THE DEVELOPMENT OF SOFTWARE FOR A FAXMODEM CARD

By Christian Phillip Stuurman

Thesis submitted in partial fulfillment of the

requirements for the Master Degree in Technology to

the Department of Electrical Engineering

at the Cape Technikon.

June 1995

CAPE TOWN

SOUTH AFRICA

DECLARATION

I declare that the contents of this thesis represents my own work and the opinions contained here are my own. It has not been submitted before for any examination at this or any other institute.

C.P. STUURMAN

[illegible]

SIGNATURE

JUNE 1995

ABSTRACT

This thesis describes the design and development of software for a FAXMODEM card for a personal computer. The software was developed to enable the designed FAXMODEM card to transmit and receive facsimiles independent of the host personal computer.

OPSOMMING

Hierdie verhandeling beskryf die ontwerp en ontwikkeling van sagteware vir 'n FAXMODEM kaart vir 'n persoonlike rekenaar. Die sagteware is so ontwerp om die FAXMODEM kaart in staat te stel om geheel onafhanklik van die persoonlike rekenaar fakse af te stuur of te ontvang.

JUNE 1995

ACKNOWLEDGMENTS

I would like to thank the following people for their assistance during the completion of this project:

Mr. P.H. Kleinhans of Cape Technikon for the commissioning of this project and for providing me with the required development and test equipment.

My numerous colleagues at the Special Investigation Section and Regional Development Centre, TELKOM, Cape Town for their support.

TELKOM SA for providing the opportunity, the facilities and resources to complete this project.

I also thank my wife Zelna, son Roderique and daughter Zelmique for supporting me during this project.

Contents

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

Introduction

The introduction of the office facsimile machines made it possible to send images of documents from your office desk to another facsimile machine anywhere in the world, provided that there is a telephone connection, fast and efficiently. Today most offices and even homes are equipped with a personal computer or computer terminal. Documents like letters, reports, statistical data, graphs, etc. are often created on a personal computer and stored on a mass storage device (hard disk). Whenever a document needs to be sent via a facsimile machine, a hard copy of that document needs to be made first. The 'FAXMODEM' card enables you to send the same document to the remote facsimile machine directly from your personal computer. By eliminating the scanning stage it is now possible to send crisp clear fax images, i.e. images without smears, distortion and other scanning imperfections to a remote fax machine.

CHAPTER 2

Objective

This thesis describes the development of software for a FAXMODEM card designed by a fellow student.

FAXMODEMS are currently using the host CPU, i.e. CPU on the personal computer's motherboard, to handle the sending and receiving of faxes. The processes managing fax reception and transmission are executed in the background by the CPU of the personal computer. These processes are usually invoked by interrupting the CPU. Since sending and receiving of facsimile data are taking place in real time (i.e. data does not wait for the CPU), interrupts from the FAXMODEM have a higher priority. The interrupts from the keyboard and other PC resources are therefore put on hold while interrupts from the FAXMODEM are being serviced. The result is a slowing down of the normal CPU foreground processes.

A FAXMODEM with enough intelligence is needed to release the PC's CPU completely from the tasks involved with the receiving and sending of fax data.

A FAXMODEM card, equipped with its own CPU, 896K RAM and 128K ROM was designed and built. This card therefore had enough hardware to perform the majority of facsimile related tasks currently being performed by the PC's CPU.

As the hardware of this card was unique in its design no software existed which could service the FAXMODEM card.

Mr. P. H. Kleinhans from the School of Electrical Engineering at the Cape Technikon commissioned this project with the following main objectives:

- To develop software that will control the FAXMODEM card enabling it to send and receive facsimile data independent of the PC.
- To develop software that will run on the PC to enable it to communicate to the FAXMODEM card, sending it commands and facsimile data retrieved from the hard disk and storing facsimile data on the hard disk received from the FAXMODEM, without interfering with any programs being executed.

CHAPTER 3

Facsimile Fundamentals

3.1 Transmission Medium

The FAXMODEM operates over the public switched telephone network (PSTN) of which the bandwidth lies between 300 and 3400 Hz. This creates an upper limit for high speed data transfers. Low quality lines also have the effect of attenuating higher frequencies more than lower frequencies. The FAXMODEM's DSP chip (R96EFX) includes three cable equalizers which can be used to improve performance when operating over low quality lines. These cable equalizers which work in both transmit and receive paths have the following characteristics:

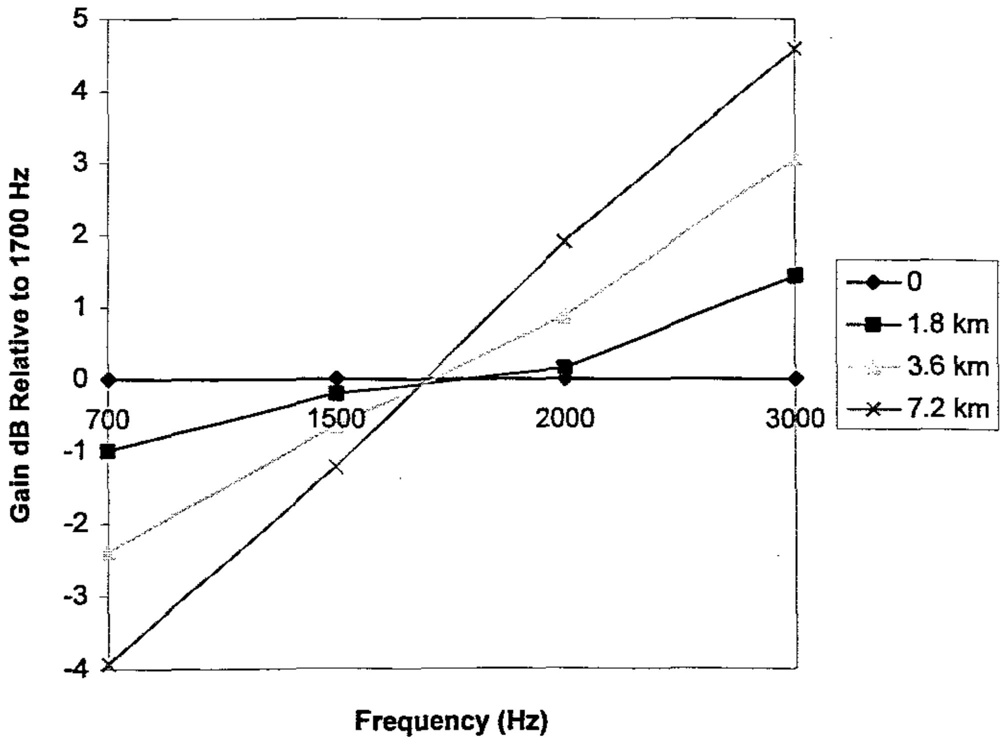


Figure 3-1 Cable Equalizer Characteristics

Table 3-1 shows the Cable1 and Cable2 configuration settings to select the required compromise equalizer.

Table 3-1 Configuring of cable equalizers

Cable2	Cable1	Equalizer
0	0	**0 km**
0	1	**1.8 km**
1	0	**3.6 km**
1	1	**7.2 km**

Facsimile apparatuses make use of ITU-T Recommendation V.29 and ITU-T Recommendation V.27*ter* for high speed image data transmission. In V.29 and V.27*ter* modes the R96EFX DSP uses an adaptive equalizer to compensate for transmission line amplitude and group delay distortion. The adaptive equalizer is set up automatically during the training sequence.

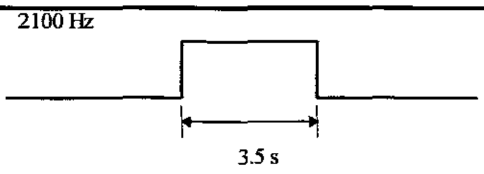
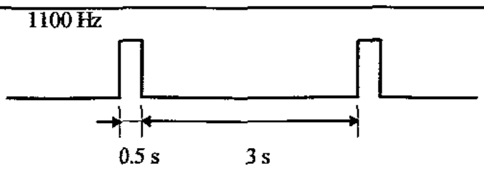
3.2 Signaling

Two types of signaling viz. binary coded and tonal, are used to interchange information between facsimile apparatus via the general switched telephone network. These signals provide the means of verifying compatibility between facsimile apparatus and therefore ensuring reliable and proper operation.

3.2.1 Tonal signaling

Tonal signaling is used by Group 1 and Group 2 facsimile apparatus for identification, confirmation and command information interchange and Group 3 facsimile apparatus for initial identification.

Table 3-2 Tones generated by the FAXMODEM card

Signal	Format	Function
Called station identification (CED)		Indicates a called non-speech terminal
Calling tone		Indicates a calling non-speech terminal

3.2.1.1 *Tone generation*

The R96EFX DSP chip can generate voice-band single or dual tones from 0 Hz to 4800 Hz with two built-in programmable tone generators. The dual tone generation allows for DTMF dialing. Single tones are generated by setting the frequency or amplitude of one of the generators to zero while the other generator is set to the desired frequency and amplitude. Equation 3-1 is used to obtain the value that should be written to the R96EFX to set the frequency. Equation 3-2 calculates the value for the power level.

Equation 3-1

$$N = 6.8267 \times (Frequency\ in\ Hz)$$

Equation 3-2

$$N = 18426 \times 10^{\frac{P_o}{20}}$$

P_o = desired output level in dBm with a 600 ohm termination.

3.2.1.2 *Tone detection*

The R96EFX DSP chip includes three programmable tone detectors. Each detector consists of two second-order filters and an averaging filter in cascade. The tone detectors can be tuned to any frequency in the 400 Hz to 3 kHz band. See Appendix C for the equations that were used to calculate the tone detector coefficients for the FAXMODEM.

3.2.2 Binary coded signaling

This type of signaling is used by Group 3 type of facsimile machines to interchange commands and information about identification and capabilities. The standard rate is 300 bps FSK using Rec V.21 channel No. 2 modulation. A signaling rate of 2400 bps DPSK using Rec V.27*ter* is optional. A preamble, i.e. a series of flags for one second in 300 bps or a long training sequence in 2400 bps mode, precedes all binary coded signaling for each line turnaround.

3.2.2.1 HDLC

All binary coded facsimile control procedures use the high level data link control (HDLC) frame structure. Figure 3-2 is an example of a binary coded signal in HDLC format.

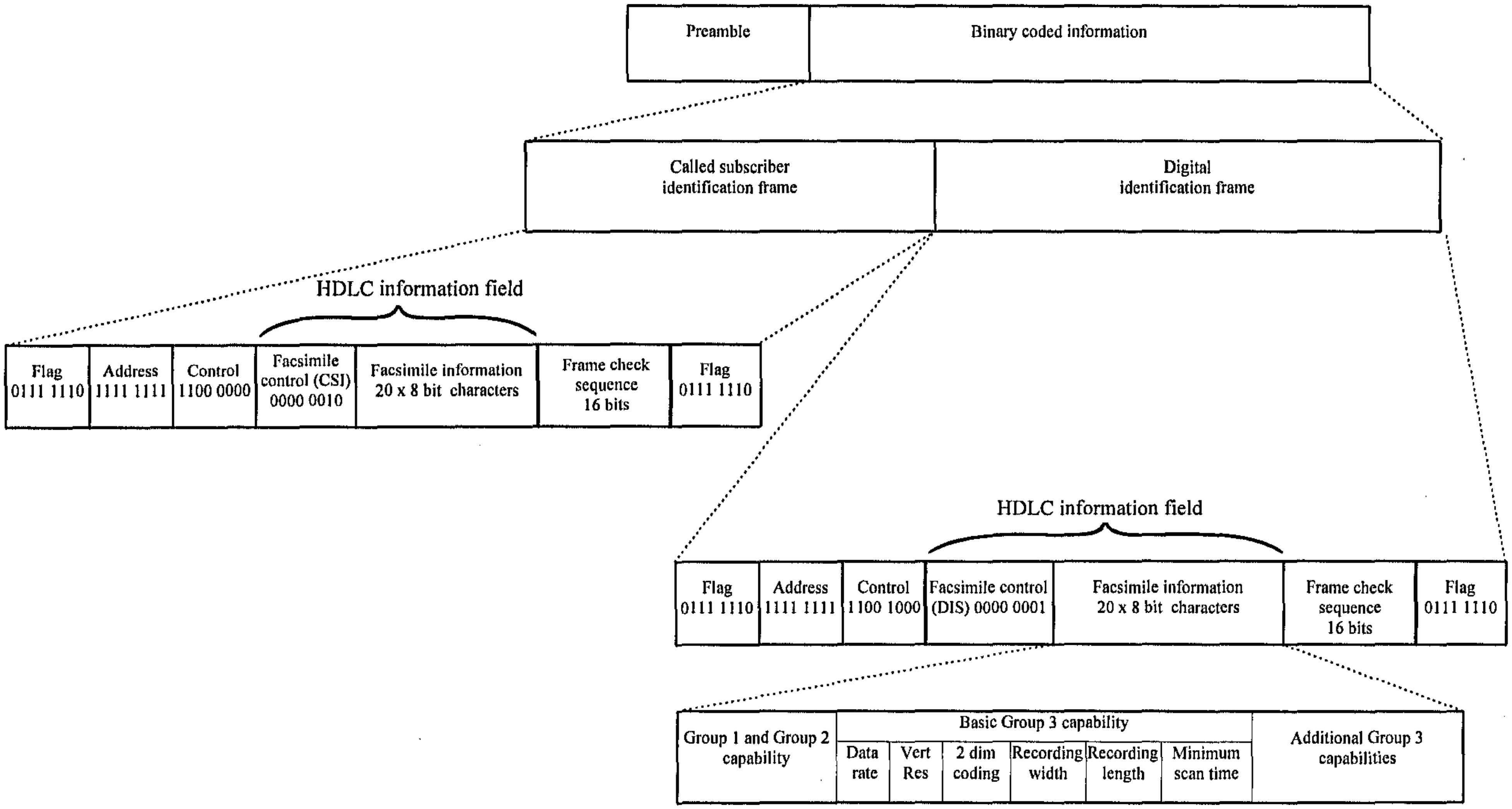

Preamble
Binary coded information
Called subscriber identification frame
Digital identification frame
HDLC information field
Flag 0111 1110
Address 1111 1111
Control 1100 0000
Facsimile control (CSI) 0000 0010
Facsimile information 20 x 8 bit characters
Frame check sequence 16 bits
Flag 0111 1110
HDLC information field
Flag 0111 1110
Address 1111 1111
Control 1100 1000
Facsimile control (DIS) 0000 0001
Facsimile information 20 x 8 bit characters
Frame check sequence 16 bits
Flag 0111 1110
Group 1 and Group 2 capability
Basic Group 3 capability
Data rate
Vert Res
2 dim coding
Recording width
Recording length
Minimum scan time
Additional Group 3 capabilities

Figure 3-2 HDLC frame structure

3.3 Image data compression

A bi-level (black and white) raster image in its uncompressed form is an array of data bits. Each bit represents one picture element or a pixel. This is a very inefficient representation of most images. The image data of the example facsimile page in Appendix E consist of 1728 x 2248 = 3874176 pixels. In uncompressed format it would consume $\frac{3874176}{8}$ = 484272 bytes. At 9600 bits per second (bps) the transmission of this image will take $\frac{484272 \times 8}{9600} \approx 404$ seconds, i.e. 6 minutes and 44 seconds.

Image data is therefore compressed to shorten transmission times.

3.3.1 The one dimensional Rec T.4 modified Huffman format

This is a compression scheme that converts rows of pixels into a series of variable length code words. Each variable code word represents a black or white run length. After modified Huffman compression the image data of the example facsimile page in Appendix E consumes only 39786 bytes. The compression ratio is therefore equivalent to $484272{:}39786 \approx 12{:}1$. At 9600 bps the transmission time will be $\frac{39786 \times 8}{9600} \approx 33$ seconds.

CHAPTER 4

Software Design Considerations

This chapter describes how the design of the software was influenced by the differences between the capabilities and hardware of existing FAXCARDS and the designed FAXCARD.

4.1 Communication protocol between PC and FAXCARD

A Group 3 facsimile communication path can be partitioned into a number of functional levels (see Figure 4-1). The protocols that are used between the PC and the FAXMODEM are grouped into classes. The class with the highest number being the class of FAXCARD with the highest level of capability.

4.1.1 Class 1

Service class 1 is defined in ANSI/TIA/EIA-578 and described in the draft recommendation T.CLASS1 of the ITU. In this class of operation the FAXMODEM provides a very basic level of services necessary to support Group 3 facsimile operation. This requires the PC to provide for signaling, HDLC data formatting, ITU Rec T.30 session management and ITU Rec T.4 or T.6 image data handling. This class of operation therefore requires a very large amount of CPU time.

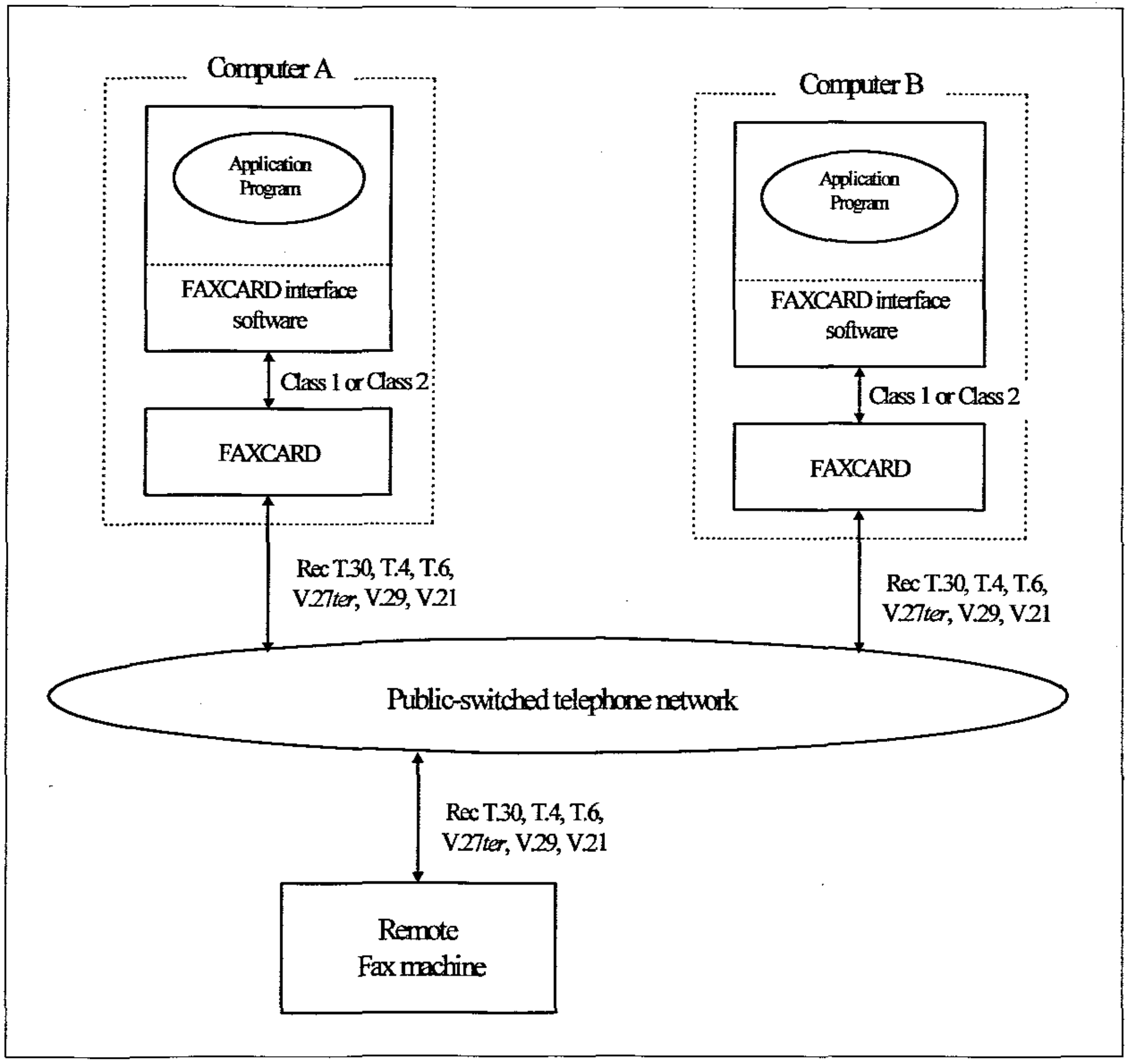
4.1.2 Class 2

Service class 2 is described in the draft recommendation T.CLASS2 of the ITU. A service class 2 FAXMODEM is responsible for conducting ITU Rec T.30 facsimile sessions and mandatory ITU Rec T.4 services. The PC prepares (e.g. separate into pages) and interprets image data in compressed form as described in ITU Rec T.4 or T.6. It then transfers this image data as required by ITU Rec T.30 on request from the FAXCARD. The FAXCARD makes and terminates calls, manages the communication session and optionally converts between T.4 and T.6 image formats. In this class of operation the PC is required to perform fewer tasks than in class 1.

4.1.3 Desired level of capability for FAXMODEM

Although a service class 2 FAXCARD provides a higher level of capability than class 1 FAXCARDS, the PC is still responsible for delivering data to the FAXCARD fast enough to maintain an uninterrupted flow of data in Phase C. The PC is also responsible for paginating and converting image data to ITU Rec T.4 or Rec T.6 format before transferring it to the FAXCARD. The software that was developed had to be of a higher level than that of class 2. The following capabilities were therefore added to enable the FAXMODEM to handle facsimile transmit and receive sessions entirely independent from the PC:

- On board facsimile transmission scheduling. The PC therefore does not have to initiate sessions. (See 6.13 **Transmit scheduling**, page 59)
- FAXMODEM is capable of completing an entire facsimile receive session without interrupting the PC. Complete facsimile documents can therefore be received and stored before transferring it to the PC's hard disk.
- FAXMODEM can convert ASCII text and TIFF images to Rec T.4 compressed data when transmitting.
- FAXMODEM can convert from Rec T.4 compressed data to TIFF images when receiving.



Computer A
Application Program
FAXCARD interface software
Class 1 or Class 2
FAXCARD
Rec T.30, T.4, T.6, V.27ter, V.29, V.21
Computer B
Application Program
FAXCARD interface software
Class 1 or Class 2
FAXCARD
Rec T.30, T.4, T.6, V.27ter, V.29, V.21
Public-switched telephone network
Rec T.30, T.4, T.6, V.27ter, V.29, V.21
Remote Fax machine

Figure 4-1 Facsimile communication

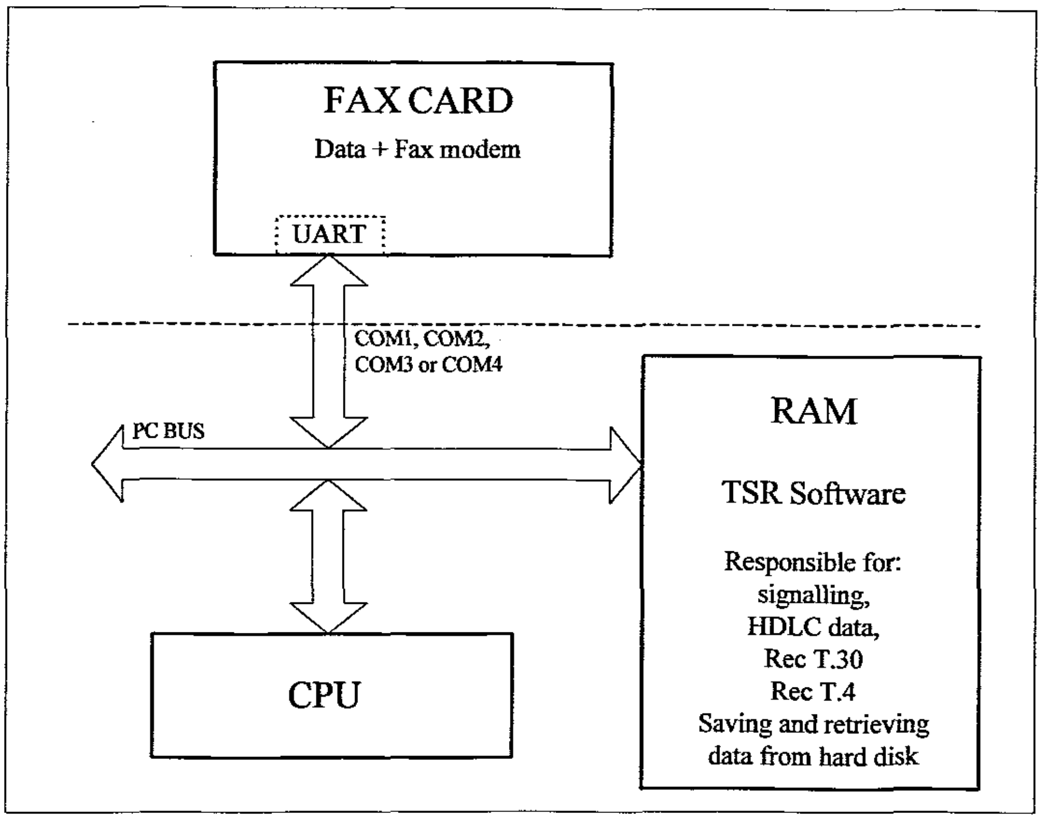


Figure 4-2 Hardware interface of a typical FAXCARD

4.2 Existing FAXCARD and PC interface

Existing processor-bus-attached "fax boards" communicate with the PC by emulating a COM port (ITU Rec V.24 serial connection). Both facsimile and data modem commands and data are transferred via this port. The hardware configuration of a typical PC and FAXCARD interface is illustrated in Figure 4-2.

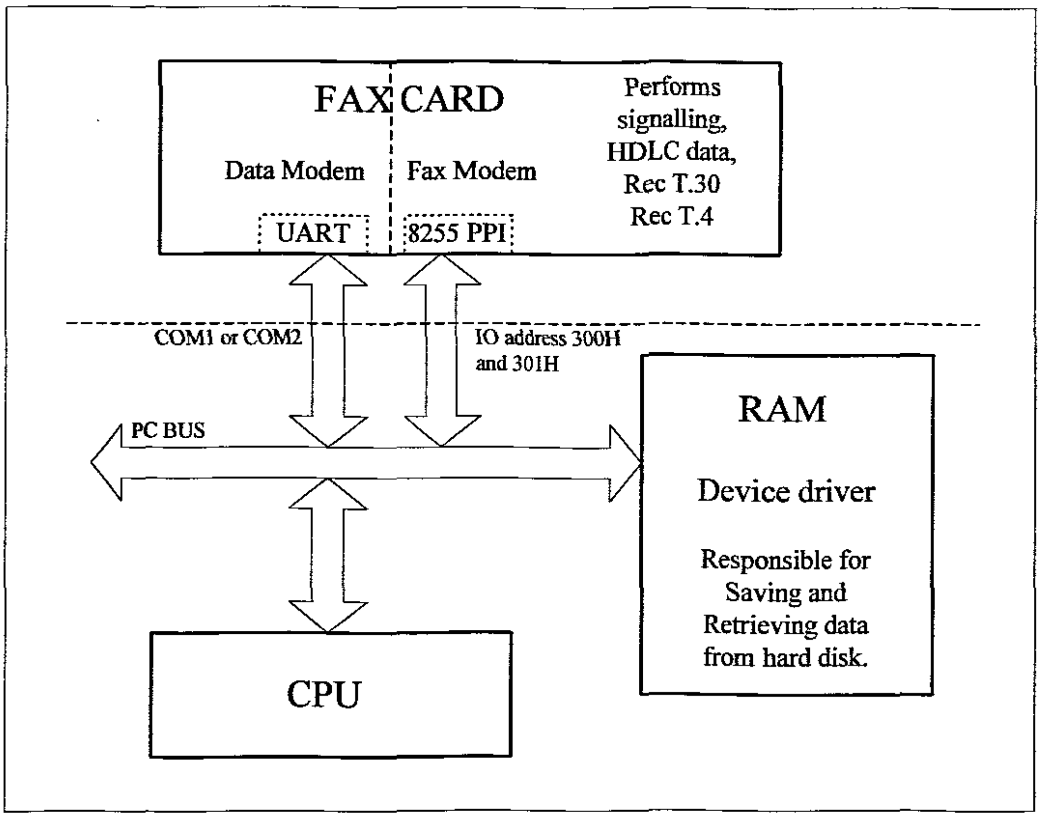

FAX CARD
Data Modem
Fax Modem
UART
8255 PPI
Performs signalling, HDLC data, Rec T.30 Rec T.4
COM1 or COM2
IO address 300H and 301H
PC BUS
RAM
Device driver
Responsible for Saving and Retrieving data from hard disk.
CPU

Figure 4-3 Hardware interface of designed FAXMODEM card

4.3 Designed FAXCARD and PC interface

The designed FAXCARD uses two communication channels to interchange information with the PC. The PC communicates with the DATA MODEM via an emulated COM port. The user can select between COM1 and COM2 by means of a jumper on the FAXCARD. The DATA MODEM uses IRQ4 (COM1) or IRQ3 (COM2) to interrupt the PC's CPU.

The FAXMODEM is accessed via two hardware ports viz. IO port 0300H (DATA) and 0301H (STATUS). An interrupt request line viz. IRQ7 is used for interrupting the PC.

Existing DATA MODEM software could therefore still be used to drive the DATA MODEM on the designed FAXCARD. Software however had to be developed to support the FAXMODEM component of the FAXCARD.

4.4 Image data

Digital image data had to be exchanged between application programs and the FAXCARD and an image data format was needed that supported both the FAXCARD and applications in the PC environment.

4.4.1 FAXMODEM Image data transfer format: TIFF

Tagged Image File Format (TIFF) files make use of tags to store the properties that describe the image, e.g. size and resolution. Programs that generate TIFF files are not constrained to use a constant file structure as tags can be located anywhere in the file. The TIFF reader only requires the necessary algorithm to locate the tags.

The TIFF format supports ITU Rec T.4 and Rec T.6 image data formats and are supported by both DOS and WINDOWS applications in the PC environment. It is also independent of operating systems, filing systems, compilers and processors.

This format was therefore selected as the default format for the storage of all received facsimile documents.

CHAPTER 5

FAXCARD Hardware

The hardware on the FAXCARD can be divided into three major components:

- Data Access Arrangement (DAA)
- DATA MODEM
- FAXMODEM

Figure 5-1 illustrates the I/O ports and interrupt lines that are utilized by the developed software to control the FAXMODEM card.

5.1 DATA ACCESS ARRANGEMENT

The data access arrangement (DAA) is the interface between the two modems and the telephone line. The DATA MODEM and FAXMODEM share one DAA. The DAA contains the following lines:

1. *FSEL.* This line is controlled by the FAXMODEM and it instructs the DAA whether it should connect the speech path to the FAXMODEM or the DATA MODEM.
2. *OH.* This line is controlled by both FAXMODEM and DATA MODEM and operates the OH (off hook) relay inside the DAA. This line is therefore used to answer a call, to seize the line for an outgoing call and to create pulses when pulse dialing.
3. *AR.* The FAXMODEM uses this line to connect the telephone to the exchange line.
4. *RING.* This output line of the DAA is connected to INT2 of the FAXMODEM CPU. This line is activated by the presence of ringing current on the exchange line.

5.2 DATA MODEM

The DATA MODEM includes a modem chip set and UART. The chip set contains a built in ROM with firmware to operate the DATA MODEM. The FAXMODEM is in control of the adaptive answering of incoming calls. The following lines provide the necessary control to the FAXMODEM over the DATA MODEM when an incoming call is to be answered:

1. *RI.* This is an input signal to the DATA MODEM and is normally connected directly to the RING output of the DAA. In this application the FAXMODEM will strobe this line when it detects a RING interrupt on INT2.

2. *AR.* This is an output line of the DATA MODEM and is normally used to connect the telephone to the exchange line. In this application it is used to inform the FAXMODEM that the DATA MODEM did not detect a carrier signal. This would result in the FAXMODEM taking over the answering procedure.

Adaptive answering is described in section 6.4, page 28.

5.3 FAXMODEM

The main building blocks of the FAXMODEM are:

5.3.1 The 80C188 CPU

The 80C188 is a very high integration 16-bit microprocessor with an external 8-bit data bus. The integrated features provided by this microprocessor are utilized as follows:

- *Programmable Interrupt Controller.* The integrated interrupt controller is programmed to operate in *fully nested mode* enabling INT0, INT1, INT2 and INT3 pins to be used as edge-triggered interrupt requests inputs.

- *Programmable Memory and Peripheral Chip Select Logic.* This integrated logic is programmed to provide the following chip select signals:

  - *Upper Chip Select (UCS) signal.* This signal selects the EPROM and is activated when memory address range E0000h to FFFFFh is accessed.

- *Lower Chip Select (LCS) signal.* This signal selects the first RAM chip and is activated when memory address range 00000h to 1FFFFh is accessed.
- *Peripheral Chip Select 0 (PCS0) signal.* This signal selects the R96EFX DSP chip and is activated when I/O address range 400h to 47Fh is accessed.
- *Peripheral Chip Select 1 (PCS1) signal.* This signal selects the 8255 PPI chip and is activated when I/O address range 480h to 4FFh is accessed.

• *Three (3) Programmable 16-bit Timers (Timer0, Timer1 and Timer2).* Timer 2 is programmed to run continuously and to generate an interrupt every one millisecond. Timer 0 and Timer 1 are programmed when required by the FAXMODEM software to perform timing of certain operations and to generate time delays.

5.3.2 The 8255 Parallel Port Interface

The FAXMODEM selects the 8255 PPI with the PCS1 peripheral chip select signal. The I/O base address of the PPI is therefore 0480h. The 8255 PPI chip is configured as follows:

- **Port A** is configured as a bi-directional port to the PC.
- **Port B** is configured as an output port controlling the following lines:
  - *PB0.* This is connected to the power on reset (POR) pin of the R96EFX DSP chip and is used to reset the DSP.
  - *PB1* enables or disables the tri-state buffer on the interrupt request line (IRQ7) to the PC.
  - *PB2* controls the off hook relay inside the DAA.
  - *PB3* connects or disconnect the telephone to the exchange line.
  - *PB4* selects between the speech paths of the FAXMODEM and DATA MODEM.
  - *PB5* sends ring pulses to the DATA MODEM.
  - *PB6* and *PB7* are used to select the cable equalizer inside the DSP chip.

- **Port C.** Five of port C's pins (PC3 to PC7) provide handshaking lines for the bi-directional buffer (Port A). The remaining lines (PC0 to PC2) are configured as input lines. Port C is therefore utilized as follows:
  - *PC0* is connected to the AR output of the DATA MODEM and signals to the FAXMODEM whether the DATA MODEM has rejected or accepted an incoming call.
  - *PC1* is connected to the INH_OBF signal of the PC IO Control circuit. This signal is used to inform the FAXMODEM of a full PC input buffer and is a request to the FAXMODEM to suspend the flow of data to the PC.
  - *PC2* is connected to the ENA_IBF signal of the PC IO Control Circuit. This signal informs the FAXMODEM when the PC's output buffer becomes empty.
  - *PC3* is connected to INT1 and is used as an interrupt request line to the FAXMODEM CPU.
  - *PC4* strobes data on the PC IO bus (DB0-DB7) into the bi-directional buffer (Port A).
  - *PC5* indicates to the PC IO Control whether the FAXMODEM has read data from Port A.
  - *PC6* transfers Port A data onto the PC IO bus, i.e. PC reads Port A.
  - *PC7* indicates to the PC IO Control whether the FAXMODEM has written to Port A.

5.3.3 The R96EFX Digital Signal Processor

The FAXMODEM CPU controls the DSP by writing data to one or more of its 32 registers. The DSP is selected by PCS0 peripheral chip select signal of the CPU and its I/O base address is therefore 0400h. The R96EFX supports the following features:

- Transmission and reception of ITU Rec V.27*ter*, V.29, V.21 signals. These signals are used for image data and handshaking information interchange.

- High-level data link control (HDLC) framing at 9600, 7200, 4800, 2400 and 300 bps. HDLC frames are used to exchange identification and capability information.
- Programmable dual / single tone generation. This is required for initial identification signaling and DTMF dialing.
- Adaptive and compromise cable equalization.
- Voice mode transmission and reception. This feature was not utilized.

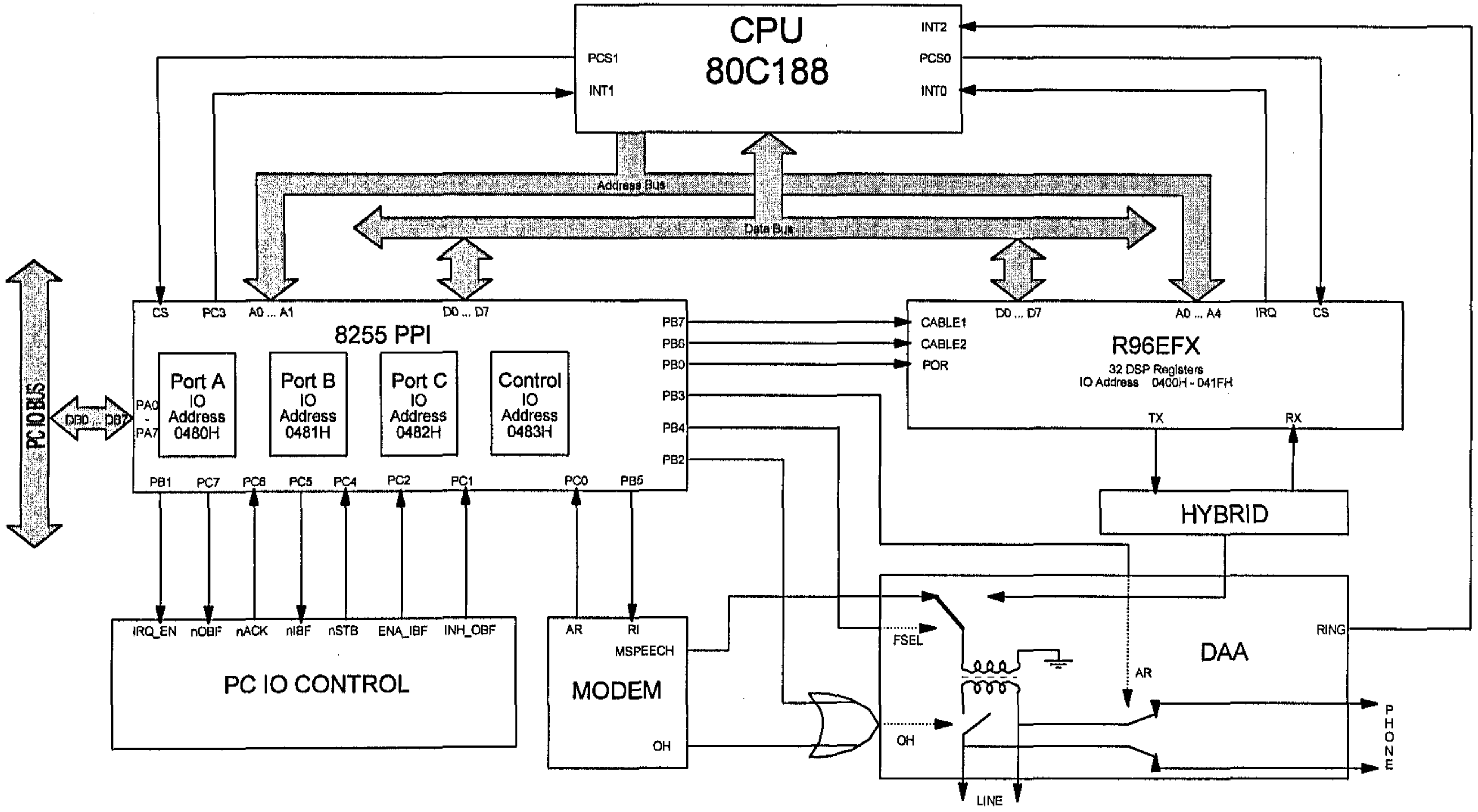


Figure 5-1 FAXMODEM IO

CHAPTER 6

FAXMODEM Software

6.1 FAXMODEM Software architecture

The software on the FAXMODEM card is divided into two principal areas viz. the foreground processes and the operating system functions.

6.1.1 Foreground processes

This includes the following principal tasks:

1. processing image data (ITU Rec T.4). This includes conversion and pagination of image data.
2. scheduling and initializing transmit sessions
3. initializing receive sessions
4. sending and receiving of facsimile documents (ITU Rec T.30)

Figure 6-1 depicts the architecture of the software responsible for the foreground processes of the FAXMODEM.

6.1.2 FAXMODEM operating system functions

This is functions and procedures that is utilized by the foreground processes to perform I/O related tasks. The operating system functions are divided into two main categories:

1) Background processes, i.e. processes which are invoked by means of hardware interrupts. This includes the following processes:

   - transfer of data to or from PC
   - detection of ringing

- maintenance of system timers
- receiving and transmitting of HDLC frames
- receiving and transmitting of facsimile image data

Functions or processes that are called directly by the foreground processes. This includes the following processes:

- memory management functions

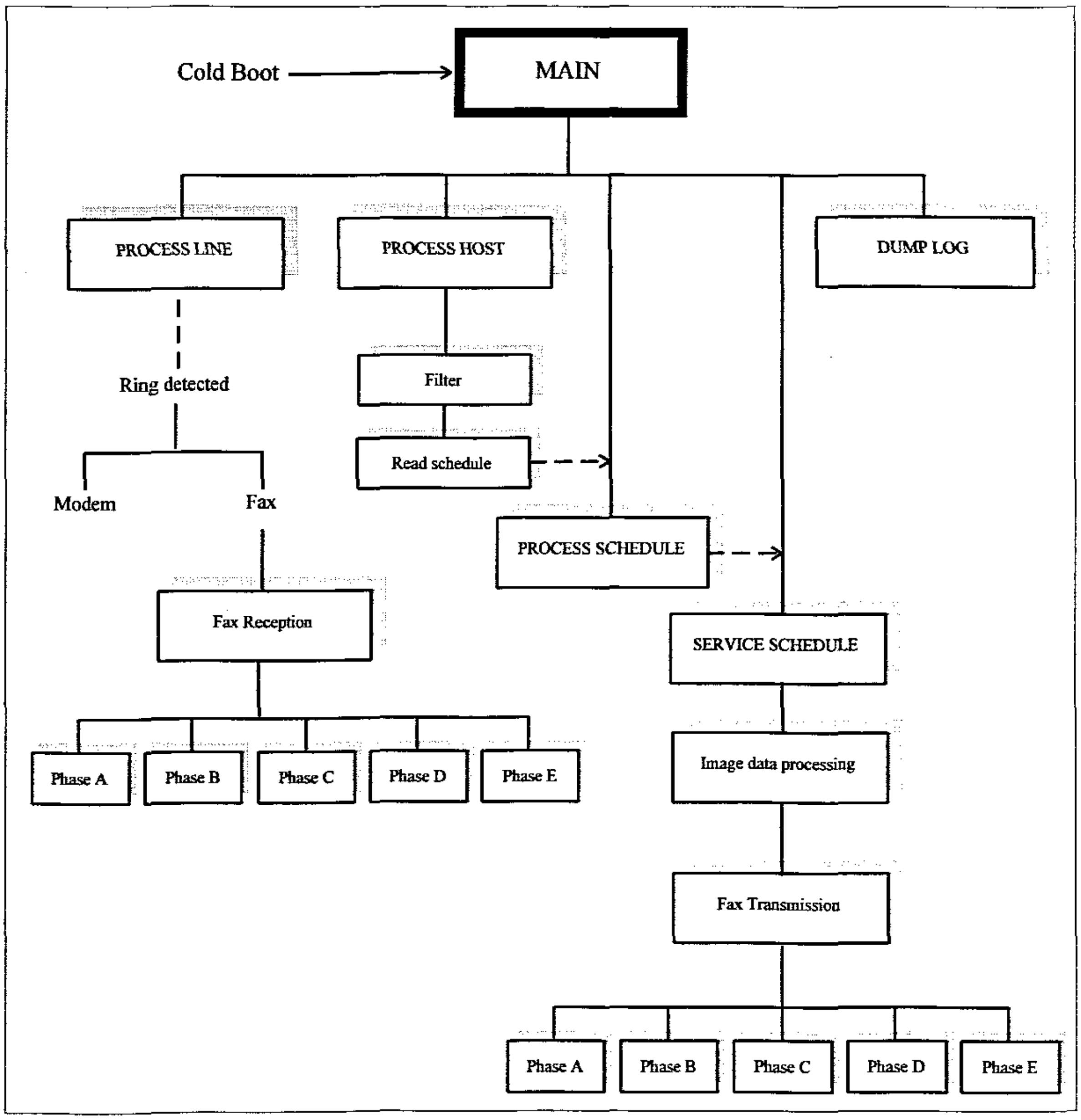


Figure 6-1 FAXMODEM foreground processes

- setting up of timers, calculation and formatting of dates
- file handling. This includes:
  - ⇒ file creation and deletion
  - ⇒ reading and writing of files

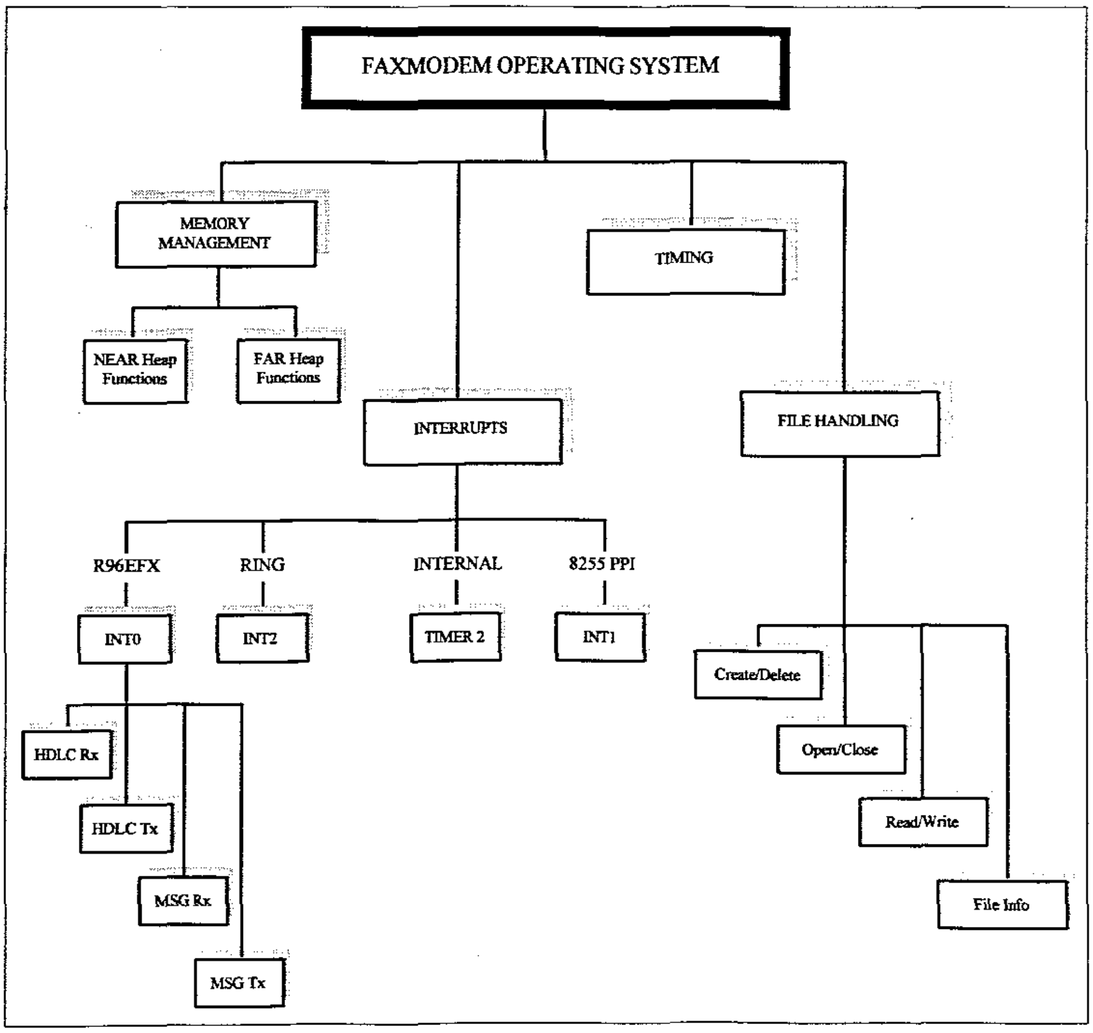


Figure 6-2 Operating system functions and processes

Figure 6-2 depicts the architecture of the operating system's functions and procedures.

The remainder of this chapter describes briefly some of the most important procedures and processes executed by the FAXMODEM software.

6.2 FAXMODEM Initialization

On power up or after a hard reset, the *COLD BOOT* procedure initializes and tests hardware to bring the FAXMODEM into a usable state by the operating system. The sequence of events following a hardware or power on reset is as follows:

1) Interrupts are disabled and all the CPU's general registers are tested.

2) The control registers of the CPU are initialize as follows:

   - The RELOCATION register is setup so that the register control block is in IO space with its base address equal to 0FF00h.
   - The LMCS (low memory chip select) register is setup so that the LCS(lower chip select) signal will address memory from 00000H to 1FFFFh (See section 8.2, page 89).
   - The UMCS (upper memory chip select) register is initialized so that the UCS (upper chip select) signal will address memory from E0000H to FFFFFh.
   - The MPCS and PACS registers are setup so that peripherals is mapped in IO space with base address equal to 400H

3) The 8255 PPI chip is initialized to operate as follows:

   - Port A as a bi-directional parallel port with the upper half of Port C (i.e. PC4 to PC7) acting as control lines.
   - Port B as an output port.
   - The lower half of Port C (i.e. PC0 to PC3) as an input port.

4) The interrupt request line to the PC (i.e. IRQ7) is disabled.

5) A RAM test is done. Memory integrity and read write capability is tested from address 00000H to DFFFFH to determine the amount of usable RAM.

6) The stack is initialized to reside immediately above the NEAR HEAP in lower memory.

7) The interrupt vector table is copied from EPROM to memory address 00000H.

8) INT0, INT1 and INT2 are initialized with INT0 having the highest and INT2 the lowest priority.

9) Control is passed to the MAIN procedure.

6.3 Performing foreground processes

6.3.1 Setting of system variables

1) Input and output buffers are initialized.

2) The 1 millisecond system timer (Timer 2) is started.

3) The NEAR and FAR heaps are created.

4) The following variables are set to their default values:

- *Number_Of_Rings* is set to 2.
- *Wait_For_Carrier* is set to 15000 milliseconds.

6.3.2 The operating system *main* loop

1) The R96EFX DSP chip is reset. This is done at the start of the loop to ensure that the state of R96EFX chip at this point will always be the same.

2) The MODEM is connected to the DAA and the telephone to the line.

3) The cable equalizer within the R96EFX chip is set.

4) The next five procedures which are called in sequence, are the main execution blocks of the FAXMODEM card.

   - ***PROCESS_LINE***: Checks for incoming calls. If an incoming FAX call is detected it will call the *Fax Reception* procedure.
   - ***PROCESS_HOST***: Checks for any incoming commands or data from the host PC. It will store all schedules, lists, and fax images as files on the FAR HEAP and set a FLAG to indicate the arrival of such data.

- ***PROCESS_SCHEDULE*:** Reads a schedule file from the FAR HEAP and converts all ASCII data lines to binary SCHEDULE_ITEMS. The SCHEDULE_ITEMS are stored on the NEAR HEAP.
- ***SERVICE_SCHEDULE*:** This procedure checks the send-time of each SCHEDULE_ITEM on the NEAR HEAP. If a SCHEDULE_ITEM is due, the required fax image files are read from the PC and the TRANSMIT_FAX procedure is called. If TRANSMIT_FAX procedure failed, it is converted to a SCHEDULE_JOB and stored on the NEAR HEAP to be retried later.
- ***DUMP_LOG*:** This procedure updates the log file on the PC's hard disk .

5) After the LOG dump control is passed back to the beginning of the operating system's main loop.

6.4 Adaptive answering of incoming calls

The *PROCESS_LINE* procedure is responsible for determining whether an incoming call is destined for the DATA MODEM or the FAXMODEM. Refer to Figure 6-3. This procedure is skipped if the RING_COUNT is below the Number_Of_Rings required before answering.

- The CPU will wait 1.5 seconds to give the DATA MODEM enough time to respond to the incoming call.
- TIMER0 is started with its time-out set to the value of the Wait_For_Carrier variable + 0.5 seconds.
- If the DATA MODEM has not responded yet the AR_FLAG is reset and the FAX RECEPTION process is started immediately.
- If the DATA MODEM did respond the CPU will set the AR_FLAG and wait until TIMER 0 has timed out or the DATA MODEM has rejected the call.
- If the DATA MODEM has rejected the call, the FAX RECEPTION process will be started.
- If TIMER 0 has timed out it is assumed that the DATA MODEM has connected. The CPU will now continue to poll the DATA MODEM until it detects DATA MODEM off-line.
- Control is returned to the main procedure after the termination of a DATA MODEM or FAXMODEM call.

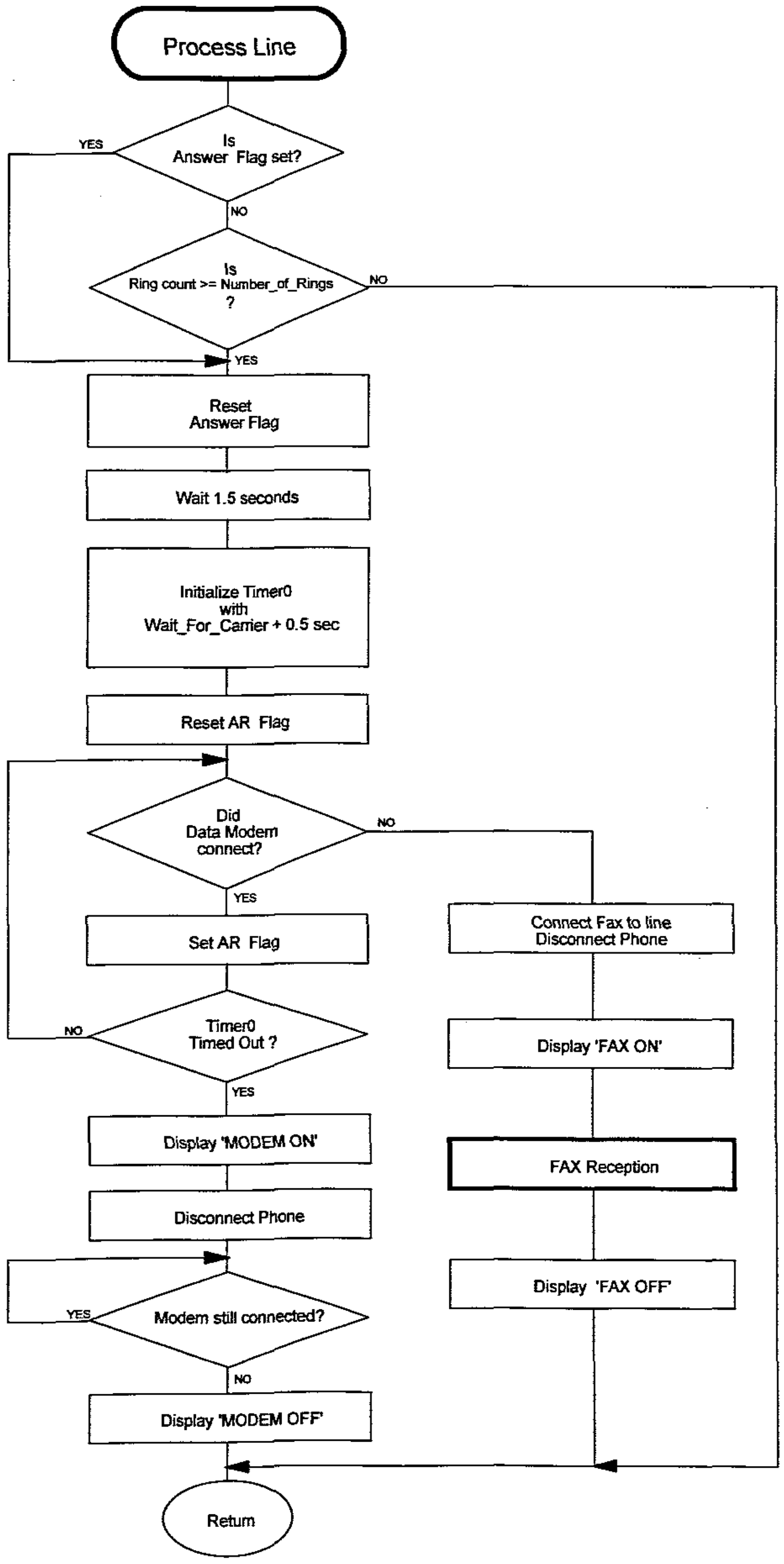


Figure 6-3 The PROCESS_LINE procedure

6.5 Fax reception

The *PROCESS_LINE* procedure will call this procedure if it detects an incoming none DATA MODEM call. This procedure will then determine if the incoming call is a fax call. If not, the telephone will be connected to the line and the operator will be informed that a voice call was detected.

6.5.1 Phase A

- The CPU connects the FAX to line and waits 0.5 seconds before configuring the R96EFX chip for FSK operation.
- The AR_FLAG is tested. This indicates whether the MODEM has already transmitted an answer tone or not. If this flag is not set, a tone of 2100 Hz is transmitted for 3.5 seconds.
- A new fax file is created to store the incoming fax document and the current time is recorded for logging purposes.

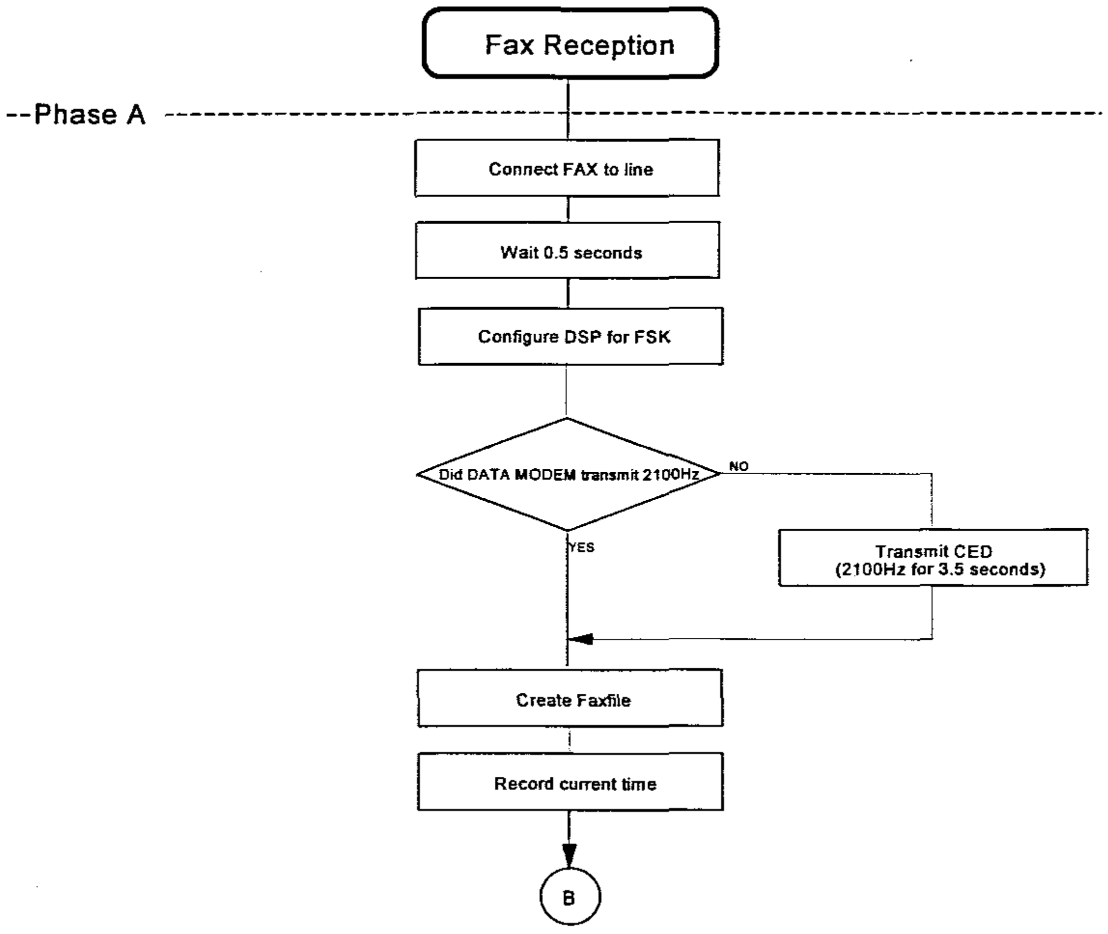


Figure 6-4 Fax reception - Initialization

6.5.2 Phase B: pre-message procedures

- After sending a digital identification signal the FAXMODEM will alert the operator if a response from the calling station is not received within 0.5 seconds. The FAXMODEM will continue to send DIS's until a response is received, a 35 second timer has timed out or the operator decides to answer the call.
- If the FAXMODEM does receive a response from the calling station, the local mode of operation is set up from the information received.
- The quality of the line is tested next. If the quality of the line is poor, the baud rate is adjusted downwards until an acceptable quality of reception is obtained.
- The FAXMODEM is now ready to receive the fax message.

6.5.3 Phase C: message reception

A single page of image data is received in phase C. This data is transferred to the file created in Phase A.

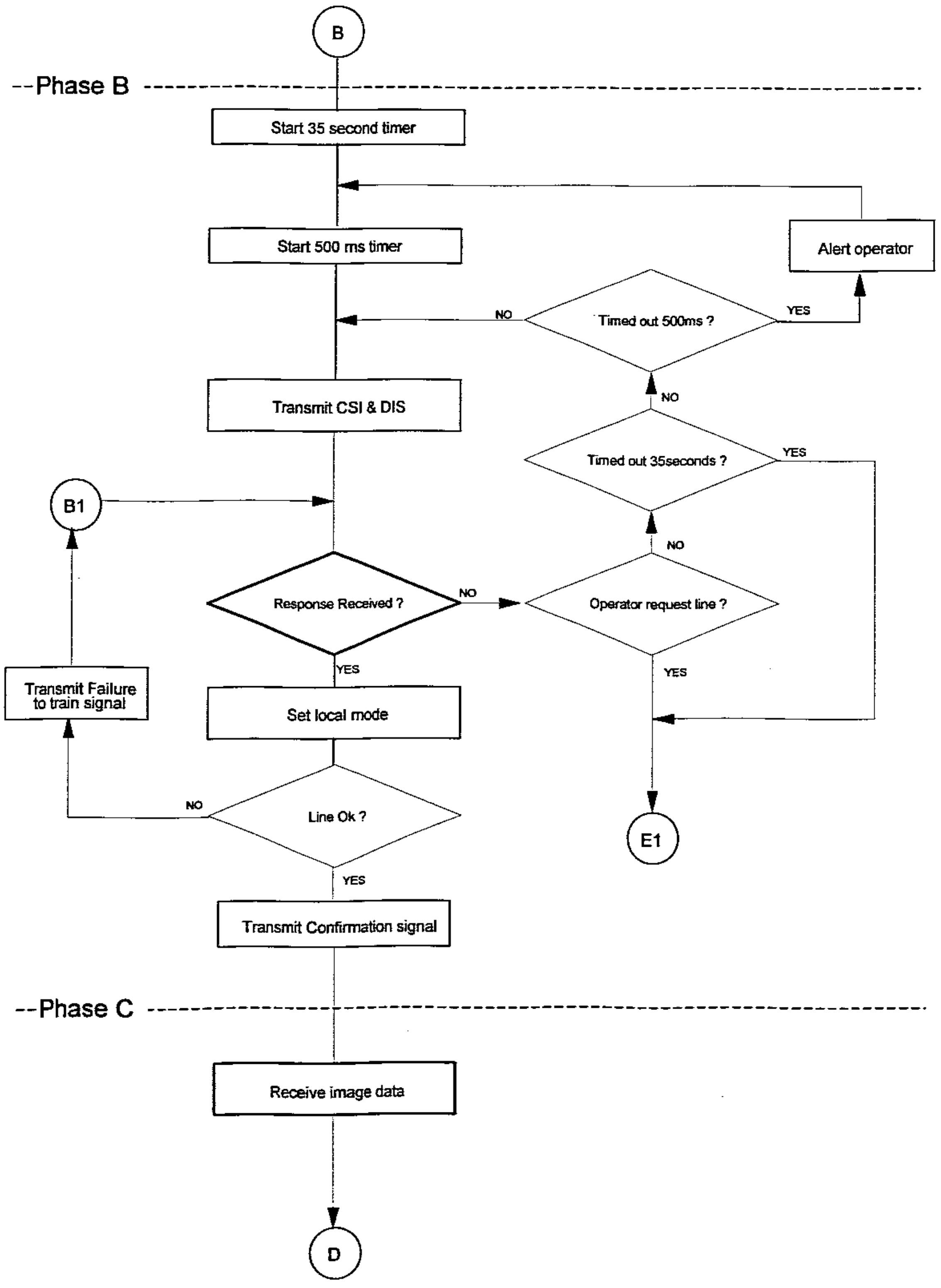


Figure 6-5 Fax reception - Handshaking and message reception

6.5.4 Phase D: post-message procedures

The quality of the image received is checked as follows:

> The run-lengths of a scan line are added up to determine the scan line's total length. If the total length of the scan line is not equal to the recording width of the facsimile document, the scan line is regarded as corrupted. The amount of 'good' and 'bad' scan lines are counted. The quality of the page received is therefore calculated by comparing the amount of corrupted scan lines against the total amount of scan lines received for that page.

If the copy quality of the page is unacceptable the FAXMODEM will request a re-transmission of the page, otherwise it will request the next page.

6.5.5 Phase E

- A disconnect signal is sent to the remote fax machine and the call is terminated.
- All fax documents are transferred to the PC's hard disk immediately after reception.

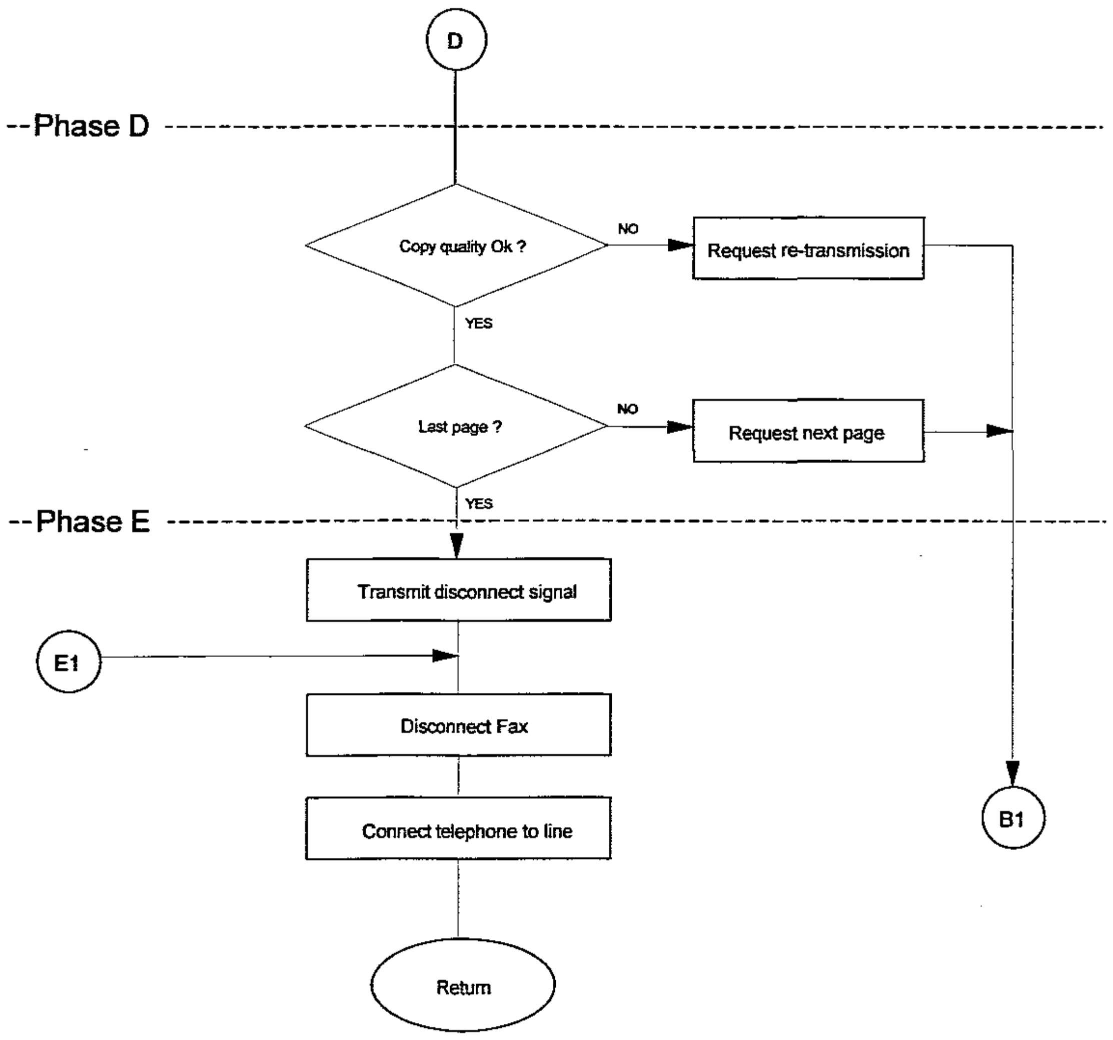


Figure 6-6 Fax reception -Post message procedures and call release

6.6 *Fax transmission*

6.6.1 Phase A: Call setup

- After detecting dial tone the remote fax machine's number is dialed in DTMF or PULSE dial mode.
- A 1100 Hz 0.5 second tone (CNG) is send every 3 seconds until an answer signal (CED) is received or a 30 second timer has timed out.

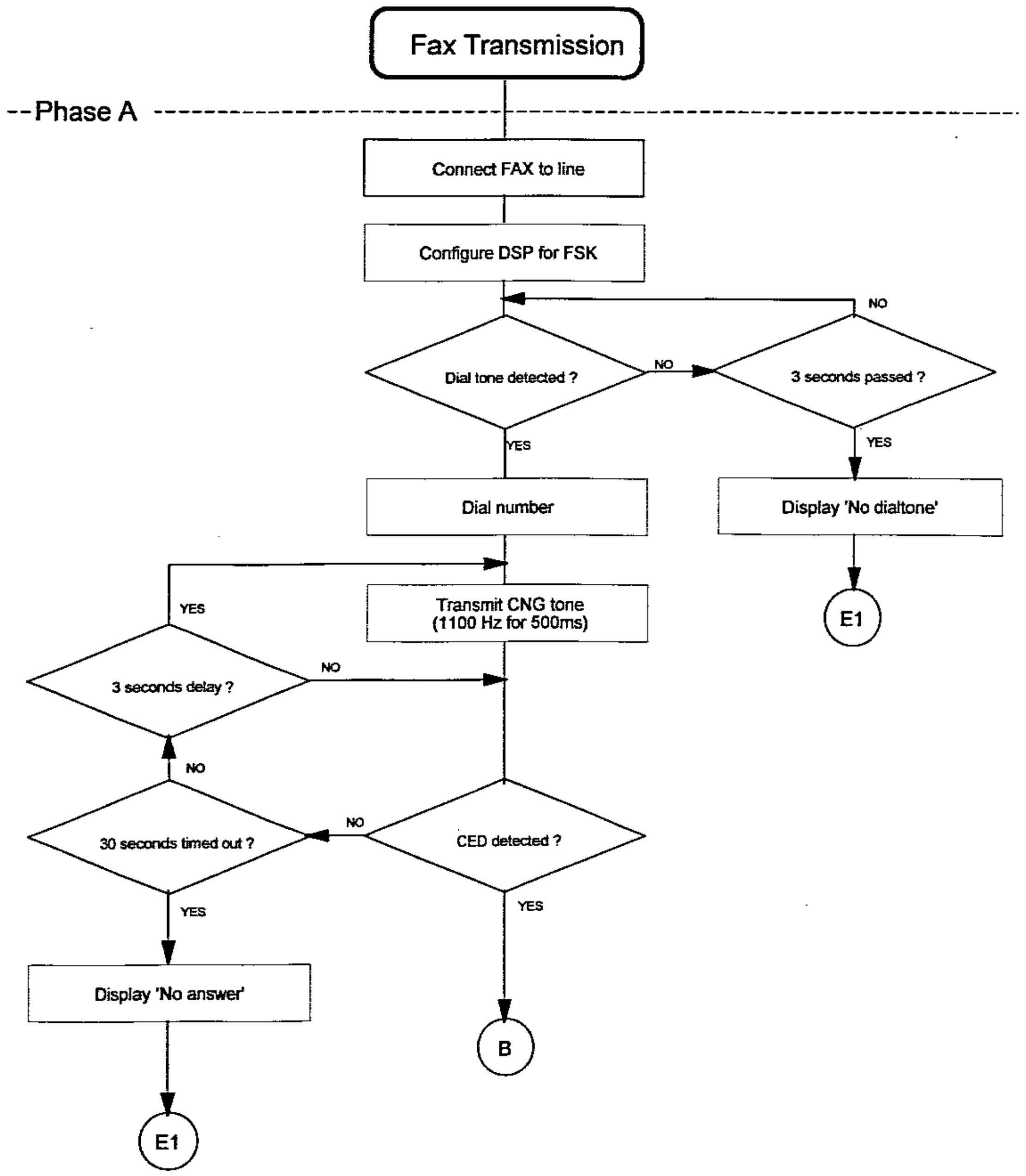


Figure 6-7 Fax transmission - Initialization

6.6.2 Phase B: pre-message procedures

- Phase B is entered upon receiving a CED signal from the called station.
- The FAXMODEM will wait up to 35 seconds for a DIS signal from the called station before aborting this procedure.
- From the information received in the DIS signal the local mode of operation is setup.
- The local mode and the dimensions and other parameters of the fax document to be transmitted are used to construct a DCS frame.
- The DCS frame is now transmitted followed by a training check (TCF) signal. The TCF is a series of zeros send at message transmission speed for 1.5 seconds. The integrity of the TCF signal received by the remote machine will determine if the speed of operation is acceptable for this line.

6.6.3 Phase C: message transmission

One page of fax image data is transmitted in phase C. A RTC signal is send to indicate the end of the page.

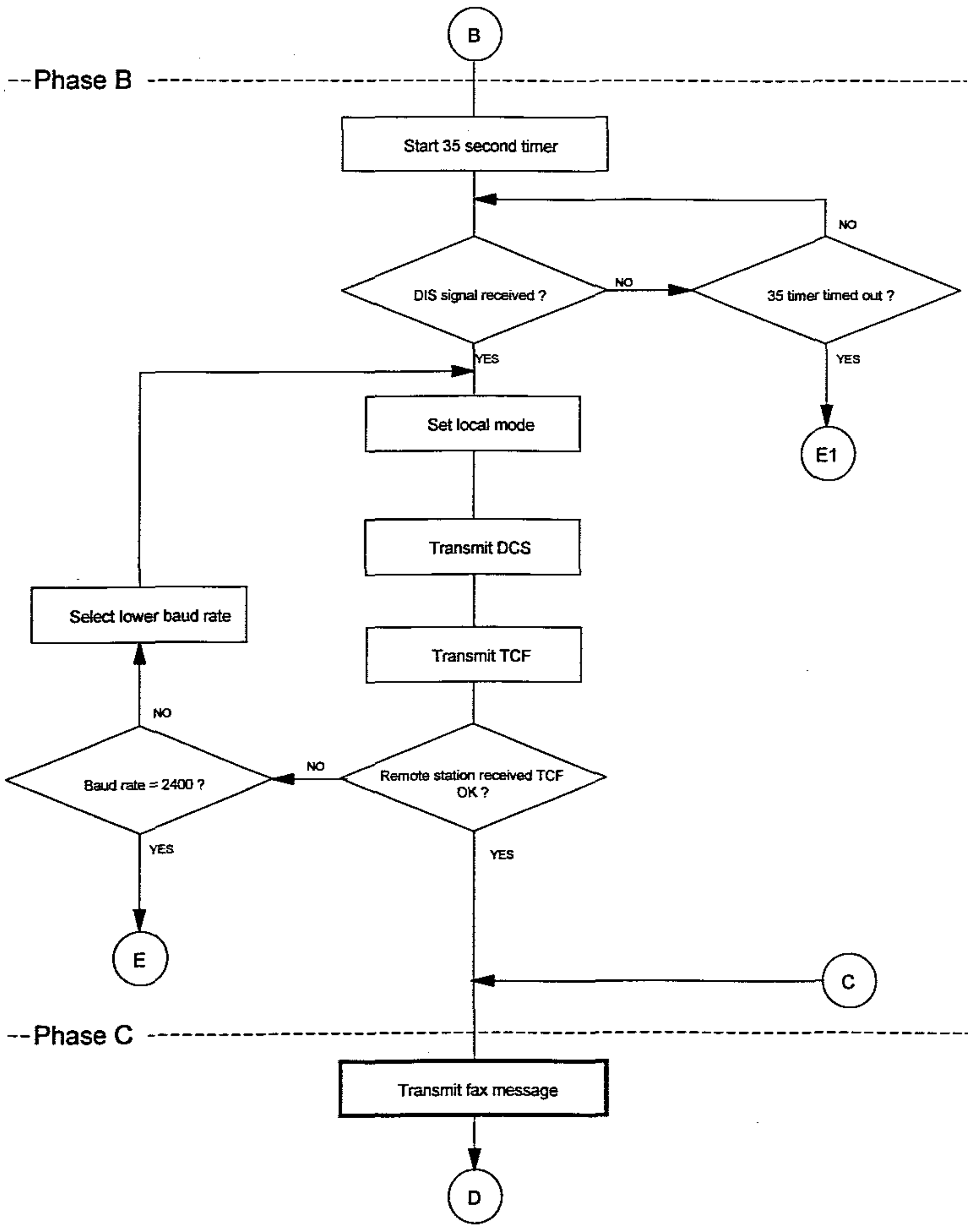


Figure 6-8 Fax transmission - Handshaking and message transmission

6.6.4 Phase D: post-message procedures

- The FAXMODEM will now wait for a response from the called station.
- If the quality of the image data received by the called station is unacceptable it will request the local station to re-transmit the same data.
- If the quality is acceptable the next page is transmitted. If there are no more pages available control is passed to phase E.

6.6.5 Phase E

The FAXMODEM sends a disconnect (DCN) signal to the remote station and terminates the call.

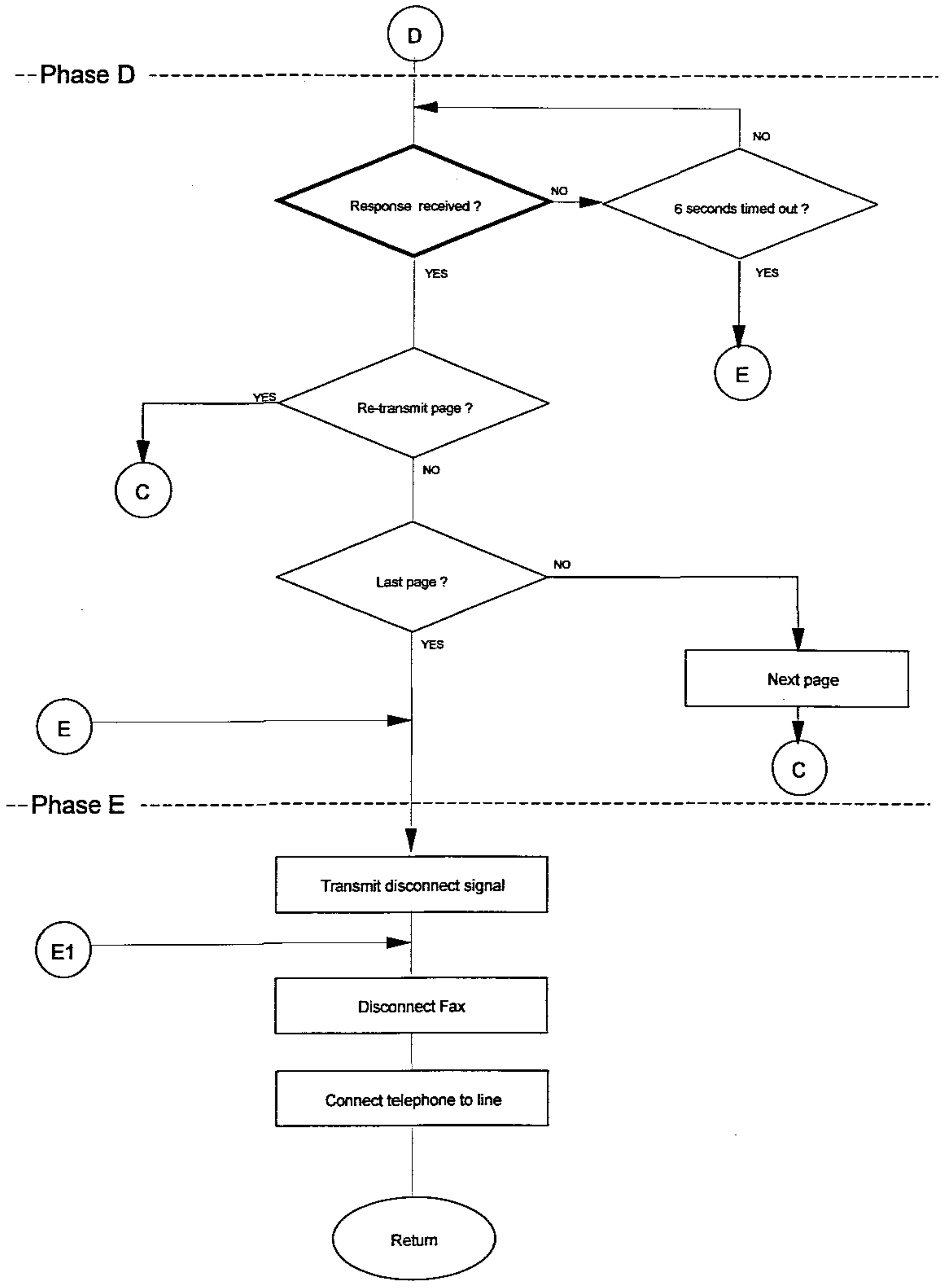


Figure 6-9 Fax transmission - Post message procedures and call release

6.7 Handling of Fax files

All fax documents are stored in the FAXMODEM RAM as files (fax files).

6.7.1 The Structure of a fax file

A fax file is structured as a linked list of fax pages. Each fax page is stored as a linked list of MEMBLOCKS. A MEMBLOCK is created on the FAR HEAP and is a fixed length of contiguous memory. A fax file should have at least one fax page and a fax page at least one MEMBLOCK. The minimum size of a fax page is therefore equal to the size of one MEMBLOCK.

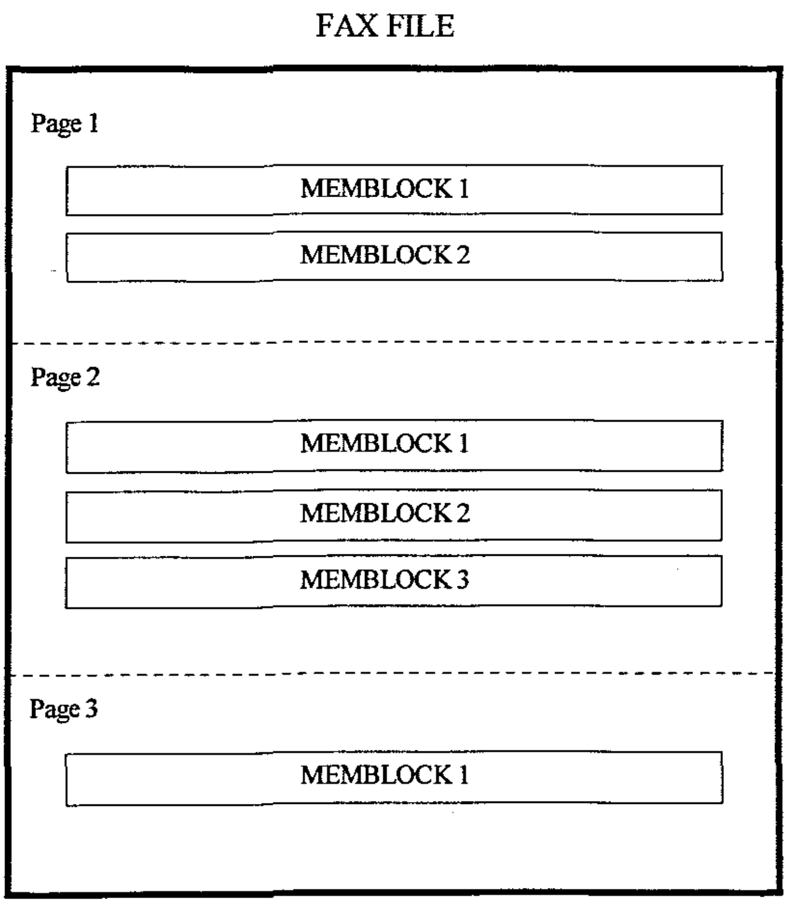


Figure 6-10 Structure of a fax file

6.7.2 Creating a Fax file

The recording width, vertical resolution, coding method and the name of the file are used when a new fax file is created. This information is written into the fax file header by the operating system of the FAXMODEM. (See Appendix B.)

6.7.3 Opening a Fax file

A file can be opened in three different modes, viz. reading, writing and append mode. Two parameters namely filename and open mode, are used to open a file.

6.7.4 Creating a Fax page

A fax file is always created with one empty page attached to it. All subsequent pages are created by performing the CREATEPAGE procedure. Four parameters are used to create a page, viz. the HANDLE, the vertical resolution of the page, the coding method used for this page and the recording width of the page.

6.7.5 Opening a Fax page

A fax page needs to be open before it can be updated or read from. The page number and HANDLE are used to open a fax page.

6.7.6 Writing to a Fax page

Data can be written to a fax file in three different modes viz. character, block and pointer mode. All three modes require a HANDLE. The validity of the HANDLE and the open mode of the file is checked before any write is performed.

6.7.6.1 Character mode

This mode requires the least amount of overhead processing, e.g. incrementing of pointers and overflow checking, and is advantageous when writing serial data to a fax file. Only two parameters viz. the data character and the HANDLE, are needed to perform character mode writing.

6.7.6.2 Block mode

This procedure writes a block of data to a fax file. It requires three parameters viz. the location of the block of data, the length of the block and the HANDLE. This method of writing is much faster than character mode when writing larger blocks (± 1000 bytes) of data.

6.7.6.3 Pointer mode

This is fastest method to write to a fax file. This procedure does not perform any writing but returns a write pointer that enables the writing of data directly to the fax file. Care must be taken not to overwrite critical data in memory. Two parameters viz. HANDLE and *data-length*, are passed to this procedure. The *data-length* is the amount of data that are to be transferred directly. This procedure returns the following values:

a) *The maximum amount of bytes that can be written* - This amount should not be exceeded when writing data directly. If this amount is smaller than the amount requested, this amount of data should be written first after which another request can be performed.

b) *The amount of bytes that cannot be written* - This is the amount of data that can not be written and is the difference between the amount of data requested and the maximum amount of data that can be written.

c) *The pointer to the write area* - This is a far pointer to the data area of the fax page.

6.7.7 Reading from a Fax page

Data can be retrieved from a fax page in three different modes viz. character, block and pointer mode. All three modes of reading check the validity of the HANDLE and the open mode of the fax file, before processing the read request.

6.7.7.1 Character mode

As with character mode writing, this method requires the least amount of housekeeping. This mode is used when reading serial data from the fax file. Only one parameter viz. the HANDLE is required to read a fax file in character mode.

6.7.7.2 Block mode

If a block of data is to be transferred to another area in memory, block mode is faster than character mode. This mode of reading requires three input parameters viz. the HANDLE, a pointer to a data transfer area and the number of bytes to read.

6.7.7.3 Pointer mode

Pointer mode is faster than block mode. This mode does not perform any data retrieval, but a pointer that points to the fax file data is returned. Data can therefore be read directly from the fax file. Two parameters viz. the HANDLE and the amount of bytes that needs to be accessed, are required. The following values are returned by this procedure:

a) *The size of the read area* - This is the amount of data that should be read before another pointer mode read is performed on this file. The read pointer in the FAXJOB is incremented by this amount.

b) *The number of bytes that cannot be accessed* - This is the amount of data that is inaccessible. Another *pointer-mode* read is required to access this data.

c) *The pointer to the read area* - This is a far pointer to the fax file data.

6.7.8 Closing a Fax page

After writing to or reading from a fax page is completed, the fax page needs to be closed. A HANDLE is required to close a fax page. An additional parameter viz. the page line count is required if closing a fax page after writing.

6.7.9 Closing a Fax file

A fax file is closed to release the HANDLE and memory occupied by FAXJOB, to the operating system.

6.7.10 Clearing a Fax page

Data within a fax page needs to be removed if it was received corrupted from a remote fax machine. A HANDLE and a page number are required to clear a fax page.

6.7.11 Deleting a Fax file

This procedure deletes a fax file from memory. Only inactive fax files, i.e. fax files that are not open, can be deleted. The name of the fax file is required as an input parameter.

6.7.12 Searching for a Fax file

This procedure requires a fax file name as an input parameter and it does a case-sensitive search in the fax file directory for a matching file name.

6.7.13 Reading Fax file information

6.7.13.1 File information

All file information pertaining the entire file is kept in the fax file header. This procedure reads the fax file header and returns the following information:

- The total amount of pages including empty pages.
- The maximum line count, i.e. the line count of the longest page in the fax file.
- The maximum recording width, i.e. the recording width of the widest pages.
- A pointer to the fax file header. See Appendix B.
- The maximum vertical resolution, i.e. the highest resolution used in this fax file.
- The maximum coding method. This indicates whether one or more pages were encoded with the optional two-dimensional coding method.
- The read and write status of this fax file. This indicates whether this file is open for reading and/or writing.

6.7.13.2 Page information

Information regarding a fax file page is stored in the fax pageInfo structure. This procedure reads this structure and returns the following data:

- The page line count. This is an indication of the physical length of the page.
- The page recording width in pixels. This equivalent to the scan line length of the page.

- The vertical resolution.
- The coding method. This information determines how the image data of the page should be decoded.
- The read and write status of this page. This indicates whether this page is open for reading and/or writing.

A HANDLE and page number are required as input parameters.

6.7.13.3 Page size

This procedure returns the length of the fax page in bytes, i.e. the amount of bytes representing the total fax page image. It requires a HANDLE and a page number as input parameters.

6.8 Image data processing

Software was developed to enable the FAXMODEM to convert between TIFF and ITU Rec T.4 formats and to convert ASCII text data to Rec T.4 format.

6.8.1 Text to T.4 conversion

Text files are sent to the FAXCARD in ASCII format. The FAXMODEM software then converts the text to a low resolution or high resolution Rec T.4 fax page(s) before transmission. The following steps describe the conversion process.

- The font table is selected. This is a table containing the raster images (bit maps) of the ASCII characters. The operator can indicate the desired font by means of an option on the command line. Two fonts viz. a normal resolution and a high resolution font are stored on the FAXMODEM EPROM. The *normal resolution* font is the default font.
- Using the selected font table, ASCII characters are converted to bit maps and stored in a separate file.
- If a text line, after conversion, is overlapping the right edge of the guaranteed reproducible area of the fax page, the overlapping words are wrapped to the next line. Text lines are not truncated and are therefore not limited to a certain length. See Appendix A.

- If an ASCII text document, after conversion, is overlapping the bottom edge of the guaranteed reproducible area of the fax page, the overlapping lines are wrapped to a new page.
- After conversion the ASCII text file is deleted from memory.

Table 6-1 shows the interpretation of corresponding ASCII control characters by the FAXMODEM when converting a text file to Rec T.4 format.

Table 6-1 Control characters for text conversion

ASCII character	Description	Effect
09h	**Tab**	**This character moves the write cursor to the next tab stop position. Tab stops are located every multiple of 128 pixels, i.e. every 15.9 mm from the left margin.**
0Ah	**Line feed**	**This character terminates the current line. The character following 0Ah will start on a new line.**
0Ch	**Form feed**	**This character terminates the current page.**
0Dh	**Carriage return**	**Terminates the current line. The character following 0Dh will start on a new line**
0Dh followed by 0Ah	**New line**	**Terminates the current line. The character following this character pair will start on a new line**
1Ah	**End of file**	**This character terminates the text file. This is an optional character.**

Table 6-2 Fax page margins

Margin	Size (pixels)	Size (mm)
LEFT	74	9.2
RIGHT	75	9.3
TOP (normal resolution)	32	8.3
TOP (high resolution)	64	8.3
BOTTOM (normal resolution)	38	9.9
BOTTOM (high resolution)	76	9.9

Table 6-2 shows the margins inserted whenever a text file is converted to Rec T.4 format.

6.8.2 TIFF to T.4 conversion

The FAXMODEM card supports the reading of the following TIFF class B (bi-level) image compression methods.

Compression method 1. Images with no compression.

Compression method 2. Images compressed with ITU Group 3 One-dimensional Modified Huffman run length encoding.

Compression method 3. Images compressed with Facsimile - compatible ITU Group 3, exactly as specified in Rec T.4.

Compression method 32773. Images compressed with PackBits compression

No margins are inserted when TIFF image files are converted. If the TIFF image is narrower than the destination Rec T.4 image, it is horizontally centered on the page. TIFF images which overlap the destination dimensions of the Rec T.4 page is scaled down in both horizontal and vertical dimensions.

6.8.3 T.4 to TIFF conversion

A fax file received from a remote station is in a raw Rec T.4 format. Transmission errors that occurred during Phase C will also be present in this data. When converting from Rec T.4 format to TIFF, corrupted lines are not converted. The scan line preceding the corrupted line is written twice in order to prevent vertical shrinkage of the image. The resulting TIFF image therefore would have corrected instead of corrupted lines. The FAXMODEM supports the writing of only one TIFF compression method, i.e. TIFF class B compression method 3. This is the default format for all received fax files.

6.8.4 T.4 to QFX conversion

The operator can set an option that will instruct the FAXMODEM to convert all received fax images to the proprietary QFX multiple page format.

6.8.5 QFX to T.4 conversion

The QFX format is proprietary multiple page fax file format. The FAXMODEM software can interpret this format for facsimile transmission.

6.9 Interrupt service routines

Three of the four available external interrupts of the 80C188 CPU are used.

6.9.1 R96EFX DSP interrupt service routine - INT0

The R96EFX DSP is programmed to interrupt the CPU when:

- It reads its data buffer (DBUFF) during transmission. This prompts the CPU to write the next byte into the data buffer.
- It writes data into DBUFF during reception. This prompts the CPU to read the byte from DBUFF.
- It detects an abort or idle sequence. This occurs when a remote transmitting machine aborts or there is a break in transmission.

The interrupt service routine (ISR) that is responsible for servicing interrupts from the R96EFX DSP calls one of four procedures. The procedure to call is determined by the current process running in the foreground. One of the following procedures can be executed:

⇒ The HDLC receive ISR.

⇒ The HDLC transmit ISR.

⇒ The Message receive ISR.

⇒ The Message transmit ISR.

The HDLC receive and transmit interrupt service routines are used during Phase B and Phase D. The Message receive and transmit interrupt service routines are used during Phase C.

6.9.1.1 The HDLC receive ISR

This procedure reads data from DBUFF and writes it to the HDLC receive buffer of the FAXMODEM. If an abort/idle condition caused the interrupt the foreground process is informed.

6.9.1.2 The HDLC transmit ISR

. This procedure reads a data byte from the HDLC transmit buffer and writes it to DBUFF.

6.9.1.3 The Message receive ISR

This procedure performs the following functions:

- Reads a data byte from DBUFF and writes it to the message buffer.
- Decode received data to detect end of line (EOL) and return to control (RTC) bit sequences.
- Check the recording width of a received line to determine if data corruption due to line errors has occurred.
- Count the number of corrupted and uncorrupted lines received. This information is used in Phase D to determine the *copy quality* of the received document.
- Check the duration of lines received. If it takes more than 5 seconds to receive a scan line, this procedure will inform the foreground process to abort this message receive session.

- Check for a return to control (RTC) byte sequence. If detected, this procedure will inform the foreground process.

6.9.1.4 The Message transmit ISR

This procedure performs the following functions:

- Read compressed T.4 data from the *message buffer* and writes it to DBUFF.
- Decode the transmitted data. This is done to detect end of line (EOL) bit sequences.
- Check the duration of a transmitted scan line. If the duration is less than the *minimum scan line time* (as determined in Phase B), this procedure will transmit *fill* bits to meet the *minimum scan line time* requirement.
- Count the amount of lines transmitted.
- Check for a return to control (RTC) byte sequence. If detected, this procedure will inform the foreground process.

6.9.2 The 8255 PPI interrupt service routine - INT1

The 8255 PPI interrupts the CPU when:

- the PC reads data from the bi-directional buffer, Port A or
- the PC writes data to Port A.

To ensure that no corruption or data loss occur the flags described in Table 6-3 are used to control the flow of data between the PC and the FAXMODEM.

Table 6-3 PC input/output control flags

Flag	Description	Location	FAXMODEM can	Function
IBFa	**Input buffer full flag**	**Port C, bit 5**	**Read only**	**Indicates if Port A contains data destined for the FAXMODEM**
OBFa	**Output buffer full flag**	**Port C, bit 7**	**Read only**	**Indicates if Port A contains data destined for the PC**
INTE_IBFa	**Input buffer interrupt enable flag**	**Port C, bit 4**	**Read/ write**	**If set, this flag enables the 8255 PPI to generate an interrupt signal when the PC writes data to Port A**
INTE_OBFa	**Output buffer interrupt enable flag**	**Port C, bit 6**	**Read/write**	**If set, this flag enables the 8255 PPI to generate an interrupt signal when the PC reads data from Port C.**
ENA_IBF	**Enable input buffer flag**	**Port C, bit 2**	**Read only**	**This flag is set by the PC when its outbuf buffer, i.e. the buffer in PC RAM which holds data destined for the FAXMODEM, contains data.**
INH_OBF	**Inhibit output buffer flag**	**Port C, bit 1**	**Read only**	**This flag is set by the PC when its input buffer, i.e. the buffer in PC RAM which holds data received from the FAXMODEM, is full.**
INTRa	**PC IO interrupt flag**	**Port C, bit 3**	**Read only**	**This flag is set by the 8255 PPI when requesting an interrupt.**
nIRQ7	**PC interrupt request disable**	**Port B, bit 1**	**Write only**	**This flag is set by the FAXMODEM CPU to disable interrupt requests to the PC.**

The interrupt service routine dealing with the 8255 PPI interrupts viz. PCIOISR calls two procedures, viz. PCIOGET and PCIOPUT, which manage the flow of data to and from Port A.

6.9.2.1 CPU reading Port A - PCIOGET procedure

- The status of the input-buffer-full flag of Port A (IBFa flag) is checked. This flag is set whenever the PC writes data into the bi-directional buffer. If this flag is not set, i.e. if Port A is empty, this procedure is terminated.
- The input-buffer-full interrupt enable flag of Port A (INTE_IBFa flag) is checked. This flag is cleared by the FAXMODEM operating system to disable the 8255 PPI from generating interrupts when the PC writes to Port A. The status of this flag therefore signifies whether this interrupt was generated by incoming data from the PC. Therefore if this flag is not set, this procedure is terminated.
- The storage area for incoming data from the PC (INPBUF) is checked.

  If the INPBUF is full the INTE_IBFa flag is cleared and this procedure is terminated.

  If the INPBUF is filled to its capacity - 1, the INTE_IBFa flag is cleared .

  If the INPBUF has space for more than one data byte , the INTE_IBFa flag is set.
- Data is read from Port A and stored in INPBUF

6.9.2.2 CPU writing to Port A - PCIOPUT procedure

- The status of the output-buffer-full flag of Port A (OBFa flag) is checked. This flag is set whenever the PC reads data from the bi-directional buffer (Port A). If this flag is not set, i.e. if Port A is not empty, this procedure is terminated.
- The output-buffer-full-interrupt-enable flag of Port A (INTE_OBFa flag) is checked. This flag is cleared by the FAXMODEM operating system to disable the 8255 PPI from generating interrupts when the PC reads data from Port A. The status of this flag therefore signifies whether this interrupt was generated by outgoing data to the PC. Therefore if this flag is not set, this procedure is terminated.

- The inhibit-output-buffer flag (INH_OBF) is checked to see if the PC's input buffer can accept more data.

  If INH_OBF is set, i.e. if the PC's input buffer is full, no data is written to Port A and software flag (INH_OBF_FLAG) is set to indicate that Port A was left empty due to a full PC input buffer. The output-buffer-interrupt-enable flag (INTE_OBFa) is reset to remove the HIGH signal from the FAXMODEM CPU's interrupt pin and the interrupt request line to the PC (IRQ7) is disabled. This procedure is then terminated without writing any data to Port A.

- The enable-input-buffer flag (ENA_IBF) is checked to see if the PC wants to send data to the FAXMODEM.

  If ENA_IBF is set, i.e. if the PC's output buffer contains data, the output-buffer interrupt enable flag (INTE_OBFa) is reset to remove the HIGH signal from the FAXMODEM CPU's interrupt pin and the interrupt request line to the PC (IRQ7) is disabled. This procedure is then terminated without writing any data to Port A.

- The storage area for outgoing data to the PC (OUTBUF) is checked.

  If the OUTBUF is empty INTE_OBFa is reset to remove the HIGH signal from the FAXMODEM CPU's interrupt pin and IRQ7 is disabled. This procedure is then terminated without writing any data to Port A.

  If the OUTBUF contains only one character, IRQ7 is enabled, INTE_OBFa is cleared and the last remaining character is read from OUTBUF and written to Port A.

  If the OUTBUF contains more than one character, IRQ7 is enabled, a character is read from OUTBUF and written to Port A.

6.9.3 The RING interrupt service routine - INT2

Ringing current will cause this interrupt to be generated every 60 milliseconds. Four consecutive interrupt cycles will signal the detection of ringing on the line. Approximately 240 milliseconds after the first interrupt a ring counter is incremented, a ring flag is set and a ring pulse of 1 second is sent to the DATA MODEM chip. During the 1 second that follows the 240 milliseconds, all ring interrupts are ignored. The detection of the next ringing current pulse will again start with the detection of four consecutive ring interrupts.

6.9.4 Timer 2 interrupt service routine

This is the interrupt with the highest priority and controls all the timing for the FAXMODEM operating system. Timer 2 is initialized during the cold boot routine. It is set up to run continuously and to generate an interrupt every 1 millisecond. The following functions are performed by the Timer 2 interrupt service routine.

- The System time is updated. All fax files are time and date stamped with the system time.
- The software timers, Timer 3 and Timer 4 are updated.
- The Ring timer is updated. This timer checks for long pauses between ring cycles. If a pause of more than approximately 5800 milliseconds is detected, the ring detection logic is reset signaling the suspension of ringing current.
- The Ring time is updated. The ring interrupt service routine initializes this software timer at the beginning of the ring pulse to the DATA MODEM. When this software timer times out, the ring pulse to the DATA MODEM is ceased and the ring flag is reset.
- The MSECOND operating system counter is decremented. This counter is used to generate time delays during the reception and transmission of facsimiles.
- The status of the OUTBUF, and the flags INH_OBF and ENA_IBF are checked. If the OUTBUF is not empty and the PC's input buffer is not full, i.e. INH_OBF is reset, and the PC's output buffer is empty, i.e. ENA_IBF is reset, the output-buffer-interrupt-enable flag (INTE_OBF) is set and software flag, INH_OBF_FLAG is reset. This action will cause an INT1 to occur if Port A is empty.
- The status of the INPBUF is checked. If the INPBUF is not full the input-buffer-interrupt-enable-flag (INTE_IBF) is set. This will cause and INT1 to occur if Port A contains data.
- The 'In Service' bit of Timer 2 interrupt is reset and the interrupt service routine is terminated.

6.10 PC input and output buffers

All data transfers to or from the PC are achieved by means of interrupts. Buffers were therefore essential to store data temporarily before being transferred to or from the PC. Two storage areas, one for incoming (INPBUF) and the other for outgoing data (OUTBUF) were set aside to store this data.

6.10.1 Output buffer (OUTBUF)

The buffer OUTBUF is a 1024 byte circular buffer. Three variables are associated with this buffer and the following functions are assigned to it.

- Read pointer: The variable *GetPntOUT* points to the tail of the circular buffer (OUTBUF). It is used by the PCIOPUT procedure to retrieve data from OUTBUF.
- Write pointer. The variable *PutPntOUT* points to the head of the circular buffer (OUTBUF). It is used by the PUTCHR procedure to write data to OUTBUF.
- Counter. The variable *OUTBUFCNT* is always equal the amount of data present in OUTBUF. When data is written *OUTBUFCNT* is incremented and when data is retrieved *OUTBUFCNT* is decremented.

6.10.1.1 Writing to the OUTBUF - The PUTCHR procedure

This procedure is used for all data transfers to the PC. It takes a single character as an input parameter and writes it to the output buffer (OUTBUF). The following steps briefly describe this procedure.

- The status of OUTBUF is checked. If it is not full the data character is written to OUTBUF.
- The status of INH_OBF is checked. If INH_OBF is not set, meaning that the PC can accept data, the output-buffer-interrupt-enable flag (INTE_OBF) is set. This will cause a INT1 to occur if Port A is or becomes empty.
- If no data was written due to a full OUTBUF, the preceding steps will be retried four times before this procedure is abandoned.

6.10.2 Input buffer (INPBUF)

The buffer INPBUF is a 1024 byte circular buffer. Three variables are associated with this buffer and the following functions are assigned to it.

- Read pointer: The variable *GetPntINP* points to the tail of the circular buffer (INPBUF). It is used by the GETCHR procedure to retrieve data from INPBUF.
- Write pointer. The variable *PutPntINP* points to the head of the circular buffer (INPBUF) and is used by the PCIOGET procedure to write data to INPBUF.
- Counter. The variable *INPBUFCNT* is always equal the amount of data present in INPBUF. When data is written to INPBUF, *INPBUFCNT* is incremented and when data is retrieved *INPBUFCNT* is decremented.

6.10.2.1 Reading the INPBUF - The GETCHR procedure

This procedure is used to read data transferred from the PC. It returns a single character read from the input buffer (INPBUF). The following steps briefly describe this procedure.

- The status of INPBUF is checked. If it is not empty a data character is read from INPBUF.
- The input-buffer-interrupt-enable flag (INTE_IBF) is set. This will cause a INT1 to occur if Port A contains incoming data.
- If no data was read due to an empty INPBUF, the preceding steps will be retried four times before this procedure is abandoned.

6.11 Interpreting incoming data from PC

Data is transferred between the FAXMODEM and the PC by packaging it into a predetermined format. This is necessary to maintain control over the flow of data between PC and FAXMODEM. Data packages transferred from FAXMODEM to PC are called *requests* and in the opposite direction it is called *commands.*

6.11.1 The command filter

Commands are filtered out by the *FILTER* procedure. The FILTER procedure is indirectly called in the foreground by the MAIN procedure. The command filter does the following:

- Call GETCHR procedure to retrieve a byte from INPBUF. If no bytes are available the FILTER procedure is immediately abandoned.
- Data bytes retrieved from GETCHR is searched for an identification sequence of bytes (COMMAND_ID). The COMMAND_ID signals the start of a command.
- The command and all associated data are now retrieved.
- Depending on the type of command, it is performed immediately or stored in memory to be performed at a later stage.

6.12 Memory management

The FAXMODEM memory is divided into four main areas, viz. the Data segment (DSEG), Code segment (CSEG), Variables segment (RSEG) and the FAR HEAP. The Data and Code segments are located inside the EPROM address range and the Variables segment is located in the first 64k of RAM, i.e. 00000 - 0FFFFh. The FAR HEAP stretch from the end of the Variables segment to the extent of the RAM.

6.12.1 Data segment

The Data segment (DSEG) contains among other data the following important system tables:

- The interrupt vector table. This table is copied to RAM address 00000 during startup, before interrupts are enabled.
- The black and white binary trees used in the decoding of Rec T.4 modified Huffman facsimile data.
- The black and white code tables used when encoding Rec T.4 modified Huffman facsimile data.

- The normal and high resolution font tables. These tables are used when converting text files to facsimile image files.

6.12.2 Code segment

The code segment contains all the operating system code of the FAXMODEM. The MAIN procedure resides in the Code segment. The Boot and Cold segments are located at the bottom of the code segment.

6.12.2.1 Cold segment

The cold start procedure is located in the cold segment. The CPU control registers are initialized in this segment.

6.12.2.2 Boot segment

The Boot segment is only sixteen bytes long and contains a long jump to the cold start procedure.

6.12.3 Variables segment

The variables segment contains all the system variables and is used as a temporary or semi-permanent storage area for system data. The following important system data areas are located in the variable segment:

- The input buffer (INPBUF). This is a circular buffer for incoming data from the PC.
- The output buffer (OUTBUF). This is a circular buffer for outgoing data to the PC.
- The message buffer (MSGBUF). A circular buffer for temporary storage of transmit and received facsimile image data.
- The NEAR HEAP. This is a scratch pad area for the FAXMODEM operating system. Data areas on the near heap are created on request by the *MALLOC* procedure and deleted on request by the *FREE* procedure. The use of the NEAR heap therefore reduces the amount of storage area necessary as data areas are dynamic and not static.
- The STACK.

6.12.4 Far heap

The FAR heap uses the same principle as the NEAR heap to allocate dynamic data storage areas. The FAR HEAP uses far pointers as it can be several segments (64k byte blocks) long. The FAR heap is primarily used to store facsimile image data, i.e. fax files.

6.13 Transmit scheduling

The FAXMODEM can perform transmit scheduling, i.e. send one or more facsimile documents to one or more remote machines at a given time. This capability greatly reduces the amount of processes the PC needs to perform when transmitting facsimile documents.

6.13.1 Reading a schedule from PC

A schedule is received from the PC in the form of a PC command. The FILTER procedure detects this data and stores it on the FAR heap as an ASCII file.

6.13.2 Processing a schedule

The PROCESS_SCHED procedure opens the ASCII schedule file and converts all its lines to SCHEDULE_ITEMS. (See Appendix B).

6.13.3 Servicing a schedule

This procedure performs the actual tasks requested in the transmit schedule, which by this stage should be converted to a list of SCHEDULE_ITEMS. The SERVICE_SCHED procedure steps through the FAX_SCHEDULE list and compares the send-time of each SCHEDULE_ITEM to the current time. If a SCHEDULE_ITEM has reached or passed its send-time, a SCHEDULE_JOB structure is created and the sessions contained in the SCHEDULE_ITEM is passed to that SCHEDULE_JOB before it is executed. The SCHEDULE_ITEM is removed from memory and deleted from the FAX_SCHEDULE. If the job, i.e. the transmission of faxes was not executed successfully due to the remote fax machine being busy, etc. the SCHEDULE_JOB is added to a list of incomplete jobs named the JOB_SCHEDULE.

6.14 Logging activities

A file is created on the PC hard disk to be used as a storage area for information regarding the progress and result of transmitted and received facsimile documents.

6.14.1 Adding to log

When transmitting facsimile documents, all information affecting the successful reception of the documents by the remote terminal equipment is written to a temporary storage area on the NEAR heap. The same is done when receiving documents from a remote machine. This logged data is stored temporary until the FAXMODEM enters the MAIN procedure.

6.14.2 Dumping log to PC

Logged data stored temporarily is transferred to the PC by the DUMP_LOG procedure. This operation takes place in the foreground, i.e. called by the MAIN procedure, so that it does not interfere with facsimile image data being transferred between PC and FAXMODEM. If a log file does not exist on the PC's hard drive, this procedure will request the PC to create a new log file.

CHAPTER 7

Personal Computer Software

Software had to be developed for the PC environment to enable communication between the PC operating system and the FAXMODEM. This software is divided into two files.

- A ***device driver*** *(FAXDRV.SYS)* containing all the code for receiving and sending of data to the FAXMODEM
- A small ***executable program*** *(FAXTSR.EXE)* responsible for loading the *configuration information* and providing the device driver with a *program segment prefix* (PSP) to enable disk IO.

This chapter describes the design and operation of this software.

7.1 Configuring the FAXMODEM

The FAXMODEM needs to be configured with the users' preferences when it is initialized. This user information is kept in the configuration file.

7.1.1 The configuration file

The default file name for the configuration file is *FAX.CFG*. This file is loaded and processed by the *FAXTSR.EXE* executable file. Table 7-1 shows the various FAXMODEM system variables that can be set in the configuration file and Figure 7-1 is an example of a typical configuration file.

Table 7-1 *FAXMODEM configuration keywords*

Keyword	Meaning	Default value	Description
DELAY	**Retry delay**	**60**	**If the FAXMODEM has failed to transfer a facsimile document to a remote machine, this shall be the minimum amount of time in seconds that it will wait before attempting another transmission.**
RING	**Ring count**	**1**	**This is the amount of rings necessary before the FAXMODEM answers. This value should be set equal to the S0 register, i.e. *Number of Rings* setting in the DATA MODEM.**
WAIT	**Wait for carrier**	**10**	**This value indicates to the FAXMODEM the time (in seconds) that the DATA MODEM will wait for a carrier signal. This value should therefore be set equal to the time set in the S7 register of the DATA MODEM.**
DIAL	**Dial mode**	**PULSE**	**This value sets the default dial mode of the FAXMODEM. Valid values are: *DTMF* or *TONE* for multi-frequency dialing and *PULSE* for pulse dialing.**
IN	**Directory for received documents**	**Current directory**	**The full path name for the received documents.**
OUT	**Directory for outgoing documents**	**Current directory**	**The full path name for outgoing documents.**
HOME	**Home directory for FAXMODEM**	**Current directory**	**The full path name for FAXMODEM data files.**
CSI	**Called subscriber ID**	**Null**	**Identification signal transmitted after answering a call.**
TSI	**Transmitting subscriber ID**	**Null**	**Identification signal transmitted when making a call.**

7.1.1.1 Important configuration settings

The FAXMODEM controls the answering of incoming calls. It distinguishes between calls destined for itself and the DATA MODEM with the aid of the *RING* and *Wait_For_Carrier* settings. (Adaptive answering of calls is described in section 6.4, page 28.) It is therefore essential for the correct handling of incoming calls that these two configuration settings are set absolutely correct.

```
DELAY = 180

RING   = 3

WAIT   = 15

DIAL   = DTMF

IN     = C:\FAXDIR\IN

OUT    = C:\FAXDIR\OUT

HOME   = C:\FAXDIR

CSI    = 27 21 555 1234

TSI    = 021 555 1234
```

Figure 7-1 Example of a FAXMODEM configuration file

7.1.2 The DATA file

This file is created by the FAXTSR.EXE program and contains the FAXMODEM configuration information in binary format. The default file name is FAX.DAT.

7.1.3 The Startup file - FAXTSR.EXE

The FAXTSR.EXE file is used to configure the FAXMODEM. It should be executed from the AUTOEXEC.BAT file initially but can subsequently be used to update the FAXMODEM. The following functions are executed:

- It reads the *DATA file* (FAX.DAT) to get the previous configuration data. If a *DATA file* does not exist, a new one is created and the directory for incoming facsimile documents is scanned to determine the *filename* for the next incoming file.

- Reads the configuration data from configuration file.
- Update the *DATA file* with the new information.
- Processes the configuration information and send it to the device driver.
- Interrogate device driver to see if it contains a program segment prefix (PSP). If it does not contain one, FAXTSR.EXE will relinquish its PSP to the device driver.

The *FAXTSR.EXE* executable should be informed of the path to the configuration file. This is accomplished by the following:

- Setting an environment variable viz. FAX equal to the path name of the configuration file. This should be done before executing FAXTSR.EXE. The environment variable is set as follows:

  ⇒ SET FAX=c:\faxdir

  Where c:\faxdir is the path name.

- Providing the path with the configuration file name as a parameter, e.g.

  ⇒ FAXTSR.EXE **c:\faxdir\myfax.cfg**

  This will override the environment variable setting.

If no path is declared the FAXTSR.EXE executable will search the current directory for the configuration file.

7.2 Device Driver

A device driver was developed to enable DOS access to the FAXMODEM. Applications running on the PC could therefore use DOS to communicate with the FAXMODEM.

7.2.1 Installing the device driver

The device driver is installed when the PC is rebooted and can be loaded into an upper memory block on 386 or higher machines. The device driver filename is *FAXDRV.SYS* and is loaded by adding one of the following lines to the CONFIG.SYS file. This will inform DOS to load the FAXMODEM device driver when rebooting:

⇒ DEVICE=C:\FAXDIR\FAXDRV.SYS

⇒ DEVICEHIGH=C:\FAXDIR\FAXDRV.SYS

The operator should replace C:\FAXDIR\ with the path name to the FAXDRV.SYS.

7.3 FAXMODEM instructions

This device driver process data originating from PC user applications by extracting only keywords with their associated information that is applicable to the FAXMODEM. Characters that follow a semicolon (';') in a text line are treated as comments and are ignored. Only keywords that start at the beginning of line with a colon (':') are recognized. The following keywords are recognized and processed:

• **:SEND**	This keyword indicates the start of a TRANSMIT SCHEDULE
• **:LISTNAME**	This indicates the start of a LIST. 'LISTNAME' should be replaced by characters representing the name of the list.
• **:END**	This keyword indicated the end of a LIST or TRANSMIT SCHEDULE

7.3.1 The TRANSMIT SCHEDULE

The TRANSMIT SCHEDULE starts with the :SEND keyword and ends with the :END keyword and consists of one or more lines of ASCII text (SCHEDULE_LINES). Each SCHEDULE_LINE is an instruction to the FAXMODEM to send one or more facsimile documents to one or more remote machines. A SCHEDULE_LINE can have one of the following formats:

⇒ Date	Time	Filename	Telephone_Number	Number_Of_Retries
⇒ Date	Time	@Filelist	Telephone_Number	Number_Of_Retries
⇒ Date	Time	Filename	@Numberlist	Number_Of_Retries
⇒ Date	Time	@Filelist	@Numberlist	Number_Of_Retries

Where:

- The fields are delimited by spaces

- *Date* = the send date in one of the following formats

⇒	YYYY-MM-DD	*e.g.*	1994-11-10
⇒	YY-MM-DD	*e.g.*	94-11-10

- *Time* = the send time in one of the following formats

⇒	HH:MM:SS	*e.g.*	23:59:50
⇒	HH:MM	*e.g.*	23:59
⇒	HH:MM am	*e.g.*	09:30 am
⇒	HH:MM pm	*e.g.*	11:59 pm

- *Filename* = the name of facsimile document to be transmitted in one of the following formats:

⇒	c:\path\filename */parameter*	*e.g.*	c:\faxdir\out\fax0001.tif
⇒	filename */parameter*	e.g.	fax0001.tif

The FAXMODEM will use the default path for outgoing facsimiles specified in the *configuration file* if no path is supplied.

The following *parameters* are recognized:

⇒	Q	indicates that the file is in QL2FAX format
⇒	X	indicates that the file is an ASCII text file
⇒	T	indicates that the file is in TIFF format
⇒	G	indicates that the file contains a list of telephone numbers or filenames
⇒	1	indicates that this file should be converted using the high resolution font table. This parameter is applicable only when the file is an ASCII text file.

- *Filelist* = The name of a ASCII text file or the name of a LIST containing the file name(s) of facsimile documents to be transmitted

- *Telephone_Number* = The telephone number of the destination facsimile machine. The following characters can be used in the telephone number string:

⇒ 0 - 9	Digits zero to nine
⇒ A,B,C,D,*,#	These characters are ignored if FAXMODEM is pulse dialing.
⇒ ,	A comma will produce a 2 second pause.
⇒ /T	Switch the FAXMODEM dial mode to DTMF dialing
⇒ /P	Switch the FAXMODEM dial mode to PULSE dialing
⇒ ( )	Round brackets can be used to group digits

- *Numberlist* = The name of a ASCII text file or the name of a LIST containing the telephone number(s) of the destination facsimile machines

Number_Of_Retries = The number of times the FAXMODEM should retry to send the facsimile documents.

```
;==========================================================================

; This line is a comment

; Example TRANSMIT SCHEDULE

; DATE = 12 JUNE 1995

:SEND

; Date          Time    Filename                          Tel No          Retry

;=====          ====    ========                          ======          =====

1995/06/12      12:00   C:\FAXDIR\OUT\EXAMPLE.TIF  /T     0215555555      3

1995/06/12      14:05   @C:\FAXDIR\MFILELST.TXT  /X       5555555         3

1995/06/12      18:00   C:\FAXDIR\OUT\AGENDA.TIF  /T      @TELNO.LST      3

:END

;==========================================================================
```

Figure 7-2 Example of a TRANSMIT SCHEDULE

7.3.2 The LIST

This can be a list of telephone numbers or facsimile document file names. The following rules should be followed when constructing a list of telephone numbers or file names.

- If the list is located in the same ASCII file as the TRANSMIT SCHEDULE, the name of the list (:LISTNAME) and the :END keyword are mandatory.
- If the list is located in a separate ASCII file, the name of the list (:LISTNAME) and the :END keyword can be omitted. The name of the ASCII file is treated as the name of the list.
- Only one list is allowed per ASCII list file.

Figure 7-3 is an example of an ASCII file containing a list of file names.

```
;______________________________________________________________________
; This is an example of a list of file names
; File name                              Description
;______                                  ________
C:\FAXDIR\OUT\AGENDA.TIF  /T             ; Agenda of meeting
C:\FAXDIR\OUT\MINUTES.TIF  /T            ; Minutes of meeting
;______________________________________________________________________
```

Figure 7-3 Example of a LIST of filenames

Figure 7-4 illustrates how lists could be merged into a TRANSMIT SCHEDULE file. This example contains a TRANSMIT SCHEDULE and two lists viz. a telephone number list (GROUP1) and a filename list (FILES1).

```
;**********************************************************************

:GROUP1

; TEL NO              COMPANY

; ______________________________

903 3762              ; PC FAXCARD

903 4549              ; FAX MACHINE

021 903 4549          ; NU TONE

:END

;**********************************************************************

:FILES1

; FILENAME

; ________

C:\OUT\EX.TXT  /X

C:\OUT\LAPCBL.TIF /T

C:\OUT\SAMPLE.TIF /T

:END

;**********************************************************************

:SEND

; DATE       TIME          FILENAME             TELNO                RETRY

; ________________________________________________________________________

94-06-13     10:00         C:\OUT\EX.TXT /T     9034549 /T1234        3

94-06-13     10:00         @FILES1              @GROUP1               3

94-07-13     08:59:10am    FAX0005.REC /T       903-4549              4

94-07-13     07:10am       FAX0003.REC /T       ((021)903 4549)       2

:END

;**********************************************************************
```

Figure 7-4 Example of a TRANSMIT SCHEDULE with a telephone number and filename list

7.4 Device Driver operation

Using a device driver simplifies the method of access to the FAXMODEM, as basic DOS INT 21h function calls can be used to write data to, or read data from the FAXMODEM. In addition the device driver provides the following functions to the FAXMODEM:

- DOS file functions, e.g. opening, closing, reading and writing of hard disk files
- Screen functions, e.g. positioning of cursor, writing to screen
- Accepting and parsing of commands and requests received from operator and / or applications.
- Reading of configuration data.
- Reading of operating system time from PC system clock.

The device driver is compiled into a file named FAXDRV.SYS and after installation it is available to the operation system as a device named FAX0.

7.4.1 Device driver initialization

All DOS device drivers are initialized before COMMAND.COM is loaded. Therefore on initialization only a limited amount of PC operating system functions are available. The following tasks are performed by FAXDRV.SYS when it is initialized:

- A boot message is displayed, informing the operator that FAXDRV.SYS is being loaded.
- The following non reentrant interrupts are trapped.
  - The screen BIOS interrupt, Int 10h.
  - The absolute disk read interrupt, Int 25h.
  - The absolute disk write interrupt, Int 26h.
  - The Disk BIOS interrupt, Int 13h.
- The MS-DOS version is read and stored in the *OS_VERSION* variable.
- The pointer to the INDOS flag is read and stored in *INDOS_FLAG_ADDR* variable.

- The base address of the video display buffer is determined and stored in the *VIDEO_SEG* variable.
- The circular buffer for incoming data from the FAXMODEM, i.e. the PC's input buffer is initialized.
- The circular buffer for outgoing data to the FAXMODEM, i.e. the PC's output buffer is initialized.
- The following interrupt vectors are hooked:
  - The DOS Idle interrupt, Int 28h.
  - The clock tick interrupt, Int 1Ch.
  - Interrupt 0Fh. This is the FAXMODEM hardware interrupt.
- The FAXMODEM is informed that the PC has finished the initialization of the FAXMODEM device driver. The PC waits for the FAXMODEM to acknowledge this signal.
- If the FAXMODEM does not acknowledge within 20 seconds the following steps are taken:
  - The hooked interrupt vectors, Int 0Fh, Int 1Ch and Int 28h are restored.
  - The trapped interrupt vectors, Int 10h, Int 13h, Int 25h and Int 26h are restored.
  - An abort message is displayed.
  - The initialization process is abandoned and the device driver is removed from memory.
- If the FAXMODEM acknowledges within 20 seconds, a confirmation message is displayed and the initialization process is exited.

7.4.2 Device Driver Components

The device driver can be divided into three major components:

- The DOS interface
- The FAXMODEM interface
- The Task Dispatcher

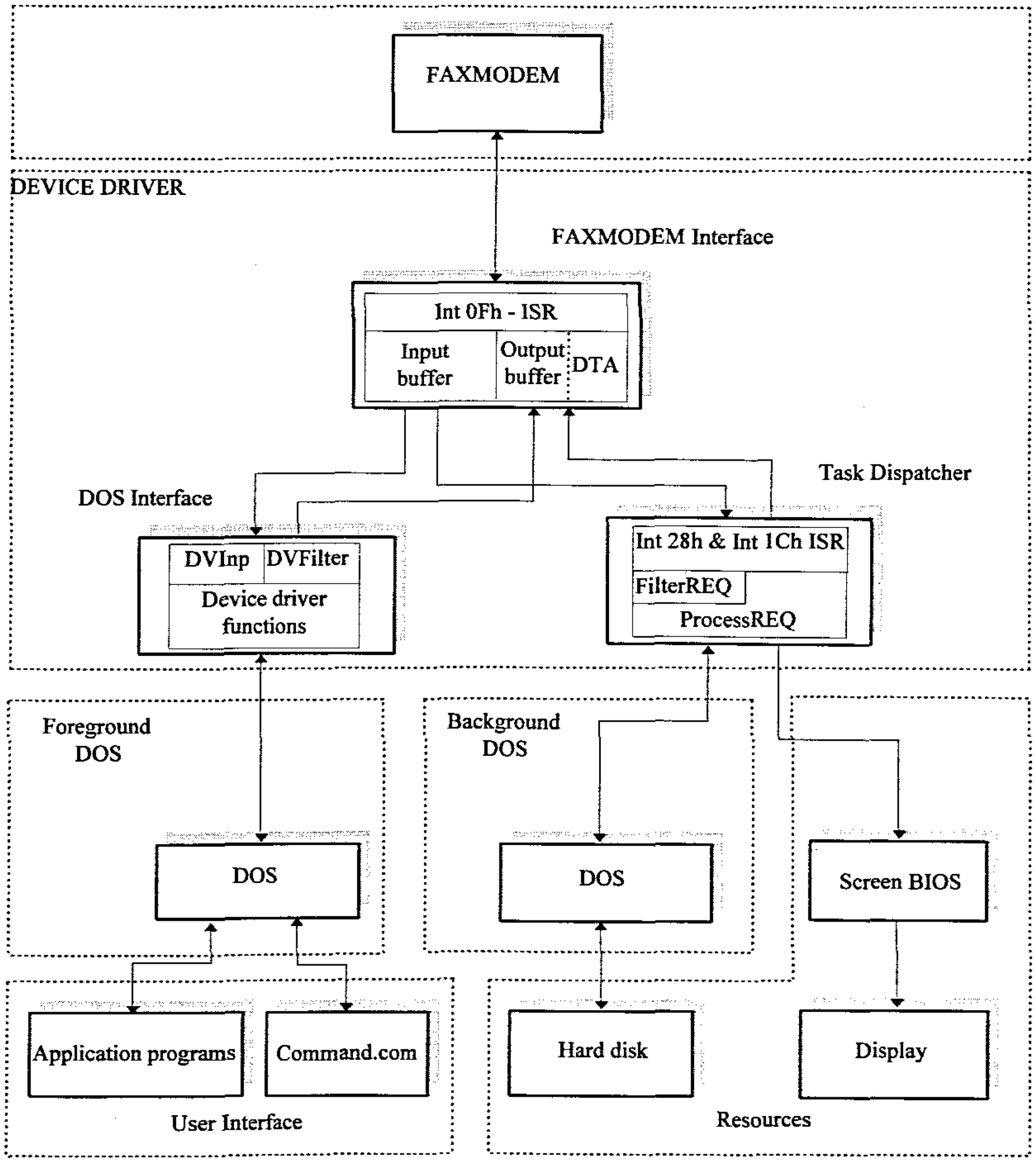


Figure 7-5 Device driver components within PC communication framework

7.4.2.1 The DOS interface

This section of the device driver communicates with the DOS environment. The device driver supports the functions listed in Table 7-2. These functions are available for use by the DOS kernel.

Table 7-2 Device Driver Functions

	Purpose	Function description
Function 00	**initialization**	**This function is performed only once, when rebooting.**
Function 03	**IOCTL read**	**The last IO control data written to the FAX device driver is returned by this function**
Function 04	**read**	**This function returns a single character read from the device driver input buffer (DVinp).**
Function 05	**non destructive read**	**This function reads DVinp without updating any pointers or counters.**
Function 06	**input status**	**This function returns the status of DVinp**
Function 07	**flush input buffers**	**This function clears DVInp.**
Function 08	**write**	**A single character is written to the FAX device driver *output filter*.**
Function 10	**output status**	**This function returns the status of the PC's output buffers.**
Function 11	**flush output buffers**	**This function is ignored.**
Function 12	**IOCTL write**	**This function sets the *BackGndPSP* variable and writes control information to the FAXMODEM.**
Function 13	**open device**	**This function clears the data transfer area (DTA) and initializes the *output filter* of the FAX device driver.**
Function 14	**close device.**	**This function writes all remaining data to the PC's output buffer.**

IO address 0301h

7	6	5	4	3	2	1	0
nOBF	INTR	INH_OBF			ENA_IBF		IBF

Figure 7-6 FAXMODEM status port

7.4.2.1.1 The Device Driver Output Filter

This procedure parse ASCII text data written by DOS to the device driver. It searches the stream of characters for the keywords described in section 7.3. If a keyword is found, its associated data are read and converted into a FAXMODEM *command* package. The package is then delivered to the FAXMODEM interface section's output buffer for transmission to the FAXMODEM.

7.4.2.2 The FAXMODEM interface

This is an interrupt service routine (ISR) for the FAXMODEM hardware interrupt (IRQ7). Two I/O ports are used to communicate with the FAXMODEM. I/O port 300h is the DATA PORT and I/O port 301h is the STATUS PORT. Figure 7-6 depicts the bit fields of the STATUS PORT.

The ISR for IRQ7 is responsible for the transfer of data between the DATA PORT and the I/O buffers of the device driver. The ISR operates as follows:

1. The ISR calls the CARDGET procedure.
2. The ISR calls the CARDOUT procedure.
3. The status port (I/O port 301h) is read. If the INTR bit is set and one or both of the nOBF or IBF bits are reset, steps 1 to 3 are repeated.
4. The In Service bit for IRQ7 in the interrupt controller is reset and the ISR is exited.

7.4.2.2.1 The CARDGET procedure

This procedure is responsible for the reading of data from the FAXMODEM. This procedure function as follows:

- The input buffer is checked. If it is full the INH_OBF bit is set and this procedure aborted.
- If the input buffer contains space for only one more character, the INH_OBF bit is set.
- If the input buffer contains space for more than one character the INH_OBF bit is reset.
- The nOBF bit is tested. If it is set, i.e. if the DATA PORT is empty, this procedure is terminated.
- The IBF bit is tested to see if the DATA PORT contains outgoing data. If the IBF bit is set, the DATA PORT is read first and the data stored in the input buffer. Thereafter old outgoing data that was in the DATA PORT is re-written.
- If the IBF bit is reset the DATA PORT is read and the data transferred to the input buffer.

7.4.2.2.2 The CARDOUT procedure

This procedure is responsible for the writing of data to the FAXMODEM. This procedure gets its data from one of two buffers:

- *the output buffer* contains data that is prepared in a packaged format, i.e. *commands,* while
- *the Data Transfer Area (DTA)* contains data read from the hard disk.

This procedure function as follows:

- The *PUTEXEC* variable is checked. This variable indicates which of the following two processes should be executed:

  ⇒ *CARDPUT.* This procedure transfers output buffer data to the DATA PORT.

  ⇒ *PUT_DTA.* This procedure transfers DTA data to the DATA PORT.

- Control is passed to one of the procedures

7.4.2.2.2.1 The CARDPUT procedure
This procedure executes as follows:

- The output buffer is checked. If it is empty the ENA_IBF bit is reset and the procedure terminated. The resetting of ENA_IBF prevents interrupts to be generated when the FAXMODEM reads the DATA PORT.
- If the output buffer contains one byte only, the ENA_IBF bit is reset.
- If the output buffer contains more than one byte, the ENA_IBF bit is set.
- The IBF bit is checked. If it is set, outgoing data will still be present in the DATA PORT hence the procedure is aborted.
- A character is read from the output buffer and written to the DATA PORT.

The characters read from the output buffer is scanned for PACKET_ID's, i.e. a string of characters signaling the start of a PACKET_HEAD. A PACKET_HEAD contains information regarding data that was read from the hard disk, i.e. data that was read into the DTA. If a PACKET_ID is detected, the *PUT_EXEC* variable is set which will cause the PUT_DTA process to be started at the next interrupt.

7.4.2.2.2.2 The PUT_DTA procedure
This procedure executes as follows:

- The DTA is checked and if it is empty:
  - ⇒ the output buffer is checked:

    if the output buffer is empty, the ENA_IBF bit is reset

    if the output buffer is not empty, the ENA_IBF bit is set.
  - ⇒ the *PUT_EXEC* variable is reset to signal that the DTA transfer is completed.
  - ⇒ the PUT_DTA procedure is terminated.
- The ENA_IBF bit is set
- The IBF bit is tested. If the IBF bit is set the PUT_DTA procedure is exited.

- A character is read from the DTA and written to the DATA PORT.

- Step 1 is repeated

- The PUT_DTA procedure is exited.

7.4.2.3 The Task Dispatcher

This section of the device driver is responsible for hard disk and display I/O. As it is necessary to run this processes in the background, the dispatcher is hooked onto the following two interrupts.

⇒ The PC's clock tick interrupt (Int 1Ch). This interrupt occurs 55 times a second. This interrupt is invoked by Int 08 which is the ISR for the IRQ0 hardware interrupt.

⇒ The DOS idle interrupt (Int 28h). This interrupt is invoked by DOS when it is waiting for command line input.

Both interrupts call the *DISPATCH* procedure. This procedure performs the hard disk and display I/O tasks.

7.4.2.3.1 The Clock Tick Interrupt

The following steps describe the execution of this interrupt service routine

- On entry the I/O buffers is checked and the following action is taken:

  ⇒ If the output buffer is not empty the ENA_IBF bit is set

  ⇒ If the input buffer is not full the INH_OBF is reset.

- The *InsideINT* flag is tested. If it is not zero this interrupt service routine is aborted. This is done because the DISPATCH procedure is non-reentrant.

- The status of PC's 8259 programmable interrupt controller unit (PICU) is read. If any other interrupt than IRQ0 is pending, this procedure is aborted. This places the ISR at a lower priority than the other hardware interrupts.

- The following flags inside the DOS kernel's data area is checked:
  - ⇒ The *INDOS_FLAG*. This flag is set by the DOS kernel when it not safe to re-enter DOS. As this ISR uses DOS for hard disk I/O, it is aborted if the INDOS_FLAG is not zero.
  - ⇒ The *CRITICAL_ERROR_FLAG*. This flag is set by the DOS kernel when processing a critical error. If this flag is not zero the ISR is aborted.
- The InsideINT flag is incremented.
- The InService bit for IRQ0 inside the 8259 PICU is reset to enable timer interrupts.
- The *DISPATCH* procedure is called,
- The InsideINT flag is decremented and the ISR is terminated.

7.4.2.3.2 The DOS Idle interrupt

The following steps describe the execution of this interrupt service routine

- On entry the I/O buffers is checked and the following action is taken:
  - ⇒ If the output buffer is not empty the ENA_IBF bit is set
  - ⇒ If the input buffer is not full the INH_OBF is reset.
- The *InsideINT* flag is tested. If it is not zero this ISR is aborted.
- The *InsideINT* flag is incremented.
- The *DISPATCH* procedure is called,
- The *InsideINT* flag is decremented and the ISR is terminated.

When DOS invokes this interrupt it is safe to re-enter DOS. The *INDOS_FLAG* and *CRITICAL_ERROR_FLAG* are therefore not checked.

7.4.2.3.3 The DISPATCH procedure

This procedure is executed as follows:

- The stack is switched to a local stack.
- The DOS extended error information is saved.
- The *REQUEST FILTER* procedure is called.
- The foreground program segment prefix (PSP) is restored.
- The foreground video status is restored.
- The DOS extended error information is restored.
- The original stack is restored and the procedure terminated.

7.4.2.3.4 The REQUEST FILTER

Requests from the FAXMODEM are filtered out by the *FILTER_REQ* procedure. This procedure does the following:

- Retrieves a byte from the input buffer. If no bytes are available the *FILTER_REQ* procedure is aborted.
- Data bytes retrieved from the input buffer is searched for an identification sequence of bytes (REQUEST_ID). The REQUEST_ID indicates the start of a request.
- The request and all associated data are retrieved.
- The request is processed by the *PROCESS_REQ* procedure in the following manner:
  - ⇒ If the request requires a DOS function to be performed, the *control-c* break is disabled before calling DOS. In addition, if the DOS function involves disk I/O, the current PSP is set to the background PSP. After a DOS call the *control-c* break is restored.
  - ⇒ If the request requires a screen BIOS function to be performed, the current foreground video mode and cursor position are saved and the cursor disabled.

- All data retrieved from the input buffer that are not packaged into a request format are dumped into the device driver input buffer (DVInp).

7.5 FAXMODEM user operation

After DOS has loaded the device driver, the device named *FAX0* is available to the operating system.

7.5.1 Instructing the FAXMODEM from an application

The following method describes what steps should be taken when an application wants to access the FAXMODEM.

- Open the device using the name FAX0 and DOS INT 21h, function 3Dh. This operation returns a HANDLE that should be used for all further IO.
- Write data using DOS INT 21h, function 40h. This data should be formatted into a transmit schedule or a list. See section 7.3 *FAXMODEM instructions* for the formats.
- Close the device using DOS INT 21h, function 3Eh.

7.5.2 Instructing the FAXMODEM from the command line

The PC operator can also use COMMAND.COM functions to write instructions to the FAXMODEM. This enables the PC operator to copy instructions from another device, e.g. the console (keyboard), or a file to the FAXMODEM.. The instructions should be in the format described in section 7.3, *FAXMODEM instructions.*

The following example uses COMMAND.COM to copy a schedule line from the console to the fax device:

```
copy con FAX0 Enter          this instructs COMMAND.COM to copy data from the console to the fax device

:SEND  Enter                 this is the keyword indicating the start of the transmit schedule

 95/07/13  13:33  C:\FAXDIR\OUTBOX\EXAMPLE.TIF /T  9034549 3  Enter

:END  Enter                  this is the keyword indicating the end of the transmit schedule

F6  Enter                    this signals the end of data input to COMMAND.COM
```

7.5.3 Instructing the FAXMODEM from a file

It is however useful in the case of larger transmit schedules, to create a text file first. The text file can then be copied to the fax device after all the schedule lines has been entered. The following example copies a text file named *sched01.txt* to the fax device:

```
copy   sched01.txt   FAX0   Enter
```

7.5.4 Sending lists to the FAXMODEM

The methods described in sections 7.5.1 to 7.5.3 can equally be used to copy file name and telephone number lists to the fax device.

CHAPTER 8

Problems encountered

Great difficulties were experienced during the development of the software for the FAXMODEM card and this chapter describes the problems encountered and the solutions.

8.1 Communication between PC and FAXMODEM card

The hardware on the FAXMODEM card had to be modified to enable reliable communication with the PC's IO channel.

The original hardware did not provide a method of handling simultaneous writes to the bi-directional buffer (i.e. 8255 PPI PORT A). This problem had to be addressed since both the PC and the FAXMODEM card would occasionally write large amounts of data (i.e. Fax image data) to one another. Corruption and loss of data did occur whenever FAXMODEM or PC tried to break into an incoming stream of data.

8.1.1 Limitations of PC - FAXMODEM interface before modification

Figure 8-1 depicts the PC I/O control circuits before it was modified. Software was written to work with this configuration but lock-ups, loss of data and corruption of data often occurred. The working of the PC I/O control circuit was analyzed and the following observations were made:

- The interrupt request line was monitored and it was noticed that there were times when it remained permanently high. This was attributed to the flip flop configuration that controlled the interrupt request line.
- Lock ups occurred more frequently when the PC's input buffer was allowed to fill up.

- The interrupt request line (IRQ5) is used as a hardware interrupt for the hard drive on the XT PC. It was observed that the BIOS interrupt service routine for this interrupt request disabled the IRQ5 line.
- A continuous stream of data was sent in both directions, i.e. from PC to FAXMODEM and from FAXMODEM to PC. Corruption and data loss occurred more frequently. This was attributed to the PC and FAXMODEM writing to the bi-directional port at the same time.

Due to the limited amount of status and handshaking lines between the PC and FAXMODEM (PC could only read OBF and IBF), it was impossible to write software that would maintain a high data transfer rate and still guaranteed data integrity. The PC I/O hardware therefore had to be modified so that it could provide the required handshaking lines.

The interrupt service routine written for the FAXMODEM interrupt had to be hooked onto an existing interrupt service routine. This meant that after finishing the ISR for the FAXMODEM the old or previous ISR had to be called. This practice ensured that the installation of the FAXMODEM software would not affect other existing programs that are using the same interrupt vector. The XT BIOS handling of the ISR was therefore not acceptable and another interrupt request line viz. IRQ7 was selected.

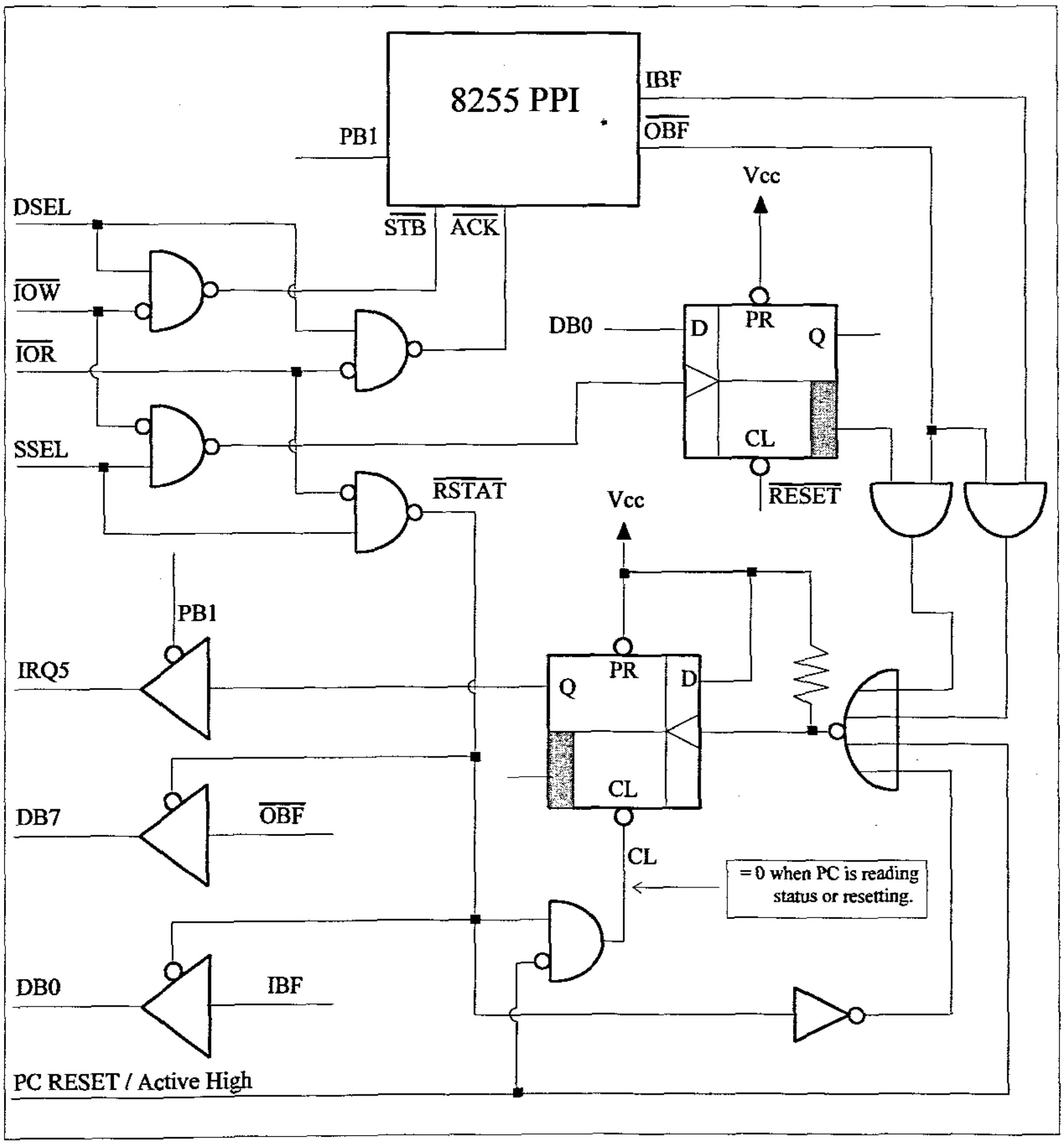


Figure 8-1 PC I/O Control before modification

8.1.2 Required control of interrupt requests

The PC required the following control over the interrupt request line:

- The ability to disable interrupts caused **only** by the FAXMODEM writing to the DATA PORT. This is required when the PC's input buffer becomes full. Disabling the interrupt will pull the interrupt request line low. If the PC subsequently read the input buffer, the interrupt can be enabled. This will cause the interrupt request line to go active if the bi-directional contains data destined for the PC.

- The ability to disable interrupts caused **only** by the FAXMODEM reading the DATA PORT. This is required when the PC's output buffer becomes empty. Disabling the interrupt will cause the interrupt request line to stay low when the FAXMODEM reads the DATA PORT. If the PC subsequently writes data to the output buffer, the interrupt can be enabled. This will cause the interrupt to go active immediately if the DATA PORT is empty.

8.1.3 Required handshaking lines

The following handshaking was required:

- A mechanism of informing the FAXMODEM when the PC's output buffer becomes empty.
- A mechanism of informing the FAXMODEM when the PC's input buffer becomes full.
- A means of informing the PC what action of the FAXMODEM caused an interrupt request to occur. This will indicate to the PC if it should read from or write to the DATA PORT.
- A status bit indicating to the PC the current state of the interrupt request line. This will indicate to the PC whether it should leave the interrupt service routine or continue reading/writing the DATA PORT.

8.1.4 PC I/O Control modification

Figure 8-2 shows the PC I/O control after the modification was done. This circuit provided the PC the means to:

- Disable IBF interrupt requests, i.e. interrupt requests generated when the FAXMODEM reads the DATA PORT. This is done by writing a '0' to the ENA_IBF flip flop.
- Disable OBF interrupt requests, i.e. interrupt requests generated when the FAXMODEM writes to the DATA PORT. This is done by writing a '1' to the INH_OBF flip flop.
- Read the following status information:

⇒ INTR = The status of the interrupt request line

⇒ IBF = The status of IBF flag. This flag indicates whether the FAXMODEM has read the DATA PORT.

⇒ OBF = The status of the OBF flag. This flag indicates whether the FAXMODEM has written to the DATA PORT.

⇒ ENA_IBF = The status of the ENA_IBF flip flop.

⇒ INH_OBF = The status of the INH_OBF flip flop.

The interrupt request line was changed from IRQ5 to IRQ7. The software applicable to this circuit is described in section 6.9.2 *The 8255 PPI interrupt service routine - INT1*, page 50 and section 7.4.2.2 *The FAXMODEM interface*, page 75.

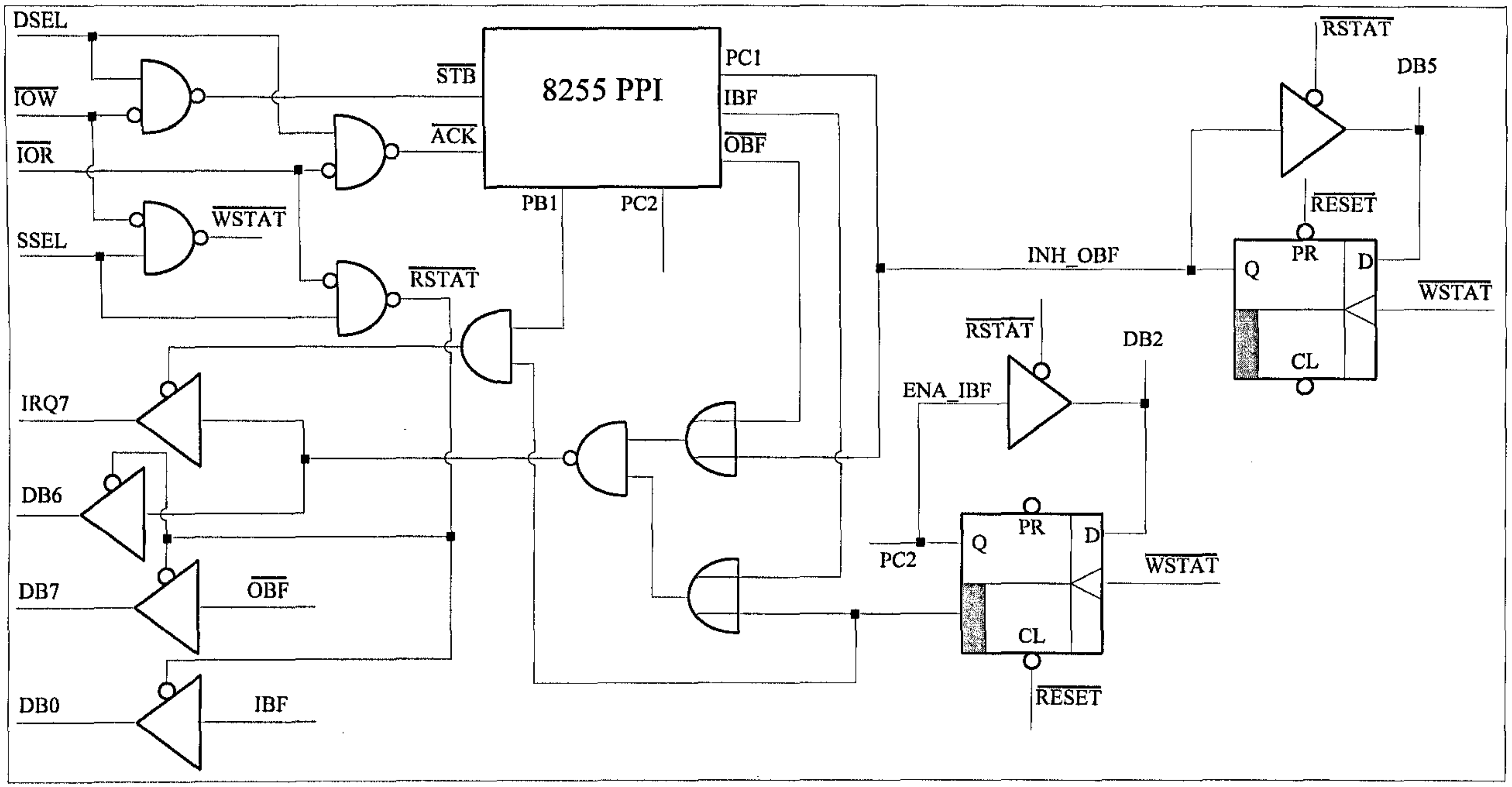


Figure 8-2 PC I/O Control after modification

8.2 Corruption of fax image data

During the initial testing phases, fax documents that were sent to a remote fax machine was often received with some of its image lines missing. It was discovered that when data was written to any of the CPU's internal control registers, the same data was written into memory.

8.2.1 Cause

Refer to Figure 8-3. Memory address space 00000 - 1FFFFh is selected by least significant output of the 74LS138 three-to-eight decoder (RAMS0).

> *When the CPU does an internal write to its IO mapped control registers, its bus controller signals the operation externally (i.e. the RD, WR, status, address, data, etc. lines are driven as in a normal bus cycle.)* - INTEL Embedded Microcontrollers and Processors, 1992

The first memory bank (RAM0) is therefore selected and its data overwritten.

8.2.2 Solution

The Lower Chip Select(LCS) signal however, does not go active when the CPU is writing to its internal control registers. The hardware was therefore modified so that LCS signal selects memory address space 00000 - 1FFFFh. See Figure 8-4.

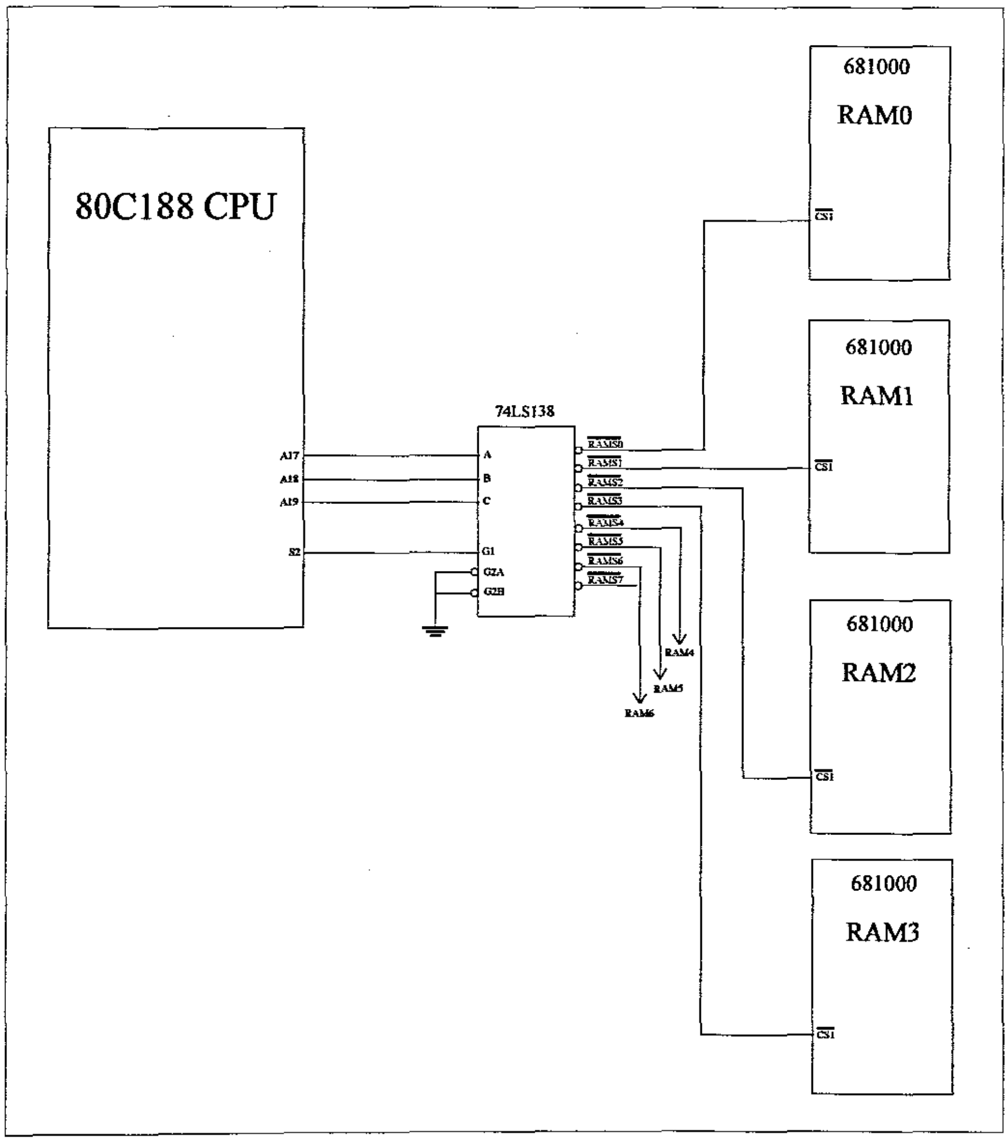


Figure 8-3 RAM Select logic before modification

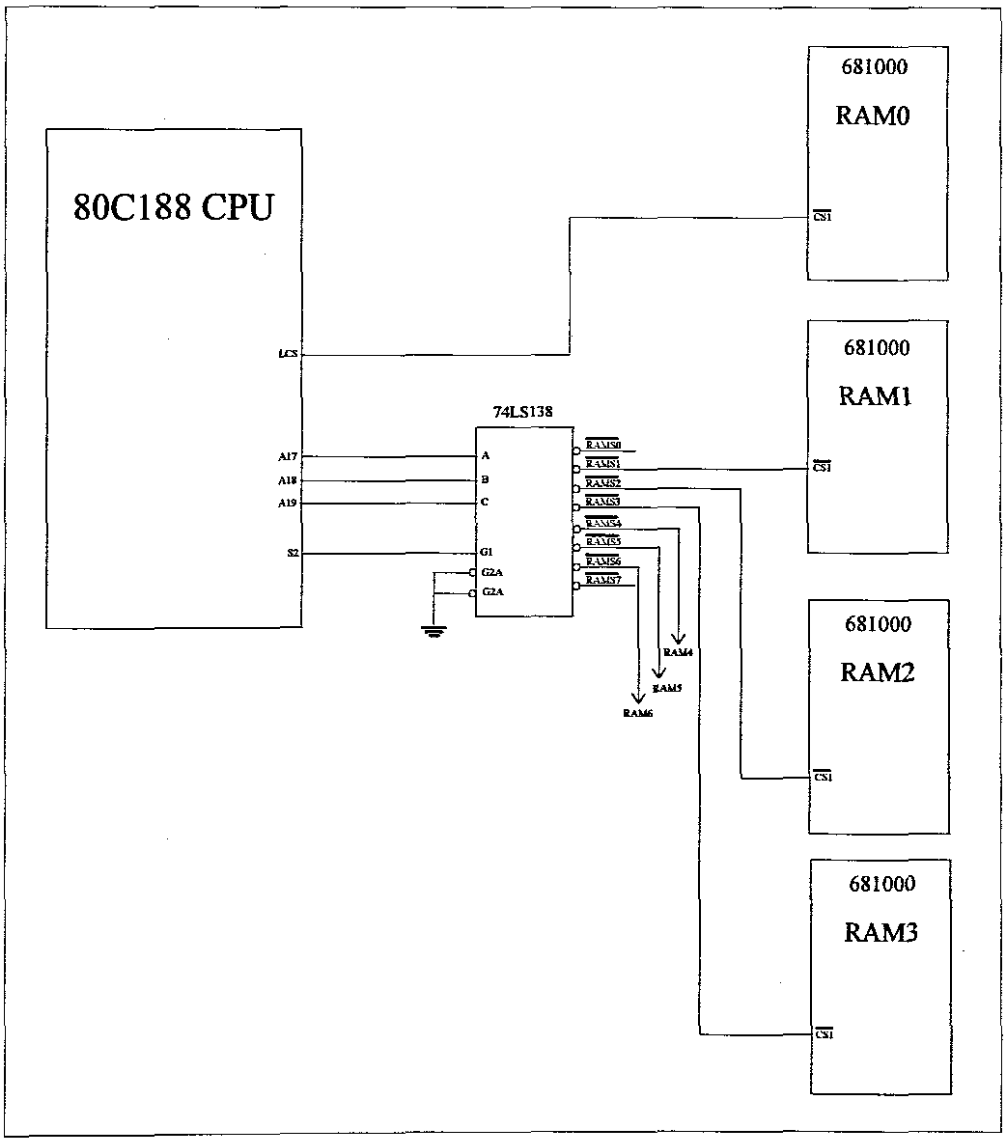


Figure 8-4 RAM Select logic after modification

8.3 Unable to receive image data above 2400 bps

Error free connections at speeds higher than 2400 bps could not be achieved due to excessive noise on the FAXMODEM card.

This problem was solved by separating the Digital and Analog ground on the board.

8.4 PC - FAXMODEM communication standards

During the design and development of this software research was done on the then existing communications protocol between Personal Computers and FAXCARDS. It was found that the majority of FAXCARDS were using the CLASS 1 protocol to communicate with the PC. The CLASS 2 protocol was only used by a few more expensive FAXCARDS. These protocols were during the development of this software still in draft form, being submitted to *STUDY GROUP 8* of the ITU-T for revision.

As discussed in section 4.1 *Communication protocol between PC and FAXCARD*, page 10 these protocols could however not be utilized for this project.

INTEL the manufacturer of the INTEL Satisfaction FAXCARD was using the Communication Application Specification (CAS) protocol to communicate with their FAXCARDS. During the development of this software the specification of the CAS protocol could not be obtained and was therefore not considered.

Due to the above reasons it was not feasible to write a software interface that was compatible with existing PC application software.

A device driver approach was therefore taken to simplify the interfacing between existing applications and the FAXMODEM card.

CHAPTER 9

Conclusion and Recommendations

The developed software for the FAXMODEM and PC interface enables the following:

- Initialization of receive and transmit sessions by the FAXMODEM. Transmit scheduling and adaptive answering is used respectively for initiating transmit and receive session.
- Reception and transmission of facsimile documents by the FAXMODEM. Handshaking and image data transmission are entirely handled by the FAXMODEM.
- Image data conversion from text, TIFF and QFX format to compressed facsimile modified Huffman and conversion from modified Huffman to TIFF and QFX formats by the FAXMODEM.
- Utilization of lower priority interrupts for background writing and reading of the PC hard drive and communication between PC and FAXMODEM. This was possible due to the non urgent nature of the data (not real time) being transferred between FAXMODEM and PC. These interrupts can therefore be interrupted while being executed.

Considering the above capabilities provided to the PC - FAXMODEM setup and the findings of this thesis, the following conclusions can be drawn:

- the software developed for the FAXMODEM card and PC conforms to all the specifications required in the objective at the beginning of this thesis,
- that facsimile transmission and reception are done independent of the PC. All facsimile protocols are handled by the FAXMODEM.
- the FAXMODEM uses the minimum amount of PC resources required for facsimile handling, i.e. only hard drive for mass storage and the screen for prompt information.

Suggestions on future software enhancements or improvements

- Optimizing the image data conversion algorithms for speed. A large amount of processing is required to convert between image data formats. Enhanced algorithms can significantly reduce the amount of time being used for image data conversions.
- Development of a software level for the PC to interface the FAXMODEM device driver with the proposed ITU Recommendation T.611. This recommendation is still in draft form and under revision of Study Group 8 of the telecommunication standardization sector of the ITU. Rec T.611 defines a Programming Communication Interface called "APPLI/COM" that provides unified access to Group 3, Group 4, telex and teletex services. The APPLI/COM image transfer format viz. TIFF is already supported by the FAXMODEM software.

Bibliography

Abel, Peter *IBM PC Assembler Language and Programming,* 1987.

Aldus Corporation / Microsoft Corporation *An Aldus / Microsoft Technical Memorandum 8/8/88 TIFF 5.0,* 1988.

Ammeraal, Leendert *C++ for programmers,* 1991

Angermeyer, John and Jaeger, Kevin *The Waite Group's MSDOS Developer's Guide,* 2nd ed. 1989

Borland International, Inc. *Borland C++, Programmers's Guide,* 1991.

Borland International, Inc. *Borland C++, Reference Guide,* 1991.

Borland International, Inc. *Turbo Assembler, Quick Reference Guide,* 1991.

Borland International, Inc. *Turbo Debugger, User's Guide,* 1991.

Campbell, Joe *C Programmers's Guide to Serial Communications,* 1987.

Cygnet Technologies Technical Services *The Spirit of TIFF class F*

Dettmann, Terry R. *DOS programmer's reference,* 1988

Graef, Gerald L. *Graphics Formats, A close look at GIF, TIFF and other attempts at a universal image format,* BYTE magazine, September 1989.

INTEL *Embedded Microcontrollers and Processors,* 1992.

Meadow, Anthony; Offner, Rocky and Budiansky, Michael *Handling Image Files with TIFF,* Dr. Dobb's Journal, May 1988.

Osborne, Adam and Kane, Gerry *OSBORNE 4 & 8 Microprocessor handbook,* 1981.

Pappas, Chris H. *Borland C++ handbook,* 3rd ed. 1992

Rockwell *9600 bps MONOFAX Modem Designers Guide,* Revision 1, October 1989.

Schildt, Herbert *C, the complete reference,* 2nd ed. 1990

Sedgewick, Robert *Algorithms in C,* 1990

Telecommunication Standardization Sector of the INTERNATIONAL TELECOMMUNICATION UNION *Current Draft of T.CLASS1,* Study Group 8 - Contribution 69 / December 1994.

Telecommunication Standardization Sector of the INTERNATIONAL TELECOMMUNICATION UNION *Draft revised Recommendation T.611 "Programming communication interface (PCI) APPLI/COM for facsimile Group 3, facsimile Group 4, teletex, telex, E-mail and file transfer services",* Study Group 8 - Report 32 / July 1994.

Telecommunication Standardization Sector of the INTERNATIONAL TELECOMMUNICATION UNION *Facsimile coding schemes and coding control functions for Group 4 facsimile apparatus, Recommendation T.6,* 1988.

Telecommunication Standardization Sector of the INTERNATIONAL TELECOMMUNICATION UNION *Procedures for document facsimile transmission in the general switched telephone network, Recommendation T.30,* Volume VII, Fascicle VII.3, 1988.

Telecommunication Standardization Sector of the INTERNATIONAL TELECOMMUNICATION UNION *Proposed draft Recommendation T.CLASS2,* Study Group 8 - Contribution 26 / September 1993.

Telecommunication Standardization Sector of the INTERNATIONAL TELECOMMUNICATION UNION *Standardization of Group 3 facsimile apparatus for document transmission, Recommendation T.4,* Volume VII. Fascicle VII.3, 1988.

The Waite Group *The Waite Groups's MSDOS Papers,* 1988.

Young, Michael J. *MS-DOS advanced programming,* 1988

APPENDIX

APPENDIX A

Implementation of ITU-T Standards

The Telecommunication Standardization sector of the International Telecommunication Union (ITU-T) formerly known as the CCITT, has issued Recommendations on procedures and signals to be used where facsimile equipment is operated over the general switched telephone network. This appendix describes the Recommendations applicable to facsimile transmission and how this was implemented with the design and development of the software.

A1.1 Recommendation V.21

This recommendation standardizes the use of 300 bits per second (bps) duplex modem for data transmission over the general switched telephone network. Rec V.21 uses Frequency Shift Keying (FSK). The FAXMODEM's digital signal processor (DSP) chip, the R96EFX, does include Rec V.21 modulation.

A1.2 Recommendation V.27*ter*

This recommendation standardizes the use of 2400 and 4800 bps modem over the general switched telephone network. Rec V.27*ter* uses Differential Phase Shift Keying (DPSK). The R96EFX does support Rec V.27*ter*.

A1.3 Recommendation V.29

This recommendation standardizes the use of 9600 bps duplex modem on point-to-point leased telephone-type circuits. Rec V.29 uses Quadrature Amplitude Modulation (QAM). The R96EFX does support this recommendation.

A1.4 Recommendation T.2

This recommendation standardizes the document transmission Group 1 type of facsimile apparatus. Group 1 facsimile apparatuses transmit an ISO A4 document in approximately six minutes. The R96EFX does not support the modulation that is used by Group 1 facsimile apparatus.

A1.5 Recommendation T.3

This recommendation standardizes the document transmission of Group 2 facsimile apparatus. Group 2 facsimile apparatuses transmit an ISO A4 document in approximately three minutes Group 2 facsimile apparatus use an amplitude modulation (AM) single sideband (SSB) method to transfer image data. The R96EFX does support Group 2 modulation. However the designed software does not support this method of operation.

A1.6 Recommendation T.4

This recommendation standardizes the document transmission of Group 3 type of facsimile apparatus. Group 3 facsimile apparatuses transmit an ISO A4 document in approximately one minute. Group 3 facsimile apparatuses use high speed synchronous data transfer methods viz. Rec V.29 and Rec V.27*ter*. Control messages use the Rec V.21 method. Optionally Rec V.27 2400 bps can be used for control signals. The basic coding scheme is Modified Huffman one-dimensional encoding.

A1.6.1 Dimensions of apparatus

The FAXMODEM software supports the following dimensions:

1. A standard vertical resolution of 3.85 line per mm (98 dpi) and a high resolution of 7.7 line/mm (196 dpi) in both receive and transmit modes.

2. An image width of 1728 picture elements (pixels) along standard scan line length of 215 mm (ISO A4 / 204 dpi) in both transmit and receive modes.

3. Image width of 864 pixels along a scan line length of 107 mm (ISO A5 / 204 dpi) in receive mode.

4. An image width of 1216 pixels along a scan line length of 151 mm (ISO A6 / 204 dpi) when receiving documents.

5. An image width of 2048 pixels along a scan line length of 255 mm (ISO B4 / 204 dpi) when receiving documents.

6. Image widths of 2432 pixels along a scan line length of 303 mm (ISO A3 / 204 dpi) when receiving documents.

A1.6.2 Transmission time per total coded scan line

The FAXMODEM software supports a minimum scan line time of 0 millisecond with a fall-back to 5 ms, 10 ms, 20 ms and 40 ms. The maximum scan line time is set to 5 seconds.

A1.6.3 Coding scheme

The one-dimensional run length coding scheme is used to transmit all image data. Other optional coding schemes viz. two-dimensional and extended two-dimensional are not supported by the FAXMODEM software.

A1.6.4 Optional modes

Other optional modes of image data transfer described in Rec T.4 viz. error correction mode and error limiting mode, are not supported by the FAXMODEM software.

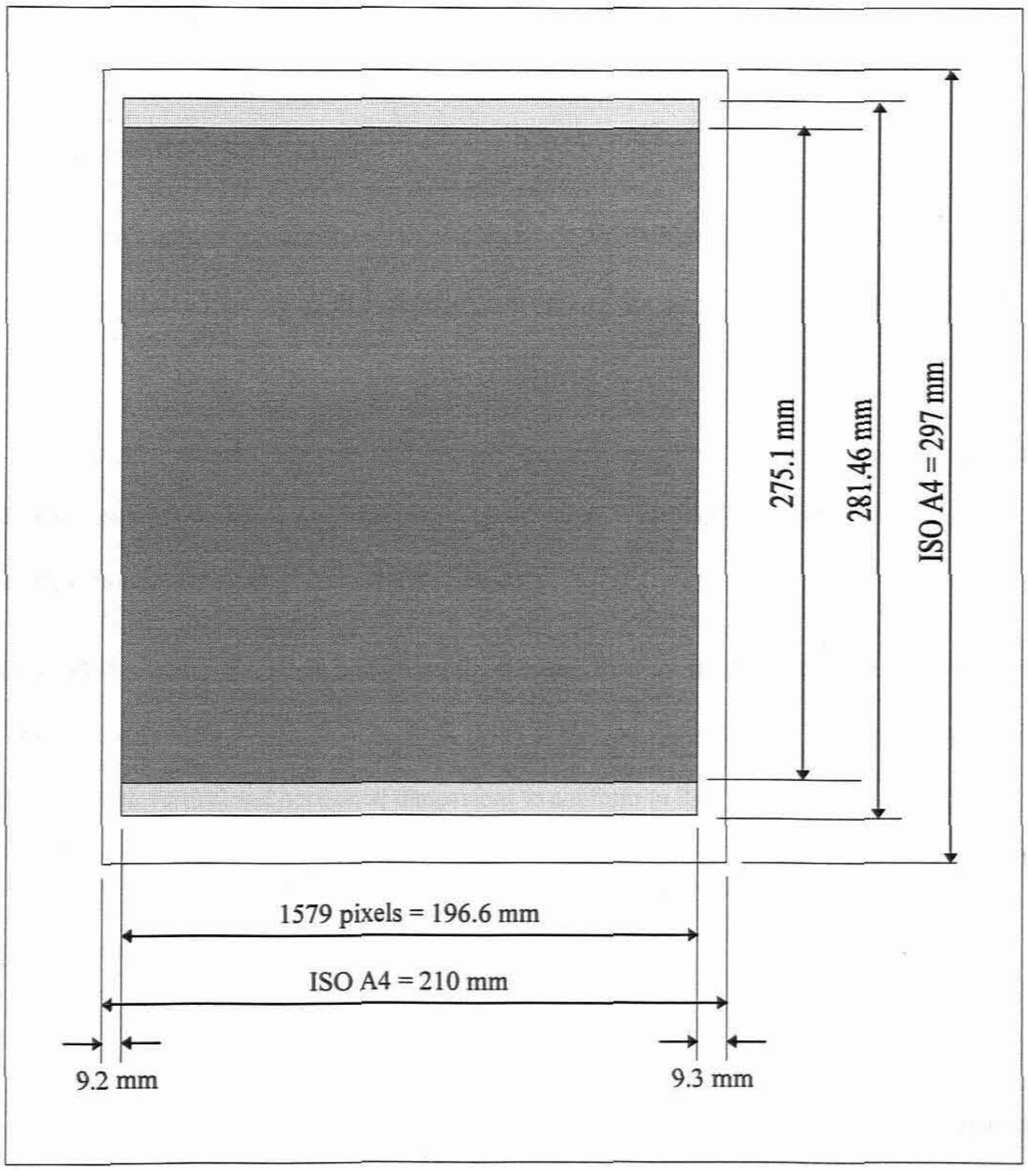


Figure A-1 Guaranteed reproducible area on a standard ISO A4 paper size

A1.6.5 Guaranteed reproducible area

Recommendation T.4 indicates a guaranteed reproducible area for Group 3 apparatus of 196.6 mm wide and 281.46 mm high on a standard ISO A4 paper size. The FAXMODEM software converts text documents to compressed Rec T.4 pages with the following dimensions:

- Left margin of 9.2 mm and right margin of 9.3 mm. These margins are added to cater for printer, enlarging, skew and positioning errors when the document is reproduced on the remote facsimile equipment.

- The minimum size of the top margin is set to 8.3 mm to cater for paper insertion loss. The minimum size of the bottom margin is set to 9.9 mm to cater for skew, scanning density tolerance and gripping losses.

The reproducible area of text documents converted by the FAXMODEM software is therefore equal to 196.6 mm wide and 275.1 mm high. This dimension falls within the guaranteed reproducible area described in Rec T.4.

The FAXMODEM software converts TIFF images to compressed Rec T.4 pages without adding any of the above mentioned margins. The following steps are taken if the dimensions of a TIFF image are not equal to the required dimensions of a Rec T.4 page:

- If the width of the TIFF image is less than the standard Rec T.4 image width of 215 mm, the TIFF image is horizontally centered on the page. If the TIFF image is wider, the TIFF image is scaled down in both vertical and horizontal dimensions to conform to Rec T.4.
- If the height of the TIFF image is more than the required height of 297 mm, the TIFF image is scaled down in both dimensions.

A1.7 Recommendation T.6

This recommendation standardizes the document transmission of Group 4 type of facsimile apparatus. Group 4 facsimile apparatuses use the public data network (PDN) as a communication medium. The basic coding scheme is in principle the same as the two-dimensional coding scheme of Rec T.4. Group 4 facsimile also supports gray scale and colour images. Rec T.6 was not implemented.

A1.8 Recommendation T.30

This recommendation standardizes the design, operation and procedures of attended or unattended Group 3 type of facsimile apparatus over the general switched public network. This includes the following:

- call establishment and call release
- compatibility checking, status and control commands

- checking and supervision of line conditions
- control function and facsimile operator recall.

The operating methods supported by the FAXMODEM software are described by five separate and consecutive phases:

- Phase A Call setup
- Phase B Pre-message procedures
- Phase C Message transmission
- Phase D Post message procedure
- Phase E Call release

A1.8.1 Phase A - call establishment

The software of the FAXMODEM was designed to support the following operating methods:

1. *Automatic to manual*

   Automatic operation at the calling station and manual operation at the called station.

2. *Automatic to automatic*

   Automatic operation at both calling and called stations.

A1.8.2 Phase B - Pre-message procedure

This procedure consists of identification of capabilities, commanding of chosen conditions and confirmation of acceptable conditions. Phase B procedures can be performed in two different ways viz. tonally, for simple procedures and binary coded, for more comprehensive procedures. The FAXMODEM software supports only binary coded procedures as a large amount of functions is performed by the card automatically. The automatic functions performed in Phase B includes the following:

- Identification section
  - group identification
  - confirmation for reception
  - subscriber identification
- Command section
  - group command
  - training
  - subscriber identification
  - polling command
  - echo suppresser disabling

A1.8.3 Phase C1 - In-message procedure

The FAXMODEM software performs error detection and line supervision during message reception.

A1.8.4 Phase C2 - Message transmission

Image data is transmitted or received according Rec T.4.

A1.8.5 Phase D - Post-message procedure

The FAXMODEM uses binary coded signaling to perform post-message procedures. When receiving, the quality of the received facsimile page is first checked by the FAXMODEM before any post-message signals are issued. Post-message procedures include the following:

- end-of-message signaling
- confirmation signaling
- multiple page signaling
- end-of-facsimile procedure signaling.

A1.8.6 Phase E - Call release

The FAXMODEM performs automatic call release. If the FAXMODEM is the commanding station it sends a disconnect signal before releasing the call.

A1.8.7 Optional procedures

The optional procedure for Group 3 facsimile transmission incorporating error correction is not supported by the FAXMODEM software.

APPENDIX B

Data Structures of the FAXMODEM

This appendix describes all the important data structures used in the operating system of the FAXMODEM. The layout of the data structures is given in MASM format.

B1.1 Dynamic memory Allocation

The FAXMODEM memory is divided into five regions viz. Variables, Code, Data and FAR HEAP. The Variables segment contains the Stack and the NEAR HEAP. The NEAR HEAP and FAR HEAP are used to allocate memory for dynamic data structures.

B1.1.1 The NEAR HEAP

The NEAR HEAP is a linked list of NEAR HEAP ITEMS that reside just below the system variables in memory. The NEAR HEAP is used to store small data structures. Below is the structure of a NEAR HEAP item.

```
NEARHEAP_ITEM     STRUC
NHI_ID            DB     'N'    ; indicates near HEAP item
NHI_EMPTY         DB     ?      ; FF = empty
NHI_NEXT          DW     ?      ; pointer to next HEAP item
NHI_PREV          DW     ?      ; pointer to previous HEAP item
NHI_DATA          LABEL BYTE    ; start of HEAP item data
NEARHEAP_ITEM     ENDS
```

B1.1.2 The FAR HEAP

The FAR HEAP is a linked list of FAR HEAP ITEMS. The FAR HEAP extends from below the NEAR HEAP to the top of memory. Below is the structure of a FAR HEAP ITEM.

```
FARHEAP_ITEM   STRUC
FHI_ID                DB      'F'     ; indicates FAR HEAP item
FHI_EMPTY             DB      ?       ; FF = empty
FHI_NEXT              DD      ?       ;  pointer to next HEAP item
FHI_PREV              DD      ?       ;  pointer to previous HEAP item
FHI_DATA              LABEL  BYTE     ; start of HEAP item data
FARHEAP_ITEM   ENDS
```

B1.2 Fax Files

B1.2.1 Static fax files structures

All static fax file structures are located in the Variables segment.

B1.2.1.1 The FILE Directory

The fax file directory is a list of near pointers to fax headers. The length of the directory puts a limit to the number of fax files that can be stored in memory.

B1.2.1.2 The JOB Table

The job table is a list of near pointers to FAXJOB structures. The length of the JOB Table determines the maximum amount of HANDLES that can be issued by the operating system. A HANDLE is merely used to index the JOB Table.

B1.2.2 Dynamic fax file structures

All dynamic Fax File Structures except MEMBLOCKS are created on the NEAR HEAP.

B1.2.2.1 The Fax File Header

A Fax file consists of one or more fax pages. The fax file header contains data relevant to the whole fax document. Below is the structure of a fax file HEADER

```
FAXHEAD           STRUC
FH_NAME           DB    MAXNAME DUP(?)     ; file name
FH_TIME           DD    ?                  ; time created
FH_MAXRECWIDTH    DW    ?                  ; recording width of widest page
FH_MAXLINECOUNT   DW    ?                  ; line count of longest page
FH_MAXRES         DB    ?                  ; highest resolution used
FH_MAXCODING      DB    ?                  ; FF if two dimensional coding used
FH_PAGECNT        DB    ?                  ; number of pages
FH_WRFLAG         DB    ?                  ; 0 = file not open for write
FH_RDFLAG         DB    ?                  ; 0 = file not open for read
FH_PAGEINFO       DW    ?                  ; pointer to 1st pageInfo
FAXHEAD      ENDS
```

B1.2.2.2 The Fax Page Header

A fax file page consists of one or more memory blocks. Each fax page has a page header which contains all data relevant for that page. Below is the structure of a page header.

```
pageInfo          STRUC
PI_NEXT           DW    ?                  ; pointer to next pageInfo
PI_LINECOUNT      DW    ?                  ; line count for this page
PI_RECWIDTH       DW    ?                  ; recording width for this page
PI_CODING         DB    ?                  ; coding method for this page
PI_RES            DB    ?                  ; vertical resolution for this page
PI_DATA           DD    ?                  ; pointer to 1st 16k MEMBLOCK
PI_NUMBER         DB    ?                  ; page number
PI_WRFLAG         DB    ?                  ; 0 = not writing to this PAGE
PI_RDFLAG         DB    ?                  ; 0 = not reading this PAGE
pageInfo          ENDS
```

B1.2.2.3 The Fax Page Memory Block

A fax page memory block (MEMBLOCK) contains image data of a fax document. A memory block has a fixed length and the last memory block of fax page is usually not filled. The use of memory blocks enables the operating system to allocate and de-allocate memory randomly. MEMBLOCK structures are created on the FAR HEAP. Below is the structure of a fax page memory block.

```
MEMBLOCK     STRUC
MB_NEXT      DD             ?      ; pointer to next block of data
MB_SIZE      DW             ?      ; size of this block / offset when writing
MB_NUMBER    DB             ?      ; block number
MB_WRFLAG    DB             ?      ; 0 = not writing to this block
MB_RDFLAG    DB             ?      ; 0 = not reading this block
MB_START     DB     MEM_BLOCKSIZE DUP(?)
MEMBLOCK     ENDS
```

B1.2.2.4 The Fax Job Structure

Whenever a fax file is opened, a FAXJOB is created on the NEAR HEAP. An integer value which is used as an index into the JOB Table is returned. This integer value is called a HANDLE. Below is the structure of a FAXJOB.

```
FAXJOB       STRUC
FJ_HEAD      DW    ?     ; pointer to FAXHEAD
FJ_PAGE      DW    ?     ; pointer to current page
FJ_BLOCK     DD    ?      ; pointer to current MEMBLOCK
FJ_BLOCKPNT  DW    ?     ; read pointer for current MEMBLOCK
FJ_TYPE      DB    ?     ; type of JOB (read/write)
FAXJOB       ENDS
```

B1.3 Transmit scheduling

B1.3.1 Static schedule structures

All static schedule structures are located in the Variables segment.

B1.3.1.1 The Fax Schedule List

The Fax Schedule List is an array of SCHEDULE_ITEM pointers. The length of this list indicates the number of transmit sessions pending.

B1.3.1.2 The Job Schedule List

The Job Schedule List is an array of SCHEDULE_JOB pointers. The length of this list is therefore an indication of the number of transmission sessions currently in progress.

B1.3.2 Dynamic schedule structures

B1.3.2.1 The SCHEDULE_ITEM Structure

The SCHEDULE_ITEM structure holds the binary equivalent of one line of an ASCII schedule file. A SCHEDULE_ITEM is converted to a SCHEDULE_JOB when its send time is reached.

```
SCHEDULEM_ITEM        STRUC
SM_TIME               DD    ?     ; date & time  of activation
SM_RETRY              DW    ?     ; retry count
SM_FILE               DW    ?     ; ptr to filename
SM_TELNO              DW    ?     ; ptr to tel no
SM_FILESHANDLE        DW    ?     ; HANDLE if a list of files
SM_TELNOSHANDLE       DW    ?     ; HANDLE if a list of telno's
SM_FILETYPE           DB    ?     ; 00: SM_FILES = ptr to a filename of a faxfile,
                                    FF: SM_FILES = ptr to a filename containing a
                                    of list of faxfile names

SM_TELNOTYPE          DB    ?     ; 00: SM_TELNOS = ptr to a telno
                                    FF: SM_TELNOS = ptr to a filename
                                    containing a list of TELNOs

SCHEDULEM_ITEM  ENDS
```

B1.3.2.2 The SCHEDULE_JOB Structure

This structure stores the data associated with a transmit session.

```
SCHEDULE_JOB        STRUC
SJ_COMPLETE         DB    ?      ; FF = schedule job can be deleted
SJ_RESULTCNT        DB    ?      ; size of result list(number of items)
SJ_RESULTS          DW    ?      ; ptr to result list
SJ_TELNOCNT         DW    ?      ; telno count
SJ_TELNOLIST        DW    ?      ; ptr to list of telno's
SJ_FILECNT          DW    ?      ; file count
SJ_FILELIST         DW    ?      ; ptr to list of filenames
SCHEDULE_JOB        ENDS
```

B1.3.2.3 The RESULT_ITEM structure

This structure is used to store the result of a transmit session. When creating a SCHEDULE_JOB, one RESULT_ITEM structure is created for every telephone number that has to be dialed.

```
RESULT_ITEM     STRUC
RI_VALUE        DB    ?      ; result
RI_RETRYCNT     DB    ?      ; retry down counter
RI_TIME         DD    ?      ; execute time
RESULT_ITEM     ENDS
```

APPENDIX C

R96EFX Programmable Tone Detectors

The R96EFX DSP incorporates three programmable tone detectors. On power up the tone detectors are centered around the frequencies listed in Table C-1.

Table C-1 Default Tone Detector frequencies

	R96EFX Tone Detection Flag	Frequency
Tone Detector 1	**FR1**	**2100 Hz**
Tone Detector 2	**FR2**	**1100 Hz**
Tone Detector 3	**FR3**	**462 Hz**

The tone detectors can be programmed to detect signals from 400 Hz to 3 kHz. A tone detector consists of three filters in cascade that are programmed as follows:

- Filter 1. This filter is programmed to a frequency just below the center frequency $(f_O - f_A)$.
- Filter 2. This filter is programmed to a frequency just above the center frequency $(f_O + f_A')$.
- Filter 3. This is an energy averaging filter and is programmed with the desired response time of the tone detector.

Figure C-1 depicts the offset frequencies and the overall response of a tone detector.

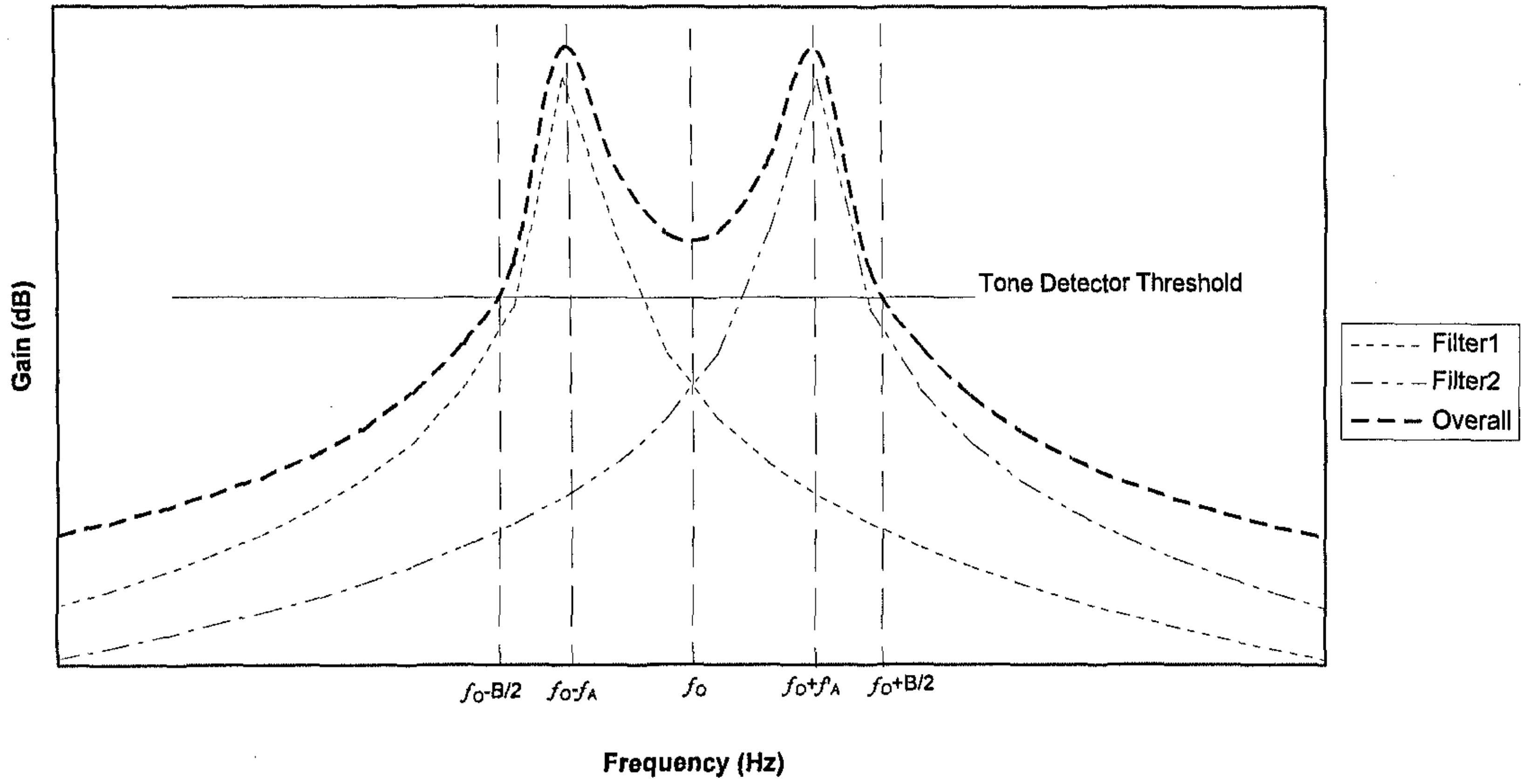


Figure C-1 Tone detector frequencies

Equation C-1 and Equation C-2 are used to calculate the center frequency offsets.

Equation C-1

$f'_A \cong 0.72 \times \frac{B}{2}$ Calculates the frequency offset above center frequency ≈ 72% of half the bandwidth.

Equation C-2

$f_A = 0.99 \times f'_A$ Calculates frequency offset below center frequency = 99% of f_A.

Equation C-3 to Equation C-5 is the transfer functions for a tone detector

Equation C-3

$$H_1(Z) = \frac{2(\alpha_0 + \alpha_1 Z^{-1} + \alpha_2 Z^{-2})}{1 - 2\beta_1 Z^{-1} - 2\beta_2 Z^{-2}}$$ Transfer function for first filter

Equation C-4

$$H_2(Z) = \frac{2(\alpha'_0 + \alpha'_1 Z^{-1} + \alpha'_2 Z^{-2})}{1 - 2\beta'_1 Z^{-1} - 2\beta'_2 Z^{-2}}$$ Transfer function for second filter

Equation C-5

$$H_3(Z) = \frac{\alpha''}{1 - \beta'' Z^{-1}}$$ Transfer function for energy averaging filter

The following equations calculate the coefficients for tone detector tuning.

Equation C-6

$$\beta_1 = r \times \cos[360° \times \frac{f_O - f_A}{f_S}]$$

Equation C-7

$$\beta'_1 = \cos[360° \times \frac{f_O + f'_A}{f_S}]$$

Equation C-8

$$\beta_2 = \beta_2' = -\frac{r^2}{2}$$

Equation C-9

$$\alpha_0 = \alpha_0' = \frac{\frac{104}{319} \times f_O - 78.62}{32767} \qquad 400 \le f_O \le 1100 \text{ Hz}$$

Equation C-10

$$\alpha_0 = \alpha_0' = \frac{\frac{44}{275} \times f_O - 104}{32767} \qquad 1100 \le f_O \le 1650 \text{ Hz}$$

Equation C-11

$$\alpha_0 = \alpha_0' = \frac{\frac{4}{45} \times f_O - 221}{32767} \qquad 1650 \le f_O \le 3000 \text{ Hz}$$

Equation C-12

$$\alpha_1 = \alpha_2 = \alpha_1' = \alpha_2' = 0 \qquad \text{set to default value} = 0$$

Equation C-13

$$\alpha'' = \frac{1}{1 + f_S \times T}$$

Equation C-14

$$\beta'' = \frac{1}{1 + \frac{1}{f_S \times T}}$$

Where:

f_O = center frequency

f_S = sample frequency = 9600 Hz for Group3

$r = 0.994030884$

$B =$ Bandwidth $=$ typically 50 Hz

$T =$ response time of tone detector, default $= 0.1$ seconds

$\alpha_0, \alpha_1, \alpha_2, \beta_1, \beta_2 =$ Coefficients for filter 1

$\alpha'_0, \alpha'_1, \alpha'_2, \beta'_1, \beta'_2 =$ Coefficients for filter 2

$\alpha'', \beta'' =$ Coefficients for filter 3

APPENDIX D

Facsimile Command Codes

The facsimile command code is sent in the facsimile control field (FCF) of the HDLC signal. Where the first bit of the FCF = x, it is set to 1 by the station which receives a valid DIS signal and is set to 0 by the station which receives a valid response to a DIS signal. It will remain unchanged until the station again enters the beginning of phase B.

Initial identification

1. DIS	=	10000000	Digital identification signal
2. CSI	=	01000000	Called subscriber identification
3. NSF	=	00100000	Non-standard facilities

Command to send

1. DTC	=	10000001	Digital transmit command
2. CIG	=	01000001	Calling subscriber identification
3. NSC	=	00100001	Non-standard facilities command

Command to receive

1. DCS	=	1000001x	Digital command signal
2. TSI	=	0100001x	Transmitting subscriber identification
3. NSS	=	0010001x	Non-standard set-up
4. CTC	=	0001001x	Continue to correct

Pre-message response signals

1. CFR	=	1000010x	Confirmation to receive
2. FTT	=	0100010x	Failure to train
3. CTR	=	1100010x	Response for Continue To Correct

Post Message Commands

1. EOM	=	1000111x	End of message
2. MPS	=	0100111x	Multi-page signal
3. EOP	=	0010111x	End of procedure
4. PRI_EOM	=	1001111x	Procedure interrupt - EOM
5. PRI_MPS	=	0101111x	Procedure interrupt - MPS
6. PRI_EOP	=	0011111x	Procedure interrupt - EOP
7. PPS	=	1011111x	Partial page signal
8. EOR	=	1100111x	End of retransmission
9. RR	=	0110111x	Receive ready ?

Post-message responses

1. MCF	=	1000110x	Message confirmation
2. RTN	=	0100110x	Retrain negative
3. RTP	=	1100110x	Retrain positive
4. PIN	=	0010110x	Procedural interrupt negative
5. PIP	=	1010110x	Procedural interrupt positive
6. RNR	=	1110110x	Receive not ready
7. ERR	=	0001110x	Response for end of retransmission

8. PPR	=	*1011110x*	Partial page request

Other line control signals

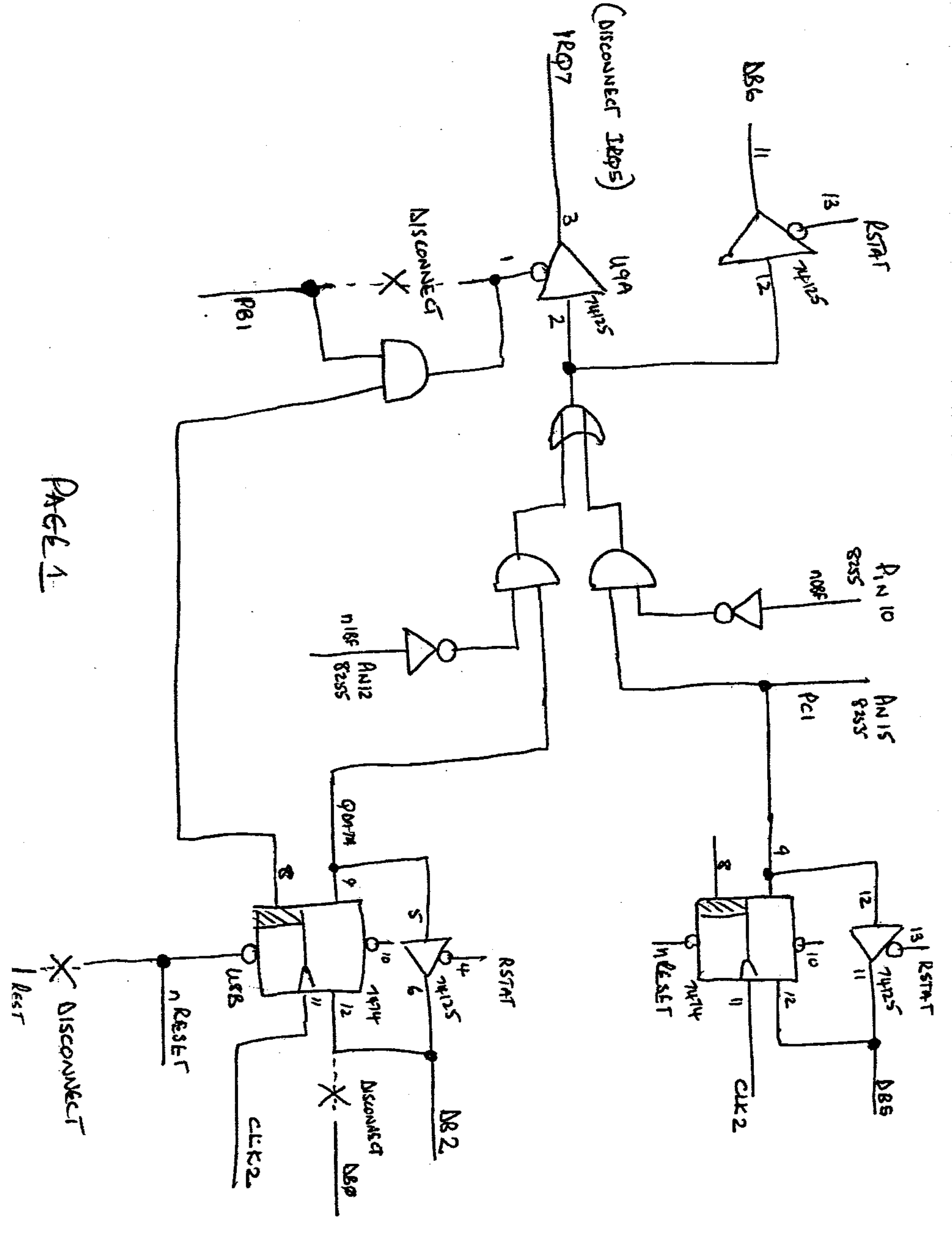
1. CRP	=	0001101x	Command repeat
2. DCN	=	1111101x	Disconnect
3. FCD	=	00000110	Facsimile coded data
4. RCP	=	10000110	Return to control for partial page

APPENDIX E

Example Facsimile Page

The following page was received by the FAXMODEM and stored on the PC's hard disk as a TIFF file.

It was then imported into a word processor and printed on a laser printer.



re E-1 The result of a test transmission

APPENDIX F

Glossary of TERMS and ABBREVIATIONS

American National Standards Institute (ANSI) This is the US member body of ISO and consists of members from computer manufacturers and users in the United States.

***Consultative Committee for International Telegraph and Telephone* (CCITT)** This was an international standards organization that included the Postal, Telegraph and Telephone authorities of the member countries. The CCITT ceased to exist as of 28 February 1993 and was replaced by the ITU Telecommunication Standardization Sector (ITU-T).

Electronics Industries Association (EIA) This is a member of ANSI and consists of members from the electronics industry in the United States.

High-level data link control (HDLC) This is an internationally agreed standard protocol controlling data exchange across point-point or *multidrop data links*.

ITU Telecommunication Standardization Sector (ITU-T) This a permanent organ of the International Telecommunication Union and is responsible for the standardization of telecommunication on a worldwide basis.

Protocol This is a set of rules formulated to govern the exchange of data between communicating parties.

Public-switched telephone network (PSTN) This term is used to describe the telephone network.

Training sequence The training sequence is a particular sequence of signals send by the transmitter before sending data. This sequence of signals allows the receiver to establish carrier detection, adjust automatic gain circuits, establish timing synchronization and converge the adaptive equalizer to the *initial line conditions*