Real Time Clock

Operators Manual

TransEra

E. J. Warchol
11-1-83

MODEL NO. 1741 - RTC
SERIAL # RO - 1835

11-1-83

TEK - 4052/4054

Real Time Clock

Operators Manual

TransEra

3707 North Canyon Road • Suite 4
Provo, Utah 84604

Phone 801-224-6550

PREFACE

This manual describes the installation and operation of the TransEra 641 and 741 Real Time Clock ROM pack for the 4050 Tektronix Graphic Computer Systems. It is organized into five sections and two appendices.

Section 1 describes the installation and use of the Real Time Clock ROM pack.

Section 2 describes the commands used to set the clock date, the time and the clock modes.

Section 3 describes the commands used to read the date and time. It also includes commands for date manipulation.

Section 4 describes the interrupt and timer commands.

Section 5 describes the CMOS memory commands.

Appendix A is a summary of the Real Time Clock Commands.

Appendix B is the complete example program and sample output.

Release 1.0 16-MAY-83

(C) - Copyright TransEra Corporation 1983, All Rights Reserved.

TABLE OF CONTENTS

SECTION ONE

GENERAL DESCRIPTION

This section contains the general description of the Real Time Clock ROM pack, the specifications, the installation instructions, and the getting started tutorial.

Section Contents

INTRODUCTION

The TransEra Model 641/741 Real Time Clock combines high level BASIC commands with a low power CMOS Real Time Clock circuit.

The high level BASIC commands allow one to set and read the date and time, specify the time format and mode, specify timed program interrupts, and store and retrieve data in the on board CMOS memory.

All the Real Time Clock BASIC commands use an exclamation point in the name to avoid conflicting with any other BASIC commands.

The date, time or both may be read from the real time clock into either BASIC strings or numbers. Either 12 or 24 hour format may be selected. The clock may be set to daylight savings time and back again.

Several date manipulation routines are included that convert between gregorian and julian dates.

Interrupt capabilities range from repeating periodic interrupts to single event interrupts with absolute or delta time settings.

The battery powered CMOS circuits keep the time intact even when the system power is off. Fourty bytes of battery powered CMOS memory provides all kinds of possiblities for data storage limited only by the imagination.

A user selectable system password may be activated to prevent unauthorized machine usage, and system start-up commands may be defined that will automatically execute each time you power-up.

SPECIFICATIONS

Resolution:

Clock	0.01 seconds
Timer	1 second

Physical Characteristics:

Weight: 114g (4oz)

Dimensions:

length	13.5 cm (5.5in)
width	6.6 cm (2.6in)
depth	2.3 cm (0.9in)

Operating temperature:
10-40 degrees Centigrade

Standard Accessories:
Operators Manual

INSTALLATION INSTRUCTIONS

The power to the 4050 should be turned off before the ROM pack is installed. After the power is shut off, it may be inserted into the firmware backpack or into a slot of the ROM Expander unit.

The Real Time Clock ROM pack should be placed into the lowest priority slot in the system. This is determined by the ROM bank number: the higher the bank number the lower the slot priority.

With the 4050E01 ROM expander installed, the slot priority may be determined from the following table.

System Priority		Backpack Slot Number	ROM Bank Number
1st	Highest	61	0
2nd		71	8
3rd		Extended Slots 61-68	16-23
4th		Extended Slots 71-78	24-31
5th		41	32
6th		51	40
7th		Extended Slots 41-48	48-55
8th	Lowest	Extended Slots 51-58	56-63

Standing in front of the 4050, the ROM Pack should be held so that the label side is showing in the upright position. This is the correct orientation for inserting the ROM Pack. Press down gently until the edge of the card connector is seated in the receptacle connector.

Note that the memory and/or operation of the 4050 may be disrupted if the 641/741RTC is inserted or removed while the power is on.

The following discussion shows how to use some of the simple Real Time Clock commands. A sample program is found in Appendix B.

To set the date and time the !SETTIME command is used as follows.

```
CALL "!SETTIME","15-APR-83 16:36:02"
```

Notice that the date is entered with a three character month, a two digit year, and that the time is specified using a 24 hour format. This sets the clock to the default 24 hour format and standard time modes.

You can display the date and time with the !DATETIME command as follows.

```
CALL "!DATETIME"
```

The system responds with: 15-APR-83 16:36:06

The date and time may also be returned in a string as follows.

```
CALL "!DATETIME",A$
```

The date may be displayed with the !DATE command.

```
CALL "!DATE"
```

The system responds with: 15-APR-83

The date may also be returned in either a string or three numbers as follows.

```
CALL "!DATE",A$
      or
CALL "!DATE",D,M,Y
```

The time can be displayed with the !TIME command.

```
CALL "!TIME"
```

The system responds with: 16:36:22.23

The time may also be returned in either a string or four numbers as follows.

```
CALL "!TIME",A$
      or
CALL "!TIME",H,M,S,B
```

The clock hour format and time mode may be displayed with the !MODES command.

```
CALL "!MODES"
```

The system responds with: 24 Hour Standard Time Mode.

Use the !12HOUR and !24HOUR commands to set the clock hour format. Use the !DAYLIGHT command to set the clock to daylight savings time mode and the !STANDARD command to set it to standard time mode.

Routines are provided that return the julian day, day-of-the-week, and seconds in the day. A routine is provided that will convert a julian day back to a day and month. These routines are described in section three.

The program may be delayed for a specified number of seconds by using the !PAUSE command as follows.

```
CALL "!PAUSE",20
```

This would delay the program for 20 seconds.

Three types of timer interrupt routines are available. They are !PERIOD, !TIMER and !ALARM. Each is described in detail in section four.

The Real Time Clock contains 40 bytes of CMOS memory storage. This memory is powered by the clock battery. These storage locations may be read and written using the !READ and !WRITE commands. The following examples demonstrate these commands.

```
CALL "!WRITE",4,1
CALL "!READ",N,1
N
```

The system responds with: 4

A string may also be read and written in the CMOS memory as follows.

```
CALL "!WRITE","HELLO",1
CALL "!READ",B$,5,1
B$
```

The system responds with: HELLO

Two routines, !SETPU and !SETPS described in section five, allow one to specify a Power-Up program or password.

The ROM pack firmware level may be displayed on the system screen by using the 641RTC or 741RTC command as follows.

```
CALL "641RTC"          for a 4051

  or

CALL "741RTC"          for a 4052 or 4054
```

The system responds with the following message.

```
741RTC Firmware Level 1.0 1830415
```

This information identifies the firmware level installed in your Real Time Clock ROM pack. This information will be required if you need to report any problems to the factory.

SECTION TWO

CLOCK SET UP COMMANDS

This section describes the commands used to set the date, time and clock time modes. The !SETTIME command is used to set the date and time. Once the date and time have been set, there are four commands that select the clock operating mode for operation in either 12 or 24 hour format and either standard time or daylight savings time mode. Once set, the clock continues to operate in the selected format and mode as long as there is enough battery power. The current clock format and time modes may be displayed with the !MODES command.

The 641/741RTC command will display the current clock firmware level number. This number is used to identify the ROM pack when reporting problems.

Section Contents

!SETTIME Set Clock Date and Time

CALL "!SETTIME",t$ "31-OCT-84 8:8:0"

t$ = Date and Time String (string variable or literal)

Purpose: This command sets the clock date and time. The 24 hour format and standard time modes are set.

General Information:
The string in this command contains the date and time to which the clock will be set. The format for this string is:

"DD-MON-YY HH:MM:SS" (i.e. "01-APR-83 10:00:02")

Where DD is a one or two digit day number, MON is a three character month representation from the following list, YY is a two digit year number, HH is a one or two digit hour number, MM is a one or two digit minute number and SS is a one or two digit second number.

Note: Twenty-four hour time format is used to set PM times, Leading zeros may be omitted and the seconds field is optional.

Months:		
	JAN	January
	FEB	February
	MAR	March
	APR	April
	MAY	May
	JUN	June
	JUL	July
	AUG	August
	SEP	September
	OCT	October
	NOV	November
	DEC	December

Example:

This example uses a literal string to set the date and time in line number 100 and displays the clock setting in line 120.

```
100 CALL "!SETTIME","01-MAY-83 10:02:00"
110 CALL "!DATETIME"

RUN
01-MAY-83 10:02:01
```

!MODES Display the Real Time Clock Operating Modes

```
CALL "!MODES"
```

Purpose: This command displays on the graphic computer system screen the current clock hour format and time modes.

General Information:
The current setting of the Real Time Clock hour format and clock time modes is displayed by this command.

Example:

This example shows the current clock modes.

```
CALL "!MODES"

24 Hour Standard Time Mode
```

This example sets the Real Time Clock to 12 hour and Daylight Savings time modes and then displays the current clock modes on the screen.

```
CALL "!12HOUR"
CALL "!DAYLIGHT"
CALL "!MODES"

12 Hour Daylight Savings Time Mode
```

!12HOUR Set Clock to 12 Hour Format

```
CALL "!12HOUR"
```

Purpose: This command sets the Real Time Clock to 12 Hour format.

General Information:
The current hour will be converted to the correct hour for display in 12 hour format. All time strings will have AM or PM appended when in 12 hour format mode.

Example:

This example sets the clock to 12 hour format in line 100, displays the time in line 110 and displays the current clock modes in line 120.

```
100 CALL "!12HOUR"
110 CALL "!TIME"
120 CALL "!MODES"

RUN
12:35:06.26 PM
12 Hour Standard Time Mode
```

!24HOUR Set Clock to 24 Hour Format

```
CALL "!24HOUR"
```

Purpose: This command sets the Real Time Clock to 24 Hour format.

General Information:
The current hour will be converted to the correct hour for display in 24 hour format.

Example:

This example sets the clock to 24 hour format in line 100, displays the time in line 110 and displays the current clock modes in line 120.

```
100 CALL "!24HOUR"
110 CALL "!TIME"
120 CALL "!MODES"

RUN
12:35:06.26
24 Hour Standard Time Mode
```

!STANDARD Set Clock to Standard Time Mode

```
CALL "!STANDARD"
```

Purpose: This command sets the Real Time Clock to standard time mode.

General Information:
The current hour is not altered.

Example:

This example sets the clock to standard time mode in line 100 and displays the current clock modes in line 110.

```
100 CALL "!STANDARD"
110 CALL "!MODES"

RUN
24 Hour Standard Time Mode
```

!DAYLIGHT Set Clock to Daylight Savings Time Mode

```
CALL "!DAYLIGHT"
```

Purpose: This command sets the Real Time Clock to daylight savings time mode.

General Information:
Setting the daylight savings time mode will not alter the current hour. When in daylight savings time mode the clock will perform two special updates. On the last sunday in April the time increments from 1:59:59 AM to 3:00:00 AM. On the last sunday in October when the time first reaches 1:59:59 AM it changes to 1:00:00 AM.

Example:

This example sets the clock to daylight savings time mode in line 100 and displays the clock modes in line 110.

```
100 CALL "!DAYLIGHT"
110 CALL "!MODES"

RUN
24 Hour Daylight Savings Time Mode
```

641/741RTC Display Firmware Level

CALL "641RTC" For 4051
CALL "741RTC" For 4052 or 4054

Purpose: This command displays on the Graphic Computer System screen the firmware level of the Real Time Clock.

NOV.18,1983

741 RTC FIRMWARE LEVEL 1,2 1830707

General Information:
The firmware level of the Real Time Clock is displayed on the screen. This information is used when reporting problems.

Example:

This example displays the firmware level as 1.0 and the release code number.

```
CALL "741RTC"

741RTC Firmware Level 1.0 1830415
```

SECTION THREE

CLOCK READ COMMANDS

This section describes the commands that read the real-time-clock. Provision is made to allow one to read either the date, time or both. Additional commands are provided that allow one to read the number of seconds in the day, the day-of-the-week in either a numeric format or a string, and the julian day. Commands are included to convert from a month and day to a julian day and back to a month and day.

Section Contents

!DATETIME Display or Return Current Date and Time

```
CALL "!DATETIME",[t$]
```

t$ = Optional target string for date and time.

Purpose: This command returns the current date and time from the clock to the target string variable or to the graphic computer system screen if the variable is omitted.

General Information:
The length of the string returned is 20 characters when the clock is in 24 hour format and 23 characters when in 12 hour format. The string format is:

"DD-MON-YY HH:MM:SS.BB [AM/PM]"

Where DD=Day, MON=Three character Month, YY=Year
HH=Hour, MM=Minute, SS=Second, BB=Sub-Seconds
AM=Optional AM/PM in 12 hour format clock mode

The sub-seconds return the current hundreths of a second. If the period timer is being used the sub-seconds are returned as zeros.

Example:

This example returns the current date and time into string A$ in line 110 and displays the date and time in line 120.

```
100 DIM A$(25)
110 CALL "!DATETIME",A$
120 PRINT A$

RUN
01-MAY-83 12:24:05 PM
```

!DATE Display or Return Current Date

CALL "!DATE",[d$]

d$ = Optional target string for date.

CALL "!DATE",d,m,y

d = day (1-31), m = month (1-12) and y = year (0-99).

Purpose: This command returns the current date from the clock to the target string or numeric variables or to the graphic computer system screen if the variables are omitted.

General Information:
The length of the message returned is 9 characters.
The string format is:

"DD-MON-YY HH:MM:SS.BB [AM/PM]"

Where DD=Day, MON=Three character Month, YY=Year

Example:

This example returns the current date into string A$ in line 110 and displays the date in line 120.

```
100 DIM A$(10)
110 CALL "!DATE",A$
120 PRINT A$
```

PRINT@55; A$

```
RUN
01-MAY-83
```

!TIME Display or Return Current Time

CALL "!TIME",[t$]

t$ = Optional target string for time.

CALL "!TIME",h,m,s,b

h = hour, m = minute, s = second and b = sub-seconds.

Purpose: This command returns the current time from the clock to the target string or numeric variables or to the graphic computer system screen if the variables are omitted.

General Information:
The length of the string returned is 11 characters when the clock is in 24 hour format and 14 characters when in 12 hour format. The string format is:

"HH:MM:SS.BB [AM/PM]"

Where DD = Day, MON = Three character Month, YY = Year
HH = Hour, MM = Minute, SS = Second, BB = Sub-Seconds
AM/PM = Optional AM/PM in 12 hour format clock mode

When the time is returned in numeric variables, the hour is always returned in 24 hour format.

The sub-seconds return the current hundreths of a second. If the period timer is being used the sub-seconds are returned as zeros.

Example:

This example returns the current time into string A$ in line 110 and displays the time in line 120.

```
100 DIM A$(15)
110 CALL "!TIME",A$
120 PRINT A$

RUN
12:26:05
```

!SECONDS Return Seconds in the Day

```
CALL "!SECONDS",n

    n = current second in the day

CALL "!SECONDS",h,m,s,n

  h = hour, m = minute, s = second and n = second in the day
```

Purpose: This command returns the current second in the day to the target variable from the clock or it calculates the second in the day from the hours, minutes and seconds specified.

General Information:

The second in the day is calculated as follows:

```
( ( (Hour*60)+ Minute) *60) + Second
```

The hours are always specified in 24 hour format.

Example:

This example returns the current second in the day into N in line 100, this value is displayed in line 110. Then the second in the day is calculated from the time specified as 12:23:15 and returned in N in line 120 and displayed in line 130.

```
100 CALL "!SECONDS",N
110 PRINT N
120 CALL "!SECONDS",12,23,15,N
130 PRINT N
140 END

RUN
37420
44595
```

!DAY Return Julian Day

```
CALL "!DAY",n

   n = current julian day

CALL "!DAY",d,m,y,j

   d = day, m = month and y = year and j = julian day
```

Purpose: This command returns the current julian day to the target variable from the clock or it calculates the julian day from the day, month and year specified.

General Information:

The julian day is the number of the day in the year. The year is used to adjust the julian day for leap years.

Example:

This example returns the current julian day into N in line 100, this value is displayed in line 110. Then the julian day is calculated from the date specified as 31-DEC-84 and returned in N in line 120 and displayed in line 130.

```
100 CALL "!DAY",N
110 PRINT N
120 CALL "!DAY",31,12,84,N
130 PRINT N

RUN
110
366
```

!MONTH Return Day and Month from Julian Day and Year

```
CALL "!MONTH",j,y,d,m

   j = julian day, y = year, d = day and m = month
```

Purpose: This command returns the day and month from the julian day and year.

General Information:
The number of the day in the year is used to calculate the month and the day in the month. The year is used to adjust for leap years.

Example:

This example returns the day and month into D and M in line 100, the day and month are displayed in line 110.

```
100 CALL "!MONTH",366,84,D,M
110 PRINT D,M

RUN
31       12
```

!WEEKDAY Return Day of the Week

```
CALL "!WEEKDAY",w

    w = day-of-week number

CALL "!WEEKDAY",d,m,y,w

    d = day, m = month, y = year and w = day-of-week number

CALL "!WEEKDAY",w$

    w$ = day-of-week name

CALL "!WEEKDAY",d,m,y,w$

    d = day, m = month, y = year and w$ = day-of-week name
```

Purpose: This command returns either the current day-of-the-week or it calculates the day-of-the-week from the specified date.

General Information:
The day-of-the-week may be returned in either a numeric variable or a nine character string. The numbers run from one, for Sunday to 7, for Saturday. The year is used to adjust the day-of-the-week for leap years.

Days:		
	1	SUNDAY
	2	MONDAY
	3	TUESDAY
	4	WEDNESDAY
	5	THURSDAY
	6	FRIDAY
	7	SATURDAY

Example:

This example returns the current day-of-the-week into N in line 100, this value is displayed in line 110. Then the day-of-the-week is calculated from the date specified as 1-MAY-83 and returned as a string in W$ in line 130 and displayed in line 140.

```
100 CALL "!WEEKDAY",W
110 PRINT W
120 DIM S$(10)
130 CALL "!WEEKDAY",1,5,83,S$
140 PRINT S$

RUN
4
SUNDAY
```

SECTION FOUR

TIMER COMMANDS

There are four timer commands available, PAUSE, PERIOD, ALARM and TIMER. The PAUSE and PERIOD commands work independently of the ALARM and TIMER commands. The ALARM and TIMER commands work together and only one or the other may be enabled at the same time.

The PAUSE command delays program execution by a specified number of seconds.

The other three timer commands cause a program interrupt to occur. The PERIOD interrupt causes the BASIC program to be interrupted at a user selected interval. The ALARM command causes the BASIC program to be interrupted according to a selected time of day. The TIMER command causes an interrupt to occur after a selected number of seconds.

All interrupts take place at the end of the current BASIC statement. In effect, a GOSUB to the specified line number is executed. This allows the user interrupt routine to return back to the previously executing program. If the RETURN is not executed, then return information will build up on the stack until all of memory is exhausted.

Because the line numbers specified in the interrupt call statements will not be renumbered, it is suggested that they be specified as numbers between 80 and 99. Line numbers in this range are not renumbered.

All interrupts are disabled when the 4050 is turned off. Interrupts need to be re-enabled when the 4050 is again turned on and a program loaded.

Section Contents

!PAUSE Delay program execution a number of seconds

CALL "!PAUSE",s

n = seconds to delay

Purpose: This command delays program execution the specified number of seconds.

General Information:
The maximum number of seconds to delay is 65535.
Control will return to the next statement following the pause after the specified number of seconds has elapsed.

Because CALL "!PAUSE" is a BASIC statement, timer interrupts will not be honored until after the end of the delay.

Example:

This example displays the current time in line 100, then the program delays for twenty seconds in line 110, and again displays the current time in line 120.

```
100 CALL "!TIME"
110 CALL "!PAUSE",20
120 CALL "!TIME"

RUN
10:45:00.45
10:45:20.47
```

!PERIOD PERIOD Timer Interrupt

CALL "!PERIOD" Disables Periodic Interrupts

CALL "!PERIOD",t,l

t = time from table below, l = BASIC program line number

t	Frequency	Time Period	t	Frequency	Time Period
0	disable interrupts				
1	1 hz	1.0 second	16	16 hz	62.5 ms
2	2 hz	500 ms	32	32 hz	31.25 ms
4	4 hz	250 ms	64	64 hz	15.625 ms
8	8 hz	125 ms	128	128 hz	7.8125 ms

CALL "!PERIOD",r,c,l

r = rate from table below, c = counts per interrupt
l = BASIC program line number

r	Frequency	Time Period	r	Frequency	Time Period
0	disable interrupts		7	128 hz	7.8125 ms
1	2 hz	500 ms	8	256 hz	3.90625 ms
2	4 hz	250 ms	9	512 hz	1.953125 ms
3	8 hz	125 ms	10	1024 hz	976.562 us
4	16 hz	62.5 ms	11	2048 hz	488.281 us
5	32 hz	31.25 ms	12	4096 hz	244.141 us
6	64 hz	15.625 ms	13	8192 hz	122.070 us

Purpose: The PERIOD interrupt command causes a program interrupt to occur at a user specified periodic interval. A BASIC program line number is specified to which the Basic system will execute a GOSUB after the currently executing BASIC line has finished.

General Information:

The user selects a time period or a rate and a count for the time period between each interrupt. Care must be exercised to not select a period so small that it locks up the 4050 system. PERIOD timer interrupts are disabled by either calling !PERIOD with no arguments or by specifying a zero for the time or rate value. While PERIOD interrupts are enabled, the sub-seconds field of the current time will be returned as a zero. A PERIOD interrupt may be enabled independently from an ALARM or TIMER interrupt.

Because the BASIC line number specified will not be renumbered, it is recommended that a line number between 80 and 99 be used for a GOTO that directs control to the actual interrupt routine line number.

Example:

This example demonstrates the PERIOD interrupt. It is divided into four sections the interrupt set up section in lines 700-720, the processing loop in lines 730-780, the interrupt target line 90, and the interrupt service routine in lines 840-860.

The interrupt set up section uses the CALL "!PERIOD" command to request that control be transferred to line number 90 four times a second. Note that the interrupt target line number 90 is used to send control to the actual service routine. This is done so that the renumbering of lines will function correctly. The service routine line number 840 in line 90 will be renumbered to match the new service routine line number. The line number 90 in the CALL "!PERIOD" command will not be renumbered.

After the interrupt has been set up, control continues into the processing loop. When an interrupt is generated by the clock, control will be transfered to line number 90 with a GOSUB after the current BASIC line is finished.

Line number 90 transfers control to the the actual interrupt service routine. This routine prints a star increments the variable I and returns control to the interrupted processing loop with a RETURN statement.

```
 90 GOTO 840

700 REM SET UP PERIOD INTERRUPT
710 I=0
720 CALL "!PERIOD",4,90

730 REM PROCESSING LOOP
740 PRINT "TEST PERIOD INTERRUPT"
750 J=0
760 IF I > 40 THEN 800
770 J=J+1
780 GOTO 760

790 REM DISABLE PERIOD INTERRUPT
800 CALL "!PERIOD"
810 PRINT "END PERIOD TEST J=";J
820 GOTO 900

830 REM PERIOD INTERRUPT SERVICE ROUTINE
840 I=I+1
850 PRINT "*";
860 RETURN
```

!TIMER Timer Interrupt

```
CALL "!TIMER"              Disable Alarm and Timer Interrupts

CALL "!TIMER",s,l

     s = seconds and l = BASIC program line number
```

Purpose: Gosub to the BASIC line number when the clock reaches the current time plus the specified number of seconds.

General Information:

The number of seconds specified may be up to 65535. The TIMER interrupt will only happen once. Enabling a TIMER interrupt will disable any outstanding ALARM interrupt. A TIMER interrupt may be enabled independently from a PERIOD interrupt.

Because the BASIC line number specified will not be renumbered, it is recommended that a line number between 80 and 99 be used for a GOTO that directs control to the actual interrupt routine line number.

Example:

This example contains the main program in lines 100 through 130, the interrupt target line 91, and the interrupt service routine in lines 1000 through 1020. The TIMER interrupt is set up for 926 seconds in line 100 after which the program loop is executed in lines 110 through 130. After 926 seconds the TIMER interrupt causes the program to GOSUB to line 91 where control is sent to the interrupt service routine. In the interrupt service routine, the message TIMER !! is printed on the screen. The RETURN in line 1020, returns control to the next statement in the main program loop.

```
91 GOTO 1010

100 CALL "!TIMER",926,91
110     .       .       .
120     .       .       .
130 GOTO 110

1000 REM TIMER INTERRUPT SERVICE ROUTINE
1010 PRINT "TIMER !!"
1020 RETURN
```

!ALARM Alarm Interrupt

```
CALL "!ALARM"              Disable Alarm and Timer Interrupts

CALL "!ALARM"[,d],h,m,s,l

   d = Optional julian day, h = hour, m = minute, s = second
   l = BASIC program line number
```

Purpose: Gosub to a BASIC line number when the clock reaches time 'h:m:s'. If the day is specified, only interrupt on julian day 'd'.

General Information:

Alarm interrupts may be set for once a year, day, hour or second. The hour is specified in 24 hour format. A value of 255 specifies a don't care value for the day, hour, minute or second. For example to cause an alarm interrupt to occur every hour at 5 minutes and 15 seconds after the hour the hour would be specified as 255, the minute as 5 and the second as 15.

Enabling an ALARM interrupt will disable any TIMER interrupt. An ALARM interrupt may be enabled independently from a PERIOD interrupt.

Because the BASIC line number specified will not be renumbered, it is recommended that a line number between 80 and 99 be used for a GOTO that directs control to the actual interrupt routine line number.

Example:

This example contains the main program in lines 100 through 130, the interrupt target line 92, and the interrupt service routine in lines 1000 through 1030. The ALARM interrupt is set up for 12:15:32 in line 100 after which the program loop is executed in lines 110 through 130. At 12:15:32 the ALARM interrupt causes the program to GOSUB to line 92 where control is sent to the interrupt service routine. In the interrupt service routine, the message ALARM !! is printed on the screen followed by the date and time. The RETURN statement in line 1030, returns control to the next statement in the main program loop.

```
92 GOTO 1010

100 CALL "!ALARM",12,15,32,92
110    .     .     .     .
120    .     .     .     .
130 GOTO 110

1000 REM ALARM INTERRUPT SERVICE ROUTINE
1010 PRINT "Alarm !!"
1020 CALL "!DATETIME"
1030 RETURN
```

!RALARM Read the Alarm Time

```
CALL "!RALARM",d,h,m,s
```

d = julian day, h = hour, m = minute and s = second

Purpose: Returns the alarm day, hour, minute and second into numeric variables if the ALARM interrupt is active.

General Information:
If the ALARM interrupt is not enabled, all zeros are returned.

Example:

This example returns the alarm day and time into the numeric variables D,H,M and S in line number 100. The values are displayed on the screen in line 110.

```
100 CALL "!RALARM",D,H,M,S
110 PRINT D;H;M;S
```

SECTION FIVE

CMOS RAM COMMANDS

There are 40 bytes of battery backed-up memory in the clock ROM pack that may be used to store information. They are numbered from 1 to 40. Commands are included to allow the CMOS memory to be read and written with user data. Additional commands are provided to allow a power-up program to be loaded or a password to be entered before allowing the 4050 to start normal operation.

Section Contents

!READ Read Byte or String from CMOS Memory

```
CALL "!READ",v,l
        v = value, l = memory location

CALL "!READ",s$,n,l
       s$ = string, n = number of bytes,
       l = starting memory location
```

Purpose: Read a byte or string back from the Clock CMOS memory.

General Information:
Two forms of the !READ call are available, one for reading a string and one for reading a single byte. The range of values for the byte is from zero to two-hundred fifty-five.

Example:

```
100 DIM A$(10)
110 CALL "!READ",A$,10,20
120 CALL "!READ",N,19
130 PRINT N,A$
```

A ten byte string is read from CMOS memory starting at location twenty and a byte is read from location ninteen.

!WRITE Write Byte or String to CMOS Memory

```
CALL "!WRITE",v,l
        v=value, l=memory location

CALL "!WRITE",s$,l
        s$=string, l=starting memory location
```

Purpose: Write a byte or string to the Clock CMOS memory.

General Information:
Two forms of the !WRITE call are available, one for writing a string and one for writing a single byte. The range of values for the byte is from zero to two-hundred fifty-five.

Example:

```
100 A$="This is a test"
110 CALL "!WRITE",A$,20
120 CALL "!WRITE",N,19
```

This example writes a string into CMOS memory starting at location twenty and a byte into location ninteen.

!SETPU Set the Power Up String

```
CALL "!SETPU"              Clear power up string

CALL "!SETPU",s$,I

        s$=power up string, I=starting CMOS memory location
```

Purpose: This command loads user specified BASIC commands into the 4050 system type ahead buffer upon power up.

General Information:
Upon 4050 power-up, the string is copied into the keyboard type ahead input buffer. It will be used as keyboard input after the power-up sequence is completed. The string may be up to 28 characters in length.

Example:

This example enables the Power-Up string to load the program "@START" from disk and RUN it. Note the REP statements used to put carriage returns into the Power-Up string A$.

```
100 A$="OLD""@START""_RUN_"
110 B$=CHR(13)
120 A$=REP(B$,12,1)
130 A$=REP(B$,16,1)
140 CALL "SETPU",A$,10
```

This example shows a small program that will be run each time the system is powered up. The program uses CMOS memory locations 1 through 16 leaving locations 17 through 40 free.

```
500 A$="1FORJ=1TO10 2PRIJ 3NEXJ RUN "
510 B$=CHR(13)
520 A$=REP(B$,12,1)
530 A$=REP(B$,18,1)
540 A$=REP(B$,24,1)
550 A$=REP(B$,28,1)
560 CALL "!SETPU",A$,1
```

When the power is restored the four lines will be entered as if they had been typed and the program run. It would look as follows.

```
1FORJ=1TO10
2PRIJ
3NEXJ
RUN
```

!SETPS Set the Power-Up Password

CALL "!SETPS" Clear password

CALL "!SETPS",s$,l

s$=password, l=starting CMOS memory location

Purpose: This command specifies the password that must be entered upon power-up to allow the 4050 system to operate.

General Information:
After 4050 power-up, enter a carriage return. This will cause the 4050 to prompt for the password. Enter the password followed by a carriage return. If the password in entered correctly, the 4050 will respond with the

System Unlocked

message and operate normally. If an incorrect password is entered, the 4050 will prompt for the password until it is entered correctly.

Warning:
There is no way of clearing an enabled password if it is forgotten. Power must be removed from the RTC chip to disable the power-up password.

Example:

CALL "!SETPS","TESTIT",10

This example disables any Power-Up string and sets the Password to "TESTIT".

Appendix A

REAL TIME CLOCK COMMAND SUMMARY

641/741RTC Display Firmware Level
```
CALL "641RTC"                    For 4051
CALL "741RTC"                    For 4052 or 4054
```

!12HOUR Set Clock to 12 Hour Format
```
CALL "!12HOUR"
```

!24HOUR Set Clock to 24 Hour Format
```
CALL "!24HOUR"
```

!ALARM Alarm Interrupt
```
CALL "!ALARM"            Disable Alarm and Timer Interrupts

CALL "!ALARM"[,d],h,m,s,l
   d = Optional julian day, h = hour, m = minute, s = second
   l = BASIC program line number
```

!DATE Display or Return Current Date
```
CALL "!DATE",[d$]
   d$ = Optional target string for date.

CALL "!DATE",d,m,y
   d = day (1-31), m = month (1-12) and y = year (0-99).
```

!DATETIME Display or Return Current Date and Time
```
CALL "!DATETIME",[t$]
   t$ = Optional target string for date and time.
```

!DAY Return Julian Day
```
CALL "!DAY",n
   n = current julian day

CALL "!DAY",d,m,y,j
   d = day, m = month and y = year and j = julian day
```

!DAYLIGHT Set Clock to Daylight Savings Time Mode
```
CALL "!DAYLIGHT"
```

```
!MODES    Display Real Time Clock Modes
        CALL "!MODES"


!MONTH    Return Day and Month from Julian Day and Year
        CALL "!MONTH",j,y,d,m
            j = julian day, y = year, d = day and m = month


!PAUSE    Delay Program Execution a number of Seconds
        CALL  "!PAUSE",s
            n = seconds to delay


!PERIOD   Periodic Timer Interrupt
        CALL "!PERIOD"          Disables Periodic Interrupts
        CALL "!PERIOD",t,l
            t=time from table, l=BASIC program line number

        CALL "!PERIOD",r,c,l
            r=rate from table, c=counts, l=BASIC program line number


!RALARM    Read Alarm Time
        CALL "!RALARM",d,h,m,s
            d = julian day, h = hour, m = minute and  s = second


!READ   Read Byte or String from CMOS Memory
        CALL "!READ",s$,n,l
                s$ = string, n = number of bytes,
                l = starting memory location

        CALL "!READ",v,l
                 v = value, l = memory location


!SECONDS   Return Seconds in the Day
        CALL "!SECONDS",n
            n = current second in the day

        CALL "!SECONDS",h,m,s,n
           h = hour, m = minute, s = second and n = second in the day


!SETPS  Set Password
        CALL "!SETPS"           Clear password

        CALL "!SETPS",s$,l
                s$=password, l=starting CMOS memory location
```

```
!SETPU  Set Power-Up String
        CALL "!SETPU"              Clear power up string

        CALL "!SETPU",s$,l
             s$=power up string, l=starting CMOS memory location


!SETTIME  Set Clock Date and Time
        CALL "!SETTIME",t$
          t$ = Date and Time String (string variable or literal)


!STANDARD    Set Clock to Standard Time Mode
        CALL "!STANDARD"


!TIME   Display or Return Current Time
        CALL "!TIME",[t$]
          t$ = Optional target string for time.

        CALL "!TIME",h,m,s,b
           h = hour, m = minute, s = second and b = sub-seconds.


!TIMER    Timer Interrupt
        CALL  "!TIMER"             Disable Alarm and Timer Interrupts

        CALL  "!TIMER",s,l
           s = seconds and l = BASIC program line number


!WEEKDAY   Return Day of the Week
        CALL "!WEEKDAY",w
           w = day-of-week number

        CALL "!WEEKDAY",d,m,y,w
           d = day, m = month, y = year and w = day-of-week number

        CALL "!WEEKDAY",w$
           w$ = day-of-week name

        CALL "!WEEKDAY",d,m,y,w$
           d = day, m = month, y = year and w$ = day-of-week name



!WRITE  Write Byte or String to CMOS Memory
        CALL "!WRITE",s$,l
               s$=string, l=starting memory location

        CALL "!WRITE",v,l
                v=value, l=memory location
```

APPENDIX - B

EXAMPLE BASIC CLOCK TEST PROGRAM

A BASIC program that exercises most of the the Real Time Clock functions is given on the following pages.

```
1 REM RTC.TEK -- Test the New 641/741RTC ROM Pack
2 GO TO 100
90 GO TO 1310
91 GO TO 1520
92 GO TO 1790

100 PAGE
110 PRINT "641/741 Real Time Clock Test Program"
120 PRINT
130 PRINT "1. Read Firmware Level and Set Clock"
140 PRINT "   Firmware message: ";
150 IF INT(RND(0)*10)<>1 THEN 180
160 CALL "641RTC"
170 GO TO 190
180 CALL "741RTC"
190 CALL "!SETTIME","02-MAY-83 10:00:00"

200 PRINT
210 PRINT "2. Simple Read Clock Commands"
220 PRINT
230 PRINT "   Current Date and Time=";
240 CALL "!DATETIME"
250 CALL "!DATETIME",B$
260 PRINT "    Date and Time String=";B$
270 PRINT
280 PRINT "   Current Date=";
290 CALL "!DATE"
300 CALL "!DATE",B$
310 PRINT "    Date String=";B$;
320 CALL "!DATE",D,T,Y
330 PRINT " Numbers=";D;T;Y
340 PRINT
350 PRINT "   Current Time=";
360 CALL "!TIME"
370 CALL "!TIME",B$
380 PRINT "    Time String=";B$;
390 CALL "!TIME",H,M,S,B
400 PRINT " Numbers=";H;M;S;B
410 PRINT
```

```
420 PRINT "3. Extended Read Clock Commands"
430 PRINT
440 CALL "!SECONDS",N
450 PRINT "   Current Seconds=";N;
460 CALL "!TIME",H,M,S,B
470 CALL "!SECONDS",H,M,S,N
480 PRINT " Find Seconds=";N
490 PRINT
500 CALL "!WEEKDAY",N
510 PRINT "   Current Weekday Number=";N;
520 CALL "!WEEKDAY",B$
530 PRINT " String=";B$
540 CALL "!WEEKDAY",D,T,Y,N
550 PRINT "      Find Weekday Number=";N;
560 CALL "!WEEKDAY",D,T,Y,B$
570 PRINT " String=";B$
580 PRINT
590 CALL "!DAY",J
600 PRINT "   Current Julian Day=";J;
610 CALL "!DAY",D,T,Y,J
620 PRINT " Find Julian Day=";J
630 CALL "!MONTH",J,Y,D,T
640 PRINT "   Julian Day=";J;" Year=";Y;" Day=";D;" Month=";T
650 PRINT

660 PRINT "4.  Clock Mode Commands"
670 PRINT
680 CALL "!24HOUR"
690 CALL "!STANDARD"
700 PRINT "   Should be 24 Hour Standard-";
710 CALL "!MODES"
720 CALL "!12HOUR"
730 PRINT "   Should be 12 Hour Standard-";
740 CALL "!MODES"
750 CALL "!24HOUR"
760 CALL "!DAYLIGHT"
770 PRINT "   Should be 24 Hour Daylight Savings-";
780 CALL "!MODES"
790 CALL "!STANDARD"
800 PRINT "   Should be 24 Hour Standard-";
810 CALL "!MODES"
820 PRINT "   End of Modes Test"
830 PRINT "Press Carriage Return to Continue.";
840 INPUT B$
850 PAGE
```

```
860 PRINT "5. Timer Commands"
870 PRINT
880 PRINT "  PAUSE Test 10 Seconds"
890 PRINT "    Start Time=";
900 CALL "!TIME"
910 CALL "!PAUSE",10
920 PRINT "      End Time=";
930 CALL "!TIME"
940 REM

950 PRINT
960 PRINT "  CMOS Memory Test"
970 FOR I=1 TO 40
980 CALL "!WRITE",I,I
990 NEXT I
1000 FOR I=0 TO 3
1010 PRINT "     ";
1020 FOR J=1 TO 10
1030 P=I*10+J
1040 CALL "!READ",N,P
1050 PRINT N;
1060 IF P=N THEN 1090
1070 PRINT "    Number Error P/J=";P;J
1080 STOP
1090 NEXT J
1100 PRINT
1110 NEXT I
1120 PRINT "    Pass Byte Read/Write"
1130 B$="TEST STRING"
1140 CALL "!WRITE",B$,1
1150 CALL "!READ",C$,LEN(B$),1
1160 IF C$=B$ THEN 1190
1170 PRINT "CMOS STRING ERROR B$/C$=";B$;C$
1180 STOP
1190 PRINT "      ";B$,C$
1200 PRINT "    Pass String Read/Write"
1210 PRINT

1220 PRINT "  Test PERIOD interrupt at 1 Hz"
1230 I=0
1240 PRINT "    ";
1250 CALL "!PERIOD",1,90
1260 REM PROCESSING LOOP
1270 J=0
1280 IF I>20 THEN 1350
1290 J=J+1
1300 GO TO 1280
1310 REM PERIOD INTERRUPT ROUTINE
1320 I=I+1
1330 PRINT "*";
1340 RETURN
1350 CALL "!PERIOD"
1360 PRINT
1370 PRINT "    End PERIOD Test J=";J
```

```
1400 PRINT
1410 PRINT "  TIMER Interrupt Test for 10 seconds"
1420 PRINT "    Start at ";
1430 CALL "!TIME"
1440 I=0
1450 CALL "!TIMER",10,91
1460 REM PROCESSING LOOP
1470 J=0
1480 IF I>0 THEN 1560
1490 J=J+1
1500 GO TO 1480
1510 REM TIMER INTERRUPT ROUTINE
1520 PRINT "    TIMER Interrupt !! at ";
1530 CALL "!TIME"
1540 I=1
1550 RETURN
1560 PRINT "    End TIMER test J=";J

1590 PRINT
1600 PRINT "  Start ALARM Interrupt Test at ";
1610 CALL "!TIME"
1620 I=0
1630 CALL "!TIME",H,M,S,B
1640 S=S+10
1650 IF S<60 THEN 1700
1660 S=S-60
1670 M=M+1
1680 IF M<60 THEN 1700
1690 H=H+1
1700 CALL "!ALARM",H,M,S,92
1710 CALL "!RALARM",J,H,M,S
1720 PRINT "    ALARM Set to ";H;":";M;":";S
1730 REM PROCESSING LOOP
1740 J=0
1750 IF I>0 THEN 1830
1760 J=J+1
1770 GO TO 1750
1780 REM ALARM INTERRUPT ROUTINE
1790 PRINT "    ALARM Interrupt !! at ";
1800 CALL "!TIME"
1810 I=1
1820 RETURN
1830 CALL "!ALARM"
1840 PRINT "    End ALARM test J=";J

1860 PRINT
1870 PRINT "  Power-Up String Test"
1880 B$="1FORJ=1TO10 2PRIJ 3NEXJ RUN "
1890 C$=CHR(13)
1900 B$=REP(C$,12,1)
1910 B$=REP(C$,18,1)
1920 B$=REP(C$,24,1)
1930 B$=REP(C$,28,1)
1940 CALL "!SETPU",B$,1
1950 PRINT "    Please turn OFF the power switch,"
1960 PRINT "    and then turn it back ON."
1970 PRINT "    Don't forget to CALL ""!SETPU"" after the test."
```