SGS-THOMSON MICROELECTRONICS

APPLICATION NOTE

SET-UP, HANDSHAKE AND COMMUNICATION MONITORING IN MODEM AND FAX MODE WITH THE ST75C502

By Laurent CLARAMOND

CONTENTS **Page**

I - INTRODUCTION

I.1 - Overall

The ST75C502 is a complete and powerful modem/fax data pump which provides all the facilities to establish modem connections from the V.32bis at 14400bps to the V.23 at 75/1200bps or 1200/75 bps and fax communications from V.17 at 14400bps to V.27ter at 4800bps.

Such a component is always used via a microcontroller (MCU). The software interface (often called modem_driver) between the MCU and the data pump will incorporate all the functions needed to use the component. To write these functions the engineer has to :

- understand how to perform the set up of the data pump,
- understand how the handshake is done and what kind of monitoring is required by the MCU,
- understand how the communication must be supervised using status given in real time by the data pump.

So the purpose of this application note is to provide all the information for the set up, the handshake and the communication monotoring. In other words, we would like to give the answers before the engineer asks for questions. Examples will be given each time they may help to understand.

I.2 - SATURN Modem Driver

SGS-THOMSON has developed an application board called SATURN. It looks like a stand alone modem/fax Hayes compatible which allows asynchronous and synchronous communications. We will take the examples from the modem driver written for this application.

I.3 - Notations

Any ***bold_italic*** command refers to reserved Name.

0x value means a value given in hexadecimal.

0b value means a value given in binary.

II - TYPICAL SOFTWARE FUNCTIONS FOR MODEM AND FAX COMMUNICATION

The main loop of the SATURN software uses the functions available in the modem driver called DP_ST.C. The set up, the handshake and the communication monitoring are separated in more than three functions (more than one for each task) to have a structured software which could be easily modified. To help the engineer to write his software we give some of these functions used in SATURN modem/fax application as example.

- Channel establishement :
  - dp_set_modulation_type()
  - dp_init_handshake()
  - dp_monitor_handshake()
  - dp_get_modulation_type()
  - dp_get_rate_sequence()
  - dp_analyzer()
- Line interface monitoring :
  - dp_carrier_detect()
  - dp_init_retrain()
  - dp_init_rate_renegotiation()
  - dp_detect_retrain()
- Fax operation :
  - dp_set_idle()
  - dp_set_v21_detector_fax()
  - dp_detect_v21_fax()
  - dp_detect_v21_in_high_speed_fax()
  - dp_facsimile_data_mode()

II.1 - Function Definition

dp_set_modulation_type() :
Configures the data pump to begin handshaking with the modulation type required. An argument could be used to determine if the data pump can fallback to slower speed if the communication cannot be established at the initial speed.

dp_init_handshake() :
Begins handshaking according to the configuration specified by dp_set_modulation_type().

dp_monitor_handshake() :
Monitors the progression of the handshake during channel establishment and give a message result. This function must be called repeatedly by the main loop until a final message is return.

dp_get_modulation_type() :
When in data mode, gets the data pump's current modulation type.

dp_get_rate_sequence() :
Give the last V.32, V.32bis rate sequence received.

dp_analyzer() :
This function is called repeatedly by the main loop. Some monitoring required by the data pump could be placed in this function.

dp_carrier_detect() :
This function tests if the received carrier is detected.

dp_init_retrain() :
This function requests to the data pump to initiate a retrain using the current configuration. This function could use two arguments, one to select the fastest modulation type to be used to start the handshake and one to allow a fallback to slower speed if the data pump can fallbak to slower speed if the communication cannot be established at the initial speed.

dp_init_rate_renegotiation() :
This function requests to the data pump to initiate a rate negotiation in V.32bis. An argument could be used to defined the new modulation type desired.

dp_detect_retrain() :
This function detects if a retrain or a rate negotiation initiated by the remote modem is in progress.

dp_set_idle() :
This function is only used in fax mode. The data pump returns in idle mode. In this mode, the data pump stops transmitting or receiving and waits for a new modulation type.

dp_set_v21_detector_fax() :
This function could be called by a fax module in originate. Prepare the data pump to detect V.21 ch2 modulation using dp_detect_v21_fax().

dp_detect_v21_fax() :
This function is only called by a fax module in originate. Interrogate the data pump about detecting V.21 ch2 modulation. This function must be called repeatedly by the main loop while sending CNG tone (ITU_T tone at 1100Hz).

dp_detect_v21_in_high_speed_fax() :
Function only used in fax mode. When the data pump is configured in high speed mode (V.27, V.29 and V.17) reception, this function tests if the remote modem is sending V.21 ch2 modulation.

dp_facsimile_data_mode() :
Function only used in fax mode. When the data pump is configured in high speed mode (V.27, V.29 and V.17) reception, this function detect if the data pump is in data mode.

III - FIRST STEP : SET-UP OF THE DATA PUMP

The set up of the ST75C502 is done using the two main commands ***CONF*** and ***MODC***.

The ***CONF*** command is essential since it is the only way to select one of the different operating modes for the Transmit and the Receive part.

The following table shows the possible operation mode :

CONF_OPER	Transmit	Receive
0000 *	Tones	Tones
0010	Voice	Tones
0100	Tone	DTMF
0110	Voice	DTMF
1000	Tones	Voice
1010	Voice	Voice
1111	Modem	Modem
Other	Not allowed	Not allowed

* Default value.

The ***CONF*** command needs 4 parameters which must be used each time. The parameters are byte size, byte 1 being the first and byte 4 the last. The Table 1 shows the meaning of the parameters.

The ***MODC*** command is used to modified default parameters value which will be used when the data pump will do the handshake and also to enable or disable some particular type of operation.

Here after we give some example of modifications that could be request by the application :
- the user can select two kind of guard tone in V.22 and V.22bis (answer mode only),
- the application requests a short train sequence instead a long train sequence in V.17 to send data while in C phase of the FAX T.30 protocol.

The selection of the modifications is done using 2 parameters (byte size) with the ***MODC*** command. The Table 2 shows the meaning of the parameters.

The purpose of the set up phase is to initialize with the good value some of the parameters of the ***CONF*** and ***MODC*** command. The others ***CONF***'s parameters and ***MODC***'s parameters will be initialized when the software will do the initialisation of the handsahke (select leased line, transmit equalizer, ...). Mainly the set up phase will select auto-mode or fixed mode and the wanted speeds.

Table 1

Field	Byte	Pos.	Value	Definition
CONF_OPER	1	3..0	-	Mode of operation see above
CONF_ANAL	1	4	0 1	Normal mode Analog loop back
CONF_PSTN	1	5	0 1	PSTN (carrier detect -43/-48dBm) Leased line (carrier detect -33/-38dBm)
CONF_AO	1	6	0 1	Answer mode (see Note) Originate mode (see Note)
CONF_MODE	2	5..0	0 1 2 3 4 5 6 7 8 9 A B C D Other	Automode Bell 103 Bell 212A V.21 V.23 V.22 V.22bis V.27ter V.29 V.17 V.32 V.32bis V.33 V.21 Channel 2 Reserved
CONF_TXEQ	2	7..6	0 1 2 3	No transmit equalizer Transmit equalizer #1 (1/2 of M1020) Transmit equalizer #2 (1/2 of M1040) Reserved
CONF_QAM	3	0	0 1	QAM/DPSK only (Automode) FSK allowed (Automode)
CONF_TCM	3	1	0 1	Treillis coding not allowed (V.32 only) Treillis coding allowed (V.32bis, V.32)
CONF_SP0	3	7..2	xxxx01 xxxx0x xxx10x xx1x0x x1xx0x 1xxx0x	300bps allowed(V.21, Bell 103) Reserved (must be set to 0) 1200bps allowed (V.22, V.22bis, V.23, Bell 212A) 2400bps allowed (V.22bis, V.27) 4800bps allowed (V.32bis, V.32, V.29, V.27) 7200bps allowed (V.32bis, V.29, V.17)
CONF_SP1	4	2..0	xx1 x1x 1xx	9600bps allowed (V.32bis, V.32, V.29) 12000bps allowed (V.32 bis, V.33, V.17) 14400bps allowed (V.32bis, V.33, V.17)

AN816-01.TBL

Table 2

Field	Byte	Pos.	Value	Definition
MODC_SH	1	6	0* 1	Normal training sequence Short training sequence
MODC_V22G	2	1..0	00* 01 10	No guard tone 1800Hz guard tone 550Hz guard tone
MODC_FPT	2	3..2	00* 10	No echo protection tone Long echo protection tone (180ms) (FAX only)
MODC_NOTA	2	4	0* 1	Answer : generate answer tone for handshake Originate : wait answer tone for handshake Answer : do not generate answer tone for handshake Orignate : do not wait answer tone for hansshake
MODC_NOSA	2	6	0* 1	Cut answer tone when receiving AA (V.32bis (short train sequence must be preceded by at least one normal training sequence), V.32). Continue answer tone when receiving AA
MODC_NOQA	2	7	0* 1	Enable V.32bis handshake on quality Disable handshake on quality

AN816-02.TBL

* Default value.

III.1 - Modem Set-up

We suggest the following flow chart (Figure 1) to illustrate the set up phase before making a handshake initialization. This flow chart explains the algorithm of the dp_set_modulation_type function() introduced on chapter II. This function uses two arguments (Mode_type, fallback) and set a flag (b_flag_half_duplex) :

- Mode_type : contains the wanted mode when not in automode.
- Fallback : 1 for automode, 0 for fixed mode.
- b_flag_half_duplex : 1 when half_duplex mode will be used, 0 for full_duplex mode.

Remark : At the end of the set up phase the dp_set_modulation_type()function only did the initialization of ***CONF***'s parameters and ***MODC***'s parameter without sending the command and the parameters to the data pump.

III.2 - FAX Set-up

In FAX mode the automode does not exist since we only use the V.21 ch2 in phase B, D and E of the T.30 protocol, and in phase C of the T.30 protocol the application will used the high speed modulation negociated in B phase (V.17, V.29 or V.27).

The possible configurations are :

V.21 ch2	CONF 0x0F 0x0D 0x00 0x00
V.17 14400	CONF 0x0F 0x09 0x00 0x04
V.17 12000	CONF 0x0F 0x09 0x00 0x02
V.17 9600	CONF 0x0F 0x09 0x00 0x01
V.17 7200	CONF 0x0F 0x09 0x80 0x00
V.29 9600	CONF 0x0F 0x08 0x00 0x01
V.29 7200	CONF 0x0F 0x08 0x80 0x00
V.29 4800	CONF 0x0F 0x08 0x40 0x00
V.27 4800	CONF 0x0F 0x07 0x40 0x00
V.27 2400	CONF 0x0F 0x07 0x20 0x00

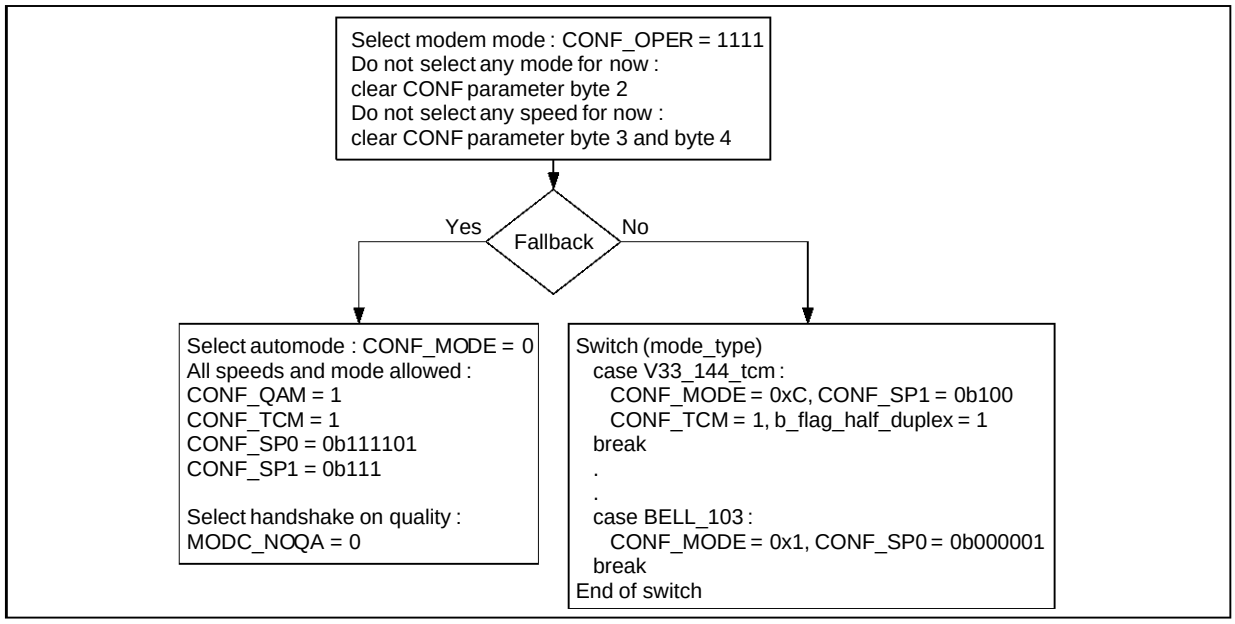
IV - SECOND STEP : HANDSHAKE

IV.1 - Modem Mode

First we must proceed to the initialization of the handshake to :

- send the ***CONF*** command with its parameters,
- send the ***MODC*** command with its parameters (optional),
- send the bulk command with its parameters in case of automode or V.32bis, V.32,
- enable ***IT1*** which is the bulk interrupt using the itmask register of the data pump,

Figure 1



- send the **DSIT** command with its parameters to select the status change which will generate **IT4** (status interrupt, optional),
- start the handshake using **HSK** or **SYNC** command (except for FSK full duplex),
- set some flag and variable for the software allowing the handshake monitoring (state variable of the state machine for handshake monitoring, ...).

We will make reference to the software function implemented in the SATURN application.

The following flow chart (Figure 2) illustres the algorithm of the dp_init_handshake() function introduced in chapter II. Dp_init_handshake() uses b_flag_half_duplex set by dp_set_mudulation_type().

The flag and variable set by dp_init_handshake() are :

- b_flag_hsh_started which will be used by dp_monitor_handshake(),
- automod_handshake_phase which will be used by dp_analizer() when monitoring repeatedly the status given by the data pump.

The next step is to make a monitoring of the handshake. This step is quite complicated since the handshake could be in automode (answer or originate) or in fixed mode. The monitoring is doing by the dp_monitor_handshake() function which must be called repeatedly by the main loop. Dp_monitor_hanshake() will return a result. Dp_monitor_handshake() uses direct status given by the data pump and the value of the state variable of a state machine implemanted in the dp_analizer() function to. This state machine using status given by the data pump.

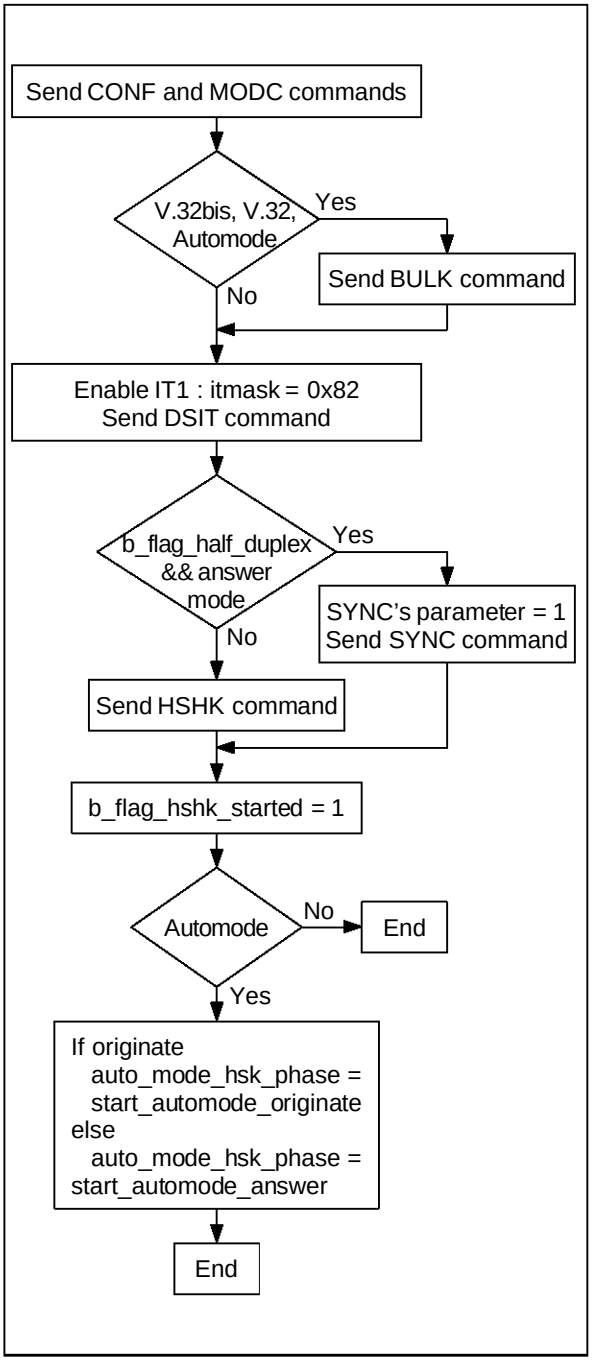
The possible values for automode_hsk_phase variable are :

- HANDSHAKE_FINISHED
- HANDSHAKE_FAILED

The data pump status used are bit0 and bit1 of the ***STATUS[1]*** byte . The 2 bits are grouped to make the b_sta_h variable which can have the values 0, 1 or 2.

Dp_monitor_handshake() will use the b_flag_hsk_started flag set by dp_init_handshake().

Figure 2



We propose the flow chart (Figure 3) of dp_monitor_handshake's algorithm as an example.

The monitoring of the handshake uses a state machine architecture placed in the dp_analyzer() function. The user could choose another architecture. To help the engineer to choose the architecture which will be the best in its software application we provide one flow chart for the automode algorithm in originate mode and one flow chart for the automode algorithm in answer mode.

A handshake progression counter contains information about the progress of the handshake in V.32bis, V.32 and V.22bis modes. This 8 bit value is available in ***STAOP2*** (optional status byte 2 of the Dual Port RAM). It can be read to examine the progression of the handshake and it contains normal and error values as show on the following tables :

Table 3 : V.22bis Originate Mode

EVENT	SHSK Normal Value
HSK	0x60
USCR1 DET	0x61
SCR1 DET	0x62
S1 DET	0x63
DATA MODE	0x70

Table 4 : V.22bis Answer Mode

EVENT	SHSK Normal Value
HSK	0x80
SCR1 DET	0x82
S1 DET	0x83
DATA MODE	0x90

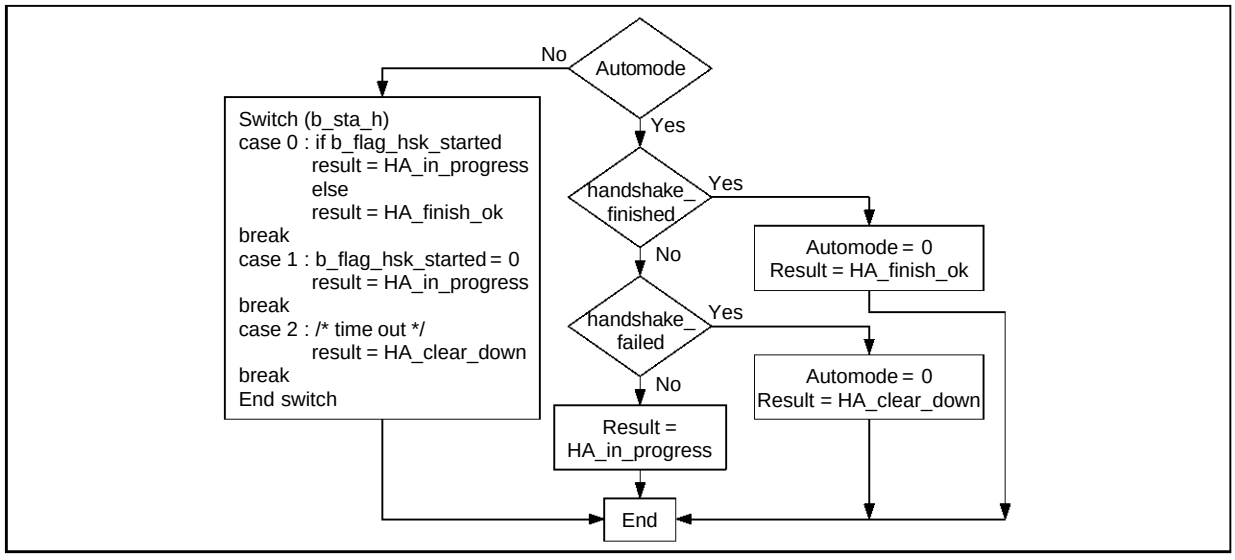
Table 5 : V.32bis Originate Mode

EVENT	SHSK Normal Value	SHSK Error Value
AC DET	0x20	
AC/CA DET	0x21	0x1
CA/AC DET	0x22	0x2
NO AC DET	0x23	0xB for RTRN, 0xC for RRN
S DET	0x24	0x4
SB DET	0x25	0x5
R1 DET	0x26	0x6
S DET	0x27	0x7
SB DET	0x28	0x8
R3 DET	0x29	0x9, 0xD no R5 det after RRN
E DET	0x2A	0xA
DATA MODE	0x30	

Table 6 : V.32bis Answer Mode

EVENT	SHSK Normal Value	SHSK Error Value
AA DET	0x40	0x8 for RTRN, 0x9 for RRN
AA/CC DET	0x41	0x1
NO CC DET	0x42	0x2
S DET	0x43	0x3
S DET2	0x44	0x4
SB DET	0x45	0x5
R2 DET	0x46	0x6, 0xA no R DET after RRN
E DET	0x47	0x7
DATA MODE	0x50	

Figure 3



The above informations are preceded by the following information (always given in ***STAOP2***) in automode mode.

Table 7 : Automode Originate Mode

EVENT	SHSK Value
Wait answer tone	0x01
Wait end of answer tone	0x02
No automode and waiting USC1	0x03
Automode waiting AC or USC1	0x04

Table 8 : Automode Answer Mode

EVENT	SHSK Value
Waiting HSK command	0x10
Generating answer tone	0x11
Generating silence	0x12

The ***_curmod*** variable gives the final negociated mode for Automode applications (especially useful for FSK) or data mode configuration. This variable is inside the ST75C502. We suggest to use the ***STAOP1*** (optional status 1 in the Dual Port Ram interface) during the handshake progression to monitor ***_curmod*** to avoid to use the ***MR*** (Memory Read) command. It will be easier for the modem driver to only read the ***STAOP1*** byte instead to send a ***MR*** command and then to read the answer. The Table 9 gives the description of _curmod is.

Figure 4 : Automode Originate

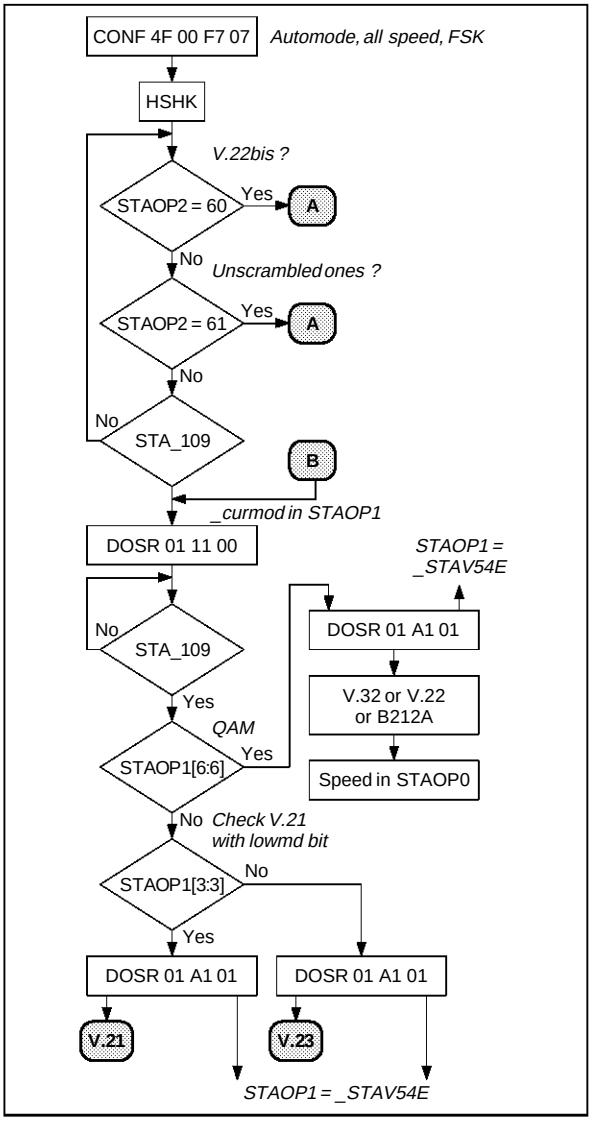


Table 9

Bit	Value	Description
D7 (ITU_T)	1 0	ITUT FSK modes Bell FSK modes
D6 (QAMMD)	1 0	QAM V.32bis, V.32, V.22bis, V.22, B212A, V.17, V.33, V.29, V.27 V.21, V.23, B103 FSK modes
D5 (TCDMMD)	1 0	Treillis mode (V.17, V.33, V.32(bis)) Non treillis mode
D4 (FDUMD)	1 0	Full duplex mode such as V.32(bis), V.22(bis), B212A, V.21, V.23, B103 Half duplex modes (fax) V.21 ch2, V.17, V.29, V.27, V.33
D3 (LOWMD)	1 0	V.27 or V.32 or V.22 or V.21 or B103 V.29 or V.32bis or V.22bis or V.23
D2 (ECCMD)	1 0	Echo canceller mode V.32bis, V.32 No echo canceller mode (others)
D1		Not used
D0 (ANSMD)	1 0	Answer mode Originate mode

AN816-03.TBL

Figure 5 : Automode Originate

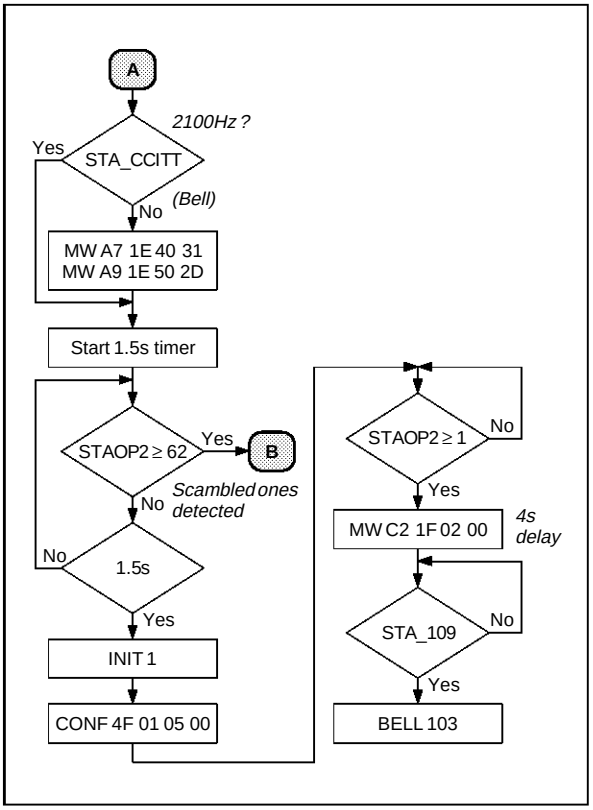


Figure 6 : Automode Answer

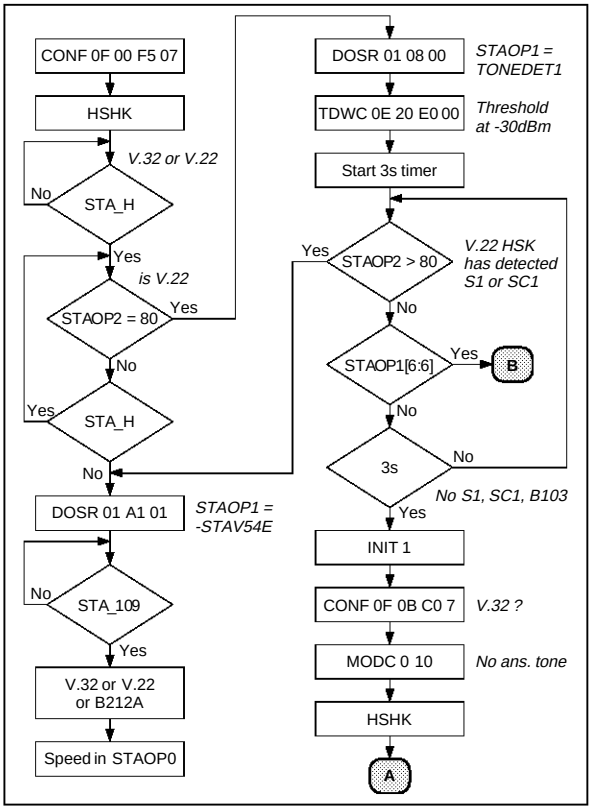


Figure 7 : Automode Answer

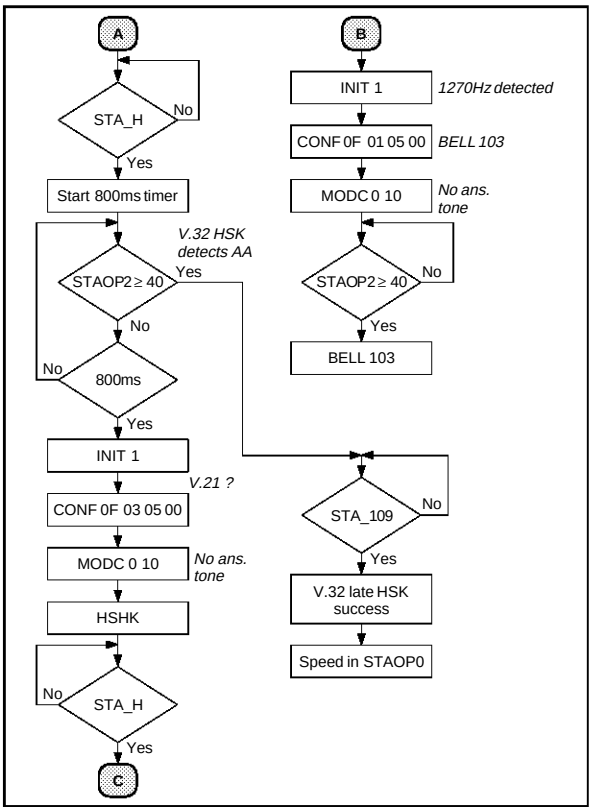


Figure 8 : Automode Answer

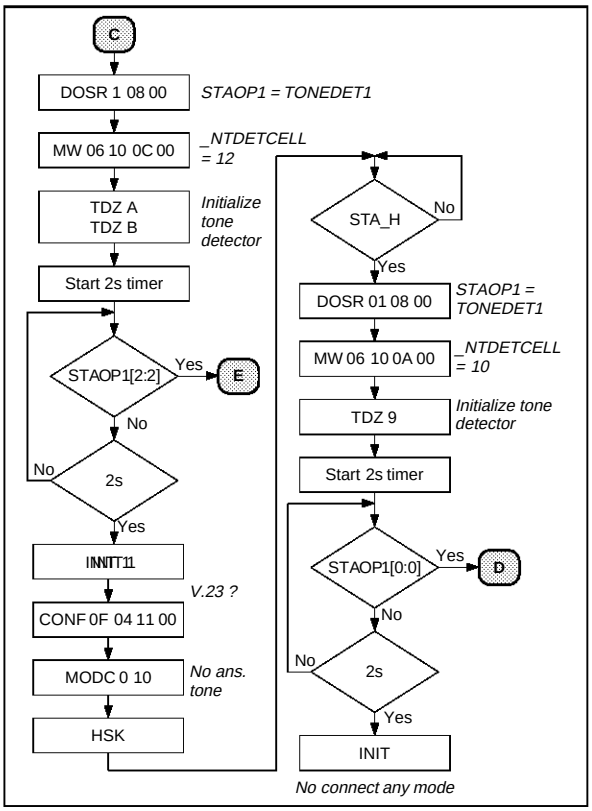
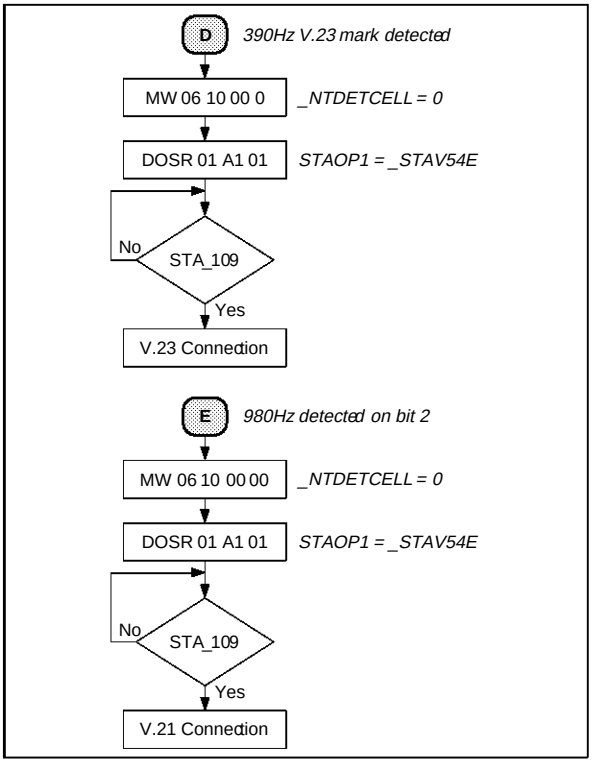


Figure 9 : Automode Answer



At the end of the handshake the MCU software place the modem board in data mode and normaly send a connect result to the terminal. Using the above information and the bit 4..1 of the ***STAOP0*** byte (optional status 0 in the Dual Port RAM) the MCU software will have all the information to send the good information to the terminal. The ***STAOP0*** information is only available for QAM or TCM modulation and provides the speed which is negotiated.

Table 10 : STAOP0 bit 4..1

Value of bit 4..1	Description
0	Reserved
1	Reserved
2	Negociated speed is 1200bps
3	Negociated speed is 2400bps
4	Negociated speed is 4800bps
5	Negociated speed is 7200bps
6	Negociated speed is 9600bps
7	Negociated speed is 12000bps
8	Negociated speed is 14400bps

IV.2 - Fax Mode

To send a carrier the only thing to do is to send the ***HSHK*** command. The user must set to the good value the ***MODC***'s parameter in case of V.17 to select the long train sequence in phase B and the short train sequence in C phase of the T.30 protocol.

The following table summarizes the parameters for the ***MODC*** command in all the case for the V.17, V.29 and V.27 modulations when the equipment is the sending unit :

Training	MODC's Parameters
Long train sequence with echo protection Long train sequence without echo protection	0x00 0x08 0x00 0x00
Short train sequence with echo protection Short train sequence without echo protection	0x40 0x08 0x40 0x00

To received a carrier the only thing to do is to send the ***SYNC 1*** command. The user must set to the good value ***MODC***'s parameters in case of V.17 to select the long train sequence in phase B and the short train sequence in C phase.

The following table summarizes the parameters for the ***MODC*** command in all the case for the V.17, V.29 and V.27 modulations when the equipment is the receiving unit :

Training	MODC's Parameters
Long train sequence	0x00 0x00
Short train sequence	0x40 0x00

V - THIRD STEP : COMMUNICATION MONITORING

V.1 - Modem Mode

While in data mode the MCU must monitoring the status given by the ST75C502 which indicate that a valid carrier is detected and that valid data are received. In all the mode execpt V.32 and V.32bis mode the status is given by the ***STA_109*** bit of the ***STATUS[0]*** byte in the Dual Port RAM interface (see above).

For the V.32 and V.32bis modulation two variables (***PWREST*** and ***RDQUA***) must be checked by the MCU.

PWREST : ***PWREST*** is the residual echo power estimate calculated by the DSP. This number is actually a correlation between the residual echo and the echo estimate. In the case of a normal connection with or without noise at any speed, the absolute value of ***PWREST*** should be near 0. If there is a local disconnect (local cable disconnect), this number's absolute value (***PWREST*** could be negative or positive) should be greater than 0x20. The address of ***PWREST*** is 1B AD.

RDQUA : We suggest detection of remote modem hang_up by monitoring the variable ***RDQUA***, equalizer error energy. ***RDQUA*** also gives an idea of signal to noise ration by the receiver. The address of ***RDQUA*** is 12 A7.

RDQUA has the following typical values in hexadecimal :

Value	RX SNR
0x0080	30dB
0x0100	27dB
0x0200	24dB
0x0400	21dB
0x0800	18dB
0x1000	15dB
0x2000	12dB
> 0x4000	Remote modem hangup

RDQUA may not be greater than 0x4000 for local cable disconnect as the local receiver may try to lock on to the large local transmit echo caused by impedance mismatching when the connection is brocken. Due to this, the quality may be quite good.

In conclusion, if both local cable disconnect and remote hang_ups are to be detected, both ***RDQUA*** and ***PWREST*** should be monitored by the MCU.

How to supervise the *RDQUA* and the *PWREST* value ?

As in data mode the ***STAOP0*** and ***STAOP2*** bytes in the dual port RAM are not used we suggest to used the ***DOSR*** command to request to the ST75C502 to put the LSB of ***PWREST*** in ***STAOP0*** and the MSB of ***RDQUA*** in ***STAOP2***. It will be easier for the MCU to read ***STAOP0*** and ***STAOP2*** instead to send twice the ***MR*** (Memory Read) command and to read each time the value given by the DSP in the ***COMREP0*** and ***COMREP1*** of the Dual Port RAM interface.

The following flow chart (Figure 10) explain how to use the ***DOSR*** command.

The monitoring of the carrier detection (hang_up detection) is made by the Dp_carrier_detect() function in SATURN.

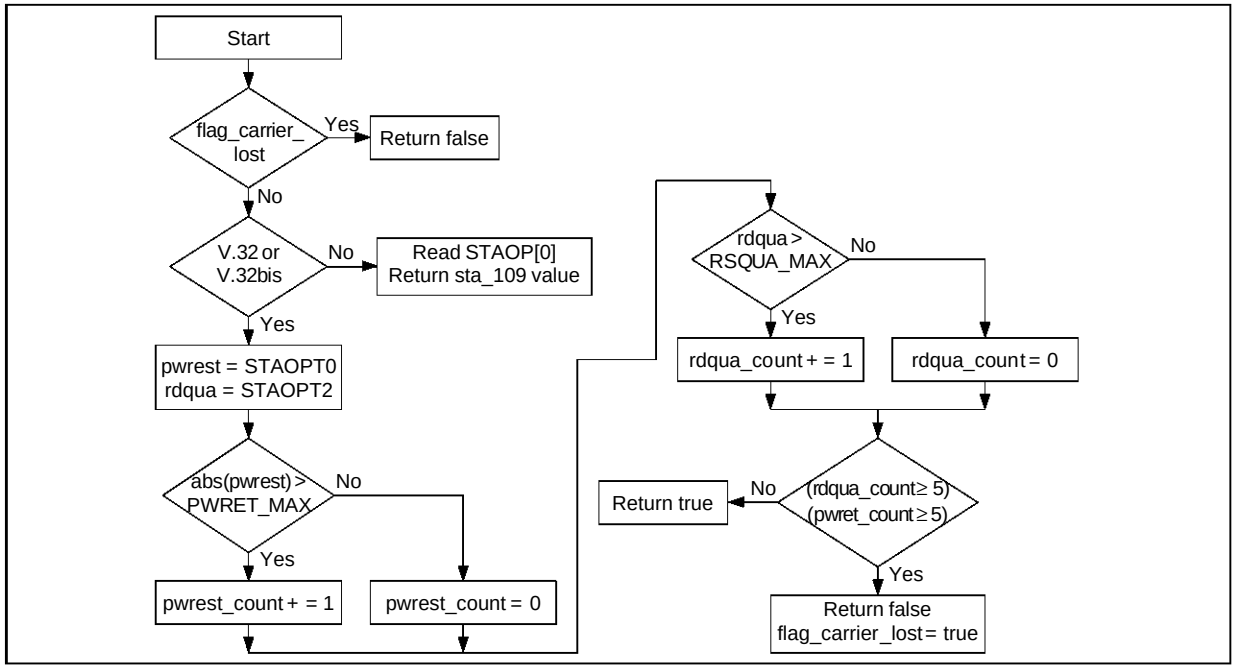
Dp_carrier_detect() is called by the main loop and return True if the carrier is detected and false if the carrier is not detected.

In case of no echo_cancellation mode the Dp_carrier_detect() function return the value of ths ***STA_109*** bit of the ***STATUS[0]*** byte in the Dual Port RAM interface.

In case of echo_cancellation mode (V.32 and V.32bis mode) Dp_carrier_detect() reads ***STAOT0*** and ***STAOPT2*** than counts the number of time the ***STAOPT0***'s value or ***STAOPT2***'s value is higher than a threshold. If ***STAOPT0***'s value or ***STAOPT2***'s value is five times higher than a threshold Dp_carrier_detect() return false. The ***PWREST***'s threshold PWREST_MAX is equal to 0x2A, and the ***RDQUA***'s threshold RDQUA_MAX is equal to 0x40.

Figure 10

Figure 11



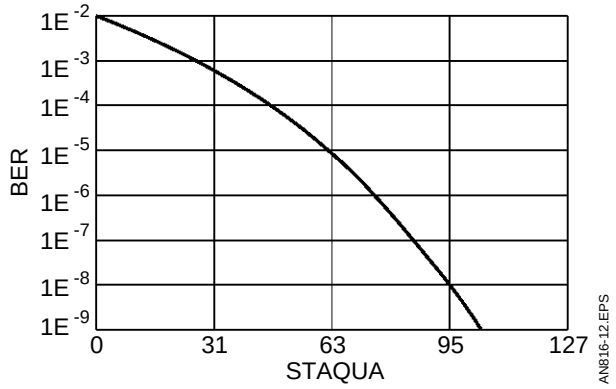
We suggest the flow chart of these function as example (Figure 11).

Note : flag_carrier_lost is set to fasle at the beginning of the data mode (just at the end of the handshake).

The application software could also monitor the Quality Status provide by the ST75C502.

The quality byte ***STAQUA*** (address 0x0B in the Dual Port RAM interface) monitors an evaluation of the line quality. It is updated once per baud and its value ranges is from 127 (perfect quality) to 0 (terrible quality). The value is automatically adjusted according to the current receiving mode (not valid in FSK modes). Refer to the Figure 12 to convert the value into its bit error rate equivalence.

Figure 12



V.2 - Fax Mode

In fax mode the monitoring is simpler. The carrier detect information is provide for all the modes by the ***STA_109*** bit in ***STAOP0*** byte.

The software could monitor the quality reading the ***STAQUA*** byte as in modem mode.

VI - SPECIAL FUNCTIONS FOR FAX COMMUNICATIONS

A fax application needs some function as V.21 ch2 flag detection, stop to send or to receive carrier, ... In SATURN the following software function group these needs : **Dp_set_idle(), Dp_set_v21_fax(), Dp_detect_v21_fax(), dp_detect_v21_in_high_speed_fax()**. We are going to describe in this chapter the algoritms of all these function. Refer to the definition of these functions given in chapter II.

VI.1 - Stop Transmit or Receive

The ***STOP*** command must be used to stop the carrier transmission, and the ***SYNC*** command with its parameter equal to 0 must be used to stop carrier reception. This is done by dp_set_idle() function.

Stop transmit : send STOP.

Stop reception : send SYNC 0.

VI.2 - Flag Detection when the Data Pump is Set-up for High Speed Reception

The application must be able to detect V.21 ch2 flag when the data pump is set up to received in high speed mode (V.27, V.29, V.17). As no particular function is available in the ST75C502 for that purpose the user has to use 2 tone detectors : tone detector cell number 0 and 1 are the only available tone cells in that case. The status given by these tone detectors will be in ***TONEDT0*** byte on bit 0 and 1. In data fax mode the ***STAOP1*** byte is available. That's why we suggest to the user to monitor ***TONEDT0*** byte via ***STAOP1*** byte using ***DOSR*** command.

The final sequence for the application will be :

- Send the ***CONF*** command to select V.17, V.29 or V.27,
- Send ***SYNC 1*** command,
- Set up the cells 0 and 1 of the tone detector,
- Use ***DOSR*** command for monitoring the ***TONEDT0*** status.

Each time the application will need to know if the far end FAX equipment send a V.21 ch2 flag sequence (preamble), the MCU will have to read the ***STAOP1*** byte and test if bit 0 and 1 are equal to 1. This is done in SATURN application by the dp_detect_v21_in_high_speed_fax()function.

Here after we provide the information to set up the tone detector cell 0 and 1.

The user has first to set up this tone detector as shows below :

/* select the reference level at - 40dBm */
TDWC's parameter 1 = 0x00 ; /* tone detector cell number */
TDWC's parameter 2 = 0x20 ; /* select static level */
TDWC's parameter 3 = 0x70 ; /* Low byte of the coefficient */
TDWC's parameter 4 = 0x00 ; /* High byte of the coefficient */
send ***TDWC*** command

/* Load the coefficients of the two biquad to detect 1650Hz using the ***TDWC*** command */

Each tone cell needs 12 coefficients given in the following table the cells 0 and 1. The coefficients are 16 bits size. The table contains 24 bytes for each cell. The table starts with the MSB of the first coefficient, then the LSB of the first coefficient, then the MSB of the second coefficient, then the LSB of the second coefficient, ...

MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB
0x01	0xC1	0x0F	0x35	0xC1	0xC3	0x00	0x00	0xC0	0x00	0x7F	0xFF
0x02	0xAB	0x11	0xC0	0xC1	0xC2	0x00	0x00	0xC0	0x00	0x7F	0xFF
0x01	0xE7	0x0D	0x83	0xC0	0xBD	0x00	0x00	0xC0	0x00	0x7F	0xFF
0x05	0x4F	0x13	0xB2	0xC0	0xBC	0x00	0x00	0xC0	0x00	0x7F	0xFF

/* Programme biquad and energy input for cell number 0 */
TDWW's parameter 1 = 0 ; /* cell number 0 */
TDWW's parameter 2 = 0 ; /* biquad energy input */
TDWW's parameter 3 = 0x11 ; /* energy estimator input */
TDWW's parameter 4 = 0x02 ; /* biquad filter signal input */
send ***TDWW*** command

/* Programme biquad and energy input for cell number 1 */
TDWW's parameter 1 = 1 ; /* cell number 1 */
TDWW's parameter 2 = 0 ; /* biquad energy input */
TDWW's parameter 3 = 0x01 ; /* energy estimator input */
TDWW's parameter 4 = 0x10 ; /* biquad filter signal input */
send ***TDWW*** command

/* Programme comparator inputs cell number 0 */
TDWW's parameter 1 = 0 ; /* cell number 0 */
TDWW's parameter 2 = 1 ; /* comparator input */
TDWW's parameter 3 = 0x30 ; /* negative comparator signal input */
TDWW's parameter 4 = 0x20 ; /* positive comparator signal input */
send ***TDWW*** command

/* Programme comparator inputs cell number 1 */
TDWW's parameter 1 = 1 ; /* cell number 1 */
TDWW's parameter 2 = 1 ; /* comparator input */
TDWW's parameter 3 = 0x21 ; /* negative comparator signal input */
TDWW's parameter 4 = 0x20 ; /* positive comparator signal input */
send ***TDWW*** command

/* clear the tone detector cell 0 */
TDZ's parameter = 0 ; /* cell number 0 */
send ***TDZ*** command

/* clear the tone detector cell 1 */
TDZ's parameter = 1 ; /* cell number 1 */
send ***TDZ*** command

Here after we provide the parameters for the ***DOSR*** command to monitor ***TONEDT0*** in ***STAOP1*** byte :
DOSR's parameter 1 = 0x01 ;
DOSR's parameter 2 = 0x07 ;
DOSR's parameter 3 = 0x00 ;
DOSR's parameter 4 = 0x00 ;
send ***DOSR*** command

Figure 13

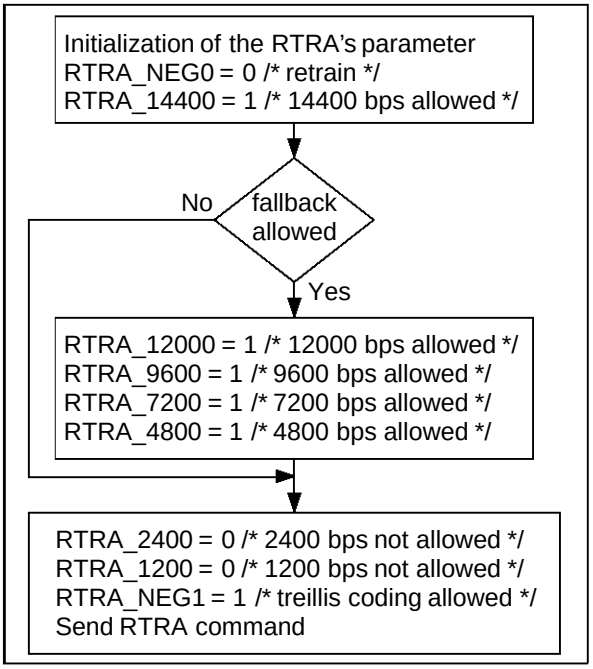
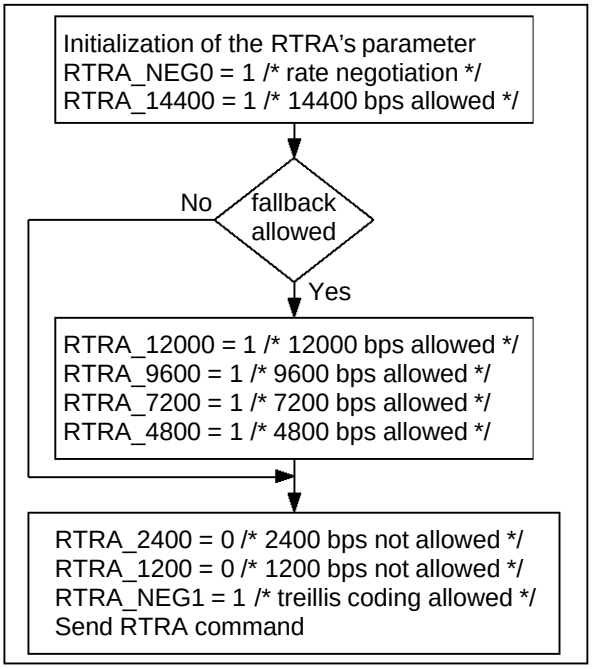


Figure 14



VI.3 - Flag Detection when the Application is Calling a Far End Fax Equipment

The equipment which is the calling unit must be able to detect V.21 ch2 modulation while sending the CNG tone (ITU_T 1100Hz tone). In this case the ST75C502 will be set up in tone mode and will provide on bit 2 of the ***STAOP1*** byte (in the Dual Port RAM interface) a status which indicates if a frequency of 1650Hz greater than -45dBm id detected.

The MCU has only to read the ***STAOP1*** byte and to test the bit 2 value. In SATURN application this done by the dp_detect_v21_fax()function.

VII - RETRAIN AND RATE RENEGOTIATION

The ST75C502 allows the application to initialize and to detect retrain or rate renegotion.

The Figure 13 explain how to initialize a retrain in V.32bis. The principle is the same in V.32 and V.22bis.

The Figure 14 explain how to initialize a rate renegotiation in V.32bis 14400. The principle is the same in V.32 and V.22bis.

After the retrain or rate renegotiation initialization, the monitoring must be done by testing the bits ***STA_H*** and ***STA_TIM*** of the ***STATUS[1]*** byte.

STA_H = Transmit synchronisation in progress. Valid only in modem mode.

STA_TIM = Handshake timeout. Valid only in data modem mode.

STA_TIM, STA_H
0 Synchronisation Completed
1 Synchronisation in Progress
2 Synchronisation Time Out

The speed negociated is given in bit 4..1 of ***STAOP0*** (see chapter IV.1 of this application for information).

The detection of a remote retrain or rate renegotiation is done by testing ***STA_RTRN*** and ***STA_RNEG*** bits in ***STATUS[1]*** byte. The following flow chart (Figure 15) give the algorithm of the dp_detect_retrain() function in SATURN application.

Figure 15