



APPLICATION NOTE 519

Using a Nonvolatile Timekeeping RAM with a Microcontroller

Abstract: This application note describes how to use a Nonvolatile Timekeeping RAM with an 8051-type microcontroller. An example schematic is provided. Source code provides an example of how to read and write the real-time clock (RTC) registers.

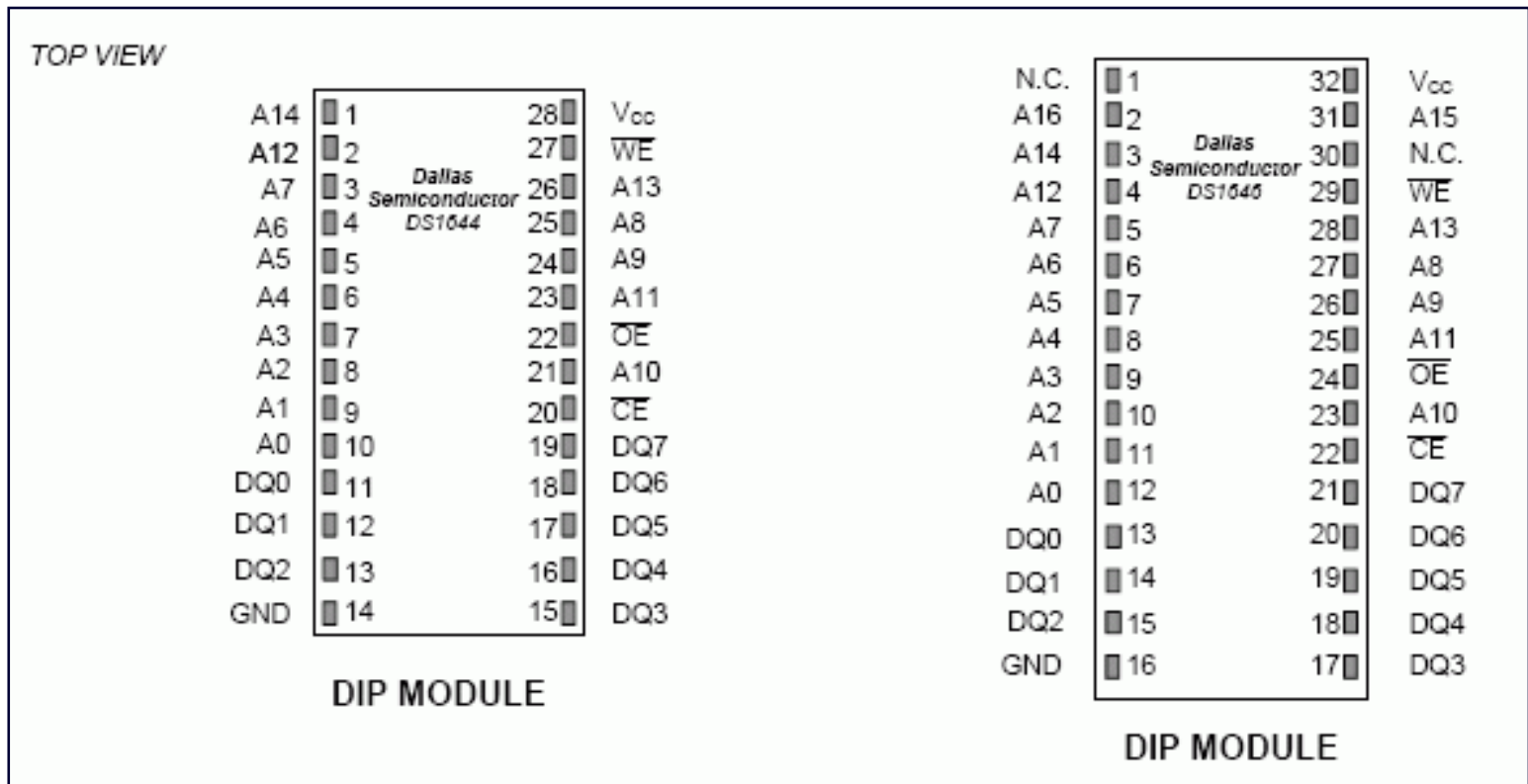
Introduction

The nonvolatile timekeeping family of real-time clock (RTC) products provide battery-backed NV SRAM as well as time and date. The clock registers are accessed identically to the SRAM. Table 1 shows a list of products in the nonvolatile timekeeping family. The DS174_ series are identical to the DS164_, with the addition of a century bit and optional 3.3V operation.

Table 1. Nonvolatile Timekeeping Products

Part	RAM Density	V _{CC} (V)
DS1642	2k × 8	5
DS1643	8k × 8	5
DS1644	32k × 8	5
DS1646	128k × 8	5
DS1647	512k × 8	5
DS1742	2k × 8	3.3 or 5
DS1743	8k × 8	3.3 or 5
DS1744	32k × 8	3.3 or 5
DS1746	128k × 8	3.3 or 5
DS1747	512k × 8	3.3 or 5

Pin Configurations



In the following example, a DS1644 or DS1646 is connected to a DS87C320. The DS87C320 high-speed microcontroller is compatible with the industry-standard 8051 architecture. Other NV RAM densities could be supported with appropriate changes to the circuit. For related application notes, please refer to the Quick View data sheets for the DS87C320 and DS1644. A reference RTC clock schematic follows the code.

DS1644/DS1646 Code

```

/*****
/* DS1644.c - Access DS1644/6, DS1744/6 using Reference RTC circuit */
/* This program if for example only and is not supported by Dallas */
/* Semiconductor Maxim. */
/*****
#include <stdio.h> /* Prototypes for I/O functions */
#include <DS5000.h> /* Register declarations for DS5000 */
#include <absacc.h> /* needed to define xdata addresses */
#include <delay.c>
/***** bit definitions *****/
/***** Defines *****/
/***** Global Variables *****/
uchar yr, mn, dt, dy, hr, min, sec, day;
unsigned int MAXADD;
sbit RSTb = P2^7;
sbit IRQb = P2^6;
sbit AD16 = P1^5;
sbit AD17 = P1^6;
sbit AD18 = P1^7;
/***** Function Prototypes *****/
void writereg();
void init_rtc();
void Disp_Clk_Regs();
void ramread();
void ramwrite();

```

```

void toggle_ft();
void init_rtc() /* ----- set the time and date ----- */
/* Note: NO error checking is done on the user entries! */
{
uchar tmp;
printf("\nEnter the year (0-99): ");
scanf("%bx", &yr);
printf("Enter the month (1-12): ");
scanf("%bx", &mn);
printf("Enter the date (1-31): ");
scanf("%bx", &dt);
printf("Enter the day (1-7): ");
scanf("%bx", &dy);
printf("Enter the hour (1-23): ");
scanf("%bx", &hr);
printf("Enter the minute (0-59): ");
scanf("%bx", &min);
printf("Enter the second (0-59): ");
scanf("%bx", &sec);
AD16 = AD17 = AD18 = 1;
XBYTE[0xffff8] = 0x80; /* set write bit for write */
XBYTE[0xffff9] = sec;
XBYTE[0xffffa] = min;
XBYTE[0xffffb] = hr;
XBYTE[0xffffc] = dy;
XBYTE[0xffffd] = dt;
XBYTE[0xffffe] = mn;
XBYTE[0xfffff] = yr;
XBYTE[0xffff8] = 0; /* clear write bit to allow update */
}
void Disp_Clk_Regs() /* ----- */
{
uchar msec, Sec, prv_sec = 99, Min, Hrs, Dte, Mon, Day, Yr;
AD16 = AD17 = AD18 = 1;
printf("\nYr Mn Dt Dy Hr:Mn:Sec");
while(!RI) /* Read & Display Clock Registers */
{
XBYTE[0xffff8] = 0x40; /* set read bit to stop updates */
Sec = XBYTE[0xffff9];
Min = XBYTE[0xffffa];
Hrs = XBYTE[0xffffb];
Day = XBYTE[0xffffc];
Dte = XBYTE[0xffffd];
Mon = XBYTE[0xffffe];
Yr = XBYTE[0xfffff];
XBYTE[0xffff8] = 0; /* clear read bit to resume updates */
delay(3); /* must allow time for transfer to occur */
if(Sec != prv_sec) /* display every time seconds change */
{
printf("\n%02.bX %02.bX %02.bX %02.bX", Yr, Mon, Dte, Day);
printf(" %02.bX:%02.bX:%02.bX", Hrs, Min, Sec);
}
prv_sec = Sec;
}
RI = 0; /* Swallow keypress to exit loop */
}
void ramread() /* ----- Read RAM ----- */
{
unsigned int j;
AD16 = AD17 = AD18 = 0;
for (j = 0; j <= MAXADD; j++)
{

```

```

if(j != 0 && !(j % 256) )
{
printf("\nPress a key or Esc to exit ");
if( _getkey() == 0x1b)
return;
}
if(!(j % 16)) printf("\n%bx%04x ", (uchar) AD16, j);
printf("%02.bx ", (uchar) XBYTE[j]);
if (j == 0xffff) break;
}
if(MAXADD == 0xffff)
{
printf("\nPress a key or Esc to exit ");
if( _getkey() == 0x1b)
return;
AD16 = 1;
for (j = 0; j <= MAXADD; j++)
if(j != 0 && !(j % 256) )
{
printf("\nPress a key or Esc to exit ");
if( _getkey() == 0x1b)
break;
}
if(!(j % 16)) printf("\n%bx%04x ", (uchar) AD16, j);
printf("%02.bx ", (uchar) XBYTE[j]);
if (j == 0xffff) break;
}
}
}

void ramwrite() /* ----- write with incrementing data ----- */
{
unsigned int j;
uchar offset = 0;
if(MAXADD == 0xFFFF)
{
AD16 = AD17 = AD18 = 0;
for (j = 0; j <= MAXADD; j++)
{
XBYTE[j] = (uchar) ( (j & 0xff) + offset);
if( (j & 0xff) == 0xff) offset++;
if(j == 0xffff) break;
}
}
AD16 = 1;
offset = 0;
for (j = 0; j < (MAXADD - 7); j++)
{
XBYTE[j] = (uchar) ( (j & 0xff) + offset);
if( (j & 0xff) == 0xff) offset++;
}
}

void ramfill(uchar dat) /* ----- fill with fixed data ----- */
{
unsigned int j;
if(MAXADD == 0xFFFF)
{
AD16 = AD17 = AD18 = 0;
for (j = 0; j <= MAXADD; j++)
{
if(j & 0x100)
XBYTE[j] = dat ^ 0xFF;
}
}
}

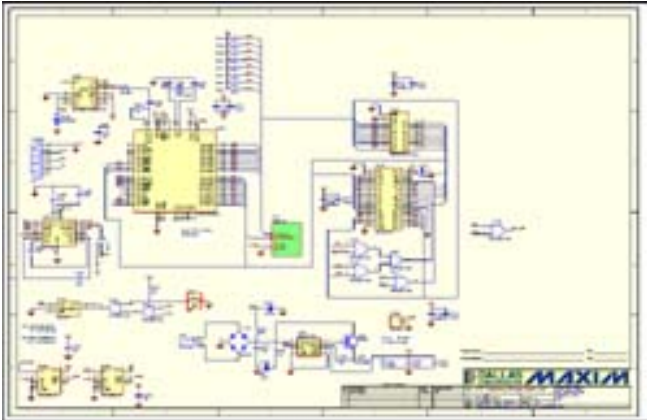
```

```

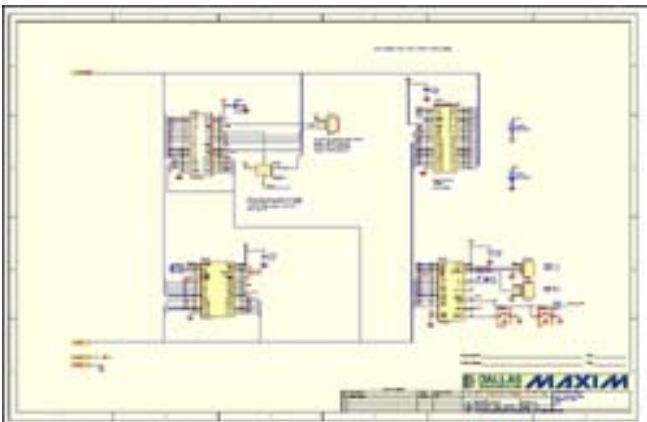
else
XBYTE[j] = dat;
if(j == 0xffff) break;
}
}
AD16 = 1;
for (j = 0; j < (MAXADD - 7); j++) /* don't disturb time & date */
{
if(j & 0x100)
XBYTE[j] = dat ^ 0xFF;
else
XBYTE[j] = dat;
}
}
void toggle_ft() /* ----- toggle the Frequency Test bit ----- */
{
uchar creg;
AD16=1;
creg = XBYTE[0xffffc];
if( (creg & 0x40) ) /* check FT bit */
{
XBYTE[0xffff8] = 0x80; /* set write bit for write */
XBYTE[0xffffc] = (creg & 0xbf); /* clear FT bit */
XBYTE[0xffff8] = 0; /* clear write bit */
}
else
{
XBYTE[0xffff8] = 0x80; /* set write bit for write */
XBYTE[0xffffc] = (creg | 0x40); /* set FT bit */
XBYTE[0xffff8] = 0; /* clear write bit */
}
}
main (void) /* ----- */
{
uchar i, M, M1;
RSTb = 1;
printf("\n1) DS1644 or 2) DS1646 ? ");
i = _getkey();
if (i == '1') MAXADD = 0x7FFF;
if (i == '2') MAXADD = 0xFFFF;
while (1)
{
printf("\nDS1644/6 build 100\n");
printf("CI Init clock CR Read Time\n");
printf("R RAM Fill RR RAM Read\n");
printf("T Toggle FT");
printf("\nEnter Menu Selection:");
M = _getkey();
switch(M)
{
case 'C':
case 'c':
printf("\rEnter Clock Routine to run: C");
M1 = _getkey();
switch(M1)
{
case 'I':
case 'i': init_rtc(); break;
case 'R':
case 'r': Disp_Clk_Regs(); break;
}
}
break;
}
}

```

```
case 'R':
case 'r':
printf("\rFill RAM A)a, 5)5, I)nc data or R)ead: ");
M1 = _getkey();
switch(M1)
{
case '5': ramfill(0x55); break;
case 'A':
case 'a': ramfill(0xaa); break;
case 'R':
case 'r': ramread(); break;
case 'I':
case 'i': ramwrite(); break;
}
break;
case 'T':
case 't': toggle_ft(); break;
}
}
}
```



[More Detailed Image](#)



[More Detailed Image](#)

Application Note 519: <http://www.maxim-ic.com/an519>

More Information

For technical questions and support: <http://www.maxim-ic.com/support>

For samples: <http://www.maxim-ic.com/samples>

Other questions and comments: <http://www.maxim-ic.com/contact>

Related Parts

DS1642: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)

DS1643: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)

DS1644: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)

DS1646: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)

DS1647: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)

DS1742: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)

DS1743: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)

DS1744: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)

DS1746: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)

DS1747: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)

AN519, AN 519, APP519, Appnote519, Appnote 519

Copyright © by Maxim Integrated Products

Additional legal notices: <http://www.maxim-ic.com/legal>