NOVEMBER 1974
Alan J. Weissberger
National Semiconductor

THE IMP-16 IN COMMUNICATION APPLICATIONS

INTRODUCTION

One of the most promising applications of LSI microprocessors is in the field of data communication. The improved reliability, flexibility and performance offered by these components makes them a natural for such tasks as message switching, smart terminal controllers, pre-processors and data concentrators. A microprocessor can handle many different codes, messages and line protocols; adapt to new operating requirements; and provide the needed real-time response to meet data transfer demands.

National's IMP-16 is a high performance 16-bit processor that is a very cost effective solution for data communication applications. Its powerful 16-bit instruction set, interrupt and input/output structure are the very qualities most important for these applications. The IMP-16 is equally adept at handling 8-bit character data or 16-bit arithmetic computations.

COMMUNICATION PROCESSOR FUNCTIONS

A communication processor may be programmed to provide a wide variety of functions at each level in the network shown in *Figure 1.* Some of these functions include:

1. Line protocol administration involving automatic dial-up, automatic answering and periodic polling of terminals to determine their status.
2. Adaptive line speed control for various speed terminals and communication lines.
3. Code conversion between terminals and the host processor or between dissimilar terminals.
4. Message assembly, syntax checking and reformatting to improve line efficiency and real-time response by off loading the host processor.
5. Data compression to improve bandwidth utilization.
6. Error control including detection of error codes (parity, CRC, block checks), detection of open line condition, inter character time-out, incorrect or no-poll responses, retransmission requests, and reporting unclearable errors.
7. Line monitoring, traffic analysis, message accounting and billing.
8. Task or address based message routing and switching.
9. Terminal control including formatting, cursor control, recognizing keyboard strokes, filling up buffers, data conversion, hard copy printouts, etc.
10. Terminal testing to locate trouble spots. Diagnostic programs can be "down line loaded" from the host or may be stored locally on a floppy disc or cassette.
11. Special functions such as communication instrument control, digital filtering or peripheral control.

ESSENTIAL COMPONENTS OF COMMUNICATIONS MICROPROCESSORS

Listed below are some of the desirable features of communications oriented microprocessors:

1. Multiple addressing modes with wide ranges. Indexed, indirect, immediate and relative addressing provide flexibility and simplify programming.
2. Good character handling, arithmetic and logic instructions for moving, manipulating and testing bits and bytes of data.
3. Architectural features such as multiple accumulators, multiple index registers, status register, and internal stack. The internal stack can be used to minimize software overhead and excessive memory references during interrupt servicing and subroutine linking. When buffer memory is full temporary data may be saved on the stack.
4. Flexible input/output (I/O) to monitor and control different types of terminals or channels. User flags and jump condition inputs are helpful for serial I/O while parallel data transfer commands and the ability

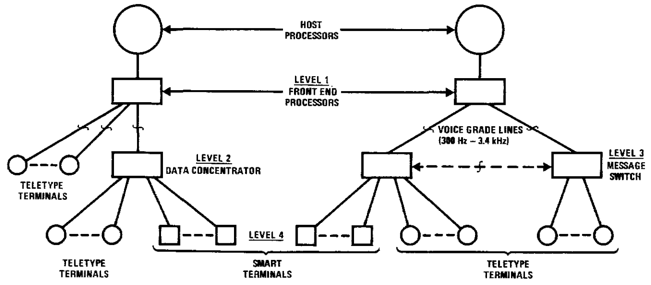

FIGURE 1. Four Levels of Application for Communication Microprocessors

to execute memory reference instructions on peripherals speeds the data flow.

5. Ability to handle synchronous or asynchronous communication lines.

6. Fast responding interrupt capabilities to meet all data transfer demands. General interrupts for simple line or terminal control, multi-level interrupts for intermixing slow and high speed devices, vectored interrupts for very fast real-time response.

7. Direct Memory Access (DMA) capability for throughput enhancement and high speed block transfers of data. The DMA scheme should be limited only by the buffer memory cycle time and not by the microprocessor.

8. Error checking features including the ability to implement a variety of error control procedures.

9. Microprogrammability for special purpose instructions to improve speed and simplify assembly language programming. Code conversion, byte swapping, and Cyclic Redundancy Check (CRC) are examples of custom instructions that could be microcoded.

ENTER THE IMP-16

The IMP-16 meets every one of the above requirements! Furthermore, its 16-bit word length makes it a more efficient data handler than currently available 8-bit processors. Let's take a closer look at 16-bits vs eight.

1. As a pre-processor or front-end, a 16-bit processor increases system thorughput when communicating with the host processor. Since data is transferred as 16 parallel bits, only one instruction or DMA transfer is needed to communicate with a 16-bit host computer; two for a host with word length of up to 32 bits. An 8-bit processor would require twice as many transfers, slowing down the system.

2. A 16-bit instruction word can specify many more operations than an 8-bit instruction. This makes the 16-bit processor easier to program. If an 8-bit processor executes 16-bit instructions, two memory references are required to fetch the instruction. Only one such memory reference is required by the IMP-16, again boosting system throughput.

3. Any one of 64k locations can be addressed in a single instruction of a 16-bit machine. An 8-bit processor must resort to address computations involving double precision arithmetic and multiple registers to formulate 16-bit addresses.

4. Processing efficiency and accuracy are improved by operating on 16 bits in parallel. This is very important in error checking, making transmission line measurements, routing analysis of communication traffic, compilation of error statistics and customer billing.

5. Extra bits in a word may be used for various control or security options. These might include: control or data character, source or destination message, background or foreground, terminal or channel address, odd or even parity.

COMMUNICATIONS SYSTEM CONFIGURATIONS

In *Figure 2* the IMP-16 is a *front end processor* for a large host computer. It is supported by its own program and buffer memories and appears as another peripheral to the host. Communication is over the host's I/O bus on a program controlled or cycle steal basis. Conventional UART's are used as the telephone line interface to the microprocessor. AN-131 provides a detailed look at the IMP-16 in this application.

A *data concentrator* or *message switch* will collect, buffer and pre-process information from several teletype or CRT terminals, and transmit that information over phone lines to a large scale computer or another terminal. The functional block diagram of *Figure 3* depicts this arrangement. The terminal controller provides a parallel interface to the IMP-16. The controller makes an interrupt request when it has an input character for the IMP-16 or when it is ready to receive an output character. The interrupt controller is essentially a priority encoder which informs the processor which terminal is requesting service. The powerful IMP-16 interrupt structure (see AN-107) is extremely important for this application. General, multi-level and vectored interrupts may be implemented depending on terminal load and response time requirements.

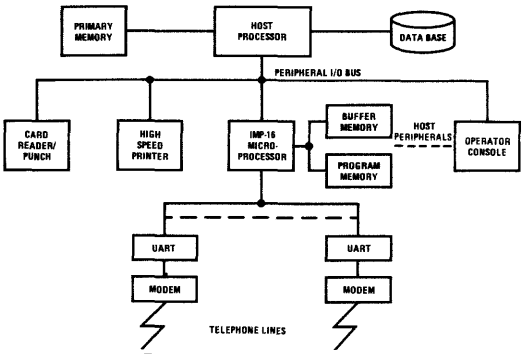

FIGURE 2. Front End Processor

A *smart terminal* or remote job entry controller is shown in *Figure 4.* Remote processing by the IMP-16 is more cost effective than a central site computer handling a group of dissimilar terminals. The microprocessor control program is specifically tailored to the configuration at the remote site and can handle much of the processing locally. The resultant reduction in communication traffic with the central site provides savings in line costs while improving real-time response and message security.

An 8-bit data memory (RAM) is used with a 16-bit instruction memory (ROM) in this application. This technique eliminates the need to pack and unpack 8-bit characters while retaining the performance benefits of a 16-bit processor. *Figure 5* illustrates one typical implementation.

Characters may be input from the terminal one at a time while IMP-16/host transfers may be in two character (one 16-bit word) units. This maximizes both data handling speed and throughput.

PROGRAMMING EXAMPLES

The power and versatility of the IMP-16 instruction set enables efficient handling of a wide variety of tasks. A few examples will illustrate this point.

Status Monitoring

Consider the task of inputting an I/O status word from a terminal, testing the busy bit in that word and requesting data from the terminal when the bit is set. If the bit is zero we are to continue to monitor the status word. This coding is shown in Table I.

The terminal is addressed as if it were a memory location in this example. It is assumed the terminal controller address is in accumulator 2 so that any arbitrary address can be used when accessing the status or data word from the terminal. This technique provides fast handling (5 cycles of 7μs to obtain the data) and programming flexibility (the data can be input into any one of three accumulators) through use of the Load (LD) instruction. Note that AC2 contains the terminal controller address. The Branch-On-Condition (BOC) instruction is used to test bit 15.

In the second example (Table II) we will input the status word, test bits 0 and 1 and take appropriate action. If bit 0 is set we will input data from the terminal and store in memory; if bit 1 is set we'll store the previous data from the terminal; if neither bit is set an alternate action is taken.

This routine might be used in a terminal controller application where two terminals A and B are assigned status bits one and zero, respectively. The data that is to be stored in SAVE will come from the terminal that is ready (status bit set) with terminal B tested first. If neither terminal is ready an alternate action (possibly a recording of the condition) is taken.

To test any number of arbitrary status bits we may shift or rotate the status bit into bit positions 0, 1 or 15 and use the BOC 3, 4 or 2 instruction as required. Alternatively we can use a mask word to test bits in parallel.

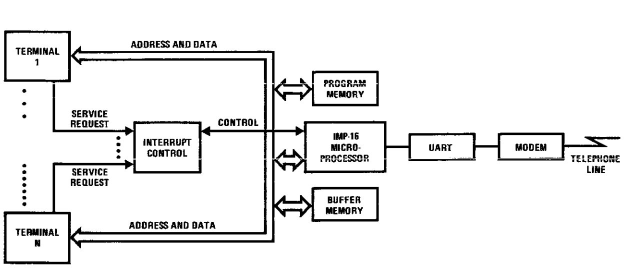

FIGURE 3. Data Concentrator or Message Switch

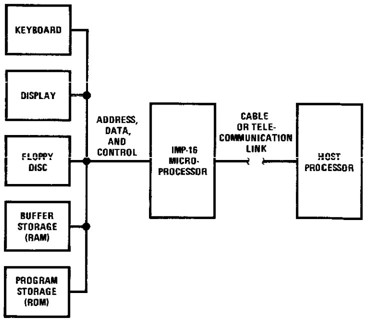

FIGURE 4. Smart Terminal

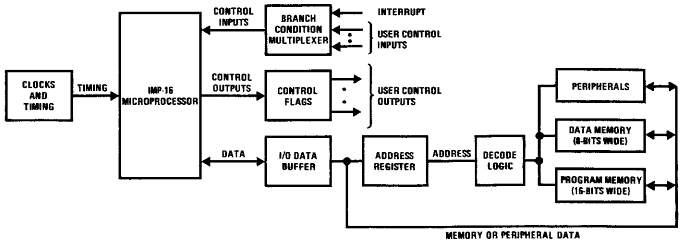

FIGURE 5. IMP-16 Microcomputer System

Table I

```
; BIT 15 OF STATUS WORD IS BUSY BIT
; = 0 IDLE
; = 1 BUSY
; ADDRESS OF TERMINAL CONTROLLER IN ACCUMULATOR 2
; STATUS = TERMINAL STATUS WORD ORDER
; DATA = TERMINAL DATA WORD ORDER
  LD          0, STATUS (2)        ; INPUT STATUS WORD
  BOC         2, . -1              ; LOOP TILL READY
  LD          0, DATA (2)          ; INPUT DATA FROM TERMINAL
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  Cycles:     14 + N(10)           where N is the number of times the
                                   status bit is zero when tested
              (196μs for 14 cycles)
  Words:      3
```

Table II

```
TERMB = TERMINAL B DATA ORDER
; TERMINAL CONTROLLER ADDRESS IN ACCUMULATOR 2
; PREVIOUS TERMINAL A DATA IN ACCUMULATOR 1
         LD      0, STATUS (2)
         BOC     3, GDATA             ; TEST BIT0 – BRANCH IF TERMINAL B
         BOC     4, GDATA + 1         ; TEST BIT1 – BRANCH IF TERMINAL A
; TAKE ALTERNATE ACTION HERE
         •
         •
         •
GDATA:   LD      1, TERMB (2)         ; GET TERMINAL B DATA
         ST      1, SAVE              ; STORE THE DATA
         •
         •
         •
SAVE;    . = . + 1                    ; MEMORY LOCATION RESERVED FOR
                                      ; TERMINAL DATA
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
         Cycles:  21 if bit 0 set        (29.4μs)
                  25 if bit 1 set        (35μs)
                  13 if neither bit set  (18.2μs)
         Words:   6
```

Note: Execution time is based on 1.4μs/cycle.

Bit Masking

In the example of Table III the status word is inputted and bits are tested for zero via the Skip if AND is Zero (SKAZ) instruction. If any bit is set, data will be input from a peripheral and added to the partial sum in accumulator 2. Note that peripheral input and addition are done in one instruction which takes only five cycles (7μs).

Data Transmission

After assembling a message block, characters must be transferred to the host or communications line. Using load (LD) and store (ST) instructions an effective rate of 130k 8-bit chars/sec (1.04 Megabits/sec) is achieved. This requires a great deal of program storage and is therefore not efficient for large data transfers.

The program controlled loop in Table IV will provide an effective transfer rate of almost 80k characters/second while using only five memory words for instructions. Two 8-bit characters are read from memory and sent out. Up to 127 16-bit words (254 characters) could be transmitted in this fashion.

There are 18 cycles in the loop. To transfer 100 8-bit characters (50 words) takes 901 cycles or 1261.4μs. Thus one character is transferred in 12.614μs providing a transfer rate of approximately 79.4k chars/sec. This provides double the throughput of an 8-bit processor with equal cycle speed.

Data transmission may be speeded up at the expense of program storage. Placing more LD and ST instructions in the loop and decrementing accumulator 2 by the number of pairs of these instructions makes the loop control instructions less of a factor in the data transfer. The actual speed vs memory tradeoff is resolved by analyzing system throughput requirements.

Message Routing and Interrupt Handling

A message switch is responsible for assigning messages to communication lines or local terminals. Assume that each bit set in a message identification word corresponds to a message which is to be serviced by the corresponding line or terminal. Up to 16 messages are to be serviced in a prioritized fashion, with bit 0 assigned the highest priority. Coding is shown in Table V.

This routine may also be used for interrupt servicing with the Interrupt Select Status Word replacing MESGID (3). The ISCAN is an extended IMP-16 instruction which shifts bits out of accumulator one until a one is detected. The number of shifts is added to accumulator two and the next instruction is skipped if accumulator one is not zero. Otherwise, the next instruction is executed.

Error Control

Cyclic Redundancy Check (CRC) is a very efficient method of accounting for transmission errors in the data link. All single bit odd number errors, all double errors, all burst errors less than the length of the CRC word and most of the burst errors longer than the CRC word can be detected.

The algorithm and associated IMP-16 coding in *Figures 6a and 6b* implement a CRC process on parallel 16-bit data. The generation polynomial chosen for this example is $1 + x^5 + x^{12} + x^{16}$, and the initial conditions are defined to cause the CRC word to start with all 1's.

The time required to go through the main loop of the program is about 582μs for 16 bits; this works out to about 37μs per bit per check, comparing quite favorably with corresponding minicomputer rates. The 16-bit format of the IMP-16 microprocessor is largely responsible for the efficiency of the instruction set and the resulting speed of execution of programs.

MEASURING PERFORMANCE

A good yardstick for evaluating data throughput is provided by measuring the time required to transfer a byte (or other convenient unit of data) of information from a peripheral device to a system memory under control of the microprocessor. This operation involves addressing the device, addressing the memory location, executing the data transfer and updating the memory address. The resulting time can be expressed conveniently as a bit rate (kilobits or megabits per second).

The IMP-16 microprocessor has a throughput rate of 1.04 Megabits measured on this basis. As comparison, other microprocessors range from 75 Kbits (4-bit machines) to 840 Kbits (8-bit machines).

Table III

```
; TERMINAL CONTROLLER ADDRESS IN ACCUMULATOR 2
; PARTIAL SUM IN ACCUMULATOR 3
          LD       0, STATUS (2)           ; INPUT STATUS WORD
          SKAZ     0, MASK                 ; MASK & TEST FOR 0
          ADD      3, DATA (2)             ; ADD PERIPHERAL DATA TO AC3
          •
          •
          •
MASK:     . WORD   X'F51A                  ; MASK 9 BITS
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
          Cycles:  10 if masked bits are zero  (14μs)
                   17 if any masked bit is one (23.8μs)
          Words:   4
```

Table IV

```
NWORDS = NUMBER OF WORDS TO BE TRANSFERRED
OUT = ADDRESS OF HOST OR COMMUNICATIONS LINE
            LD      3, OUTADR     ; HOST ADDRESS IN AC3
            LI      2, NWORDS     ; NO. WORDS TO BE TRANSMITTED
LOOP:       LD      0, IN -1(2)   ; 16 BIT WORD FROM BUFFER MEMORY
            ST      0, OUT (3)    ; OUTPUT THE WORD TO HOST
            AISZ    2, -1         ; DECREMENT & TEST = 0
            JMP     LOOP          ; LOOP BACK
            •
            •                     ; TRANSMISSION COMPLETE
            •
            .BSECT
IN:         . = . + NWORDS        ; NUMBER OF WORDS IN BUFFER MEMORY
OUTADR:     . WORD X'4000         ; ADDRESS OF HOST COMPUTER
```

Table V

```
MESGID = MESSAGE ID WORD ORDER
; (OUTADR) IN ACCUMULATOR 3
            LD        2, ADTABL       ; ADDRESS OF POINTER TABLE IN AC2
            LD        1, MESGID (3)   ; INPUT MESSAGE ID WORD
CHECK:      ISCAN                     ; SCAN FOR BIT = 1
            JMP       NOMESG          ; NO MESSAGES LEFT
            JSR       @ (2)           ; SERVICE THE MESSAGE
            JMP       CHECK           ; LOOP BACK
NOMESG:     •
            •
            •
            •
            •
ADTABL:     . WORD    TABLE-1         ; ADDRESS OF POINTER TABLE
TABLE:      . WORD    LINE 1          ; ADDRESS OF LINE 1 SUBROUTINE
            •         •               •
            •         •               •
            •         •               •
            . WORD    LINE 16         ; ADDRESS OF LINE 16 SUBROUTINE
```

In conclusion, it can be seen that the characteristics of the IMP-16 make it adaptable to various communication environments. The results of 16-bit operations on 8-bit data is increased flexibility, performance and throughput. The potential and power of the IMP-16 can be used to serve a wide segment of the communication market.

```
SETUP:   COUNT  ←  16          ; INITIALIZE BIT COUNT
         CRC    ←  –1          ; INITIALIZE CRC WORD
         POLY   ←  $1021_{16}$ ; BINARY EQUIVALENT OF CRC POLYNOMINAL

LOOP:    (A) ← DATA(7) $\veebar$ CRC(15)
         CRC ← $\underset{0}{\uparrow}$ CRC
         IF A THEN CRC ← CRC $\veebar$ POLY
         DATA ← $\underset{0}{\uparrow}$ DATA
         COUNT ← COUNT –1
         IF COUNT ≠ 0, GO TO LOOP
         GET NEXT DATA WORD
```

FIGURE 6a. CRC Algorithm

```
CRCSHI
                              TITLE  CRCSHIFT

          0003     ODD        =       3
          0002     SEL        =       2
          000A     CYOV       =       10

                   ;          JSR     FCRC
                   ;          NORMAL RETURN           AC0=CRC IN 2 8 BIT BYTES.
                   ;          ON ENTRY :
                   ;          AC3= ADDRESS OF P(X).
                   ;          AC2= G(X).
                   ;          AC0= NUMBER OF BYTES +2 IN P(X).
                   ;          IF NUMBER OF BYTES ARE ODD
                   ;          FIRST BYTE IS IN BITS 8 TO 15.
                   ;          BITS 0 TO 7 ARE ZERO.

                   FCRC:
     0000 AD18 A              ST      3,POINT         ;STORE POINTER TO LIST.
     0001 1315 A              BOC     ODD,SEXTB       ;SET EXTRA BYTE.
     0002 4F10 A              LI      3,16            ;SET BIT COUNT.
                   LETSG:
     0003 5CFF A              SHR     0,1             ;MAKE WORD COUNT.
     0004 A115 A              ST      0,WORDC         ;STORE WORD COUNT.
     0005 4CFF A              LI      0,-1            ;GET INITIAL COND OF REG.
     0006 2101 A              JMP     .+2             ;FIRST TIME.
                   CRCBK:
     0007 4F10 A              LI      3,16            ;SET BIT COUNT.
     0008 9510 A              LD      1,@POINT        ;GET NEXT P(X) TERM.
     0009 790F A              ISZ     POINT           ;UPDATE ADDRESS.
                   CRC:
     000A 3000 A              RADD    0,0             ;SHIFT CRC WORD.
     000B 1A07 A              BOC     CYOV,CK1        ;BRANCH IF MSB IS A 1.
     000C 3500 A              RADD    1,1             ;SHIFT AC1 1 LEFT.
     000D 1A07 A              BOC     CYOV,XOR        ;IF MSB A 1 DO XOR.
                   OUT:
     000E 4BFF A              AISZ    3,-1            ;CHECK IF NEW WORD NEEDED.
     000F 21FA A              JMP     CRC             ;NO.
     0010 7D09 A              DSZ     WORDC           ;UP DATE WORD COUNT,SKIP IF DONE.
     0011 21F5 A              JMP     CRCBK           ;KEEP GOING.
     0012 0200 A              RTS                     ;DONE RETURN.
                   CK1:
     0013 3500 A              RADD    1,1             ;SHIFT LEFT 1.
     0014 1AF9 A              BOC     CYOV,OUT        ;IF MSB =0 DO XOR.
                   XOR:
     0015 3202 A              RXOR    2,0             ;NOW XOR.
     0016 21F7 A              JMP     OUT             ;GET NEXT.
                   SEXTB:
     0017 4F08 A              LI      3,8             ;SET BIT COUNT TO 8.
     0018 21EA A              JMP     LETSG

     0019 0000 A   POINT:     .WORD   0
     001A 0000 A   WORDC:     .WORD   0

          0000                .END
```

FIGURE 6b. CRC Coding

National Semiconductor Corporation
2900 Semiconductor Drive, Santa Clara, California 95051, (408) 732-5000/TWX (910) 339-9240
National Semiconductor GmbH
808 Fuerstenfeldbruck, Industriestrasse 10, West Germany, Tele. (08141) 1371/Telex 05-27649
National Semiconductor (UK) Ltd.
Larkfield Industrial Estate, Greenock, Scotland, Tele. (0475) 33251/Telex 778-632