ISSN-L: 2223-9553, ISSN: 2223-9944
Vol. 2, No. 3, May 2012

# MICROPROCESSOR-BASED REMOTE DATA ACQUISITION SYSTEM WITH AUTOMATIC BAUD-RATE DETECTION

**Prof. Hyacinth C. Inyiama**
Electronic and Computer Engineering Department
Nnamdi Azikiwe University
Awka, NIGERIA.
drhcinyiama@yahoo.com

**Christiana C. Okezie**
Electronic and Computer Engineering Department
Nnamdi Azikiwe University
Awka, NIGERIA.
christianaobioma@yahoo.com

**Nkolika O. Nwazor**
Research and Development Department, Electronic Development Institute Awka,
NIGERIA.
nkolinks81@yahoo.com

## ABSTRACT

*A Microprocessor-Based Remote Data Acquisition System with Automatic Baud-Rate Detection has been presented in this paper. The implementation featured an Intel 8085 Microprocessor as the controller and the 8251A Universal Synchronous and Asynchronous Receiver / Transmitter (USART) as a Programmable Communication Interface (PCI). Both hardware and Software aspects were covered and the technique for determining automatically, the baud-rate of a distant peripheral when that peripheral sends a space character, is also detailed in this presentation. This means that this data acquisition system always responds with the correct baud-rate when interacting with a distant data source and this makes for accurate data communication. Input / Output Multiplexing was featured to enable the data acquisition system cope with a variety of peripheral devices and an opto-coupler-based novel approach to peripheral device selection was used in the implementation.*

**Keywords:** Microprocessor, Remote, Data-acquisition, Multiplexing, Interfacing

## INTRODUCTION

A common feature of the peripheral devices used for Remote Data Acquisition is the use of serial data transmission and reception. That is a character byte is normally transmitted as a series of 1's and 0's which is re-assembled at the receiver. This is cost-effective compared to the exorbitant cost of transferring 8bits in parallel. However, only one serial input port and one serial output port are available in the 8085 Microprocessor system used in this project. Thus, only one of the peripheral devices may use this pair of serial 1/0 ports at any one time. Therefore, the serial output lines and serial input lines of the peripherals (when a number of them are in use) need to be multiplexed to the serial input line and serial output line respectively, of the 8085, to allow only the desired peripheral to take control of the serial 1/0 lines, at any given time.

The number of bits of information that may be transferred per second (baud rate) to or from a peripheral vary widely among peripherals, the standard baud rates being from 110 baud to 19, 200 baud. The serial communication software contained in the 8085 monitor is intended for a VDU or a Teletypewriter (TTY) terminal operating at 110 baud. It is desirable to provide for faster peripherals, a definite advantage when transferring large volumes of data via a costly data communication network. It is an advantage if the 8085 Microprocessor is able to detect automatically the speed of transmission of peripherals (other than the local VDU/TTY terminal and cassette recorder) and to respond at the detected baud rate. This option was taken in this application and it allows a wide range of peripheral devices to communicate with the data acquisition processor without any hardware speed adjustments by the end-user.

Line drivers and receivers are required to transmit/receive data on a line between a Microprocessor and terminals. These drivers convert standard TTL logic levels to communication levels before transmission while the line receivers convert from communication levels to TTL levels compatible with microprocessor circuits. The use of standardized communication levels allows any peripheral equipment compatible with the standard to communicate with the data acquisition processor. The

standard communication level used in this application is the Electronic Industries Association (EIA) RS- 232 standard for serial data transmission systems [1].
The rest of this paper is divided into three sections namely:

a. The programmable communication interface to the remote data source.
b. The hardware interfaces to VDU, teletype and cassette recorder, including the logic circuit which converts between TTL and EIA standards; and
c. The multiplexing of the serial peripheral devices to the single pair of serial I/0 lines of the 8085 Microprocessor used to achieve Remote Data Acquisition.

The remote site first sends a space character. The 8085 Microprocessor receives the character into its Serial Input Data (SID) line via the phone modem. A software application in the 8085 uses the space character to estimate the baud rate in use at the remote data source that sent the space character. Thereafter, the 8085 data acquisition processor programs the USART to communicate with the remote terminal with the computed baud rate.

Serial data from the remote data source are directly received by the USART via the phone modem. The USART converts each serial bit stream into an 8-bit character byte which is then passed on to the 8085 Microprocessor. The 8085 data acquisition will then pass each received character byte to either the tape drive or to the VDU or to the teletypewriter. This sequence of operations continues until all the data from the remote data source is received. Similarly any information or feedback from the 8085 microprocessor to the remote data source is passed one byte at a time from the 8085 to the USART. The USART serializes each byte and passes its bit-stream to the remote data centre via the phone modem.

The keyboard interfaced to the 8085 permits data entry at the near end while the LCD or VDU serves to display the baud rate in use or other status information about the USART, peripherals or remote data sources.

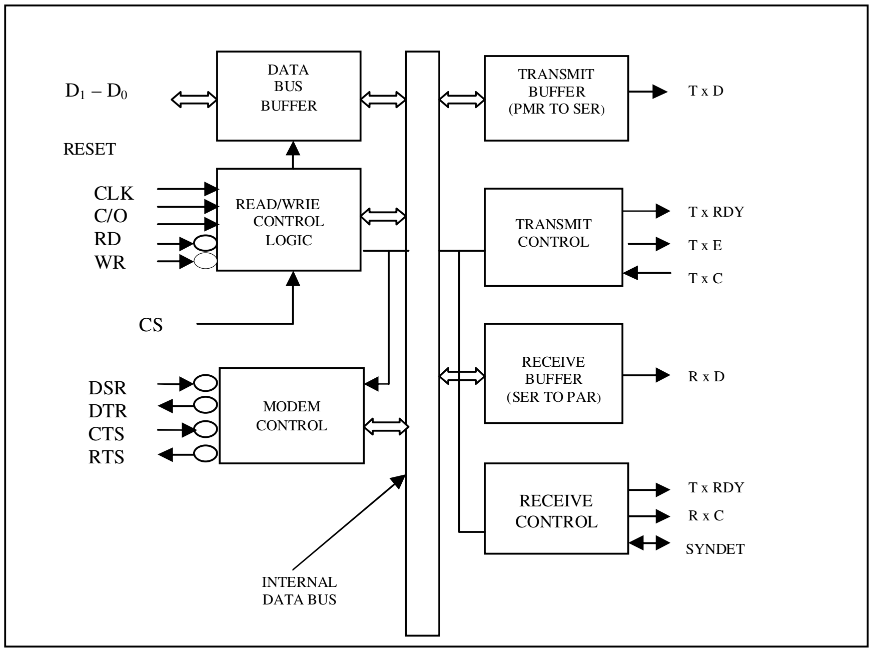


*Fig 1 Block Diagram of the 8251A USART [2]*

ISSN-L: 2223-9553, ISSN: 2223-9944
Vol. 2, No. 3, May 2012

**Table 1: Pin Functions Of The Usart [2]**

| | |
|---|---|
| D7 | DO DATA BUS |
| DSR | DATA SET READY |
| C/D | CONTROL OR DAYA BYTE |
| DTR | DATA TERMINAL READY |
| RD | READ DATA |
| SYNDET | SYNC/BREAK DETECT |
| WR | WRITE DATA |
| RTS | REQUEST TO SEND DATA |
| CS | CHIP SELECT |
| CTS | CLEAR TO SEND DATA |
| CLK | CLOCK PULSE (TTL) |
| TXE | TRANSMITTER EMPTY |
| RESET | RESET SIGNAL |
| VCC | +5 VOLTS |
| TXC | TRANSMITTER CLOCK |
| GND | GROUND |
| TXD | TRANSMITTER DATA |
| RXD | RECEIVER DATA |
| RXC | RECEIVER CLOCK |
| RXCDY | RECEIVER CHARACTER READY |
| TXCDY | TRANSMITTER READY FOR NEW CHARACTER |

## Programmable Communication Interface

The Intel 8251A Programmable Communication Interface (PCI) used to interface to the remote data source is the enhanced version of the industry standard, Intel 8251 Universal Synchronous/ Asynchronous Receiver/Transmitter (USART), compatible with the 8085 microprocessor. The 8251A USART can be programmed to accept data from the CPU in a parallel format and to convert them into a continuous serial data stream before transmitting them. Simultaneously the 8251A can receive serial data streams and convert them into parallel data characters for the 8085 CPU. The USART will signal the CPU whenever it is ready to accept a new character for transmission or whenever it has received a new character for the CPU. The CPU can read the status of the USART at any time, including data transmission control and error signals.

FIG 1 is a block diagram of the 8251A USART showing its essential features. The pin functions are summarized in Table 1 while Table 2 shows the data-direction control signals. The Read and Write (RD and WR) of the 8085 are applied directly to the RD and WR inputs to the USART chip. The two methods used to pass data from a peripheral device to a microprocessor are I/O-mapped-I/O and Memory-mapped-I/O respectively. I/O-mapped-mapped-I/O makes the peripheral device appear as an I/O port address to the processor. Typically only 8-bits are used to address an I/O port. Memory-mapping involves making the peripheral device appear like a memory location anywhere in the full address space of the processor. This type of interfacing often calls for the full decoding of the address bus which is longer than an I/O port address used in I/O-mapped-I/O. However, only IN and OUT instructions can be used to operate a peripheral when I/O-mapped-I/O is used, whereas the full power of the processor's instruction set can be used to interact with a peripheral when memory mapping is in use.

It was convenient to use I/O-mapped-I/O (involving the use of a less complex port-address decoder compared to memory-mapping) to interface the USART to the 8085 as the full power if the 8085 instruction set (available when memory mapping is used) is not required. The IN and OUT instructions of the 8085 are sufficient to transfer data between the 8085 and the 8251A USART. Table 3 shows the I/O port addresses for the USART, chosen such that when the address line A3 is applied directly to the C/D input of the USART the conditions specified for the control signals (Table 2) are met. When the "IN F6" command is executed by the 8085, the USART receiver data is read. An "OUT 6" command effects data transfer from the 8085 accumulator to the USART transmitter. In both cases address line A3 (and hence C/D) is a logic 0.

**Table 2: Usart Control Signals**

| NAME C/D | RD | WR | CS | | FUNCTION |
|---|---|---|---|---|---|
| DATAIN | 0 | 0 | 1 | 0 | Read the receiver data |
| DATOUT | 0 | 1 | 0 | 0 | Output data to transmitter |
| STATIN | 1 | 0 | 1 | 0 | Read the status word |
| MODCMD | 1 | 1 | 0 | 0 | Output mode/command word |
| | X | X | X | 1 | USART not selected |
| | X = | DON'T CARE | | | |

**Table 3: Usart I/O – Port Assignments**

| | | A3=C/D | RD | WR | CS | PORTS |
|---|---|---|---|---|---|---|
| DATAIN | A7 – A4, A2 – A0 | A3 | 0 1 | 0 | 6F | |
| DATOUT | ALL LOGIC 1's | 0 | 1 | 0 | 0 | 6F |
| STATIN | | 1 | 0 | 1 | 0 | 7F |
| MODCMD | | 1 | 1 | 0 | 1 | 7F |

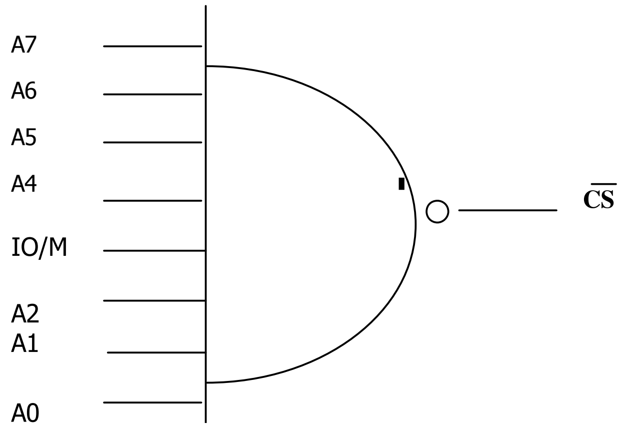


***FIG 3 Chip Select Logic for Mapping the 8251a USART to Ports 6f*** *And 7f*

Similarly, for "IN F7" (Read USART status word) or "OUT F7" (OUTPUT USART Mode/Command Word) A3 (and hence C/D) is a logic 1 as required. The signal line indicating whether the address referenced by the 8085 is an I/O port or a memory location (i.e. IO/M) is logic 1 only when an IN or an OUT instruction is being executed by the 8085. Thus by decoding when IO/M = 1 and the bit pattern on address lines $A_0$ – $A_7$ corresponds to F6 or F7, and by applying the decoder output to the Chip Select (CS) input of the USART (FIG 3), the 8251A is effectively mapped I/O ports F6 and F7. Where the remote data source is located within 100 feet of the USART, a twisted pair of coaxial cables may be used to transfer information between the USART and the data source after the signals had been transformed to RS – 232 standard levels. The telephone or GSM network may also be used as a data link where the distance between the remote data source and the USART justifies this. Figs 4 and 5 illustrate the two options available when using the telephone network for data transfers. The four control signals of the USART, namely, Data Set Ready (DSR), Data Terminal Ready (DTR), Request to Send (RTS) and Clear To Send (CTS) serve as handshake signals between the USART and Communication interface units called Modems. DSR is a general purpose input signal to the USART which information indicates when the Data Set is ready. It can be tested by the CPU using a Status Read Operation. DTR is a 1 – bit inverting output port which can be programmed by software. It is normally used to indicate when the Data Terminal is ready or as a rate select.

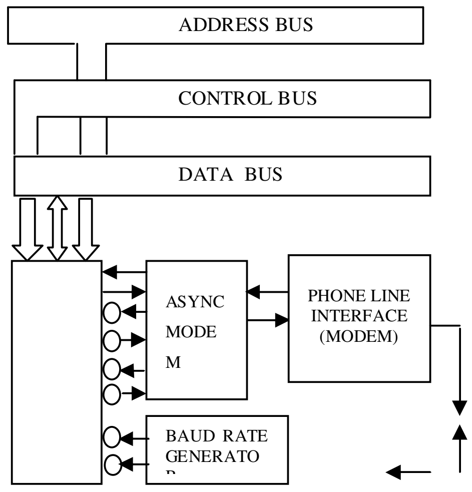


FIG. 4 ASYNCHRONOUS INTERFACE TO TELEPHONE LINES [3]

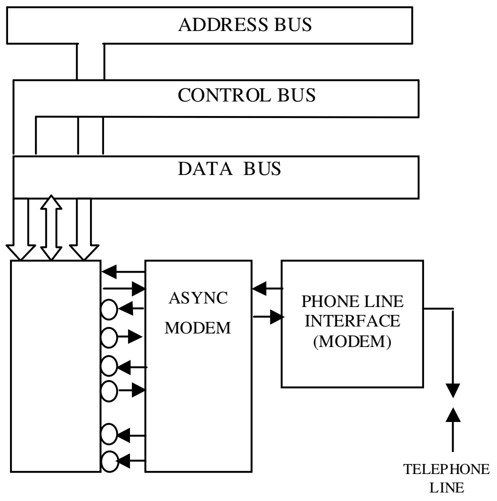


FIG. 5 ASYNCHRONOUS INTERFACE TO TELEPHONE LINES [3]

**Table 4. The Clock Frequencies Required For Different Bit Rates And Baud Rate Factors [3]**

| | **Clock Frequency Required for Baud Rate Factors** | | |
|---|---|---|---|
| **Bit Rate** | **X1** | **X16** | **X64** |
| 110 | 110 HZ | 1.76 KHZ | 7.04 KHZ |
| 150 | 150 HZ | 2.4 KHZ | 9.6 KHZ |
| 300 | 300 HZ | 4.8 KHZ | 19.2 KHZ |
| 600 | 600 HZ | 9.6 KHZ | 38.4 KHZ |
| 1,200 | 1,200 HZ | 19.2 KHZ | 76.8 KHZ |
| 2,400 | 2,400 HZ | 38.4 KHZ | 153.6 KHZ |
| 4,800 | 4, 800 HZ | 76.8 KHZ | 307.2 KHZ |
| 9,600 | 9, 600 HZ | 153.6 KHZ | 614.4 KHZ |
| 19,200 | 19, 200 HZ | 307.2 KHZ | 1,228.4 KHZ |

The RTS output signal is used by the transmitting station (i.e. the USART) to indicate readiness to transmit data. The receiving station forces the CTS input low, when it is ready to receive data. Communication Modems Compatible with the 8251A USART Modem is readily available [4].

The USART has three programmable baud rate factors namely x1, x16, and x64, which divides the transmitter and receiver input clock frequencies internally to obtain a desired data transfer baud rate. The various combinations of input frequencies and baud rate factors which yield the standard baud rates are shown in Table 4. In an application where the baud rate may vary from time to time depending on the peripheral device currently in use, a means must be provided for selecting one of a number of alternative input frequencies to be applied to the Transmitter clock (TXC) and receiver clock (RXC) inputs of the USART. The selected frequency must be at least four times less than the frequency applied to the clock (CLK) input of the USART for asynchronous communications and 32 times for synchronous communication [3]. The RXC and TXC frequencies may differ if data transmission rates are different from the reception rate for the same data source/destination. This is uncommon and an unnecessary sophistication, therefore provision is made here for the situation where RXC is always equal to TXC for a given data source/destination. Fig 6 illustrates how the 8085 system clock may be used to derive the alternative RXC and TXC frequencies. The 3.072 MHZ clock of the 8085 is divided by 2 to obtain a 1.536 MHZ output. Both clocks (3.072 MHZ and 1.536 MHZ) serve as the primary clock sources to a frequency divider chain comprised of two 4 – bit counters 974LS90) and 74LS93. A1 – of – data selects either the 3.072 MHZ primary clock or the 1.536 MHZ frequency derived from it, and applies the selected frequency to the clock input of the divider network producing five different outputs, namely F/10, F/20, F/40, F/80 and F/160, where F is the selected input to the divider chain. The illustration shows how a 1 – out-of – 8 data selector may be connected to select one of the five outputs by means of a 3-bit code from the 8085 microprocessor.

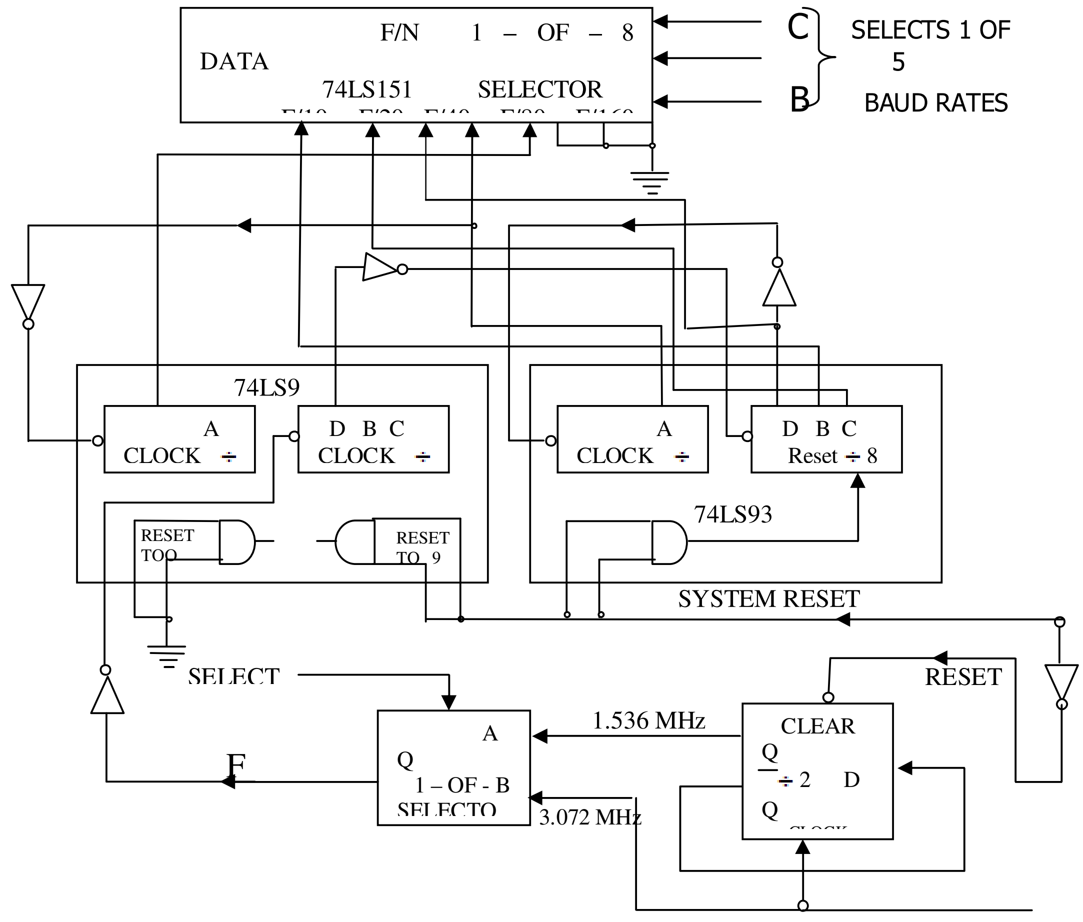


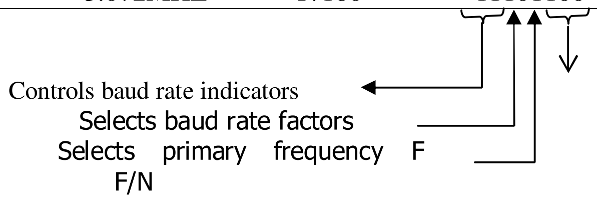
**FIG 6: AUTOMATIC BAUDRATE SELECTOR LOGIC** 8085 CLOCK

The selected output is applied to the R X C and T X C inputs of the USART. The combinations of primary clock source, selected output of the divider chain, and baud rate factors which yield the standard baud rates are summarized in Table 5.

When the source/destination of data is an equipment other than the local VDU/Teletype terminal, the user has the option either to specify the desired baud rate for the data transfer by inputting one of the user-input codes shown against the baud rates in Table 5 or by inputting a one – character code 'C' (meaning calculate which instructs the microprocessor to estimate automatically the baud rate used at the distant station. When the 'C' option is chosen the selected peripheral must initiate the data transfer by sending a space character after the "Ready" message from the 8085. The bit pattern of the space character and the resultant waveform at the serial input port of the 8085 is shown in Fig 7. The 8085 estimates the time interval from the falling edge of the waveform to its rising edge. This corresponds to the time taken to transmit the number of bits of the low level portion of the waveform (6 bits long) and is used to estimate the transmission speed in number of bits per second (baud rate). In this implementation, whenever the estimated baud rate does not match one of the standard baud rates in Table 5 exactly, the closest standard baud rate to the estimated value is assumed.

**Table 5. Baud Rate Control Scheme**

| INPUT MODE | BAUD RATE SELECTED | BAUD RATE FACTOR | PRIMARY FREQUENCY F | F/N | CONTROL BIT PATTERN GENERATED |
|---|---|---|---|---|---|
| 0 | 150 | X64 | 1.536MHZ | F/160 | 00010100 |
| 1 | 300 | X64 | 3.072MHZ | F/160 | 00111100 |
| 2 | 600 | X64 | 3.072MHZ | F/160 | 01011011 |
| 3 | 1200 | X64 | 3.072MHZ | F/160 | 01111010 |
| 4 | 2400 | X64 | 3.072MHZ | F/160 | 10011001 |
| 5 | 4800 | X64 | 3.072MHZ | F/160 | 10111000 |
| 6 | 9600 | X1 | 1.536MHZ | F/160 | 11000100 |
| 7 | 19200 | X1 | 3.072MHZ | F/160 | 11101100 |


Controls baud rate indicators
Selects baud rate factors
Selects primary frequency F
F/N

It is a advantageous and would facilitate fault-tracing if during operation the 8085 always indicates the current mode of operation, that is: the baud rate in use, baud rate factor programmed into the USART, and the primary frequency input to the divider chain. Since only 3 bits of a control byte are required to select one of

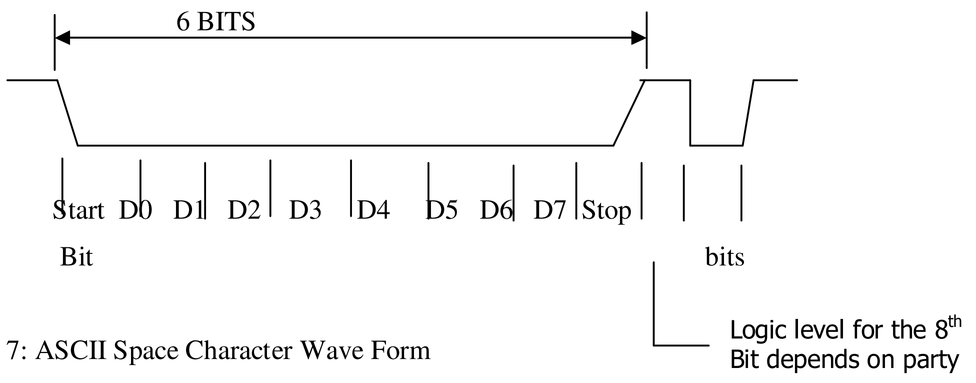


Fig 7: ASCII Space Character Wave Form

the five frequency divider chain outputs to be used as the current R X C and T X C inputs to the USART it is convenient to provide the mode information using the remaining 5 bits of the control byte. Three of the five bits would suffice to indicate one of the 8 possible standard baud rates when decoded, one bit is required for the baud rate factor and the 5th bit would serve as the primary frequency indicator (a logic 1 = 3.072 MHZ and logic 0 = 1.536 MHZ). This bit assignment summarized in Table 5 was used in this implementation while software details are shown in Table 6. All status indications are displayed on the liquid crystal display (LCD) panel of Fig 1. More information on the uses of the 8251A USART are readily available [2]

## VDU, TELETYPE AND MAGNETIC TAPE INTERFACES

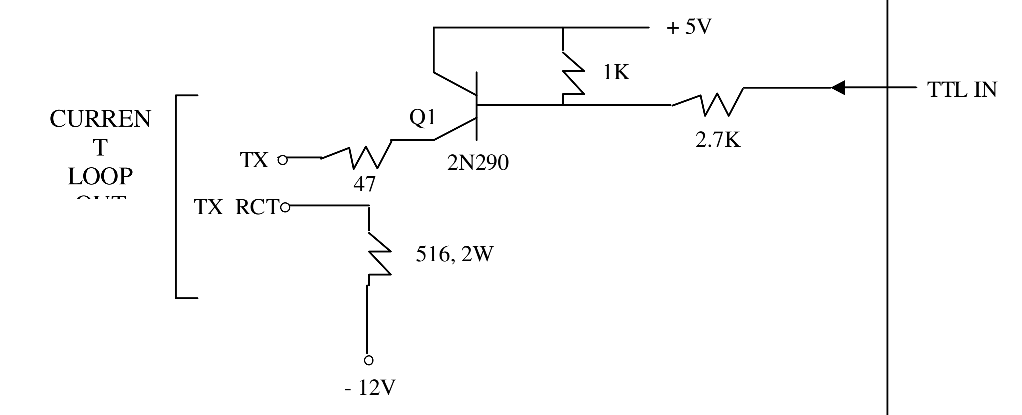


**FIG 8: MICROPROCESSOR – TELETYPE INTERFACE**

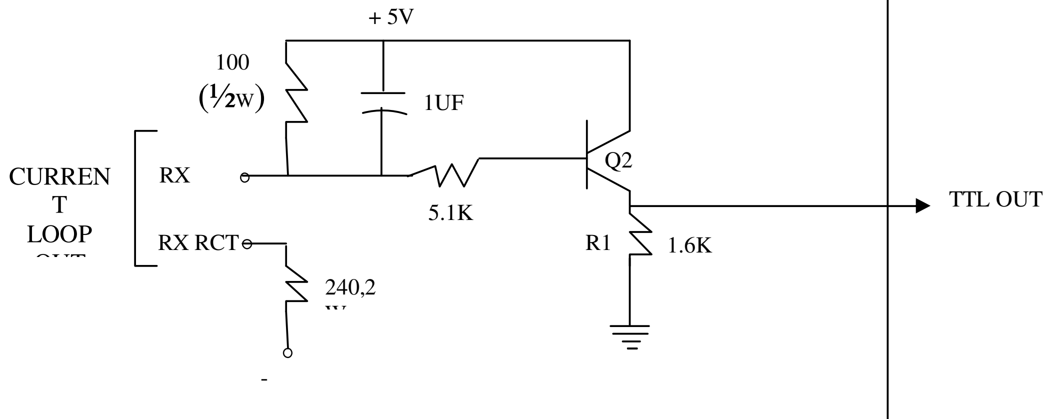


**FIG 9: TELETYPE – MICROPROCESSOR INTERFACE**

The next few paragraphs review briefly the electronic circuits which enable peripheral devices such as Teletypewriters, Cassette (or Magnetic Tape) recorders, and VDU's to communicate with the microprocessor.

The Teletypewriter (TTY) terminals in common usage communicate via a standard 20mA current loop. That is, their input and output logic 1 and logic 0 levels correspond to the presence or absence of 20mA current respectively. Therefore, additional interface circuits are required to translate the 20mA current loop into TTL logic levels compatible with the processor. FIG. 8 illustrates how a single switching transistor may be connected as a TTL – TO – 20mA level converter for data transmission from the microprocessor to a TTY or any other 20mA current-loop equipment. The component values shown match the power supplies indicated (i.e. +5V and -12V). In this implementation, a logic 0 TTL level causes Q1 to turn ON, allowing a 20mA current (whose loop is completed inside the TTY equipment) to flow via transmit (TX) output and back via the TX RETURN (TX RET) line. A logic 1

TTL input cuts off Q1. For the data path from the teletype to the microprocessor (FIG 9), Q2 is out off when the (2omA current loop is open thus allowing a logic 1 TTL output to reach the serial data input of the processor. Q2 and its biasing arrangement are chosen such that when Q2 conducts (20mA loop closed), most of the 5V supply voltage is dropped across Q2 and only a small current flows through R1 leading to a logic 0 TTL OUT level to the processor.

8085 system manufacturers supply a compatible cassette recorder interface circuit. Data is transmitted with each bit composed of a tone burst, followed by a pause. The first third of a bit period is always a tone burst, the mid a pause corresponding to, respectively a one or zero, and the final third is always a pause, as shown in FIG 10. The interface circuit (FIG 11) is comprised of one LM 325 quad operational amplifiers (op-amp) and a few standard value discrete components. On playback, A2 buffers the incoming signal, and A3 inverts it. The peaks of these two signals are transmitted through D1 or D2 and are filtered by an RC network. Comparator A4 then squares up the output and produces the logic signal read by the microprocessor. Since the op-amps are powered by the single 5 volt supply, a 2.0 volt reference level is obtained from a resistive voltage divider.

The microprocessor interprets the incoming data as follows: if the tone burst is longer than the pause, the bit is one otherwise it is zero. Since only the time ratio is considered, any variation in tape speed will not affect data determination.

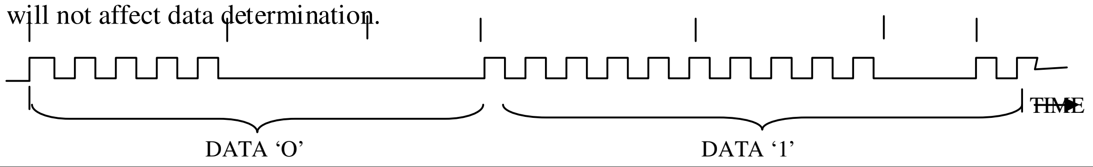


**FIG 10 TAPE INTERFACE DATA RECORDING SCHEME (8)**

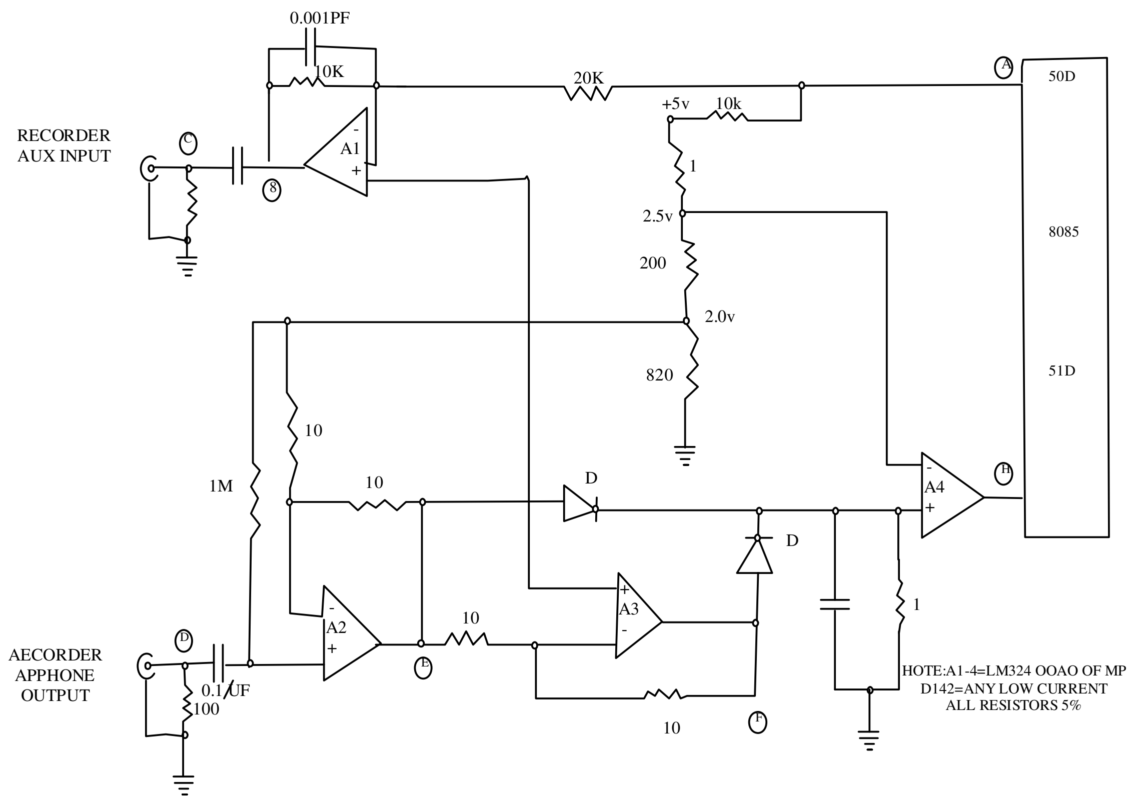


**FIG 11 ONE CHIP MAGNETIC TAPE INTERFACE SCITEMATIC (8)**

Software details for cassette data retrieval and storage are given in Table 6.

**Table 6A: Subroutine Bdrate**

| | | |
|---|---|---|
| BDRATE | CALL BRID: | ; Estimate the bit time used by the distant Peripheral |
| | | ; Shift the rang ± 10 to between 0 and 20 by adding 10 to the |
| | LXI D, AH | ; computed bit Time 10 = A (Hex). |
| | DAD D | ; Add 10 in DE to Bit Time in HL. |
| | XCHG | ; Transfer adjusted Bit Time from HL to DE |
| | MVI C, 7 | ; Initialize register C to the maximum number of entries in BITABL. |
| | LXI H BITABL | ; Point to the first entry of the Table of Bit Times with HL |
| BDLOOP: | PUSH D: | ; Save adjusted bit time on the stack (ie the Bit Time computed by |
| | MOV A, E | BRID) |
| | SUB M | ; Put low order byte of adjusted Bit Time into A |
| | | ; Subtract low order byte of Bit Time in Table from low order byte |
| | MOV E, A | ; of computed Bit Time (E) |
| | INX H | ; Put result back into E |
| | MOV A, D | ; Point to high order byte of Bit Time in BITABLE |
| | SBB M | ; Put high order byte of Bit Time computed by BRID into A |
| | MOV D, A | ; Subtract high order byte of computed Bit Time in Bitabl |
| | | ; and the current setting of the borrow flag from previous |
| | CALL CHECK | ; subtraction. Put result back into E |
| | POP D | ; Check if remainder in DE is within 0 to 20 and set carry flag if so. |
| | JC THERE | ; Retrieve the adjusted Bit Time into DE as it was before subtraction |
| | INX H | ; If carry is set, a match has been found in BITABL for the compute |
| | DCR C | Time |
| | JNZ BDLOOP | ; If carry not set point to the next Bit Time entry in BITABL |
| | | ; Decrement count of entries |
| | MVIA, 15 | ; If not end of BITABL go back and check for a match computed |
| | CALL ERRORX | TIME, |
| | JMP CODE | ; else a non – valid baud rate was estimated by BRID. |
| THERE: | LXI H, | ; Put error code for INVALID BAUD RATE into register A |
| | CDTABL | ; Output error message to user |
| | MVI A, 7 | ; Abort current command and go for another command code |
| | SUB C | ; Point to baud rate programming code Table byte 1. |
| | | ; Put maximum number of entries in BITABL into S |
| | MVI D, O | ; Subtract remainder in C to get index into Table of USART baud rate |
| | MOV E, A | ; programming codes |
| | DAD D | ; Put index into register pair DE |
| | MOV A, M | ; |
| | STA BDCODE | ; Add DE to HL pointer to index baud rate code Table |
| CHECK: | RET | ; Put the baud rate into A |
| | XRA A | ; Store it in location BDCODE for future use |
| | ORA D | ; Return to calling routine |
| | JNZ N0 | ; Zero register A and compare with register D – higher byte of remain |
| | MVI A, 14 H | ; If high order byte of remainder 10 then remainder > 20 |
| | CMP E | ; If remainder > 20 go and reset carry flag |
| | JM N0 | ; Put 20 decimal (14 Hex) into A |
| | STC | ; Compare with low order byte of remainder |
| N0: | RET | ; If low order byte of remainder > 20 then go and reset carry flag |
| | STC | ; Else the estimated baud rate matches that in BITABL pointed to by |
| | CMC | ; Set carry and return to calling routine |
| | RET | ; Make sure carry is a 1 |
| | | ; Then reset it to zero |
| | | ; Return to calling routine. |

ISSN-L: 2223-9553, ISSN: 2223-9944
Vol. 2, No. 3, May 2012

**Table 6B: Subroutine Modcmd**

| | | |
|---|---|---|
| MODCMD: | MVI A, 40 H | ; Set bit 6 of register A |
| | OUT 7 FH | ; Output to Internal Reset pin of USART to return it to Mode/Comn |
| | | ; programming state |
| | LHLD MODE | ; Put Mode Word for X64 into register H and that for X1 into regist |
| | LDA BDCODE | ; Retrieve the Baud rate control pattern into A |
| | ANI 10H | ; Clear all bits but bit 4 which defines X64 or X1 |
| | MOV A, L | ; Put Mode Word for X1 into A |
| | JZ OUTPUT | ; If bit 4 of BDCODE is zero MODE WORD for X1 is used |
| | MOV A, H | ; Else Mode Word for X64 is used |
| OUTPUT: | OUT 7FH | ; Set USART Mode |
| | MVI A, 15H | ; Put the Command Word which clears all USART errors and enabl |
| | | ; transmitter and receiver into Register A |
| | OUT 7F | ; Output Command Word to USART |
| | RET | ; Done |

**Table 6C. Subroutine Trsmit**

| | | |
|---|---|---|
| TRSMIT: | PUSH PSN | ; Save Data Byte Now In Register A On The Stack |
| TRS2: | IN7FH | ; Input USART Status |
| | ANI 1 | ; Mask out all but the TX ready bit |
| | JZ TRS2 | ; If Not Set Wait Until TX (Transmitter) is ready |
| | POP PSW | ; Retrieve Data Byte From The Stack |
| | OUT 6FH | ; Output Data Byte Via USART |
| | RET | ; Done |

**Table 6D: Subroutine Recvr**

| | | |
|---|---|---|
| RECVR: | IN 7FH | ; Input Usart Status |
| | ANI 2 | ; Mask out all but Rx (Receiver) ready bit |
| | JZ RECR | ; If Not Set Wait Until RX is Ready |
| | IN 6FH | ; Input data byte from USART |
| | RET | ; Done. |

**Table 6E: Subroutine Blkrcd**

| | | |
|---|---|---|
| BLKRCD: | MVI A, CSETON | ; Output Code to select Cassette Interface: CSETON = 01 |
| | OUT 9 | ; Output to peripheral multiplexer |
| | MVI C, LEADER | ; Set up LEADER burst length (250 decimal) |
| | MVI A, CH | ; Set Accumulator (register A) to result in burst |
| | CALL BURST | ; Output tone, i.e a LEADER bit |
| | DCR C | ; Decrement LEADER bit count |
| | JNZ BRl | ; If not zero, go back and finish |
| BRl | XRA A | ; Clear accumulator and output space so that |
| | CALL BURST | ; the start of the first data byte can be detected |
| | MVI D, O | ; Initialize byte count in block to maximum: 0 = 256 |
| | CALL OUTAPE | ; Go and output the 256 data bytes to tape |
| | LXI H, CRCLOW | ; Point to Low order byte of CRC-16 check |
| | MVI D, 2 | ; Digits for the block transmitted and specify no of CRC bytes on ta |
| | RET | ; Done |
| OUTAPE: | MOV C, M | ; Get next data byte from buffer pointed to by HL |
| | CALL TAPEO | ; Record data byte on tape |
| | INX H | ; Point to next data byte with HL |
| | DCR D | ; Decrement number of byte to record |
| | JNZ OUTAPE | ; If not zero, go back and finish |
| | RET | ; Done, return to calling routine |

Table 6F: Subroutine Playbk

| | | |
|---|---|---|
| PLAYBK: | MVI A, CSETON | ; Select Cassette Interface |
| | OUT 9 | |
| | MVI C, LDRCHK | ; LDRCHK = 250. Successive highs must be |
| PBl: | CALL BITIN | ; read to verify that the LEADER is present |
| | JNC PLAYBK | ; and electronics has stabilized |
| | DCR C | ; End of Leader? |
| | JNZ PBl | ; If not continue reading LEADER bits |
| | MVI D, O | ; Initialize D to maximum number of bytes in a block |
| | CALL PB2 | ; Go and input 256 bytes into memory |
| | MVI H, | ; Point to low order byte of CRC to buffer location |
| | CRCLOW | ; Initialize D to the number of CRC bytes |
| | MVI D, 2 | ; Go and input 2 CRC bytes into CRCLOW and CRCHIGH |
| | CALL PB2 | ; Perform CRC – 16 check on received data including CRC bytes. |
| | CALL CRCHEK | ; Output error message inside CRCHEK if error and abort comn else |
| | RET | ; Return to calling routine |
| | CALL TAPEIN | ; Get data from Recorder |
| | MOV M, C | ; Store in memory |
| PB2: | INX H | ; Advance memory pointer |
| | DCR D | ; Decrement count of bytes to record |
| | JNZ PBZ | ; If not zero, go back and finish |
| | RET | ; Return to calling routine |

Table 6G. Subroutines Blkout And Blkin

| | | |
|---|---|---|
| BLKOUT: | LXI H, OUTADR | ; Point to Table of Output Routine Addresses |
| | JMP IOCASE | ; Jump to program section shared with BLKIN |
| BLKIN: | LXI H, INPADR | ; Point to Table of Input Routine Addresses |
| IOCASE: | PUSH H | ; Save the selected Table's address on the stack |
| | MVI C, O | ; Clear count of Peripheral Codes in PERIPH |
| | LXI H, PERIPH | ; Point to Peripheral Code Table bytes 1 |
| | LDA IOSWCH | ; Retrieve the Code for the user-specified peripheral |
| IOCAS2: | CMP M | ; Compare with content of current PERIPH location |
| | JZ IOCAS3 | ; If the same go to form address of peripheral I/O routine |
| | INX H | ; Else point to the next peripheral code entry in PERIPH |
| | INR C | ; Increment count of codes processed so far |
| | MOV B, A | ; Save peripheral code specified by user in B |
| | MVI A, 3 | ; Check if the last peripheral code in PERIPH |
| | CMP C | ; has been compared with user specification |
| | JNZ IOCAS2 | ; If not, go back and continue searching |
| | MVI A, 10H | ; Else put Error Code for "INVALID PERIPHERAL |
| | CALL ERRORX | ; CODE" into A and go and output Error message to user |
| | JMP CODE | ; Go back for another User Command and abort this one |

A local terminal may be either a VDU or a teletypewriter, so, some means is needed to switch the serial data path from the processor between the corresponding interfaces. It is also desirable to have a flexible situation permitting a data source/destination other than the local terminal) to communicate by means of either TTL level (e.g. a TTY interface) or RS-232 levels (as for VDUs) and to cause the data path from the processor to be routed automatically via the peripheral interface circuit which handles the transmission level being used.

Fig 12 illustrates an arrangement which satisfies these requirements. The user indicates a choice of local terminal (i.e. VDU or TTY) by means of a custom made solid state opto-switch. When the plunger shown in the illustration is pressed down (TTY selected) it blocks the LIGHT PATH OF A slotted opto-switch producing a logic 1, TTY SELECT signal and a logic 0 VDU SELECT signal. Conversely when the VDU is selected (plunger in the "UP" position), the VDU SELECT line goes to a logic 1 while the TTY SELECT becomes a logic 0. The SELECT signals operate the quad bilateral analog switch (4066) which connects either the TTY serial output data line to the processor according to which SELECT (control) signal is currently a logic 1. Simultaneously, and by means of the SELECT control signals, the quad bilateral switch connects the serial output data line from the processor to the serial data input line of the selected device. All data

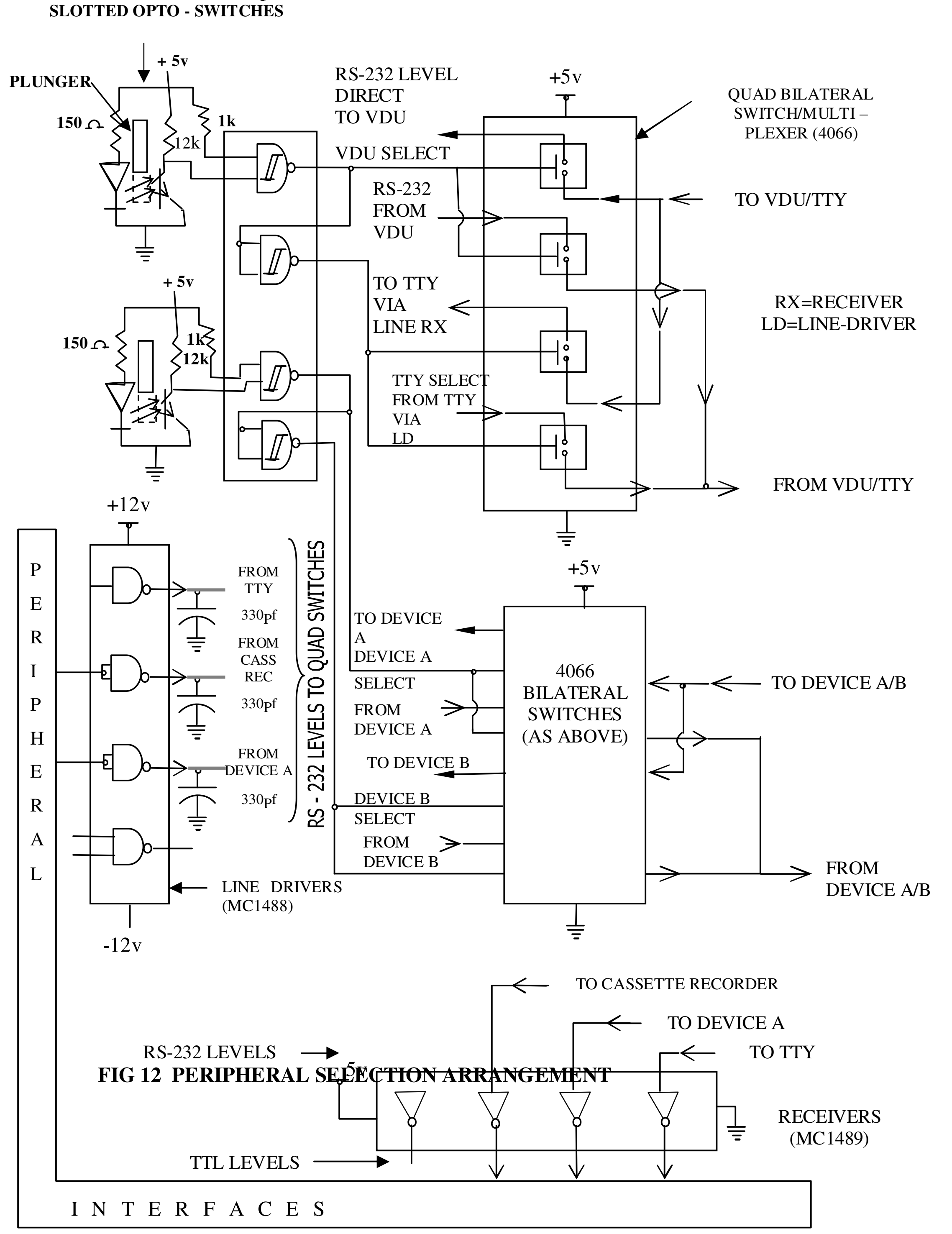


FIG 12 PERIPHERAL SELECTION ARRANGEMENT

transmission is at standard RS-232 levels. Thus, the TTL levels from a Teletype interface are first passed via a line driver (MC 1488) which produces an RS-232 level from a TTL input. The output of the MC1488 is then applied to the quad bilateral switch as the TTYserial output data line.

A similar arrangement is used to select either device A (using a TTL interface) or device B (already at RS – 232 level like the VDU) when data is transferred between a source/destination other than the local terminal, and the CPU, leading to a second pair of serial I/O lines for devices A and B. The third pair of serial I/O lines shown in the illustration (Fig. 12 applies to the cassette recorder interface whose I/O lines at TTL levels need transformation to RS – 232 levels as for the TTY interface.

Incoming signals to the TTY, Cassette recorder, and Device A interfaces are RS – 232 levels and need to be translated down to TTL levels by means of a line receiver (MC 1489) as shown.

## MULTIPLEXING OF SERIAL PERIPHERAL I/O LINES

Four serial output lines, one from each of VDU/TTY, Device A/Device B, Cassette Recorder, and the remote data system need to be multiplexed to the single serial input Data (SID) line available on the 8085. Considering that the 8215A USART used to communicate to the remote data source uses an I/O mapped-I/O interface which permits the use of I/O ports and the 8085 data bus for data transfers, it may not be apparent why the serial data from the remote data source needs to be multiplexed to the 8085 SID line along with those of other peripherals. This multiplexing is necessary because the USART plays a dual role in this application: first, as a means of communicating with the remote data system and secondly as a transparent data source/destination for peripherals other then those at the local terminal. This arrangement enables the processor to communicate to such peripherals by programming the desired baud rate into the USART and using the USART for all data transfers involving any of these peripherals.

In-coming signals from the peripheral devices including the remote data source are translated from the standard RS – 232 levels used in transmission, to TTL levels compatible with the processor, by means of a line receiver, the MC1489 (FIG 12). The outputs of the line receiver chip are opto-coupled to the microprocessor using a quad opto-isolator and a quad 2-input NAND Schmitt trigger (F2). The AND gates in the quad 2-input AND buffer (7048) connected to the outputs of F2 as shown, perform an OR logic function to provide a single serial input to the SID line of the processor. With this arrangement, only one of the peripheral devices may be active at any one time.

The data applied to the SID line of the processor is also applied to the Receiver Data (R X D) input of the USART. When the USART is not enabled, this input has no effect on it. The processor enables the USART only when a peripheral device other than a local terminal (VDU, TTY or Cassette Recorder) is specified by the user. This allows data transfers from the remote data source, Device A, Device B to the processor to be handled by the USART.

The processor is thus enabled to use the same input/output software drivers for all external peripherals, leading to a significant reduction in software costs.

The information intended for an external peripheral is transferred in parallel by the processor via the USART's data-bus interface. The USART converts the parallel data into serial data and produces an output data stream at the programmed baud rate. A two-bit control pattern from the microprocessor routes the T X D serial output of the USART to the desired destination by means of a dual 1 – of – 4 decoder/demultiplexer and a quad 2-input NAND-buffer Table 7 shows the selected device for each combination of the control bits. The NAND-gates N1 – N4 correspond to VDU/TTY, Cassette Recorder, USART/remote data source, and Device A /Device B, respectively. One input of each NAND gate is connected to the demultiplexer output which becomes active when the corresponding peripheral device is selected. The USART Transmitter Data is connected to the second input lines of N3 and N4 which communicate to peripheral devices other than the local terminal and Cassette Recorder. The second input lines of N1 and N2 which correspond to the local (VDU/TTY) terminal and Cassette Recorder respectively, are fed from the serial output Data (SOD) line of the processor Via a 7408 buffer. This allows the use of the 110 baud rate software available in the 8085 monitor for VDU/TTY, and a software input/output driver for Cassette Recorder. FIG 12 A shows how the data being transmitted via N1 – N4 is opto-coupled to the MC1488 line driver circuit which uses + 12v and

– 12v supplies compared to + 5v for the other logic circuits. The outputs of the line drivers at RS – 232 standard levels are then transmitted to the respective peripheral interfaces.

## CONCLUSION

A virile remote data acquisition system has been designed and developed using an 8085 Microprocessor and an 8251A USART as the peripheral Communication Interface. The transmission and reception when communicating with remote terminals is via a telephone modem. The remote data acquisition system features both automatic baud-rate detection novel means of selecting a peripheral device at the near end. This system can serve a wide range of remote data acquisition needs in industry involving tape storage and retrieval of data received from remote sources.

## REFERENCES

[1] Industries Association. Engineering Dept. (1969). *EIA standard RS-232-C: Interface between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchan*ge. Washington: Electronic OCLC 38637094

[2] Free Microprocessor Projects *Interfacing with Intel 8251A (USART)*. http://www.8085projects.info/INTEL-8251A-%28USART%29.html. Retrieved 24th August 2011.

[3] .D.A. Godse, A.P. Godse. (2009). *Microprocessors and Interfacing Tech*niques.

[4] P. K. Ghosh, P. R. Sridhar. (2009). *Introduction To Microprocessors For Engineers and Scientists*, 2Nd Ed