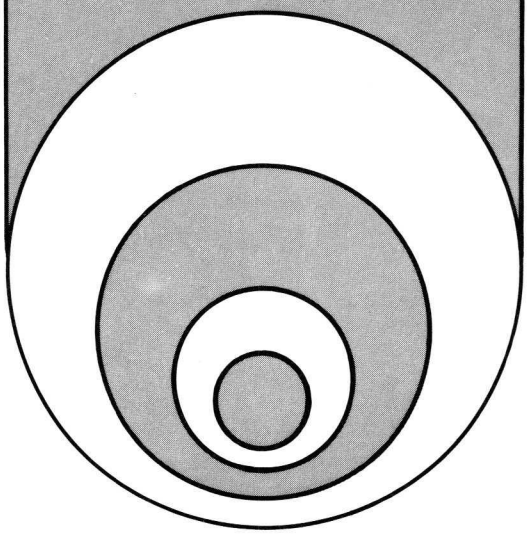
signetics

MOS
MICROPROCESSOR



GENERAL
DELAY
ROUTINES
AS52

SUMMARY

In microprocessing applications, delay times are often required. A typical example is a delay time for a serial Teletypewriter interface. While delay times can be generated by counters, monostables, multivibrators, and other hardware, it is often simpler and more economical to use a short software routine.

This applications memo describes several ways of writing software delay time routines for the Signetics 2650 microprocessor. Time restrictions and formulas for calculating the delay time are given for each routine.

DELAY ROUTINES

In general, a delay can be implemented by setting a counter with a number N and decrementing this number by one until it is zero. If decrementing the number takes one clock period, then the total delay time is N clock periods.

In the 2650 microprocessor, the internal registers may be used as counters. The most useful instructions for decrementing are the "Branch on Decrementing Register" (BDRR and BDRA) instructions, which also test the content of a register for zero.

Figure 1 illustrates a flowchart of a delay routine. This routine consists of a setup part and a count loop. The count loop will be executed n times and the setup only once. Hence, the delay time is:

$$t_d = t_{su} + n \cdot t_{ct}$$

It is possible to increase the delay time by increasing n or by making t_{ct} longer. The latter can be done by inserting a fixed delay such as a No Operation (NOP) instruction in the count loop.

DELAY ROUTINE FLOWCHART

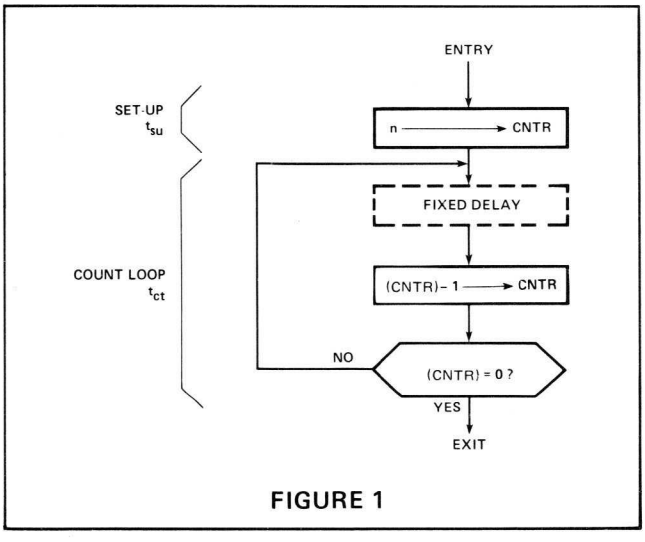


FIGURE 1

The program of the routine shown in Figure 1 is as follows:

	LODI, Rx n	Load n into register Rx	6 cp*
LOOP	NOP	No operation; fixed delay of 6 cp	6 cp
	BDRR, Rx LOOP	Decrement Rx; branch to loop if the result is not zero	9 cp

*cp = clock periods

With one NOP, the delay time is: $t_d = (6 + 15 \cdot n)$ cp. Without the NOP, the delay time is: $t_d = (6 + 9 \cdot n)$ cp. The maximum delay time is obtained when Rx is loaded with zero, since Rx will cycle through all the 256 possible states. When Rx = R0, the LODI, R0 0 instruction can be replaced by the EORZ R0 instruction, which saves one byte of code.

DELAY ROUTINE WITH FOUR REGISTERS

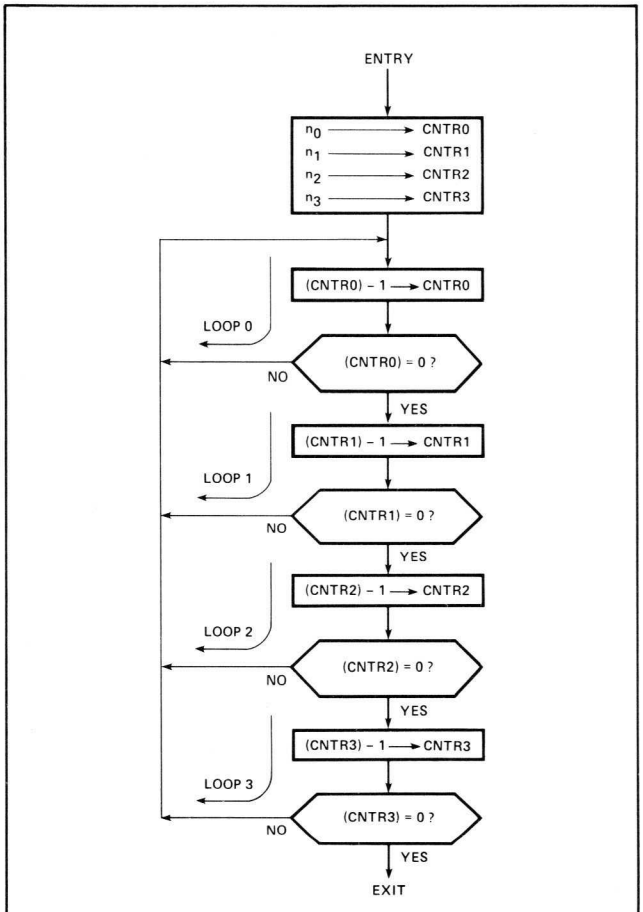


FIGURE 2

Another possible way of increasing the delay time is to repeat the count loop of Figure 1 several times. This can be done by repeating the instructions or by counting the repetitions of the count loop in another register. For example, this latter method can be expanded to include four internal registers. A flowchart of a delay routine using this technique is illustrated in Figure 2.

The number of times the processor executes the different loops shown in Figure 2 are:

loop 3 n_3

loop 2 $n_2 + (n_3 - 1)\ 256$

loop 1 $n_1 + (n_2 - 1)\ 256 + (n_3 - 1)\ 256^2$

loop 0 $n_0 + (n_1 - 1)\ 256 + (n_2 - 1)\ 256^2 + (n_3 - 1)\ 256^3$

Hence, the delay time of this routine is:

$$t_d = [24 + \{n_0 + n_1 + (n_1 - 1)\ 256 + n_2 + (n_2 - 1)\ (256 + 256^2) + n_3 + (n_3 - 1)\ (256 + 256^2 + 256^3)\}\ 9]\ cp$$

(If Rx is loaded with a zero, then n = 256 in the formula):

Table 1 shows six different delay routine programs along with specifications for each program. The delay time for these routines can be computed from the following equations.

Routine	Delay Time
a	$t_d = (6 + 9 \cdot n_0)\ cp$
b	$t_d = (6 + 15 \cdot n_0)\ cp$
c	$t_d = (2310 + 9 \cdot n_0)\ cp$
d	$t_d = \{12 + [n_0 + n_1 + (n_1 - 1)\ 256]\ 9\}\ cp$
e	$t_d = \{18 + [n_0 + n_1 + (n_1 - 1)\ 256 + n_2 + (n_2 - 1)\ (256^2 + 256)]\ 9\}\ cp$
f	$t_d = \{24 + [n_0 + n_1 + (n_1 - 1)\ 256 + n_2 + (n_2 - 1)\ (256^2 + 256) + n_3 + (n_3 - 1)\ (256^3 + 256^2 + 256)]\ 9\}\ cp$

TABLE 1

ROUTINE	POSSIBLE DELAY TIME (cp)		DELAY STEP (cp)	NUMBER OF BYTES	NUMBER OF REGISTERS	PROGRAM
	MIN*	MAX				
a	15	2310	9	4	1	LODI, R0 n_0 LOOP BDRR, R0 LOOP
b	21	3846	15	5	1	LODI, R0 n_0 LOOP NOP BDRR, R0 LOOP
c	2319	4614	9	6	1	LODI, R0 n_0 LOP 1 BDRR, R0 LOP 1 LOP 2 BDRR, R0 LOP 2
d	30	592.140	9	8	2	LODI, R0 n_0 LODI, R1 n_1 LOOP BDRR, R0 LOOP BDRR, R1 LOOP
e	45	≈151.6 x 10^6 **	9	12	3	LODI, R0 n_0 LODI, R1 n_1 LODI, R2 n_2 LOOP BDRR, R0 LOOP BDRR, R1 LOOP BDRR, R2 LOOP
f	60	≈38.8 x 10^9 ***	9	16	4	LODI, R0 n_0 LODI, R1 n_1 LODI, R2 n_2 LODI, R3 n_3 LOOP BDRR, R0 LOOP BDRR, R1 LOOP BDRR, R2 LOOP BDRR, R3 LOOP

* cp = clock period. For 1MHz clock 1 cp = 1 μs.

** For 1MHz clock this is about 2.5 minutes.

*** For 1MHz clock this is about 10.46 hours.