SERIAL

We all know microcomputers use parallel data for internal operations. Computers - mini, micro or maxi - are often specified with the "bus data width" as a key parameter. This is the number of parallel bits which participate in operations at one time. Typical microprocessors now available have bus data widths of 4,8,12, or 16 bits. If you employ a used minicomputer in your system you may enjoy a 12,16,18, or 24 or even 32-bit wide bus. When shuffling data to or from memory and peripherals, the parallel lines of the bus are defined simultaneously - and you have to run at least as many physical wires to each interfaced subsystem.

When wire is at a premium you can get by with only one channel if the data is sent in a time-ordered sequence, one bit at a time. Communications and peripheral interfaces thrive on diets of serial bits provided the speed is relatively low.

In this article, Don Lancaster provides us with an excerpt from his forthcoming book, **TV Typewriter Cookbook**, *to be published by Howard W. Sams, Indianapolis, Indiana. Don describes the basics of parallel to serial conversion and its inverse, using UART technology to do the transformation. His article this month concerns UARTs and serial interfaces which are relatively self-contained - local wires, tape recorders, etc. He also covers the communications aspects of serial data...radio and telephone network modem hardware.*

While Don wrote the article from the point of view of the TV Typewriter technology which he pioneered, the problems he discusses are just as applicable to the home brew computer context...simply read "computer" every time you see TVT in the text. Don's comments on the serial cassette interface will provide one input to the discussion of various possible recording interfaces in the pages of BYTE.

. . . CARL

by
Don Lancaster
Box 1112
Parker AZ 85344

Most TV typewriter circuits need all their ASCII character and command bits simultaneously available in parallel form. This is also true of most electronic keyboard encoders and the bidirectional data buses of many minicomputers and microprocessors. In simple systems, we can connect all these parallel sources and loads together as needed without any further interface circuitry.

Sometimes, it's far more useful or convenient to have the bits march by one by one in *serial* form. While serial form is much slower, it has one big advantage — only a single wire or communications channel is needed, instead of multiple signal lines. Another benefit of serial form is that it can be made slow enough to communicate over ordinary phone lines, cassette tapes, radio channels, or electromechanical teletype systems. Some of the places where we'd like to use serial transmission are:

Remote Keyboards, where a single pair interconnection gets used instead of expensive multiple conductor cable.

Teletypes, where the bits have to be converted to signals based on current or no current in a wire loop.

Industry Standard Interfaces, such as the RS232-C and the newer RS422 and GPIB, allowing signals to travel relatively long distances.

Cassette Recorders, where we can store and exchange characters and programs with properly designed single channel, speed independent circuitry.

Radio Transmission, where only two tones on a single transmitted frequency are often used. This is typical of ham RTTY.

Modems, or Modulator-Demodulators that let us exchange data over the telephone line, either one way, or both ways at once.

We can call the circuits

INTERFACE

that get us from parallel to serial and back again *serial interface*. Usually, there are two distinct parts to the interface problem. The first is to convert from parallel to serial (or vice versa) at *logic level*, staying compatible with CMOS or TTL integrated circuits. This is often done in an industry standard single integrated circuit called a UART, short for Universal Asynchronous Receiver Transmitter. UARTs will simply and cheaply do the conversion and back again, along with providing all the necessary housekeeping bits, control signals, and noise immunity provisions.

The second portion of our conversion process gets us from logic levels to whatever form of signal the serial part of the system uses — such as dc currents for teletypes, bipolar signals for standard interface, and carefully selected tones suitable for cassette recording or transmission over a radio channel or a phone line. We'll be taking a detailed look at most of these techniques later.

How Fast?

There are two basic types of serial transmission we can use, synchronous and asynchronous.

In synchronous transmission, all the characters (called "words") are locked into system timing. We know the exact time position of each piece of data. If some time is to go by without doing anything useful, do-nothing words called *nulls* are provided. Timing signals must somehow be supplied to each end of a synchronous serial data system so we can tell when each word is to start. This usually means a separate timing channel or track or some sort of elaborate timing recovery circuit. Synchronous systems are usually fast and complex, but they are rarely used in most TV Typewriter applications.

With asynchronous transmission, the data words are not locked into system timing and can arrive with almost any spacing between words. To tell the beginning and end of a word, we have to add some new bit groupings, called *start* and *stop* bits, to the data. We don't have to provide any other locking signal between the source and destination of the data. Asynchronous data is commonly used in TVT systems.

Both ends of a serial transmission system have to *exactly* agree on a system speed, usually called the *Baud Rate*. The Baud rate is simply how many bits per second are going to be transmitted, including any start and stop bits. Fig. 1 shows us some common Baud rates. The most popular of these include 110, 300, 600 and 1200 bits per second. 1200 BPS is usually the fastest that can be handled by the phone company without fully dedicated lines. Inside systems, and where special lines can be provided, faster synchronous standard Baud rates of 2400, 4800 and 9600 bits per second may be used.

Should the Baud rate change between transmission and reception, such as when two recorders are used, or the batteries on one recorder age, the receiving end of the system has to be able to adjust itself to the new effective Baud rate if errors are to be avoided.

110 Baud is very popular for limited speed TV Typewriter uses. This rate is compatible with the ASR-33 eight bit Teletype code and corresponds to a 100 word per minute typing rate. While this is the fastest that most teletypewriter systems can be driven, and is easily handled by two-way 103 style phone modems, it takes painfully long to fill the screen. Even with a 512 character screen,

Serial Uses

- Remote keyboards
- Teletypes
- Cassette recorders
- Modems

Fig. 1. Standard serial communications speeds.

BAUD RATE	TYPE	TELETYPE COMPATIBLE?	DDD PHONE COMPATIBLE?	FRAME UPDATE COMPATIBLE?	TIME TO LOAD 512 CHARACTERS
110 BPS	Asynchronous	YES	TWO WAY	YES	51.2 SECONDS
300 BPS	Asynchronous	NO	TWO WAY	YES	18.7 SECONDS
600 BPS	Asynchronous	NO	ONE WAY	MAYBE	9.3 SECONDS
1200 BPS	Asynchronous	NO	ONE WAY	NO	4.6 SECONDS
2400 BPS	Synchronous	NO	NO	NO	2.3 SECONDS
4800 BPS	Synchronous	NO	NO	NO	1.2 SECONDS
9600 BPS	Synchronous	NO	NO	NO	0.6 SECONDS

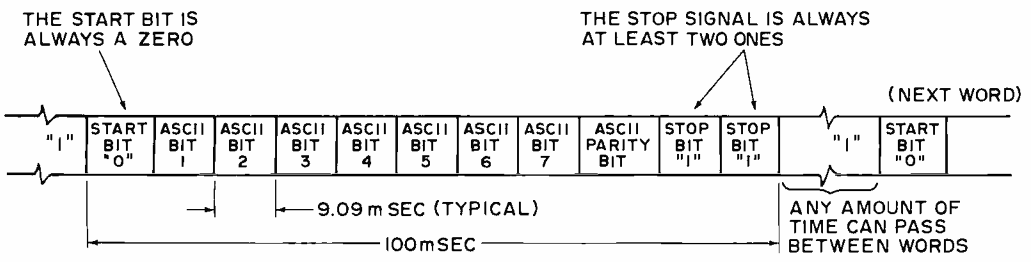


Fig. 2. 110 Baud, 100 word per minute code. "0" is a space or an open line. "1" is a mark or a shorted line.

UART = Universal Asynchronous Receiver Transmitter

RTTY = Radio Teletype

TVT = Television Typewriter

it takes 51.2 seconds, or almost a minute to load or retransmit the screen.

300 Baud is equal to 30 characters per second or 300 words per minute. This rate is the fastest normally used by a two frame update cursor system. It is also a rate easily handled by a cassette recorder, and by many two-way or full duplex modem systems. About 18.6 seconds are needed to load or dump a 512 character screen. These frame update and retransmission rates can be minimized by using a "virtual" update where one page is viewed while the other is updated, and by creative use of carriage return commands to return after the last character of each line, rather than going on out to the end of the line.

Faster Baud rates usually take more in the way of circuit design, and are usually limited to one-way modem transmission, premium recording techniques, and a Direct Memory Access type of update in the TV Typewriter.

A 110 Baud Standard

The 110 Baud, 100 word per minute code is an industry standard for slow data exchange. It is compatible with the Model 33 and Model 35 Teletype systems, and other teleprinters using an 8 level code. The code takes 100 milliseconds to send a character. The next character can follow immediately or can be sent any time later. Fig. 2 shows us the standard.

The original Teletype notation still carries over to this code. A Mark is a digital "1", a shorted line, or a completed connection. A Space is a digital "0", an open line or a broken connection. Between words, the teletype line or digital output is constantly putting out "1"s or marks, and is thus marking time. One of the reasons this was originally done was so that any break in communications is immediately known.

There are eleven bits to the code. Each bit is an identical 9.09 milliseconds long for a total code time of 100 milliseconds per word. Each word begins with a start bit. The start bit is always a zero and tells the receiving circuitry that a new character is to begin. The start bit is essential since some ASCII words will begin with one or more "1"s and there is no way to tell a marking time "1" from a "1" bit in an ASCII character code.

The ASCII bits follow in sequential order, starting with bit B1 or the least significant bit. After the seven character bits, an eighth bit is sent either as a "1" or providing a parity check bit for the rest of the word. At least two stop bits must follow the word. The stop bits are "1"s or marks, and any number of additional marking "1"s can follow between characters. The stop bits give the receiving circuitry a chance to shut itself down and await a new word.

The receiver can be electronic in the case of a UART or electromechanical in the case of a teletype. Between words, the receiver just waits. Since the data transmission is asynchronous, the receiver has no way of knowing ahead of time when a new word is to arrive, so it has to wait for a new start bit before it can do anything. The arrival of this bit activates the receiver, which—then goes through a sequential procedure that sorts out the bits, puts them in parallel form, and outputs them.

With a UART, sequential time intervals of 9.09 milliseconds each are electronically generated and the center of each interval window is tested against the incoming code to see whether a "1" or a "0" is received. These are accumulated in a shift register, error tested, and output as a parallel word at the end of the interval. The stop bits are used to reset and shut off the circuitry.

With an electromechanical teletype, the break in line current caused by the stop bit releases a one-turn clutch on a mechanical scanning commutator that goes once around in 100 milliseconds. It sequentially routes the incoming code to a group of scanning solenoid magnets. These set up the code in parallel form, and at the end of the word, the scanner resets and the code is typed or output on paper tape.

Tolerances

It's extremely important that both the transmitter and receiver are clocking bits out at the same 9.09 millisecond rate, and that nothing happens in the channel to speed up or slow down the bits. There are many possible sources of error. If the bit positions jitter around or are differentially delayed to by any tone keying, the filtering, or channel response, we get a bias error. Bias errors put individual bits ahead of or behind where they actually belong. One source of bias in a two tone modem or cassette system occurs when one tone is delayed more than the other in any filtering circuit. This is called the group delay distortion problem.

If the basic transmission and reception rates differ so that the bits can get ahead of or behind where they're supposed to be, we have a *skew error.* Note that bias errors apply to individual bits, while skew errors are progressive, making each sequential bit decision that much more difficult to detect without error. We get a skew error if there is an absolute timing difference between transmitter and receiver. Cassette recording systems introduce a potential skew error if the record and playback rates differ. This easily happens with cheaper units susceptible to speed variations with battery voltage, and almost is inevitable if the recording is done on one machine and playback on a second. Recording skew errors are correctable if the recording signals are designed to include speed information, and the receiver is capable of using this information to speed up or slow down as needed to eliminate this error source.

How accurate do we have to be? This is easy to calculate. Assume temporarily that there are zero jitter and zero bias errors in the channel, and that we are using an electronic receiver that very narrowly samples for valid data. The last data bit we are interested in is the parity bit. The center of the parity bit is 8½ bits removed from the beginning of the start bit, a delay of 77.26 milliseconds. The receiver delay is also supposed to be 77.26 milliseconds. If our sampling is narrow enough, we can be up to just under half a bit slow or fast and still be able to read the parity bit without error. This corresponds to a time error of 4.53 milliseconds either way, or slightly over 5%.

But, this figure leaves no room for bias and jitter errors and doesn't give the slower electromechanical circuits enough time width to reliably respond to the incoming data. As a practical rule, *the receiver and transmitter bit times must match to well within plus or minus one percent.* It is absolutely essential to hold things this close for low-error communications.

A 300 Baud asynchronous timing system is very similar to Fig. 2 and uses the same eleven bit code of equally spaced bits. The only difference is that the per-bit time is 3.33 milliseconds, corresponding to a 300 Hertz clock rate. Optionally, only a single stop bit may be used. This rate is cassette and TV Typewriter compatible and may be used for full duplex (two way) operation in most modem circuits, but it is too fast for teletype use.

Using UARTs

Parallel to serial conversion and back again can obviously be done with CMOS or TTL circuits. Basically, you parallel load a shift register and serially clock out data or serially clock in data to a shift register and then latch its parallel outputs when the data is valid. By the time you add all the error testing circuitry, housekeeping bits, synchronization, and so on, the circuits tend to get specialized and complex.

Instead of this route, you can use an industry standard MOS integrated circuit called a UART for virtually any serial to parallel conversion and return process. Several pin compatible UARTs appear in Fig. 3. These are general purpose, programmable devices that let you select the number of start and stop bits, the word length, type of parity, and so on to suit your particular system. Dedicated UART-like devices are also available for use with specific microprocessors. The Intel 8201 and the Motorola 6850 are typical of these.

The standard UART comes in a 40 pin package, and has supply voltages of +5 routed to pin #1, -12 to pin #2, and ground to pin #3. The later versions of these devices are N channel types that need no -12 supply, and that pin is left unconnected. The General Instruments AY-5-1014 is one of these.

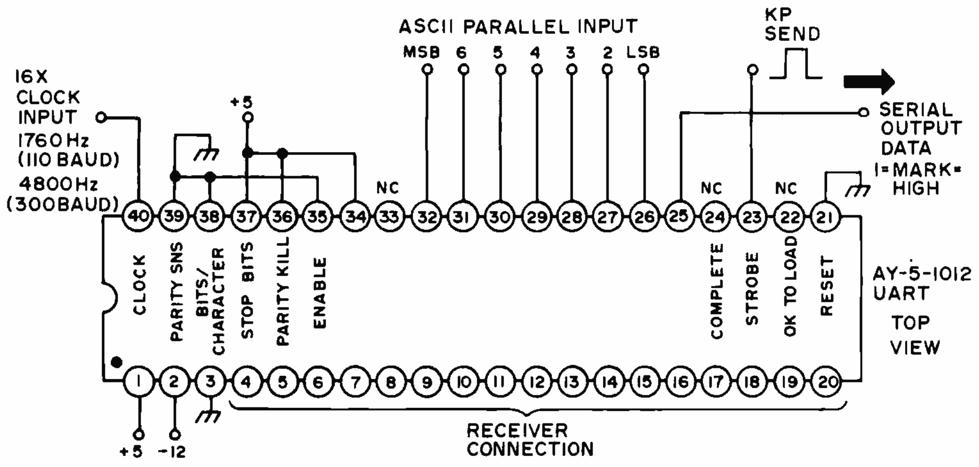
Fig. 3. Pin compatible UARTS.

S1883 (American Microsystems)
AY-5-1012 (General Instruments)
2536 (Signetics)
COM2502 (Standard Microsystems)
TMS6012 (Texas Instruments)
TR1602 (Western Digital)

The low number pins (1-20) are the receive portion of the UART, while the high number pins (21-40) are the transmit portion. Except for common word length and parity programming, the two halves of the circuit are separate, although they often are used as a send-receive pair.

Both the receiver and transmitter portions of the circuit need a clock. The clock frequency is usually *sixteen* times the Baud rate. This high frequency lets the UART do things like sample the center of each data interval and recheck for valid start signals and similar good things. For instance, a 110

Fig. 4. UART circuit to transmit code of Fig. 2.



Baud circuit needs a clock of 1760 Hertz, while a 300 Baud one uses a 4800 Hertz clock and so on.

The clock signals can be derived from a CMOS or 555 type of astable oscillator, but it is far better to digitally derive clock frequencies from TVT system timing or another stable source. Remember these clock signals must be held to well within one percent and ideally shouldn't have any adjustments. If our TVT has a 15,840 horizontal rate, we can divide this by nine to exactly get 1760 for a 110 Baud system. With a 15,720 rate, we get 1746.6 Hertz, a figure a bit low, but still useful and less than one percent under.

The receiver and transmitter clock inputs are on separate pins. They are often tied to a common clock source in simple send-receive circuits. One important exception is when a UART is used as part of a speed-independent cassette interface. In this case, the *receiver* clock frequency is derived from the tape during playback. While it is nominally the same as the transmitter frequency, its exact value is set by speed information recovered from the recorder. This can be used to eliminate much of the skew error that would normally result from a change in tape speed from time to time or machine to machine.

Fig. 4 shows us the connections for transmission of the 11 unit code of Fig. 2. The input ASCII code goes on pins 26 through 33, with the least significant or b1 bit on pin 26. A 16X clock goes into pin 40, and a KP send command goes to pin 23. The leading edge of this send command starts transmission, but the input data must be valid for the entire time the command is positive. Normally, this is a narrow pulse a few milliseconds wide, derived from a keypressed command on a keyboard. Serial output data appears on pin 25.

Pins 34-39 program the UART for different bit lengths and codes. 34 is an enable that normally remains high. Pin 35 provides a parity bit if it is grounded and omits one if it is high. Pin 36 picks the number of stop bits. Ground gives you one and high gives you two. 37 and 38 together decide how many *data* bits are to be sent, ranging from 5 to 8. Both grounded provide for 5 data bits, useful for Baudot RTTY transmission. The connection shown gives us a seven bit data word. Note that if you use the parity bit, it *adds to* the number of data words. The code of Fig. 2 uses our start bit, seven data bits, one parity bit, and one stop bit for an eleven unit code. Pin 39 picks even or odd parity with ground giving odd parity. An optional reset input is provided on pin 21. It is normally grounded. Bringing it high resets the UART. Without resetting, the first word transmitted after power is first applied can be wrong.

The UART transmitter is double buffered. This means you can load a new character as soon as the one already inside begins its transmission. Two optional outputs are provided. Pin 22 tells you when it is OK to provide a new character by going high. Pin 24 tells you that a character has been completely sent when it goes high.

There are two ways you can use a UART transmitter, either unconditionally or handshaking. In the unconditional mode, any time a character arrives, it gets sent. This is the simplest, but you have to make absolutely certain that characters don't arrive spaced or grouped too closely together. While pairs of inputs can be closely spaced much the same way that two key rollover works in a keyboard, you have to be absolutely certain that the long term average is never exceeded by the word rate of the UART. This means a 100 millisecond character spacing for 110 Baud, and around one third that for a 300 Baud system. In the handshaking mode, the UART decides when it wants to receive a new character, using the pin 22 and 24 outputs. Circuits driving the UART are set up to provide characters only when they are asked for them. For most TVT uses, unconditional UART transmission is simpler and easier to use.

Fig. 5 shows us the UART receiver circuit, again set up for the code of Fig. 2. The receiver logic has elaborate noise elimination provisions, made possible by the sixteen times higher clock frequency. Whenever a start bit is purportedly received, that bit is retested later and verified to prevent a random noise pulse from generating an unwanted character output. All data bits are narrowly sampled in the middle of their possible time slots, allowing considerable bias and skew distortion to exist without error.

The same pin 34-39 inputs that programmed the transmitter's word length and format identically program the receiver portion of the UART. Although the receiver and transmitter can be used in totally different circuits and at different baud rates, they have to operate with a common format, set by these pins.

The receiver needs its own clock of sixteen times its Baud rate, applied to pin 17. In send-receive systems, we can often tie the receiver and transmitter clocks together. In speed-independent cassette interface circuits, the receive clock is reconstructed from speed information recovered from the recorder to eliminate skew errors.

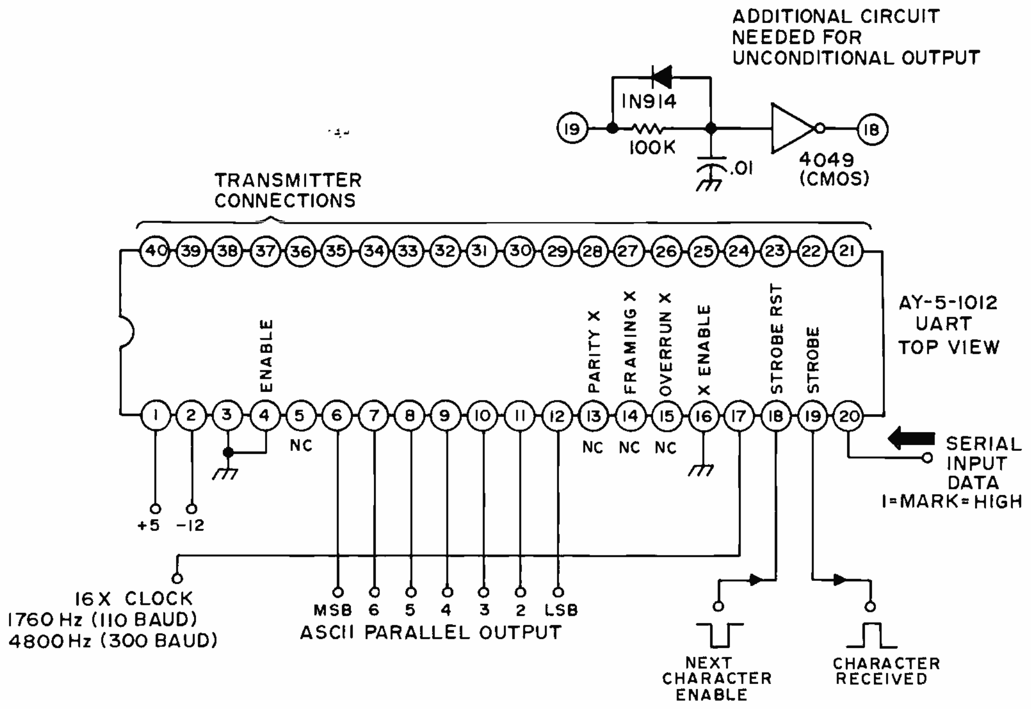
The serial input is routed to pin 20 and converted to the equivalent seven bit parallel ASCII output on pins 6 through 12, with the least significant bit on pin 12. When the output is valid, a strobe on pin 19 goes high as an output.

This output strobe must be reset before a new character can be output. If we're operating in a handshaking mode, the TVT circuitry accepting the character sends back an acknowledgement or completion signal that momentarily drives pin 18 low. If we are using an unconditional output mode, you somehow have to delay the pin 19 strobe output, invert it, and reapply it to the strobe reset input pin 18. Fig. 5 shows us one way to do this with a RC network and a CMOS inverter. Many UART receiver problems are caused by failing to reset the strobe after each character — watch this particular detail very closely.

Several additional outputs are available for use in fancier systems. Parity, framing, and overrun errors produce respective high outputs on pins 13 through 15. These can be used to ask for a repeat or put in a question mark to indicate a transmission error. All receiver UART outputs are tri-state and may be floated in systems where the UART's outputs must share a common or a bidirectional data buss. Making pin 4 positive disables the ASCII outputs, while making pin 16 positive disables the error outputs.

Besides its usual use as a two way serial to parallel converter, there are other

Fig. 5. UART circuit to receive code of Fig. 2. Parity, word length, and stop bits set by transmitter programming, pins 34-39.



useful circuit tricks you can do with a UART. For instance, by connecting the parallel outputs of the receiver back to the parallel inputs of the transmitter, you can change Baud rates. This is handy in speeding up slow data for use on a fast channel, and for correcting speed errors on cassette systems.

If the UART is to accept data from several sources you can either tri-state combine the sources onto a single input buss, or else use an input eight pole, double throw selector switch to pick one of two input channels. This is common in TVT service, where the keyboard forms one channel and the screen retransmission output buss forms the other. 4502 hex tri-state drivers or 4019 four pole selectors are suitable CMOS devices to use. In general, tri-state lines onto a common buss are much preferable to selectors, but few keyboard encoders have inherent tri-state output, and the output buss on the TVT often has to continously drive the display if we want to view the retransmission process.

Teletype Interface

There are two common types of teletype systems in use today. The older type is the five code bit machine, typical examples of which are the Teletype model 28, the Creed model 75, and various Kleinschmidt models. While these machines are commercially obsolete, they still see usage for ham RTTY and some deaf communications systems. Their reasonable price availability makes them attractive for home computer hard copy as well, although the available character presentation is extremely limited. These older machines all use the more or less obsolete Baudot code and are not directly ASCII and TVT compatible, unless conversion ROMs and figures-letters logic is added to them. We'll note in passing that a UART may be used in the Baudot code by applying the code to pins 26-30 with the least significant bit on pin 26, and making pin 35 high and grounding pins 37 and 38. Older UARTs will generate two stop bits, while more recent ones (such as the TR1602B) will automatically generate the needed 1.42 stop bits in this mode.

The second common type of machine is the computer and timesharing standard teletype — the Teletype models 33 and 35, particularly the ASR-33. These use a standard eight bit ASCII code and follow the format of Fig. 2. They are directly compatible with TVT system coding.

Either type of system is based on breaking current in a dc loop. This current is often either 20 or 60 milliamperes. Transmission occurs when a mechanically coded commutator generates an output code by once-around breaking the current as often as needed.

Reception reverses the process. A momentary break representing the start bit releases a once-around commutator that distributes the code breaks to magnets which set up a pattern for printing when the scan is complete.

Fig. 6 shows us the interface for a Model 33 teletype. For optimum use, the teletype is internally programmed to a 20 milliampere current loop and full duplex operation. This is done following the teletype's maintenance manual. Full duplex operation means that the keyboard and printer aren't connected to each other. The keyboard can send and the printer receive both at the same time.

The transmitter interface provides a 20 milliampere current for a mark or a one and an open circuit for a space or a zero. The receiver senses a closed contact for a mark or a one and an open contact for a space or a zero. Extra inverters are added as shown to make the codes correspond so that a "1" from the UART is read as a "1" by the teletype. Be sure to observe the line polarities shown. The transistors can be almost any medium power, reasonable gain devices. More information on teletype interface appears in the Intel MCS-8 Users Manual.

Similar current-no current interface loops can be used with older teletype systems. However, some of these machines need substantially higher currents and are notorious as transient and ground noise generators. With these older machines, total isolation is strongly recommended, using small, high speed reed relays or else opto-isolator circuits.

With any teletype system, the transmit Baud rate of the UART must match the needs of the teletype to within one percent. In the case of the ASR-33, a 110 Baud rate is needed, resulting in a 16X UART clock of 1760 Hertz.

Industrial Interface

There are presently quite a few "standard" interfaces used to get between

Fig. 6. UART-teletype (ASR-33) interface. Teletype must be internally set to 20 mA loop current and full duplex operation.

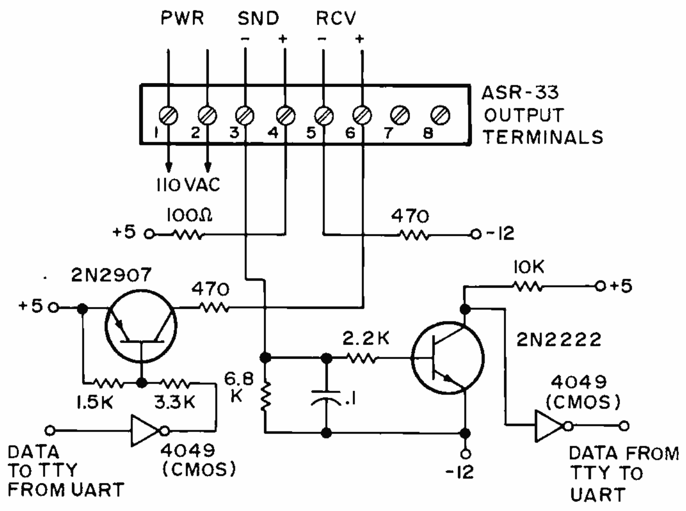
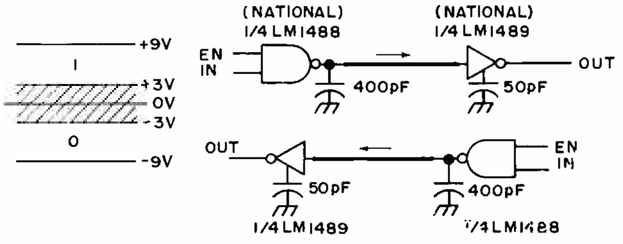
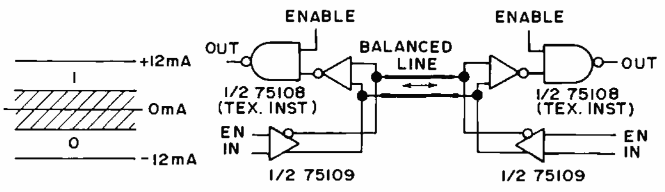


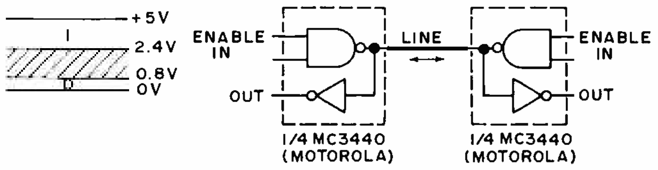
Fig. 7. Industrial interfaces for cable interconnections.



(a) RS232-C



(b) RS422



(c) GPIB

commercial modems, printers, test equipment, computers, and anywhere else you need a reasonable distance, noise free interface. Three of the most common include the RS232, the RS422, and the General Purpose Interface Buss or GPIB. Fig. 7 shows us what's involved in the way of signal levels and interface techniques.

RS232-C is an old EIA standard that predates IC techniques and is somewhat unwieldy. It is widely used in commercial modem circuits and for most large scale computer serial interface. The signal is bipolar, with a logic "1" being defined as +3 to +9 volts or more and a logic "0" being defined as -3 to -9 volts or more. Capacitors on the drivers limit risetimes to 30 volts per microsecond or less to minimize ringing and transient effects. Capacitors on the receivers limit the response as needed to reject noise but pass the highest transmitted Baud rate. The 1488 and 1489 integrated circuits form a typical interface pair.

RS422 is a newer EIA standard that uses balanced transmission lines and differential current sensing to eliminate any common mode noise. A current, typically of 6 to 12 milliamperes in one direction, defines a "1", while the reversed current defines a "0". The balanced line may be used either single direction or bidirectional, depending on how the receiver enables are used. In any bidirectional or party line system, it's extremely important to enable only one driver at a time. The 75108 and 75109 are typical balanced line drivers and receivers.

The General Purpose Interface Buss or GPIB uses TTL compatible levels, but combines them with a terminated line, high drive capability, and receivers with hysteresis for good noise immunity. The GPIB is most often used in parallel form to interface test and measuring equipment with computers and calculators. It can also be used as an effective serial interface that takes no special power supplies. The 3440 is a quad bus transceiver useful for GPIB service.

More information on EIA standards are available from the Electronic Industries Association, 2001 Eye St. NW, Washington DC 20006, while information on the GPIB interface is available from Hewlett Packard, 1501 Page Mill Rd., Palo Alto CA 94304.

Cassette Interface

Magnetic storage in the form of tape drive, disc files, and floppy discs have long been a standard and expensive way of storing bulk serial data for computer use. An obvious and extremely low cost substitute for these would seem to be the ordinary audio cassette recorder. Besides providing bulk storage, the cassette can replace paper tape and punched cards, and handle programs as well. One big advantage of cassettes is potentially low cost duplication and distribution and exchange of programs.

Cassette recorders present several serious design problems when used for storage of digital data. Even a quality machine will vary its speed by a percent or more, and the machine to machine variations, particularly on lower cost units, can far exceed the amount of skew

Fig. 8 Speed independent cassette standards.

110 BAUD (CODE PER FIG. 2)
ARK = 1 = 16 CYCLES OF 1760 HZ
SPACE = 0 = 8 CYCLES OF 880 HZ

300 BAUD (FIG. 2 CODE WITH 300 HZ BIT RATE)
MARK = 1 = 16 CYCLES OF 4800 HZ
SPACE = 0 = 8 CYCLES OF 2400 HZ

600 BAUD (FIG. 2 CODE WITH 600 HZ BIT RATE)
MARK = 1 = 8 CYCLES OF 4800 HZ
SPACE = 0 = 4 CYCLES OF 2400 HZ
(clock doubling required)

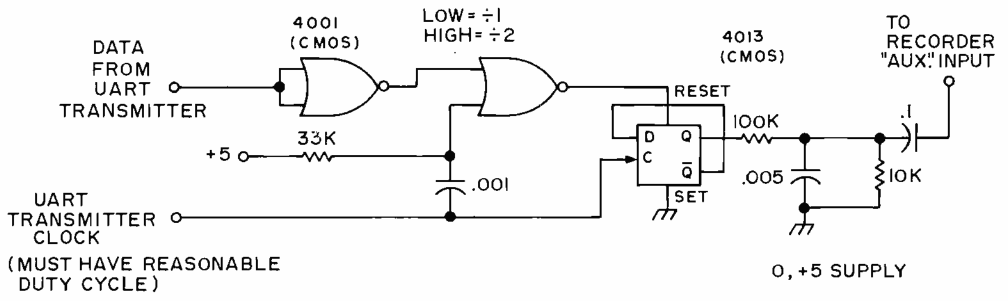
distortion the code of Fig. 2 can stand — if the receiver UART is running at a constant clock rate. Some of the techniques used to send data over modems and radio channels will work with a single, quality recorder, but these circuits are far from optimum as they have no way to compensate for recorder speed variations. Similarly, many of the standard computer tape recording techniques may work, but they are based on a different type of recording head and system — one that works with pulses, saturated flux changes, and sense amplifers, rather than one that records and plays back sine waves more or less linearly.

An ideal cassette interface would handle any machine and provide for machine to machine variations. We'd also like to supply some protection against dropouts, perhaps in the form of integration or multiple voting on what constitutes a one or a zero. A low cost, simple system using a single supply voltage and a minimum number of non-critical adjustments is obviously desirable, particularly if the system can be very tolerant of recorder levels and settings, and need no exotic codings, preambles or other limitations.

We can base such a speed variation tolerant or "speed independent" system on a unique property of the UART. For every received one or zero bit, the UART needs exactly sixteen receiver clock pulses. By designing a standard so that sixteen cycles of clock are recovered from a one recorded on the tape, and sixteen cycles of clock are recovered from a zero on the tape, the UART will always receive just the exact number of clock pulses it needs per one or zero. Tape speed variations of plus and minus 30 percent or more should be possible.

Fig. 9. Speed independent cassette interface. Values shown for 300 Baud rate. Recorder speed may vary ±30%.

(a) Record circuit.



(b) Playback circuit.

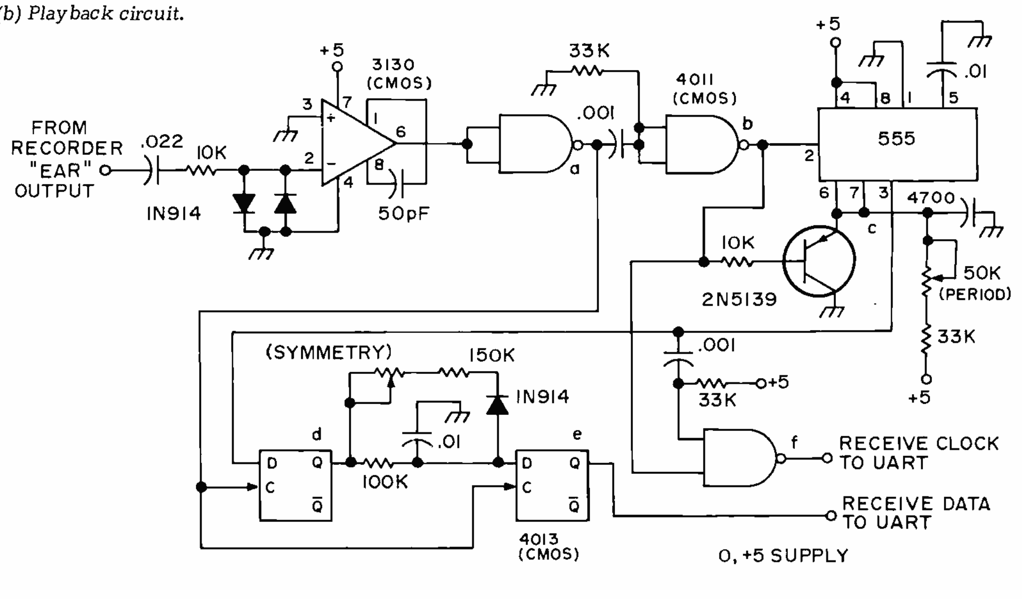


Fig. 8 shows us a set of standards for speed independent cassette interface. All three standards are based on the code and bit timing of Fig. 2. A 110 Baud standard records 16 cycles of 1760 Hertz for a one and eight cycles of 880 Hertz for a zero. The 300 Baud version, which is recommended for most uses, records 16 cycles of 4800 Hertz for a "1" and 8 cycles of 2400 Hertz for a zero. A 600 Baud version can be based on the 300 Baud standard by halving the length and doubling the clock recovery.

Fig. 9 shows us the 300 Baud circuitry involved, along with the key waveforms of Fig. 10. Even including the $5 to $10 cost of the UART (usable elsewhere in the TVT anyway), the circuit is extremely simple, non-critical, and cheap.

Fig. 9A shows the record circuit. As usual, a UART transmit clock of 16 times the Baud rate must be provided. For recorder use, the duty cycle of this clock must be stable and nearly 50%. Note that the UART serial data will always change synchronously with the UART clock, after a brief propagation delay. Clock and Data from the UART transmitter are sent to a gate and flip flop that divides the clock by two if the data is a zero and divides by one if the data is a one. This is done by resetting the flip flop about half way through a clock cycle if the data is a "1". We get sixteen clock frequency cycles with a "1" and eight half clock frequency cycles with a "0". Since the inputs are synchronized, there are no transient problems, and all cycles are full length. This output is 10:1 attenuated and moderately filtered to round the rise and fall times. This prevents excess peaking of the high frequency compensation inside the recorder. The output is capacitively coupled to the recorder AUX input.

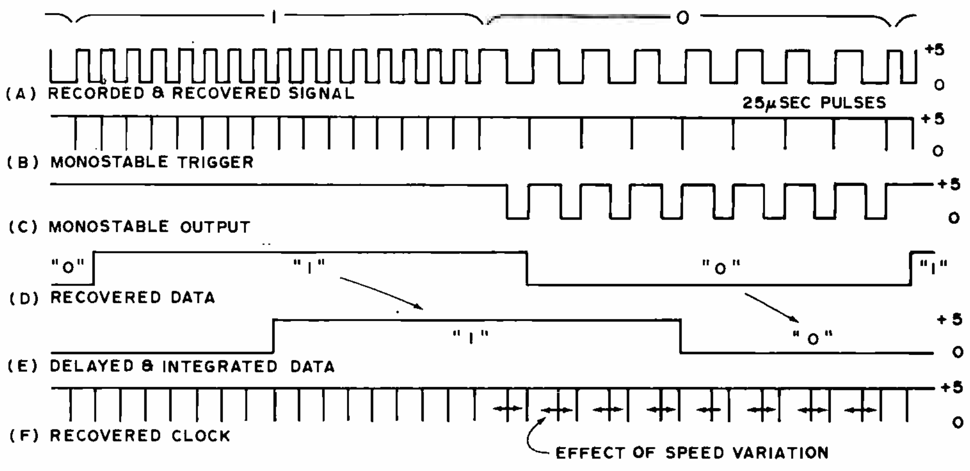
The receiver gets its signal from the recorder EAR output. This signal is high pass filtered and doubly limited, first by a pair of diodes and then by a CMOS op amp. This particular op amp lets you run the inputs at the same voltage as the "negative" supply. If you use a standard op amp here, you'll need either a negative supply or positive input bias. A CMOS gate following the op amp improves the rise and fall time.

The gate's output is shortened and generates a pulse that lasts for 25 microseconds or so, coincident with the positive edge of the transmitter clock. This trips a negative recovery monostable set to two thirds the period of the lower frequency clock. If a string of "1"s is received, the monostable keeps on being retriggered and never drops its output. If a string of "0"s is received, the monostable drops its output for the final one third of each cycle. The net result is that you get a train of 8 negative going pulses out for each zero and nothing for each one. The leading edges of these outputs are sensed and shortened to 25 microseconds, and then combined with the input 25 microsecond pulses. The result is 16 clock pulses routed to the UART receive clock — either sixteen from the data for a one, or eight from the data and eight from the monostable for a zero.

The monostable's output also goes to a flip flop to recover the data. We get a "1" out of the first flip flop whenever 16 cycles of high frequency data are received and a zero whenever 8 cycles of low frequency data are received. The second flip flop integrates the output of the first one to eliminate any noise pulses and to take an average of several sequential ones or zeros before providing an output.

As the tape speeds up or slows down, the input square waves will also speed up or slow down. The monostable's output can absorb almost a 33 percent increase (from half to 2/3) of the high frequency clock period, or a 33 percent decrease (from 1 to 2/3) of the low frequency clock period without error.

Fig. 10. Key waveforms of speed independent cassette interface.



Oddly enough, though the circuit will work with poor quality recorders, the tape quality should be good — bad tape means bad data.

The spacing of alternate zero clock pulses to the UART will change as the speed changes — but the UART doesn't care about this so long as the clock pulses don't actually overlap. This jitter is automatically eliminated by circuits inside the UART.

You calibrate the system by inputting a string of "1"s and noting the pot position where errors first happen. You then input a string of zeros and note the pot position where errors first happen. Then you set the pot one third of the distance from the limiting one to the limiting zero settings. This is a non-critical adjustment. The symmetry control can optionally be adjusted with a scope, or else by inputting alternate ones and zeros and adjusting for one half the supply voltage (as read on a meter) at the UART receiver data output.

There is one little detail that must be checked. *The positive edge of the UART transmit clock must be the same edge that, when received, is tripping the negative recovery monostable.* If it isn't, you may get partial pulses instead of a nice uniform eight pulse train during a zero. Recorders may vary in their internal circuitry and may provide a phase inversion between input and output. If you have this problem, either use the $\overline{Q}$ output of the recorder flip flop, or interchange the connections on pins 2 and 3 of the receiver limiter. Once set for a given recorder, there should be no further problems along this line.

While the circuit seems to work with amazingly poor recorders, best operation is gotten with a clean, medium to better grade recorder, preferably one that has an automatic level control on the input and tone controls available, along with an AUX input and an EAR output. Best operation will normally result with the output volume control about half way up and the tone control set to maximum treble boost.

A very important point that's often overlooked is the tape quality. Bad tape means bad data. Use only fine grain premium tapes (Radio Shack Supertape is typical). The key thing to watch for is whether the amplitude variation is guaranteed to less than one decibel. If the variation isn't specified — *don't use the tape.* The cost difference between good and cheap tape is negligible. You should also always "certify" your tape before you use it by writing a repetitive and obvious pattern over the entire tape length and checking for errors, splices or dropouts.

Extra controls to start and stop the recorder can optionally be added to make the storage more efficient. Data should never be entered or removed until the recorder is up to speed, and the UART's output lines can be gated to prevent garbage from getting into the system. Formatting data by setting it up on the TVT screen and then using a screen read to load the tape will give you dense storage.

One final detail that may need watching is to make sure the cassette recorder can't *overspeed* the system if you happen to record on a slow machine and playback on a fast one. This could output characters faster than can be accepted by a teletype or a TVT frame update system. The way around this problem is to set some maximum possible character rate that will let the return characters speed up without problems — use of a 7.5 Hertz rate on a 110 Baud system or a 22.5 Hertz rate on a 300 Baud system is one example. A second potential solution is to use two separate UARTs — programming the transmitter for two stop bits and the receiver for only one. When going from a cassette to a teletype, data can be resynchronized by connecting the UART parallel receiver outputs to the UART parallel transmitter inputs. Once again, make sure you can't overspeed the system.

Radio Data Links

One of the more common methods of sending serial digital data over a radio channel is to use a two frequency *frequency shift keyer* method in which one frequency represents a digital one and the other a digital zero.

Ham RTTY provides us with a typical example. At the audio baseband, two tones are used, defined as 2125 Hertz for a mark, or one, and 2925 Hertz for a zero or space. These tones represent the fifth and seventh harmonic of 425 Hertz.

These tones may be digitally generated the same way the modem tones of the next section are produced, or may be generated by a voltage controlled oscillator such as a 555, 8038 or a 566. These frequency shifted tones are used to frequency modulate an rf carrier. Alternately, the carrier itself can remain at its normal frequency for a mark and can be shifted down 850 Hertz (the difference between 2125 and 2925) for a zero, with the audio differences being picked up by mistuning the receiver by 2125 Hertz.

Fig. 11 shows us a typical receiver demodulator circuit. The carrier is received and detected by a FM receiver, adjusted to output audio tones of 2125 and 2925 Hertz. These tones are limited and routed to two bandpass filters, one set to the upper and one set to the lower frequency. Outputs are amplitude detected and compared, resulting in a one out for a frequency of 2125 and a zero for 2925. This output may be routed to a UART for serial to parallel conversion. Normally a 7.42 unit code of 60 or 100 words per minute, using Baudot encoding, is used for ham RTTY.

Any radio carrier system must follow the rules and regulations for the particular frequencies used. Best performance of a frequency shifted keyed system normally results when the generated frequencies are sinewaves and are switched, transient free, at their zero crossings. Receiver filters should delay both sets of frequencies identically to prevent ones from getting ahead of zeros or vice versa, and thus creating times when neither a one or a zero, or both of them together are simultaneously present. As with any serial interface, input and output code formats and Baud rates must closely agree.

Modems

Modems, or modulator-demodulators, are ways to get tones or tone groups onto the telephone line and off again in order to send and receive digital data. Two common ways of coupling modems to the phone line are to use small speakers to acoustically couple to a standard handset, or to directly connect to the phone line through a suitable protective network or data access arrangement. Acoustical coupling can be used anywhere on an unmodified telephone, but has problems with frequency response, microphonics, and second harmonic distortion caused by the carbon transmitter. Direct coupling gives better control and better performance, but has to meet certain telephone company regulations and interconnect restrictions, needs a physical connection to the phone line, and needs its own hybrid or means of separating transmitted and received data.

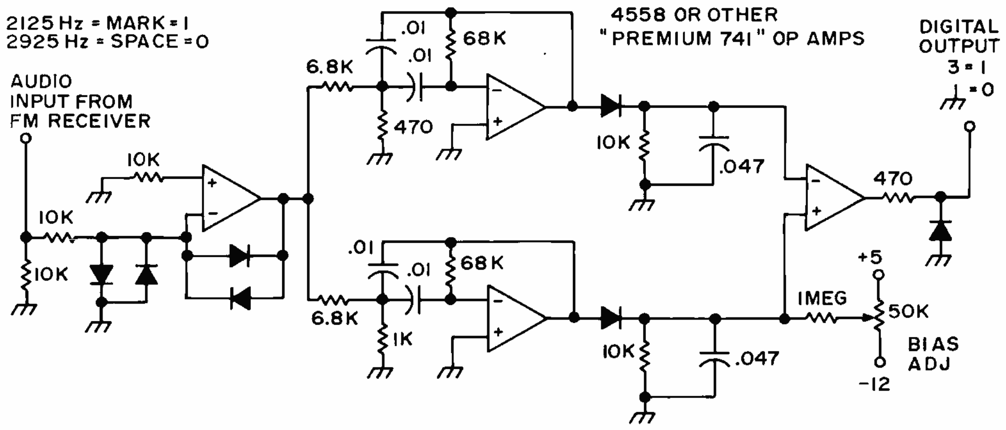
There are several basic ways to use modems. Simplex transmission goes one way only. Simplex with a back channel goes one way only, but provides for some low frequency communications, often limited to 4 Baud or less, in the other direction. This can provide for handshaking, message acknowledgement, etc..., but is far too slow to return data. Half duplex systems can send or receive data, but not simultaneously. Either the transmitter is off or in a mark condition while data is being received, or the receiver is disabled while data is being sent. In full duplex systems, data can be sent both ways, independently, and at the same time.

There are at least three basic types of modem systems that are useful over ordinary telephone lines, based on the Bell 103, 202 and 400 series systems. You can rent these from the phone company or others, buy them outright from modem firms, or design your own with the guidelines of this chapter. Most of the commercial modems use RS232-C interface standards and include such logic and switching functions as automatic answer and hangup, carrier detect, and other housekeeping signals.

The 400 systems are based on the touch tone ringing frequencies and accept contacts as inputs and are limited in the number of characters and the Baud rate. Baud rates of 10 to 20 characters per second are usually the maximum, and operation is normally simplex, with a separate unit needed for transmission and reception.

The 202 systems are half-duplex modems that can run up to 1200 Baud over the

Fig. 11. Audio processor for RTTY receiver.



phone line, but cannot simultaneously communicate in both directions unless a special four wire system is used.

The 103 modems are full duplex and may be used at 110 and 300 Baud rates. For the majority of TVT uses, this series, run at either Baud rate, is the most practical. Unlike the other serial interface circuits of this article, proper design of a good 103 style modem circuit with reasonable noise performance is a major job, particularly if your circuit has to operate over the dialup network for long distances and is to reliably communicate with commercial modems on the other end.

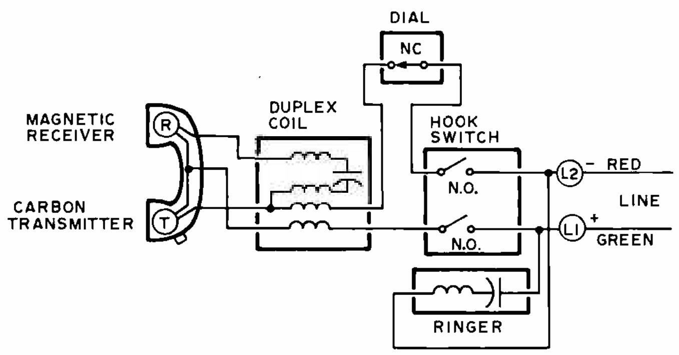


Fig. 12. Telephone schematic.

Phone Characteristics

A simplified schematic of a standard 500 telephone set is shown in Fig. 12. A carbon variable resistance transmitter and a magnetic headphone-style receiver are connected to the line by way of a duplex coil, a normally closed dial contact, and a pair of open when unused hookswitch contacts.

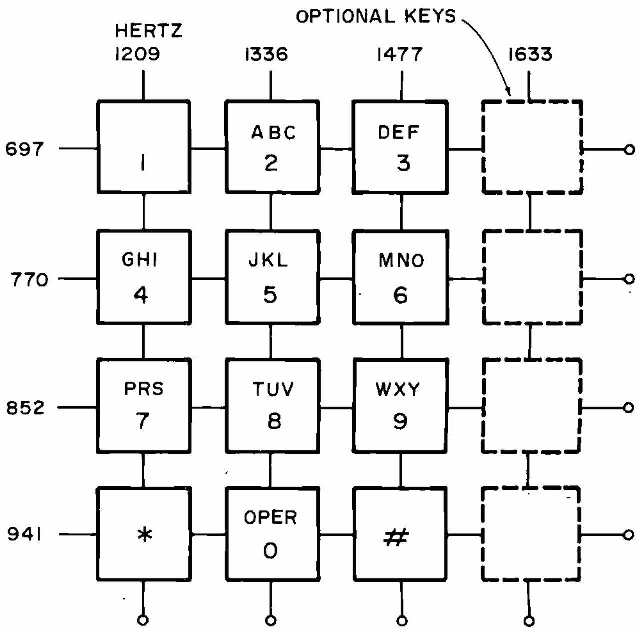
The duplex coil makes sure that outgoing signals reach the line and that ingoing signals reach the receiver with a minimum of interference. This is done by having two transmitter windings induce nearly equal and opposite signals into the receiver windings. This effectively cancels much of the local transmitter's signal into the receiver. The net result is to keep transmitter energy from being wasted in its own receiver and minimizes a "hear yourself" sidetone that psychologically makes people speak much more quietly. The duplex coil attenuates the transmitted signal by four decibels (to 60% voltage), the received signal by two decibels (to 80% voltage), and the sidetone by seventeen decibels (to 14% voltage).

The line is powered by a 48 volt central office battery supply, and the ac impedance of the line is nominally 600 Ohms, but varies with distance and quality of service. The audio signal levels at the line terminals are fractions of a volt. Normally, the loudest permissible modem tones are around a quarter of a volt, measured on the outgoing line. Received signals are lower still, typically one half to one tenth this value for local service, and even less on long distance loops.

When the phone is connected to the line by lifting the hookswitch, the line voltage drops to around 6 volts or so. The traditional dial signals by breaking this connection to deliver a group of mechanically spaced pulses that jump the line voltage between the open circuit and phone-off-the-hook values. Touch tone systems replace the dial with a low impedance that sums the tones of the next section (see Fig. 13) onto the line for signalling, a pair of tones at a time.

The ringer is capacitively coupled across the line and is resonant to some low frequency in the 20 to 47 Hertz range. Ringing voltage is an ac signal of 86 volts RMS. Selective ringing of a party line can be done in three wire systems with a ground return by ringing one phone from L1 to ground and the other one from L2 to ground. In two wire systems, ringer circuits with different

Fig. 13. Standard touch tone frequencies. Each key simultaneously generates two tones as shown.



resonant frequencies can be selectively rung by changing the frequency of the ring signal.

For direct entry modems, either a protective network or a data access arrangement such as the Bell CBS or CBT units can be used. These networks simulate the impedance of the telephone when activated and prevent any supply voltages from going onto or coming off of the power line. Under no circumstances should dc power be applied to or removed from the phone system lines, or any impedance be placed across the line or to ground that would degrade normal telephone services.

400 Style (Touch Tone) Modems

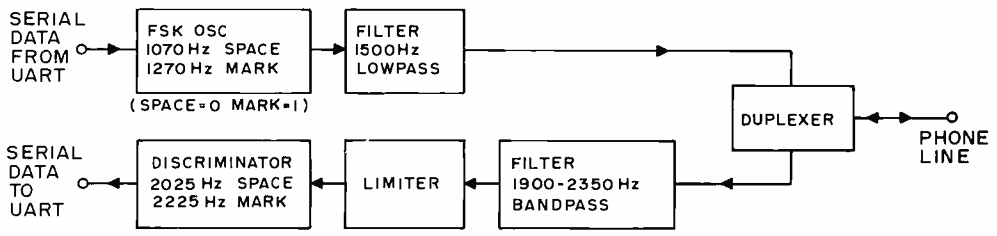
Modems based on touch tone signalling frequencies are usually limited to low data rates and often to a limited number of available characters. Touch tone signalling is based on simultaneously sending a pair of carefully chosen tones, following the code of Fig. 13.

The tone pair must exist for 40 milliseconds and the minimum time between tone pairs is 40 milliseconds, with a resulting maximum character rate of 12 per second. Touch tones are normally entered at signal levels somewhat higher than other voice and modem signals, being around three quarters of a volt RMS for the high frequencies and around half a volt for the low group. Line characteristics equalize these amplitudes by the time they get to recognition circuits.

A touch tone modem transmitter can simply be the touch tone dial of a remote phone, or it can be a circuit to generate two sine waves of proper amplitude and frequency simultaneously. Unlike other modems, and much of the serial interface of this chapter, the code is activated directly by contact closures. One closure, rather than a serial code, is all that is needed to send one of twelve or one of sixteen separate pieces of information.

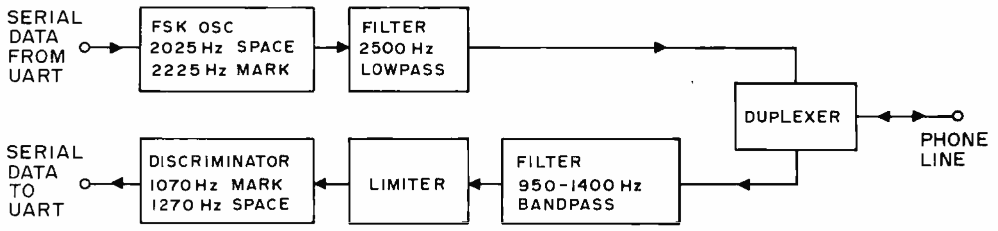
Fig. 14. Full duplex 300 Baud modems.

(a) Originate modem (unit making call).



SERIAL DATA FROM UART
FSK OSC 1070 Hz SPACE 1270 Hz MARK
(SPACE=0 MARK=1)
FILTER 1500Hz LOWPASS
DUPLEXER
PHONE LINE
SERIAL DATA TO UART
DISCRIMINATOR 2025 Hz SPACE 2225 Hz MARK
LIMITER
FILTER 1900-2350 Hz BANDPASS

(b) Answer modem (unit receiving call).



SERIAL DATA FROM UART
FSK OSC 2025 Hz SPACE 2225 Hz MARK
FILTER 2500 Hz LOWPASS
DUPLEXER
PHONE LINE
SERIAL DATA TO UART
DISCRIMINATOR 1070 Hz MARK 1270 Hz SPACE
LIMITER
FILTER 950-1400 Hz BANDPASS

Additional tones or tone combinations can be added, such as in the Bell 401L or 402C systems that offer 99 or 256 characters.

Touch tone reception consists of three parts. First, the signals need sharply filtered with bandpass group filters whose response is 650 to 1000 Hertz for the low band and 1150 to 1700 Hertz for the high band. Adequate prefiltering is absolutely essential for most tone detection schemes. Tones are then detected, using limiters and slicers somewhat similar to Fig. 9, using narrow bandpass filters and detectors, or using phase lock loop tone detectors such as Signetics 567 tone decoder. Finally, the detected tones are combined with suitable two of eight digital logic.

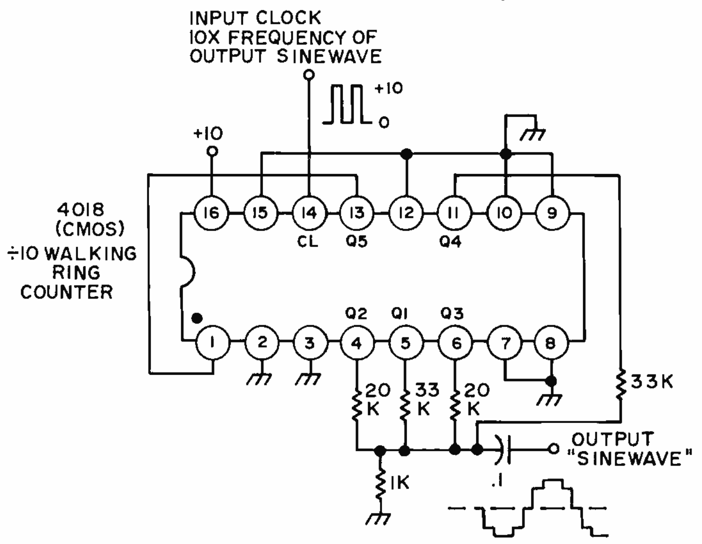
Additional information on touch tone techniques appears in the March 1963 *IEEE Transactions on Applications and Industry,* the Signetics *Linear IC Applications Manual,* and various issues of the *Bell System Technical Journal.*

103 Style (300 Baud, Full Duplex) Modems

"103" style modems are often the best choice for TVT use, as they offer full duplex, two way, operation at 110 or 300 Baud rates over the ordinary phone line. Fig. 14 shows us a block diagram of this type of modem.

The circuits are used in pairs. The modem at the end that's doing the calling is called an originate modem. It sends a 1070 Hertz sine wave for a space or zero and a 1270 Hertz sine wave for a mark or one, usually at a phone level of -10 DBM or around a quarter of a volt RMS. The modem that's doing the receiving is called an answer modem, and it

Fig. 15. Digital "sine wave" modem transmitter is easily filtered as first strong harmonics (-20 dB) are the ninth and eleventh.



receives and responds to these two frequencies. In turn, the answer modem transmits a 2025 Hertz sine wave for a space or a zero and a 2225 Hertz sine wave for a mark or a one. These, in turn, are acceptable to the originate modem.

These frequencies are carefully chosen to allow two-way conversation without interaction. The answer modem always transmits on the high frequency since a 2025 Hertz note is needed to automatically disable echo suppressors used on long distance phone lines, and to provide a standard recognition signal for automatic dialing equipment. (Echo suppressors effectively convert long distance lines into voice keyed one way lines. Two way transmission on a long line is not possible unless these suppressors are defeated.)

There are several very important things to consider when you are designing a modem. The transmitted signal must be a low distortion sine wave. Particularly, its second harmonic must be extremely low to prevent the originate modem's transmitter splattering its own receive spectrum with its second harmonic. When acoustical coupling is used, the transmit level must be held low enough that the rather bad second harmonic distortion of the carbon mike doesn't raise its signals to an intolerable level. Partial compensation of this carbon mike effect can be gotten by summing a sine wave of plus one half the third harmonic with the fundamental. This causes some cancellation and allows a higher level of transmission. Since most modem detectors use only the zero crossing information, it's important to coherently switch between these two frequencies, changing only when the sine wave goes through zero. The coherent operation eliminates "short cycles" that will jitter the received data.

Input signals to either modem *must* be strongly filtered to get rid of the other channel tones, as well as interference from speech, noise, touch tone coding, and other signals. The duplex coil in the phone set reduces but does not eliminate sidetone coupling. If you build your own duplexer instead, the same cancellation is only partial because of changing telephone line impedances.

In addition to getting rid of unwanted signals, there's a second severe restriction to the input filter. Both the ones and zeros going through the filter must be delayed an equal amount. Otherwise the ones and zeros will get out of step with each other and cause timing errors.

There are two basic ways to go about building this style of modem. An analog modem generates and decodes its signals using gated oscillators, RC networks, and phase lock loop detectors. A digital modem uses all digital logic for the frequency generation and detection. Analog modems should be avoided for several reasons. The transmitters inherently have less stability, need field

Fig. 16. Digitally derived modem frequencies.

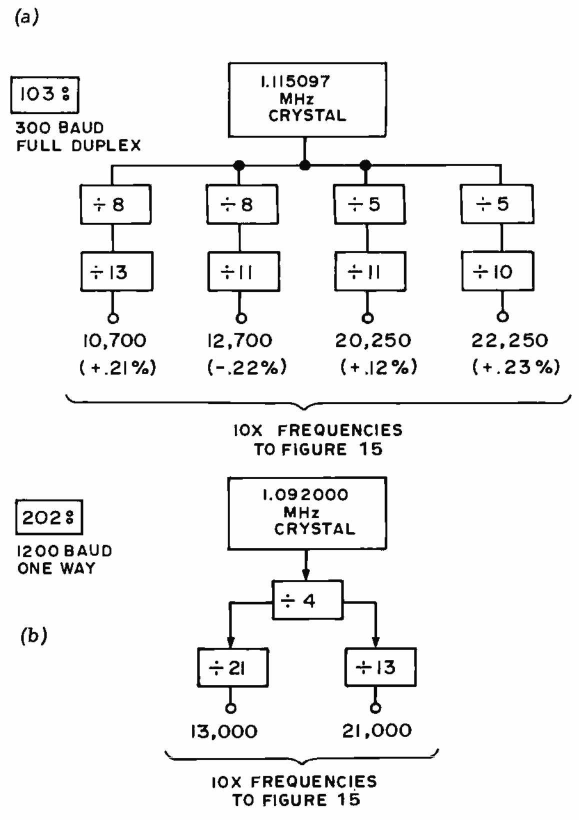
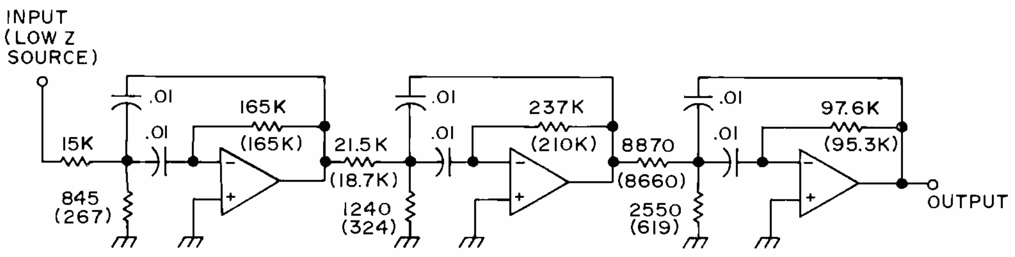


Fig. 17. Modem receive filter. 4558 or 7415 op amps. 950-1400 Hz = normal values = answer filter. 1900-2350 Hz = (parenthetical values) originate filter.



adjustments, and potentially have a stronger second harmonic, besides needing calibration. They are harder to coherently switch at zero crossings to eliminate transients. Analog receivers also must be calibrated and able to accurately resolve a small frequency difference, again with adjustment and calibration.

At this writing, there is no such thing as a modem on a chip. The Motorola MC6860 is one IC that handles approximately one fourth of the circuitry needed for a digital 103 modem. Two Exar chips, the 2207 FSK generator and the 210 FSK demodulator provide around half the circuitry for an analog system. A premium set of four hybrid integrated circuits from Cermetek Electronics is available that does the whole job in their Minimodem CH1213, 1214, 1252, and 1257 devices.

Figs. 15-18 show several techniques that might be of use in your own modem designs. Fig. 15 is a CMOS digital IC sine wave generator; it produces a sine wave in response to a 10X digital clock input. It is based on summing phases of a walking ring or Johnson counter and has negligible harmonic output up to the ninth and eleventh, which are both twenty decibels down (1/10 the amplitude) and easily filtered. The output can also be used to coherently synchronize input switching. Fig. 16 shows a digital timing sequence that starts with a crystal and produces all four modem frequencies needed for the Fig. 15 circuit. It can be built with a CMOS 4520 and a gate or two. Fig. 17 shows us some active filters useful as pre-filters with controlled group delay distortion. Fig. 18 shows an adjustment and calibration free receiver digital discriminator.

More information on 103 style modem designs are available in Motorola applications note AN731, Exar data sheets XR210 and XR2207, *The Active Filter Cookbook* (Sams), and Cermetek Microelectronics Minimodem data sheets.

202 Style (1200 Baud, Half Duplex) Modems

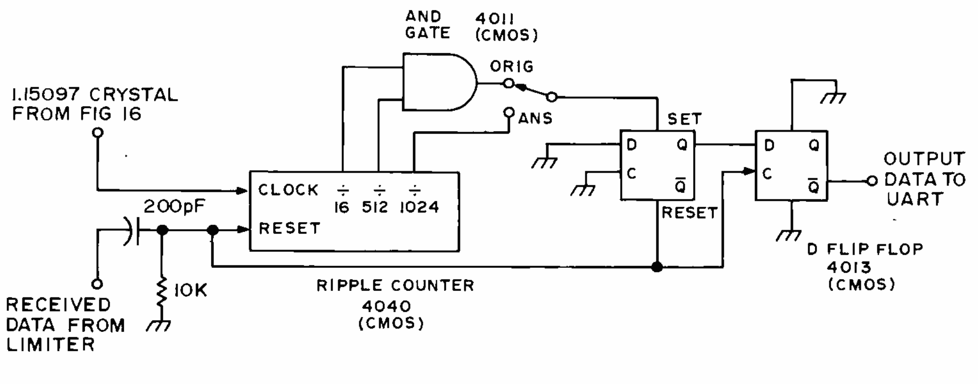
The 202 style modem circuits are both faster and simpler than the 103 versions and require less in the way of circuitry. Their big disadvantage is that most of them are strictly simplex or half duplex devices when used on the ordinary two wire phone line. Simultaneously transmitting and receiving is ordinarily not possible.

Like the 103, 202 standards use frequency shift keying. Bell standards call for a 1200 Hertz mark or one and a 2200 Hertz space or zero, while international standards call for a 1300 Hertz space or zero and a 2100 Hertz mark or one. An additional tone may need generation to provide for automatic answering. Commercial units also sometimes provide a back channel of four or five Baud for acknowledgement.

The circuit design techniques for both types of modem are similar. Because of the faster Baud rate, control of group delay distortion in any filtering is extremely important. Detection circuitry must not differentially delay ones with respect to zeros. The Rockwell 10371 Digital Telecommunications Data Interface handles much of the non-filtering aspects of this type of modem. This IC also has a built in UART.

Several additional sources of modem information include the Microdata *Communications Handbook*, *Data Modem Evaluation Guide* by V. V. Villips, and various issues of *Data Communications* and *Telecommunications*.

Fig. 18. Digital discriminator needs no adjustments or calibration.


AND GATE
4011 (CMOS)
ORIG
ANS
1.15097 CRYSTAL FROM FIG 16
200pF
RECEIVED DATA FROM LIMITER
10K
CLOCK ÷16 ÷512 ÷1024
RESET
RIPPLE COUNTER 4040 (CMOS)
SET
D Q
C Q̄
RESET
D Q
C Q̄
OUTPUT DATA TO UART
D FLIP FLOP 4013 (CMOS)