T3,X       ; clear T3
        STAA    TIMERX,X   ; timer is not expired
        LDX     EQUAD_OFFSET
        CLRA
        STAA    TADC,X     ; NO RAI
        LDX     MEM_OFFSET
        LDAA    #$80
```

```
        STAA    PORTA       ;  LED OFF
        LDAA    #$02
        STAA    STATE,X     ; set state to 2
STATES1 BRA     STATES2     ; go to STATES

S2      TBA                 ; copy event to acca
        EORA    #$01        ; event1?
        BNE     NOTS2E1
        LDX     EQUAD_OFFSET
        LDAA    #$08
        STAA    TADC,X      ; RAI
        LDAA    FSTAT,X
        ANDA    #$FD        ; CLEAR 8MSDIS
        STAA    FSTAT,X
        LDAA    TRANC,X     ; DON'T force E bits to ONE
        ANDA    #$FB
        STAA    TRANC,X
        LDX     MEM_OFFSET
        CLRA
        STAA    PORTA       ; LED ON
        LDAA    #$01
        STAA    STATE,X     ; set state to 1
        BRA     STATES2     ; go to STATES

S3      BRA     REALS3      ; BRANCH TO FAR

NOTS2E1 TBA
        EORA    #$03
        BNE     NOTS2E3
        LDX     EQUAD_OFFSET
        LDAA    #$00
        STAA    TADC,X      ; NO RAI
        LDX     MEM_OFFSET
        LDAA    #$80
        STAA    PORTA       ; LED OFF
        LDAA    #$03
        STAA    STATE,X     ; set state to 3
        BRA     STATES2     ; go to STATES

S4      BRA     S4A         ; BRANCH TOO FAR

NOTS2E3 TBA
        EORA    #$04
        BNE     NOTS2E4
        LDX     EQUAD_OFFSET
        LDAA    #$08
        STAA    TADC,X      ; RAI
        LDAA    FSTAT,X
        ORA     #$02        ; SET 8MSDIS
        STAA    FSTAT,X
        LDAA    TRANC,X     ; force E bits to ONE
        ORA     #$04
        STAA    TRANC,X
        LDX     MEM_OFFSET
        CLRA
        STAA    PORTA       ; LED ON
        LDAA    #$04
        STAA    STATE,X     ; set state to 4
STATES2 BRA     STATES3     ; go to STATES

NOTS2E4 TBA
        EORA    #$05
        BNE     LEAVE1
        LDX     EQUAD_OFFSET
        LDAA    #$08
        STAA    TADC,X      ; RAI
        LDX     MEM_OFFSET
        CLRA
        STAA    PORTA       ; LED ON
        LDAA    #$05
        STAA    STATE,X     ; set state to 5
LEAVE1  BRA     STATES3     ; go to STATES

REALS3  TBA                 ; copy event to acca
        EORA    #$01        ; event1?
        BNE     LEAVE
```

```
        LDX     EQUAD_OFFSET
        LDAA    #$08
        STAA    TADC,X      ; RAI
         LDAA    FSTAT,X
        ANDA    #$FD        ; CLEAR 8MSDIS
         STAA    FSTAT,X
        LDAA    TRANC,X     ; DON'T force E bits to ONE
        ANDA    #$FB
        STAA    TRANC,X
         LDX     MEM_OFFSET
        CLRA
        STAA    PORTA       ; LED ON
        LDAA    #$01
        STAA    STATE,X     ; set state to 1
        BRA     STATES      ; go to STATES

STATES3 BRA     STATES      ; go to STATES

S4A  TBA                  ; copy event to acca
        EORA    #$03        ; event3?
         BNE     NOTS4E3
        LDX     EQUAD_OFFSET
        CLRA                ; EVENT3 occurred
        STAA    TADC,X      ; NO RAI
         LDAA    FSTAT,X
        ANDA    #$FD        ; CLEAR 8MSDIS
         STAA    FSTAT,X
        LDAA    TRANC,X     ; DON'T force E bits to ONE
        ANDA    #$FB
        STAA    TRANC,X
         LDX     MEM_OFFSET
        LDAA    #$80
        STAA    PORTA       ; LED OFF
        LDAA    #$03
        STAA    STATE,X     ; set state to 3
        RTS     STATES      ; go to STATES

NOTS4E3  TBA                  ; copy event to acca
        EORA    #$01        ; event1?
         BNE     LEAVE
        LDX     EQUAD_OFFSET
        LDAA    #$08        ; EVENT1 occurred
        STAA    TADC,X      ; RAI
         LDAA    FSTAT,X
        ANDA    #$FD        ; CLEAR 8MSDIS
         STAA    FSTAT,X
        LDAA    TRANC,X     ; DON'T force E bits to ONE
        ANDA    #$FB
        STAA    TRANC,X
         LDX     MEM_OFFSET
        CLRA
        STAA    PORTA       ; LED ON
        LDAA    #$01
        STAA    STATE,X     ; set state to 1
LEAVE   RTS     STATES      ; go to STATES

; *** PART 2 STATES MATRIX ***

STATES  LDX     EQUAD_OFFSET        ; LOAD QUADRANT OFFSET
        LDAA    FRMR_ALARM_STAT,X   ; LOAD ALARM STATUS
        LDX     MEM_OFFSET          ; LOAD MEMORY OFFSET
        STAA    ALARM_STAT,X        ; SAVE ALARM STATUS
        LDX     EQUAD_OFFSET        ; LOAD QUADRANT OFFSET
        LDAA    FRMR_ALARM_INT,X    ; LOAD INTERRUPT STATUS
        LDX     MEM_OFFSET          ; LOAD MEMORY OFFSET
        STAA    INT_STAT,X          ; SAVE INTERRUPT STATUS
        LDAA    INT_STAT,X
        ANDA    #AISI_M             ; CHECK IF AISI IS SET
        BNE     AISI
        LDAA    INT_STAT,X
        ANDA    #REDI_M             ; CHECK IF REDI IS SET
        BNE     REDI
        LDAA    INT_STAT,X
        ANDA    #RRAI_M             ; CHECK IF RRAI IS SET
        BNE     RRAI_1
        RTS                         ; IF NONE OF THE ABOVE, LEAVE
```

```
AISI    LDAA    ALARM_STAT,X
        ANDA    #$04        ;  CHECK IF AIS IS SET
        BNE     AIS_1

AIS_0   LDAA    FSTATE,X    ; CHECK FOR FSTATE4
        ANDA    #F4
        BNE     AIS_F4
        RTS

AIS_F4  LDAA    #$02        ; USER MPH-EI 2
        STAA    UMPH,X
        LDAA    #F3
        STAA    FSTATE,X    ; CHANGE TO FSTATE3
        BRA     REDI

AIS_1   LDAA    FSTATE,X    ;  CHECK FOR FSTATE 1
        ANDA    #F1
        BNE     AIS_F1
        LDAA    FSTATE,X    ;  CHECK FOR FSTATE 2
        ANDA    #F2
        BNE     AIS_F2
        LDAA    FSTATE,X    ;  CHECK FOR FSTATE 3
        ANDA    #F3
        BNE     AIS_F3
        LDAA    FSTATE,X    ;  CHECK FOR FSTATE 5
        ANDA    #F5
        BNE     AIS_F5
        RTS

AIS_F1  LDAA    #$01
        STAA    UPH,X       ; USER PH-DI
        LDAA    #$03
        STAA    UMPH,X      ; USER MPH-EI 3
        LDAA    #F4
        STAA    FSTATE,X    ; CHANGE TO FSTATE4
        RTS

AIS_F2  LDAA    #$03
        STAA    UMPH,X      ; USER MPH-EI 3
        LDAA    #F4
        STAA    FSTATE,X    ; CHANGE TO FSTATE4
        RTS

AIS_F3  LDAA    #$03
        STAA    UMPH,X      ; USER MPH-EI 3
        LDAA    #F4
        STAA    FSTATE,X    ; CHANGE TO FSTATE4
        RTS

AIS_F5  LDAA    #$03
        STAA    UMPH,X      ; USER MPH-EI 3
        LDAA    #F4
        STAA    FSTATE,X    ; CHANGE TO FSTATE4
        RTS

RRAI_1  BRA     RRAI_2

REDI    LDAA    ALARM_STAT,X
        ANDA    #$08        ; CHECK IF RED IS SET
        BNE     RED_1
RED_0   LDAA    FSTATE,X
        ANDA    #F3         ; CHECK FOR FSTATE3
        BNE     RED_F3
        LDAA    FSTATE,X
        ANDA    #F4         ; CHECK FOR FSTATE4
        BNE     RED_F4
        RTS

RED_1   LDAA    FSTATE,X
        ANDA    #F1         ; CHECK FOR FSTATE1
        BNE     RED_F1
        LDAA    FSTATE,X
        ANDA    #F2         ; CHECK FOR FSTATE2
        BNE     RED_F2
        LDAA    FSTATE,X
```

```
        ANDA    #F5         ; CHECK FOR FSTATE5
        BNE     RED_F5
        RTS

RED_F1  LDAA    #$01
        STAA    UPH,X       ; USER PH-DI
        LDAA    #$02
        STAA    UMPH,X      ; USER MPH-EI 2
        LDAA    #F3
        STAA    FSTATE,X    ; CHANGE TO FSTATE3
        RTS

RED_F2  LDAA    #$02
        STAA    UMPH,X      ; USER MPH-EI 2
        LDAA    #F3
        STAA    FSTATE,X    ; CHANGE TO FSTATE3
        RTS

RED_F5  LDAA    #$02
        STAA    UMPH,X      ; USER MPH-EI 2
        LDAA    #F3
        STAA    FSTATE,X    ; CHANGE TO FSTATE3
        RTS

RRAI_2  BRA     RRAI

RED_F3  LDAA    #$00
        STAA    UPH,X       ; USER PH-AI
        LDAA    #$00
        STAA    UMPH,X      ; USER MPH-AI
        LDAA    #F1
        STAA    FSTATE,X    ; CHANGE TO FSTATE1
        BRA     RRAI

RED_F4  LDAA    #$00
        STAA    UPH,X       ; USER PH-AI
        LDAA    #$00
        STAA    UMPH,X      ; USER MPH-AI
        LDAA    #F1
        STAA    FSTATE,X    ; CHANGE TO FSTATE1
        BRA     RRAI

RRAI    LDAA    ALARM_STAT,X
        ANDA    #$80        ; CHECK IF RRA BIT IS SET
        BNE     RRA_1
RRA_0   LDAA    FSTATE,X
        ANDA    #F2         ; CHECK IF IN FSTATE2
        BNE     RRA_F2
        LDAA    FSTATE,X
        ANDA    #F5         ; CHECK IF IN FSTATE5
        BNE     RRA_F5
        RTS

RRA_1   LDAA    FSTATE,X
        ANDA    #F1         ; CHECK IF IN FSTATE1
        BNE     RRA_F1
        RTS


RRA_F1  LDAA    #$01
        STAA    UPH,X       ; USER PH-DI
        LDAA    #$01
        STAA    UMPH,X      ; USER MPH-EI 2
        LDAA    #F2
        STAA    FSTATE,X    ; CHANGE TO FSTATE2
        RTS

RRA_F2  LDAA    #$00
        STAA    UPH,X       ; USER PH-AI
        LDAA    #$00
        STAA    UMPH,X      ; USER MPH-AI
        LDAA    #F1
        STAA    FSTATE,X    ; CHANGE TO FSTATE1
        RTS

RRA_F5  LDAA    #$00
```

```
        STAA    UPH,X       ; USER PH-AI
        LDAA    #$00
        STAA    UMPH,X      ; USER MPH-AI
        LDAA    #F1
        STAA    FSTATE,X    ; CHANGE TO FSTATE1
        RTS

; *** END OF SOURCE ***
```

APPENDIX B EXPLANATION OF SOFTWARE RESPONSE TO C.4.3 (TBR 004 B.5.2)

This section explains how the software routine in Appendix A responds to the ETS 300 011 C.4.3 test stimulus. It describes each step of the test in terms of stimulus, explanation of stimulus, expected response, and explanation of response. Each step corresponds to a line in the C.4.3 stimulus description.

The following table corresponds the naming of the State Table parameters with the variables in the C code pertaining to Appendices B, C, and D.

Table 5. Correspondence Between States Table and C Code

State Table Variable	EQUAD_SR.C Variable
MSDIS8	MSDIS8
CMFACT	CMFACT
E1	EVENT1
E2	EVENT2
E3	EVENT3
E4	EVENT4
E5	EVENT5
FALSE	FALSE
NOF	NOF
OOCMF	OOCMF

State Table Variable	EQUAD_SR.C Variable
OOCMFI	OOCMFI
OOF	OOFI
RAI	RAI
S1	STATE1
S2	STATE2
S3	STATE3
S4	STATE4
S5	STATE5
T3	timer_count
TRUE	TRUE

It is assumed that the EQUAD is initially in STATE1.

1) Stimulus:

BIT2=1, FAS
#

Explanation of Stimulus:

This stimulus should be sufficient for the IUT to find basic FAS alignment.

Expected Response:

NOF

Explanation of Response:

The EQUAD will find FAS alignment. It will interrupt with the status indicating:

OOFI=1
OOF=0

This indicates EVENT2 has occurred and with the EQUAD at STATE1 the routine will:

Move to STATE2
Reset timer_count
Assert NOF

The EQUAD will then find MFAS alignment. It will interrupt with the status indicating:

OOFI=0
OOCMFI=1

This indicates EVENT3 has occurred and with the EQUAD at STATE2 the routine will:

Move to STATE3
Assert NOF

2) Stimulus:

BIT 2=1, /FAS

Explanation of Stimulus:

This stimulus consists of a single corrupted FAS.
This stimulus stresses the IUTs ability to stay in basic FAS alignment, since this stimulus should not be sufficient to cause loss of FAS alignment (three consecutive corrupted FASs are necessary to lose FAS alignment).

Expected Response:

NOF

Explanation of Response:

The EQUAD will not interrupt during this stimulus.

3) Stimulus:

BIT2=1, FAS
#

Explanation of Stimulus:

This stimulus should keep the IUT in FAS alignment.

Expected Response:

NOF

Explanation of Response:

The EQUAD will not interrupt during this stimulus.

4) Stimulus:

BIT 2=1, /FAS, BIT2=1, /FAS

Explanation of Stimulus:

This stimulus consists of two consecutive corrupted FASs. This stimulus stresses the IUTs ability to stay in basic FAS alignment, since this stimulus should not be sufficient to cause loss of FAS alignment (three consecutive corrupted FASs are necessary to lose FAS alignment).

Expected Response:

NOF

Explanation of Response:

The EQUAD will not interrupt during this stimulus.

5) Stimulus:

BIT2=1, FAS
#

Explanation of Stimulus:

This stimulus should keep the IUT in FAS alignment.

Expected Response:

NOF

Explanation of Response:

The EQUAD will not interrupt during this stimulus.

6) Stimulus:

BIT 2=1, /FAS, BIT2=1, /FAS, BIT2=1, /FAS

Explanation of Stimulus:

This stimulus consists of three consecutive corrupted FASs. This stimulus stresses the IUTs ability to detect loss of FAS alignment. This stimulus should be sufficient to cause loss of FAS alignment.

Expected Response:

RAI

Explanation of Response:

The EQUAD will lose FAS alignment. It will interrupt with the status indicating:
OOFI=1

OOF=1
CMFACT=0

This indicates EVENT1 has occurred and with the EQUAD at STATE3 the routine will:

Move to STATE1
Assert RAI

7) Stimulus:

BIT2=1, FAS, BIT2=1, FAS
#

Explanation of Stimulus:

This stimulus checks the IUT's ability to find FAS alignment. This should be sufficient to find FAS alignment.

Expected Response:

NOF

Explanation of Response:

The EQUAD will find FAS alignment. It will interrupt with the status indicating:

OOFI=1
OOF=0

This indicates EVENT2 has occurred and with the EQUAD at STATE1 the routine will:

Move to STATE2
Reset timer_count
Assert NOF

The EQUAD will then find MFAS alignment. It will interrupt with the status indicating:

OOFI=0
OOCMFI=1
OOCMF=0

This indicates EVENT3 has occurred and with the EQUAD at STATE2 the routine will:

Move to STATE3
Assert NOF

8) Stimulus:

BIT 2=1, /FAS, BIT2=1, /FAS, BIT2=1, /FAS

Explanation of Stimulus:

This stimulus consists of three consecutive corrupted FASs.
This stimulus stresses the IUTs ability to detect loss of FAS alignment. This stimulus should be sufficient to cause loss of FAS alignment.

Expected Response:

RAI

Explanation of Response:

The EQUAD will lose FAS alignment. It will interrupt with the status indicating:

OOFI=1
OOF=1
CMFACT=0

This indicates EVENT1 has occurred and with the EQUAD at STATE3 the routine will:

Move to STATE1
Assert RAI

9) Stimulus:

BIT2=1, FAS, BIT2=1, /FAS
#

Explanation of Stimulus:

This stimulus stresses the IUT's algorithm for finding FAS alignment. This stimulus should not be sufficient to find FAS alignment, since two consecutive correct FASs are required.

Expected Response:

RAI

Explanation of Response:

The EQUAD will not interrupt during this stimulus.

10) Stimulus:

BIT 2=1, FAS

Explanation of Stimulus:

This stimulus consists of a single correct FAS. This stimulus stresses the IUTs ability to find FAS alignment. This stimulus should not be sufficient to find FAS alignment.

Expected Response:

RAI

Explanation of Response:

The EQUAD will not interrupt during this stimulus.

11) Stimulus:

BIT2=0, FAS
#

Explanation of Stimulus:

This stimulus stresses the IUT's algorithm for finding FAS alignment. This stimulus should not be sufficient to find FAS alignment, since two consecutive correct FASs are required with BIT2=1 in-between.

Expected Response:

RAI

Explanation of Response:

The EQUAD will not interrupt during this stimulus.

12) Stimulus:

BIT 2=1, FAS
#

Explanation of Stimulus:

This stimulus stresses the IUTs ability to find FAS alignment. This stimulus should be sufficient to find FAS alignment.

Expected Response:

NOF

Explanation of Response:

The EQUAD will find FAS alignment. It will interrupt with the status indicating:

OOFI=1
OOF=0

This indicates EVENT2 has occurred and with the EQUAD at STATE1 the routine will:

Move to STATE2
Reset timer_count
Assert NOF

The EQUAD will then find MFAS alignment. It will interrupt with the status indicating:

OOFI=0
OOCMFI=1
OOCMF=0

This indicates EVENT3 has occurred and with the EQUAD at STATE2 the routine will:

Move to STATE3
Assert NOF

13) Stimulus:

BIT2=0, FAS, BIT2=1, FAS, BIT2=1, FAS
#

Explanation of Stimulus:

This stimulus consists of single BIT2=0. This stimulus stresses the IUT's algorithm for losing FAS alignment. This stimulus should not be sufficient to lose FAS alignment, since three consecutive BIT2=0 are required.

Expected Response:

NOF

Explanation of Response:

The EQUAD will not interrupt during this stimulus.

14) Stimulus:

BIT2=0, FAS, BIT2=0, FAS, BIT2=1, FAS
#

Explanation of Stimulus:

This stimulus consists of two consecutive BIT2=0. This stimulus stresses the IUT's algorithm for losing FAS alignment. This stimulus should not be sufficient to lose FAS alignment, since three consecutive BIT2=0 are required.

Expected Response:

NOF

Explanation of Response:

The EQUAD will not interrupt during this stimulus.

15) Stimulus:

BIT2=0, FAS, BIT2=0, FAS, BIT2=0, FAS
#

Explanation of Stimulus:

This stimulus consists of three consecutive BIT2=0. This stimulus stresses the IUT's algorithm for losing FAS alignment. This stimulus should be sufficient to lose FAS alignment.

Expected Response:

RAI

Explanation of Response:

The EQUAD will lose FAS alignment. It will interrupt with the status indicating:

OOFI=1
OOF=1
CMFACT=0

This indicates EVENT1 has occurred and with the EQUAD at STATE3 the routine will:

Move to STATE1
Assert RAI

16) Stimulus:

BIT 2=1, FAS
#

Explanation of Stimulus:

This stimulus stresses the IUTs ability to find FAS alignment. This stimulus should be sufficient to find FAS alignment.

Expected Response:

NOF

Explanation of Response:

The EQUAD will find FAS alignment. It will interrupt with the status indicating:

OOFI=1
OOF=0

This indicates EVENT2 has occurred and with the EQUAD at STATE1 the routine will:

Move to STATE2
Reset timer_count
Assert NOF

The EQUAD will then find MFAS alignment. It will interrupt with the status indicating:

OOFI=0
OOCMFI=1

This indicates EVENT3 has occurred and with the EQUAD at STATE2 the routine will:

Move to STATE3
Assert NOF

17) Stimulus:

BIT2=1
FRAMEB
#

Explanation of Stimulus:

This stimulus consists of two consecutive frames with a mimic framing pattern in Timeslot 31. This stimulus stresses the IUT's robustness against mimic FASs. This stimulus should not be sufficient to change FAS alignment.

Expected Response:

NOF

Explanation of Response:

The EQUAD will not interrupt during this stimulus.

18) Stimulus:

6 x FRAME C

Explanation of Stimulus:

This stimulus consists of six repetitions of a two frame sequence where the FAS and BIT2 are corrupted in their current position, but another correct FAS and BIT2 are available in Timeslot 31. This stimulus stresses the IUTs ability to change FAS alignment. This stimulus should be sufficient to force a change of FAS alignment.

Expected Response:

RAI --> NOF

Explanation of Response:

The EQUAD will lose FAS alignment. It will interrupt with the status indicating:

OOFI=1
OOF=1
CMFACT=0

This indicates EVENT1 has occurred and with the EQUAD at STATE3 the routine will:

Move to STATE1
Assert RAI

Second, the EQUAD will find FAS alignment (in what was previously Timeslot 31). It will interrupt with the status indicating:

OOFI=1
OOF=0

This indicates EVENT2 has occurred and with the EQUAD at STATE1 the routine will:

Move to STATE2
Reset timer_count
Assert NOF

Not find MFAS

19) Stimulus:

FRAME B
(For 4 to 8 ms)

Explanation of Stimulus:

This stimulus consists of two consecutive frames with a mimic framing pattern in Timeslot 31. This stimulus will keep the IUT aligned on Timeslot 31. However, since there is no MFAS present, a re-search for basic FAS alignment should be forced eventually because MFAS alignment cannot be found (as recommended in G.706 Section 4.2).

Expected Response:

RAI --> NOF (at least once)

Explanation of Response:

The EQUAD will be forced out FAS alignment by the circuitry trying to find MFAS alignment. It will interrupt with the status indicating:

OOFI=1
OOF=1
CMFACT=1
T3 is not expired

This indicates EVENT5 has occurred and with the EQUAD at STATE2 the routine will:

Move to STATE5
Assert RAI

Second, the EQUAD will find FAS alignment (in one of the two FAS alignments present). It will interrupt with the status indicating:

OOFI=1
OOF=0

This indicates EVENT2 has occurred and with the EQUAD at STATE5 the routine will:

Move to STATE2
Assert NOF

APPENDIX C EXPLANATION OF SOFTWARE RESPONSE TO C.4.4 (TBR 004 B.5.3)

This section explains how an implementation of the recommended Transient States Matrix (refer to Section 4.4.4) passes the ETS 300 011 (A1) C.4.4 test. It describes each step of the test in terms of stimulus, explanation of stimulus, expected response, and explanation of response. Each step corresponds to a line in the Amendment 1 C.4.4 stimulus description.

Assume the EQUAD is initially in STATE1

1) Stimulus:

FRAME B
#

Explanation of Stimulus:

This is an undetermined length of time in which frames with a mimic FAS are sent in order to stress the IUT's ability to find FAS and MFAS in the presence of a mimic FAS.

Expected Response:

NOF

Explanation of Response:

The EQUAD will first find FAS alignment. It will interrupt with the status indicating:

OOFI=1
OOF=0

This indicates EVENT2 has occurred and with the EQUAD at STATE1 the routine will:

Move to STATE2
Reset timer_count
Assert NOF

If the FAS alignment is correct the EQUAD will find MFAS alignment. It will interrupt with the status indicating:

OOFI=0
OOCMFI=1

This indicates EVENT3 has occurred and with the EQUAD at STATE2 the routine will:

Move to STATE3
Assert NOF

If the FAS alignment was incorrect (i.e. aligned to a mimic) disregard the above EVENT3 and the then the EQUAD will force a reframe after 8ms. It will interrupt indicating:

OOFI=1
OOF=1
CMFACT=1
timer_count not expired(i.e. timer_expired=FALSE)

This indicates EVENT5 has occurred and with the EQUAD at STATE2 the routine will:

Move to STATE5
Assert RAI

Next, the EQUAD will find the correct FAS. It will interrupt indicating:

OOFI=1
OOF=0

This indicates EVENT2 has occurred and with the EQUAD at STATE5 the routine will:

Move to STATE2
Assert NOF

The EQUAD will then find MFAS. It will interrupt indicating:

OOFI=0
OOCMFI=1

This indicates EVENT3 has occurred and with the EQUAD at STATE2 the routine will:

Move to STATE3

Assert NOF

2) Stimulus:

/FAS, BIT2=1, /FAS, BIT2=1
/FAS, BIT2=1

Explanation of Stimulus:

The E1 basic framing format (described in ITU-T G.704 Section 2.3.2) alternates between frames with Timeslot 0 containing the frame alignment signal (FAS frames) and frames containing Bit 2 of Timeslot 0 set to logic one (NFAS frames). This stimulus presents three consecutive FAS frames with corrupted FASs alternated with three consecutive uncorrupted NFAS frames.
As specified by ITU-T G.706 Section 4.1.1, basic "frame alignment will be assumed to have been lost when three consecutive incorrect frame alignment signals have been received." Therefore, this stimulus is meant to force the IUT out of FAS alignment.

Expected Response:

RAI

Explanation of Response:

The EQUAD will lose FAS alignment. It will interrupt with the status indicating:

OOFI=1
OOF=1
CMFACT=0

This indicates EVENT1 has occurred and with the EQUAD at STATE3 the routine will:

Move to STATE1
Assert RAI

3) Stimulus:

MF A

Explanation of Stimulus:

This is a correct CRC-4 multiframe. However, a single MF A is not sufficient to find MFAS alignment (see ITU-T G.706 Section 4.2).

Expected Response:

NOF

Explanation of Response:

The EQUAD will find FAS alignment. It will interrupt with the status indicating:

OOFI=1
OOF=0

This indicates EVENT2 has occurred and with the EQUAD at STATE1 the routine will:

Move to STATE2
Reset timer_count
Assert NOF

4) Stimulus:

4 X MF B

Explanation of Stimulus:

This is four consecutive multiframes with incorrect MFASs. This is equivalent to 8 ms of incorrect MFAS. The point of this stimulus is to test the IUTs compliance with ITU-T G.706 Section 4.2 which states: "If multiframe alignment cannot be achieved within 8 ms, it should be assumed that frame alignment is due to a spurious frame alignment signal and a re-search for frame alignment should be initiated."

Expected Response:

NOF

Explanation of Response:

The EQUAD will interrupt twice during this stimulus. First, after 8 ms of unsuccessfully trying to find MFAS alignment, the EQUAD will interrupt indicating:

OOFI=1
OOF=1
CMFACT=1
timer_count not expired

This indicates EVENT5 has occurred and with the EQUAD at STATE2 the routine will:

Move to STATE5
Assert RAI

Secondly, the EQUAD will interrupt again as soon as it finds FAS alignment, indicating:

OOFI=1
OOF=0

This indicates EVENT2 has occurred and with the EQUAD at STATE5 the routine will:

Move to STATE2
Assert NOF

5) Stimulus:

MF A

Explanation of Stimulus:

This is a correct CRC-4 multiframe. However, a single MF A is not sufficient to find MFAS alignment (see ITU-T G.706 Section 4.2).

Expected Response:

NOF

Explanation of Response:

The EQUAD will not interrupt during this stimulus.

6) Stimulus:

37 X MF B

Explanation of Stimulus:

This stimulus is 37 consecutive CRC-4 multiframes with incorrect MFASs. The point of this stimulus is to stress the lower bound (100 ms) specified for the timer, as given in ITU-T G.706 Section 4.2 Note 2.
By the end of this stimulus, 37+6=43 multiframes will have passed since the EQUAD began looking for MFAS alignment. Each CMF has a period of 2 ms, therefore 86 ms will have passed.

Expected Response:

NOF, transition to RAI and back to NOF (depending on implementation)

Explanation of Response:

The EQUAD will interrupt twice every 8 ms. Every 8 ms, the EQUAD will interrupt when the CMF hunt algorithm forces a re-search for FAS alignment, then, after a few FASs, the EQUAD will interrupt when it finds find FAS alignment.

After 8 ms of unsuccessfully trying to find MFAS alignment, the EQUAD will interrupt indicating:

OOFI=1
OOF=1
CMFACT=1
timer_count not expired(i.e. timer_expired=FALSE)

This indicates EVENT5 has occurred and with the EQUAD at STATE2 the routine will:

Move to STATE5
Assert RAI

Secondly, the EQUAD will interrupt again as soon as it finds FAS alignment, indicating:

OOFI=1
OOF=0

This indicates EVENT2 has occurred and with the EQUAD at STATE5 the routine will:

Move to STATE2
Assert NOF

The result will be that every 8 ms, the IUT will assert RAI for about 500µs then deassert it for the duration of this stimulus.

7) Stimulus:

MF A, MF B, MF A, MF B, MF A, MF B

Explanation of Stimulus:

This is an alternation between correct and incorrect MFASs. As specified in ITU-T G.706 Section 4.2, this stimulus should be sufficient to find MFAS alignment.

Expected Response:

NOF

Explanation of Response:

Somewhere during this stimulus, the EQUAD will find MFAS alignment. At that time, it will interrupt indicating:

OOFI=0
OOCMFI=1

This indicates EVENT3 has occurred and with the EQUAD at STATE2 the routine will:

Move to STATE3
Assert NOF

The EQUAD will not interrupt again during this stimulus.

8) Stimulus:

MF B

Explanation of Stimulus:

This is a multiframe with incorrect MFAS.

Expected Response:

NOF

Explanation of Response:

The EQUAD will not interrupt during this stimulus

9) Stimulus:

251

Explanation of Stimulus:

The previous stimulus (MF B) is repeated at least 251 times. This is to stress the upper limit (500 ms) specified for the timer in ITU-T G.706 Section 4.2.
251 multiframes corresponds to 502 ms.

Expected Response:

NOF

Explanation of Response:

The EQUAD will not interrupt during this stimulus

10) Stimulus:

/FAS, BIT2=1, /FAS, BIT2=1
/FAS, BIT2=1

Explanation of Stimulus:

This stimulus presents three consecutive frames with corrupted FASs alternated with three consecutive uncorrupted NFAS frames.
As specified by ITU-T G.706 Section 4.1.1, basic "frame alignment will be assumed to have been lost when three consecutive incorrect frame alignment signals have been received." Therefore, this stimulus is meant to force the IUT out of FAS alignment.

Expected Response:

RAI

Explanation of Response:

The EQUAD will lose FAS alignment. It will interrupt with the status indicating:

OOFI=1
OOF=1
CMFACT=0

This indicates EVENT1 has occurred and with the EQUAD at STATE2 the routine will:

Move to STATE1
Assert RAI

11) Stimulus:

MF B

Explanation of Stimulus:

This is a single multiframe with a corrupted MFAS. The basic FAS in not corrupted however, so the framer should find FAS alignment.

Expected Response:

NOF

Explanation of Response:

The EQUAD will find FAS alignment. It will interrupt with the status indicating:

OOFI=1
OOF=0

This indicates EVENT2 has occurred and with the EQUAD at STATE1 the routine will:

Move to STATE2
Reset timer_count
Assert NOF

12) Stimulus:

#250
MF B

Explanation of Stimulus:

This is 251 consecutive CMFs sent with corrupted MFASs. The basic FAS is not corrupted.
This stimulus stresses the upper limit (500ms) specified for the timer in ITU-T G.706 Section 4.2. 251 multiframes corresponds to 502ms.

Expected Response:

NOF -> RAI -> NOF(at least once) then stable RAI

Explanation of Response:

The EQUAD will interrupt many times during this interrupt before the timer expires. The EQUAD will first interrupt indicating:

OOFI=1
OOF=1
CMFACT=1
timer_count not expired(i.e. timer_expired=FALSE)

This indicates EVENT5 has occurred and with the EQUAD at STATE2 the routine will:

Move to STATE5
Assert RAI

Next, the EQUAD will interrupt when it finds FAS alignment, indicating:

OOFI=1
OOF=0

This indicates EVENT2 has occurred and with the EQUAD at STATE5 the routine will:

Move to STATE2
Assert NOF

These last two interrupt responses will be repeated every 8ms until the timer expires. After the timer expires, the EQUAD will interrupt indicating:

OOFI=1
OOF=1
CMFACT=1
timer_count has expired(i.e. timer_expired=TRUE)

This indicates EVENT4 has occurred and with the EQUAD at STATE2 the routine will:

Move to STATE4
Assert RAI
Set MSDIS8 bit

13) Stimulus:

MF A, 4 x MF B

Explanation of Stimulus:

This is one multiframe with a correct MFAS followed by four consecutive multiframes with corrupted MFASs.
This stimulus tests to see that the IUT does not prematurely declare MFAS alignment. A single correct MFAS should not be sufficient to find MFAS alignment.

Expected Response:

RAI

Explanation of Response:

The EQUAD will not interrupt during this stimulus, so the response from the previous stimulus, RAI, will be maintained.

14) Stimulus:
MF A, 2 x MF B, MF A
MF A, 2 x MF B, 2 x MF A

Explanation of Stimulus:

This stimulus should be sufficient for the IUT to find MFAS alignment. Since there are at least two correct MFASs present in each 8ms window, separated by an integral multiple of 2ms (as per the G.706 requirement). This stimulus causes the EQUAD to interrupt several times.

Expected Response:

NOF

Explanation of Response:

First is due to an errata the EQUAD will locate MFAS alignment based on the first MF A. The software should ignore the first MF A with proper FAS. The EQUAD will interrupt with the status indicating:

OOFI=0
OOCMFI=1

This indicates EVENT3 has occurred and with the EQUAD at STATE4 the routine will:

Go to state 3
Assert NOF

15) Stimulus:

MF B, MF A
#

Explanation of Stimulus:

This stimulus consists of multiframes with alternating correct and corrupted MFASs, repeated indefinitely.
This stimulus stresses the IUTs ability to stay in MFAS alignment, since this stimulus should not be sufficient to cause loss of MFAS alignment

Expected Response:

NOF

Explanation of Response:

The EQUAD will not interrupt during this stimulus.

APPENDIX D EXPLANATION OF SOFTWARE RESPONSE TO C.4.5 (TBR 004 B.4.2)

This section explains how an implementation of the recommended Transient States Matrix (refer to Section 4.4.4) passes the ETS 300 011 (A1) C.4.5 test. It describes each step of the test in terms of stimulus, explanation of stimulus, expected response, and explanation of response. Each step corresponds to a line in the Amendment 1 C.4.5 stimulus description.

Assume the EQUAD is initially in STATE1.

1) Stimulus:

SMF A

Repeat more than 1 second

Explanation of Stimulus:

This is 1 second of sub-multiframes having correct generation of C1 to C4 bits.

Expected Response:

NOF, No E bit set to zero.

Explanation of Response:

The EQUAD will find FAS alignment. It will interrupt with the status indicating:

OOFI=1
OOF=0

This indicates EVENT2 has occurred and with the EQUAD at STATE1 the routine will:

Move to STATE2
Reset timer_count

The EQUAD will then find MFAS. It will interrupt indicating:

OOFI=0
OOCMFI=1
OOCMF=0

This indicates EVENT3 has occurred and with the EQUAD at STATE2 the routine will:

Move to STATE3

2) Stimulus:

SMF B

Explanation of Stimulus:

One sub-multiframe having incorrect generation of C1 to C4 bits.

Expected Response:

One E bit set to zero.

Explanation of Response:

The EQUAD will not interrupt during this stimulus.

3) Stimulus:

SMF A
#

Explanation of Stimulus:

Continuous sub-multiframes having correct generation of C1 to C4 bits.

Expected Response:

No E bit set to zero.

Explanation of Response:

The EQUAD will not interrupt during this stimulus.

4) Stimulus:

SMF B, SMF, B

Explanation of Stimulus:

Two consecutive sub-multiframes having incorrect generation of C1 to C4 bits.

Expected Response:

Two contiguous E bits set to zero.

Explanation of Response:

The EQUAD will not interrupt during this stimulus.

5) Stimulus:

SMF A
Repeated more than 1 second

Explanation of Stimulus:

This is 1 second of sub-multiframes having correct generation of C1 to C4 bits.

Expected Response:

No E bit set to zero.

Explanation of Response:

The EQUAD will not interrupt during this stimulus.

6) Stimulus:

914 X SMF B

Explanation of Stimulus:

914 consecutive sub-multiframes having incorrect generation of C1 to C4 bits.

Expected Response:

914 contiguous E bits set to zero.

Explanation of Response:

The EQUAD will not interrupt during this stimulus.

7) Stimulus:

86 X SMF A

Explanation of Stimulus:

86 consecutive sub-multiframes having correct generation of C1 to C4 bits.

Expected Response:

86 contiguous E bits set to ONE.

Explanation of Response:

The EQUAD will not interrupt during this stimulus.

8) Stimulus:

914 X SMF B

Explanation of Stimulus:

914 consecutive sub-multiframes having incorrect generation of C1 to C4 bits.

Expected Response:

914 contiguous E bits set to zero.

Explanation of Response:

The EQUAD will not interrupt during this stimulus.

9) Stimulus:

SMF A
Repeated more than 1 second

Explanation of Stimulus:

This is 1 second of sub-multiframes having correct generation of C1 to C4 bits.

Expected Response:

No E bit set to zero.

Explanation of Response:

The EQUAD will not interrupt during this stimulus.

10) Stimulus:

915 X SMF B

Explanation of Stimulus:

915 continuous sub-multiframes having incorrect generation of C1 to C4 bits.

Expected Response:

RAI (at least once)

Explanation of Response:

The EQUAD will not interrupt during this stimulus.

11) Stimulus:

85 X SMF A

Explanation of Stimulus:

85 consecutive sub-multiframes having correct generation of C1 to C4 bits.

Expected Response:

RAI (at least once)

Explanation of Response:

The EQUAD will not interrupt during this stimulus.

12) Stimulus:

915 X SMF B

Explanation of Stimulus:

915 consecutive sub-multiframes having incorrect generation of C1 to C4 bits.

Expected Response:

RAI (at least once) -> NOF

Explanation of Response:

The EQUAD has lost FAS alignment. It will interrupt with the status indicating.

OOFI=1
OOF=1
CMFACT=0

This indicates EVENT1 has occurred and with the EQUAD at STATE3 the routine will:

Move to STATE1
Assert RAI
set MSDIS8=1

The EQUAD will then find FAS alignment. It will interrupt with the status indicating:

OOFI=1
OOF=0

This indicates EVENT2 has occurred and with the EQUAD at STATE1 the routine will:

Move to STATE2
Reset timer_count

The EQUAD will then find MFAS. It will interrupt indicating:

OOFI=0
OOCMFI=1

This indicates EVENT3 has occurred and with the EQUAD at STATE2 the routine will:

Move to STATE3
Assert NOF

13) Stimulus:

SMF A
#

Explanation of Stimulus:

This is continuous sub-multiframes having correct generation of C1 to C4 bits.

Expected Response:

NOF, No E bit set to zero.

Explanation of Response:

The EQUAD will not interrupt during this stimulus.

14) Stimulus:

/FAS, BIT 2 = 1, /FAS, BIT2 = 1

Explanation of Stimulus:

Two consecutive frames with incorrect FAS, incorrect bits C1 to C4 and incorrect MFAS in timeslot 0.

Expected Response:

RAI, No E bit set to zero.

Explanation of Response:

The EQUAD has lost FAS alignment. It will interrupt with the status indicating.

OOFI=1
OOF=1
CMFACT=0

This indicates EVENT1 has occurred and with the EQUAD at STATE3 the routine will:

Move to STATE1
Assert RAI
set MSDIS8=0

15) Stimulus:

/FAS, BIT2 = 1
#

Explanation of Stimulus:

Continuous frames with incorrect FAS, incorrect bits C1 to C4 and incorrect MFAS in timeslot 0.

Expected Response:

RAI, No E bit set to zero.

Explanation of Response:

The EQUAD will not interrupt during this stimulus.

NOTES

PMC-9803xx Issue date: February 1998
Printed in Canada