Def Sci J, Vol 37, No 3, July 1987, pp 297-304.

Software for Bus Monitor

T.V. Rama Murthy
National Aeronautical Laboratory, Bangalore-560 017

and

Sanath Kumar Varambally
Karnataka Regional Engineering College, Srinivasnagar-574 157

ABSTRACT

Software for Bus Monitor (SOBUM) is a package developed for MIL-STD-1553B based on Intel's Microprocessor Development System (MDS). SOBUM, consisting of modules in ASM 86 and PASCAL 86 when used with proper hardware interface can transfer the bus messages to the RAM in real time. SOBUM is then used in the off-line analysis of the message traffic on the bus. It displays or prints the data gathered in very useful and interpretive formats.

1. INTRODUCTION

More than a decade ago the military aircraft designers visualised an airborne central computer complex for the management of aircraft subsystems. However, the wiring to the remote sensors, actuators and electronics was a significant design obstacle. After many reviews the military standard MIL-STD-1553B aircraft internal time division command/response multiplex bus evolved[1] to simplify the interconnection task in the aircraft. As an example the aircraft A-7E has 1581 multiplexed signals – 721 avionic and 860 electrical power distribution signals.[2] The application of the standard in some of the existing aircrafts has been studied by the author.[3]

The MIL-STD-1553B describes the method of communication and the electrical interface requirements for subsystems connected to the data bus. Some of the key-elements are the bus-controller (BC), the embedded sensor, the standalone remote terminal (RT), twisted shielded pair wire data bus and the optional isolation couplers

Received 14 May 1986, revised 9 March 1987

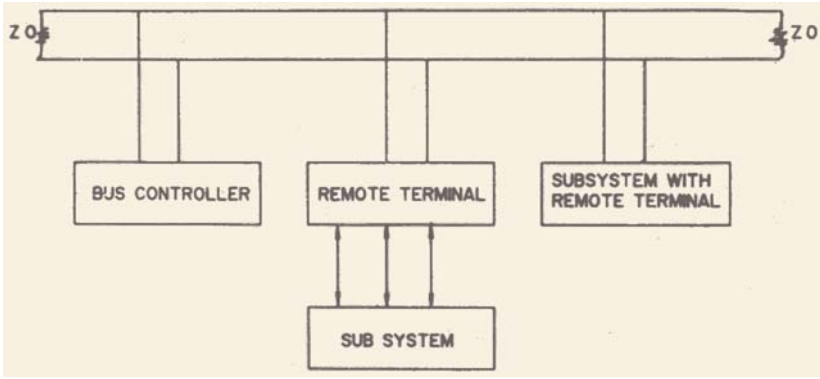

Figure 1. MIL-STD-1553B multiplex data bus architecture.

(Fig 1). In this paper single level bus topology is considered. In this type of network, a single shielded coaxial cable with terminations on either side constitutes the 1 MHz bus. Each terminal called as a remote terminal is connected through a transformer on to the bus. These RT's are functionally independent units like flight control computer fire control computer, display and key-board panel etc.

Within each RT, there is a further partition called sub-address. There are 31 RT's and 31 sub-addresses altogether. Each sub-address can have utmost 32 words (word-count) which can be accessed through the bus. The units defining the protocol is shown in Fig. 2.

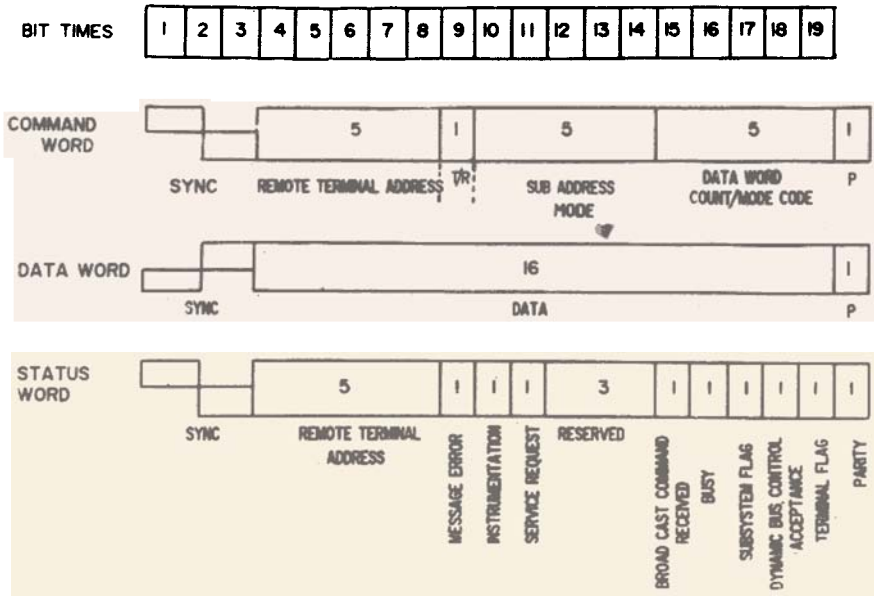

Figure 2. MIL-STD-1553B word formats.

The transactions are controlled by the BC. It issues the command word containing the address of the specific remote terminal to transmit or receive the data on the bus. The RT responds to the BC and hence this type of protocol is called as command/response protocol. Five types of message transmissions issued are :

(a) BC to RT – called as receive command where the T/R bit is set to logic 0. The RT receives the specified number of data words and later RT responds with a status word.

(b) RT to BC – called as transmit command where T/R bit is set to logic 1. The RT transmits a status word and follows it with data words.

(c) RT to RT – The BC first designates the receiver and immediately issues a second command word to designate the transmitter. The status word flows from the transmitter and follows it with the data words. The receiver answers with its status word.

(d) Broadcast – The controller issues a receive command with RT address-31 followed by data words. All those with broadcast options recognise that RT address of 31 implies broadcast message transfer and recieve the data. In RT to RT type of broadcast message transfer, the transmit command is issued to the specific RT which transmits the data word.

(e) Mode code – To analyse the status word further, the BC has to issue a mode command to the unique terminal. Mode code is indicated by setting the sub address in the command word to 31. However the mode code is used to resolve the conflicts in the bus management. In many aircrafts, the mode code for only the dynamic bus control is used.[4]

All the transmissions on the bus is governed by the bus controller using the command/response method. The monitor performs two distinct functions – (i) It listens to all addresses or a subset of addresses and stores the data for further processing. (ii) It acts on the data to resolve higher level system configuration or bus controller bugs.

2. SOFTWARE FOR BUS MONITOR (SOBUM)

SOBUM is an example of how the Intel MDS can be used to match the performance of the high cost stand-alone units and also improve on them. The users of SOBUM can easily change the software to suit to their needs. SOBUM performs the task of collecting a preset number of bus words on the 1553B bus in real time and then switch to off-line mode when it will dump the collected data into a disc file. It later invokes procedures to group each bus word into units defined in 1553B protocol and displays the entire bus transactions that has taken place. There are various options available to the user to extract the information flow on the bus. The software can be modified to make the unit as a talker and it can interrogate the sub-systems and test their responses.

The message transactions that take place on the bus has to be analysed and presented in user understandable formats – units which make up BC address, RT address, T/R bit, sub address, mode code, data word etc. A typical situation where

SOBUM works is to track the message exchanges that takes place between two subsystems, say flight control computer (FCC) and fire control computer. For a particular mode that has been chosen, say, air to air or air to surface role, the designer would like to know how the system is responding to the command. Assuming bus transactions to take place at a task rate of 10 Hz and memory capacity to store bus transactions as 32K, the worst case timing consists of one control word, one status word and one data word. There will be about 1000 words in the memory amounting to data exchanges of 100 sec duration which can be analysed by application programs. The MDS will have to be invariably used in the development phase because of the support it gives for the multiprocessor based systems.

3. BUS MONITOR INTERFACING

The data bus can be integrated with MDS by making use of Data Device Corporation's (DDC) BUS-1553 and Intel's SDK-86 card as shown in Fig. 3 and 4. Bus-1553 is interfaced with the data bus on one end and on the other end its 16 bit data output lines are connected to ports B of 8255(1) and 8255 (2). Ports A and C of 8255 (1) and port A of 8255 (2) are programmed as output ports. Port B of 8255 (1) and ports B and C of 8255 (2) are programmed as input ports. A brief description of the relevant pins of the DDC's BUS-1553 module[4] is given below :

Pin	fan in/out	description
TX	in	External input logic '0' disables the receiver while logic '1' inhibits the transmitter
RST	out	Receiver Sync type logic '1' – command logic '0' – data
RB	out	Receiver busy
VWR	out	Valid word received. VWR level goes high only after valid sync, 16 data bits and odd parity have been received. (refer fig. 5 for waveforms)

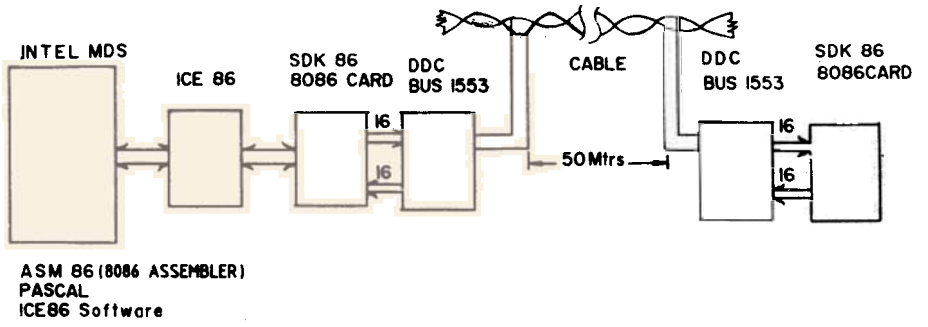

Figure 3. MIL-STD-1553B development system schematic.

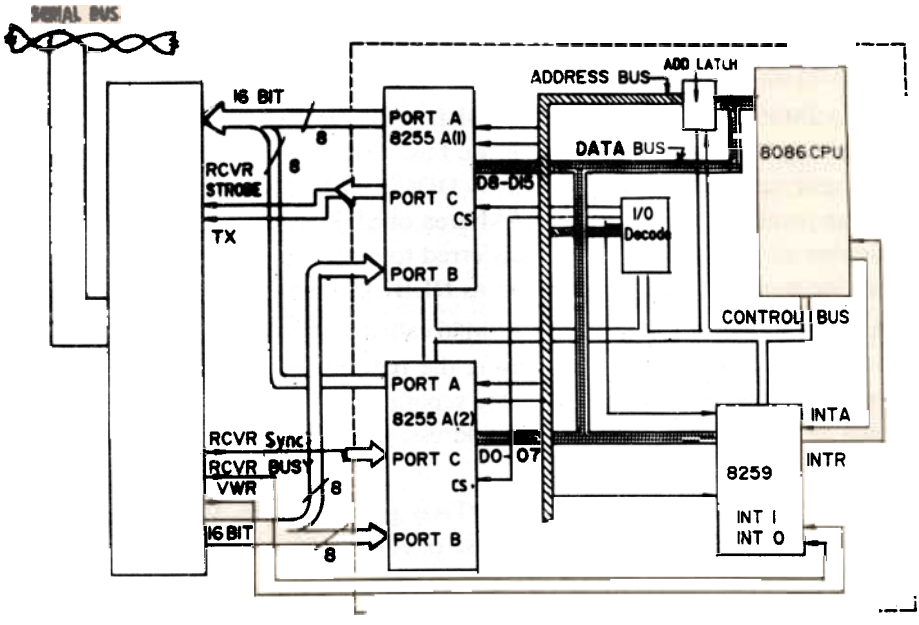

Figure 4. Hardware configuration to interface the 1553B with 8086 up.

As the device is hooked as a listener, the transmitter mode is inhibited. Receiver sync type outputed as RST has to be taken into know whether it is a command word or not. RB and VWR signal the presence of valid RST and the presence of valid word on the 16 bit data bus. Hence RB, VWR will be connected as two interrupts to 8259 so that the interrupt service routines can take in sync word and bus word respectively through the input ports. Two lines of port C of 8255 (1) are used to make BUS-1553 operate in receiver mode. This is made by keeping receiver strobe at logic 1. The transmitter is inhibited by applying logic 1 to TX of BUS-1553. The pin receive sync type (RST) of BUS-1553 is connected to the input port C of 8255 (2).

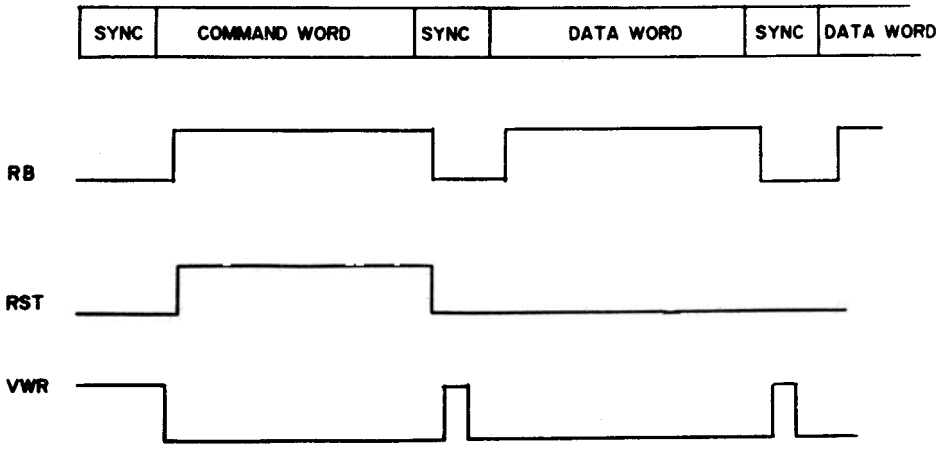

Figure 5. Receiver timing waveforms.

4. BUS MONITOR PROGRAM

SOBUM consists of the folowing modules to perform the task of bus monitoring.

(a) An ASM86 routine called BUS-WORD. ASM invoked in ICE-86 mode starts collecting the words that flow on the bus. The synchronisation is achieved via the interrupts RB and VWR given by DDC's BUS-1553B device. The bus words are sequentially stored in the RAM area of SDK-86 card. At the end of preset number of words, the data is transferred to disc storage area and then it enters off-line mode called by the system as DEBUG86 mode.

(b) Another module of SOBUM in ASM86 called as ANALYS. ASM is then called which transfers the saved data using the register pairs ES and DI with CX as the count register. This performs the separation of the elements of the bus word such as RT address, T/R bit, sub address, mode code, word count, data word and status word.

(c) An ASM86 routine called as CMD-TAB generates the ASCII value for each digit.[5] This used as display, makes use of ASCII value of each character.

(d) The Pascal procedure REM-ADDR is called to communicate with the user to select the mode-codes or terminal activity selection.[6] The subroutine MODE-CODE compares the particular mode with the allowed mode-codes and hence determines whether it is legal or illegal. The procedure WRITE-MODE displays the mode command. The terminal activity selection will lead to the choice of whether all RTs activity is to be displayed or that of a selected RT. The Pascal procedure WRITEIT takes in the above information given by the user. In case of selected RT activity is desired by the user, the program compares and displays the transactions on the bus concerned with the selected RT only. The procedure WRITEIT takes the variables RT-ADDR, TR-REC, SUB-ADDR, WC-MC of the bus-word as array of four characters. If the bus-word is a command word then this programme writes its RT-ADDR, TR-REC, SUB-ADDR, WC-MC and writes it as command word. If the bus-word is a status word, then this programme writes each digit of status word in binary representation. This is achieved by using a look-up table where binary representation of Hex digit are written in an external file called NUMS. Every time a Hex digit is picked the corresponding binary representation is read from NUMS and is written into the display. If the bus-word is a data word, then it is written in binary format and the word 'DATA WORD' is written into the display.

A pseudo coded program of the above routines are as follows :

```
BUS-WORD. ASM
 begin
  initialise 8086 registers, 8255 ports and 8259;
  while preset < num do {num = required no. of data-words}
   begin
     while interrupt1 do
      begin
```

```
        clear interrupt1;
        input RST and move it to BUSWD [I];
        set interrupt;
        increment I;
        increment preset
        end; {interrupt 1}
       while interrupt 2 do
        begin
         input data-word and move it BUSWD [I];
         increment I;
         increment preset
        end {interrupt 2}
         end {preset}
        end {prog}.
ANALYS ASM
begin {main}
initialise the system routines;
set up registers ES, SI, DI for accessing the global file where SAVE system
has dumped the 1553B bus transactions;
for cx: = 0 to count do
  move the BUSWD determined by ES, DI and move it to syncword SYNC [DI];

 WRITEIT
 {seek user requirements like unique RT transactions, unique RT-RT
 transactions or mode code transactions or RT-RT transactions or all;
   seek disc file or VDU options}
   reinitialise SI, DI values;
   for temp:= 0 to count do
   begin
     move SYNC [SI] to unit-word;
   if unit-word is 0 then DATA: =1
     else begin {decompose}
       decompose unit-word to protocol terms like RT-address, T/R bit,
Subaddress/mode, word count/mode-code, instrumentation bit etc;
      If RT-address is 31 then BRD-CST: =1
      else if Sub-address is 31 then
        begin {mode}
          MODE: =1;
          MODE-CODE, WRITE-MODE; {procedures for checking lega l or
          illegal mode codes and displaying the details of the code}
        end; {mode}
      else if the instrumentation bit is 1 then STATUS: =1
      else COMMAND: =1:
```

```
end; {decompose}
obtain the ASCII value of the unit-word;
call Pascal subprogram to display or store the transactions in disc file
end {for temp}
end.
```

5. CONCLUSION

SOBUM, a software package for data bus monitoring has been developed as debugging tool in the development phase of data communication on the 1553B bus based on the Intel MDS. SOBUM run on the Intel MDS with its resident software tools and multibus environment provides a cost-effective and better alternative to some of the existing data bus monitors. Further work needs to be carried out to resolve higher level bus problems with the aid of the latest hardware support.[7]

ACKNOWLEDGEMENT

The authors wish to thank the Director, National Aeronautical Laboratory for permitting to publish this paper. The authors are grateful to the Head, Systems Engineering Division, National Aeronautical Laboratory and Prof. K.M. Hebbar, Head of the department of Electronics and Communication of Karnataka Regional Engineering College for encouraging in this work.

REFERENCES

1. A triservice military standard for aircraft internal time division command/response multiplex data bus: MIL-STD-1553B (US Air Force, US Govt Printing office), 21 September 1978.
2. Jones, J.L., *et al*, Application of general purpose multiplex system to the A-7E avionics, Proc. NAECON/IEEE, Vol 1 (1978), p. 122.
3. Rama Murthy, T.V., Review of multiplexing techniques in modern combat aircrafts, Technical Memorandum, National Aeronautical Laboratory, Bangalore. TM-SE-427/1-83, November 1983.
4. Designers guide MIL-STD-1553B, (ILC Data Device Corporation, New York), 1982.
5. ASM86 language reference manual No. 121703 (Intel Corporation, USA), 1982.
6. PASCAL86 users guide No. 121539 (Intel Corporation, USA), 1982.
7. DDC product catalog supplement, (ILC Data Device Corporation, New York), October 1986.