ANALOG DEVICES

AN-328
APPLICATION NOTE

ONE TECHNOLOGY WAY • P.O. BOX 9106 • NORWOOD, MASSACHUSETTS 02062-9106 • 617/329-4700

Generate 4 Channels of Analog Output Using AD7542 12-Bit D/A Converters and Control the Lot with Only Two Wires

by John Wynne

This application note describes how a Universal Asynchronous Receiver Transmitter (UART) can be configured to drive four AD7542 12-bit D/A converters. The AD7542 is ideal in this application because of its data loading structure. The functional diagram of Figure 1 shows the AD7542 to consist of three 4-bit data registers, a 12-bit DAC register, address decoding logic and a 12-bit multiplying DAC. Data is loaded into the data registers in three 4-bit nibbles, and subsequently transferred to the 12-bit DAC register. These data move operations are controlled by the address inputs A0 and A1. Each character received by the UART contains one 4-bit nibble of data and four bits of associated control information.

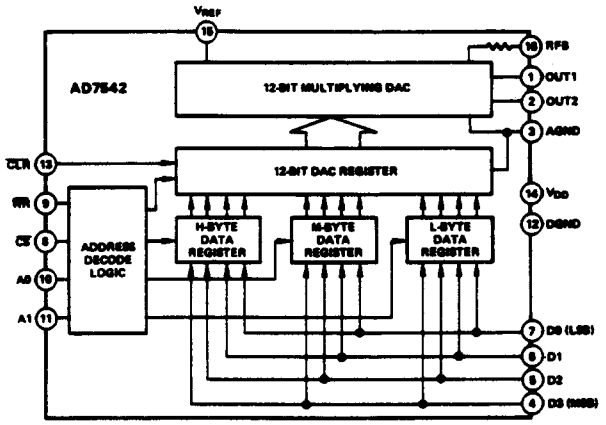

Figure 1. AD7542 Functional Block Diagram

A block diagram of the circuit is shown in Figure 2. All data and control signals pass over one opto-coupler allowing very simple wiring between a host processor and remote analog output channels. The power consumption of the complete circuit is kept very low due to the fact that every active component in the circuit is constructed with either CMOS or low power Bifet technology.

SERIAL DATA FORMAT

The format of the serial data used by the receiving IM6402 is shown in Figure 3. This consists of one start bit, eight data bits (LSB first) and two stop bits. Transmission may be either from another UART or directly from a microcomputer. Many modern microcomputers have dedicated serial input/output ports which can be used for transmission. However, small variations between microcomputers do exist and so it will be necessary to tailor the receiving UART to allow different formats to be accommodated.

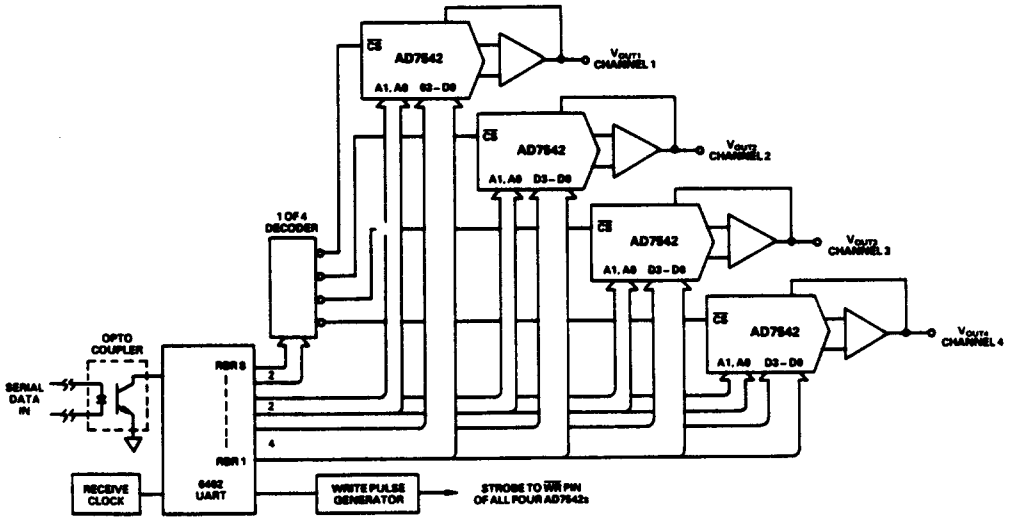

Figure 2. Block Diagram of Circuit

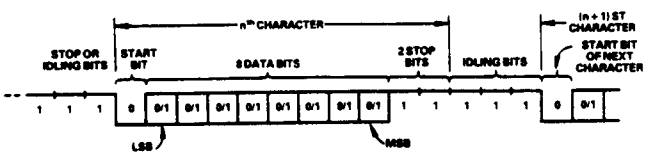

Figure 3. Serial Data Format

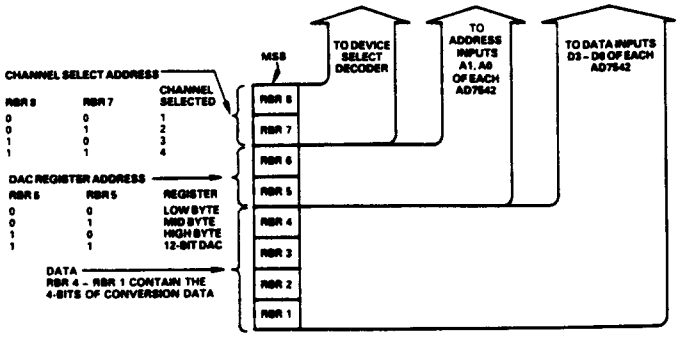

Figure 4. Breakout of the 8-Bit Character Received by the UART

As a character is received at the UART, the start and stop bits are stripped off and the data is transferred to the Receiver Buffer Register. A high level on the Data Received (DR) pin indicates that a new character has been received. Each 8-bit wide character contains 4-bits of conversion data and 4-bits of control data. A detailed functional breakout of each bit in the 8-bit wide character is shown in Figure 4.

The two most significant bits go to the decoder to select one of the four DACs to be updated. Character bits RBR6 and RBR5 go to address inputs A1, A0 respectively of each DAC to select one of the internal registers for a data move operation. The four least significant bits (RBR4 – RBR1) contain the data to be loaded to one of the 4-bit nibble registers.

The simple 1-of-4 decoder is shown in Figure 5. The truth table is shown in Table I. The four decoded outputs go to the Chip Select (CS) inputs of the DACs. This signal must be low for a data move operation to occur.

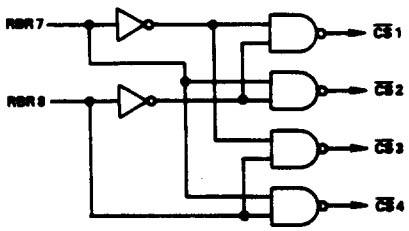

Figure 5. Channel Select Decoder

RBR6	RBR7	$\overline{CS1}$	$\overline{CS2}$	$\overline{CS3}$	$\overline{CS4}$
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

Table I. Truth Table for Figure 5 Decoder

The final signal which the DACs require is a Write ($\overline{WR}$) pulse tocontrol DAC loading. This signal is common to all four DACs and is generated by means of the Data Received (DR) output (see Figure 6). As previously mentioned, a high level on the DR pin indicates that a new character has been received by the UART. The combination of the CMOS inverters and the resistor-capacitor networks of Figure 6 produces a delayed low-going signal on the Data Received Reset ($\overline{DRR}$) input. This clears the DR output low which in turn drives the $\overline{DRR}$ input high again. This low-going pulse on the $\overline{DRR}$ input (equal to the width of the delay) is used as the $\overline{WR}$ pulse for the DACs.

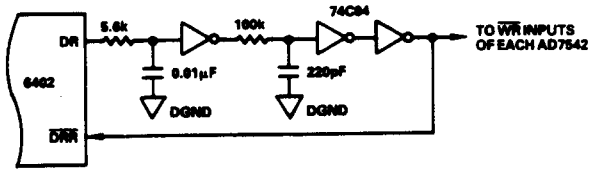

Figure 6. Write Pulse Generator Circuit

The complete circuit is shown in Figure 7. The optocoupler used is a high speed HCPL-2530 manufactured by Hewlett Packard. The Receiver Register Clock (RRC) is generated by the ICM7555, a low power CMOS version of the popular 555 timer. Note that this clock rate must be 16 times the receiver baud rate. The values of the timing components, RA, RB and C in Figure 7, have been chosen for a 9600 baud rate. The 10k ohms potentiometer should be adjusted (adjusted value is approximately 9k ohms) until the RRC frequency is 153.6kHz. The published relationship between the timing components and the generated clock frequency on the RRC input is

$$f_{CLK} = \frac{1.46}{(RA + 2RB)C}$$

In practice this relationship is valid only at low frequencies. At the required RRC clock frequency the measured frequency was typically 30% lower than the calculated frequency. Different baud rates can be accommodated by changing the timing components.

DAC OUTPUT CIRCUITRY

Note that each DAC of Figure 7 is configured for Unipolar Binary Operation. This configuration provides straight (binarily-related) attenuation of the input reference signal, which can be either a.c. or d.c., current or voltage. The application examples shown in the following section use this mode with a fixed –10V reference to the four DACs. Other modes of operation of an AD7542 are possible, for instance, 4-quadrant multiplication or even single supply operation. The availability or otherwise of suitable power supplies at the remote location determines the output voltage ranges and DAC configurations which are possible. For the application examples a remote power supply of ±15V is required. For information on these other DAC configurations the reader is referred to the AD7542 data sheet and to Ref [1,2], all of which are available from Analog Devices.

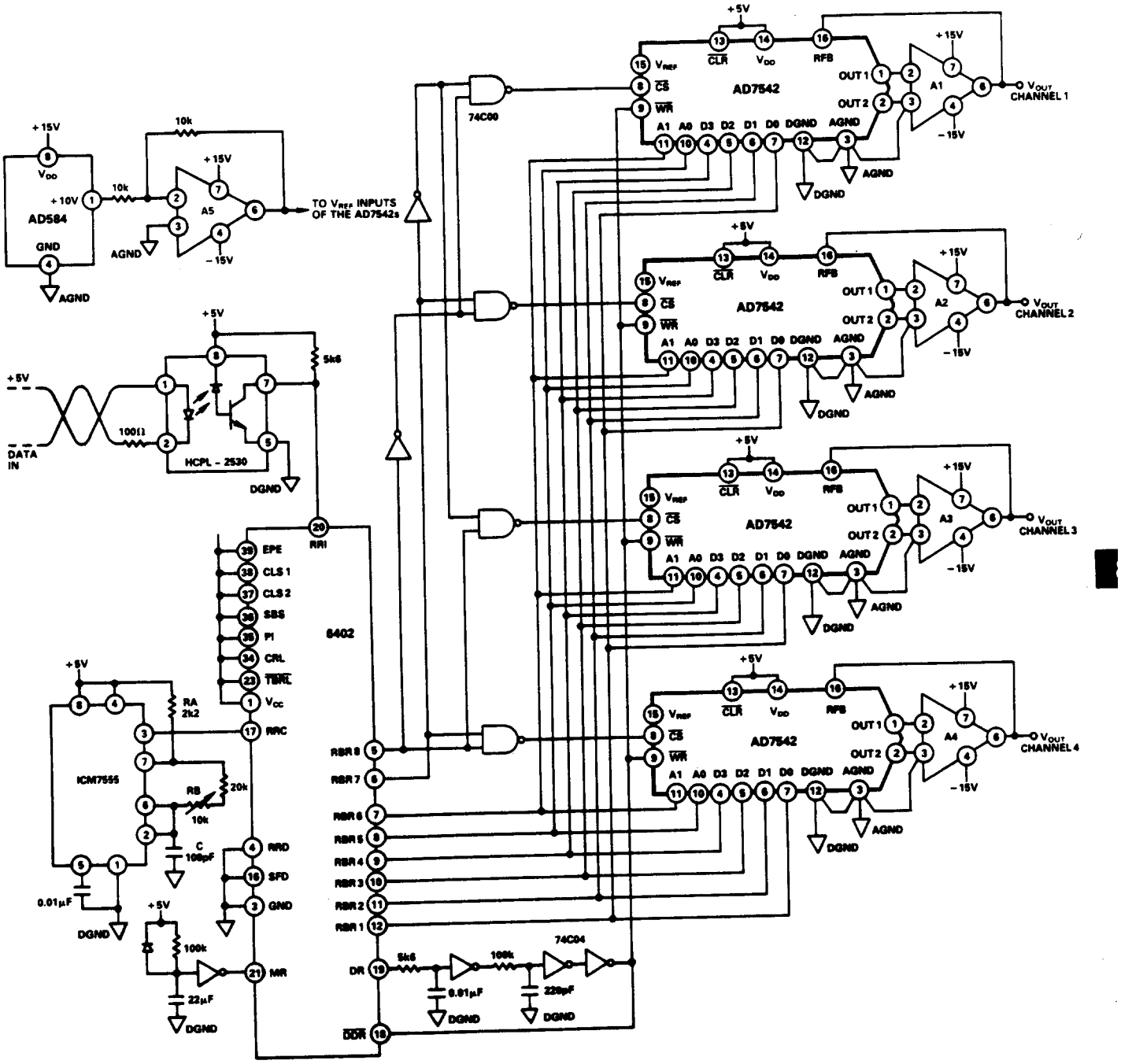

Figure 7. Detailed Circuit Diagram

THE SOFTWARE INTERFACE

Before data transmission can occur the 4-bits of conversion data must be packed with the proper 4-bits of control data to form an 8-bit character. This character is then passed to the transmitter for serial transmission. If the transmitter is a UART similar to the receiving one, the 8-bit character need only be written into the UARTs transmit register and the UART commanded to transmit. Start and stop bits are automatically added by the UART itself. If the transmitter is based upon a microprocessor then some extra software is required to supervise the serial transmission. An example is given which loads 12-bits of data to one of the four DACs and immediately updates the analog output. This involves transmitting four characters sequentially to the addressed DAC, i.e., the 3 data nibbles and the load DAC register command. The software is written in the form of nested subroutines; the first, called PACK, assembles the 8-bit character prior to transmission; the second, called XMIT, transmits this character with the necessary start and stop bits; the final one, called DELAY, determines the baud rate. The sequence of events in the PACK subroutine are shown in Figure 8 with the actual program listing shown in Table II. The other routines are discussed in Appendix A1. The PACK routine of Figure 8 ensures four characters are sequentially sent. It can be considerably simplified to allow single 4-bit data loading to any AD7542 register and also to defer updating of a DAC's analog output until required.

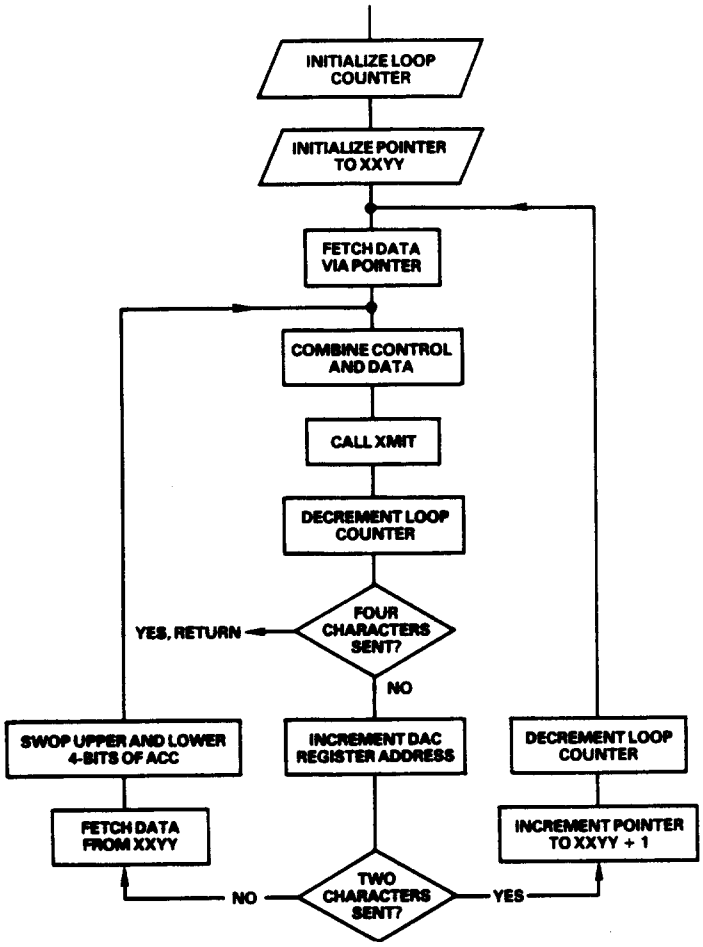

Figure 8. Sequence of Events in Pack Routine

The microprocessor used in the example is an Intel 8085A. New 12-bit data to be written to a DAC is stored in memory locations XXYY and XXYY + 1. The four most significant bits are assumed to occupy the lower half of XXYY + 1. Register pair HL acts as a pointer and allows the 12-bit data to be passed from the main user program to the nested subroutines. In the program listing of Table II, location 2070H is used for XXYY. Register D is used as a counter to ensure that four characters have been sent. The two most significant bits of Register E contain the channel address. The channel address is loaded into Register E before the program enters the nested subroutines and does not change during the program execution.

The next two most significant bits of Register E contain the AD7542 register address. Accumulator A is used to combine the conversion data loaded from XXYY (or XXYY + 1) with the control data of Register E. The combined word is then moved to Register C. Register C is used to pass the 8-bit character from the PACK subroutine to the XMIT subroutine. To completely update the four channels requires 16 characters to be transmitted. Table III shows the truth table for the control portion of the 8-bit character. The two most significant bits of the character determine which channel is to be updated. Before calling the PACK subroutine to transmit new data, the user must first initialize the E Register with the proper channel address. For instance

```
MVI E, 00
CALL PACK
```

addresses channel 1. The other channels are similarly addressed; the E Register being loaded with 40H for channel 2, 80H for channel 3 and COH for channel 4.

LOCATION	OP CODE	MNEMONIC	STATEMENT
2044 PACK	16 05	MVI D, 05	Load 05 into Loop Counter
2046	21 70 20	LXI HL 2070	Load 2070 into HL Pointer
2049	7E	MOV A,M	Fetch Conversion Data
204A	E6 OF	ANI OF	Combine Conversion Data and Control Data
204C	B3	ORA E	
2040	4F	MOV C,A	New Character Ready for Transmission
204E	CD 00 20	CALL 2000	Call XMIT Routine
2051	15	DCR D	Decrement Loop Counter
2052	C8	RZ	Finished?
2053	3E 10	MVI A, 10	Increment DAC Register Address
2055	83	ADD E	
2056	5F	MOV E,A	
2057	7A	MOV A,D	Check that LOW- and MID-Bytes Have Been Sent
2058	C6 00	ADI 00	
205A	EA 65 20	JPE 2065	
205D	7E	MOV A,M	Fetch Conversion Data
205E	OF	RRC	Swop Upper and Lower 4-Bits of ACC
205F	OF	RRC	
2060	OF	RRC	
2061	OF	RRC	
2062	C3 4A 20	JMP 204A	
2065	23	INX H	Increment Pointer
2066	15	DCR D	
2067	C3 49 20	JMP 2049	

Table II. Program Listing for PACK Routine

CHARACTER TO BE TRANSMITTED								HEX EQUIVALENT	REGISTER ADDRESSED
CONTROL				DATA					
MSB							LSB		
0	0	0	0	X	X	X	X	0X	DAC 1 Low-Byte Register
0	0	0	1	X	X	X	X	1X	DAC 1 Mid-Byte Register
0	0	1	0	X	X	X	X	2X	DAC 1 High-Byte Register
0	0	1	1	X	X	X	X	3X	DAC 1 DAC Register
0	1	0	0	X	X	X	X	4X	DAC 2 Low-Byte Register
0	1	0	1	X	X	X	X	5X	DAC 2 Mid-Byte Register
0	1	1	0	X	X	X	X	6X	DAC 2 High-Byte Register
0	1	1	1	X	X	X	X	7X	DAC 2 DAC Register
1	0	0	0	X	X	X	X	8X	DAC 3 Low-Byte Register
1	0	0	1	X	X	X	X	9X	DAC 3 Mid-Byte Register
1	0	1	0	X	X	X	X	AX	DAC 3 High-Byte Register
1	0	1	1	X	X	X	X	BX	DAC 3 DAC Register
1	1	0	0	X	X	X	X	CX	DAC 4 Low-Byte Register
1	1	0	1	X	X	X	X	DX	DAC 4 Mid-Byte Register
1	1	1	0	X	X	X	X	EX	DAC 4 High-Byte Register
1	1	1	1	X	X	X	X	FX	DAC 4 DAC Register

Table III. Truth Table For Control Data

8

APPENDIX A1

A simple latch is all that is required with the 8085A to generate the serial out stream (see Figure A1). In practice the least significant bit of a spare I/O port (Port A) on an 8355 was used as the serial out pin. This port was designated Port 00. The sequence of events for the XMIT routine is shown in Figure A2 and the corresponding program appears in Table AI.

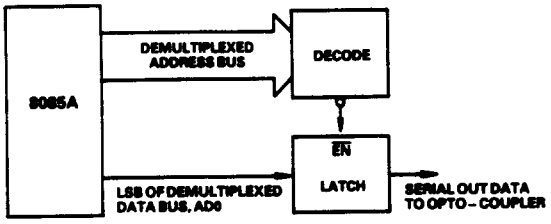

Figure A1. Serial Out Generator

The XMIT subroutine is entered with the character to be transmitted contained in register C. A bit counter is set to 11_{10} and is decremented as each bit is transmitted. The 11 bits are made up of 1 start bit, 8 data bits and 2 stop bits. The DELAY subroutine is called after each bit is transmitted (see Table AII). By varying the length of the delay different transmission rates can be achieved. For a rate of 9600 bits/second, "XX" in the delay subroutine was set to 11H (17_{10}). With this baud rate 872.7 11-bit characters per second are transmitted giving a true data rate of 6981.6 bits/second. At this speed an analog output can be updated in 4.5mS. Table AIII shows the baud rates obtained for different values of "XX" with an 8085A clock frequency of 3.1 MHz. Some provision should be made to allow the UART to clock itself out of an error condition. This could arise if the serial transmission link is broken or on power-up of the remote circuit. This requirement is met if, after transmitting four characters to one DAC, transmission ceases for a period equal to the time taken

LOCATION	OP CODE	MNEMONIC	STATEMENT
2000 XMIT	06 B	MVI B, B	Set Bit Counter to 11
2002	3E FF	MVI A, FF	Initialize Port 00 as Output
2004	D3 02	OUT 02	
2006	79	MOV A,C	Fetch Character
2007	B7	ORA A	Clear Carry
2008	17	RAL	
2009	D3 00	OUT 00	Send to Port
200B	CD 20 20	CALL 2020	Call DELAY Routine
200E	1F	RAR	Rotate Next Bit
200F	37	STC	
2010	05	DCR B	
2011	C2 09 20	JNZ 2009	Finished?
2014	C9	RET	Return

Table AI. Program Listing for XMIT Routine

to transmit a single character. During this time the serial data line must be held high to hold the RRI input high. This happens automatically in fact, since at the end of each character the line goes high for the stop bit and remains high until the low start bit of the next character occurs.

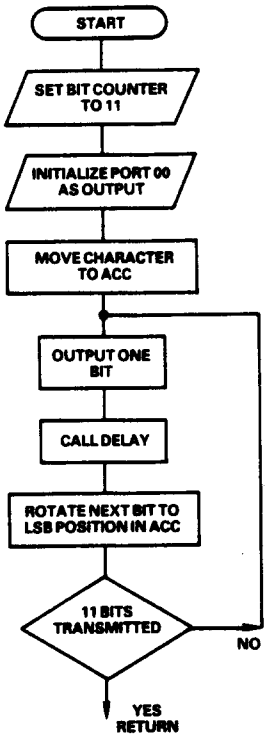

Figure A2. Sequence of Events in XMIT Routine

LOCATION	OP CODE	MNEMONIC	STATEMENT
2020 DELAY	D5	PUSH DE	Save DE on Stack
2021	1E XX	MVI E, XX	Load Required Baud Rate
2023	1D	DCR E	Delay
2024	C2 23 20	JNZ 2023	Finished?
2027	D1	POP DE	Pop DE from Stack
2028	C9	RET	Return

Table AII. Program Listing for DELAY Routine

BAUD RATE	"XX"
9600 bits/sec	11H
4800 bits/sec	27H
2400 bits/sec	56H
1200 bits/sec	BOH

Table AIII. Baud Rate Variation vs. XX for Delay Routine

ACKNOWLEDGEMENTS

To Phil Burton for originally suggesting the idea and also to Ray Speer for his help.

REFERENCES

1. Phil Burton, "CMOS D/A Converter Circuits for Single +5V Supplies", PUB. NO. E828-10-4/84
2. CMOS DAC Application Guide, PUB. NO. G872-30-10/84