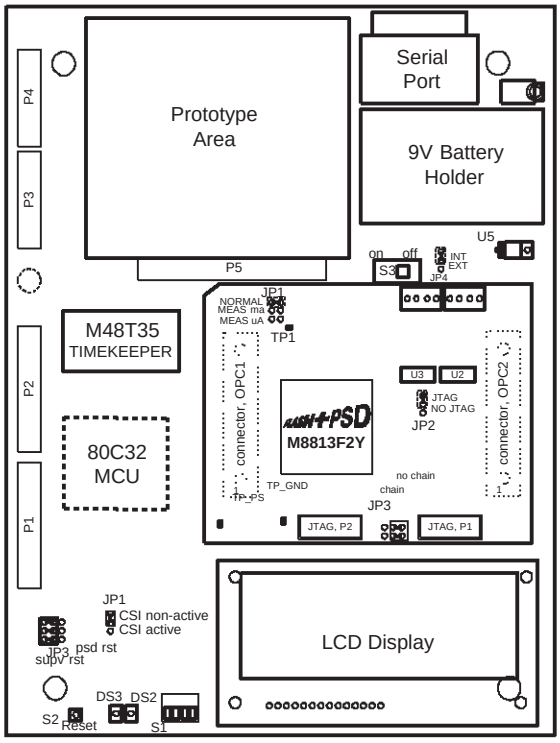
AN1259
APPLICATION NOTE

M8813F2Y FLASH+PSD and M48T35Y TIMEKEEPER Demonstration

FLASH+PSD, from STMicroelectronics, is a family of Flash memory based programmable system devices (PSDs) for 8 bit micro-controllers. The M8813F2Y-90K1 is an 8-bit FLASH+PSD with 128Kx8 bit of primary Flash memory, 8Kx8 bit of secondary boot Flash memory, 2Kx8 bit of SRAM, 27 I/O port pins, and a JTAG interface for in system programming (ISP).

The 80C32-based M8DK8051 Evaluation Board, shown in Figure 1, illustrates the versatility of the FLASH+PSD. For example, it illustrates its extensibility to include extra battery backed SRAM, and a real time clock (RTC), in its address space. This has been added in the form of a 32KX8 bit CMOS TIMEKEEPER SRAM, the M48T35Y-70PC1. This has an on-chip RTC, which allows the time of day to be read continuously out to the liquid crystal display on the evaluation board. This document describes how this can be achieved.

Figure 1. Layout of the 80C32-based M8DK8051-K52-110 Evaluation Board



AI03762

The evaluation board is delivered with C code already embedded. This can be modified, as described next.

First, a file can be created, called timekeeper.h, in which to define the following variables and a function prototype:

```
uchar xdata myear _at_ 0x7fff;
uchar xdata mmonth _at_ 0x7ffe;
uchar xdata mdate _at_ 0x7ffd;
uchar xdata mday _at_ 0x7ffc;
uchar xdata mhour _at_ 0x7ffb;
uchar xdata mminutes _at_ 0x7ffa;
uchar xdata mseconds _at_ 0x7ff9;
uchar xdata mcontrol _at_ 0x7ff8;
void  gettimekeeper(void);
```

The data type xdata is one that the Keil C compiler recognizes for the 80C31. It stands for external data. The variables are defined for the registers of the M48T35 (as shown in Table 1) and are used in the gettimekeeper() function.

Table 1. M48T35 Register Map

Address	Data								Function	Range (in BCD Format)
	D7	D6	D5	D4	D3	D2	D1	D0		
7FFFh	10 Years				Year				Year	00-99
7FFEh	0	0	0	10M	Month				Month	01-12
7FFDh	0	0	10 Date		Date				Date	01-31
7FFCh	0	FT	0	0	0	Day			Day	01-7
7FFBh	0	0	10 Hours		Hours				Hour	00-23
7FFAh	0	10 Minutes			10 Minutes				Minute	00-59
7FF9h	ST	10 Seconds			Seconds				Second	00-59
7FF8h	W	R	S	Calibration					Control	

Notes: 1. S = Sign bit
2. FT = Frequency Test bit (must be reset to 0 for normal clock operation when powered-up
3. R = Read bit
4. W = Write bit
5. ST = Stop bit
6. 0 = must be reset to 0

The first code fragment was added to evaltest.c:

```
lcd_clear();
lcd_string_display(0,0,”The time is:”);
while(1)
   {
   gettimekeeper();
}
```

The LCD is a 2x16 character display. It accepts and displays ASCII characters. The lcd_clear() function clears the display by writing the space character (0x20h) to all 32 locations of the LCD display, and is already supplied with the source code for the board. The function lcd_string_display(0,0,”The time is:”); writes “The time is:” to the first line of the display at the first position. The program then goes into an infinite loop in which the function gettimekeeper(), to get and display the time, is called over and over again. This infinite loop gives the display the appearance of a digital clock.

The next code fragment defines the function gettimekeeper(), and can be added to the other functions in evl_io.c:

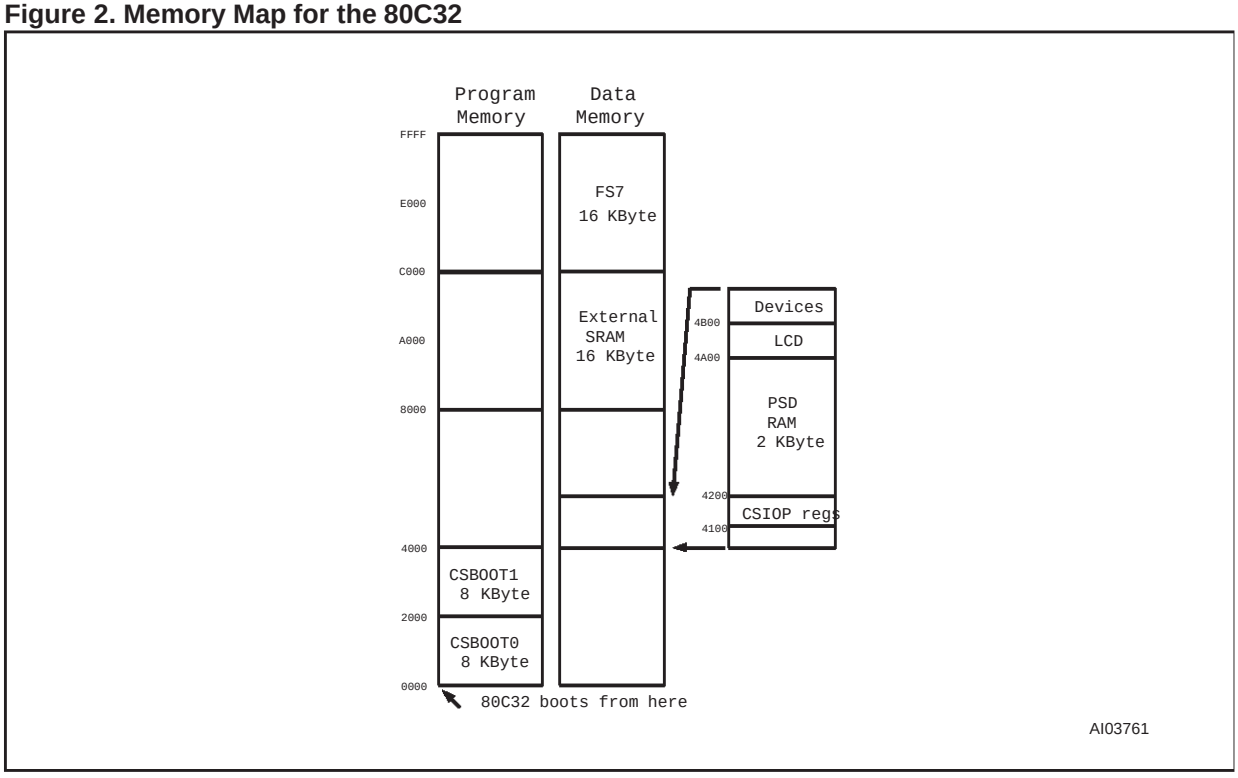
```
void gettimekeeper(void)
   {
   uchar chr;

   chr=(mhour&0xf0)/0x10|0x30;
   lcd_char_display(1,0,chr);
   chr=(mhour&0x0f)|0x30;
   lcd_char_display(1,1,chr);
   lcd_char_display(1,2, ’:’ );
   chr=(mminutes&0xf0)/0x10|0x30;
   lcd_char_display(1,3,chr);
   chr=(mminutes&0x0f)|0x30;
   lcd_char_display(1,4,chr);
   lcd_char_display(1,5, ’:’ );
   chr=(mseconds&0xf0)/0x10|0x30;
   lcd_char_display(1,6,chr);
   chr=(mseconds&0x0f)|0x30;
   lcd_char_display(1,7,chr);
   }
```

This routine reads the TIMEKEEPER registers, converts the values to ASCII characters, ready to be sent to the appropriate position on the LCD display. For example, the first line of this function reads the hour byte from location 0x7FFBh. This byte contains the hour of the day, which consists of two binary coded decimal digits in the upper and lower nibbles of the byte. The first digit is converted to an ASCII character by ANDing it with 0xF0. The data is then shifted right four places by dividing the byte by 16 (0x10). Finally the byte is ORed with 0x30 to form the ASCII byte to be sent to the display. The lower nibble is converted

to ASCII in the same way except it is masked, by ANDing it with 0x0F, and does not need any shifting right. After each conversion, the byte is sent to the bottom row of the display in the right position using the lcd_char_display() function that comes with the source for the board. The remaining digits are sent to the display in the same manner. The colons (:) are sent to the display to give the format of the time the correct appearance.

Figure 2 shows the memory map for this application. The program is stored in, and executed from, CSBOOT0.

Figure 2. Memory Map for the 80C32



There is one additional change that needs to be made to the abel file. The SRAM registers are based at 0x7FF0 in the C code. This means they must be mapped into the address space in the abel file for the project. This can be achieved by adding a term to the chip select (cs_ram) equation in the abel file.

Appearance of the equation before the change has been made:

```
cs_ram_ =  ((address >= ^h0000) & (address <= ^h3FFF) & (page == X) & !es0 & es1 )
        #( (address >= ^h8000) & (address <= ^hBFFF) & (page == X)& !(!es0 & es1 ) )
```

Appearance of the equation after the change has been made:

```
cs_ram_ =  ((address >= ^h0000) & (address <= ^h3FFF) & (page == X) & !es0 & es1 )
        #( (address >= ^h8000) & (address <= ^hBFFF) & (page == X)& !(!es0 & es1 ) )
        #( (address >= ^h7FF0) & (address <= ^h7FFF) & (page == X));
```

For current information on M88 FLASH+PSD products, please consult our pages on the world wide web:
www.st.com/flashpsd

If you have any questions or suggestions concerning the matters raised in this document, please send them to the following electronic mail addresses:

apps.flashpsd@st.com	(for application support on FLASH+PSD)
apps.nvram@st.com	(for application support on TIMEKEEPER)
ask.memory@st.com	(for general enquiries)

Please remember to include your name, company, location, telephone number and fax number.