National Défense
Defence nationale

MULTIPURPOSE DATA INTERFACE BOARD (DIB)

by

Capt Yves Simoneau and Caroline Tom

DEFENCE RESEARCH ESTABLISHMENT OTTAWA
REPORT NO. 1332

Canada

July 1998
Ottawa

Abstract

Defence Research Establishment Ottawa is pursuing an in-house research activity in spread-spectrum technology to support development of robust, anti-jam satellite communications for the military. The in-house effort consists of developing a system simulator, including both a ground terminal processor and a payload processor, to research the techniques involved in spread-spectrum synchronization. For these experiments, a multipurpose data interface board is required for different data operations, and is the subject of this report. The board is composed mainly of an erasable programmable logic device to reduce the number of integrated circuits and to add flexibility to the design. The board was designed to perform three functions. The first function is the data format conversion between a ground terminal processor and a data source, and likewise, a payload processor and a data sink. The second function is to provide an interface to a separate direct link between the payload and ground terminal subsystems for transmitting a reference pulse for synchronization. The third function is to provide a set of debug latches for the user. In this document, the software and hardware details are provided along with a user's guide for the board.

Résumé

Le Centre de recherche de la défense Ottawa, dans un projet de recherche interne, travaille au développement de technologies à spectre étalé reliées à un système de communications militaires par satellite robuste et à l'épreuve de l'interférence. Le projet interne se résume au développement d'un simulateur pour des recherches sur certains aspects difficiles, telle la synchronisation du spectre étalé. Pour ces expériences, une carte d'interface multi-fonctions pour les données est requise pour les différentes opérations sur les données et est le sujet de ce rapport. La carte a été construite à l'aide d'une puce logique effaçable et programmable pour remplacer les puces et pour donner un maximum de flexibilité à la carte. La carte a egalement été conçue pour satisfaire trois différentes fonctions. La première fonction est la conversion du format des données entre le processeur du terminal au sol et une source de données, et de même, le processeur de la charge utile et un collecteur de données. La seconde est l'interface à un lien direct séparée entre la charge utile et la station terrestre produisant une impulsion pour la synchronisation. La troisième est de produire trois paneaux contrôleurs. Dans ce document, tous les détails sont données ainsi qu'un guide pour l'utilisateur.

Executive Summary

Defence Research Establishment Ottawa is pursuing an in-house research activity in spread-spectrum technology to support development of robust, anti-jam satellite communications for the military. The MILSATCOM (Military Satellite Communications) group is developing a system simulator to research some critical techniques such as spread-spectrum synchronization. In the simulation of such a system, a data source and a data sink are used at the transmitter and receiver for data communications once synchronization is achieved. Data generated by the source is in the form of a serial bit stream which must be converted to an appropriate data word for modulation and transmission. At the receive end, data is demodulated into data words which must be converted back into a bit stream for the data sink. The transformation of this type of data to the proper format will be done via a multipurpose data interface board (DIB), designed and developed at DREO. This report describes the interface board, its functions and its operation.

The board is designed to perform three distinct functions. The first, is the data format conversion between a ground terminal processor or payload processor and a data source or sink respectively. The second function is to provide a means to access a direct link between the payload and ground terminal simulator subsystems. The direct link is used to transmit a reference pulse to initiate the synchronization process. The third function of the DIB is to provide a set of debug latches for the user. The multipurpose data interface board was designed to be able to operate on the transmit or receive side of the link. Thus, both the ground terminal and payload simulator subsystems use the same board to interface with a data source and sink respectively.

Flexibility was a key aspect of the board design. In addition to designing the board to operate at either end of the link, an erasable programmable logic device (EPLD) was used to implement much of the digital circuitry. The EPLD helps in saving physical space on the board and gives the user flexibility to rapidly make modifications or corrections.

The EPLD was found to be very useful in replacing the numerous small integrated circuits normally required for any design. The inner functions of the EPLD were divided into five main logic blocks. They are the clock generator, the address decoder, the command register, the transmit shift and the receive shift. The design and programming of the EPLD was done on a personal computer simplifying design changes. Substituting one EPLD for all of the usual integrated circuits meant rapid development of the hardware as well as easy customization of the hardware interface. Future expansion is also possible since the EPLD is used at only 80% of its capacity. There is also board space for additional interface logic devices.

The DIB was successfully tested and has been integrated in the ground terminal and payload simulator subsystems. The DIB will be used in the upcoming Skynet trials for uplink synchronization and data communication.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

BER	Bit Error Rate
CRC	Communication Research Centre
D/C	Down Converter
DIB	Data Interface Board
DIP	Dual Inline Package
DREO	Defence Research Establishment Ottawa
DSP	Digital Signal Processor
EHF	Extremely High Frequency
EPLD	Erasable Programmable Logic Device
FSK	Frequency-Shift-Keying
GT	Ground Terminal
I/O	Input/Output
IOE	Input/Output Enable
LDR	Low Data Rate
MDR	Medium Data Rate
MILSATCOM	Military Satellite Communication
MOU	Memorandum Of Understanding
N/A	Not Applicable
NWR	Not Write/Read
PC	Personal Computer
PL	Payload
PLCC	Plastic Leadless Chip Carrier
PN	Pseudo Noise
RESDIP	Resistor Dual Inline Package
RX	Receive
RXOVR	Receive Overflow
RXRDY	Receive Ready
SATCOM	Satellite Communication
SHF	Super High Frequency
TTL	Transistor Transistor Logic
TX	Transmit
TXMT	Transmit Empty
TXUND	Transmit Underflow
U/C	Up Converter
UK	United Kingdom

1. Introduction

1.1 Background

The Military Satellite Communications (MILSATCOM) group at Defence Research Establishment Ottawa (DREO) and Communications Research Center (CRC) have been involved in the research and development of Extremely High Frequency (EHF) Satellite Communications (SATCOM) to better assess the associated capabilities, limitations and complexities. Wide-band frequency-hopping and onboard satellite processing features of EHF SATCOM can reduce the effects of electronic interference thereby offering communications robustness. For onboard processing, uplink and downlink synchronization play a key role. The MILSATCOM group in-house effort includes examining the synchronization aspects of EHF SATCOM.

Through a Memorandum of Understanding (MOU) with the United Kingdom (UK), DREO and CRC are using the Skynet 4A EHF transponder to carry out the synchronization experiments. The Skynet transponder receives an EHF signal and translates it to an SHF signal to be retransmitted. Uplink and downlink synchronization techniques were examined separately. The downlink portion has already been completed and documented [1]. Work has commenced on the uplink synchronization and communications experiments.

As in the downlink synchronization experiments, ground terminal (GT) and payload (PL) simulator subsystems for the uplink synchronization experiments are developed in-house. These two subsystems are ground-based and are integrated with their respective transmit and receive antenna terminals, which are separated by a distance of approximately 1 km. Each simulator subsystem consists of a processor and various interface boards connected to IF and RF equipment. Nominal transmit and receive frequencies are 44.6 and 7.6 GHz with a hopping bandwidth of 50 MHz. A general system block diagram is given in Fig. 1. A more detailed description of the system is provided in Section 2.2.

In an actual system, uplink synchronization only begins once downlink synchronization is achieved. For the uplink synchronization experiments, the process of downlink synchronization is simulated by providing the GT simulator subsystem with a reference of the PL simulator subsystem clock via a direct serial link. The GT uses this reference to initiate the uplink synchronization process. Synchronization probes are transmitted by the GT simulator subsystem to the PL simulator subsystem based on different hypotheses. The PL simulator subsystem processes the signals received relative to its own clock and formulates a synchronization response which must be relayed back to the GT. The synchronization response indicates whether the synchronization probes were detected. Once detected, the GT clock is corrected. Fine synchronization probes are then transmitted and the synchronization responses received for these probes are used to fine-tune the GT clock. Once this is completed, synchronization is achieved and data communication can begin. The modulation scheme used for data communication is 8-ary frequency-shift-keying (FSK) and at a rate of 2.4 kb/s [2].

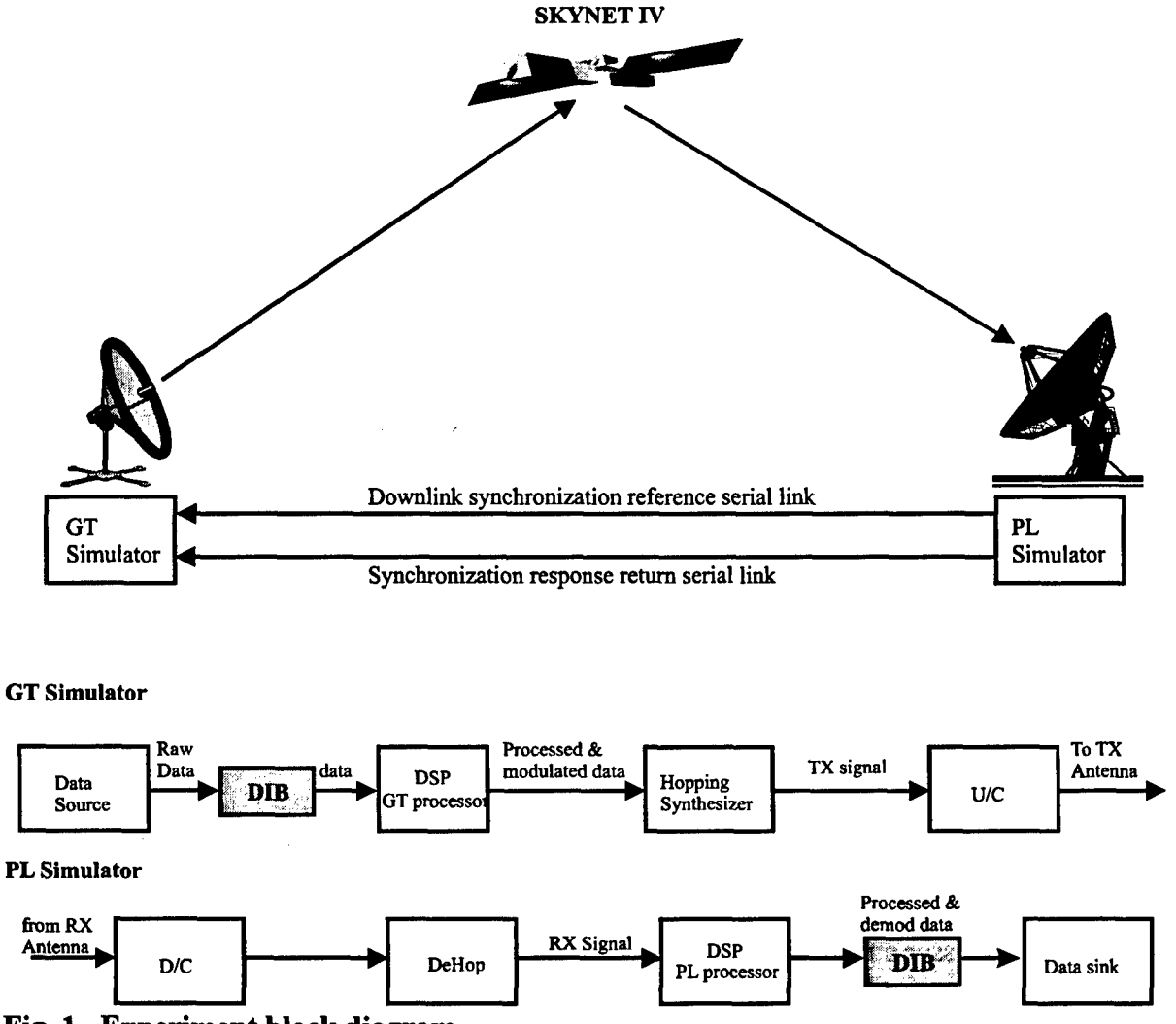

Fig. 1. Experiment block diagram

1.2 The Task

The uplink synchronization trials setup includes a GT simulator subsystem (located in Building 2 at CRC) which operates as the EHF transmit station to the Skynet satellite. At the satellite, the EHF transmit signal is translated to SHF and retransmitted to a PL simulator subsystem (located in T85 at DREO). In addition to the satellite link setup, a direct serial link, called the "downlink synchronization reference serial link", is used to transmit a reference pulse of the PL clock which serves as a starting point for the synchronization process. The PL simulator subsystem is responsible for generating the reference pulse to be received by the GT simulator subsystem. It is important to note that even with the reference pulse, there is still a difference between the GT and PL clocks due to the delays experienced with the satellite and serial link transmissions. An additional serial link, called the "synchronization response return serial link", is used to return the synchronization responses formulated by the PL simulator subsystem to the GT simulator subsystem.

For the downlink synchronization reference serial link, an interface is required between each subsystem processor and the serial link. Furthermore, for data communications, HP1645 data error analyzers are used as the data source and sink. The HP1645 generates and accepts a synchronous serial RS232 bit stream of data. As the modulation scheme is 8-ary FSK for the trials, a formatter is required to convert the serial bit stream to an appropriate data word format for modulation. Both the interface to the downlink synchronization reference link and the data formatting requirements are handled by a multipurpose data interface board.

The multipurpose Data Interface Board (DIB) was designed and fabricated to perform three different functions. Its first function is to handle the format conversion requirements between a data source and the GT processor, and similarly, the PL processor and a data sink. From the point of view of the GT, serial data is converted from RS232 to transistor-to-transistor logic (TTL) levels and formatted into 12-bit words. The reverse is done on the PL processor side. The DIB's second function is to provide an interface to the downlink synchronization reference serial link between the PL and GT. This link is used in the uplink synchronization experiments to transmit from the PL to the GT a reference pulse as an estimate of the PL clock and the PL's position in the hopping pattern. The reference pulse provides a starting point for the GT to begin the uplink synchronization process and is analogous to the information available if downlink synchronization had been performed. The third function of the DIB is to provide a set of debug latches. These latches may be used as a debugging tool by the user at any point of the experiment to locate any problems in the user's subsystems.

1.3 Objectives and Report Outline

The primary objective of this report is to describe the design and implementation of the three functions of the DIB. The other objective of this report is to provide a detailed user's guide for the operation of the DIB and the information necessary to carry out any modifications to the DIB.

In Section 2 of this report, the design concept of the DIB is described for each of the three functions as well as the addressing concept for the board. The hardware description (Section 3) includes reasons behind the choice of each element of the DIB. The report also includes a description of the testing of the DIB in Section 4 to help the user to understand how the DIB functions and enable testing of the DIB if modifications are made. The details of the hardware and firmware implementation are contained in Appendix A. Details include a description of the various logic circuit blocks in the EPLD and of the physical layout of the DIB. Appendix B provides examples of the operations of the DIB with sample code written for use on a TMS320C25 digital signal processor (DSP) board built by Spectrum Signal Processing Inc. Appendix C lists the programs used for the testing described in Section 4.

2. Design Concept

2.1 General

The design of this board was based on a prototype board providing similar functions used in the downlink synchronization experiments [1]. Additional functions have been added to accommodate uplink synchronization experiment requirements. In this section, the overall system setup will be described briefly to explain where and why the DIB is positioned where it is. The three different functions of the DIB will also be explained.

2.2 System Description

As described in Section 1.1, the uplink synchronization experiments are carried out over the Skynet 4A EHF transponder. The use of the EHF transponder requires the development of both GT and PL simulator subsystems as ground-based systems and results in a setup as shown in Fig. 1. The GT and PL processor functions are developed on digital signal processing (DSP) boards installed in personal computers. Each processor also communicates with a hopping synthesizer controller (HSC) and a data source (GT) or a data sink (PL). In the GT simulator subsystem, the HSC computes the frequency of the next hop to be generated by the frequency synthesizer for transmission. In the PL simulator subsystem, the HSC is used to dehop the received signal before being processed. Other components of the subsystems include the interface to a downlink synchronization reference serial link and to a synchronization response return serial link described below.

For the uplink synchronization experiments, a reference of the PL clock is transmitted via a serial link, referred to in this report as the downlink synchronization reference serial link, to provide the GT with a starting point for synchronization. The serial link simulates the downlink portion of a practical SATCOM system with onboard processing. Once the reference is received and the GT has established a starting point, the GT simulator subsystem transmits synchronization probes to the PL simulator subsystem. The PL simulator subsystem processes the received data and returns to the GT an estimate of the synchronization accuracy for the probes sent. The estimate is transferred through a separate link referred to as the synchronization response return serial link.

Once synchronization is achieved, data generated by a data source can also be transmitted from the GT to the PL. The modulation scheme used in the uplink synchronization experiments is 8-ary frequency-shift-keying (FSK). The data source generates a bit stream of data which is formatted and stored in a data latch. The GT processor subsequently reads the latch and forwards the data (three bits at a time) to the hopping synthesizer controller [3] to be transmitted. At the PL receiver, the received signal is dehopped and demodulated. The demodulated data is then transferred by the PL processor to a data latch and ultimately to the data sink. The interface between each processor and the data source or sink, and between each processor and the downlink synchronization reference serial link, is provided by a data interface board (DIB) which is the subject of this report.

2.3 DIB General Description

The primary function of the DIB is to format the data transferred between the data source and GT processor, and similarly between the data sink and PL processor (Fig. 2). For the uplink synchronization experiments and for this report, the data source and sink is a Hewlett Packard HP 1645 data error analyzer. For data flowing from the data source to the GT, the HP 1645 provides a serial bit stream through an RS232 link to the DIB. The DIB formats the serial data into 12-bit words and stores them in a latch until the GT processor reads the data.

The operation is reversed for data flowing from the PL processor to the data sink and is implemented on the same DIB. Hence the DIB is designed for use by either the GT or PL simulator subsystems. More details are given in Sections 3.3.4 and 3.3.5 (Hardware/EPLD) and in Appendix A on the way the data is converted from one format to the other.

There are two other secondary functions of the DIB. One is to provide an interface to a downlink synchronization reference serial link between the PL and GT for the PL clock reference signal. The other is to provide three debug latches. These latches can be used as a debugging tool by the user at any point of the experiment to access intermediate values. Each of these functions will be discussed in more detail in the following subparagraphs.

2.3.1 Data Operation

The DIB provides the necessary data conversion function between the GT and PL processors and the HP 1645. On the PL side, the parallel data must be disassembled into a serial bit stream and converted from TTL levels to RS232 levels. The reverse is true on the GT side. The direction of data flow is determined by the address port accessed by the DSP and the DSP operation (read or write). This is discussed further in Section 2.4.2.

The block diagram of the data transformation process for the data link is given in Fig. 2. The DIB contains two separate interfaces. One side of the DIB is connected to an HP 1645 data error analyzer via an RS232 interface. On the other side, the DIB communicates with a DSP board. For the uplink synchronization experiments, Spectrum Signal Processing Inc. DSP boards with the DSPLINK interface are used. The DSPLINK interface has 16 data lines of which 12 are used for data transferred between the DIB and the GT or PL processor. From the PL point of view, a 12-bit data word needs to be converted to a single bit stream in order to transfer it through the RS232 connection to the data error analyser. The data conversion is achieved by loading the 12-bit data word into a shift register. The 12-bit data word is then clocked out one bit at a time. The resulting data bit stream will then go through a line driver to transform the initial TTL-level signal into an RS232-level signal before being transferred to the HP1645A. The next 12-bit word is loaded into the shift register once all the previous 12 bits have been transferred. If no data is loaded into the shift register before the next bit is to be clocked out, then an underflow flag bit is set on the status register of the DIB. The underflow bit is described further in Section 2.4.1.

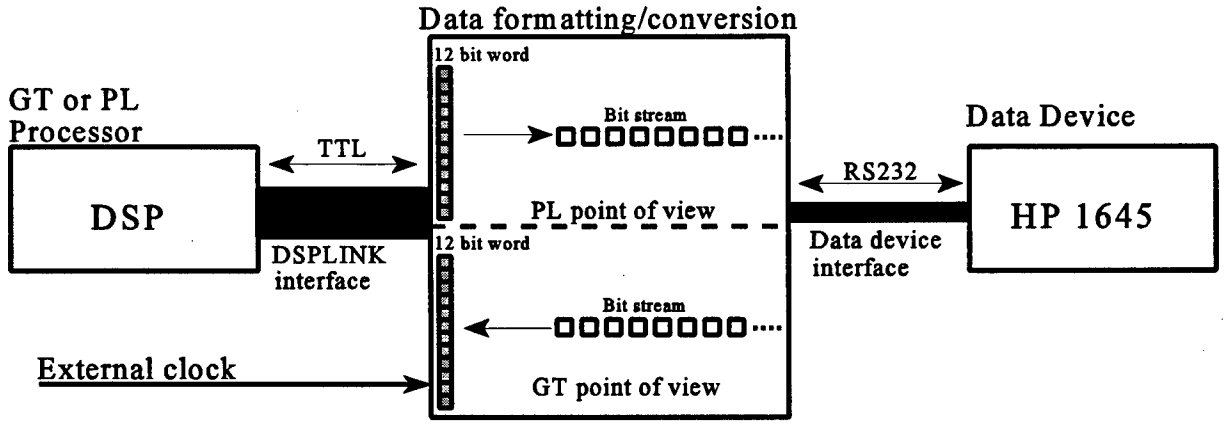

Fig. 2. Data transforming process

From the GT point of view, the data is assembled from a single bit stream to a 12-bit parallel stream. The data goes through a line receiver that transforms the data format from RS232 to TTL levels. The data is then assembled into a 12-bit word using a shift register and loaded into a latch. The DIB will then set a flag that will tell the user when the data is available to be read. Once the 12-bit word is read by the GT processor, the next 12 bits assembled in the shift register are loaded in the latch. If the previous data in the latch is not read by the GT processor before the next 12 bits are ready to be transferred from the shift register, a data overflow bit is set on the status register. The data overflow bit is described in Section 2.4.1.

The use of only twelve out of the sixteen data lines first came about as a requirement in the downlink synchronization experiments [1]. In the implementation of the DIB for the uplink synchronization experiments, the 12-bit format was retained as it is a suitable grouping of the data bits for the 8-ary FSK modulation specified for data transmission [2]. It is noted that a grouping of 15 bits is also suitable for the data modulation scheme, however, the clock signals required for the data conversion is facilitated if the word length remains an even number.

In both cases, an external clock signal is required for the DIB operations. The clock frequency required is the data bit rate. For the uplink synchronization experiments, the data rate is 2.4 kb/s. This clock signal is used in the transformation of the data from one data line to 12 data lines and vice versa by clocking data in and out of shift registers. A derivative of the clock signal is generated by the DIB and is used by the DIB as a control signal for the transformation process.

2.3.2 Frame Zero Pulse

In practice, a GT performs downlink synchronization to obtain an estimate of the PL clock before initiating uplink synchronization. For the uplink synchronization trials, since only the uplink portion will be implemented over the satellite, there is a need to simulate the downlink portion that

the GT will use as a basis for beginning uplink synchronization. It was decided that an estimate of the PL clock would be provided in the form of a pulse which would be transmitted from the PL simulator subsystem to the GT simulator subsystem via a direct link. The rising and falling edges of the pulse, which is referred to as the Frame Zero (FR0) pulse, will be used to identify the start and end of the zeroth frame in the pseudorandom frequency hop sequence used for the experiment. This pulse signal can also be used in tracking the PL clock.

The FR0 pulse is transmitted from the PL simulator to the GT simulator via a direct serial RS232 link between them. This serial link is referred to as the downlink synchronization reference serial link. The interface between the processors and the direct link is provided by the DIB. The pulse is generated by the PL processor when a "1", followed by a "0", is written to the appropriate command data line (FR0_out) of the DIB command register (details of the command register are found in Sections 2.4.2 and 3.3.3 of the main document, and Section 4.2.4 of Appendix B). The FR0_out data line goes to a line driver where the signal is converted from TTL to RS232 levels. The signal is then transmitted through the serial link between the PL and GT simulator subsystems. When received on the other DIB the signal is converted back to TTL levels and is detected by the GT processor on one of the DIB status lines (FR0_in) (details of the status register are found in Sections 2.4.2 of the main document and Section 4.2.4 of Appendix B). The GT processor will then use the status information to verify the alignment of its clock and update the appropriate counters as needed. A flow diagram of the process is shown in Fig. 3.

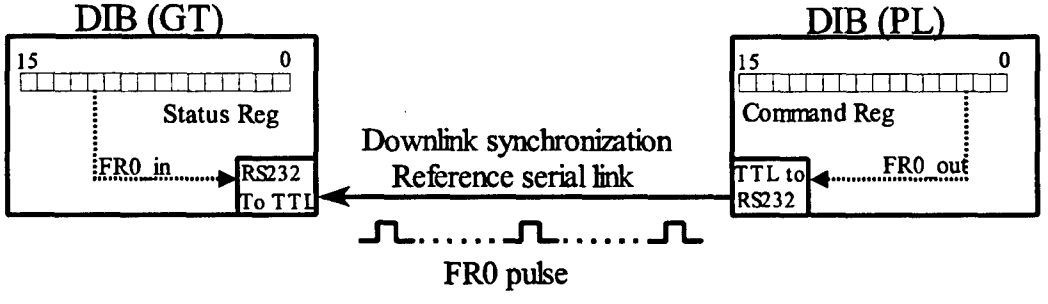

Fig. 3. Frame zero pulse flow diagram

2.3.3 Debug latches

This board is also equipped with three 16-bit debug latches. These latches are there to help the user in programming and testing the system. This function does not affect the data operation function of the board. These latches will store any data the user writes to them. The user can therefore read the data on the latches with a logic analyser to verify the data and/or identify any

hardware or software problems. Data can be written to the latches as often as is required. An example of writing to a debug latch is given in the user's guide (Appendix B). The latches are read by probing the pins of the appropriate integrated circuit. In addition, debug latch #1 can be accessed conveniently by a connector (26-pin) that has been set up on the front panel.

2.4 Addressing Concept

The interface between the DIB and the DSP is specified by the DSPLINK interface of Spectrum Signal Processing Inc. DSP boards. The 50-line DSPLINK interface is a high-speed bidirectional interface supporting 16-bit data transfers. To communicate with the DIB, the signals required are provided by the minimum subset of DSPLINK shown in Table 1. With 4 available address lines, there are 16 possible addresses. The DIB uses two of the sixteen addresses thereby allowing other boards to use the same DSPLINK interface without addressing conflicts.

Signal name	In/Out	Description
D0-D15	In/Out	16 fully buffered bi-directional data lines
A0-A3	Out	4 buffered address lines
NWR	Out	Read/Write line to indicate direction of data transfer
NIOE	Out	I/O enable line, indicates access to one of 16 standard I/O ports only
NRESET	Out	Reset line, same as reset into processor
GND	-	Signal ground

Table 1. DSPLINK interface [4] signal subset used by the DIB.

2.4.1 Base Address

For the uplink synchronization experiments, several interface boards are present in both PL and GT simulator subsystems which use the DSPLINK interface. As a result, a DSP backplane with the DSPLINK interface was developed to facilitate the connection of the various boards to the DSP. In order to address each of these interface boards properly, each board must use non-conflicting addresses. Furthermore, added flexibility is available if the assignment of addresses is selectable. The DIB requires one address line and the NWR line to decode the necessary ports within its address space. This is described further in the next section. The three remaining addresses lines can be used to select any block of 2 address assignments among the 16 available. For the uplink synchronization trials, the base address is set by enabling the appropriate address lines using the DIP switch on each board. Details of the DIP configuration for the DIB are included in Fig. A1 in Appendix A and Table B1 in Appendix B.

2.4.2 On Board Addressing

The addressing on the DIB is decoded using one address line (A0) and the NWR line, thus giving 4 separate I/O ports. All other address lines going to the board will be used only to select the base address as described above. The functions of the four I/O ports of the DIB are listed in Table 2. Each function is described further in the paragraphs below.

A0	NWR	Address Name	Function
0	0 (write)	Write Data	sends the data to the data sink or to a debug latch
0	1 (read)	Read Data	gets the data from the data source
1	0 (write)	Write Command	commands the DIB and selects destination of Write Data
1	1 (read)	Read Status	reads status from the DIB

Table 2. DIB addressing

2.4.2.1 Write Data

The purpose of this address port is to allow data to be transferred to either the data sink or to one of the three debug latches. The designated device is chosen by giving the appropriate command specified in the command register (described in the 'Write Command' paragraph) prior to accessing this port. An example of this operation can be found in Appendix B.

2.4.2.2 Read Data

The purpose of this address port is to allow data to be transferred from the data source. The Read Data operation is prompted by the RXRDY flag in the status register which is set when data is available. The data, which has been formatted by the DIB, is stored in a latch on the DIB. An example of this operation can be found in Appendix B.

2.4.2.3 Write Command

The purpose of this address port is to allow a command to be sent to the DIB. The user can either give a software reset, address one of the 3 debug latches, address the data latch or generate the Frame Zero pulse. Each function is specified by a bit or combination of bits in the command register as shown in Fig. 4 and Table 3. Note that for the data transfer operations, data lines D0 and D1 are used only to select the destination of the transfer to Write Data. To complete the operation, the Write Command is immediately followed by a Write Data where the actual data transfer takes place (described above). An example of this operation can be found in Appendix B.

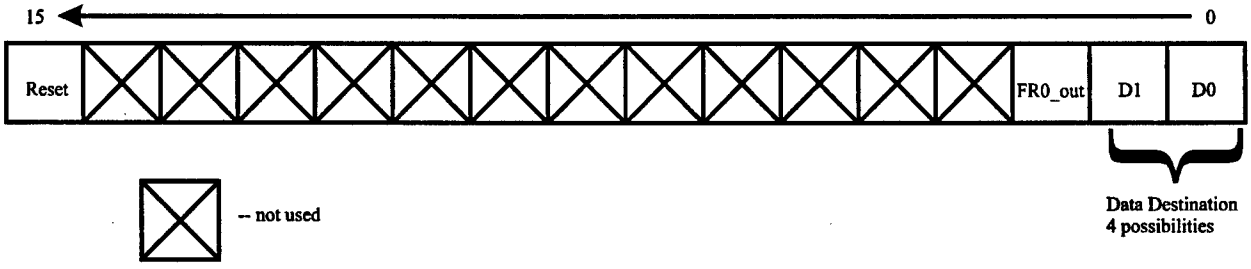

Fig. 4. Command register bits

Command Register	Type	Function
0000000000000**00**	TX Hold	selects the data latch to be the destination for the data (followed by conversion to a single bit stream)
0000000000000**01**	Debug #1	selects debug latch #1 to be the destination for the data
0000000000000**10**	Debug #2	selects debug latch #2 to be the destination for the data
0000000000000**11**	Debug #3	selects debug latch #3 to be the destination for the data
0000000000000100	FR0_out	puts a '1' on the Frame Zero pulse to be transmitted through the serial link
1000000000000000	Reset	Gives an active-high software reset.

Table 3. Write command functions

2.4.2.4 Read Status

The purpose of this address port is to read the 5 status bits of the DIB. The location of the 5 status bits in the status register is shown in Fig. 5. Four status bits are used to monitor the transfer of data to and from the data sink and source. In addition, one status bit is allocated for receiving the FR0 pulse. Each of the status bits is described in the following subparagraphs. Examples of the operations carried out involving the status register can be found in Appendix B.

a. **RXRDY**: This status bit will tell the user when the data is ready to be read from the DIB data latch. This bit will be set when the DIB has accumulated 12 bits of data from the data source and has transferred them into the data latch on the DIB.

b. **RXOVR**: This status bit will tell the user that while transferring data from the data source the 12 data bits waiting on the DIB data latch were not read before the next 12 bits were available and thus were overwritten resulting in loss of data.

c. **TXMT**: This status bit will tell the user when the DIB is ready to accept the next word of data into the hold register. This bit is set when the DIB has transferred the data from the hold register to the shift register to be converted into a serial bit stream for the data sink.

d. **TXUND**: This status bit will tell the user that the next 12-bit word was not available in the hold register to be transferred to the shift register after the previous 12 bits were clocked out of the shift register to the data sink.

e. **FR0_in**: This status bit indicates the state of the FR0 pulse transmitted on the downlink synchronization reference serial link from the PL simulator subsystem.

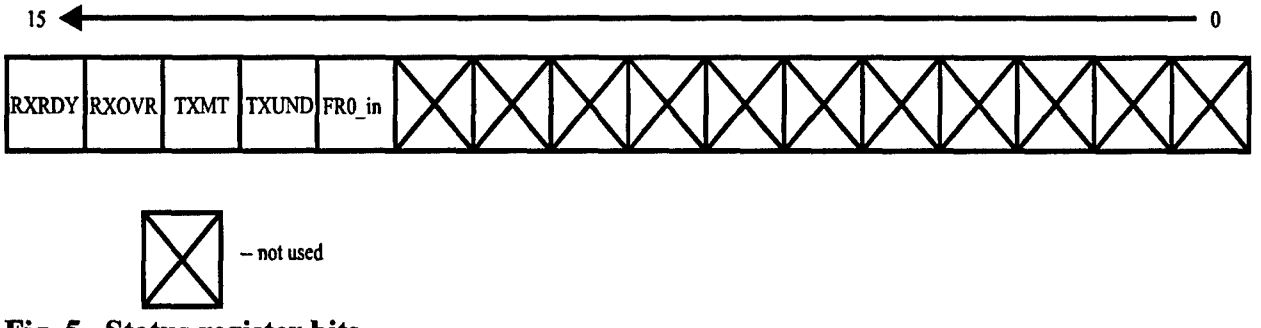

Fig. 5. Status register bits

3. Hardware

3.1 General

In this section, details will be given about the physical components of the DIB and on the different components of the Erasable Programmable Logic Device (EPLD) that are used to implement the various functions of the DIB.

3.2 Board Construction

The elements of the DIB include the data formatting circuitry, the data conversion clock, the debug latches, the TTL/RS232 conversion circuitry, the Frame Zero pulse interface and the DSPLINK interface. The bulk of the functions of the DIB are implemented on an EPLD. The EPLD used here is an Altera EPM5128JC-1. Fig. 6 represents the general DIB block diagram. The DIB was implemented on a custom printed circuit board. The physical details of the board are given in Appendix A. The DIB schematic is given in Fig. A1 of Appendix A.

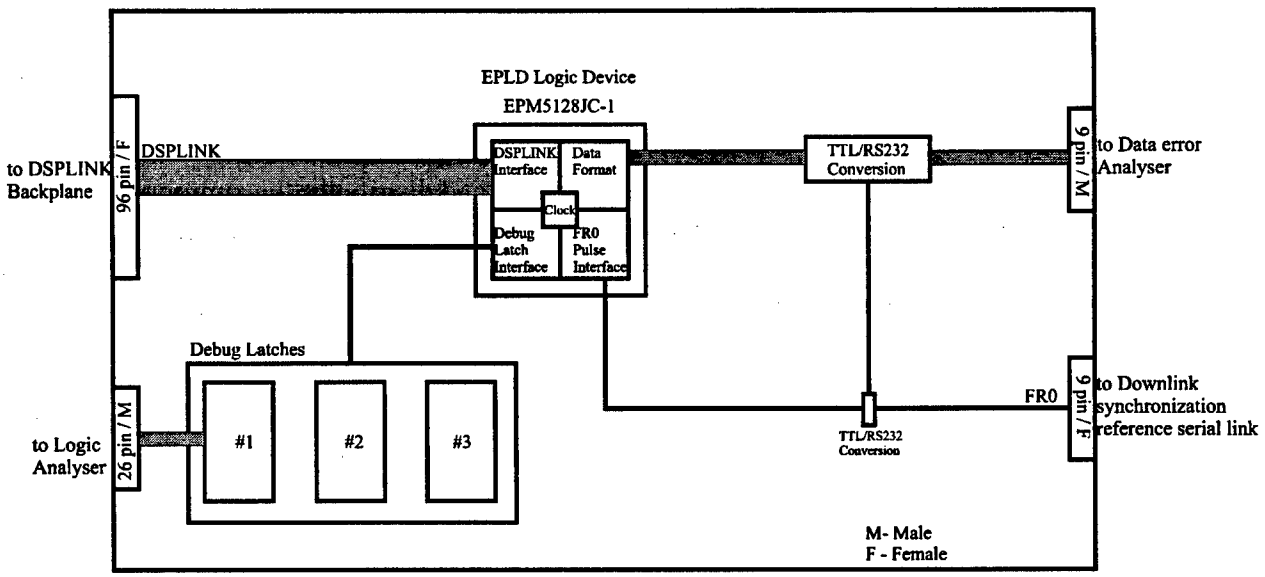

Fig. 6. DIB block diagram

3.3 Erasable Programmable Logic Device

The EPLD is an integrated circuit that consists of many macrocells. The collection of elements or macrocells which are used to implement a particular function in the EPLD produce a macro. A macro can be made of other macros. The EPLD is flexible as it can be erased and reprogrammed to implement modifications for the DIB. The EPLD was found to be very useful in replacing the numerous small integrated circuits thus reducing the physical space occupied on the board. The design and programming of the EPLD was done on a personal computer. Substituting one EPLD for all of the usual integrated circuits meant rapid development of the hardware as well as easy customization of the hardware interface. The internal chip configuration and the pin usage (input, output or both) are programmed into the EPLD. Modifications can be made on the EPLD by erasing it using ultraviolet light and then reprogramming the EPLD.

The functions implemented in the EPLD include the clock generator, the address decoder as the logic interface between the DSP and the DIB, the command register, the transmit shift, the receive shift and status bit generation. These functions are described in greater detail in the following paragraphs and in Appendix A. The EPLD used in this design is the Altera EPM5128JC-1 which consists of 128 macrocells in a 68 pin PLCC package. A schematic of the represented EPLD circuit is given in Fig. A8 of Appendix A.

3.3.1 Clock Generator

The clock generator function of the EPLD provides and distributes the appropriate clock to the other EPLD functions. This function transforms an external clock signal to the frequency needed by the board to support its operation. The DIB provides two external clock options. The user can provide the actual 2.4 kHz clock needed for the uplink synchronization experiments, or can provide a 432 kHz clock from which the clock generator divides it down to a 2.4 kHz clock. The 432 kHz clock option was initially used in the downlink synchronization experiments [1]. The selection of the external clock is made by the DIP switch on the DIB (this is further described in Section 3 of Appendix B). For the uplink synchronization experiments, the 2.4 kHz external clock is selected.

In the clock generator circuit, the 2.4 kHz clock is called the 'Shift Clock' and is used to transfer data to and from the data sink and source, respectively to satisfy the 2.4 kb/s data rate used in the uplink synchronization experiments [2] . Another clock is created for internal use, called the 'Latch Clock'. The DIB generates the 'Latch Clock' by dividing the 2.4 kHz clock by a factor of 12, thus giving a 200 Hz signal . This is used to signal the DSP when 12 bits of data have been assembled or disassembled. A schematic of the represented circuit for the clock generator is given in Fig. A9 of Appendix A.

3.3.2 Address Decoder

This EPLD function covers the decoding of the various addresses and sets the base address of the DIB. The base address is specified by address lines A1,A2, and A3 of the DSPLINK interface as described in Section 2.4.1. The selection of the base address is made by the user and is applied by correctly setting the DIP switches on the DIB (this is further described in Section 3 of Appendix B). The DIP switch settings are then compared with the corresponding address lines to enable the 3-to-8 decoder when there is a match. The 3-to-8 decoder uses address line 0 (A0) and the NWR line to generate the various I/O port signals required for the DIB (described in Table 2 of Section 2.4.2). In addition to the enable signal produced by comparing A0-A3 to the DIP switch settings, the NIOE signal is also used to enable the 3-to-8 decoder. A schematic of the represented circuit for the address decoder is given in Fig. A10 of Appendix A.

3.3.3 Command Register

This EPLD function provides three different operations on the DIB. The command register enables the user to select whether data is transferred to either the data latch or to one of three debug latches on the DIB. The command register also allows the user to access the downlink synchronization reference serial link in order to transmit the FR0 pulse. Finally the command register allows a software reset of the DIB. The operations are identified by specific data bits of the command register as described in Section 2.4.2 and are actioned when the Write Command port is accessed. The reset and FR0 pulse generation are straightforward in that these operations are carried out by writing a "0" or a "1" to the appropriate bit. However, as discussed in Section 2, when the user wishes to transfer data to either one of the debug latches or to the data latch for format conversion, the Write Command operation must be followed by a Write Data operation where the actual data is transferred. In this implementation, the Command register bits D0-D1 are used to select the destination for the data transfer. D0-D1 are further decoded to enable the appropriate destination device (data latch or debug latch). A schematic of the represented circuit for the command register is given in Fig. A11 of Appendix A.

3.3.4 Transmit Shift

This EPLD function covers the conversion of 12-bit data word stored in a latch into a serial bit stream for the data sink. After being latched, all the data lines are directed into shift registers that will convert the data coming from 12 parallel data lines to 12 consecutive bits on a single data line. The Shift Clock (2.4 kHz) described in Section 3.3.1 controls the conversion process. Data from the DSP is transferred in the lower 12 bits of the 16 data lines on DSPLINK. On the DIB, the twelve bits are shifted out sequentially such that the first bit out of the shift register corresponds to the least significant bit (LSB) of the data word, (i.e. D0). The transmit shift logic interface also provides two status bits for the user to monitor data flow through the DIB (TXMT, TXUND). Further details on the status bit can be found in Section 2.4.2. A schematic of the represented circuit for the transmit shift is given in Fig. A12 of Appendix A.

3.3.5 Receive Shift

This EPLD function covers the conversion of a serial bit stream from the data source into a 12-bit data word stored in a latch. A data stream is directed into a shift register which, with the help of the Shift Clock, will transform 12 bits from the data stream to a 12-bit word and store it in a 16-bit data latch. The data latch consists of the upper 4 bits being zeroed and the lower 12 bits holding the 12-bit word assembled by the shift register. The result of the shifting process is that the first bit of the data stream is stored in the LSB of the data latch (i.e. D0). The receive shift logic interface also provides two status bits for the user to monitor data flow through the DIB (RXRDY, RXOVR).

This macro also contains the circuitry for assembling the DIB status bits that form the status register. The description of the status bits was given in Section 2.4.2. A schematic of the represented circuit for the receive shift is given in Fig. A13 of Appendix A.

3.3.6 Status Bit Generation

In both the "Transmit Shift" and "Receive Shift" macros, there are two macros which are identified in the schematics as "Rise_Det" and "DFF_Plus". They are used to generate the 4 status bits that are related to the data transfer and format conversion (TXMT, TXUND, RXRDY, RXOVR). In the "Rise_Det" macro, a D-type flip flop is connected so that on the rising edge of the 200Hz "Latch Clk", a "high" signal is generated at the output. This signal is used to signal when data is ready to be read by the GT (i.e. RXRDY) or when the DIB is ready for the next data to be sent by the PL (i.e. TXMT). The "high" signal is cleared when the GT or PL subsequently reads or transfers, as appropriate, the next data word. The "DFF_Plus" macro follows the "Rise_Det" and consists of another D-type flip flop used to latch the output of the "Rise_Det" macro. The same 200Hz "Latch Clk" is used as the clock signal. If the output of the "Rise_Det" macro has not been cleared before the rising edge of the clock signal, then the output of the "DFF_Plus" macro will latch a "high" signal to its output. The output of the "DFF_Plus" is used to signal when data has been overwritten because of an overflow at the GT simulator subsystem (i.e. RXOVR) or when data has not been made available by the PL simulator subsystem thereby causing an underflow condition (i.e. TXUND). Schematics of the represented circuit for the "Rise_Det" and "DFF_Plus" are given in Fig. A14 and A15 of Appendix A.

3.4 Debug Latches

There are three 16-bit debug latches on the DIB which can be used as a debugging tool. Each 16-bit latch is composed of two octal D-flip-flops. The configuration of the latches is such that there is a one-to-one correspondence of the DSPLINK data lines to the debug latch data lines. All three debug latches can be accessed at any time by the user. The debug latches can be read by probing the

integrated circuit. In addition, debug latch #1 can also be read directly from a 26-pin connector mounted at one end of the DIB. Details of the 26-pin connector are included in Section 2.2 of Appendix A.

3.5 External Interfaces

There are four external interfaces on the DIB. The first is a 96-pin connector which is connected to the DSP backplane developed for the uplink synchronization experiments and provides the interface between the DSP and the DIB. The DSP backplane includes the DSPLINK expansion interface of Spectrum Signal Processing Inc. DSP boards. The DSPLINK interface [4] is a high-speed, bidirectional bus that allows data transfer with the DSP, thereby avoiding the PC bus bottleneck. The DSPLINK supports 16-bit data transfers. The signals which make up the DSPLINK interface and are used for the DIB are described in Section 2.4. The DIB can be driven by any of the Spectrum Signal Processing Inc. DSP boards which support the DSPLINK interface. Two 9-pin RS232 connectors are also present on the DIB. One RS232 connector is used for the downlink synchronization reference serial link which carries the FR0 pulse. The other RS232 connector is for the data source or sink connection. A 26-pin connector is also available and provides off-board access to debug latch #1. The four external interfaces are described in Appendix A.

4. Testing

A description of the testing done on the DIB is given in the following section. The purpose of testing is to verify the three functions of the board: data conversion/transfer; transmission and detection of the Frame Zero pulse; and use of debug latches. The setup used for testing is illustrated in Fig. 7. For testing purposes only, the clock required for the operation of the DIB is provided by a function generator.

4.1 Test Setup

Upon initial completion of the DIB, all the functions of the DIB were verified for proper operation. Fig. 7 illustrates the test setup for the DIB. As the DIB implements both the functions for the GT and the PL simulator subsystems, a single DIB is set up in a loop back configuration so that data flowing in both directions can be tested.

The external clock was generated by the function generator shown in Fig. 7. The Xantrax power supply was used for DC power necessary for the board to operate. The power supply is used only for the test setup as the DIB is installed in a DSPLINK backplane which provides power for the DIB during the uplink synchronization experiments. The data error analyser was used as both the data source and sink to verify the data flow from the source to the destination. The FR0_out and FR0_in lines are connected together so that when a FR0 pulse is transmitted, it is immediately reflected on the status register of the DIB. The test program used can be found in Appendix C. A description of the tests carried out for each of the functions is included in the following paragraphs.

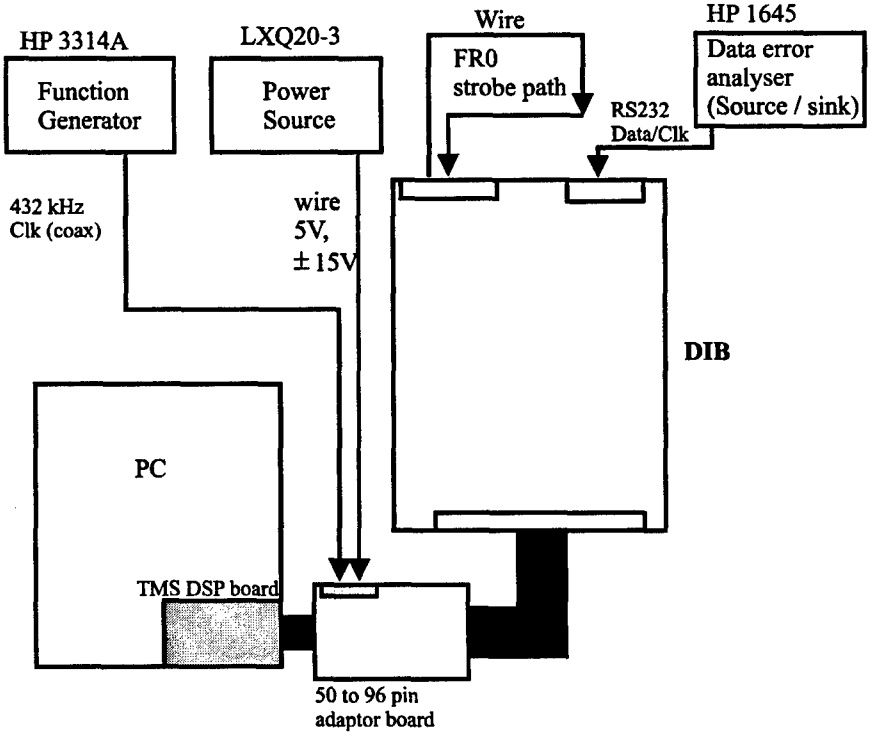

Fig. 7. Test setup diagram

4.1.1 Data Conversion/transfer test

The purpose of the test is to verify that the onboard (DIB) data conversion and transfer are being performed successfully.

The raw data came from the data error analyser (HP1645) and is sent through the DIB where it was transformed from a single bit stream into consecutive 12-bit words and sent to the DSP board installed inside a PC (simulated subsystem). Once on the DSP board, the data was written back to the DIB and reformatted from 12-bit words to a single bit stream. The data was then forwarded to the data error analyser. More information on data operation can be found in section 2.3.1.

To verify that the data has successfully been received, transmitted and transformed, the data error analyser was monitored during the testing. The testing process was repeated several times to ensure reliability. The indicator on the data error analyser read "0" meaning that the conversion and transfer portion of the DIB was working properly. Therefore, the testing of this portion was successful.

The listing of the test program for the DIB data conversion and transfer can be found in Appendix C of this document.

4.1.2 Tx/Rx Frame Zero test

The purpose of this test was to verify that the transmission and reception of the Frame Zero pulse were being performed successfully.

The Frame Zero pulse was generated by the TMS DSP board using the DIB command register causing a pulse on the J5 connector. For testing purpose, a cable was connected between the output of the Frame Zero port and its input, creating a loop back of the Frame Zero pulse. Once the pulse is looped back on the DIB, it can be read by the DSP from the DIB status register. Therefore, both the reception and the transmission of the Frame Zero pulse are tested simultaneously. To verify that the Frame Zero pulse was successfully transmitted and received, a logic analyser was also used to monitor the Frame Zero pulse port. The reader is referred to Section 2.3.2 for more information on the timing of the Frame Zero pulse.

The listing of the test program for the DIB Frame Zero pulse can be found in Appendix C of this document.

4.1.3 Debug latch test

The purpose of the test is to verify that the onboard 16-bit latches work properly.

The testing of the debug latches consists of simply writing a user-defined 16-bit word onto debug latch #1 and monitoring the data lines of the latch via the external connector. In addition, the debug latch operation was tested to ensure it did not affect the other functions of the DIB. This was verified to be true. The test was performed and was successful for all three debug latches. The reader is referred to section 2.3.3 for more information on the debug latches.

The listing of the test program for the DIB data conversion and transfer can be found in Appendix C of this document.

5. Conclusion

A multipurpose DIB was designed and implemented for upcoming uplink synchronization trials at DREO. The DIB can be used in both the GT and PL simulator subsystems. The DIB formats a serial bit stream of data from an HP1645 data error analyser into consecutive 12-bit words to be transferred to the GT processor for data transmission. Prior to formatting the bit stream, the DIB converts the original RS232 level signal to TTL levels required for the DIB circuitry. The reverse operation is also performed by the DIB for the PL processor side. The DIB also provides the interface between the GT and PL simulator subsystems and the downlink synchronization reference serial link. The link is used to carry a reference of the PL system clock, called the FR0 pulse, that is generated by the PL. The FR0 pulse is used by the GT as a starting point for uplink synchronization. Again, this interface includes the TTL/RS232 conversion of the reference signal. The DIB also provides three 16-bit debug latches for the user to use as a debug tool. Data lines from one of the three latches have been brought out to a connector for easier access.

The DIB was implemented and fabricated on a custom PCB. Flexibility was a key aspect of the DIB design. In addition to designing both the GT and PL requirements for data formatting and conversion on the same board, an EPLD was used to minimize the number of ICs on the board. Substituting the various circuits by an EPLD meant rapid development of the hardware as well as easy customization of the hardware interface. The EPLD is currently programmed to 80% of its capacity. The circuits incorporated into the EPLD include the interface between the DSP and the DIB, the data formatting operation, clock signal generation, and command and status register operations.

The DIB is driven by either a GT or PL processor. The processors for the uplink synchronization trials are implemented on Spectrum DSP boards and have a DSPLINK interface. The DSPLINK interface was implemented on a backplane which contains the DIB as well as other custom interface boards used in the experiments. The DIB is also connected to a data device (HP1645) as well as a downlink synchronization reference serial link.

A user's guide is provided in an Appendix B which describes the proper configuration and operation of the DIB. The DIB was tested and all the functions were verified to work properly. Appendix C includes the program written for the Spectrum TMS320C25 DSP board to independently test the DIB.

References

1. Addison, R., Seed, W., "Implementation of an EHF Frequency-Hopping Simulator", DREO Report 1279, December 1995.

2. Lambert, J.D., "DREO/CRC Joint Data Link Standard for Low Data Rate Service to EHF Ground Terminal Payload Simulators", DREO Report 1069, February 1991.

3. Addison, R., "Modified Hopping Synthesizer Controller", DREO Report 1304, December 1996.

4. TMS320C25 Processor Board User's Manual, Spectrum Signal Processing, Inc. Version 2, September 1988.

5. The TTL Data Book, Texas Instruments, Volume 2, 1985.

6. MAX+PLUS, Programmable Logic Development System, ALTERA Corporation, 1995.

Appendix A: Hardware & Firmware Details

1. General

This appendix contains the schematics, component list and layout of the DIB. The custom macros of the EPLD for the DIB functions can also be found in this appendix. The external interface connectors of the DIB are also described.

2. DIB

In this section of Appendix A, a graphic description of the circuitry of the DIB, its hardware components and its layout description can be found.

2.1 Schematic

The schematic for the multipurpose DIB is given in Fig. A1

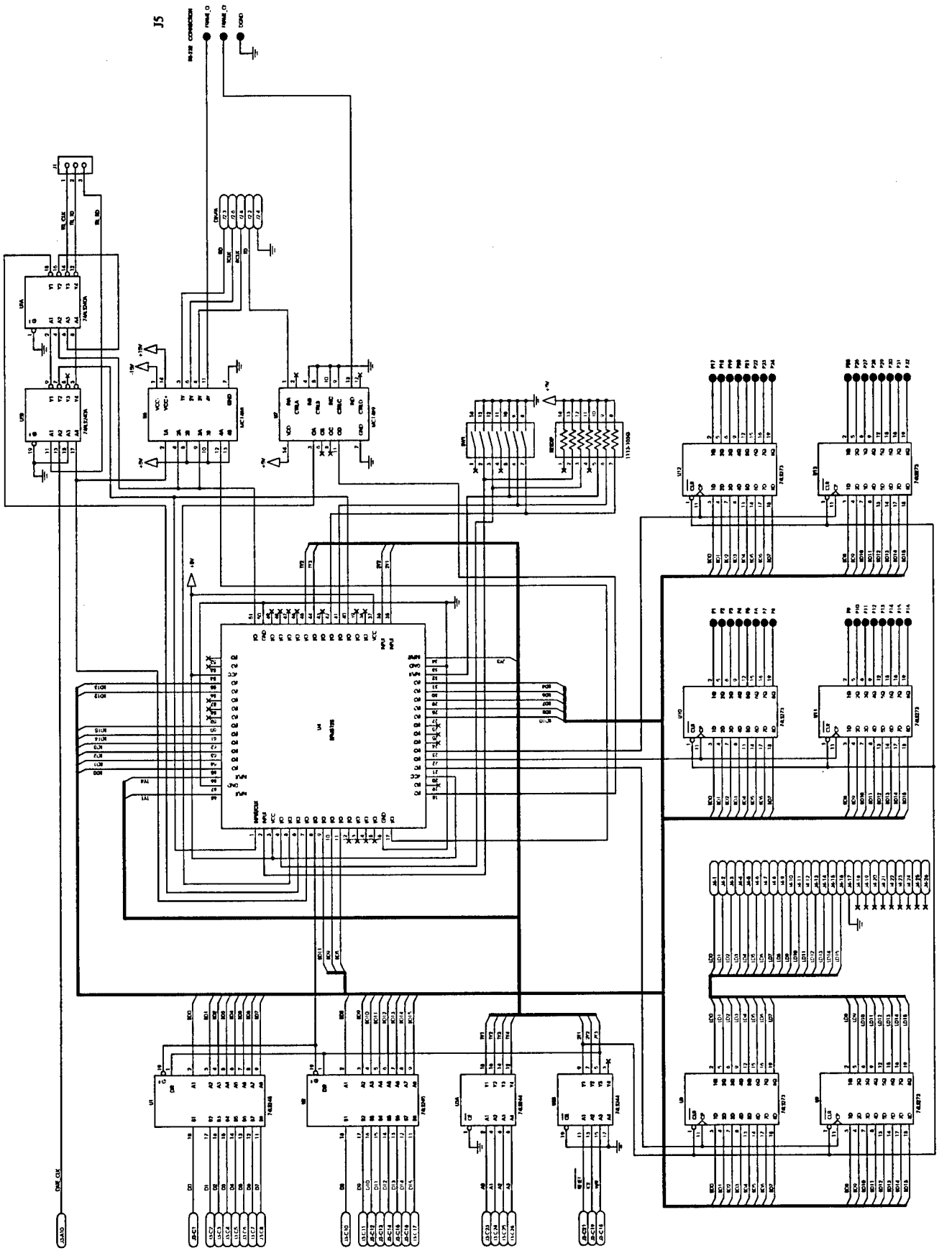

Fig. A1 Data interface board schematic

2.2 Layout Description

The printed circuit board layout of the DIB is shown in Fig. A2. All parts are standard off-the-shelf. The EPLD is the only component that will need to be modified (programmed). This EPLD contains most of the interface logic for the address decoder, data formatting, the clock signal generation, command and status register operations. Fig. A2 reveals the position of the six 8-bit latches used to form the three 16-bit debug latches which are used for debugging purposes. Line drivers used to convert TTL signal to RS232 signal and vice versa are also shown. The buffers are used to isolate the signal for DSPLINK lines coming from DSPLINK backplane. There is an unused IC socket on the board (Spare) to allow an TTL IC to be implemented for debugging purposes or for future expansion.

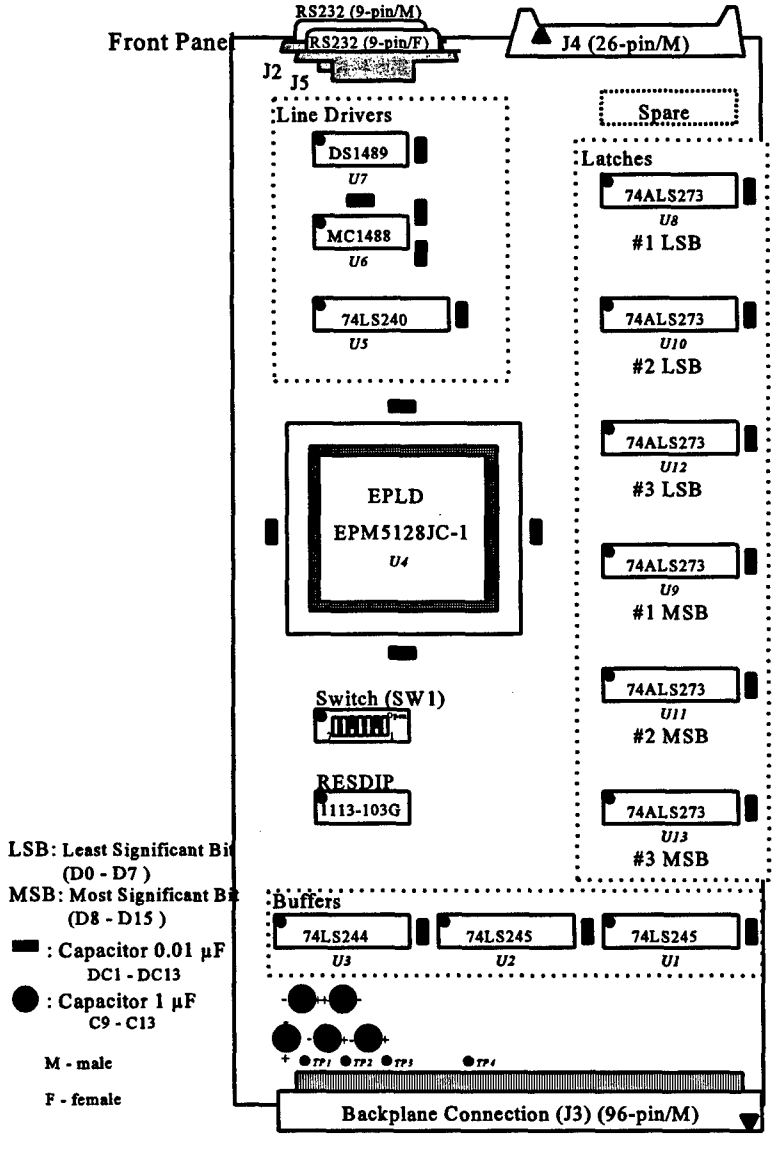

Fig. A2. Multipurpose DIB printed circuit board layout

2.3 External Interface Connectors

2.3.1 Front Panel Connectors

The front panel layout as seen when the DIB is installed in the DSPLINK backplane is shown in Fig. A3 and consists of 3 connectors J2, J4 and J5 which are described below. The position of the DIB interface connectors shown in Fig. A3 correspond exactly to its associated connector on the printed circuit board. Fig. A4 to A6 show the pinout of those three different connectors. Furthermore, Tables A1 and A2 identify the signals for connectors J2 and J5 respectively

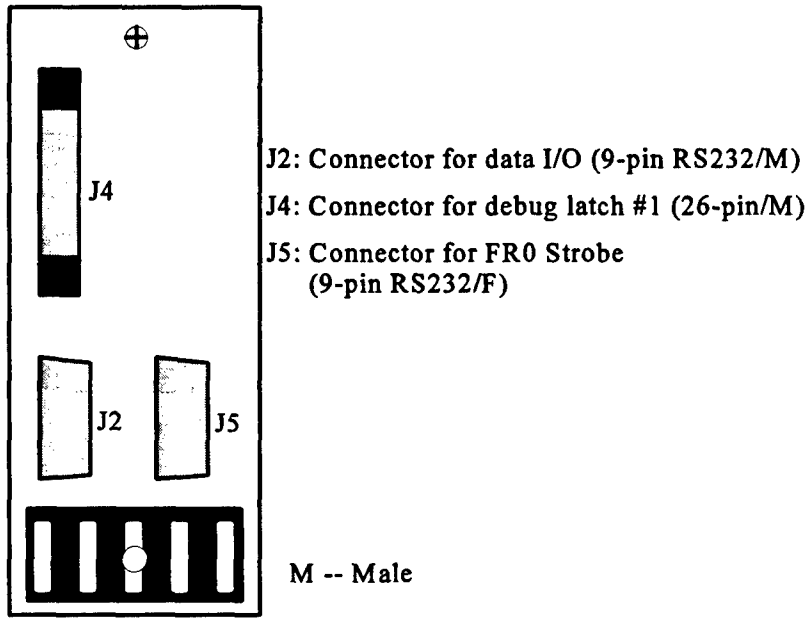

Fig. A3. DIB front panel layout

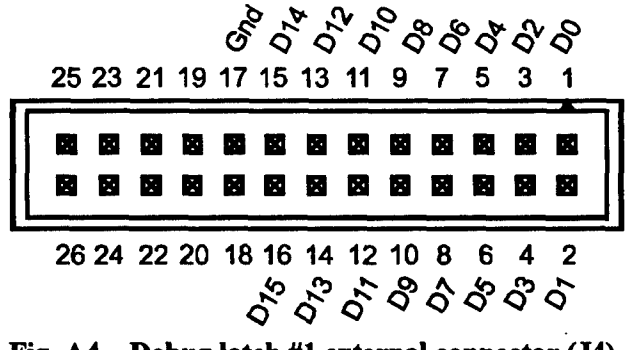

Fig. A4. Debug latch #1 external connector (J4)

D0-D15	Sixteen (16) TTL data lines for transferring data from the debug latch #1 to a monitoring device.
Gnd	Digital ground.
Pins 18-26	Not used

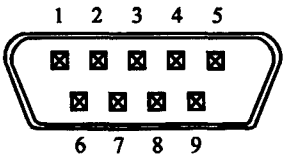

Fig. A5. Data source/sink external connector (J2) male

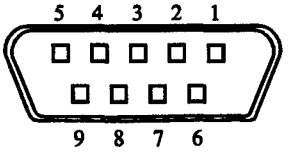

Fig. A6. FR0 pulse external connector (J5) female

PIN	Description
1	not used
2	Transmit data
3	Read data
4	Ground
5	not used
6	Transmit clock
7	not used
8	Read clock
9	not used

Table A1. Data source/sink external connector (J2) pinout description

PIN	Description
1	not used
2	FR0 in
3	FR0 out
4	not used
5	Ground
6	not used
7	not used
8	not used
9	not used

Table A2. FR0 pulse external connector (J5) pinout description

2.3.2 Backplane Connector

On the side opposite the front panel is a standard 96-pin connector (J3), connecting into the backplane which contains the DSPLINK interface defined in row C from Table A3. The DIB makes use of the DSPLINK interface (row C) and the data clock. Descriptions of the pinout configuration for each of the pins are included below.

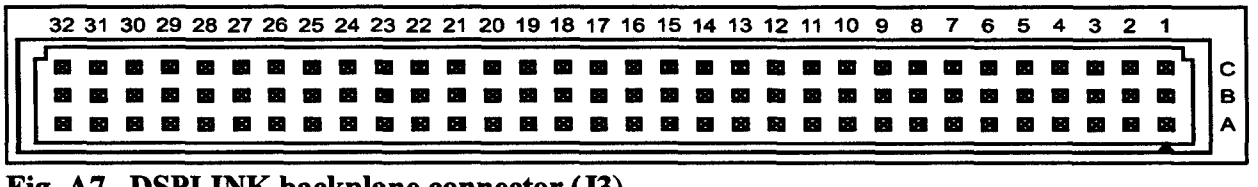

Fig. A7. DSPLINK backplane connector (J3)

PIN	ROW A	ROW B	ROW C
1	HOP CLK		D0
2	NHOP CLK		D1
3			D2
4		RESERVED	D3
5		RESERVED	D4
6		RESERVED	D5
7		RESERVED	D6
8		RESERVED	D7
9	GND	RESERVED	GND
10	DATA CLK	RESERVED	D8
11	GND	RESERVED	D9
12			D10
13			D11
14			D12
15	GND		D13
16			D14
17	GND		D15
18			NWR
19	GND		NIOE
20	RESERVED	GND	NINT0
21	RESERVED		RESET
22	RESERVED		CLK/2
23			A0
24		GND	A1
25			A2
26			A3
27			FLAG IN
28		-5V ANALOG	FLAG OUT
29		+5V ANALOG	
30		A GND	
31	-15V	+5V STBY	+15V
32	+5V	+5V	+5V

Table A3. DSPLINK backplane connector (J3) pinout configuration and description

D0-D15	Sixteen bi-directional TTL data lines of DSPLINK
GND	Digital ground
NWR	DSPLINK read/not write line originating from the DSP to signal the direction of data transfer. The "not write" nomenclature denotes an active-low signal for the WRITE signal. The direction is determined from the point of view of the DSP (i.e. a WRITE refers to data being transferred from the DSP to the DIB.
NIOE	An active-low, input/output enable signal indicating an access on the DSPLINK originating from the DSP.
NINT0	A negative-edge triggered, or active-low interrupt signal on DSPLINK. This signal is not used by the DIB.
RESET	DSPLINK reset line. This signal is active-low.
CLK/2	General purpose clock signal on DSPLINK originating from the DSP. This signal is not used by the DIB.
A0-A3	Four buffered TTL address lines of DSPLINK.
FLAGIN	General purpose input line on DSPLINK readable by the DSP. This signal is not used by the DIB.
FLAGOUT	General purpose output line on DSPLINK writeable by the DSP.
15V	15 volts power supply.
-15V	-15 volts power supply
5V	5 volts power supply
-5V ANALOG	-5 volts analog power supply
5V ANALOG	5 volts analog power supply
AGND	Analog ground
5V STDBY	5 volts standby power supply. This signal is not used by the DIB.
HOP CLK	Hop clock signal provided by another source. This pin is not used by the DIB.
NHOP CLK	Inverse hop clock signal. This pin is not used by the DIB.
DATA CLK	Data clock signal provided by another source to the DSPLINK backplane.
RESERVED	Reserved lines for the DSPLINK backplane.

2.4 Key Component List

A list of the components which were used for the DIB are provided in Tables A4, A5, and A6. The component labels listed in each of the tables correspond to the labels used on the printed circuit schematic in Fig.A1.

2.4.1 Integrated Circuits

Board Component	**Type**	Name
U1, U2	74LS245	Octal bus transceiver
U3	74LS244	Octal line driver
U4	EPM5128JC-1	Altera 128 macro-EPLD, 30 ns
U5	74ALS240	Octal inverting line driver
U6	MC1488	Quad line driver
U7	MC1489	Quad line receiver
U8,U9,U10,U11,U12,U13	74LS273	Octal D flip-flop with clear

Table A4. Integrated circuits

2.4.2 Discrete Components

Component	Type
DC1 - DC13	0.01 μF
RESDIP	Resistor DIP, 7 wide, 20 kΩ
C9 - C13	10 μF Tantalum
TP1 - TP4	Test point post TP1: monitors line B31 on J3 (+5V STBY) TP2: monitors line A22 on J3 (free) TP3: monitors line A27 on J3 (free) TP4: monitors line A29 on J3 (free)
Switch (SW1)	DIP switch, single throw, 7 wide
Spare	Unused socket position available for expansion

Table A5. Discrete components

2.4.3 Other Components

Component	Type
J1	Not used
J2	9 pin connector, RS232 type, male
J3	96 pin connector, triple row, 0.1" spacing, male
J4	26 pin connector, double row, 0.1" spacing, male
J5	9 pin connector, RS232 type, female

Table A6. Other components

3. EPLD Description

3.1 General

This section graphically describes the firmware used in the EPLD Altera EPM5128JC-1. The EPLD was configured with the MAX+plusII compiler installed in a PC and then loaded on the EPLD chip using an interface board and a programming unit.

3.2 EPLD Schematics

The EPLD, Altera 5128, is an erasable programmable logic device that replaces a collection of TTL logic and reduces significantly the number of ICs. The EPLD is programmed using the MAX II + Plus software from Altera. At present, the macros have used about 80% of the EPLD space. The EPLD schematic shown in Fig. A8 includes several custom macros in order to simplify the total logic circuit of the EPLD. Each macro is identified by the box drawn in a dotted line. The custom macros are listed in Table A7. Each of the macros are described in the following sections.

Macro Name	Description of Macro	Reference Fig.
Clocks	Clock generation	Fig. A9
Address	Address decoder	Fig. A10
Comd_reg	Command register	Fig. A11
Tx_Shift	Transmit Shift - conversion from 12-bit word to a 1-bit data stream	Fig. A12
Rx_Shift	Receive Shift - conversion from a 1-bit data stream to a 12-bit word	Fig. A13
Rise_Det	Rise Detector - status bit generation	Fig. A14
DFF_Plus	D-flip-flop Plus - status bit generation	Fig. A15

Table A7. List of EPLD Macros for the DIB Functions

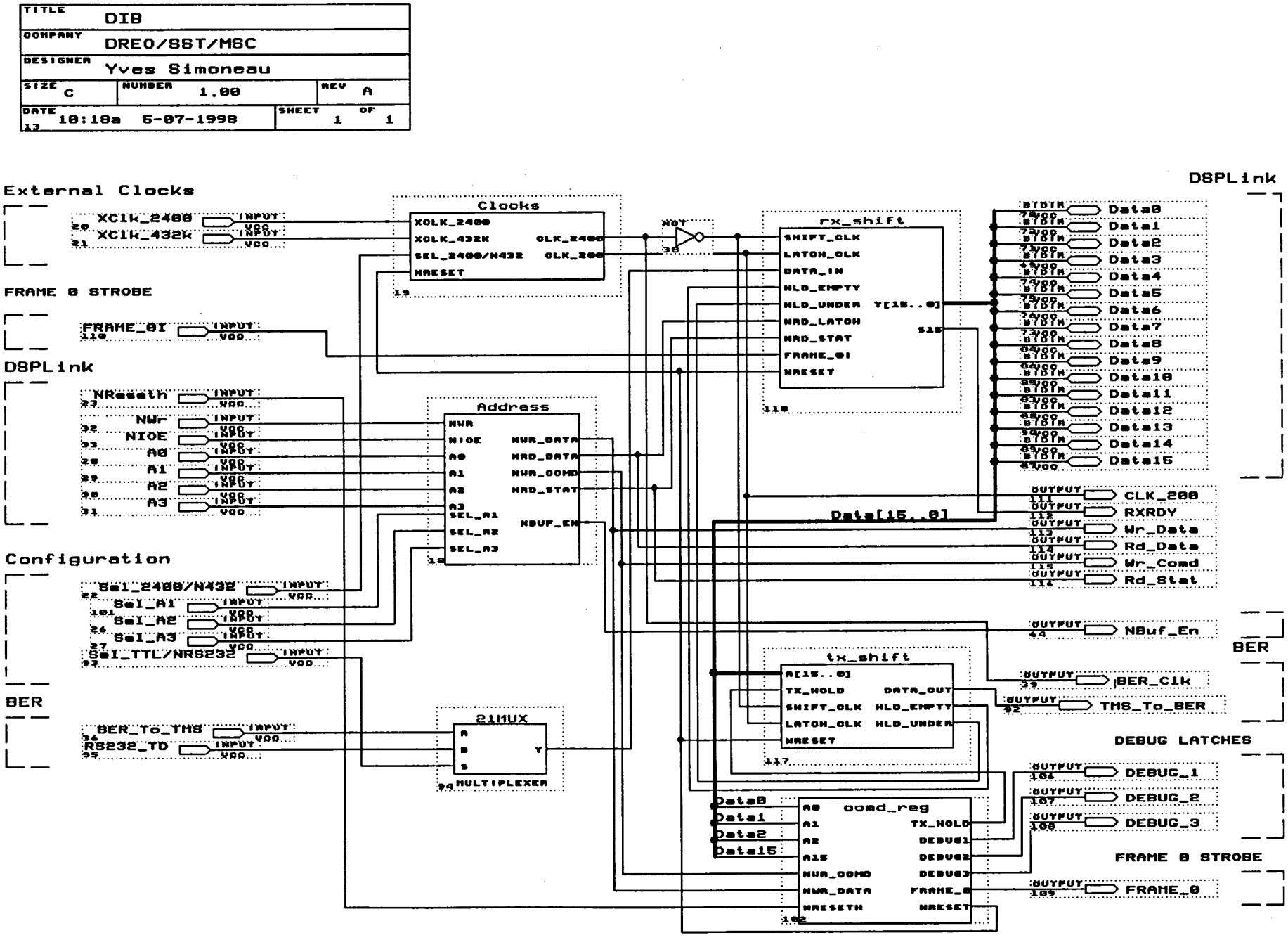

Fig. A8. EPLD diagram

3.2.1 Clock Generator Macro

The clock generation macro shown in Fig. A9 produces two clock signals which are required for the data transformation process. The two clock signals required are a 2.4 kHz data clock signal and a 200 Hz clock signal. The 2.4 kHz signal is used to shift the bit stream to and from the data device. The 200 Hz signal, derived from the 2.4 kHz data clock signal is used for moving 12 bits of data.

There are two ways in which the two clock signals may be generated. The first method is a straightforward one where the 2.4 kHz data clock signal is directly provided by the user via the external clock input on the DIB. This is represented by the XClk_2400 signal from Fig. A9. The 200 Hz signal is subsequently produced by dividing the 2.4 kHz signal using a divide-by-12 circuit (7492). The divide-by-12 process actually occurs in two stages: a divide-by-2, followed by a divide-by-6 stage as indicated by the feedback from the QA output to the CLKB input of the 7492 device. The resulting 200 Hz signal, labelled Clk_200, is produced at the QD output of the divide-by-12 circuit.

The second method for producing the required clock signals consists of the user providing a 432 kHz clock signal via the external clock input on the DIB. The 432 kHz input signal option originated from requirements for the DIB used in the downlink synchronization experiments. This option was retained to allow flexibility in expanding the number of clock signals that could be generated. The clock generation macro again divides the 432 kHz input clock signal to produce the required signals. The generation of the 2.4 kHz signal is achieved by dividing the input clock signal by 180. This is implemented in two stages: a divide-by-90 and a divide-by-2 stage. The divide-by-90 function is effected by an 8-bit up/down counter which is loaded with an initial value of 90. In this implementation, the counter is configured to count down. When the counter reaches "0", it is reset and reloaded with a value of 90 and the process is repeated. The output of the down counter is a 4.8 kHz (432 kHz/90) signal which is then divided by two via a toggle flip-flop (TFF in Fig. A9) to produce the needed 2.4 kHz. As in the first method, the required 200 Hz signal is generated by dividing the 2.4 kHz by 12 as described in the previous paragraph.

A multiplexor is used to allow the user to select which method is used by the EPLD to produce the required clock signals. The selection is determined by the Sel_2400/N432 line shown in Fig. A9. When the line is "high", the 2400 Hz clock is used. Conversely, when the Sel_2400/N432 line is "low", the option to use a 432 kHz signal as the input clock signal is selected. The Sel_2400/N432 line is set by a dip switch on the board which is described in Appendix B.

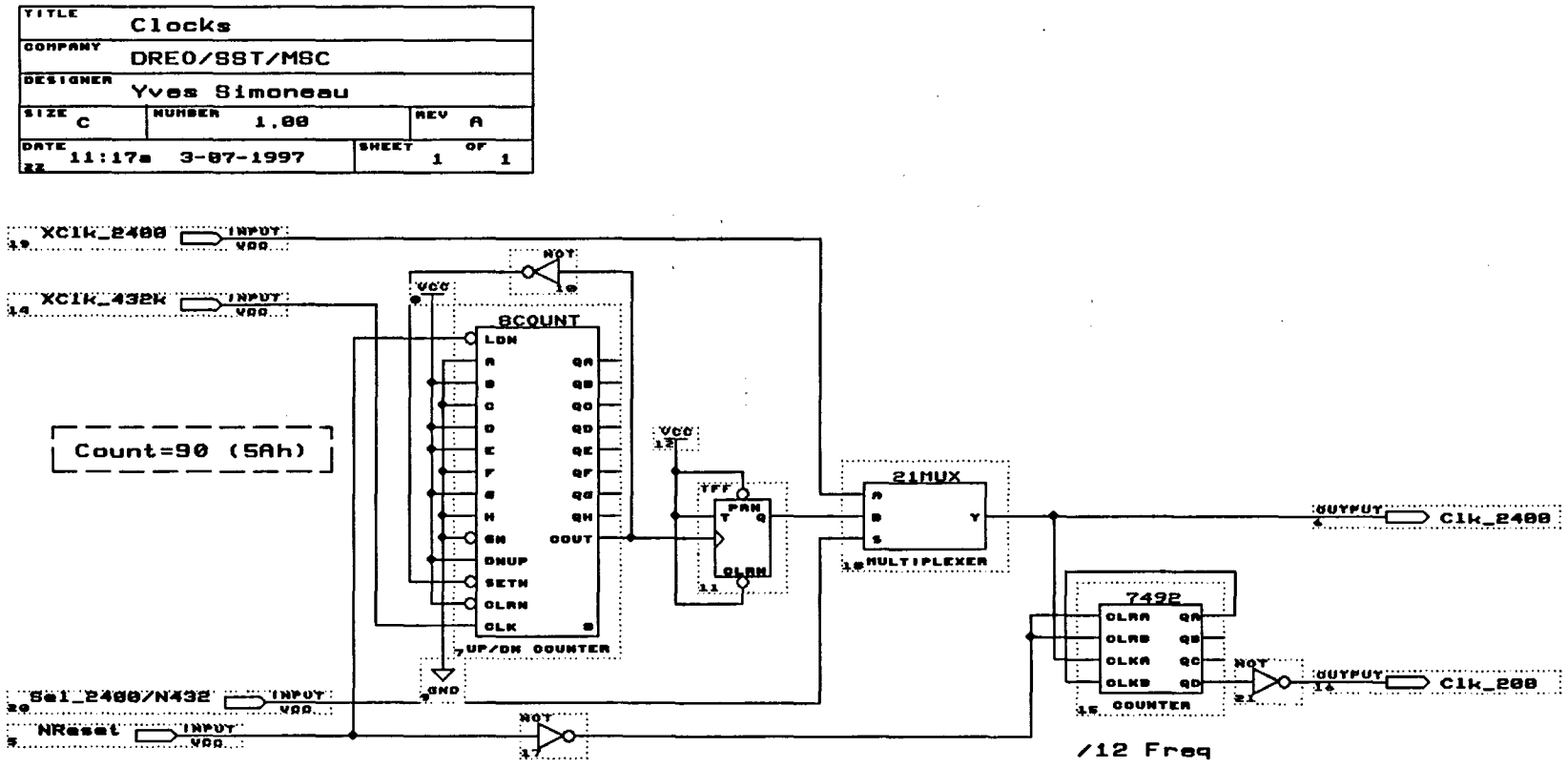

Fig. A9. Clock Generator Macro

3.2.2 Address Decoder Macro

The address decoder macro is illustrated in Fig. A10. The address decoder generates the I/O port select lines for the EPLD. The I/O port select lines are labelled NWr_Data, NRd_Data, NWr_Comd, and NRd_Stat. These lines correspond to Write Data, Read Data, Write Command, and Read Status respectively described in Section 2.4.2 in the main document. The four I/O port select lines are represented by four outputs of a 3-to-8 decoder. The select lines are generated by decoding the NWR (read/write) line and the A0 address line both from the DSPLINK interface. A third input is tied to ground. A 3-to-8 decoder was used rather than a 2-to-4 decoder to facilitate future expansion.

With the 3-to-8 decoder, two enable signals are available for the address decoder macro to ensure that a valid I/O access is taking place and that the DIB is being addressed. The first enable signal is provided by the NIOE signal from the DSPLINK interface. The NIOE is an active-low signal which is generated on DSPLINK when the DSP initiates an I/O access to an external board. The other enable signal is provided by the output of a comparator which is included in the address decoder macro. The comparator is used to verify the base address of the I/O operation. The base addressing scheme ensures that multiple interface boards using the DSPLINK do not use conflicting addresses and is described in Section 2.4.1 of the main document . The dip switch settings of the DIB for selecting the base address are compared with the A1, A2, and A3 address lines of the I/O operation. A "low" signal is produced at the comparator output only if the base address used by the DSP matches the dip switch settings.

In order to isolate the data lines on the DIB from the DSPLINK data lines when the DIB is not being addressed, two 8-bit buffers, 74LS245, are used on the DIB. The 74LS245 buffers include an enable line which can be used to disable the device, effectively isolating the data lines. An additional signal, NBuf_En, is generated in the address decoder macro to perform this function. The NBuf_En signal is an active-low signal which is generated by a logic OR of the NIOE signal and the output of the comparator. In other words, a "low" signal is produced to enable the buffers when both a valid I/O operation is occurring (NIOE is "low") and the board is selected (output of the comparator is "low").

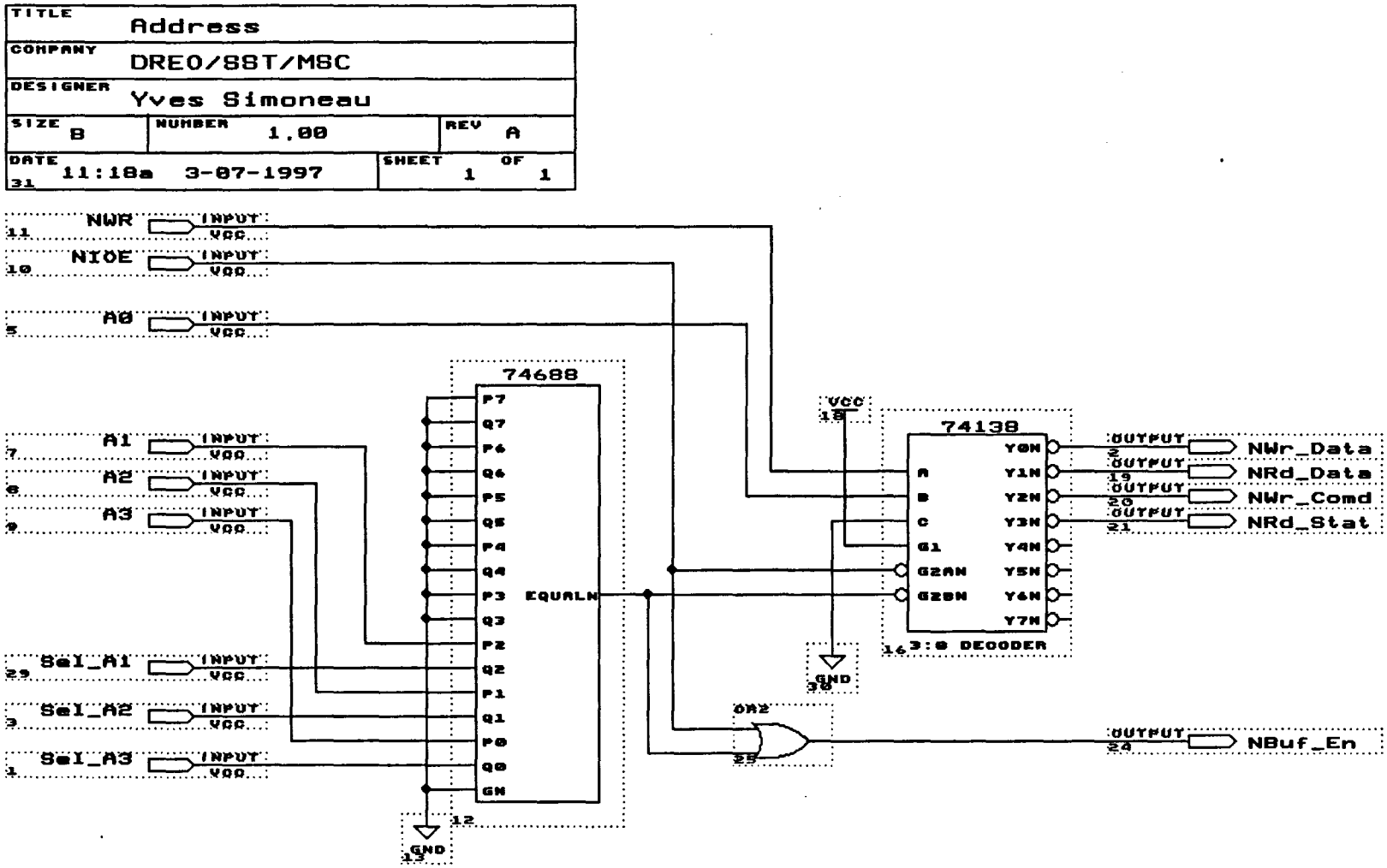

Fig. A10 Address Decoder Macro

3.2.3 Command Register Macro

The Comd_Reg macro is illustrated in Fig. A11. The Comd_Reg macro provides the necessary logic to allow the user to select whether data is to be transferred to the data buffer or to one of three debug latches on the DIB. It also allows the user to issue a software reset of the DIB or to send a FR0 pulse. These operations are described in Section 2.4.2 of the main document. When a Write Command operation is performed by the DSP, the NWR_COMD select line generated by address decoder macro causes data lines A[0], A[1], and A[2] to be stored in a latch and A[15] to be stored on a flip flop. A[0], A[1], A[2], and A[15] correspond to the data bits (D0, D1, D2, and D15 respectively) on the command register described in Section 2.4.2. Hence, A[0] and A[1] are the data destination selection lines, A[2] is the FR0_out line, and A[15] is the software reset line. When the FR0_out line is latched, the output of the latch, labelled FRAME_0 in Fig. A.11, is transferred directly to the TTL/RS232 conversion circuitry. The software reset line is combined with the DSPLINK reset line, NRESETH, to generate the board reset line, NRESET. The inclusion of the DSPLINK reset line allows the DIB to be reset using either a hardware or software reset.

As the DIB allows the user to transfer data to multiple locations on the DIB, a second level of decoding is required to produce the appropriate select line for transferring data to the DIB. The select lines, Tx_Hold, DEBUG_1, DEBUG_2, and DEBUG_3 are generated by decoding the latched A[0] and A[1] lines, labelled Q1 and Q2 in Fig. A9, using a 3-to-8 decoder. Again, a 3-to-8 decoder is used in lieu of a 2-to-4 decoder to allow additional select lines to be generated in the future if required. The generation of the appropriate select line takes place upon a Write Data operation whereby a "low" NWr_Data signal is produced from the address decoder macro to enable the 3-to-8 decoder. The actual data transfer occurs during the Write Data operation.

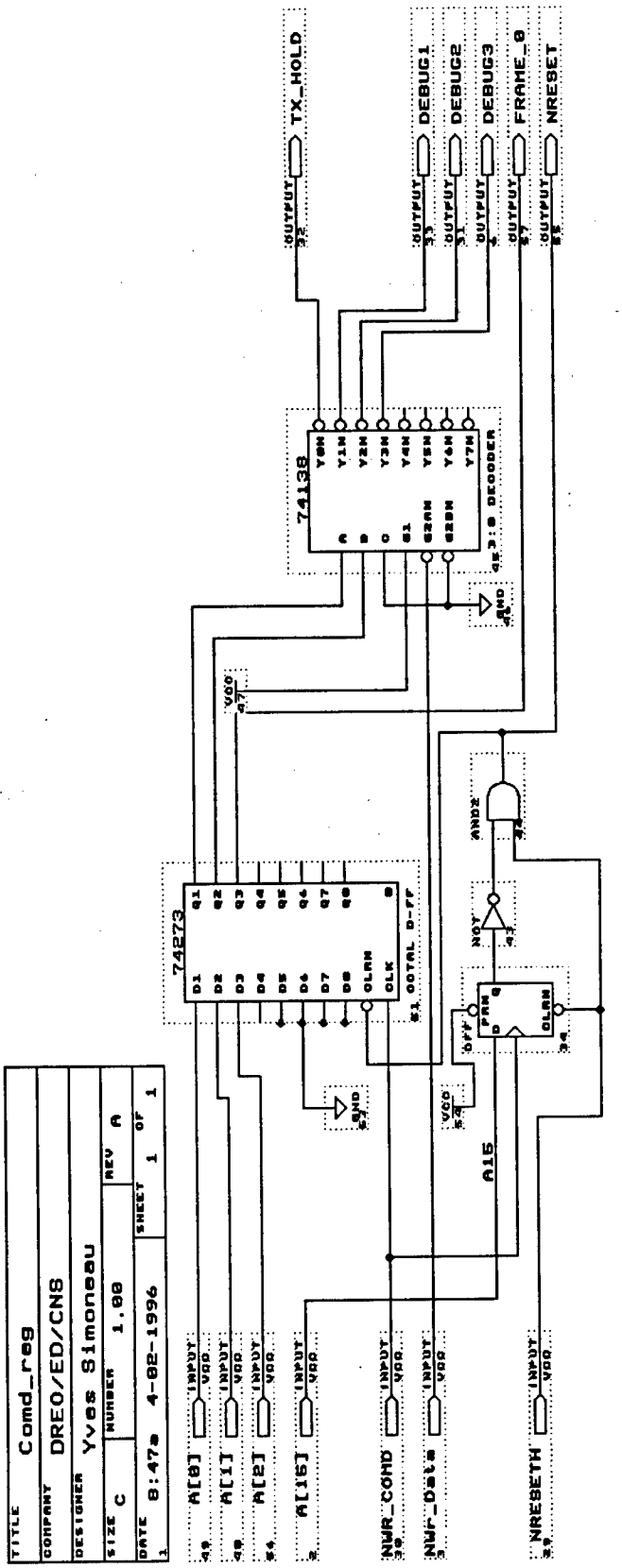

Fig. A11. Command Register Macro

3.2.4 Transmit Shift Macro

The purpose of the TX_Shift macro is to take a 12-bit word, transferred from the DSP, and convert it to a single bit stream that is to be forwarded to the data sink. The 12-bit word is located in the lower 12 bits of the 16-bit DSPLINK data interface. From Fig. A12, the 16 data lines from the DSPLINK interface, labelled A[15..0], lead to two 8-bit latches. Data on the lines are stored on the latches upon the rising edge of the Tx_Hold signal. The Tx_Hold signal is the I/O port select line generated by the secondary level of decoding of the command register bits D0 and D1 which is described in Section 4.2.3 of this appendix. The Tx_Hold signal is selected or active during a Write Data operation where the data buffer is selected as the destination of the data transfer (i.e. D0 and D1 of the command register are both zero). The latched data is transferred to two 8-bit shift registers which are connected in series. The data is "shifted" out one bit at a time on the Data_Out line using the Shift_Clk signal. For the uplink synchronization experiments, the Shift_Clk is the 2.4 kHz clock provided by the user. It is noted that the data lines are bit reversed between the data latches and the shift registers so that the least significant bit (LSB) is shifted out first. The shift process is controlled by the Latch_Clk signal which is generated by dividing the 2.4 kHz data clock by 12 yielding a 200 Hz signal (See Section 4.2.1 of this appendix). The Latch_Clk signal is used to generate a low signal to load the next 16-bit word from the data latches to the shift registers. By generating the load signal using the Latch_Clk, the shift process is effectively limited to 12 bits after which the next data is transferred to the shift registers.

There are also two status lines which are generated by the TX_Shift macro. They are the Hld_Empty and Hld_Under lines, which allow the user to monitor the data flow process of this macro. The two lines correspond to the TXMT and TXUND bits of the status register as described in Section 2.4.2 of the main document. The Hld_Empty signal is generated by a flip flop circuit defined under the Rise_Det macro which is described in Section 4.2.6 of this appendix. A "high" signal is produced on the Hld_Empty line on every rising edge of the Latch_Clk to signal that the 16-bits from the data latch have been transferred to the shift registers and that the data latches are ready to store the next 16 bits. The Hld_Empty signal is cleared only when the next 16-bits are transferred from the DSP, i.e. when a Tx_Hold signal is generated. If no data is transferred to the data latches by the next rising edge of the Latch_Clk, the "high" level Hld_Empty signal is latched onto the Hld_Under line indicating that an underflow condition has occurred.

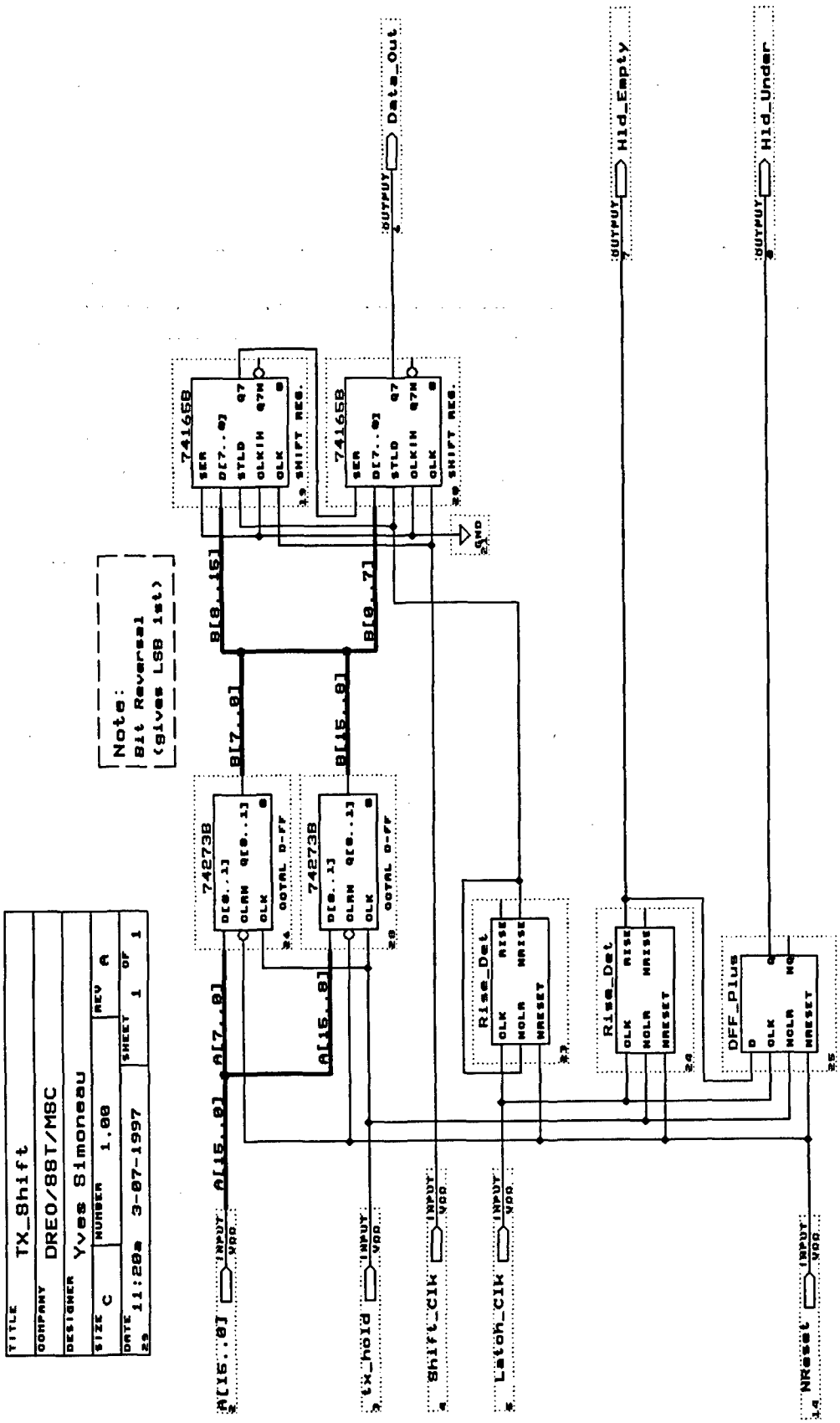

Fig. A12. Transmit Shift Macro

3.2.5 Receive Shift Macro

In the RX_Shift macro, there are two separate circuits which are implemented. The first is the transformation of a bit stream of data coming from a data source, the Data_In line in Fig. A13 to a 12-bit word. This process is basically the reverse of that described for the TX_Shift macro. The second circuit contains the logic required to form the status register which is read by the DSP.

For the data transformation, two 8-bit shift registers are connected in series as for the TX_Shift macro. A single bit stream of data (Data_In) is shifted into the shift registers using the 2.4 kHz Shift_Clk signal. When 12 bits have been shifted in, the Latch_Clk signal will trigger the 12 bits to be transferred in parallel to the internal latch with the 4 upper bits tied to ground. When a Read Data operation occurs, the multiplexor transfers the data lines from the internal latch to a 16-bit buffer where it can be read by the DSP.

The status register is defined by 5 bits as described in Section 2.4.2 in the main document. The lines labelled "S15" and "S14" are related to the data transformation process described in the paragraph above and correspond to status bits RXRDY and RXOVR. The two lines are generated in the same manner as the Hld_Empty and Hld_Under lines in the TX_Shift macro described earlier. For the RX_Shift macro, a "high" signal is generated at the output of the Rise_Det macro on every rising edge of the Latch_Clk. This "high" signal is the RXRDY signal on the status register which informs the DSP (user) that 12 bits are available to be read. The RXRDY signal is cleared when the data is read from the buffer. If the data is not read by the time the next 12 bits are available, a second flip flop (labelled DFF_Plus) produces a "high" Rx_Data_Overflow (or RXOVR on the status register) signal indicating an overflow condition has occurred. The third and fourth bits of the status register are the Hld_Empty and Hld_Under lines which are produced by the TX_Shift macro. The fifth status bit is the FR0 line from the separate serial link connecting the PL and the GT. The five status bits are presented to the other port of the multiplexor with the lower 10 bits set to "0". During a read status operation, where the NRd_Stat line becomes active ("low"), the status register bits are transferred through the multiplexor to the buffer (labelled, Tri_B) to be read by the DSP.

3.2.6 Rise Detector Macro and D flip-flop Macro

The Rise_Det and DFF_Plus macros, shown in Fig. A14 and A15 respectively, are used in the TX_Shift and RX_Shift macros to generate the status register bits. They are both based on a D-type flip flop and are cleared by an appropriate NClr or NReset board reset signal. The selection of the NClr signal is discussed in the TX_Shift and RX_Shift macros. The NReset signal is described in the Comd_Reg macro.

In the Rise_Det macro, the input of the D flip flop is tied to Vcc. Consequently, on the rising edge of the Clk signal, the Rise_Det macro produces a "high" signal at its positive output. The output is cleared (i.e. returned to "0") only when an active-low NCLR or NRESET is received. This macro is used to generate the two status bits TXMT and RXRDY as described in Section 4.2.4 and 4.2.5 above.

In the DFF_Plus macro, the rising edge of the Clk signal causes the input to be latched to the output of the flip-flop. Again, the output is cleared when an active-low NCLR or NRESET is received. This macro is used to generate the RXOVR and TXUND status bits as described in Sections 4.2.4 and 4.2.5 of this appendix.

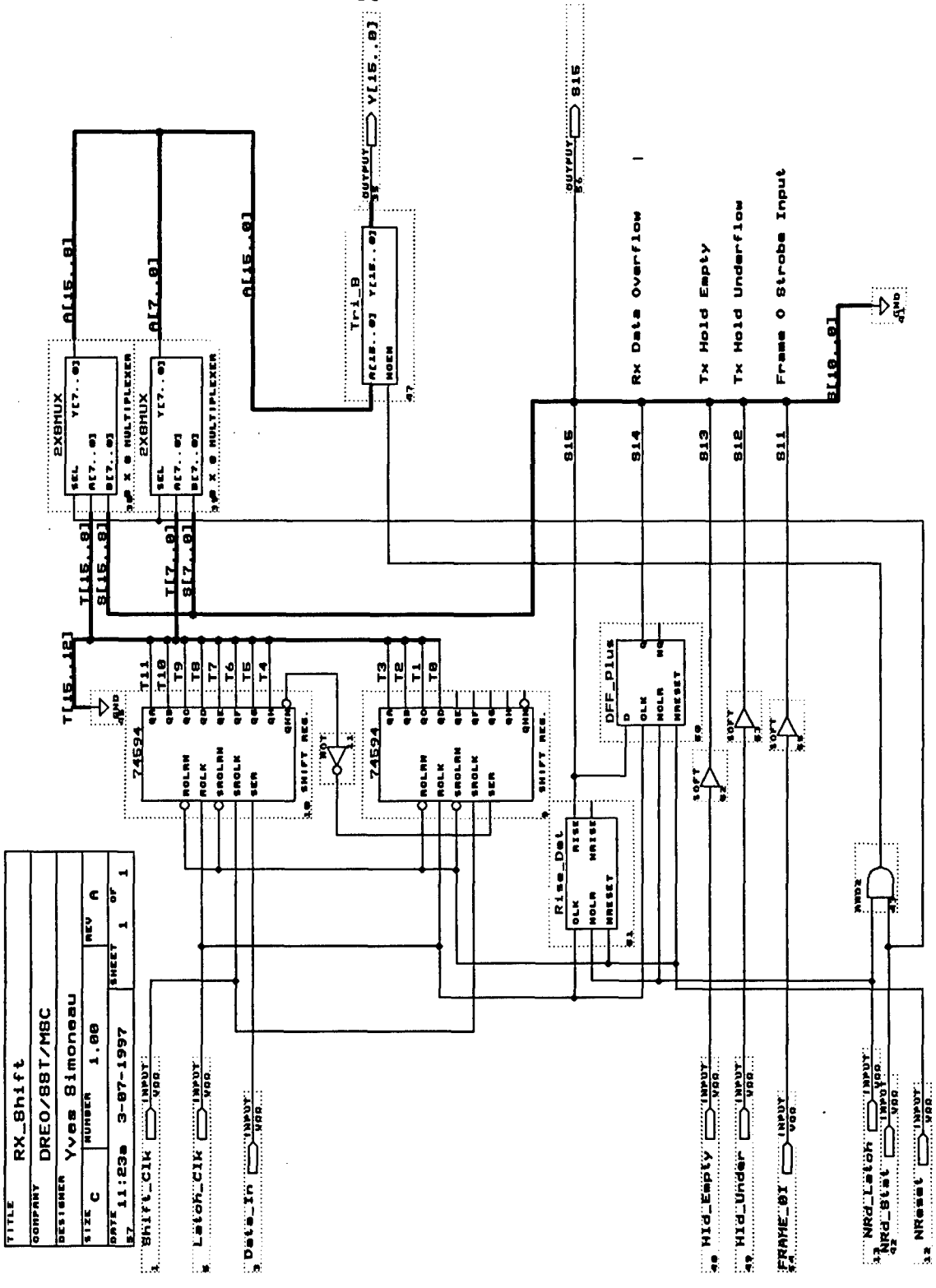

Fig. A13 Receive Shift Macro

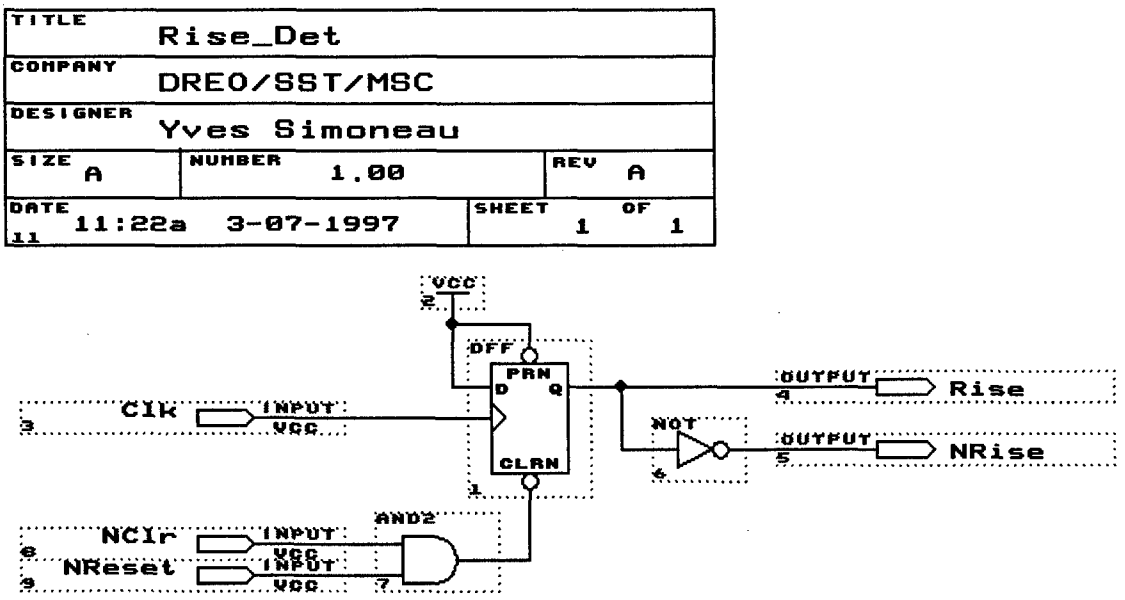

Fig. A14. Rise Dectector Macro

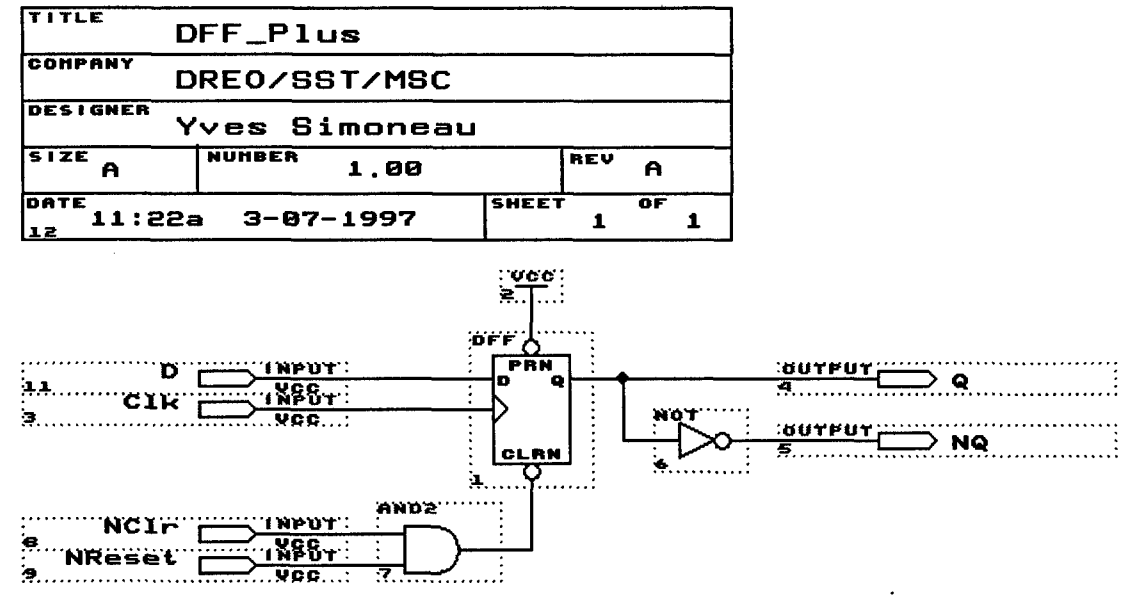

Fig. A15. D-Flip-Flop Macro

Appendix B: DIB User's Guide

1. General

This appendix covers the information required to install, configure and use the DIB. The latter consists of a description of the modes of operation and examples of working with the DIB. The examples given are: reading the status; giving a command; reading the data from a data source; sending the data to a data sink; resetting the DIB; writing to the debug latches; and transmitting/receiving the FR0 pulse. Sample code for each of these operations is included.

2. Installation

In order to facilitate the implementation of all the interface boards for the uplink synchronization experiments, a chassis which provides a DSPLINK interface and additional clock signals on a single backplane is used. The DIB is installed by sliding it onto the backplane, connecting its 96-pin connector to the rear of the backplane itself. The backplane is connected to the DSP by an extension card and a 50-pin ribbon cable. The extension card was fabricated to map the 96-pin backplane to the 50-pin DSPLINK interface on the DSP board. It is recommended that the cable from the backplane to the computer be no longer than 3 feet to minimize losses which would corrupt the signal.

Once the board is correctly installed onto the backplane, the connections to the front panel can take place. The data port, labelled as J2 in Fig.A3, should be connected by an RS232 cable to a designated data device. For the uplink synchronization experiments, the data device used is the Hewlett Packard 1645 data error analyser. The DIB will either send serial data through that port or receive it.

The FR0 port, labelled as J5 in Fig.A3, is connected to the FR0 port of another DIB via a separate RS232 cable to realize the downlink synchronization reference link. In the uplink synchronization experiments, the RS232 cable from the DIB located at the PL simulator subsystem is connected to one of the ports of a statistical multiplexer (stat mux). The stat mux is connected to a serial communication link setup between the transmit and receive locations at DREO/CRC. At the other end of the serial link, the connection is passed through another stat mux to the port connected to the FR0 port of the GT DIB.

A 26-pin connector is also provided to the user to access the data lines for debug latch #1. This facilitates the use of a logic analyser to observe the data stored on the debug latch. The data lines for debug latches #2 and #3 may be accessed by probing the chip.

3. Configuration

In the middle of the DIB, a 7-pin wide DIP switch is accessible (see Fig. A1 of Appendix A). The switch is used to configure the board and includes the selection of the base address of the DIB. The details of the DIP switch settings are given in Table B1.

Switch	Selects	Position	TTL Value	Value
SW1	Not Used	X	X	X
SW2	External Clock Selection	Closed/On Open/Off	0 1	432 kHz 2.4 kHz
SW3	I/O Format	Closed/On Open/Off	0 1	RS232 TTL (not used)
SW4	Not Used	X	X	X
SW5	Base Address selection/ Address Line #3 (MSB)	Closed/On Open/Off	0 1	Disabled Enabled
SW6	Base Address selection/ Address Line #2	Closed/On Open/Off	0 1	Disabled Enabled
SW7	Base Address selection/ Address Line #1 (LSB)	Closed/On Open/Off	0 1	Disabled Enabled

(X - don't care)

Table B1. Configuration DIP switch settings.

The switches, that are being used, are described in the following paragraphs.

External Clock Selection (SW2): The DIB can work with two different input external clock speeds. One being at 2.4 kHz and the other at 432 kHz. The default setting is at 2.4 kHz which is the rate required for the uplink synchronization trials.

I/O Format (SW3): The DIB is designed to be able to transmit and receive user data in two possible formats. The first possibility is an RS232-level signal and the other possibility is a TTL-level signal. There is currently no port setup for the TTL I/O signal on the DIB. Thus, the switch must be closed to select the RS232 signal option.

Address Lines 1,2,3 (SW 7,6,5): These three dip switches are used to assign a base address for the DIB. Eight possible base addresses for the DIB are provided by these dip switches. Address lines 1, 2, and 3 (A_1, A_2, A_3) are compared with address lines A1, A2, and A3 of DSPLINK to determine whether to enable the DIB for the DSPLINK I/O access. For example, if $A_1 = 1$, $A_2 = 0$, $A_3 = 1$, a base address of 10 is assigned to the DIB and along with address line A0 of DSPLINK, only I/O accesses through DSPLINK using addresses 10 and 11 would enable the DIB.

3.1 DIB Applications

For the uplink synchronization experiments, two DIBs are needed to establish an end-to-end data transmission link. Each DIB is used differently depending on whether it is located at the transmit or receive stations. On the GT transmit side, the DIB is used with the HP1645A as a data source generating the data to be transmitted while on the PL receive end, the HP1645 is connected to the DIB as a data sink. There are no physical settings or switches which need to be set to differentiate the two operations. The selection of the application is done automatically through the manner in which the DIB is addressed. The same is true for the FR0 pulse. The FR0 port of the DIB is used to relay a reference pulse from the PL to the GT. Transmitting a pulse through the FR0 pulse involves writing "1" and "0" to the appropriate bit in the command register. By the same token, receiving the FR0 pulse is achieved by reading the status register.

4. Operational Procedure

The following section describes in further detail the procedure or sequence of events required to operate the DIB correctly. The process of retrieving data from and transferring data to the DIB is explained. The procedure followed to receive and transmit the FR0 pulse, to address the debug latches and to give a reset to the DIB is also included. These procedures are shown in sample code written for a Spectrum Signal Processing Inc TMS320C25 DSP board.

4.1 General Operation

4.1.1 Reset

There are two different ways to reset this board. The DIB can be reset using either a hardware reset or a software reset. Generally a hardware reset takes place upon power up of the DSP board via the RESET line of the DSPLINK interface. During a hardware reset, all components of the board are reset or cleared. For the software reset, all components except the debug latches are cleared. Since the output of the debug latches was of interest only when a value was written to them, it was decided that a software reset capability was not required for the latches. A software reset is given by the user via the command register of the DIB as described in Section 2.4.2.

The software reset pulse for the DIB is an active-high signal. In order to generate a software reset pulse for the DIB the user must first write a "1" to bit 15 of the command register. The DIB circuitry will then generate the appropriate "high" or "low" signal for the various components on the board. After the reset occurs, the command register must be returned to its initial state (i.e. the reset must be cleared). The user must write a "0" to bit 15 of the command register to clear the reset line.

The following is sample code written for the TMS320c25 DSP board to perform software reset of the DIB:

```
BOARD          .set          0              ;base address of DIB = 0

COMMND         .set          BOARD+1        ;COMMAND port address

RESHI          .set          8000h          ;Set reset bit (D15) to "high" command
RESLO          .set          0h             ;Clear reset bit command

; Sample code begins here

               OUT           RESHI,COMMND   ;send a "1" to D15
               NOP
               NOP
               NOP
               OUT           RESLO,COMMND   ;send a "0" to D15,clear reset

; End of sample code
```

4.1.2 Selecting the destination for the data transferred to the DIB

The process of transferring data to either the data buffer or one of the debug latches involves two steps as described in Section 2.4.2. First the user must select the destination which is represented by D0 and D1 of the command register. To select the destination, D0 and D1 of the command register must be set to one of the values defined in Table B2. Thus, for normal operation of the DIB, bits D0 and D1 should always be set to '0'.

Command Register D1/D0 value	Destination selected
00	Data buffer
01	Debug Latch #1
10	Debug Latch #2
11	Debug Latch #3

Table B2. Settings for selecting destination of data transfer to DIB

The following sample code for the TMS320C25 DSP would be used to select debug latch #1 as the destination of the data transfer:

```
BOARD          .set          0                    ;base address of DIB = 0
COMMND         .set          BOARD+1              ;COMMAND port address
SEL_LAT1       .set          1                    ;select debug latch #1
; Sample code begins here
               OUT           SEL_LAT1,COMMND
; End of sample code
```

The procedure for transferring data after the destination is selected is described further in the next section.

4.2 Data Operation

4.2.1 Read Data Example

The following paragraphs describe the procedure for reading data from the data buffer of the DIB. It is assumed that the board has a base address of '0' (i.e. A1, A2, A3 = '0') and the read operation is continuous. For a read operation, the setting of bits D0 and D1 of the command register is not necessary. However, it is assumed that D0 and D1 are set to the default setting (00) for normal operation. During a read operation, single bit stream data which has been converted from a RS232 to TTL levels and formatted into a 12-bit word is transferred to the DSP.

The procedure for a read operation is as follows: Test and wait for bit #15 (RXRDY) of the status register to be set to '1'. When the RXRDY bit is set to '1', read the 12 data bits from the DIB buffer. Reading the data clears the RXRDY flag. The data occupies the lower 12 bits of the 16 data lines on DSPLINK. If the data is not read before the next 12 bits are available from the data source, the RXOVR bit is set to a '1' meaning that an error has occurred.

A sample code for the read operation written for the TMS320C25 DSP follows:

```
BOARD          .set          0              ;base address of DIB = 0
RXDATA         .set          BOARD+0        ;Read DATA port address
RDSTAT         .set          BOARD+1        ;Read STATUS port address
RXRDY          .set          0h             ;C25 bit code for receive ready status bit, D15
               .bss          STAT           ;status register variable
               .bss          HOLD           ;data buffer variable
; Sample code begins here
RD_LOOP:
WAIT_RDY:
               IN            STAT,RDSTAT    ;read DIB status register
               BIT           STAT,RXRDY     ;test bit 15
               BBZ           WAIT_RDY       ;keep waiting if bit not set
RD_DATA:
```

```
                IN              HOLD,RXDATA     ;read data from buffer into HOLD

                B               RD_LOOP         ;End of loop

; End of sample code
```

4.2.2 Send Data Example

The procedure for sending data to the DIB is similar to reading data from the DIB. Again, it is assumed that the board has a base address of '0' (i.e. A_1, A_2, A_3 = '0'). It is also assumed that the data is to be transferred to the data buffer. As a result, the command register bits D0 and D1 must be set to '0' prior to the data transfer (see Section 4.1.1 of this appendix). During a send operation, 12-bit words are transferred to the DIB from the DSP and formatted into a single data bit stream. The bit stream is converted from TTL to RS232 levels and then transferred to the data device.

The procedure for a send operation is as follows: Test and wait for bit #13 (TXMT) of the status register to be set to '1'. When the TXMT bit is set to '1', transfer the data to the data buffer. Sending the data clears the TXMT flag. The data is transferred in the lower 12 bits of the 16 data lines on DSPLINK. If the data is not sent, the TXUND bit is set to a '1' meaning that an error has occurred.

```
BOARD           .set            0               ;base address of DIB = 0

TXDATA          .set            BOARD+0         ;Read DATA port address
RDSTAT          .set            BOARD+1         ;Read STATUS port address
COMMND          .set            BOARD+1         ;Command port address

TXMT            .set            2h              ;C25 bit code for transmit empty status bit, D13

SEL_BUFF        .set            0               ;select data buffer

DATA            .set            99              ;Arbitrary data value to be transferred

                .bss            STAT,1          ;status register variable


; Sample code begins here

TX_LOOP:

WAIT_MT:
                IN              STAT,RDSTAT     ;read DIB status register
                BIT             STAT,TXMT       ;test bit 13
                BBZ             WAIT_MT         ;keep waiting if bit not set

SEND_DATA:
                OUT             SEL_BUFF,COMMND;select destination to be data buffer
                NOP
                OUT             DATA,TXDATA     ;transfer data to DIB

                B               TX_LOOP         ;End of loop

; End of sample code
```

4.2.3 Writing to Debug Latches

In cases where a user would like to verify the value of a register, for example, it is possible to write the value to one of three debug latches on the DIB. The procedure for writing data to a debug latch mirrors that of writing to the data buffer. The difference comes from the selection of the destination in bits D0 and D1 of the command register as discussed above. Once the data transfer to the latch is complete, the command register is set back to its default so that the data buffer is selected as the destination. For this example, it is assumed that the board has a base address of '0'. It is also important to mention that there are three debug latches on the DIB, but only one (#1) is brought out on a connector (26-pin) to the front panel. A user can write data to a latch anytime during the program.

The procedure for writing data to a debug latch is as follows: The user sets the command register to address the specific debug latch (bit 0,1) according to Table B2. Then, the data is sent to the latch using the TXDATA command in the program. Finally, the command register is returned to its initial state (zero) in order for the program to continue with the normal path of the data.

Sample code is provided below to transfer a data value to debug latch #1.

```
BOARD           .set            0               ;base address of DIB = 0

TXDATA          .set            BOARD+0         ;Read DATA port address
COMMND          .set            BOARD+1         ;Command port address

SEL_BUFF        .set            0               ;select data buffer as destination
SEL_LAT1        .set            1               ;select debug latch #1 as destination

DATA            .set            99              ;Arbitrary data value to be transferred


; Sample code begins here

SEND_DATA:
                OUT             SEL_LAT1,COMMND;select destination to be debug latch #1
                NOP
                OUT             DATA,TXDATA     ;transfer data to debug latch
                NOP
                OUT             SEL_BUFF,COMMND;select destination to be data buffer

; End of sample code
```

4.2.4 Frame Zero Pulse

For the uplink synchronization experiment, a downlink synchronization reference serial link is implemented to transmit an estimate of the payload clock to the ground terminal in order to begin the process of uplink synchronization. The estimate is transmitted in the form of a pulse. The rising edges and falling edges refer to specific times in the PL system hop sequence. The DIB provides the interface between the simulators and the serial link for both the generation and detection of the FR0 pulse.

To generate the rising edge of the pulse, the FR0_out bit (bit 2) of the command register is set to '1' (default is '0'). For the falling edge, the FR0_out bit is returned to '0'. For the uplink synchronization experiments, the PL continuously generates this reference pulse for the GT to use for syncrhonization.

To receive the FR0 pulse, a user repeatedly reads the status register and monitors bit D11 (FR0_in) of the status register. The GT processor adjusts its clock based on the detection of the rising and falling edges of the pulse. For the uplink synchronization experiments, the rising edge of the FR0 pulse corresponds to the zeroth hop of the zeroth frame in the hop sequence. The falling edge of the FR0 pulse corresponds to the zeroth hop of the first frame.

The sample TMS320C25 DSP code provided directly below illustrates the procedure for transmitting one FR0 pulse.

```
BOARD          .set          0              ;base address of DIB = 0

COMMND         .set          BOARD+1        ;COMMAND port address

FRM0_HI        .set          4              ;set D2 of COMMAND reg to '1'
FRM0_LO        .set          0              ;clear D2 of COMMAND reg

; Sample code begins here
; It is assumed that the FR0 bit is originally set to be 0 (default)

               OUT           FRM0_HI,COMMND ;put a '1' on FR0 bit of COMMAND reg
               NOP                          ;delay
               NOP
               NOP
               NOP
               NOP
               NOP
               OUT           FRM0_LO,COMMND ;put '0' on FR0 bit of COMMAND reg

; End of sample code
```

The sample code written for the TMS320C25 DSP board for detecting the FR0 pulse is as follows:

```
BOARD          .set          0              ;base address of DIB = 0

RDSTAT         .set          BOARD+1        ;Read STATUS port address

FRM0_RX        .set          4h             ;C25 bit code for FR0 bit on status reg (D11)

               .bss          STAT,1

; Sample code begins here

DET_FR0:
               IN            STAT,RDSTAT    ;read status register of DIB
               BIT           FRM0_RX,STAT   ;test bit D11
               CALL          CHK_4_EDGE     ;call subroutine to check for fall/ris edge
                                            ;(not included here)
```

Appendix C: Test Program for DIB

1. General

This appendix contains the listings of the program used by the PC (station) to test the DIB according to Fig. 7 of the main document. The testing steps are briefly described in section 4 of this document. There will be comments after the program codes when needed to explain the meaning of that particular code.

2. Program Listings

The program listings for the testing of the DIB are found in the following pages.

2.1 **C Listing (Berhost).** User interface program for a PC.
2.2 **Assembler listing (Bertest).** DSP assembler program.
2.3 **Berhost.MAK.** To compile and link C program.
2.4 **Bertest.MAK.** To compile and link assembler program.
2.5 **Bertest.CMD.** Linker file used to produce output file to be loaded into DSP memory.

2.1 C Listing

Berhost.C

```
#include <stdlib.h>
#include <conio.h>
#include <stdio.h>
#include "TIC25.h"                    /* TMS320 development Lib. header file.  */

#define PROC_BD 3                     /* Corresponds to I/O ports 390-397 */

#define MAXWAIT 50000L
#define TIMWAIT 2000L

#define CLEAR   0
#define TBEGIN  1
#define TSTAT   2
#define TRESLT  3
#define TDEMOD  4

#define C25IN   0
#define C25P1   1
#define C25P2   2
#define C25P3   3
#define PCIN    4
#define PCP1    5
#define PCP2    6
#define PCP3    7
#define FR0_FLG 8
#define DEBUG     9

ushort mail_base;
int stop_flag = 0;

void init_board(int board);
void send_and_wait(ushort type,ushort p1,ushort p2,ushort p3,long count);
ushort get_mail(ushort *p1,ushort *p2,ushort *p3,long count);
ushort mread(ushort addr);
void mwrite(ushort addr,ushort val);
void checkkey(void);
void display_mail(ushort type,ushort p1,ushort p2,ushort p3);

void main()
{
    ushort ans,t1,t2,t3,mtype;
    long i,j;

    init_board(PROC_BD);
    send_and_wait(TBEGIN,0,0,0,MAXWAIT);
    exit(0);
}
```

```
void init_board(int board)
{
    int loadstatus;

    SelectBoard(board);
    AssertReset();
    UnHold();
    RemoveFlag();
    CommRWDis();
    CommRRDis();
    ReadStatReg();
    ReadStatReg();
    loadstatus = LoadObjectFile("bertest.out");
    if (loadstatus != 0) {
      printf("Error!  Not able to load \bertest.out\ down to the board\n");
      exit(1);
    }
    RelReset();
    WaitComWr(MAXWAIT);
    mail_base = RdCommReg();
    return;
}


void send_and_wait(ushort type,ushort p1,ushort p2,ushort p3,long count)
{
    long i,j;

    mwrite(C25P1,p1);
    mwrite(C25P2,p2);
    mwrite(C25P3,p3);
    mwrite(C25IN,type);
    mwrite(DEBUG,1);

        for (j=0;j<2000;j++) {
      for (i=0;i<500;i++) {
            /*nothing*/
            }
      mwrite(FR0_FLG,0);
      for (i=1;i<500;i++) {
            /*nothing*/
            }
      mwrite(FR0_FLG,1);
      }
      mwrite(DEBUG,0);
    return;
}
```

```
ushort get_mail(ushort *p1,ushort *p2,ushort *p3,long count)
{
    long i;
    ushort mtype;
    if (count != 0) {
      for (i=0;i<count;i++) if ((mtype=mread(PCIN)) != CLEAR) break;
      if (i == count) return CLEAR;
    } else {
      while ((mtype=mread(PCIN)) == CLEAR) ;
    }
    *p1 = mread(PCP1);
    *p2 = mread(PCP2);
    *p3 = mread(PCP3);
    mwrite(PCIN,CLEAR);
    return mtype;
}

ushort mread(ushort addr)
{
    return GetMem16('D',addr+mail_base);
}

void mwrite(ushort addr,ushort val)
{
    PutMem16('D',addr+mail_base,val);
    return;
}

void checkkey(void)
{
    int c;

    if (kbhit() == 0) return;
    c = getch();
    switch (c) {
      case 'q':
      case 'Q':
      case 'x':
      case 'X':
      case 'e':
      case 'E':
      case '\033':
          stop_flag = 1;
          break;
      case 0:
          c = getch();
          break;
      default:
          break;
    }
    return;       }
```

```
void display_mail(ushort type,ushort p1,ushort p2,ushort p3)
{
    char buff[80];

    switch (type) {
      case CLEAR:
          printf("No message - timeout\n");
          break;
      case TBEGIN:
          printf("Illegal BEGIN message\n");
          break;
      case TSTAT:
          ltoa((long) p1,buff,2);
          printf("STATUS %16s  (%X)\n",buff,p1);
          break;
      case TRESLT:
          ltoa((long) p1,buff,2);
          printf("  Res = %12s  (%X)\n",buff,p1);
          break;
      case TDEMOD:
          ltoa((long) p1,buff,2);
          printf("Demod = %12s  (%X)\n",buff,p1);
          break;
      default:
          printf("Illegal message type %d  (%d, %d, %d)\n",type,p1,p2,p3);
          break;
    }
    return;
}
```

2.2 Assembler Listing

Bertest.ASM

```
            .global     MAIN
            .global     RESET,INT0,INT1,INT2,TINT,RINT,XINT,USER,BADINT

PORT0:      .set  0

BOARD:      .set  8
RXDATA:     .set  BOARD + 0         ; Receive data (read only)
TXDATA:     .set  BOARD + 0         ; Transmit data (write only)
SFLAGS:     .set  BOARD + 1         ; Status flags (read only)
HOPEDG:     .set  15                ; C25 bit code for Bit position 0 (LSB)
FIFOMT:     .set  14                ; C25 bit code for Bit position 1
FRMZERO:    .set  4                 ; C25 bit code for Bit position 11
TXUND:      .set  3                 ; C25 bit code for Bit position 12
TXMT:       .set  2                 ; C25 bit code for Bit position 13
RXOVR:      .set  1                 ; C25 bit code for Bit position 14
RXRDY:      .set  0                 ; C25 bit code for Bit position 15 (MSB)
COMMND:     .set  BOARD + 1         ; Command - used for reset (write only)
ADDATA:     .set  BOARD + 2         ; A/D FIFO data (read only)

CLEAR:      .set  0
TBEGIN:     .set  1
TSTAT:      .set  2
TRESLT:     .set  3
TDEMOD:     .set  4


            .sect "IRUPTS"
RESET:      B     MAIN
INT0:       B     BADINT
INT1:       B     BADINT
INT2:       B     BADINT
            .space      16 * 16
TINT:       B     BADINT
RINT:       B     BADINT
XINT:       B     BADINT
USER:       B     BADINT



            .sect "IniDat"
MAILBAS:
C25IN:      .word CLEAR
C25P1:      .word 0
C25P2:      .word 0
C25P3:      .word 0
PCIN:       .word CLEAR
PCP1:       .word 0
```

```
PCP2:       .word 0
PCP3:       .word 0
FR0_FLG     .word 0
DEBUG       .word 0

TMPIN:      .word 0
TMPP1:      .word 0
TMPP2:      .word 0
TMPP3:      .word 0

XOLD:       .word 0
YOLD:       .word 0
X:          .word 0
Y:          .word 0

STAT:       .word 0
HOLD:       .word 0
FLAG:       .word 0
TEMP:       .word 0
C0:         .word 0

UN:         .word 1
DEUX:       .word 2
TROIS:      .word 3
FRM_0:      .word 4
QZ:         .word 8000h
NUM         .word 0AAAAh

            .text

BADINT:
            SXF
BADLOP:     B     BADLOP


;================================================
;                        Main Program
;------------------------------------------------

MAIN:
      DINT                      ; Ensure interrupts disabled
      ROVM                      ; Reset overflow mask
      LARP  AR1                 ; Set ARP to AR0
      LDPK  0                   ; Point to page 0 (control area)
      LACK  0                   ; Mask off all interrupts
      SACL  4
      RSXM                      ; Disable sign extension
      SPM   0                   ; No shift on P register
```

```
      LDPK  MAILBAS          ; Point to data area

      LALK  MAILBAS          ; Send mail base address to PC
      SACL  TEMP
      OUT   TEMP,PORT0
      LALK  0
      SACL  FLAG

WAITBG:
      CALL  WAITM            ; Waits for FLAG to begin program
      SUBK  TBEGIN           ;
      BNZ   WAITBG           ;

LOOP:
      LAC   FR0_FLG          ; Checks for FRM_0 and makes it
      BNZ   CONT1            ; vary for testing purpose (simulate)
      OUT   FRM_0,COMMND     ;
      B     CONT2            ;
CONT1:                       ;
      OUT   C0,COMMND        ;

CONT2:
      IN    STAT,SFLAGS      ; Receives status bits from the BIB

      LAC   FLAG
      BNZ   L1
      BIT   STAT,RXRDY       ; Checks for the RX_ready status bit
      BBZ   L1               ; If the bit is at '0' go to L1
      IN    HOLD,RXDATA      ;stores data from BIB in HOLD

      LALK  1
      SACL  FLAG

      B     LOOP

L1:
      LAC   FLAG
      BZ    L2
      BIT   STAT,TXMT        ; Checks for the RX_ready status bit
      BBZ   L2               ; If the bit is at '0' go to L2
      NOP
      OUT   HOLD,TXDATA      ; send data in HOLD to the BIB
      LALK  0
      SACL  FLAG

      LAC   DEBUG
      BZ    LOOP
      OUT   UN,COMMND        ; Adresses Debug latche #1
      OUT   HOLD,TXDATA      ; Puts the content of HOLD on latch
      OUT   C0,COMMND        ; Back to initial state
      B     LOOP
```

```
L2:
      B     LOOP


;_________WAIT BEGIN________

WAITM:
      LAC   C25IN
      BZ    WAITM
      SACL  TMPIN
      LAC   C25P1
      SACL  TMPP1
      LAC   C25P2
      SACL  TMPP2
      LAC   C25P3
      SACL  TMPP3
      LALK  CLEAR
      SACL  C25IN
      LAC   TMPIN
      RET
            .end
```

2.3 Berhost.MAK

```
berhost.exe:    berhost.obj
      link berhost,,,sti25dev;

berhost.obj:    berhost.c
      cl /c /Od berhost.c
```

2.4 Bertest.MAK

```
BERTEST.OUT:      BERTEST.OBJ
      c:\tms\DSPLNK BERTEST.CMD

BERTEST.OBJ:      BERTEST.ASM
      c:\tms\DSPA BERTEST.ASM -l
```

2.5 Bertest.CMD

```
 BERTEST.OBJ                  /* BER Board Test Software          */

 -e MAIN                      /* Entry point                      */
 -o BERTEST.OUT               /* Executable file                  */
 -m BERTEST.MAP               /* Map file                   */
```

```
MEMORY
{
      PAGE 0:     VECTORS: origin =     0H,       length =    03FH
                  PROG:  origin =  400H,  length = 01C00H
      PAGE 1:     DATA:  origin = 8000H,  length = 02000H
}

SECTIONS
{
      IRUPTS:     { } > VECTORS     PAGE 0
      .text:            { } > PROG  PAGE 0
      .data:            { } > PROG  PAGE 0
      IniDat:     { } > DATA  PAGE 1
      .bss:             { } > DATA  PAGE 1
}
```

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM
(highest classification of Title, Abstract, Keywords)

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Defence Research Establishment Ottawa
Ottawa, Ontario
K1A 0Z4

2. SECURITY CLASSIFICATION (overall security classification of the document including special warning terms if applicable)

UNCLASSIFIED

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C or U) in parentheses after the title.)

Multipurpose Data Interface Board (DIB) (U)

4. AUTHORS (Last name, first name, middle initial)

Simoneau (Capt), Yves and Tom, Caroline

5. DATE OF PUBLICATION (month and year of publication of document)

July 1998

6a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc.)

84

6b. NO. OF REFS (total cited in document)

6

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.

DREO Report

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include the address.

5ca11

9a. PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant)

9b. CONTRACT NO. (if appropriate, the applicable number under which the document was written)

10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)

DREO REPORT 1332

10b. OTHER DOCUMENT NOS. (Any other numbers which may be assigned this document either by the originator or by the sponsor)

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(X) Unlimited distribution
( ) Distribution limited to defence departments and defence contractors; further distribution only as approved
( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
( ) Distribution limited to government departments and agencies; further distribution only as approved
( ) Distribution limited to defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). however, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

Unlimited Announcement

13. **ABSTRACT** (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

Defence Research Establishment Ottawa is pursuing an in-house research activity in spread-sprectrum technology to support development of robust, anti-jam satellite communications for the military. The in-house effort consists of developing a system simulator, including both a ground terminal processor and a payload processor, to research techniques involved in spread-spectrum synchronization. For these experiments, a multipurpose data interface board is required for different data operations, and is the subject of this report. The board is composed of mainly an erasable programmable logic device to reduce the number of integrated circuits and to add flexibility to the design. The board was designed to perform three functions. The first function is the data format conversion between a ground terminal processor and a data source, and likewise, a payload processor and a data sink. The second function is to provide an interface to a separate direct link between the payload and ground terminal subsystems for transmitting a reference pulse for synchronization. The third function is to provide a set of debug latches for the user. In this document, the software and hardware details are provided along with a user's guide for the board.

14. **KEYWORDS, DESCRIPTORS or IDENTIFIERS** (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

data format conversion
data interface board
erasable programmable logic device
EPLD