POWER LINE
MODEMS & APPLICATIONS

1st EDITION

SEPTEMBER 1994

USE IN LIFE SUPPORT DEVICES OR SYSTEMS MUST BE EXPRESSLY AUTHORIZED

SGS-THOMSON PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF SGS-THOMSON Microelectronics. As used herein:

1. Life support devices or systems are those which (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided with the product, can be reasonably expected to result in significant injury to the user.

2. A critical component is any component of a life support device or system whose failure to perform can reasonably be expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

TABLE OF CONTENTS

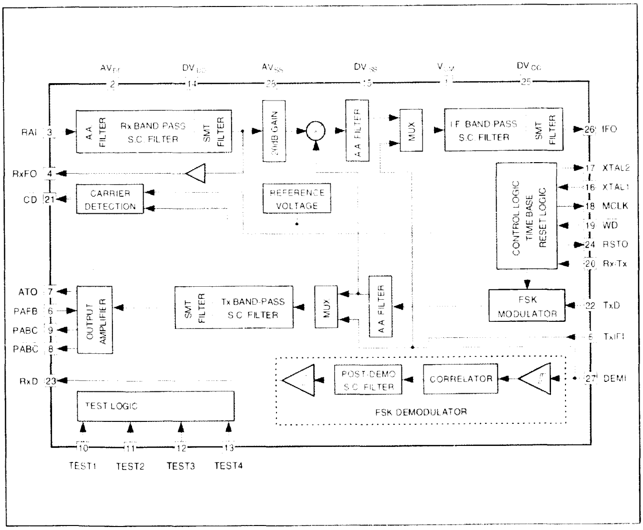
Dedicated Modem Chip Targets Home Automation Systems

Sending data over 220V/110V power lines, the ST7537 Home Automation modem lets appliances communicate without extra cabling.

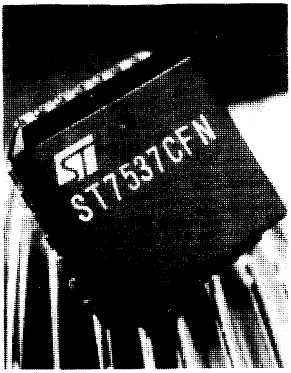
by Joël Huloux
& Jérôme Gilbert

In the latest generation of home automation systems, appliances can exchange information by transmitting data over the domestic mains wiring. As a result there is no need to install extra control cables and appliances can be connected to the "network" simply by plugging them into the nearest wall socket. Apart from the obvious saving in installation cost, this virtual network also makes modification and enhancement very simple since new devices just have to be plugged into a wall socket to be instantly connected to the network.

What makes these systems feasible is a new dedicated modem integrated circuit, the SGS-THOMSON ST7537 Home Automation Modem IC, developed specifically for this new high volume consumer market as part of a European Community "ESPRIT" project on domestic automation.



Inside the ST7537 are all of the basic functions needed in a power line modem, plus carrier detection and watchdog functions. The only external components needed are a transformer and a simple line driver.



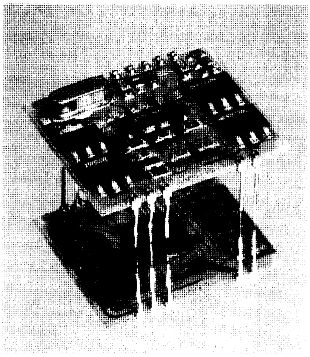
Designed specifically for home automation applications, the SGS-THOMSON Microelectronics ST7537 Home Automation Modem IC sends data at 1200bps over 220/110V power lines using a 132.45kHz carrier, this eliminating the need for extra cabling. A PLCC28 package is used.

THE MODEM CHIP

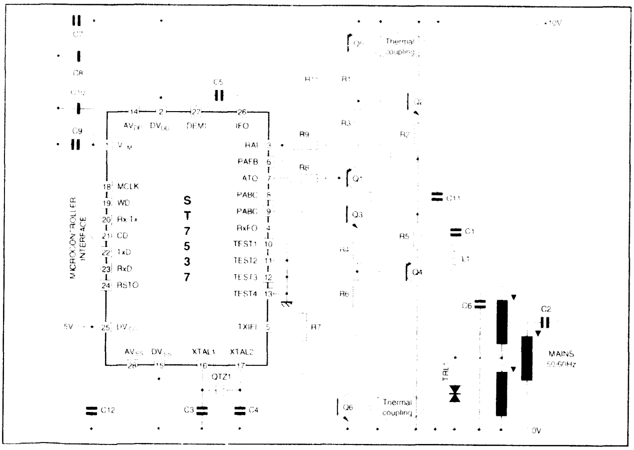
Fabricated in analog CMOS technology, the ST7537 sends and receives data at 1200bps in half duplex mode using a carrier frequency of 132.45kHz, complying with Europe's CENELEC ES 50065 standard (which specifies the use of band 125kHz to 140kHz carrier frequencies for home automation) and US FCC regulations.

Frequency-shift keying is used for transmission, a fundamental design choice that makes it possible to achieve rugged transmission in a very noisy electrical environment at an affordable cost for high-volume consumer markets. Among the alternatives, amplitude- shift keying is too susceptible to noise and Spread Spectrum, though theoretically more reliable, requires complex and costly circuits. Moreover, field trials in a critical remote utility meter reading application have proven the dependability of the SGS-THOMSON approach.

Included on the chip are all of the functional blocks necessary for the transmission and reception of data over power lines. In addition to this IC the only external components needed are a line driver and a transformer, plus, of course, the microcontroller that prepares and interprets message data. The photo shows a complete module made by Landis & Gyr, the schematic of which is given below. In this module the ST7537 die is mounted directly on the under side of the board without a conventional package, though the device is also supplied in a PLCC28 package.


This compact modem module designed by Landis & Gyr mounts the ST7537 die directly on the underside of the board. Surface mounting transistors are used to make the line driver.

Transmit data enters the FSK modulator asynchronously with a nominal intra-message data rate of 1200bps. Inside the modulator the data is transformed into two frequencies (133.05kHz for a "0" and 131.85kHz for a "1"), derived from an inexpensive 11.0592 MHz crystal.



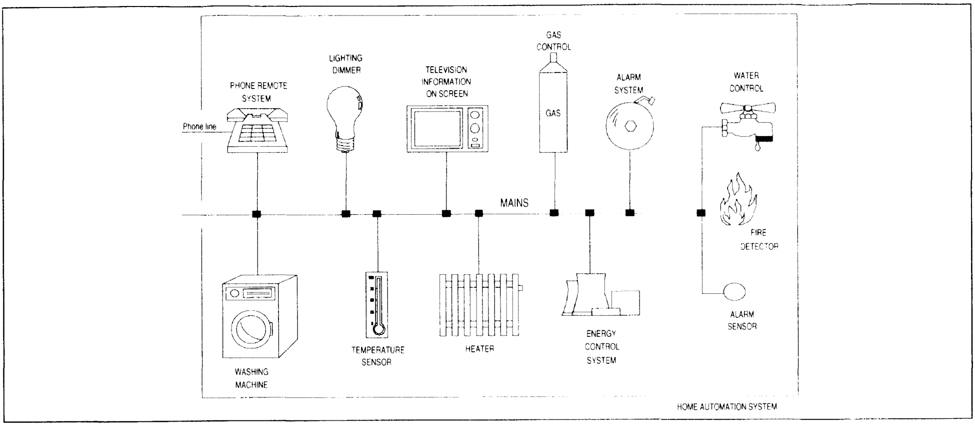
Schematic of the application board shown in the photo above.

The modulated signal from the FSK modulator is filtered by a switched-capacitor bandpass filter (TX bandpass) to limit the output spectrum and to reduce the level of harmonic components. The final stage of the transmit path consists of an operational amplifier which needs a feedback signal from the power amplifier.

In the receive section, the incoming signal is applied at the RAI input (with a sensitivity of 10mV) where it is first filtered by a switched-capacitor bandpass filter with a pass band of around 12kHz, centered on the carrier frequency. The output of the filter is amplified by a 20dB gain stage which provides symmetrical limitation for overvoltages. The resulting signal is downconverted by a mixer which receives a local oscillator synthesized by the FSK modulator block.

Finally, an intermediate frequency bandpass filter whose central frequency is 5.4kHz improves the signal-to-noise ratio before entering the FSK demodulator. The coupling of the intermediate frequency filter output to the FSK demodulator input is made by an external capacitor which cancels the receive path offset.

In the ST7537 there are two important additional functions: the carrier detector and the watchdog. Carrier detection is needed because in practically all applications more than two appliances will be connected to the power line. Before attempting to transmit an appliance must first check that there is no carrier present, and



In a typical home each appliance or device will have an ST7537 module and communicate with other devices using the power cable.

if there is, it must wait and retry later.

The watchdog function is provided to ensure that the modem's control micro is functioning correctly. Software in the micro must include instructions that send a pulse to the watchdog input of the ST7537 at least once every 1.5 seconds. If no negative transition is observed at this input for 1.5s a reset signal is generated to restart the micro. This watchdog monitor scheme ensures that any disruptions caused by glitches are quickly corrected.

During full operation the ST7537 dissipates less than 300mW. Supplies of 10V and 5V are used.

THE HOME SYSTEM PROJECT

SGS-THOMSON Microelectronics is a member of an ESPRIT project on home automation, together with EDF (Electricité de France, the French electricity authority), Landis & Gyr, Daimler Benz, Merlin Gerin and Thomson Consumer Electronics. These industrial partners have built a platform for development tools that work with a common protocol, called Home System (HS).

One of the partners, Landis & Gyr, already uses the new ST7537 modem chip in a new family of home automation products. Thanks to products like these we enter a new era of appliances that are not only smart, but they communicate with each other, too. A typical household scenario is shown above, where various appliances, sensors, utility controls, a telephone interface and a TV screen display are all connected to the power line using Landis & Gyr modules.

If this automated house catches fire the appropriate detector will send a warning message over the power line. This will be picked up by the gas control which can cut off the gas supply, by an alarm system that can alert anyone in the house, and even by the telephone interface that can call the emergency services.

The telephone interface also allows the householder to give instructions to appliances from outside. You might, for example, phone home and tell the air conditioner to precool certain rooms at a specified time.

Where there is a limit on energy consumption, or where demand energy pricing is used (now that the technology is available this is likely to be applied extensively in future) various appliances can negotiate power requirements through an energy control system. For example, a washing machine can agree with the heating system when it can start a cycle to avoid sudden and unnecessary peaks of demand.

The cover illustration appeared first in the magazine Elektronik Informationen.

GENERAL INDEX

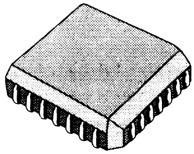
ST7536 DATASHEET

SGS-THOMSON
MICROELECTRONICS

ST7536

POWER LINE MODEM

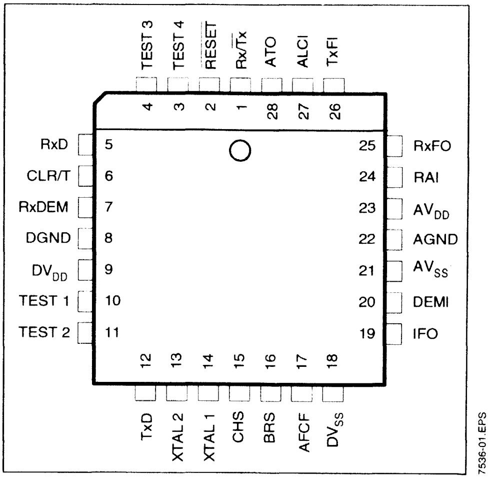
- HALF DUPLEX SYNCHRONOUS FSK MODEM
  - Two programmable channels for 600bps data rate
  - Two programmable channels for 1200bps data rate
- AUTOMATICALLY TUNED Rx AND Tx FILTERS
- TX CARRIER FREQUENCIES SYNTHESIZED FROM EXTERNAL CRYSTAL
- LOW DISTORTION Tx SIGNAL (S/H2 ≥ 50dB)
- AUTOMATIC LEVEL CONTROL ON Tx SIGNAL
- Rx SENSITIVITY : 2mV$_{RMS}$ (600bps)
  3mV$_{RMS}$ (1200bps)
- Rx CLOCK RECOVERY
- POWER-DOWN MODE
- SUITABLE TO APPLICATION IN ACCORDANCE WITH DH028/29 ENEL, EN50065-1 CENELEC AND FCC SPECIFICATIONS



PLCC28
(Plastic Package)

ORDER CODE : ST7536CFN

PIN CONNECTIONS



DESCRIPTION

The ST7536 is a half duplex synchronous FSK MODEM designed for power line communication network applications.

It operates from a dual power supply +5V and -5V, and requires an external interface for the coupling to the power line. It offers two programmable data rate with two programmable channels each.

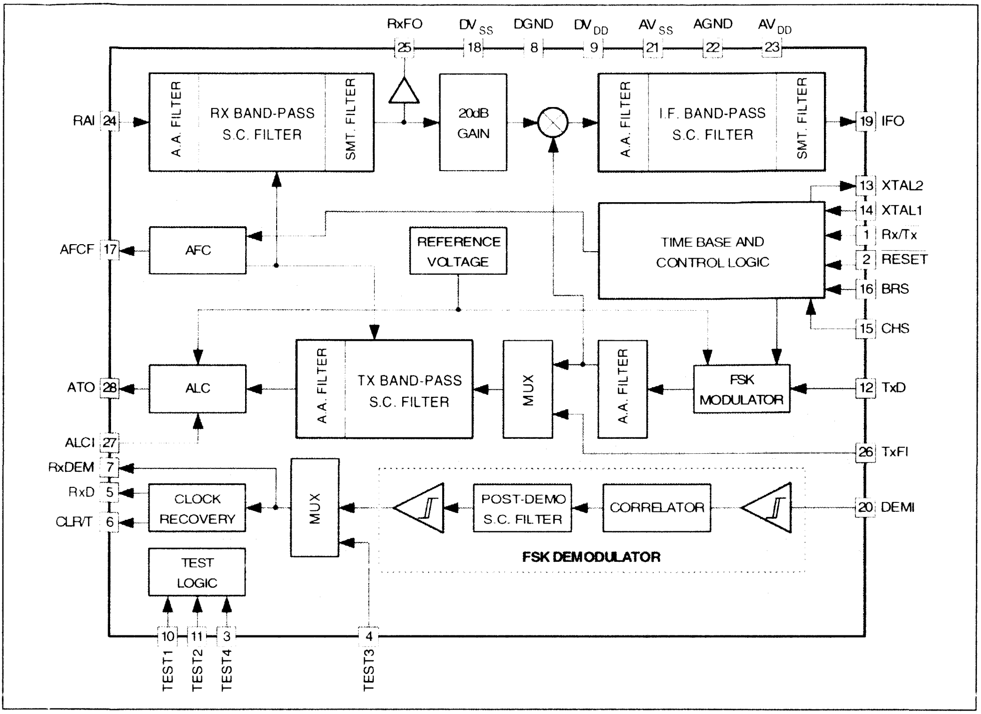
This is advance information on a new product now in development or undergoing evaluation. Details are subject to change without notice.

PIN DESCRIPTION

Pin Name	Pin Number	Pin Type	Description
$\text{Rx}/\overline{\text{Tx}}$	1	Digital	Rx or Tx mode selection input
$\overline{\text{RESET}}$	2	Digital	Logic reset and power-down mode input. Active when low.
TEST4	3	Digital	Test input which selects the Tx band-pass filter input (TxFI) when high.
TEST3	4	Digital	Test input which gives an access to the clock recovery input stage. This input is selected when TEST1 is high.
RxD	5	Digital	Synchronous receive data output
CLR/T	6	Digital	Rx or Tx clock according to the functional mode
RxDEM	7	Digital	Demodulated data output
DGND	8	Supply	Digital ground
DVDD	9	Supply	Digital positive supply voltage : 5V ± 5%
TEST1	10	Digital	Test input which cancels the Tx to Rx mode automatic switching and validates TEST3 input. Active when high.
TEST2	11	Digital	Test input which reduces the Tx to Rx mode automatic switching time. Active when high.
TxD	12	Digital	Transmit data input
XTAL2	13	Digital	Crystal oscillator output
XTAL1	14	Digital	Crystal oscillator input
CHS	15	Digital	Channel selection input
BRS	16	Digital	Baud rate selection input
AFCF	17	Analog	Automatic frequency control output for connecting compensation network.
DVSS	18	Supply	Digital negative supply voltage : -5V ± 5%
IFO	19	Analog	Intermediate frequency filter output
DEMI	20	Analog	FSK demodulator input
AVSS	21	Supply	Analog negative supply voltage : -5V ± 5%
AGND	22	Supply	Analog ground : 0V
AVDD	23	Supply	Analog positive supply voltage : 5V ± 5%
RAI	24	Analog	Receive analog input
RxFO	25	Analog	Receive filter output
TxFI	26	Analog	Transmit filter input (selected when TEST4 is high)
ALCI	27	Analog	Automatic level control input
ATO	28	Analog	Analog transmit output

7536-01.TBL

BLOCK DIAGRAM



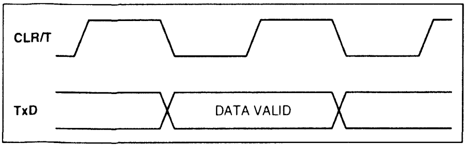
7536-02.EPS

TRANSMIT SECTION

The transmit mode is set when Rx/$\overline{\text{Tx}}$ = 0, if Rx/$\overline{\text{Tx}}$ is held at 0 longer than 3 seconds, then the device switches automatically in the Rx mode. A new activation of the Tx mode requires Rx/$\overline{\text{Tx}}$ to be returned to 1 for a minimum 2 microsecond period before being set to 0.

The Transmit Data (TxD) is sampled on a positive edge of CLR/T which delivers the transmit bit clock when the transmit mode is selected. This data enters a FSK modulator whose two basic frequencies are selected by the Baud Rate Selection pin (BRS) and the Channel Selection pin (CHS) according to the Table 1.

Figure 1 : Tx Data Input Timing



7536-03.EPS

Table1

BRS	CHS	Baud Rate (Baud)	Tx Frequencies (kHz) TxD=1 - TxD=0
0	0	600	81.75 - 82.35
0	1	600	67.2 - 67.8
1	0	1200	71.4 - 72.6
1	1	1200	85.95 - 87.15

These frequencies are synthesized from a 11.0592MHz crystal oscillator ; their precision is the same as the crystal one's (100 ppm).

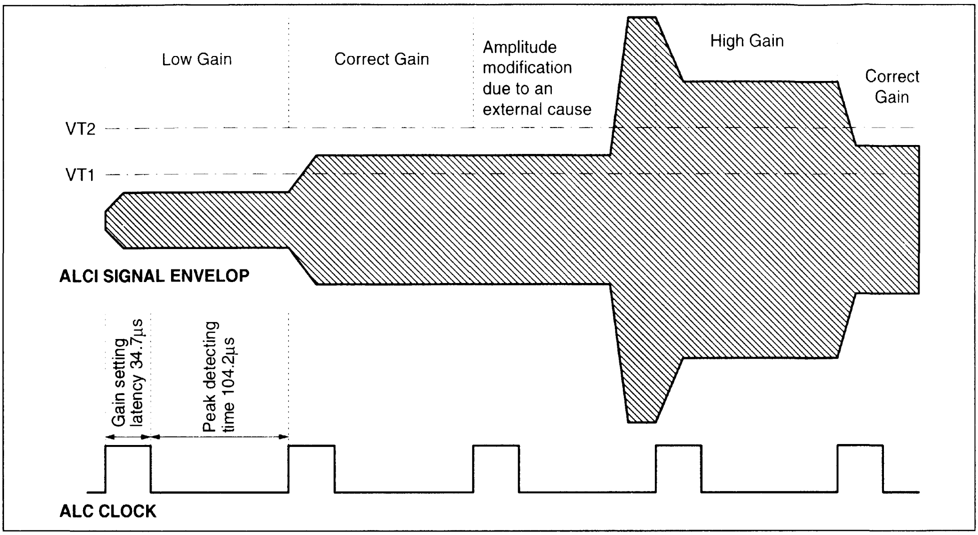
The modulated signal coming out of the FSK modulator is filtered by a switched-capacitor band-pass filter (Tx band-pass) in order to limit the output spectrum and to reduce the level of harmonic components.

The output stage of the Tx path consists of an Automatic Level Control (ALC) system which keeps the output signal (ATO) amplitude independant of the line impedance variations. This ALC is a variable gain system (with 32 discrete values) controlled by an analog feed-back signal ALCI. The ALC gain range is 0dB to -26dB and gain change is clocked at 7200Hz. Gain steps are of magnitude 0.84dB typically.

A period of this clock is decomposed into a 34.7µs gain settling latency and a 104.2µs peak detecting time. The gain change is related to the result of a peak detection obtained by making a direct comparison of ALCI maximum value (during detecting time) with two threshold voltages V_{T1} and V_{T2}.

- max (VALCI)< V_{T1} - The next gain is increased by 0.84dB
- $V_{T1} \leq$ max (VALCI)$\leq V_{T2}$ - No gain change
- V_{T2} < max (VALCI) - The next gain is decreased by 0.84dB.

Figure 2 : Automatic Level Control Timing Chart


Low Gain
Correct Gain
Amplitude modification due to an external cause
High Gain
Correct Gain
VT2
VT1
ALCI SIGNAL ENVELOP
Gain setting latency 34.7µs
Peak detecting time 104.2µs
ALC CLOCK
7536-04.EPS

RECEIVE SECTION

The receive section is active when Rx/$\overline{\text{Tx}}$ = 1.

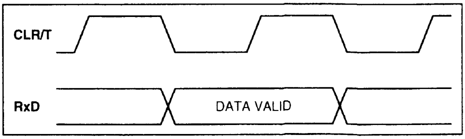
The baud rate and channel selection is also made according to Table 1.

The Rx signal is applied on RAI with a common mode voltage of 0 volt and filtered by a band-pass switched capacitor filter (Rx band-pass) centered on the received carrier frequency and whose bandwidth is around 6 kHz. The input voltage range on RAI is $2mV_{RMS}$ - $2V_{RMS}$.

The Rx filter output is amplified by a 20dB gain stage which provides symmetrical limitations for large voltage. The resulting signal is down-converted by a mixer which receives a local oscillator synthesized by the FSK modulator block. Finally an intermediate frequency band-pass filter (IF band-pass) whose central frequency is 2.7kHz when BRS = 0 and 5.4kHz when BRS = 1 improves the signal to noise ratio before entering the FSK demodulator. The coupling of the intermediate frequency filter output (IFO) to the FSK demodulator input (DEMI) is made by an external capacitor C5 (1µF ±10%, 10V) which cancels the Rx path offset voltage.

A clock recovery circuit extracts the receive clock (CLR/T) from the demodulated output (RxDEM) and delivers synchronous data (RxD) on the positive edge of CLR/T.

Figure 3 : Rx Data Output Timing



ADDITIONAL DIGITAL AND ANALOG FUNCTIONS

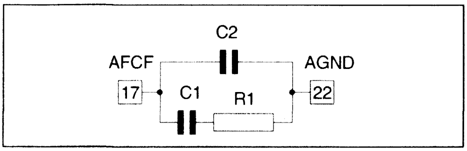
A reset intput ($\overline{\text{RESET}}$) initializes the device.

When $\overline{\text{RESET}}$ = 0, the device is in power-down mode and all the internal logic is reset. When $\overline{\text{RESET}}$ = 1, the device is active.

A time base section delivers all the internal clocks from a crystal oscillator (11.0592MHz). The crystal is connected between XTAL1 and XTAL2 pins and needs two external capacitors C3 and C4 depending on the crystal characteristic typically 22pF ±10% for proper operation. It is also possible to provide directly the clock on pin XTAL1 ; in this case C3 and C4 should be removed.

An Automatic Frequency Control (AFC) Section adjusts the central frequency of Rx and Tx band-pass filter to the carrier central frequency. The stability of the AFC loop is ensured by an external compensation network C1 (470nF ±10%, 10V), C2 (47nF ±10%, 10V) and R1 (1.5kΩ ±5%) connected to pin AFCF.

Figure 4 : Automatic Frequency Loop Filter



TESTING FEATURES

- An additionnal amplifier allows the observation of the Rx band-pass filter output on pin RxFO.
- A direct input to the Tx band-pass filter (TxFI) is available and selected when TEST4 = 1.
- The 3 second normal duration of the Tx to Rx mode automatic switching is reduced to 1.48ms when TEST2 = 1.
- When TEST1 = 1 the Tx to Rx mode automatic switching is desactivated and the functional mode of the circuit is controlled by $Rx/\overline{Tx}$ as follow : when $Rx/\overline{Tx}$ = 0 the circuit is transmitting continuously, when $Rx/\overline{Tx}$ = 1 the clock recovery block is disconnected from the FSK demodulator for testing purpose, in this configuration TEST 3 is the data input of the clock recovery block, RXDEM follow TEST3 and RxD delivers the resynchronized data.

POWER SUPPLIES WIRING AND DECOUPLING PRECAUTIONS

The ST7536 has two positive power supply pins, two negative power supply pins and two ground pins in order to separate internal analog and digital supplies. The analog and digital terminals of each supply pair must be connected together externally and require special routing precautions in order to get the best receive sensitivity performances. The three major routing requirements are :

- The ground impedance should be as low as possible, for this purpose the AGND an DGND terminals can be connected via a local plane.
- The positive and negative power supplies (AV_{DD}, DV_{DD}, AV_{SS}, DV_{SS}) should be star-connected, avoiding common current path for the digital and analog power supplies terminals.
- Five decoupling capacitors located as close as possible to the power supply terminals should be used. Two 2.2µF tantalum and two 100nF ceramic capacitors perform the main decoupling function in the vicinity of the analog power supplies and a 100nF ceramic capacitor in the vicinity of the positive digital power supply is used to reduce the high frequency perturbations generated by the logic part of the circuit.

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
AV_{DD}/DV_{DD}	Positive Supply Voltage (1)	-0.3, +7	V
AV_{SS}/DV_{SS}	Negative Supply Voltage (1)	-7, +0.3	V
$V_{AGND/DGND}$	Voltage between AGND and DGND	-0.3, +0.3	V
V_I	Digital Input Voltage	DGND-0.3, DV_{DD}+0.3	V
V_O	Digital Output Voltage	DGND-0.3, DV_{DD}+0.3	V
I_O	Digital Output Current	-5, +5	mA
V_i	Analog Input Voltage	AV_{SS}-0.3, AV_{DD}+0.3	V
V_o	Analog Output Voltage	AV_{SS}-0.3, AV_{DD}+0.3	V
I_o	Analog Output Current	-5, +5	mA
P_D	Power Dissipation	500	mW
T_{oper}	Operating Temperature	- 25, + 70	°C
T_{stg}	Storage Temperature	- 65, + 150	°C

7536-02.TBL

Notes : 1. The voltages are referenced to AGND and DGND.

2. Latch-up problems can be overcome with 2 reverse biased schottky diodes connected respectively between A/DV_{DD} & A/DGND and A/DV_{SS} & A/DGND.

3. Absolute maximum ratings are values beyond which damage to device may occur. Functional operation under these conditions is not implied.

GENERAL ELECTRICAL CHARACTERISTICS

The test conditions are A/DV$_{DD}$ = +5V, A/DV$_{SS}$ = -5V, A/DGND = 0V,
T$_{amb}$ = -10 to 70°C unless otherwise specified

Symbol	Parameter	Test Conditions	Min.	Typ	Max	Unit
AV$_{DD}$/DV$_{DD}$	Positive Supply Voltage		4.75	5	5.25	V
AV$_{SS}$/DV$_{SS}$	Negative Supply Voltage		-5.25	-5	-4.75	V
AI$_{DD}$ + DI$_{DD}$	Positive Supply Current in Tx Mode	$\overline{RESET}$ = 1, RX/$\overline{Tx}$ = 0		30	35	mA
AI$_{DD}$ + DI$_{DD}$	Positive Supply Current in Rx Mode	$\overline{RESET}$ = 1, RX/$\overline{Tx}$ = 1		29	34	mA
AI$_{SS}$ + DI$_{SS}$	Negative Supply Current in Tx Mode	$\overline{RESET}$ = 1, RX/$\overline{Tx}$ = 0	- 34	- 29		mA
AI$_{SS}$ + DI$_{SS}$	Negative Supply Current in Rx Mode	$\overline{RESET}$ = 1, RX/$\overline{Tx}$ = 1	- 33	- 28		mA
AI$_{DD}$ + DI$_{DD}$	Positive Power-down Current	$\overline{RESET}$ = 0, RX/$\overline{Tx}$ = 1 XTAL1 = 1			1.2	mA
AI$_{SS}$ + DI$_{SS}$	Negative Power-down Current		- 1.2			mA
V$_{IH}$	High Level Input Voltage	Digital inputs except XTAL1	2.2			V
V$_{IL}$	Low Level Input Voltage	Digital inputs			0.8	V
V$_{OH}$	High Level Output Voltage	Digital outputs, I$_{OH}$ = - 400µA	2.4			V
V$_{OL}$	Low Level Output Voltage	Digital outputs, I$_{OL}$ = 1.6mA			0.4	V
V$_{IH}$	High Level Input Voltage	XTAL1 input	3.6			V
DC	XTAL1 Clock Duty Cycle	External clock	40		60	%

7536-03.TBL

TRANSMITTER ELECTRICAL CHARACTERISTICS

The test conditions are A/DV$_{DD}$ = +5V, A/DGND = 0V, A/DV$_{SS}$ = -5V,
T$_{amb}$ = -10 to +70°C unless othewise specified

Symbol	Parameter	Test Conditions	Min.	Typ	Max	Unit
V$_{TAC}$	Max Carrier Output AC Voltage	R$_L$ = 2kΩ, V$_{ALCI}$ < V$_{T1}$	2.8	3.2	3.7	V$_{PP}$
HD2	Second Harmonic Distortion	R$_L$ = 2kΩ, V$_{ALCI}$ < V$_{T1}$			0.32	%
FD	FSK Peak-to-peak Deviation	BRS = 0 BRS = 1		600 1200		Hz Hz
TRxTx	Carrier Activation Time	After Rx/$\overline{Tx}$ 1 → 0 transition			1	ms
TALC	Carrier Stabilisation Time	ALC maximum settling time 32 gain steps			5	ms
DRNG	ALC Dynamic Range		25	26	27	dB
VT1	ALC Low Threshold Voltage		1.81	1.87		V
VT2	ALC High Threshold Voltage			2.12	2.18	V
GST	ALC Gain Step			0.84		dB
PSRR1 PSRR2	Power supply rejection ratio on ATO (1)	V$_{IN}$ = 200mV$_{PP}$, f$_{IN}$ = 50Hz on V$_{DD}$ or V$_{SS}$	35 10			dB dB

7536-04.TBL

Note 1 : This characteristic is guaranteed by correlation.

RECEIVER ELECTRICAL CHARACTERISTICS

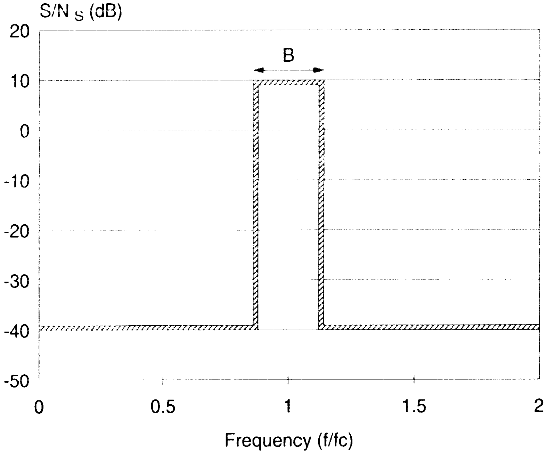
The test conditions are A/DV$_{DD}$ = +5V, A/DGND = 0V, A/DV$_{SS}$ = -5V, T$_{amb}$ = -10 to +70°C unless othewise specified

Symbol	Parameter	Test Conditions	Min	Typ	Max	Unit
V$_{IN}$	Maximum Input Signal				2	V$_{RMS}$
R$_{IN}$	Input Impedance		100			kΩ
RCJ	Recovered Clock Jitter	Percentage of the nominal clock	- 5		+ 5	%
PSRR1 PSRR2	Power supply rejection ratio on RxFO (1)	V$_{IN}$ = 200mV$_{PP}$, f$_{IN}$ = 50Hz on V$_{DD}$ or V$_{SS}$	35 10			dB dB
V$_{IN0}$ V$_{IN1}$	Rx sensitivity (1)	Typical measured BER < 10^{-5} BRS = 0 BRS = 1			2 3	mV$_{RMS}$
BER1 BER2	Bit error rate at minimum Rx signal (1)	White Noise, S/N = 15dB RAI = 2mV$_{RMS}$, BRS = 0 RAI = 3mV$_{RMS}$, BRS = 1		$2 \cdot 10^{-5}$ $3 \cdot 10^{-4}$	10^{-3} 10^{-3}	
BER3	Bit error rate at maximum Rx signal (1)	RAI = 2V$_{RMS}$, White Noise S/N = 25dB		10^{-7}	10^{-3}	
BER4	Bit error rate at medium Rx signal (1)	RAI= 0.6V$_{RMS}$, S/N= 15dB		10^{-6}	10^{-3}	
BER5	Bit error rate with impulsive noise (1)	RAI = 90mV$_{RMS}$, N = 5V$_{PP}$ pulse wave, f = 100Hz, duty cycle = 10%			10^{-3}	
BER6 BER7	Bit error rate with modulated sinusoidal noise Ns (1)	S+ Ns < 0.2V$_{RMS}$, Ns = sine carrier with 80% AM modul., f$_m$ = 1kHz, See Figure 5 S$_{min}$ = 2mV$_{RMS}$, BRS = 0 S$_{min}$ = 3mV$_{RMS}$, BRS = 1			10^{-3} 10^{-3}	

7536-05.TBL

Note 1 : This characteristic is guaranteed by correlation

Figure 5 : S/N Mask for 80% AM Sine Noise



B = 20kHz at 600 Bit/s (BRS = 0)
B = 40kHz at 1200 Bit/s -BRS = 1)

fc : Central Carrier Frequency

7536-07.EPS

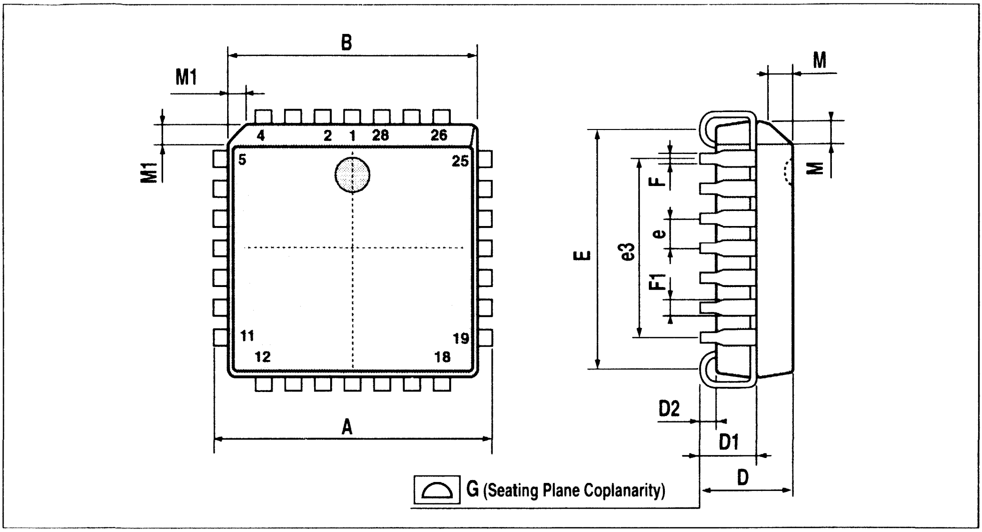
FILTER TEMPLATES

Frequency (kHz)	Test Conditions	Amplitude (dB)		
		Min.	Typ.	Max.
RECEIVE AND TRANSMIT FILTER				
54	BRS = 0, CHS = 0			- 35
79.05		- 4	- 3	- 2
Ref 82.05			0	
85.05		- 4	- 3	- 2
123				- 35
44.4	BRS = 0, CHS = 1			- 35
65		- 4	- 3	- 2
Ref 67.46			0	
69.93		- 4	- 3	- 2
101.13				- 35
47.57	BRS = 1, CHS = 0			- 35
69.64		- 4	- 3	- 2
Ref 72.28			0	
74.92		- 4	- 3	- 2
108.36				- 35
57.08	BRS = 1, CHS = 1			- 35
83.57		- 4	- 3	- 2
Ref 86.74			0	
89.91		- 4	- 3	- 2
130.03				- 35
INTERMEDIATE FREQUENCY FILTER				
1.2	BRS = 0			- 35
2.15		- 5	- 3	- 2
Ref 2.7			0	
3.25		- 5	- 3	- 2
5.8				- 35
2.4	BRS = 1			- 35
4.3		- 5	- 3	- 2
Ref 5.4			0	
6.5		- 5	- 3	- 2
11.6				- 35

7536-06.TBL

PACKAGE MECHANICAL DATA

28 PINS - PLASTIC CHIP CARRIER



PMPLCC28.EPS

Dimensions	Millimeters			Inches		
	Min.	Typ.	Max.	Min.	Typ.	Max.
A	12.32		12.57	0.485		0.495
B	11.43		11.58	0.450		0.456
D	4.2		4.57	0.165		0.180
D1	2.29		3.04	0.090		0.120
D2	0.51			0.020		
E	9.91		10.92	0.390		0.430
e		1.27			0.050	
e3		7.62			0.300	
F		0.46			0.018	
F1		0.71			0.028	
G			0.101			0.004
M		1.24			0.049	
M1		1.143			0.045	

PLCC28.TBL

ST7536 APPLICATION NOTE

SGS-THOMSON
MICROELECTRONICS

APPLICATION NOTE

ST7536

By Joël HULOUX

SUMMARY

Page

I - INTRODUCTION TO THE ST7536

The ST7536 is a half duplex synchronous FSK-modem, and has been designed to operate on powerline networks. For a complete communication system, a micro-controller and a powerline-interface (PLI) are needed (see Figure 1).

Such a system is able to transmit and receive on 4 different channels with 2 different data rates (600 and 1200 baud). The baudrate (BRS) and channel (CHS) selection is made, according to the table 1:

Table 1

BRS	CHS	Bitrate	Xmit Freq (KHz) TxD = 1	Xmit Freq (KHz) TxD = 0
0	0	600	81.75	82.35
0	1	600	67.20	67.80
1	0	1200	71.40	72.60
1	1	1200	85.95	87.15

The ST7536 is a half duplex modem, as it has two operation modes; receive or transmit data. The mode selection is made with a Rx/Tx control input.

Data input and output are related to the clock signal; it's a synchronous modem. This clock signal is generated by the ST7536.

Only a few external components have to be added for full operation of the ST7536: a crystal, four resistors and five capacitors.

II - ST7536 DESCRIPTION

The ST7536 is a single chip modem; all the electrical circuits needed for a complete modem are inside the chip. The modem is available in 28 pins PLCC (see Figure 2).

In transmit mode the Transmit Data (TXD) is sampled on the positive edge of the clock (CLR/T). Then the data enters the FSK modulator. The frequency on which this modulator operates is set by the time base and control logic. In normal operation the multiplexer (MUX) selects the FSK modulator signal to be send to the transmit filter. This filter is a switched capacitor band-pass filter. The time base and control logic uses the Automatic Frequency Control (AFC) to set this filter at the transmit frequency, corresponding to the selected channel. After filtering, the transmit signal is sent to an Automatic Level Control (ALC). This control is used to overcome problems with line impedance variations. The powerlines on which the modem has to operate, have variations in their line characteristics, which are very frequent and totally unpredictable. The automatic level control uses a feed back signal (ALCI) from the powerline interface to adjust the transmit output (ATO).

In receive mode the signal enters the chip on the Receive Analog Input (RAI). The received signal is filtered in the receive band-pass filter. It's just like the transmit filter , a switched capacitor filter.The automatic frequency control is used to set it on the right frequency. After being amplified the signal is down converted and filtered in the intermediate frequency band-pass filter. The resulting signal is sent to the FSK demodulator. The coupling of the intermediate frequency filter output (IFO) to the FSK DEModulator Input (DEMI) is made by an external capacitor which cancels an eventual offset voltage. A clock recovery circuit extracts the receive clock (CLR/T) from the demodulated output (RXDEM) of the FSK demodulator. Synchronous received data (RXD) is delivered on the positive edge of the clock.

A time base section delivers all the internal clock signals from a crystal oscillator running at 11.0592 MHz. The crystal is connected between the XTAL1 and XTAL2 pins. It is also possible to provide directly a clock signal on XTAL1 instead of using a crystal.

To debug the chip and test external circuits the ST7536 provides some test options. The transmit band-pass filter can be observed using a direct input on the filter. This input (TXFI) is selected by the multiplexer if TEST4 = 1. The Receive band-pass Filter Output (RXFO) is provided at pin 25. Finally the clock recovery can be observed when TEST1 = 1. In this case the TEST3 input gives a direct input to the clock recovery block.

Figure 1

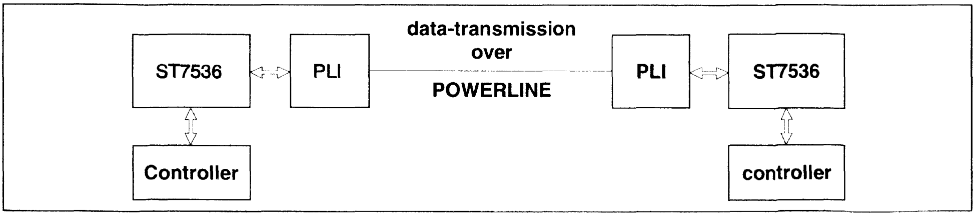
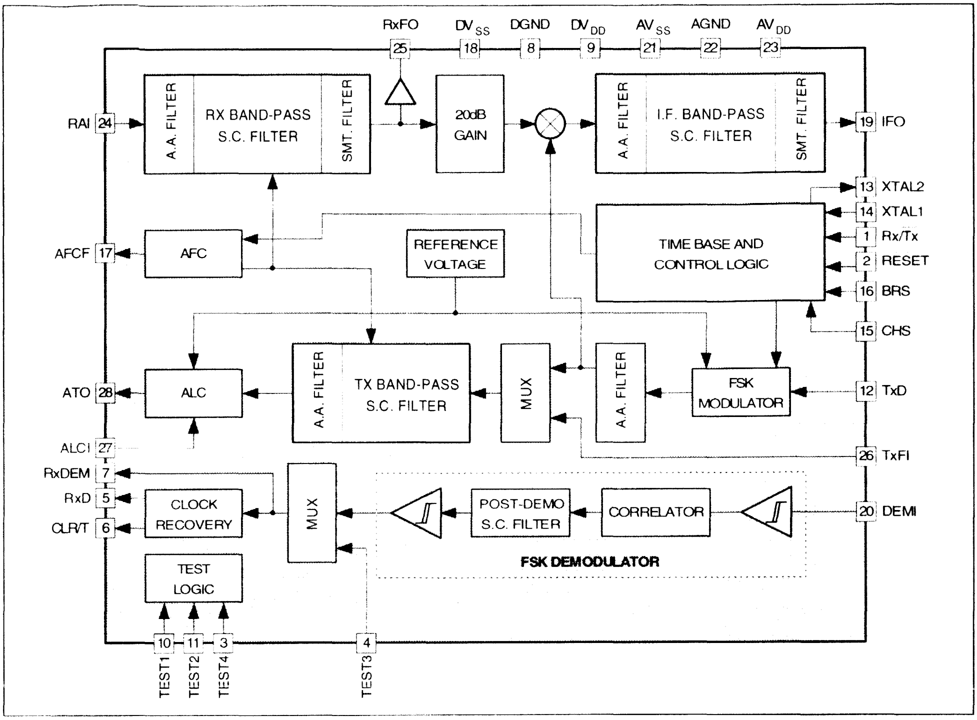


Figure 2 : Block Diagram



III - ST7536 PIN DESCRIPTION

The pin decription is not given in numerical order, but the pins are described in relation with their function and consequently sometimes with other pins.

- power supply input
- channel selection
- crystal oscillator input
- AFCF stabilisation
- automatic level control input
- data input and output
- test inputs
- IFO/DEMI output/input
- transmit output and receive input
- Rx/Tx control input
- reset input

III.1 - Power Supply Input

- pin 8 (DGND) : Digital ground (0V)

- pin 9 (DV$_{DD}$) : Digital positive supply voltage (+5V)

- pin 18 (DV$_{SS}$) : Digital negative supply voltage (-5V)

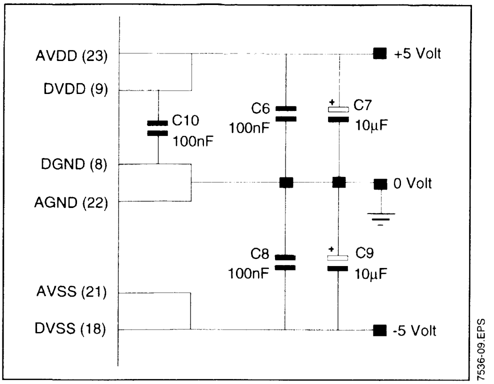
- pin 21 (AV$_{SS}$) : Analog negative supply voltage (-5V)

- pin 22 (AGND) : Analog ground (0V)

- pin 23 (AV$_{DD}$) : Analog positive supply voltage (+5V)

Internally the ST7536 has separated power supplies: the digital and analog circuits are separated. Externally the power supplies should be connected together. For decoupling, both the positive and negative supplies are decoupled with 2 capacitors. C6 and C7 decouple the positive, C8 and C9 the negative supplies. For proper operation the digital positive supply voltage should be decoupled with a capacitor (C10) mounted close to pin 9. C6,C8 and C10 are 100nF/16V ceramic capacitors, C7 and C9 10µF/16V tantal capacitors (see Figure 3).

Figure 3



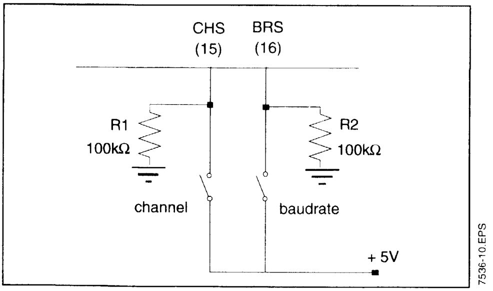
III.2 - Channel Selection

- pin 15 (CHS) : Channel selection input
- pin 16 (BRS) : Baudrate selection input

Both inputs are digital inputs (0/+5V). The ST7536 operates with two bitrates: 600 and 1200 baud. These bitrates are selected with pin 16 (BRS). For both bitrates the ST7536 offers two channels, which are selected with pin 15 (CHS).

A logical "0" is represented by 0V, a "1" by +5V. R1 and R2 are pull-down resistors, creating a logical "0". Closing a switch gives a "1". The selection is made according to table I.

Figure 4



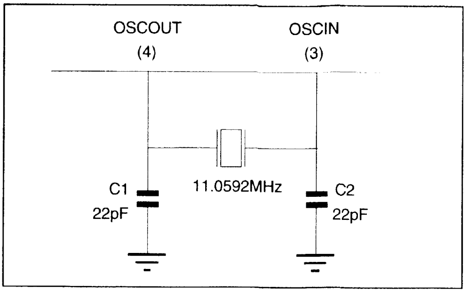
III.3 - Crystal Oscillator Input

- pin 13 (XTAL2) : Crystal oscillator output
- pin 14 (XTAL1) : Crystal oscillator input

The internal crystal oscillator of the ST7536 needs an external crystal. This one should be a 11.0592MHz crystal. Two capacitors (C1 and C2) have to be added for proper operation. They are typically 22pF/10V ceramic capacitors.

It is also possible to connect directly a clock signal to the oscillator input, in this case the crystal and the capacitors should be removed. On the application board this option is not used. The ST7536 clock signal is the time reference of the system.

Figure 5

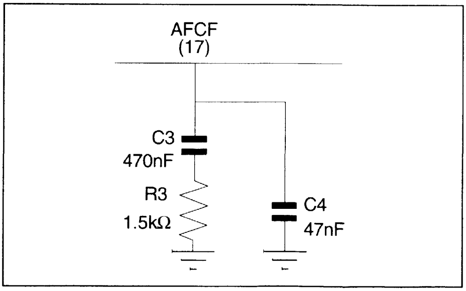


III.4 - AFCF Stabilisation

- pin 17 (AFCF) : Automatic frequency control output

In the ST7536 an automatic control section adjusts the central frequency of the receive and transmit band-pass filters. The stability of this section has to be ensured with an external RC network.

Figure 6



III.5 - Automatic Level Control Input

- pin 27 (ALCI) : Automatic level control input

The output stage of the transmit path consists of an automatic level control (ALC). It offers the possibility to keep the output voltage of the power amplifier independent of variations of the powerline network. The impedance of these networks can be anywhere in the range of 5 - 100Ω. If the impedance of the powerline changes, the output of the amplifier will change. With the ALC input it is possible to correct these output variations. To control the output of the powerline interface a feed-back signal is needed. This signal is sent through an amplifier. The automatic level control can decrease the maximum transmit output in 32 steps of 0.84dB. The gain range is 0dB → -26dB. A peak detection is done on the signal presented on the ALCInput and the ALC compares it to two reference voltages, VT1 (1.87V) and VT2 (2.12V).

If max. VALCI < VT1 the next gain is increased by 0.84dB.

If VT1 < max. VALCI < VT2 there is no gain change.
If VT2 < max. VALCI the next gain is decreased by 0.84dB.

The gain of the feed-back amplifier is such that the feed-back signal peak voltage falls between V_{T1} and V_{T2}.

Example:
The wanted interface output voltage is 0.5V(peak).
For a 0.5V output peak voltage

$$G = \frac{V_{out\ peak}}{\left(\frac{V_{T1} + V_{T2}}{2}\right)} = \frac{0.5}{2} = 4 \text{ (12dB).}$$

Then the feed-back amplifier should have a gain of 4x (= +12dB). The ST7536 starts up. VALCI = 0V (VALCI < VT1). The ATO output is increased with a gain of +0.84dB. On a certain moment the output voltage over the powerline will become 0.5 V(peak). This signal is amplified to 2.0 V(peak). Then the ALC stops increasing the ATO output, which will remain at its actual level. If the line impedance increases, the power amplifier of the interface might deliver more output voltage. If the output voltage of this amplifier increases, the ALCI voltage will be higher than VT2. The ALC will then immediately decrease the ATO output. And so the output of the interface can be made independent of impedance variations of the powerline.

Of course this will operate only if the power amplifier in the interface is able to drive all the impedances at the required output voltage. Let's say the impedance of the line becomes 0.1Ω. The ALC will increase the output of the ATO. But if the power amplifier is not able to drive such low impedances, the only result will be an output signal with a large distortion. Therefore on the application board the ALCInput is set at 0V with a resistor (R4). The ATO will be always at maximum output (1.25 Vrms). The powerline interface has been designed to drive all impedances from 0.5 - 100Ω with this input. To be able to do some experiments with the ALC, a resistor is used to set the ALCI at 0V. It gives the possibility to inject a signal on the ALCI. This would not have been possible if on the printed circuit board a short circuit to ground had been made (see Figure 7)

III.6 - Data Input and Output

- pin 5 (RxD)	:	Synchronous receive data output
- pin 6 (CLR/T)	:	Receive and transmit clock
- pin 7 (RxDEM)	:	Demodulated data output
- pin 12 (TxD)	:	Transmit data input

The ST7536 is a synchronous modem; data input and output are related to the clock (CLR/T). In transmit mode the ST7536 generates this clock signal. The transmit data are sampled on the positive edge of CLR/T. Therefore the TxD should be valid at that moment. In receive mode the demodulated (receive) data is available at pin 7 (RxDEM). A clock recovery circuit extracts the clock signal from the demodulated data and delivers synchronous data (RxD) on the positive edge of CLR/T.

On the application board the RxDEM data output is not used. All the data signals from and to the ST7536 (RxD, TxD) are related to the clock (CLR/T) (see Figure 8).

Figure 7

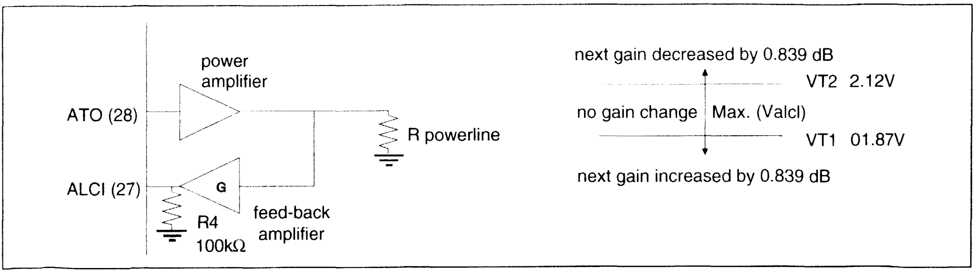
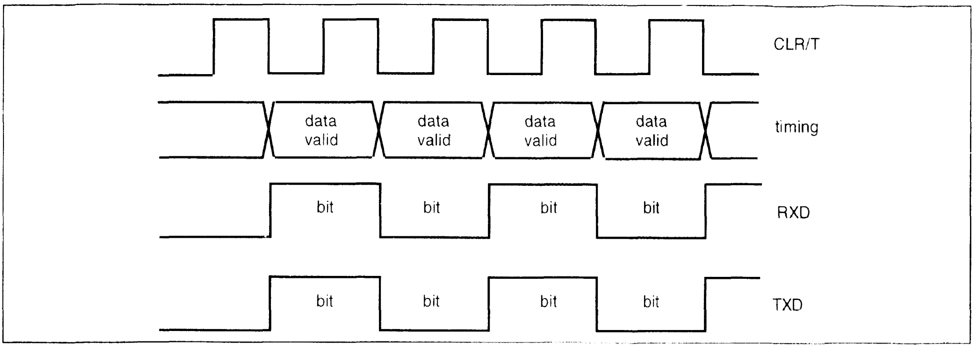


Figure 8



III.7 - Test Inputs

- **pin 3 (TEST4)** : Test input, with a "1" on this pin the multiplexer selects the transmit band-pass filter input (TXFI).
- **pin 4 (TEST3)** : Test input which gives a direct acces to the clock recovery circuit.This input is selected when TEST1 = "1".
- **pin 10 (TEST1)** : Test input, a "1" on this pin cancels the automatic switching from transmit to receive mode, and validates the TEST3 input to the clock recovery circuit.
- **pin 11 (TEST2)** : Test input, a "1" on this pin reduces the automatic switching time (from transmit to receive mode) to 1.48 msec.

On the application board TEST 2/3/4 are not used, and pin 3, 4, and 11 are therefore set at 0V. With a switch TEST1 can be set at "0" or "1". See also the Rx/Tx control input.

III.8 - IFO/DEMI Output / Input

- **pin 19 (IFO)** : Intermediate frequency filter output
- **pin 20 (DEMI)** : FSK demodulator input

The connection between the intermediate frequency filter output and the FSK demodulator input should be made externally with a capacitor (C5, 1µF/10V).

III.9 - Transmit Output and Receive Input

- **pin 24 (RAI)** : Receive analog input
- **pin 28 (ATO)** : Analog transmit output

Pin 24 is the receive input of the ST7536. The receive output of the powerline interface should be connected to this pin. The maximum input voltage is 2V_{RMS}. The receive sensitivity of the ST7536 is 2mV_{RMS} f on channel 1 and 2 (600 baud), and 3mV_{RMS} on channel 3 and 4 (1200 baud).

Pin 28 is the transmit output of the ST7536. The transmit input of the powerline interface should be connected to this pin. The ATO output is regulated by the ALCI circuit. The maximum output voltage is 3.5V_{PP}. The second harmonic distortion is about -53dB.

III.10 - Rx/$\overline{\text{Tx}}$ Control Input

- **pin 1 (Rx/$\overline{\text{Tx}}$)** : Receive or transmit mode selection input

The ST7536 is a half duplex modem and has therefore two operation modes: receive and transmit. This mode selection is done with the Rx/$\overline{\text{Tx}}$ input. The transmit mode is selected when Rx/$\overline{\text{Tx}}$ is "0". If Rx/$\overline{\text{Tx}}$ is held at "0" longer than 3 seconds, the ST7536 switches back to receive mode. To set the ST7536 again in transmit mode, Rx/$\overline{\text{Tx}}$ should be held at "1" for a minimum of 3 microseconds before being set to "0".

The carrier activation time is 1 msec.

To be able to observe the transmit output of the ST7536 on the power line interface for a longer time than 3 seconds it is possible to use the test 1 Input. If this input is set at "1" the automatic switching is disactivated.

Then it is possible to transmit a signal but not to receive.

III.11 - Reset Input

- pin 2 (RESET) : logic reset and power-down input

When this input is set at "0" the ST7536 is in power-down mode. All the internal logic is then reset. For normal operation this input should be set at "1". On the application board this input is controlled by the micro-controller.

IV - POWERLINE INTERFACE

The power line interface (PLI) connects the ST7536 to the powerlines.The following PLI has been designed according to the ENEL (italian electricity distributor) specifications : (This PLI is suitable to CENELEC european specification and the FCC USA spec) (see Figure 9)

transmit output :
R powerline > 5Ω → 1 - 2 V_{RMS}
R powerline < 5Ω → 200-400mA_{RMS}
Second Harmonic Distortion ← 72dB
for R powerline = 18Ω
receive sensitivity : 1.5mV_{RMS}

In transmit mode the powerline interface amplifies and filters the transmit signal (ATO) from the ST7536. The maximum output current that can be taken from ATO is 1mA. Therefore a buffer is used to protect the ST7536 and in order to drive the next stages in the powerline interface. The Second Harmonic Distortion (HD2) of the transmit signal from the ST7536 is -53dB. To suppress the harmonics a low pass filter (LPF) is used. The filtered signal is then sent to a power amplifier, which must drive powerlines with impedances from 1 to 100Ω,via the transformer. The transformer is not only used to put signals on the powerlines. It's also used as a band pass filter, in order to suppress the second harmonic of the transmit signal to a level of less than -72dB.

In receive mode the transformer extracts the signal from the powerline. Before sending it to the receive input (RAI) of the ST7536, it is amplified with a level of 34dB in the preamplifier.

The buffer and power amplifier are switched off in receive mode,in order to avoid the low output impedance of the power amplifier attenuating the received signals.

IV.1 - Buffer and Low Pass Filter

These two functions are build up around UA1 (see Figure 10).

Figure 9

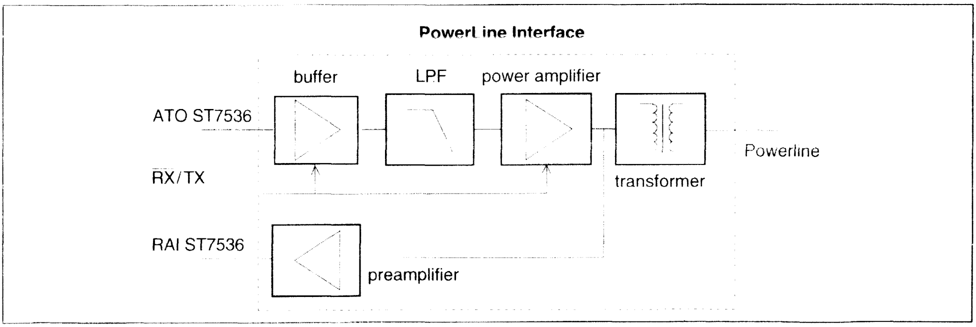
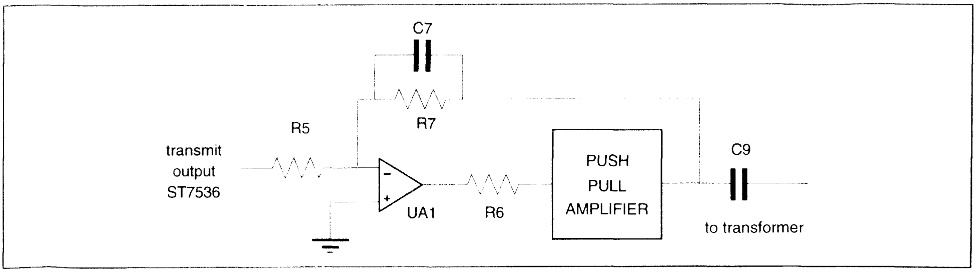


Figure 10



A feed back from the output of the power amplifier to the operational amplifier is done with R7/C7. This gives a low pass function and therefore the possibility to create a low pass filter.The ST7536 operates on 4 channels : 67, 72, 82 and 86kHz. With R7 and C7 the cut off frequency of this filter is set.If this frequency is set at 75kHz, the difference between 75kHz and 150kHz (second harmonic) signals is only 3dB,because such a filter has already an attenuation of 3dB at the cutoff frequency (see figures below).

Figure 11

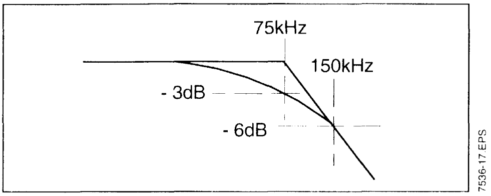
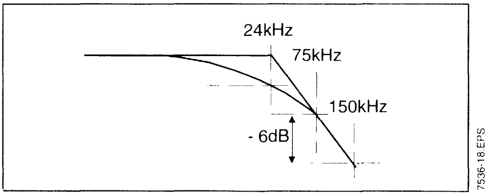


Figure 12



To ensure an attenuation of 6dB of the second harmonic, the cut off frequency has been set at 24kHz.

With R7 = 10kΩ
f = 1/(6.28*R7*C7) = 24kHz
C7 = 680pF

The ratio R7/R5 provides sufficient amplification on the transmit frequency, to drive the power amplifier at optimum performances.The frequency differences of the four channels result in a different output of the low pass filter. Therefore the ratio of R7/R5 is not the same for all the four channels.

Channel	R7 (Ω)	R5 (Ω)	ATO (Vpp)
1	10k	1500	3.3
2	10k	1800	3.6
3	10k	1800	3.5
4	10k	1500	3.2

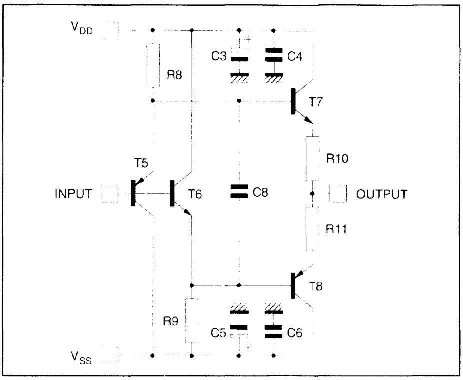
The connection of the operational amplifier to the power amplifier is done with R6. This resistor is added to avoid oscillation. Without this resistor stable operation cannot be guaranteed. The value of R6 is determinated with experiments to be 330Ω.

An other function of R6 is to increase the load impedance seen by the operational amplifier. The impedance is R6 plus the input impedance of the power amplifier. If this impedance is to low the operational amplifier will not be able to drive the power amplifier in optimum performances. The maximum voltage swing will decrease and the second harmonic distortion will increase. Different operational amplifiers have been tested. The TL071C gives the best performances.

IV.2 - Power Amplifier

The power amplifier increases the output signal of the operational amplifier and low pass filter (UA1).

Figure 13



The input impedance is increased because it's multiplied by the Beta of T5/6, which are no longer used as diodes. Therefore R8 and R9 could be decreased, to deliver more current to T7/T8. The optimum performances of the amplifier were obtained with a value of 820Ω for R8 and R9. An other solution to deliver more current to the output transistors is the addition of C8. It will decrease the input impedance, but also deliver extra current to T7 by T6, and to T8 by T5. Other transistors have been used also a BD237 for T6/T7, a BD238 for T5/T8. These transistors can deliver more output power, and are not much expensive than the 2N2222/2N2907. Furthermore, the collectors are connected to the (metal) package. This gives the possibility for a mechanical connection of T5/T8 and T6/T7. This will result in the same temperature in both transistors, what will avoid thermal runaway. To decouple the power supplies C3/C5 (22μF/16V) and C4/C6 (100nF/16V) are used, mounted close to T7 and T8.

Using this configuration, it is possible to provide 1 → 2 V_{RMS} in powerlines with impedances from 5 → 100Ω.

IV.3 - Transformer

A transformer is used to connect the power amplifier and the preamplifier to the powerline.This transformer has to :

- separate the rest of the interface from the powerline
- put the transmit signal on the powerline
- extract the received signal from the powerline
- filter the 50/60Hz signal coming from the powerline
- filter the second harmonic of the transmit signal.

The transformer is a TOKO T1002N,which has two primary windings and one secondary winding. The ratios of the windings are 4:1:1 (turns) (see also the Figure 14). Typical values of the transformer are:

L1t windings : 9.4 µH
L4t winding : 140 µH

The primary windings of the transformer are used to create a bandpass filter. The resonance frequency is set at the transmit frequency with C10/C11. These capacitors are in parallel with the primary windings (1t/4t). The equivalent value for those two windings can be calculated according to:

$$Leq = L1t + L4t + 2M$$

$$M = k \cdot \sqrt{L1t \cdot L4t} \quad \left(k = \frac{1}{\sqrt{2}}\right) \qquad (16)$$

With the given values:

$$M = (9.4\ \mu H * 140\ \mu H)^{0.5}$$
$$= (1316\ \mu H)^{0.5} = 36.3\ \mu H$$
$$Leq = L1t + L4t + 2 \cdot (L1t \cdot L4t)^{0.5}$$
$$= 9.4\mu H + 140\mu H + 2 \cdot 25.6\mu H = 200\mu H$$

The resonance frequency of this LC network is dependent of Ceq = Cp = C10//C11 and Leq according to:

$$f_{res} = \frac{1}{2\pi \times \sqrt{L_{eq} \cdot C_p}} \qquad (17)$$

$$C_P = \frac{\left(\frac{1}{2\pi \cdot f_{res}}\right)^2}{Leq} \qquad (18)$$

As this filter is very sharp, there are different values for Cp on each (transmit) frequency.

channel 1 : f = 82kHz → C_P = 18nF = 10nF // 6.8nF
channel 2 : f = 67kHz → C_p = 28nF = 22nF // 6.8nF
channel 3 : f = 72kHz →
C_P = 24nF (only 1 capacitor)
channel 4 : f = 86kHz → C_P = 17nF = 10nF // 5.6nF

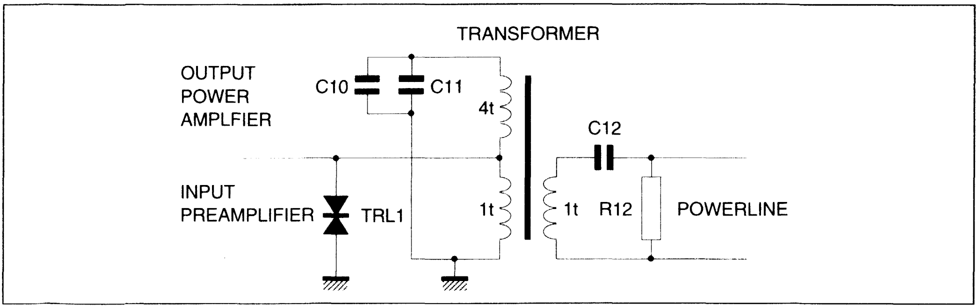
On channel 3 only 1 capacitor is needed and therefore C11 doesn't exist. On a printed circuit board the capacitors should be mounted close to the transformer. In order to get the best filter performances.The capacitors (C10/C11) have to be linear, such as the KS (styroflex) types.

C12 is used to filter the 50/60Hz signal from the powerline. The capacitor filters low frequencies 50/60Hz and lets the high (transmit) frequencies pass. The capacitor is a class X2 capacitor. These capacitors have a short circuit protection, which is absolutely necessary,because in case of a short circuit in the capacitor, the 50/60Hz filtering is lost, and the powerline interface will be destroyed, or might be dangerous for persons working with the interface and the ST7536.

As a final protection against any possible spikes, a transil (TRL1) is used. It is a 6.8V bipolar type. If a voltage 6.8V appears, the transil will act as a short circuit to ground, protecting the other parts of the powerline interface from damage.

R12 is added to discharge C12 after disconnecting the interface from the powerline. Without this resistor, C12 will not be discharged and shock hazard might occur if someone touches the powerline connector. This resistor is only usefull in evaluation systems. In all other cases when disconnecting from the powerline never takes place R12 can be removed.

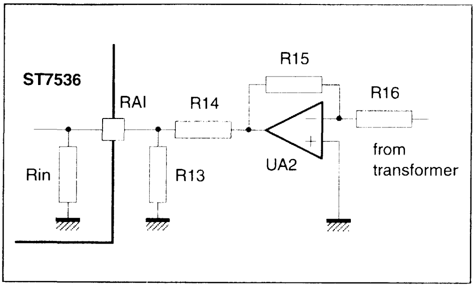
Figure 14



IV.4 - Preamplifier

Receive signals on the powerline are extracted by the transformer and (pre)amplified before sending them to the Receive Analog Input (RAI) of the ST7536. This is done to have, according to the specifications, a receive sensitivity of 1.5mV$_{RMS}$. The sensitivity of the ST7536 is 2mV$_{RMS}$ for channel 1 & 2, and 3mV$_{RMS}$ for channel 3 & 4. To increase the sensitivity the received signal is filtered in the transformer, and then amplified with a gain of 40dB. A limiter is used to protect the ST7536 against signals > 2V$_{RMS}$.

Figure 15



In receive mode the power amplifier is virtually disconnected from the power supply, in order to avoid its low output impedance attenuating the received signals. Signals that are extracted from the powerline are filtered in the transformer, in the same way that the transmitted signals.

After filtering, the signals are amplified. This is done with UA2, an inverting amplifier. The gain of this amplifier is set with R15 and R16.

gain = R15 / R16 = 100k / 1k = 100 x = +40dB

The maximum input level at the RAI is 2V$_{RMS}$.Therefore the signals coming from the preamplifier have to be limited to avoid transmodulation to the ST7536. Amplifier UA2 operates with a power supply of -5V and +5V. The maximum output voltage of the amplifier is then ± 4V. With R13 and R14 a simple limiter has been created. The output voltage of this limiter is the voltage over R13. The input resistance of the RAI (Rin) is 100kΩ.

The gain of the limiter
= (R$_{IN}$//R13) / (R$_{IN}$//R13 + R14)
= (100k//47k) / (100k//47k + 47k)
= 0.4 x (= -8dB)

With a maximum output of the amplifier of 4V, the maximum output of the limiter is set at 0.4 x 4V = 1.6V. Strong input signals are clamped by UA2, but tests showed that this clamping has no effect on correct demodulation.

The total gain of the preamplifier is: +40dB + -8dB = +32dB providing the required receive sensitivity of 1.5mV$_{RMS}$.

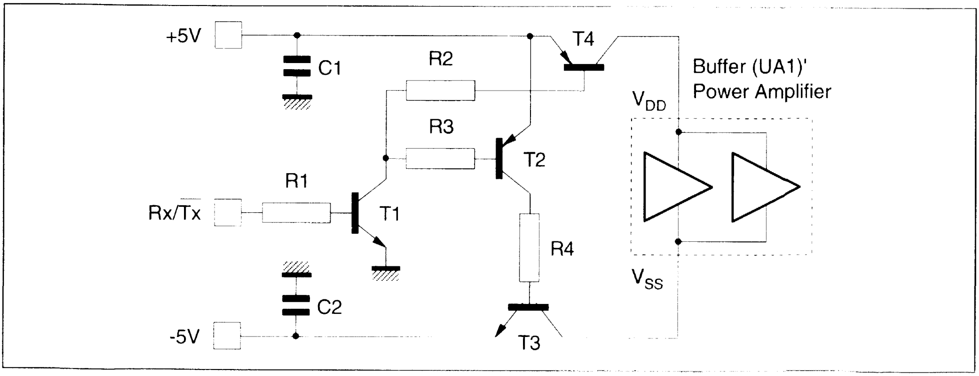
IV.5 - Power on/off Switch

The powerline interface has two operation modes: transmit and receive. Normally the ST7536 system (and therefore the interface) is in received mode, waiting for commands or data requests from the master system.The interface will be used in transmit mode ,only when the system has to respond to the master,

To save energy costs, the buffer and power amplifier in the transmit path are switched off. Also if the interface is used in a master system, which will be often in transmit mode, this switching can be usefull.

A second reason to switch off the transmit power amplifier is the fact that its low output impedance will attenuate the incoming signals in receive mode. Therefore the power amplifier is virtually disconnected from the power supply (see Figure 16).

Figure 16



Switching the positive (V_{DD}) and the negative (Vss) input voltage is done with T3 & T4. If these transistors are switched off the high resistance of the collectors will provide the virtual disconnection. In transmit mode these transistors are switched on, and the voltage lost over the transistors (V_{CE}) will be 0.2V.

The Control of the switch is done with a $Rx/\overline{Tx}$ control line from the controller. In transmit mode this line is +5V, in receive mode 0V. The +5V will open T1, which delivers the base current for T2 and T4. T3 is switched by T2.

In transmit mode the buffer and power amplifier will operate with HF-signals (the transmit signals have frequencies 67...86kHz). Therefore the input (± 5V) of the switching transistors has to be decoupled. This is done with C1 and C2, which have both a value of 100nF.

R1 is 47kΩ, to create a high input impedance. R2, R3 and R4 are 270Ω. T1 and T3 are a 2N2222, T2 and T4 the equivalent pnp version; a 2N2907. These transistors can deliver a maximum current of 0.8A, more than enough for the buffer and power amplifier.

IV.6 - Building-up the Powerline Interface

The whole described parts make a complete powerline interface. The interface has to be connected to the ST7536 as described before.

Because the interface is supposed to operate with the ST7536, the input and output names correspond to the related pin names of the ST7536. for instance: the ATO pin of the ST7536 should be connected to the ATO pin of the powerline interface.

The ATO and RAI are the analog output and input from/to the ST7536. The $Rx/\overline{Tx}$ control input is connected to the controller. The controller switches the interface from transmit mode to receive mode and vice versa. The +/-5 Volt inputs are connected to the power supply connections of the application board. These inputs are HF-decoupled on the board. See also the schematic of the ST7536. If the interface has to operate separated from the application board, using an external power supply, the ± 5V inputs should be decoupled with four capacitors (see Figure 17)

The operation mode of the interface is set with the $Rx/\overline{Tx}$ input line. A high input (+5V) on this line selects the transmit mode, a low input (0V) selects the receive mode. A micro-controller has to be used to control this input.

The 'power line' outputs are the powerline connections. On the application board these connections are located close to C12 and the transformer, to avoid long tracks carrying high voltage.

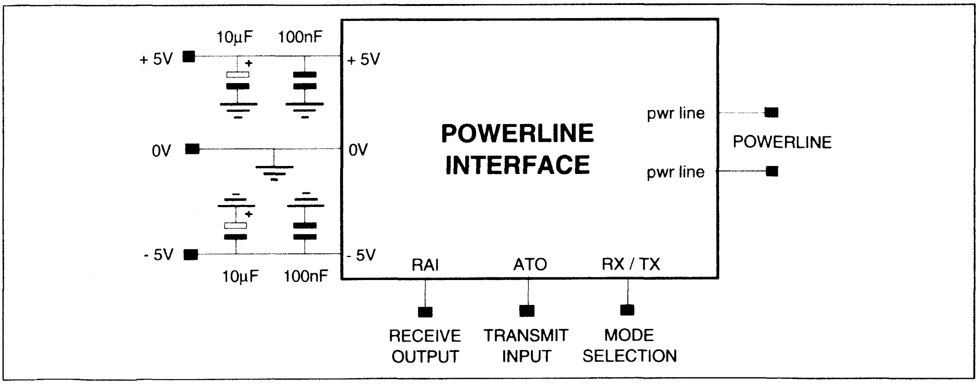
IV.7 - Performances of the Powerline Interface

The following tests have been done on the powerline interface :

- power consumption
- transmit output
- receive sensitivity

All the tests are done with the powerline interface connected to the ST7536.

Figure 17



IV.7.1 - Power consumption

The power consumption is measured both in transmit and receive mode.

In both modes the powerline has been simulated with a 5Ω resistor (worse case simulation). In transmit mode the data input (TxD) was a logical 0 (0V).

The results remained the same for the four channels.

The current consumption :

The input voltage :	- 5.00V, + 5.00V
transmit mode :	- 150mA$_{RMS}$, + 180mA$_{RMS}$
receive mode :	- 1mA$_{RMS}$, + 1mA$_{RMS}$

The power consumption :

transmit mode :	-5V x -150mA + +5V x +180mA 0.75W + 0.9W = 1.65W
receive mode :	-5V x -1mA + +5V x +1mA 0.005W + 0.005W = 0.01W (= 10mW)

In transmit mode the powerline interface delivered 0.340 W into a 5Ω load. With an input of 1.65 W the efficiency is 20%. This does not imply a waste of energy. A ST7536 system is almost allways in receive mode, and the lost of energy is consequently limited.

In receive mode the buffer and the power amplifier are switched off. Power is only consumed by the preamplifier. This explanes the low power consumption in receive mode.

Test equipment : Keithley 165 Multimeter
Test conditions : T = +25°C

IV.7.2 - Transmit output

The transmit output of the powerline interface is measured with the powerline simulated by resistors.

The interface is tested on the four channels. On each channel the ST7536 uses two signals : one for TxD = 1 (lower freq.) and one or TxD = 0 (higher freq.) Therefore the output on each channel is measured for TxD = 1 and TxD = 0. This makes 4 (channels) x 2 (TxD 0/1) = 8 signals to test.

The powerline is simulated with resistors. Six different impedances are tested : R = 0.5, 1, 5, 10, 50, 100Ω.

A spectrum analyzer is used to test the output of the powerline interface It measures the output power and generates a frequency spectrum plot. With this plot the harmonic distortion can be calculated (see Figure 18)

Test results. (see ANNEXE B)

With the spectrum analyzer the output power on the transmit frequency (H1) is measured. Then the power of the harmonics is measured. The difference between those two signals is the harmonic distortion.

Example : TxD = CHS = BRS = 0
(channel 1, txd = 0 → 81.75kHz.)
R powerline = 5Ω.

H1 : f= 81.75kHz, measured power = +15.2dBm.
H2 : f = 163.5kHz, measured power = -58.8dBm.

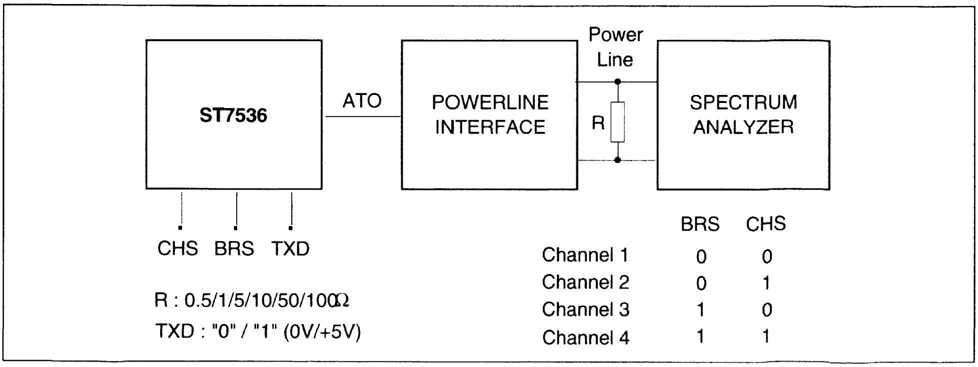
The difference between H1 and H2 is + 15.2dB - -58.8dB = 74dB.

The second harmonic of the signal is in this case suppressed to a level of -74dB (compared to H1).

The measured output power of H1 = +15.2dBm. Then the output voltage can be calculated.

0 dBm is 1mW power into a resistor of 50Ω. So +15.2 dBm is 33mW power into a resistor of 50Ω. Vout(rms) is therefore (33mW * 50Ω)^0.5. In this case the output voltage is 1.29V$_{RMS}$.

Figure 18



R : 0.5/1/5/10/50/100Ω
TXD : "0" / "1" (0V/+5V)

	BRS	CHS
Channel 1	0	0
Channel 2	0	1
Channel 3	1	0
Channel 4	1	1

The output current is also calculated :
Iout(rms) = Vout(rms) / R.
For example; the output voltage with R powerline = 0.5Ω is 0.18V_{RMS}. Then the output current is 360mA_{RMS}.

Channel 1 : 82kHz

Rline (Ω)	Output (V_{RMS}) TxD = 1	Output (V_{RMS}) TxD = 0	H2 (dB) TxD = 1	H2 (dB) TxD = 0
0.5	0.18	0.18	-51	-54
1	0.31	0.31	-51	-55
5	1.29	1.29	-74	-76
10	1.70	1.68	-77	-81
50	2.06	2.02	-74	-77
100	2.09	2.06	-74	-76

Channel 2 : 67kHz

Rline (Ω)	Output (V_{RMS}) TxD = 1	Output (V_{RMS}) TxD = 0	H2 (dB) TxD = 1	H2 (dB) TxD = 0
0.5	0.16	0.16	-67	-68
1	0.28	0.28	-68	-68
5	1.21	1.20	-75	-75
10	1.70	1.65	-78	-75
50	2.16	2.11	-83	-75
100	2.21	2.16	-84	-75

Channel 3 : 72kHz

Rline (Ω)	Output (V_{RMS}) TxD = 1	Output (V_{RMS}) TxD = 0	H2 (dB) TxD = 1	H2 (dB) TxD = 0
0.5	0.16	0.16	-65	-67
1	0.28	0.28	-66	-67
5	1.17	1.16	-73	-76
10	1.62	1.56	-75	-79
50	2.02	1.95	-74	-75
100	2.06	2.00	-73	-75

Channel 4 : 86kHz

Rline (Ω)	Output (V_{RMS}) TxD = 1	Output (V_{RMS}) TxD = 0	H2 (dB) TxD = 1	H2 (dB) TxD = 0
0.5	0.17	0.17	-56	-52
1	0.29	0.30	-60	-57
5	1.21	1.18	-77	-77
10	1.55	1.53	-82	-82
50	1.86	1.80	-82	-78
100	1.88	1.84	-80	-78

Summary of the test results :

Channel 1 :

R < 5Ω	:	310-360mA_{RMS}
R > 5Ω	:	1.3 - 2.1v_{RMS}
R 10/50Ω	:	H2 < -74dB

Channel 2 :

R < 5Ω	:	280-320 mA_{RMS}
R > 5Ω	:	1.2 - 2.2 V_{RMS}
R 10/50Ω	:	H2 < -75dB

Channel 3 :

R < 5Ω	:	280-320mA_{RMS}
R > 5Ω	:	1.2 - 2.0V_{RMS}
R 10/50Ω	:	H2 < -74dB

Channel 3 :

R < 5Ω	:	290-340mA_{RMS}
R > 5Ω	:	1.2 - 1.9V_{RMS}
R 10/50Ω	:	H2 < -78dB

With impedances < 5Ω the output current is for all the four channels in the range 280-360mA_{RMS}.

The output voltage on impedances > 5Ω is both on channel 3 and 4 in the range 1.2 - 2.0V_{RMS}. On channel 1 and 2 it's in the range 1.2 - 2.2V_{RMS}.

On all the channels the second harmonic of the signals is < -74dB, on channel 4 the second harmonic is even < -78dB.

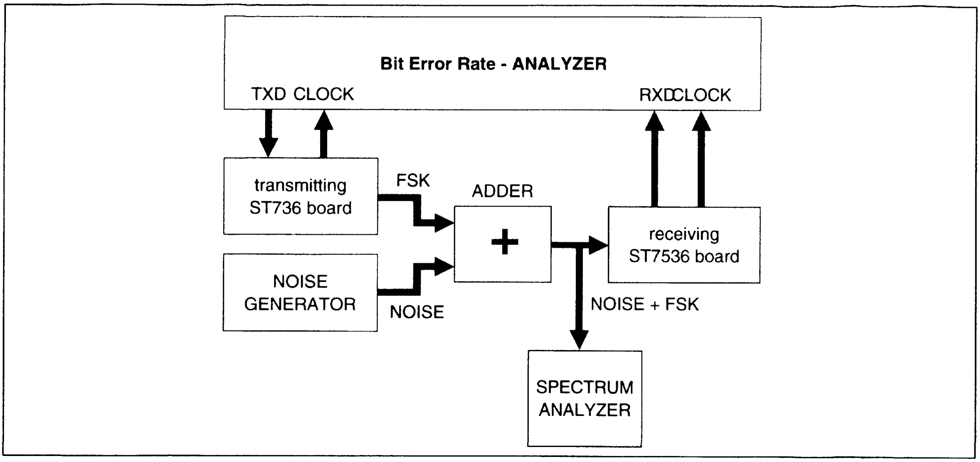
IV.7.3 - Receive sensitivity

The receive sensitivity of the powerline interface is measured with a Bit Error Rate (B.E.R.) test. The Bit Error Rate is the amount of wrong bits in a received bit pattern. For example, if 2 out of 1000 received bits is wrong detected, the B.E.R. is 2/1000 = 2 E-3. If the B.E.R. with an input of 1mV_{RMS} is worse than with an input of 5mV_{RMS}, the receive sensitivity is not 1mV_{RMS} but 5mV_{RMS} (or more).

Test configuration
In this test two ST7536 boards are used. Each board has a ST7536 + powerline interface.

One board is in transmit mode, the other in receive mode. A Bit Error Rate Analyzer is used to generate bit patterns, and to compare these patterns with the receive patterns. Because the ST7536 is a synchronous modem, both the received data (RxD) and transmitted data (TxD) are related to the clock signal generated by the ST7536. Therefore the clock signals of the boards are delivered to the analyzer (see Figure 19).

Figure 19



The output of the transmitting board is a Frequency Shift Keying (FSK) signal. This signal is added with the signal from a noise generator. This to observe the B.E.R. under different Signal/Noise-ratio conditions. In the adder the FSK signal is attenuated to a level of 0.5 - 5mV$_{RMS}$. The output signal is then send to the receiving board.

A spectrum analyzer is used to measure all the signals.

Measurements

Two tests are done on channel 3 (72kHz/1200 baud).

First B.E.R. test is made with a FSK input of 1mV$_{RMS}$ (= -60dBV). With the noise level set at -68....-74dBV. This gives a S/N ratio from 8....14dB.

A second test is done with a FSK input of 5mV$_{RMS}$ (= -46dBV). With the noise level set at -54...-60dBV. These values are set by adjusting the mixer, and measured with the spectrum analyzer.

The spectrum analyzer measurements are made in a spectrum of 1200Hz. This is done because the FSK signal has two main frequencies on 1200 Hz distance from each other. The noise signal is therefore measured in this band. In annexe B example plot are given from all the tests, with a S/N ratio of 10dB.

The B.E.R. is calculated from the number of errors counted by the B.E.R. analyzer.

Example :

For instance the bit rate is 1200 baud. In 10 minutes the analyzer counted 800 errors. The measure time is then 10 x 60 seconds is 600 seconds. Each second 1200 bits are transfered, so in 600 seconds 720000 bits. Then the bit error rate is 800/720000 = 1.1 E-3.

→ B.E.R. = number of errors / (time in seconds x bit rate)

Test results (see ANNEXE A)

The results of the B.E.R. test are almost the same for both 1mV$_{RMS}$ and 5mV$_{RMS}$ (FSK signal level) input. Compared to B.E.R. test results of a stand alone ST7536, the results are even 1dB (S/N ratio) better.

These results demonstrate that the receive sensitivity is at least 1mV$_{RMS}$, and therefore the other channels are tested with an input of 1mV$_{RMS}$.

Channel 1, 2 and 4 are tested with a FSK signal input of 1mV$_{RMS}$. On those channels the results are also compared to B.E.R. test results of a stand alone ST7536. On channel 2 there is no difference between the B.E.R. of a stand alone ST7536 and a ST7536 + powerline interface. On channel 1 and 4 the B.E.R. is 0.5dB better than a stand alone ST7536.

Typical B.E.R. : (input FSK 1mV)

S/N (dB)	Channel 1	Channel 2	Channel 3	Channel 4
8	1.2E-2	1.2E-2	2.0E-2	1.0E-2
10	2.0E-3	2.0E-3	4.5E-3	1.2E-3
12	1.0E-4	1.0E-4	4.5E-4	1.0E-4
14	4.0E-6	4.0E-6	4.5E-5	3.0E-6

IV.7.4 - Conclusions

The B.E.R. tests confirm a receive sensitivity of 1mV$_{RMS}$. This is according to the specifications under which the powerline interface has to operate. Moreover, the B.E.R. tests showed that the powerline interface improved the performances of the ST7536 ; the results of a ST7536 in combination with the powerline interface are better than a stand alone ST7536.

Remark :
To test if these results are not only valid for a laboratory set up, both boards have been connected to the 220V powerline network. The distance between the two boards was 30 meter. After a measure period of 15 minutes, not even 1 error was detected !

Test equipment : (see Figure 20)
Shlumberger SI 7703B B.E.R. analyzer
HP3562A Spectrum analyzer
Rohde & Schwarz SUF2 Noise generator
Mixer :

V - HEATING CONTROL APPLICATION

V.1 - Introduction

We will do a heating control system, using the ST7536 and a ST6 micro controller.

We have two boards (see Figure 21) :
- MASTER : control and set temperature in each room
- SLAVE : temperature reading,switch on/off of heater.

Figure 20

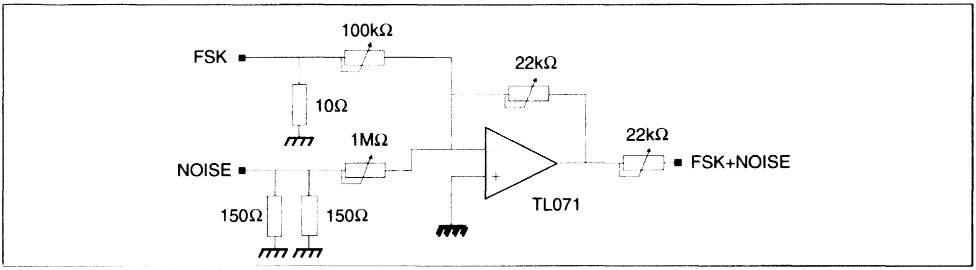
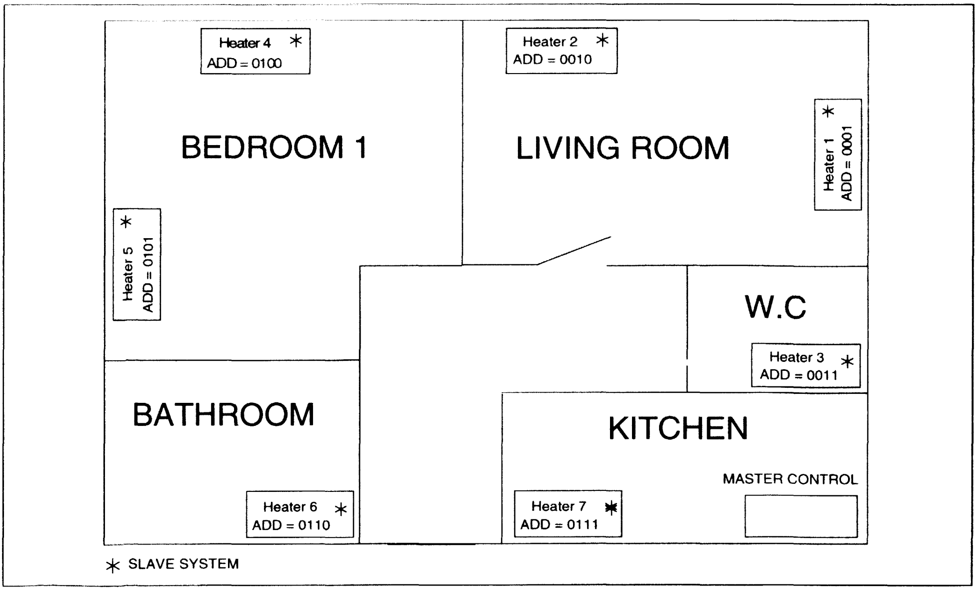


Figure 21

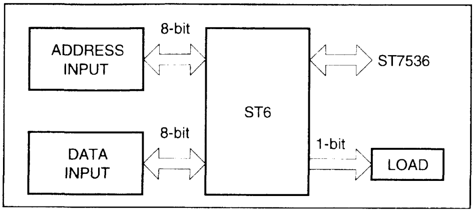


V.2 - Micro-controllers

Two different micro-controllers have been set up, one for the slave systems, and one for the master system. The main differences between the two controllers are the different input/output facilities.

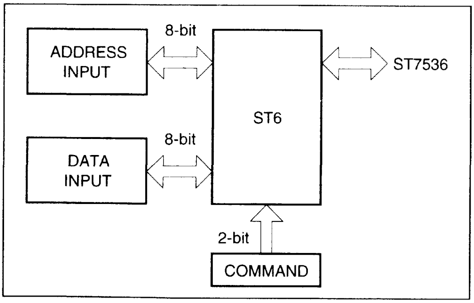
The slave version needs one 8-bit data input to initialize its own address, and one 8-bit input to read data from an external measure system. It should also provide an output that switches a load. This load will be simulated by a LED.

Figure 22 : Slave Micro-controller



The master version will have its own address initialized in the software. Therefore no data input is needed for that. Data input (8-bit) is needed to read the destination (slave) address. To display data, an 8-bit data output has to be provided. Furthermore, it needs a 2-bit command input.

Figure 23 : Master Micro-controller



Both the master and the slave version need also data exchange with the ST7536; the clock, transmit data, receive data, reset and Rx/$\overline{\text{Tx}}$ control lines.

V.3 - Hardware

As a micro-controller the ST6 has been chosen. This controller provides 20 data input/output pins, a reset and a non maskable interrupt input. Only a few external components have to be added to this micro-controller for full operation. The used ST6 is a 2K byte program memory EPROM version; the ST62E15. The ST6 has an internal oscillator circuit. One machine cycle takes 13 oscillator pulses. This means that with a clock frequency of 8MHz a machine cycle takes 1.625µs. Most of the instructions (load instructions, bit manipulations) take 4 machine cycles. The maximum bitrate the ST6 has to serve is 1200 baud. One bit has a lenght of 833µs, which is equal to 512 machine cycles. This means that during each bit about 130 instructions can be executed.

The ST6 has an on chip watchdog circuit. There are two different versions of the ST62E15. On one there is a software selectable watchdog, and on the other (the hardware version) this watchdog is allways activated. The version that is used for these micro-controllers is the hardware version.

V.4 - Slave

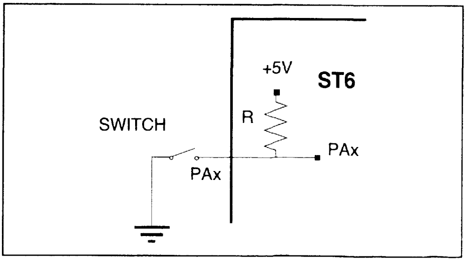
This micro-controller is in fact just an ST6 with a very few external components. A few switches, resistors, capacitors, a crystal and a 74LS04 are connected to have a complete controller. Each of these components is used to set the ST6 in the correct configuration.

Pin configuration slave controller

For each pin a short discription is given, such as the configuration chosen for this micro-controller.

- **pin 27..20 (PA0..PA7) :** Input/output port A. Port A of the ST6 is used for reading the (8-bit) home address of the slave system. Switches are used to set each bit. The ST6 provides an internal pull-up resistor which will cause an "1". Closing a switch (to 0V) will cause a "0".

Figure 24

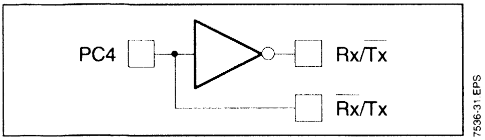


- **pin 19..12 (PB0..PB7) :** Input/output port B. Port B of the ST6 is used to read (8-bit) data from an external measure system. This system is simulated by switches. The same as for port A, the ST6 provides an internal pull-up resistor which will cause an "1". Closing a switch (to 0V) will cause a "0".
- **pin 9 (PC4) :** Port C bit4. This output is used as the Rx/$\overline{\text{Tx}}$ control. Transmit mode of the ST7536 and the powerline interface is selected if this output is "1". Receive mode is

selected with an "0".

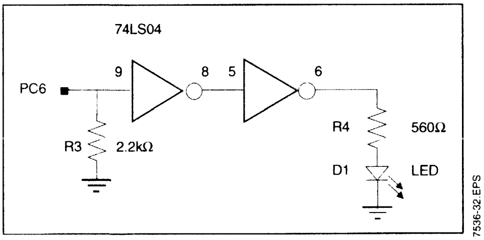
- **pin 8 (PC5) :** Port C bit5.
  This pin is used as the transmit data (TxD) output to the ST7536.

Figure 25



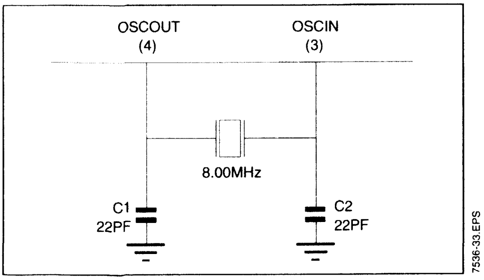
- **pin 6 (PC7) :** Port C bit7.
  This pin is used as the receive data input (RxD) from the ST7536.
- **pin 7 (PC6) :** Port C bit6.
  This is the load switching output. The load is simulated by a LED. Two inverters are used as a buffer between the ST6 and the LED. Because the ST6 outputs are in high impedance during a reset, a pull down resistor (R3/2k2) is used to avoid the load switching on.

Figure 26



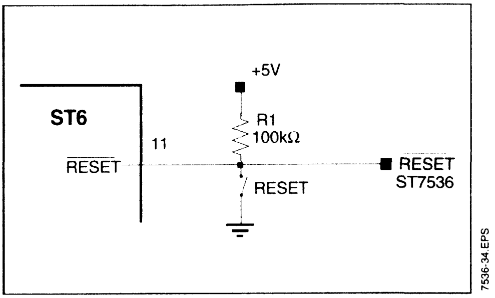
- **pin 4 (OSCOUT) :** Oscillator output.
- **pin 3 (OSCIN) :** Oscillator input.
  Between these pins a 8.00MHz crystal has to be connected. If the internal oscillator of the ST6 runs at 8MHz, one machine cycle is 1.625µs. This speed is needed to be able to serve the 1200 baud bitrate from the ST7536.

Figure 27



- **pin 5 (NMI) :** Non maskable interrupt.
  The NMI is used as input of the (inverted) clock of the ST7536. The NMI is falling edge sensitive. An external pull-up resistor (R2/100k) is added to provide +5V for debugging the controller without the 74LS04 (the inverter).
- **pin 11 (RESET) :** Reset input.
  The reset of the ST6 is active low. To restart the microcontroller at the beginning of its program, this pin should be set to 0V by closing the switch. For normal operation the +5V is provided by a pull-up resistor (R1/100k).

Figure 28



- **pin 2 (TIMER) :** Timer output, not used.
- **pin 10 (TEST) :** Test input.
  The test pin is used to set the ST6 in a special operation mode. For normal operation this pin is set at 0V.
- **pin 1 (V_{DD}) :** Power supply, +5V.
- **pin 28 (V_{SS}) :** Ground, 0V.

V.5 - Master

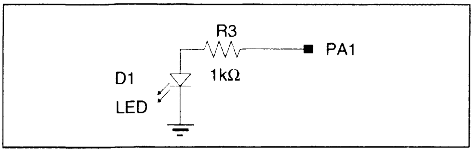
The main difference between the master and the slave version of the micro-controller is the fact that the master needs one extra input/output pin. The slave version has 1 output to control a load, where the master needs 2 inputs to read a command. Therefore one input/output (PC5) has been multiplexed, it serves both the RxD and TxD lines to the ST7536.

Pin configuration master controller

The pin configuration of the master differs from the slave on the next pins:

- **pin 27..20 (PA0..PA7) :** Port A.
  The master uses port A to display data. Light Emitting Diodes (LED's) are used to do this. The maximum current that can be taken from each pin is 5mA. Therefore the serial resistor has a value of 1kΩ (current = U/R = 5-0.6/1k = 4.4mA).

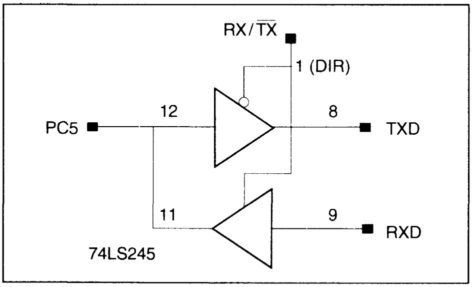
Figure 29



7536-35.EPS

- **pin 19..12 (PB0..PB7)** : Port B.
  The hardware configuration of these pins is the same as on the slave, but on the master these pins are used to read a (destination) address.
- **pin 7/6 (PC6/7)** : Port C bit6/bit7.
  On the master these pins are used to read a command (see also the software discription). The hardware configuration is the same as for port B.
- **pin 9 (PC5)** : Port C bit5.
  The slave uses PC7 as received data input (RxD). Because the master already uses this pin for reading a command, PC5 has to be multiplexed. This is done with an 74LS245. It is a (8-bit) bus receiver/transceiver. The Rx/Tx line is used to select whether the RxD should be send to PC5, or the data from PC5 to the TxD (that's the opposite direction).

Figure 30



7536-36.EPS

V.6 - Software

The software that has been developed for the micro-controllers has to be regarded as an introduction to more complex communication protocols. Therefore a very simple but effective protocol has been set up. With this protocol it should be possible to evaluate the performances and possible applications of a ST7536 system.

V.7 - Protocol

The protocol has been set up in such a way that all kind of features can be added easily. A simple but powerfull frame format is used. It gives the possibility to use error correction and detection.

Each frame consists of a preamble, a system address, a destination address, a control block and a data block. The preamble and the system address length is 2 bytes, the destination address, the control block and the data block are 3 bytes long.

The preamble is used to train both the transmitting and receiving ST7536. It consists of two 8-bit patterns (10101010). The receiving ST7536 needs it to train its clock recovering. Because the 3 first bits transmitted by an ST7536 are not guaranteed to be correct, the preamble is also used to overcome unreliable data in the beginning of a transmission. This because the preamble doesn't contain data.

The system address is used to be able to have more than one ST7536 system operating on a certain powerline network. For example a remote metering system and a traffic light control system. It is also used to avoid interference with other (no ST7536) systems. The lenght of the address is only 8 bits, and therefore it's send twice, to avoid unwanted activation of a group that has not been called.

The lenght of the preamble and system address together is 4 bytes (32 bits) (see Figure 31)

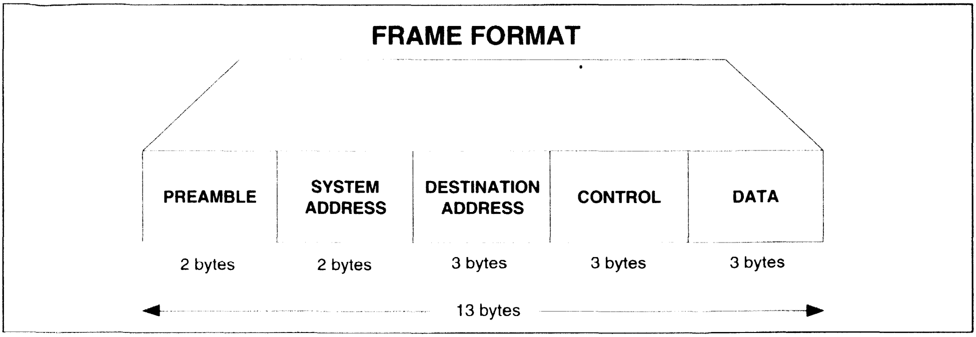
The preamble and the system address inside the frame (see Figure 32).

The received destination address, control block and data block should be very reliable, and therefore an error correction is done. To be able to do this all these data is send 3 times. The destination address has a lenght of 1 byte (8 bits), which is send 3 times: in block 1, block 2, and block 3. This is the same for the control byte and the data byte. As an example the destination address inside the frame (see Figure 33)

So all the blocks (block 1/2/3) contain the same byte. The error correction uses them to extract the correct byte out of the 3 that have been received. The destination address is used to select 1 user (slave) in a system group. All the slaves in a system have their own 'home' address. To activate a slave, it has to recognize the received destination address inside a frame as its own 'home' address.

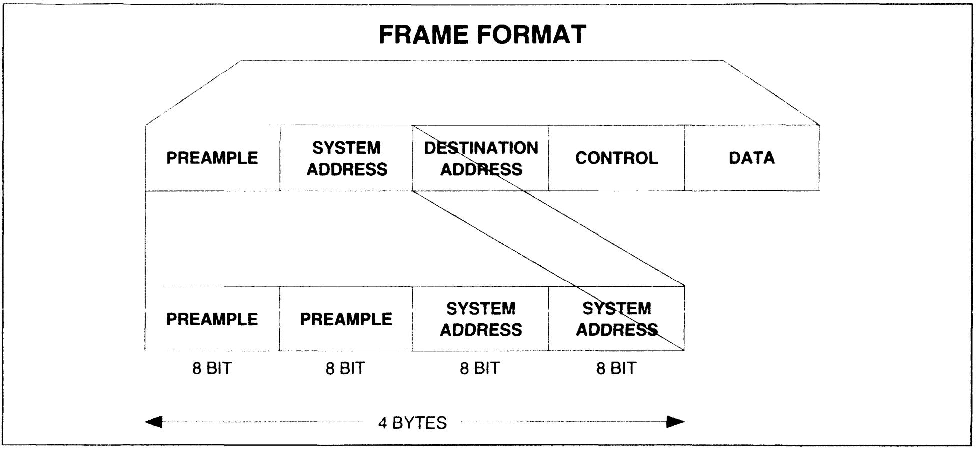
In this simple protocol there is only communication between a master and the slaves. Therefore the destination address is transmitted by each slave is the master address. In the frame which transmitted by the master, the destination address is the home address of the slave that is called.

Figure 31



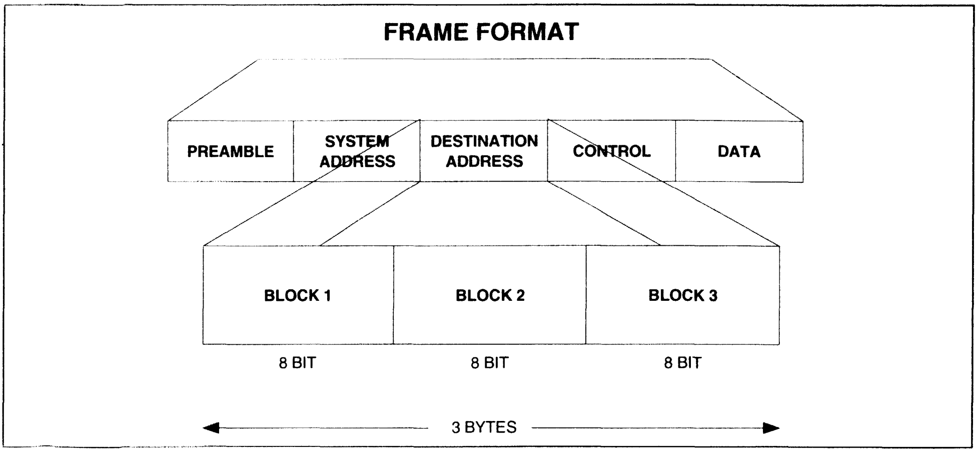
7536-37 EPS

Figure 32



7536-38 EPS

Figure 33



7536-39 EPS

The control byte can be used for all kind of information about the frame. In this protocol the control block is only used to say if the frame is a command or a response. This is done with bit 7. If this bit is set it means that the data byte contains a command (from the master to the slave), and the data byte will contain the command. If this bit is reset it means that the frame is a response (from the slave to the master), and the data byte will contain the requested data.

Control Byte	Status	Data Byte
10000000	command	command
00000000	response	requested data

In the control byte only bit 7 is used. Bit 0...6 are reset. They can be used to add severall features to the protocol.

Error detection and correction

A possible feature that can be added is error detection. In the protocol this feature is not available. This because, to be really usefull, error detection would require the possibility to send a message from the receiver to the transmitter, indicating that an error has been detected. It will need a more detailed protocol, which uses the free bits in the control byte. The intention of this protocol was to be very simple and clear. Therefore the error detection is not provided.

Although there is no error detection, the protocol provides an error correction. It would be very unrealistic to assume that all the bits in a received frame are correct. Therefore the most important parts of the frame (destination address, control byte and data byte) are protected with an error correction.

The error correction is done with bit-overlay. This is a very powerfull method to correct bytes that are transmitted over very noise lines. Each byte is transmitted (and received) 3 times. The software uses the 3 received bytes to extract the (probably) correct byte. This is done by performing a bitwise majority decision on all the received blocks.

Even if all the 3 received blocks contain errors, it's still possible to extract the correct byte out of these blocks.
example :
The first received block contains 3 errors (b6/b4 and b1), the second block contains 2 errors (b7 and b3) and the third block contains 3 errors (b5/b2 and b0) (see Figure 34).

The error corrector will take bitwise a decision what is probably the correct bit. If two out of three bits are "1", the resulting bit will be "1". If two out of three bits are "0", the resulting bit will be "0".

This system can correct 1 error out of 3 bits. If more blocks are send, let's say 9, it would be possible to correct 4 out of 9 bits. This is a very interesting method to overcome problems on very noisy powerlines.

Figure 34

ERROR CORRECTION

b7	6	5	4	3	2	1	0	
0	0	1	0	1	0	1	1	transmitted byte
0	1	1	1	1	0	0	1	received block 1
1	0	1	0	0	0	1	1	received block 2
0	0	0	0	1	1	1	0	received block 3
0	0	1	0	1	0	1	1	corrected byte

7536-40.EPS

V.8 - Application software

The protocol has been designed to demonstrate typical applications of the ST7536.

All the slaves have been programmed with one program. With this program it is possible to set a load (simulated by a LED) on or off. This load can be for example (depending on the application) a traffic light. With this program it's also possible to read data (simulated by 8 switches) from an external measure system, and send this data to the master if requested. The measure system, for example, can be reading an electricity meter. Remote reading these meters, can save the costs for manual reading (such a system is allready operational in Italy). So there has been written 1 program for the slaves, that can simulate different applications.

Each typical application should be programmed in the master. It gives the possibilty to demonstrate different applications without reprogramming all the slaves.

One application program has been developed for the master.It demonstrates the good functioning of the system: a remote heater control. With this application it's possible to control in a building in each room the heater (which is equipped with an ST7536). The LED on the slave simulates in this case the heater. With the master each heater can be set manual on or off, and even more, the master can regulate automatically the heater, by reading out the room temperature. This temperature is simulated with the 8 switches on the slave. The master has 4 different commands:

1:(00)	manual off	00000000	heater on
2:(01)	manual on Temperature :	00001111	
3:(10)	not used	00010000	
4:(11)	automatic control →	11111111	heater off

V.8.1 - ST6 programs

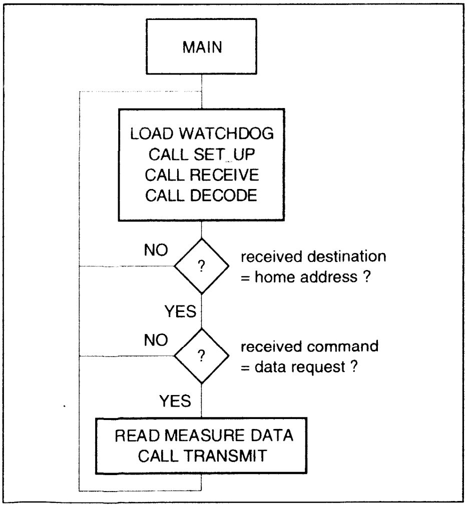
The programs for both the master and the slave have been written in assembly language. An assembler is used to create the executable code. A special ST6 kit is used to debug the programs. The EPROMs are also programmed with this kit. The program memory size of the ST62E15 is 2K byte. The slave program is 1.6K byte, the application program for the master 1.8K byte long.

Some subroutines are used in both the master and the slave program, like the transmit/receive subroutines and the error decoding.

In annexe C the slave program is given, and in annexe D the application program for the master. The register declaration file is given in annexe E. From both programs the most important subroutines are described on the next pages. The flow charts that are used do not give a detailed representation of the subroutines, but are used to explain the structure of the subroutine.

V.8.1.1 - Slave program

Figure 35



In the main program first of all the watchdog is (re)loaded. The watchdog is a down-counter that generates a reset when it's not in time reloaded. It provides a recovery from a software upset.

Then the home address is read from the switches in the set-up routine. The slave will go in receive mode and be maintened until a complete frame is received.

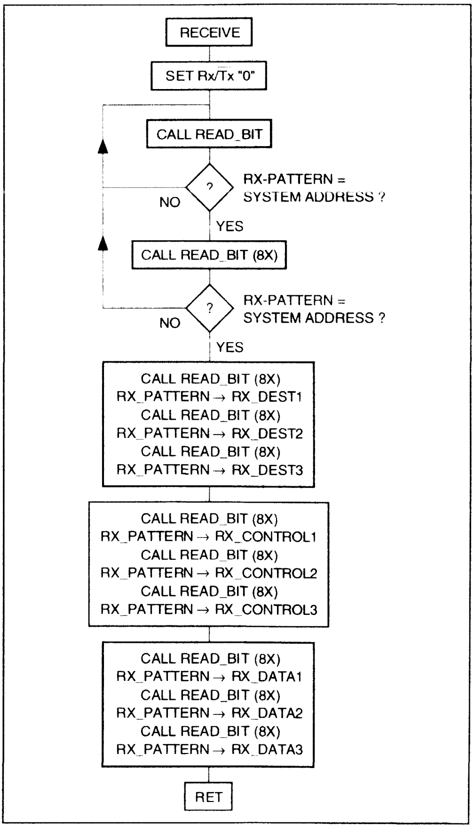
After that the contents of this frame is decoded. In the decode subroutine the bytes are corrected and depending on the received command the led is set on or off.

If the received destination address was the slaves home address and the received command was a data request the (simulated) measure data is read and then transmitted to the master.

Receive subroutine

The receive subroutine is used to read a frame. It ignores all the RxD until the system address is received. When it is received for the second time, the next bytes of the frame are read and stored. To read the RxD this subroutine uses the read_bit subroutine which is discribed on the next page.

Figure 36



First of all the Rx/$\overline{\text{Tx}}$ line is reset. The ST7536 and the powerline interface will then be in receive mode. Then the read_bit subroutine is called, whlch will add the next received bit to the Rx_pattern. As long as the system address is not received, the programs continue read the RxD.

When the system address is received, the next 8 bits will be loaded using the read_bit subroutine, and after that the Rx_pattern should contain again the system address. If this is not the case this procedure starts again. Else the next bytes will be read and stored.

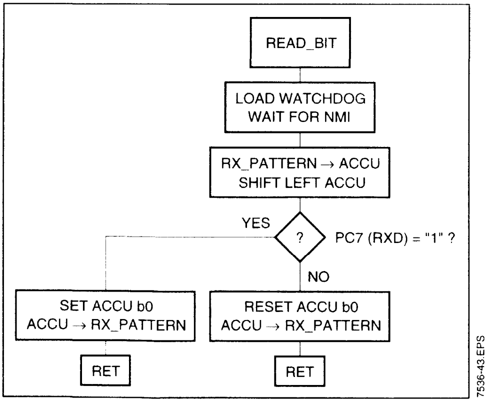
Flrst the 3 destlnatlon addresses. The read_bit subroutine is called 8 times and the Rx_pattern will then contain the next byte (Rx_dest1) which is stored. The next 2 bytes (Rx_dest2, Rx_dest3) are read in the same way.

When the destination addresses are received the control bytes and the data bytes are received and stored in the same way.

Read-bit routine

This routine is used to read the RxD is presented on PC7 (for the master on PC5). The ST7536 delivers valid data on the positive edge of its clock. The inverted clock is used as the NMI input of the ST6. This NMI is falling edge sensitive - > an NMI will be generated on the positive edge of the ST7536 clock. This means that the RxD should be read immediately after a NMI interrupt. The received bit is then added to the (Rx_)pattern.

Figure 37



First of all the watchdog is reloaded.

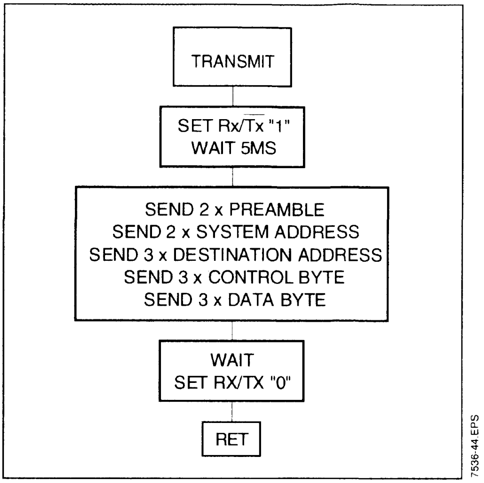
Then the ST6 will wait for the NMI (interrupt). The Rx_pattern is loaded into the accumulator and then shifted left. If the received data bit is a "1" the next bit (b0 in the accumulator) is set to "1". If the received bit is a "0", this bit will be set to "0".

At the end the new pattern is stored.

Transmit subroutine

This subroutine uses the send 8-bit subroutine to send a 8-bit pattern. This subroutine is discribed on the next page.

Figure 38



In the transmit subroutine first off all the Rx/$\overline{\text{Tx}}$ line is set "1". The ST7536 and the powerline interface are then in transmit mode. Typical carrier stabilisation time of the ST7536 is 5ms. Therefore the program waits this time before sending all the bytes.

First the preamble (10101010) is send 2 times and then the system address. The destination address, the control byte and the data byte are send 3 time. This is done with the send 8-bit subroutine.

Send 8bit pattern subroutine

The ST7536 samples the TxD on the positive edge of the clock. The inverted clock is used as the NMI input of the ST6. On the positive edge of the clock (CLR/T) an NMI interrupt occurs. The software waits then 304 machine cycles before changing the TxD. When the TxD is changed it waits for the next NMI and again 304 cycles before storing the next TxD on PC5. Using these delays it is possible to present the ST7536 valid TxD on the positive edge of its clock. Both 600 (channel 1/2) and 1200 (channel 3/4) baud bitrates can be handled this way.

Figure 39

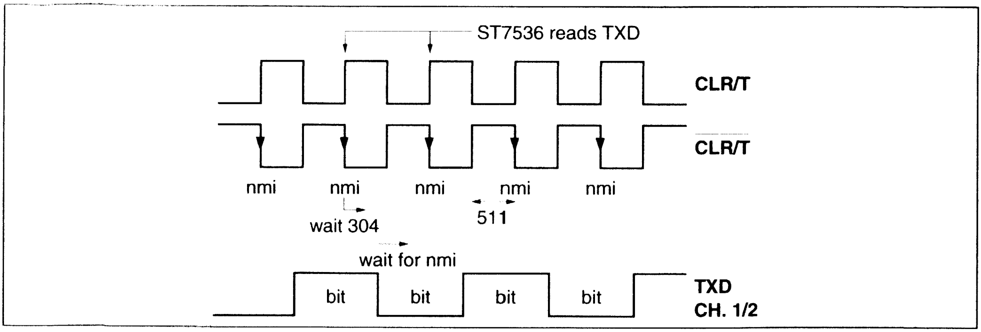


Figure 40

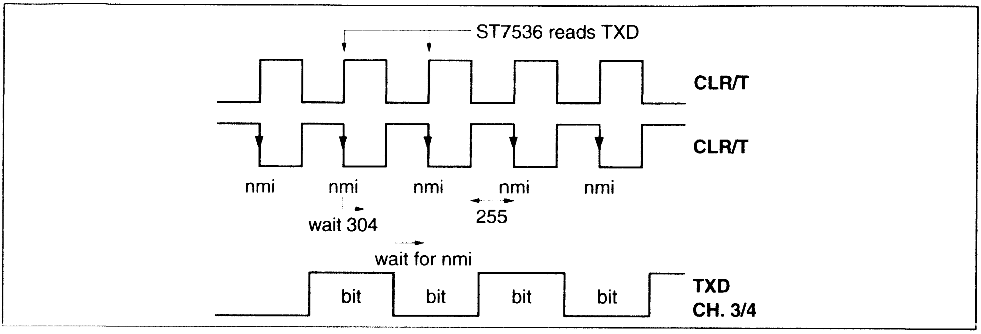
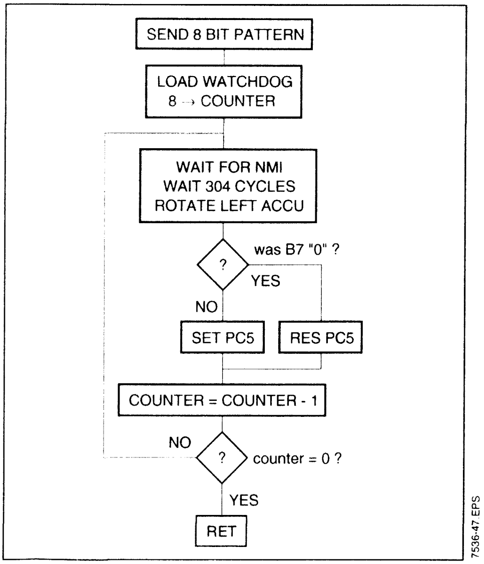


Figure 41



In the subroutine the watchdog is (re)loaded. Then the counter is set at 8. The ST6 waits for the NMI and then 304 cycles. The accumulator contains the pattern that has to be transmitted. This pattern is shift left; the carry bit will contain b7, b7 will contain b6 etc. If b7 was a "0" PC5 (the TxD output of the ST6) is reset, if it was a ' 1 ' PC5 is set. Bit7 is then transmitted.

The counter is then decreased by 1, the software waits the same time, and the accumulator is shift left again. The carry bit will then contain b6 of the pattern that should be transmitted. The same as before, this bit is transferred to the TxD output.

This is also done with b5..b0. If b0 is transferred(transmitted) the counter will be 0. And then the ST6 will jump out of the subroutine.

Decode subroutine

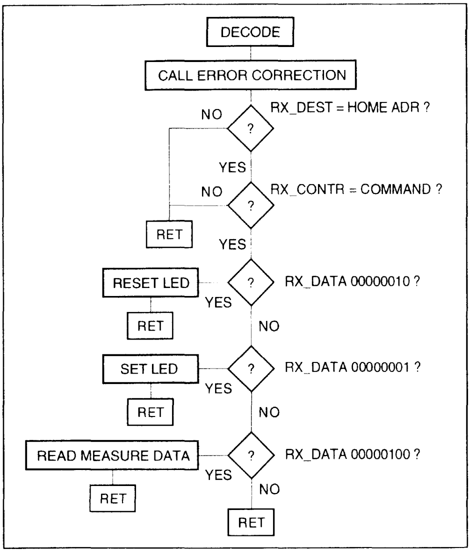
When a frame is received the main program jumps to this subroutine. In this subroutine first of all the error correction is called. After the correction, described below, the subroutine checks if the destination address of the frame was the home address of the slave. In this case, and then the control byte indicates a command in the data byte, the subroutine decodes the data byte. Then there is a jump out of the subroutine (see Figure 42)

Three commands are used in this program :

1: data = 0000 0001	→	LED off.
2: data = 0000 0010	→	LED on.
3: data = 0000 0100	→	data request.

Depending on the received data byte the led will be set on or off, or in case of a data request, the ST6 reads the data on the system.

Figure 42



7536-48.EPS

Error correction subroutine

In this subroutine the bits of the 3 received bytes are compared. The resulting should be set ("1") if at least 2 out of 3 received bit are set. If 2 or more received bits are reset, the resulting bit should be reset ("0"). All the bits of the bytes are tested this way. This is done bit by bit (bit x), using the same procedure (see Figure 43).

V.8.1.2 - Master program

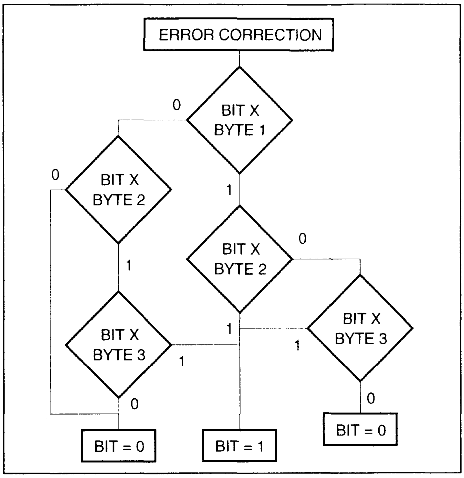
For the master one application program is set up. It presents the ST7536 in a central heating control. The receive, transmit and error correction subroutines are almost the same as described for the slave program. The master has a 2 bit command input (PC6/PC7).

After reading the command it is checked if it's a new command or an old command. The program continues if a new command is read (from the switches). Then this command is decoded :

PC6/7: 00	→	heater off
PC6/7: 01	→	heater on
PC6/7: 10	→	not used
PC6/7: 11	→	automatic control

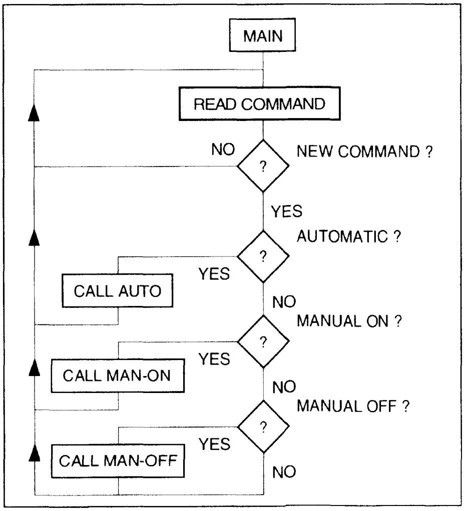
The program will jump to the subroutine corresponding to the command that is decoded; the automatic, manual on or manual off subroutine (see Figure 44).

Figure 43



7536-49.EPS

Figure 44



7536-50.EPS

Automatic control subroutine

In this subroutine the master compares the receive data (temperature) from the slave to the reference value (00001 1 1 1). If the data < reference the heater is set on, else the heater is set off.

First the master sends a datarequest command to the slave. Then it goes in receive mode, to wait for the response of the slave. The received frame is then corrected, and the received destination address and control byte are checked. If the control byte indicates a response the received data is compared to the reference value (00001111). If the received data < the reference, a heater on command will be transmitted, else a heater off command. After that the command input is checked to be still an automatic control command. In this case, the master will continue with controlling the heater (see Figure 45).

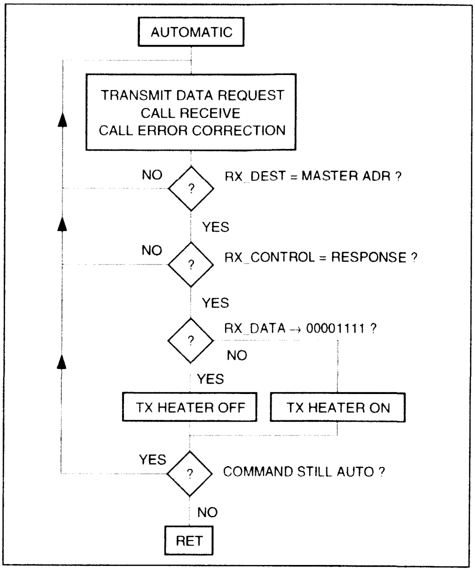
Manual on/off subroutines

Although these routines are very simple, their functions are given in the flow charts below, to have a complete overview of the application program. The manual on subroutine sends a frame containing a heater on command, the manual off routine a frame containing a heater off command (see Figure 46).

V.8.1.3 - Evaluation of the software

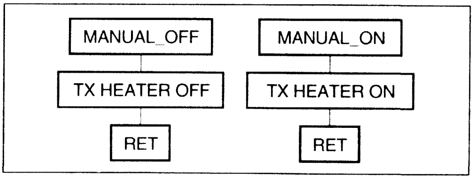
The used protocol seems to be rather effective and functional. The system has been tested on very noise powerline networks, and no (software) problems occured. The ST6 programs have been written and tested step by step. When the programs were operating according to the protocol they have not been 're-written'. Therefore the programs might not be as well structured as possible. For final application software, it might be usefull to evaluate the programs, and then rewrite some subroutines. The program size may be decrease, and a 'clean up' will make the programs easier to understand.

Figure 45



7536-51.EPS

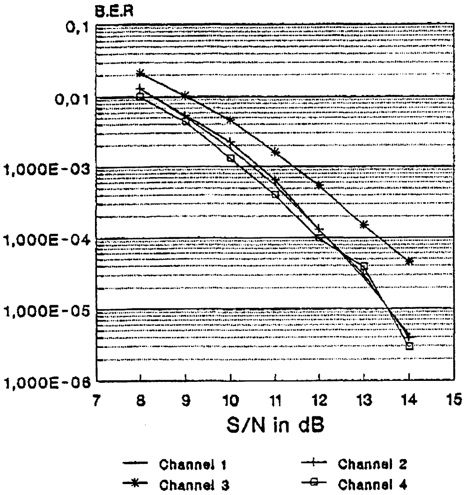
Figure 46



7536-52.EPS

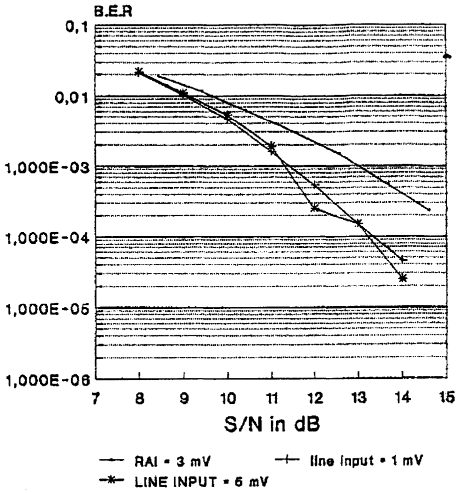
ANNEXE A

Figure 47 : B.E.R. ST7536 Application Board Comparison of channel 1/2/3/4
T = 25°C, Line Input = 1mV



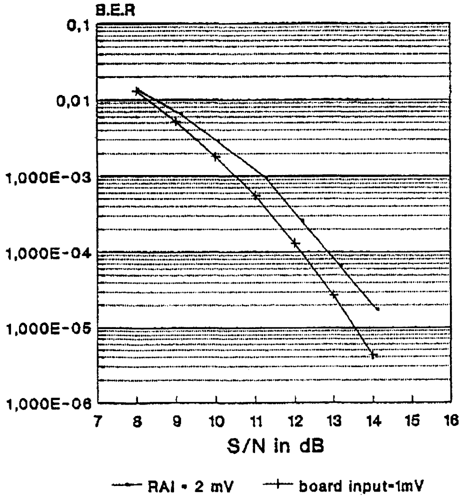
7536-53.TIF

Figure 48 : B.E.R. Channel 3 (72kHz.) ST7536 board input 1mV & 5mV versus stand alone ST7536 RAI input 3mV



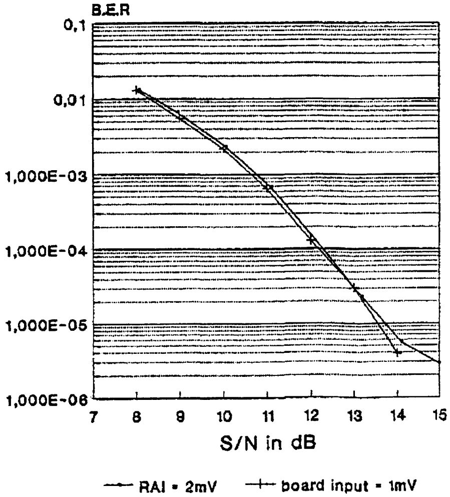
7536-54.TIF

Figure 49 : B.E.R. comparison - ST7536 board versus stand alone ST7536
T = 25°C, Channel 1 (82kHz)



7536-55.TIF

Figure 50 : B.E.R. comparison - ST7536 board versus stand alone ST7536
T = 25°C, Channel 2 (67kHz)



7536-56.TIF

Figure 51 : B.E.R. comparison - ST7536 board versus stand alone ST7536 T = 25°C, Channel 3 (72kHz)

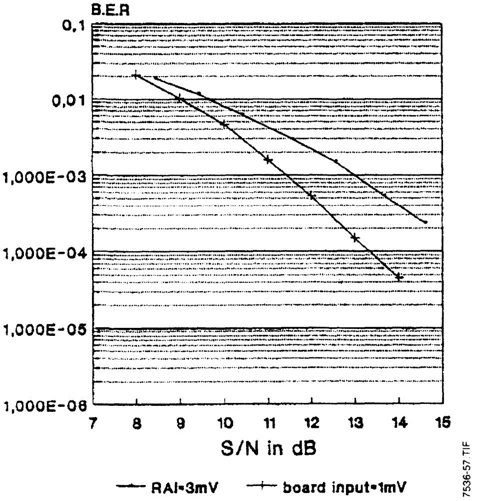
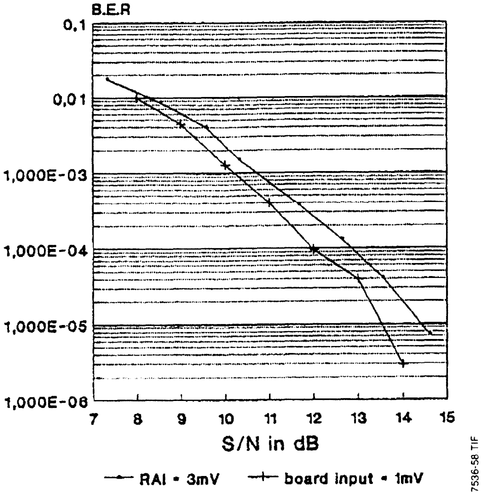
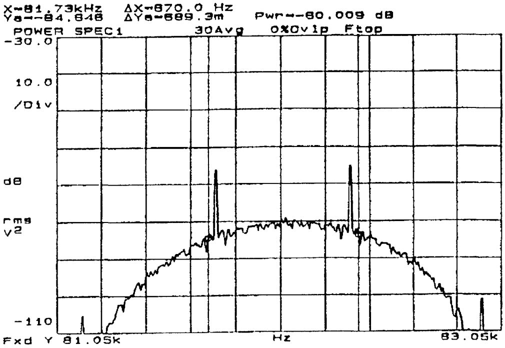


Figure 52 : B.E.R. comparison - ST7536 board versus stand alone ST7536 T = 25°C, Channel 4 (86kHz)



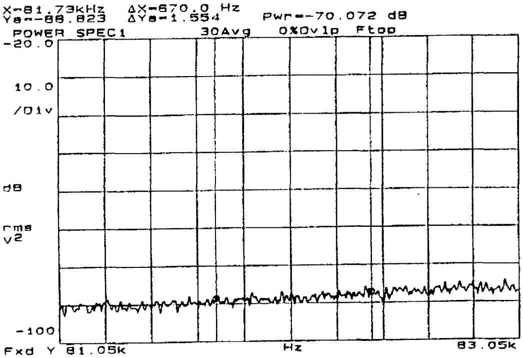
ANNEXE B : B.E.R. ST7536 APPLICATION BOARD (Signal/Noise frequency spectra)

Figure 53 : Channel 1 (82kHz) - S/N = -10dB, Input 1mV = -60dB
FSK (-60dB) - 1mV



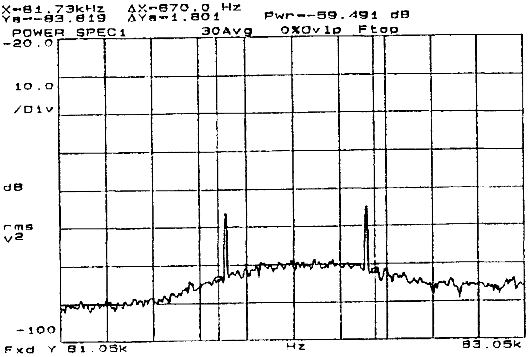
7536-59.TIF

Figure 54 : Channel 1 (82kHz) - S/N = -10dB, Input 1mV = -60dB
Noise (-70dB)



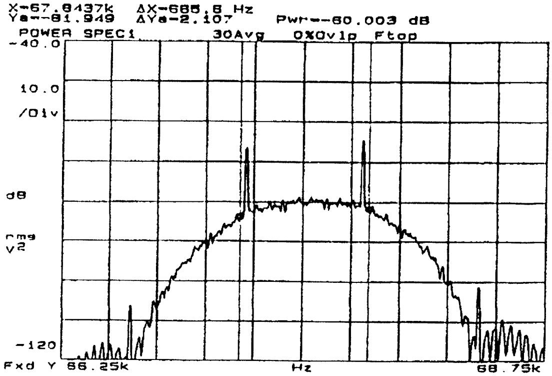
7536-60.TIF

Figure 55 : Channel 1 (82kHz) - S/N = -10dB, Input 1mV = -60dB
FSK + Noise



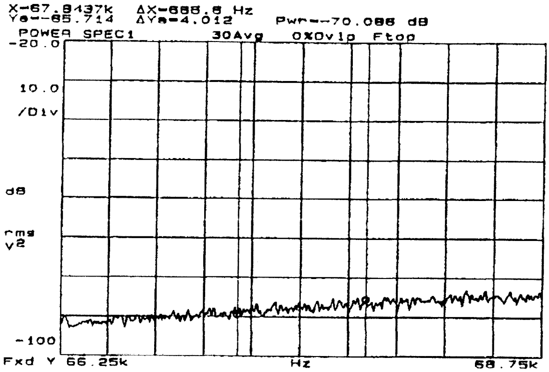
7536-61.TIF

Figure 56 : Channel 2 (67kHz) - S/N = -10dB, Input 1mV = -60dB
FSK (-60dB) - 1mV



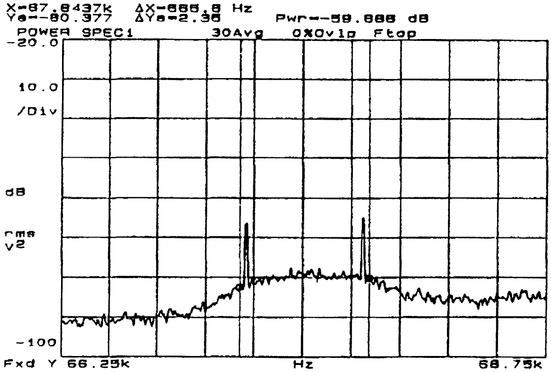
7536-62.TIF

Figure 57 : Channel 2 (67kHz) - S/N = -10dB, Input 1mV = -60dB
Noise (-70dB)



7536-63.TIF

Figure 58 : Channel 2 (67kHz) - S/N = -10dB, Input 1mV = -60dB
FSK + Noise



7536-64.TIF

Figure 59 : Channel 3 (72kHz) - S/N = -10dB, Input 1mV = -60dB
FSK (-60dB) - 1mV

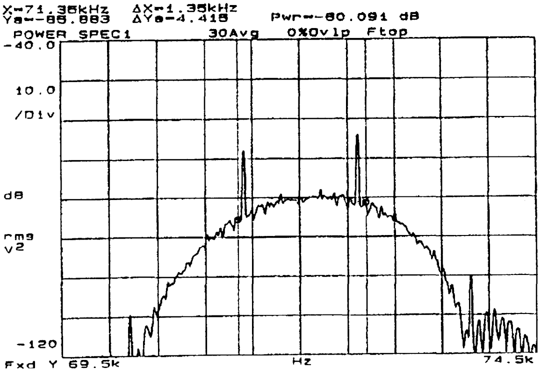


Figure 60 : Channel 3 (72kHz) - S/N = -10dB, Input 1mV = -60dB
Noise (-70dB)

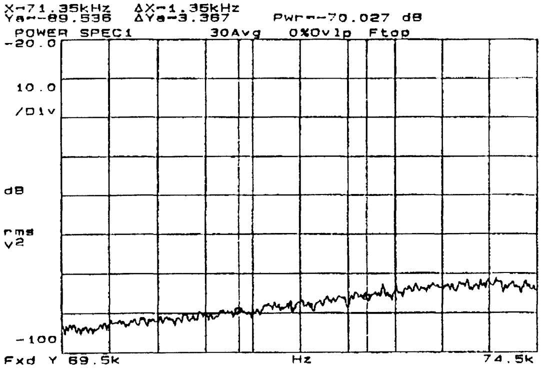


Figure 61 : Channel 3 (72kHz) - S/N = -10dB, Input 1mV = -60dB
FSK + Noise

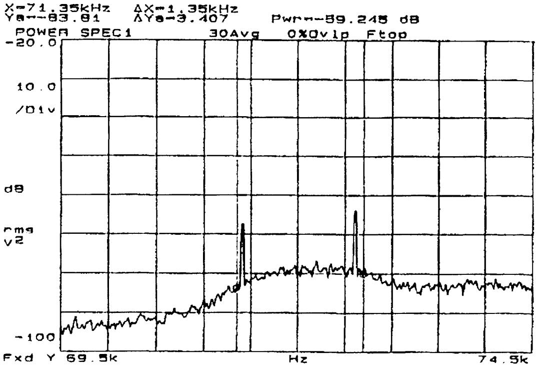
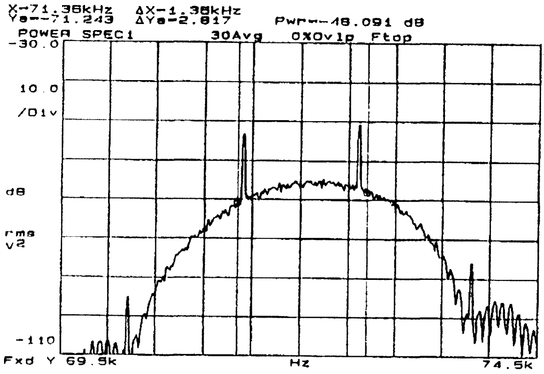
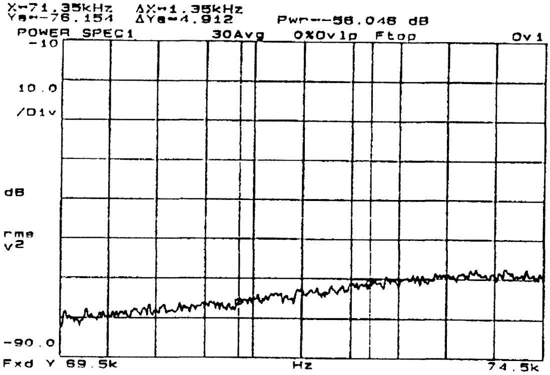


Figure 62 : Channel 3 (72kHz) - S/N = -10dB, Input 5mV = -46dB
FSK (-46dB) - 5mV



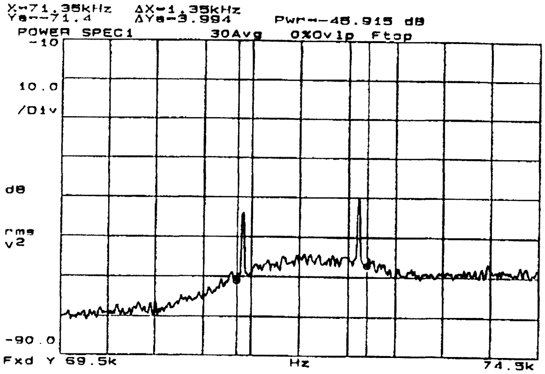
7536-68 TIF

Figure 63 : Channel 3 (72kHz) - S/N = -10dB, Input 5mV = -46dB
Noise (-56dB)



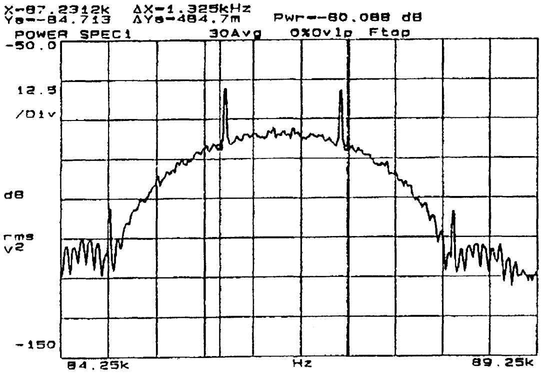
7536-69 TIF

Figure 64 : Channel 3 (72kHz) - S/N = -10dB, Input 5mV = -46dB
FSK + Noise



7536-70 TIF

Figure 65 : Channel 4 (86kHz) - S/N = -10dB, Input 1mV = -60dB
FSK (-60dB) - 1mV



7536-71.TIF

Figure 66 : Channel 4 (86kHz) - S/N = -10dB, Input 1mV = -60dB
Noise (-70dB)

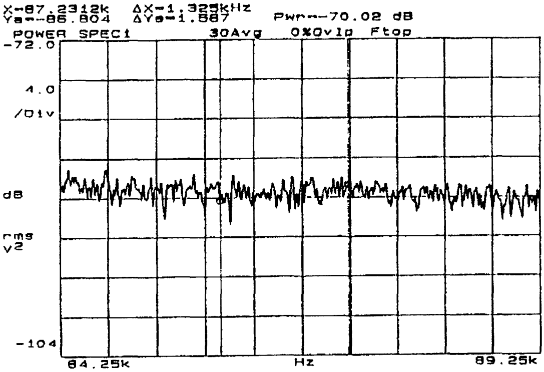
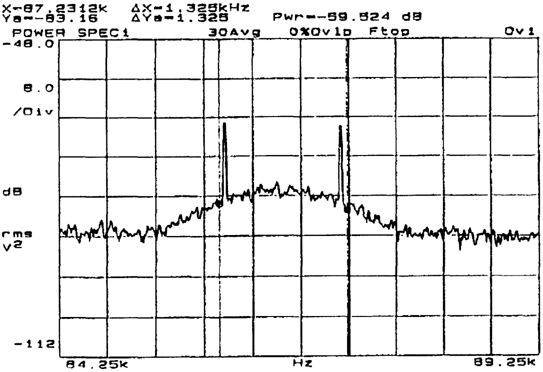


Figure 67 : Channel 4 (86kHz) - S/N = -10dB, Input 1mV = -60dB
FSK + Noise



ANNEXE C : ST6 MASTER PROGRAM

```
;ST7536 POWERLINE COMMUNICATION SYSTEM
;APPLICATION 1 :HEATER CONTROL
;
; MASTER PROGRAM
;
; frame format:                                PC6     PC7      command
; 1: preamble         2 x 1 byte
; 2: syst.address     2 x 1 byte                0       0       heater off
; 3: dest.address     3 x 1 byte                0       1       heater on
; 4: control          3 x 1 byte                1       0       not used
; 5: data             3 x 1 byte                1       1       automatic
;
; data is displayed on Port A
; destination address is read from Port B
; command is read from Port C
;
;PC7 = command input
;PC6 = command input
;PC5 = RXD/TXD
;PC4 = tx/rx

                     .title "APPLICATION_1"
                     .vers "ST6215"
counter              .def 80h
wait_count           .def 81h

preamble             .equ 10101010b
syst_adr             .equ 36h
dest_adr             .def 82h

tx_pattern           .def 83h
tx_contr             .def 84h
tx_data              .def 85h

rx_pattern           .def 86h

rx_cntr              .def 87h
rx_cntr1             .def 88h
rx_cntr2             .def 89h
rx_cntr3             .def 8Ah

rx_data              .def 8Bh
rx_data1             .def 8Ch
rx_data2             .def 8Dh
rx_data3             .def 8Eh

rx_mast              .def 8Fh
rx_mast1             .def 90h
rx_mast2             .def 91h
rx_mast3             .def 92h

byte_1               .def 93h
byte_2               .def 94h
byte_3               .def 95h
byte_c               .def 96h

last_com             .def 97h

                     .romsize 2
                     .pp_on
                     .input "6215_reg.asm"
```

```
;--------------------------initialization----------------------------
                .section 1
                .org 0A0h

reset           ldi wdr,0feh            ;load the watchdog

                ldi ddra,11111111b      ;port A as output
                ldi ora,11111111b       ;to display data
                ldi dra,11111111b       ; ( push-pull )

                ldi ddrb,00000000b      ;port B as input
                ldi orb,00000000b       ;to read dest address
                ldi drb,00000000b       ;pull up R / no interrupts

                ldi ddrc,00111111b      ;pc4   rx/tx   output
                ldi orc,00111111b       ;pc5   rxd/txd input/output
                ldi drc,00101111b       ;pc6/7 command input

                ldi ior,10h             ;enable all interrupts
                ldi tscr,00h            ;disable timer interrupt
                ldi adcr,00h            ;disable adc interrupt


                clr dra                 ;clear display
                reti                    ;end of reset routine

;---------------MAIN PROGRAM-----------------------------------------

                ldi a,00000001b
                ld last_com,a

start           ldi wdr,0feh            ;reload watchdog
                ldi ddrc,00111111b
                ldi orc,00111111b
                ldi drc,00101111b

new_com         ld a,drb                ;get destination address
                ld dest_adr,a           ;store destination address
                clr dra                 ;clear display
                ld a,drc
                andi a,11000000b        ;read PC6/7 (command)
                cp a,last_com           ;new command?
                jrnz loop
                ldi wdr,0feh            ;reload watchdog
                jp new_com

;---------------

loop            ld last_com,a           ;store last command
                cpi a,11000000b         ;auto command?
                jrnz next1
                call automatic
                jp loop_end

next1           cpi a,10000000b         ;manual on command?
                jrnz next2
                call man_on
                jp loop_end

next2           cpi a,00000000b         ;manual off command?
                jrnz next3
                call man_off
                jp loop_end
```

```
next3               cpi a,01000000b          ;not used
                    jrnz loop_end
                    nop
                    jp loop_end

loop_end            jp start

;-------------- automatic heater control ----------------------------

automatic           ldi tx_contr,10000000b   ;command
                    ldi tx_data,00000100b    ;data request
go                  call transmit            ;tx data request
                    call receive             ;read requested data
                    call error_cor           ;correct received data
                    ld a,rx_mast
                    cpi a,11111111b          ;master address?
                    jrnz go
                    jrs 7,rx_cntr,go         ;rx_cntr = command?
                    ld a,rx_data             ;get temp
                    ld dra,a                 ;display temperature
                    jrs 4,rx_data,h_off      ;analyze temperature
                    jrs 5,rx_data,h_off
                    jrs 6,rx_data,h_off
                    jrs 7,rx_data,h_off

h_on                ldi tx_data,00000001b    ;heater on command
                    call transmit            ;tx command
                    jp h_end

h_off               ldi tx_data,00000010b    ;heater off command
                    call transmit            ;tx command

h_end               ld a,drc
                    andi a,11000000b         ;check if the PC6/7 command
                    cpi a,11000000b          ;is still automatic control
                    jrnz auto_end
                    jp automatic

auto_end            ret

;-------------- manual off ------------------------------------------

man_off
                    ldi tx_contr,10000000b   ;command
                    ldi tx_data,00000010b    ;heater off command
                    call transmit
                    ret

;-------------- manual on -------------------------------------------

man_on
                    ldi tx_contr,10000000b   ;command
                    ldi tx_data,00000001b    ;heater on command
                    call transmit
                    ret

;-------------- transmit --------------------------------------------

transmit            ldi wdr,0feh
                    set 4,drc                ;rx/tx = 1
                    call pc5_output          ;select txd
                    ldi counter,0ffh
```

```
car_wait        dec counter
                ldi wdr,0feh              ;wait 5 milliseconds
                ldi wdr,0feh              ;for carrier stabilisation
                jrnz car_wait

                ldi a,preamble            ;preamble -> a
                ld tx_pattern,a           ;a -> tx_pattern
                call send                 ;tx preamble
                call send                 ;tx preamble

                ldi a,syst_adr            ;syst_adr -> a
                ld tx_pattern,a           ;a -> tx_pattern
                call send                 ;tx syst_adr
                call send                 ;tx syst_adr

                ld a,dest_adr             ;dest_adr -> a
                ld tx_pattern,a           ;a -> tx_pattern
                call send                 ;tx dest_adr block 1
                call send                 ;            block 2
                call send                 ;            block 3

                ld a,tx_contr             ;tx_contr -> a
                ld tx_pattern,a           ;a -> tx_pattern
                call send                 ;tx control  block 1
                call send                 ;            block 2
                call send                 ;            block 3

                ld a,tx_data              ;tx_data -> a
                ld tx_pattern,a           ;a -> tx_pattern
                call send                 ;tx data     block 1
                call send                 ;            block 2
                call send                 ;            block 3

                wait                      ;wait
                wait                      ;wait for end last tx-bit
                res 4,drc                 ;rx/tx = 0
                call pc5_input            ;select rxd
                ret

;---------------sent 8-bit pattern ---------------------------------

send            ldi wdr,0feh              ;reload watchdog
                ld a,tx_pattern
                ldi counter,8             ;8 -> counter

tx_loop         wait                      ;wait for nmi

                ldi wait_count,50         ;wait 304 cycles
d_w_c           dec wait_count
                jrnz d_w_c                ;continue if wait_count=0

                rlc a                     ;bit 7 -> carry
                jrnc tx_bit_zero
tx_bit_one      set 5,drc                 ;tx bit = 1
                jp loop_test
tx_bit_zero     res 5,drc                 ;tx bit = 0

loop_test       dec counter
                jrz end
next            jp tx_loop
end             ret                       ;end of tx pattern
```

```
;---------------receive ------------------------------------------

receive         nop                     ;begin of receive routine

syst_read       call read_bit           ;get next bit
                ld a,rx_pattern         ;load rx_pattern
                cpi a,syst_adr          ;rx_pattern = syst_adr ?
                jrz syst_ok
syst_not_ok     jp syst_read

syst_ok         ldi counter,8           ;get next 8 bits
syst2_read      call read_bit           ;get next bit
                dec counter
                jrnz syst2_read

                ld a,rx_pattern         ;load pattern
                cpi a,syst_adr          ;rx_pattern = syst_adr ?
                jrz syst2_ok
syst2_not_ok    jp receive              ;restart
;---------------
syst2_ok        ldi counter,8           ;get next 8 bits
mast1_read      call read_bit           ;get next bit
                dec counter
                jrnz mast1_read

                ld a,rx_pattern         ;load rx_pattern
                ld rx_mast1,a           ;store rx_mast1

                ldi counter,8           ;get next 8 bits
mast2_read      call read_bit           ;get next bit
                dec counter
                jrnz mast2_read

                ld a,rx_pattern         ;load rx_pattern
                ld rx_mast2,a           ;store rx_mast2

                ldi counter,8           ;get next 8 bits
mast3_read      call read_bit           ;get next bit
                dec counter
                jrnz mast3_read

                ld a,rx_pattern         ;load rx_pattern
                ld rx_mast3,a           ;store rx_mast3
;---------------
                ldi counter,8           ;get next 8 bits
contr1_read     call read_bit           ;get next bit
                dec counter
                jrnz contr1_read

                ld a,rx_pattern         ;load pattern
                ld rx_cntr1,a           ;store rx_cntr1

                ldi counter,8           ;get next 8 bits
contr2_read     call read_bit           ;get next bit
                dec counter
                jrnz contr2_read

                ld a,rx_pattern         ;load pattern
                ld rx_cntr2,a           ;store rx_cntr2

                ldi counter,8           ;get next 8 bits
```

```
contr3_read     call read_bit               ;get next bit
                dec counter
                jrnz contr3_read

                ld a,rx_pattern             ;load pattern
                ld rx_cntr3,a               ;store rx_cntr3
;---------------
                ldi counter,8               ;get next 8 bits
data1_read      call read_bit               ;get next bit
                dec counter
                jrnz data1_read

                ld a,rx_pattern             ;load rx_pattern
                ld rx_data1,a               ;store rx_data1

                ldi counter,8               ;get next 8 bits
data2_read      call read_bit               ;get next bit
                dec counter
                jrnz data2_read

                ld a,rx_pattern             ;load rx_pattern
                ld rx_data2,a               ;store rx_data2

                ldi counter,8               ;get next 8 bits
data3_read      call read_bit               ;get next bit
                dec counter
                jrnz data3_read

                ld a,rx_pattern             ;load rx_pattern
                ld rx_data3,a               ;store rx_data3

                ret                         ;end of reading frame

;-------------- read bit ------------------------------------------

read_bit        ldi wdr,0feh                ;reload watchdog
                wait
                ld a,rx_pattern             ;load pattern
                sla a                       ;shift left accu
                jrs 5,drc,pc5_set           ;jump if PC5 is set
pc5_res         res 0,a                     ;accu bit0 = 0
                ld rx_pattern,a             ;store rx_pattern
                ret
pc5_set         set 0,a                     ;accu bit0 = 1
                ld rx_pattern,a             ;store rx_pattern
                ret

;-------------- error correction ----------------------------------

error_cor       ld a,rx_mast1               ;correct rx master address
                ld byte_1,a
                ld a,rx_mast2
                ld byte_2,a
                ld a,rx_mast3
                ld byte_3,a

                clr byte_c
                call det_byte_c
                ld a,byte_c
                ld rx_mast,a
```

```
;---------------
                   ld a,rx_cntr1           ;correct  rx control
                   ld byte_1,a
                   ld a,rx_cntr2
                   ld byte_2,a
                   ld a,rx_cntr3
                   ld byte_3,a

                   clr byte_c
                   call det_byte_c
                   ld a,byte_c
                   ld rx_cntr,a
;---------------
                   ld a,rx_data1           ;correct rx data
                   ld byte_1,a
                   ld a,rx_data2
                   ld byte_2,a
                   ld a,rx_data3
                   ld byte_3,a

                   clr byte_c
                   call det_byte_c
                   ld a,byte_c
                   ld rx_data,a

                   ret

;-------------- determinate correct byte (byte_c) -------------------

det_byte_c                                 ;correction of byte
det_bit0           jrr 0,byte_1,a0         ;correction of bit 0
                   jrr 0,byte_2,b0
                   jp bit01

a0                 jrr 0,byte_2,bit00
                   jrr 0,byte_3,bit00
                   jp bit01

b0                 jrr 0,byte_3,bit00

bit01              set 0,byte_c
                   jp det_bit1
bit00              res 0,byte_c
;----------------------------------
det_bit1           jrr 1,byte_1,a1         ;correction of bit 1
                   jrr 1,byte_2,b1
                   jp bit11

a1                 jrr 1,byte_2,bit10
                   jrr 1,byte_3,bit10
                   jp bit11

b1                 jrr 1,byte_3,bit10
bit11              set 1,byte_c
                   jp det_bit2
bit10              res 1,byte_c
;----------------------------------
det_bit2           jrr 2,byte_1,a2         ;correction of bit 2
                   jrr 2,byte_2,b2
                   jp bit21
```

```
a2              jrr 2,byte_2,bit20
                jrr 2,byte_3,bit20
                jp bit21

b2              jrr 2,byte_3,bit20

bit21           set 2,byte_c
                jp det_bit3
bit20           res 2,byte_c
;-----------------------------------
det_bit3        jrr 3,byte_1,a3             ;correction of bit 3
                jrr 3,byte_2,b3
                jp bit31

a3              jrr 3,byte_2,bit30
                jrr 3,byte_3,bit30
                jp bit31

b3              jrr 3,byte_3,bit30

bit31           set 3,byte_c
                jp det_bit4
bit30           res 3,byte_c
;-----------------------------------
det_bit4        jrr 4,byte_1,a4             ;correction of bit 4
                jrr 4,byte_2,b4
                jp bit41

a4              jrr 4,byte_2,bit40
                jrr 4,byte_3,bit40
                jp bit41

b4              jrr 4,byte_3,bit40

bit41           set 4,byte_c
                jp det_bit5
bit40           res 4,byte_c
;-----------------------------------
det_bit5        jrr 5,byte_1,a5             ;correction of bit 5
                jrr 5,byte_2,b5
                jp bit51

a5              jrr 5,byte_2,bit50
                jrr 5,byte_3,bit50
                jp bit51
b5              jrr 5,byte_3,bit50

bit51           set 5,byte_c
                jp det_bit6
bit50           res 5,byte_c
;-----------------------------------
det_bit6        jrr 6,byte_1,a6             ;correction of bit 6
                jrr 6,byte_2,b6
                jp bit61

a6              jrr 6,byte_2,bit60
                jrr 6,byte_3,bit60
                jp bit61

b6              jrr 6,byte_3,bit60

bit61           set 6,byte_c
                jp det_bit7
```

```
bit60           res 6,byte_c
;---------------------------------
det_bit7        jrr 7,byte_1,a7            ;correction of bit 7
                jrr 7,byte_2,b7
                jp bit71

a7              jrr 7,byte_2,bit70
                jrr 7,byte_3,bit70
                jp bit71

b7              jrr 7,byte_3,bit70

bit71           set 7,byte_c
                jp det_end
bit70           res 7,byte_c

det_end         ret                        ;end of byte correction

;-------------- PC5 input/output --------------------------------------

pc5_input       res 5,ddrc                 ;program PC5 as input
                res 5,orc
                set 5,drc
                ret

pc5_output      set 5,ddrc                 ;program PC5 as output
                set 5,ddrc
                set 5,ddrc
                ret

;-------------- NMI interrupt routine ---------------------------------

nmi_int         reti                       ;not used

;-------------- interrupt vector routines -----------------------------

                .section 32
                .org 00h
adc             nop
                reti
timer           nop
                reti
int2            nop
                reti
int1            nop
                reti

                .org 0ch
nmi             jp nmi_int
res             jp reset
;--------------END------------------------------------------------------
```

ANNEXE D : ST6 SLAVE PROGRAM

```
;ST7536 POWERLINE COMMUNICATION SYSTEM
;
; SLAVE PROGRAM
;
; frame format:                               command:
; 1: preamble         2 x 1 byte              00000001  led on
; 2: syst.address     2 x 1 byte              00000010  led off
; 3: dest.address     3 x 1 byte              00000100  data request
; 4: control          3 x 1 byte
; 5: data             3 x 1 byte
;
; home address is read from PA
; data is read from PB
;
;PC7 = RXD            = input,  no pull up R, no interrupt
;PC6 = LED output     = output, push pull
;PC5 = TXD __         = output, push pull
;PC4 = tx / rx        = output, push pull

                .title "SLAVE"
                .vers "ST6215"

counter         .def 80h
wait_count      .def 81h

preamble        .equ 10101010b
syst_adr        .equ 36h
master_adr      .equ 11111111b
home_adr        .def 82h

tx_pattern      .def 83h
tx_contr        .def 84h
tx_data         .def 85h

rx_pattern      .def 86h

rx_home         .def 87h
rx_home1        .def 88h
rx_home2        .def 89h
rx_home3        .def 8Ah

rx_cntr         .def 8Bh
rx_cntr1        .def 8Ch
rx_cntr2        .def 8Dh
rx_cntr3        .def 8Eh

rx_data         .def 8Fh
rx_data1        .def 90h
rx_data2        .def 91h
rx_data3        .def 92h
byte_1          .def 93h
byte_2          .def 94h
byte_3          .def 95h
byte_c          .def 96h

                .romsize 2
                .pp_on
                .input "6215_reg.asm"
```

```
;--------------initialisation ---------------------------------
                .section 1
                .org 0A0h

reset           ldi wdr,0feh              ;load the watchdog

                ldi ddra,00000000b        ;port A as adr.input
                ldi ora,00000000b         ;no interrupts
                ldi dra,00000000b         ;pull up resistor

                ldi ddrb,00000000b        ;port B as data input
                ldi orb,00000000b         ;no interrupts
                ldi drb,00000000b         ;pull up resistor

                ldi ddrc,01111111b        ;port C as output
                ldi orc,01111111b         ;  (push pull)
                ldi drc,11111111b         ;pc7 rxd input

                ldi ior,10h               ;enable all interrupts
                ldi tscr,00h              ;disable timer interrupt
                ldi adcr,00h              ;disable adc interrupt

                res 6,drc
                reti

;-------------- MAIN PROGRAM ----------------------------------

start           ldi wdr,0feh              ;reload watchdog

                call set_up

                call receive              ;go in rx-mode
                call decode               ;decode received frame

                ld a,rx_home              ;home address ?
                cp a,home_adr
                jrz home_ok
                jp start                  ;if not then restart
home_ok         ld a,rx_data
                cpi a,00000100b           ;command=data request?
                jrz d_request
                jp start                  ;if not restart
d_request       ld a,drb                  ;read data
                ld tx_data,a              ;store data
                call transmit             ;tx requested data

                jp start

;-------------- set up ----------------------------------------

set_up          clr tx_contr

                ld a,dra                  ;dra -> a
                ld home_adr,a             ;a -> home_adr
                ret

;-------------- transmit --------------------------------------
transmit        set 4,drc                 ;transmit mode
                ldi wait_count,0ffh
car_wait        dec wait_count
                ldi wdr,0feh              ;wait 5 milliseconds
                ldi wdr,0feh              ;for carrier stabilisation
                jrnz car_wait
                ldi a,preamble          ;preamble -> a
                ld tx_pattern,a         ;a -> tx_pattern
                call send               ;tx preamble
                call send               ;tx preamble

                ldi a,syst_adr          ;syst_adr -> a
                ld tx_pattern,a         ;a -> tx_pattern
                call send               ;tx syst_adr
                call send               ;tx syst_adr

                ldi a,master_adr        ;master_adr -> a
                ld tx_pattern,a         ;a -> tx_pattern
                call send               ;tx centr_adr block 1
                call send               ;             block 2
                call send               ;             block 3

                ld a,tx_contr           ;tx_contr -> a
                ld tx_pattern,a         ;a -> tx_pattern
                call send               ;tx tx_contr  block 1
                call send               ;             block 2
                call send               ;             block 3

                ld a,tx_data            ;tx_data -> a
                ld tx_pattern,a         ;a -> tx_pattern
                call send               ;tx tx_data   block 1
                call send               ;             block 2
                call send               ;             block 3

                wait                    ;wait
                wait                    ;wait for end last tx-bit
                res 4,drc               ;receive mode
                ret

;-------------- send 8-bit pattern ----------------------------------

send            ldi wdr,0feh            ;reload watchdog
                ld a,tx_pattern
                ldi counter,8           ;8 -> counter

tx_loop         wait                    ;wait for nmi

                ldi wait_count,50       ;wait 304 cycles
d_w_c           dec wait_count
                jrnz d_w_c              ;continue if wait_count=0

                rlc a                   ;bit7 -> carry
                jrnc bit_zero
bit_one         set 5,drc               ;tx bit = 1
                jp loop_test
bit_zero        res 5,drc               ;tx bit = 0

loop_test       dec counter
                jrz end
next            jp tx_loop
end             ret                     ;end of tx pattern

;-------------- receive ---------------------------------------------

receive         res 4,drc               ;receive mode

syst_read       call read_bit           ;get next bit
                ld a,rx_pattern         ;load rx_pattern
                cpi a,syst_adr          ;rx_pattern = syst_adr ?
                jrz syst_ok
```

```
syst_not_ok     jp syst_read
;----------------------------------------------------------------
syst_ok         ldi counter,8           ;get next 8 bits
syst2_read      call read_bit           ;get next bit
                dec counter
                jrnz syst2_read

                ld a,rx_pattern
                cpi a,syst_adr          ;rx_pattern = syst_adr ?
                jrz syst2_ok
syst2_not_ok    jp receive              ;receive
;---------------
syst2_ok        ldi counter,8           ;get next 8 bits
home1_read      call read_bit           ;get next bit
                dec counter
                jrnz home1_read
                ld a,rx_pattern         ;load rx_pattern
                ld rx_home1,a           ;store rx_home1


                ldi counter,8           ;get next 8 bits
home2_read      call read_bit           ;get next bit
                dec counter
                jrnz home2_read

                ld a,rx_pattern         ;load rx_pattern
                ld rx_home2,a           ;store rx_home2

                ldi counter,8           ;get next 8 bits
home3_read      call read_bit           ;get next bit
                dec counter
                jrnz home3_read

                ld a,rx_pattern         ;load rx_pattern
                ld rx_home3,a           ;store rx_home3
;---------------
                ldi counter,8           ;get next 8 bits
contr1_read     call read_bit           ;get next bit
                dec counter
                jrnz contr1_read

                ld a,rx_pattern         ;load pattern
                ld rx_cntr1,a           ;store rx_cntr1

                ldi counter,8           ;get next 8 bits
contr2_read     call read_bit           ;get next bit
                dec counter
                jrnz contr2_read

                ld a,rx_pattern         ;load pattern
                ld rx_cntr2,a           ;store pattern

                ldi counter,8           ;get next 8 bits
contr3_read     call read_bit           ;get next bit
                dec counter
                jrnz contr3_read

                ld a,rx_pattern         ;load rx_pattern
                ld rx_cntr3,a           ;store rx_cntr3
;---------------
                ldi counter,8           ;get next 8 bits
data_1_read     call read_bit           ;get next bit
                dec counter
                jrnz data_1_read
                ld a,rx_pattern         ;load rx_pattern
                ld rx_data1,a           ;store rx_data1
                ldi counter,8           ;get next 8 bits
data_2_read     call read_bit           ;get next bit
                dec counter
                jrnz data_2_read

                ld a,rx_pattern         ;load rx_pattern
                ld rx_data2,a           ;store rx_data2

                ldi counter,8           ;get next 8 bits
data_3_read     call read_bit           ;get next bit
                dec counter
                jrnz data_3_read

                ld a,rx_pattern         ;load rx_pattern
                ld rx_data3,a           ;store rx_data3

                ret                     ;end of reading frame

;-------------- read bit ------------------------------------------

read_bit        ldi wdr,0feh            ;reload watchdog
                wait
                ld a,rx_pattern         ;load rx_pattern
                sla a                   ;shift left accu
                jrs 7,drc,pc7_set       ;jump if PC7 is set
pc7_res         res 0,a                 ;accu bit0 = 0
                ld rx_pattern,a         ;store rx_pattern
                ret
pc7_set         set 0,a                 ;accu bit0 = 1
                ld rx_pattern,a         ;store rx_pattern
                ret

;---------------packet decoding-----------------------------------

decode          call error_cor

                ld a,rx_home
                cp a,home_adr
                jrz continue
                ret

continue        ld a,rx_cntr
                cpi a,10000000b
                jrz command
                cpi a,00000000b
                jrz response

unknown         ret

response        nop                     ;master use only
                ret
command         ld a,rx_data
                cpi a,00000001b         ;led on command
                jrnz next1
                jp instr_1
next1           cpi a,00000010b         ;led off command
                jrnz next2
                jp instr_2
next2           cpi a,00000100b         ;data request
                jrnz next3
                jp instr_3
next3           cpi a,00001000b         ;not used
```

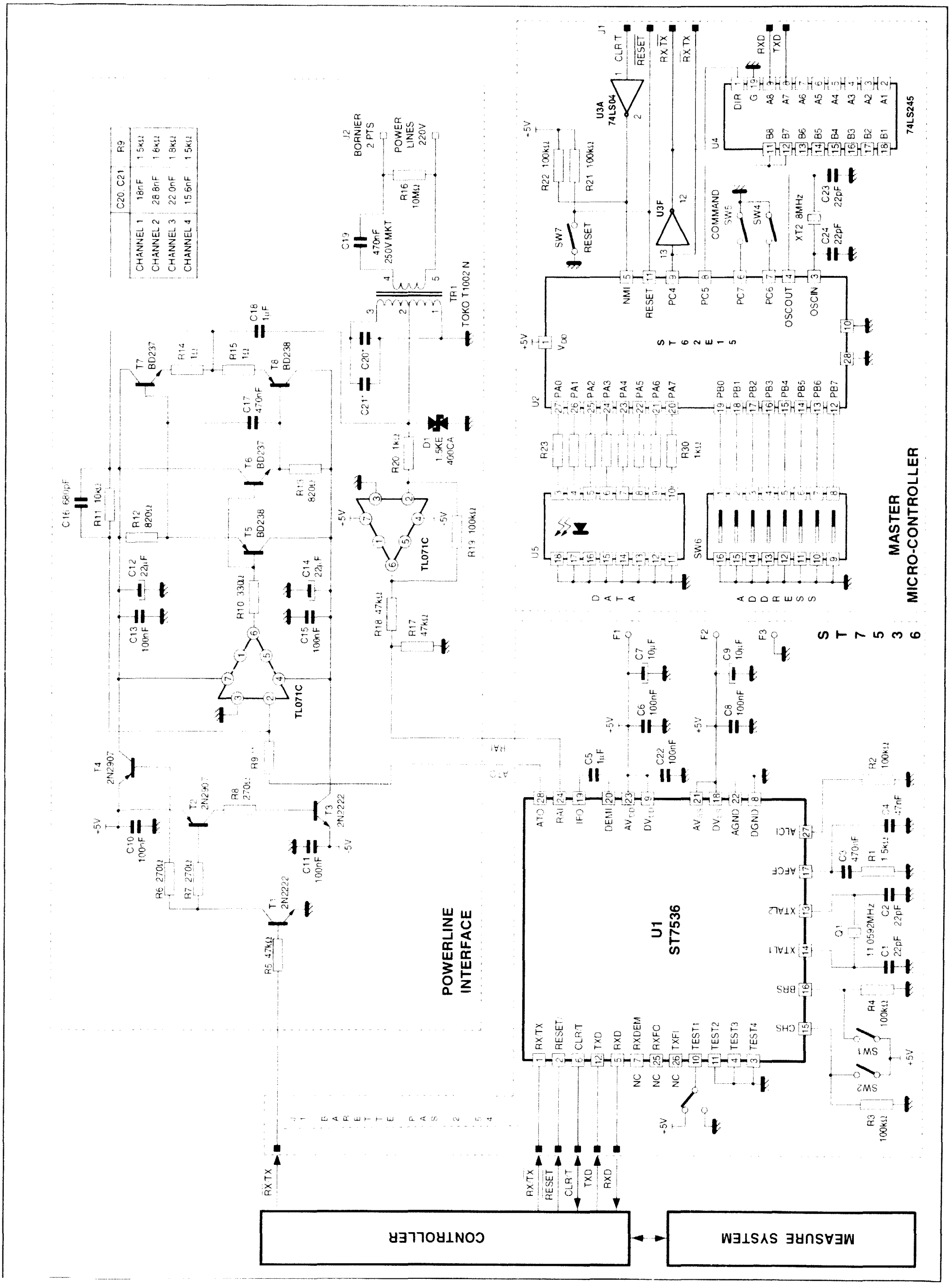
```
                jrnz no_command
                jp instr_4

no_command      ret

instr_1         set 6,drc               ;led on
                ret
instr_2         res 6,drc               ;led off
                ret
instr_3         ld a,drb                ;read data
                ld tx_data,a            ;store data
                ret
instr_4         nop                     ;not in use
                ret

;---------------error correction--------------------------

error_cor       ld a,rx_home1           ;correct rx_home
                ld byte_1,a
                ld a,rx_home2
                ld byte_2,a
                ld a,rx_home3
                ld byte_3,a

                call det_byte_c
                ld a,byte_c
                ld rx_home,a

                ld a,rx_cntr1           ;correct rx_cntr
                ld byte_1,a
                ld a,rx_cntr2
                ld byte_2,a
                ld a,rx_cntr3
                ld byte_3,a

                call det_byte_c
                ld a,byte_c
                ld rx_cntr,a

                ld a,rx_data1           ;correct rx_data
                ld byte_1,a
                ld a,rx_data2
                ld byte_2,a
                ld a,rx_data3
                ld byte_3,a

                call det_byte_c
                ld a,byte_c
                ld rx_data,a

                ret
;---------------determinate correct byte (byte_c)-----------

det_byte_c      ldi wdr,0feh            ;error correction
det_bit0        jrr 0,byte_1,a0         ;correction of bit 0
                jrr 0,byte_2,b0
                jp bit01

a0              jrr 0,byte_2,bit00
                jrr 0,byte_3,bit00
                jp bit01
```

```
b0              jrr 0,byte_3,bit00

bit01           set 0,byte_c
                jp det_bit1
bit00           res 0,byte_c
;----------------------------------
det_bit1        jrr 1,byte_1,a1              ;correction of bit 1
                jrr 1,byte_2,a1
                jp bit11

a1              jrr 1,byte_2,bit10
                jrr 1,byte_3,bit10
                jp bit11

b1              jrr 1,byte_3,bit10

bit11           set 1,byte_c
                jp det_bit2
bit10           res 1,byte_c
;----------------------------------
det_bit2        jrr 2,byte_1,a2              ;correction of bit 2
                jrr 2,byte_2,b2
                jp bit21

a2              jrr 2,byte_2,bit20
                jrr 2,byte_3,bit20
                jp bit21

b2              jrr 2,byte_3,bit20

bit21           set 2,byte_c
                jp det_bit3
bit20           res 2,byte_c
;----------------------------------
det_bit3        jrr 3,byte_1,a3              ;correction of bit 3
                jrr 3,byte_2,b3
                jp bit31

a3              jrr 3,byte_2,bit30
                jrr 3,byte_3,bit30
                jp bit31

b3              jrr 3,byte_3,bit30

bit31           set 3,byte_c
                jp det_bit4
bit30           res 3,byte_c
;----------------------------------
det_bit4        jrr 4,byte_1,a4              ;correction of bit 4
                jrr 4,byte_2,b4
                jp bit41

a4              jrr 4,byte_2,bit40
                jrr 4,byte_3,bit40
                jp bit41

b4              jrr 4,byte_3,bit40

bit41           set 4,byte_c
                jp det_bit5
bit40           res 4,byte_c
;----------------------------------
det_bit5        jrr 5,byte_1,a5              ;correction of bit 5
                jrr 5,byte_2,b5
                jp bit51

a5              jrr 5,byte_2,bit50
                jrr 4,byte_3,bit50
                jp bit51

b5              jrr 5,byte_3,bit50

bit51           set 5,byte_c
                jp det_bit6
bit50           res 5,byte_c
;----------------------------------
det_bit6        jrr 6,byte_1,a6          ;correction of bit 6
                jrr 6,byte_2,b6
                jp bit61
a6              jrr 6,byte_2,bit60
                jrr 6,byte_3,bit60
                jp bit61

b6              jrr 6,byte_3,bit60

bit61           set 6,byte_c
                jp det_bit7
bit60           res 6,byte_c
;----------------------------------
det_bit7        jrr 7,byte_1,a7          ;correction of bit 7
                jrr 7,byte_2,b7
                jp bit71

a7              jrr 7,byte_2,bit70
                jrr 7,byte_3,bit70
                jp bit71

b7              jrr 7,byte_3,bit70

bit71           set 7,byte_c
                jp det_end
bit70           res 7,byte_c

det_end         ret                      ;end of byte correction

;-------------- NMI interrupt routine ---------------------------------

nmi_int         reti                     ;not used

;-------------- interrupt vector routines ----------------------------
                .section 32
                .org 00h
adc             nop
                reti
timer           nop
                reti
int2            nop
                reti
int1            nop
                reti

                .org 0ch
nmi             jp nmi_int
res             jp reset
;----------------------------------------------------------------------
```

ANNEXE E : ST6 REGISTER FILE

```
;
;                    ST6210/15 and ST6220/25 Registers Declaration
;                     Use this file with the .input directive.
;

x        .def 80h!m              ; Index register.
y        .def 81h!m              ; Index register.
v        .def 82h                ; Short direct register.
w        .def 83h                ; Short direct register.

a        .def 0ffh!m             ; Accumulator.

dra      .def 0c0h               ; Port a data register.
drb      .def 0c1h               ; Port b data register.
drc      .def 0c2h               ; Port c data register.

ddra     .def 0c4h               ; Port a direction register.
ddrb     .def 0c5h               ; Port b direction register.
ddrc     .def 0c6h               ; Port c direction register.

ior      .def 0c8h               ; Interrupt option register.
drwr     .def 0c9h               ; Data rom window register.

ora      .def 0cch               ; Port a option register.
orb      .def 0cdh               ; Port b option register.
orc      .def 0ceh               ; Port c option register.

adr      .def 0d0h               ; A/D data register.
adcr     .def 0d1h               ; A/D control register.

psc      .def 0d2h               ; Timer prescaler register.
tcr      .def 0d3h!m             ; Timer counter register.
tscr     .def 0d4h               ; Timer status control register.

wdr      .def 0d8h               ; Watchdog register.
```

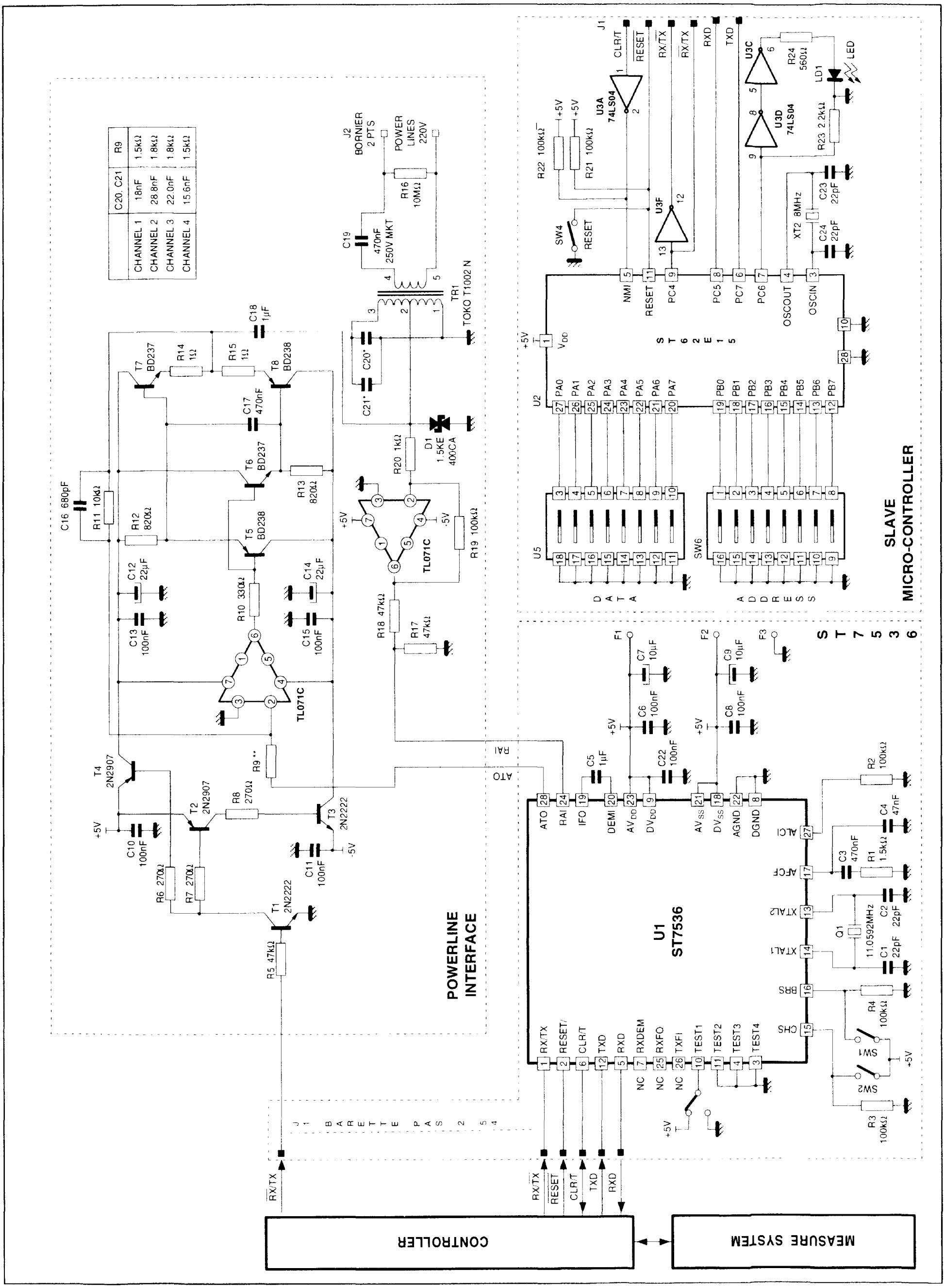
Figure 68 : ST7536 Master



	C20, C21	R9
CHANNEL 1	18nF	1.5kΩ
CHANNEL 2	28.8nF	1.8kΩ
CHANNEL 3	22.0nF	1.8kΩ
CHANNEL 4	15.6nF	1.5kΩ

7536-74.EPS

Figure 69 : ST7536 Slave



	C20, C21	R9
CHANNEL 1	18nF	1.5kΩ
CHANNEL 2	28.8nF	1.8kΩ
CHANNEL 3	22.0nF	1.8kΩ
CHANNEL 4	15.6nF	1.5kΩ

7536-75.EPS

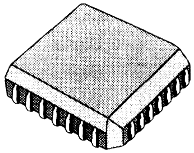
ST7537 DATASHEET

SGS-THOMSON
MICROELECTRONICS

ST7537

HOME AUTOMATION MODEM

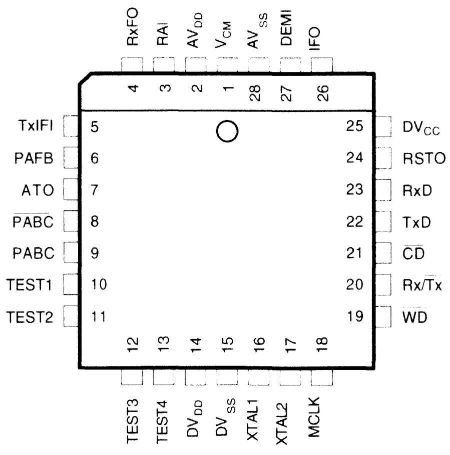
- HALF DUPLEX ASYNCHRONOUS 1200bps FSK MODEM
- Tx CARRIER FREQUENCY SYNTHESIZED FROM EXTERNAL CRYSTAL
- LOW DISTORTION Tx SIGNAL
- 10mVrms Rx SENSITIVITY
- CARRIER DETECTION
- WATCH-DOG INPUT
- RESET AND MASTER CLOCK OUTPUTS FOR MICROCONTROLLER
- POWER AMPLIFIER BIAS CURRENT CONTROL (HIGH IMPEDANCE IN Rx MODE)
- SIMPLE AND ECONOMICAL APPLICATION SCHEMATICS
- COMPATIBLE WITH CENELEC EN 50065-1 AND FCC SPECIFICATION
- CARRIER DETECT CLAMPING ON RxD PROGRAMMABLE (ALLOWING DEMODULATION ON VERY LOW RECEIVE LEVEL, 1mV$_{RMS}$ TYPICALLY)



PLCC28
(Plastic Chip Carrier)

ORDER CODE : ST7537

PIN CONNECTIONS



7537-01.EPS

DESCRIPTION

The ST7537 is a half duplex asynchronous FSK MODEM designed for home automation communication on the domestic electric mains which complies with the EN 50065-1 CENELEC standard.

It mainly operates from a 10 V power supply and a 5V power supply for the microcontroller digital interface.

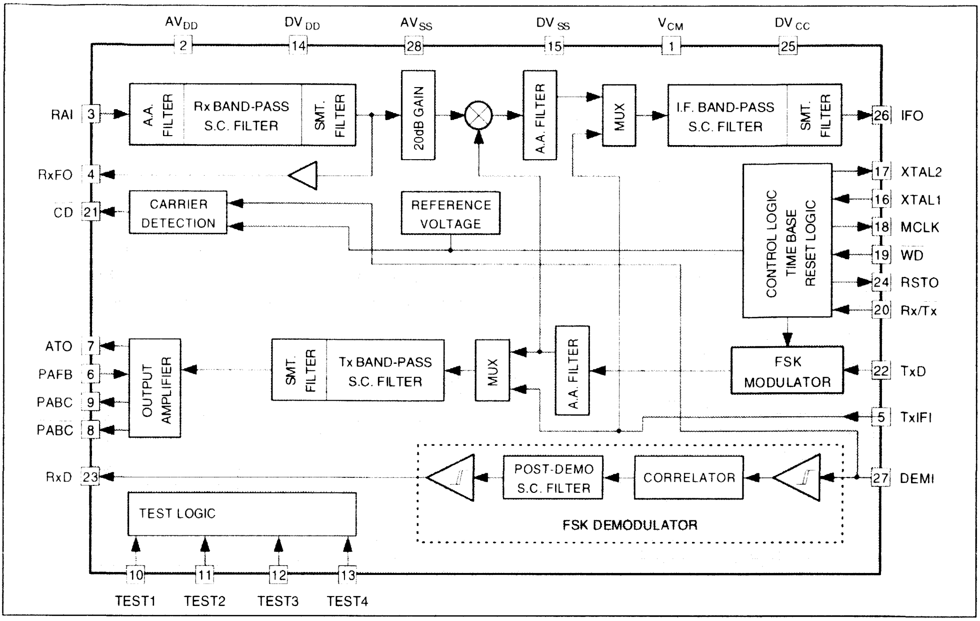
It is interfaced to the power line by an external driver, and a transformer (see Application Schematic Diagram). Its data transmission rate is 1200 bps and its carrier frequency is 132.45kHz.

PIN DESCRIPTION

Pin Name	Pin Number	Pin Type	Description
V_{CM}	1	Analog	Common Mode Voltage
AV_{DD}	2	Supply	Analog Power Supply : 10V ±5 %
RAI	3	Analog	Receive Analog Input
RxFO	4	Analog	Receive Filter Output
TxIFI	5	Analog	Transmit and Intermediate Frequency Filters Test Input (mode TEST3)
PAFB	6	Analog	Power Amplifier Feed-back Input
ATO	7	Analog	Analog Transmit Output
$\overline{PABC}$	8	Digital (10V)	Power Amplifier Bias Current Control Complementary Output
PABC	9	Digital (10V)	Power Amplifier Bias Current Control Output
TEST1	10	Digital	Tx to Rx Automatic Mode Switching Control Input
TEST2	11	Digital	Automatic Mode Switching Time and Watch-dog Time Reduction Control Input
TEST3	12	Digital	TxIFI Selection Input
TEST4	13	Digital	Undelayed Reset Input
DV_{DD}	14	Supply	Digital Power Supply : 10V ±5%
DV_{SS}	15	Supply	Digital Ground : 0V
XTAL1	16	Digital (10V)	Crystal Oscillator Input
XTAL2	17	Digital (10V)	Crystal Oscillator Output
MCLK	18	Digital	Master Clock Output
$\overline{WD}$	19	Digital	Watch-dog Input
$Rx/\overline{Tx}$	20	Digital	Rx or Tx Mode Selection Input
$\overline{CD}$	21	Digital	Carrier Detect Output
TxD	22	Digital	Transmit Data Input
RxD	23	Digital	Receive Data Output
RSTO	24	Digital	Reset Output
DV_{CC}	25	Supply	Digital Buffers Supply Voltage : 5 V ±5 %
IFO	26	Analog	Intermediate Frequency Filter Output
DEMI	27	Analog	Demodulator Input
AV_{SS}	28	Supply	Analog Ground : 0V

7537-01.TBL

BLOCK DIAGRAM



7537-02 EPS

TRANSMIT SECTION

The transmit mode is set when Rx/$\overline{\text{Tx}}$ = 0. if Rx/$\overline{\text{Tx}}$ is held at 0 longer than 1 second, then the device switches automatically in the Rx mode. A new activation of the Tx mode requires Rx/$\overline{\text{Tx}}$ to be returned to 1 for a minimum 2 microsecond period before being set to 0.

The Transmit Data (TxD) enter asynchronously the FSK modulator with a nominal intra-message data rate of 1200 bps.

The basic transmit frequencies are :
- f(TxD=0) = 133.05kHz
- f(TxD=1) = 131.85kHz

These frequencies are synthesized from a 11.0592MHz crystal oscillator; their precision is the same as the crystal one's (100ppm).

The modulated signal coming out of the FSK modulator is filtered by a switched-capacitor band-pass filter (Tx band-pass) in order to limit the output spectrum and to reduce the level of harmonic components.

The final stage of the Tx path consists of an operational amplifier which needs a feed-back signal (PAFB) from the power amplifier as shown on Application Schematic Diagram.

In Tx mode the Receive Data (RxD) signai is set to 1.

RECEIVE SECTION

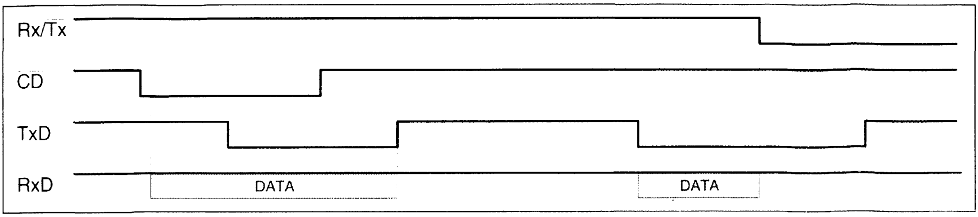
The receive section is active when Rx/$\overline{\text{Tx}}$ = 1.

The Rx signal is applied on RAI and filtered by a band-pass switched capacitor filter (Rx band-pass) centered on the carrier frequency and whose bandwidth is around 12kHz.The Rx filter output is amplified by a 20dB gain stage which provides symetrical limitations for large voltage. The resulting signal is down-converted by a mixer which receives a local oscillator synthesized by the FSK modulator block. Finally an intermediate frequency band-pass filter (IF band-pass) whose central frequency is 5.4kHz improves the signal to noise ratio before entering the FSK demodulator. The coupling of the intermediate frequency filter output (IFO) to the FSK demodulator input (DEMI) is made by an external capacitor C5 (100nF ±10%, 10V) which cancels the Rx path offset voltage.

The RxD output delivers the demodulated signal if the carrier detect ($\overline{\text{CD}}$) signal is low and is set to high level when CD = 1.

The RxD output can delivers the demodulated signal whatever the level of $\overline{\text{CD}}$ (0 or 1) if Rx/$\overline{\text{Tx}}$ = 1 and TxD = 0 (see Figure 1).

Figure 1 : Data Timing Chart



ADDITIONAL DIGITAL AND ANALOG FUNCTIONS

Time base

A time base section delivers all the internal clocks from a crystal oscillator (11.0592MHz). The crystal is connected between XTAL1 and XTAL2 pins and needs two external capacitors C3 and C4 (22pF ±10%, 10V) for proper operation.

Reset and watch-dog

The reset output (RSTO) is driven high when the supply voltage is lower than Vrh (typically 7.6V) with an hysteresis Vrh-Vrl (typically 300mV) or when no negative transition occurs on the watch-dog input ($\overline{WD}$) for more than 1.5 second (see the timing chart on Figure 2). When a reset occurs RSTO is held high for at least 50ms.

Signal detection

The Carrier Detect output ($\overline{CD}$) is driven low when the input signal amplitude on RAI is greater than V_{CD} for at least T_{CD} (typically 6ms see the timing chart on Figure 3). When the input signal desappears or becomes lower than V_{CD}, $\overline{CD}$ is held low for at least Tcd before returning to a high level. V_{CD} is the carrier detection threshold voltage which is set internally to detect $5mV_{RMS}$ typically.

External power amplifier bias control

Two dedicated digital output (PABC and $\overline{PABC}$) delivering a signal between 0V and 10V are driven low respectively high, when the circuit is set in the receive mode (Rx/$\overline{Tx}$=1) or when the transmit mode time out (1 second) is exceeded; in the same time the output ATO is put in a high impedance state.

TESTING FEATURES

- An additionnal amplifier allows the observation of the Rx band-pass filter output on pin RxFO.
- A direct input to the Tx band-pass filter and to the IF filter (TxIFI) is selected when TEST3 = 1.
- The 1 second normal duration of the Tx to Rx mode automatic switching is reduced to 488µs and the 1.5 second watch-dog time out is reduced to 46.3µs when TEST2 = 1.
- When TEST1 = 1 the Tx to Rx mode automatic switching is desactivated and the functional mode of the circuit is fully controlled by Rx/$\overline{Tx}$.
- TEST4 is a reset input which allows an undelayed control of RSTO and of the internal state of the circuit.

POWER SUPPLIES WIRING PRECAUTIONS

The ST7537 has two positive power supply terminals (AV_{DD},DV_{DD}) and two ground terminals (AV_{SS},DV_{SS}) in order to separate internal analog and digital supplies. The analog and digital terminals of each supply pair must be connected together externally for proper operation.

The V_{DD} must be protected against short-circuit for proper operation.

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
AV_{DD}/DV_{DD}	Supply Voltage (1)	- 0.3, + 12	V
V_I	Digital Input Voltage	DV_{SS} - 0.3, DV_{DD} + 0.3	V
V_O	Digital Output Voltage (microcontroller interface)	DV_{SS} - 0.3, DV_{CC} + 0.3	V
V_O	Digital Output Voltage (PABC and $\overline{PABC}$)	DV_{SS} - 0.3, DV_{DD} + 0.3	V
I_O	Digital Output Current	- 5, + 5	mA
V_I	Analog Input Voltage	AV_{SS} - 0.3, AV_{DD} + 0.3	V
V_O	Analog Output Voltage	AV_{SS} - 0.3, AV_{DD} + 0.3	V
I_O	Analog Output Current	- 5, + 5	mA
P_D	Power Dissipation	500	mW
T_{oper}	Operating Temperature	0, + 70	°C
T_{stg}	Storage Temperature	- 55, + 150	°C

7537-02 TBL

Notes : 1. The voltages are referenced to AV_{SS} and DV_{SS}.
2. Absolute maximum ratings are values beyond which damage to device may occur. Functional operation under these conditions is not implied.

GENERAL ELECTRICAL CHARACTERISTICS

(A/DV_{DD} = 10V, A/DV_{SS} = 0V, DV_{CC} = 5V and 0°C ≤ T_{amb} ≤ 70°C, unless otherwise specificied)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
AV_{DD} DV_{DD}	Supply Voltage		9.5	10	10.5	V
AI_{DD} + DI_{DD}	Supply Current			30		mA
DV_{CC}	Digital Output Supply Voltage		4.75		5.25	V
DI_{CC}	Digital Output Supply Current			1.5		mA
V_{IH}	High Level Input Voltage	Digital Inputs	4.2			V
V_{IL}	Low Level Input Voltage	Digital Inputs			0.8	V
V_{OH}	High Level Output Voltage	I_{OH} = -100µA • Digital Outputs • Digital Outputs PABC and $\overline{PABC}$	 4.9 9.8			 V V
V_{OL}	Low Level Output Voltage	I_{OL} = 100µA • Digital Outputs • Digital Outputs PABC and $\overline{PABC}$			 0.1 0.2	 V V
DC	Duty Cycle	MCLK Output, C_L = 15pF	40		60	%

7537-03 TBL

TRANSMITTER ELECTRICAL CHARACTERISTICS

(A/DV_{DD} = 10V, A/DV_{SS} = 0V, DV_{CC} = 5V and 0°C ≤ T_{amb} ≤ 70°C, unless otherwise specificied)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
VTAC	Max Carrier Output AC Voltage	R_L = 5.6kΩ $R_L(AV_{SS})$ = 5.6kΩ R(ATO, PAFB) = 1kΩ	0.8	1.0	1.3	V_{RMS}
HD2	Second Harmonic Distortion			- 50		dB
HD3	Third Harmonic Distortion			- 60		dB
FD	FSK Peak-to-peak Deviation			1200		Hz

7537-04 TBL

RECEIVER ELECTRICAL CHARACTERISTICS

(A/DV$_{DD}$ = 10V, A/DV$_{SS}$ = 0V, DV$_{CC}$ = 5V and 0°C ≤ T_{amb} ≤ 70°C, unless otherwise specificied)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_{IN}	Input Sensitivity			1	10	mV$_{RMS}$
V_{IN}	Maximum Input Signal				2	V$_{RMS}$
R_{IN}	Input Impedance		15			kΩ
GRx	Receive Gain	f = 132.45kHz		20		dB
BER	Bit Error Rate (1)	S/N = 15dB, S = 10mV$_{RMS}$, N : white		10^{-5}	10^{-3}	
t_{DEM}	Demodulation Time	Alternate 0 , 1 sequence		3		T bit
V_{CD}	Carrier Detection Level	f = 132.45kHz, sine wave		5	10	mV$_{RMS}$

7537-05.TBL

Note 1 : This parameter is guaranteed by correlation

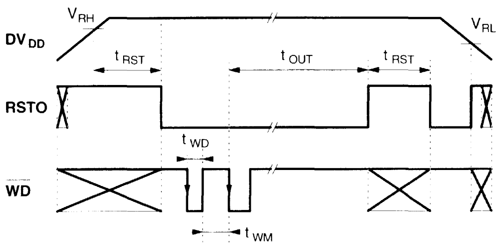
ADDITIONAL DIGITAL AND ANALOG FUNCTIONS ELECTRICAL CHARACTERISTICS

(A/DV$_{DD}$ = 10V, A/DV$_{SS}$ = 0V, DV$_{CC}$ = 5V and 0°C ≤ T_{amb} ≤ 70°C, unless otherwise specificied)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_{RH}	High Level Reset Voltage	See Figure 2		7.9		V
V_{RL}	Low Level Reset Voltage	See Figure 2		7.6		V
t_{RST}	Reset Time	See Figure 2	50			ms
t_{WD}	Watch-dog Pulse Width	See Figure 2	500			ns
t_{WM}	Watch-dog Pulse Period	See Figure 2	800			µs
t_{OUT}	Watch-dog Time Out	See Figure 2			1.5	s
t_{CD}	Carrier Detection Time	See Figure 3	3		6.5	ms

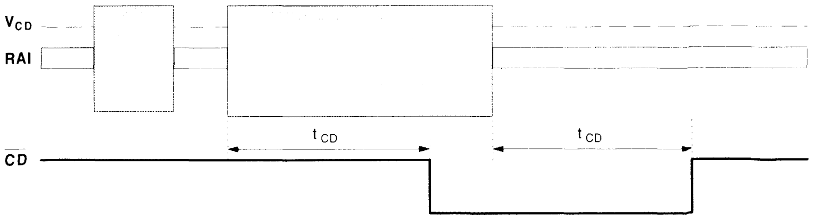
7537-06.TBL

Figure 2 : Reset and Watch-dog Timing Chart



7537-04.EPS

Figure 3 : Carrier Detection Timing Chart



FILTER TEMPLATES

Receive and Transmit Filter

Frequency (kHz)	Gain (dB)		
	Min.	Typ.	Max.
92			- 30
126.45	- 5	- 3	- 2
Ref 132.45		0	
138.45	- 5	- 3	- 2
180			- 30

Intermediate FrequencyFilter

Frequency (kHz)	Gain (dB)		
	Min.	Typ.	Max.
2.4			- 35
4.3	- 4	- 3	- 1
Ref 5.4		0	
6.5	- 5	- 3	- 2
11.6			- 35

7537-07 TBL

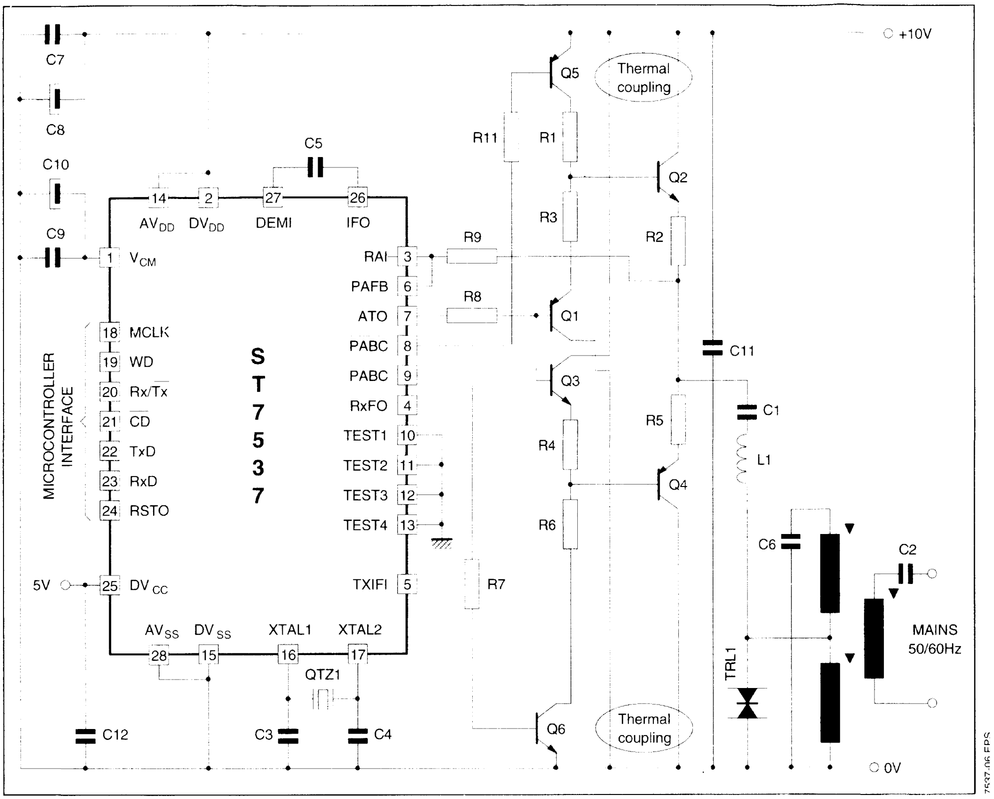
APPLICATION SCHEMATIC INFORMATIONS

RESISTORS			CAPACITORS			
R1	180Ω		C1	1μF		Ceramic 50
R2	2.2Ω		C2	470nF		Paper, class X2
R3	2.2Ω		C3 (2)	22pF	10%	Ceramic 10V
R4	2.2Ω		C4 (2)	22pF	10%	Ceramic 10V
R5	2.2Ω		C5	100nF	10%	Ceramic 10V
R6	180Ω		C6	6.8nF	5%	Plastic Film 50V
R7	47kΩ		C7	100nF		Ceramic 10V
R8	1kΩ		C8	2.2μF		
R9	1kΩ	5%	C9	100nF		Ceramic 10V
R11	47kΩ		C10	2.2μF		
INDUCTOR			C11 (1)	100nF		Ceramic 10V
L1	10μH	≅ 1.5Ω	C12 (1)	100nF		Ceramic 10V
TRANSISTORS			**TRANSIL**			
Q1 : 2N2907 Q2 : 2N2222 Q3 : 2N2222 Q4 : 2N2907 Q5 : 2N2907 Q6 : 2N2222			TRL1 : SGS-THOMSON P6KE6V8CP			
			TRANSFORMER			
			TR1 : TOKO T1002 N			
			CRYSTAL			
			QTZ1 : 11.0592MHz parallel resonance			

7537-08 TBL

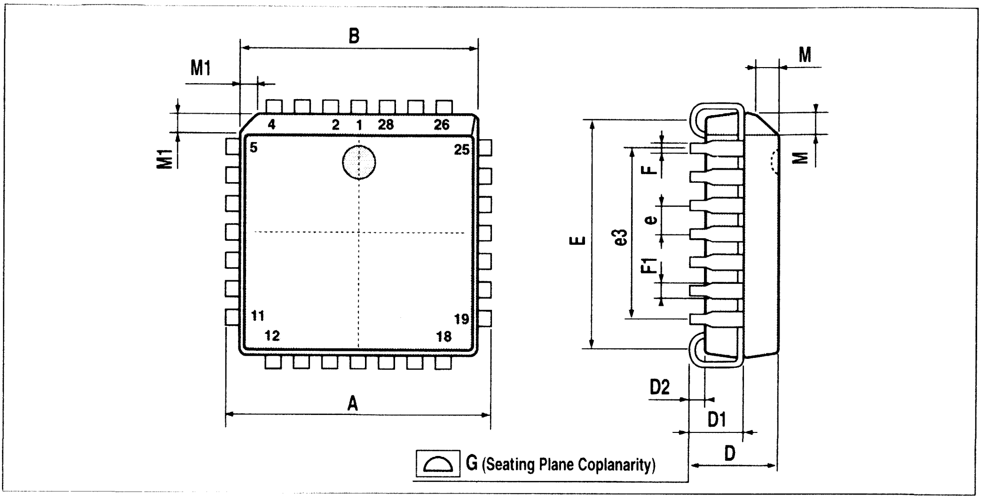
Notes : 1. These capacitors might not be necessary if the overall power supplies decoupling is sufficient.
2. The value of these capacitors depends on the crystal parameters.

APPLICATION SCHEMATIC DIAGRAM



PACKAGE MECHANICAL DATA

28 PINS - PLASTIC CHIP CARRIER



PMPLCC28.EPS

Dimensions	Millimeters			Inches		
	Min.	Typ.	Max.	Min.	Typ.	Max.
A	12.32		12.57	0.485		0.495
B	11.43		11.58	0.450		0.456
D	4.2		4.57	0.165		0.180
D1	2.29		3.04	0.090		0.120
D2	0.51			0.020		
E	9.91		10.92	0.390		0.430
e		1.27			0.050	
e3		7.62			0.300	
F		0.46			0.018	
F1		0.71			0.028	
G			0.101			0.004
M		1.24			0.049	
M1		1.143			0.045	

PLCC28.TBL

ST7537 APPLICATION NOTE

SGS-THOMSON
MICROELECTRONICS

APPLICATION NOTE

ST7537
POWER LINE MODEM APPLICATION

By Joël HULOUX and Laurent HANUS

SUMMARY **Page**

I - FOREWORD : HOME AUTOMATION CONCEPT

Kenneth P. Wacks, consultant to the home automation industry, has written an article clearly defining the concept of home automation. An extract is given below :

"... Over the past six years a new industry called "home automation" has been developing. This industry will create the next generation of consumer appliances. The primary value added by home automation is the integration of products and services for household use. A few small companies are marketing home automation systems. Large companies and institutions are exploring this emerging industry to determine the market potential.

A communication network in the house will provide the infra-structure for linking appliances, sensors, controllers, and control panels inside the house. This has become feasible by tailoring the communications technologies developed for office automation to the home environment.

I.1 - Home Automation Appliances

In home automation, the term "appliances" refers not only to the familiar kitchen, audio/video, and portable appliances, but also to the components of a heating and cooling system, a security system, and lighting features. Home automation covers a broad range of products and services intended for consumer use. These items are expected to share some common attributes, among which are :

- Emphasis on Subsystems :
  Most appliances in houses today are self-contained in metal or plastic cabinets. Each appliance operates independently to the others. Each appliance has a different set of user control.
  Appliances in a home automation environment are able to exchange data. This allows appliances to be grouped into subsystems. Examples range from familiar subsystems, such as security and audio/video systems, to sophisticated lighting controls with preset dimming levels for banks of lights. A future subsystem might permit a washing machine or a dish-washer to request that a water heater preheat water when needed or when the energy cost is lowest.
- Incorporation of Communications Standard :
  Some of the subsystems mentioned already exist. However, the components of each are interconnected using custom-designed technologies and custom wiring. Home automation standards will relieve the manufacturer of the need to invent an ad hoc communications protocol and to provide wiring for data signals.
- Diverse Locations :
  Once communications standards are developed, manufacturers will be able to locate components of appliances outside the cabinet. Control panels could be placed where convenient for the user, not necessarely mounted on the cabinet. Related appliances, such as clothes washer and a clothes dryer, could share a control panel so the knobs and dials are consistent and easier to operate.

I.2 - The Growth of the Industry

Communications technology and standards play important roles in forecasting the home automation industry. However, the development of applications to use these technologies will set the growth rate that simplify routine activities, spark a desire consumers, or save money.

Thus, the growth rate of the home automation industry is ultimately determinated by the actions of appliance manufacturers. Key among these decisions are :

- Adoption of an Emerging Communications Standard :
  The appliance manufacturers will greatly influence the establishment of a particular communications standard. They may even force an amalgamation of standards from among the current contenders.
- Create New Appliances or Appliances Features :
  The development of standard communications methods can benefit manufacturers and consumers. The design staff would more likely be encouraged and financed to invent appliances that depend on the exchange of data if a communications infra-structure were already in the house..."

II - INTRODUCTION

In the latest generation of home automation systems, appliances can exchange information by transmitting data over the domestic mains wiring. As a result there is no need to install extra control cables and appliances can be connected to the "network" simply by plugging them into the nearest wall socket. Apart from the obvious saving in installation cost, this virtual network also makes modification and enhancement very simple since new devices just have a wall socket to be instantly connected to the network.

What makes these systems feasible is a new dedicated modem integrated circuit, the SGS-THOMSON ST7537 Home Automation Modem IC, developed specifically for this new high volume consumer market as part of a European Community "ESPRIT" project on domestic automation.

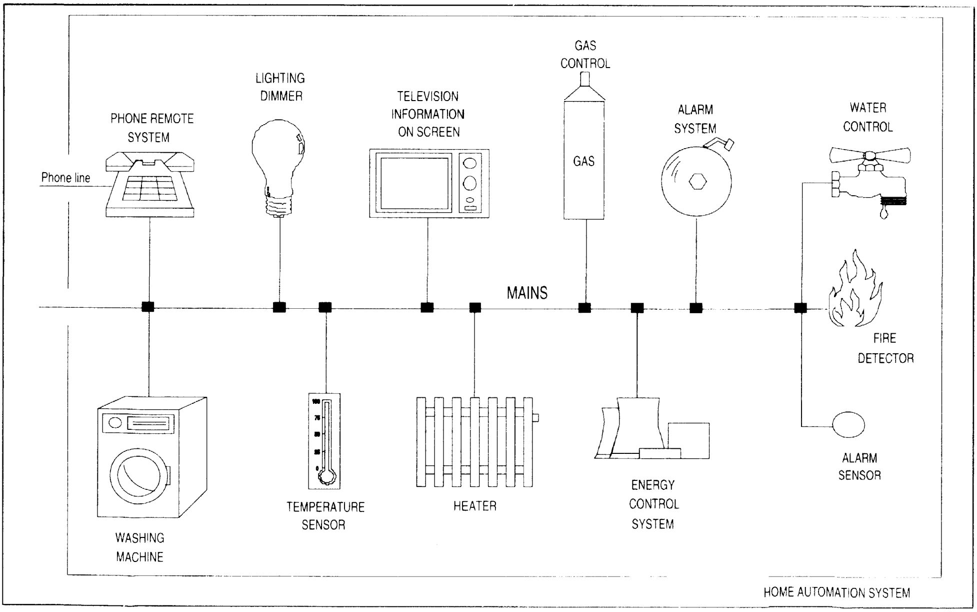
A typical household scenario is shown in Figure 1, where various appliances, sensors, utility controls, a telephone interface and a TV screen display are all connected to the power line using power line modem.

If this automated house catches fire the detector will send a warning message over the line. This will be picked up by the gas control which can cut off the gas supply, by an alarm system that can alert anyone in the house, and even by the telephone interface that can call the emergency services.

The telephone interface also allows the householder to give instructions to appliances from outside. You might, for example, phone home and tell the air conditioner to precool certain rooms at a specified time.

Where there is a limit on energy consumption, or where demand energy pricing is used (now that the technology is available this is likely to be applied extensively in future) various appliances can negociate power requirements through an energy control system. For example, a washing machine can agree with the heating system when it can start a cycle to avoid sudden and unnecessary peaks of demand.

Figure 1 : Typical Household Scenario



III - THE ELECTRICAL NETWORK

Research has been done on the communication properties of the residential power circuit by J.B O'Neal Jr. An extract of his written work is presented below :

"... The primary objective in most residential power line carrier systems is to communicate information from one power outlet in a residence to another. The communication medium, therefore, consists of everything connected on power outlets. This includes house wiring in the walls of the building, appliance wiring, the appliances themselves, the service panel, the triplex wire connecting the service panel to the distribution transformer and the distribution transformer itself. Since distribution transformers usually serve more than one residence, the loads and wiring of all residences connected to the same transformer must be included.

III.1 - Impedance of Power Lines

The most extensive data on this subject has been published by Malack and Engstrom of IBM (Electromagnetic Compatibility Laboratory), who measured the RF impedance of 86 commercial AC power distribution systems in six European countries (see Figure 2).

These measurements show that the impedance of the residential power circuits increases with frequency and is in the range from about 1.5 to 80Ω at 100kHz. It appears that this impedance is determined by two parameters - the loads connected to the network and the impedance of the distribution transformer. The loads at a neighbor's residence can effect this impedance. Wiring seems to have a relatively small effect. The impedance is usually inductive.

For typical resistive loads, signal attenuation is expected to be from 2 to 40dB at 150kHz depending on the distribution transformer used and the size of the loads. Moreover, it may be possible for capacitive loads to resonate with the inductance of the distribution transformer and cause the signal attenuation to vary wildly with frequency.

III.2 - Noise

The principal source of noise is caused by appliances connected to the same transformer secondary to which the power line carrier system is connected. The two primary sources of noise will be triacs used in light dimmers and universal motors. Triacs generate noise synchronous with the 50Hz power signal and this noise appears as harmonics of 50Hz. Universal motors found in mixers, sewing machines, and sanders also create noise, but it is not as strong as light dimmer noise, and not generally synchronous with 50Hz. Furthermore, light dimmers are often left on for long periods of time whereas universal motors are used intermittently. The Figure 3 shows noise sources as well as background noise in a typical residential environment.

Figure 2 : Aggregate European Power Line Impedance (by Malack and Engstrom)

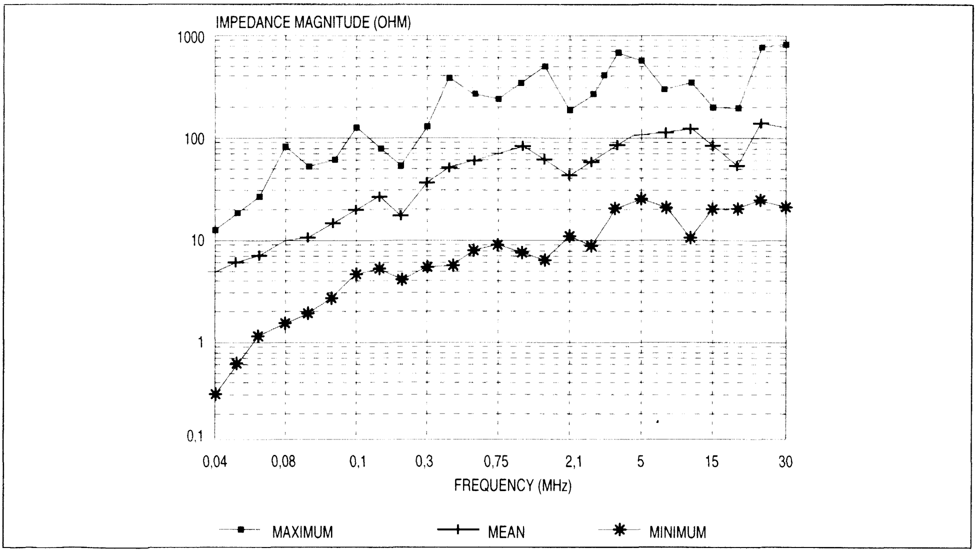
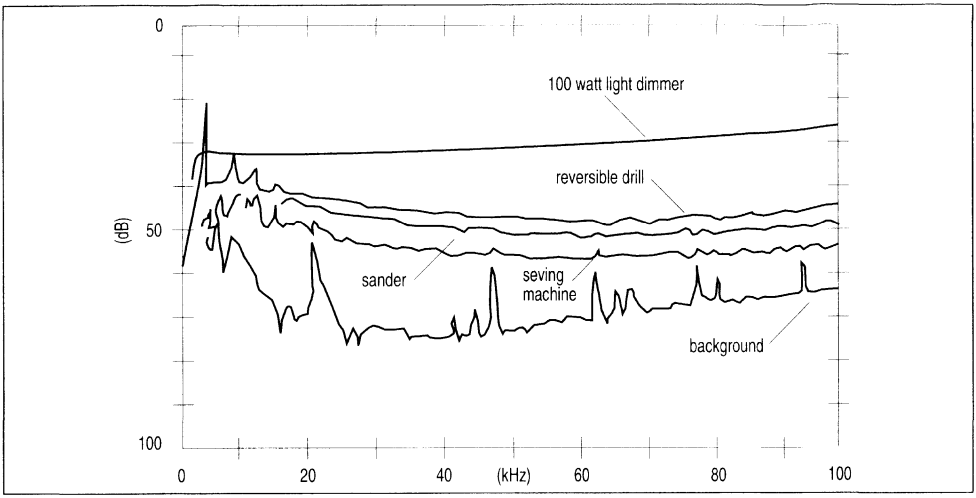


Figure 3 : Voltage spectra for 3 universal motors compared to light dimmers operating into the 60Hz power circuit (by Vines, Trussel, Gale and O'Neal Jr.)



III.3 - Standing Waves

Standing wave effects will begin to occur when the physical dimensions of the communication medium are similar to about one-eight of a wavelenght, which is about 375 and 250 meters at 100 and 150kHz respectively. The length of the communication path on the secondary side of the power distribution system will be determined primarily by the length of the triplex wire connecting the residences to the distribution transformer. Usually, several residences use the same distribution transformer. It would be rare that a linear run of this wiring would exceed 250 meters in length although the total length of branches might occasionally exceed 250 meters. Thus standing wave effects would be rare at frequencies below 150kHz for residential wiring..."

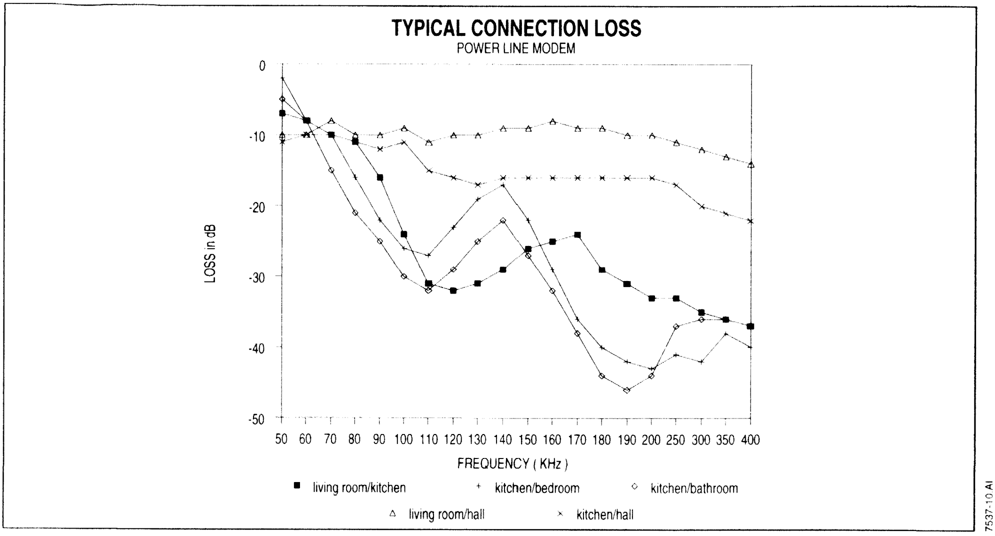
III.4 - Typical Connection Loss (see Figure 4)

We notice two classes of value at a transmit frequency of about 130kHz :

- from 10dB to 15dB : in this case, the transmitter and the receiver are connected to the same branch circuit.
- from 20dB to 30dB : in that case, the transmission path goes from one branch circuit to another through the service panel which induces an additional attenuation of 10dB to 20dB.

Therefore, the transmit range of a home automation system depends on the physical topology of the electric power distribution network inside the building where the system is installed.

Figure 4 : Static Attenuation for Several Paths (by Daniel CHAFFANJON)



IV - ST7537 POWER LINE MODEM

Fabricated in analog CMOS technology, the ST7537 transmits and receives data up to 1200bps in half duplex mode using a carrier frequency of 132.45kHz, complying with Europe's CENELEC EN 50065 standard (which specifies the use of 125kHz to 140kHz carrier frequencies for home automation) and US FCC regulations (which specifies the use of carrier frequencies lower than 450kHz).

Frequency-shift keying is used for transmission, a fundamental design choice that makes it possible to achieve rugged transmission in a very noisy electrical environment at an affordable cost for high volume consumer markets. Among the alternatives, amplitude-shift keying is too susceptible to noise and spread-spectrum, though theoretically more reliable, requires complex and costly circuits. Moreover, field trials in a critical remote utility meter reading application have proven the dependability of the SGS-THOMSON approach.

Included on the chip are all of the functional blocks necessary for the transmission and reception of data over power lines. In addition to this IC the only external components needed are a line driver and a transformer, plus, of course, the microcontroller that prepares and interprets message data.

Transmit data enters the FSK modulator asynchronously with a nominal intra-message data rate of 1200bps. Inside the modulator, the data is transformed into two frequencies (133.05kHz for a "0" and 131.85kHz for a "1"), derived from an inexpensive 11.0592MHz crystal.

The modulated signal from the FSK modulator is filtered by a switched-capacitor bandpass filter (TX bandpass) to limit the output spectrum and to reduce the level of harmonic components. The final stage of the transmit path consists of an operational amplifier which needs a feedback signal from the power amplifier.

In the receive section, the incoming signal is applied at the RAI input (with a typical sensitivity of 1mV$_{RMS}$) where it is first filtered by a switched-capacitor bandpass filter with a pass band of around 12kHz, centered on the carrier frequency. The output of the filter is amplified by a 20dB gain stage which provides symetrical limitation for overvoltages. The resulting signal is downconverted by a mixer which receives a local oscillator synthesized by the FSK modulator block.

Finally, an intermediate frequency bandpass filter whose central frequency is 5.4kHz improves the signal-to-noise ratio before entering the FSK demodulator. The coupling of the intermediate frequency filter output to the FSK demodulator input is made by an external capacitor which cancels the receive path offset.

In the ST7537 there are two important additional functions: the carrier detector and the watchdog. Carrier detection is needed because in practically all applications more than two appliances will be connected to the power line. Before attempting to transmit, an appliance must first check that there is no carrier present, and if there is, it must wait and retry later.

The watchdog function is provided to ensure that the modem's control micro is functioning correctly. Software in the micro must include instructions that send a pulse to the watchdog input of the ST7537 at least once every 1.5s. If no negative transition is observed at this input for 1.5s a reset signal is generated to restart the micro. This watchdog monitor scheme ensures that any disruption caused by glitches are quickly corrected.

V - DEMO BOARD FEATURES

Power line interface

The power line interface has been designed in order to follow the CENELEC EN 50065-1 and US FCC specification. It has to amplify and filter the output signal of the ST7537.

Test pin

It is possible to program the different test modes of the ST7537 with the switches SW1, SW2, SW3 and SW4 corresponding to TEST1, TEST2, TEST3 and TEST4. The most important test mode is TEST1 which allows continuous transmission.

RS232C interface

On the application board, there is an RS232C interface allowing you to debug your system. This interface is connected to the ST7537 by four switches SW5, SW6, SW7 and SW8.

Remark : It is mandatory to provide the watchdog clock to the ST7537.

Wrapping area

You can wire your application and do its debug by connecting relevant digital signals to SW5, SW6, SW7 and SW8 (pin not used) and watchdog, master clock and RSTO.

VI - HARDWARE DESCRIPTION

VI.1 - About CENELEC Specifications

The CENELEC specifications are given for an imaginary network (50Ω/ 50µH + 5Ω) simulating the power line. This network looks like a 54Ω impedance at a transmit frequency of 132.45kHz. The transmitted signal is measured in relation to a reference of this network (see Annexe B). With this configuration, some of the specifications are :
- maximum output level : 116dBµV
- harmonics level of less than 46dBµV mean.

In this chapter, the transmitted signal is measured between the phase and the neutral of the simulated power line. Then, the measured voltages are twice the ones measured with CENELEC test configuration. Thus, it is necessary to add 6dBµV to the specifications given above :
- maximum output level : 122dBµV
- harmonics level of less than 52dBµV mean.

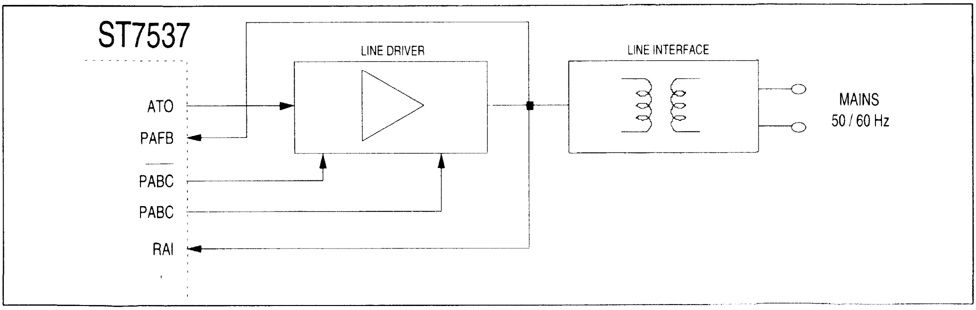
Henceforth, these values will be used .

VI.2 - Power Line Interface

The power line interface connects the ST7537 to the power lines and meets the CENELEC and FCC specifications. It has the following functions :
- in transmit mode : to amplify and filter the transmit signal (ATO) from the ST7537
- in receive mode : to provide received signal from powerlines to the receive input (RAI) of the ST7537
- protection against spikes and overvoltages.

It is composed of a line driver and a line interface as it is shown in Figure 5.

Figure 5 : Power Line Interface Description



In transmit mode, the power line interface has to be able to drive, via the line interface, power lines with impedances from 1 to 100Ω. The line interface is not only used to put signals on the power line. It is also used as a bandpass filter, in order to reduce the harmonics of the transmit signal to a level of less than 52dBμV .

In receive mode, the line driver is switched off to avoid the low output impedance of the line driver attenuating the received signals and to save energy costs.

VI.2.1 - The Line Driver

The line driver has to amplify the output signal (ATO) of the ST7537 (see Figure 6).

First, a normal Push-Pull amplifier has been set up with two bipolar transistors Q4 (2N2222) and Q3 (2N2907). These types of transistors (2N2222 and 2N2907) have been chosen as they are cheap and widely used.

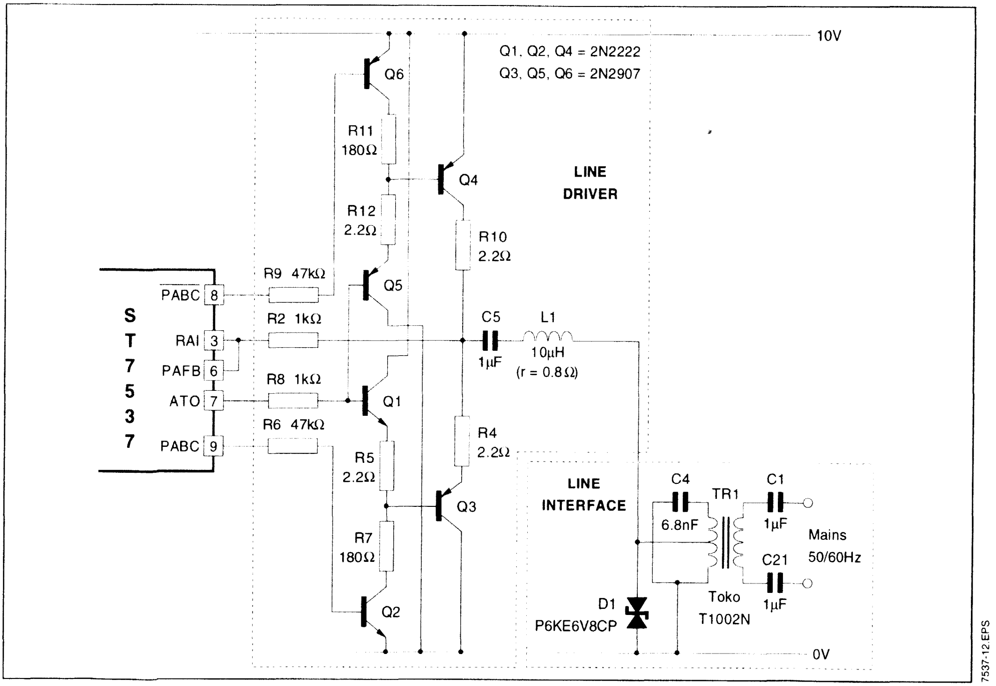
The resistors R4, R5, R10 and R12 degenerate the emitter of Q5, Q4, Q1, Q3 in order to define the bias current of the ouput branch independently of the mismatch of the transistors. The Push-Pull is polarized with two common collector amplifiers composed of Q1 (2N2222) and Q5 (2N2907). As far as resistors R7 and R11 are concerned, their value (180Ω) has been defined to obtain the optimum performances of the amplifiers thus define the bias current of the system.

The bipolar transistors Q2 (2N2222) and Q6 (2N2907) are used to switch off the power amplifier during the receive mode, thanks to the ST7537 output signals PABC and $\overline{\text{PABC}}$ which follow the Rx/Tx mode.

In order to avoid thermal runaways, it is mandatory to connect thermically Q1/Q4 and Q3/Q5. This is possible since the collectors of the transistors used are connected to the metal package. Consequently, both transistors will have the same temperature.

Main characteristics of the line driver :
- voltage gain = 1
- high input impedance
- low output impedance

Figure 6 : Power Line Interface Schematics



VI.2.2 - The Line Interface

In order to adapt the line driver to the power line, a transformer is used (see Figure 6). This transformer has :

- to isolate the rest of the interface from the power line
- to put the transmit signal on the power line
- to extract the received signal from the power line
- to filter 50Hz/60Hz signal coming from the power line
- to filter the harmonics of the transmit signal.

The used transformer is a TOKO T1002N. It has two primary windings and one secondary winding. The ratios of these windings are 4:1:1 (turns). Typical values of the transformer are :

- L1t windings : 9.4µH
- L4t windings : 140µH.

The primary windings of the transformer are used to create a bandpass filter. The resonance frequency is set at the transmit frequency with C4. This capacitor is in parallel with the primary winding (1t/4t). The equivalent value for those two windings can be calculated according to :

$$Leq = L1t + L4t + 2M$$

$$M = k \cdot \sqrt{L1t \cdot L4t}$$

With the given values :

$$k = 1/2^{1/2}$$
$$M = (9.4\mu H \cdot 140\mu H / 2)^{1/2} = 25.7\mu H$$
$$Leq = L1t + L4t + 2 \cdot M = 200.7\mu H$$

The resonance frequency of this LC network is dependant of C4 and Leq according to :

$$Fres = \frac{1}{2\pi \cdot \sqrt{Leq \cdot C4}}$$

$$C4 = \frac{1}{Leq \cdot (2\pi \cdot Fres)^2}$$

For Fres = 132.45kHz → C4 = 7.2nF (6.8nF is chosen since it is the nearest capacitor value available).

The capacitor C4 must be very linear in order avoid harmonic distortion. That's why a KS (styroflex or NPO ceramic capacitor) capacitor has been used.

In order to filter the 50Hz/60Hz signal from the powerlines, C1 is used. The capacitor filters the low frequencies (50Hz/60Hz) and lets the high (Transmit) frequencies pass. It is a class X2 capacitor. These capacitors have a short circuit protection, which is absolutely necessary. Indeed if a short circuit in the capacitor occurs, the 50Hz/60Hz filtering is lost, and the powerline interface will be destroyed, or worse, danger might occur for persons working with the interface and the ST7537. Moreover, since the TOKO transformer cannot overcome higher than 800V spikes, the safety norms are not met and the capacitor C1 is required to comply with them. An additional capacitor C21 is used as the phase location is unknown.

As a final protection against any possible spikes, a transil (TRL 1) is used. It is a 6.8V bidirectional type. If a voltage greater than 6.8V appears, voltage between pins of the system will be set to 6.8V, protecting the other parts of the power line interface from damage.

R1 is added to discharge C1 after disconnecting the interface from the powerline. Without this resistor, C1 will not be discharged and schock hazard might occur if someone touches the powerline connector. This resistor is only useful in evaluation systems. In all other cases where disconnection from the power line never takes place, R1 can be removed, saving undesired energy loss.

VI.2.3 - The Power Line Interface

The complete power line interface has been described in the two preceding parts. The interface has to be connected to the ST7537 as described in Figure 7.

The ATO and RAI are the analog output and input from/to the ST7537. The control of the transmit/receive mode is made with PABC and $\overline{PABC}$ signals from the ST7537. A high output (+10V) on PABC line selects the transmit mode, whereas a low output (0V) selects the receive mode.

The "pwr" outputs are the power line connections. On the application board, these connections are located close to C1 and the transformer in order to avoid long tracks carrying high voltage.

VI.2.4 - Performances of the power line interface

The following tests have been done on the power line interface :

- output impedance of the powerline interface versus the frequency
- Bit Error Rate (BER) test
- spectrum analysis of the transmit signal.

VI.2.4.1 - OUTPUT IMPEDANCE OF THE POWER LINE INTERFACE VERSUS THE FREQUENCY

The output impedance of the power line interface is measured with an impedance analyzer as it is shown in Figure 8. The board is set in receive mode.

The results are given in annexe B.

Test equipment : 41924 LF Impedance Analyzer
5Hz-13MHz (Hewlett Packard)
Test conditions : T = +25°C

Figure 7 : Power Line Interface Inputs and Outputs

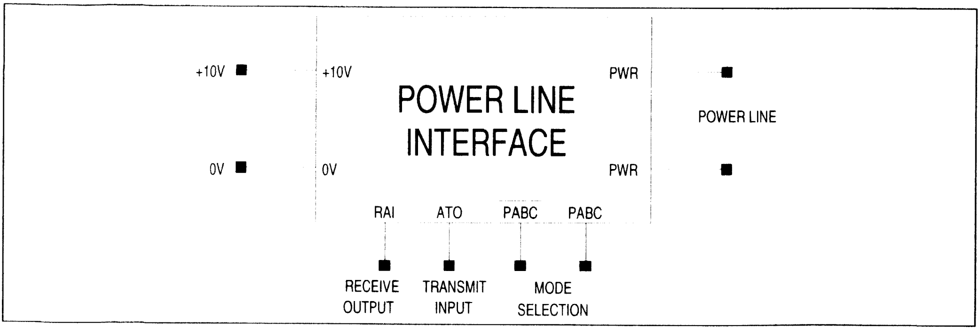
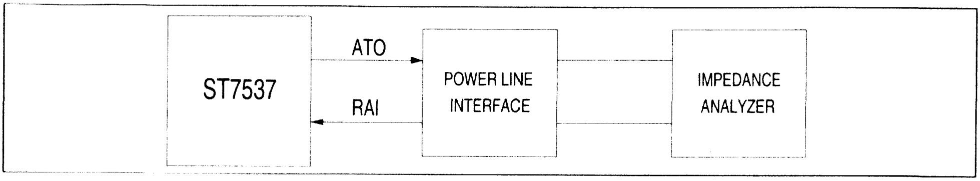


Figure 8 : Output Impedance Measurement Configuration



VI.2.4.2 - BER TEST

Two boards are required : one for the transmission, the other one for the reception.

White noise is added to the ATO transmit output of the ST7537 thanks to a mixer. The aim is to measure the BER under different Signal/Noise ratio conditions. The mixed signal is transmitted to the RAI receive input of the modem. The digital signal injected in TxD is a 2^{15}-1 pseudo-random pattern long, generated by a bit error rate analyzer (with internal 1.2kHz asynchronous clock).

In the reception board, a 1.2kHz clock (CRX) is built thanks to the ST7537 MCLK clock. The received digital signal RxD is amplified (RxDL) and synchronized with the CRX clock. Both of them (CRX and RxDL) are analyzed by the BER analyzer.

The measurements are made with different RAI input level. The Figures 10 and 11 gives respectively the B.E.R with a RAI input level of 10.023mV$_{RMS}$ and 1.14mV$_{RMS}$.

Conclusion

Under the test conditions of the ST7537 specification (RAI = 10mV$_{RMS}$ and S/N = 15dB) the BER is 4.10-7. With an RAI input level of 1.14mV$_{RMS}$ the BER is around 10-4 with the same S/N ratio. Therefore, the ST7537 is able to communicate with low input signal level of about 1mV$_{RMS}$. This test illustrates the high sensitivity of the power line modem.

In Figure 10, the measured BER (with an RAI input level of 10mV$_{RMS}$) is compared with the theorical BER of a conventional BFSK modulator/demodulator.

Test equipment :	SI7703B BER analyzer Rhode and Schwartz noise generator
Test condition :	T = +25°C

Figure 9 : BER Test Configuration

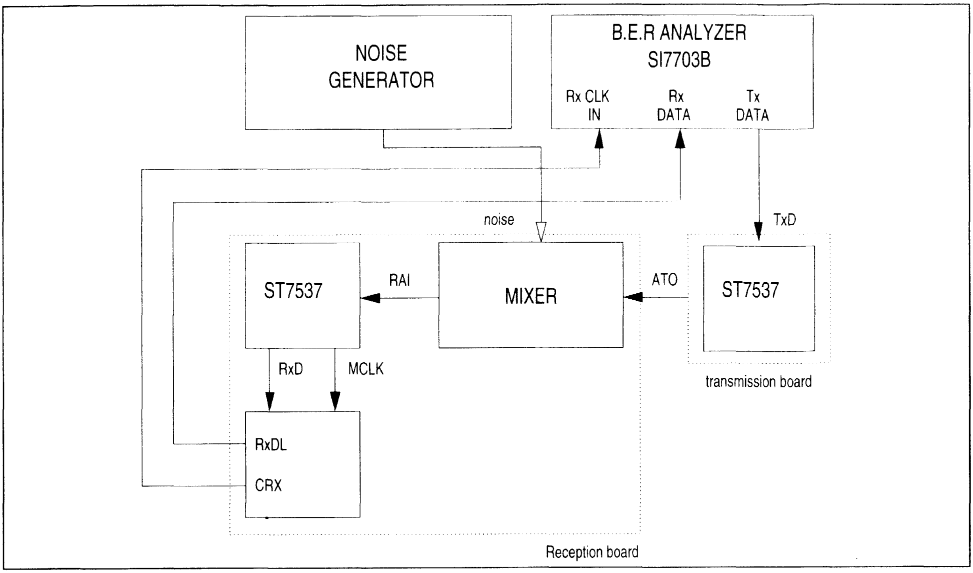


Figure 10 : BER Test for an RAI Input Amplitude of 10.023mV$_{RMS}$

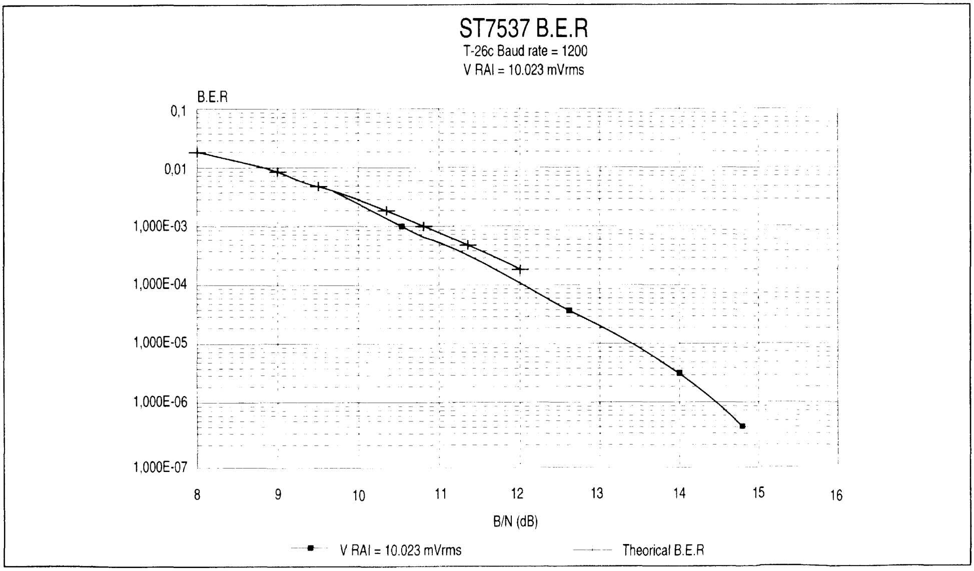
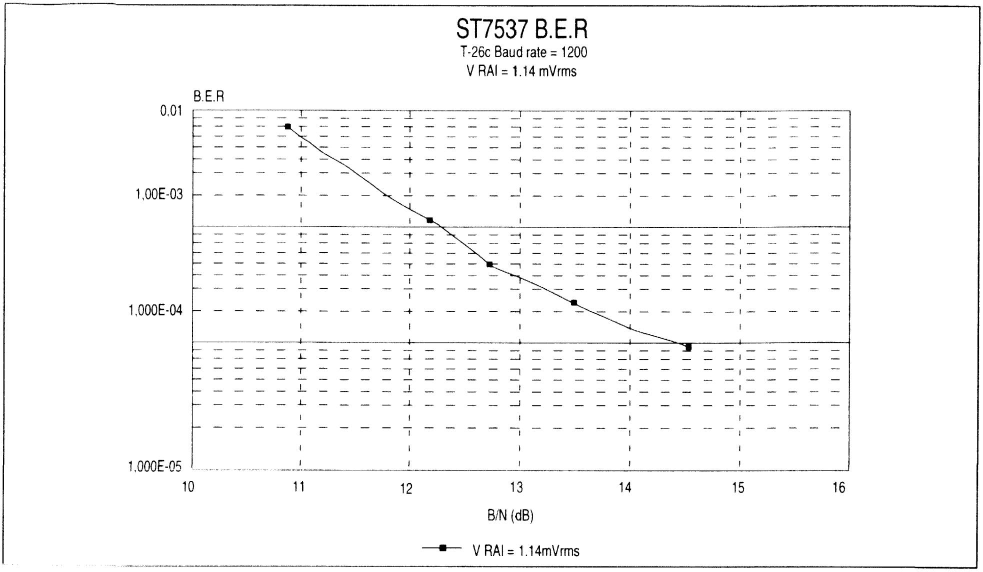


Figure 11 : BER Test for an RAI Input Level of 1.14mV$_{RMS}$



VI.2.4.3 - TRANSMIT SIGNAL SPECTRUM ANALYSIS

The transmit output signal of the power line interface is measured with the power line simulated by resistors : R = 1, 5, 10, 50, 100Ω.

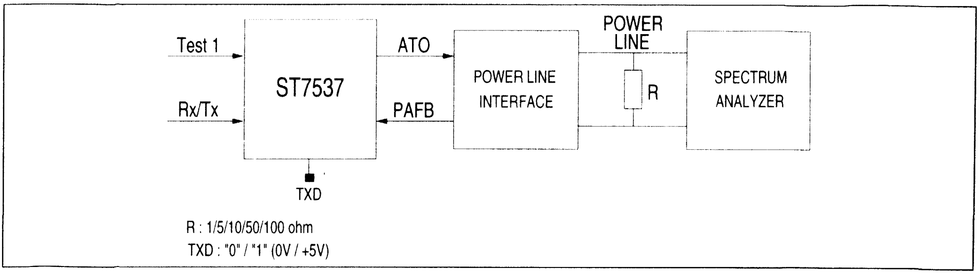
A spectrum analyzer is used to display the output signal frequency spectrum of the power line interface (see Figure 12).

In a first design of the board, a 2.2Ω resistor was used instead of the inductance L1. In this configuration, whatever the power line impedance, the output level was at least 106dBμV up to 119dBμV (see Figure 13). Thus no communication problems had been noticed during the test session.

To improve the frequency spectrum of the transmit signal, the resistor has been replaced by an inductance L1 of 68μH, 1.6Ω (see Figures 14 and 15). However, tests on a real site showed that the transmit level was very low with this inductance in case of low power line impedance : with an impedance of 1Ω, the output level is 87dBμV, so that communication difficulties occur. At the transmit frequency (132.45kHz), the inductance looks like an impedance of about 56Ω, which introduces significant attenuations on the transmit signal compared to those induced by the 2.2Ω resistor.

To improve the output signal amplitude, the inductance value must be modified. A compromise has to be found between filtering the pertubation voltages and lowering the impedance of the inductance at the transmit frequency. An inductance of 10 μH (0.8Ω) has been chosen which looks like an impedance of 8Ω at 132.45kHz frequency (see Figures 16 and 17).

Figure 12 : Spectrum Analysis Configuration



R : 1/5/10/50/100 ohm
TXD : "0" / "1" (0V / +5V)

Figure 13 : Output Transmit Level (dBµV) with 2.2Ω Resistor

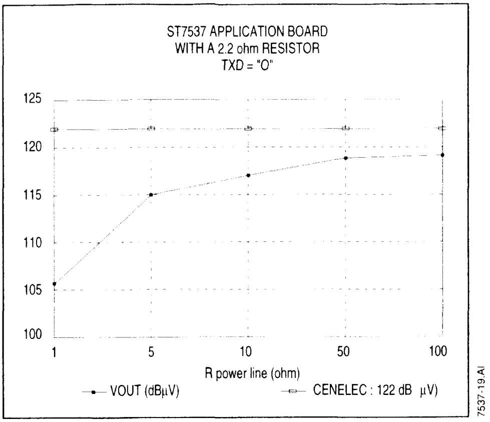


Figure 14 : Output Transmit Level (dBµV) with 68µH Inductance

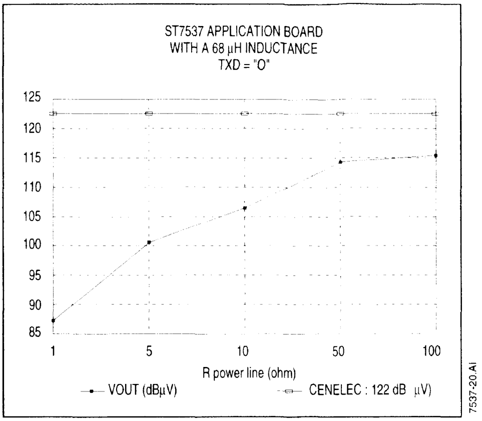


Figure 15 : Second and Third Harmonics Level (dBµV) with 68µH Inductance

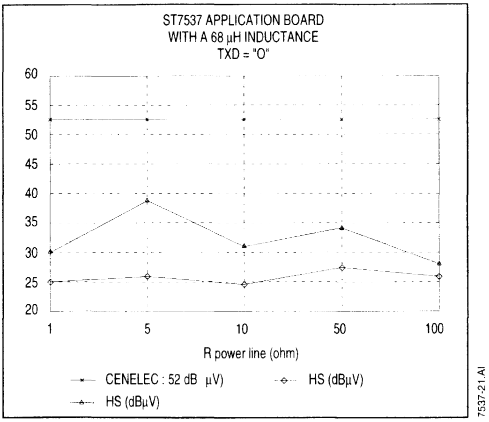


Figure 16 : Output Transmit Level (dBµV) with 10µH Inductance

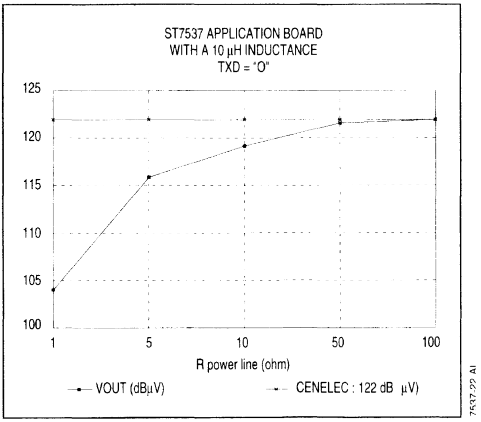
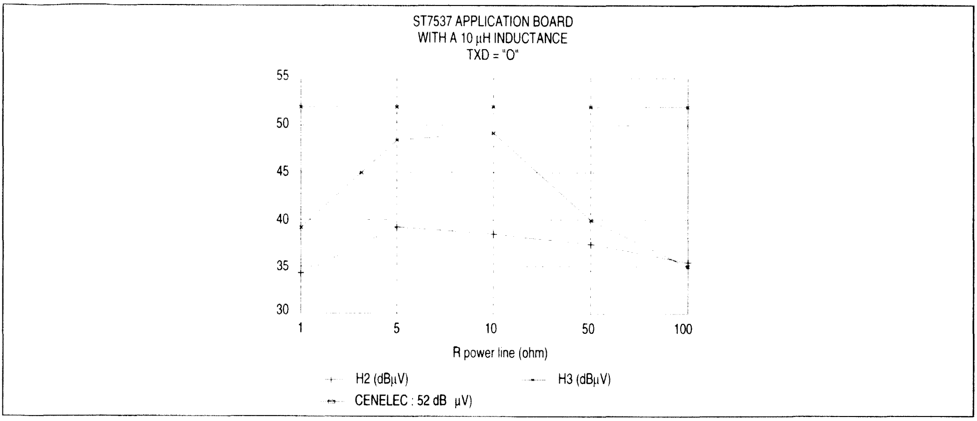


Figure 17 : Second and Third Harmonics Level (dBμV) with 10μH Inductance



VOUT/H2 and VOUT/H3 variations with the 10μH inductance versus the power line impedance are given in Annexe C.

Test results (with L1 = 10μH)	CENELEC specifications	FCC specifications
VOUT < 122 dBμV	VOUT < 122 dBμV,	
H2 < 39 dBμV	H2 < 56 dBμV mean	H2 < 48 dBμV (extended to 60 dBμV)
H3 < 49 dBμV	H3 < 52 dBμV mean	H3 < 48 dBμV (extended to 60 dBμV)
VOUT/H2 > 70 dB		
VOUT/H3 > 65 dB		

Conclusion

With L1 = 10 μH, the required harmonics level is reached and the output voltage is smaller than 122 dBμV. Therefore, the power line interface is fully operating according to the CENELEC and FCC specifications. Moreover, for very low power line impedances, the output transmit level is high enough to ensure a good communication quality.

Test equipment : 3585A Spectrum Analyzer 20Hz-40MHz (Hewlett Packard)
Test conditions : T = +25°C

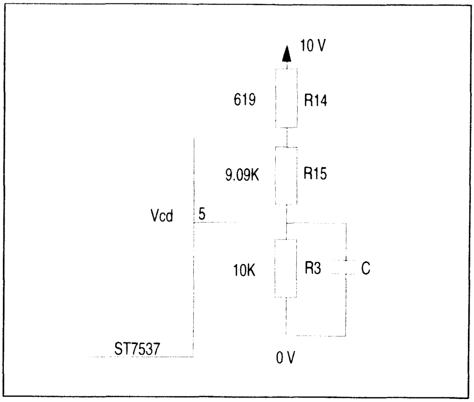
VI.3 - Carrier Detect

The carrier detect output ($\overline{CD}$) is driven low when the input signal amplitude on RAI is greater than (V_{CD} - V_{CM})/Grx for at least T_{CD} (typically 4ms). When the input signal disappears or becomes lower than (V_{CD} - V_{CM})/Grx, $\overline{CD}$ is held low for at least Tcd before returning to a high level. Vcd input is the carrier detection threshold voltage which is set externally and Grx is the receive gain (typically 20dB). The minimum detection level has to be $10mV_{RMS}$ in order to meet the CENELEC specifications.

$$\frac{V_{CD} - V_{CM}}{Grx} = 10mV \Rightarrow V_{CD} = 5.089V$$

This voltage is set with a resistor bridge as it is shown in the Figure 18.

Figure 18 : Carrier Detect



Since V_{CD} has to be very precise, 1% resistors must be used. With the resistors chosen :

$$\boxed{V_{CD} = \frac{R13}{R3 + R4 + R13}}$$

V_{CD} = 5.07V
⇒minimum detection level = 5.8mV$_{RMS}$

Calculation of the V_{CD} voltage range according to resistor tolerance :

$$\boxed{V_{DC\,max.} = \frac{R3max.}{R3max. + R4min. + R15min.} = 5.12V}$$

⇒ minimum detection level = 9.85mV$_{RMS}$

$$\boxed{V_{DC\,min.} = \frac{R3min.}{R3min. + R4min. + R15max.} = 5.02V}$$

⇒ minimun detection level = 1.7mV$_{RMS}$

Thus, with 1% resistors whose value is given above, the CENELEC specification is met.

The graph, given in Annexe D, represents the minimum amplitude of the received signal which can be detected (which corresponds to $\overline{CD}$ = 0) according to the frequency. For frequencies near 132.45kHz, the minimum level complies with the calculation made above. Thus input signals at a frequency of 133.05kHz (high logic level) and 131.85kHz (low logic level) can de detected at a very low level. For frequencies smaller than 129kHz or greater than 150kHz, the detection is made at a very high level of input signal. Therefore, only significant frequencies received signals are detected.

In the final version of the ST7537, the carrier detector level will be fixed inside the device at 5mV, so that there will be no need of external components on V_{CD}.

VI.4 - Communication with a RS232C Interface

The application board can be connected to a Personal Computer (PC) thanks to the RS232C interface. As the electrical levels of the RS232 port (±12V) do not match the electrical levels of the ST7537 (TTL levels 0/+5V), a MAX232 is used to make communication possible. This device has two RS232 receivers to convert RS232 levels into TTL levels and two RS232 transmitters to convert TTL levels into RS232 levels.

The connections between the ST7537 and the RS232 interface are given in figure 19. Not all the pins from the RS232 port are used. The RXD, TXD and Carrier Detect (CD) signals are directly converted. The Request To Send (RTS) line is used to set the ST7537 in receive or transmit mode, but also to give the PC a Clear To Send (CTS) signal. The Data Set Ready (DSR) line is connected to the Data Terminal Ready (DTR) line. This simulates the transmission of the DSR signal by the power line modem when the PC is ready. The RI output of the PC is only used for telephone network modems, and therefore it is not connected.

If the RS232 port of the PC is used, it is necessary to provide the board with a watchdog clock (e.g : 1kHz) in order to get the PC communication working. A suggested clock generator is given Figure 20. It uses a NE555 timer working in astable mode.

Figure 19 : Connections between ST7537 and RS232 Interface

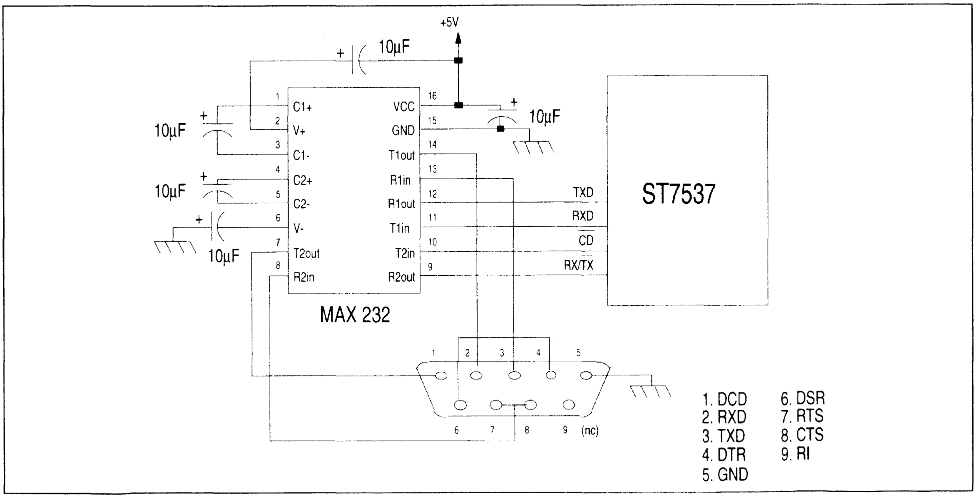
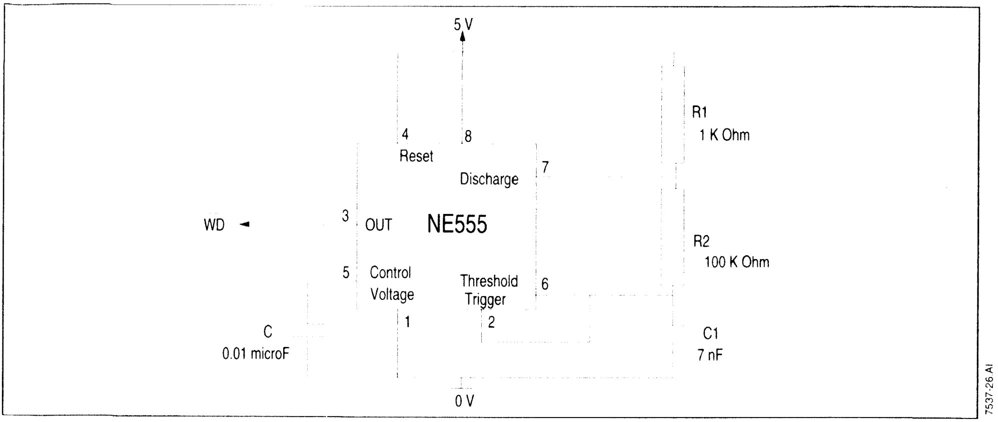


Figure 20 : Watchdog Clock



The output HIGH time of the clock is :
$t_H = 0.693*(R1 + R2)*C1$

The output LOW time of the clock is :
$t_L = 0.693*(R2)*C1$

Thus the total period T is : $T = t_H + tL$

The frequency of oscillation is : $f = 1/T = 1/(t_H + t_L)$

Calculations provides the following results :
- R1 = 1kΩ
- R2 = 100kΩ
- C1 = 7nF.

VI.5 - Demo-board Power Consumption

The demo-board tested is composed of :
- the ST7537
- the power line interface
- the MAX232
- the watchdog clock generator

The power consumption is measured both in transmit and receive modes. In both modes, the power line has been simulated with a 1Ω resistor (worst case simulation). In transmit mode, the data input (TxD) was a logical 0 (0V) (see Figure 21).

Current consumption

Power line impedance	:	1Ω
Input voltage	:	10V
Transmit mode		
- ST7537	:	30mA$_{RMS}$
- power line interface	:	112mA$_{RMS}$
- MAX232	:	2.5mA$_{RMS}$
- watchdog clock generator	:	2.5mA$_{RMS}$
Total	:	147mA$_{RMS}$
Receive mode		
- ST7537	:	30mA$_{RMS}$
- MAX232	:	2.5mA$_{RMS}$
- watchdog clock generator	:	2.5mA$_{RMS}$
Total	:	35mA$_{RMS}$

Total power consumption

Transmit mode : 1.47W
Receive mode : 350mW

A ST7537 system is almost always in receive mode, and the transmit mode only lasts 1 second. The need of energy is therefore limited.

In receive mode the line driver is switched off, which explains the low consumption.

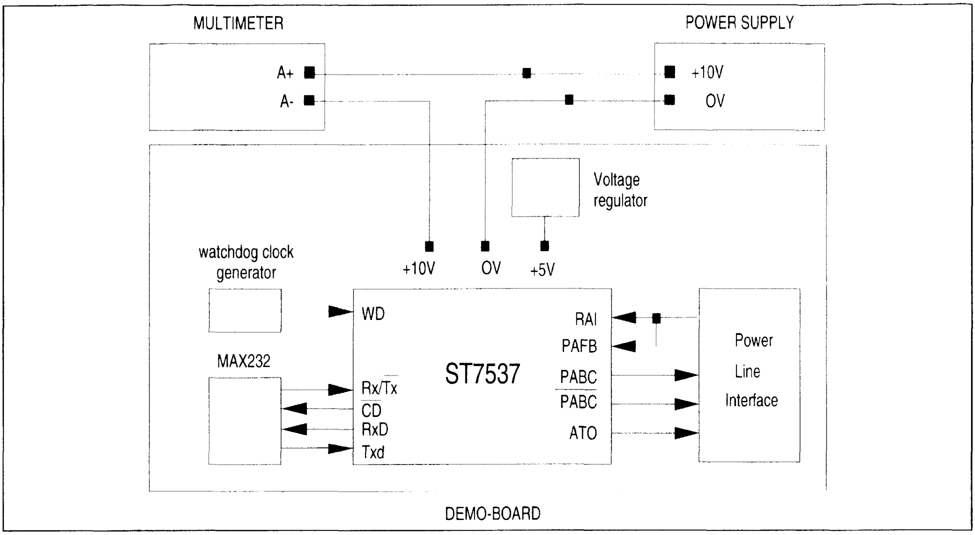
Test equipment : FLUKE 45
(Dual Display Multimeter)
Test conditions : T = +25°C

VI.6 - Demoboard Communicating Application

The ST7537 power line modem enables you to design "communicating" appliances, which meet your specific requirements and comply with the CENELEC specifications. Equipped with a single low-cost ST90E28 microcontroller, it makes it possible to build a "smart" home network, where each device is able to use any information required either if it is local (sensors) or remote (inside any other communicating appliance).

This paragraph is intended to provide design basics for the implementation of the ST90E28 on the ST7537 demoboard.

Figure 21 : Power Consumption Test



VI.7 - Overview of the ST90E28 MCU

The ST90E28 microcontroller chosen to equip the ST7537 demoboard is a 16Kbyte program memory EPROM version with 256 bytes of RAM and 256 bytes of register file. Within this file, 224 general purpose registers are available as RAM, accumulators or index pointers, allowing code efficiency. This MCU has an internal clock generator, a 16-bit watchdog timer for system integrity, a powerful serial communications interface (SCI) with included baud rate generator and outstanding character search capability, and a 16-bit multifunction timer for complex user applications; it provides a reset input and up to 36 input/output pins, including 7 external interrupts and a non-maskable interrupt.

Most of the instructions take 14 clock cycles: with a clock frequency of 11.0592MHz, one instruction lasts about 90ns. Connected to the ST7537, the microcontroller has to deliver a maximum bit rate of 1200 bauds: one bit is at least 833µs long.

VI.8 - Implementation of the ST90E28 MCU

Two configurations have been set up, one for the slave appliances, and one for the master system. Both versions will have their address initialized in the software in this first release. Besides, they use one data output to display information about the main program execution by means of a led: you know that the main program is running well, when this led is blinking as the appliance is powered on. The main differences between the two controllers are the input/ouput facilities.

The slave configuration provides an ouput that switches a load. This load will be simulated by a LED (see Figure 22).

The master configuration provides a 3-bit command input to control the slaves. This command will be simulated by a KEYBOARD : one key is available for each slave, and one specific key enables the user to supervise all the slaves inside a room at once. This configuration also uses a 3-bit data output to let you know whether a particular slave is on, or whether the room is lit up. This information will be displayed by one led attached to the key dedicated to a particular device (see Figure 23). All the slaves addresses will be stored in the master version of the software.

Furthermore, both configurations need a 7 bit data exchange with the ST7537 : clock, transmit data, receive data, reset, Rx/$\overline{Tx}$ control lines (see Figure 24). No external component is needed to interface the microcontroller with the power line modem, allowing cost savings.

Figure 22 : Slave Configuration

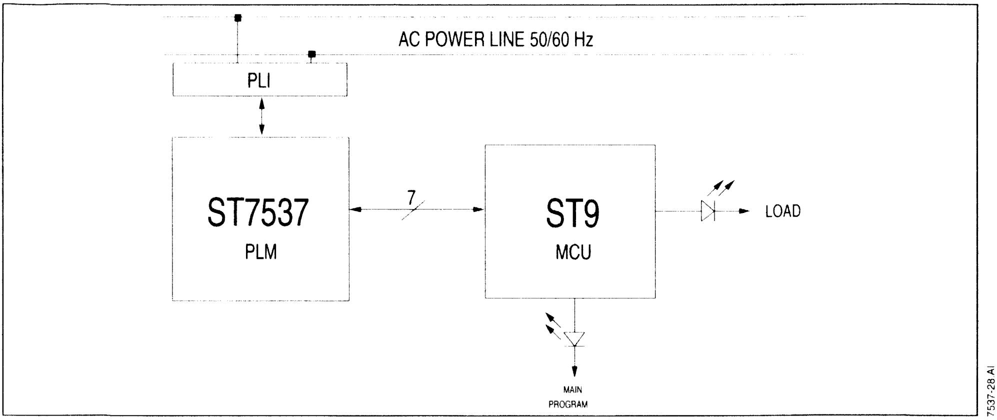


Figure 23 : Master Configuration

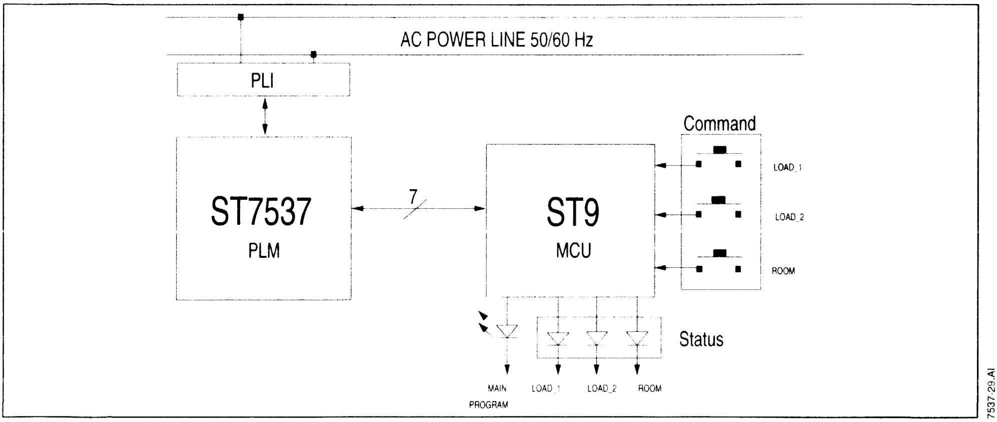
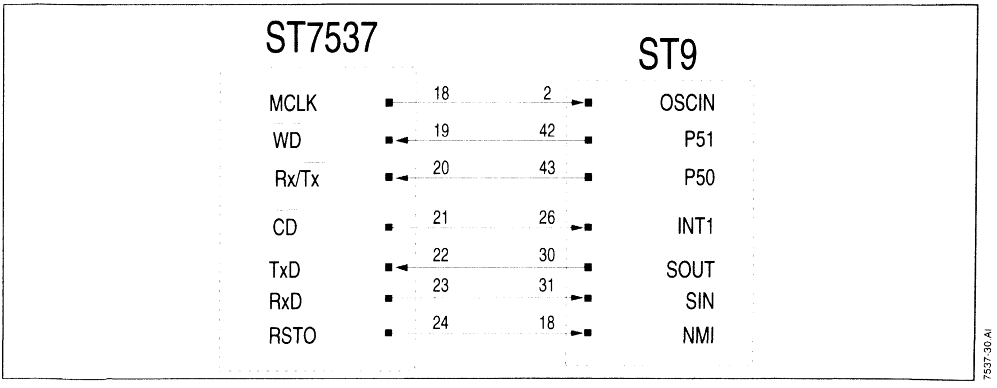


Figure 24 : Interface between ST7537 and ST90E28



- OSCIN (Pin 2) : The MCU oscillator is driven with the PLM master clock, so that no additional crystal is needed. In this case, the oscillator output pin must stay unconnected.
- Port 5 bit 1 (Pin 42) : This output bit provides the PLM watchdog input with negative transitions, before the timeout end is reached. The watchdog pulses must be at least 500ns wide with a period of at least 800µs and up to 1.5s.
- Port 5 bit 0 (Pin 43) : This output controls the Rx/Tx mode. When this bit is 0, the transmit mode is set, otherwise the receive mode is selected. Remember that the ST7537 switches automatically in the receive mode, when this bit is held at 0 longer than 1s.
- INT1 (Pin 26) : The PLM carrier detect signal channels through this external interrupt input pin, which is triggered on falling edge. On signal detection, the carrier detect output is driven low and generates an interrupt request.
- SOUT (Pin 30) : The microcontroller provides the ST7537 with Tx data by means of the SCI output.
- SIN (Pin 31) : The ST7537 provides the microcontroller with Rx data through the SCI input.
- NMI (Pin 18) : The PLM reset output signal acts as an MCU external watchdog, in order to detect hardware or software failures. This signal channels through the MCU external non maskable interrupt input pin, which is triggered on rising edge. When the power supply is too low or when no negative transition occurs on the PLM watchdog input for more than 1.5s, the reset ouput is driven high and generates a top level interrupt request, which resets the microcontroller. As for the MCU internal watchdog timer, the watchdog mode is disabled, so that a second 16-bit programmable timer is available for customer applications.

VI.8.1 - Applicative Pin Configuration

- V_{SS} (Pin 1) : Digital Circuit Ground
- V_{DD} (Pin 21) : Main Power Supply Voltage +5V. A decoupling capacitor of 47µF is connected between V_{DD} and V_{SS} pins. The V_{DD} of the microcontroller should be connected also to the DV_{CC} of the ST7537 in order to reference the digital level of the ST7537.
- RESET (Pin 3) : This input is active low. To restart the microcontroller, the reset key has to be pressed (see Figure 25). A capacitor (2.2µF) will keep the input low for a minimum startup period, whereas a pull-up resistor (100kΩ) will keep it high for normal operation.
- Display Output : Light emitting diodes are used to display data. The maximum current provided by each output pin is 0.8mA. Therefore the serial resistor R has a minimum value of 4.7kΩ (see Figure 26 : current = (4.2-0.6)/4.7e3 = 0.77mA).

The slave configuration uses 2 display output pins.

Port 2 bit 3 (Pin 25)	:	blinking led
Port 2 bit 5 (Pin 27)	:	load (slave led)

The master configuration uses 4 display output pins.

Port 2 bit 3 (Pin 25)	:	blinking led
Port 2 bit 5 (Pin 27)	:	load 1 status
Port 2 bit 6 (Pin 28)	:	load 2 status
Port 5 bit 5 (Pin 38)	:	room status

Figure 25 : Reset Command

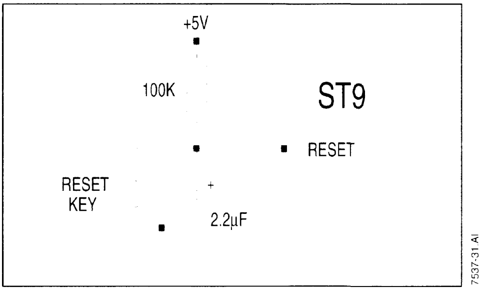
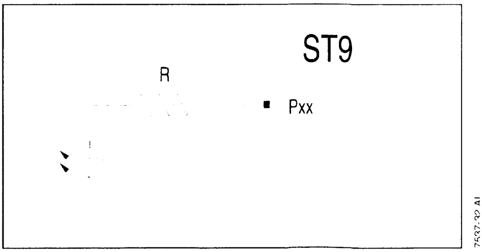


Figure 26 : Display Output

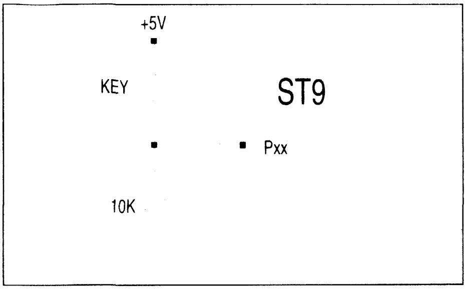


- Keyboard Input : Switch keys are used to enter commands. The keyboard pin is active high (see Figure 27). A pull-down resistor of 10kΩ keeps the input low, whereas a key press holds it high for active operation.

  The master configuration uses 3 keyboard input pins.

Port 5 bit 2 (Pin 41)	:	load 1 command
Port 5 bit 3 (Pin 40)	:	load 2 command
Port 5 bit 4 (Pin 39)	:	room command

Figure 27 : Keyboard Input



VI.8.2 - Power Consumption

The power consumption of each configuration has been measured. Both master and slave boards were connected to the AC power mains : the slave led and all master status leds are switched ON by pressing the master room key (worst case simulation).

The current consumption is measured with a digitizing oscilloscope (channel 2) by means of a serial resistor, which value is small enough to avoid big supply voltage drops (about 1Ω typically).

A dual tracking power supply provides each board with the same power voltage, which value is displayed on a multimeter.

Test equipment : Fluke 45 Multimeter, Tektronix TDS460 Digitizing Oscilloscope

Test conditions : R = 1.04Ω , Valim = +10.006 V
T = +25°C

- Slave board : the oscilloscope is triggered on the falling edge of the Carrier Detect ($\overline{CD}$) signal displayed on channel 1 (see Figure 28). Therefore, the current consumption is displayed on channel 2 in receive mode on stand-by ($\overline{CD}$ = 1) and active ($\overline{CD}$ = 0) states.

  Current consumption (Rx mode) :+146mA$_{RMS}$

  Power consumption :

  (+10.006V - 1.04Ω · 146mA) · 146mA = +1.44W

  Slave board current consumption test results (see Figure 29)

Channel 1 :	Carrier Detect signal
Channel 2 :	Supply current

- Master board : the oscilloscope is triggered on the falling edge of the Rx/$\overline{Tx}$ signal on channel 1 (see Figure 30). The current consumption is displayed on channel 2 in both receive and transmit modes.

  Current consumption :

Rx mode	+160mA$_{RMS}$
Tx mode	+230mA$_{RMS}$

  Power consumption :

  Rx mode (+10.006V - 1.04Ω · 160mA) · 160mA = +1.57W

  Tx mode (+10.006V - 1.04Ω · 230mA) · 230mA = +2.25W

Master board current consumption test results (see Figure 31)

Channel 1 :	Rx/$\overline{Tx}$ signal
Channel 2 :	Supply current

Figure 28 : Slave Board Current Consumption Test

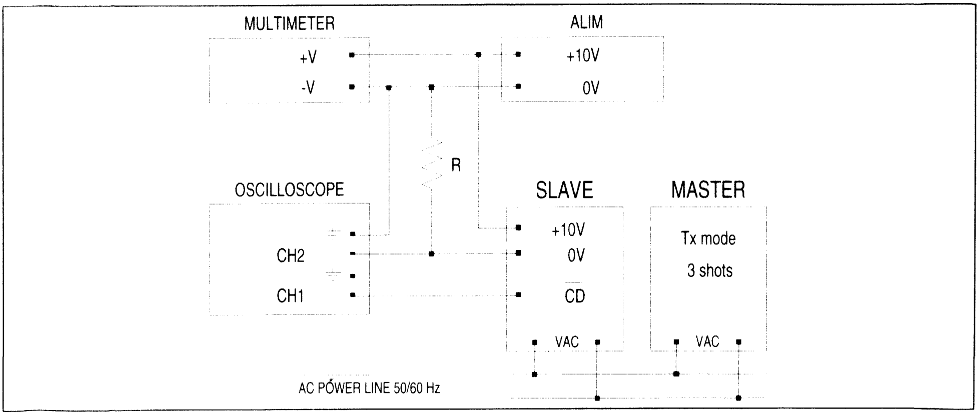


Figure 29 : Slave Board Current Consumption Test Results

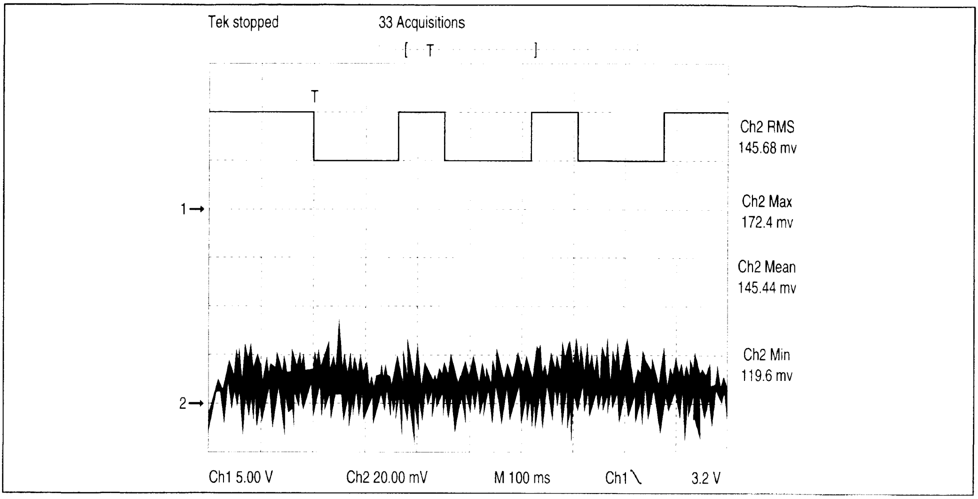


Figure 30 : Master Board Current Consumption Test

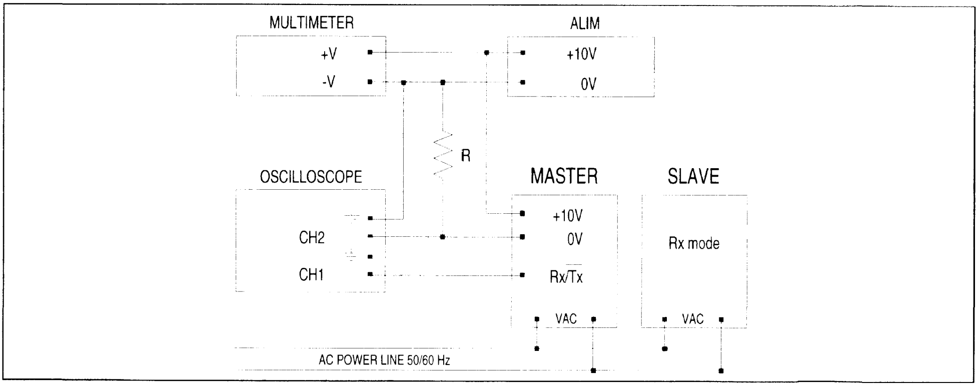
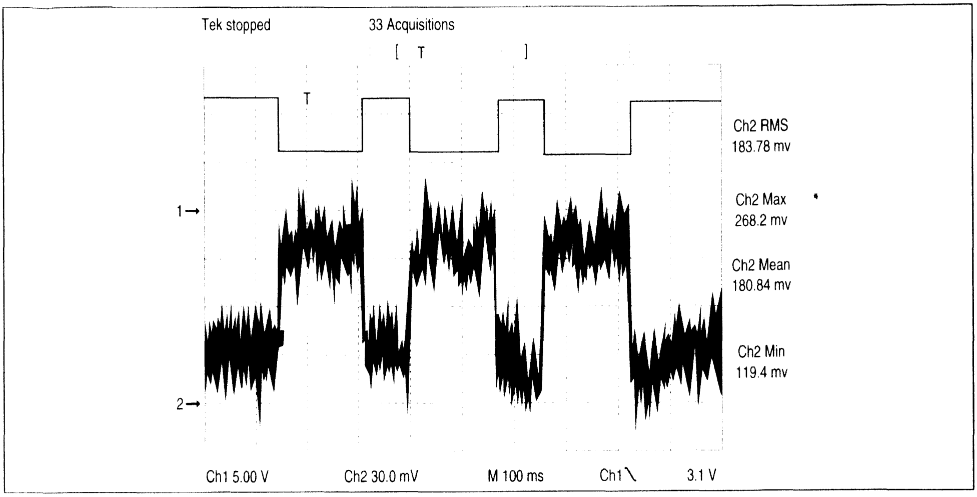


Figure 31 : Master Board Current Consumption Test Results



V.9 - Power Supply

V.9.1 - Power supply features

The power supply features are :

- one reference voltage of 10 V_{DC}
- output current of 400 mA

The 5 V_{DC} voltage needed for the numeric part of the application is provided by a voltage regulator LM 7805, which already exists on the board.

The power supply schematic is given in Figure 32 :

The LM317T regulator is ajustable between 1.2V and 37V thanks to the R1 & R2 resistors. It could be replaced by a +10V regulator.

V.9.2 - Power supply sizing

The rectified voltage between pins of the capacitor C1 is shown in Figure 33 :

Uca = transformer secondary voltage (V_{RMS})

Ucc = voltage between pins of the capacitor C1

Urtt = ripple voltage

U = minimum voltage which has to exist between input and output of the voltage regulator

Us = output power supply voltage

Ud = rectifier diodes voltage drop

I = output power supply current

Hypothesis :

- I = 400mA
- Umin = 3V
- Ud = 1V

The minimum voltage the transformer has to provide is :

$$\boxed{Uca = (Us + Umin + Urtt + 2Ud) / 2}$$

The ripple voltage is :

$$\boxed{Urtt = 10 * I / C1 \text{ (with I in mA and C1 in } \mu F)}$$

Figure 32 : Power Supply Schematics

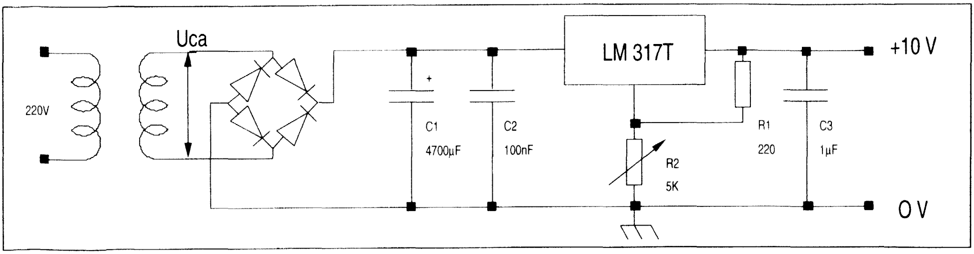
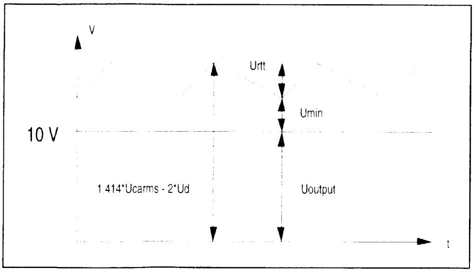


Figure 33 : Rectified Voltage Parameters



V.9.3 - Using a 2x6 V secondary voltage transformer

The transformer must be able to supply I = 400mA, so that a 5 VA transformer is required.

The maximum value of Urtt is :

$$\boxed{\begin{array}{c}\text{Urtt max} = 2^*\text{Uca} - \text{Us} - \text{Umin} - 2^*\text{Ud}\\ \text{Urtt max} = 2\text{V}\end{array}}$$

$\Rightarrow$ C1 min = 10*I / Urtt max
C1 min = 2000µF

We choose a C1 capacitor value of : 4700µF

The maximum voltage Vmax which can be applied between C1 pins has to be higher than the maximum secondary voltage of the transformer. Therefore, with a safety margin of 25% :

$$\boxed{\text{Vmax} = (2 * \text{Uca}) * 1.25 = 21.2\text{V}}$$

The maximum power dissipated by the voltage regulator is :

$$\boxed{\text{Pd} = \text{U} * \text{I}}$$

U = 2*Uca - Us - Urtt - 2*Ud
Urtt = (10 * 400) / 4700 = 0.85V
$\Rightarrow$Pd = 1.6W

In short, the power supply sizing is :

- secondary voltage of the transformer : 2x6V
- 5 VA transformer
- C1 = 4700µF with a maximum voltage of 25V between its pins.

VII - PC SOFTWARE

With the application board, we provide you a communication program written in Turbo C language which allows :

- to drive the RS232 interface
- to transmit data via power lines thanks to the ST7537
- to receive data from power lines thanks to the ST7537
- to process data
- to run character error test.

It is possible to transmit :

- characters
- text (maximum 80 characters)
- hexadecimal data (maximum 64 bytes)
- file.

The communication program allows you to run different types of communication :

- communication between 2 computers.
- communication between 2 ports COM on the same computer.

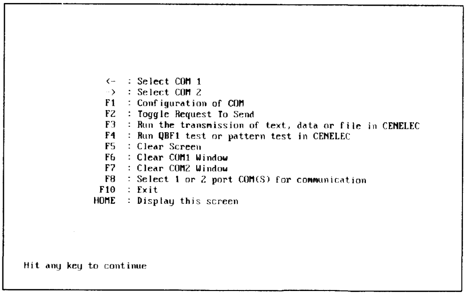
VII.1 - Software Running and Presentation

The description of what is executed in PLMCOM3 is given in annexe E (flow chart 1).

To run the software, type "PLMCOM3" and press ENTER. The presentation of the software is displayed.

Then, the following menu appears. It recalls the different command function keys used in the program (details about these keys are given in chapter VII.3).

Figure 34

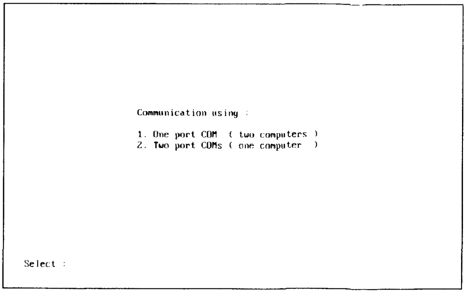


```
  <-  : Select COM 1
  ->  : Select COM 2
  F1  : Configuration of COM
  F2  : Toggle Request To Send
  F3  : Run the transmission of text, data or file in CENELEC
  F4  : Run QBF1 test or pattern test in CENELEC
  F5  : Clear Screen
  F6  : Clear COM1 Window
  F7  : Clear COM2 Window
  F8  : Select 1 or 2 port COM(S) for communication
 F10  : Exit
HOME  : Display this screen

Hit any key to continue
```

Hit any key to continue.

Next display :

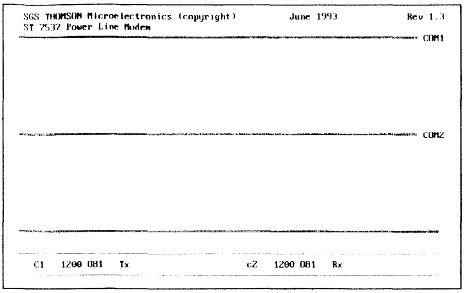
Figure 35



```
Communication using :

1. One port COM  ( two computers )
2. Two port COMs ( one computer  )

Select :
```

Press "1" if you want to select option 1 (1 port COM) or press "2" to select option 2 (2 port COMs). Option 2 is set by default.
When the option is chosen, the main screen appears. One of its configuration is given below.

Figure 36



```
SGS THOMSON Microelectronics (copyright)        June 1993         Rev 1.3
ST 7537 Power Line Modem
__________________________________________________________________ COM1

__________________________________________________________________ COM2

______________________________________________________________________
C1   1200 O81   Tx                    c2   1200 O81   Rx
```

It is divided into 3 parts :
- COM1 window
- COM2 window
- status port COM menu

The status port COM menu gives information about the two ports COM :
- C1 and c2 : indicates which port COM information refers to (the port COM selected is displayed in upper case letters)
- 1200 O81 : indicates the current RS232 port programming (example : 1200 O81 means 1200 bauds, Odd parity, 8 data bits and 1 stop bit)
- Rx and Tx : indicates that the port COM is used in receive mode (Rx) or in transmit mode (Tx)
- CER : appears only if the test mode is selected. It means that character error rate test is running
- CD : indicates that the port COM is receiving a signal (possible display only during receive mode).

VII.2 - Frame Format

The program transmits frames composed of :
- a preamble
- a data field (text, hexadecimal data, file or sentence test).

The preamble is composed of 4 bytes which are :
- FFh
- AAh
- data field size
- data format

As the RS232 interface is asynchronous, FFh and AAh are transmitted to allow to train the FSK demodulator and to allow a good synchronisation for next character.
The data field size corresponds to the number of bytes transmitted in the frame. A byte corresponds to a character or one hexadecimal data.
The data format gives information about the format of the transmit data. It can be text, hexadecimal data or test (a file is considered as text).
The Figure 37 shows a transmit frame.

VII.3 - Using the Command Function Keys

VII.3.1 - Arrow keys

The left arrow key (←) selects the port COM1 whereas the right key (→) selects the port COM2. When a port COM is selected, its name (C1 or C2) is displayed in upper case letters in the status port COM menu. Then, the transmission and the programming of the RS232 port are possible.

VII.3.2 - F1 Key

Press F1 key if you want to program the selected port COM. The software will ask you to enter the new port COM parameters : see Figure 38
The number of data bits is always 8.
When the programming is finished, the value of the new parameters appears in the status port COM menu within the part corresponding to the selected port COM.
The programmation by default is 1200 bauds, odd parity and 1 stop bit.
To communicate, transmitter and receiver ports COM have to be programmed in the same way.

VII.3.3 - F2 Key (see flow chart 4)

Pressing F2 key allows you to choose between Tx and Rx mode for the selected port COM. Changing from one port COM to another let the unselected port COM in Rx mode.
When you are in Tx mode, you can transmit characters by typing on the keyboard.
Remark : when the Tx mode is selected, the Rx/$\overline{\text{Tx}}$ pin of the ST7537 is not set to a low level (0V) since the powerline modem can't transmit longer than 1s. The program controls the Rx/Tx mode selection pin.

Figure 37 : Frame Format

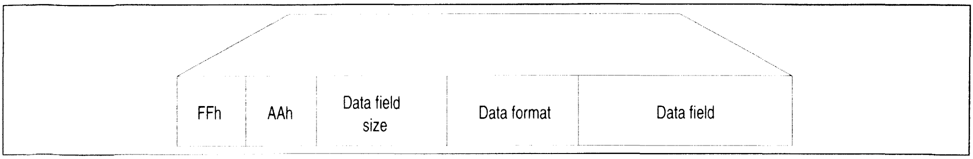
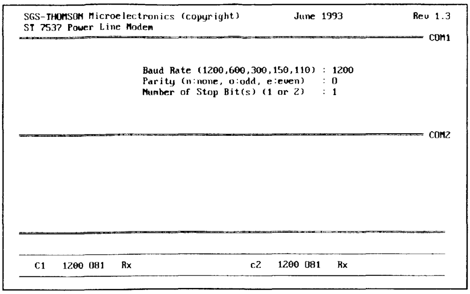


Figure 38



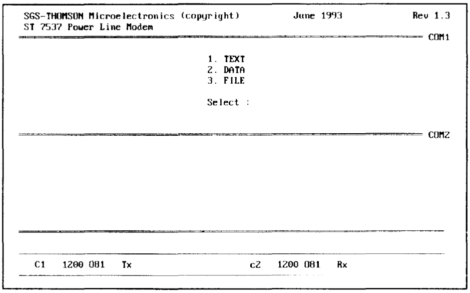
```
SGS-THOMSON Microelectronics (copyright)        June 1993          Rev 1.3
ST 7537 Power Line Modem
_________________________________________________________________ COM1

                Baud Rate (1200,600,300,150,110) : 1200
                Parity (n:none, o:odd, e:even)   : O
                Number of Stop Bit(s) (1 or 2)   : 1

_________________________________________________________________ COM2

C1   1200 081   Rx                   c2   1200 081   Rx
```

7537-44.PCX

VI.3.4 - F3 Key (see flow chart 5)

The F3 key can be used only in Tx mode. Press it to transmit text, hexadecimal data or a file. The following options are displayed :

Figure 39



```
SGS-THOMSON Microelectronics (copyright)        June 1993          Rev 1.3
ST 7537 Power Line Modem
_________________________________________________________________ COM1
                         1. TEXT
                         2. DATA
                         3. FILE

                          Select :

_________________________________________________________________ COM2

C1   1200 081   Tx                   c2   1200 081   Rx
```

7537-45.PCX

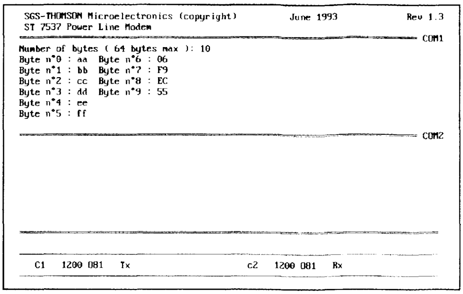
VII.3.4.1 - TRANSMISSION OF TEXT

Press "1" to transmit text. The program will ask you to enter the text (maximum 80 characters). The program calculates the text size and fills the transmit buffer with your text. The Rx/Tx pin of the ST7537 is set to a low level (0V). Then the preamble is transmitted and next the transmit buffer. At the end of the transmission, it is necessary to wait 30ms before returning in Rx mode, in order to save the last transmit byte. Then the Rx/Tx pin of the ST7537 is set to a high level (+5V).

VII.3.4.2 - TRANSMISSION OF HEXADECIMAL DATA

Press "2" to transmit hexadecimal data. The program will ask you how many bytes you want to transmit (maximum 64 bytes). Then you will have to enter hexadecimal data as it is shown in the following screen.

Figure 40



```
SGS-THOMSON Microelectronics (copyright)        June 1993          Rev 1.3
ST 7537 Power Line Modem
_________________________________________________________________ COM1
Number of bytes ( 64 bytes max ): 10
Byte n°0 : aa  Byte n°6 : 06
Byte n°1 : bb  Byte n°7 : F9
Byte n°2 : cc  Byte n°8 : EC
Byte n°3 : dd  Byte n°9 : 55
Byte n°4 : ee
Byte n°5 : ff

_________________________________________________________________ COM2

C1   1200 081   Tx                   c2   1200 081   Rx
```

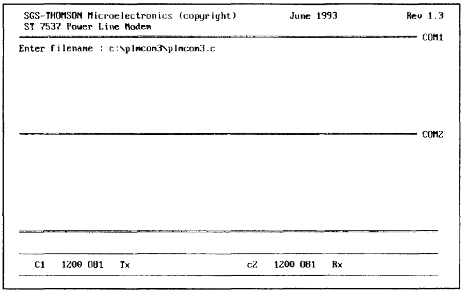
7537-46.PCX

The hexadecimal data are set in a buffer and are transmitted in the same way than the text (see Transmission of text).

VII.3.4.3 - TRANSMISSION OF A FILE

Press "3" to transmit a file. The program will ask you to enter the filename. You have to enter the complete file path. An example is given in the following figure.

Figure 41



```
SGS-THOMSON Microelectronics (copyright)        June 1993          Rev 1.3
ST 7537 Power Line Modem
_________________________________________________________________ COM1
Enter filename : c:\plmcom3\plmcom3.c

_________________________________________________________________ COM2

C1   1200 081   Tx                   c2   1200 081   Rx
```

7537-47.PCX

Then the program opens the file and transmits it line per line. For that, it puts the line in the transmit buffer and transmits it in the same way than the text (see Transmission of text).

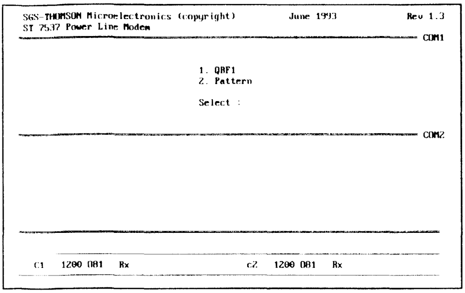
If the file does not exist, an error message appears : "CANNOT OPEN FILE".

VII.3.5 - F4 Key (see flow chart 7 and 9)

You have to be in Tx mode for the use of this key. When you press F4 key, you run a character error rate (CER) test using QBF1 sentence or your specific pattern.

The program will ask you what kind of sentence you want to transmit : See Figure 42

Figure 42



```
SGS-THOMSON Microelectronics (copyright)        June 1993        Rev 1.3
ST 7537 Power Line Modem
---------------------------------------------------------------- COM1

                        1. QBF1
                        2. Pattern

                        Select :

---------------------------------------------------------------- COM2



------------------------------------------------------------------
  C1   1200 081   Rx                  c2   1200 081   Rx
```

7537-48.PCX

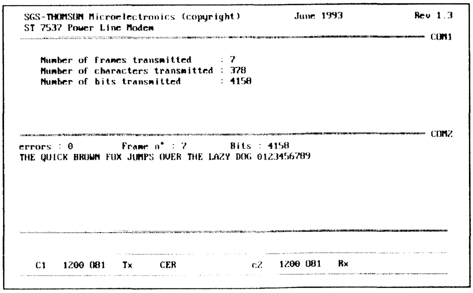
By pressing "1", you start a CER test using the sentence : "THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG 0123456789". By pressing "2", the software will ask you to enter the specific pattern. QBF1 test is set by default.

In this configuration you have to press F4 key on receiver in order to enter the right pattern to check. The program will ask you to choose the right sentence (QBF1 or pattern).

Once the sentence is defined, you must enter the number of frames you want to transmit.

Then, the program calculates the sentence size and fills the transmit buffer. At each frame transmission, the Rx/Tx pin of the ST7537 is set to a low level (0V). Then the preamble, the total number of frames and the frame order are transmitted. Next the transmit buffer is sent. At the end of each frame transmission it is necessary to wait 30ms before returning in Rx mode. Then the Rx/Tx pin of the ST7537 is set to a high level (+5V) and the screen displays (see Figure 43)

Figure 43



```
SGS-THOMSON Microelectronics (copyright)        June 1993        Rev 1.3
ST 7537 Power Line Modem
---------------------------------------------------------------- COM1

   Number of frames transmitted     : 7
   Number of characters transmitted : 378
   Number of bits transmitted       : 4158


---------------------------------------------------------------- COM2
errors : 0          Frame n° : 7         Bits : 4158
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG 0123456789



------------------------------------------------------------------
  C1   1200 081   Tx     CER           c2   1200 081   Rx
```

7537-49.PCX

The number of frames, characters and bits transmitted are updated at each frame transmission. The number of characters depends on the transmitted sentence size. The number of bits depends on the sentence size and the programming of the port COM (for 1200 bauds, 1 parity bit, 8 data bits and 1 stop bit, there are 11 bits transmitted per character).

A description of the transmit test frame is given in Figure 44.

Remark : For 600bauds or slower, the QBF1 sentence is too long to be transmitted in less than 1s.

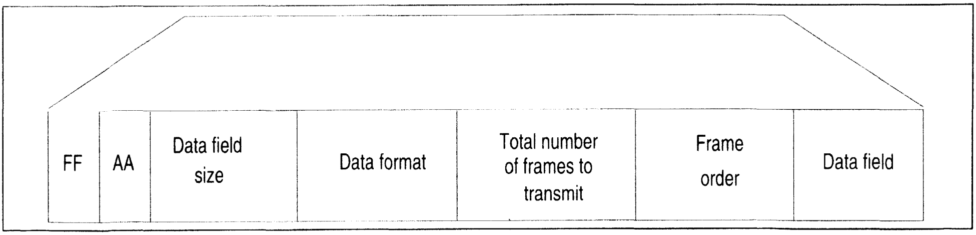
VII.3.6 - F5, F6, F7 Keys

Press F5 key to clear COM1 and COM2 window.
Press F6 key to clear COM1 window.
Press F7 key to clear COM2 window.

VII.3.7 - F8 Key

Press F8 to select 1 or 2 port(s) COM for communication.

Figure 44 : Test Frame Format



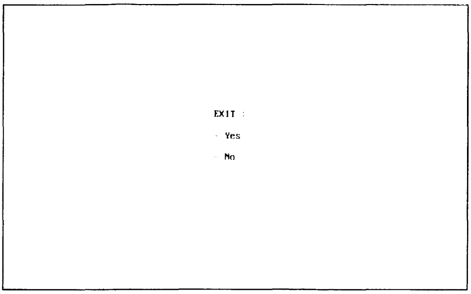
FF	AA	Data field size	Data format	Total number of frames to transmit	Frame order	Data field

7537-50.AI

VII.3.8 - F10 Key

Press F10 key to exit PLMCOM3. The program will ask you to confirm your choice.

Figure 45



```
EXIT :

 - Yes

 - No
```

7537-51.PCX

Hit "y" to exit or hit "n" to return to the communication program.

VII.3.9 - HOME key

The HOME key recalls the different command function keys which are used in the program.

VII.4 - Reception

To receive data, you enter the Rx mode (use F2 key). The reception works under interruptions. When the carrier detect of the ST7537 is held low, an interruption is run which puts the received data in a receive buffer (see flow chart 10). Once the ST7537 carrier detect is set high, the program displays the data according to their format (see flow chart 2).

First of all, the program tests if AAh is received for synchronisation. If so, the data field size and the data format are read.

In case of text, hexadecimal data or file format, the corresponding data are displayed in the receiver COM window. As far as the file is concerned, it is received and displayed line per line.

In case of CER test, on reception of each frame the program reads :

- the total number of frames transmitted
- the frame order

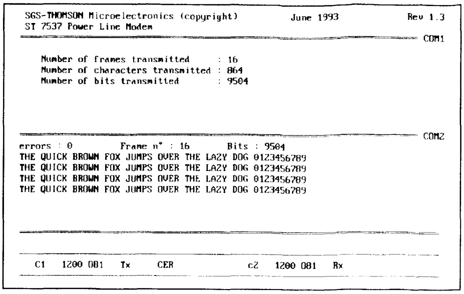
Then it displays the received sentence, the number of character errors, the number of received frames and the number of received bits (see Figure 46).

Once the last frame is received, the program calculates the CER and displays the following screen : see Figure 47.

The frame errors correspond to the frames which contain at least one error and to those which had not been received.

The character errors correspond to the number of received characters which are wrong.

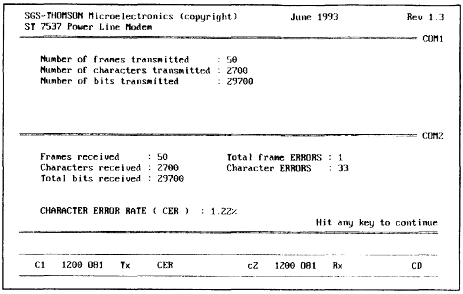
Figure 46



```
SGS-THOMSON Microelectronics (copyright)        June 1993              Rev 1.3
ST 7537 Power Line Modem
                                                                          COM1
    Number of frames transmitted     : 16
    Number of characters transmitted : 864
    Number of bits transmitted       : 9504

                                                                          COM2
errors : 0          Frame n° : 16        Bits : 9504
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG 0123456789
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG 0123456789
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG 0123456789
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG 0123456789

   C1   1200 081   Tx     CER              c2   1200 081   Rx
```

7537-52.PCX

Figure 47



```
SGS-THOMSON Microelectronics (copyright)        June 1993              Rev 1.3
ST 7537 Power Line Modem
                                                                          COM1
    Number of frames transmitted     : 50
    Number of characters transmitted : 2700
    Number of bits transmitted       : 29700

                                                                          COM2
    Frames received      : 50        Total frame ERRORS : 1
    Characters received  : 2700      Character ERRORS   : 33
    Total bits received  : 29700

    CHARACTER ERROR RATE ( CER )  : 1.22%
                                                   Hit any key to continue

   C1   1200 081   Tx     CER              c2   1200 081   Rx           CD
```

7537-53.PCX

VIII - TYPICAL APPLICATION : LOAD MANAGEMENT

VIII.1 - Protocol Design

The software described in the following parts provides you with a simple efficient protocol kernel, which is fully interrupt handled and uses almost no CPU time. Therefore it enables you to develop friendly interactive applications with a short response time.

This protocol uses a packet encapsulation mechanism with two level error detection capability, both for the packet level and for the byte level. During reception, burst noise can affect the communication channel, so that a frame check sum is used to detect excessive errors. In many cases, impulsive noise may cause unpredictable data loss without modifying the frame check sum. Therefore, each byte is transmitted and received in an asynchronous mode inside a 11-bit type word including a start bit, one stop bit, and an odd parity bit to ensure byte integrity.

VIII.1.1 - Frame Format (see Figure 48)

Each frame consists of a preamble, a header, a house address, a link control, a source address, a destination address, a data block, and a frame check sum.

The preambule is 8-bit field with a fixed value FFh:

it trains the FSK demodulator, allows a good uart synchronisation for next character. The header consists of a 8-bit pattern AAh chosen with a low probability of wrongly detecting noise or preamble as the header. On a message reception, a matching test is run on the house address field to overcome perturbations coming from a neighbouring home network.

VII.2 - Use of the ST90E28 resources

- The Watchdog/Timer :
  The watchdog mode is disabled and the timer is operated in continuous mode.
  On each timer interrupt request, network access parameters, keyboard delay time, common system clock parameters are updated. Besides, the ST7537 watchdog input is reset.
- The Serial Communication Interface (SCI) :
  The SCI is configured in asynchronous mode to exchange data between the power line modem and the microcontroller. Every character sent (or received) by the SCI has the following format: 1 start bit, 8 data bits, 1 parity bit (odd parity selected), 1 stop bit. The transmit rate is 1200 bauds.
  To start transmitting a frame, the transmitter buffer register is loaded with the preambule value FFh in order to run the SCI. Each data byte end of transmission results in the generation of an TXHEM (transmitter buffer empty) interrupt request to load the next transmit data byte.
  An outstanding character search is performed to detect the header of an incoming frame (see Figure 49). This is achieved by comparing each received data byte to the content of the data compare register. If the incoming character matches, an RXA (receiver address match) interrupt is requested to enable the analysis of the next data frame fields. Every time the reception of a data byte is completed, a RxD (receive data) interrupt request is generated to store the received data byte.
- The Register File (see Figure 50) :
  Among the 224 available global purpose registers, 16 registers are reserved as a transmit frame buffer, another group of 16 registers is reserved as a receive frame buffer, 48 registers are dedicated to the protocol kernel, and another group of 48 registers is allocated to the system & user stacks, which leaves 96 registers for storage of applicative values.
- The Input/Output Ports :
  Two of the port pins must be used for the Rx/Tx (P5.0) and WD (P5.1) output signals. Four must be initialized as alternate function for the RSTO (P2.0), CD (P2.4), RxD (P3.6) and TxD (P3.7) signals. Details concerning the initialization of these ports are given in next section.

Figure 48 : Frame Fields

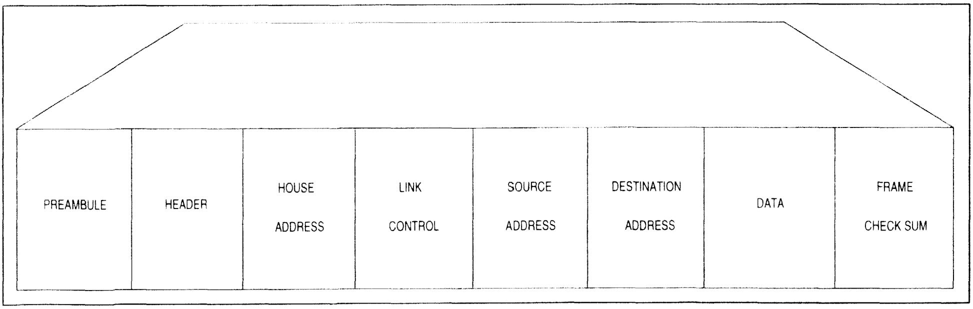


Figure 49 : Character Search Function

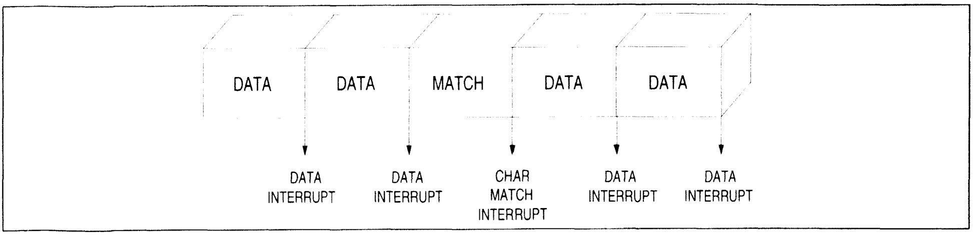
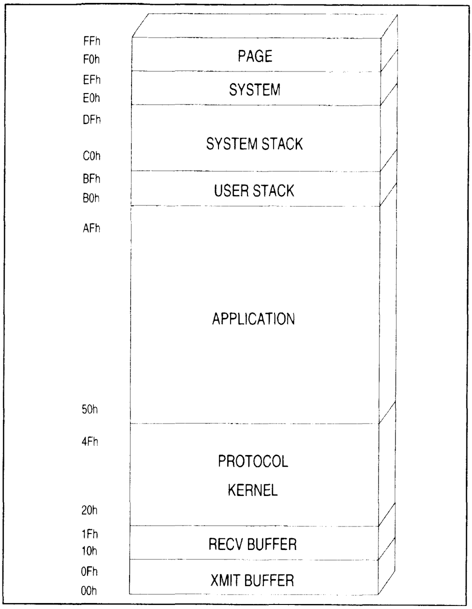


Figure 50 : Register File Map



VII.2.1 - Initialization of ST90E28 core and on-chip peripherals

- Core initialization : The user and system stacks are set up in the internal register file. The internal clock frequency is set to 11.0592MHz. The priority level of the main program is set to 7 (lowest), whereas the non-maskable interrupt (RSTO signal) has the top level priority.
- Initialization of the Input/Output ports : Only six input/outputs are required to exchange data between the ST7537 and the ST90E28. The corresponding pins are initialized as follows :

NMI	(Port 2 bit 0)	→	Alternate function, open drain, TTL
CD	(Port 2 bit 4)	→	AF, OP, TTL
RxD	(Port 3 bit 6)	→	AF, OP, TTL
TxD	(Port 3 bit 7)	→	Alternate function, Push pull, TTL
Rx/Tx	(Port 5 bit 0)	→	Output, Push pull, TTL
WD	(Port 5 bit 1)	→	OUT, PP, TTL

The NMI pin is programmed rising edge sensitive, whereas the CD/ input signal triggers an external interrupt request on a falling edge (INT1 pin) with a priority level set to 1.

As for the applicative features, each port pin is initialized as follows :

display pin	→	Output, push pull, TTL
keyboard pin	→	Input, tristate, TTL

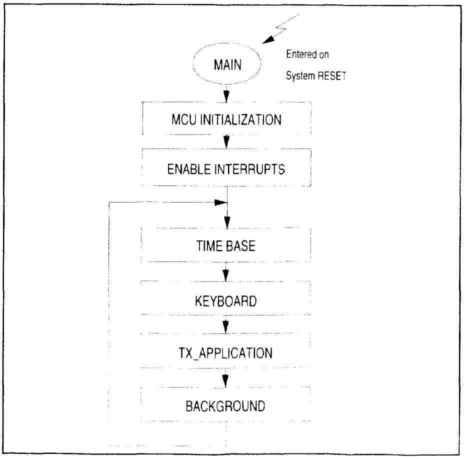
- Timer : The watchdog mode is disabled. Continuous mode is selected with count down from a fixed value of 767, each underflow resulting in an interrupt request and reload of the fixed initial counter value. The internal clock rate, prescaler and initial count value are chosen to give an interrupt request every 555.56µs (1.8kHz = 36*50Hz = 30*60Hz). The timer counter is loaded with the value 767 to complete an end of count every 555.56µs. On each counter underflow an interrupt request (INT0) is generated with a priority level set to 0 (high).
- Serial Communication Interface : The asynchronous mode is selected. The serial interface programmed characteristics are : 8-bit word length, odd parity generation and detection, 1 stop bit generation, AAh header search. In this mode, each data bit is sampled 16 times, so that each data bit period will be 16 SCI clock periods long. The counter of the baud rate generator is loaded with the fixed value 576 to set the SCI clock rate to 16*1200 = 19200 bauds. The priority level of all SCI interrupts (RXA, RxD, TXHEM) is set to 1.

VIII.2.2 - Main Program

The main is automatically entered on system reset, and first initializes the internal clock, stacks, ports, register file, serial communication interface, and timer. Then the timer starts counting down towards zero from an initial value of 767. Each time the counter clears to zero, an high priority interrupt request will be generated, which will initiate an update of the network access parameters.

The main program loops around the main modules.

Figure 51 : Main Program Flow Chart



ANNEXE A : DEMOBOARD OUTPUT IMPEDANCE

Figure 52

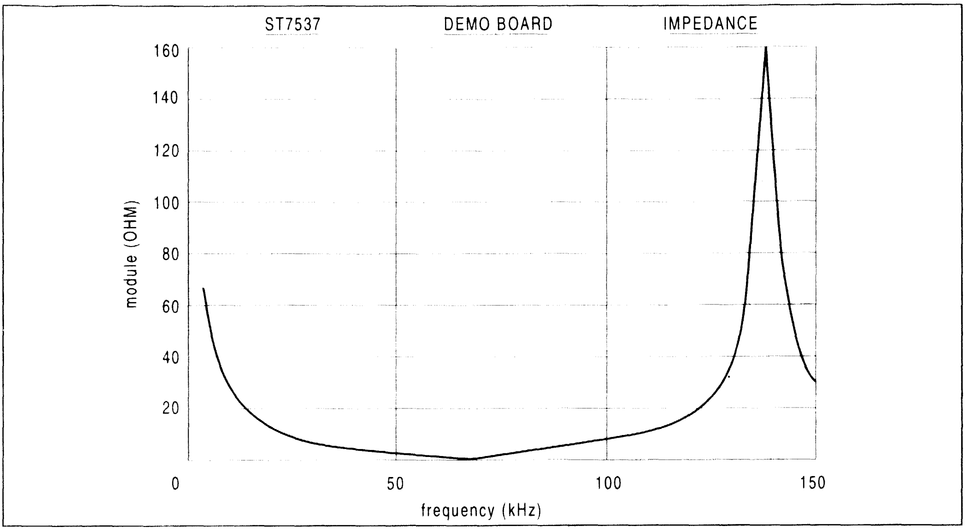
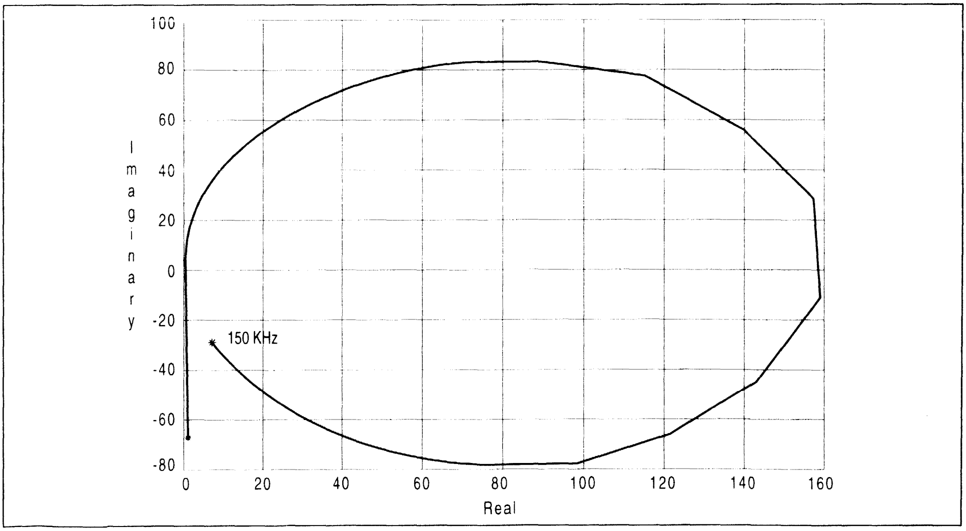
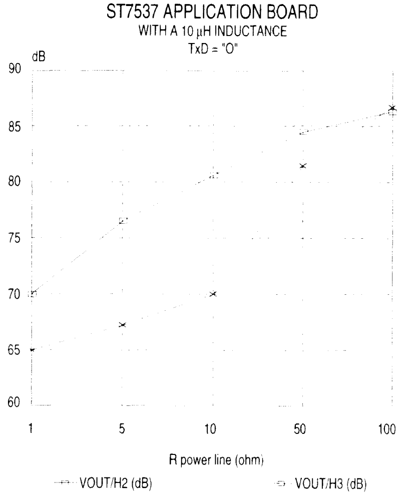


Figure 53



ANNEXE B : DEMO-BOARD TRANSMIT PERFORMANCE VOUT/H2 (dB) and VOUT/H3 (dB) with the 10µH inductance

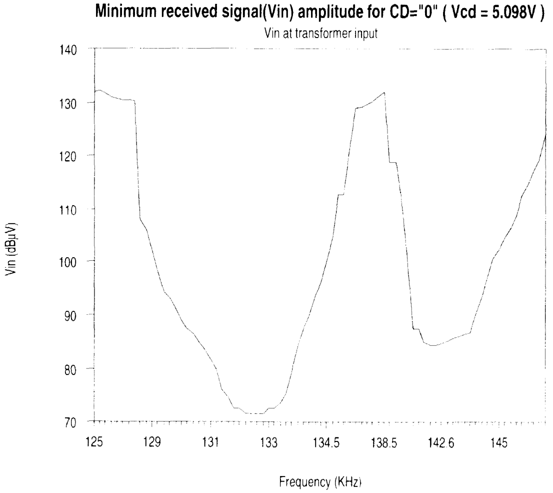
Figure 54



7537-60 AI

ANNEXE C : RAI INPUT MINIMUM DETECTION LEVEL

Figure 55



7537-61 AI

ANNEXE D : PC COMMUNICATION PROGRAM FLOW CHARTS

Figure 56 : Flow chart 1 : PLMCOM3

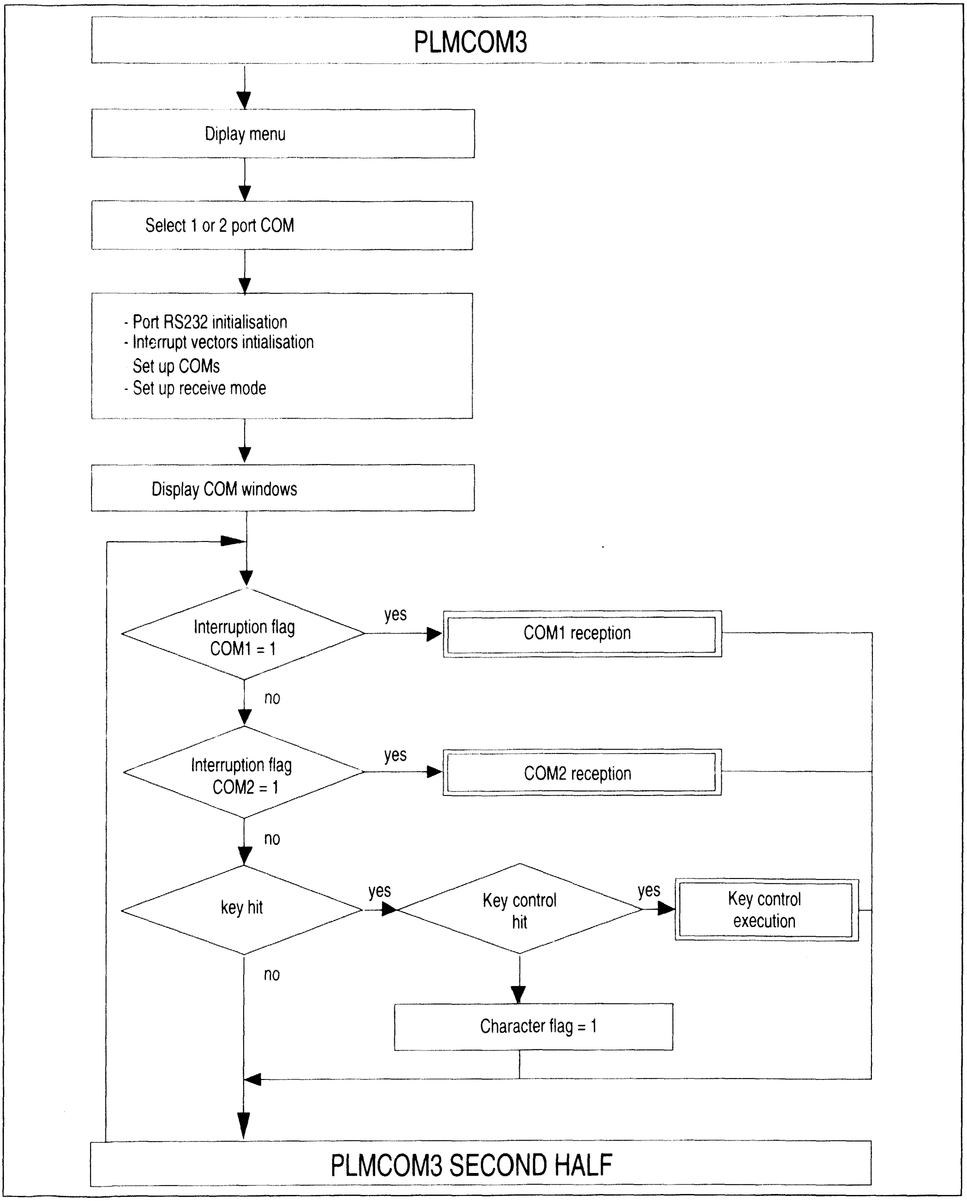


Figure 57 : Flow chart 1 : PLMCOM3 second half

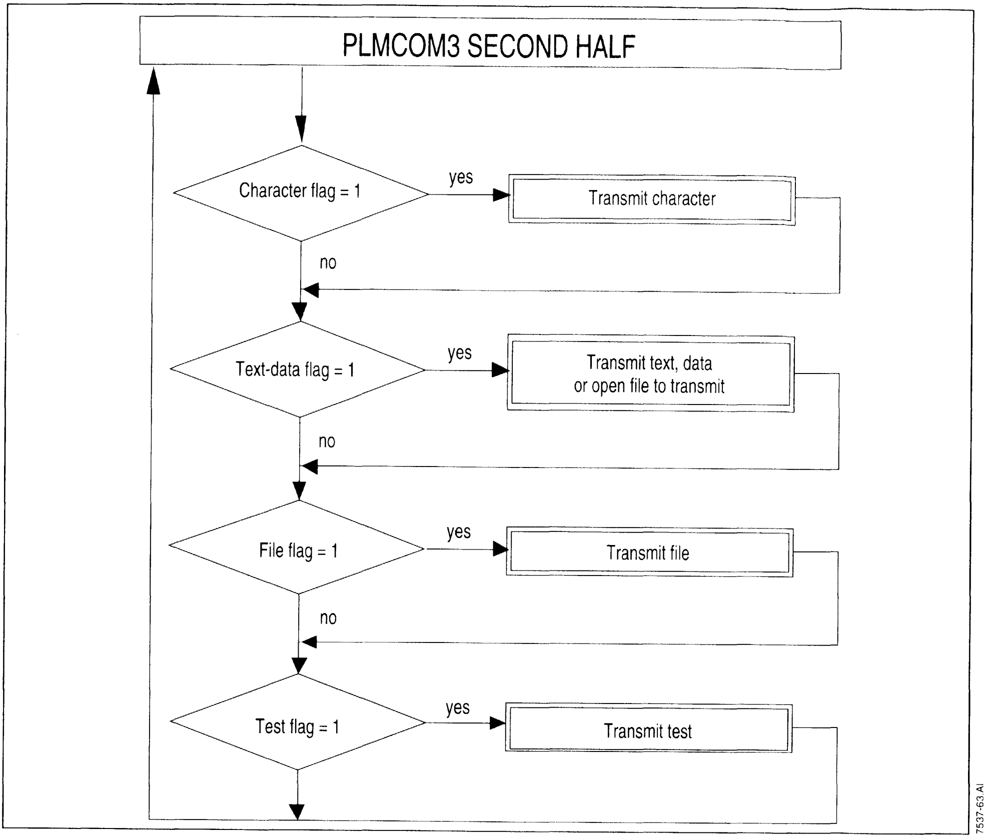
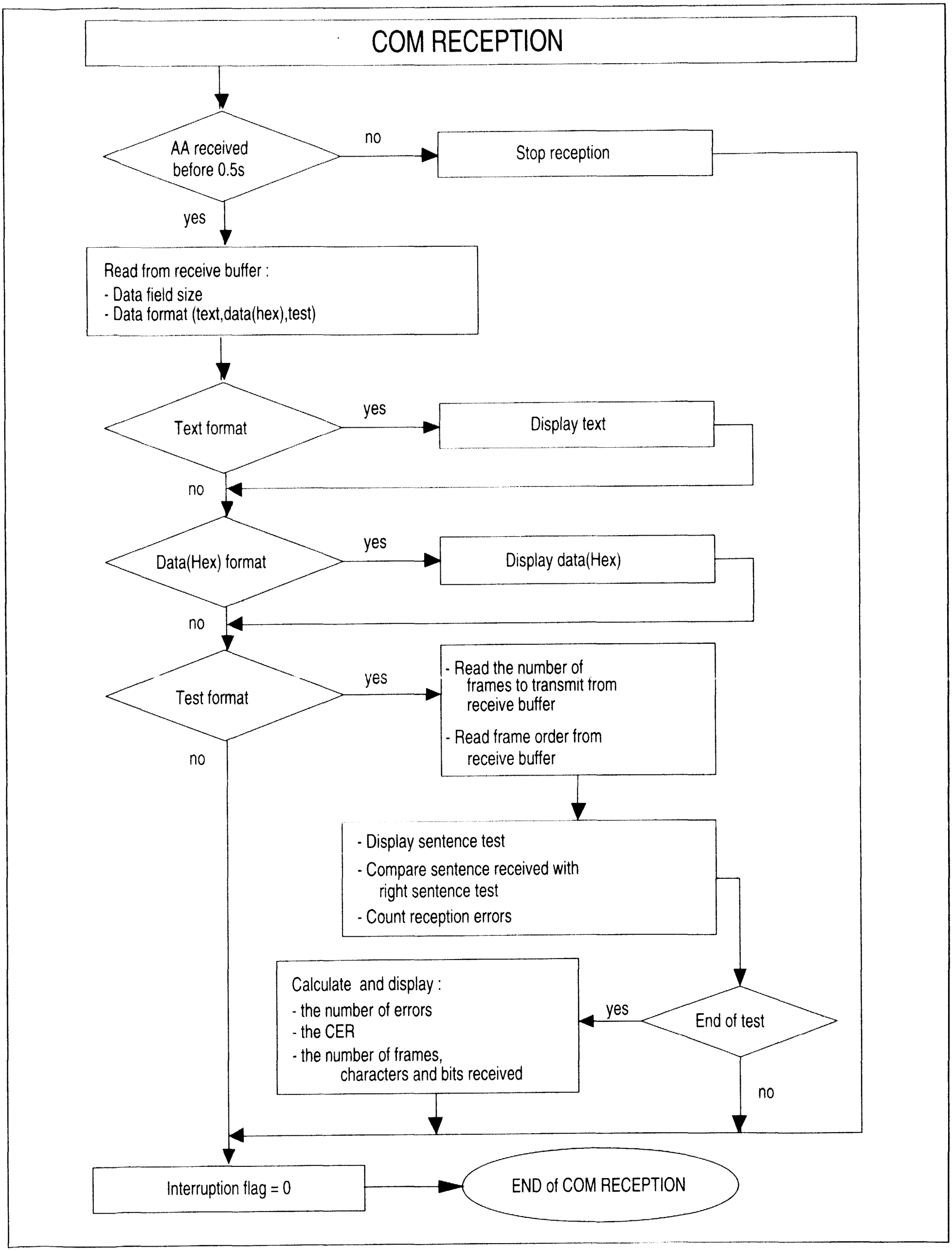


Figure 58 : Flow chart 2 : Com reception



7537-64.AI

Figure 59 : Flow chart 3 : Key control execution

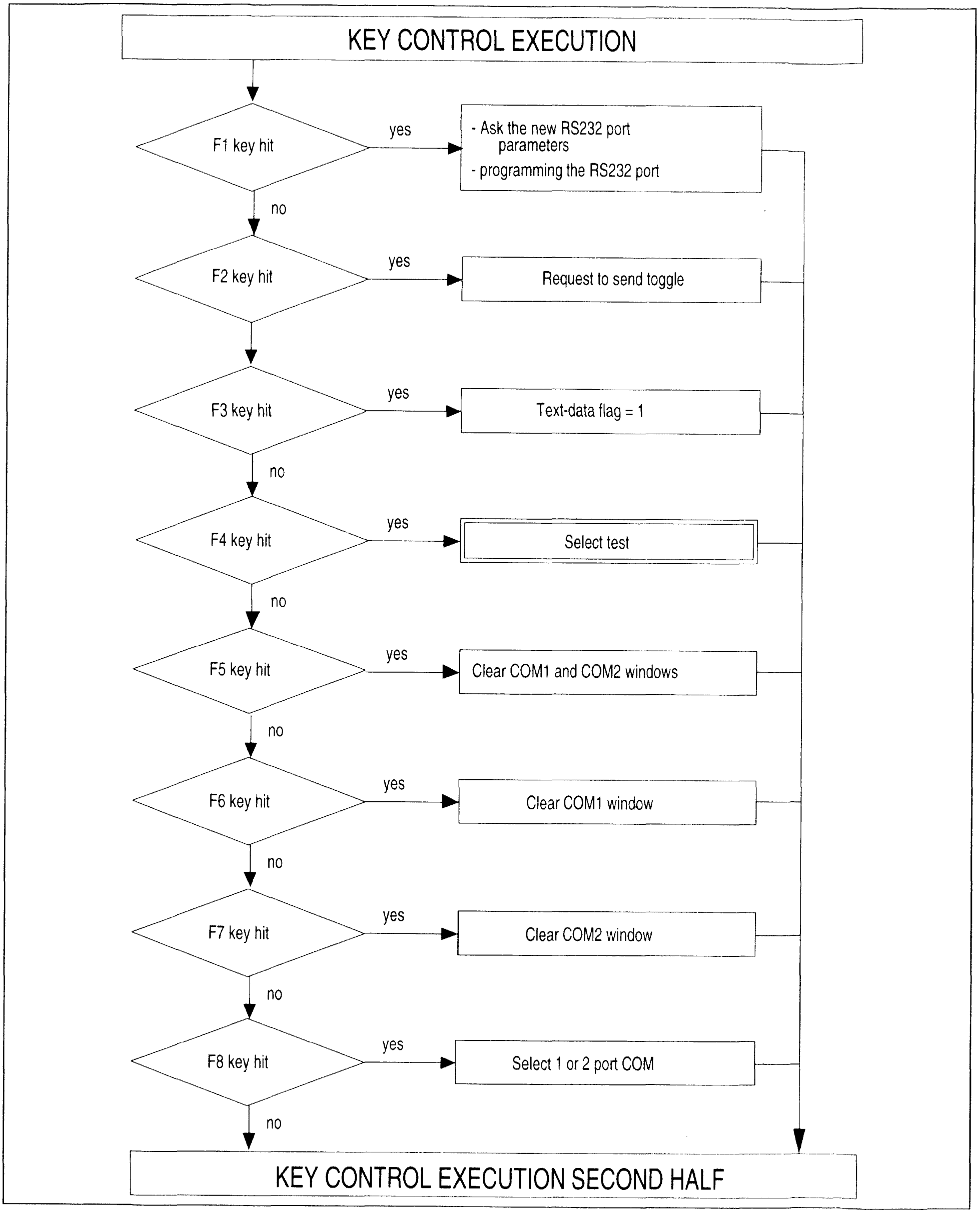


Figure 60 : Flow chart 3 : Key control execution second half

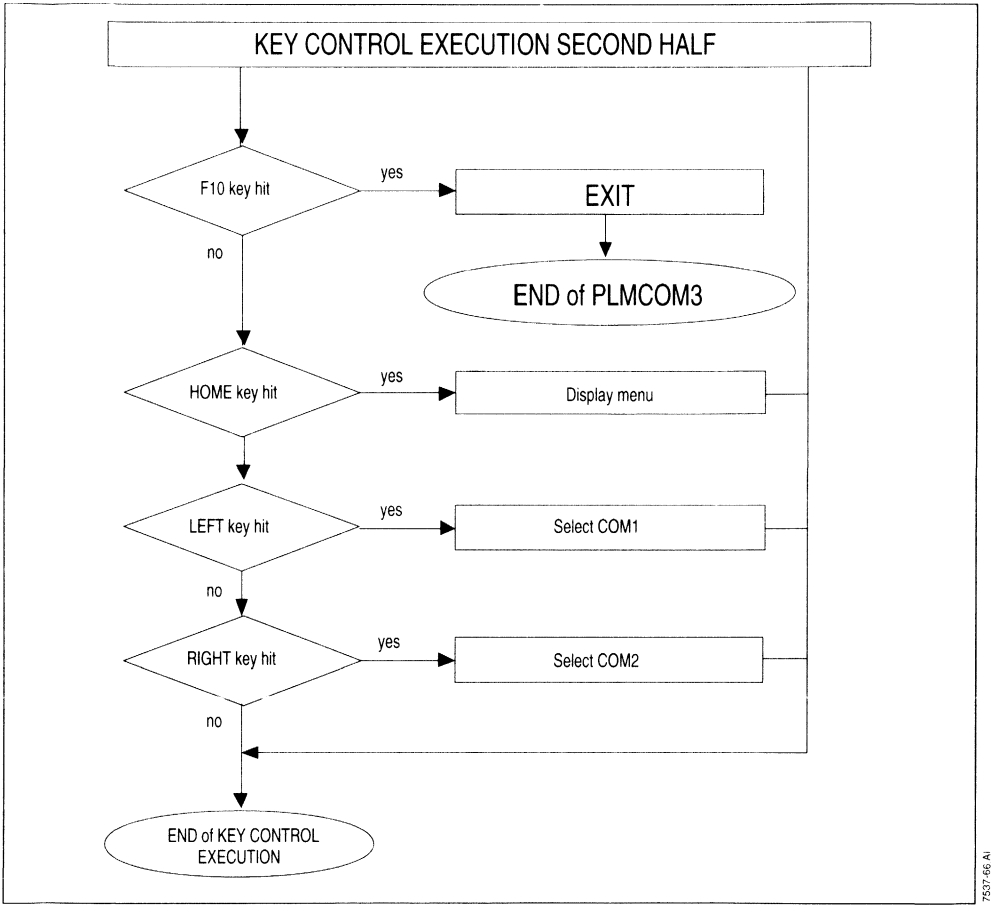
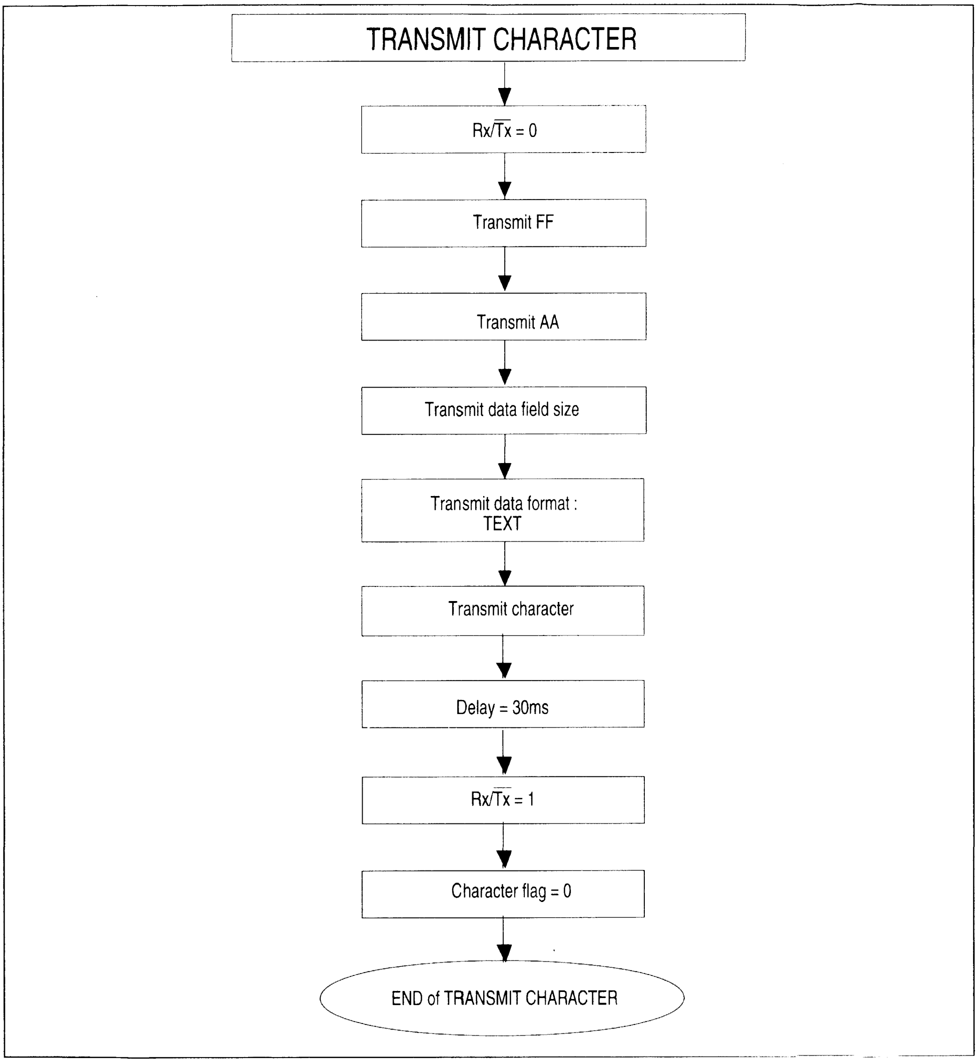


Figure 61 : Flow chart 4 : Transmit character



7537-67 AI

Figure 62 : Flow chart 5 : Transmit text or data

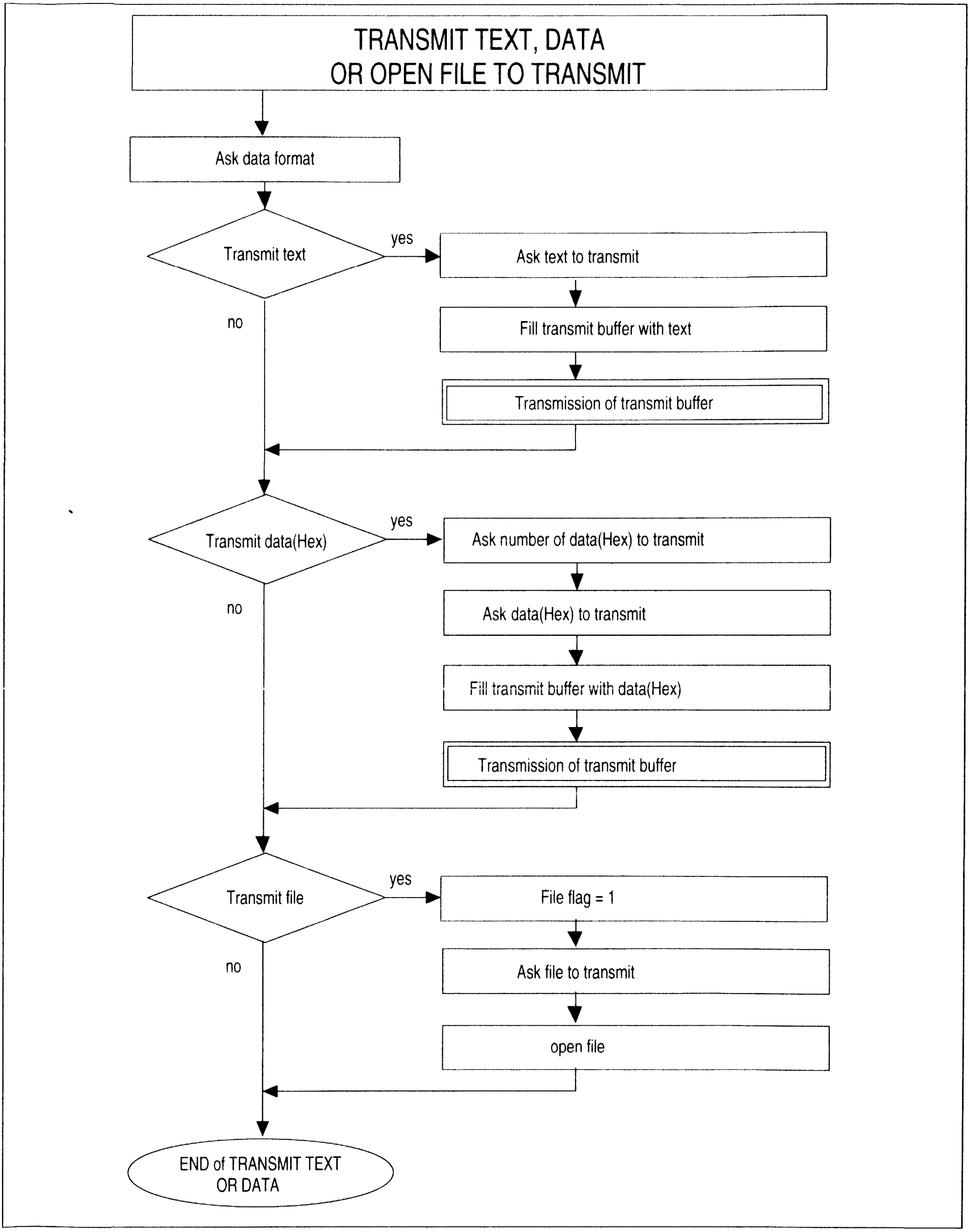


Figure 63 : Flow chart 6 : Transmit file

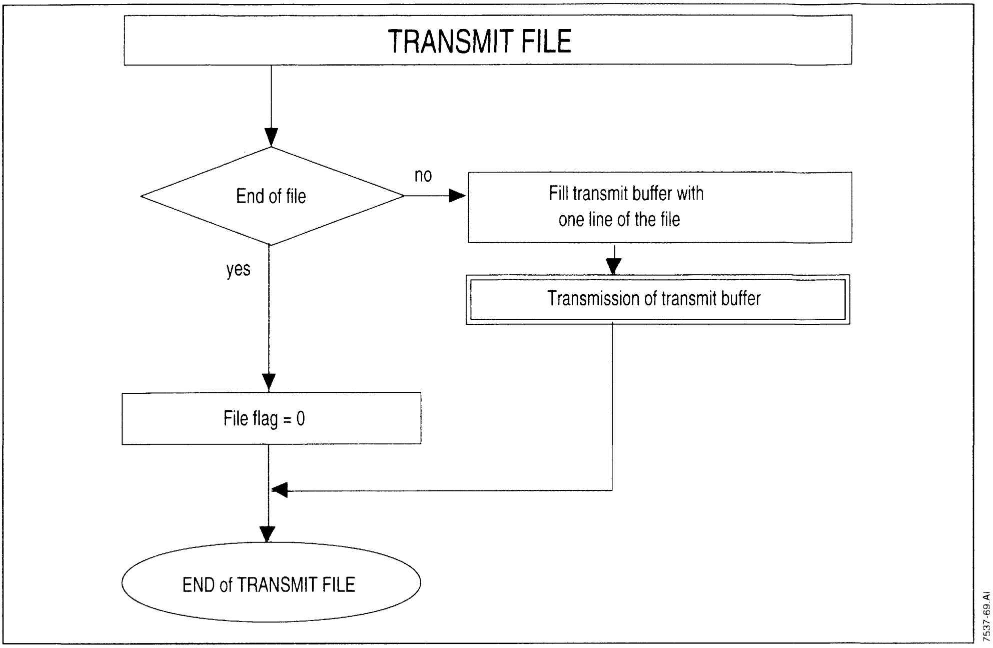

TRANSMIT FILE
End of file
no
Fill transmit buffer with one line of the file
Transmission of transmit buffer
yes
File flag = 0
END of TRANSMIT FILE
7537-69.AI

Figure 64 : Flow chart 7 : Transmit test

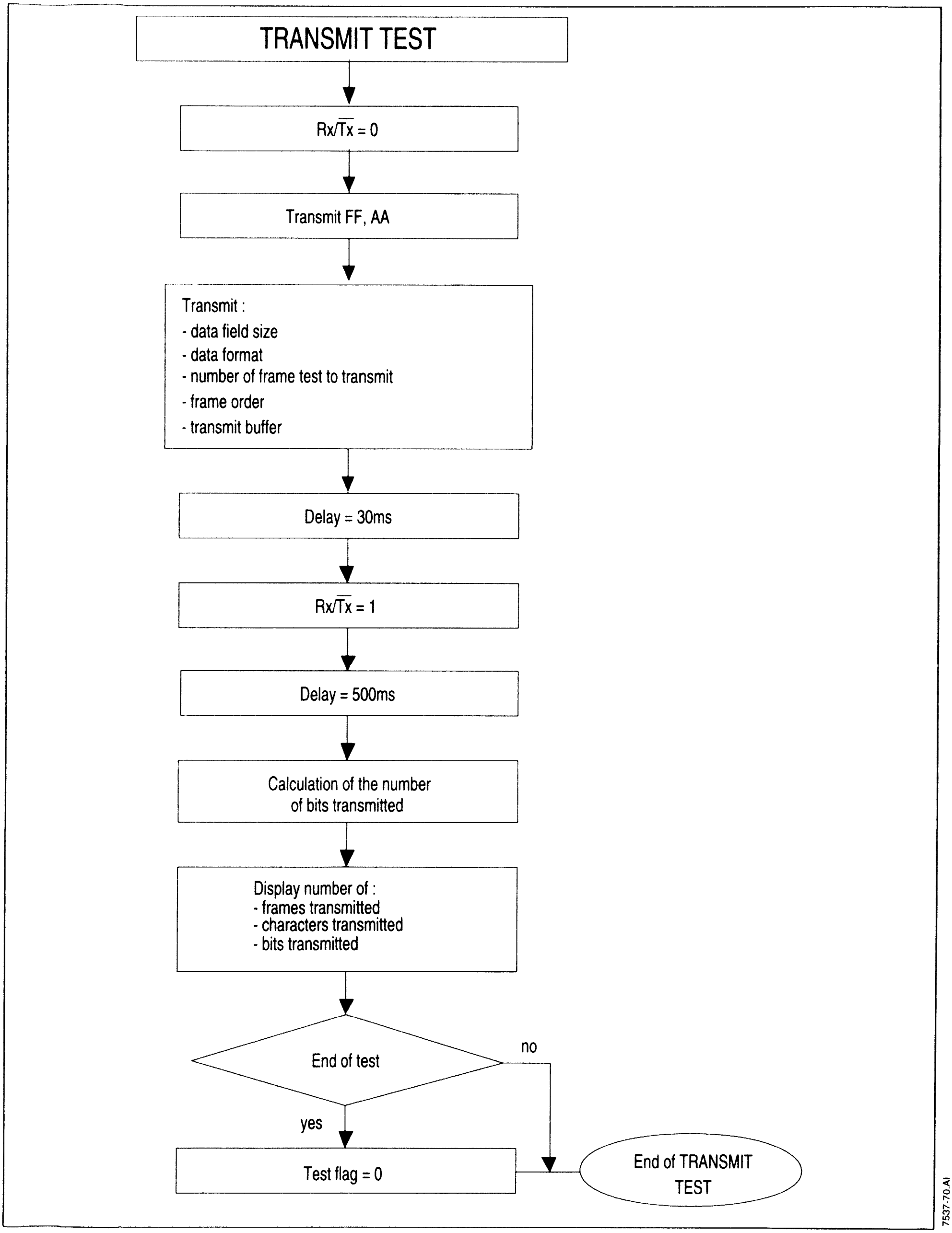


Figure 65 : Flow chart 8 : Transmission of tranmit buffer

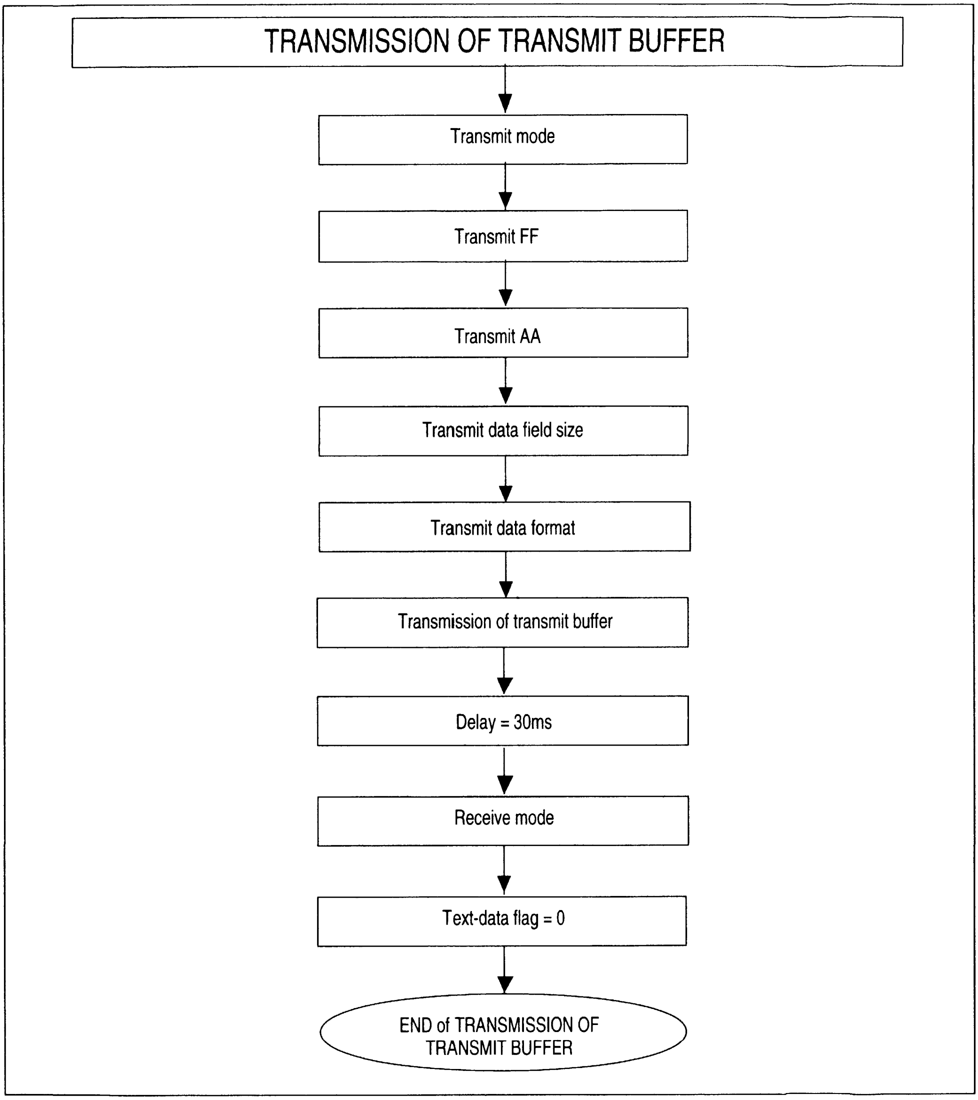
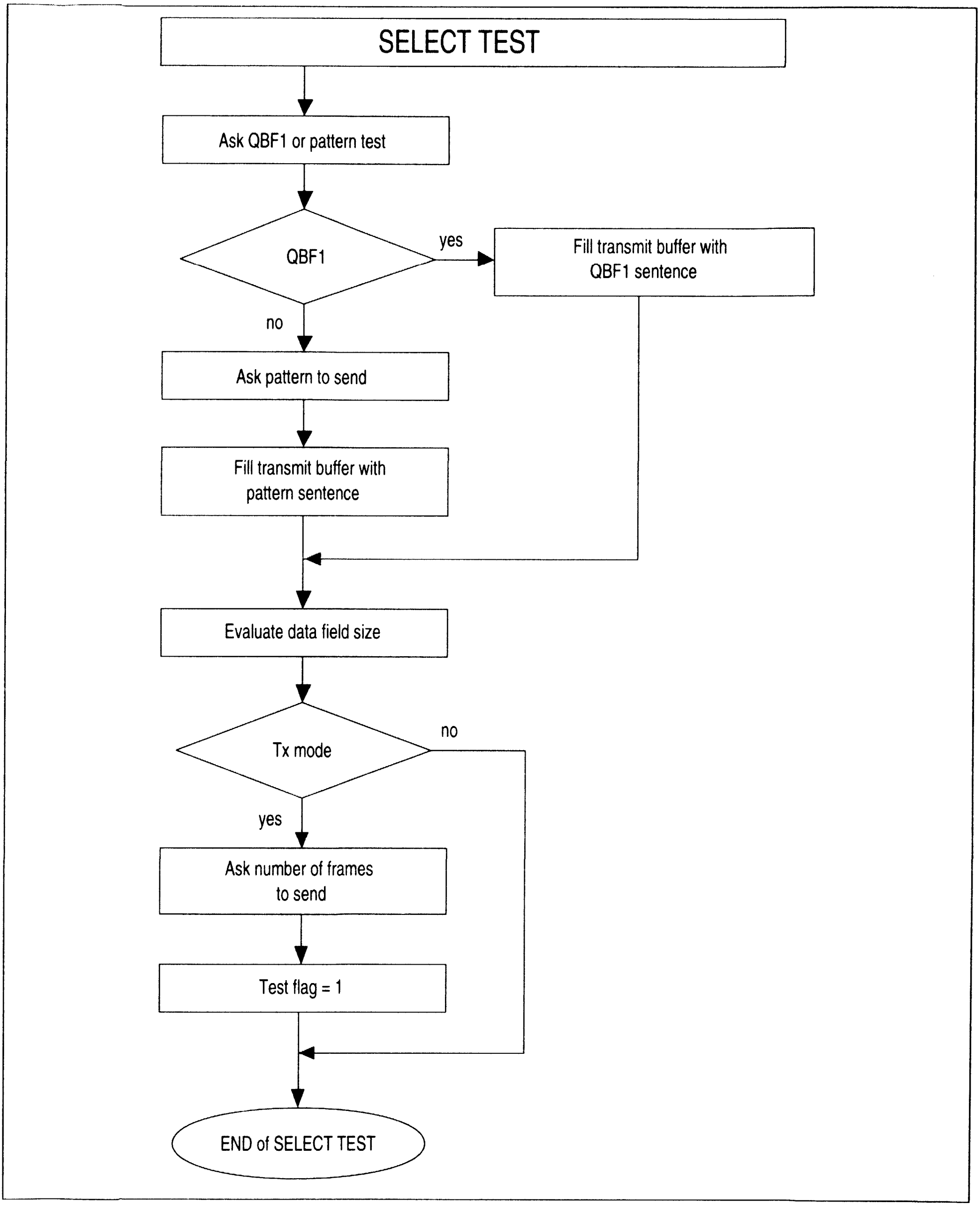
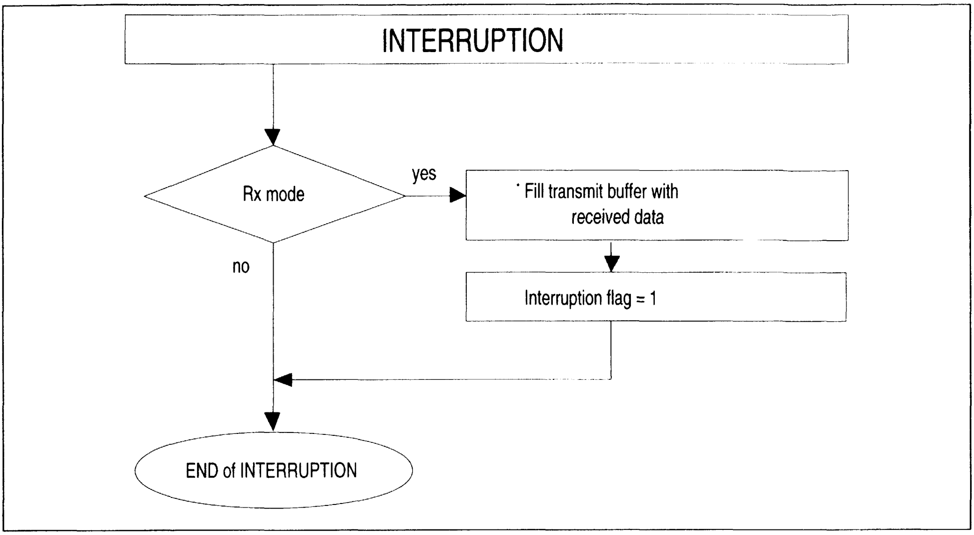


Figure 66 : Flow chart 9 : Select test



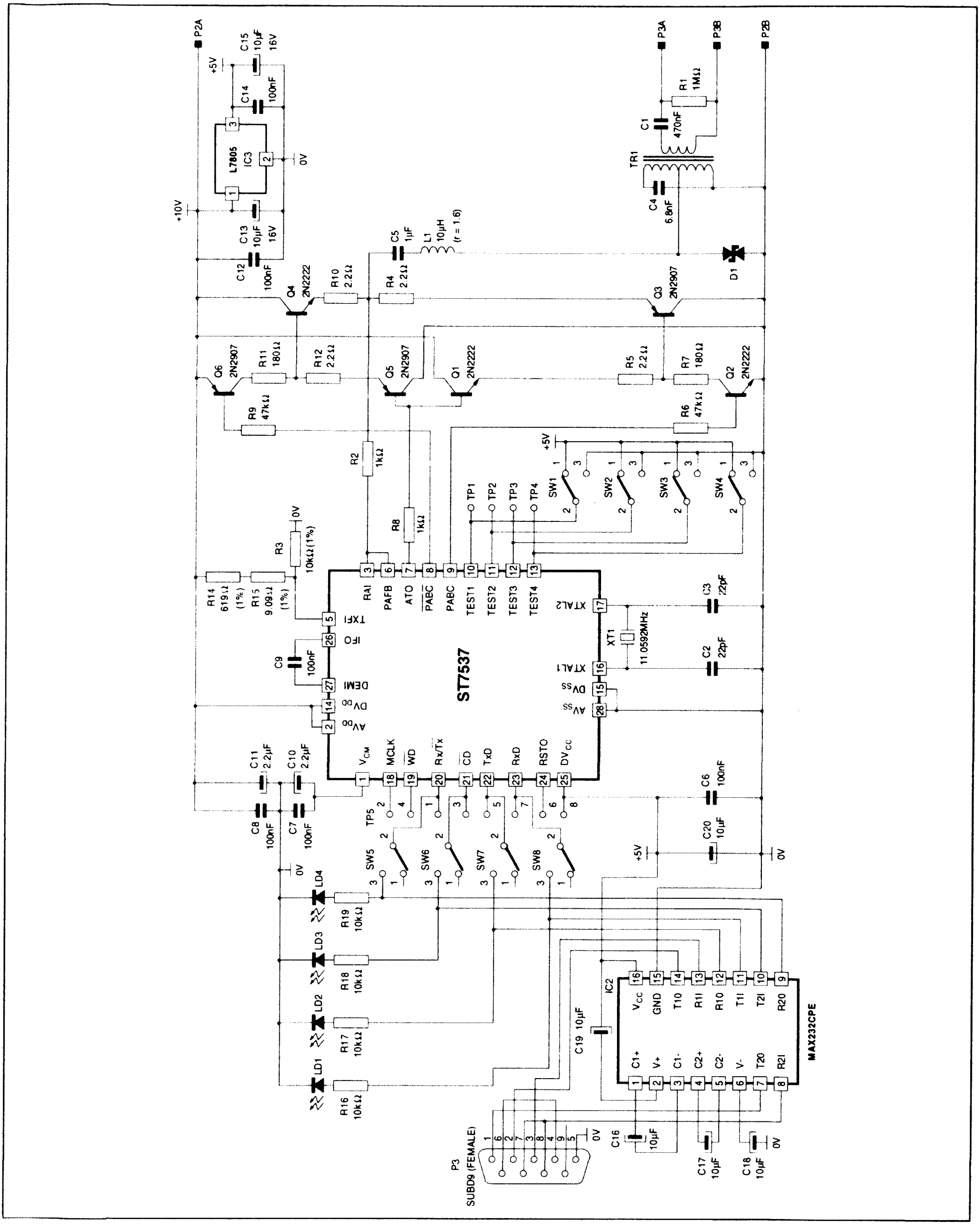
7537-72.AI

Figure 67 : Flow chart 10 : Interruption



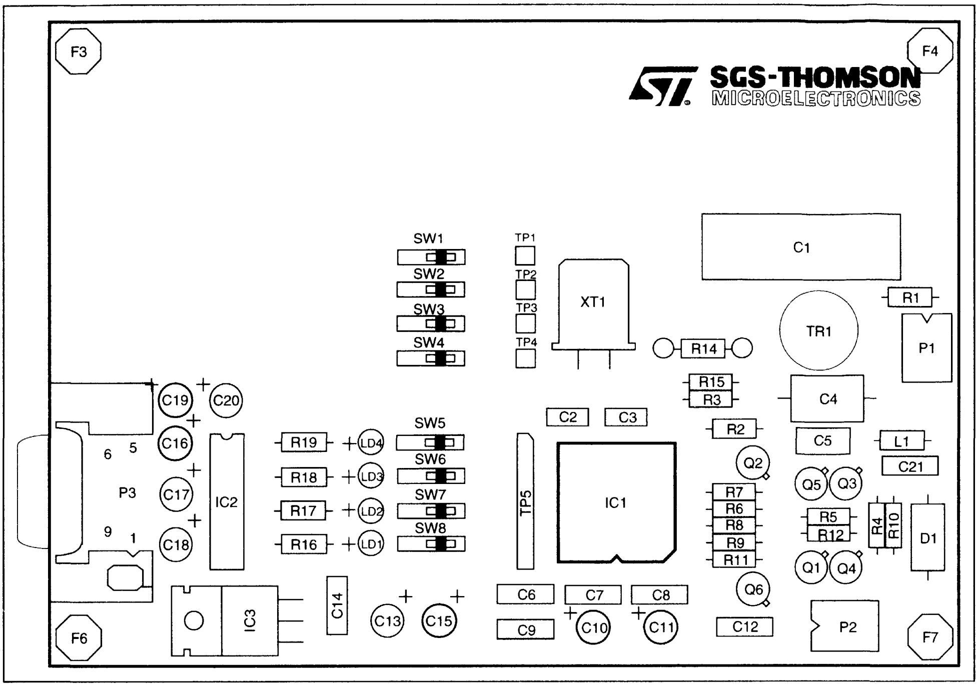
ANNEXE E : DEMOBOARD SCHEMATICS & LAY OUT

Figure 68 : Application Board



7537-74.EPS

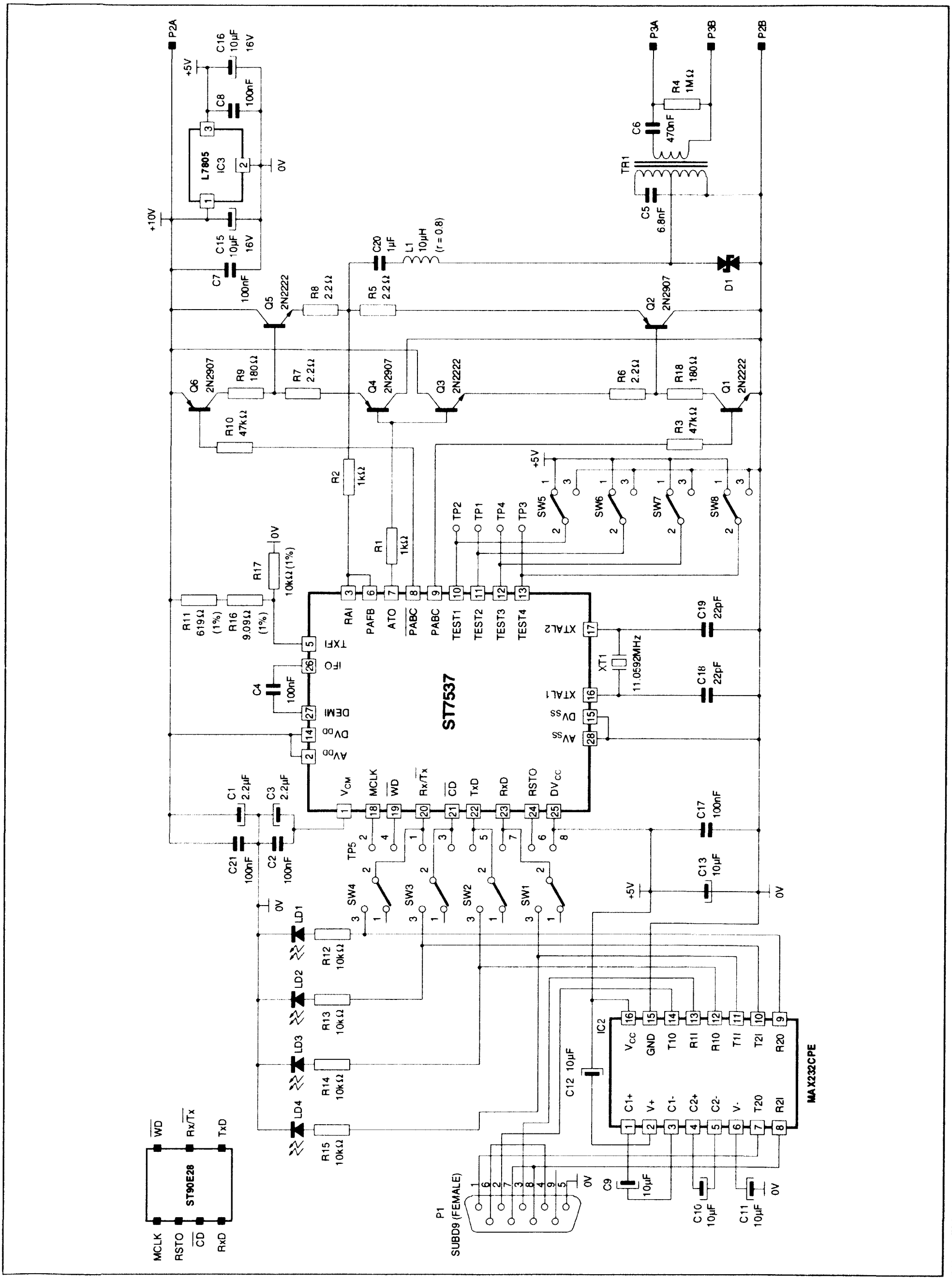
Figure 69 : Layout



Bill Of Materials

Item	Qty.	Reference	Part
1	2	C11,C10	2.2µF
2	6	C7,C6,C8,C9,C12,C14	100nF
3	4	LD4,LD1,LD2,LD3	LED
4	1	IC1	ST7537
5	8	SW8, SW1, SW2, SW3, SW4, SW5, SW6, SW7	
6	1	XT1	CRYSTAL
7	2	R8, R2	1kΩ
8	2	R6, R9	47kΩ
9	3	Q2, Q1, Q4	2N2222
10	3	Q3, Q5, Q6	2N2907
11	1	C4	6.8nF
12	1	C1	470nF
13	1	R1	1MΩ
14	4	R4, R5, R10, R12	2.2Ω
15	2	R11, R7	180Ω
16	1	IC2	MAX232CPE
17	1	IC3	LM7805
18	1	R14	619 (1%)
19	4	R19, R16, R17, R18	10kΩ
20	5	C16, C17, C18, C19, C20	10µF
21	1	C21	15nF
22	2	PICO1, PICO2	PICO
23	2	C13, C15	10nF/16V
24	1	L1	10µH (r=0.8)
25	1	D1	DIODE
26	5	TP2, TP1, TP3, TP4, TP5	POINT
27	1	P3	SUBD9 (FEMALE)
28	1	P2	ALIM
29	1	P1	ALIM+
30	1	TR1	TOKO
31	1	R15	9.09kΩ (1%)
32	1	R3	10kΩ (1%)
33	2	C2, C3	22pF
34	1	C5	1µF

Figure 70 : Application Board



7537-76.EPS

Figure 71 : Master Configuration Board

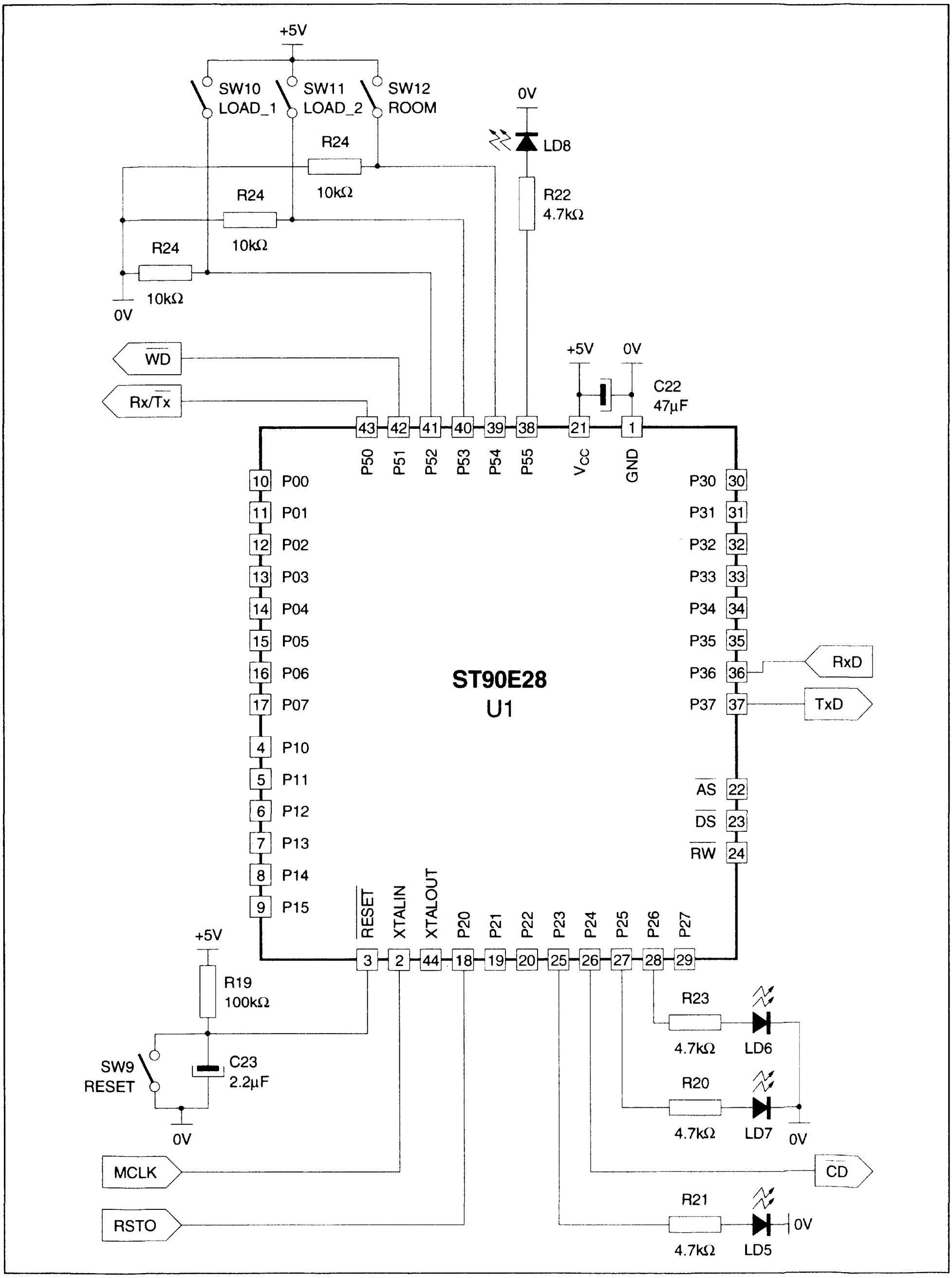
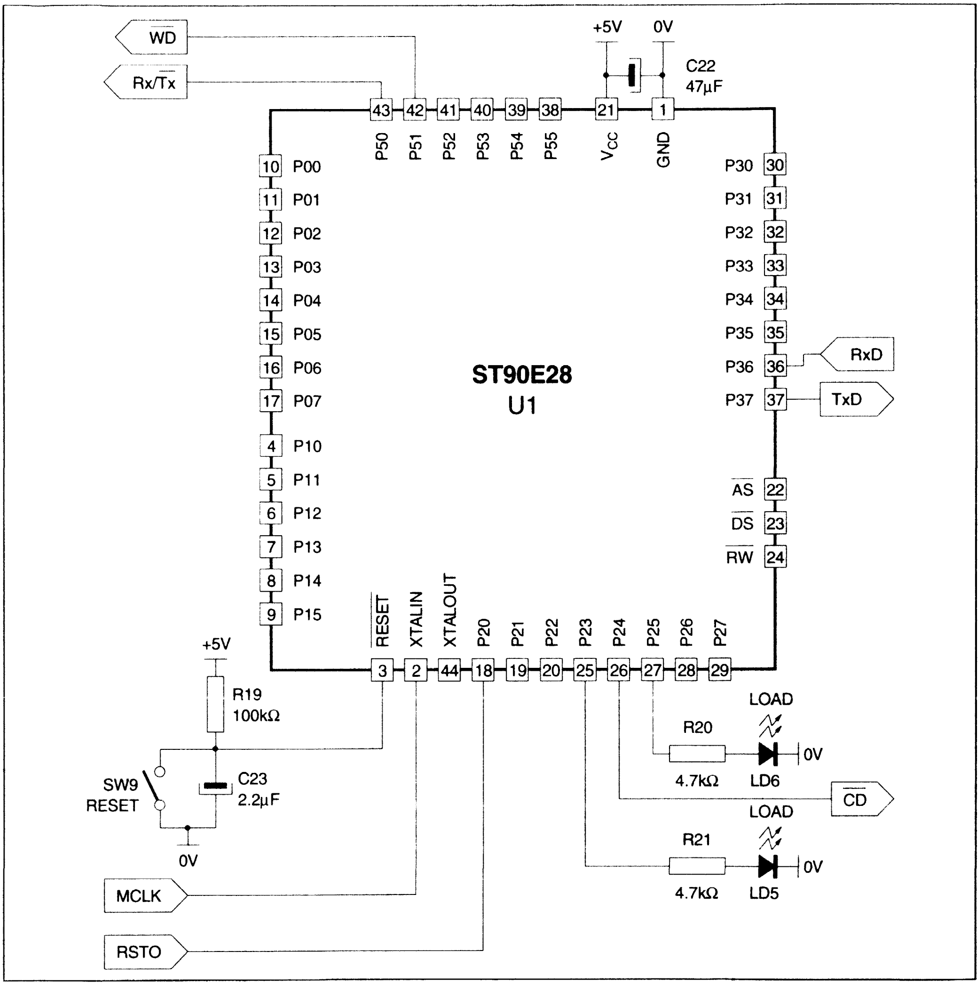


Figure 72 : Slave Configuration Board



PROGRAM

```
/**********************************************************************
           POWER LINE MODEM  COMMUNICATION PROGRAM
                      ST7537

           Revision : 1.3
           Date     : June 1993
           Author   : SAF Jean-Pierre
*********************************************************************/


#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
#include <bios.h>
#include <time.h>
#include <ctype.h>
#include <process.h>


/*####################################################################
  ######################    INITIALISATION    ########################
  ####################################################################*/

FILE *f;

void display();
void menu();
void interrupt (*old_it1)();
void interrupt (*old_it2)();
void connect_it();

char Tx_buff_text[100],Rx_buff_text1[100],Rx_buff_text2[100],buff[100];
char QBF1[100] ="THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG 0123456789";
char PATTERN[100] ,QBF1_PATTERN[100];
char test, ex, pr, pr1, pr2;
char path, file_char;

int Tx_buff_data[100] ,Rx_buff_data1[100] ,Rx_buff_data2[500];
int RxTx1 ,RxTx2 ,Tx_text_data ,Rx_size1 ,Rx_size2 ,Rx_size;
int Tx_num ,Rx_num11 ,Rx_num22 ,Rx_num1 ,Rx_num2;
int baud_rate ,stop_bit ,parity ,nb_data_bit ,settings;
int sb ,sb1 ,sb2 ,ndb ,ndb1 ,ndb2 ,br ,br1 ,br2 ,nbbit1 ,nbbit2;
int pc_select ,exit1 ,regF8 ,regFC , regFE ,COM ,select_com;
int x1 ,x2 , y1, y2, inter1, inter2, C1_C2, err;
int qbf1_pattern ,nb_qbf1_pattern ,cmp_qbf1 ,cmp_qbf1r ,cmp_qbf2r;
int Rx_nb_frame1 ,Rx_nb_frame2 ,frame_err;
int Tx_size_test, flag_file, flag_test, flag_Tx,flag_char;
int size_ref;

float cer;

long nbchar1,nbchar2;

clock_t start, end;
```

```
/*###################################################################
 #########################   Rx_Tx SETTING    #########################
 ####################################################################*/

/* Set, Clear or Toggle Request TO Send */

void Rx_Tx(int mode)      {
   switch (mode) {
   case 0: outportb( regFC, (inportb(regFC) | 0x02)); break;    /*Set   */
   case 1: outportb( regFC, (inportb(regFC) & 0x0D)); break;    /*Clear*/
   case 2: {if (select_com == 0)      /*  COM1 selected */
            { RxTx1 = !RxTx1; }   /* Toggle RxTx1    */
        else                      /*  COM2 selected */
            { RxTx2 = !RxTx2; }   /* Toggle RxTx2    */
        }
   break;        }
               }


/*###################################################################
 #################   Clear screen management    #####################
 ####################################################################*/

/* Clear  : - line in the selected window */
/*         - COM1 window                  */
/*         - COM2 window                  */
/*          - both windows                 */

void clear_line(int xc, int yc)      {
   if(select_com == 0)          /* COM1 selected */
     /*----------------> Clear line */
      {gotoxy(xc, yc);
       clreol();      }
   else                         /* COM2 selected */
     /*----------------> Clear line */
      {gotoxy(xc, yc+9);
       clreol();  }          }


void clear_win_COM1()    {
   int k;
   /*----------------> Clear COM1 window */
   for(k=4;k<12;k++)        {
      gotoxy(1,k);
      clreol();             }
               }


void clear_win_COM2()    {
   int k;
   /*----------------> Clear COM2 window */
   for(k=13;k<21;k++)       {
      gotoxy(1,k);
      clreol();             }
               }


void clear_win()    {
   int k;
   /*----------------> Clear window */
   for (k=4; k<12; k++)     {
      gotoxy(1, k);
      clreol();
      gotoxy(1, k+9);
      clreol();             }
             }
 *##################################################################
  ###############   Select one or two port COM   ##################
  ##################################################################*/

void sel_pc()    {
   char pc_buff[5];
   char pc;
   int f10;
   pc_buff[0] = 2;

   clrscr();
   gotoxy(23, 10);
   printf("Communication using :");
   gotoxy(23, 12);
   printf("1. One port COM  ( two computers )");
   gotoxy(23, 13);
   printf("2. Two port COMs ( one computer  )");
   do    {
      f10=0;
      gotoxy(2, 24);
      printf("Select : ");
      cgets(pc_buff);
      if(pc_buff[2]==0x00)  {      /* Default : one computer */
      pc_select = 2;
      break;              }
      else                   {
      pc=pc_buff[2];
      switch(pc)              {
      case '1' : pc_select = 1; break;          /* One computer   */
      case '2' : pc_select = 2; break;          /* Two computers */
      default   : f10 = 1;     }
                    }
      }
   while(f10 == 1);
   clrscr();
          }


/*##################################################################
 #################    RS232 PARAMETERS    #########################
 ###################################################################*/

void prog_RS232()           {
   int f1, f2, i;
   char *br_test, *sb_test;
   char br_buff[10],pr_buff[10], sb_buff[10];

   /*********  BAUD RATE ***********/
   do {
      f1 = 0;
      do {
      clear_line(24, 6);
      printf("Baud Rate (1200,600,300,150,110) : ");
      gets(br_buff);
      f2 = 0;

      for ( br_test = br_buff; *br_test; br_test++)
      /*---------------> Test if character is an integer */
      if (!isdigit(*br_test))      {
         f2=1;
         break;                 }
      /*---------------> Default : 1200 bauds */
         if(br_buff[0]==0x00) {
          br = 1200;
          baud_rate = 0x80;
          f2 = 0;
          break;           }
     }
      while (f2 == 1);
      if(br_buff[0] != 0x00) {
      sscanf(br_buff,"%d",&br);   /* conversion in an integer */
      switch(br)             {
      case 1200 : baud_rate = 0x80; break;       /* 1200 bauds */
      case 600  : baud_rate = 0x60; break;       /* 600 bauds */
      case 300  : baud_rate = 0x40; break;       /* 300 bauds */
      case 150  : baud_rate = 0x20; break;       /* 150 bauds */
      case 110  : baud_rate = 0x00; break;       /* 110 bauds */
      default   : f1 = 1;      }
                    }
     }
   while (f1 == 1);


   /***************  PARITY  ***************/
   do {
      f1 = 0;
      clear_line(24, 7);
      printf ("Parity (n:none, o:odd, e:even)   : ");
      cgets(pr_buff);
      /*----------------> Default : odd parity */
      if(pr_buff[2]==0x00)   {
      pr = ’O’;
      parity = 0x08;
      break;              }
      else                   {
      pr=toupper(pr_buff[2]);
      switch(pr)             {
      case ’N’ : parity = 0x00; break;           /* no parity   */
      case ’O’ : parity = 0x08; break;           /* odd parity  */
      case ’E’ : parity = 0x18; break;           /* even parity */
      default  : f1 = 1;     }
                     }
     }
   while (f1 == 1);


   /*************** NUMBER OF STOP BIT **********/
   do {
      f1 = 0;
      do {
      clear_line(24, 8);
         printf ("Number of Stop Bit(s) (1 or 2)    : ");
      gets(sb_buff);
      f2 = 0;

      for ( sb_test = sb_buff; *sb_test; sb_test++)
      /*----------------> Test if character is an integer */
      if (!isdigit(*sb_test))      {
         f2=1;
         break;                  }
      /*----------------> Default : 1 stop bit */
      if(sb_buff[0]==0x00)  {
          sb = 1;
          stop_bit = 0x00;
          f2 = 0;
          break;            }
     }
      while ( f2 == 1);

      if(sb_buff[0] != 0x00)  {
      sscanf(sb_buff,"%d",&sb);   /* conversion in an integer */
      switch(sb)              {
         case 1  : stop_bit = 0x00; break;        /* 1 stop bit */
      case 2  : stop_bit = 0x04; break;        /* 2 stop bits */
      default : f1 = 1;       }
                     }
     }
   while (f2 == 1);


   /*************** BIT / CHARACTER **********/
   ndb = 8;
   nb_data_bit = 0x03;

   clear_win();
}
```

```
/*##################################################################
 ##################   RS232 PROGRAMMATION    ########################
 ###################################################################*/

 void RS232()      {
   /*---------------> COM1 */
   if (select_com == 0)    {
      br1 = br;
      pr1 = pr;
      ndb1 = ndb;
      sb1 = sb;             }
   /*---------------> COM2 */
   else                    {
      br2 = br;
      pr2= pr;                       /* COM2 */
      ndb2 = ndb;
      sb2 = sb;             }

 settings = ( baud_rate | parity | stop_bit | nb_data_bit);
 bioscom( 0, settings, COM );
 }


/*##################################################################
 #########    RS232 INITIALISATION   1200bps ,8bit+1stop,odd   #####
 ###################################################################*/

 /*---------------> Initialisation 1200 bauds           */
 /*                           Odd parity          */
 /*                        8 bits/character     */
 /*                        1 stop bit           */

void init_RS232(){
   int set_init;
   baud_rate = 0x80;
   parity = 0x08;
   stop_bit = 0x00;
   nb_data_bit = 0x03;
   br1 =br2 = 1200;
   pr1 =pr2 = 'O';
   sb1 =sb2 = 1;
   ndb1=ndb2= 8;

   set_init = ( baud_rate | parity | stop_bit | nb_data_bit );
   if(pc_select==1) bioscom( 0, set_init, 0);
   if(pc_select==2)              {
      bioscom( 0, set_init, 1);
      bioscom( 0, set_init, 0); }
}


/*##################################################################
 ######################     TRANSMISSION    ########################
 ###################################################################*/

/* Transmission of : - Text                        */
/*                - Data in hexadecimal format */
/*                    - File                       */

void sel_text_data()   {
   char text_data_buff[5], txt_dt;
   char file_char;
   int f1,i;
   div_t  div_size;
   text_data_buff[0] = 2;
   /*---------------> Ask transmission format */
   clear_line(36, 5);
   printf("1. TEXT");
   clear_line(36, 6);
   printf("2. DATA");
   clear_line(36, 7);
   printf("3. FILE");
   do {
      f1=0;
      clear_line(36, 9);
      printf("Select : ");
      cgets(text_data_buff);
      if(text_data_buff[2]==0x00)    {       /* Default QBF1 */
      Tx_text_data = 0x00;
      break;                     }
      else                          {
      txt_dt=text_data_buff[2];
      switch(txt_dt)          {
      case '1' : Tx_text_data = 0x00; break;          /* QBF1 */
      case '2' : Tx_text_data = 0x01; break;          /* Pattern */
      case '3' : {Tx_text_data=0x03;
                     flag_file = 1;}     break;           /* FILE */
      default  : f1 = 1;    }
                        }
     }
   while(f1 == 1);

   if(select_com==0) clear_win_COM1();
   else clear_win_COM2();
}

void transmit_text()     {
   char empty_getchar, td, *p;
   char num_byte_buff[50], *num_byte_test;
   char byte_buff[50], *byte_test;
   char file_char1;
   int Tx_size, Tx_num_byte, td1, f6, f7, x, y, i;

   if ((inportb(regFE) & 0x80) != 0x80)  {      /*  Rx/Tx=0 and CD=1 */
           /* Transmission of text */

   /*---------------> Transmission of text */
      if (Tx_text_data == 0x00)    {
      Tx_num = 0;
     /*--------------->Read text and fill transmit buffer */
      clear_line(1,4);
      printf ("Enter the text to send : ");
      clear_line(1, 5);
     /*--------------->80 characters max */
      Tx_buff_text[0]=81;
      cgets(Tx_buff_text);
      Tx_size = Tx_buff_text[1];
     /*--------------->transmission */
      Rx_Tx(1);
      Rx_Tx(0);
      /*-------------->Preamble */
      bioscom( 1, 0xFF, COM );
         bioscom( 1, 0xFF, COM );
      bioscom( 1, 0xAA, COM );            /* send AA           */
      bioscom( 1, Tx_size, COM );         /* send size of text */
      bioscom( 1, Tx_text_data, COM );    /* send text         */
      /*-------------->Send the text */
      for (Tx_num = 2; Tx_num < Tx_size+2; Tx_num++)     {
      bioscom( 1, Tx_buff_text[Tx_num], COM );          }
      /*--------------> Wait before reception to save last character */
      delay(30);
      /*--------------> select reception */
      Rx_Tx(1);
      }
```

```
/*---------------> transmission of data in hexadecimal format */
   if( Tx_text_data == 0x01)     {
   Tx_num = 0;
   do {
      /*---------------> ask the number of bytes to send */
      clear_line(1, 4);
      printf("Number of bytes ( 64 bytes max ): ");
      gets(num_byte_buff);
      f6 = 0;

      /*---------------> test if the character is an integer */
      for(num_byte_test=num_byte_buff; *num_byte_test; num_byte_test++)
      if( (!isdigit(*num_byte_test)) || (num_byte_buff[0]==0x00)) {
         f6 = 1;
         break;                                                 }
      /*---------------> Conversion in integer */
      sscanf(num_byte_buff, "%d", &Tx_num_byte);
      }
   while((f6 == 1) || num_byte_buff[0]==0x00 || Tx_num_byte>64);


   /*---------------> Clear line 7 */
   clear_line(1, 7);
   /*---------------> Read the bytes */
   if(select_com == 0) y = 5;
   else y = 14;
   x = 1;

   for(Tx_num = 0; Tx_num < Tx_num_byte; Tx_num++)    {
      do {
         gotoxy(x, y);
         printf("Byte n<F128><144><F255>%d : ", Tx_num);
         gets(byte_buff);
         y = wherey();
         f7 = 0;

   /*---------------> test if hexadecimal character */
         for(byte_test=byte_buff; *byte_test; byte_test++)
         if( !isxdigit(*byte_test) )    {
         f7 = 1;
         y = wherey();
         y = y - 1;
         clreol();
         break;                    }

   /*---------------> Management of the screen */
         if (y == 11)  {
         x = x + 15;
         y = 5;      }
         if ( y == 20 )  {
         x = x + 15;
         y = 14;      }

         if (x > 70 )     {
         x = 2;
         if (select_com == 0) {
            y = 5;
            clear_win_COM1();   }
         else                 {
            y = 14;
            clear_win_COM2();   } }

        }
      while( f7 == 1);

      sscanf(byte_buff, "%x", &Tx_buff_data[Tx_num]);
   }
```

```
      /*---------------> Transmission mode */
      Rx_Tx(0);
      /*---------------> Preamble */
      bioscom( 1, 0xFF, COM);
      bioscom( 1, 0xFF, COM);
      bioscom( 1, 0xAA, COM);
      bioscom( 1, Tx_num_byte, COM);
      bioscom( 1, Tx_text_data, COM);
      /*---------------> Send bytes */
      for (Tx_num = 0; Tx_num < Tx_num_byte; Tx_num++) {
         bioscom( 1, Tx_buff_data[Tx_num], COM);        }

      /*---------------> Wait before reception to save last character */
      delay(30);
      /*---------------> Reception mode */
      Rx_Tx(1);
   }

   /*---------------> Transmission of file */
      if(Tx_text_data==0x03)   {
      clear_line(1,4);
      printf ("Enter filename : ");
      scanf("%s",&path);

      /*--------------->Test presence of File */
      if(( f=fopen(&path, "r")) == NULL)   {
         clear_win();
         clear_line(32,6);
         printf("Cannot open file");
         fclose(f);
         flag_file=0;
         clear_line(57, 11);
         printf("Hit any key to continue");
         while(kbhit()==0);
         getch();
         x1=x2=1;
         y1=5;
         y2=14;
         clear_win();                   }
                      }
   }
}


/*###################################################################
  ####################   FILE transmission   #######################
  ###################################################################*/

void Tx_FILE()   {
   int i;
   int Tx_char_size=0;

   /*---------------> Test if character different from EOF */
   if(file_char!=EOF)   {
      clear_line(35,6);
      printf("WORKING...");
      Tx_num=0;
      /*---------------> Fill buffer with one line of the file */
      while((fscanf(f,"%c",&file_char)==1) && file_char!=0x0A)   {
      Tx_buff_text[Tx_num]=file_char;
      Tx_num=Tx_num++;
         Tx_char_size=Tx_char_size++;                     }

      Tx_num=0;
      Rx_Tx(0);
      bioscom( 1, 0xFF, COM);
      bioscom( 1, 0xFF, COM);
      bioscom( 1, 0xAA, COM);
      bioscom( 1, Tx_char_size, COM);
      bioscom( 1, 0x00, COM);

      for(Tx_num=0; Tx_num<Tx_char_size; Tx_num++)      {
      bioscom( 1, Tx_buff_text[Tx_num], COM);          }
      delay(30);
      /* Reception mode */
      Rx_Tx(1);
      delay(500);

      }

      /*---------------> Test if End Of File */
      if (file_char==EOF){
      fclose(f);
         if(select_com==0) clear_win_COM1();
      else clear_win_COM2();
         file_char=0;
      flag_file=0;    }
}


/*####################################################################
 #####################      RECEPTION    #########################
 ####################################################################*/




void interrupt Rx_it1()  {                     /* COM1 interrupt routine */
   disable();                                  /* Forbid interruptions    */
   inter1=1;
   if(RxTx1==1)  inter1=1;
   else inter1=0;
   Rx_buff_text1[Rx_num1] = inportb(0x3F8);    /* Fill reception buffer   */
   Rx_buff_data1[Rx_num1] = inportb(0x3F8);
   Rx_num1 = Rx_num1++;
   if(inter1==0) Rx_num1=0;
   outportb( 0x20, 0x24);                  /* End of interrupt       */
   enable();            }                      /* Interruptions allowed   */


void interrupt Rx_it2()   {                     /* COM2 interrupt routine */
   disable();                               /* Forbid interruptions    */
   if(RxTx2==1 && pc_select==2) inter2=1;
   else inter2 = 0;                             /* Flag of interruption    */
   Rx_buff_text2[Rx_num2] = inportb(0x2F8);    /* Fill reception buffer   */
   Rx_buff_data2[Rx_num2] = inportb(0x2F8);
   Rx_num2 = Rx_num2++;
   if(inter2==0) Rx_num2=0;
   outportb( 0x20, 0x23);                /* End of interrupt      */
   enable();              }                     /* Interruptions allowed */


void enable_it()          {
   disable();
   /* ----------------------------------------> COM1 */
   /* -----> Line control register */
   outportb(0x3FB, (inportb(0x3FB) & 0x7F));          /* DLAB=0                */
   /* -----> Authorization interruption register */
   outportb(0x3F9, 0x01);                             /* Data received and ready */
   /* -----> Control register of modem signals */
   outportb( 0x3FC, ((inportb(0x3FC) | 0x0C) & 0x0F)); /* OUT1=OUT2=1           */
   /* -----> Interruption register */
   outportb(0x21, (inportb(0x21) & 0xEF));

   /* ----------------------------------------> COM2 */
   /* -----> Line control register */
   outportb(0x2FB, (inportb(0x2FB) & 0x7F));          /* DLAB=0                */
   /* -----> Authorization interruption register */
   outportb(0x2F9, 0x01);                             /* Data received and ready */
   /* -----> Control register of modem signals */
   outportb( 0x2FC, ((inportb(0x2FC) | 0x0C) & 0x0F)); /* OUT1=OUT2=1           */
   /* -----> Interruption register */
   outportb(0x21, (inportb(0x21) & 0xF7));
   enable();              }
```

```
void connect_it()             {
   disable();
   outportb(0x21, (inportb(0x21) | 0x18));     /* Interruption allowed on COM1 and COM2 */
   enable();
   /* --------------->   init */
   Rx_num1 = 0;
   Rx_num2 = 0;
   inter1 = 0;
   inter2 = 0;
   /*--------------->   connect ITs vectors */
   old_it1 = getvect(0x0C);                   /* Keep old COM1 IT vector */
   setvect(0x0C, Rx_it1);                      /* Set new COM1 IT vector   */
   old_it2 = getvect(0x0B);                /* keep old COM2 IT vector */
   setvect(0x0B, Rx_it2);                      /* Set new COM2 IT vector   */
   enable_it();
 }


void disconnect_it()  {
   setvect(0x0C, old_it1);          /* Set old COM1 and COM2 IT vectors */
   setvect(0x0B, old_it2);
   disable();                       /* disable IT for i8259A */
   outportb(0x21, (inportb(0x21) | 0x18));
   outportb(0x3F9, 0x00);              /* disable IT for i8250 */
   outportb(0x2F9, 0x00);
   enable();
   }

/*---------------> Reception of the preamble on COM1 */
void preamble1()  {
   int testaa, i;
   gotoxy(35, 24);
   printf("CD");
   testaa = 0;
   Rx_num11=0;

   start = clock();
   do              {
      delay(10);
      /* --------------> Test if AA is received for synchronisation */
      if( Rx_buff_data1[Rx_num11] != 0xAA ) testaa = 1;
      else testaa = 0;
      Rx_num11=Rx_num11++;
      end = clock();

      /* --------------> if AA not received before 0.5s */
      if (((end - start) / CLK_TCK) > 0.5)      {
         gotoxy(35, 24);
      printf("  ");
      inter1 = 0;
      break;                           }
      }
   while (testaa == 1);

   Rx_size1 = Rx_buff_data1[Rx_num11];      /* Size of the frame      */
   Rx_num11=Rx_num11++;
   Tx_text_data = Rx_buff_data1[Rx_num11]; /* Text or data received */
   Rx_num11=Rx_num11++;    }


void exec_rx1()         {
  int i, j, k, l, Rx_qbf1, err1;
  long bit1;
```

```
/*----------------------------> Reception of text */
  if ( Tx_text_data == 0x00)  {
    /*-------------------------->  Print of the text received */
    if (y1 > 10)      {
       clear_win_COM1();
       x1=1;
       y1=5;          }
    gotoxy(x1, y1);
    for(i=Rx_num11; i < (Rx_size1+Rx_num11); i++)          {
    gotoxy(x1, y1);
    putchar(Rx_buff_text1[i]);
    x1=x1++;
    if (x1 >  80)      {
        x1 = 1;
        y1 = y1++;  }                         }

    }

/*---------------> Reception of data in hexadecimal format */
   if ( Tx_text_data == 0x01)
   {
      /*--------------> Print of the datas */
      gotoxy(x1,y1);
      for(i=Rx_num11; i<(Rx_size1+Rx_num11); i++)      {
      gotoxy(x1, y1);
      cprintf("%x", Rx_buff_data1[i]);
      x1 = x1 + 3;

      /*--------------> Management of screen */
      if (x1 >  76)  {
         x1 = 1;
         y1 = y1++;  }
         if (y1 > 10)   {
         clear_win_COM1();
         x1 = 1;
         y1 = 5;     }
                               }
   }

/*---------------> Reception of QBF1 test */
   if(Tx_text_data == 0x02)   {
      /*---------------> Clear screen at first reception */
      if(cmp_qbf1r == 0)   {
       clear_win_COM1();
       x1=1;
       y1=5;            }

      cmp_qbf1r=cmp_qbf1r++;

      Rx_qbf1 = Rx_buff_data1[Rx_num11];       /* Read number of frames sent */
      Rx_num11= Rx_num11++;
      Rx_nb_frame1 = Rx_buff_data1[Rx_num11]; /* Read frame number           */
      Rx_num11=Rx_num11++;

      err1=0;

      /*---------------> Print frame */
      for(i= Rx_num11; i < (Rx_size1+Rx_num11); i++)  {
      gotoxy(x1, y1);
      putchar(Rx_buff_text1[i]);
      nbchar1=nbchar1++;
      /*------------> Test if character error */
      if (Rx_buff_text1[i] != QBF1_PATTERN[i-Rx_num11]) {
         err = err++;
         err1=1;      }
      x1=x1++;                             }

     /*---------------> Management of screen */
      y1 = wherey();
         if (x1 >  79)      {
         x1 = 1;
         y1 = y1++;       }
      if (y1 > 9)    {
         for (j=5; j<12; j++)  {
            gotoxy(1, j);
            clreol();           }
         x1 = 1;
         y1 = 4;      }
    /* if at least one error in a frame, increment frame_err */
    if(err1) frame_err = frame_err++;
    err1=0;
    /* calculation of received bits number */
    bit1 = nbchar1*((long) nbbit1);

    gotoxy(1,4);
    printf("errors : %d", err);
    gotoxy(20,4);
    printf("Frame n<F128><144><F255> : %d", cmp_qbf1r);
    gotoxy(40,4);
    printf("Bits : %ld", bit1);

    /* Empty reception buffer */
    for(l=0;l<80;l++)          {
      Rx_buff_text1[l] = 0x00;
      Rx_buff_data1[l] = 0x00;}

    /*----------------> Test end of test */
    if ((cmp_qbf1r == Rx_qbf1) || (Rx_nb_frame1==Rx_qbf1))     {
      clear_win_COM1();

      /* frame error = not received frame + wrong received frame */
      frame_err = frame_err + Rx_qbf1 - cmp_qbf1r;

      gotoxy(5,5);
      printf("Frames received     : %d",cmp_qbf1r);
      gotoxy(40,5);
      printf("Total frame ERRORS : %d",frame_err);
      gotoxy(5,6);
      printf("Characters received : %ld",nbchar1);
      gotoxy(40,6);
      printf("Character ERRORS   : %d", err);
      gotoxy(5,7);
      printf("Total bits received : %ld", bit1);

      /*----------------> CER */
      cer=( ((float) err)/( ((float) nbchar1)) ) *100;
      gotoxy(5,10);
      printf("CHARACTER ERROR RATE ( CER )  : %4.2f%",cer);

      gotoxy(57,11);
      printf("Hit any key to continue");
      while(kbhit()==0);
      clear_win();
      getch();
      y1=4;
      cmp_qbf1r=0;
      nbchar1 =0;
      err = 0;
      frame_err=0;    }
     }

   if(Rx_size1 > 1)      {
       y1=y1++;
       x1=1;            }

   for(i=0;i<100;i++)         {
      Rx_buff_text1[i]=0x00;
      Rx_buff_data1[i]=0x00;   }

   inter1 = 0;                    /* Set flag of interruption to 0 */
   Rx_num1 = 0;
   Rx_size1 = 0;
   C1_C2=0;
   gotoxy(35, 24);
   printf("  ");}                 /* Clear CD                      */


void preamble2(){
   int testaa;
   gotoxy(75, 24);
   printf("CD");
   testaa = 0;
   Rx_num22 = 0;
```

```
/* Test if AA is received for synchronisation */
   start = clock();
   do    {
      delay(10);
      if( Rx_buff_data2[Rx_num22] != 0xAA ) testaa = 1;
      else testaa = 0;
      Rx_num22 = Rx_num22++;
      end = clock();
      if (((end - start) / CLK_TCK) > 0.5)      {
      gotoxy(75, 24);
      printf("  ");
      inter2 = 0;
      break;}
      }
   while (testaa == 1);

   Rx_size2 = Rx_buff_data2[Rx_num22];     /* Size of the frame */
   Rx_num22 = Rx_num22++;
   Tx_text_data = Rx_buff_data2[Rx_num22]; /* Text or data received */
   Rx_num22 = Rx_num22++;}


void exec_rx2()          {
   int i, j, k, l, Rx_qbf2, errl;
   long bit2;

/*---------------> Reception of text */
   if ( Tx_text_data == 0x00) {              /* Reception of text   */

      if (y2 > 19)     {
      clear_win_COM2();
      x2 = 1;
      y2 = 14;      }

      gotoxy(x2, y2);
      for(i=Rx_num22; i < (Rx_size2+Rx_num22); i++) {    /* Print text received */
      gotoxy(x2, y2);
      putchar(Rx_buff_text2[i]);
      x2=x2++;
      if (x2 >  80)    {
         x2 = 1;
         y2 = y2++;    }                    }

   }

/*---------------> Reception of data in hexadecimal format */
   if ( Tx_text_data == 0x01)
   {

      /* Print of the datas */
      for(i=Rx_num22; i<(Rx_size2+Rx_num22); i++)     {
      gotoxy(x2, y2);
      cprintf("%x", Rx_buff_data2[i]);
      x2 = x2 + 3;

/* Management of screen */
      y2 = wherey();
         if (y2 > 19)    {
         clear_win_COM2();
         x2 = 1;
         y2 = 14;     }
      if (x2 >  76)    {
         x2 = 1;
         y2 = y2++;    }
       }
   }

/*---------------Reception of Tests */
   if(Tx_text_data == 0x02)    {
      if(cmp_qbf2r == 0)    {
       clear_win_COM2();
       x1=1;
       y1=14;            }
      cmp_qbf2r=cmp_qbf2r++;
   Rx_qbf2 = Rx_buff_data2[Rx_num22];
   Rx_num22= Rx_num22++;
   Rx_nb_frame2 = Rx_buff_data2[Rx_num22];
   Rx_num22=Rx_num22++;
   err1=0;

   for(i= Rx_num22; i < (Rx_size2+Rx_num22); i++)           {
   gotoxy(x2, y2);

   putchar(Rx_buff_text2[i]);
   nbchar2=nbchar2++;
   if (Rx_buff_text2[i] != QBF1_PATTERN[i-Rx_num22])      {
      err = err++;
      err1 = 1;                                   }
   x2=x2++;}

  /*----------------> Management of screen */
   y2 = wherey();
   if (x2 >  79)       {
      x2 = 1;
      y2 = y2++;       }
   if (y2 > 18)     {
      for (j=14; j<21; j++)  {
         gotoxy(1, j);
         clreol();            }
      x2 = 1;
      y2 = 13;       }

 if(err1) frame_err = frame_err++;
 err1=0;

 bit2 = nbchar2*((long) nbbit2);

 gotoxy(1,13);
 printf("errors : %d", err);
 gotoxy(20,13);
 printf("Frame n<F128><144><F255> : %d", cmp_qbf2r);
 gotoxy(40,13);
 printf("Bits : %ld", bit2);

 for(l=0;l<80;l++)           {
   Rx_buff_text2[l] = 0x00;
   Rx_buff_data2[l] = 0x00; }

 if ((cmp_qbf2r >= Rx_qbf2) || (Rx_nb_frame2==Rx_qbf2))   {
   clear_win_COM2();
   frame_err = frame_err + Rx_qbf2 - cmp_qbf2r;
   gotoxy(5,14);
   printf("Frames received     : %d",cmp_qbf2r);
   gotoxy(40,14);
   printf("Total frame ERRORS : %d",frame_err);
   gotoxy(5,15);
   printf("Characters received : %ld",nbchar2);
   gotoxy(40,15);
   printf("Character ERRORS   : %d", err);
   gotoxy(5,16);
   printf("Total bits received : %ld", bit2);

   cer=( ((float) err)/( ((float) nbchar2)) ) *100;
   gotoxy(5,19);
   printf("CHARACTER ERROR RATE ( CER )  : %4.2f%",cer);

   gotoxy(57,20);
   printf("Hit any key to continue");
   while(kbhit()==0);
   clear_win();
   getch();
   y2=13;
   cmp_qbf2r=0;
   err = 0;
   nbchar2=0;
   frame_err=0;     }
  }

if(Rx_size2 > 1)      {
    y2=y2++;
    x2=1;               }
   for(i=0;i<100;i++)          {
      Rx_buff_text2[i]=0x00;
      Rx_buff_data2[i]=0x00;    }

   inter2 = 0;                    /* Set flag of interruption to 0 */
   Rx_num2 = 0;
   Rx_size2 = 0;
   C1_C2=0;
   gotoxy(75, 24);
   printf("  ");
}


/*###################################################################
 ##################   QBF1 and PATTERN tests    ####################
 ###################################################################*/

/* select an tranmit QBF1 test or Pattren test */

void nb_QBF1_pattern(){
   char qbf1_pattern_buff[10], *qbf1_pattern_test;
   int f8;

   do {
      clear_line(31, 5);
      printf ("Number of frames : ");
      gets(qbf1_pattern_buff);
      f8 = 0;
      for ( qbf1_pattern_test = qbf1_pattern_buff; *qbf1_pattern_test; qbf1_pattern_test++)
      /*----------------> Test if character is an integer */
      if (!isdigit(*qbf1_pattern_test))       {
      f8 = 1;
      break;                              }
      }
   while ((f8 == 1) || qbf1_pattern_buff[0]==0x00);

   sscanf(qbf1_pattern_buff,"%d",&nb_qbf1_pattern);    /* conversion in an integer */

   clear_win();
}


void sel_QBF1_pattern()    {
   char qbf1_pattern_buff[5], qb_pt;
   int f1,i;
   qbf1_pattern_buff[0] = 2;

   /*---------------> Selection between QBF1 test and pattern test */
   clear_line(35, 6);
   printf("1. QBF1");
   clear_line(35, 7);
   printf("2. Pattern");
   do    {
      f1=0;
      clear_line(35, 9);
      printf("Select : ");
      cgets(qbf1_pattern_buff);
      if(qbf1_pattern_buff[2]==0x00)    {       /* Default : QBF1 */
      qbf1_pattern = 1;
      break;                }
      else                     {
      qb_pt=qbf1_pattern_buff[2];
      switch(qb_pt)            {
      case '1' : qbf1_pattern = 1; break;            /* QBF1 */
      case '2' : qbf1_pattern = 2; break;            /* Pattern */
      default  : f1 = 1;    }
                    }
      }
   while(f1 == 1);
   clear_win();

   /*---------------> QBF1 test selected */
   if(qbf1_pattern==1)    {
      for(i=0;i<54;i++) QBF1_PATTERN[i]=QBF1[i];
      Tx_size_test=54;     }
   /* --------------> Pattern test selected */
   if(qbf1_pattern==2)    {
      clear_line(1,4);
      printf ("Enter the pattern : ");
      clear_line(1, 5);
      /*--------------------------->80 characters max */
      PATTERN[0]=81;
      cgets(PATTERN);
      Tx_size_test=PATTERN[1];
      for(i=0;i<Tx_size_test;i++) QBF1_PATTERN[i]=PATTERN[i+2];

      clear_win();         }
   }


void test_QBF1(){
   int Tx_size ,i ,j, k;
   long bit;

   if (RxTx1==0) i=5;
   else if (RxTx2==0) i=14;

   /*---------------> Test if all frames sent */
   if(cmp_qbf1 < nb_qbf1_pattern)    {
      Rx_Tx(0);
      bioscom( 1, 0xFF, COM );
      bioscom( 1, 0xFF, COM );
      bioscom( 1, 0xAA, COM );                  /* send AA                   */
      bioscom( 1, Tx_size_test, COM );          /* send size of text         */
      bioscom( 1, 0x02, COM );                  /* send test                 */
      bioscom( 1, nb_qbf1_pattern, COM);        /* send number of QBF1 test  */
      bioscom( 1, (cmp_qbf1+1), COM);           /* send frame number         */
      for(j=0; j<Tx_size_test; j++)      {      /* send frame                */
      bioscom( 1, QBF1_PATTERN[j], COM );  }
      delay(30);
      Rx_Tx(1);
      delay(500);

      /* Calculation of bit number sent */
      if(RxTx1==0) bit = ((long)(cmp_qbf1+1)*((long)nbbit1)*((long)Tx_size_test));
      else if(RxTx2==0)  bit = ((long)(cmp_qbf1+1)*((long)nbbit2)*((long)Tx_size_test));

      gotoxy(5,i);
      printf("Number of frames transmitted     : %d", cmp_qbf1+1);
      gotoxy(5,i+1);
      printf("Number of characters transmitted : %d", (cmp_qbf1+1)*Tx_size_test);
      gotoxy(5,i+2);
      printf("Number of bits transmitted       : %ld",  bit);

      cmp_qbf1 = cmp_qbf1++;  }

   else {
      if(pc_select==1)    {
         gotoxy(57,i+6);
         printf("Hit any key to continue");
         while(kbhit()==0);
         getch();
         clear_win();}

      /*----------------> Empty QBF1_PATTERN buffer */
      for(k=0;k<100;k++) QBF1_PATTERN[k]=0x00;
      Tx_text_data = 0;
      flag_test = 0;
      menu();
      cmp_qbf1 = 0;
      x1 = x2 = 1;
      y1 = 5;
      y2 = 14;

         }
```

```
/*###################################################################
 ######################   Menu bar   ###############################
 ####################################################################*/

void menu(){
   gotoxy(1, 24);   /* print menu bar */
   clreol();
   if(select_com == 0){ /* selection of COM1 */
      gotoxy(4, 24);
      printf("C1");
      gotoxy(44, 24);
      printf("c2");   }
   else               { /* selection of COM2 */
      gotoxy(4, 24);
      printf("c1");
      gotoxy(44, 24);
      printf("C2");   }

   gotoxy(9, 24);
  printf("%d %c%d%d",br1,pr1,ndb1,sb1);/*baud rate,parity,data bit,stop bit*/
  gotoxy(49, 24);
  printf("%d %c%d%d", br2, pr2, ndb2, sb2);

   if(select_com == 0)     {     /* Rx, Tx */
      gotoxy(20, 24);
      if (RxTx1 == 1) printf("Rx");
      else printf("Tx");
      gotoxy(60, 24);
      printf("Rx");     }
   else                 {
      gotoxy(60, 24);
      if (RxTx2 == 1) printf("Rx");
      else printf("Tx");
      gotoxy(20, 24);
      printf("Rx");     }

   if(select_com == 0)              {     /* Print QBF1 */
      gotoxy(27, 24);
      if (flag_test) printf("CER");
      else printf("   ");             }
   else                              {
      gotoxy(67, 24);
      if (flag_test) printf("CER");
      else printf("   ");             }

   gotoxy(80, 25);
}


/*###################################################################
 ######   Calculation of number of bits in one character   #########
 ####################################################################*/

void nb_bit()   {
   if(pr1=='N') nbbit1 = 1 + ndb1 + sb1;
   else if( (pr1=='O') || (pr1=='E') ) nbbit1 = 1 + ndb1 + 1 + sb1;
   if(pr2=='N') nbbit2 = 1 + ndb2 + sb2;
   else if( (pr2=='O') || (pr2=='E') ) nbbit2 = 1 + ndb2 + 1 + sb2;
          }

/*###################################################################
 ######################    display   ##############################
 ####################################################################*/

void display(){
   int lig = 0;
   gotoxy(1,1);
   printf(" SGS-THOMSON Microelectronics (copyright)\n");
   printf(" ST 7537 Power Line Modem");
   gotoxy(52,1);
   printf("June 1993");
   gotoxy(74,1);
   printf("Rev 1.3");
   for (lig =1 ; lig < 76; lig++){     /* print of the bordure */
      gotoxy(lig, 3);
      printf("%c", 0xCD);
      gotoxy(lig, 12);
      printf("%c", 0xCD);        }
   for (lig =1 ; lig < 80; lig++){
      gotoxy(lig, 23);
      printf("%c", 0xC4);
      gotoxy(lig, 25);
      printf("%c", 0xC4);
      gotoxy(lig, 21);
      printf("%c", 0xCD);      }

   gotoxy(77,3);
   printf("COM1");
   gotoxy(77,12);
   printf("COM2");
}

/*######################################################################
 #########################   Keys   ##############################
 ######################################################################*/

void keys(){
   clrscr();
   gotoxy(15, 7);
   printf("  <-  : Select COM 1");
   gotoxy(15, 8);
   printf("  ->  : Select COM 2");
   gotoxy(15, 9);
   printf("  F1  : Configuration of COM");
   gotoxy(15, 10);
   printf("  F2  : Toggle Request To Send");
   gotoxy(15, 11);
   printf("  F3  : Run the transmission of text, data or file in CENELEC");
   gotoxy(15, 12);
   printf("  F4  : Run QBF1 test or pattern test in CENELEC");
   gotoxy(15, 13);
   printf("  F5  : Clear Screen");
   gotoxy(15, 14);
   printf("  F6  : Clear COM1 Window");
   gotoxy(15, 15);
   printf("  F7  : Clear COM2 Window");
   gotoxy(15, 16);
   printf("  F8  : Select 1 or 2 port COM(S) for communication");
   gotoxy(15, 17);
   printf(" F10  : Exit");
   gotoxy(15, 18);
   printf("HOME  : Display this screen");
   gotoxy(2, 24);
   printf("Hit any key to continue");
   while(kbhit() == 0);
   getch();
   clrscr();
}

/*######################################################################
 #############   SGS THOMSON Microelectronics   ####################
 ######################################################################*/

void sgs(){
   clrscr();
   gotoxy(1,5);
   printf("\n        ***** ***** *****      ***** *   * ***** ** ** ***** ***** *   *   ");
   printf("\n        *     *     *            *   *   * *   * * * *     *   * **  *   ");
   printf("\n        ***** *  ** ***** ****   *   ***** *   * *   * ***** *   * * * *   ");
   printf("\n            * *   *     *        *   *   * *   * *   *     * *   * *  **   ");
   printf("\n        ***** ***** *****        *   *   * ***** *   * ***** ***** *   *   ");
   gotoxy(1,15);
   printf("\n         ** ** * *** *** *** *** *   *** *** *** *** *** *  * * *** ***  ");
   printf("\n         * * * * *   * * * * *   *   *   *    *  * * * * ** * * *   *    ");
   printf("\n         *   * * *   **  * * **  *   **  *    *  **  * * * ** * *   ***  ");
   printf("\n         *   * * *   * * * * *   *   *   *    *  * * * * *  * * *     *  ");
   printf("\n         *   * * *** * * *** *** *** *** ***  *  * * *** *  * * *** ***  ");
```

```
while(kbhit()==0);
clrscr();
getch();
gotoxy(1,2);
printf("\n             ***** ***** * * * ***** *****     *     * *   * ***** ");
printf("\n             *   * *   * * * * *     *   *     *     * **  * *     ");
printf("\n             ***** *   * * * * ***   ****      *     * * * * ***   ");
printf("\n             *     *   * * * * *     *  *      *     * *  ** *     ");
printf("\n             *     ***** ** ** ***** *   *     ***** * *   * ***** ");

gotoxy(1,8);
printf("\n                         ** ** ***** ****  ***** ** ** ");
printf("\n                         * * * *   * *   * *     * * * ");
printf("\n                         *   * *   * *   * ***   *   * ");
printf("\n                         *   * *   * *   * *     *   * ");
printf("\n                         *   * ***** ****  ***** *   * ");

gotoxy(1,14);
printf("\n                     ***** *****   ***** ***** ***** ***** ");
printf("\n                     *       *         * *         *     * ");
printf("\n                     *****   *        *  *****   ***    *  ");
printf("\n                         *   *       *       *     *   *   ");
printf("\n                     *****   *       *   ***** *****   *   ");

gotoxy(2,24);
printf("Rev 1.3");
gotoxy(70,24);
printf("June 1993");

while(kbhit()==0);
clrscr();
getch();
}

/*##################################################################
 ############   Select COM (register definition)   #################
 ###################################################################*/

void com(){
   if (select_com == 0)     {     /* COM1 */
      regFC = 0x3FC;
      regFE = 0x3FE;
      regF8 = 0x3F8;
      COM = 0;          }
   else                 {       /* COM2 */
      regFC = 0x2FC;
      regFE = 0x2FE;
      regF8 = 0x2F8;
      COM = 1;          }
}


/*##################################################################
  ####################   Key execution   ###########################
  ##################################################################*/

/*----------------> RS232 programming */
void F1_exec()   {
   clear_win();
   prog_RS232();
   RS232();
   nb_bit();
   gotoxy(1, 22);
   clreol();
   gotoxy(1, 24);
   clreol();
   connect_it();
   menu();
   x1 = x2 = 1;
   y1 = 5;
   y2 = 14;     }

/*----------------> Transmission of Text, Data or File */
void F3_exec()   {
   if(select_com==0) clear_win_COM1();
   else clear_win_COM2();
   sel_text_data();
   transmit_text();
   flag_Tx=0;
   display();
   menu();        }


/*----------------> transmission of QBF1 test or Pattern test */
void F4_exec()   {
   int i;
   clear_win();
   if((RxTx1==1) && (RxTx2==1))   {
      for(i=0;i<100;i++) QBF1_PATTERN[i]=0x00;
      sel_QBF1_pattern();}

   else if((RxTx1==0) || (RxTx2==0))    {
      for(i=0;i<100;i++) QBF1_PATTERN[i]=0x00;
      flag_test=1;
      menu();
      x1 = x2 = 1;
      y1 = 5;
      y2 = 14;
      sel_QBF1_pattern();
      nb_QBF1_pattern();
      Tx_text_data=0x02;     }
   err = 0;
   nbchar1 = nbchar2 = 0;
   frame_err = 0;
   cmp_qbf1 = 0;
   cmp_qbf1r = cmp_qbf2r = 0;
}

void char_exec()    {
   Rx_Tx(0);
   bioscom(1, 0xFF, COM);
   bioscom(1, 0xFF, COM);
   bioscom(1, 0xAA, COM);
   bioscom(1, 0x01, COM);
   bioscom(1, 0x00, COM);
   bioscom(1, test, COM);
   delay(30);
   Rx_Tx(1);
   flag_char=0;
                  }

/*---------------> Exit */
void F10_exec()   {
   clrscr();
   do  {
      exit1 = 0;
      gotoxy(37, 10);
      printf("EXIT :");
      gotoxy(37, 12);
      printf("- Yes");
      gotoxy(37, 14);
      printf("- No");
      ex = toupper(getch());
      switch(ex)   {
      case 'Y' : {   clrscr();
               disconnect_it();
               exit(0);      }
               break;
      case 'N' : {   clrscr();
               display();
               menu();
               exit1 = 0; }
               break;
      default  : exit1 = 1;     }
      }
      while(exit1 == 1);   }
```

```
/*#################################################################
 #######################   Main Routine   #########################
 ##################################################################*/

main(){
      int k;
      sgs();
      keys();
      sel_pc();
/*--------------->  Initialisation of COM1, COM2 screen display */
      x1=1;
      y1=5;
      x2=1;
      y2=14;
      C1_C2 = 0;
/*--------------->  initialisation of QBF1-PATTERN test */
      file_char=0;
      flag_test = 0;
      flag_file=0;
      cmp_qbf1  = 0;
      cmp_qbf1r = 0;
      cmp_qbf2r = 0;
      nbchar1 = 0;
      nbchar2 = 0;
      err = 0;
      frame_err = 0;

      init_RS232();                     /* Initialisation of port RS232 */
      nb_bit();                    /* nb of bits in one character  */
      RxTx1 = 1;                     /* Set ST7537 in Rx mode        */
      RxTx2 = 1;
      Tx_text_data=0;
      select_com = 0;               /* Set-up COM1                 */
      com();
      display();                    /* Management of screen        */
      menu();
      connect_it();                   /* Init interrupt vectors       */

      while(1)
      {
      if (inter1)   {         /* if interruption on COM1 */
         preamble1();
         if(Tx_text_data!=0x02) Rx_size=Rx_size1+4;
         else Rx_size=Rx_size1+6;
         if( (Rx_num1 > Rx_size) || C1_C2)   exec_rx1();
                 }

      else if (inter2)       {
         preamble2();
         if(Tx_text_data!=0x02) Rx_size=Rx_size2+4;
         else Rx_size=Rx_size2+6;
         if( (Rx_num2 > Rx_size) || C1_C2)   exec_rx2();
                     }
      else      {
      /*----------------> Key hit */
      if (kbhit())    {
         test = getch();
         if( test == 0 )       {
         test = getch();
         switch(test)      {
/* F1 */       case 59 : F1_exec();  break;

/* F2 */       case 60 : {   clear_win();
                    Rx_Tx(2);
                    x1 = x2 = 1;
                    y1 = 5;
                    y2 = 14;
                    menu(); }
                break;

/* F3 */       case 61 : {   if ((RxTx1 == 0) || (RxTx2 == 0)) {
                    flag_Tx=1;}}
                break;

/* F4 */       case 62 : F4_exec();  break;
/* F5 */        case 63 : {   x1 = x2 = 1;
                     y1 = 5;
                     y2 = 14;
                              err = 0;
                     nbchar1 = nbchar2 = 0;
                     frame_err = 0;
                     cmp_qbf1 = 0;
                     cmp_qbf1r = cmp_qbf2r = 0;
                     clrscr();
                     display();
                     menu(); }
                 break;

/* F6 */        case 64 : {   x1 = 1;
                     y1 = 5;
                     err = 0;
                     nbchar1 = 0;
                     frame_err = 0;
                     cmp_qbf1 = cmp_qbf1r = 0;
                     clear_win_COM1();
                     }
                 break;

/* F7 */        case 65 : {   x2 = 1;
                     y2 = 14;
                              err = 0;
                     nbchar2 = 0;
                     frame_err = 0;
                     cmp_qbf1 = cmp_qbf2r = 0;
                     clear_win_COM2();}
                 break;

/* F8 */        case 66 : {   sel_pc();
                     init_RS232();
                              x1 = x2 = 1;
                     y1 = 5;
                     y2 = 14;
                     display();
                     menu();}
                 break;

/* HOME */      case 71 : {   keys();
                     display();
                     menu();     }

/* LEFT */      case 75 : {   select_com = 0;
                     RxTx2 = 1;
                     com();
                     menu();     }
                break;

/* RIGHT */     case 77 : {   select_com = 1;
                     RxTx1 = 1;
                     com();
                     menu();     }
                break;

/* F10 */       case 68 :   F10_exec();  break;

          default : menu();

        } /* switch */
     } /* if test = 0 */
```

```
/* if one character hit */
      else        {
         /*---------------> Transmission of one character */
         if ((RxTx1 == 0) || (RxTx2 == 0))  flag_char=1;
            }
      }
      }
      /*---------------> Transmission of tests */
      if (flag_test)   {
       if(pc_select==2)   {
          connect_it();
          C1_C2=1;         }
       test_QBF1();
       if(pc_select==1) connect_it();
                 }

      if (flag_Tx)  {
         if(pc_select==2)   {
          connect_it();
          C1_C2=1;         }
      F3_exec();
      if(pc_select==1) connect_it();        }

      /*---------------> Transmission of file */
      if (flag_file)  {
      if(pc_select==2)   {
          connect_it();
          C1_C2=1;         }
      Tx_FILE();
      if(pc_select==1) connect_it();        }

      if (flag_char)  {
         if(pc_select==2)   {
         connect_it();
         C1_C2=1;         }
      char_exec();
      if(pc_select==1) connect_it();        }

    }
 }
```

REFERENCES

1. **WACKS (Kenneth P.)** Utility load management using home automation, IEEE Transactions on Consumer Electronics, Vol 37, N°2, pp 168-174, May 1991.
2. **O'NEAL (J.B, Jr.),** The residential power circuit as a communication medium, IEEE Transactions on Consumer Electronics, Vol CE-36, N°3, pp 567-577, August 1986.
3. **VINES (Roger M.),** TRUSSEL (Jel), GALE (Louis J.), Noise on Residential power distribution circuits, IEEE Transactions on Electromagnetic Compatibility, Vol EMC-26, N°24, pp 161-168, November 1984.
4. **LEWART (Cass),** Modem handbook for the communications professional, Elsevier Science Publishing Co., 1987.
5. **SGS-THOMSON Microelectronics,** ST9 family 8/16 bit MCU programming manual, 1991, ST9 serie.
6. **SGS-THOMSON Microelectronics,** ST9 family 8/16 bit MCU technical manual, 1991, ST9 serie.
7. **BORLAND,** TURBO C : User's manual, 1988.
8. **CHAFFANJON D.,** Courants porteurs sur installation électrique d'un logement (aspects physiques).

ST7537 STARTER KIT

ST7537 POWER LINE MODEM

By Joël HULOUX

SUMMARY

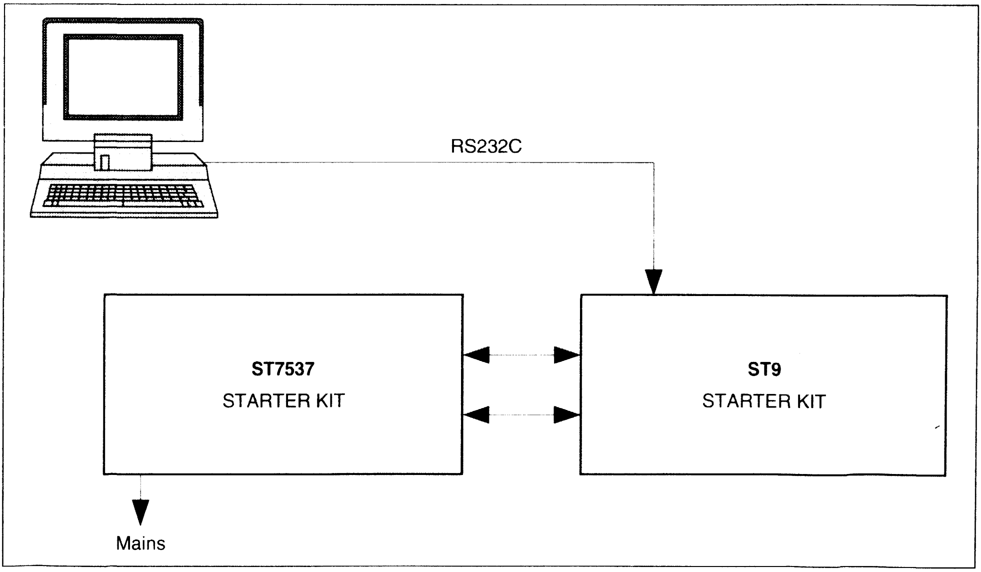
I - INTRODUCTION

The ST7537 starter kit has been developped in order to help customer for designing a Home Automation System.

Everything needed for using the micro-controller ST9 is on board and the ST7537 starter kit is fully compatible with the ST9 starter kit.

All connections for ST7537 and ST9 are on board and the only wiring you have to do is your application.

The board has been designed for the ST90E40 so you are not limited on resources and you can choose the microcontroller you want.



II - STARTER KIT FEATURES

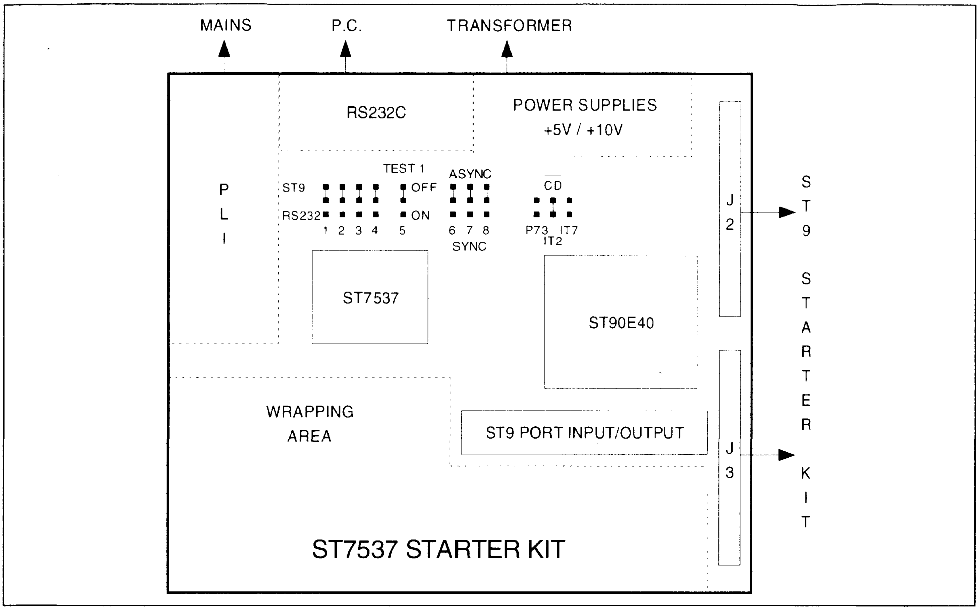
On the starter kit 2 applications are selectable :

- RS232C application
- ST9 application

With the starter kit you have :

- application note
- starter kit board
- power transformer
- RS232C cable
- demo software

III - BOARD CONFIGURATION

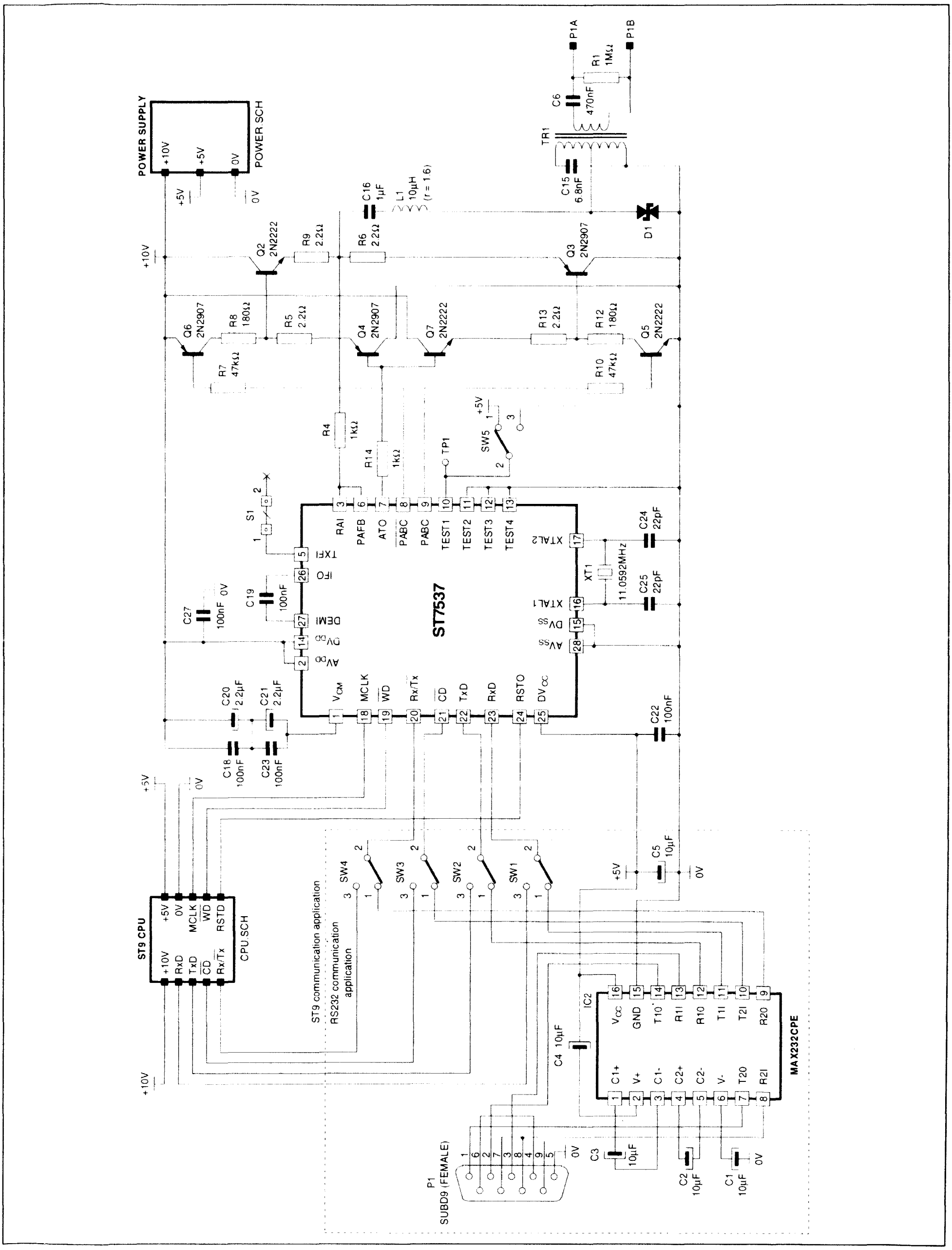


- Application choice is made thanks to SW 1,2,3 and 4. You can select ST9 or RS232C application.
- During your developpment you could need to transmit more than 1 sec (specified by CENELEC), you can do so with SW5 (Pin TEST1 of 7537).
- The Carrier Detector can be connected either to INT2, INT5 or INT7 by using the JP1.
- The starter can be configured to work in asynchronous or in synchronous mode with :

SW6	→	Rxd connected to Sin (P70)	Async
		Rxd connected to P74	Sync
SW7	→	Txd connected to Sout (P71)	Async
		Txd connected to P75	Sync

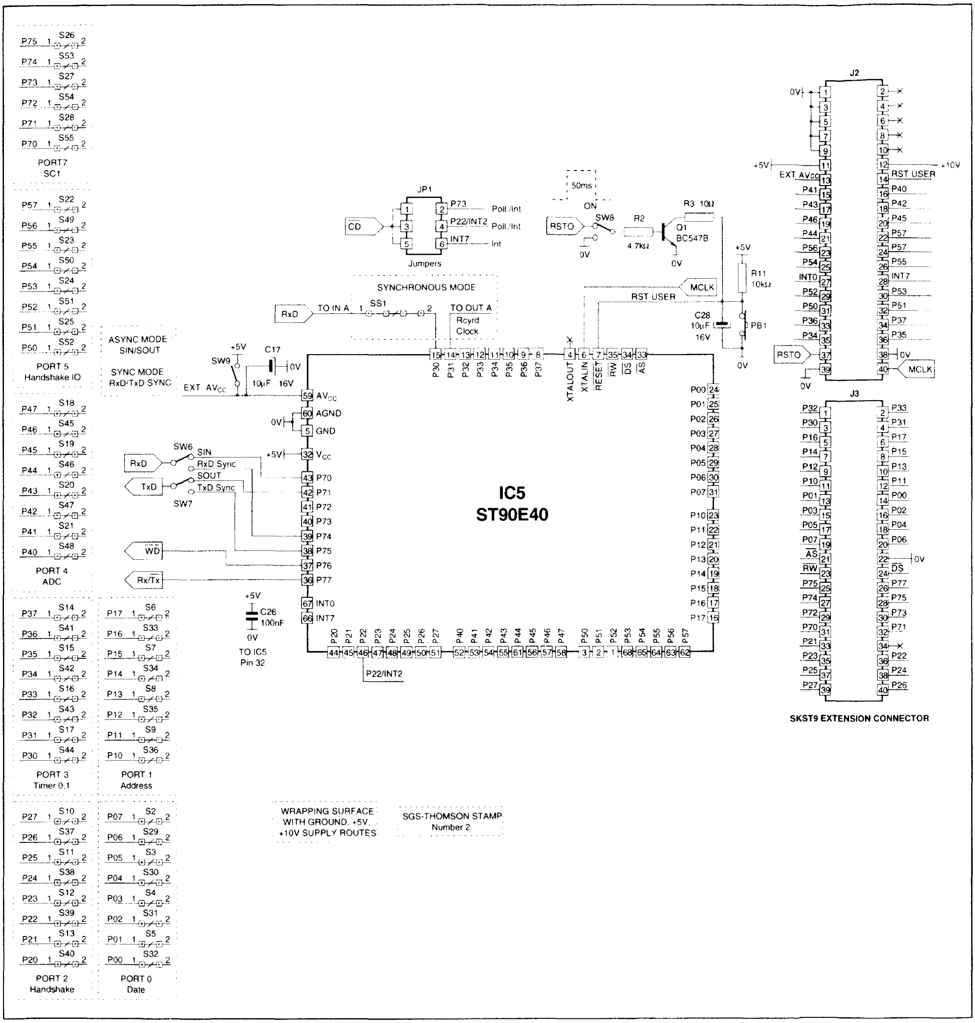
IV - SCHEMATICS

Figure 1



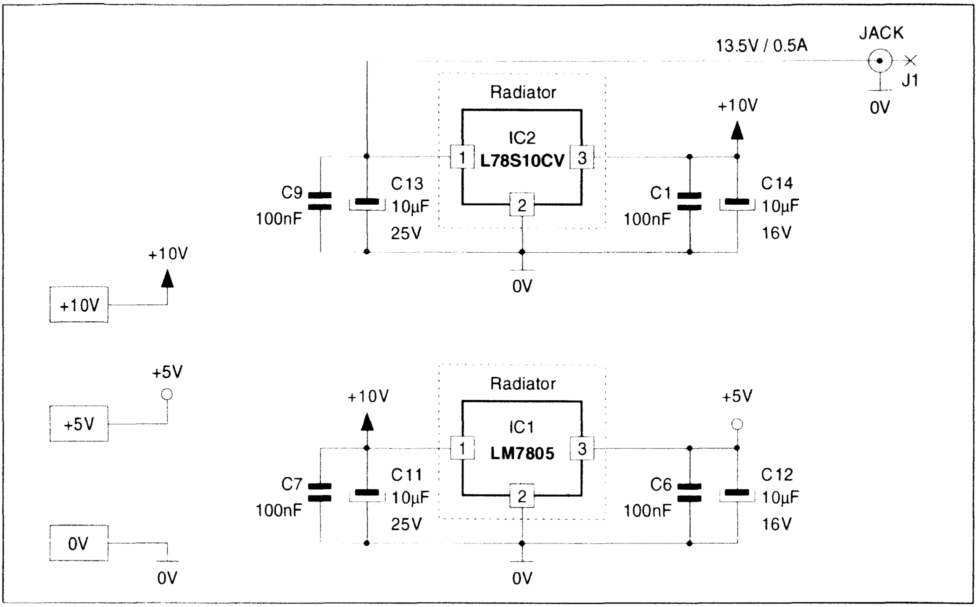
7537-81.EPS

Figure 2



7537-82.EPS

Figure 3



7537-83 EPS

ST9 FAMILY
8/16 BIT MCUs

ST9 APPLICATION TAILORED MCU

The ST9 family of 8/16 bit Microcontrollers (MCUs) was designed after the requirements of the most advanced applications in computer, consumer, telecom, industrial and automotive Segments.

Processed with the same proprietary CMOS EPROM and EEPROM technologies that have established SGS-THOMSON as a world leading supplier of non-volatile memories, the ST9 provides high speed computing with reduced power consumption.

Built around a high performance, register based core, the ST9 family offers different program and data memory sizes and a wide range of on-chip peripherals to meet the needs of most systems.

Time to market is minimized with ST9's well defined, socket compatible, evolution path, from application evaluation with EPROMs, to prototyping using OTPs, up to the high volume production using cost effective ROM versions.

All standard ST9 devices include a Serial Peripheral Interface, a Watchdog Timer to ensure system integrity against externally generated malfunctions, bit configurable I/Os, prioritizable Interrupts for real-time data handling, and DMA for fast data transfers with handshake (HSHK).

In addition ST9 family variants include up to three Multi-Function Timers, two Serial Communication Interfaces (SCI), an Analog to Digital converter (A/D) and On-Screen Display and Data-Slicer for TV control.

REGISTER BASED ARCHITECTURE

The Register based architecture provides more efficient data handling and reduced code size compared to an accumulator based MCU. It also provides the capability for fast context switching.

224 of the 256 8-bit Registers in the ST9 Register File are available as accumulators, index registers, or stack pointers and can be cascaded to perform all these functions as 16-bit registers. The remaining registers are dedicated to system and peripheral control.

This architecture is common to all ST9 devices.

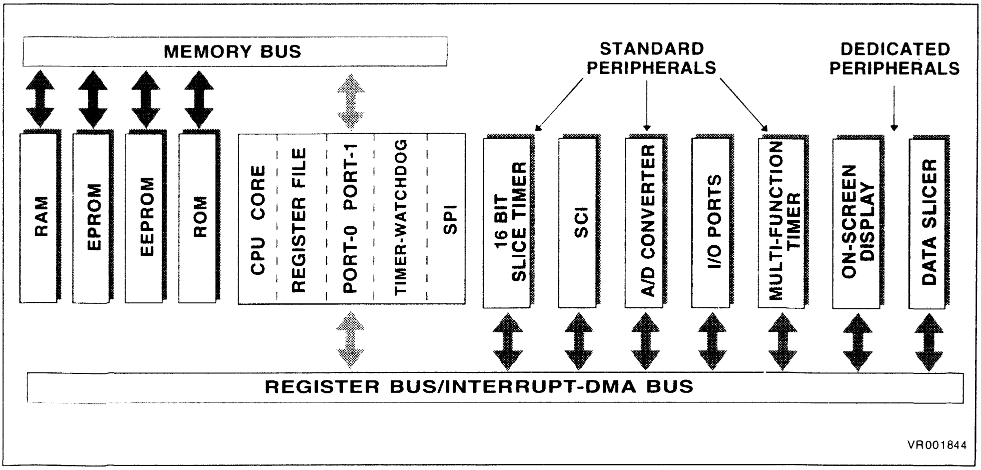
FLEXIBLE I/O

The flexibility of the ST9 I/O pins allow designers to match the MCU to the application, and not the application to the MCU.

Most I/Os can be individually programmed as input (TTL or CMOS thresholds), output (open-drain or push-puil), bidirectional, or as the Alternate Function of a peripheral, such as a Timer or an A/D Converter.

COMPREHENSIVE SCI

Serial communication is easily implemented, using formats and facilities offered by the ST9 Serial Communication Interface.



ST9 Architectural Block Diagram

This peripheral provide full flexibility in character format (5,6,7,8 databits), odd, even or no parity, address bit, 1, 1.5, or 2 stop bits in asynchronous mode, and an integral baud rate generator allowing communication at up to 370k baud in asynchronous mode or 1.5Mbyte/s in synchronous mode.

Industrial, telecom and communication systems users can furthermore benefit from the self-test and address bit wake-up facility offered by the character search mode.

FAST A/D WITH ANALOG WATCHDOG

Up to 8 analog input voltages can be sequentially converted by the Analog to Digital converter including on-chip sample and hold.

The 11µs conversion time, and the possibility to trigger conversions either by the on-chip timers, or by external sources, allows real time processing of analog data.

CPU loading is also reduced by the analog watchdog on two channels, the peripheral interrupts the ST9 when the analog input voltage moves out of a preset threshold window.

UNIVERSAL SPI

A universal Serial Peripheral Interface, providing basic I^{2}C-bus, Microwire-Bus and S-BUS functionality, allows efficient communication with low-cost external peripherals or serial access memories such as EEPROMs.

MULTI-FUNCTION TIMERS

The 16 bit up/down counter operating in 13 modes gives the ST9 Multi-function Timer the possibility to cover most application timing requirements.

Two input pins, programmable as external clock, gate or trigger, allow 16 modes of operation, including auto-discrimination of the direction of externally generated signals.

Pulse Width Generation can easily be implemented, using the overflow/underflow signal and the two 16 bit comparison registers, each of them able to independently set, reset, toggle or ignore two output bits.

The Multifunction Timer outputs may also generate interrupts for system scheduling, and trigger DMA transactions of a data byte to or from a data table in memory through an I/O Port with handshake.

ON-SCREEN DISPLAY

Interactive information display for television control is easily implemented with the powerful ST9 On-Screen Display. With up to 34 characters in 15 rows, and colour, italic, underline, flash, tranparent and fringe options, the 128 character set can be adapted for all needs.

DATA-SLICER

Closed Caption Data can be easily extracted from the video signal with the ST9 Data Slicer. When used in conjunction with the ST9 On-Screen Display, a powerful TV controller can be achieved with the minimum of components.

POWERFUL INSTRUCTION SET

The ST9 has 14 addressing modes and instructions (including multiply, divide, table search and block move) to cover all data manipulation needs, bit, byte and word, at the speed required by even the most demanding control application.

Its instruction set was conceived to facilitate the software designer's task, and to improve programming efficiency.

FULL DEVELOPMENT SUPPORT

ST9 Development Tools are designed for application development efficiency.

A high level macro assembler (with IF/THEN, DO/WHILE, SYSTEM/CASE, PROCedure C language constructs in the assembler) is available, as well as an incremental linker able to link up to 16 Mbytes of program and data, a library maintainer for archiving common software routines and Software Simulation of the code execution, allowing off-line code development and timing analysis.

The validated ANSI standard C Compiler generates optimised code for the ST9. In addition a GNU C cross-compiler and Linker allows development support under the Microsoft Windows 3^{TM} environment.

Cost effective emulation is provided either through a software module running on standard PCs, or with the ST9 Starter Kit, offering hardware emulation capability.

Full real time hardware emulation is provided by the ST9-HDS system.

ST9 FAMILY OVERVIEW

ST9 FAMILY OVERVIEW

All devices have 256 byte Register File with 224 General Purpose Registers (Accumulators/RAM), TWD and SPI Peripherals.

DEVICE	ROM x8	EPROM OTP ROM[1] x8	RAM x8	EEPROM x8	MFT	SCI	A/D Inputs	BSS	MAX I/O	HSHK
ST9026	16K		256		1	1			40	1
ST9027	16K		256		1	1			32	1
ST9028	16K		256		1	1			36	1
ST90E26		16K	256		1	1			40	1
ST90E27		16K	256		1	1			32	1
ST90E28		16K	256		1	1			36	1
ST90T26		16K[1]	256		1	1			40	1
ST90T27		16K[1]	256		1	1			32	1
ST90T28		16K[1]	256		1	1			36	1
ST90R26	-	-	256		1	1			32	1
ST9030	8K				2	1	8		56	1
ST90E30		8K			2	1	8		56	1
ST90T30		8K[1]			2	1	8		56	1
ST90R30	-	-			2	1	8		40	1
ST9032	12K	-			2	1	8		56	1
ST9036	16K		256		2	1	8		40	1
ST90E36		16K	256						40	
ST90T36		16K[1]	256						40	
ST9040	16K		256	512	2	1	8		56	1
ST90E40		16K	256	512	2	1	8		56	1
ST90T40		16K[1]	256	512	2	1	8		56	1
ST90R40	-	-	256	512	2	1	8		40	1
ST90R50	-	-			3	2	8	1	56	2
ST90R51	-	-			3	2	8	1	54	2
ST9054	32K		1280		3	2	8	1	72	2
ST90E54		32K	1280		3	2	8	1	72	2
ST90R54			1280		3	2	8	1	72	2
ST9292	24K		384				3[2]		41	
ST92E92		24K	384				3[2]		41	
ST92T92		24K[1]	384				3[2]		41	
ST9293	32K		640				4[2]		41	
ST92E93		32K	640				4[2]		41	
ST92T93		32K[1]	640				4[2]		41	

Keys:

TWD	Timer/Watchdog	SCI	Serial Communications Interface
SPI	Serial Peripheral Interface	A/D	8 bit 8 channel A/D Converter
MFT	Multi-Function Timer	BSS	Bankswitch logic 16M byte address range
I/O	In :TLL/CMOS, Out : OD/PP Alternate Functional Peripheral	HSHK	# Ports with Handshake Capability

Notes :

1. OTP ROM = One Time Programmable
2. 6 bit A/D Converter

DEVICE	OTHER FEATURES	PACKAGE (Operating temperature)			PAGE
		DIP	LCC	QFP	
ST9026		P48 (1,6)			9
ST9027		P40 (1,6)			9
ST9028			P44 (1,6)		9
ST90E26		C48W (1)			11
ST90E27		C40W (1)			11
ST90E28			C44W (1)		11
ST90T26		P48 (6)			11
ST90T27		P40 (6)			11
ST90T28			P44 (6)		11
ST90R26		P48 (6)			13
ST9030			P68 (1,6)	P80 (1)	15
ST90E30			C68W (1)	C80W (1)	17
ST90T30			P68 (6)	P80 (1)	17
ST90R30			P68 (6)		19
ST9032			P68 (1,6)	P80 (1)	21
ST9036			P68 (1,6)	P80 (1)	23
ST90E36			C68W (1)	C80W (1)	25
ST90T36			P68 (6)	P80 (1)	25
ST9040			P68 (1,6)	P80 (1)	27
ST90E40			C68W (1)	C80W (1)	29
ST90T40			P68 (6)	P80 (1)	29
ST90R40			P68 (6)		31
ST90R50			P84 (6)		33
ST90R51				P80 (1)	35
ST9054			P84 (1,6)		37
ST90E54			C84W (1)		39
ST90R54			P84 (6)		41
ST9292	OSD, STM, DSL, PWM	PS42 (1)			43
ST92E92		CS42W (1)			45
ST92T92		PS42 (1)			45
ST9293	OSD, STM	PS42 (1)			47
ST92E93		CS42W (1)			49
ST92T93		PS42 (1)			49

Keys :

OSD	On Screen Display	Pxx	Plastic Package
STM	Slice Timer	PSxx	Plastic Shrink DIP Package
DSL	Data Slicer extracting Closed Caption Data	CxxW	Ceramic Package with window
PWM	Pulse Width Modulation outputs	CSxxW	Ceramic Shrink DIP Package with window

Temperature Ranges :

(1)One version available, 0 to +70°C (6)One version available, -40 to+85°C
(1,6)Two versions available, 0 to +70°C and -40 to+85°C

DEVELOPMENT TOOLS

ST9 STARTER KIT

EVALUATION KIT FOR ST9 MCU FAMILY

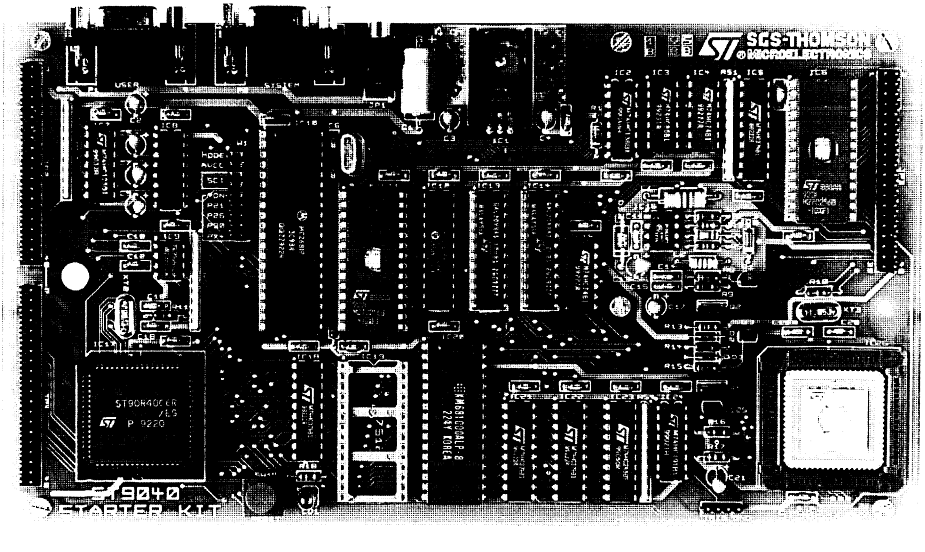
- Full Evaluation Kit for ST9 Family
- Emulation Capability
- Windowed and Command Line interfaces

GENERAL DESCRIPTION

The ST9 Starter Kit includes all the hardware, software and documentation required to evaluate the ST9 family of 8/16-bit MCUs and to develop simple applications. The ST9 Starter Kit includes ROMless (ST90R40) and EPROM-based (ST90E40) microcontrollers, the ST9 family documentation, the ST9 software tools package and an evaluation board for debugging an application and programming the ST90E40.

The board, which measures 225 x 125mm, is based on the ST90R40, which offers all the most important features of the ST9 family, including a built-in DMA controller, a Serial Peripheral Interface (supporting S-bus, I^{2}C-bus and IM-bus), 256 bytes of internal RAM, 512 bytes of internal EEPROM, 16-bit multifunction timers, A/D converter and a full duplex serial communication interface. On-board memories provide storage for emulated programs and data, the monitor program and breakpoint information.

For maximum flexibility, the board can run in three different modes. In the stand-alone mode, up to 64Kbytes of program space and 64Kbytes of data space are available. In the emulation mode, the board is driven by a monitor program allowing the use of all registers and memories, while single step, software trace and breakpoints are supported on both program and data spaces through debugging software running on a PC host. The third mode is the programming mode, which allows debugged software to be downloaded to an ST90E40 using the ZIF socket provided.



SOFTWARE TOOLS

The software tools include a full macro-assembler which supports modular programming, an incremental linker, an archiver that manages relocatable objects modules, a functional simulator, a windowed debugger which drives the ST9 evaluation board and the EPROM programming software. The fully symbolic debugger allows access to all ST9 resources (memories, registers) while the windowed and menu-driven interface, on-line help and intuitive access to commands make it very easy to use.

The ST9 Starter Kit also includes full documentation on the ST9040 Family, on how to connect and program it and the software tools manuals which describe how to use the ST9 development tools included in the starter kit, as well as a floppy disk containing several application programs for ST9 devices.

Starter Kit Command Line Summary

COMMAND	DESCRIPTION
ARchive	Archive symbols and macros
ASm	On-line Assembler
BAse	Change base of numbers
BYE	Exit from debugger program
CLOSE	Close I/O Channel
CLS	Clear screen
CM	Compare memory
DEfine	Define symbols and macros
DIsasm	On-Line disassembler
DM	Display memory
DO	Execute macro
DR	Display register
DUmp	Save current setup
ENDFOR	End for loop (see FOR)
ENDIF	End conditional block (see IF)
FM	Fill memory with pattern
FOR	Loop command execution
FR	Fill registers with pattern
GO	Execute program
Help	On-Line help
IF	Conditional command execution
JUMP	Go to label
LIstsymbol	List symbols and macros
LOad	Load program from file
LOCATE	Set cursor position at given coords
MAP	Set/Display mapping
MB	Modify memory breakpoints settings
MM	Move memory block
NEXT	Execute program steps

Starter Kit Command Line Summary (Continued)

OPEN	Open I/O channel
PAUSE	Pause for number of seconds
Print	Print strings and values
Quit	Return to Graphical Interface
REset	Reset emulated CPU
SAve	Save memory contents to file
SB	Set/Display memory breakpoints
SEarch	Search for pattern in memory
SET	Set/Reset emulator options
SM	Set memory
SR	Set/Display registers
SYstem	Exit temporarily to operating system
Trace	Display trace
UNdefine	Remove symbols
USE	Execute command file
VER	Print version information
WAtch	Display/Create watch data
WR	Display current working register set
<value>	Evaluate expression
!	Execute single system command
?	Display symbols having a given value
:	Comment line for macros/command files

SGS-THOMSON MICROELECTRONICS

ST9-GNU TOOLCHAIN

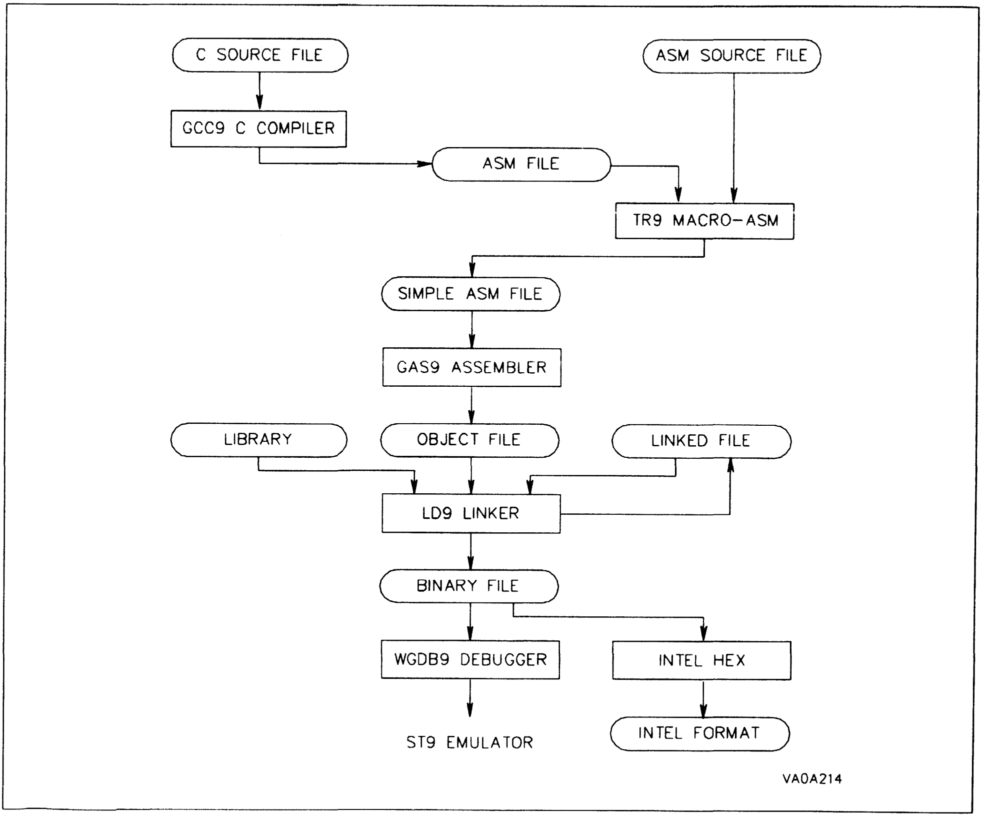
C COMPILER, ASSEMBLER, LINKER AND SOURCE DEBUGGER FOR ST9 MCU FAMILY

CONTENTS

- Optimised C compiler with options for different standards: traditional C, ANSI C, and GNU extensions
- Macro-assembler with powerful pre-processor
- Linker/loader
- Source-level debugger with Microsoft WINDOWS 3™ graphic interface
- Available for SUN 4 (SPARC STATION) under the UNIX system

GENERAL DESCRIPTION

The GNU Toolchain offers the software developer a full set of resources for the development of code for the ST9 microcontroller. This is achieved through the optimised GNU C Compiler, the Macro-assembler, Linker/Loader and Library Archiver. The Assembler is fully compatible at source level with previous version of the ST9 Assembler. Program debugging is made easier with the C Language Source Level Debugger, which runs under MSDOS or Microsoft Windows 3™.



GNU C COMPILER

- All standard types allowed (char, int, short, long, signed or unsigned, float, and double) with Float types respecting the IEEE 754 standard
- Libraries delivered include string handling, conversion, I/O routines and mathematics
- Direct access to the Register File of the ST9, allowing access to all registers and on-chip peripherals
- Allows inclusion of assembly language instructions, with access to C program symbols
- Options to generate code for one or two memory spaces, one or two stacks and interrupt routines
- Optimisation phase included at final stage

General Description

The GNU C Compiler for the ST9 allows the programmer to write C source code (using traditional C (Kernigan & Richie), ANSI C, or GNU Extensions) and to produce assembly language source code. When used with the Assembler and Linker, it allows the generation of executable object code for all members of the ST9 family.

The generated assembly source file may include interleaved C lines and assembly language lines, and provides information for source-level debugging.

ASSEMBLER

The Assembler pre-processor allows macro substitution, file inclusion and conditional assembly and is fully compatible at source level with the ST9 assembler (AST9) pseudo-instructions and pseudo-macros.

Source level debugging information is generated with the object file by the assembler.

Assembly language programs are fully mixable with C language programs and accept 3 sections (TEXT, DATA, BSS).

LINKER

- Combines object code files issued by the assembler
- Supports incremental linking

General Description

The Linker resolves references to external symbols and searches libraries for necessary modules to produce an output file in a binary format, downloadable by the debugger to the ST9 emulator.

A map file is generated, including all mapping information on sections, files, and symbols and separate files are produced to support ST9 bank switch mechanism

The three sections generated by the C compiler and used by the assembler are accepted.

Options are available for setting the base addresses of sections and stacks and management of two spaces with script files to define memory mapping.

DEBUGGER

- Runs on MS-DOS based PC, with or without WINDOWS 3™
- Connected by serial line to the ST9 hardware emulator
- Offers a WINDOWS 3™ - based graphic interface, supporting all standard features
- Mouse supports access to context sensitive help
- Offers a line mode command interface (able to run on MS-DOS or within a WINDOWS 3™ DOS box) supporting command files
- Includes a window for access to the low-level SDBST9 debugger
- Dumps ST9 memories, system registers, Register File and paged registers

General Description

The ST9 Debugger allows source level debug for C language and assembly language programs, even with optimized C language programs.

The debugger is able to generate trace information, with hardware information interleaved with source lines, and to display the local symbols of the current C procedure and the stack based on the C language source level.

Source lines are displayed, with or without disassembly of memory interleaved with the source lines with symbols under their real types.

Requires 386 class PC or higher with at least 4 Mbytes of memory, under MS-DOS 4.01 or higher.

UTILITIES

- Archiver
- Formatter of INTEL HEX industrial format, allowing download of program to an EPROM programmer
- Binary file deformatter

Note: Windows 3 is a trademark of the Microsoft Corporation.

SGS-THOMSON MICROELECTRONICS

APPLICATION NOTE

SYNCHRONOUS POWER LINE MODEM COMMUNICATION WITH ST9 MULTIFUNCTION TIMER

Required tools : ST9/ST7537 PLM Starter Kit, By O. GARREAU

INTRODUCTION

This application note provides an example of an ST9 MFT application handling a Home Automation synchronous protocol. It presents a way to easily communicate on a synchronous Network. Each node of this network may consist of the Power Line Modem (PLM) Starter Kit or of the Home Service (HS) macro-component, provided that it includes the ST9 plus ST7537 modem chipset.
This PLM Starter Kit helps also in developing the ST9 version of 'HOME SERVICE' European Home Automation Protocol.

The ST7537 modem may work in both synchronous and asynchronous modes at a Baud rate of 1200. There is no problem in building an asynchronous interface with the ST9 due to the capabilities of its SCI (Serial Communication Interface). The asynchronous protocol may be programmed directly and all work is done by the SCI. However the synchronous protocol has different requirements.

In order to program a synchronous protocol, it is not possible to use the SCI lines that are reserved for asynchronous communications. The solution resides in using the Multifunction Timer of the ST9.

The most simplified synchronous protocol may consist in a simple 3-wire link (The Receive Data line, RxD, the Transmit Data line, TxD and the signal Ground).

Similar to the asynchronous mode, no clock signal is available in the PLM synchronous mode and the time reference is included in the RxD signal. It is clear that the clock frequency should be known and determined in advance by both emitter and receiver. For our application working as the LAYER 0 of the 'HS' Protocol, this frequency is 1200Hz and generates thus a 1200 baud rate. Note that the signals concerned (RxD and TxD) are 'NRZ, Non Return to Zero' signals.

The first task that the following procedure performs is 'Transmission BitClock Recovery'. Here the Multifunction Timer T0 of the ST9 is used in the background. This basic clock signal is also output on a timer pin (T0OUTA).

The procedure that tests the link between distant modems, consists in sending from the master modem a standard frame, compatible with the 'HS' protocol or not, and in sending back from the slave modem the received frame in order to make a match test, for example, in the master system (which is, in our case, controlled by a Windows 3.1(TM) program).

This method can even validate asynchronous modes. The test frame is stored in the internal memory of the ST9 (in the first 128 bytes of the Register File).

In addition to the RxD and TxD lines, it is possible to improve protocol management with extra lines called RSTO, CDn, RXTXn and WDn. These lines mean respectively RESET modem controller, CARRIER DETECT, RECEIVE OR TRANSMIT, and WATCHDOG signal. 'n' means that the signal is active LOW.

These lines are specific to the ST7537 modem as it is working in its HALF-DUPLEX mode. The direction of data flow is chosen by RXTXn state. CDn indicates that the modem is about to receive data. The WATCHDOG signal is used by the modem to check the presence and activity of the CPU (ST9) and if activity (meaning a minimum of one negative transition per second) is absent, the modem will try to reset the CPU.

To prevent this shut down, WDn may be connected to the recovered BitClock (1200 transitions per second) or to an IO port (P76 in this case).

Note : To fully understand this Note, it is recommended that the reader refers to the relevant chapters of the ST9 databook for the Multifunction Timers.

I - HARDWARE LINK BETWEEN ST9 AND ST7537

Figure 1 shows the schematic of this link. As can be seen, no additional hardware or components are needed. This interface needs only 5 point-to-point wires plus the signal ground. It is the lowest cost way to connect a ST7537 to an ST9.

All wires are TTL compatible and mono-directional. Only one 8-bit port 7 of the ST9 is required, Port 3.0 (T0INA) is also reserved for this application, signal on T0OUTA (P3.1) is the recovered clock, event trigger and control the timer, P7.3 P7.4 P7.5 P7.6 P7.7 are used as simple TTL I/O bits.

The RxD line is managed by a technique which attaches a double function to this single signal. RxD triggers the timer and is acting as a synchroniser. Its level indicates also the input bit state. The kernel of this application is a sophisticated software PLL.

P7.3 is programmed as an input and reads Carrier Detect signal (CDn).
P7.4 is programmed as an input and reads the current input bit state (RxD).
P7.5 is programmed as an output and sends the current output bit (TxD).
P7.6 is programmed as an output and activates the WatchDog signal (WD).
P7.7 is programmed as an output and chooses the direction of data flow (RXTXn).

The corresponding timing chart of these signals is given in the next section. Note that it is not necessary to connect ST7537 signals like DVCC (Digital Output Supply Voltage), RSTO (Reset Output) and MCLK (Master Clock) when this application is running on the ST9 STARTER KIT. The signal ground is of course essential and common to all these lines. When the ST7537 Starter Kit is in stand-alone mode, the ST9 uses RSTO and MCLK

II - TIMING CHARTS OF CONTROL AND DATA SIGNALS

Figure 2 shows the timing and event charts considering an example of 3 bits input or output or both. The first chart displays Carrier Detect signal, which enables the start of transfer.

The second timing shows the input signal used as trigger and data signal. Each '0' or '1' pulse lasts 833.3 microSeconds. The third line is a relative time axis for the internal counter T0. The next axis is the 'Event' axis. The next chart displays the output waveform for the recovered BitClock; this preferably should have a 50% cyclic ratio.

The last chart is an example of the TxD output signal.

Figure 1 : Direct Connection in the Stand-alone Mode

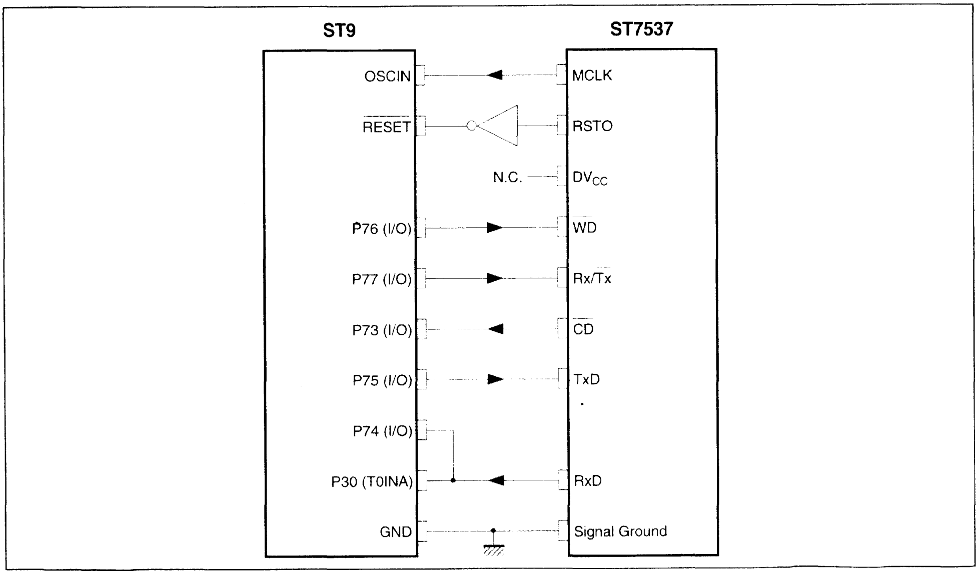
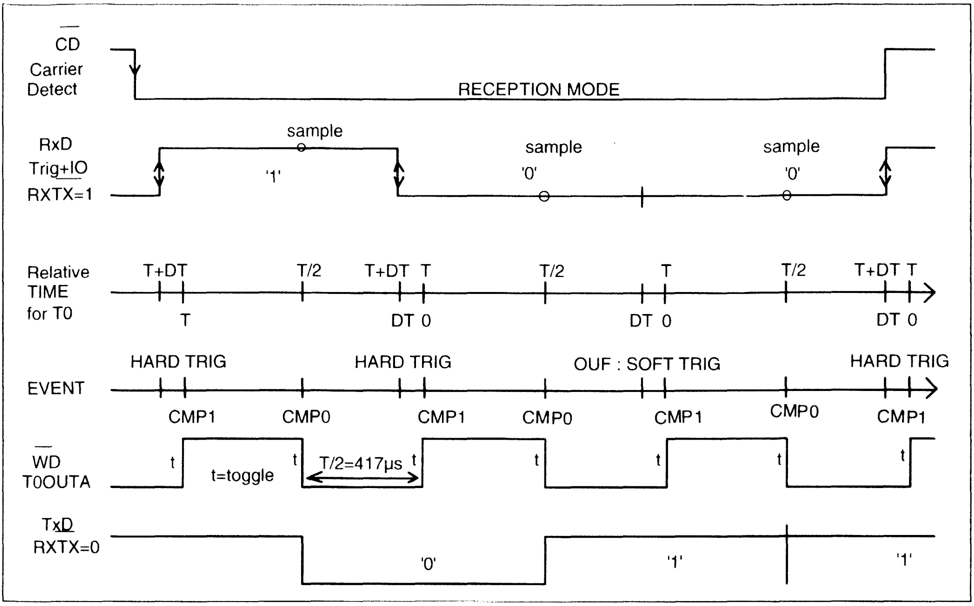


Figure 2 : Timing Chart of Signal and Data Signals



AN430-02.EPS

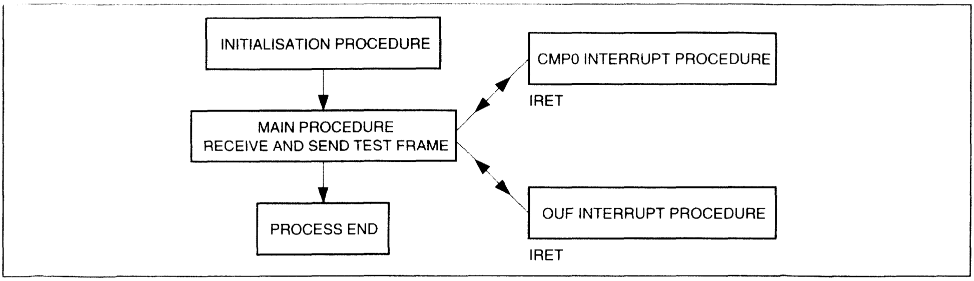
All the powerful features of a ST9 timer are used in this application. These include Hardware Trigger (HT), Software Trigger (ST), Compare (CMP0, CMP1) and Over/UnderFlow (OUF) Events. Both HT and ST events reload TIMER0 from the REG0R Load register. The HT event is automatically generated by every low to high or high to low transition on the RxD line.

The CMP1 event is used to rebuild the initial clock frequency (of the emitter). The CMP0 event samples the input signal on RxD or outputs the transmitted signal on TxD or both.

The OUF event (over-under/flow) allows the counter to be reloaded by an ST action, this acts as a WatchDog, controlled under software and corrects (synchronises) for frequency shift, frequency fluctuations and phase shift.

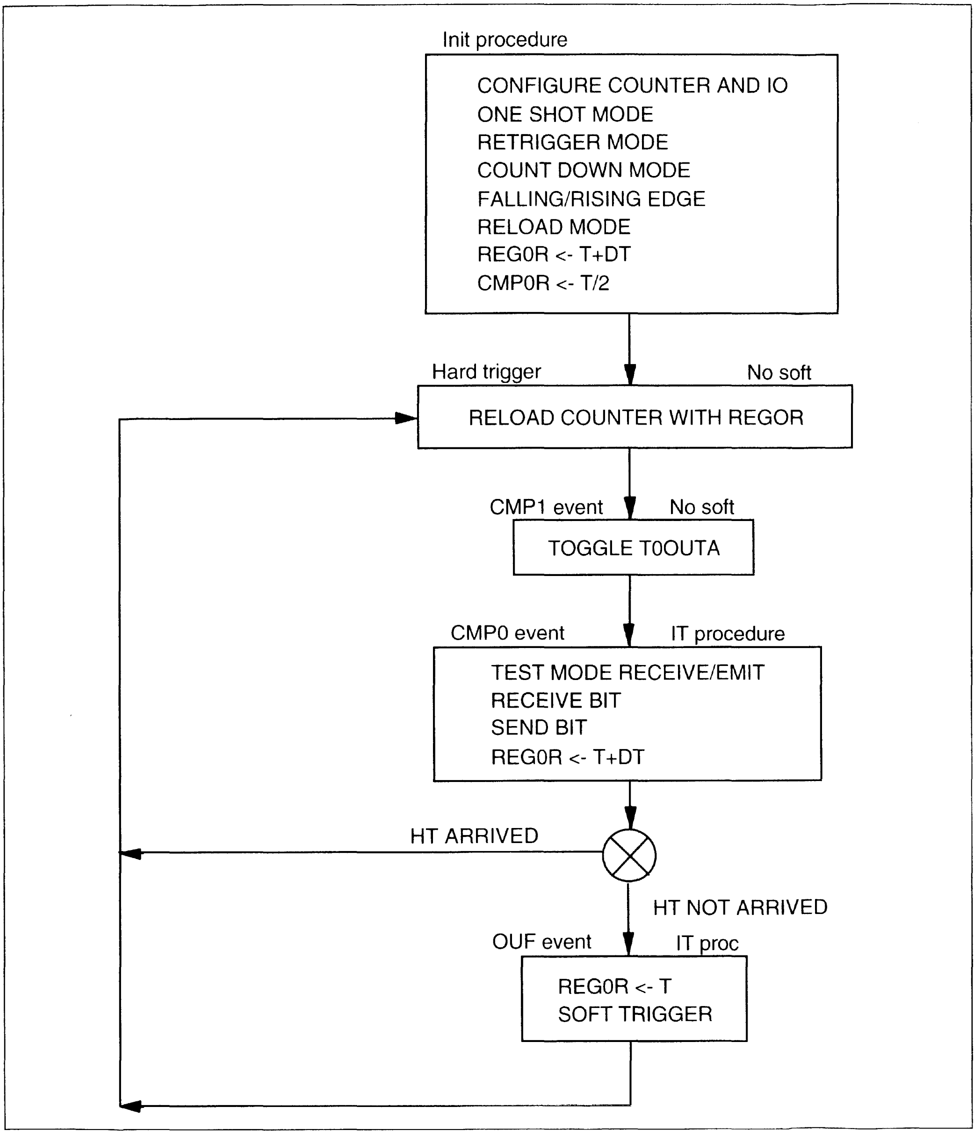
During the time gap (called DT), these parameters may vary and can be taken into account, if their variation is not too wide. A software parameter controls this correction : 'variation' represents a percentage of the whole bit pulse period, called T.

Figure 3 : Software Structure



AN430-03.EPS

Figure 4 : Procedure Algorithm



III - SUMMARY

The aim of this application note is to demonstrate to Hardware designers just how simple it is to interface an ST7537 with a member of the ST9 family. The schematic described in part one is an example of the electrical link, however it also depends on a good quality and conformity of input signals.

First, it is obvious that the RxD signal should be perfectly stable and defined during the acquisition phase. Then, its working frequency (1200Hz) should be reasonably stable and the stability should not be worse than 90%. With practical testing, it is shown that the application works correctly between 1150Hz and 1250Hz. All these characteristics are software programmable.

This routine has been tested at various transmission rates, from 300Hz up to 9600Hz. The software defines constants called 'Rxxxxbds' that correspond to these rates. That way, this application software will stay compatible with new generations of multi-speed ST7537 modem.

Warning : the user must RECALCULATE these constants if the ST9 has an external clock (crystal) DIFFERENT to the frequency used in this example.

The nominal working frequency should be 1200Hz or 2400Hz in the case of the 'HS' protocol.

In this protocol, data packets are composed of a particular frame. Each frame starts with a specific header. Typically the beginning of such a header is 4 bytes: generally FFh, FFh, AAh and AAh. Many bits of the first FFh may be lost but the synchronisation is made actually on the AAh bytes. It is thus possible to improve the accuracy of the process. The byte AAh is a succession of bits at '1' and '0' and can allow frequency optimisation. Using first the timer 0 in its 'Capture' mode will define the precise distant frequency, a pre-calibration.

This software improvement may be useful in case of a distant frequency very far from the expected working frequency (1200Hz in this case).

The user is free to modify these routines. The bits of Port 7 (P73, P74, P75, P76 and P77) are used here as I/O bits. It is clear that another port may be defined as I/O port in order to free access to PORT 7.

Finally, the user can test a link between two distant nodes, or on a bigger network, and use equally synchronous or asynchronous protocol .

As a conclusion, the ST9 microcontroller is a cost effective solution for home automation modem applications and a synchronous protocol like 'HS' or other customer protocol can be performed in the background, requiring little CPU time.

IV - APPENDIX 1 : ST9 SOFTWARE LISTING

```
;__________________________________________________________________________
;                          MODEM+ST9 APPLICATION NOTE
;             FILE: HS.ST9(compile with AST9)
;             First version: 26/05/93
;             Last revision: 13/10/93
;             Author  : Olivier Garreau
;             Department PPG, Section ST9 SUPPORT/APPLICATION (Grenoble)
;             Running on ST7537 STARTER KIT, here in stand-alone 11MHz
;             ST7537 starter kit version : MB076b
;             ST7537 version  : V 3.0
;__________________________________________________________________________

;*************************************************************************
;       goal     : SYNCHRONOUS BIT CLOCK RECOVERY, I/O BUFFERS MANAGER
;                  Validation software for a serial link between two
;                  based on ST 7537 modems, Full Duplex capability
;                  one modem driver is a PC, second modem driver is the ST9
;
;       example of application   : 'HOME SERVICE' AUTOMATION PROTOCOL
;       modem used       : ST7537 (always working in half duplex mode)
;       input signals    : RxD connected to T0INA(P30) AND P74
;                          CDn (carrier detect) connected to P73 (by jumper)
;       output signals    : Bit clock recovered on T0OUTA (P31).
;                          P76 output on modem WatchDog WDn.
;                          RXTXn (select direction) connected to P77
;                          TxD connected to P75
;       modified         : Input Bit buffer permanently filled
;                          Output Bit Buffer permanently sent
;
;       use underflow, compare0 and compare1 event interrupts
;       -OUF interrupt reload the timer
;       -COMPARE0 interrupt sample data on RxD line and update Bit Buffer
;                 and manage TxD output signal
;       -COMPARE1 generate output recovered clock (no procedure)
;       Process running in background:
;             - read a synchronous frame (max 128*8 bits)
;             - write it back to the emitter
;             - then stand by to check watchdog circuit (wait for reset)
;*************************************************************************


;****************************
;* Include file definition *
;****************************
.include        "..\\..\\include\st904x.inc"
```

```
; REGISTERS DEFINITION
;*********************

data            =       r0      ; data.b2 contains binary information
status          =       r1      ; hold flag of timer
ptr             =       r2      ; read buffer pointer
num_bit_r       =       r3      ; bit number in input byte
ptw             =       r4      ; output buffer pointer
num_bit_w       =       r5      ; bit number in output byte
mode            =       r6      ; mode selection(input,ouput or full duplex)
tempol          =       r7      ; 8 bits register
tempo           =       rr8     ; 16 bits register
sys_stack       =       0CFh    ; system stack
end_buf         =       07Fh    ; common end of input and output buffer
start_buf       =       000h    ; start of both buffers
;*********************
; CONSTANTS DEFINITION
;*********************

pageF           =       (0Fh*2)
page8           =       (08h*2)

;time_step      =       250     ; minimum time step is 250 nanoseconds
;period         =       833333  ; 833333 nanoseconds corresponds to 1200 Hz
;T              =       period/time_step        ; period for timer
;DT             =       T/variation             ; Delta T for timer

                                ; different clock rate
R9600bds        =       416     ; 9600 baud
R4800bds        =       833     ; 4800 baud
R2400bds        =       1666    ; 2400 baud
R1200bds        =       3072    ; 1200 baud modified for 11.0592 Mhz
                                ; should be 3333 with 24 Mhz crystal (/2)
R600bds         =       6666    ; 600 baud
R300bds         =       13333   ; 300 baud

.defstr mode_transmit   "r6.0"  ; flag for transmission
.defstr mode_receive    "r6.1"  ; flag for reception

RxD             =       4         ; position of RxD bit
TxD             =       00100000b ; TxD bit
CD              =       3         ; position of Carrier Detect bit
RXTX            =       10000000b ; RXTX bit
WATCHDOG        =       01000000b ; WDn signal
```

```
T               =       R1200bds; 'Home Service' clock frequency
variation       =       10      ; +-variation around correct signal edge
DT              =       T/variation     ; allow +-10% variation
                        ; the effective receive frequency is 1200.12 Hz

t0_vect         =       010h    ; Start of Timer 0 vector table.

;******************
; MACRO DEFINITIONS
;******************

.macro t0start                          ; start counter
        begin [PPR] {
        spp #T0D_PG
        or  T_IDMR,#gtien               ; T0 Global interrupt mask disabled.
        or  T_TCR,#cen                  ; Counter enabled.
        }
.endm

.macro t0stop                           ; stop counter
        begin [PPR] {
        spp #T0D_PG
        and T_TCR,#~cen                 ; Counter disabled.
        and T_IDMR,#~gtien              ; T0 Global interrupt mask enabled.
        }
.endm

.macro  set_mode_transmit               ; set transmit flag and line RXTX
        bset    mode_transmit           ; set flag
        begin [PPR] {                   ; save PPR
        spp     #P7D_PG                 ; set port7 data page
        and     P7DR,#~RXTX             ; select transmit mode of modem
        }
.endm

.macro  reset_mode_transmit             ; reset transmit flag and line RXTX
        bres    mode_transmit           ; reset flag
        begin [PPR] {                   ; save PPR
        spp     #P7D_PG                 ; set port7 data page
        or      P7DR,#RXTX              ; select receive mode of modem
        }
.endm
```

```
.macro  set_mode_receive                ; set receive flag and line RXTX
        bset    mode_receive            ; set flag
        begin [PPR] {                   ; save PPR
        spp     #P7D_PG                 ; set port7 data page
        or      P7DR,#RXTX              ; select receive mode of modem
        }
.endm

.macro  reset_mode_receive              ; reset receive flag and line RXTX
        bres    mode_receive            ; reset flag
        begin [PPR] {                   ; save PPR
        spp     #P7D_PG                 ; set port7 data page
        or      P7DR,#RXTX              ; select receive mode of modem
        }
.endm

.macro  wait_CD_high    ?loop_wait      ; wait for "1 to 0" transition on CD
        begin [PPR] {
        spp     #P7D_PG
        loop_wait:
        ld      data,P7DR               ; read input CD bit (data.b4)
        btjt    data.CD,loop_wait       ; wait for low level
        }
.endm

.macro  wait_CD_low     ?loop_wait      ; wait for "0 to 1" transition on CD
        begin [PPR] {
        spp     #P7D_PG
        loop_wait:
        ld      data,P7DR               ; read input CD bit (data.b4)
        btjf    data.CD,loop_wait       ; wait for high level
        }
.endm

.macro  do_tempo
        clr     tempo1                   ; first loop
        ldw     tempo,#3000              ; tempo value
        loopw   [tempo] {
                loop    [tempo1] {
                        }
        }
.endm
```

```
;******************
; INTERRUPT VECTORS
;******************

power_on::

.word   main                    ; RESET vector.
.word   main                    ; Divide by 0 vector not used
.word   main                    ; no top level interrupt

.org    t0_vect                 ; table of timer interrupt vectors
.word   ouf_proc                ; vector of ouf interrupt
.word   error_proc              ; not a vector
.word   error_proc              ; no capture procedure, but event used
.word   comp_proc               ; vector of compare0 interrupt

;**************
; STOP on error
;**************

error_proc::
        halt                    ; stops if capture event

;*****************
; Routine COMPARE0
;*****************

comp_proc::                     ; middle of bit pulse

        begin   [PPR,RP0R,RP1R] { ; save PPR and RPP

        spp     #P7D_PG         ; set page for port7
        xor     P7DR,#WATCHDOG  ; signal ST9 activity
        srp     #page8          ; select working register, bank 8
        btjf    mode_receive,no_receive ; mode receive not selected
        ld      data,P7DR       ; read input RxD bit (data.b4)
        cpl     num_bit_r       ; prepare a reset mask
        and     (ptr),num_bit_r ; reset read bit in buffer (default case)
        cpl     num_bit_r       ; restore initial value
        btjf    data.RxD,bit_zero ; jump if RxD = 0
        or      (ptr),num_bit_r ; put read bit in buffer
bit_zero::
        ror     num_bit_r       ; shift from bn to bn-1
        adc     ptr,#0          ; increment pointer if end of byte
        and     ptr,#end_buf    ; prevent from overflow of input buffer
no_receive::
        btjf    mode_transmit,no_transmit ;mode transmit not selected
        tm      (ptw),num_bit_w ; test bit to send
        jrne    bit_high        ; jump if bit = "1"
        and     P7DR,#~TxD      ; clear TxD line
        jr      bit_ok          ; bit state is OK
bit_high::
        or      P7DR,#TxD       ; set TxD line
bit_ok::
        ror     num_bit_w       ; shift from bn to bn-1
        adc     ptw,#0          ; test end of byte
        and     ptw,#end_buf    ; force buffer end
no_transmit::
        spp     #T0D_PG         ; page data for timer 0
        ldw     T_REG0R,#T+DT   ; load T+DT into REG0R
                                ; adjust timing : delay of DT
        ld      status,T_FLAGR  ; status flag, should read compare pending FL
        clr     T_FLAGR         ; reset pending bits
        }
        iret

;************
; Routine OUF
;************

ouf_proc::                      ; hardware watchdog -> no transition on RxD

        begin   [PPR,RP0R,RP1R] { ; save PPR and RPP

        spp     #T0D_PG         ; page data for timer 0
        ldw     T_REG0R,#T      ; load (T+DT)-DT into REG0R
                                ; adjust timing : advance of DT(synchronise)
        or      T_FLAGR,#cp0    ; launch counter with advanced value
        or      status,T_FLAGR  ; save pending bit (ouf pending bit)
        clr     T_FLAGR         ; reset pending bits
        }
        iret
```

```
;********************
; Routine periph_init
;********************

proc    periph_init     [PPR,RP0R,RP1R]       {

;************ init T0OUTA and T0INB ****************************************

        spp #P3C_PG             ; direction of signals     ST9 <==> ST7537
                                ; b0=T0INA=P3.0 : IN,TRI,CMOS   <--- RxD
        ld P3C2R,#00000000b     ; b1=T0OUTA=P3.1: AF,PP          ---> Clock
        ld P3C1R,#00000010b     ; b2-b7:unused
        ld P3C0R,#00000011b     ;


        spp #P7C_PG

        ld P7C2R,#00000000b     ; b0-b3         : free for SCI use
        ld P7C1R,#11100000b     ; b6=P7.6       : OUT,PP I/O    ---> WDn
        ld P7C0R,#00011000b     ; b7=P7.7       : OUT,PP I/O    ---> RXTX
                                ; b4=P7.4       : IN,TRI CMOS   <--- RxD
                                ; b5=P7.5       : OUT,PP I/O    ---> TxD
                                ; b3=P7.3       : IN,TRI CMOS   <--- CDn


;************ init timer modes ********************************************

        spp #T0D_PG             ; T0 data page
        srp #pageF              ; To access bank F with "r" addressing mode.
        ldw t_reg0r,#T+DT       ; load REG0R with period + variation
        ldw t_cmp0r,#T/2        ; center to the middle of the pulse
        ldw t_cmp1r,#T          ; initiate (set) clock pulse
        ld  t_tcr,#0            ; Disable counter
                                ; Count down
        ld  t_tmr,#oe0|co       ; T0OUTB disable as timer output
                                ; T0OUTA enable
                                ; REG0R reload
                                ; No ECK clock
                                ; Retrigger mode
                                ; One shot mode
        ld t_icr,#ab_ti|exa_rf  ; T0INA trigger, T0INB I/O
                                ; T0INB nop, T0INA rising+falling edge
        ld t_prsr,#0            ; No prescaling
        ld t_oacr,#c0_res|c1_set|ou_nop ; CMP0R used to reset clock signal
                                ; CMP1R used to set clock signal
                                ; OUF not used
                                ; EOC not used
                                ; preload with '0'
        ld t_idmr,#gtien|cm0i|oui ; enable interrupt
                                ; enable compare0 interrupt
                                ; enable OUF interrupt
                                ; disable compare1 interrupt

        spp #T0C_PG

        ld t0_ivr,#t0_vect      ; pointer into vector table
        ld t0_idcr,#0C6h        ; priority level 6

        }

;*************
; MAIN PROGRAM
;*************

main::

        ld MODER,#11000000b     ; Ext clock NOT prescaled by 2.
                                ; Internal system and user stacks.
        ldw SSPR,#sys_stack     ; System stack pointer.

        spp #WDT_PG             ; select watchdog page
        ld  WCR,#wdgen          ; Watch dog mode disabled, no wait states.
        ld  EIMR,#0             ; Mask all channels interrupts.

;**********************  initialise I/O and timer  **********************

        call periph_init        ; Initialize timer and port

;**********************  init buffers and pointers  *********************

        srp     #page8
        ld      ptr,#start_buf  ; set read pointer to start
        ld      num_bit_r,#080h ; first bit to read : MSB
        ld      ptw,#start_buf  ; set write pointer to start
        ld      num_bit_w,#080h ; first bit to output : MSB

        clr     mode            ; reset both modes
```

```
;***********************   process itself  ********************************

        ei                       ; enable interrupts
        t0start                  ; start counter (down)

        ;***************   Read a frame   ********************************

        wait_CD_high             ; wait for the modem to receive
        set_mode_receive         ; activate RXTX line and fill input buffer
        wait_CD_low              ; wait for end of incoming data
        reset_mode_receive       ; stop receive mode
                                 ; got the frame

        ;***************   Wait for a while   ****************************

        do_tempo                 ; do a tempo

        ;***************   Emit the received frame   *********************

        set_mode_transmit        ; clear RXTX line and use output buffer

        while   [ ptw != ptr ] {
                                 ; last input byte pointed by ptr
                                 ; wait for the output buffer to be sent
        }                        ; stop at the last input byte

        while   [ num_bit_w != num_bit_r ] {
                                 ; position of last bit pointed by num_bit_r
                                 ; wait for the output buffer to be sent
        }                        ; stop at the last input bit (+1)

        reset_mode_transmit      ; stop emission mode

        di                       ; stop WatchDog to ST7537
        halt                     ; wait for reset

;***********************   end of file   ************************
```

V - APPENDIX 2 : PROCEDURE UNDER WINDOWS 3.1™

```
/*
;________________________________________________________________
;|
;|Routine Name   : PROCTEST
;|File Name      : PROCTEST.H(compile with MS QuickC)
;|Action         : is the Header of proctest.c file
;|Author(FN/LN)  : Olivier Garreau / PPG / GRENOBLE
;|First rev date: 06/10/93
;|Last rev date : 13/10/93
;|Input paramet.: none
;|Output paramet: none
;|Modified var. : none
;|Global varia. : none
;|Comments       : define the constants of test procedure
;|      to be defined in a .MAK file
;|_______________________________________________________________
*/


#define delay_RXTX_start6// wait 6 millisecond before first byte
#define delay_RXTX_end6// wait 6 millisecond after last byte
#define reset_delay2000// wait 2 seconds for Board2 ST9 reset
#define proper_buffer800// wait 0.8 second to flush input buffer
#define Watchdog3000// wait 3 seconds for ST9 answer

#define number_of_test10// perform 10 test loops MAX

#define size_buffer_in1024
#define size_buffer_out1024


#define test_string"This is a string to verify the links !"
#define length_stringsizeof (test_string)


voidinit_com_port(void);
voidsend_frame(void);
voidreceive_frame(void);
intmatch_frames (void);
voiddisplay_result (int);
voidwait (DWORD);
```

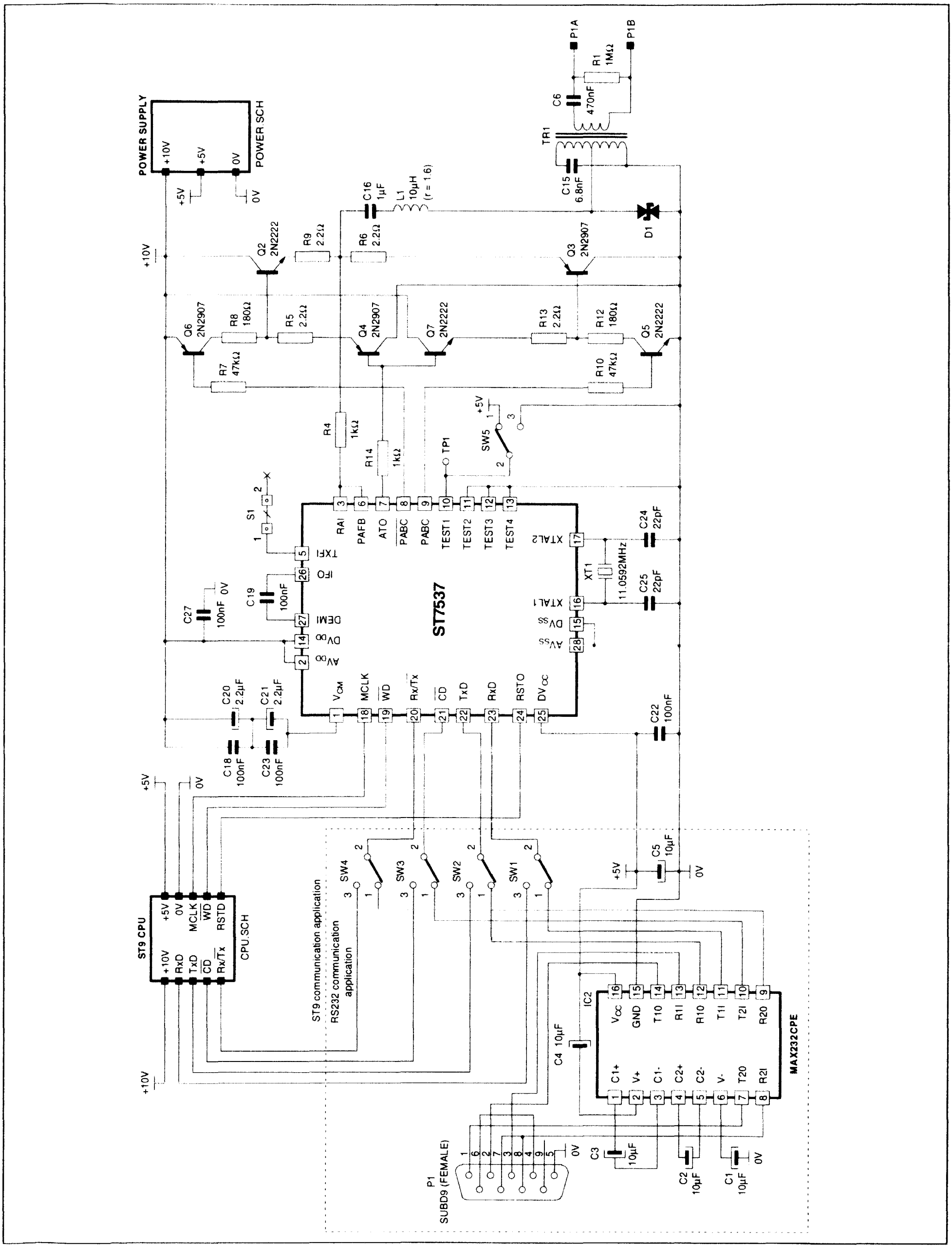
```
/                                                                      *
;|
;|Routine Name   : PROCTEST
;|File Name      : PROCTEST.C(compile with MS QuickC)
;|Action         : Performs software validation of MB076b hardware
;|Author(FN/LN)  : Olivier Garreau / PPG / GRENOBLE
;|First rev date: 06/10/93
;|Last rev date : 13/10/93
;|Input paramet.: none
;|Output paramet: none
;|Modified var. : none
;|Global varia. : none
;|Comments      : build a link this way :
;|
;|Windows PC->modem1->modem2->ST9 ->!
;|                                     ! (TEMPO)
;|          Match test<-PC<-modem1<-modem2 <-!
;|
;|               So it checks both links of network.
;|Compile with MS QuickC (Project: QuickWin EXE).
;|

*/

#include <windows.h>
#include <stdio.h>
#include <io.h>
#include <string.h>
#include <conio.h>

#include "proctest.h"

DCBdcb_comm;
COMSTATcom_stat;
charbuffer_in[size_buffer_in];
charbuffer_out[size_buffer_out] = test_string;// init string to send
intcom_device,test_key;



voidwait (tempo)

DWORDtempo;// perform a tempo of 'tempo' milliseconds
{
```

```
DWORDnow;
now = GetCurrentTime();

        while (GetCurrentTime() < now + tempo) {
// this loop wait for 'tempo' delay
        }
}


voidflush_in (void)

    {
int err;

err=FlushComm(com_device,1);
if(err < 0) {
      printf ("**** failed to flush input buffer ******\n");
      return ;
      }
      strcpy (buffer_in ,"Time out buffer IN !!");
}

voidflush_out (void)

    {
int err;

err=FlushComm(com_device,0);
if(err < 0) {
      printf ("**** failed to flush output buffer ******\n");
      return ;
      }
}

voidinit_com_port(void)// init COM1 for bidir link

{
interr;

        com_device=OpenComm("COM1",size_buffer_in,size_buffer_out);
      if(com_device < 0) {
      printf ("**** failed to open RS232 port ******\n");
      return ;
      }
      err = BuildCommDCB("COM1:1200,N,8,1",&dcb_comm );
      if(err < 0) {
      printf ("**** can not setup COM1 ******\n");
      return ;
      }

      else {

      dcb_comm.fBinary=TRUE;
       dcb_comm.fRtsDisable=TRUE;
       dcb_comm.fOutxCtsFlow=FALSE;
       dcb_comm.fOutxDsrFlow=FALSE;
       dcb_comm.fDtrDisable=TRUE;
       dcb_comm.fOutX=FALSE;
       dcb_comm.fInX=FALSE;
       dcb_comm.fDtrflow=FALSE;
       dcb_comm.fRtsflow=FALSE;
}

      err = SetCommState(&dcb_comm);
if(err < 0) {
printf ("**** cannot setup COM1 ******\n");
      return ;
        }

        else {

        printf ("Please reset both board 1 and board 2\n");

        printf("type 'ENTER' when done...\n");
             test_key=getchar();
            }
}


voidsend_frame (void)

{
interr;

     err = EscapeCommFunction(com_device, SETRTS);// SET RXTX LOW
    if(err < 0) {
      printf ("**** can not clear RXTX signal ******\n");
      return ;
      }
    wait (delay_RXTX_start); // wait for RXTX to be set up LOW

    err = WriteComm(com_device,buffer_out,length_string);
// SEND BUFFER
    if(err < 0) {
      printf ("**** can not send test frame ******\n");
      return ;
            }

          do {

        err = GetCommError(com_device, &com_stat);
// WAIT TIL BUFFER OUT EMPTY
    if(err < 0) {
      printf ("**** can not wait for buffer
transmitted ******\n");
      return ;
            }
            }
     while (com_stat.cbOutQue != 0);

    wait (delay_RXTX_end);// wait for RXTX to be set up HIGH

     err = EscapeCommFunction(com_device, CLRRTS);// SET RXTX HIGH
    if(err < 0) {
      printf ("**** can not set RXTX signal ******\n");
      return ;
            }

}

voidreceive_frame(void)

{

interr;
DWORDread_time;

        wait(proper_buffer);// wait to have a proper input buffer
        flush_in();// flush input buffer

        read_time = GetCurrentTime();
        do {

        err = GetCommError(com_device, &com_stat);
// WAIT TIL BUFFER IN FULL
    if(err < 0) {
      printf ("**** can not wait for buffer input ***** \n");
      return ;
            }
            }
while ((com_stat.cbInQue
<length_string)&((GetCurrentTime()-read_time)<Watchdog));

// Wait until size_buffer_in

 err = ReadComm(com_device,buffer_in,length_string);
//Read frame
    if(err < 0) {
      printf ("**** can not receive back frame ******\n");
      return ;
            }
}

intmatch_frames(void)

{
printf("String sent    : %s\n",buffer_out);
printf("String received: %s\n",buffer_in);

if(strcmp(buffer_in,buffer_out) == 0)

{ return 1 ;}

else

{ return 0 ;}

}

voiddisplay_result(int value)

{
     if (value)

     {printf("----- GOOD MATCH !!!!\n");}

     else

     {printf("***** BAD MATCH !!!!\n");}
}
```

```
// ********************** MAIN PROCEDURE ****************************

voidmain(void)

{
intcounter,result,fail,err;



    printf("Validation Software for ST7537 Starter Kit \n");
printf("performs a bidirectional test from : \n");
printf("MB076b board 1 to MB076b board 2\n");
printf("boundary of board 1 is a PC\n");
printf("boundary of board 2 is the ST9\n\n");

while(TRUE)

{
init_com_port();// init RS232 com

fail = 1;// number of communication attempts

for (counter=0;counter<number_of_test;counter++)

{

send_frame();// send frame through network

receive_frame();// wait until back frame

result = match_frames(); // test if identical

        display_result(result);// display result of attempt n

if(result)
{
 break; // get out of the loop
 }
else
{
 fail ++; // increment attempt number
 wait(reset_delay);
// wait during 2 seconds to reset Board 2 ST9
}

}

err = CloseComm(com_device);
if(err < 0) {
      printf("**** cannot close COM ******\n");
      return ;
            }

    printf("End of test !!.\n\n");

if (result)
{
printf("The current boards are GOOD after %dattempt(s).\n"
,fail);
}
else
{
printf("The current boards are NOT GOOD !!\n");
printf("Please send them back to designer.\n");
}

printf("End of procedure\n\n");

   printf("type 'ENTER' to test new boards ...\n");
test_key=getchar();


    }

}
```

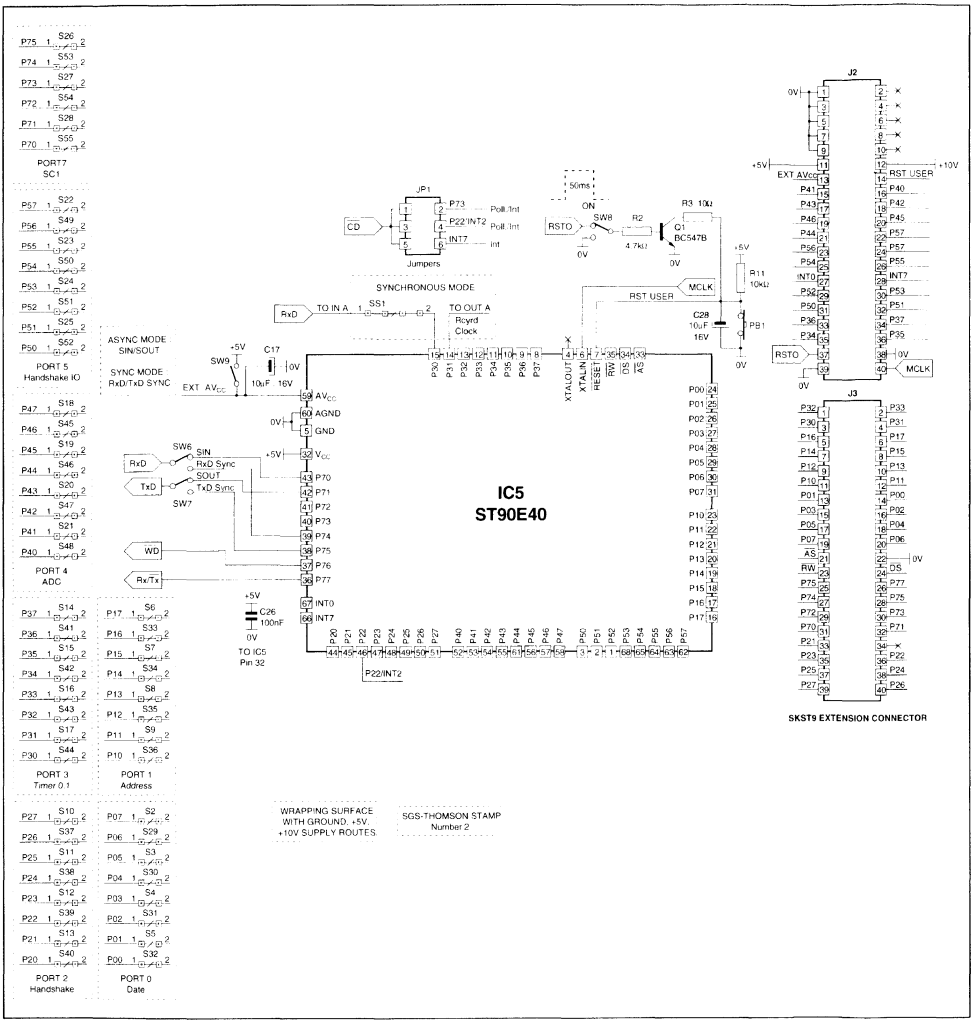
VI - APPENDIX 3 : SCHEMATICS OF ST7537 STARTER KIT

Figure 5 : PLM Interface



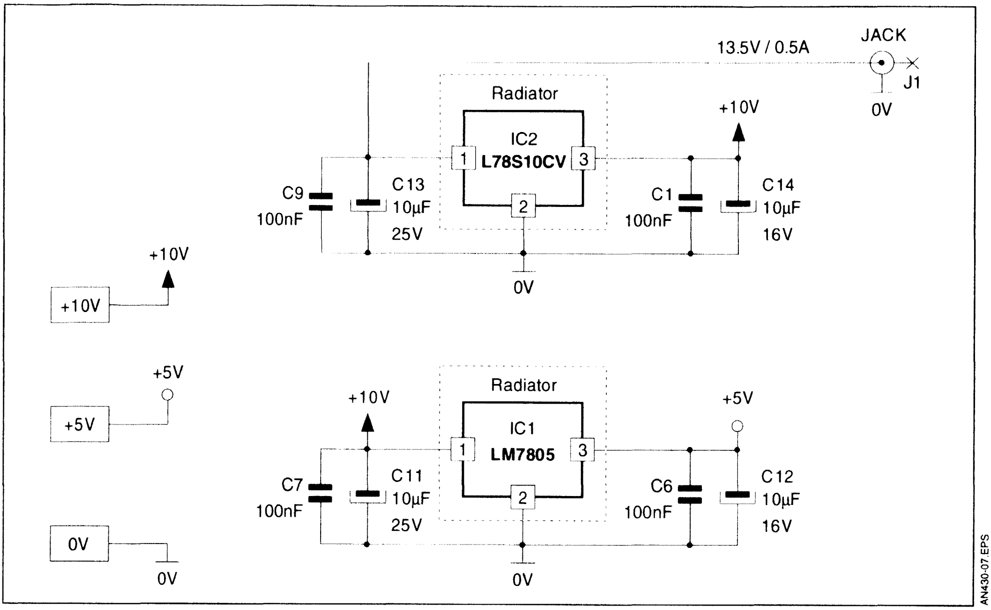
AN430-05.EPS

Figure 6 : ST9 Connections



AN430-06.EPS

Figure 7 : Power Supply



THE SOFTWARE INCLUDED IN THIS NOTE IS FOR GUIDANCE ONLY. SGS-THOMSON SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM USE OF THE SOFTWARE.

TRIAC APPLICATION NOTE

SGS-THOMSON MICROELECTRONICS

APPLICATION NOTE

PROTECT YOUR TRIACS

By P. RAULT

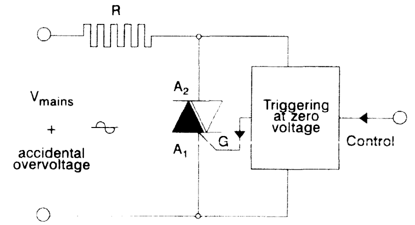
In most of their applications, Triacs are directly exposed to overvoltages transmitted by the mains. When used to drive resistive loads (temperature regulation), it is essential to provide them with efficient protection.

WHY PROTECTION ?

In a typical circuit (Figure 1), an overvoltage superimposed on the network voltage can turn on the Triac by exceeding its avalanche voltage.

Figure 1 : Typical Circuit

The Triac is directly connected to the distribution network : risk of damage.



Under these conditions, because of its internal structure, only a part of the Triac is effectively turned on and it can thus withstand only very low di/dt. This explains the considerable danger of damage to the component when used to drive purely resistive loads. In reality, the di/dt when turning on can, in this case, reach very high values (> 100 A/µs) since only the inductance of the connections limits the rate at which the current can increase.

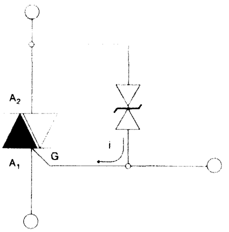
WHAT WE PROPOSE

The principle of the protection which we have studied consists of turning on the Triac by the gate as soon as the voltage across it exceeds a certain value (Figure 2), thus ensuring a high level of safety. To do this we use a bidirectional TRANSIL diode whose current/voltage characteristic is shown in Figure 3.

When the voltage applied to the Triac reaches the V_{BR} voltage of the TRANSIL, the latter conducts, producing a current in the Triac gate and turning it on (Figure 4). The Triac continues to conduct till the half cycle current passes through zero (Figure 5).

Figure 2 : Protection of the Triac by a Bidirectional TRANSIL Diode

The Triac is turned on by gate current (i) as soon as voltage A2 exceeds the voltage V_{BR} of the TRANSIL.



THE ADVANTAGES OF THIS SOLUTION

- The Triac will always operate within the voltage limits given by the manufacturer (VDWM) and thus far from the avalanche zone.
- Not much power is dissipated in the Triac during the disturbance before the turn-on, the dissipated power is localized in the protection component (the TRANSIL is made for that !).

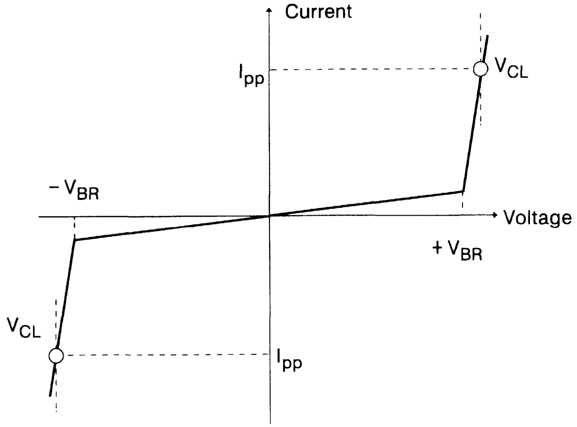
The Triac is turned on by a gate current which will ensure optimal di/dt conditons.

THE RESULTS

We have carried out tests with repetitive overloads (1Hz) under various conditions :
Exponential shock waves of about 1ms, calibrated in voltage (up to 2000V) and controlled in di/dt (500 A/µs max).

The tests were carried out with steep-edged voltage pulses (dV/dt > 1000 V/µs) and also with gradual slopes (< 50 V/µs).
All these tests were successful : zero failure.

Figure 3 : Voltage-current Characteristics of a TRANSIL Diode



V_{BR} specified at 1mA (tolerance 5 or 10%)
V_{CL} limitation voltage is given for a high I_{PP} current level (from several amperes to several tens of amperes, depending on the type)

Figure 4 : Characteristics of the TRIAC + TRANSIL Assembly. Case of a 600V/12A Triac Protected by a 440V TRANSIL Diode (the dotted line gives the characteristics of the Triac alone).

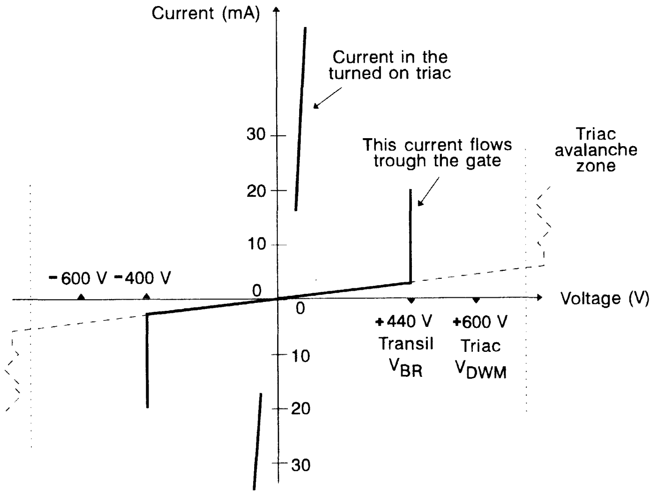
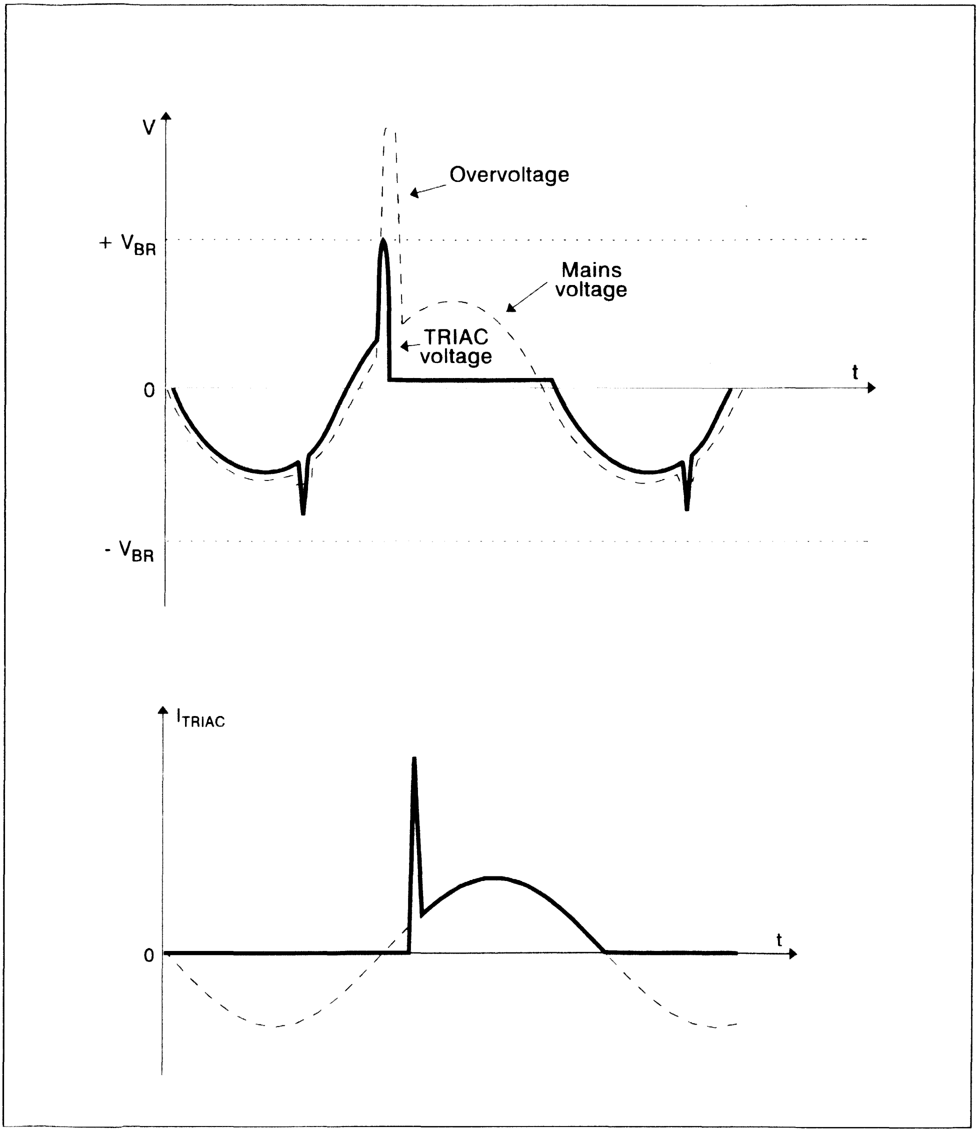


Figure 5 : Behaviour of a Triac Protected by a TRANSIL Diode (the triac is turned on by the gate at the beginning of the overvoltage and continues through the rest of the half-wave)



SELECTION OF THE TRANSIL DIODE REQUIRED FOR PROTECTING A TRIAC

Voltage : V_R

Obviously the Triac associated with the TRANSIL diode should not be turned on by the maximum mains voltage. An additional safety margin should be given to prevent untimely turning on by the small voltage spikes, often repetitive, which are always present on a "normally" disturbed mains line.

$$V_R > V_{mains} \times \sqrt{2} + \text{safety margin.}$$

In the absence of accurate specifications, add 20% for the safety margin.

Example : 220V network :

$$V_R > 220\sqrt{2} + 20\% = 375V$$

Power

The TRANSIL only conducts when turning on the triac ($t \equiv 1\mu s$).

The current during this time can reach very high levels (several tens of amperes) in the case of disturbances with steep edges (> 1000V/S), however the dissipated power remains well within the capability of TRANSILS.

The BZW 04(400W/1ms) is suitable for all cases.

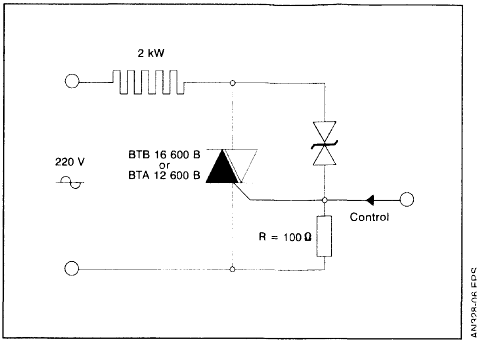
PRACTICAL EXAMPLE

Drive circuit for a 2kW heating element on 220V mains (Figure 6).

The BZW04-376B type TRANSIL perfectly protects the BTB16-600B triac ($V_{DWM} = \pm 600V$).

The 100Ω resistor, R, between the gate and A1 is not absolutely indispensable, but it preserves the dV/dt characteristic of the Triac which would be reduced (by about 20%) by the junction capacitance of the TRANSIL between anode and gate.

Figure 6 : Practical Example of the Protection of a 12 or 16A Triac against Overvoltages



CONCLUSION

With the protection circuit proposed, the Triac always operates under perfectly defined conditions in case of overvoltages :

- The voltage remains limited to the maximum specified for the triac
- Turn-on is ensured by a gate current.

This circuit, which we have tested in a number of different setups (different lads, high amplitude overvoltages, disturbances of long duration, etc...), enables a considerable increase in the reliability of circuits using triacs and is indispensable for driving resistive loads on highly disturbed networks.

SGS-THOMSON
MICROELECTRONICS

APPLICATION NOTE

TRIAC + MICROCONTROLLER SAFETY PRECAUTIONS FOR DEVELOPMENT TOOL

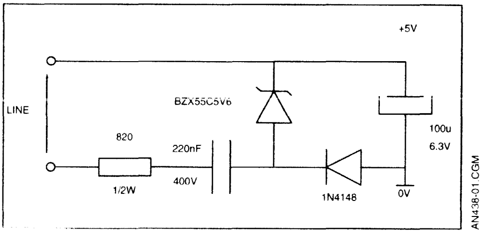
By P. RABIER

The goal of this paper is to analyse the different ways to configure a micro-controller and a development tool during the debbugging phase. The major problem is due to the direct connection of the computer I/O lines with the mains power. Some precautions have to be taken during the emulation in order to avoid destruction.

I - LOW COST POWER SUPPLY

In most low cost applications the step down transformer is not used and the power supply delivers low current, as shown for example in Figure 1.

Figure 1 : Unisulated Power Supply



In domestic appliance applications, one of the most important power switches is the Triac.

The function of driving the Triac becomes more and more complex. For this reason, microcontrollers are becoming more and more common. Furthermore, sensitive Triacs with high commutation parameters, for example LOGIC LEVEL triacs can be triggered directly by the microcontroller without any buffer. Sensitive triacs and microcontrollers allow decrease in power consumption.

In this way the power supply can be optimized to reduce the cost. Optimisation can be achieved by removing the transformer.

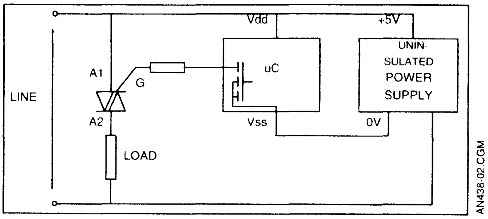
The consequence is that there is no insulation, the microcontroller is connected directly on the line !

When the software is emulated on the application board, the output port (RS232 port) of the computer is connected on the line via the emulator.

If some precaution is not taken "something" will be destroyed !

Figure 2 gives an example of an application using a triac and a microcontroller.

Figure 2 : Triac and Microcontroller on the Line

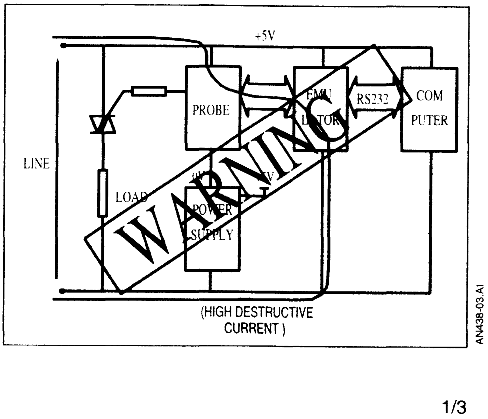


In this case the micro-controller is supplied by an uninsulated +5V power supply connected directly to the line, and a low level (0V) on the output ports of the micro-controller is needed to trigger the Triac.

II - USE OF A DEVELOPMENT TOOL

During the debbuging phase, the micro-controller is removed and is replaced by the emulation probe. The circuit corresponding to the emulation phase of the previous example is shown in Figure 3.

Figure 3 : Circuit without protection
(beware : this circuit is dangerous)



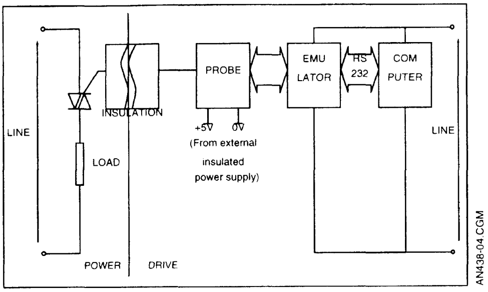
The line is connected directly to the +5V of the emulator and a high (destructive) current can flow through the emulator and/or the computer.

III - INSULATED SYSTEM

To avoid destruction of the development tool it is necessary to have an insulation between line and probe. This insulation can be achieved by optocouplers, pulse transformers, or insulation transformers.

Figure 4 shows the topology of the most common insulation.

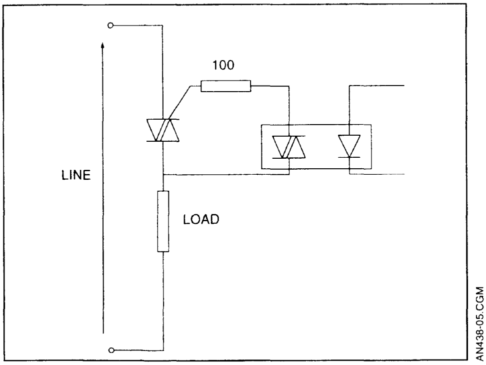
Figure 4 : Conventinal Insulation



III.1 - Optotriac

Figure 5 shows the circuit with Triac and Optotriac.

Figure 3 : Optotriac Drive



The Triac is working in the 1st and in the 3rd quadrants.

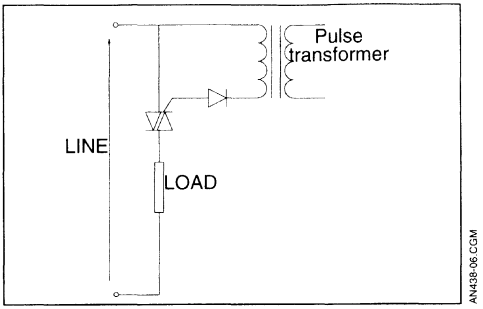
The main advantage of a such system is the low cost of the Optotriac, but it needs an isolated auxiliary power supply.

For a zero crossing optotriac, the Triac is triggering with a gate current equal to the gate trigger current with a very low dI_G/dt. This does not allow high di/dt at turn on. That is to say the control of high current resistive load is not recommended with this method.

III.2 - The Pulse Transformer

Figure 6 shows the circuit with a Triac and a pulse transformer.

Figure 6 : Pulse Transformer Insulation



The Triac is working in the 2nd and 3rd quadrants.

This system is simple to use when the Triac was initially driven by a buffer transistor, but it needs an external power supply. The high dI_G/dt through the gate allows high current resistive loads to be driven. Due to the saturation of the magnetic material, this system cannot drive small loads because the gate current is cancelled before the latching current has been reached.

For more information refer to the application note "Triac Control by Pulse Transformer".

III.3 - The Line Insulation Transformer

In the previous examples,the insulation was between the Triac and the microcontroller (see Figure 7).

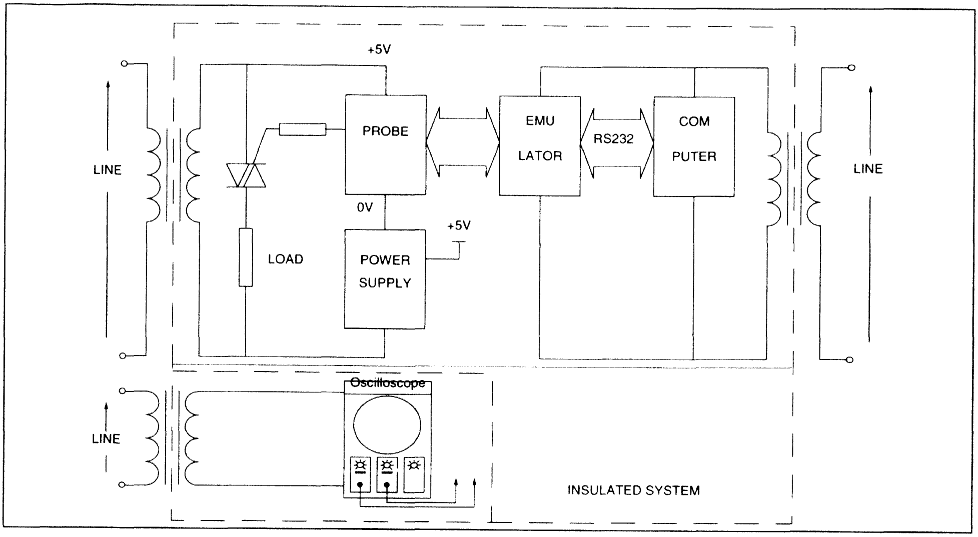
Another solution is to supply each equipment connected to the board from the mains through an insulation transformer.

If an oscilloscope is used, it also has to be separately insulated.

The main advantage of this system is that we do not need to modify the target system during the debbuging phase and it can be used with the microcontroller.

When a transformer is used between line and Triac it should be noted that the line impedance is modified and then the behaviour of the Triac, load and line set can be different (waveform of current).

Figure 7 : Insulation with transformers



IV - SUMMARY

New LOGIC LEVEL and SNUBBERLESS Triacs can be connected directly to the microcontroller without buffers or insulation. Furthermore, low cost power supplies without a transformer are becoming more common. There is an increasing number of applications supplied directly from the mains, and the microcontroller is directly connected to it.

During the debbuging phase when connecting the development tool, a galvanic insulation is absolutely necessary.

This insulation can be done in 3 ways.

- With Optotriacs :
  - Need modifications on the target system
  - Need external power supply
- With pulse transformer :
  - Need modifications (transistor to drive the pulse transformer)
  - Need an external power supply
  - Cannot drive small loads
- With insulation transformer :
  - No modification on the application board
  - Modification of the line impedance due to the transformer between line and load

Therefore a microcontroller operating on the mains with a Triac may be directly connected to the line.

SALES OFFICES

EUROPE

DENMARK

2730 HERLEV
Herlev Torv, 4
Tel. (45-44) 94.85.33
Telex: 35411
Telefax: (45-44) 948694

FINLAND

LOHJA SF-08150
Ratakatu, 26
Tel. (358-12) 155.11
Telefax. (358-12) 155.66

FRANCE

94253 GENTILLY Cedex
7 - avenue Gallieni - BP. 93
Tel.: (33-1) 47.40.75.75
Telex: 632570 STMHQ
Telefax: (33-1) 47.40.79.10

67000 STRASBOURG
20, Place des Halles
Tel. (33-88) 75.50.66
Telefax: (33-88) 22.29.32

GERMANY

85630 GRASBRUNN
Bretonischer Ring 4
Postfach 1122
Tel.: (49-89) 460060
Telefax: (49-89) 4605454
Teletex: 897107=STDISTR

60327 FRANKFURT
Gutleutstrasse 322
Tel. (49-69) 237492-3
Telefax: (49-69) 231957
Teletex: 6997689=STVBF

30695 HANNOVER 51
Rotenburger Strasse 28A
Tel. (49-511) 615960-3
Teletex: 5118418 CSFBEH
Telefax: (49-511) 6151243

90491 NÜRNBERG 20
Erlenstegenstrasse, 72
Tel.: (49-911) 59893-0
Telefax: (49-911) 5980701

70499 STUTTGART 31
Mittlerer Pfad 2-4
Tel. (49-711) 13968-0
Telefax: (49-711) 8661427

ITALY

20090 ASSAGO (MI)
V.le Milanofiori - Strada 4 - Palazzo A/4/A
Tel. (39-2) 57546.1 (10 linee)
Telex: 330131 - 330141 SGSAGR
Telefax: (39-2) 8250449

40033 CASALECCHIO DI RENO (BO)
Via R. Fucini, 12
Tel. (39-51) 591914
Telex: 512442
Telefax: (39-51) 591305

00161 ROMA
Via A. Torlonia, 15
Tel. (39-6) 8553960
Telex: 620653 SGSATE I
Telefax: (39-6) 8444474

NETHERLANDS

5652 AR EINDHOVEN
Meerenakkerweg 1
Tel.: (31-40) 550015
Telex: 51186
Telefax: (31-40) 528835

SPAIN

08004 BARCELONA
Calle Gran Via Corts Catalanes, 322
6th Floor, 2th Door
Tel. (34-3) 4251800
Telefax: (34-3) 4253674

28027 MADRID
Calle Albacete, 5
Tel. (34-1) 4051615
Telex: 46033 TCCEE
Telefax: (34-1) 4031134

SWEDEN

S-16421 KISTA
Borgarfjordsgatan, 13 - Box 1094
Tel.: (46-8) 7939220
Telex: 12078 THSWS
Telefax: (46-8) 7504950

SWITZERLAND

1218 GRAND-SACONNEX (GENEVA)
Chemin Francois-Lehmann, 18/A
Tel. (41-22) 7986462
Telex: 415493 STM CH
Telefax: (41-22) 7984869

UNITED KINGDOM and EIRE

MARLOW, BUCKS
Planar House, Parkway
Globe Park
Tel.: (44-628) 890800
Telex: 847458
Telefax: (44-628) 890391

AMERICAS

BRAZIL

05413 SÃO PAULO
R. Henrique Schaumann 286-CJ33
Tel.: (55-11) 883-5455
Telex: (391)11-37988 "UMBR BR"
Telefax : (55-11) 282-2367

CANADA

NEPEAN ONTARIO K2H 9C4
301 Moodie Drive Suite 307
Tel.: (613) 829-9944
Telefax: (613) 829-8998

U.S.A.

NORTH & SOUTH AMERICAN
MARKETING HEADQUARTERS
55 Old Bedford Road
Lincoln, MA 01773
Tel.: (617) 259-0300
Telefax: (617) 259-4421

SALES COVERAGE BY STATE

ALABAMA
Huntsville - Tel.: (205) 533-5995
Fax: (205) 533-9320

ARIZONA
Phoenix - Tel.: (602) 867-6217
Fax: (602) 867-6200

CALIFORNIA
Santa Ana - Tel.: (714) 957-6018
Fax: (714) 957-3281
San Jose - Tel.: (408) 452-8585
Fax: (452) 1549

COLORADO
Boulder - Tel.: (303) 449-9000
Fax: (303) 449-9505

FLORIDA
Boca Raton - Tel.: (407) 997-7233
Fax: (407) 997-7554

GEORGIA
Norcross - Tel.: (404) 242-7444
Fax: (404) 368-9439

ILLINOIS
Schaumburg - Tel.: (708) 517-1890
Fax: (708) 517-1899

INDIANA
Kokomo - Tel.: (317) 455-3500
Fax: (317) 455-3400
Indianapolis - Tel.: (317) 575-5520
Fax: (317) 575-8211

MICHIGAN
Livonia - Tel.: (313) 953-1700
Fax: (313) 462-4071

MINNESOTA
Bloomington - Tel.: (612) 944-0098
Fax: (612) 944-0133

NORTH CAROLINA
Cary - Tel.: (919) 469-1311
Fax: (919) 469-4515

NEW JERSEY
Voorhees - Tel.: (609) 772-6222
Fax: (609) 772-6037

NEW YORK
Poughkeepsie - Tel.: (914) 454-8813
Fax: (914) 454-1320

OREGON
Lake Oswego - Tel.: (503) 635-7650

TENNESSEE
Knoxville - Tel.: (615) 524-6239

TEXAS
Austin - Tel.: (512) 502-3020
Fax: (512) 346-6260
Carrollton - Tel.: (214) 466-8844
Fax: (214) 466-8130
Houston - Tel.: (713) 376-9936
Fax: (713) 376-9948

FOR RF AND MICROWAVE POWER TRANSISTORS CONTACT THE FOLLOWING REGIONAL OFFICE IN THE U.S.A.

PENNSYLVANIA
Montgomeryville - Tel.: (215) 361-6400
Fax: (215) 361-1293

ASIA / PACIFIC

AUSTRALIA

NSW 2220 HURTSVILLE
Suite 3, Level 7, Otis House
43 Bridge Street
Tel. (61-2) 5803811
Telefax: (61-2) 5806440

HONG KONG

WANCHAI
22nd Floor - Hopewell centre
183 Queen's Road East
Tel. (852) 8615788
Telex: 60955 ESGIES HX
Telefax: (852) 8656589

INDIA

NEW DELHI 110019
Liaison Office
3rd Floor, F-Block
International Trade Tower
Nehru Place
Tel. (91-11) 644-5928/647-9415
Telex: 031-70193 STMI IN
Telefax: (91-11) 6443054

MALAYSIA

SELANGOR, PETALING JAYA 46200
Unit BM-10
PJ Industrial Park
Jalan Kemajuan 12/18
Tel.: (03) 758 1189
Telefax: (03) 758 1179

PULAU PINANG 10400
4th Floor - Suite 4-03
Bangunan FOP-123D Jalan Anson
Tel. (04) 379735
Telefax (04) 379816

KOREA

SEOUL 121
8th floor Shinwon Building
823-14, Yuksam-Dong
Kang-Nam-Gu
Tel. (82-2) 553-0399
Telex: SGSKOR K29998
Telefax: (82-2) 552-1051

SINGAPORE

SINGAPORE 2056
28 Ang Mo Kio - Industrial Park 2
Tel. (65) 4821411
Telex: RS 55201 ESGIES
Telefax: (65) 4820240

TAIWAN

TAIPEI
11th Floor
105, Section 2 Tun Hua South Road
Tel. (886-2) 755-4111
Telex: 10310 ESGIE TW
Telefax: (886-2) 755-4008

THAILAND

BANGKOK 10110
54 Asoke Road
Sukhumvit 21
Tel.: (662) 260 7870
Telefax: (662) 260 7871

JAPAN

TOKYO 108
Nisseki - Takanawa Bld. 4F
2-18-10 Takanawa
Minato-Ku
Tel. (81-3) 3280-4121
Telefax: (81-3) 3280-4131

Information furnished is believed to be accurate and reliable. However, SGS-THOMSON Microelectronics assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of SGS-THOMSON Microelectronics. Specification mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. SGS-THOMSON Microelectronics products are not authorized for use as critical components in life support devices or systems without express written approval of SGS-THOMSON Microelectronics.

SGS-THOMSON Microelectronics GROUP OF COMPANIES
Australia - Brazil - France - Germany - Hong Kong - Italy - Japan - Korea - Malaysia - Malta - Morocco - The Netherlands - Singapore - Spain - Sweden - Switzerland - Taiwan - Thailand - United Kingdom - U.S.A.

Recycled and chlorine free paper