STEREO SOUND IN SYNC

LDM 1903/00/01 (8928 190 30001)

LDM 1904/00/01 (8928 190 40001)

MONITORING

Contents

STEREO SOUND IN SYNC

LDM 1903/00/01 (8928 190 30001)

LDM 1904/00/01 (8928 190 40001)

MONITORING

1. INTRODUCTION

Most of the cards in the Stereo Sound in Sync system have fault indication LEDs and fault outputs to the mother board to indicate whether some part of the system has failed. The fault outputs are taken to a monitoring card, Monitor 1 which provides remote fault indications on the basis of the inputs sent to it.

2. MONITOR 1 - HARDWARE

This card is a Z80 processor system with interfacing to the rest of the system. Thus the action of the card depends on the software supplied for it.

24 fault input lines are provided, not all of which are currently used. These are taken to R/S flip-flops which are set by the fault input going active. The outputs of the flip-flops appear as three read-only ports at address 00, 20 and 40 Hex. The flip-flops assigned to each port are reset by the program reading from that port.

Main and secondary fault outputs are provided along with six open-collector outputs. The main fault output is driven from a monostable which has to be regularly triggered by the program to keep the output in the 'system good' state. This is so that, if the program crashes, the monitor will declare a fault. The main and secondary fault outputs and the open-collector outputs are driven from a write-only port at 60 Hex.

The circuit is provided with 16k bytes of RAM, 16k bytes of EPROM which is copied to RAM on switch-on and a 16k byte BBC BASIC interpreter EPROM. A CTC chip is supplied to provide a real-time clock.

There is also an 8-bit data bus, which is not used in this application and various input and output control or timing signals.

A 4-pole switch S3 is set to indicate to the software the circumstances

in which the card is working. The switch positions have the following significance:

1.	On:	Coder	Off: Decoder
2.	On:	Main facilities	Off: Full facilities, i.e. in the case of the Coder, when an Audio Decoder module is fitted, in the case of the Decoder, when an Audio Comparator is fitted.
3.	On:	Not used	Off: Not used.
4.	On:	No audio modules	Off: With audio modules.

The main fault output controls a green 'System Good' LED which protrudes through the front panel and a 2-pole changeover relay whose contacts are connected to the 37-way 'D' type connector on the back panel.

The secondary output controls a red LED behind the front panel and another 2-pole changeover relay connected to the 37-way 'D' connector.

Six other LEDs with corresponding open-collector outputs to the 37-way D connector indicate fault inputs in logical groupings whenever the main fault output comes on. The fault groupings are determined by the software.

Two RS232 ports are provided on the front of the module: one for interactive control or modification of the software via a terminal; the other for loading a program from a microcomputer or sending a program from RAM to a PROM Programmer. In both cases the baud rate is 9600, there must be one or more stop bits, parity is ignored and there is no handshake. The data output to a PROM programmer is in 'Intel Intellec' format. A TERMINAL ENABLE switch and ESCAPE button are provided for use with these facilities. The two RS232 ports are labelled X2 and the connections are as follows (Pin 1 at the top):

1.	0V	)
		)
2.	Transmit	) Terminal
		)
3.	Receive	)

4. Transmit to PROM PROGRAMMER

5. Receive from BBC microcomputer

6. 0V.

A small HELP screen is available by typing '*' on the terminal.

3. MONITOR 1 - SOFTWARE

3.1 What It Does

Most of the software is concerned with controlling the main fault output. On the coder the secondary fault output is used to indicate that the incoming video already carries sound in syncs, so it can be used to drive an external bypass relay. On the decoder it is not used. The six open-collector outputs are currently used to give information about the nature of the fault once the system has been declared faulty.

The software is designed so that brief fault inputs do not cause the main fault output to come on, unless too many occur in a given time. Also the inputs must be fault free for a specified time before the fault output is switched off again.

More precisely, for each input the following variables can be set: TL%, TC%, TTG%, NL%, FP%.

TL% — If an input is continuously faulty for a period of TL% then the main fault output comes on. Note that since the inputs are fed via R/S flip-flops, as long as the input is showing a fault at least once in each program cycle the input will appear to be continuously faulty (e.g. a rapidly pulsing fault input would appear as a continuous fault). If TL% is set to -1 then the input is allowed to go faulty continuously without triggering the main fault output, though intermittent faults can still trigger the fault output (see NL%, FP% below).

TC% — If after being faulty an input is good for a period of less than TC% then that good period is ignored for the purpose of timing the length of the fault, i.e. the input is considered to be continuously faulty with the good period being bad. If TC% is longer than TL% and two short faults occur within a period of TC% (but separated by more than TL%) then the main fault output will come on.

NL%, FP% — These variables control intermittent fault detection. If more than NL% faults occur in a period of FP%, then the main fault output comes on (note that faults separated by less than TC% count as one fault). Counting take place in separate periods so it is possible for more than NL% faults to occur in a period of FP% without triggering the main fault output if, say, half of them occur at the end of one timing period and the other half at the start of the next, with less than or equal to NL% in each period. Once a timing period has completed with less than or equal to NL% faults the next period does not start

until another fault appears. Once more than NL% faults have been detected a new timing period starts immediately and the input is considered intermittently faulty until a timing period expires with less than or equal to NL% faults counted.

TTG% Once the main fault output has come on it will not go off again until each input has been fault free for the preceding period of TTG%. This applies to all inputs, not just to those that caused the main fault output to come on in the first place.

Note: If an intermittent fault has been detected the main output cannot go good until a period of FP% has expired. This may well be longer than TTG%. TTG% applies only to the actual fault input, it does not time from when an intermittent fault clears. If an intermittent fault clears then the main fault output will go off immediately so long as the inputs have already been fault free long enough.

The program's accuracy in timing faults depends on the frequency at which it polls its inputs. This is variable. If all the inputs are static then the period is of the order of 0.1s. If the inputs change, the program then has additional processing to do and this delays the next poll by about 0.7s, though a worst case of between 1 and 1.5s is possible.

The most obvious effect is that short faults appear longer than they really are. Once a fault has been detected the fact that it has been cleared will not be detected for around 0.7s. Therefore there is no point in setting any of TL%, TC%, or TTG% to less than about one or two seconds, unless they are actually set to zero.

On the coder the secondary fault output is used to indicate that the incoming video already carries sound in sync. This is done under the control of a timer so the output does not come on until sound in sync has been detected for a period of BP_{on} and it does not go off until there has been no sound in sync for a period of BP_{off}. The main fault output may or may not be required to come on when sound in sync is detected; the value of TL% pulses should be set to -1 if not.

The two inputs SYNC FAIL and STANDBY ON are treated as special cases on the coder. SYNC FAIL does not cause the main fault output to come on as long as STANDBY ON comes on within the period of the variable TSD (see Sh. 595-7). The main fault output also comes on if Standby is on after a delay of TSD from SYNC FAIL going off. If it is required that SYNC FAIL or STANDBY ON always turn the main fault output on then the DATA table should be altered so that TL% is not set to -1 for these inputs.

The open-collector outputs are all off while the main fault output is off. When the main fault output is on they are controlled by combinational logic applied to the fault inputs. They are intended to give information about the cause of the fault.

3.2 How It Does It

Each time round the main loop a procedure PROCpoll is called which polls the inputs and logs their behaviour.

Variables 'Stable' and 'Marker' are controlled by PROCfaulty and PROCOK to indicate respectively whether the main fault output will have to change state at some time, if the inputs remain in the current state, and if so, when. While the main fault output is off they are updated each time an input or the intermittent fault detector changes state. While the main fault output is on they are only updated once the marker has been reached or passed and an input or the intermittent fault detector has changed state since they were last updated. This is because with the main fault output on, no changes can move Marker earlier, so it is safe to wait until it is reached before updating it. See REM statements for PROCfaulty and PROCOK on Sh. 595-14.

Each time round the loop Stable and Marker are checked to see whether the main output needs to change state.

Each time a new fault appears the intermittent fault detector checks whether the current timing period has expired. If it has the fault count is reset to zero and a new period is started. A new period is also started if the current period has not expired but this is the first fault since it started.

If the fault count exceeds NL% then an intermittent fault is declared, the count is reset and a new timing period is started. If no new faults appear the intermittent fault is considered to have cleared once this period has expired, but note that, as mentioned above, if a new fault does appear then a new timing period starts, which will have to expire before the intermittent fault is considered to have cleared. Thus one fault will delay the clearing of the intermittent fault even if not enough faults occur to retrigger the detector.

The variable InMark stores the earliest time at which any intermittent fault period is due to expire. By the time it is reached it may be incorrect, i.e. the period which was due to expire may have been curtailed and a new one started. In any case once it is reached all the periods for which an intermittent fault is currently declared are checked to see if they have expired and if so the declaration is cancelled. InMark is then reset for the next period that is due to expire.

3.3 Modifying the Program

The BASIC program held in EPROM is copied to RAM on switch-on and then run. With a terminal connected as mentioned in Para. 2 the program can be modified and the modified version then run. When an EPROM is required containing the modified program, the RAM contents may be sent to a PROM programmer (Para. 2) with the command SAVE.

The modifications that are most likely to be made to the program are alterations to the DATA tables at the end of the program. For convenience the data tables for coder and decoder are shown below, without line numbers but including an explanatory REM statement that is omitted from the EPROM version due to lack of space.

```
REM****************************** Coder DATA ************************
REM       TL%       TC%       NL%       FP%       TTG%      CHECK
DATA      -1,       0,        2,        20,       0,        Sync
DATA      -1,       0,        2,        20,       0,        Standby
DATA      2,        2,        2,        20,       10,       DataADP
DATA      2,        2,        2,        20,       10,       SigFail
DATA      -1,       0,        2,        20,       0,        Fldsync
DATA      5,        2,        2,        20,       10,       Mute
DATA      2,        2,        2,        20,       10,       ClkADP
DATA      2,        2,        2,        20,       10,       Parity
DATA      2,        2,        2,        20,       10,       Pulses
DATA      1,        0,        1,        20,       10,       ROMerr
DATA      2,        2,        2,        20,       10,       Buffov
DATA      1,        0,        1,        20,       10,       RAMerr
DATA      2,        2,        2,        20,       10,       Buffun
DATA      2,        2,        2,        20,       10,       Ctrlerr
DATA      2,        1,        2,        20,       10,       Insert
DATA      -1,       0,        2,        20,       0,        Spare1
DATA      -1,       0,        2,        20,       0,        Spare2
DATA      -1,       0,        2,        20,       0,        Spare3
DATA      -1,       0,        2,        20,       0,        Spare4
DATA      -1,       0,        2,        20,       0,        Spare5

REM       TSD       TTSG
DATA      3,        10

REM       BPon      BPoff
DATA      0,        0
```

```
REM DECDATA Stereo SiS Ver 0.2
AUTO9000
REM************************ Decoder DATA ***************************
REM       TL%       TC%       NL%       FP%       TTG%      CHECK
DATA      2,        2,        2,        20,       10,       Sync
DATA      2,        2,        2,        20,       10,       AmpRef
DATA      2,        2,        2,        20,       10,       DataRec
DATA      2,        2,        2,        20,       10,       SigFail
DATA      -1,       0,        2,        10,       10,       Fldsync
DATA      5,        2,        2,        20,       10,       Mute
DATA      2,        2,        2,        20,       10,       ClkBSR
DATA      2,        2,        2,        20,       10,       Parity
DATA      2,        2,        2,        20,       10,       Pulses
DATA      -1,       0,        2,        20,       0,        Spare1
DATA      2,        2,        2,        20,       10,       Buffun
DATA      -1,       0,        2,        20,       0,        Spare2
DATA      2,        2,        2,        20,       10,       Buffov
DATA      2,        2,        2,        20,       10,       Just
DATA      2,        2,        2,        20,       10,       DataFail
DATA      2,        2,        2,        20,       10,       DataLow
DATA      -1,       2,        10,       5,        10,       AGC
DATA      2,        2,        2,        20,       10,       DataHi
DATA      -1,       0,        2,        20,       0,        Spare3
DATA      5,        2,        2,        20,       10,       DACfail
```

Any other alterations to the behaviour of the main fault output would involve major changes and would probably mean a complete rewrite of the software.

The function of the secondary fault output (currently unused on the decoder) could be altered, though this should only be done by someone who is sure they know what they are doing.

The behaviour of the six open-collector outputs can be modified. They are simply controlled by combinational logic applied to the fault inputs. No timing is involved. If modification to this combinational logic is required the part of the program to be altered is the procedure called PROCassessC (for the Coder) or PROCassessD (for the Decoder). They use the function FNi(x), which returns the value TRUE if the input x is currently faulty and FALSE if it is not. BASIC variables with names describing each fault input have been set up with the required value of x, e.g. FNi(Sync) returns TRUE if line sync has failed. Various FNi(x) functions are ANDed and ORed together to determine the required states of the six open-collector outputs. The outputs are actually controlled by setting or resetting the six least significant bits of the variable OP. This is done by:

```
OP = OP OR y          to set a bit      (LED/output off)
```

or

```
OP = OP AND NOT y     to reset a bit    (LED/output on)
```

where

```
y =  1 for bit 0
     2 for bit 1
     4 for bit 2
     8 for bit 3
    16 for bit 4
    32 for bit 5.
```

It is vital that the two most significant bits of OP are not altered as these control the main and secondary fault outputs. If there is any doubt about whether they will be altered then the line:

```
TEMP = OP
```

should be put immediately after the line DEFPROCassess, and the line

```
OP = (OP AND &3F) OR (TEMP AND &C0)
```

immediately before the line ENDPROC. This will ensure that the two most significant bits have the same value when the procedure finishes as when it started.

Conditions such as (SWITCH AND 2) = 2 are included to ensure that apparent faults due to intentionally missing cards are ignored. The positions of Poles 1 to 4 of switch S3 are read into the variable SWITCH (bits 3 to 0 respectively) at the beginning of the program, ON having a bit value 0 and OFF a bit value 1.

For convenience PROCassessC and PROCassessD are listed on Sh. 595-9 and 10, without line numbers but with explanatory REM statements which are omitted from the version supplied in EPROM. Equally useful for implementing changes will be the logic diagram representations of PROCaccessC and PROCassessD shown on Sh. 595-11 and 12.

```
REM****************************** PROCassess (Coder) ********************************
REM Attempts to give some information about the cause of the problem when the
    system has been declared faulty. This information is given on the LEDs and
    o/c outputs. Note that if the inputs change this information will change
    immediately; it is not controlled by timers in the way that the main outputs
    are. The outputs have the following meanings:
REM output 5 The video input is failing intermittently. Note that a permanent
             failure is not considered as a fault, as the coder will switch to
             standby and the sound will be correctly transmitted. Switching to
             and from standby frequently will however break up the sound.
REM output 4 The incoming video already carries SiS; the video output of this
             unit will be unusable.
REM output 3 Either, i) Incoming line syncs are at the wrong rate,
REM                  ii) If using external data the data is not valid NICAM-728
                         data,
REM                 iii) If using the internal NICAM-728 coder, it is faulty.
REM output 2 Any of the signals (ie data, data clock, C0, C0 clock) from the
             source selected by the Clock & I/O module have failed.
REM output 1 The decoded bitstream is not a valid NICAM-728 bitstream, or has a
             high error rate.
REM output 0 There is a hardware fault.
REM NB hardware faults could of course cause any of the above indications to
    come on, but the listed faults are those most likely to be the cause of the
    problem.

DEFPROCassessC
DSPFail=((SWITCH AND 1)=1) AND (FNi(ROMerr) OR FNi(RAMerr))
NoSig  =FNi(SigFail)
BadWire=FNi(DataADP) AND NOT NoSig
NonNic =(FNi(Buffov) OR FNi(Buffun)) AND NOT FNi(DataADP) AND NOT SBF
Coded  =FNi(Pulse)
InVideo=INFLT AND (BITMASK%(Sync) OR BITMASK%(Fldsync))
SBfault=SBF OR SBW
InsFail=FNi(Insert) AND NOT NoSig AND NOT NonNic AND NOT Coded AND NOT InVideo

ProcClk=FNi(ClkADP)
BadNic =((FNi(Mute) OR FNi(Parity)) AND ((SWITCH AND 4)=4)) OR (FNi(Ctrlerr) AND
((SWITCH AND 1)=1))

OP=OP OR &3F
IF InVideo THEN OP=OP AND NOT 32
IF Coded THEN OP=OP AND NOT 16
IF NonNic THEN OP=OP AND NOT 8
IF NoSig THEN OP=OP AND NOT 4
IF BadNic THEN OP=OP AND NOT 2
IF SBfault OR DSPFail OR BadWire OR ProcClk OR InsFail THEN OP=OP AND NOT 1
ENDPROC
```

```
REM************************* PROCassess (Decoder) ******************************
REM Attempts to give some information about the cause of the problem when the
    system has been declared faulty. This information is given on the LEDs and
    o/c outputs.
REM Note that if the inputs change this information will change immediately,
    i.e. it is not controlled by timers in the way that the main outputs are.
REM The outputs have the following meanings:
REM output 5 Incoming field syncs are failing intermittently, probably means
             that coder is switching between normal and standby.
REM output 4 No incoming video, or SiS pulses are non existent or defective,
             e.g. wrong amplitude.
REM output 3 The bitstream error rate is high.
REM output 2 Any of the signals (ie data, data clock, C0, C0 clock) from the
             source selected by the Clock & I/O module have failed.
REM output 1 There is an audio decoding fault.
REM output 0 There is a hardware fault.
REM NB hardware faults could of course cause any of the above indications to
    come on, but the listed faults are those most likely to be the cause of the
    problem.

DEFPROCassessD
FldSyncInt=(INFLT AND BITMASK%(Fldsync))>0
Novideo=FNi(Sync)
Notcoded=FNi(Pulses) AND NOT Novideo
BadPulses=(FNi(AmpRef) OR FNi(DataLow) OR FNi(DataHi) OR FNi(Just)) AND NOT
Novideo AND NOT Notcoded
BadSignal=Novideo OR Notcoded OR BadPulses OR FldSyncInt
NoSig=FNi(SigFail)
NICerrors=((FNi(Buffov) OR FNi(Buffun)) AND NOT BadSignal AND NOT FNi(AGC)) OR
(NOT NoSig AND ((FNi(Mute) OR FNi(Parity)) AND ((SWITCH AND 1)=1)))
NoNICdata=NOT BadSignal AND (FNi(DataFail) OR FNi(DataRec))
DACfault=FNi(DACfail) AND ((SWITCH AND 4)=4)
Hardware=NoNICdata OR FNi(AGC) OR (FNi(ClkBSR) AND NOT NICerrors AND NOT
BadSignal)

OP=OP OR &3F
IF FldSyncInt THEN OP=OP AND NOT 32
IF Novideo OR Notcoded OR BadPulses THEN OP=OP AND NOT 16
IF NICerrors THEN OP=OP AND NOT 8
IF NoSig THEN OP=OP AND NOT 4
IF DACfault THEN OP=OP AND NOT 2
IF Hardware THEN OP=OP AND NOT 1
ENDPROC
```

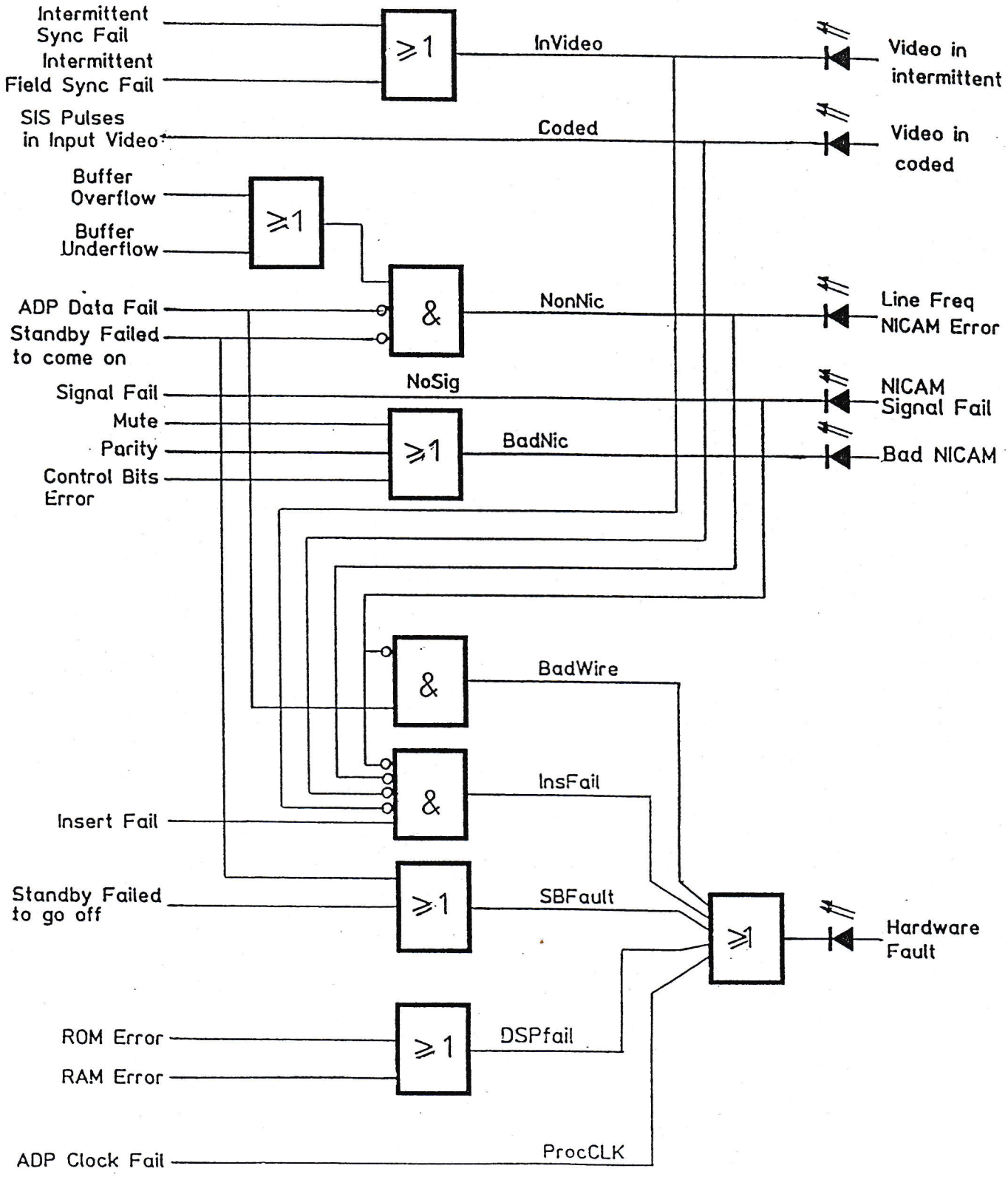

Fig. 1 Representation of PROCassessC

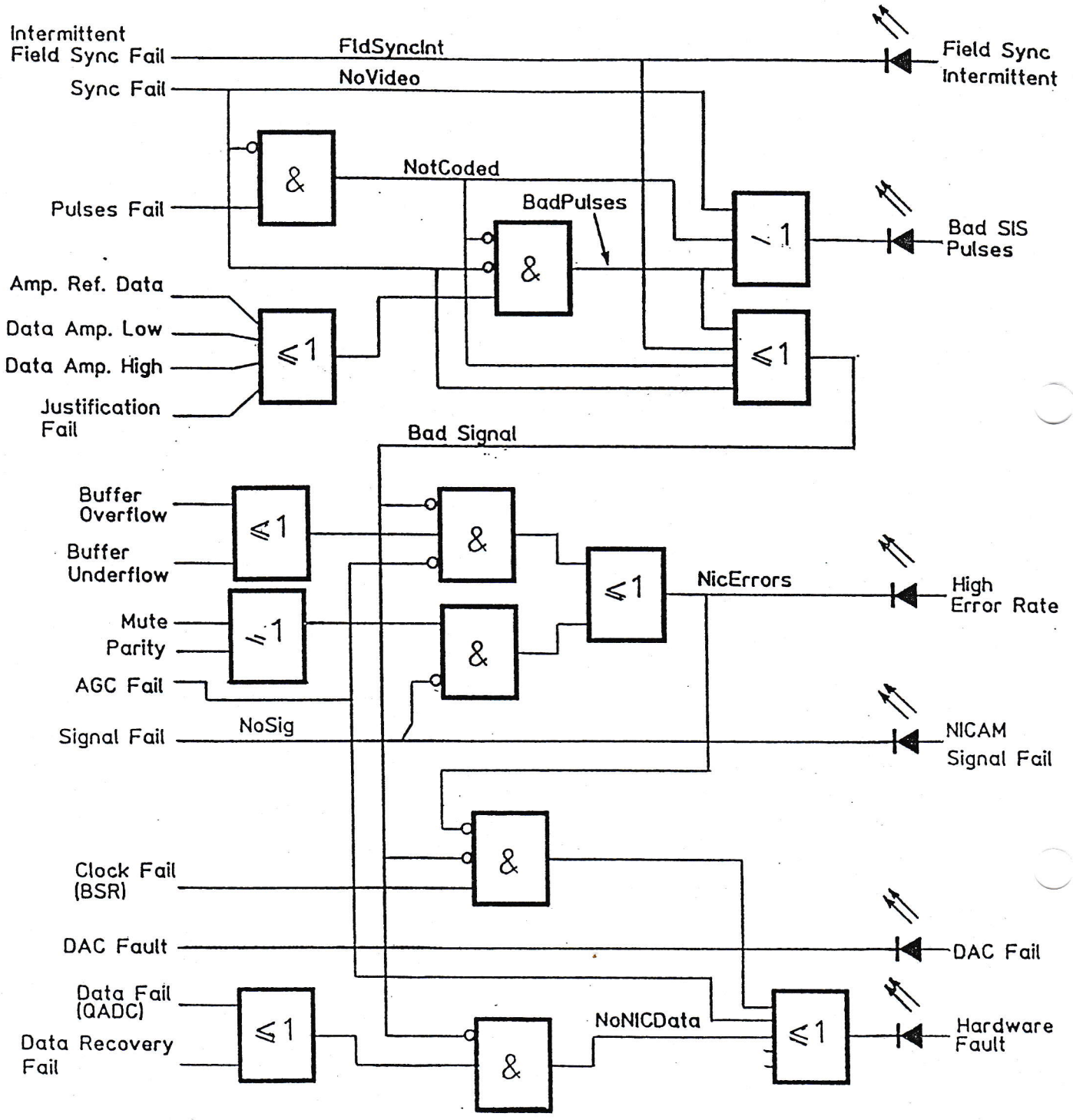

Fig. 2 Representation of PROCassessD

In several other cases REM statements have been omitted from the version of the program supplied in EPROM, so they are given below to aid understanding of the program.

```
REM********************************* MAIN LOOP ***********************************
REM Calls PROCpoll each time round the loop, which reads the inputs and keeps
    records of their behaviour. Also calls, when required, the procedures which
    decide whether the system is good or not and the procedures which control
    the outputs. Exactly which procedures are called depends on the state of
    various flags and timers.

REM* PROCstate controls the main output.
REM  It should only alter bit 7 of OP and leave the other bits unchanged.

REM************** PROCsecondary controls the secondary output.****************
REM  It should only alter bit 6 of OP and leave the other bits unchanged.

REM********************* PROCind controls the o/c outputs.************************
REM  It should only alter bits 0 to 5 of OP and leave bits 6 & 7 unchanged.

REM***************************** PROCpoll ************************************
REM Input monitoring procedure.

REM Returns the following information: FAULT contains current state of inputs,
    ie bit n =1 if input n is declaring a fault.
REM CHANGE indicates which inputs have changed state since PROCpoll was last
    called, ie bit n =1 if input n has changed.
REM INFLT indicates which inputs are considered to be intermittently faulty.

REM Further information is contained in the arrays TNF%( ) and TFC%( ).
REM TNF%(n) gives the time at which a new fault appeared on input n.
REM TFC%(n) gives the time when input n last went good.

REM Definitions:

REM "new fault": an input indicates a fault after being good for at least TC%(n)
    centiseconds, if the input has been good for less than that length of time
    the fault is considered to be a continuation of the previous fault.

REM "intermittently faulty": more than NL%(n) new faults have been counted in a
    period of FP%(n). Note that counting takes place in separate periods, so it
    is possible for more than NL%(n) faults to occur in a period of FP%(n) if
    some of them occur at the end of one counting period and some at the start
    of the next, with less than NL%(n) occuring in each period.
```

```
REM********************************* PROCfaulty ************************************
REM Should be called whenever an input or the intermittent fault detector
    changes state and the system good LED is on.
REM  It returns a flag, Stable, which is TRUE if the system is to remain in the
    good state indefinitely. Otherwise it also returns, in the variable Marker,
    the time at which the system should be declared bad, assuming no other
    changes take place before then.


REM********************************* PROCOK ************************************
REM When called returns the time, in Marker, at which the system can be declared
    good again, assuming no input changes before then. If there is no time at
    which the system can be declared good the flag Stable will be set TRUE and
    Marker will give the earliest time at which the system could go good
    assuming the inputs change as required, i.e. there is no need to call the
    procedure again before Marker is reached.


REM****************************** Time functions ********************************
REM Note the following conventions: The value of the real time clock is stored
    in the variable Tpoll at the time of polling the inputs and Tpoll is used as
    the current time throughout the program.
REM  The highest two bits of Tpoll are held at zero. The same goes for any other
    variable storing a time. This is so that modulo arithmetic can be performed
    without risk of overflow.

REM  FNtto(T) returns the time till T, which may be in the past or future. NB
    due to the modulo nature of TIME events that were in the past will
    eventually move into the future, this happens after about two months.


REM************************** Input testing function ****************************
REM FNi(n) is set TRUE if input n is currently showing a fault.


REM***************************** PROCpowerup ************************************
REM Provides a delay so that transient fault indications at switch on do not
    result in the system being declared bad for a long time; the system good
    output does not come on until this delay has expired.
```

3.4 Output Indications

Output	Coder	Decoder
Main	System Fail	System Fail
Secondary	Video Input Coded	Not Used
o/c 5	Video Input Intermittent	Field Syncs Intermittent
o/c 4	Video Input Coded	Bad SIS Pulses
o/c 3	Line Freq/NICAM Data Error	High Error Rate
o/c 2	NICAM Signal Fail	NICAM Signal Fail
o/c 1	Bad NICAM	DAC Fail
o/c 0	Hardware Fault	Hardware Fault

Note: Relays are energised (i.e. not 'normal') for system good. Open collector output polarity is ground for system good.

4. REAR PANEL CONNECTIONS

Coder and Decoder: 37-way D-type plug

Pin No.	Function
1	Open collector output 0
20	Open collector output 1
2	Open collector output 2
21	Open collector output 3
3	Open collector output 4
22	Open collector output 5
4	Earth (0V)
23	Earth (0V)
5	Main alarm relay - normally closed contact 1
24	Main alarm relay - common contact 1
6	Main alarm relay - normally open contact 1
25	Main alarm relay - normally closed contact 2
7	Main alarm relay - common contact 2
26	Main alarm relay - normally open contact 2
8	Secondary alarm relay - normally closed contact 1
27	Secondary alarm relay - common contact 1
9	Secondary alarm relay - normally open contact 1
28	Secondary alarm relay - normally closed contact 2
10	Secondary alarm relay - common contact 2
29	Secondary alarm relay - normally open contact 2
11	0V
30	Parity error flag output (open collector)
12	Not used
31	Not used
13	Not used
32	Not used
14	Not used
33	Not used
15	Not used
34	Not used
16	Not used
35	Audio Ch A output red (600 ohms balanced)
17	Audio Ch A output blue (600 ohms balanced)
36	0V
18	0V
37	Audio Ch B output red (600 ohms balanced)
19	Audio Ch B output blue (600 ohms balanced)