

## Physics 116B Winter 2005: Problem Set 9

1. Write a MAS program as follows based on the example in the M68000 programming notes.
  - (a) Start with an array of 40 ASCII characters in the array msg (use the dc.b directive to set up the 40 character array rather than reading it from the keyboard). The idea is to form a sum of any characters that can be interpreted as hex digits. Thus A-F (or a-f) are also considered numbers.
  - (b) Recognize the characters with a subroutine (name it cnvhex). When you call the subroutine, the character to be tested should be in the register d0. On return from the subroutine, d0 should contain the numerical value of the character for a valid hex character (range 0-15) or 0 if the character is not a valid hex character.
  - (c) The main program will step through the msg array, call the subroutine for each character and total the sum.
  - (d) At the end of the main program, display the sum on the screen.

A table of ASCII characters is included below (example: 'A' = \$41):

Least Significant Digit	Most Significant Digit							
	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	'	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	\$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

From MC68000 Programmer's Ref. Manual, © Motorola, 1992

2. Consider the attached M68000 MAS program. The lines are numbered in the left column.
- (a) Does line 6 translate directly into M68000 assembly language (like line 10, for example)? If not, why is it there?
  - (b) The M68000 has many addressing modes. What mode of addressing is used for the destination address in the move.b instruction in line 32?
    - i. Explain how this mode of addressing works to achieve the desired result of entering successive characters into memory.
    - ii. Is there any reason not to use register A5 for this purpose throughout this Mac program instead of A1? Explain what A5 is used for. Would the instruction in line 37 use A5 in some manner?
  - (c) What kind of addressing is used for the source address in line 38? Where does this information (the number 37) reside prior to the execution of the instruction (i.e., in the data area, one of the registers or in the program area)? explain briefly.
  - (d) Explain how the instruction on line 33 works to control the loop. Under what specific circumstance will the instruction on line 37 be executed?
  - (e) Does any branch instruction perform a logic test which makes use of the Z bit in the condition code register? If so, which one(s)? (Hint: see Table 3-19 in the M68000 writeup).
  - (f) Is the run-time stack used by this program? If so, how or why? At any time, how is the last occupied memory location in the stack indicated?
  - (g) Modify the program so it changes lower case letters to upper instead of vice versa. Just list the lines which you change. (An ASCII chart is given in Prob. 1.)
  - (h) Modify the program to store the original characters in msg and the message with possibly changed characters in a new array called newmsg. You will have to add some lines. Indicate what they are and where they go (i.e., “insert xxx after line yy” etc.).

Printed: Wednesday, March 9, 2005 7:24:49 PM

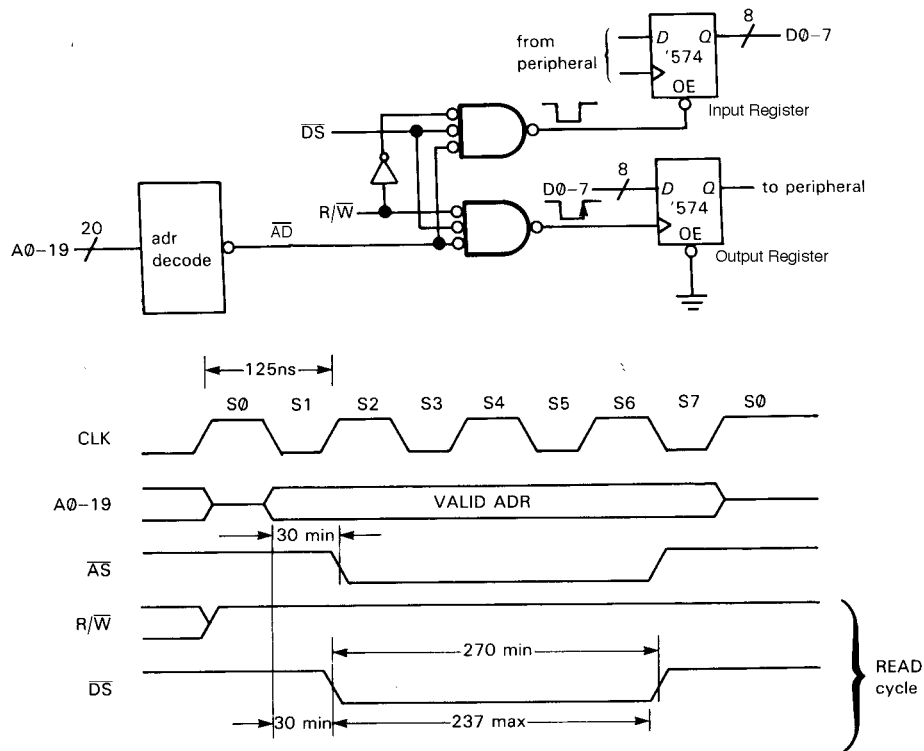
```
1 ; Program to read up to 100 ASCII characters, sum up any integers, change upper to lower case
2 ; and store in msg.
3 ; Combined solution to exercises 1-3 of M68000 Programming Notes, 6/10/99
4 ; -D. Pellett 3/18/04 (beta version)
5
6     xref getchar, strout, decin, decout, newline, stop
7
8 start:  lea    msg1,a0          ; ask for number of characters to read (100 max)
9         move.w #50,d0
10        jsr    strout
11        jsr    decin          ; get the number (in d0)
12        jsr    newline
13        cmp.w  #100,d0        ; see if it fits
14        ble   fits           ; it fits - branch
15        move.w #100,nchar     ; too big - set nchar = 100
16        lea   msg2,a0        ; send message to input 100 max
17        move.w #46,d0
18        jsr    strout
19        jsr    newline
20        jmp   readem
21 fits:  move.w d0,nchar       ; enter number in nchar
22        lea   msg3,a0        ; send message to input characters
23        move.w #26,d0
24        jsr    strout
25        jsr    newline
26 readem: lea   msg,a1         ; put address of msg in a1
27        move.w nchar,d1      ; number of bytes to read
28        clr   d2             ; clear the register for the sum
29        jmp   enter         ; enter loop at end
30 loop:  jsr    getchar       ; getchar puts the character in d0
31        jsr    addch         ; subroutine to test and add integers, change case
32        move.b d0,(a1)+     ; move the character to memory
33 enter: dbra   d1,loop       ; subtract 1 from d1 and see if done
34
35 ; now output the information
36
37        lea   msg4,a0
38        move.w #37,d0        ; output info message
39        jsr    strout
40        jsr    newline
41        lea   msg,a0         ; set up for outputting character string
42        move.w nchar,d0
43        jsr    strout        ; output the string
44        jsr    newline
45        lea   msg5,a0        ; output sum message
46        move.w #31,d0
47        jsr    strout
48        move.w d2,d0         ; output the sum
49        jsr    decout
50        jsr    newline
51        jsr    stop         ; end of program
52
53
54
55
56 ; Subroutine addch tests ASCII character in d0
57 ; If it is a number, adds it to the sum in d2
```

```
58 ; If it is an upper case letter, changes it to lower case
59 ; Lower case letter or number is returned in d0
60
61 addch: and.b   #$7F,d0      ; mask off parity bit of character
62        cmp.b   #$30,d0     ; see if it is less than $30
63        blt     skip        ; if so, skip
64        cmp.b   #$39,d0     ; see if it is greater than $39
65        bgt     skip        ; if so, skip
66        move.w  d0,d3       ; is a number - put in d3 (don't change d0)
67        and.w   #$000F,d3   ; get the number by masking off upper bits
68        add.w   d3,d2       ; add to sum in d2
69        rts                    ; return from subroutine
70 skip:  cmp.b   #$41,d0     ; see if it is less than $41 (A)
71        blt     skip1       ; if so, skip1
72        cmp.b   #$5A,d0     ; see if it is greater than $5A (Z)
73        bgt     skip1       ; if so, skip1
74        add.w   #$20,d0     ; upper case letter - convert to lower
75 skip1: rts
76
77        data
78 msg1:  dc.b   'Enter the number of characters to read (100 max): '
79 msg2:  dc.b   'Your entry was too big. Enter 100 characters: '
80 msg3:  dc.b   'Now enter the characters: '
81 msg4:  dc.b   'Here is the string (all lower case): '
82 msg5:  dc.b   'The sum of numbers entered is: '
83 msg:   ds.b   100          ; set aside 100 bytes of storage
84 nchar: dc.w   20           ; number of characters to read
85        even
86        end
```

3. Diagrams below from Horowitz and Hill show connections of an input register and an output register to an M68008 microprocessor. Also shown is a write cycle timing diagram with  $R/\overline{W}$ , VALID ADDRESS (lumped together as usual), etc. Specifications for the 74HCT574 octal D latch are attached.

(a) On the timing diagram, show the address decoder output,  $\overline{AD}$ , and the  $\overline{OE}$  signal on the input register octal D latch when it is accessed with its valid address during a read cycle. Also show the data bus signals from the register (D0-7 lumped together like A0-19).

(b) Explain why the input register must have tri-state output lines.



# SN54HCT574, SN74HCT574 OCTAL EDGE-TRIGGERED D-TYPE FLIP-FLOPS WITH 3-STATE OUTPUTS

SCLS177E – MARCH 1984 – REVISED AUGUST 2003

- Operating Voltage Range of 4.5 V to 5.5 V
- High-Current 3-State Noninverting Outputs Drive Bus Lines Directly or Up To 15 LSTTL Loads
- Low Power Consumption, 80- $\mu$ A Max  $I_{CC}$
- Typical  $t_{pd} = 22$  ns
- $\pm 6$ -mA Output Drive at 5 V
- Low Input Current of 1  $\mu$ A Max
- Inputs Are TTL-Voltage Compatible
- Bus-Structured Pinout

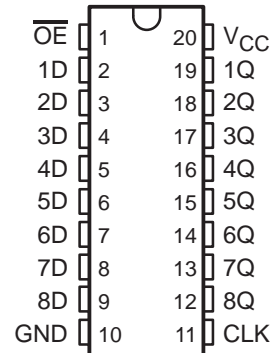
## description/ordering information

These octal edge-triggered D-type flip-flops feature 3-state outputs designed specifically for bus driving. The 'HCT574 devices are particularly suitable for implementing buffer registers, I/O ports, bidirectional bus drivers, and working registers.

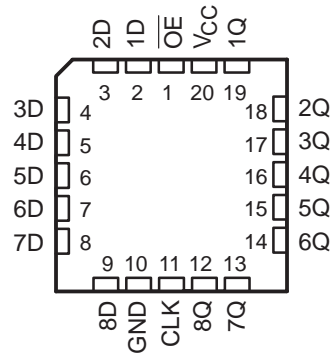
The eight flip-flops enter data on the low-to-high transition of the clock (CLK) input.

A buffered output-enable ( $\overline{OE}$ ) input can be used to place the eight outputs in either a normal logic state (high or low logic levels) or the high-impedance state. In the high-impedance state, the outputs neither load nor drive the bus lines significantly. The high-impedance state and increased drive provide the capability to drive bus lines without interface or pullup components.

SN54HCT574 . . . J OR W PACKAGE  
SN74HCT574 . . . DB, DW, N, NS, OR PW PACKAGE  
(TOP VIEW)



SN54HCT574 . . . FK PACKAGE  
(TOP VIEW)



## ORDERING INFORMATION

$T_A$	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
-40°C to 85°C	PDIP – N	Tube of 20	SN74HCT574N	SN74HCT574N
	SOIC – DW	Tube of 25	SN74HCT574DW	HCT574
		Reel of 2000	SN74HCT574DWR	
	SOP – NS	Reel of 2000	SN74HCT574NSR	HCT574
	SSOP – DB	Reel of 2000	SN74HCT574DBR	HT574
	TSSOP – PW	Tube of 70	SN74HCT574PW	HT574
		Reel of 2000	SN74HCT574PWR	
Reel of 250		SN74HCT574PWT		
-55°C to 125°C	CDIP – J	Tube of 20	SNJ54HCT574J	SNJ54HCT574J
	CFP – W	Tube of 85	SNJ54HCT574W	SNJ54HCT574W
	LCCC – FK	Tube of 55	SNJ54HCT574FK	SNJ54HCT574FK

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at [www.ti.com/sc/package](http://www.ti.com/sc/package).



Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

UNLESS OTHERWISE NOTED this document contains PRODUCTION DATA information current as of publication date. Products conform to specifications per the terms of Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

 **TEXAS  
INSTRUMENTS**

POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

Copyright © 2003, Texas Instruments Incorporated

# SN54HCT574, SN74HCT574 OCTAL EDGE-TRIGGERED D-TYPE FLIP-FLOPS WITH 3-STATE OUTPUTS

SCLS177E – MARCH 1984 – REVISED AUGUST 2003

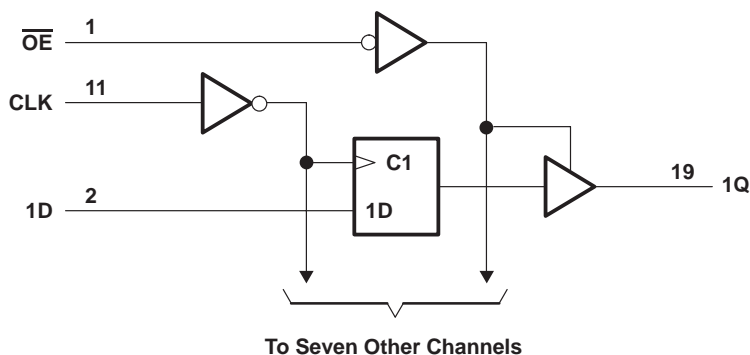
## description/ordering information (continued)

$\overline{OE}$  does not affect the internal operations of the flip-flops. Old data can be retained or new data can be entered while the outputs are in the high-impedance state.

FUNCTION TABLE  
(each flip-flop)

INPUTS			OUTPUT
$\overline{OE}$	CLK	D	Q
L	↑	H	H
L	↑	L	L
L	H or L	X	$Q_0$
H	X	X	Z

## logic diagram (positive logic)

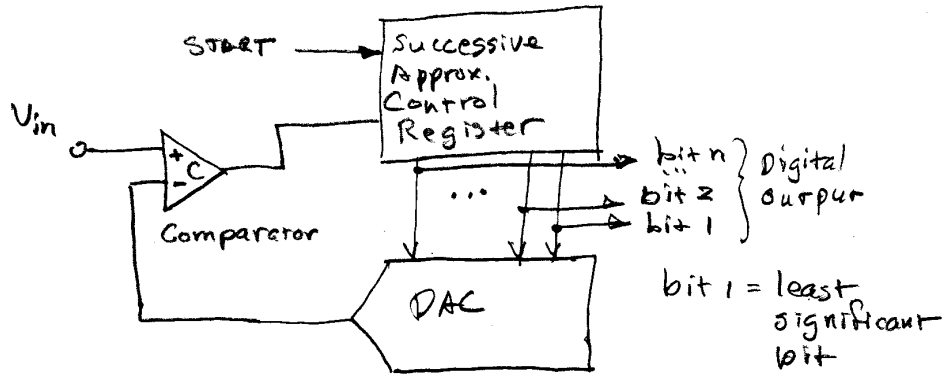


## absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Supply voltage range, $V_{CC}$	-0.5 V to 7 V
Input clamp current, $I_{IK}$ ( $V_I < 0$ or $V_I > V_{CC}$ ) (see Note 1)	$\pm 20$ mA
Output clamp current, $I_{OK}$ ( $V_O < 0$ or $V_O > V_{CC}$ ) (see Note 1)	$\pm 20$ mA
Continuous output current, $I_O$ ( $V_O = 0$ to $V_{CC}$ )	$\pm 35$ mA
Continuous current through $V_{CC}$ or GND	$\pm 70$ mA
Package thermal impedance, $\theta_{JA}$ (see Note 2):	
DB package	70°C/W
DW package	58°C/W
N package	69°C/W
NS package	60°C/W
PW package	83°C/W
Storage temperature range, $T_{stg}$	-65°C to 150°C

† Stresses beyond those listed under “absolute maximum ratings” may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under “recommended operating conditions” is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

- NOTES: 1. The input and output voltage ratings may be exceeded if the input and output current ratings are observed.  
2. The package thermal impedance is calculated in accordance with JESD 51-7.



4. A block diagram of a successive approximation ADC is shown above. Bit 1 is the least significant bit of the output binary number. The voltage to be measured is  $v_{in}$ .
- Consider a 3 bit ADC as an example ( $n=3$  in the diagram above). What is the first binary number sent to the DAC by the successive approximation control register?
    - If the comparator output is low, what 3 bit number is sent next?
    - As a result, if the comparator output is now high, what number is sent next?
    - Now the comparator output is low. What is the 3 bit ADC value for this input voltage?
  - What is the main advantage of this ADC over a simple counting (or counter-ramp) ADC?
  - Suppose the number of bits is 8 instead of 3. What advantage does a *flash ADC* have over this design? What is the main disadvantage? Explain briefly.