accuracy by using the ratio of two integers to approximate the constant.

For example, $^{355}/_{113}$ approximates π to within (8.5E–6)%. Therefore, you can replace the statement a=b*3.1415926539 with a=(b*355)/113. You've thus substituted an integer multiply and divide for a floating-point multiply. In doing so, you generally get a substantial improvement in execution speed. The function in **Listing 1** determines the best integers to use for a given word size. **Table 1** gives you approximations for several mathematical constants.

You can realize even greater efficiency by making the denominator a power of 2 and substituting right shifts for the divide. **Listing 2** gives a function to calculate ratios of this type. **Table 2** shows the approximations for the constants in **Table 1**. (DI #1774) EDN

To Vote For This Design, Circle No 369.

Clock-recovery PLL fits into single PLD

RICARDO MONLEONE, AGIE LTD, LOSONE, SWITZERLAND

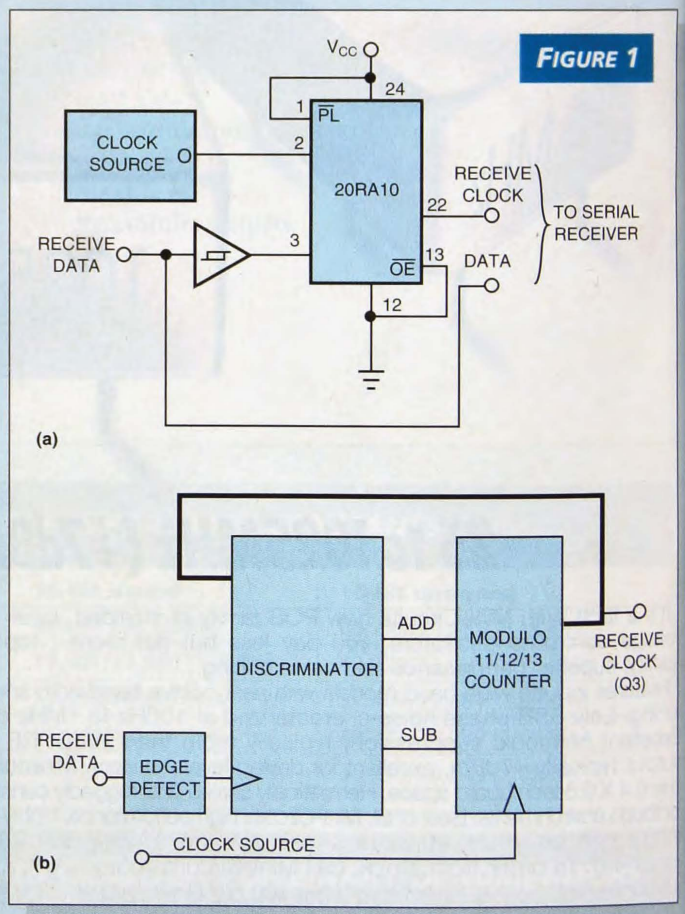
BBS Synchronous serial protocols, HDLC, for example, require that you synchronize the transmitter and the receiver. Sometimes, this synchronization requires an extra wire to carry the serial clock. However, many serial-communication ICs can save this wire thanks to an internal PLL clock-recovery circuit that reconstructs a clock signal phase-locked to the serial data stream.

This internal clock-recovery unit is often the limiting factor for the maximum baud rate; many ICs can handle higher baud rates if you supply the serial clock externally. For instance, Intel's 80C152 µC can handle baud rates up to 2.4 Mbaud with an external serial clock (via a separate wire or an external digital PLL) and only 2 Mbaud using its internal digital PLL. Similar considerations apply to Zilog's Z85C30 SCC (4 vs 1 Mbaud) and many other chips.

The circuit in **Figure 1a** is a digital PLL implemented in a single 20RA10-type PAL that provides a recovered serial clock to a receiver IC. The circuit suits NRZ/NRZI data streams and has been tested for baud rates of 2 Mbaud and over. The clock source must be a quartz oscillator, such as the processor's clock source. **Figure 1b** is the block diagram of the logic implemented in the PAL. The logic consists of an edge detector (equations "delay" and "edge_det"), a self-reloading modulo 11/12/13 counter (equations "Q0" through "Q3"), and a timing discriminator (equations "add," "sub," and "reset"). The external oscillator, which must have a frequency of 12×B Hz, where B is the baud rate, directly clocks the counter.

The edge detector identifies rising and falling edges in the incoming data stream. Normally, the circuit expects edges when the counter is near state 6 (**Figure 2**). The circuit sets either the add or sub flip-flop, depending on whether an edge arrives after or before state 6, respectively. If the add flip-flop is set (the edge arrived too late), the counter will add one count to the actual period, thereby counting to 13. Inversely, if sub is set, the counter will count only to 11. If neither flip-flop is set, the counter has a default period of 12 clocks. An internal reset signal controls the modulo of the counter. If the reset signal is 1, the counter will go to state 0 at the following clock. The sub and reset signals clear when the counter reaches state 0; the add signal clears in state 12.

The recovered clock of the digital PLL is the Q3 output (bit



This PAL-based PLL (a) recovers a serial clock from the incoming data. The corresponding block diagram (b) shows the logic in the PAL.

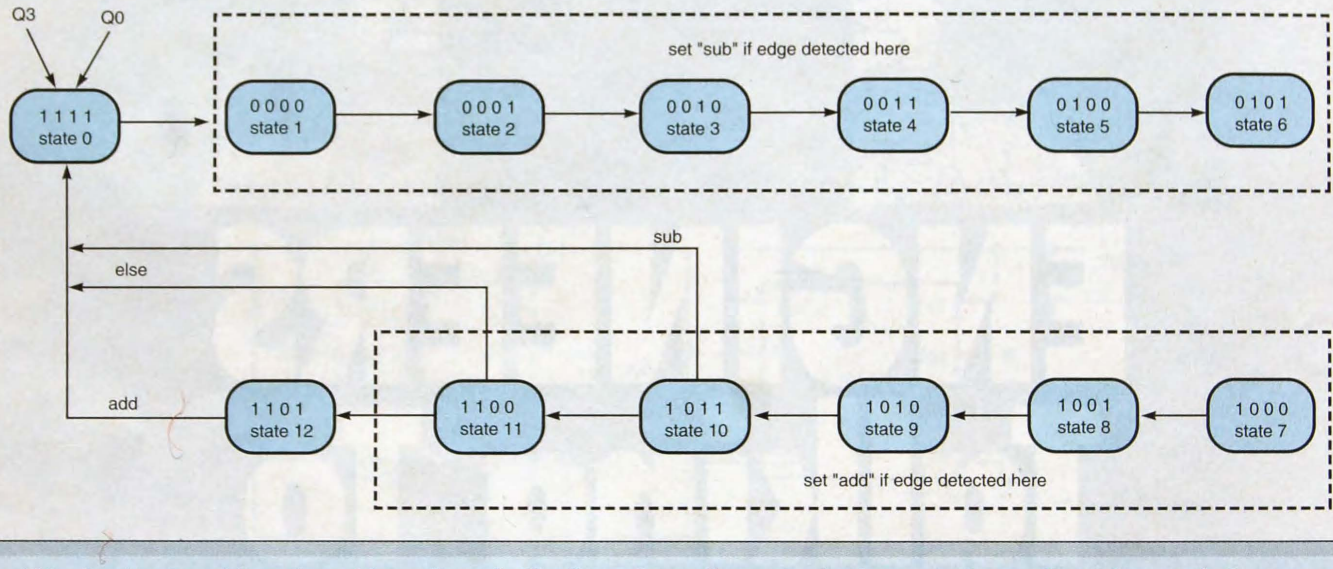
3) of the counter. This circuit is intended for those serial-communication chips that shift in data with the falling edge of the serial clock. For those requiring a rising edge, it is sufficient to invert the declaration of Q3.

The lock range is B±B/13. To keep the PLL locked in, a certain minimum activity on the serial data line is necessary. This minimum depends on the quality of the oscillator, the accuracy of the incoming data stream's baud rate, and the selected data coding (NRZ or NRZI). However, practical tests with NRZ coding and a standard quartz clock show that, after inactivity periods of 10000/B (seconds), the PLL is still well in phase. In High Level Data Link Control transmission, you can increase this time if the transmitter sends flags while idling, if you add preambles to the frames, and if you use NRZI coding.

The optional Schmitt trigger buffer of **Figure 1a** reshapes the received serial signal for better operation of the edge detector. Using standard 30-nsec 20RA10 PALs, this digital PLL can handle data rates up to 1 Mbaud. 20RA10-20 parts can handle 2 Mbaud, and GAL20RA10 parts can handle even more. You can also integrate the design into field-programmable gate arrays. Using the *EDN* bulletin-board system, you can download PALASM and CUPL listings from **EDN BBS /DI_SIG #1793.** (DI #1793) EDN

To Vote For This Design, Circle No. 370

FIGURE 2



The PLL circuit sets either the add or the sub flip-flop, depending on whether an edge arrives after or before state 6, respectively.

Interface ties eight-channel ADC to 8051 μC

DHANANJAY VASUDEO GADRE, INTER-UNIVERSITY CENTRE OF ASTRONOMY AND ASTROPHYSICS, PUNE, INDIA

The circuit in **Figure 1** shows a precision, multiple-channel-ADC interface for the 8051 series of embedded controllers. The ADC interface is based on the MAX186 (Maxim Integrated Products (Sunnyvale, CA)) low-power, eight-channel, serial 12-bit ADC. The **figure** shows how you can connect the ADC's simple four-wire interface to an 80C32, an approach that results in compact code with reduced software overhead.

The MAX186 can operate in either internal- or external-clock mode. In the external-clock mode, the circuit must write a control byte into the D_{IN} pin before any conversion can occur. The SSTRB output of the MAX186 goes high for one clock cycle after the control byte clocks in. The D_{OUT} pin outputs the result of the conversion on the next 12 falling edges of SCLK. In the external-clock mode, the MAX186 requires that SCLK has a frequency between 0.1 and 2 MHz.

You can program the serial interface of an 8051-family processor to work in a variety of modes. The serial port supports the standard RS-232C protocol, a multiprocessor communication protocol, and a half-duplex synchronous-communication mode. This synchronous-communication mode (mode 0) is really a shift register, which you can use to shift data out or in at any given time.

The serial-port TxD and RxD pins work as clock and data,