REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	2 SEP 97	Final 1 May 96 - 30 Apr 97

4. TITLE AND SUBTITLE
"Computer Aided Synthesis or Measurement Schemas For Telemetry Applications"

5. FUNDING NUMBERS
Grant - F49620-96-1-0186

6. AUTHOR(S)
Professor Peter H. Sydenham, Investigator
Peter Evdokiou, Researcher

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Australian Center for Test and Evaluation
University of South Australia
Building 7, Smith Road
Salisbury East
South Australia 5109

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Asian Office of Aerospace Research and Development (AoARD)
Unit 45002
APO AP 96337-5002

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AOARD 96-01

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report describes the design, implementation, testing and operation of a subminiature telemetry (SMT) configuration tool. The configuration tool is comprised of software and hardware components. The softtware, written in Microsoft Visual Basic Version 4.0, offers a graphical user interface for specifying SMT configurations. It also includes a function for automating the tedious task of setting up a telemetry frame structure. The program features automatic report generation, programming of the SMT devices and setting up a test environment to configure and verify correct operation of the SMT devices in a laboratory before going out in the field. The hardware includes a programming interface between a PC and an SMT device and a programmable test signal generator to test and verify the correct SMT configuration. The SMT configuration tool has reduced a labour skilled intensive process to an automated, efficient and user friendly computer aided approach. The tool has been used in realistic environments to support test and evaluation missions of high-tech state-of-the-art conventional armanent.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Telemetry, subminature, programming.	140
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

DEPARTMENT OF THE AIR FORCE
ASIAN OFFICE OF AEROSPACE RESEARCH AND DEVELOPMENT
AIR FORCE RESEARCH LABORATORY/OFFICE OF SCIENTIFIC RESEARCH
AOARD UNIT 45002, APO AP 96337-5002

10 Jun 98

MEMORANDUM FOR Defense Technical Information Center
8725 John J. Kingman Road, STE 0944
Fort Belvoir VA 22060-6218

FROM: AOARD
Unit 45002
APO AP 96337-5002

SUBJECT: Submission of Document

1. A Final Technical Report entitled, "Computer Aided Synthesis of Measurement Schemas for Telemetry Applications" is attached.

2. Please contact our Administrative Officer, Dr. Jacque Hawkins, AOARD, DSN: 315-229-3388, DSN FAX, 315-229-3133, e-mail: hawkinsj@aoard.yokota.af.mil, if you need additional information.

KOTO WHITE
Director, AOARD

Attachment:
AOARD 96-01, w/SF 298 and DTIC 50

FINAL TECHNICAL

REPORT

Computer Aided Synthesis of

Measurement Schemas

for

Telemetry Applications

Principal Investigator: Prof. Peter H. Sydenham
Principal Researcher: Peter Evdokiou

from

Australian Center for Test and Evaluation
University of South Australia
Building 7, Smith Road
Salisbury East
South Australia 5109

Grant Number: F49620-96-1-0186

Tuesday, 2 September 1997

Final Technical Report: Created at:

Australian Centre for Test and Evaluation

University of South Australia

Adelaide SA, Australia

Period: 1 May 96 to 30 April 97

Prof. Peter Sydenham
Australian Centre for Test & Evaluation
University of South Australia
Salisbury SA, Australia

Peter Evdokiou, Meng, Beng (Hons)
Australian Centre for Test & Evaluation
University of South Australia
Salisbury SA, Australia

Table of Contents

List of Figures

List of Tables

Preface

This is the Final Technical Report of the collaborative project between the Australian Centre for Test & Evaluation (ACTE) at the University of South Australia (UniSA) and Wright Laboratory of Eglin Air Force Base. The Author wishes to acknowledge the work of the following people:

- ACTE (South Australia)
  - Prof. Peter Sydenham, Director of ACTE
  - David Harris, Manager of ACTE,
  - Mark Dvorak, Project Leader at ACTE,
  - Peter Evdokiou, Principal Researcher on the system conceptualisation, requirements definition, hardware/software implementation, test and evaluation, and operation.
  - Eric Lammerts (hardware/software implementation),
  - Howy Truong (software debugging)
  - La Cuong (literature review on modulation schemes).
- Eglin Air Force Base (Florida)
  - Ed Keller (Eglin AFB)
  - John Cesulka
  - David Kerr
- AFOSR (Tokyo)
  - Dr. Thomas Davis (AFOSR),
  - USAF.

Peter Evdokiou

Adelaide, September 2, 1997

Abstract

This report describes the design, implementation, testing and operation of a subminiature telemetry (SMT) configuration tool. The configuration tool comprises of software and hardware components. The software, written in Microsoft Visual Basic Version 4.0, offers a graphical user interface for specifying SMT configurations. It also includes a function for automating the tedious task of setting up a telemetry frame structure. The program features automatic report generation, programming of the SMT devices and setting up a test environment to configure and verify correct operation of the SMT devices in a laboratory before going out in the field.

The hardware includes a programming interface between a PC and an SMT device, and a programmable test signal generator to test and verify the correct SMT configuration.

The SMT configuration tool has reduced a labour skilled intensive process to an automated, efficient and user friendly computer aided approach. The tool is been used in realistic environments to support test and evaluation missions of high-tech state-of-the-art conventional armament at Eglin Air Force Base.

1. Introduction

The University of South Australia is conducting research in collaboration with the Royal Australian Air Force and Wright Laboratory of Eglin Air Force Base (Florida, USA) to optimise telemetry stream data structures in an effort to improve the traceability between test requirements and measured data.

Modern complex systems require rigorous test and evaluation programmes to be conducted to assure that they operate reliably to prescribed standards. Conducting a test and evaluation programme requires the acquisition, management, processing and analysis of large volumes of test data. The measured data must be traceable to internationally accepted standards, and the data processing chain must be well defined to guarantee that data of known quality is delivered to customers.

The colaborative research has investigated how to adapt already developed ACTE specification tools to create highly-marketable, off the shelf, Sub-Miniature Telemetry (SMT) modules for widespread domain use. These domains are applications where the cost per system is affordable to users who previously could not make use of telemetry due to high cost, large bulk, and lack of sufficient technical expertise. This research has explored set-up configuration tools and the knowledge needed to put telemetry in the hands of non-expert users as an effective, affordable, system. Examples of potential users of "affordable telemetry" include:

- Ambulatory medical patient monitoring,
- Machine tool automation,
- Process Control,
- Wild/farm animal and fish health monitoring,
- "Just in time", air, sea and land vehicle test,
- etc.

The aim of the collaborative research was focused on conceptualising and developing demonstrable structured computer aided methodologies, that will aid non-expert users of SMT technology, in the process of establishing domain specific measurement/data-acquisitions schemas from an overall generic telemetry model. The model is based on the structural, functional, and behavioural capabilities of the current SMT technology.

This report describes the conceptualisation, design, development, test and evaluation, and operation of a software configuration tool for SMT modules.

2. Objective

2.1. Aim

The aim of this hardware/software development project shall be to conceptualise, design, develop, test and evaluate a full prototype system of what shall be called the "Sub-Miniature Telemetry (SMT) Hardware Test-Bed and Software Configuration Tool. This tool shall form part of the overall research project currently being undertaken by the Principal Researcher, Peter Evdokiou. The overall research project is known as "Computer Aided Synthesis of Measurement Schemas for Telemetry Applications".

2.2. Problem Definition

The hardware/software development project can best be defined by considering **Error! Reference source not found.**.

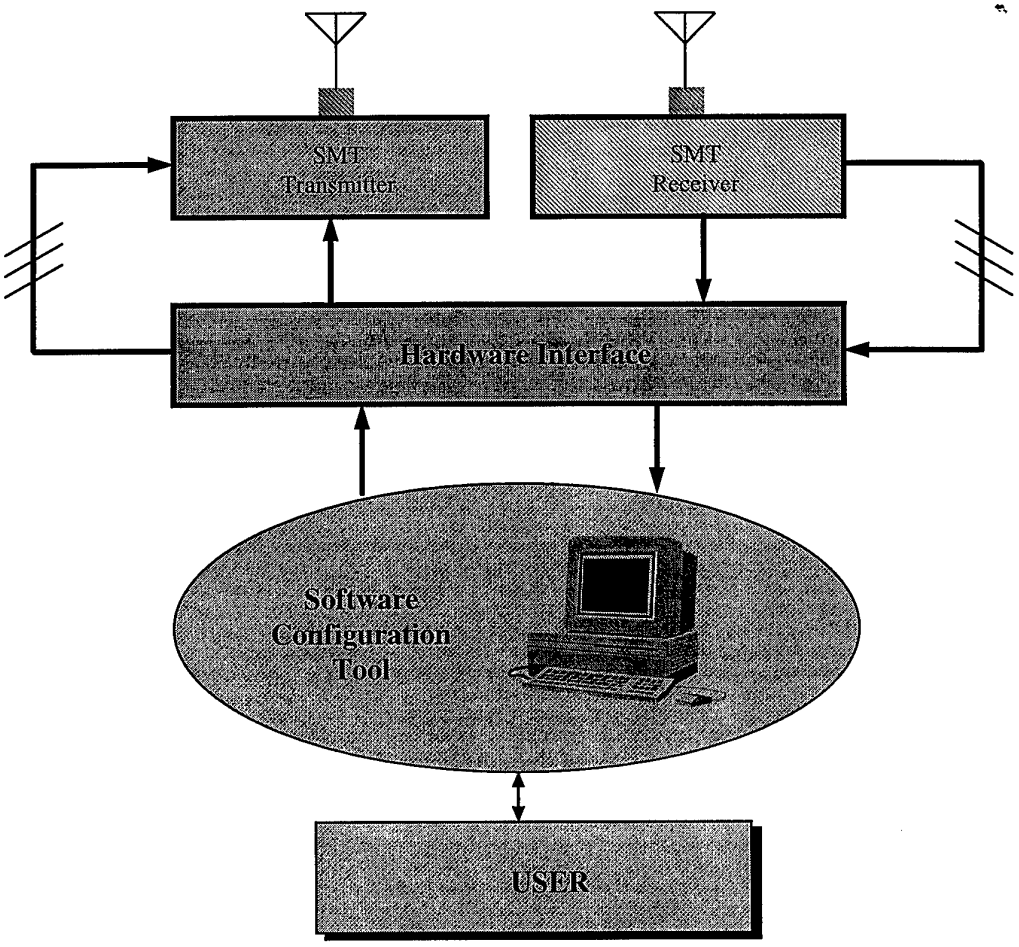


Figure 1. SMT Hardware Test-Bed & Software Configuration Tool System.

Figure 1 shows the system as a whole when in operation. It shows structurally that the SMT transmitter and receiver shall be configured through a Software Configuration Tool via a Hardware Test-Bed Interface between the SMT's and the PC environment. A description of the Software Configuration Tool and the Hardware Test-Bed Interface appear in the following sections.

2.2.1. Software Configuration Tool

The Software Configuration Tool serves three main purposes and hence can be considered to have three modes of operation:

- ***Mode 1:*** **"SMT Set-Up"**

  Mode 1 of operation involves the set-up of the SMT EEPROMS on board the SMT modules. The EEPROMS on board the SMT's contain the program that configure the SMT's for an specific measurement schema. Given a file containing a list of parameters such as:

  - number of inputs,
  - type of inputs (ie., analog, digital, discrete),
  - sampling rates,
  - analog filter bandwidths,
  - TM format,
  - sync words,
  - transmit frequency,
  - gains, etc.

  An EEPROM file is required to be generated in the proper syntax and format, as specified by Harris Corporation Engineers, comprising the above instantiated parameters.

  In this mode of operation the Software configuration tool should be able to take a file comprising of the above instantiated parameters and produce an EEPROM file that complies to the Harris Corporation syntax and format, and then download to the EEPROM on the SMT's (using a special driver supplied by Harris Corporation) via a Hardware Test-Bed Interface that connects the PC and SMT's together.

  Configuring the SMT's for an specific measurement schema is only part of the problem. Verification must be made about the correctness of the set-up procedure to ensure that what is intended to be transmitted will in actual fact be what will be received. This brings rise to the next mode of operation. That being Mode 2, "SMT Run & Verify".

- ***Mode 2:*** **"SMT Run & Verify"**

  In Mode 2 of operation the Software Configuration Tool must verify the correctness of the set-up procedure in Mode 1, and also perform tests to

obtain performance results on each input and its corresponding receiver output. The Verification phase of Mode 2 shall simply involve the sending of appropriate signals to the inputs of the SMT and verifying that they have been received at the receiver. A report must be produced to show the test method, test signals at the transmitter inputs, and the test signals received at the receiver, and any other information directly relevant to the Verification phase.

Once the Verification Phase has been completed and is successful then the user may wish to apply certain test signals to the inputs of the transmitter and view the received signals in real time on the computer for performance related purposes. Details of this Test Phase of Mode 2 will be investigated at a later date. Once the SMT set-up has been verified and tested to comply with the requirements of the user, a detailed report needs to be generated that includes all the information relating to the initial requirements of the user, set-up and verification results of Modes 1 and 2.

- *Mode 3:* **"System Options Set-Up for System Expert"**

  Mode 3 simply contains special set-up options to configure the Hardware Test-Bed for special Engineering purposes. This mode should not relate to the needs of the user but merely for the Telemetry expert to set-up other interesting features on the Hardware Test-Bed Interface. Possible features have yet to be determined. This is left to the discretion of the Engineer undertaking this project.

2.2.2. Hardware Test-Bed Interface

The Hardware Test-Bed Interface shall be a physical interface between the PC environment that runs the Software Configuration Tool and the SMT transmitter and Receiver. This interface shall provide the correct connections between the physical elements concerned, and also contain the necessary electronics to perform the various functions depicted in the three modes of operation of the Software Configuration Tool.

The Hardware Test-Bed Interface shall meet the interface requirements of the SMT's concerned and also the interface requirements of a PC, and the relevant communication protocols associated with each of these.

The Hardware Test-Bed Interface shall be robust, small in size and easy to be handled in field environments. Field Environments can span various domains from military, medical down to home applications.

The Hardware Test-Bed Interface shall be built with custom off the shelf hardware. Cost of design and parts shall be kept to a minimum as possible.

3. Introduction to Subminiature Frame Telemetry

In this chapter, first an introduction to telemetry in general, and frame telemetry in particular will be given. It is followed by a description of the device used in this project, the Harris 3003274. Because this device is highly integrated and therefore very small, we speak about Subminiature Telemetry (SMT). The description is split into two parts: The physical characteristics and the programming of the EEPROM of the device.

3.1. Introduction to telemetry

The Federal Communications Commission (FCC) defines telemetry as the "use of telecommunication for automatical indicating or recording measurements at a distance from the measuring instrument". Telemetering systems are used in many environments, like spacecraft sending data to Earth, control of satellite communications systems for voice and video communications, monitoring body signals of sick persons in hospitals and missile and aircraft testing. Figure 2 shows the basic structure of a telemetry system.

In most cases data from multiple measurements has to be sent through one channel. This calls for a multiplexing method. Well-known multiplexing methods include Frequency Division Multiple Access (FDMA), Time Division Multiple Access (TDMA) and Code Division Multiple Access (CDMA). In the first case, the bandwidth of the channel is divided into subbands, in which data can be sent. In TDMA measurements are sent after each other. In CDMA, the data is coded in such a way that the correlation with the other coded data is zero. This way, multiple signals can be sent simultaneously in the same bandwidth.

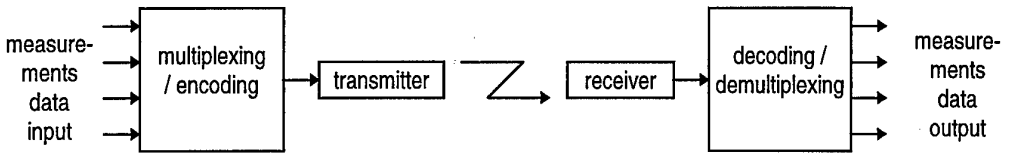


Figure 2. Telemetry system overview

Nowadays data is usually sent over the communications channel as digital data using Pulse Code Modulation (PCM). Analog measurements are converted to a digital equivalent. The multiplexing method that fits best to digital data channels is TDMA. For the receiver to be able to demultiplex the data, the data must be sent in a structured way. Data from multiple sources is packed in a frame. If the frames are transmitted continuously, we speak of **frame telemetry**. Another method of structuring data is to generate packets of data and send them when they are available, much like the format used for most computer-to-computer communication. This is called **packet telemetry**.

In this project a telemetry device manufactured by Harris Corp. will be used. This device is a complete telemetry package in itself. While the size of the device is very small (2" x 2" x 0.25"), it features 4 analog, 8 digital and 1 serial inputs, generates a

telemetry frame, encodes it and modulates it onto a carrier frequency generated internally. All of these functions are fully programmable.

3.2. Frame telemetry

The **Inter-Range Instrumentation Group** (IRIG) has defined two frame formats for telemetry purposes. These are called Class I and Class II specification. They are illustrated in **Table 1**. The whole frame is called the major frame. This major frames consists of a number of minor frames. In Table 2, the minor frames are shown as rows. In Class I, the minor frames are of a fixed length. This length is divided into a number of fixed-length words, containing the telemetry data. At the start of each minor frame a synchronization word is required. This allows the receiver to determine which field contains which data.

Table 1: IRIG Class I and II telemetry

Parameter	Class I specification	Class II specification
Data bits/words per minor frame	≤8192 bits or ≤512 words	≤16384 bits or ≤512 words
Minor frame length	Fixed	Variability allowed
Fragmented words	Not allowed	Allowed
Format changes	Not allowed	Allowed
Asynchronous formats	Not allowed	Allowed
Bit rate	>10bps	> 5Mbps
Independent subframes	Not allowed	Allowed
Supercom spacing	Uniform in minor frame	"Even as practical"
Data format	Unsigned binary / Complement binary	Others allowed
Word length	4 to 16 bits	16 to 64 bits

In a typical application, not all measurements have to be done at the same rate. To accomodate this and still use bandwidth efficiently, the IRIG standard allows supercommuted and subcommutated signals.

Commutated signals are sent each minor frame, and their position within the minor frame is fixed. The sampling rate of these signals therefore equals the minor frame rate. Examples in Table 2 of these signals are the field labeled 1, 2, 3 and n-1.

Supercommutated signals are sent more than once per minor frame. Therefore, a higher sampling rate is possible. Usually, the fields are positioned in the minor frame in such a way that the time difference between each occurrence of the field is the same (Class I requires this). An example of this are the Supercom-*n*-fields in Table 2.

Subcommutated signals are sent less than once per minor frame. (Without subcommutated signals, there would be no reason to have more than one minor frame per major frame). Subcommutated signals are put in a subframe, shown in Table 2 with a grey background. Each signal appears once per major frame. Example: i+1

Super-subcommutated signals are signals sent in a subframe, but appearing more than once per major frame. An example of this is field i in Table 2.

Table 2. IRIG telemetry frame format

Sync Word	1	2	3	...	Supercom1	SFID=1	i	...	Supercom2	n-1
Sync Word	1	2	3	...	Supercom1	SFID=2	i+1	...	Supercom2	n-1
Sync Word	1	2	3	...	Supercom1	SFID=3	i	...	Supercom2	n-1
Sync Word	1	2	3	...	Supercom1	SFID=4	i+3	...	Supercom2	n-1
Sync Word	1	2	3	...	Supercom1	SFID=5	i	...	Supercom2	n-1
Sync Word	1	2	3	...	Supercom1	SFID=6	i+3	...	Supercom2	n-1
Sync Word	1	2	3	...	Supercom1	SFID=7	n	...	Supercom2	n-1
Sync Word	1	2	3	...	Supercom1	SFID=8	i+4	...	Supercom2	n-1

In Class I Telemetry, subframes are synchonized with the major frame. That means a subframe starts when a major frame starts. To allow the receiver to distinguish between the minor frames, a field called **subframe identifier** (SFID) is used. This is a counter that is increased or decreased every minor frame, and is reset when another major frame starts.

In Class II Telemetry, there are much more variations allowed than in Class I. For example, it is allowed to have subframes that are not synchronized with the major frame. Then the subframe must have its own synchronization method. It is also allowed to fragment a data word. All fragments have to be transmitted within the same minor frame. **Table 1** sums up the specifications of the two classes.

The synchronization word, with which each minor frame begins, is a fixed bit pattern of 8 to 33 bits. IRIG has compiled a table of optimal bit patterns that can be used. The patterns are optimal in the sense that the chances are minimal that the receiver locks onto the wrong words. The optimal patterns are listed in Table 3.

Table 3: IRIG recommended synchronization patterns

length	pattern	length	pattern
16	111 010 111 001 000 0	25	111 110 010 110 111 000 100 000 0
17	111 100 110 101 000 00	26	111 110 100 110 101 100 110 000 00
18	111 100 110 101 000 000	27	111 110 101 101 001 100 110 000 000
19	111 110 011 001 010 000 0	28	111 101 011 110 010 110 011 000 000 0
20	111 011 011 110 001 000 00	29	111 101 011 110 011 001 101 000 000 00
21	111 011 101 001 011 000 000	30	111 110 101 111 001 100 110 100 000 000
22	111 100 110 110 101 000 000 0	31	111 111 100 110 111 110 101 000 010 000 0
23	111 101 011 100 110 100 000 00	32	111 111 100 110 101 100 101 000 010 000 00
24	111 110 101 111 001 100 100 000	33	111 110 111 010 011 101 001 010 010 011 000

3.3. The Harris 3003274 SMT device

The Harris SMT device is a complete telemetry package. While the size of the device is very small (2" x 2" x 0.25"), it features 4 analog, 8 digital and 1 serial inputs, generates a telemetry frame, encodes it and modulates it onto a carrier frequency generated internally. All of these functions are programmable. The device is available is two versions: an FSK (Frequency Shift Keying) and a QPSK (Quadrature Phase Shift Keying) version.

3.3.1. Structure of the SMT device

A block diagram of the QPSK version of the device is given in Figure 3. Each part of the device will be described below. The transmitter section will be described only briefly because this project does not deal with its characteristics.

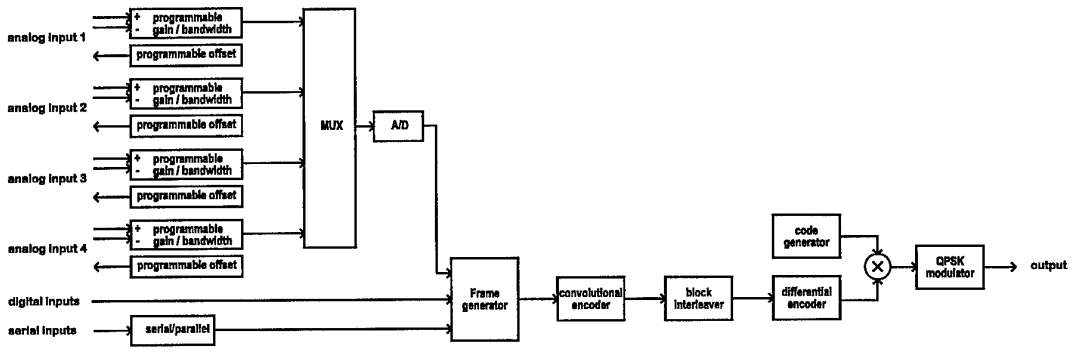


Figure 3. SMT device block diagram

- **Analog signal conditioning**

  Each analog channel features a differential input, programmable offset and gain, and a programmable anti-aliasing filter. The analog inputs are differential inputs. The offset can be applied by connecting the voltage offset output from the device to the negative input. What remains then is a single-end input. So the user can choose either the offset capability or a differential input.

  Possible offset values are: -2.5 V - 2.5 V in steps of 0.3125 V. A value of 0 V, however, is not possible. But if no offset is needed, the negative input can be connected to ground instead of to the offset output voltage.

  Possible gain values are: 1, 2, 4, 8, 16, 32. The accuracy of the gain stage is 0.4%.

  This filter consists of a discrete-time filter (using switched-capacitor techniques) functioning as an anti-aliasing filter for the A/D convertor, preceded by a continuous-time filter which provides anti-aliasing for the discrete-time filter. Unfortunately, the documentation of the device doesn't specify the order of the filters.

  Possible anti-aliasing filter cut-off frequencies are: 122 Hz, 244 Hz, 488 Hz, 976 Hz, 1953 Hz and 3906 Hz. The accuracy of the cut-off frequency is 1%.

- **Multiplexer**

  The multiplexer selects an analog channel from the inputs. Because all the data words are time-multiplexed in the frame structure anyway, the usage of only one A/D converter is no drawback. However, due to setup times it is not possible to transmit two analog values after each other. So in the frame structure an analog channel has to be followed by a non-analog channel.

- **A/D converter**

  The A/D converter is of the successive-approximation type. It has an input range of -2.5 V to 2.5 V. The resolution is 8 bits, the linearity 1/2 LSB and the accuracy 3% of full-scale.

- **Frame generator**

  The frame generator is fully programmable. As only data words of 8 bit are generated, the frame generator is programmed byte by byte. This implies that the synchronization word must be either 16, 24 or 32 bits long, and that the SFID field must be either 8 or 16 bits long. When programming the table that the device traverses to generate the frame, bytes with value <128 are taken as channel addresses, while byte >=128 are taken as literal values to be transmitted. Because sync words and the SFID field have to be programmed as literal values, each 8-bit part of the sync words and SFID field has to start with a "1"-bit. This limits the choice of sync words. Especially the IRIG recommended syncwords of length 24 and 32 are not possible to program.

- **Convolutional encoder & block interleaver**

  The convolutional encoder uses Viterbi R=1/2, k=7 encoding. This means the data rate is increased by a factor of two. The available data rate is therefore reduced by a factor of two when the user chooses to use Viterbi encoding. The block interleaver is used to equalize the spectrum.

- **Differential encoder**

  This encoder encodes the bit stream using the well-known Manchester encoding technique.

- **Code generator and multiplier**

  The code generator generator a pseudo-random bit sequence with which the data stream is multiplied. This generates a spread-spectrum signal. The receiver side should use the same pseudo-random generator seed. A maximum of 12 SMT devices can operate using the same frequency provided they are programmed with different seeds that cause bit sequences having a cross-correlation function equalling 0. The seed is 11 bits long.

- **QPSK modulator**

  The QPSK modulator encodes the resulting data stream on a carrier frequency generated internally. The carrier frequency can be set from 2320 Mhz to 2380 Mhz in steps of 10 Mhz.

3.3.2. SMT device connections

All connections to the SMT, except for the transmitter output, are made with one connector. This is a 37-pin subminiature D-connector. On the SMT device, the gender of the connector is female.

Table 4 lists the pin assignments. The connections can be divided in six categories: Analog, Digital, Serial, Setup, Control and Power. The first three are inputs for the measurements (and the auxillary outputs Serial Clock and A1-A4 Voltage Offset). The Setup signals are used for programming the device. The Control signals are used for setting run-time parameters: The transmitter is switched on by tying the pin Transmitter On (33) to ground, and a beacon signal can be switched on by providing a TTL high level to the pin Beacon Enable (3). The beacon signal allows a receiver to determine the position of the device, and causes only a moderate current drawn from the power supply (much less than while transmitting). The power connections are used to connect the two supply voltages (a positive and a negative) to the SMT device. These voltages must both be between 5.2V and 7.4V. The device draws a maximum of 400mA from the positive supply and 40mA from the negative supply. When the transmitter is switched off however, the current drawn is much lower.

Table 4: SMT device pin layout

pin	name	group	type	pin	name	group	type
0	Setup Done	Setup	TTL out	19	Setup Data	Setup	TTL in
1	A3 -	Analog	Analog in	20	Ready for Data	Setup	TTL out
2	Setup Data Clock	Setup	TTL in	21	A2 +	Analog	Analog in
3	Beacon Enable	Control	TTL in	22	D7	Digital	TTL in
4	A4 +	Analog	Analog in	23	A3 +	Analog	Analog in
5	D1	Digital	TTL in	24	D3	Digital	TTL in
6	D5	Digital	TTL in	25	D4	Digital	TTL in
7	A2 -	Analog	Analog in	26	D2	Digital	TTL in
8	D0	Digital	TTL in	27	A1 +	Analog	Analog in
9	Serial Clock	Serial	TTL out	28	D6	Digital	TTL in
10	Serial Byte Data Input	Serial	TTL in	29	A1 Voltage Offset	Analog	Analog out
11	A1 -	Analog	Analog in	30	A2 Voltage Offset	Analog	Analog out
12	A3 Voltage Offset	Analog	Analog out	31	Signal Ground		
13	A4 -	Analog	Analog in	32	A4 Voltage Offset	Analog	Analog out
14	Daisy Out	Control	TTL in	33	Transmitter On	Control	
15	MCM TEMP	Control		34	Negative Supply Return (+)	Power	
16	Negative supply (-)	Power		35	Positive Supply (+)	Power	
17	Positive Supply (+)	Power		36	Positive Supply Return (-)	Power	
18	Positive Supply Return (-)	Power					

Nota Bene: When the Signal Ground (31) is connected to the Power Supply Return pins (18, 34, 36), the current limiting circuitry inside the device is rendered inoperational. It is therefore advisable to use separate power supplies for the device and the surrounding circuitry.

3.4. SMT device EEPROM format

This section describes how the SMT device interprets the contents of its EEPROM. The programming of the SMT device EEPROM is generally done by a program called ITPDL.EXE which is provided by Harris Corp. The connection to the SMT device is made using a standard PC parallel port. The input ITPDL needs a file in ASCII text format. In this file, each line should consist of two decimal values, optionally followed by text. The text is ignored by the program. This makes it possible to add comments to the file, making it more readable for humans. The first value on each line is the EEPROM address, and the second value is the data to program into that location.

The file consists of two parts:

- **Command part**

  The command part contains instructions that the SMT device performs when it is powered up. The instructions involve setting up transmission parameters, analog channel conditioning and scantable length.

- **Scantable**

  When the SMT device had been put in run mode, the SMT device stops executing commands, and starts to scan the scantable and generates output bytes according to the table entries. This is how the telemetry frames are generated.

3.4.1. Command part

Commands consists of an opcode (operation code) and parameters. The 3 most significant bits of the first byte of a command form the opcode. This allows for 8 opcodes. The remaining 5 bits can be used as parameters. Also, there can be more bytes following the first byte, providing more parameter space. The opcodes are:

- **0**

  This opcode takes the 3 LSB's and programs them as the 3 MSB's of the spread-spectrum pseudorandom generator seed (PN seed). It takes a second byte as parameter and programs that as the 8 LSB's of the PN seed.

- **1**

  This opcodes takes three bytes as parameter and sends them to the FSK/QPSK modulator and carrier frequency generator. When the first parameter byte is 0x80, the carrier frequency is set. When the first parameter byte is 0, the reference word is set. The second and third byte function as one 16-bit parameter (MSB first, then LSB).

  For FSK versions, the carrier frequency equals the second parameter / 8 [Mhz], and the reference word should be 0x103.

  For QPSK versions, the carrier frequency equals the second parameter, and the reference word should be 0x23.

- **2**

  This opcode takes the 4 LSB's and programs them into the Single Bit Setup Register. This registers controls the bitrate, encoding and modulation type:

parameter	setup
0x0	200kb/s FSK, no Viterbi coding / block interleaving
0x4	200kb/s FSK, Viterbi coding / block interleaving
0x3	2Mb/s FSK, no Viterbi coding / block interleaving
0x7	2Mb/s FSK, Viterbi coding / block interleaving
0x8	200kb/s QPSK, no Viterbi coding / block interleaving
0xC	200kb/s QPSK, Viterbi coding / block interleaving

- **3**

  The 3 LSB's are sent to the ADAC (A/D converter) as an opcode. The second byte provides the ADAC address, while the third byte provides the ADAC data.

- **4**

  The 3 LSB's are sent to the ADAC (A/D signal conditioner) as an opcode. The following byte is sent as ADAC address.

- **5**

  This opcode puts the SMT device is run mode. It takes two bytes as parameters, and uses this 16-bit value as the scantable start address. The scantable end address is always 8191. After reaching that value, the scantable adress counter wraps back to the start address.

  After this command, the SMT device stops processing command bytes and starts its normal operation.

- **6 & 7**

  These opcodes are unspecified.

3.4.1.1.ADAC opcodes:

- **0**

  Initialize ADAC. This opcode needs to be sent twice, with an adress parameter of 0, to reset the ADAC.

- **1**

  Program offset:

Param.	Offset	Param.	Offset	Param.	Offset	Param.	Offset
0	0.3125 V	4	1.5625 V	8	-2.5 V	0xC	-1.25 V
1	0.625 V	5	1.875 V	9	-2.1875 V	0xD	-0.9375 V
2	0.9375 V	6	2.1875 V	0xA	-1.875 V	0xE	-0.625 V
3	1.25 V	7	2.5 V	0xB	-1.5625 V	0xF	-0.3125 V

- **2**

  Address program. Each time this command is sent, the next channel of the ADAC is programmed with its address (given as parameter to his command). This opcode should be sent after initializing and before gain/offset/filter setup of a particular channel. The address is used when programming gain/offset/filter properties and in the scantable.

- **3**

  Program bandwidth of anti-aliasing filter (200kb/s version):

Parameter	Cut-off frequency
2	122 Hz
3	244 Hz
4	488 Hz
5	976 Hz
6	1953 Hz
7	3906 Hz

- **4**

  This opcode is unspecified

- **5**

  Program gain:

Parameter	Gain
0	Power Down
1	1
2	2
3	4
4	8
5	16
6	32
7	Invalid

- **6**

  Reset COA function (it is not specified what the COA function is).

- **7**

  This opcode is unspecified

3.4.2. Scantable part

A byte in the scantable is interpreted as a channel address when the most significant bit is 0, and taken as a literal value when the most significant bit is 1. The channel addresses that can be used are 0, indicating the serial input, 1, indicating the digital input, and any of the analog channels addresses, which are determined by the user using the ADAC Address Program commands (opcode 2). The maximum size of the scantable is 7168 bytes. Note that it is the user's task to take care of synchronization / subframe identifier fields. These fields are implemented by putting literal values in the scantable.

4. Overview of the SMT Configuration Tool

4.1. Overview of the configuration system

The structure of the SMT configuration system is depicted in Figure 4. The user sets up the SMT system using the software configuration tool. The software configuration tool determines the necessary configurations of the three hardware components and configures those components through the PC hardware interface. In the following sections the components that make up the system are described in short.

- **Test signal generator**

  The test signal generator is used to provide signals to the SMT transmitter inputs in order to test whether the SMT transmitter and receiver are setup properly. The requirements of these signals vary with the SMT transmitter setup. Therefore, the test signal generator must be programmable by the configuration tool.

- **SMT transmitter**

  The SMT transmitter integrates of a data acquisition system providing analog and digital inputs and a transmitter. All of its functions are programmed through the PC hardware interface.

- **SMT receiver**

  The SMT receiver is able to receive signals from multiple SMT transmitters. In this configuration system only one transmitter is used at a time. The SMT receiver is used to verify the correct operation of the SMT transmitter. It is configured though a serial interface connected to the PC hardware interface.

- **PC hardware interface**

  The PC hardware interface forms the connection between the software and the hardware components of the system. On the PC side, it connects to a standard parallel port. Therefore there are no special requirements on the PC.

- **Software configuration tool**

  The configuration tool enables users without expertise on telemetry systems to program the SMT transmitter. This allows for easy configuration. The software can also generate reports on the actions performed for easy documentation.

4.2. Overview of the SMT configuration software

The software configuration tool can be divided in functional parts, shown in Figure 5. In the following paragraphs, these parts and their interaction are described.

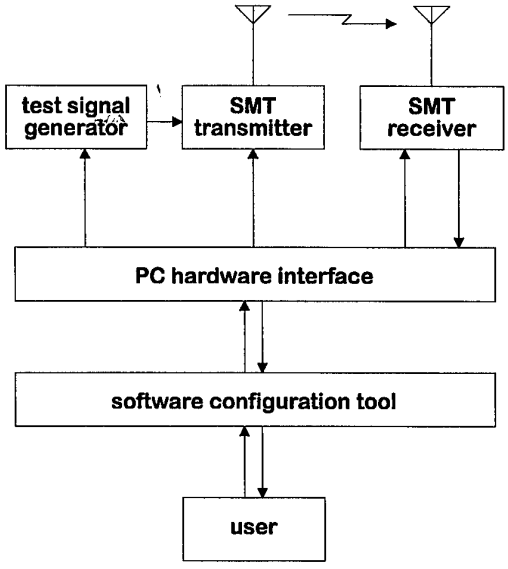


Figure 4: System overview

- **SMT Device Specification database**

  The information about the device we want to configure is contained in this database. The reason to use this database is to allow other device types to be added in later, without rewriting the software. The information can be divided into two categories: the device capabilities, and how to setup the device.

- **SMT configuration**

  This part of the tool comprises a graphical user interface to enter an SMT configuration. Information from the databases described above helps the user to enter this information quickly and reliably.

- **Storage system**

  Complete configurations can be saved to and retrieved from disk.

- **SMT configuration file generation**

  From the information entered by the user, an SMT configuration file is generated. This file describes the contents of the EEPROM of the SMT device. As this file conforms to the specification by Harris Corp., it allows the use of the program ITPDL, written by Harris Corp., to program the SMT devices.

- **SMT programming**

  This part consists of the ITPDL program. It programs the SMT device using a PC parallel port according to the configuration file. Regrettably this program cannot provide feedback about the results of the programming to the SMT configuration tool. It only reports to the user, on the screen. Therefore, the results of the programming cannot be included in the report that can be generated automatically (see below).

- **Test signal generation, test signal generator setup**

  For each configuration, test signals are defined that allow the user to verify each aspect of the SMT programming, for instance the analog input range and filter cut-off frequency. When the verification starts, the test signal generator is configured through the PC hardware interface.

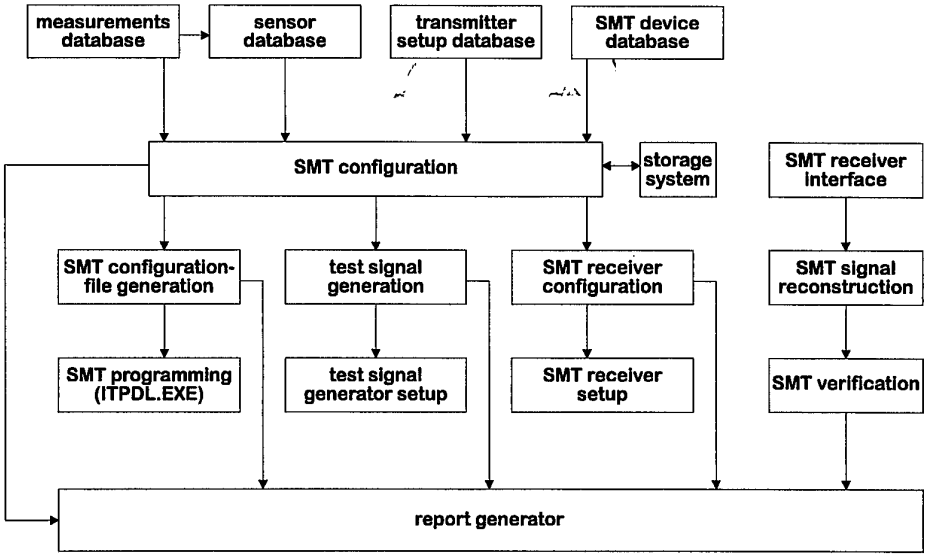


Figure 5: Software configuration tool overview

- **SMT receiver configuration, SMT receiver setup**

  The receiver setup must of course match the transmitter setup. If the receiver is directly connected to the PC the configuration can be done automatically. If the receiver is a stand-alone system, the user can use the information in the report to configure it properly.

- **SMT receiver interface**

  The receiver interface is the driver which reads data from the SMT receiver through the PC hardware interface.

- **SMT signal reconstruction, SMT verification**

  From the frames received by the receiver interface, the individual analog and digital values are retrieved. Then, the signals received are verified to match the signals generated by the test signal generator.

- **Report generator**

  The report generator is an important part of the system. It produces a report about all of the configuration settings. The report is created in Microsoft Word using OLE (object linking and embedding). The reason to do this is the rich feature set of Word, and the fact that Word is the mostly used word processor on PCs. It enabled users to easily integrate the report in a larger document.

5. Design of the SMT Configuration Tool

5.1. Hardware

5.1.1. Programming interface

Because the programming of the SMT Device is performed by the program ITPDL.EXE provided by Harris Corp., the interface hardware has to conform to Harris' specification. This specification is depicted in Figure 6. The schematic needs little explanation. The "done"-LED should light up when the programming of the device is completed. The "daisy-chain"-LED should be off when the "On"-switched is closed. In the actual design, two AND gates are put between D0/D1 and Setup Data Clock/Setup Data. This allow the user to turn off the power of the SMT device and remove all signals without shutting off the PC.

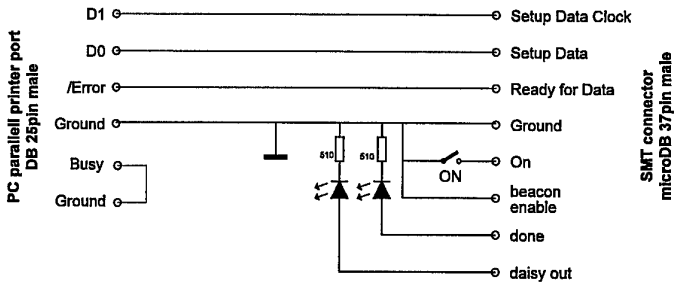


Figure 6: SMT Interface specification

5.1.2. Test signal generator

The test signal generator has to provide the signals needed to test whether the SMT device is setup and programmed properly. Of the analog channels, gain, offset and low-pass filter characteristics should be tested. To implement this, a square-wave generator was designed with programmable frequency, gain and offset.

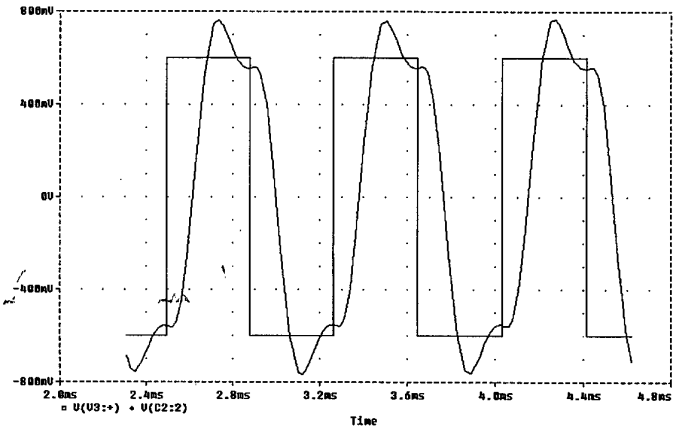


Figure 7: Low-pass filter input and output, $f_{-3dB}=3f$

The frequency of the generator is set to approx. 1/3 of the filter cut-off frequency. Then the 3rd harmonic of the square-wave is still there after the filter (attenuated 3dB), while higher-order harmonics are significantly reduced in amplitude. At the receiver side, this should be clearly visible. If the filter setting is not set correctly, either the 3rd harmonic disappears (cut-off frequency too low), or higher-order frequencies appear (cut-off frequency too high). Because the transfer function of the low-pass filter is not given (only the -3dB frequency), the output cannot be predicted in what it will look like. To get some idea of the shape of the output waveform, a plot was made of a square wave of 1.3 kHz, and the same signal filtered by 4th-order Butterworth filters with -3db frequencies of 3.9 kHz and 6.5 kHz (Figure 7 and Figure 8, resp.). The 5th harmonic is clearly visible

in Figure 8. The amplitude of the signal is set to 0.6 times the range of the analog input. One reason for this is the fact that after filtering the voltage span of the signal gets larger (see Figure 7 and Figure 8). Another reason is to allow for some tolerance in the signal generator, allowing cheaper components to be used without the need for calibration.

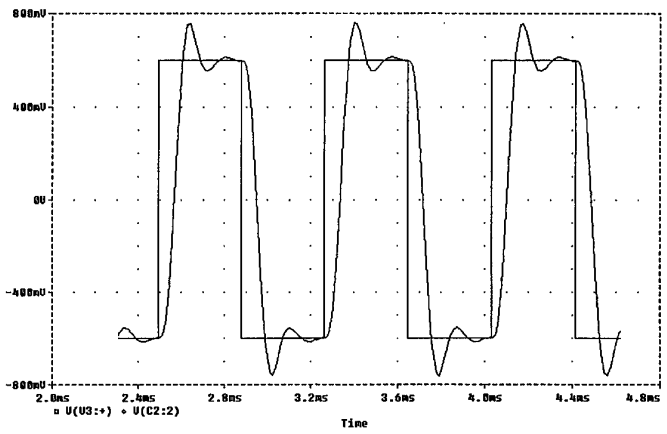


Figure 8: Low-pass filter input and output, f_{-3dB}=5f

Because the SMT device has a very limited set of possible settings (16 offset values, 6 gain values and 6 cut-off frequencies), the test signal generator doesn't need complex circuitry. Is is therefore possible to build it with standard, off-the-shelf components, allowing a low-cost design.

Because the SMT device has 4 input channels, four test signal generators should be provided. To keep the circuit simple, we use one test signal generator which is time-shared by the four channels. Every second the PC switches the output of the test signal generator to another analog input. The other analog inputs receive 0 V input.

The complete circuit diagram is depicted in `Figure 9. A discussion of the function of all components appears in the following section.

5.1.2.1.PC interface

All of the functions, except switching on/off the SMT device power supply, are setup by the PC though its parallel port. For this we need 12 bits (3 to select the frequency, 3 to select the gain, 4 to select the offset and 2 to select the channel). Because there's 12 output bits on the PC parallel port, of which 2 are taken by the SMT program interface, we need to extend the number of output bits. This is done by a shift register (IC5). The software provides data signals on D5 and clock signals on D6. An advantage of using an extra IC here is the fact that the output characteristics of IC5 are well-known. This allows a 4-bit D/A convertor to be built with a simple 4-resistor network (R8-R11). This can't be done reliably on the parallel port output because signal levels and output impedance aren't specified precisely enough (the only specification you can rely on is that the output signals are TTL compatible).

IC1a and IC1b provide a means for the software to check whether the hardware is connected to a parallel port (and to which parallel port).

5.1.2.2.Analog test signals

The test signal frequencies are generated by IC1c and IC2. IC1c generates a signal of 10.4 kHz. IC2 divides this frequency to 1.3 kHz, 650 Hz, 325 Hz, 163 Hz, 81 Hz, 41 Hz, 20 Hz and 10 Hz. One of these is selected by IC3, a multiplexer. After this, the DC component of the signal is removed by the high-pass filter formed by R1-R6 and

C10. The cut-off frequency of this filter is 0.85 Hz. Because the lowest possible signal frequency is 20 Hz, this is sufficient. The gain selection circuit is built around R1-R6 and IC4. Multiplexer IC4 selects the signal attenuation by connecting the input of IC6a to a point in the resistor ladder. IC6b buffers the signal, which is then converted to a current by P2. The inverting input of IC6c provides a virtual ground. Therefore, the signal current generated by IC6b / P2 is independent of R8-R11, P3 and R20. To the signal current the offset current is added. This offset current is generated by 4-bit D/A converter IC5 / R8-R11. Because we need positive as well as negative offset currents, the DC level of the offset is adjusted to 0 V by P3. The signal current plus offset current is then converted to an output voltage by IC6c / R20.

The component values follow from:

$$V_{out} = -R_{20}\left(n\frac{4.7V}{30k\Omega} + A\frac{V_{signal}}{P2} - \frac{4.7V}{P3}\right)$$

Where $n \in \{0,1,...,6,7,9,10,...,16\}$, $A \in \{1,\frac{1}{2},\frac{1}{4},\frac{1}{8},\frac{1}{16},\frac{1}{32},0\}$ and V_{signal} is a square-wave with an amplitude of 4.7 V / 2 = 2.35 V.

Note the omission of n=8, caused by $R8 = \frac{1}{9}R9$ instead of $R8 = \frac{1}{8}R9$. This conforms to the omission of 0V in the possible offset values of the SMT device.

Because n=8 corresponds to 0V offset, the DC level of the output voltage equals:

$$V_{out} = -R_{20}\left(8\frac{4.7V}{30k\Omega} - \frac{4.7V}{P3}\right)$$

This needs to be zero, so $P3 = 3.75k\Omega$.

The amplitude of the output needs to be 0.6*2.5V=1.5V in case the gain of the SMT device is 1x. Therefore, $P2 = R_{20}\frac{2.35V}{1.5V} = 3.125k\Omega$.

The resulting analog signal is connected to one of the analog input of the SMT device by IC8. R14-R17 ensure that the input voltages the inputs not currently connected to the signal generator output equal 0V. To enable the offset capabilities of the SMT device, the offset output voltages are connected to the inverting inputs of the SMT device.

5.1.2.3.**Digital test signals**

As digital test signals, the output from the frequency divider is taken. At the receiver side one can check the frequency of each digital input, or, if the digital inputs are presented as one byte, the value of that byte increases every $\frac{1}{1.3kHz} = 0.769ms$.

5.1.2.4.Serial test signal

Since access to documentation on the serial input was not available, it was not possible to define a meaningful test signal. Therefore, the serial input was connected to the first digital input, D0. One might be able to check the serial input function by comparing the serial channel values with those of D0.

5.1.2.5.Power supply

The test signal generator itself needs a +5v supply for the digital IC's and a positive and negative supply for the analog output. The SMT device needs a +5.2 - 7.4V at 400mA and -5.2 - 7.4V at 40mA supply. The power supply circuit is set up to provide +6.5V, +4.7V, -4.7V and -6.5V, and works as follows: If an external power adaptor is used, only the circuit of IC10 with surrounding components is used. This circuit is well-known. A bridge rectifier (D1) and a buffer capacitor (C2) are provided to be able to connect a DC as well as AC power adaptor, and to eliminate the need to pay attention to the polarity of the input voltage.

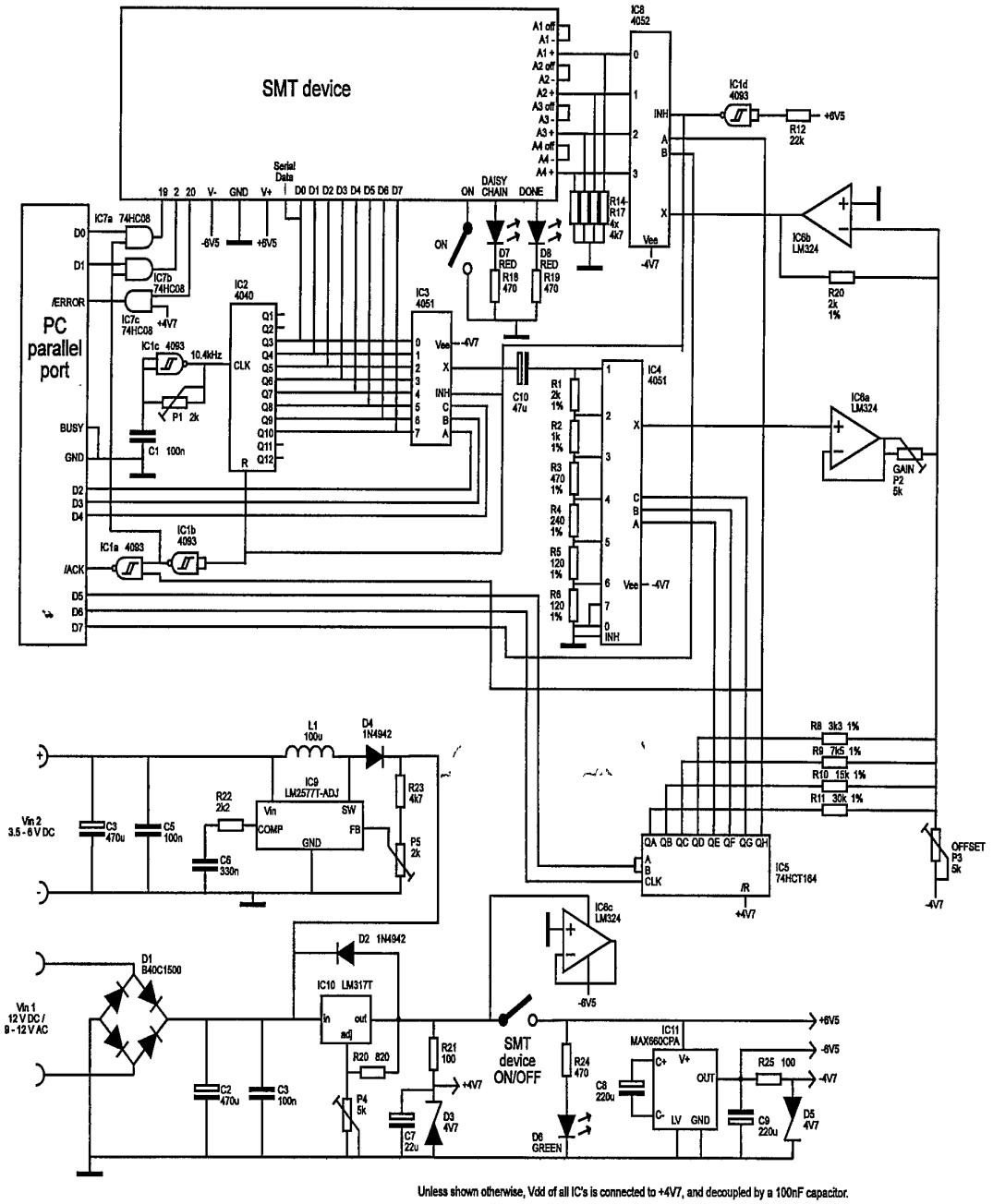


Figure 9: Test signal generator

If power is drawn from the PC keyboard connector, the output voltage has to be higher than the input voltage. Therefore a switching power supply is needed. This is built using IC9 and surrounding circuitry. IC9 is a simple-to-use, almost completely integrated switching power supply. It works as follows: IC9 turns on and off its output switch, which is connected between SW and GND, at a frequency of 52 kHz. When the switch is closed, energy is accumulated in inductor L1. When the switch opens, the side of the inductor connected to D4 flies above the input voltage (this is caused by the negative dI_{L1} / dt of the inductor), thereby charging C7. IC9 regulates the output voltage by changing the duty cycle of the output switch signal. The shorter the switch is closed, the less energy is accumulated in L1 and the lower the output voltage will be.

Because the switching power supply has no current-limiting circuitry, the output is connected to the voltage-regulating circuitry of IC10. Another advantage of this setup is that the power supplies to the SMT device contains less high-frequency interference generated by the switching. A disadvantage of this approach is the extra energy loss incurred by IC10. A higher current is therefore drawn from the PC. This current equals:

$$I_{\text{sup }ply} = \frac{I_{SMT} + I_{generator}}{\eta} \frac{V_{IC9,out}}{V_{IC9,in}}$$

Where $I_{SMT} \leq 450mA$ (400mA for the positive supply and 50mA for the negative supply). and $I_{generator} \leq 100mA$. η denotes the effiency of IC9, and is typically 80%, according to the datasheet. $V_{IC9,out} = 6.5V + 2V$ (IC10 needs 2V power drop) and $V_{IC9,out} = 5V$. Therefore, $I_{\text{sup }ply} \approx 1.2A$. PC power supplies are quite able to provide this amount of current.

Because power to IC10 is supplied through either D1 or D4, it is possible to have both power circuits connected at the same time, while only one is providing output power. This saves a switch and is easier for the user. From the 6.5V delivered by IC10, a 4.7V voltage is derived with R21 and D3. Because the components using this power require very little power, no further regulation is necessary. The negative power supply is derived from the positive supply by IC11, a charge-pump voltage inverter. Its main features are a voltage drop of less than 0.3V at an output current of 60mA and a conversion efficiency of 88% (typ.). Only two external capacitors are needed for the device to function. The -4.7V supply is derived from the -6.5V supply the same way as for the positive supply voltage.

When the SMT device is switched off, the negative voltages are also switched off because they're only needed for the SMT device and the analog test signal generator. Besides that, all signals to the SMT device are switched off so the device can be unplugged safely without turning off the PC. This is accomplished through IC7a-c, IC1b,d and R12/R13. When the SMT device is switched off, the input of IC1d drops to 0V (R12 is provided to protect the input against the 6.5V input, which is higher than the supply voltage of IC1d (4.7v)). The output of IC1d then disables the multiplexers IC3 and IC8, and resets IC2. Then all digital and analog inputs of the SMT device are 0V. Through IC1b and IC7a,b the programming interface pins of the SMT are set to 0V. IC7c decouples the SMT from a pull-up resistor that might be present on the /ERROR input of the parallel port.

Note that the signal ground is connected to the power supply return pins. The SMT device specifications discourages this because the internal current limiting circuitry doesn't work anymore. The reason for this action is because at the time the circuit was designed this information was not yet available. And changing the circuit afterwards would take too much time. The consequences are not too serious because the power supply circuit has a current limiting circuit of its own.

5.1.3. Receiver

Because a receiver nor the specification of one was available, it was impossible to integrate a receiver interface into the design. This means that the correctness of the SMT device setup cannot be verified by the configuration tool and has to be done manually.

5.2. Software

5.2.1. Visual Basic

Visual Basic is a programming language developed by Microsoft to provide programmers with a quick and easy method of developing Windows applications. It provides the programmer with an integrated environment where he can use tools to create a graphical user interface and use event driven programming techniques. A developer can quickly and easily create a user interface, then write the code to respond to specific events which occur as a result of user input. The integrated development environment allows you to attach code quickly to the interface created for each event which is applicable for any type of object on the interface.

Advantages of using Visual Basic are:

- Quick development of nice-looking programs.
- Easy integration with other Microsoft applications.
- Built-in extensive database capabilities.
- Easy creation of programs for both Windows 3.x and Windows 95.

Disadvantages of using Visual Basic are:

- The Basic language is a not so powerful language compared to C / C++.
- A large run-time library is required to run VB programs and must be distributed with the program.
- It's almost impossible to port programs to other platforms than Windows.
- Run-time efficiency is traded in for easy development. As a result, programs run slower and need more memory.

In this project, quick development was a must. Therefore, Visual Basic was chosen to program the SMT configuration tool.

The information the SMT software uses is managed by the Visual Basic Database Engine. Advantages of this approach are the possibilities of data exchange with other applications, good integration with Visual Basic controls and easy maintenance by

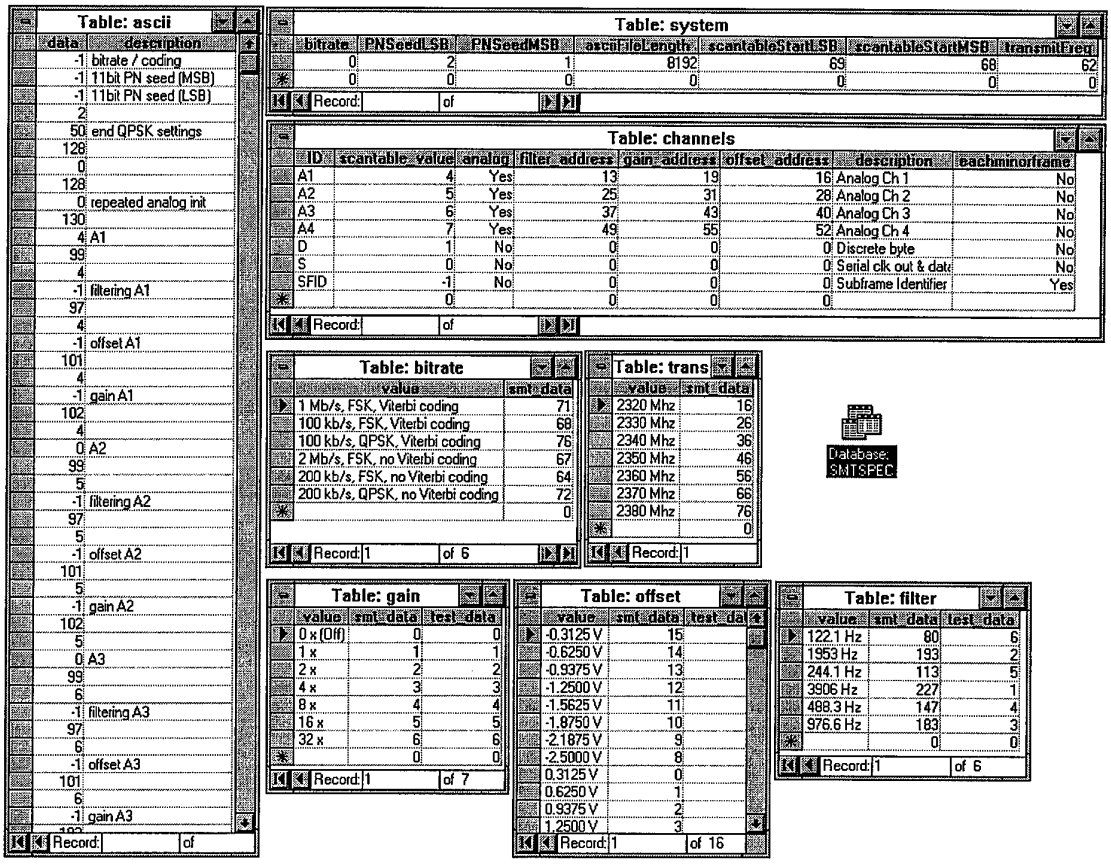


Figure 10: SMT Specification database

less experienced programmers. Furthermore, it allows rapid development without worrying about file formats and low-level file operations.

Disadvantages are the probably lower speed and an increase in the program size and required memory. Speed, however, is not a concern because there are only small amounts of data involved in the SMT setup. And because all database functions are contained in the Visual Basic run-time library, program size isn't a problem either.

5.2.2. SMT specification database format

To make is possible to program future types of SMT devices, the information about the device is put in a Microsoft Access database. When a new type of SMT device becomes available, the software doesn't need to be modified. Only the database has to be altered. However, some basic assumptions about the device programming structure have to be made. The database structure is depicted in Figure 10. The database consists of 8 tables:

- **ascii: ASCII file template**

  This table consists of the bytes that are the same in each configuration. Positions that have to be filled in afterwards are set to -1. The table also contains a description of each field, making the ASCII file more readable for humans.

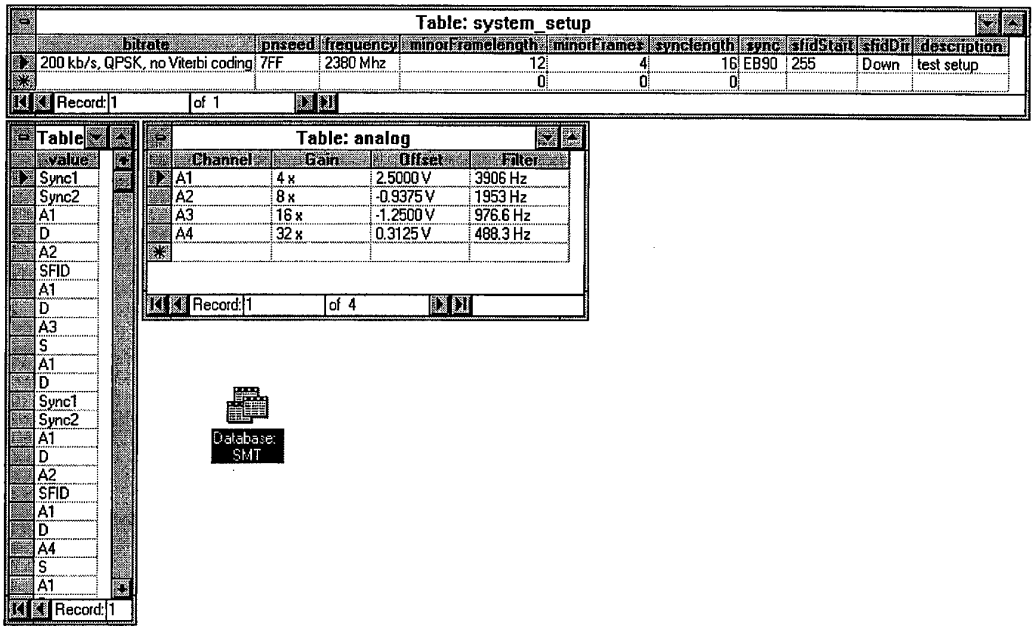


Figure 11: SMT Database format

- **system: System parameters**

  This table contains the ASCII file length, and the position in the ASCII file of the system parameters: bitrate, transmission frequency, PN Seed and scantable start address.

- **channels: Channel characteristics**

  In this table the channels supported by the SMT device are listed. Sync words are not listed, because they are a special case. The properties of each channel are: name, description, value to be put in the scantable, whether the channel is analog or digital, position of gain/offset/filter setting in the ASCII file (only applicable for analog channels) and whether this value should appear at the same position in each minor frame.

- **transmitfreq, bitrate, gain, offset, filter**

  In these table the possible values for each parameter are listed, along with the value to program the SMT device with that value. The Gain/Offset/Filter entries also have a test_data field, which is used to program the test signal generator.

The way the database is setup makes it easy to change the characteristics SMT device. You can add more channels, change transmission specifications, bitrates and so forth.

5.2.3. SMT specification format

It is possible to load/save the SMT configuration from/to disk. The Microsoft Access Database format was chosen to do this. This allows other programs (like Visual Basic applications or MS Office applications) to interact with this data. The database is depicted in Figure 11. The table 'system_setup' contains the system parameters line bitrate, PN seed etc. The table 'analog' contains the properties of the analog channels. The table 'scantable' contains the scantable in a one-dimensional way: All minor frames are put after each other. The reason not to use a two-dimensional format is that

is is more difficult and slower to add/remove fields when the frame dimensions are changed. Furthermore, it takes more effort to write software to interact with a database with a variable number of fields. That means it would be harder for external programs to interact with the database.

5.2.4. User interface

The user interface consists of the main editing window and some additional windows used for special functions. The main editing window allows the user to edit all of the SMT configuration properties. The most important additional windows are the Frame Generation window, the View Errors window, the SMT EEPROM file window and the Test Signal Generator status window.

A description of how to use the software is given in the next chapter. No description of the source code will be given, as it should be clear by itself. Where useful, some comments were included, so one should be able to understand / modify the code. The only exception is the frame generation code, which deserves a little more explanation.

5.2.5. Frame structure generation

The algorithm generating the frame structure should take as inputs the sampling frequency requirements of the channels, and produce on its output either the frame structure or an indication that the input requirements cannot be met.

The input sampling frequencies are in fact given as integers describing the period time of the sampling, expressed in the byte transmission time. For example, if the byte rate is 25kb/s and the desired sampling frequency is 1kHz, then the period time is 25. The reason the inputs are given like this is that a period time of n means that the channel value should be repeated every n bytes.

The frame structure generated always consists of only one minor frame. The reason why this is done is that this allows a more efficient use of bandwidth, because there are less sync words needed, and no subframe identifiers are needed.

Example: Consider the following two frame structures:

Sync1	Sync 2	A1	D	A2	SFID	A1	A3
Sync1	Sync 2	A1	D	A2	SFID	A1	S
Sync1	Sync 2	A1	D	A2	SFID	A1	A3
Sync1	Sync 2	A1	D	A2	SFID	A1	A4

Sync 1	Sync 2	A1	D	A2		A1	A3
		A1	D	A2		A1	S
		A1	D	A2		A1	A3
		A1	D	A2		A1	A4

The left frame structure consists of 4 minor frames, each consisting of 8 bytes. The right frame structure consists of 1 minor frame consisting of 32 bytes (displayed as an 8-by-4 matrix). Since the word rate is the same for both frames, it is clear that the sampling frequencies of each channel are the same for both frames. The ten empty cells in the frame on the right can be filled in with more data.

So if one minor frame is always preferable over multiple minor frames, one might ask himself why minor frames are used at all. The reason for this is that the Class I IRIG

standard prescribes a maximum minor frame length of 8192 bits or 512 words. For example a frame structure of 64x16 bytes can't be replaced by a 1024x1 structure.

The SMT device used in this project however only has 6 channels (4 analog, a digital and a serial). There's no need to use frame structures with as total length of more than 512 words. Even future devices with 8 or 16 channels would hardly need such large frame structures.

The frame structure generator generates every second frame field. It does this because the SMT device can't handle two analog channels directly after each other. So in between analog channels there can only be the digital channel(s), the serial channel(s) and the sync words. This provides so much bandwidth for the digital and serial channels that they are not included in the automatic frame structure generation.

To improve readability, in the rest of this section only the analog channels are mentioned, as if there are no other channels. The other channels get weaved in later.

The analog channel period times are refered to as $n_1,\ldots\ n_{N-1}$. There are N-1 analog channels. gcd() denotes the greatest common divisor of its arguments, and lcm() denotes the least common multiple of its arguments.

Because the IRIG standards require the spacing of each channel to be uniform within the minor frame, the minor frame length must be a multiple of n_i , i=1...N-1. Therefore, the minor frame length should be lcm($n_1,\ldots\ n_{N-1}$) or a multiple thereof. Using a multiple has no advantages, and it therefore not considered. Knowing the frame length, we know the period time of the sync word, for which a space should also be reserved. We denote the period time of the sync word by n_N, and treat the sync word as a normal channel. So we have N channels in total.

Now the remaining problem is how to fill the frame structure in such a way that all channels fit in without colliding. For two channels not to collide, it is required and sufficient that their period times share a common factor, gcd(n_1,n_2). For example, if the period times are 4 and 5, there will always be a collision no matter how you shift the cells. This is shown in Table 5.

Table 5: Collision of channels that do not share a common factor

A1					A1					A1					A1				
A2				A2				A2				A2				A2			
	A2				**A2**				A2				A2				A2		
		A2				A2				**A2**				A2				A2	
			A2				A2				A2				**A2**				A2

For more than two channels not to collide, their period times should at least share a common factor, gcd($n_1,\ldots\ n_N$). If this common factor is greater or equal to the number of channels, it fits. Otherwise, the channels should be divided among gcd($n_1,\ldots\ n_N$) groups. In every such group the same requirements apply, except that the period times should be divided by gcd($n_1,\ldots\ n_N$).

To make this description a little bit more readable, Figure 12 shows two examples on how channels can be fitted in. The first example shows how three channels fit in directly because their common factor is large enough. In the second channel, the common factor is 2 which is less than the number of channels. The channels are therefore divided into two groups. The first one only contains n_1, the second one contains n_2 ... n_4. In the second group, the common factor divided by the common factor of the previous group (2) equals 3. This equals the number of channels. Therefore it is possible to fit these channels into the frame structure.

Generally there can be a lot of ways to divide channels among groups. The algorithm implemented in SCANTBL.BAS (listed in Appendix A) tries all combinations. It tries the possibilities that are most likely to succeed (those with equal distribution over the groups) first. The algorithm uses the function "try_permutation" recursively to distribute channels among the groups, and the function "try_subtable" to check each group. When a valid distribution is found, the resulting positions of the channels in the frame structure are recorded in the array "offset". When the user accepts the results, the procedure "generateScantable" puts the values in the frame structure. The

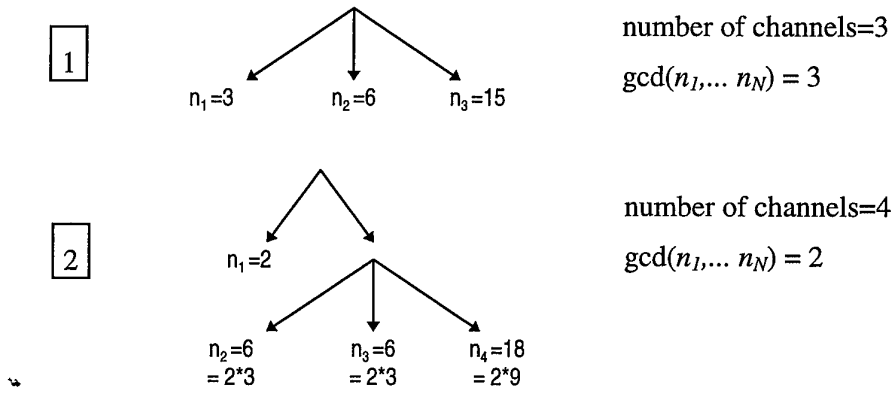


Figure 12: Examples of channel distribution

positions are shifted in such a way that the sync word is the first word.

6. Test and Evaluation of SMT Interface

Unfortunately during the initial testing and evaluation process conducted in Australia access to an SMT device and SMT receiver was impossible. The only testing that could be done was therefore the software, the test signals generated by the Test Signal Generator and the power supply. Later however, when testing was performed at Wright Laboratory (Eglin Air Force Base, Florida USA), the testing focussed on the software as the primary objective. It was agreed that the hardware would be left for someone else to test and modify.

6.1. Software

The software has been tested with different SMT device specifications, and different setups. Minor frame lengths of 3-512 have been tested and 1-256 minor frames per major frame. A bug still existing in the program is that the EEPROM file is generated within a string of maximum length 64 kbytes, and displayed in a textbox with a maximum string length of 32 kbytes. Because the average line length is approx. 25, this means that problems occur when the total frame length exceeds 1280 bytes. A solution to this problem may be writing the EEPROM file directly to disk instead of accumulating the text in a string.

6.1.1. Debugging

6.1.1.1. Emphasise Combo Box

The function of the emphasise combo box is emphasise on one data channel. That is it tries to re-arrange the frame into an N rows x M columns matrix such that the data channel to be emphasised is shown down one column. This function is only available when there is only one Minor Frame Per Major Frame. The TableWidth TextBox value would be modified (to the value M) to reflect on the change. The problems and solutions for this combo box are given in the table below:

No	Problem	Solution
1	The combo box does not offer a means to undo the emphasise command.	This was fixed by adding a new item in the combo box called "Normal", which reverses the emphasise function.
2	The program would crash when the user tried to modify the Scantable whilst in emphasise mode.	Fixed by not allowing editing whilst in amphasise mode.

6.1.1.2. Transmit Frequency Combo Box

The function of the Transmit Frequency Combo Box is to show the user the possibile frequencies available for configuration of the SMT device. The problems and solutions for this combo box are given in the table below:

No	Problem	Solution
1	The frequencies shown were out of order.	Fixed by setting the "sort" property of the ComboBox to true. Double listing of frequencies were also removed.
2	Depending on the modulation type (QPSK or FSK) chosen in the BitRate ComboBox, only certain frequencies were available to the Transmit Frequency ComboBox.	Fixed by using a simple number filter to select the frequencies required and reloading the Transmit Frequency ComboBox everytime the BitRate was changed.
3	As the user clears the Transmit Frequency ComboBox the BitRate ComboBox was changed. This was necessary as some frequencies were not allowed under QPSK and vise versa for FSK. The user might have chosen a correct frequency under QPSK, but is incorrect if the user decides to change the format to FSK. This offered inconvenience to the user.	Fixed by setting up a new database table to enter a default value in the Box. This allows the user enter their own default value by changing the value in the database table.

6.1.1.3. PN SEED Text Box

The function of the PN SEED Text Box is required when selecting modulation type of QPSK. The problems and solutions for this combo box are given in the table below:

No	Problem	Solution
1	Text Box is enabled for both types of modulation.	Fixed by enabling the edit option of the Text Box when QPSK is selected and disabled when FSK is selected.

6.1.1.4. Minor Frame Length Text Box

The function of the Minor Frame Length Text Box is to specify how many cells should be in each minor frame. The problems and solutions for this combo box are given in the table below:

No	Problem	Solution
1	The Text Box does not respond to the user after Enter is pressed. The program would only respond after the user clicks the mouse elsewhere on the screen.	Fixed by creating an event to respond to the "Enter" and "Tab" keys.
2	The Text Box cuts off data. This arises when a user decides to change the number in the Minor Frame Length Text Box when the frame is still under emphasis, the program assumes the original Minor Frame Length with only M columns, hence results in cutting off data in the second row onwards and appends the new cells from cell M onwards.	Before changing the length of the frame, check if Scantable is in emphasis. If it is then Normalise before changing length. Problem fixed.

6.1.1.5. Minor Frames Per Major Frame Text Box

The function of this Text Box is to specify how many minor frames are within the major frame. The problems and solutions for this combo box are given in the table below:

No	Problem	Solution
1	The text box does not respond to the user after Enter is pressed.	Fixed by creating an event to respond to the "Enter" and "Tab" keys.

6.1.1.6. SFID Column Width

The problems and solutions for this are given in the table below:

No	Problem	Solution
1	When the user specifies a column in the Scantable to be the SFID column, "SFID" is not shown in entirety due to the width of the cells.	Fixed by checking if the column is designated "SFID" and if so the cell width is adjusted.

6.1.1.7. Generate Report

The function of the generate report menu item is to activate the procedure which generates an MS Word 6 document detailing the configuration information entered in the current project. The problems and solutions for this are given in the table below:

No	Problem	Solution
1	If there is only one minor frame in the major frame, the code generates a two column grid for the scantable in the document. This is wasteful of paper, and unprofessional in the presentation. There was also redundant information printed in the docuement relating to the testing of signals. This is not required at present.	Fixed by commenting out the code to emphasise the scantable together with code regarding the testing of signals.

6.1.2. Inclusion of Extra Features

The following extra features were built into the software to make it more user friendly, easy to use, and robust.

6.1.2.1. Analog Filter Pop Up Menu

The function of this is to pop up a menu of available frequencies for the filters. The prequirements and solutions for this are given in the table below:

No	Requirement	Solution
1	There are only certain frequencies for 200Kb/s and a different set of frequencies for 2Mb/s.	An extra menu has been created to cater for the seclection of different sets based on the BitRate.

6.1.2.2. Repeat Command Box

The function of this Repeat Command Box allows the user to highlight the set of cells to be repeated in sequence to the end of the minor frame length. If the pattern copied is incomplete at the end of the minor frame then the user is notified with a message to bring attention to the matter.

6.1.2.3. Scroll Bars

The software was also intended to be used with a lap-top computer out in the field. Currently the user interface is set to medium screen resolution. If the software is installed on computers with lower screen resolutions than the current user interface screen resolution then the users will not be able to see the entire user interface. To overcome this, scroll bars have been introduced to allow the user to scroll to parts of the user interface cut-off by lower screen resolutions.

The software module used to incorporate scroll bars on the user interface comes from a third party provider. This evident when the software is first run. A message appears to inform the user that a third party module is in use and needs to be registered. This is left to the USAF to proceed with any further action on this matter.

6.1.3. Program Commenting

All the code in the software contains brief comments to aid programmers in understanding what the code does and to allow for easy readability and modification of the code in the future. The complete software code is attached in Appendix B.

6.2. Hardware

After calibration, the hardware was tested using a Tektronix 2225 oscilloscope. The measurements include frequency, offset and amplitude of the test signal. Tests on long-term stability of the test signals and the influence of disturbing factors like temperature variations and load variations on the power supply were not performed as this product is only considered a concept phase demonstrator.

6.2.1. Frequency

Because all frequencies are derived from one source (an RC generator) by binary divider circuitry, only one measurement suffices. The frequency was measured on the output for channel 1 with settings: gain: 1x, offset: 0.3125V and filter 3.9kHz. The frequency should be 1.3kHz in this case. Because the measurement was done just after calibration, it may not come as a surprise that the frequency was exactly what it should be.

6.2.2. Gain and Offset

The gain and the offset were measured for in several combinations. The test results are enumerated in the following table (the first value denotes the measured amplitude in volts, the second the measured offset in volts):

	Gain		
Offset	1x	4x	32x

	Gain		
Offset	1x	4x	32x
-2.5 V	1.4	0.33	45m
	-2.4	-2.35	-2.3
-1.25 V	1.4	0.33	45m
	-1.15	-1.2	-1.15
-0.3125 V	1.45	0.34	45m
	-0.25	-0.24	-0.27
+0.3125 V	1.5	0.32	45m
	0.30	0.32	0.33
1.25 V	1.45	0.35	45m
	1.4	1.2	1.2
2.5 V	1.45	0.34	45m
	2.5	2.4	2.4

The relative inaccuracy of these results is shown below:

	Gain		
Offset	1x	4x	32x
-2.5 V	-6.7%	-12%	-10%
	-4%	-6%	-8%
-1.25 V	-6.7	-12%	-10%
	-8%	-4%	-8%
-0.3125 V	-3.3%	-9.4%	-10%
	-20%	-23.2%	-13.6%
+0.3125 V	0%	-4.7%	-10%
	-4%	2.4%	5.6%
1.25 V	-3.3%	-6.7%	-10%
	12%	4%	4%
2.5 V	-3.3%	-9.3%	-10%
	0%	-4%	-4%

The inaccuracies measured are pretty large to what you would expect from a signal generator. However, the purpose of the test signal generator is only to test whether the setup was done correctly. For this purpose, variations of up to 10 percent are not a problem. There are some measurements that fall out of this range. Reasons for this can be the poor power supply stabilization of the 4.7V supply and inaccuracy of resistors.

6.3. Power supply

The Test Signal Generator has two power supply options: An AC or DC power back, or the PC. After building the circuit, some problems arose using the PC power supply. Inductor L1 became very hot, after which the generated voltage collapsed. This is most likely caused by an incorrect type of inductor used for L1, causing it to be driven into saturation. This can cause a significant dissipation. Therefore, only one power option remains.

The quality of the standard voltage regulating circuitry was not fully tested. It may be expected however that the IC used for this purpose (the LM317T) is up to its task.

7. User's Guide to the Hardware and Software

7.1. Software

7.1.1. Creating SMT configurations

Figure 13 shows the SMT configuration screen. The configuration is divided into 4 groups: descriptions, transmission setup, channel setup and frame structure setup.

7.1.1.1. SMT configuration settings

The descriptions are for human use only. A project name, a description of this particular setup and your name can be filled in.

The transmission setup consists of settings for bitrate/encoding, PN Seed (QPSK version only, don't care for the FSK version), transmission frequency and synchronization word setup. You can pick most settings from the drop-down boxes, which contain all possible values for the setting. When filling in the sync word, note that the length of the syncword should match the 'sync word length' field.

The channel setup is organized as a matrix. In the different rows the analog channels are listed. The digital and serial channel(s) don't have properties to setup and are therefore omitted from the matrix. In the columns, the gain, offset and bandwidth properties are listed. When you click on a cell, a pick list pops up, allowing you to choose a value for the selected property. It is also possible to select an entire column

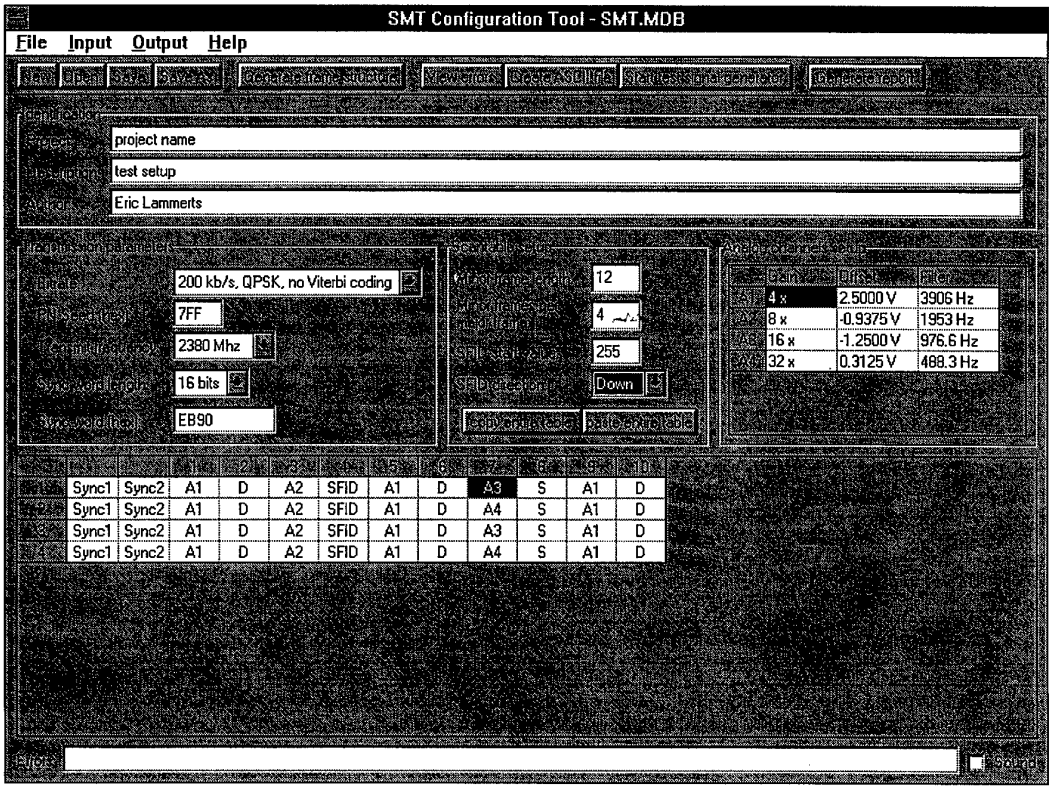


Figure 13: Main configuration screen

by clicking on the column header or dragging the mouse across the matrix. All selected properties are filled in then. When using the keyboard, you can move around with the cursor keys and select values using the spacebar or enter key. Multiple cells can be changed by using the shift+cursor keys.

The frame structure setup consists of the settings of the minor frame length, the number of minor frames, the subframe-identifier (SFID) (only if there's more than one minor frame) and the frame itself. The sync words are already filled in, since their position is fixed. The cells are named according to IRIG standards (the first field after the sync words is numbered 1). Editing the frame structure works just like the analog channels: When you click a cell, or select a region, a pick list pops up, allowing you to choose the cell value. You can also choose to clear the cells, copy them to the clipboard or paste the clipboard into the frame structure. The format used on the clipboard is text with each row on a line, columns separated by tabs. This format allows easy integration with Microsoft Excel and other applications that use data in a tabular format. There are also two command buttons that deal with the clipboard: Copy Entire and Paste Entire. Copy Entire copies the whole frame to the clipboard. This is equivalent to selecting the whole frame structure and choosing Copy from the pick list. The Paste Entire has some added functionality: It examines the clipboard and adjusts the settings for minor frame length and the number of minor frames. So if you prepared a frame structure in a different application, you paste it with ease.

The SFID settings consist of a start value and a direction. The start value determines he values of the SFID in the first minor frame. Each following minor frame gets a number one higher or lower than in the previous frame.

When there's only one minor frame, things change: The frame structure can be viewed not only as one row with several columns, it can also be viewed as a matrix. This has two purposes: First, giving a better view of the frame structure, since it usually eliminates the need to scroll the screen. Second, giving a clear view of how often a cell value is repeated. You can set the number of columns to any number the minor frame length is divisible by. An easier way to change the number of columns is click the Emphasize Channel control. When you select a channel from the list, the program emphasizes the position of that channel, and tries to adjust the column width so that the channel appears once a row, in the same column. This only works when the channel values are spaced uniformly in the minor frame (this is an IRIG requirement).

The SFID setting disappears because there's no need for an SFID.

7.1.1.2.Error messages

When you enter an inappropriate value, or violate the requirements imposed by the SMT device capabilities, an error message is generated. This error message is displayed in de bottom of the window. If you enable the Sound checkbox, a beep is generated when an error occurs. The software was deliberately made in such a way that the user can continue editing without being interrupted with message windows stating things you might know already. There can be only one error message in the text box, and the text box only displays errors in the item you are currently editing. If you want a list of all error, choose View Errors on the Output menu. Note that still only one error message per setting is displayed.

7.1.1.3.File Menu

The File menu offers you the familiar New/Open/Save/Save As options. The only thing different from most Windows applications is that the SMT configuration files are saved in the Microsoft Access database format. You can save the configuration data to an existing database, even if it doesn't already contain the tables used to hold the SMT data. The tables are automatically created then. When the tables do already exist, their properties should be compatible with the requirements of the configuration tool. Otherwise, you are informed why things are not working.

The other options in the File menu (Open/Edit SMT Device Specification) allow you to use other SMT device specifications than the default one.

7.1.1.4.Input Menu

Currently, the only item on this menu is the automatic frame structure generation. You can specify the sample frequencies you want by typing them in the boxes. With the button "Generate Frame Structure" you can check whether the chosen combination is possible. Unfortunately the user gets little help when it comes to selecting appropriate values. There was no time left to improve this part. With the button "Use this frame structure", the frame structure on the main form is filled with the channels specified.

7.1.2. Output options

The SMT Configuration Tool has three ways to output the SMT configuration: Creating an SMT EEPROM file necessary to program the SMT device, setting up the test signal generator and creating a report of the configuration.

7.1.2.1.EEPROM file creation

If you choose this option, the program generates the EEPROM file and displays it in a window. Comments are added to the file, so it is possible to inspect the system settings, channel settings and scantable. After inspection you can save it, and optionally start ITPDL to program the device. Refer to the ITPDL documentation for instruction about this.

7.1.2.2.Test Signal Generator

When you activate this option, the program investigates the parallel ports on the PC to check whether the hardware is connected. The standard port addresses 0x3BC (LPT1), 0x378 (LPT1/2) and 0x278 (LPT2/3) are checked. If the hardware is found, it is configured to generate the correct test signals. You can choose to test one channel, or have the program alternate between the four channels automatically. Don't forget that when the channel is set to a certain value, the input voltage of the other channels equals 0.

7.1.2.3.Report generation

To generate the report, functionality of Microsoft Word is borrowed. This means you have to have Word installed on your machine. The program creates a link to Word using OLE (object linking and embedding). It sends the configuration data to Word and formats it. The information included is:

- Title, project name, description, author, date&time, number of pages
- Transmission Parameters
- Analog channel properties
- Digital channels properties
- Frame structure
- Test signals
- EEPROM file

A sample report is included in Appendix A.

For the Analog and Digital channels, the sampling frequency is automatically deduced from the frame structure and bitrate. When non-uniform spacing of supercommutated signal occurs, this is printed as well. Strictly speaking, non-uniform spacing is not allowed by the IRIG Class I standards.

When Word is ready creating the report, it saves it with the name you chose. You may want to print it too, but you can of course do this later as well.

7.1.3. How to add new SMT device types

To add new SMT devices, you need to generate an SMT device specification file. The easiest way is to copy the existing SMTSPEC.MDB file to a new name and editing the tables. That way, the table structures are automatically correct. The following tables are to be edited:

- **ascii**

  This table contains the configuration part of the EEPROM. All commands are pre-entered into this table. You need to figure out these commands yourself. Parameters and such are filled in by the program. The locations of those parameters are listed in the tables "system" and "channels". Locations that need to be filled in are given the data value -1. Since this is an invalid value for the EEPROM, it allow the configuration tool to detect that information is missing.

- **system**

  The program uses only the first record of this table. The fields contain the locations in the EEPROM configuration part of various system parameters. Exceptions are the fields "asciiFileLength" and "scantableMaxLength". These contain the EEPROM size and the maximum size of the scantable, resp.

- **bitrate & transmitfreq**

  These tables contain the valid values for these system parameters. The field "smt_data" contains the values to program into the EEPROM locations.

- **channels**

  This table describes the properties of the input channels of the device. For all channels, this includes the name, description and the boolean field "eachminorframe". This field determines whether it's required to have this field in each minor frame in the same position. For the digital, serial and analog channels, the value with which the channel is identified in the scantable is given in the field "scantable_value". Finally, only for analog channels, the locations in the

EEPROM configuration part of the gain, offset and filter properties of the channel are given.

- **gain, offset & filter**

  These tables contain the valid choices for the Gain, Offset and Filter settings of the analog channels. For each choice, the value to put into the EEPROM configuration part is given.The field "test_data" is used by the Test Signal generator. It contains the codes to setup the test hardware correctly.

7.1.4. How to change the report layout

You may want to alter the paper size, margins etc. of the report. If you want to change the default values the program uses, start Word and edit the file REPORT.DOC, residing in the SMT configuration tool's program directory. This file contains the template used for creating the file. You can change the paragraph style by changing the styles "Heading 1" (used for the title), "Heading 2" (used for the header of each section), "SMT_EEPROM" (used for the EEPROM file) and "Normal" (used for normal text and tables). Other styles are not used.

7.2. Hardware

7.2.1. Calibration procedure

The test signal generator has three calibration points: Frequency (P1), Gain (P2) and Offset (P3). These calibrations do not influence each other, so they can be done in any order. Below a calibration procedure is described (there is of course more than one way to do it). For this procedure an oscilloscope is needed. Only one channel is needed, and the bandwidth of the signals measured is so small (below 100kHz) that almost any scope will do.

- **Step 1: Frequency**

  Connect the scope to the CLK pin of IC2 (pin 10) and turn P1 so that the frequency equals 10.4 kHz.

- **Step 2: Offset**

  Run the SMT configuration tool, set for channel 1 the gain to 32, the offset to 2.5V and the filter bandwidth to 3.9kHz. Start the test signal generator, select channel 1. Connect the scope input to the A1+ pin of the SMT connector. A square wave with a small amplitude (something like 50mV) and frequency of 1.3kHz should be visible. Turn P3 so that the mean level of the signal equals 2.5V.

- **Step 3: Gain**

  Change the gain setting of channel 1 to 1 and the offset to 0.3125V. Start the test signal generator, select channel 1. Connect the scope input to the A1+ pin of the SMT connector. Turn P2 so that the peak-peak value of the square wave equals 3V.

This completes the hardware calibration.

8. Conclusions and Recommendations

During this project, an SMT configuration tool was developed and tested. Hardware as well as software has been designed and tested. The software allows the user to easily put together a set of specifications to program the SMT device. The hardware allows the user to actually program the device and test its setup.

Using off-the-shelf database technology, easy integration with other programs is possible.

An aim that is not attained fully is the computer-aided frame structure generation. It is therefore highly recommended that further research be conducted in this area.

An aim than could not be attained at all is the integration of the receiver side of the telemetry system. This is because no information was available about that system.

9. References

Stephen Horan, *"Introduction to PCM Telemetering Systems"*, CRC Press, Inc., 1993

O.J. Strock, *"Telemetry Computer Systems: The New Generation"*, Instrument Society of America, 1988

"IRIG Standard 106-93: Telemetry Standards", Telemetry Group, Range Commanders Council, 1993

10. Appendix A: A Sample Report

SMT Configuration Report

Project: test project

Description: test setup

Author: Eric Lammerts

Date of creation: Monday, Sep 16 1996 16:16

Number of pages: 4

Transmission parameters

Bitrate	200 kb/s, QPSK, no Viterbi coding
PN Seed	7FF
Transmit frequency	2380 Mhz

Analog channels

Name	Gain	Offset	Filter	Sampling frequency
A1	4 x	2.5000 V	3906 Hz	6250 Hz
A2	8 x	-0.9375 V	1953 Hz	2083 Hz
A3	16 x	-1.2500 V	976.6 Hz	1042 Hz
A4	32 x	0.3125 V	488.3 Hz	1042 Hz

Digital channels

Name	Sampling frequency
D	6250 Hz
S	2083 Hz

Frame Structure

Minor frame length: 12

Minor frames per major frame: 4

Sync words: EB 90 (hex)

SFID Start value: 255

SFID Direction: Down

Frame Structure:

			1	2	3	4	5	6	7	8	9	10
Minor frame 1	Sync1	Sync2	A1	D	A2	SFID	A1	D	A3	S	A1	D
Minor frame 2	Sync1	Sync2	A1	D	A2	SFID	A1	D	A4	S	A1	D
Minor frame 3	Sync1	Sync2	A1	D	A2	SFID	A1	D	A3	S	A1	D
Minor frame 4	Sync1	Sync2	A1	D	A2	SFID	A1	D	A4	S	A1	D

Test Signals

Name	Type	Frequency [Hz]	Amplitude [V]	Offset [V]
A1	square wave	2600	3.75E-01	-2.5
A2	square wave	1300	1.88E-01	0.9375
A3	square wave	650	9.38E-02	1.25
A4	square wave	325	4.69E-02	-0.3125
D(0)	digital	2600		
D(1)	digital	1300		
D(2)	digital	650		
D(3)	digital	325		
D(4)	digital	162		
D(5)	digital	81		
D(6)	digital	40		
D(7)	digital	20		

SMT EEPROM Description file (ASCII file)

```
   0    72    (0x48) bitrate / coding -> 200 kb/s, QPSK, no Viterbi coding
   1     7     (0x7) 11bit PN seed (MSB) -> 0x7FF
   2   255    (0xFF) 11bit PN seed (LSB)
   3     2     (0x2)
   4    50    (0x32) end QPSK settings
   5   128    (0x80)
   6     0     (0x0)
   7   128    (0x80)
   8     0     (0x0) repeated analog init
   9   130    (0x82)
  10     4     (0x4) A1
  11    99    (0x63)
  12     4     (0x4)
  13   227    (0xE3) filtering A1 = 3906 Hz
  14    97    (0x61)
  15     4     (0x4)
  16     7     (0x7) offset A1 = 2.5000 V
  17   101    (0x65)
  18     4     (0x4)
  19     3     (0x3) gain A1 = 4 x
  20   102    (0x66)
  21     4     (0x4)
  22     0     (0x0) A2
  23    99    (0x63)
  24     5     (0x5)
  25   193    (0xC1) filtering A2 = 1953 Hz
  26    97    (0x61)
  27     5     (0x5)
  28    13     (0xD) offset A2 = -0.9375 V
  29   101    (0x65)
  30     5     (0x5)
  31     4     (0x4) gain A2 = 8 x
  32   102    (0x66)
  33     5     (0x5)
  34     0     (0x0) A3
  35    99    (0x63)
  36     6     (0x6)
  37   183    (0xB7) filtering A3 = 976.6 Hz
  38    97    (0x61)
  39     6     (0x6)
  40    12     (0xC) offset A3 = -1.2500 V
  41   101    (0x65)
  42     6     (0x6)
  43     5     (0x5) gain A3 = 16 x
  44   102    (0x66)
  45     6     (0x6)
  46     0     (0x0) A4
  47    99    (0x63)
  48     7     (0x7)
  49   147    (0x93) filtering A4 = 488.3 Hz
  50    97    (0x61)
  51     7     (0x7)
  52     0     (0x0) offset A4 = 0.3125 V
  53   101    (0x65)
  54     7     (0x7)
  55     6     (0x6) gain A4 = 32 x
  56   102    (0x66)
  57     7     (0x7)
  58     0     (0x0)
  59    32    (0x20)
  60   128    (0x80)
  61     9     (0x9)
  62    76    (0x4C) transmit frequency -> 2380 Mhz
  63    32    (0x20)
  64     0     (0x0)
  65     0     (0x0)
  66    35    (0x23)
  67   160    (0xA0)
  68    31    (0x1F) scan table pointer MSB
  69   208    (0xD0) scan table pointer LSB
8144   235    (0xEB) Sync word #1 ======================
8145   144    (0x90) Sync word #2
8146     4     (0x4) Analog Ch 1
8147     1     (0x1) Discrete byte
8148     5     (0x5) Analog Ch 2
8149   255    (0xFF) Subframe Identifier
8150     4     (0x4) Analog Ch 1
8151     1     (0x1) Discrete byte
```

```
8152    6     (0x6) Analog Ch 3
8153    0     (0x0) Serial clk out & data in
8154    4     (0x4) Analog Ch 1
8155    1     (0x1) Discrete byte
8156  235    (0xEB) Sync word #1 ======================
8157  144    (0x90) Sync word #2
8158    4     (0x4) Analog Ch 1
8159    1     (0x1) Discrete byte
8160    5     (0x5) Analog Ch 2
8161  254    (0xFE) Subframe Identifier
8162    4     (0x4) Analog Ch 1
8163    1     (0x1) Discrete byte
8164    7     (0x7) Analog Ch 4
8165    0     (0x0) Serial clk out & data in
8166    4     (0x4) Analog Ch 1
8167    1     (0x1) Discrete byte
8168  235    (0xEB) Sync word #1 ======================
8169  144    (0x90) Sync word #2
8170    4     (0x4) Analog Ch 1
8171    1     (0x1) Discrete byte
8172    5     (0x5) Analog Ch 2
8173  253    (0xFD) Subframe Identifier
8174    4     (0x4) Analog Ch 1
8175    1     (0x1) Discrete byte
8176    6     (0x6) Analog Ch 3
8177    0     (0x0) Serial clk out & data in
8178    4     (0x4) Analog Ch 1
8179    1     (0x1) Discrete byte
8180  235    (0xEB) Sync word #1 ======================
8181  144    (0x90) Sync word #2
8182    4     (0x4) Analog Ch 1
8183    1     (0x1) Discrete byte
8184    5     (0x5) Analog Ch 2
8185  252    (0xFC) Subframe Identifier
8186    4     (0x4) Analog Ch 1
8187    1     (0x1) Discrete byte
8188    7     (0x7) Analog Ch 4
8189    0     (0x0) Serial clk out & data in
8190    4     (0x4) Analog Ch 1
8191    1     (0x1) Discrete byte
```

11. Appendix B:

Software Code

frmAbout - 1

```
Option Explicit

Private Sub cmdAboutOk_Click()
    Unload Me
End Sub

Private Sub cmdACTE_Click()
    frmACTE.Show
End Sub
```

frmAbout - 1

```
VERSION 5.00
Begin VB.Form frmAbout
   BorderStyle     =   1  'Fixed Single
   Caption         =   "About..."
   ClientHeight    =   4320
   ClientLeft      =   3048
   ClientTop       =   4092
   ClientWidth     =   3900
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   PaletteMode     =   1  'UseZOrder
   ScaleHeight     =   4320
   ScaleWidth      =   3900
   Begin VB.CommandButton cmdACTE
      Caption         =   "Info"
      Height          =   435
      Left            =   3060
      TabIndex        =   3
      Top             =   120
      Width           =   615
   End
   Begin VB.CommandButton cmdAboutOk
      Caption         =   "OK"
      Height          =   375
      Left            =   1176
      TabIndex        =   0
      Top             =   3828
      Width           =   1455
   End
   Begin VB.Label txtAbout
      Alignment       =   2  'Center
      Caption         =   "October, 1996"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   9.6
         Charset         =   0
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   312
      Index           =   6
      Left            =   312
      TabIndex        =   8
      Top             =   2760
      Width           =   3252
      WordWrap        =   -1  'True
   End
   Begin VB.Label txtAbout
      Alignment       =   2  'Center
      Caption         =   "Modified by Peter Evdokiou"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   9.6
         Charset         =   0
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   312
      Index           =   5
      Left            =   540
      TabIndex        =   7
      Top             =   2388
      Width           =   2832
      WordWrap        =   -1  'True
   End
   Begin VB.Label txtAbout
      Alignment       =   2  'Center
      Caption         =   "Copyright 1996"
      BeginProperty Font
```

```
      Name            =   "MS Sans Serif"
      Size            =   9.6
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   312
   Index           =   4
   Left            =   1176
   TabIndex        =   6
   Top             =   3360
   Width           =   1512
   WordWrap        =   -1  'True
End
Begin VB.Label txtAbout
   Alignment       =   2  'Center
   Caption         =   "June 10th - September 13th, 1996"
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   9.6
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   315
   Index           =   3
   Left            =   360
   TabIndex        =   5
   Top             =   1920
   Width           =   3255
   WordWrap        =   -1  'True
End
Begin VB.Label txtAbout
   Alignment       =   2  'Center
   Caption         =   "written by Eric Lammerts"
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   9.6
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   315
   Index           =   2
   Left            =   540
   TabIndex        =   4
   Top             =   1560
   Width           =   2835
   WordWrap        =   -1  'True
End
Begin VB.Label txtAbout
   Alignment       =   2  'Center
   Caption         =   "Australian Centre for Test and Evaluation"
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   9.6
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   555
   Index           =   1
   Left            =   840
   TabIndex        =   2
   Top             =   120
   Width           =   2115
```

```
      WordWrap        =   -1  'True
   End
   Begin VB.Image imgUnisa
      Height          =   540
      Left            =   180
      Picture         =   (Bitmap)
      Top             =   120
      Width           =   420
   End
   Begin VB.Label txtAbout
      Alignment       =   2  'Center
      Caption         =   "SMT Configuration Tool"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   12
         Charset         =   0
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   495
      Index           =   0
      Left            =   300
      TabIndex        =   1
      Top             =   1020
      Width           =   3315
      WordWrap        =   -1  'True
   End
End
```

Australian Centre for Test and Evaluation

SMT Configuration Tool

written by Eric Lammerts

June 10th - September 13th, 1996

Modified by Peter Evdokiou

October, 1996

frmACTE - 1

```
Option Explicit

Private Sub cmdAboutOk_Click()
    Unload Me
End Sub
```

```
frmACTE - 1

VERSION 5.00
Begin VB.Form frmACTE
   BorderStyle     =   1  'Fixed Single
   Caption         =   "What is ACTE?"
   ClientHeight    =   5676
   ClientLeft      =   1416
   ClientTop       =   1812
   ClientWidth     =   8412
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   PaletteMode     =   1  'UseZOrder
   ScaleHeight     =   5676
   ScaleWidth      =   8412
   Begin VB.CommandButton cmdAboutOk
      Cancel          =   -1  'True
      Caption         =   "OK"
      Default         =   -1  'True
      Height          =   375
      Left            =   3480
      TabIndex        =   0
      Top             =   5160
      Width           =   1455
   End
   Begin VB.Label lblACTE
      Caption         =   "For more information, look at the ACTE Web page at http://www.acte.
      Height          =   255
      Index           =   4
      Left            =   120
      TabIndex        =   5
      Top             =   4740
      Width           =   8175
      WordWrap        =   -1  'True
   End
   Begin VB.Label lblACTE
      Caption         =   <...>
      Height          =   855
      Index           =   3
      Left            =   120
      TabIndex        =   4
      Top             =   3840
      Width           =   8235
      WordWrap        =   -1  'True
   End
   Begin VB.Label lblACTE
      Caption         =   <...>
      Height          =   855
      Index           =   2
      Left            =   120
      TabIndex        =   3
      Top             =   3000
      Width           =   8175
      WordWrap        =   -1  'True
   End
   Begin VB.Label lblACTE
      Caption         =   "What is ACTE?"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   12
         Charset         =   0
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   315
      Index           =   1
      Left            =   120
      TabIndex        =   2
      Top             =   2640
      Width           =   1695
      WordWrap        =   -1  'True
   End
   Begin VB.Image imgACTE
```

```
      Height          =   1548
      Left            =   120
      Picture         =   (Bitmap)
      Top             =   120
      Width           =   6564
   End
   Begin VB.Label lblACTE
      Caption         =   <...>
      Height          =   435
      Index           =   0
      Left            =   120
      TabIndex        =   1
      Top             =   2160
      Width           =   8175
      WordWrap        =   -1  'True
   End
End
```

UNIVERSITY OF SOUTH AUSTRALIA

AUSTRALIAN CENTRE FOR TEST AND EVALUATION

The Australian Centre for Test and Evaluation (ACTE) is located at the University of South Australia's Salisbury Campus, which is located some 17 kilometres north of Adelaide, South Australia.

What is ACTE?

The Australian Centre for Test and Evaluation (ACTE) has been established to provide a focus for Test and Evaluation in the Asia Pacific Region, to expand recognition of the value of T&E and to develop the skill levels of its practitioners. It is the first such activity in a non-defence university, and is one of several R&D centres in the School of Physics and Electronic Systems Engineering.

T&E is a scientifically based disciplined process used to help ensure that new complex technological systems and products are maturing at an expected rate, that simulations and models of those systems and products are faithful and that they behave as expected in the user environment over their useful lives. T&E is a principal mechanism for reducing technological risk. ACTE offers a range of services to assist this.

For more information, look at the ACTE Web page at http://www.acte.unisa.edu.au

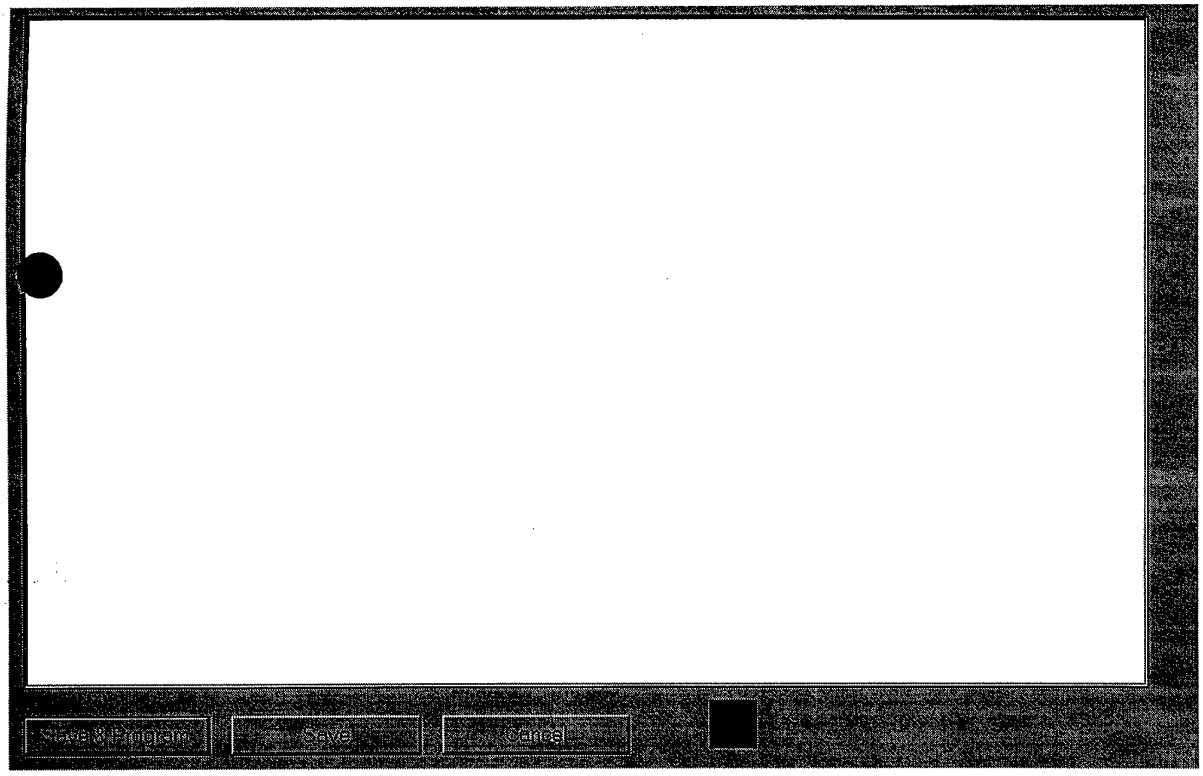
[redacted]

frmAscii - 1

```
Option Explicit

Private Sub save()
Dim file, i As Integer
Dim filename As String

    With commondialogAscii
        .Flags = _
            cdlOFNHideReadOnly Or cdlOFNNoReadOnlyReturn Or _
            cdlOFNOverwritePrompt Or cdlOFNPathMustExist
        filename = frmMain.commondialogFile.filename
        For i = Len(filename) To 1 Step -1
            If Mid(filename, i, 1) = "."
                Or Mid(filename, i, 1) = "\" Then Exit For
        Next
        If filename <> "" Then
            If Mid(filename, i, 1) = "." Then Mid(filename, i) = ".DAT"
        End If
        .filename = filename
        On Error GoTo errAsciiSave
        .ShowSave
        On Error GoTo 0
        If .filename <> "" Then
            file = FreeFile
            Open .filename For Output Access Write As #file
            Print #file, txtAscii
            Close #file
        End If
    End With
    Exit Sub
errAsciiSave:
    MsgBox "Error saving ASCII file", vbOKOnly + vbExclamation, "Error"
End Sub

Private Sub cmdAsciiDontSave_Click()
    Unload Me
End Sub

Private Sub cmdAsciiSave_Click()
    save
    Unload Me
End Sub

Private Sub cmdAsciiSaveProgram_Click()
    save
    Shell App.Path & "\ITPDL.PIF " & commondialogAscii.filename, vbNormalFocus
    Unload Me
End Sub
```

```
VERSION 5.00
Object = "{F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.1#0"; "COMDLG32.OCX"
Begin VB.Form frmAscii
   BorderStyle     =   1  'Fixed Single
   Caption         =   "ASCII File"
   ClientHeight    =   6060
   ClientLeft      =   876
   ClientTop       =   1248
   ClientWidth     =   9600
   Icon            =   (Icon)
   LinkTopic       =   "Form2"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   PaletteMode     =   1  'UseZOrder
   ScaleHeight     =   6060
   ScaleWidth      =   9600
   Begin VB.CommandButton cmdAsciiSaveProgram
      Caption         =   "Save && Program"
      Default         =   -1  'True
      Height          =   315
      Left            =   120
      TabIndex        =   0
      Top             =   5640
      Width           =   1515
   End
   Begin VB.CommandButton cmdAsciiDontSave
      Cancel          =   -1  'True
      Caption         =   "Cancel"
      Height          =   315
      Left            =   3480
      TabIndex        =   2
      Top             =   5640
      Width           =   1515
   End
   Begin VB.CommandButton cmdAsciiSave
      Caption         =   "Save"
      Height          =   315
      Left            =   1800
      TabIndex        =   1
      Top             =   5640
      Width           =   1515
   End
   Begin VB.TextBox txtAscii
      BeginProperty Font
         Name            =   "Courier New"
         Size            =   8.4
         Charset         =   0
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   5355
      Left            =   120
      Locked          =   -1  'True
      MultiLine       =   -1  'True
      ScrollBars      =   2  'Vertical
      TabIndex        =   3
      TabStop         =   0   'False
      Top             =   120
      Width           =   9375
   End
   Begin MSComDlg.CommonDialog commondialogAscii
      Left            =   5640
      Top             =   5520
      _ExtentX        =   847
      _ExtentY        =   847
      _Version        =   327681
      CancelError     =   -1  'True
      DefaultExt      =   "dat"
      DialogTitle     =   "Create ASCII File"
      Filter          =   "SMT Ascii files (*.dat)|*.dat"
      MaxFileSize     =   64
   End
```

```
frmAscii - 2

End
```

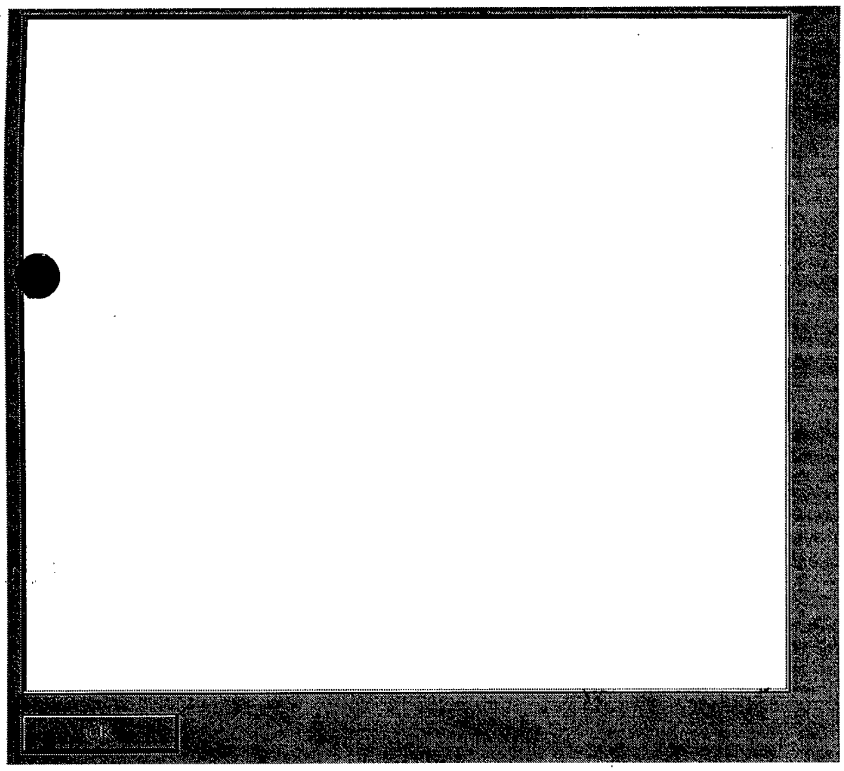

Save & Program
Save
Cancel

frmErrors - 1

```
Option Explicit

Private Sub cmdViewErrorsOk_Click()
    Unload Me
End Sub

Private Sub Form_Deactivate()
    Unload Me
End Sub

Private Sub Form_Load()
    txtViewErrors = ""
End Sub
```

frmErrors - 1

```
VERSION 5.00
Begin VB.Form frmErrors
   BorderStyle     =   1  'Fixed Single
   Caption         =   "Error list"
   ClientHeight    =   6036
   ClientLeft      =   1092
   ClientTop       =   1512
   ClientWidth     =   6720
   Icon            =   (Icon)
   LinkTopic       =   "Form2"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   PaletteMode     =   1  'UseZOrder
   ScaleHeight     =   6036
   ScaleWidth      =   6720
   WindowState     =   1  'Minimized
   Begin VB.CommandButton cmdViewErrorsOk
      Cancel          =   -1  'True
      Caption         =   "OK"
      Default         =   -1  'True
      Height          =   315
      Left            =   120
      TabIndex        =   0
      Top             =   5640
      Width           =   1275
   End
   Begin VB.TextBox txtViewErrors
      Height          =   5415
      Left            =   120
      Locked          =   -1  'True
      MultiLine       =   -1  'True
      ScrollBars      =   2  'Vertical
      TabIndex        =   1
      TabStop         =   0   'False
      Top             =   120
      Width           =   6495
   End
End
```

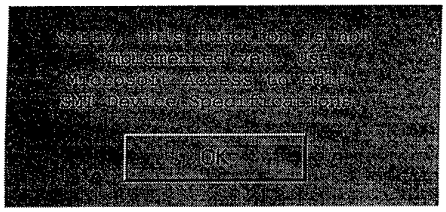


frmFileEditSmtDeviceSpecification - 1

```
Option Explicit

Private Sub cmdFileEditSmtDeviceSpecification_Click()
    Unload Me
End Sub
```

frmFileEditSmtDeviceSpecification - 1

```
VERSION 5.00
Begin VB.Form frmFileEditSmtDeviceSpecification
   BorderStyle     =   3  'Fixed Dialog
   Caption         =   "Edit SMT device specification"
   ClientHeight    =   1644
   ClientLeft      =   2052
   ClientTop       =   3156
   ClientWidth     =   3468
   ControlBox      =   0   'False
   Icon            =   (Icon)
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   PaletteMode     =   1  'UseZOrder
   ScaleHeight     =   1644
   ScaleWidth      =   3468
   ShowInTaskbar   =   0   'False
   Begin VB.CommandButton cmdFileEditSmtDeviceSpecification
      Caption         =   "OK"
      Height          =   375
      Left            =   960
      TabIndex        =   1
      Top             =   1080
      Width           =   1455
   End
   Begin VB.Label lblFileEditSmtDeviceSpecification
      Alignment       =   2  'Center
      Caption         =   "Sorry, this function is not implemented yet. Use Microsoft Access t
      Height          =   735
      Left            =   360
      TabIndex        =   0
      Top             =   240
      Width           =   2655
      WordWrap        =   -1  'True
   End
End
```


OK

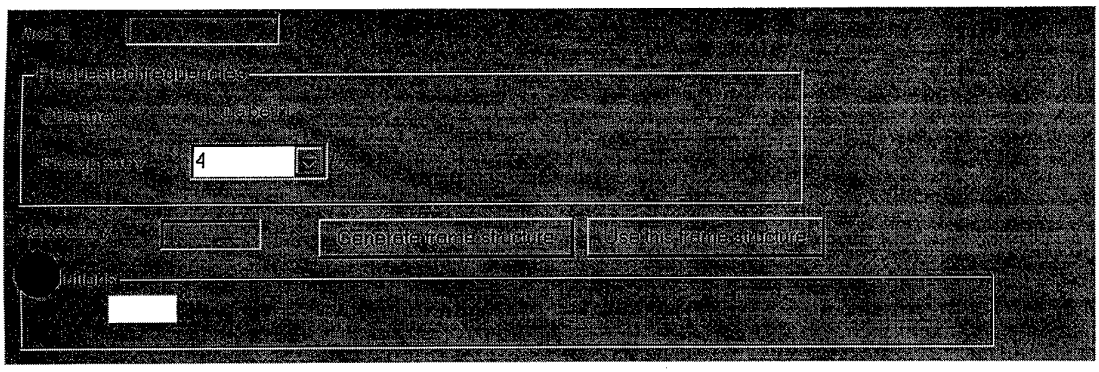
frmgen - 1

```
Option Explicit
Dim numchannels As Integer
Dim channelfreq() As Integer

Private Sub updateChannelfreq()
Dim i As Integer
Dim tot As Double
    For i = 1 To numchannels
        If Val(cmbRequestedFreq(i)) > 0 Then
            channelfreq(i) = wordrate / 2 / Val(cmbRequestedFreq(i))
        Else
            channelfreq(i) = 0
        End If
    Next
    tot = 0
    For i = 1 To numchannels
        If Val(cmbRequestedFreq(i)) > 0 Then tot = tot + Val(cmbRequestedFreq(i))
    Next
    txtCapacityUsed = Format(tot / (wordrate / 2), "Percent")
End Sub
Private Sub cmbRequestedFreq_Click(Index As Integer)
    updateChannelfreq
End Sub

Private Sub cmbRequestedFreq_LostFocus(Index As Integer)
    cmbRequestedFreq_Click (Index)
End Sub

Private Sub cmdGenerate_Click()
Dim scantable() As Integer
Dim i As Long

    updateChannelfreq
    channelfreq(numchannels + 1) = 1
    i = lcm(channelfreq)
    channelfreq(numchannels + 1) = i

    If scantablePossible(channelfreq) Then
        With grdAnalogFreq
            .cols = numchannels + 1
            .row = 0
            For i = 1 To numchannels
                .col = i
                If channelfreq(i) <> 0 Then
                    .text = Format(wordrate / 2 / channelfreq(i), "0.00")
                Else
                    .text = ""
                End If
            Next
        End With
    Else
        MsgBox "Scantable not possible", vbOKOnly + vbExclamation, "Error"
    End If
End Sub

Private Sub Command1_Click()
    generateScantable
    frmMain.cmbEmphasize.ListIndex = 0
    Unload Me
End Sub

Private Sub Form_Load()
Dim i, j, colw, spacing As Integer

    computeBitrateWordrate
    computeSampleFreqs
    Select Case wordrate
        Case Is >= 1000000
            txtWordrate = wordrate / 1000000 & " Mwords/s"
        Case Is >= 1000
            txtWordrate = wordrate / 1000 & " kwords/s"
        Case Else
            txtWordrate = wordrate & " words/s"
    End Select
```

```
    numchannels = frmMain.grdAnalog.rows - 1
    ReDim channelfreq(1 To numchannels + 1)
    spacing = lblChannel(1).Width + 50

    frmMain.grdAnalog.col = 0
    frmMain.grdAnalog.row = 1
    lblChannel(1) = frmMain.grdAnalog
    cmbRequestedFreq(1) = ""
    For i = 2 To numchannels
        Load lblChannel(i)
        lblChannel(i).Left = lblChannel(i - 1).Left + spacing
        frmMain.grdAnalog.row = i
        lblChannel(i) = frmMain.grdAnalog
        lblChannel(i).Visible = True

        Load cmbRequestedFreq(i)
        cmbRequestedFreq(i).Left = cmbRequestedFreq(i - 1).Left + spacing
        cmbRequestedFreq(i).TabIndex = cmbRequestedFreq(i - 1).TabIndex + 1
        cmbRequestedFreq(i) = ""
        cmbRequestedFreq(i).Visible = True

    Next
    frmMain.grdAnalog.col = 0
    For i = 1 To numchannels
        frmMain.grdAnalog.row = i
        For j = LBound(channels) To UBound(channels)
            If channels(j) = frmMain.grdAnalog Then Exit For
        Next
        cmbRequestedFreq(i) = Format(Abs(samplefreqs(j)), "0.##")

        cmbRequestedFreq(i).AddItem "off"
        For j = 2 To 10
            cmbRequestedFreq(i).AddItem Format(wordrate / j / 2, "0.##")
        Next

    Next
    updateChannelfreq
    With grdAnalogFreq
        .cols = numchannels + 1
        .rows = 10
        .FixedRows = 0
        .FixedCols = 1
        colw = lblChannel(2).Left - lblChannel(1).Left - _
            (.ColPos(2) - .ColPos(1) - .ColWidth(1))
        .ColWidth(0) = lblChannel(1).Left - .Left
        For i = 1 To numchannels
            .ColWidth(i) = colw
        Next
        .Width = .ColPos(numchannels) + .ColWidth(numchannels) + 300
    End With
    fraSolutions.Width = 2 * grdAnalogFreq.Left + grdAnalogFreq.Width
    fraRequested.Width = fraSolutions.Width
    frmgen.Width = 2 * fraSolutions.Left + fraSolutions.Width + 120
End Sub
```

```
frmgen - 1

VERSION 5.00
Object = "{A8B3B723-0B5A-101B-B22E-00AA0037B2FC}#1.0#0"; "GRID32.OCX"
Begin VB.Form frmgen
   Caption         =   "Frame structure generator"
   ClientHeight    =   2880
   ClientLeft      =   360
   ClientTop       =   1680
   ClientWidth     =   8808
   LinkTopic       =   "Form1"
   PaletteMode     =   1  'UseZOrder
   ScaleHeight     =   2880
   ScaleWidth      =   8808
   Begin VB.CommandButton Command1
      Caption         =   "Use this frame structure"
      Height          =   315
      Left            =   4680
      TabIndex        =   2
      Top             =   1740
      Width           =   1935
   End
   Begin VB.TextBox txtCapacityUsed
      BackColor       =   &H8000000F&
      Height          =   285
      Left            =   1260
      Locked          =   -1  'True
      TabIndex        =   12
      TabStop         =   0   'False
      Top             =   1740
      Width           =   855
   End
   Begin VB.TextBox txtWordrate
      BackColor       =   &H8000000F&
      Height          =   285
      Left            =   960
      Locked          =   -1  'True
      TabIndex        =   10
      TabStop         =   0   'False
      Top             =   120
      Width           =   1275
   End
   Begin VB.CommandButton cmdGenerate
      Caption         =   "Generate frame structure"
      Height          =   315
      Left            =   2520
      TabIndex        =   1
      Top             =   1740
      Width           =   2055
   End
   Begin VB.Frame fraSolutions
      Caption         =   "Solutions"
      Height          =   675
      Left            =   120
      TabIndex        =   6
      Top             =   2100
      Width           =   7875
      Begin MSGrid.Grid grdAnalogFreq
         Height          =   315
         Left            =   180
         TabIndex        =   7
         TabStop         =   0   'False
         Top             =   240
         Width           =   2535
         _Version        =   65536
         _ExtentX        =   4471
         _ExtentY        =   556
         _StockProps     =   77
         BackColor       =   16777215
         BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
            Name            =   "MS Sans Serif"
            Size            =   7.8
            Charset         =   0
            Weight          =   400
            Underline       =   0   'False
            Italic          =   0   'False
```

```
            Strikethrough   =   0   'False
         EndProperty
         Rows            =   1
         FixedRows       =   0
         MouseIcon       =   {Binary}
      End
   End
   Begin VB.Frame fraRequested
      Caption         =   "Requested frequencies"
      Height          =   1155
      Left            =   120
      TabIndex        =   4
      Top             =   480
      Width           =   6315
      Begin VB.ComboBox cmbRequestedFreq
         Height          =   300
         Index           =   1
         Left            =   1380
         TabIndex        =   0
         Text            =   "4"
         Top             =   660
         Width           =   1095
      End
      Begin VB.Label Label1
         Caption         =   "Frequency (Hz):"
         Height          =   195
         Index           =   1
         Left            =   180
         TabIndex        =   9
         Top             =   720
         Width           =   1215
      End
      Begin VB.Label Label1
         Caption         =   "Channel:"
         Height          =   195
         Index           =   0
         Left            =   180
         TabIndex        =   8
         Top             =   360
         Width           =   735
      End
      Begin VB.Label lblChannel
         Alignment       =   2  'Center
         BorderStyle     =   1  'Fixed Single
         Caption         =   "Label1"
         Height          =   255
         Index           =   1
         Left            =   1380
         TabIndex        =   5
         Top             =   300
         Width           =   1095
      End
   End
   Begin VB.Label lbl1
      Caption         =   "Capacity used:"
      Height          =   195
      Index           =   1
      Left            =   120
      TabIndex        =   11
      Top             =   1740
      Width           =   1095
   End
   Begin VB.Label lbl1
      Caption         =   "Word rate:"
      Height          =   195
      Index           =   0
      Left            =   120
      TabIndex        =   3
      Top             =   180
      Width           =   795
   End
End
```


Product-specific properties
4
Create terrain structure
Create transmission structure

```
frmGenerating - 1

Option Explicit

Private Sub cmbAbort_Click()
    abort = 1
End Sub

Private Sub Form_Load()
    lblGenerating = "Generating frame structure of size " & scantableLength * 2
End Sub
```

frmGenerating - 1

```
VERSION 5.00
Begin VB.Form frmGenerating
   BorderStyle     =   3  'Fixed Dialog
   Caption         =   "Generating frame structure"
   ClientHeight    =   1188
   ClientLeft      =   3120
   ClientTop       =   3060
   ClientWidth     =   3636
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   PaletteMode     =   1  'UseZOrder
   ScaleHeight     =   1188
   ScaleWidth      =   3636
   ShowInTaskbar   =   0   'False
   Begin VB.CommandButton cmbAbort
      Cancel          =   -1  'True
      Caption         =   "Abort"
      Height          =   375
      Left            =   1260
      TabIndex        =   0
      Top             =   660
      Width           =   1215
   End
   Begin VB.Label lblGenerating
      Alignment       =   2  'Center
      Caption         =   "-"
      Height          =   315
      Left            =   120
      TabIndex        =   1
      Top             =   180
      Width           =   3375
   End
End
```

```
Option Explicit
'
' If you have any questions about the source code, feel free to email me
'
'   Eric (eric@scintilla.utwente.nl)
'
Private Declare Function checkHardwarePresence Lib "hwaccess" () As Integer

Dim dbSpec, dbSmt As Database
Dim rsSpec, rsSmt As Recordset
Dim grdScantableIn, grdAnalogIn As Boolean
Public smtChanged As Boolean  ' indicates whether the user should be asked to save the data
Const scantableColwidthMin As Long = 300
Const scantableColwidthMax As Long = 570


Private Function checkAll() As Boolean
    txtDescription.SetFocus
    DoEvents        ' generate lostfocus event for the control
                    ' the user is editing right now
    checkAll = True
    If checkBitrate Then Exit Function
    If checkTransmitFreq Then Exit Function
    If checkPNSeed Then Exit Function
    If checkSyncword Then Exit Function
    If checkSfid Then Exit Function
    If checkMinorframelength Then Exit Function
    If checkMinorframes Then Exit Function
    If checkAnalog Then Exit Function
    If checkScantable Then Exit Function
    checkAll = False
End Function

Private Sub viewAll()
    txtDescription.SetFocus
    DoEvents        ' generate lostfocus event for the control
                    ' the user is editing right now
    checkBitrate
    checkTransmitFreq
    checkPNSeed
    checkSyncword
    checkSfid
    checkAnalog
    checkMinorframelength
    checkMinorframes
    checkScantable
End Sub
Private Sub logError(errorMsg As String)
    If frmErrors.Visible Then
        frmErrors.txtViewErrors = frmErrors.txtViewErrors & errorMsg & Chr(13) & Chr(10)
    Else
        txtError = errorMsg
    End If
End Sub
Private Sub clearError()
    txtError = ""
End Sub
Private Function checkBitrate() As Boolean
Dim i As Integer

    clearError
    checkBitrate = False
    With cmbBitrate
        For i = 0 To .ListCount - 1
            If .text = .list(i) Then Exit Function
        Next
    End With
    logError "Invalid bitrate"
    checkBitrate = True
End Function
Private Function checkTransmitFreq() As Boolean
Dim i As Integer
```

```
    clearError
    checkTransmitFreq = False
    With cmbTransmitFreq
        For i = 0 To .ListCount - 1
            If .text = .list(i) Then Exit Function
        Next
    End With
    logError "Invalid transmit frequency"
    checkTransmitFreq = True
End Function
Private Function checkPNSeed() As Boolean
    clearError
    If Val("&H" & txtPNSeed) < 0 Or Val("&H" & txtPNSeed) > &H7FF Then
        logError "PN Seed should be between 0 and 0x7FF"
        checkPNSeed = True
    Else
'        txtPNSeed = Hex(Val("&H" & txtPNSeed))
        checkPNSeed = False
    End If
End Function

Private Function checkMinorframelength()
    clearError
    If minorframelength > maxMinorframelength Or minorframelength < syncwords Then
        logError "Invalid minor frame length"
        checkMinorframelength = True
    Else
        checkMinorframelength = False
    End If
End Function

Private Function checkMinorframes()
    clearError
    If minorframes > maxMinorFrames Or minorframes < 1 Then
        logError "Invalid number of minor frames per major frame"
        checkMinorframes = True
    ElseIf minorframes * minorframelength > maxscantablelength Then
        logError "No more than " & maxscantablelength & " bytes per major frame allowed"
        checkMinorframes = True
    Else
        checkMinorframes = False
    End If
End Function

Private Function checkAnalog() As Boolean
Dim i, j, row As Integer
Dim err As String

    clearError
    With grdAnalog
        On Error GoTo analogError
        For i = 1 To .rows - 1
            .row = i

            .col = 1
            For j = 1 To 1000
                If mnuAnalogGainEntry(j).Caption = .text Then Exit For
            Next

            .col = 2
            For j = 1 To 1000
                If mnuAnalogOffsetEntry(j).Caption = .text Then Exit For
            Next

            .col = 3
            For j = 1 To 1000
                If mnuAnalogFilterEntry(j).Caption = .text Then Exit For
            Next
        Next
    End With
    checkAnalog = False
    Exit Function
analogError:
    row = grdAnalog.row
```

```
    grdAnalog.row = 0
    err = "Invalid value in " & grdAnalog & " setting of channel "
    grdAnalog.row = row
    grdAnalog.col = 0
    logError err & grdAnalog & "."
    checkAnalog = True
End Function

Private Function checkScantable() As Boolean
Dim r, c As Integer
Dim err, reqd As String
Dim prevanalog, analog, eachminorframe, sfid As Boolean

    clearError
    checkScantable = True
    saveScantable            ' save grdScantable in array 'scantable'
    sfid = False             ' no SFIDs encountered yet
    For c = syncwords To minorframelength - 1       ' check scantable column-wise
        If scantable(0, c) = "SFID" Then sfid = True     ' SFID encountered!
        For r = 0 To minorframes - 1
                                                    ' empty cells not allowed
            If scantable(r, c) = "" Then
                err = "Scantable field should be filled"
                GoTo rowcol
            End If
                                                    ' check whether channel exists
            If Not isinlist(channels, scantable(r, c)) Then
                err = "Unknown scantable entry"
                GoTo rowcol
            End If
            eachminorframe = isinlist(eachminorframes, scantable(r, c))
            If r = 0 Then
                If eachminorframe Then
                    ' remember that field should be the same in every minor frame
                    reqd = scantable(r, c)
                Else
                    reqd = ""
                End If
            Else
                If reqd <> "" Then
                    If reqd <> scantable(r, c) Then
                        ' a field that should be the same in every minor frame
                        ' occured in row 0 but not in a row>0
                        err = "Value should be the same as in previous minor frame"
                        GoTo rowcol
                    End If
                Else
                    If eachminorframe Then
                        ' a field that should be the same in every minor frame
                        ' occured in a row>0 but not in row 0
                        err = "Value not allowed if not present in all minor frames"
                        GoTo rowcol
                    End If
                End If
            End If
        Next
    Next
    If sfid And minorframes = 1 Then
       logError ("Subframe identifier in a frame with no subframes")
       GoTo endCheckScantable
    End If
    If Not sfid And minorframes <> 1 Then
       logError ("No subframe identifier in a frame with subframes")
       GoTo endCheckScantable
    End If
    ' check for consecutive analog cells
    For r = 0 To minorframes - 1
        prevanalog = isinlist(analogs, scantable(r, syncwords))
        For c = syncwords + 1 To minorframelength - 1
            analog = isinlist(analogs, scantable(r, c))
            If analog And prevanalog Then
                GoTo twoanalog
            End If
            prevanalog = analog
```

```
        Next
    Next
    checkScantable = False
    GoTo endCheckScantable

twoanalog:
    err = "Two analog channel entries after each other not allowed"
    GoTo rowcol
rowcol:
    If minorframes = 1 Then
        grdScantable.row = c \ columns + 1
        grdScantable.col = 0
        err = err & " in row " & grdScantable
        grdScantable.col = c Mod columns + 1
        grdScantable.row = 0
    Else
        grdScantable.row = r + 1
        grdScantable.col = 0
        err = err & " in row " & grdScantable
        grdScantable.col = c + 1
        grdScantable.row = 0
    End If
    logError err & ", column " & grdScantable & "."
endCheckScantable:
    Erase scantable
End Function

Private Function checkSfid()
    clearError
    checkSfid = False
    If minorframes = 1 Then Exit Function ' with 1 minor frame you don't use sfid's
    checkSfid = True
    If Val(txtSfidStart) < 128 Or Val(txtSfidStart) > 255 Then
        logError "SFID should be between 128 and 255"
        Exit Function
    End If
    If cmbSfidDir = "Up" Then
        If Val(txtSfidStart) + (minorframes - 1) > 255 Then
            logError "SFID exceeds 255 in minor frame " & 256 - Val(txtSfidStart)
            Exit Function
        End If
    ElseIf cmbSfidDir = "Down" Then
        If Val(txtSfidStart) - (minorframes - 1) < 128 Then
            logError "SFID is lower than 128 in minor frame " & Val(txtSfidStart) - 127
            Exit Function
        End If
    Else
        logError "SFID Direction should be 'Up' or 'Down'"
        Exit Function
    End If
    txtSfidStart = Val(txtSfidStart)
    checkSfid = False
End Function
Private Function checkSyncword()
Dim i As Integer

    clearError
    ' check sync words that should not be used
    For i = 0 To 3 - syncwords
        If syncwordbytes(i) <> 0 Then
            logError "Sync word too long"
            checkSyncword = True
            Exit Function
        End If
    Next
    ' check sync words that should be used
    For i = 4 - syncwords To 3
        If syncwordbytes(i) < &H80 Then
            logError "Sync words should be between 0x80 and 0xFF"
            checkSyncword = True
            Exit Function
        End If
    Next
    checkSyncword = False
```

```
End Function

Public Sub update_form()
Dim r, c, row, col, colw As Integer
Dim one As Boolean

    one = (minorframes = 1)
    ' if 1 minor frame, then show the appropriate controls (columns & emphasize channel)
    lblDisplay(0).Visible = one
    lblDisplay(1).Visible = one
    txtCols.Visible = one
    cmbEmphasize.Visible = one

    ' if >1 minor frame, then show the appropriate controls (SFID)
    lblSfid(0).Visible = Not one
    lblSfid(1).Visible = Not one
    cmbSfidDir.Visible = Not one
    txtSfidStart.Visible = Not one

    With grdScantable
        mouseHourglass ' it might take a second
        If minorframes = 1 Then
            .cols = columns + 1
            .rows = minorframelength \ columns + 1
        Else
            .cols = minorframelength + 1
            .rows = minorframes + 1
        End If
        row = .row
        col = .col
        .FixedAlignment(0) = 2 'centered
        .col = 0
        For r = 1 To .rows - 1
            .row = r
            .col = 0
            If minorframes = 1 Then
                .text = "+" & (r - 1) * columns
            Else
                .text = r
            End If
            If r = 1 Or minorframes > 1 Then
                For c = 1 To syncwords
                    .col = c
                    .text = "Sync" & c
                Next
            End If
        Next
        .row = 0

        If .cols * scantableColwidthMax > .Width Then
            colw = (.Width - (1 + syncwords) * scantableColwidthMax - 120) \ (.cols - 1 - syn
cwords)
            If colw < scantableColwidthMin Then colw = scantableColwidthMin
        Else
            colw = scantableColwidthMax
        End If
        For c = 1 To .cols - 1
            .col = c
            If c > syncwords Then
                .ColWidth(c) = colw - 15
                .text = c - syncwords
            Else
                .ColWidth(c) = scantableColwidthMax
                .text = ""
            End If
            .ColAlignment(c) = 2 'centered
            .FixedAlignment(c) = 2 'centered
        Next
        .row = row
        .col = col
        .SelStartCol = 2
        .SelEndCol = 1
        .SelStartRow = 1
        .SelEndRow = 1
```

```
        mouseNormal
    End With
    txtMinorframelength = minorframelength
    txtMinorFrames = minorframes
    txtCols = columns
    txtSyncword_Click
End Sub

Public Sub fillgrdScantable(text As String)
    On Error GoTo End_Function
    With grdScantable
        If .CellSelected Then
            If .SelStartRow = 0 Then .SelStartRow = 1
            If .SelStartCol <= syncwords Then .SelStartCol = syncwords + 1
            .FillStyle = 1
            .text = text
            'hkt one line
            If (.text = "SFID") And (txtMinorframelength > 21) Then .ColWidth(.col) = 1.1 * T
extWidth(.text)
            .FillStyle = 0
'            If .SelStartCol <= syncwords Then update_form
        End If
    End With
End_Function:
End Sub
Private Function closeSmtFile() As Boolean
Dim file As String

        If Not smtChanged Then
        closeSmtFile = True
        Exit Function
    End If
    If commondialogFile.filename = "" Then
        file = "(Untitled)"
    Else
        file = commondialogFile.filename
    End If
    Select Case MsgBox(file & " has changed!" & Chr(13) & Chr(13) & "Save changes?", _
        vbYesNoCancel Or vbQuestion, frmMain.Caption)
        Case vbCancel
            closeSmtFile = False
            Exit Function
        Case vbYes
            mnuFileSave_Click
        Case vbNo
    End Select
    closeSmtFile = True
End Function


Private Sub addtolist(list(), S As String)
    On Error Resume Next
    err.clear
    ReDim Preserve list(1 To UBound(list) + 1)
    If err.Number > 0 Then ReDim list(1 To 1)
    list(UBound(list)) = S
End Sub

Private Function isinlist(list(), S As String) As Boolean
Dim ls As Variant
    isinlist = True
    For Each ls In list
        If ls = S Then Exit Function
    Next
    isinlist = False
End Function

Private Sub generateTestSignals()
Dim i As Integer
    With grdAnalog
        ReDim testSignalSetup(1 To 3, 0 To .rows - 2)

        Set rsSpec = dbSpec.OpenRecordset("gain")
        rsSpec.Index = "value"
```

```
        .col = 1
        For i = 0 To .rows - 2
            .row = i + 1
            rsSpec.Seek "=", .text
            testSignalSetup(.col, i) = rsSpec("test_data")
        Next
        rsSpec.Close

        Set rsSpec = dbSpec.OpenRecordset("offset")
        rsSpec.Index = "value"
        .col = 2
        For i = 0 To .rows - 2
            .row = i + 1
            rsSpec.Seek "=", .text
            testSignalSetup(.col, i) = rsSpec("test_data")
        Next
        rsSpec.Close

        Set rsSpec = dbSpec.OpenRecordset("filter")
        rsSpec.Index = "value"
        .col = 3
        For i = 0 To .rows - 2
            .row = i + 1
            rsSpec.Seek "=", .text
            testSignalSetup(.col, i) = rsSpec("test_data")
        Next
        rsSpec.Close
    End With
End Sub
Private Sub updateFormCaption(ByVal filename As String)
Dim i As Integer

    For i = Len(filename) To 1 Step -1
        If Mid(filename, i, 1) = "\" Then
            filename = Mid(filename, i + 1)
            Exit For
        End If
    Next
    frmMain.Caption = "SMT Configuration Tool - " & filename
End Sub
Private Function createAsciiFile()
Dim asciiFileLength, bitrate, PNSeedLSB, PNSeedMSB As Integer
Dim scantableStartLSB, scantableStartMSB, transmitFreq As Integer
Dim i, j, row, col, b, v, addr, recordCount As Integer
Dim data() As Integer
Dim descrip() As String
Dim rsSpecProp As Recordset
Dim prop As String
Dim string1, string2 As String

    If checkAll Then
        Beep
        createAsciiFile = False
        Exit Function
    End If
    mouseHourglass
    asciiFile = ""
    Set rsSpec = dbSpec.OpenRecordset("system")

    ' get system parameters addresses
    asciiFileLength = rsSpec("asciiFileLength")
    bitrate = rsSpec("bitrate")
    PNSeedLSB = rsSpec("PNSeedLSB")
    PNSeedMSB = rsSpec("PNSeedMSB")
    scantableStartLSB = rsSpec("scantableStartLSB")
    scantableStartMSB = rsSpec("scantableStartMSB")
    transmitFreq = rsSpec("transmitFreq")
    rsSpec.Close

    ' get ascii table
    Set rsSpec = dbSpec.OpenRecordset("ascii")
    rsSpec.MoveLast
    recordCount = rsSpec.recordCount
    rsSpec.MoveFirst
```

```
    ReDim data(recordCount)
    ReDim descrip(recordCount)
    addr = 0
    Do Until rsSpec.EOF
        If VarType(rsSpec("description")) = vbString Then
            descrip(addr) = rsSpec("description")
        Else
            descrip(addr) = ""
        End If
        data(addr) = rsSpec("data")
        addr = addr + 1
        If addr > recordCount Then
            ' this shouldn't be necessary, but rsSpec.recordCount is not always right!
            recordCount = addr + 10
            ReDim Preserve data(recordCount)
            ReDim Preserve descrip(recordCount)
        End If
        rsSpec.MoveNext
    Loop
    ' this shouldn't be necessary, but rsSpec.recordCount is not always right!
    ' very weird
    recordCount = addr
    rsSpec.Close

    ' fill in bitrate
    Set rsSpec = dbSpec.OpenRecordset("bitrate")
    rsSpec.Index = "value"
    rsSpec.Seek "=", cmbBitrate
    data(bitrate) = rsSpec("smt_data")
    descrip(bitrate) = descrip(bitrate) & " -> " & cmbBitrate
    rsSpec.Close

    ' fill in transmitFreq (line 61)
    Set rsSpec = dbSpec.OpenRecordset("transmitFreq")
    rsSpec.Index = "value"
    rsSpec.Seek "=", cmbTransmitFreq
    ' data(transmitFreq) = rsSpec("smt_data")
    descrip(transmitFreq) = descrip(transmitFreq) & " -> " & cmbTransmitFreq
    ' fill in the calculated values for lines 61 & 62 (Modified by Peter Evdokiou)
    ' Check to see the type of Modulation Scheme selected
    string1 = InStr(cmbBitrate, "QPSK")
    string2 = InStr(cmbBitrate, "FSK")
    If string1 = 0 Then     ' if Mod Scheme is FSK then set data for lines 61 & 62 as

        data(61) = Int(Val(cmbTransmitFreq) * 0.03125)
        data(62) = Val(cmbTransmitFreq) * 8 - (256 * Int(Val(cmbTransmitFreq) * 0.03125))

    Else        ' if Mod Scheme is QPSK then set data for lines 61 & 62 as

        data(61) = 9
        data(62) = rsSpec("smt_data")

    End If
    rsSpec.Close

    ' fill in the values of lines 65 & 66 according to type of Modulation scheme
    Set rsSpec = dbSpec.OpenRecordset("bitrate")
    rsSpec.Index = "value"
    rsSpec.Seek "=", cmbBitrate
    string1 = InStr(cmbBitrate, "QPSK")
    string2 = InStr(cmbBitrate, "FSK")
    If string1 = 0 Then
        data(65) = 1    ' Value taken if Mode Scheme selected is FSK
        data(66) = 3    ' Value taken if Mode Scheme selected is FSK
    Else
        data(65) = 0    ' Value taken if Mode Scheme selected is QPSK
        data(66) = 35   ' Value taken if Mode Scheme selected is QPSK
    End If
    rsSpec.Close


    ' fill in PNSeed, scantableStart
    i = Val("&H" & txtPNSeed)
    data(PNSeedLSB) = i Mod 256
```

```
    data(PNSeedMSB) = i \ 256
    descrip(PNSeedMSB) = descrip(PNSeedMSB) & " -> 0x" & Hex(i)
    b = asciiFileLength - minorframes * minorframelength
    data(scantableStartLSB) = b Mod 256
    data(scantableStartMSB) = b \ 256

    ' fill in analog properties
    Set rsSpec = dbSpec.OpenRecordset("channels")
    rsSpec.Index = "ID"
    For i = 1 To grdAnalog.cols - 1
        grdAnalog.col = i
        grdAnalog.row = 0
        prop = grdAnalog
        Set rsSpecProp = dbSpec.OpenRecordset(prop)
        rsSpecProp.Index = "value"
        For j = 1 To grdAnalog.rows - 1
            grdAnalog.row = j
            grdAnalog.col = 0
            rsSpec.Seek "=", grdAnalog
            grdAnalog.col = i
            rsSpecProp.Seek "=", grdAnalog
            addr = rsSpec(prop & "_address")
            data(addr) = rsSpecProp("smt_data")
            descrip(addr) = descrip(addr) & " = " & grdAnalog
        Next
        rsSpecProp.Close
    Next

    For addr = 0 To recordCount - 1
        If (data(addr) < 0) Then
            MsgBox "SMT Specification Database invalid in line " & addr + 1, _
                vbOKOnly + vbExclamation, "Error"
            createAsciiFile = False
            Exit Function
        End If
        asciiFile = asciiFile & _
            Format(addr, "@@@@") & "  " & _
            Format(data(addr), "@@@") & "  " & _
            Format("(0x" & Hex(data(addr)), "@@@@@@") & ") " & _
            descrip(addr) & Chr(13) & Chr(10)
    Next
    ReDim descrip(1)
    Erase data

    ' recordset("channels") is still open
    saveScantable
    For row = 0 To minorframes - 1
        For col = 0 To minorframelength - 1
            If col < syncwords Then
                v = syncwordbytes(4 - syncwords + col)
                descrip(0) = "Sync word #" & col + 1
                If col = 0 Then descrip(0) = descrip(0) & " ====================="
            Else
                rsSpec.Seek "=", scantable(row, col)
                v = rsSpec("scantable_value")
                descrip(0) = rsSpec("description")
                If v = -1 Then
                    'SFID
                    If cmbSfidDir = "Up" Then
                        v = txtSfidStart + row
                    Else
                        v = txtSfidStart - row
                    End If
                End If
            End If
            asciiFile = asciiFile & _
                Format(b, "@@@@") & "  " & _
                Format(v, "@@@") & "  " & _
                Format("(0x" & Hex(v), "@@@@@@") & ") " & _
                descrip(0) & Chr(13) & Chr(10)
            b = b + 1
        Next
    Next
    rsSpec.Close
```

```
    mouseNormal
    createAsciiFile = True
End Function

Private Sub cmbBitrate_Click()
    'by hkt except line 3
Dim stringQPSK As Integer   'string has qpsk modulation
Dim string_kbs As Integer   'string has 200kb/s format
'Dim string_is_QPSK As Boolean      'FSK or QPSK
'Dim string_string_is_kbs As Boolean       '200 kb/s or 2Mb/s
Dim def_qpsk, def_fsk As String    'default value for 2Mbs and 200kbs Transmission
    'original line
    smtChanged = True
    stringQPSK = InStr(cmbBitrate, "QPSK")  'check if string has QPSK
    If stringQPSK > 0 Then
        string_is_QPSK = True              'set flag if it has
    Else: string_is_QPSK = False
    End If

    string_kbs = InStr(cmbBitrate, "200 kb/s")  'check if string has 200 kbs
    If string_kbs > 0 Then      'set flag if it has
        string_is_kbs = True
    Else: string_is_kbs = False
    End If
    'if QPSK then enable PN Sêed, else disable
    If string_is_QPSK = True Then
        txtPNSeed.Enabled = True
    Else:
        txtPNSeed.Enabled = False
    End If

    'clear transmit freq and reload
    cmbTransmitFreq.clear
    Set rsSpec = dbSpec.OpenRecordset("transmitFreq")
    If string_is_QPSK = True Then   'if QPSK then reload only these freq
        Do Until rsSpec.EOF
            If rsSpec("value") = "2310 Mhz" Then
                cmbTransmitFreq.AddItem (rsSpec("value"))
                rsSpec.MoveNext
            ElseIf rsSpec("value") = "2320 Mhz" Then
                cmbTransmitFreq.AddItem (rsSpec("value"))
                rsSpec.MoveNext
            ElseIf rsSpec("value") = "2330 Mhz" Then
                cmbTransmitFreq.AddItem (rsSpec("value"))
                rsSpec.MoveNext
            ElseIf rsSpec("value") = "2340 Mhz" Then
                cmbTransmitFreq.AddItem (rsSpec("value"))
                rsSpec.MoveNext
            ElseIf rsSpec("value") = "2350 Mhz" Then
                cmbTransmitFreq.AddItem (rsSpec("value"))
                rsSpec.MoveNext
            ElseIf rsSpec("value") = "2360 Mhz" Then
                cmbTransmitFreq.AddItem (rsSpec("value"))
                rsSpec.MoveNext
            ElseIf rsSpec("value") = "2370 Mhz" Then
                cmbTransmitFreq.AddItem (rsSpec("value"))
                rsSpec.MoveNext
            ElseIf rsSpec("value") = "2380 Mhz" Then
                cmbTransmitFreq.AddItem (rsSpec("value"))
                rsSpec.MoveNext
            Else: rsSpec.MoveNext
            End If
        Loop
    Else        'if FSK then load all
        Do Until rsSpec.EOF
            cmbTransmitFreq.AddItem (rsSpec("value"))
            rsSpec.MoveNext
        Loop
    End If
    rsSpec.Close    'close datatable
    Set rsSpec = dbSpec.OpenRecordset("startfreq")  ' give combo box default value
    def_qpsk = rsSpec("value")  'default value of QPSK
    rsSpec.MoveNext
    def_fsk = rsSpec("value")   'default value of FSK
```

```
    rsSpec.Close

    If string_is_QPSK = True Then
        cmbTransmitFreq.text = def_qpsk
    Else: cmbTransmitFreq.text = def_fsk
    End If
End Sub

Private Sub cmbEmphasize_Click()
Dim i, first As Integer
'hkt next 7 line
Dim string_normal As Integer

    string_normal = InStr(cmbEmphasize, "Normal")    'if normal chosen then normalise
                                                     'minor frame
    If string_normal > 0 Then
        txtCols = minorframelength
        txtCols_Click
        Exit Sub
    End If

    first = -1
    With grdScantable
        For i = 0 To minorframelength - 1

            '.col = i Mod columns + 1 remmed by hkt
            '.row = i \ columns + 1   remmed by hkt
            'hkt next 2
            .col = i Mod (columns + 1)
            '.row = i Mod (columns + 1)

            If .text = cmbEmphasize Then
                If first = -1 Then
                    first = i
                Else
                    txtCols = i - first
                    txtCols_Click
                    GoTo found
                End If
            End If
        Next
        If first = -1 Then Exit Sub
        txtCols = minorframelength
        txtCols_Click
found:
        .SelStartCol = first Mod columns + 1
        .SelEndCol = first Mod columns + 1
        .SelStartRow = 1
        .SelEndRow = .rows - 1
    End With
End Sub

Private Sub cmbSyncwordLength_Click()
Dim newsyncwords As Integer
    newsyncwords = Val(cmbSyncwordLength) \ 8
    If newsyncwords <> syncwords Then
        If newsyncwords < syncwords Then
            With grdScantable
                .SelStartCol = newsyncwords + 1
                .SelEndCol = syncwords
                .SelStartRow = 1
                .SelEndRow = .rows - 1
                .FillStyle = 1
                .text = ""
                .FillStyle = 0
                .SelEndCol = 1
                .SelEndRow = 1
            End With
        End If
        syncwords = newsyncwords
        update_form
        smtChanged = True
    End If
End Sub
```

```
Private Sub cmdButtonBar_Click(Index As Integer)
    Select Case Index
        Case 0
            mnuFileNew_Click
        Case 1
            mnuFileOpen_Click
        Case 2
            mnuFileSave_Click
        Case 3
            mnuFileSaveAs_Click
        Case 4
            mnuFileGenerateScantable_Click
        Case 5
            mnuFileViewErrors_Click
        Case 6
            mnuFileCreateAscii_Click
        Case 7
            mnuFileTestsignalgenerator_Click
        Case 8
            mnuGenerateReport_Click
    End Select
End Sub


Private Sub Form_Activate()
    Unload frmgen
End Sub

Private Sub Form_Initialize()
    minorframelength = 4
    columns = 4
    minorframes = 1
End Sub

Private Sub cmbSfidDir_Click()
    checkSfid
    smtChanged = True
End Sub

Private Sub cmbTransmitFreq_Click()
    smtChanged = True
End Sub

Private Sub cmdCopyEntire_Click()
Dim c As Integer
    c = columns
    txtCols = minorframelength
    txtCols_Click
    With grdScantable
        .SelStartCol = 1
        .SelEndCol = .cols - 1
        .SelStartRow = 1
        .SelEndRow = .rows - 1
        Clipboard.SetText .Clip + Chr(13)
        .SelEndCol = 1
        .SelEndRow = 1
    End With
    txtCols = c
    txtCols_Click
End Sub

Private Sub cmdPasteEntire_Click()
Dim clipdata As String
Dim i, rows, cols, curcols As Integer
Dim content As Boolean

    smtChanged = True
    clipdata = Clipboard.GetText
    ' clipboard analysis; cols are separated by tab
    ' (tab=chr(9)), rows by cr/lf (cr=chr(13))
    rows = 0
    cols = 0
    curcols = 1
```

```
    For i = 1 To Len(clipdata)
        If Asc(Mid(clipdata, i, 1)) >= 32 Then
            content = True
        End If
        If Mid(clipdata, i, 1) = Chr(9) Then
            curcols = curcols + 1
        End If
        If Mid(clipdata, i, 1) = Chr(13) Then
            rows = rows + 1
            ' take largest value of all rows:
            If curcols > cols Then cols = curcols
            curcols = 1
            content = False
        End If
    Next
    ' if the last line doesn't end with a cr:
    If content Then rows = rows + 1
    If rows = 0 Or cols <= syncwords Then
        Beep
        Exit Sub
    End If

    grdScantable.rows = rows + 1
    grdScantable.cols = cols + 1
    With grdScantable
        .SelStartCol = 1
        .SelEndCol = cols
        .SelStartRow = 1
        .SelEndRow = rows
        ' copy data:
        .Clip = clipdata
        .SelEndCol = 1
        .SelEndRow = 1
    End With

    minorframelength = cols
    columns = cols
    minorframes = rows
    update_form
End Sub

Private Sub Form_Load()
Dim i, j As Integer
Dim control As Variant
Dim rsSpecID As String

    ' position window in the middle of the screen:
    frmMain.Left = Screen.Width / 2 - frmMain.Width / 2
    frmMain.Top = Screen.Height / 2 - frmMain.Height / 2
    ' frmerrors.Hide
    If sSmtSpecDatabasefile = "" Then
        sSmtSpecDatabasefile = App.Path + "\SMTSPEC.MDB"
    End If
    'hkt 2 lines
    'set the scroll bars for laptop useage
    vsViewPort1.VirtualWidth = vsViewPort1.Width + 2000
    vsViewPort1.VirtualHeight = vsViewPort1.Height + 2000

    On Error GoTo dbSpecError
    Set dbSpec = DBEngine.Workspaces(0).OpenDatabase(sSmtSpecDatabasefile, , True)
    Set rsSpec = dbSpec.OpenRecordset("system")
    maxscantablelength = rsSpec("scantableMaxLength")
    rsSpec.Close

    Set rsSpec = dbSpec.OpenRecordset("bitrate")
    i = 0
    Do Until rsSpec.EOF
        cmbBitrate.AddItem (rsSpec("value"))
        rsSpec.MoveNext
        i = i + 1
    Loop
    rsSpec.Close
    'rem by hkt next 6
    'Set rsSpec = dbSpec.OpenRecordset("transmitFreq")
```

```
    'Do Until rsSpec.EOF
    '    cmbTransmitFreq.AddItem (rsSpec("value"))
    '    rsSpec.MoveNext
    'Loop
    'rsSpec.Close

    Set rsSpec = dbSpec.OpenRecordset("channels")
    rsSpec.Index = "ID"
    i = 1
    grdAnalog.col = 0
    Erase eachminorframes
    Erase channels
    Erase analogs
    Erase digitals
    Do Until rsSpec.EOF
        rsSpecID = rsSpec("ID")
        addtolist channels, rsSpecID
        If rsSpec("eachminorframe") Then
            addtolist eachminorframes, rsSpecID
        ElseIf rsSpec("Analog") Then
            grdAnalog.rows = grdAnalog.rows + 1
            grdAnalog.row = grdAnalog.rows - 1
            grdAnalog = rsSpecID
            addtolist analogs, rsSpecID
        Else
            addtolist digitals, rsSpecID
        End If
        cmbEmphasize.AddItem (rsSpecID)
        Load mnuScantableEntry(i)
        mnuScantableEntry(i).Caption = rsSpecID
        i = i + 1
        rsSpec.MoveNext
    Loop

    rsSpec.Close
    'hkt one line
    'add "normal" to "emphasize" box
    cmbEmphasize.AddItem "Normal"
    With grdAnalog
        .FixedAlignment(0) = 2
        .ColWidth(0) = 400
        .ColWidth(1) = 800
        .ColWidth(2) = 900
        .ColWidth(3) = 900
        .FixedRows = 1
        .row = 0
        .col = 1
        .text = "Gain"
        .col = 2
        .text = "Offset"
        .col = 3
        .text = "Filter"
    End With
    mnuScantableEntry(1000).Visible = False

    Set rsSpec = dbSpec.OpenRecordset("gain")
'    rsSpec.Index = "value"
    i = 1
    Do

        mnuAnalogGainEntry(i).Caption = rsSpec("value")
        rsSpec.MoveNext
        If rsSpec.EOF Then Exit Do
        i = i + 1
        Load mnuAnalogGainEntry(i)
    Loop
    rsSpec.Close

    Set rsSpec = dbSpec.OpenRecordset("offset")
'    rsSpec.Index = "value"
    i = 1
    Do
        mnuAnalogOffsetEntry(i).Caption = rsSpec("value")
        rsSpec.MoveNext
```

```
        If rsSpec.EOF Then Exit Do
        i = i + 1
        Load mnuAnalogOffsetEntry(i)
    Loop
    rsSpec.Close

    Set rsSpec = dbSpec.OpenRecordset("filter")
'    rsSpec.Index = "value"
'hkt
'load in freq for pop up menu analog filter
    i = 1
    Do
        mnuAnalogFilterEntry(i).Caption = rsSpec("value")
        mnuAnalogFilterkbsEntry(i).Caption = rsSpec("value")
        mnuAnalogFiltermbsEntry(i).Caption = rsSpec("value")

        rsSpec.MoveNext
        If rsSpec.EOF Then Exit Do
        i = i + 1
        Load mnuAnalogFilterEntry(i)
        Load mnuAnalogFilterkbsEntry(i)
        Load mnuAnalogFiltermbsEntry(i)


        Loop
    rsSpec.Close



    'by hkt 28
    'filter the freq required for 200 kb/s format
    filter_item_max = i     'max number in filter
    For i = 1 To filter_item_max
        Select Case mnuAnalogFilterkbsEntry(i).Caption
            Case "122.1 Hz"
            Case "244.1 Hz"
            Case "488.3 Hz"
            Case "976.6 Hz"
            Case "1953 Hz"
            Case "3906 Hz"
            Case Else
                mnuAnalogFilterkbsEntry(i).Visible = False
                'mnuAnalogFilterkbsEntry(i).Enabled = False
        End Select
    Next i

    'by hkt
    'filter freq for 2 Mb/s format
    For i = 1 To filter_item_max
        Select Case mnuAnalogFiltermbsEntry(i).Caption
            Case "152.5 Hz"
            Case "305 Hz"
            Case "610 Hz"
            Case "1220 Hz"
            Case "2440 Hz"
            Case "4880 Hz"
            Case "19530 Hz"
            Case "9766 Hz"
            Case Else
                mnuAnalogFiltermbsEntry(i).Visible = False
                'mnuAnalogFiltermbsEntry(i).Enabled = False
        End Select
    Next i


        On Error GoTo 0

    cmbSyncwordLength.ListIndex = 0
    smtChanged = False
    mnuFileNew_Click
    update_form

    Exit Sub
dbSpecError:
```

```
    On Error Resume Next
    MsgBox "Invalid SMT Specification Database", vbOKOnly + vbExclamation, "Error"
    For Each control In frmMain
        If Val(control.Tag) = 1 Then control.Enabled = False
    Next
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    If closeSmtFile = False Then Cancel = True
    If UnloadMode <> vbFormCode Then End
End Sub

Private Sub grdAnalog_KeyDown(KeyCode As Integer, Shift As Integer)
'var by hkt
Dim string_kbs, i, j As Integer
    smtChanged = True
    With grdAnalog
        Select Case KeyCode
            Case vbKeyReturn, vbKeySpace
                Select Case .col
                    Case 1
                        PopupMenu mnuAnalogGain, vbPopupMenuCenterAlign, _
                            fraAnalog.Left + .ColPos(.col) + .Left, _
                            fraAnalog.Top + .RowPos(.row) + .Top
                    Case 2
                        PopupMenu mnuAnalogOffset, vbPopupMenuCenterAlign, _
                            fraAnalog.Left + .ColPos(.col) + .Left, _
                            fraAnalog.Top + .RowPos(.row) + .Top
                    Case 3
                        'hkt
                        If string_is_kbs = True Then
                        PopupMenu mnuAnalogFilterkbs, vbPopupMenuCenterAlign, _
                            fraAnalog.Left + .ColPos(.col) + .Left, _
                            fraAnalog.Top + .RowPos(.row) + .Top

                        Else
                            PopupMenu mnuAnalogFiltermbs, vbPopupMenuCenterAlign, _
                                fraAnalog.Left + .ColPos(.col) + .Left, _
                                fraAnalog.Top + .RowPos(.row) + .Top
                        End If
                End Select
        End Select
    End With
End Sub

Private Sub grdAnalog_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single
)
    With grdAnalog
        grdAnalogIn = X <= .ColPos(.cols - 1) + .ColWidth(.cols - 1) And _
            Y <= .RowPos(.rows - 1) + .RowHeight(.rows - 1)
    End With

End Sub

Private Sub grdAnalog_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)

    With grdAnalog

        If Button = vbLeftButton And grdAnalogIn Then
            smtChanged = True
            Select Case grdAnalog.col
                Case 1
                    PopupMenu mnuAnalogGain
                Case 2
                    PopupMenu mnuAnalogOffset
                Case 3
                    'PopupMenu mnuAnalogFilter
                    'by hkt next 5


                    If string_is_kbs = True Then
                            PopupMenu mnuAnalogFilterkbs
                    Else
                            PopupMenu mnuAnalogFiltermbs
```

```
                    End If
                End Select
            End If
        End With
End Sub

Private Sub grdAnalog_SelChange()
    grdAnalog.SelEndCol = grdAnalog.SelStartCol
End Sub

Private Sub grdScantable_KeyDown(KeyCode As Integer, Shift As Integer)
    smtChanged = True
    With grdScantable
        Select Case KeyCode
            Case vbKeyDelete, vbKeyBack
                fillgrdScantable ("")
            Case vbKeyReturn, vbKeySpace
                PopupMenu mnuScanTable, vbPopupMenuCenterAlign, _
                    .ColPos(.col) + .Left, .RowPos(.row) + .Top
        End Select
    End With
End Sub

Private Sub grdScantable_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Sin
gle)
    With grdScantable
        grdScantableIn = _
            X <= .ColPos(.cols - 1) + .ColWidth(.cols - 1) And _
            Y <= .RowPos(.rows - 1) + .RowHeight(.rows - 1)
    End With
End Sub

Private Sub grdScantable_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Singl
e)
    With grdScantable
        'If Button = vbLeftButton And grdScantableIn Then
        If Button = vbLeftButton Then
            smtChanged = True
            PopupMenu mnuScanTable
        End If
    End With
End Sub



Private Sub lblErrorSound_Click()
    chkErrorSound = 1 - chkErrorSound
End Sub

Private Sub mnuAnalogFilterEntry_Click(Index As Integer)
    smtChanged = True
    With grdAnalog
        If .SelStartRow = 0 Then .SelStartRow = 1
        If .CellSelected Then
            .FillStyle = 1
            .text = mnuAnalogFilterEntry(Index).Caption
            .FillStyle = 0
        End If
    End With
    checkAnalog
End Sub

Private Sub mnuAnalogFilterkbsEntry_Click(Index As Integer)
'by hkt
'enter the text into the cell selected in grid analog filter
'for 200 kb/s menu
    smtChanged = True
    With grdAnalog
        If .SelStartRow = 0 Then .SelStartRow = 1
        If .CellSelected Then
            .FillStyle = 1
            .text = mnuAnalogFilterkbsEntry(Index).Caption
```

```
            .FillStyle = 0
        End If
    End With
    checkAnalog

End Sub

Private Sub mnuAnalogFiltermbsEntry_Click(Index As Integer)
'by hkt
'enter text in cell chosen from the 2 Mb/s menu
    smtChanged = True
    With grdAnalog
        If .SelStartRow = 0 Then .SelStartRow = 1
        If .CellSelected Then
            .FillStyle = 1
            .text = mnuAnalogFiltermbsEntry(Index).Caption
            .FillStyle = 0
        End If
    End With
    checkAnalog
End Sub

Private Sub mnuAnalogGainEntry_Click(Index As Integer)
    smtChanged = True
    With grdAnalog
        If .SelStartRow = 0 Then .SelStartRow = 1
        If .CellSelected Then
            .FillStyle = 1
            .text = mnuAnalogGainEntry(Index).Caption
            .FillStyle = 0
        End If
    End With
    checkAnalog
End Sub

Private Sub mnuAnalogOffsetEntry_Click(Index As Integer)
    smtChanged = True
    With grdAnalog
        If .SelStartRow = 0 Then .SelStartRow = 1
        If .CellSelected Then
            .FillStyle = 1
            .text = mnuAnalogOffsetEntry(Index).Caption
            .FillStyle = 0
        End If
    End With
    checkAnalog
End Sub

Private Sub mnuFileCreateAscii_Click()
    If Not createAsciiFile Then Exit Sub
    Load frmAscii
    frmAscii.txtAscii = asciiFile
    frmAscii.Show
End Sub

Private Sub mnuFileGenerateScantable_Click()
    cmbSyncwordLength.ListIndex = 0
    clearError
    If checkBitrate Then
        Beep
        Exit Sub
    End If
    frmgen.Show
End Sub

Private Sub mnuFileNew_Click()
Dim i, j As Integer

    If closeSmtFile = False Then Exit Sub
    updateFormCaption "(Untitled)"
    cmbBitrate.ListIndex = -1
    txtPNSeed = "7FF"
    cmbTransmitFreq.ListIndex = -1
    cmbSyncwordLength.ListIndex = 0
```

```
    txtSyncword = "EB90"
    minorframelength = 10
    columns = 10
    minorframes = 1
    txtSfidStart = ""
    cmbSfidDir.ListIndex = -1
    txtProject = ""
    txtDescription = ""
    txtAuthor = ""

    update_form
    With grdScantable
        .SelStartCol = syncwords + 1
        .SelEndCol = .cols - 1
        .SelStartRow = 1
        .SelEndRow = .rows - 1
        .FillStyle = 1
        .text = ""
        .FillStyle = 0
        .SelEndCol = 1
        .SelEndRow = 1
    End With
    With grdAnalog
        .SelStartCol = 1
        .SelEndCol = .cols - 1
        .SelStartRow = 1
        .SelEndRow = .rows - 1
        .FillStyle = 1
        .text = ""
        .FillStyle = 0
        .SelEndCol = 1
        .SelEndRow = 1
    End With

    smtChanged = False
End Sub

Private Sub mnuFileOpen_Click()
Dim i, j As Integer

    If closeSmtFile = False Then Exit Sub
    With commondialogFile
        .Flags = cdlOFNFileMustExist Or cdlOFNHideReadOnly _
            Or cdlOFNNoReadOnlyReturn
            .DialogTitle = "Open"
        .filename = ""
        On Error GoTo errFileOpenCancel
        .ShowOpen
        On Error GoTo 0
        If .filename = "" Then Exit Sub
    End With

    On Error GoTo errFileOpenFile
    mouseHourglass
    Set dbSmt = DBEngine.Workspaces(0).OpenDatabase(commondialogFile.filename)

    On Error GoTo errFileOpenRS
    Set rsSmt = dbSmt.OpenRecordset("system_setup")
    On Error GoTo errFileOpen
    cmbBitrate = rsSmt("bitrate")
    txtPNSeed = rsSmt("pnseed")
    cmbTransmitFreq = rsSmt("frequency")
    cmbSyncwordLength.ListIndex = -1
    For i = 0 To cmbSyncwordLength.ListCount - 1
        If Val(cmbSyncwordLength.list(i)) = rsSmt("synclength") Then
            cmbSyncwordLength.ListIndex = i
            Exit For
        End If
    Next
    txtSyncword = rsSmt("sync")
    minorframelength = rsSmt("minorFramelength")
    columns = minorframelength
    grdScantable.cols = minorframelength + 1
    minorframes = rsSmt("minorFrames")
```

```
    grdScantable.rows = minorframes + 1
    If minorframes <> 1 Then
        txtSfidStart = rsSmt("sfidStart")
        cmbSfidDir = rsSmt("sfidDir")
    End If
    On Error Resume Next
    txtProject = ""
    txtProject = rsSmt("project")
    txtDescription = ""
    txtDescription = rsSmt("description")
    txtAuthor = ""
    txtAuthor = rsSmt("author")
    rsSmt.Close
    mouseHourglass

    On Error GoTo errFileOpenRS
    Set rsSmt = dbSmt.OpenRecordset("analog")
    On Error GoTo errFileOpen
    With grdAnalog
        For i = 1 To .rows - 1
            If rsSmt.EOF Then Exit For
            .row = i
            .col = 1
            .text = rsSmt("Gain")
            .col = 2
            .text = rsSmt("Offset")
            .col = 3
            .text = rsSmt("Filter")
            rsSmt.MoveNext
       Next
    End With
    rsSmt.Close

    On Error GoTo errFileOpenRS
    Set rsSmt = dbSmt.OpenRecordset("scantable")
    On Error GoTo errFileOpen
    With grdScantable
        For i = 1 To .rows - 1
            .row = i
            For j = 1 To .cols - 1
                If rsSmt.EOF Then GoTo scantableDone
                .col = j
                .text = rsSmt("value")
                rsSmt.MoveNext
            Next
        Next
    End With
scantableDone:
    rsSmt.Close

    dbSmt.Close
    update_form
    mouseNormal
    On Error GoTo 0
    updateFormCaption commondialogFile.filename
    smtChanged = False
errFileOpenCancel:
    Exit Sub

errFileOpen:
    MsgBox "Error opening file:" & Chr(13) & _
        err.Description & Chr(13) & _
        "table=" & rsSmt.Name, vbOKOnly + vbExclamation, "Error"
    dbSmt.Close
    GoTo errFileOpenFinish
errFileOpenRS:
    dbSmt.Close
errFileOpenFile:
    MsgBox "Error opening file:" & Chr(13) & _
        err.Description, vbOKOnly + vbExclamation, "Error"
errFileOpenFinish:
    On Error GoTo 0
    smtChanged = False
    mnuFileNew_Click
```

```
    Exit Sub
End Sub

Private Sub mnuFileOpenSmtDeviceSpecification_Click()
    If closeSmtFile = False Then Exit Sub
    With commondialogSpecFile
        .Flags = cdlOFNFileMustExist Or cdlOFNHideReadOnly _
            Or cdlOFNNoReadOnlyReturn
        On Error GoTo ErrFileOpenSmtDeviceSpecificationCancel
        .ShowOpen
        On Error GoTo 0
        If .filename = "" Then Exit Sub
        sSmtSpecDatabasefile = .filename
    End With
'    frmMainReload.Hide
    smtChanged = False
    Unload frmMain
    frmMain.Show
ErrFileOpenSmtDeviceSpecificationCancel:
End Sub

Private Sub mnuFileSave_Click()
Dim i, j As Integer
Dim tdSmt As TableDef
Dim fldSmt As Field

    If commondialogFile.filename = "" Then
        mnuFileSaveAs_Click
        Exit Sub
    End If
    On Error GoTo errFileSaveOpen
    Set dbSmt = DBEngine.Workspaces(0).OpenDatabase(commondialogFile.filename)
    On Error GoTo errFileSaveFatal

    dbSmt.BeginTrans

    On Error GoTo errFileSave_system_setup
    Set rsSmt = dbSmt.OpenRecordset("system_setup")
    On Error GoTo errFileSaveFatal
    If rsSmt.EOF Then rsSmt.AddNew Else rsSmt.Edit
    rsSmt("bitrate") = cmbBitrate
    rsSmt("pnseed") = txtPNSeed
    rsSmt("frequency") = cmbTransmitFreq
    rsSmt("synclength") = Val(cmbSyncwordLength)
    rsSmt("sync") = txtSyncword
    rsSmt("minorFramelength") = minorframelength
    rsSmt("minorFrames") = minorframes
    rsSmt("sfidStart") = txtSfidStart
    rsSmt("sfidDir") = cmbSfidDir
    rsSmt("project") = txtProject
    rsSmt("description") = txtDescription
    rsSmt("author") = txtAuthor
    rsSmt.Update
    rsSmt.Close

    On Error GoTo errFileSave_scantable
    Set rsSmt = dbSmt.OpenRecordset("scantable")
    On Error GoTo errFileSaveFatal
    With grdScantable
        For i = 1 To .rows - 1
            .row = i
            For j = 1 To .cols - 1
                .col = j
                If rsSmt.EOF Then rsSmt.AddNew Else rsSmt.Edit
                rsSmt("value") = .text
                rsSmt.Update
                If Not rsSmt.EOF Then rsSmt.MoveNext
            Next
        Next
    End With
    Do Until rsSmt.EOF
        rsSmt.Delete
        rsSmt.MoveNext
    Loop
```

```
    rsSmt.Close

    On Error GoTo errFileSave_analog
    Set rsSmt = dbSmt.OpenRecordset("analog")
    On Error GoTo errFileSaveFatal
    With grdAnalog
        For i = 1 To .rows - 1
            If rsSmt.EOF Then rsSmt.AddNew Else rsSmt.Edit
            .row = i
            .col = 0
            rsSmt("Channel") = .text
            .col = 1
            rsSmt("Gain") = .text
            .col = 2
            rsSmt("Offset") = .text
            .col = 3
            rsSmt("Filter") = .text
            rsSmt.Update
            If Not rsSmt.EOF Then rsSmt.MoveNext
       Next
    End With
    rsSmt.Close
    dbSmt.CommitTrans
    dbSmt.Close
    On Error GoTo 0
    smtChanged = False

    Exit Sub

errFileSaveOpen:
    On Error GoTo errFileSaveCreate
    'create database
    Set dbSmt = DBEngine.Workspaces(0).CreateDatabase(commondialogFile.filename, _
        dbLangGeneral)
    On Error GoTo 0
    Resume Next

errFileSave_system_setup:
    On Error GoTo errFileSaveFatal
    'create system_setup
    Set tdSmt = dbSmt.CreateTableDef("system_setup")
    Set fldSmt = tdSmt.CreateField("bitrate", dbText, 50)
    fldSmt.AllowZeroLength = True
    tdSmt.Fields.Append fldSmt
    Set fldSmt = tdSmt.CreateField("pnseed", dbText, 4)
    fldSmt.AllowZeroLength = True
    tdSmt.Fields.Append fldSmt
    Set fldSmt = tdSmt.CreateField("frequency", dbText, 10)
    fldSmt.AllowZeroLength = True
    tdSmt.Fields.Append fldSmt
    Set fldSmt = tdSmt.CreateField("minorFrameLength", dbInteger)
    tdSmt.Fields.Append fldSmt
    Set fldSmt = tdSmt.CreateField("minorFrames", dbInteger)
    tdSmt.Fields.Append fldSmt
    Set fldSmt = tdSmt.CreateField("synclength", dbInteger)
    tdSmt.Fields.Append fldSmt
    Set fldSmt = tdSmt.CreateField("sync", dbText, 8)
    fldSmt.AllowZeroLength = True
    tdSmt.Fields.Append fldSmt
    Set fldSmt = tdSmt.CreateField("sfidStart", dbText, 5)
    fldSmt.AllowZeroLength = True
    tdSmt.Fields.Append fldSmt
    Set fldSmt = tdSmt.CreateField("sfidDir", dbText, 4)
    fldSmt.AllowZeroLength = True
    tdSmt.Fields.Append fldSmt
    Set fldSmt = tdSmt.CreateField("project", dbText, 80)
    fldSmt.AllowZeroLength = True
    tdSmt.Fields.Append fldSmt
    Set fldSmt = tdSmt.CreateField("description", dbText, 80)
    fldSmt.AllowZeroLength = True
    tdSmt.Fields.Append fldSmt
    Set fldSmt = tdSmt.CreateField("author", dbText, 80)
    fldSmt.AllowZeroLength = True
    tdSmt.Fields.Append fldSmt
```

```
    dbSmt.TableDefs.Append tdSmt
    Resume 0

errFileSave_scantable:
    On Error GoTo errFileSaveFatal
    'create scantable
    Set tdSmt = dbSmt.CreateTableDef("scantable")
    Set fldSmt = tdSmt.CreateField("value", dbText, 5)
    fldSmt.AllowZeroLength = True
    tdSmt.Fields.Append fldSmt
    dbSmt.TableDefs.Append tdSmt
    Resume 0

errFileSave_analog:
    On Error GoTo errFileSaveFatal
    'create analog
    Set tdSmt = dbSmt.CreateTableDef("analog")
    Set fldSmt = tdSmt.CreateField("Channel", dbText, 4)
    fldSmt.AllowZeroLength = True
    tdSmt.Fields.Append fldSmt
    Set fldSmt = tdSmt.CreateField("Gain", dbText, 20)
    fldSmt.AllowZeroLength = True
    tdSmt.Fields.Append fldSmt
    Set fldSmt = tdSmt.CreateField("Offset", dbText, 20)
    fldSmt.AllowZeroLength = True
    tdSmt.Fields.Append fldSmt
    Set fldSmt = tdSmt.CreateField("Filter", dbText, 20)
    fldSmt.AllowZeroLength = True
    tdSmt.Fields.Append fldSmt
    dbSmt.TableDefs.Append tdSmt
    Resume 0

errFileSaveCreate:
    MsgBox "Error creating file:" & _
        Chr(13) & err.Description, vbOKOnly + vbExclamation, "Error"
    On Error GoTo 0
    Exit Sub

errFileSaveFatal:
    dbSmt.Rollback
    MsgBox "Error saving file:" & Chr(13) & _
        err.Description & Chr(13) & _
        "table=" & rsSmt.Name, vbOKOnly + vbExclamation, "Error"
    On Error GoTo 0
End Sub

Private Sub mnuFileSaveAs_Click()
    With commondialogFile
        .Flags = cdlOFNOverwritePrompt Or cdlOFNHideReadOnly _
            Or cdlOFNNoReadOnlyReturn Or cdlOFNPathMustExist
        .DialogTitle = "Save As"
        On Error GoTo ErrFileSaveAsCancel
        .ShowSave
        On Error GoTo 0
        If .filename = "" Then Exit Sub
    End With
    mnuFileSave_Click
    updateFormCaption commondialogFile.filename
ErrFileSaveAsCancel:
End Sub

Private Sub mnuFileTestsignalgenerator_Click()
Dim i As Integer

    If checkAnalog Then
        Beep
        Exit Sub
    End If

    frmTests.pport = checkHardwarePresence
    If frmTests.pport = 0 Then
        MsgBox "No Test Signal Generator found on any parallel port, " & _
            "or SMT Device is switched off", vbOKOnly + vbExclamation, frmMain.Caption
        Exit Sub
```

```
    End If
    generateTestSignals
    frmTests.Show
End Sub

Private Sub mnuFileViewErrors_Click()
    frmErrors.Show
    viewAll
    If frmErrors.txtViewErrors <> "" Then
        frmErrors.WindowState = 0
    Else
        Unload frmErrors
        MsgBox "No errors in SMT configuration", vbOKOnly, frmMain.Caption
    End If
End Sub

Private Sub mnuGenerateReport_Click()
Dim cr, tb As String
Dim ch As Variant
Const normal As String = "Normal"
Const heading1 As String = "Heading 1"
Const heading2 As String = "Heading 2"
Const tableformat As Integer = 20
Dim filename, shortfilename As String

Dim i, j, freq, cols, maxcols As Integer
Dim gain, offset As Double
Dim fontsize As Integer, colw As String

    cr = Chr(13)
    tb = Chr(9)
    If checkAll Then
        Beep
        Exit Sub
    End If
    createAsciiFile
    With commondialogReportFile
        If commondialogFile.filename <> "" Then
            filename = commondialogFile.filename
            For i = Len(filename) To 1 Step -1
                If Mid(filename, i, 1) = "."
                    Or Mid(filename, i, 1) = "\" Then Exit For
            Next
            If Mid(filename, i, 1) = "." Then Mid(filename, i) = ".DOC"
            .filename = filename
        End If
        .Flags = cdlOFNOverwritePrompt Or cdlOFNHideReadOnly _
            Or cdlOFNNoReadOnlyReturn Or cdlOFNPathMustExist
        On Error GoTo errSaveReportCancel
        .ShowSave
        On Error GoTo 0
        If .filename = "" Then Exit Sub
        filename = .filename
    End With
    For i = Len(filename) To 1 Step -1
        If Mid(filename, i, 1) = "\" Then Exit For
    Next
    If Mid(filename, i, 1) = "\" Then
        shortfilename = Mid(filename, i + 1)
    End If

    On Error GoTo errOLE
    Set oleReport = CreateObject("Word.basic")

    ' close existing Word documents with the same name (without saving)
    For i = 1 To oleReport.CountWindows
        If oleReport.windowname(i) = shortfilename Then
            oleReport.WindowList i
            oleReport.fileclose 2
            Exit For
        End If
    Next

    oleReport.AppMinimize 1
```

```
    oleReport.FileNew App.Path + "\REPORT.DOC"
    oleReport.Style heading1
    oleReport.Insert "SMT Configuration Report" + cr
    oleReport.Style normal
    oleReport.Insert "Project: " + txtProject + cr
    oleReport.Insert "Description: " + txtDescription + cr
    oleReport.Insert "Author: " + txtAuthor + cr
    oleReport.Insert "Date of creation: " + Format(Now, "dddd, mmm d yyyy hh:nn") + cr
    oleReport.Insert "Number of pages: ##pages##" + cr

    oleReport.Style heading2
    oleReport.Insert "Transmission parameters" + cr
    oleReport.Style normal
    oleReport.TableInsertTable , 2, 2, "4 cm", , tableformat, 129
    oleReport.Insert "Bitrate"
    oleReport.nextcell
    oleReport.TableColumnWidth "7 cm"
    oleReport.Insert CStr(cmbBitrate)
    oleReport.nextcell
    oleReport.Insert "PN Seed"
    oleReport.nextcell
    oleReport.Insert CStr(txtPNSeed)
    oleReport.nextcell
    oleReport.Insert "Transmit frequency"
    oleReport.nextcell
    oleReport.Insert CStr(cmbTransmitFreq)
    oleReport.LineDown

    DoEvents
    computeSampleFreqs
    oleReport.Style heading2
    oleReport.Insert "Analog channels" + cr
    oleReport.Style normal
    oleReport.TableInsertTable , 5, 2, "2 cm", , tableformat, 161
    oleReport.Tableheadings
    oleReport.TableColumnWidth "1.25 cm"
    oleReport.Insert "Name"
    oleReport.nextcell
    oleReport.Insert "Gain"
    oleReport.nextcell
    oleReport.Insert "Offset"
    oleReport.nextcell
    oleReport.Insert "Filter"
    oleReport.nextcell
    oleReport.TableColumnWidth "4 cm"
    oleReport.Insert "Sampling frequency"
    For i = 1 To grdAnalog.rows - 1
        grdAnalog.row = i
        grdAnalog.col = 0
        For j = LBound(channels) To UBound(channels)
            If channels(j) = grdAnalog Then Exit For
        Next
        oleReport.nextcell
        oleReport.Insert CStr(grdAnalog)
        grdAnalog.col = 1
        oleReport.nextcell
        oleReport.Insert CStr(grdAnalog)
        grdAnalog.col = 2
        oleReport.nextcell
        oleReport.Insert CStr(grdAnalog)
        grdAnalog.col = 3
        oleReport.nextcell
        oleReport.Insert CStr(grdAnalog)
        oleReport.nextcell
        oleReport.Insert Format(Abs(samplefreqs(j)), "0") + " Hz"
        If samplefreqs(j) < 0 Then
            oleReport.Bold 1
            oleReport.Insert " (non-uniform)"
            oleReport.Bold 0
        End If
    Next
    oleReport.charright 2

    DoEvents
```

```
    oleReport.Style heading2
    oleReport.Insert "Digital channels" + cr
    oleReport.Style normal
    oleReport.TableInsertTable , 2, 2, "2 cm", , tableformat, 161
    oleReport.Tableheadings
    oleReport.TableColumnWidth "1.25 cm"
    oleReport.Insert "Name"
    oleReport.nextcell
    oleReport.TableColumnWidth "4 cm"
    oleReport.Insert "Sampling frequency"
    For Each ch In digitals
        oleReport.nextcell
        oleReport.Insert ch
        oleReport.nextcell
        For j = LBound(channels) To UBound(channels)
            If channels(j) = ch Then Exit For
        Next
        oleReport.Insert Format(Abs(samplefreqs(j)), "0") + " Hz"
        If samplefreqs(j) < 0 Then
            oleReport.Insert " (non-uniform)"
        End If
    Next
    oleReport.charright 2

    DoEvents
    oleReport.Style heading2
    oleReport.Insert "Frame Structure" + cr
    oleReport.Style normal
    oleReport.Insert "Minor frame length: " & minorframelength & cr
    oleReport.Insert "Minor frames per major frame: " & minorframes & cr
    oleReport.Insert "Sync words: "
    For i = 4 - syncwords To 3
        oleReport.Insert Hex(syncwordbytes(i)) + " "
    Next
    oleReport.Insert "(hex)" + cr
    oleReport.Insert "SFID Start value: " & txtSfidStart & cr
    oleReport.Insert "SFID Direction: " & cmbSfidDir & cr
    oleReport.Insert "Frame Structure: " & cr

    If minorframelength > 14 Then
        maxcols = 18
        fontsize = 8
        colw = "0.8 cm"
    Else
        maxcols = 14
        fontsize = 10
        colw = "1 cm"
    End If
    'rem by hkt 3 lines
    'disable the emphasis facility to give neater presentation
    'If minorframes = 1 Then
    '    txtCols = maxcols
    '    txtCols_Click
    'End If
    With grdScantable
        .FixedRows = 0
        .SelStartRow = 0
        .SelEndRow = .rows - 1
        For i = 0 To .cols - 2 Step maxcols
            DoEvents
            If i + maxcols < .cols - 1 Then
                cols = maxcols
            Else
                cols = .cols - 1 - i
            End If
            .SelStartCol = i + 1
            .SelEndCol = i + cols
            oleReport.Insert .Clip + Chr(13)
            oleReport.LineUp
            .col = 0
            For j = .rows - 2 To 0 Step -1
                .row = j + 1
                If minorframes > 1 Then oleReport.Insert "Minor frame "
                oleReport.Insert .text + tb
```

```
                oleReport.LineUp
                oleReport.StartOfLine
            Next
            oleReport.Insert tb
            oleReport.StartOfLine
            oleReport.ExtendSelection
            oleReport.EndOfDocument
            oleReport.TextToTable , , , colw, , tableformat, 161
            oleReport.fontsize fontsize
            oleReport.TableColumnWidth , 0
            oleReport.CenterPara
            oleReport.Cancel
            If minorframes > 1 Then
                oleReport.StartOfLine
                oleReport.TableColumnWidth "2.5 cm"
            End If
            oleReport.EndOfDocument
            oleReport.Insert cr
        Next
        .FixedRows = 1
        .SelEndCol = 1
        .SelEndRow = 1
    End With
    'reduce the redundant information generated in the report
    'hkt remmed generate Test Signal
    'generateTestSignals
    'oleReport.Style heading2
    'oleReport.Insert "Test Signals" + cr
    'oleReport.Style normal
    'oleReport.TableInsertTable , 5, 2, "4 cm", , tableformat, 161
    'oleReport.Tableheadings
    'oleReport.TableColumnWidth "1.25 cm"
    'oleReport.Insert "Name"
    'oleReport.nextcell
    'oleReport.TableColumnWidth "2.5 cm"
    'oleReport.Insert "Type"
    'oleReport.nextcell
    'oleReport.Insert "Frequency [Hz]"
    'oleReport.nextcell
    'oleReport.Insert "Amplitude [V]"
    'oleReport.nextcell
    'oleReport.Insert "Offset [V]"
    'With grdAnalog
    '    For i = 0 To UBound(testSignalSetup)
    '        .row = i + 1
    '        .col = 0
    '        oleReport.nextcell
    '        oleReport.Insert CStr(grdAnalog)
    '        oleReport.nextcell
    '        oleReport.Insert "square wave"
    '        oleReport.nextcell
    '        oleReport.Insert Str(2600 / 2 ^ testSignalSetup(3, i))
    '        oleReport.nextcell
    '        If testSignalSetup(1, i) > 0 Then
    '            gain = 1 / (2 ^ (testSignalSetup(1, i) - 1))
    '        Else
    '            gain = 0
    '        End If
    '        oleReport.Insert Format(2.5 * 0.6 * gain, "Scientific")
    '        oleReport.nextcell
    '        If testSignalSetup(2, i) >= 8 Then
    '            offset = (testSignalSetup(2, i) - 7) * 0.3125
    '        Else
    '            offset = testSignalSetup(2, i) * 0.3125 - 2.5
    '        End If
    '        oleReport.Insert CStr(offset)
    '    Next
    'End With
    'DoEvents
    'freq = 2600
    'For i = 0 To 7
    '    oleReport.nextcell
    '    oleReport.Insert "D(" & i & ")"
    '    oleReport.nextcell
```

```
    '    oleReport.Insert "digital"
    '    oleReport.nextcell
    '    oleReport.Insert CStr(freq)
    '    oleReport.nextcell
    '    oleReport.nextcell
    '    freq = freq \ 2
    'Next
    'DoEvents
    'oleReport.charright 2

    oleReport.insertPageBreak
    oleReport.Style heading2
    oleReport.Insert "SMT EEPROM Description file (ASCII file)" + cr
    oleReport.Style "SMT_EEPROM"
    oleReport.Insert cr + asciiFile

    oleReport.EndOfDocument
    i = oleReport.SelInfo(1)
    oleReport.StartOfDocument
    oleReport.EditReplace "##pages##", CStr(i), , , , , , , 1
    oleReport.StartOfDocument

    oleReport.FileSaveAs filename
    oleReport.AppMinimize 0
'    oleReport.fileclose
    Exit Sub

errOLE:
    MsgBox "Error using OLE to create report:" & Chr(13) & _
        err.Description, vbOKOnly, frmMain.Caption
errSaveReportCancel:
End Sub



Private Sub mnuScanTableClear_Click()
    smtChanged = True
    fillgrdScantable ("")
    update_form
    checkScantable
End Sub

Private Sub mnuScanTableCopy_Click()
    Clipboard.SetText grdScantable.Clip + Chr(13)
End Sub
Private Sub mnuScanTableCut_Click()
    smtChanged = True
    mnuScanTableCopy_Click
    mnuScanTableClear_Click
    checkScantable
End Sub

Private Sub mnuScantableEntry_Click(Index As Integer)
    smtChanged = True
    fillgrdScantable (mnuScantableEntry(Index).Caption)
    checkScantable
End Sub

Private Sub mnuFileEditSmtDeviceSpecification_Click()
    frmFileEditSmtDeviceSpecification.Show
End Sub
Private Sub mnuFileExit_Click()
    End
End Sub
Private Sub mnuHelpAbout_Click()
    frmAbout.Show
End Sub
Private Sub mnuScanTablePaste_Click()
Dim clipdata As String
Dim i, rows, cols, curcols As Integer
Dim content As Boolean

    smtChanged = True
    clipdata = Clipboard.GetText
```

```
    ' clipboard analysis; cols are separated by tab
    ' (tab=chr(9)), rows by cr/lf (cr=chr(13))
    rows = 0
    cols = 0
    curcols = 1
    For i = 1 To Len(clipdata)
        If Asc(Mid(clipdata, i, 1)) >= 32 Then
            content = True
        End If
        If Mid(clipdata, i, 1) = Chr(9) Then
            curcols = curcols + 1
        End If
        If Mid(clipdata, i, 1) = Chr(13) Then
            rows = rows + 1
            ' take largest value of all rows:
            If curcols > cols Then cols = curcols
            curcols = 1
            content = False
        End If
    Next
    ' if the last line doesn't end with a cr:
    If content Then rows = rows + 1
    If rows = 0 Or cols = 0 Then Exit Sub

    With grdScantable
        ' adjust selection to clipboard content,
        ' taking care of grid boundaries:
        .SelStartCol = .col
        If .col + cols > .cols Then
            .SelEndCol = .cols - 1
        Else
            .SelEndCol = .col + cols - 1
        End If
        .SelStartRow = .row
        If .row + rows > .rows Then
            .SelEndRow = .rows - 1
        Else
            .SelEndRow = .row + rows - 1
        End If
        ' copy data:
        .Clip = clipdata
        .SelEndCol = .col
        .SelEndRow = .row
    End With
    update_form
    checkScantable
End Sub


Private Sub repeat_Click()
Dim clipdata As String                    'clipboard data
Dim i, rows, cols, curcols As Integer     'col and row var
Dim content As Boolean                    'correct content status
Dim s_col, s_row, rpt_times As Integer    'start row and start col, and number of repeats to do
Dim hi_start, hi_end, no_col As Integer   'highlighted cells and cols
Dim msg_string, Response As String        'msg at end
Dim rpt_left_over As Boolean              'incomplete copy status
Dim string_Sync As Integer      'sync cells string check
Dim col_Sync As Integer         'no of sync cols

    On Error GoTo End_Function
    smtChanged = True
    msg_string = "incomplete pattern copied at tail of minor frame" 'set message

    rpt_left_over = False   'set var

    s_row = grdScantable.SelStartRow    'set var
    hi_start = grdScantable.SelStartCol
    hi_end = grdScantable.SelEndCol
    no_col = (grdScantable.SelEndCol - grdScantable.SelStartCol) + 1
    s_col = hi_start

    'Clipboard.clear
```

```
    'clipdata = Clipboard.GetText


    Clipboard.SetText grdScantable.Clip + Chr(13)
    ' clipboard analysis; cols are separated by tab
    ' (tab=chr(9)), rows by cr/lf (cr=chr(13))
    clipdata = Clipboard.GetText
    'initialse variable
    rows = 0
    cols = 0
    curcols = 1
    'check if highlighted area has correct data
    For i = 1 To Len(clipdata)
        If Asc(Mid(clipdata, i, 1)) >= 32 Then
            content = True
        End If
        If Mid(clipdata, i, 1) = Chr(9) Then
            curcols = curcols + 1
        End If
        If Mid(clipdata, i, 1) = Chr(13) Then
            rows = rows + 1
            ' take largest value of all rows:
            If curcols > cols Then cols = curcols
            curcols = 1
            content = False
        End If
    Next
    ' if the last line doesn't end with a cr:
    If content Then rows = rows + 1
    If rows = 0 Or cols = 0 Then Exit Sub
    'calc the number of times we nêed to repeat the copy command
    rpt_times = ((grdScantable.cols - grdScantable.col) + 1) \ no_col
    'check for sync cells
    col_Sync = 0
    For i = 1 To 5
        string_Sync = 0
        grdScantable.col = 1
        string_Sync = InStr(grdScantable.text, "Sync")
        If string_Sync > 0 Then
            col_Sync = col_Sync + 1
        End If
    Next i

    With grdScantable
        ' adjust selection to clipboard content,
        ' taking care of grid boundaries:
        .SelStartRow = s_row
        .SelEndRow = s_row
    'copy highlighted cells til end of minor frame
    For i = 1 To rpt_times
        s_col = s_col + cols 'set s_col for loop
        If Val(s_col) >= Val(.cols) Then Exit For 'if greater than minor frame
        .SelStartCol = s_col
        If Val(.SelStartCol + cols - 1) >= Val(.cols) Then
            grdScantable.SelEndCol = grdScantable.cols - 1
            rpt_left_over = True
        Else
            .SelEndCol = Val(s_col + cols - 1)
        End If
        ' copy data:
        .Clip = clipdata
    Next i
    End With
    'if an incomplete copy at tail of minor frame then give message
    If rpt_left_over = True Then
        Response = MsgBox(msg_string, vbOKOnly, "Message")
    End If
    update_form
    checkScantable
End_Function:

End Sub

Private Sub txtCols_Click()
```

```
    If Val(txtCols) = columns Then Exit Sub
    columns = Val(txtCols)
    Do While minorframelength Mod columns <> 0
        columns = columns - 1
    Loop
    If columns <= syncwords Then columns = syncwords
    saveScantable
    update_form
    restoreScantable
    smtChanged = True

End Sub

Private Sub txtCols_KeyDown(KeyCode As Integer, Shift As Integer)
'by hkt
'txtcol responds to keyboard events
    Select Case KeyCode
        Case vbKeyReturn
            If Val(txtCols) = columns Then Exit Sub
            columns = Val(txtCols)
            Do While minorframelength Mod columns <> 0
                columns = columns - 1
            Loop
            If columns <= syncwords Then columns = syncwords
            saveScantable
            update_form
            restoreScantable
            smtChanged = True
            Exit Sub
        Case vbKeyTab
            If Val(txtCols) = columns Then Exit Sub
            columns = Val(txtCols)
            Do While minorframelength Mod columns <> 0
                columns = columns - 1
            Loop
            If columns <= syncwords Then columns = syncwords
            saveScantable
            update_form
            restoreScantable
            smtChanged = True
            Exit Sub
    End Select
End Sub

Private Sub txtCols_LostFocus()
    txtCols_Click
End Sub

Private Sub txtDescription_Change()
    smtChanged = True
End Sub

Private Sub txtError_Change()
    If txtError <> "" Then
        txtError.BackColor = &H80FFFF
        If chkErrorSound Then Beep
    Else
        txtError.BackColor = &HFFFFFF
    End If
End Sub

Private Sub txtMinorframelength_Click()
'hkt var
Dim changelength As Integer 'value of new length

    If Val(txtMinorframelength) = minorframelength Then Exit Sub
    'hkt next 4
    'check if in emphasis state, and if so normalise before changing
    If Val(txtCols) <> Val(txtMinorframelength) Then
        changelength = txtMinorframelength
        txtCols = minorframelength
        txtCols_Click
    End If
```

```
    minorframelength = Val(txtMinorframelength)
    If minorframelength > maxMinorframelength Then minorframelength = maxMinorframelength
    If minorframelength < syncwords Then minorframelength = syncwords
    If minorframes = 1 Then columns = minorframelength
    smtChanged = True
    update_form
End Sub

Private Sub txtMinorframelength_KeyDown(KeyCode As Integer, Shift As Integer)
'by hkt
Dim changelength As Integer 'value of new length

    Select Case KeyCode
        Case vbKeyReturn
            If Val(txtMinorframelength) = minorframelength Then Exit Sub
            'hkt next 4
            'check if in emphasis state, and if so normalise before changing
            If Val(txtCols) <> Val(txtMinorframelength) Then

                changelength = txtMinorframelength
                txtCols = minorframelength
                txtCols_Click
            End If

            minorframelength = Val(txtMinorframelength)
            If minorframelength > maxMinorframelength Then minorframelength = maxMinorframele
ngth
            If minorframelength < syncwords Then minorframelength = syncwords
            If minorframes = 1 Then columns = minorframelength
            smtChanged = True
            update_form
            Exit Sub
        Case vbKeyTab
            If Val(txtMinorframelength) = minorframelength Then Exit Sub
            'hkt next 4
            'check if in emphasis state, and if so normalise before changing
            If Val(txtCols) <> Val(txtMinorframelength) Then
                changelength = txtMinorframelength
                txtCols = minorframelength
                txtCols_Click
            End If

            minorframelength = Val(txtMinorframelength)
            If minorframelength > maxMinorframelength Then minorframelength = maxMinorframele
ngth
            If minorframelength < syncwords Then minorframelength = syncwords
            If minorframes = 1 Then columns = minorframelength
            smtChanged = True
            update_form
            Exit Sub
    End Select
End Sub

Private Sub txtMinorframelength_LostFocus()
    txtMinorframelength_Click
End Sub

Private Sub txtMinorframes_Click()

    If Val(txtMinorFrames) = minorframes Then Exit Sub
    minorframes = Val(txtMinorFrames)
    If minorframes > maxMinorFrames Then minorframes = maxMinorFrames
    If minorframes < 1 Then minorframes = 1
    If minorframes = 1 Then columns = minorframelength
    smtChanged = True
    update_form
End Sub

Private Sub txtMinorframes_KeyDown(KeyCode As Integer, Shift As Integer)
'by hkt
'responds to keyboard events
    Select Case KeyCode
        Case vbKeyReturn
            If Val(txtMinorFrames) = minorframes Then Exit Sub
```

```
            minorframes = Val(txtMinorFrames)
            If minorframes > maxMinorFrames Then minorframes = maxMinorFrames
            If minorframes < 1 Then minorframes = 1
            If minorframes = 1 Then columns = minorframelength
            smtChanged = True
            update_form
            Exit Sub
        Case vbKeyTab
            If Val(txtMinorFrames) = minorframes Then Exit Sub
            minorframes = Val(txtMinorFrames)
            If minorframes > maxMinorFrames Then minorframes = maxMinorFrames
            If minorframes < 1 Then minorframes = 1
            If minorframes = 1 Then columns = minorframelength
            smtChanged = True
            update_form
            Exit Sub
    End Select

End Sub

Private Sub txtMinorframes_LostFocus()
    txtMinorframes_Click
End Sub

Private Sub txtPNSeed_Change()
    checkPNSeed
    smtChanged = True
End Sub

Private Sub txtPNSeed_Click()
    txtPNSeed_LostFocus
End Sub

Private Sub txtPNSeed_LostFocus()
    txtPNSeed = Hex(Val("&H" & txtPNSeed))
End Sub

Private Sub txtSfidStart_Change()
    checkSfid
    smtChanged = True
End Sub

Private Sub txtSyncword_Change()
    smtChanged = True
End Sub

Private Sub txtSyncword_Click()
Dim b, sw As Integer
Dim i As Long
    i = Val("&H" & txtSyncword & "&")
    txtSyncword = Hex(Val(i & "&"))
    For sw = 3 To 0 Step -1
        b = i Mod &H100
        If b < 0 Then
            b = b + &H100
            i = i - &H80000000  ' work around Bill Gates' brain damage
            i = i \ &H100
            i = i + &H800000
        Else
            i = i \ &H100
        End If
        syncwordbytes(sw) = b
    Next
    checkSyncword
End Sub

Private Sub txtSyncword_LostFocus()
    txtSyncword_Click
End Sub
```

```
VERSION 5.00
Object = "{F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.1#0"; "COMDLG32.OCX"
Object = "{A8B3B723-0B5A-101B-B22E-00AA0037B2FC}#1.0#0"; "GRID32.OCX"
Begin VB.Form frmMain
   BorderStyle     =   1  'Fixed Single
   Caption         =   "SMT Configuration Tool"
   ClientHeight    =   8208
   ClientLeft      =   72
   ClientTop       =   1740
   ClientWidth     =   11880
   Icon            =   (Icon)
   MaxButton       =   0   'False
   MinButton       =   0   'False
   PaletteMode     =   1  'UseZOrder
   ScaleHeight     =   8208
   ScaleWidth      =   11880
   Begin VB.PictureBox vsViewPort1
      Height          =   7035
      Left            =   720
      ScaleHeight     =   6984
      ScaleWidth      =   9924
      TabIndex        =   50
      Top             =   480
      Width           =   9975
      Begin VB.Frame fraAnalog
         Caption         =   "Analog channels setup"
         Height          =   2355
         Left            =   8160
         TabIndex        =   47
         Top             =   1440
         Width           =   3615
         Begin MSGrid.Grid grdAnalog
            Height          =   1935
            Left            =   120
            TabIndex        =   48
            Tag             =   "1"
            Top             =   300
            Width           =   3375
            _Version        =   65536
            _ExtentX        =   5953
            _ExtentY        =   3413
            _StockProps     =   77
            BackColor       =   16777215
            BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
               Name            =   "MS Sans Serif"
               Size            =   7.8
               Charset         =   0
               Weight          =   400
               Underline       =   0   'False
               Italic          =   0   'False
               Strikethrough   =   0   'False
            EndProperty
            Rows            =   1
            Cols            =   4
            FixedRows       =   0
            ScrollBars      =   2
            MouseIcon       =   {Binary}
         End
      End
      Begin VB.Frame fraScantable
         Caption         =   "Scantable setup"
         Height          =   2355
         Left            =   5100
         TabIndex        =   31
         Top             =   1440
         Width           =   3075
         Begin VB.CommandButton cmdPasteEntire
            Caption         =   "paste table"
            Height          =   315
            Left            =   1080
            TabIndex        =   40
            Tag             =   "1"
            Top             =   1920
            Width           =   915
```

```
End
Begin VB.TextBox txtSfidStart
   Alignment       =   1  'Right Justify
   Height          =   285
   Left            =   1680
   TabIndex        =   39
   Tag             =   "1"
   Text            =   "255"
   Top             =   1140
   Width           =   555
End
Begin VB.ComboBox cmbSfidDir
   Height          =   300
   Left            =   1680
   Style           =   2  'Dropdown List
   TabIndex        =   38
   Tag             =   "1"
   Top             =   1500
   Width           =   855
End
Begin VB.CommandButton cmdCopyEntire
   Caption         =   "copy table"
   Height          =   315
   Left            =   120
   TabIndex        =   37
   Tag             =   "1"
   Top             =   1920
   Width           =   975
End
Begin VB.TextBox txtCols
   Height          =   300
   Left            =   2010
   TabIndex        =   36
   Tag             =   "1"
   Text            =   "-1"
   Top             =   1125
   Width           =   555
End
Begin VB.ComboBox cmbEmphasize
   Height          =   300
   Left            =   1980
   Style           =   2  'Dropdown List
   TabIndex        =   35
   Tag             =   "1"
   Top             =   1500
   Width           =   855
End
Begin VB.TextBox txtMinorframelength
   Height          =   315
   Left            =   1680
   TabIndex        =   34
   Tag             =   "1"
   Text            =   "-1"
   Top             =   300
   Width           =   555
End
Begin VB.TextBox txtMinorFrames
   Height          =   315
   Left            =   1680
   TabIndex        =   33
   Tag             =   "1"
   Text            =   "-1"
   Top             =   720
   Width           =   555
End
Begin VB.CommandButton repeat
   Caption         =   "repeat"
   Height          =   315
   Left            =   1980
   TabIndex        =   32
   Top             =   1920
   Width           =   915
End
Begin VB.Label Label1
```

```
      Caption         =   "Minor frames per major frame:"
      Height          =   495
      Index           =   3
      Left            =   120
      TabIndex        =   46
      Top             =   660
      Width           =   1335
      WordWrap        =   -1  'True
   End
   Begin VB.Label Label1
      Caption         =   "Minor frame length:"
      Height          =   195
      Index           =   2
      Left            =   120
      TabIndex        =   45
      Top             =   360
      Width           =   1395
   End
   Begin VB.Label lblSfid
      Caption         =   "SFID start value:"
      Height          =   195
      Index           =   0
      Left            =   120
      TabIndex        =   44
      Top             =   1200
      Width           =   1275
   End
   Begin VB.Label lblSfid
      Caption         =   "SFID direction:"
      Height          =   195
      Index           =   1
      Left            =   120
      TabIndex        =   43
      Top             =   1560
      Width           =   1095
   End
   Begin VB.Label lblDisplay
      Caption         =   "Table width (on screen):"
      Height          =   195
      Index           =   0
      Left            =   120
      TabIndex        =   42
      Top             =   1200
      Width           =   1875
   End
   Begin VB.Label lblDisplay
      Caption         =   "Emphasize channel:"
      Height          =   195
      Index           =   1
      Left            =   120
      TabIndex        =   41
      Top             =   1560
      Width           =   1515
   End
End
Begin VB.Frame fraTransmission
   Caption         =   "Transmission parameters"
   Height          =   2355
   Left            =   0
   TabIndex        =   20
   Top             =   1440
   Width           =   5115
   Begin VB.TextBox txtSyncword
      Alignment       =   1  'Right Justify
      DataField       =   "sync1"
      DataSource      =   "datSmtSystemSetup"
      Height          =   285
      Left            =   1800
      TabIndex        =   25
      Tag             =   "1"
      Text            =   "EB90"
      Top             =   1920
      Width           =   1155
   End
```

```
Begin VB.ComboBox cmbBitrate
   Height          =   288
   Left            =   1800
   Style           =   2  'Dropdown List
   TabIndex        =   24
   Tag             =   "1"
   Top             =   360
   Width           =   2832
End
Begin VB.ComboBox cmbTransmitFreq
   Height          =   288
   Left            =   1800
   Sorted          =   -1  'True
   Style           =   2  'Dropdown List
   TabIndex        =   23
   Tag             =   "1"
   Top             =   1080
   Width           =   1308
End
Begin VB.ComboBox cmbSyncwordLength
   Height          =   300
   Left            =   1800
   Style           =   2  'Dropdown List
   TabIndex        =   22
   Tag             =   "1"
   Top             =   1500
   Width           =   855
End
Begin VB.TextBox txtPNSeed
   Alignment       =   1  'Right Justify
   Height          =   285
   Left            =   1800
   TabIndex        =   21
   Tag             =   "1"
   Top             =   720
   Width           =   555
End
Begin VB.Label Label3
   Caption         =   "PN Seed (hex):"
   Height          =   255
   Left            =   240
   TabIndex        =   30
   Top             =   780
   Width           =   1395
End
Begin VB.Label label4
   Caption         =   "Transmit frequency:"
   Height          =   255
   Left            =   240
   TabIndex        =   29
   Top             =   1140
   Width           =   1575
End
Begin VB.Label Label2
   Caption         =   "Bitrate:"
   Height          =   255
   Index           =   0
   Left            =   240
   TabIndex        =   28
   Top             =   420
   Width           =   855
End
Begin VB.Label Label5
   Caption         =   "Sync word (hex):"
   Height          =   255
   Index           =   0
   Left            =   240
   TabIndex        =   27
   Top             =   1980
   Width           =   1575
End
Begin VB.Label Label5
   Caption         =   "Sync word length:"
   Height          =   255
```

```
      Index           =   1
      Left            =   240
      TabIndex        =   26
      Top             =   1560
      Width           =   1575
   End
End
Begin VB.Frame Frame1
   Caption         =   "Identification"
   Height          =   1455
   Left            =   0
   TabIndex        =   13
   Top             =   0
   Width           =   11775
   Begin VB.TextBox txtDescription
      Alignment       =   1  'Right Justify
      Height          =   285
      Left            =   1080
      TabIndex        =   16
      Tag             =   "1"
      Top             =   600
      Width           =   10395
   End
   Begin VB.TextBox txtProject
      Alignment       =   1  'Right Justify
      Height          =   285
      Left            =   1080
      TabIndex        =   15
      Tag             =   "1"
      Top             =   240
      Width           =   10395
   End
   Begin VB.TextBox txtAuthor
      Alignment       =   1  'Right Justify
      Height          =   285
      Left            =   1080
      TabIndex        =   14
      Tag             =   "1"
      Top             =   960
      Width           =   10395
   End
   Begin VB.Label Label2
      Caption         =   "Description:"
      Height          =   255
      Index           =   1
      Left            =   120
      TabIndex        =   19
      Top             =   660
      Width           =   855
   End
   Begin VB.Label Label2
      Caption         =   "Author:"
      Height          =   255
      Index           =   4
      Left            =   120
      TabIndex        =   18
      Top             =   1020
      Width           =   855
   End
   Begin VB.Label Label2
      Caption         =   "Project:"
      Height          =   255
      Index           =   5
      Left            =   120
      TabIndex        =   17
      Top             =   300
      Width           =   855
   End
End
Begin MSGrid.Grid grdScantable
   Height          =   3315
   Left            =   0
   TabIndex        =   49
   Tag             =   "1"
```

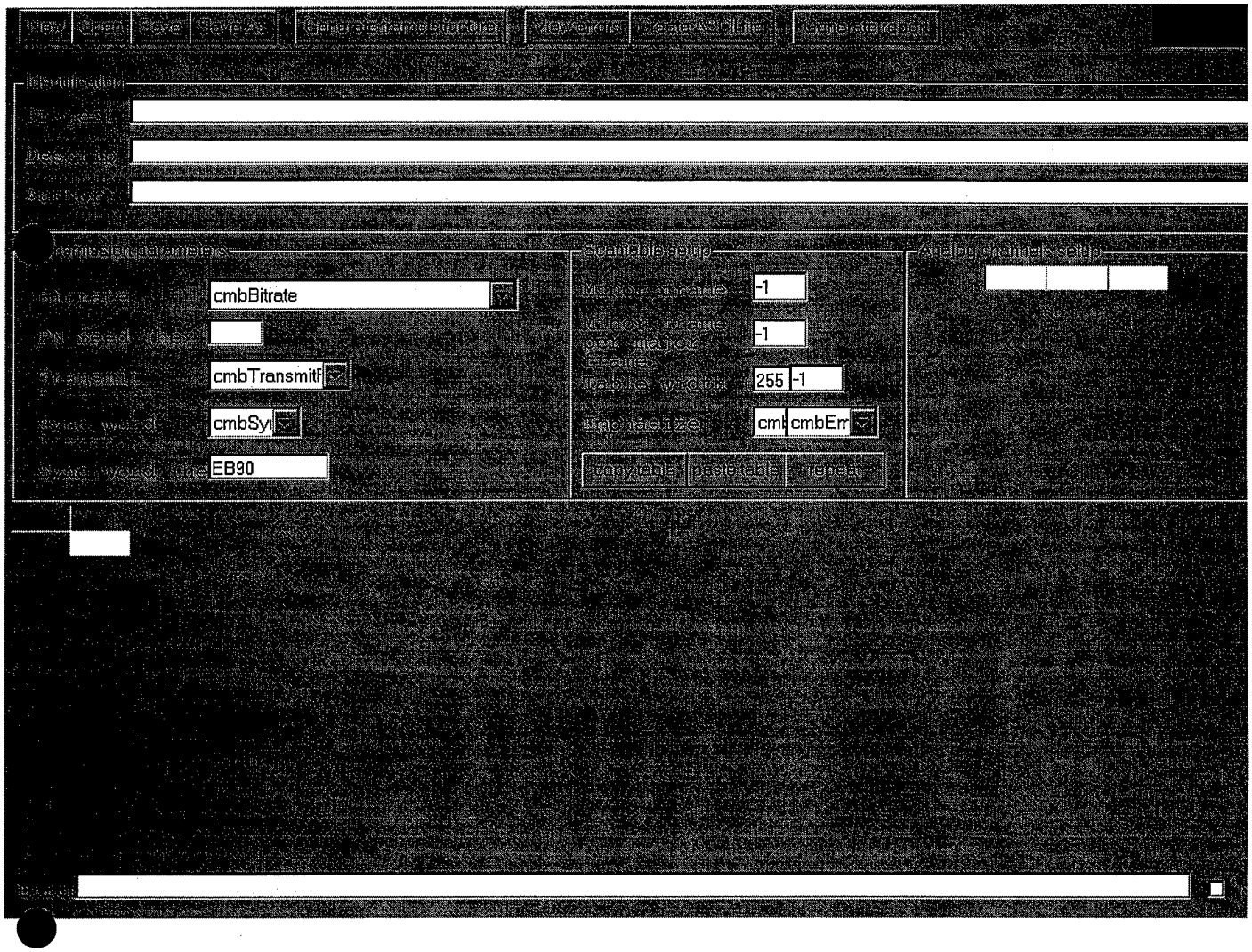
```
      Top             =   3780
      Width           =   11775
      _Version        =   65536
      _ExtentX        =   20770
      _ExtentY        =   5847
      _StockProps     =   77
      BackColor       =   16777215
      BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
         Name            =   "MS Sans Serif"
         Size            =   7.8
         Charset         =   0
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      MouseIcon       =   {Binary}
   End
End
Begin VB.CommandButton cmdButtonBar
   Caption         =   "Generate report"
   Height          =   315
   Index           =   8
   Left            =   7200
   TabIndex        =   12
   TabStop         =   0   'False
   Top             =   60
   Width           =   1335
End
Begin VB.CommandButton cmdButtonBar
   Caption         =   "Start test signal generator"
   Height          =   315
   Index           =   7
   Left            =   8616
   TabIndex        =   11
   TabStop         =   0   'False
   Tag             =   "1"
   Top             =   60
   Visible         =   0   'False
   Width           =   1995
End
Begin VB.CommandButton cmdButtonBar
   Caption         =   "Create ASCII file"
   Height          =   315
   Index           =   6
   Left            =   5700
   TabIndex        =   10
   TabStop         =   0   'False
   Tag             =   "1"
   Top             =   60
   Width           =   1335
End
Begin VB.CommandButton cmdButtonBar
   Caption         =   "View errors"
   Height          =   315
   Index           =   5
   Left            =   4740
   TabIndex        =   9
   TabStop         =   0   'False
   Tag             =   "1"
   Top             =   60
   Width           =   975
End
Begin VB.CommandButton cmdButtonBar
   Caption         =   "Generate frame structure"
   Height          =   315
   Index           =   4
   Left            =   2640
   TabIndex        =   8
   TabStop         =   0   'False
   Tag             =   "1"
   Top             =   60
   Width           =   1935
End
```

```
   Begin VB.CommandButton cmdButtonBar
      Caption         =   "Save As"
      Height          =   315
      Index           =   3
      Left            =   1680
      TabIndex        =   7
      TabStop         =   0   'False
      Tag             =   "1"
      Top             =   60
      Width           =   795
   End
   Begin VB.CommandButton cmdButtonBar
      Caption         =   "Save"
      Height          =   315
      Index           =   2
      Left            =   1140
      TabIndex        =   6
      TabStop         =   0   'False
      Tag             =   "1"
      Top             =   60
      Width           =   555
   End
   Begin VB.CommandButton cmdButtonBar
      Caption         =   "Open"
      Height          =   315
      Index           =   1
      Left            =   600
      TabIndex        =   5
      TabStop         =   0   'False
      Tag             =   "1"
      Top             =   60
      Width           =   555
   End
   Begin VB.CommandButton cmdButtonBar
      Caption         =   "New"
      Height          =   315
      Index           =   0
      Left            =   120
      TabIndex        =   4
      TabStop         =   0   'False
      Tag             =   "1"
      Top             =   60
      Width           =   495
   End
   Begin VB.CheckBox chkErrorSound
      Caption         =   "Check1"
      Height          =   195
      Left            =   10980
      TabIndex        =   0
      Top             =   7860
      Width           =   255
   End
   Begin VB.TextBox txtError
      Alignment       =   1  'Right Justify
      BackColor       =   &H00FFFFFF&
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   7.8
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   285
      Left            =   660
      Locked          =   -1  'True
      TabIndex        =   1
      TabStop         =   0   'False
      Tag             =   "1"
      Top             =   7800
      Width           =   10215
   End
   Begin MSComDlg.CommonDialog commondialogReportFile
```

```
      Left            =   11040
      Top             =   0
      _ExtentX        =   847
      _ExtentY        =   847
      _Version        =   327681
      CancelError     =   -1  'True
      DefaultExt      =   "mdb"
      DialogTitle     =   "Save Report As"
      Filter          =   "Document files (*.doc)|*.doc"
      MaxFileSize     =   64
   End
   Begin MSComDlg.CommonDialog commondialogFile
      Left            =   10440
      Top             =   0
      _ExtentX        =   847
      _ExtentY        =   847
      _Version        =   327681
      CancelError     =   -1  'True
      DefaultExt      =   "mdb"
      Filter          =   "SMT Configuration files (*.mdb)|*.mdb"
      MaxFileSize     =   64
   End
   Begin MSComDlg.CommonDialog commondialogSpecFile
      Left            =   10740
      Top             =   0
      _ExtentX        =   847
      _ExtentY        =   847
      _Version        =   327681
      CancelError     =   -1  'True
      DefaultExt      =   "mdb"
      DialogTitle     =   "Open SMT Configuration Specification file"
      Filter          =   "SMT Config Spec files (*.mdb)|*.mdb"
      MaxFileSize     =   64
   End
   Begin VB.Line Line1
      X1              =   0
      X2              =   11880
      Y1              =   420
      Y2              =   420
   End
   Begin VB.Label lblErrorSound
      Caption         =   "Sound"
      Height          =   195
      Left            =   11280
      TabIndex        =   3
      Top             =   7860
      Width           =   495
   End
   Begin VB.Label Label2
      Caption         =   "Errors:"
      Height          =   255
      Index           =   2
      Left            =   120
      TabIndex        =   2
      Top             =   7860
      Width           =   555
   End
   Begin VB.Menu mnuFile
      Caption         =   "&File"
      Begin VB.Menu mnuFileNew
         Caption      =   "&New..."
         Tag          =   "1"
      End
      Begin VB.Menu mnuFileOpen
         Caption      =   "&Open..."
         Tag          =   "1"
      End
      Begin VB.Menu mnuFileSave
         Caption      =   "&Save"
         Tag          =   "1"
      End
      Begin VB.Menu mnuFileSaveAs
         Caption      =   "Save &As..."
         Tag          =   "1"
```

```
      End
      Begin VB.Menu mnuSepBar1
         Caption         =   "-"
      End
      Begin VB.Menu mnuFileOpenSmtDeviceSpecification
         Caption         =   "Open SMT &device specification..."
      End
      Begin VB.Menu mnuFileEditSmtDeviceSpecification
         Caption         =   "&Edit SMT device specification..."
      End
      Begin VB.Menu mnuSepBar4
         Caption         =   "-"
      End
      Begin VB.Menu mnuFileExit
         Caption         =   "E&xit"
      End
   End
   Begin VB.Menu mnuInput
      Caption         =   "&Input"
      Begin VB.Menu mnuFileGenerateScantable
         Caption         =   "&Generate frame structure..."
         Tag             =   "1"
      End
   End
   Begin VB.Menu mnuReport
      Caption         =   "&Output"
      Begin VB.Menu mnuFileViewErrors
         Caption         =   "&View errors"
         Tag             =   "1"
      End
      Begin VB.Menu mnuSepBar10
         Caption         =   "-"
      End
      Begin VB.Menu mnuFileCreateAscii
         Caption         =   "&Create ASCII file"
         Tag             =   "1"
      End
      Begin VB.Menu mnuGenerateReport
         Caption         =   "&Generate Report..."
      End
      Begin VB.Menu mnuSepBar3
         Caption         =   "-"
      End
      Begin VB.Menu mnuFileTestsignalgenerator
         Caption         =   "S&tart test signal generator"
         Tag             =   "1"
      End
   End
   Begin VB.Menu mnuHelp
      Caption         =   "&Help"
      NegotiatePosition=   3  'Right
      Begin VB.Menu mnuHelpContents
         Caption         =   "&Contents"
         Shortcut        =   {F1}
         Visible         =   0   'False
      End
      Begin VB.Menu mnuSepBar2
         Caption         =   "-"
         Visible         =   0   'False
      End
      Begin VB.Menu mnuHelpAbout
         Caption         =   "&About..."
      End
   End
   Begin VB.Menu mnuScanTable
      Caption         =   "scantable_popup"
      Visible         =   0   'False
      Begin VB.Menu mnuScantableEntry
         Caption         =   "dummy"
         Index           =   1000
      End
      Begin VB.Menu mnuScanTable_
         Caption         =   "-"
      End
```

```
      Begin VB.Menu mnuScanTableCut
         Caption         =   "Cut"
      End
      Begin VB.Menu mnuScanTableCopy
         Caption         =   "Copy"
      End
      Begin VB.Menu mnuScanTablePaste
         Caption         =   "Paste"
      End
      Begin VB.Menu mnuScanTableClear
         Caption         =   "Clear"
      End
   End
   Begin VB.Menu mnuAnalogGain
      Caption         =   "analogGain_popup"
      Visible         =   0   'False
      Begin VB.Menu mnuAnalogGainEntry
         Caption         =   ""
         Index           =   1
      End
   End
   Begin VB.Menu mnuAnalogOffset
      Caption         =   "analogOffset_popup"
      Visible         =   0   'False
      Begin VB.Menu mnuAnalogOffsetEntry
         Caption         =   ""
         Index           =   1
      End
   End
   Begin VB.Menu mnuAnalogFilter
      Caption         =   "analogFilter_popup"
      Visible         =   0   'False
      Begin VB.Menu mnuAnalogFilterEntry
         Caption         =   ""
         Index           =   1
      End
   End
   Begin VB.Menu mnuAnalogFilterkbs
      Caption         =   "analogFilterkbs_popup"
      Visible         =   0   'False
      Begin VB.Menu mnuAnalogFilterkbsEntry
         Caption         =   ""
         Index           =   1
      End
   End
   Begin VB.Menu mnuAnalogFiltermbs
      Caption         =   "analogFiltermbs_popup"
      Visible         =   0   'False
      Begin VB.Menu mnuAnalogFiltermbsEntry
         Caption         =   ""
         Index           =   1
      End
   End
End
```

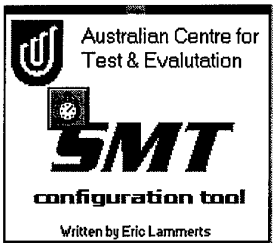

New
Open
Save
Save As
Generate frame structure
Viewers
Player Settings
Generate report
Identification
Description
Author
Transmission parameters
cmbBitrate
cmbTransmitf
cmbSy
EB90
Scrambler setup
Minimum frames
-1
-1
255
-1
cmb
cmbEn
copyable
pastable
Audio channels setup

frmStartup - 1

```
Option Explicit

Private Sub Form_Load()
    ' position window in the middle of the screen:
    frmStartup.Left = Screen.Width / 2 - frmStartup.Width / 2
    frmStartup.Top = Screen.Height / 2 - frmStartup.Height / 2
    frmMain.Hide
    Form_Paint
End Sub

Private Sub Form_Paint()
    ' start wait time
    tmrStartup.Interval = 3000
    tmrStartup.Enabled = False
    tmrStartup.Enabled = True
End Sub

Private Sub tmrStartup_Timer()
    frmMain.Show
    Unload Me
End Sub
```

frmStartup - 1

```
VERSION 5.00
Begin VB.Form frmStartup
   BorderStyle     =   0  'None
   Caption         =   "Form1"
   ClientHeight    =   2352
   ClientLeft      =   3588
   ClientTop       =   2916
   ClientWidth     =   2700
   ControlBox      =   0   'False
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   PaletteMode     =   1  'UseZOrder
   ScaleHeight     =   2352
   ScaleWidth      =   2700
   ShowInTaskbar   =   0   'False
   Begin VB.Timer tmrStartup
      Left            =   360
      Top             =   660
   End
   Begin VB.Image imgStartup
      Height          =   1884
      Left            =   0
      Picture         =   (Bitmap)
      Top             =   0
      Width           =   2160
   End
End
```


Australian Centre for
Test & Evalutation
SMT
configuration tool
Written by Eric Lammerts

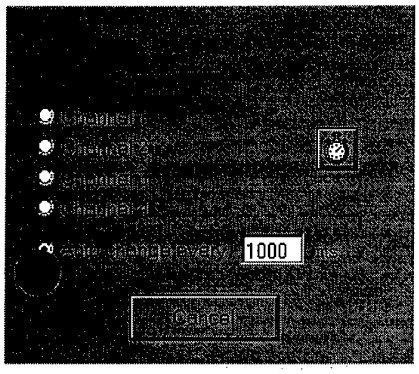
frmTests - 1

```
Option Explicit
Private Declare Sub setupTestSignal Lib "hwaccess" (ByVal port As Integer, ByVal gain As Inte
ger, ByVal off As Integer, ByVal freq As Integer, ByVal chan As Integer)
Dim channel As Integer
Public pport As Integer
Const channels = 4

Private Sub cmdCancel_Click()
    Unload Me
End Sub

Private Sub Form_Deactivate()
    Unload Me
End Sub

Private Sub Form_Load()
    channel = 3
    lblPortAddress = "Using parallel port at address 0x" & Hex(pport)
    tmrInterval_Timer
End Sub


Private Sub optChannel_Click(Index As Integer)
    If Index = 0 Then
        tmrInterval.Enabled = True
        tmrInterval.Interval = Val(txtInterval)
    Else
        tmrInterval.Enabled = False
        tmrInterval.Interval = 0
        channel = Index - 1
        setupTestSignal pport, _
            testSignalSetup(1, channel), _
            testSignalSetup(2, channel), _
            testSignalSetup(3, channel), channel
        lblTestChannel.Caption = "Testing analog channel " & channel + 1
    End If
End Sub

Private Sub tmrInterval_Timer()
    channel = ((channel + 1) Mod channels)
    lblTestChannel.Caption = "Testing analog channel " & channel + 1
    setupTestSignal pport, _
        testSignalSetup(1, channel), _
        testSignalSetup(2, channel), _
        testSignalSetup(3, channel), channel

End Sub

Private Sub txtInterval_Change()
    tmrInterval.Interval = Val(txtInterval)
    tmrInterval.Enabled = False
    tmrInterval.Enabled = True
End Sub
```

frmTests - 1

```
VERSION 5.00
Begin VB.Form frmTests
   BorderStyle     =   1  'Fixed Single
   Caption         =   "Test Signal Generator"
   ClientHeight    =   2904
   ClientLeft      =   1248
   ClientTop       =   2772
   ClientWidth     =   3312
   Icon            =   (Icon)
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   PaletteMode     =   1  'UseZOrder
   ScaleHeight     =   2904
   ScaleWidth      =   3312
   Begin VB.OptionButton optChannel
      Caption         =   "Channel 4"
      Height          =   195
      Index           =   4
      Left            =   240
      TabIndex        =   3
      Top             =   1560
      Width           =   1875
   End
   Begin VB.OptionButton optChannel
      Caption         =   "Channel 3"
      Height          =   195
      Index           =   3
      Left            =   240
      TabIndex        =   2
      Top             =   1320
      Width           =   1875
   End
   Begin VB.OptionButton optChannel
      Caption         =   "Channel 2"
      Height          =   195
      Index           =   2
      Left            =   240
      TabIndex        =   1
      Top             =   1080
      Width           =   1875
   End
   Begin VB.OptionButton optChannel
      Caption         =   "Channel 1"
      Height          =   195
      Index           =   1
      Left            =   240
      TabIndex        =   0
      Top             =   840
      Width           =   1875
   End
   Begin VB.TextBox txtInterval
      Height          =   285
      Left            =   1920
      TabIndex        =   5
      Text            =   "1000"
      Top             =   1860
      Width           =   555
   End
   Begin VB.Timer tmrInterval
      Interval        =   1000
      Left            =   2520
      Top             =   1020
   End
   Begin VB.CommandButton cmdCancel
      Cancel          =   -1  'True
      Caption         =   "Cancel"
      Default         =   -1  'True
      Height          =   375
      Left            =   1020
      TabIndex        =   6
      Top             =   2340
      Width           =   1215
   End
   Begin VB.OptionButton optChannel
```

```
      Caption         =   "Auto change every"
      Height          =   255
      Index           =   0
      Left            =   240
      TabIndex        =   4
      Top             =   1860
      Value           =   -1  'True
      Width           =   1695
   End
   Begin VB.Label lblPortAddress
      Alignment       =   2  'Center
      Height          =   195
      Left            =   300
      TabIndex        =   9
      Top             =   120
      Width           =   2715
   End
   Begin VB.Label lblTestChannel
      Alignment       =   2  'Center
      Height          =   195
      Left            =   300
      TabIndex        =   8
      Top             =   480
      Width           =   2715
   End
   Begin VB.Label Label1
      Caption         =   "ms"
      Height          =   255
      Index           =   1
      Left            =   2520
      TabIndex        =   7
      Top             =   1920
      Width           =   255
   End
End
```


1000
Cancel

modScantableGenerator - 1

```
Option Explicit

Public abort As Boolean
Dim gcdtable(), gcds As Integer
Dim offset() As Integer
Public scantableLength As Long
Dim ch As chSet
Dim chRepeatTime() As Integer


Private Function try_subtable(ch As chSet, depth, pos As Integer) As Boolean
' tries to fit a subtable in the space provided
' * ch is the table to try
' * depth is the depth of recursion of try_subtable
' * pos indicates where in the scantable the channels should go
Dim subch() As chSet
Dim gcd, i As Integer

    ' make Windows respond to other events (other applications and the abort button)
    DoEvents

    If depth > UBound(gcdtable) Then ReDim Preserve gcdtable(depth)
    ' gcdtable(depth) is greatest common divisor of the repeat times of the channels involved
    gcdtable(depth) = ch.gcd(chRepeatTime)
    ' gcd is the greatest common divisor of the repeat times of the channels involved
    ' divided by the table cell repeat time
    gcd = gcdtable(depth) \ gcdtable(depth - 1)
    If gcd >= ch.length Then
        ' enough space for all channels
        For i = 0 To ch.length - 1
            offset(ch.peek(i)) = i * gcdtable(depth - 1) + pos
        Next
        try_subtable = True
    Else
        ' not enough space for all channels, try to divide channels into subtables
        ReDim subch(gcd)
        For i = 0 To gcd - 1
            Set subch(i) = New chSet
        Next
        try_subtable = try_permutation(ch, subch, 0, gcd, ch.length, depth + 1, pos)
    End If
End Function

Private Function try_permutation(ch, subch() As chSet, hint, gcd, _
    chlen, depth, pos As Integer) As Boolean
' tries all possibilities of fitting the ch channels in gcd subtables
' * ch is the table containing channels to be divided amongst subch
' * subch is the array of subtables where the channels should go
' * hint is a parameter that allow try_permutation to check even distributions first,
'       greatly reducing processing time
' * gcd is the amount of subtables used
' * chlen is the number of channels in the original table
' * depth is the depth of recursion of try_subtable
' * pos indicates where in the scantable the channels should go
Dim i, j As Integer

    If abort Then
        ' user aborted scantable generation
        try_permutation = False
        Exit Function
    End If
    If ch.length > 0 Then
        ' recurse further
        For i = 0 To gcd - 1
            j = (i + hint) Mod gcd
            ' move channel from table to subtable
            subch(j).push (ch.pop)
            try_permutation = try_permutation(ch, subch, (hint + 1) Mod gcd, gcd, _
                chlen, depth, pos)
            ' reverse that move
            ch.push (subch(j).pop)
            If try_permutation Then Exit Function
        Next
    Else
```

```
        ' all channels are allocated to a subtable
        ' now check if all subtables are possible
        try_permutation = False
        For i = 0 To gcd - 1
            If subch(i).length = chlen Then Exit Function ' subtable same as table, worthless
            If subch(i).length > 0 Then ' subtables of length 0 are always possible
                If Not try_subtable(subch(i), depth, i * gcdtable(depth - 2) + pos) Then
                    Exit Function
                End If
            End If
        Next
        try_permutation = True
    End If
End Function

Public Function scantablePossible(channelRepeatTime() As Integer) As Boolean
' this function takes a requirement provided in channelRepeatTime() and produces offset(),
' the array that determines which channel has to go where
' return value indicates whether scan table is possible
Dim i As Integer

    Set ch = New chSet
    ch.clear

    ' make local copy of channelRepeatTime() to be used later by generateScantable
    ' fill channel array
    ReDim chRepeatTime(LBound(channelRepeatTime) To UBound(channelRepeatTime))
    For i = LBound(channelRepeatTime) To UBound(channelRepeatTime)
        If channelRepeatTime(i) <> 0 Then
            ch.push (i)
            chRepeatTime(i) = channelRepeatTime(i)
        End If
    Next
    ReDim offset(LBound(chRepeatTime) To UBound(chRepeatTime))

    ' scantable length is the least common multiple of all channel repeat times
    scantableLength = ch.lcm(chRepeatTime)

    ' if scantable length doesn't conform to IRIG, return error
    If 2 * scantableLength > maxMinorframelength Then
        scantablePossible = False
        Exit Function
    End If

    ReDim gcdtable(10)
    gcdtable(0) = 1
    gcds = 1
    abort = False
    ' show form indicating the algorithm is busy
    frmGenerating.Show
    scantablePossible = try_subtable(ch, 1, 0)
    Unload frmGenerating
End Function

Public Sub generateScantable()
' takes the offset() array generated by scantablePossible() and puts the channel
' values in frmmain.grdScantable
Dim i, j, chn As Integer
Dim Shift As Integer

    With frmMain.grdScantable
        frmMain.smtChanged = True
        minorframes = 1
        minorframelength = scantableLength * 2  ' digitals inbetween analogs req'd
        columns = scantableLength * 2
        frmMain.update_form                     ' resize grid
        .row = 1
        .SelStartCol = syncwords + 1
        .SelEndCol = .cols - 1
        .SelStartRow = 1
        .SelEndRow = 1
        .FillStyle = 1
        .text = digitals(LBound(digitals))      ' fill grid with first digital channel
        .FillStyle = 0
```

```
        .SelEndCol = 1
        .SelEndRow = 1
    End With

    ' shift scantable so syncword is in first position
    Shift = scantableLength - offset(ch.peek(ch.length - 1))
    For i = 0 To ch.length - 2
        chn = ch.peek(i)
        For j = (offset(chn) + Shift) Mod chRepeatTime(chn) To scantableLength - 1 _
        Step chRepeatTime(chn)
            'fill grdScantable with analog channel
            frmMain.grdScantable.col = 2 * j + 1
            frmMain.grdScantable = frmgen.lblChannel(chn)
        Next
    Next
End Sub
```

```
Public Const maxMinorframelength As Integer = 514  ' IRIG class I
Public Const maxMinorFrames As Integer = 256       ' IRIG class I
Public maxscantablelength As Long

Public sSmtSpecDatabasefile As String

Public testSignalSetup() As Integer
Public bitrate, wordrate As Long
Public channels(), eachminorframes(), analogs(), digitals(), dummy As String
Public samplefreqs() As Double

Public minorframelength, minorframes As Integer

Public columns As Integer

Public syncwords As Integer
Public syncwordbytes(0 To 3) As Byte

Public oleReport As Object

Public grdScantableChanged As Boolean
Public scantable() As String

Public asciiFile As String
'global var by hkt
Public filter_item_max As Integer   'max item in filter data table
Public string_is_QPSK As Boolean          'QPSK/FSK format chosen
Public string_is_kbs As Boolean           '200 kbs or 2 Mbs chose


Public Function doGcd(ByVal a, ByVal b As Integer) As Integer
' compute greatest common divisor
' Euclid's algorithm
    Do While (a <> b)
        If a > b Then a = a - b Else b = b - a
    Loop
    doGcd = a
End Function

Public Function gcd(a() As Integer) As Integer
' compute greatest common divisor of array
' ignoring zero entries
Dim i As Integer
Dim g As Long

    g = a(LBound(a))
    For i = LBound(a) + 1 To UBound(a)
        If a(i) <> 0 Then g = doLcm(g, a(i))
    Next
    gcd = g
End Function

Public Function doLcm(ByVal a, ByVal b As Long) As Long
' compute least common multiple
    doLcm = a * b / doGcd(a, b)
End Function

Public Function lcm(a() As Integer) As Long
' compute least common multiple of array,
Dim i As Integer
Dim l As Long

    l = 1
    For i = LBound(a) To UBound(a)
        If a(i) <> 0 Then l = doLcm(l, a(i))
    Next
    lcm = l
End Function

Public Sub mouseHourglass()
        Screen.MousePointer = 11 ' hourglass
End Sub
```

```
chSet - 1

Option Explicit

Private chList() As Integer
Private listLen As Integer

Private Sub Class_Initialize()
    ReDim chList(0)
    listLen = 0
End Sub

Public Sub clear()
    listLen = 0
    ReDim chList(0)
End Sub

Public Function length() As Integer
    length = listLen
End Function

Public Sub push(i As Integer)
    If listLen > UBound(chList) Then
        ReDim Preserve chList(listLen)
    End If
    chList(listLen) = i
    listLen = listLen + 1
End Sub

Public Function pop() As Integer
    If listLen > 0 Then
        listLen = listLen - 1
        pop = chList(listLen)
    Else
        err.Raise vbObjectError, , "Pop from empty set"
    End If
End Function

Public Function peek(ByVal i As Integer) As Integer
    If i < listLen Then
        peek = chList(i)
    Else
        err.Raise vbObjectError, , "Peek beyond end of set"
    End If
End Function

Public Function gcd(chfreq() As Integer) As Integer
Dim i, g As Integer

    If listLen = 0 Then err.Raise vbObjectError, , "Gcd from empty set"
    g = chfreq(chList(0))
    For i = 1 To listLen - 1
        g = doGcd(g, chfreq(chList(i)))
    Next
    gcd = g
End Function

Public Function lcm(chfreq() As Integer) As Integer
Dim i As Integer
Dim l As Long

    If listLen = 0 Then err.Raise vbObjectError, , "Lcm from empty set"
    l = CLng(chfreq(chList(0)))
    For i = 1 To listLen - 1
        l = doLcm(l, chfreq(chList(i)))
    Next
    lcm = l
End Function
```

Module1 - 2

```
Public Sub mouseNormal()
        Screen.MousePointer = 1 ' arrow
End Sub
Public Sub computeBitrateWordrate()
Dim i As Integer

    bitrate = Val(frmMain.cmbBitrate)
    For i = 1 To Len(frmMain.cmbBitrate)
        Select Case LCase(Mid(frmMain.cmbBitrate, i, 1))
            Case "k"
                bitrate = bitrate * 1000
            Case "m"
                bitrate = bitrate * 1000000
            Case "a" To "z"
                Exit For
        End Select
    Next
    wordrate = bitrate / 8
End Sub

Public Sub computeSampleFreqs()
Dim r, c, ch  As Integer
Dim i, pos, pos1, dtmin, dtmax, tot As Long

    ReDim samplefreqs(LBound(channels) To UBound(channels))
    saveScantable
    computeBitrateWordrate
    For ch = LBound(channels) To UBound(channels)
        pos1 = -1
        dtmin = minorframelength * minorframes
        dtmax = 0
        tot = 0
        i = 0
        For r = 0 To minorframes - 1
            For c = 0 To minorframelength - 1
                If scantable(r, c) = channels(ch) Then
                    If pos1 < 0 Then
                        pos1 = i
                    Else
                        If i - pos < dtmin Then dtmin = i - pos
                        If i - pos > dtmax Then dtmax = i - pos
                    End If
                    pos = i
                    tot = tot + 1
                End If
                i = i + 1
            Next
        Next
        If pos1 < 0 Then
            samplefreqs(ch) = 0
        Else
            pos1 = pos1 + minorframelength * minorframes
            If pos1 - pos < dtmin Then dtmin = pos1 - pos
            If pos1 - pos > dtmax Then dtmax = pos1 - pos
            If dtmin = dtmax Then
                samplefreqs(ch) = wordrate / dtmin
            Else
                samplefreqs(ch) = -(wordrate / (minorframelength * minorframes)) * tot
            End If
        End If
    Next
End Sub
Public Sub saveScantable()
Dim r, c As Integer
Dim msg As String
ReDim scantable(minorframes - 1, minorframelength - 1) ' allocate memory

    With frmMain.grdScantable
        For r = 1 To .rows - 1
            .row = r
            For c = 1 To .cols - 1
                .col = c
                If minorframes = 1 Then
                    scantable(0, (r - 1) * (.cols - 1) + (c - 1)) = .text
```

```
                Else
                    scantable(r - 1, c - 1) = .text
                End If
            Next
        Next
    End With
End Sub

Public Sub restoreScantable()
Dim r, c As Integer

    With frmMain.grdScantable
        For r = 1 To .rows - 1
            .row = r
           For c = 1 To .cols - 1
                .col = c
                If minorframes = 1 Then
                    .text = scantable(0, (r - 1) * columns + (c - 1))
                Else
                    .text = scantable(r - 1, c - 1)
                End If
            Next
        Next
    End With
    Erase scantable  ' free memory
End Sub
```